33 PAGES OF EXPERT TUTORIALS

web designer

FREE 72 MINUTES OF HTML VIDEOS

HTML5 CSS3 jQuery JS JavaScript @WebDesignerMag /WebDesignerUK

GET STARTED VUE.JS

A fast track guide to the framework

AUGMENTED REALITY IN THE BROWSER

Implement WebAR on mobile & desktop

NEW GENERATION CSS

WEAVE THE LATEST MUST-KNOW TOOLS & RULES INTO YOUR PROJECTS

Get creative with CSS Grid • Build with Web Animations API
Optimise with Critical CSS • Code Custom Properties

HOW TO CODE A REACT APP

CUSTOM SOCIAL MEDIA BUTTONS

JS FRAMEWORKS: MAKE THE RIGHT CHOICE

SAY HELLO TO ANGULAR 4

A practical look at the latest release

Future
ISSUE 262

20i™ FROM THE CREATORS OF RESELLER HOSTING

GET YOUR FIRST MONTH OF **UNLIMITED RESELLER HOSTING** FOR

£1 THEN ONLY £29.99 A MONTH

NO CONTRACT
+ 30 DAY MONEY BACK GUARANTEE

SIGN UP AT
20i.com/design

Reseller Web Hosting on a 100% SSD, Autoscaling Cloud Hosting Platform.

TIM & JONATHAN BREALEY
FOUNDING DIRECTORS OF 20i & INDUSTRY VETERANS

UNLIMITED HOSTING

Includes unlimited SSD web space, bandwidth, websites, user accounts, 1GB MySQL databases, 10GB mailboxes and free SSL certificates.

EASY MIGRATION

Move your domains, websites, email and databases easily with our automated one-stop migration centre.

UNIQUE TO 20i

SSD AS STANDARD

We only fit top of the range solid-state drives in our servers to ensure blazing fast speeds and exceptional reliability.

AUTOSCALING

Our platforms scale resources on demand to ensure your website stays online and fast, even during the busiest times.

MULTIPLE PLATFORMS

Host websites on performance-tuned Linux, Windows and WordPress optimised platforms.

UNIQUE TO 20i

100% BRANDABLE

Give your customers their own customised control panel. It's sleek, packed with features, and user-friendly.

UNIQUE TO 20i

VISIT US
20i.com/design

EMAIL US
sales@20i.com

CALL US
0333 344 2733

Welcome to the issue

THE WEB DESIGNER MISSION

To be the most accessible and inspiring voice for the industry, offering cutting-edge features and techniques vital to building future-proof online content

Steven Jenkins

Highlight

JavaScript has developed far enough that mostly jQuery isn't really needed anymore

Digital pioneers wild tell us they are heading to the frontier of digital design

Page 36

CSS is growing up

CSS was a sensitive soul, once an awkward teenager, but now, with the help of browser support, it is starting to grow and realise its potential.

Our latest lead feature (page 44) takes a closer look at the latest tools and rules that are taking the king of styling forward to a brave new era. The darling of the moment CSS Grid is here. We take a look at positioning and offer a host of useful tips and tricks to get more from the spec. Discover the benefits that Custom Properties, or CSS Variables if you prefer, will bring to your coding. Plus, find out what you can do with CSS Element Queries, how to build with the Web Animations API and how to speed up your website with Critical CSS.

While CSS is bigging itself up JavaScript frameworks have been building in abundance. There is no shortage of them, in fact, and with so many it's hard to decide which one to use. Help is at hand on page 66. We take a look at the features of the more popular frameworks, Angular, React, Ember, Meteor, plus a few up-and-coming options, and help you choose the right one for your projects. Talking of JS frameworks we show you how to install and get started with the popular Vue.js on page 74.

Elsewhere learn how to utilise React to build a dashboard application, find out what's new in Angular 4 and create custom social media buttons. Enjoy the issue.

> Many of us have been using variables in CSS for a few years through the use of preprocessors like Sass and LESS, but now we can use them in native CSS

Follow us on Twitter for all the news & conversation @WebDesignerMag

Visit our blog for opinion, freebies & more www.gadgetdaily.xyz

FREE – exclusive with this issue

78 Designer resources

Video Tuition – 72 minutes of expert video guides on HTML/CSS/JS troubleshooting from Pluralsight (www.pluralsight.com)

Assets – 16 Orange and Teal Photoshop actions
70 vintage wood textures
– Tutorial files and assets

FileSilo

www.filesilo.co.uk/webdesigner

This issue's panel of experts

Welcome to that bit of the mag where we learn more about the featured writers and contributors...

"With full spec implementation blunders out of the way, CSS has a bunch of exciting new features on the horizon"

Steven Roberts

Steven is a front-end developer from Teesside. In this article he'll be taking a look at what's new, upcoming and exciting in CSS. He offers top tips on using CSS Grid, how to use Custom Properties (aka CSS Variables), work with the Web Animations API & peek inside CSS Element Queries. **Page 44**

Mark Shufflebottom

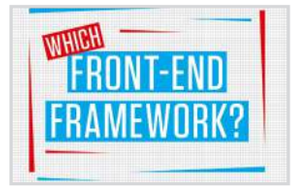
Mark is a Professor of Interaction Design at Sheridan College of Advanced Learning near Toronto. Mark is exploring the power of AR.js to bring augmented reality through WebRTC and WebGL to the browser. **Page 54**

Tam Hanna

Tam has an ambivalent relationship with JavaScript and is always interested in new frameworks promising to streamline the development process. This issue he gets up close and personal with the JavaScript framework Vue. **Page 74**

Simon Jones

Simon likes his code quick, clean and maintainable, which means modularity and well-defined design principles. Frameworks help bring these benefits to the web. In this issue he helps you decide which is the best framework for you. **Page 66**



Sarah Maynard

Sarah is a freelance WordPress developer who likes to make colourful websites. In this issue she'll be giving a demonstration on making good-looking social media icons without diving into the backend of WordPress. **Page 60**

Jemmima Knight

Jemmima leads a team of software engineers, developing and maintaining enterprise-scale applications. This issue she takes a closer look at Angular and demonstrates how to make use of the latest release, 4. **Page 80**

David Howell

David is an experienced writer, author and journalist who runs his own company, Nexus Publishing. This issue he gets the lowdown on Austrian agency wild. He finds out how the digital pioneers operate on a day-to-day basis. **Page 36**

Matt Crouch

Matt is a front-end developer based in London. In this issue, Matt takes a beginner-friendly look at React and how to make a dashboard application you can use to keep up to date with important statistics within a project. **Page 86**

Leon Brown

Leon is a freelance web developer and trainer who assists web developers in creating efficient code for projects. This issue he reveals a host of techniques, as seen in the top-class websites featured in our Lightbox. **Page 14**

Got web skills?
We're always looking for the hottest web-design talent. Email **webdesigner@imagine-publishing.co.uk** with examples of your creative work.

web designer

Future Publishing Ltd
Richmond House, 33 Richmond Hill
Bournemouth, Dorset, BH2 6EZ
☎ +44 (0) 1202 586200
Web: **www.gadgetdaily.xyz**
www.greatdigitalmags.com
www.futureplc.com

Editorial
Editor **Steve Jenkins**
steve.jenkins@futurenet.com
☎ 01202 586233
Designer **Harriet Knight**
Production Editor **Rachel Terzian**
Photographer **James Sheppard**
Senior Art Editor **Will Shum**
Editor in Chief **Amy Hennessey**

Contributors
Mark Stuckert, Arabella Lewis-Smith, Annie Hall, Mark Billen, Leon Brown, Deutsch, wild, David Howell, Steven Roberts, Mark Shufflebottom, Sarah Maynard, Simon Jones, Tam Hanna, Jemmima Knight, Matt Crouch

Advertising
Digital or printed media packs are available on request.
Commercial Sales Director **Clare Dove**
Advertising Manager **Mike Pyatt**
☎ 01225 687538
michael.pyatt@futurenet.com
Account Director **George Lucas**
george.lucas@futurenet.com
Advertising Sales Executive **Chris Mitchell**
chris.mitchell@futurenet.com

Licensing
Web Designer is available for licensing. Contact the licensing department to discuss further opportunities.
Head of International Licensing **Matt Ellis**
☎ +44 (0) 1225 442244
matt.ellis@futurenet.com

Subscriptions
For all subscription enquiries:
webdesigner@servicehelpline.co.uk
☎ 0844 848 8413
☎ Overseas +44 1795 592 878
www.imaginesubs.co.uk
Head of subscriptions **Sharon Todd**

FileSilo.co.uk
Assets and resource files for this magazine can be found on this website.
Register now to unlock thousands of useful files.
Support filesilohelp@futurenet.com

Circulation
Circulation Director **Darren Pearce**
☎ 01202 586200

Production
Head of Production US & UK **Mark Constance**
Production Project Manager **Clare Scott**
Advertising Production Manager **Joanne Crosby**
Digital Editions Controller **Jason Hudson**
Production Manager **Nola Cokely**

Management
Finance & Operations Director **Marco Peroni**
Creative Director **Aaron Asadi**
Art & Design Director **Ross Andrews**

Printing & Distribution
Printed by Southernprint Ltd, 17-21 Factory Road Upton Industrial Estate, Poole, Dorset, BH16 5SN

Distributed in the UK, Eire & the Rest of the World by Marketforce, 5 Churchill Place, Canary Wharf, London, E14 5HU
☎ 0203 787 9060 www.marketforce.co.uk

Distributed in Australia by Gordon & Gotch Australia Pty Ltd, 26 Rodborough Road, Frenchs Forest, New South Wales 2086
☎ + 61 2 9972 8800 www.gordongotch.com.au

Disclaimer
The publisher cannot accept responsibility for any unsolicited material lost or damaged in the post. All text and layout is the copyright of Future Publishing Ltd. Nothing in this magazine may be reproduced in whole or part without the written permission of the publisher. All copyrights are recognised and used specifically for the purpose of criticism and review. Although the magazine has endeavoured to ensure all information is correct at time of print, prices and availability may change. This magazine is fully independent and not affiliated in any way with the companies mentioned herein.

If you submit material to Future Publishing via post, email, social network or any other means, you automatically grant Future Publishing an irrevocable, perpetual, royalty-free licence to use the material across its entire portfolio, in print, online and digital, and to deliver the material to existing and future clients, including but not limited to international licensees for reproduction in international, licensed editions of Future Publishing products. Any material you submit is sent at your risk and, although every care is taken, neither Future Publishing nor its employees, agents or subcontractors shall be liable for the loss or damage.

© 2017 Future Publishing Ltd

ISSN 1745-3534

recycle
When you have finished with this magazine please recycle it.

Future
Future is an award-winning international media group and leading digital business. We reach more than 57 million international consumers a month and create world-class content and advertising solutions for passionate consumers online, on tablet & smartphone and in print.

Future plc is a public company quoted on the London Stock Exchange (symbol: FUTR).
www.futureplc.com

Chief executive Zillah Byng-Thorne
Non-executive chairman Peter Allen
Chief financial officer Penny Ladkin-Brand

Tel +44 (0)1225 442 244

Save on broadband, TV, phone!

Just some of our friendly UK based team

Save up to £577* per year

PRICE COMPARISON CALCULATOR APPROVED BY Ofcom

Let us do the hard work for you!
All it takes is one quick call.

83% of our customers save!**

1. Call free and speak to a UK based impartial expert.
2. We review over 6,000 deals from the top UK providers to find the right one for you.
3. Our experts do the hard work, answer your questions and even arrange your installation!

THIS MONTH'S TOP DEALS

Best TV, Broadband & Phone Bundle simplify digital 2010 AWARDS

Call us for best value up to 20Mb broadband + TV + calls package

sky

Best Super-fast Broadband simplify digital 2010 AWARDS

Call us for a great value, superfast fibre-optic broadband deal

Virgin media

Call us free and we'll do the hard work!

0800 840 1712

Mon-Fri 8am-8pm, Sat 9am-5.30pm, Sun 10am-5.30pm

www.techradarstores.co.uk/digitalsave

simplify digital

*20% of Simplifydigital customers received an average saving of £577. Data based on 484 records between 1st January 2010 and 17th May 2010. ** Out of 445 consultations between 01/01/2010 and 03/05/2010, 368 Simplifydigital customers saved money by signing up to our recommended deal.

inside issue 262

contents

Cutting-edge features, techniques and inspiration for web creatives

Chat with the team and other readers and discuss the latest tech, trends and techniques. Here's how to stay in touch...

webdesigner@imagine-publishing.co.uk @WebDesignerMag www.gadgetdaily.xyz

Quick look...

08 Browsers and the current state of support
If only browser support was the same for all. We look at what's going on with the latest versions

10 Webkit: The best must-try resources out there
Discover the libraries and frameworks that will make your site a better place to visit

11 Fantasy UX
Lead designer at Sequence, Mark Stuckert, talks fantasy inspiration for real-world UX

14 Lightbox
A showcase of inspirational sites and the techniques used to create them

28 Walking on water
Discover how creative agency Deutsch delivered a serious message via an engaging digital experience

36 Wild design
With an insatiable appetite to discover the frontier of digital design, **Web Designer** simply had to talk to prospective pioneers wild

44 New-generation CSS
Work with CSS Grid, code Custom Properties and build with the Web Animations API

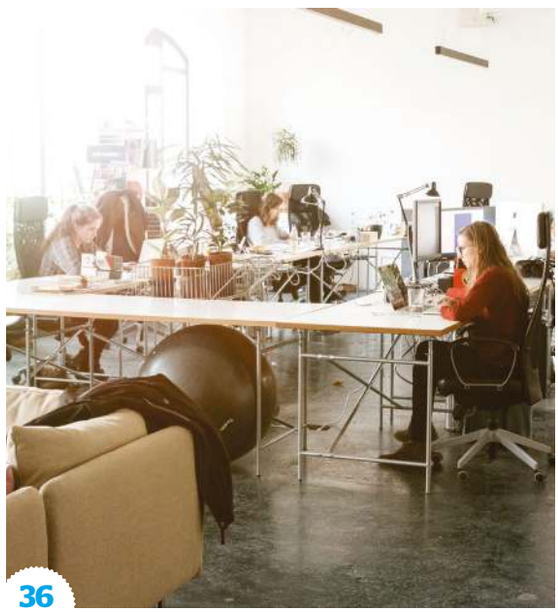
66 Front-end frameworks: make the right choice
With so many options out there, which one do you need? We help you make the best choice

74 Get started with Vue.js
Find out how to install and start using the popular JavaScript framework

92 Course listings
Want to start learning online? Check out what courses are out there with this list

94 Hosting listings
An extensive list of web hosting companies. Pick the perfect host for your needs

98 Next month
What's in the next issue of **Web Designer?**



ProFile: wild
Pioneering folk at the frontier of digital design

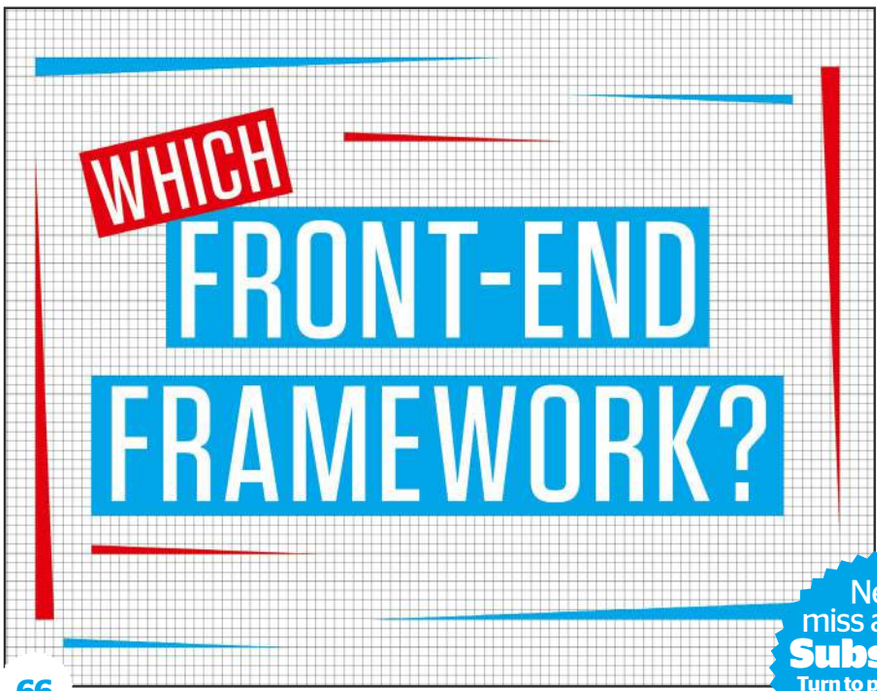
Lightbox: Three Cents
Selling through illustration and animation

FileSilo

96 Get the latest must-have resources and videos

A comprehensive collection of free designer resources!

- 72 minutes of troubleshooting video guides from Pluralsight
- 70 vintage wood textures
- 16 Orange and Teal PS actions



66 Which front-end framework do you need? We help you decide

Never miss an issue
Subscribe
Turn to page 34 now
USA readers turn to page 85 for an exclusive offer

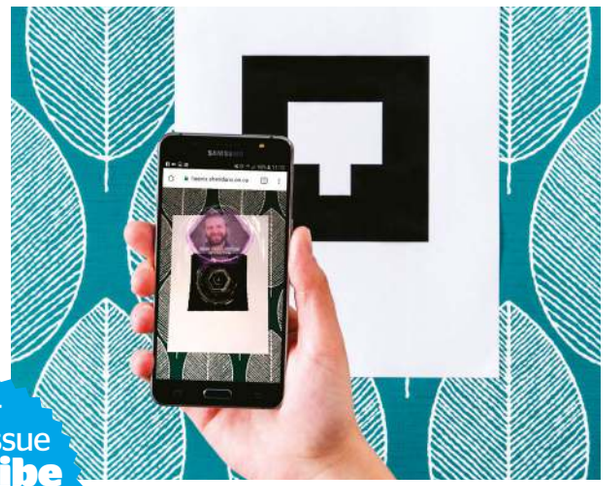
Visit the WEB DESIGNER online shop at
Future myfavouritemagazines
myfavouritemagazines.co.uk
for the latest issue, back issues and specials

<tutorials>

Web gurus take you step-by-step through professional techniques

54 **Code augmented reality markers**
Learn how augmented reality can be achieved in the browser using native web technologies

60 **Create custom social media buttons**
Create great social media buttons using the WordPress menu tool, built-in icon fonts and a little CSS



<header>

The tools and trends to inspire your web projects

08 **The current state of browser support**
What's being supported and what's not

10 **Webkit: The best must-try resources out there**
Need a new framework or library? Then this is where you should start

11 **Comment: Mark Stuckert**
Lead designer at Sequence discusses how fantasy inspiration could one day affect real-world UI and UX

> Nowadays we even have to avoid HTML code as hacking text. Audiences are just too easily pulled out of the illusion

Mark Stuckert

web workshop

52 **Animate multiple elements**
tram-house.com
Introduce multiple animations across a single page

58 **Code draggable navigation**
poligraph.live
Let the user take control

64 **Create interactive navigation**
ivanaivazovsky.life
Implement different background images on hover

web developer

74 **Get started with Vue.js**
Find out how to install and start using the popular JavaScript framework

80 **Say hello to Angular 4**
There's been some fundamental changes to how Angular works. See how they can benefit your application

86 **Create a dashboard app with React**
Make the most out of reusable components to create a highly flexible data visualisation screen

Header

The tools, trends and news to inspire your web projects

The current state of browsers

What browser supports what, and which browsers are people using? **Web Designer** takes a closer look

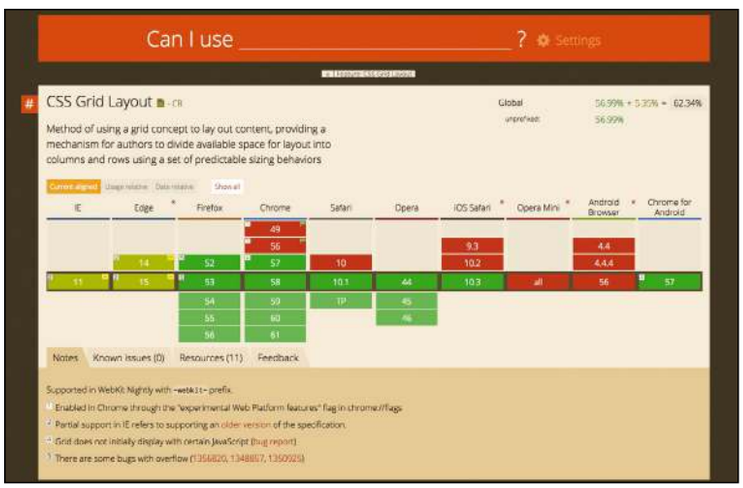
New specifications are hamstrung by browser support. Until the moment that major browser vendors decide to offer/include support for a new feature, it is effectively a plaything. You can try a feature out, but using it in production is still a no-no. CSS Grid is a good example of how the process works. Firefox and Chrome have offered full support since the beginning of March 2017, and Safari 10.1 introduced it at the end of March 2017, around the same time as Opera 44. Microsoft Edge 15 does offer support, but for an older specification. On mobile Chrome for Android it is fully supported and so is Safari on iOS. The actual CSS Grid specification had a working draft as far back as 2011, but with support now (almost) universal it is quickly being adopted by designers and developers everywhere. This is a typical scenario and one that all designers and developers have to contend with continuously.

Here we take a quick look at the current state of browsers and see what their latest versions have to offer designers and developers. Let's start with everyone's favourite, Microsoft Edge. The latest version at the time of writing (this applies to all browsers in the article) is 15. So, what does this have included in the latest version? Notable additions are preview support for the new Payment Request API, added support for CSS Custom Properties (aka CSS Variables) and WebRTC 1.0 enabled by default. Find out more at **bit.ly/2q4Xnzu**. The latest version of Chrome includes full screen for progressive web apps, Firefox includes JSON Viewer enabled by default and a selection of tweaks. Check out the full list at **developer.mozilla.org/en-US/Firefox/Releases/53**. Safari 10.1 has support for Custom Elements, the Gamepad API and CSS Grid. Check out more at **apple.co/2o8cqdp**. If looking for a specific feature, for example WebGL, the best place to find out if support exists is the popular Can I Use site over at **caniuse.com**.

Adoption of a specification by the browser vendors is when it becomes a reality. But, this still doesn't solve the old issue of whether you should wait for total support or simply start using a feature when the major browsers have support. Does anyone still consider IE11 when thinking of support? It's not hard to guess what the most popular browser is – Google Chrome. When it comes to worldwide usage it is streets ahead. According to StatCounter, Chrome has over 53% market share for all devices. Change this to desktop and mobile and it is still much the same. Safari comes in a distant second with 12%, UC Browser 8% and Firefox 7%. Change location to Europe and Chrome stays out in front, but Firefox and Safari come in with around 12% of the market share. Swap to North America and again Chrome is way out in front with over 50%, but Safari comes in with 25% while Firefox is below 10%. In Africa Chrome has taken over from Opera in the last year. Lots of information to absorb and lots of permutations to consider, so browse the stats at your leisure at **gs.statcounter.com**.

We at **Web Designer** look forward to the day when we can wonder the web with a utopian vision of the browser. But, until that day (will it ever happen?) we will sit back, adopt the major browsers (especially Chrome) and pretend the rest don't exist. It may prove to be foolish, but we will be a lot saner for the choice.

> ***"When it comes to worldwide usage it is streets ahead. Chrome has over 53% market share for all devices"***



If you need to know if a specific element is supported in a browser, caniuse.com should be your first port of call

Get Web Designer digital editions

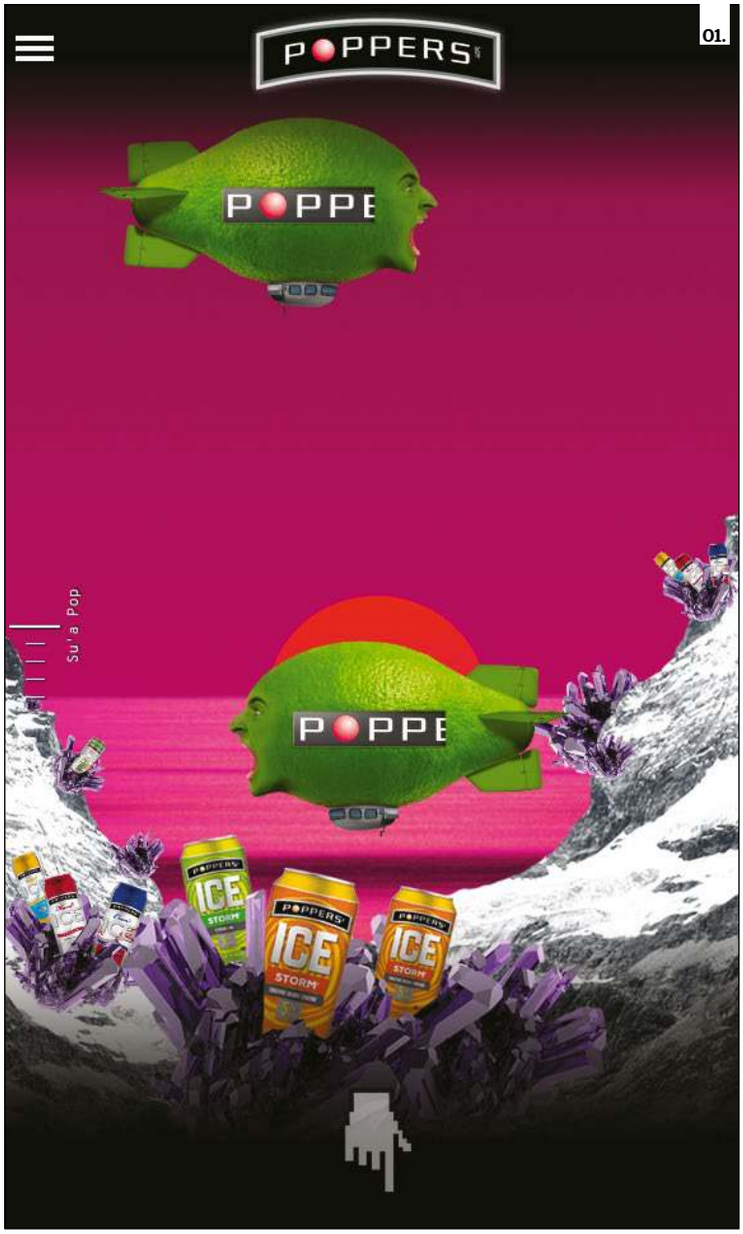
Get your hands on a collection of the best contemporary techniques

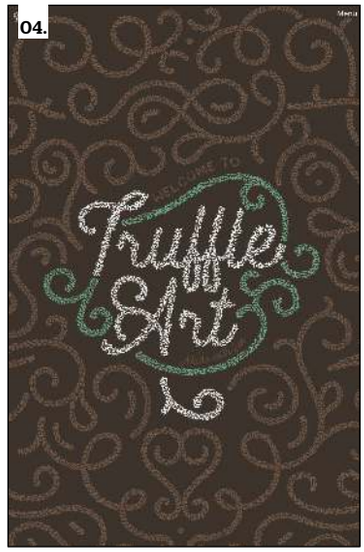
Need an instant hit of **Web Designer**? Then check our digital specials. *The Web Apps Handbook Volume 2* and *The Developer's Guide to JS* reveal the best tools and methods to build apps with web technologies and learn how to get started with ES6 and NPM.

Don't forget *Animate with HTML & CSS* and *3D and the Web*. We cherry-pick a collection of the best animation and 3D-related techniques to add interest to any project.

Plus, check out *Advanced Angular, 20 Quickfire HTML & CSS Techniques* and *Amazing Websites and How to Build them Vol 04*. How can you get your hands on a **Web Designer** digital edition? Head to **bit.ly/1hsGYgl**, download the free **Web Designer** app and find them waiting for you as an in-app purchase.

Sites of the month





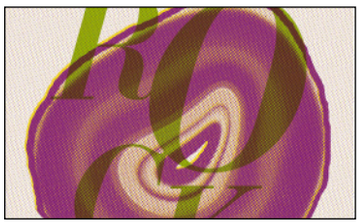
01. Poppers
popperscoolers.com/fr
Lots of design craziness here. Plenty to keep the viewer amused and engaged.

02. Qantas Dreamliner
dreamliner.qantas.com/explore
Discover what the inside of a Dreamliner looks like in this interactive 3D experience.

03. I Buongiorno
ibuongiorno.com/en
A vertical and horizontal scrolling experience guides the user to tell its tale.

04. Truffle Art
truff.co
Smart, smooth animations coupled with imaginative transitions and design.

Graphics
Halftone Posters
bit.ly/2p3Pg9q
We love a halftone graphic. Check out this collection of posters from the talented hand of Jake Arnold.



Colour picker
thinking of you...
bit.ly/2pd1E1U

#D4DE25
#96CB65
#63A765
#3C8E8B
#E26DAA

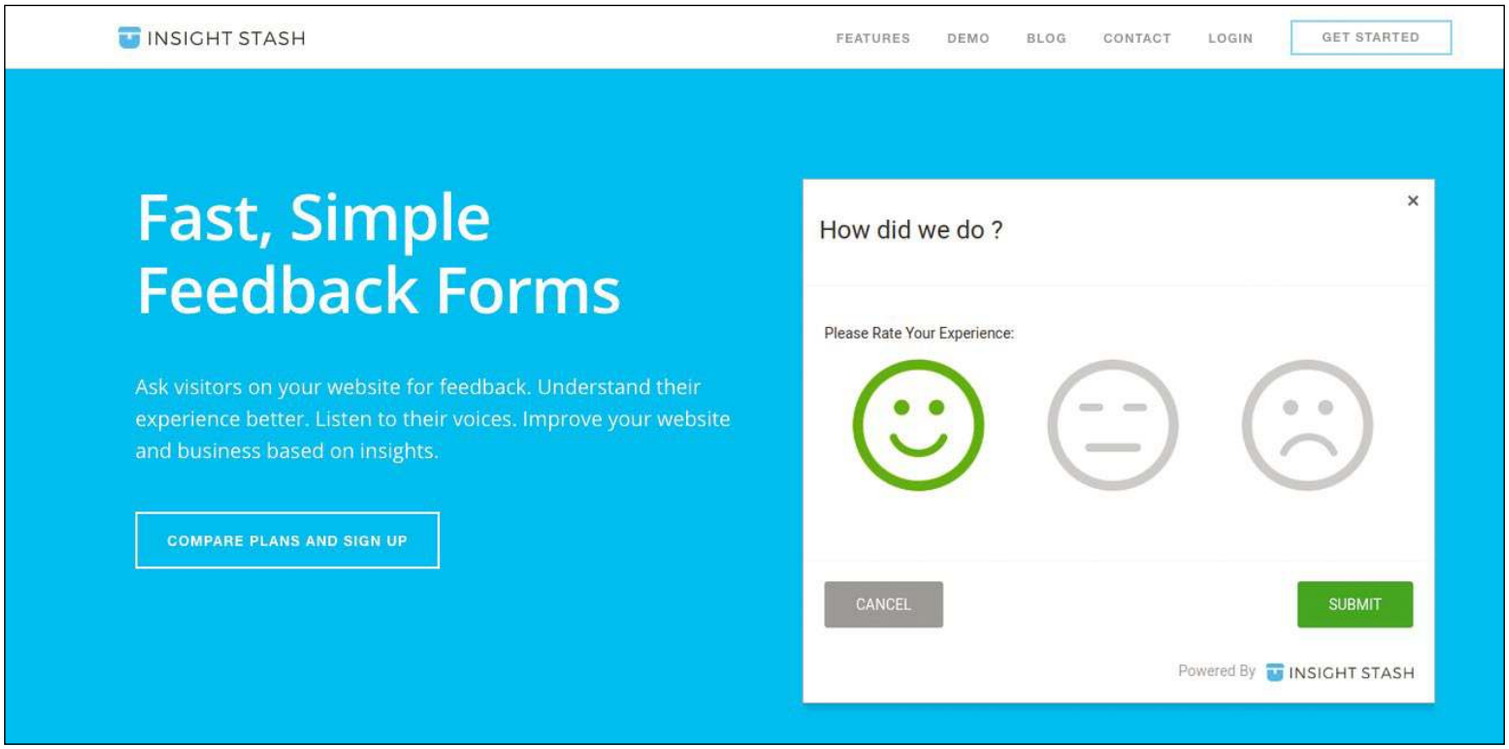
Typesetter
Goldana
bit.ly/2qKUQcQ
Inspired by Art Deco, Goldana is very versatile with a host of options to make it three dimensional.

WordPress
Seese
bit.ly/2qp9slH
eCommerce theme with many talents, contemporary layouts and colour schemes, plus some very useful search features.

CONTACT US AT: webdesigner@imagine-publishing.co.uk | @WebDesignerMag

webkit

Discover the must-try resources that will make your site a better place



Insight Stash

insightstash.com

Every site needs feedback. Giving the user what they want means that more people will visit, use and appreciate your site and ultimately come back. This software uses a familiar and intuitive interface to help build surveys, polls and testimonials. Users can customise the look by uploading their own custom CSS files and include their own branding.

Prepack

prepack.io

Prepack is a tool to help you optimise your JavaScript code. Currently in early-stage development, help out and tell them what you think.

SmartPhotos.js

appleple.github.io/SmartPhoto

An easy-to-use responsive image viewer especially for mobile devices. Features intuitive gestures and offers accessibility from keyboards and screen-readers.

CSS Peeper

csspeeper.com

A Chrome extension with a touch of style, its purpose is to 'Extract CSS and build beautiful styleguides.' A viewer that allows designers to focus on design.

TOP 5 Web conferences – June 2017

Get yourself a seat at the biggest and best conferences coming your way soon

Web Rebels

www.webrebels.org

There's a great selection of speakers at this two-day, one-track event in Norway, with 16 speakers in total.

Generate

bit.ly/2qSmdC4

The conference heads to San Francisco and brings with it a host of speakers across two tracks for one day.

UX Scotland

uxscotland.net/2017

A UX and design conference that 'allows participants to connect and learn from their peers and leaders in the industry.'

Kerning

2017.kerning.it

Two days of workshops and a one-day conference in the heart of Italy dedicated solely to typography.

Patterns Day

patternsday.com

An event for web designers and developers on design systems, pattern libraries, style guides and components.

Real-world UX and fantasy inspiration

Mark Stuckert explains how fantasy is evolving to outrun real-world UI

Mark Stuckert | Lead designer | thesequencegroup.com

Designing UI for the real world is no easy challenge. You need to consider easy-to-use navigation, functionality, readability and more. But when it comes to designing UX for sci-fi and fantasy, we are swamped with a realm of possibilities. The differences between fictional and real-world UX are huge. When we have a scene that needs to be filled with fantasy UI, we can just fill it. The images don't have coding behind them. Unlike real-world UX, it doesn't require any practical purpose, it just needs to look good.

Our biggest challenge now is that audiences are so desensitised to UI – if the smallest detail isn't believable, they are easily pulled out of the story. From phones to consoles, daily life at the office and even self-service at the supermarket, audiences know what is believable and what is not. Compared to 15 years ago, viewers are far more easily pulled out of the fantastical experience.

Our team has a lot of experience in designing UI/UX for motion comics, art and animation. It's really important for us, and anyone creating concepts for the fantasy and sci-fi landscape, to make sure UI aligns with that world. If I'm watching an animation and the screen flashes 'downloading virus', I'm immediately pulled out of the experience because it is inherently unbelievable in any context. In addition, if we were to create UX for a real-world setting such as live action – with flesh and blood actors – overly fantastical UI can break that experience. You need to consider each medium differently when designing UI. Nowadays we even have to avoid HTML code as hacking text. Audiences are just too easily pulled out of the illusion from these small, yet crucial, elements.

As part of this trend, we have also seen skeuomorphism become outdated – something I am personally glad to see. Fantasy UI followed this trend, which really broke out with *Oblivion*. I think that was a watershed moment for FUI, and we're still continuing with the trend. Fantasy design is becoming more real and aesthetically functional in appearance.

> **"Our biggest challenge now is that audiences are so desensitised to UI – if the smallest detail isn't believable, they are pulled out of the story"**

When we begin a project, we create team rules and guidelines based on the new material. When we created UI for *Halo: The Fall of Reach*, we had a lot of franchise history to pull from – colour, line work, typography and more. The next step was for us to create a UI that looked good in the specific sequences. We take into consideration whether the UI will be in focus or out of focus in the shot. If it's the main focal point, it will relate to the story and have a purpose – so looking good is a key concern.

While art is an obvious inspiration, we also pull a lot from real-world technology – places you don't expect! I prefer using a lot of older, industry-specific references and operating systems as inspiration. For example, I use reference images from software used by the US Air Force and NASA. These are programs that aren't consumer-facing and appear complex to average audiences. I even turn to some operating systems for inspiration, including macOS, iOS, early versions of Windows and really old Xerox interfaces. Older legacy examples are busy and complex; they easily amalgamate into a sci-fi setting.

When creating the UI, we likewise think about the overall shape of what we're trying to build. In some sci-fi universes, a futuristic shape can have a lot of extraneous angles, edges, details and other random bits. In others, like *Ex Machina* and *Batman V Superman*, the interfaces look like a Dieter Rams version of macOS or Windows. It's about what works in that world. In animation, it's about how the little assets interact and make the whole screen feel both alive and realistic.

That said, if you look at an actual computer screen there's probably not much animating or blinking – you have to dramatise it for film and cinematics. Another option is creating a hyper-minimal interface, which means you don't necessarily need a lot going on.

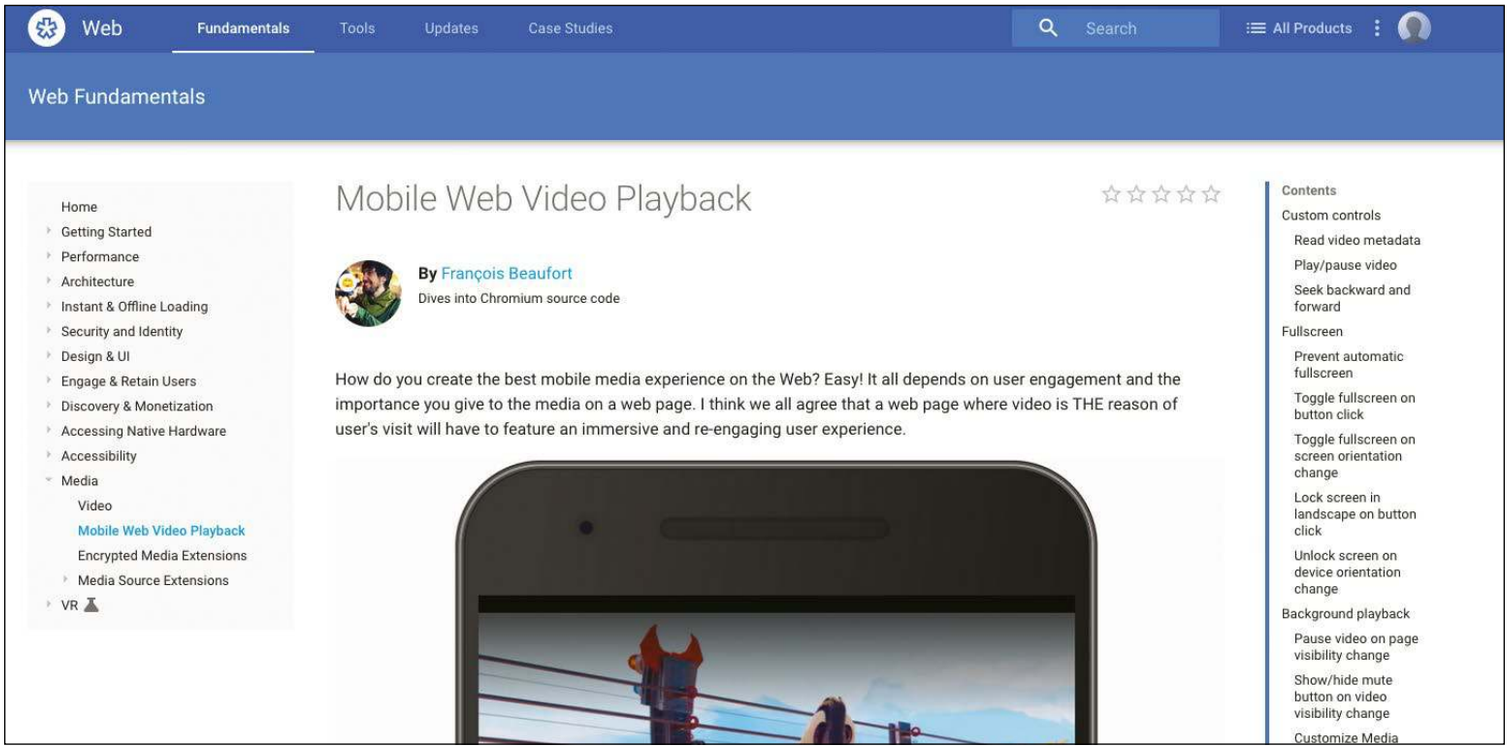
In short, we no longer need to sell the 'computer-ness' of technology in fantasy and sci-fi, unlike when desktop computers were first emerging and many people didn't have a working knowledge of them. Our target audiences all know about interfaces now, so we have a lot more freedom to get creative.

Both fantasy and real UI/UX are developing faster and faster as technology becomes more mainstream. Future UI is going to continue delving into every dimension of the modern world going forward – from phone UI to computers and smart cars. I envision a convergence of UI and fantasy UI in the years to come.

CONTACT US AT: webdesigner@imagine-publishing.co.uk | @WebDesignerMag

webkit

Discover the must-try resources that will make your site a better place



Mobile Web Video Playback

bit.ly/2qJIu8u

How do you create the best mobile media experience? This in-depth guide from Google Developers delves into a host of techniques that will help ensure that the video experience on mobile is spot on.

ReactXP

microsoft.github.io/reactxp

This is a 'library for building cross-platform apps.' It enables developers to share most of their code between web, iOS, Android and Windows.

stylelint

stylelint.io

A modern CSS linter to help you enforce consistent conventions and avoid errors in your stylesheets. Use via a host of plugins for most tools and text editors.

Barba.js

barbajs.org/index.html

A small (4.4kB minified and gzipped), flexible and dependency-free library that helps you create fluid and smooth transitions between your site's pages.

TOP 5 WordPress e-commerce themes

Make the online shopping experience more enjoyable with this collection of themes

Handmade

bit.ly/2pYtEbm

A big image slider takes centre stage with offers and products. Sitting neatly underneath is a new-product carousel.

Gon CakeShop

gonthemes.com/cakeshop

Visual impact is king here, from the full-width hero image through to the gallery, featured products and latest blog.

Sash

althemist.com/sash

Big full-width hero image and associated buttons get straight to the point and give the viewer an easy-to-make decision.

MaxStore

maxstore.kutethemes.net

A contemporary layout is brought up a level by smart positioning of call-to-action boxouts and simple designs.

More

bit.ly/2pnkjlq

A grid-based layout that wastes no time is getting the user to click on a product. Includes neat and engaging transitions.

Web design – art or science?

In the highest league of web design, does data trump design or is creativity king?

Arabella Lewis-Smith (Founder) **& Annie Hall** (Planner) **at Salad** | saladcreative.com

Annie: Data-driven design often gets a bad rep in industry discourse – Google's 41 shades of blue testing is a well-cited example. However, there's no getting away from the fact that digital product design must be user-centric. The most commonly agreed stat suggests we have 50 milliseconds to make a positive impression on a web visitor and that can't be achieved by design alone.

If a user has to wait or work too hard to find what they're looking for, if there's too many clicks between making a decision and acting on it, or the landing page simply doesn't seem to match the search they entered to get there – you've already lost them.

And your users aren't the only people who will punish you for getting this wrong. Google will too. Google wants to remain the most popular search engine by serving the best websites. Google penalises sites that fail to meet ever-changing standards around content, design, relevance and speed.

> **"Truly great design is better for business. It builds excitement and loyalty amongst your customers"**

To make matters more complicated, user behaviour changes as well. The ever-widening range of mobile, platform and connected device is impacting the way we shop and search. Web design must be adaptable and responsive. Making use of A/B testing and agile development can ensure a level of longevity that creative ad campaigns, by comparison, can't live up to.

Whether you've been working on the same site for decades, or work on a different site every month, understanding the goals and perceptions of any possible visitor over time takes more than creative intuition. Assumptions and ideas must be continuously tested and validated. This requires thoughtful research methods, objective experimentation and insight gathered quickly and at scale.

In 2016, it was reported that 40 per cent of all internet users – over one billion people – had bought products online, and UK e-tail sales reached over £52.25 million in 2015. Yet research has found just a one-second delay in website loading time can result in a seven per cent loss in conversion. In fact, 40 per cent of web users will abandon a website if it takes longer than three seconds to load.

No matter how practised an art may be, when there's that much riding on your website's performance, can you really afford to trust creative hands over a scientific mind?

Bella: While that's a valid argument, if every website were left to science alone, we would experience the death of beauty in the digital sphere. The science may be able to navigate people through a digital experience, but surely we can argue that the 'enjoyment' of that experience is largely due – at least in part – to the aesthetics. Colour, composition, the use of photography, film and even typography can influence the overall enjoyment we experience when navigating a website, because we are visual creatures.

Even the way we communicate with one another offline remains predominantly non-verbal – around 55 per cent is attained through our body language, facial expressions and so on. It would be dangerous to assume that just because we're communicating through a screen, we lose visitors' desire for clean, attractive and easy-to-follow design.

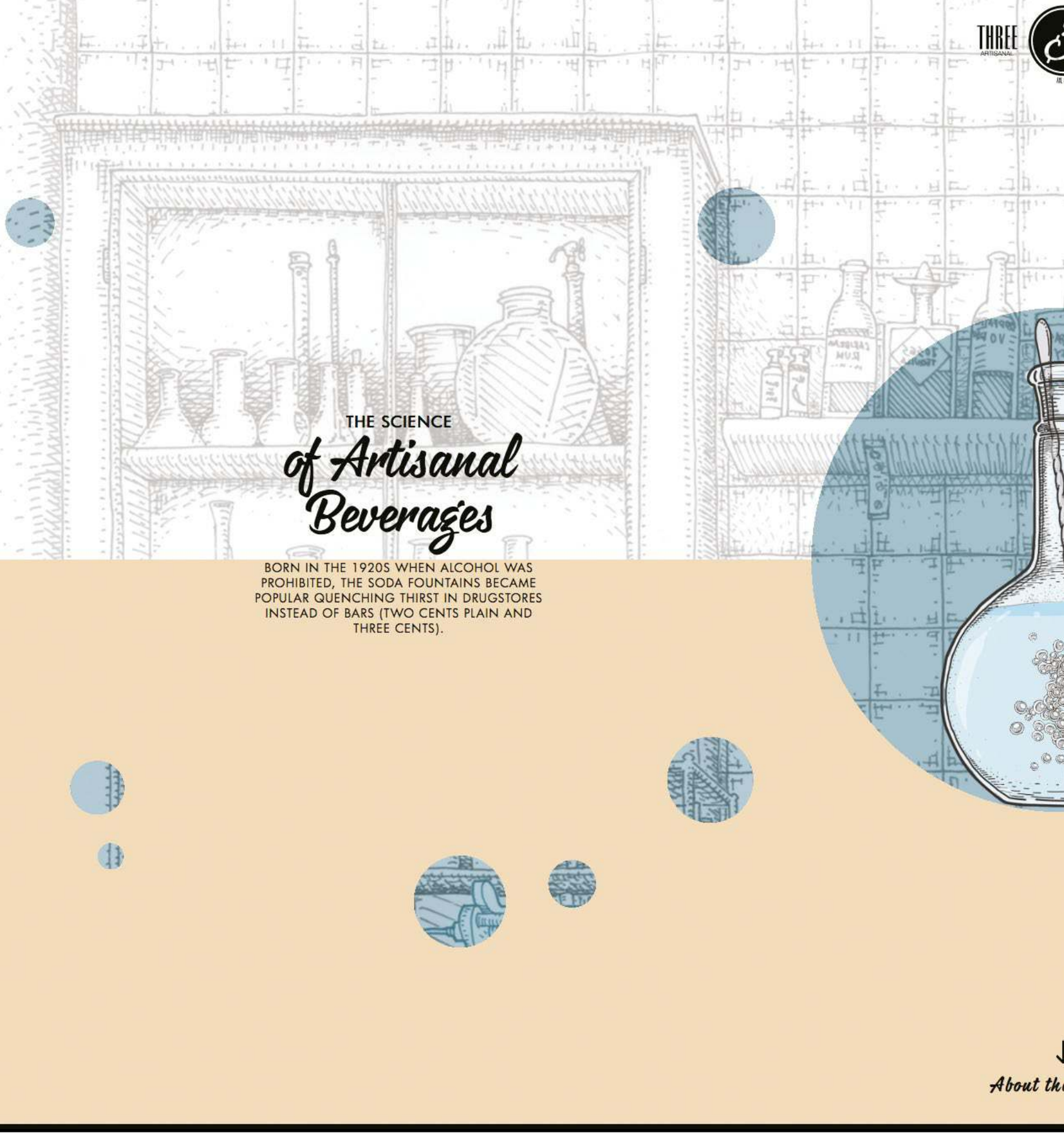
Beautiful, imaginative and innovative design transforms an intelligent digital experience from one of function to an enriching and joyful experience. Great design aims to make you feel something and the experience should ultimately influence the choices you make. Design is the reason a customer will select your product or service, because they have enjoyed the journey that got them to the choice.

Brands that invest in design are more attractive and ultimately sell more products, because by owning a beautifully designed product, customers feel it says something about them. Everyone enjoys beautiful design, whether it's a shirt, book sleeve or a San Pellegrino can – which, playing devil's advocate, you could argue is a Fanta in costume. The reality is that truly great design is better for business. It builds excitement and loyalty amongst your customers and pride amongst employees.

Ultimately, it can be argued that a laser-focus on the science of UX data is vital but only half of the picture. In the same way, web copy written purely to maximise organic SEO leaves little room for brand personality.

A mixed-method approach is always advocated, where quantitative data is evaluated in context alongside qualitative feedback and the gut-feel of our studio experts. When the science of design becomes too tactical (identifying the optimum shade of blue, for example) it begins to suffocate creativity capable of building new and exciting experiences. Analysts, designers and developers need to work collaboratively, which involves learning to speak each other's language.

Salad's value proposition 'Beautifully Effective' underpins everything and perfectly captures the promise made to clients. It's about striking the balance between beautiful design and measurable effectiveness in every project.



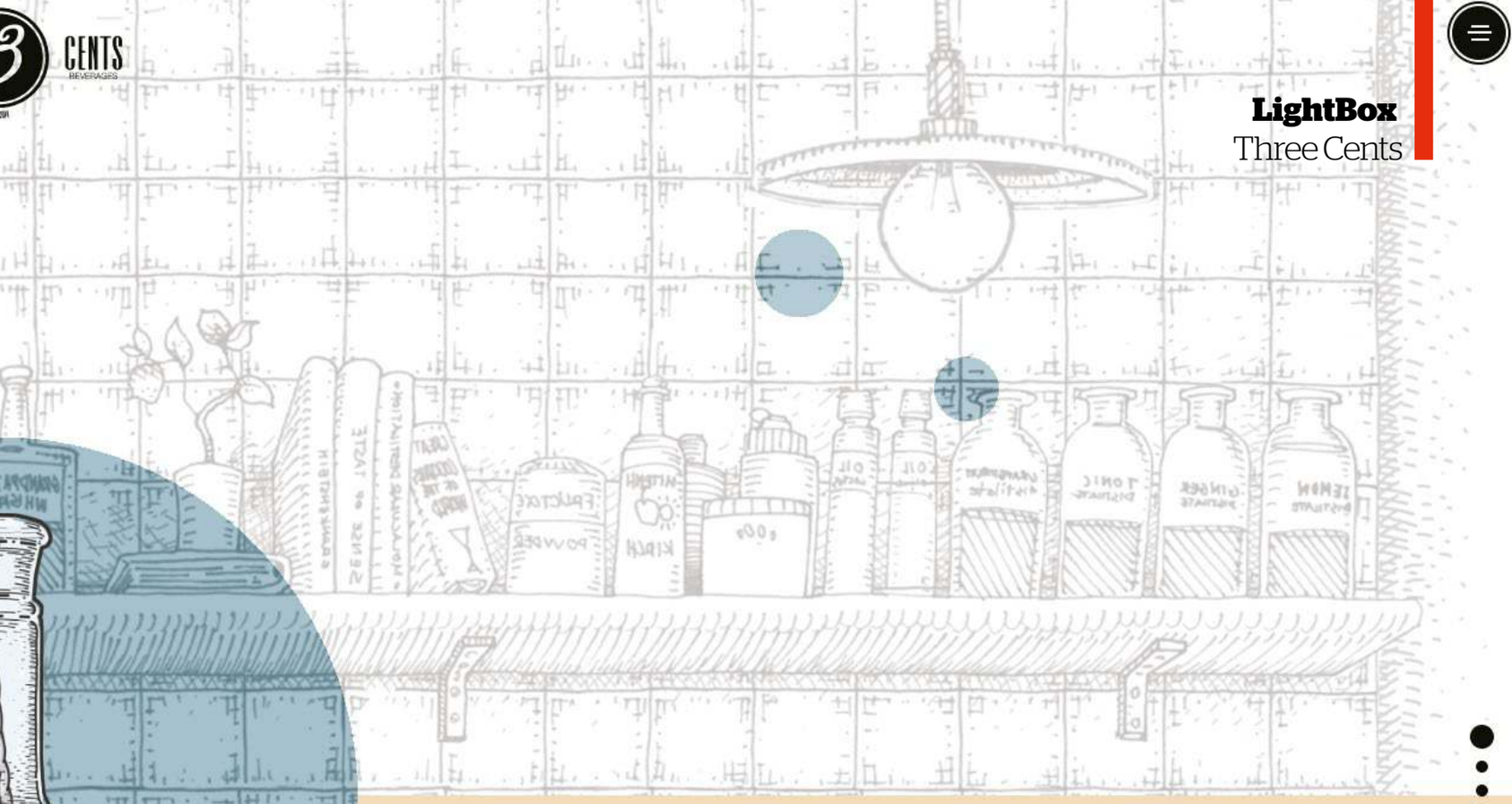
Three Cents

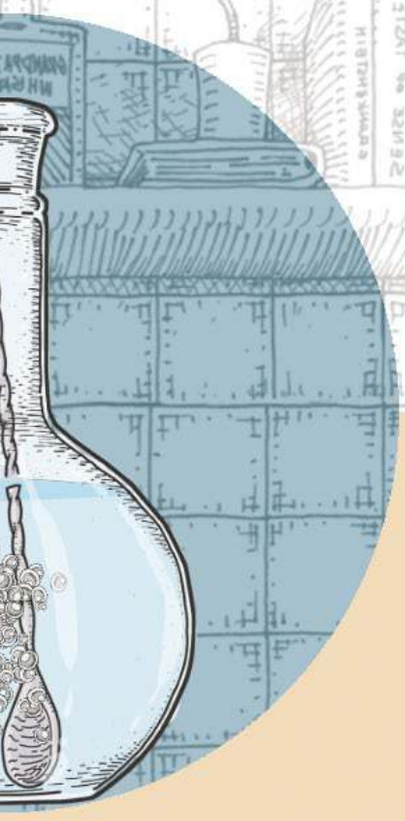
threecents.gr

Designer: Red Collar Digital – redcollar.digital

Development technologies WordPress, jQuery, WooCommerce, PHP

"Distinctively rustic in design, this promo site for a line of boutique beverages blends illustrations with subtle scrolling effects"





e Science

#FFFFFF

#ADC7D2

#EED9BA

#D1E6EF

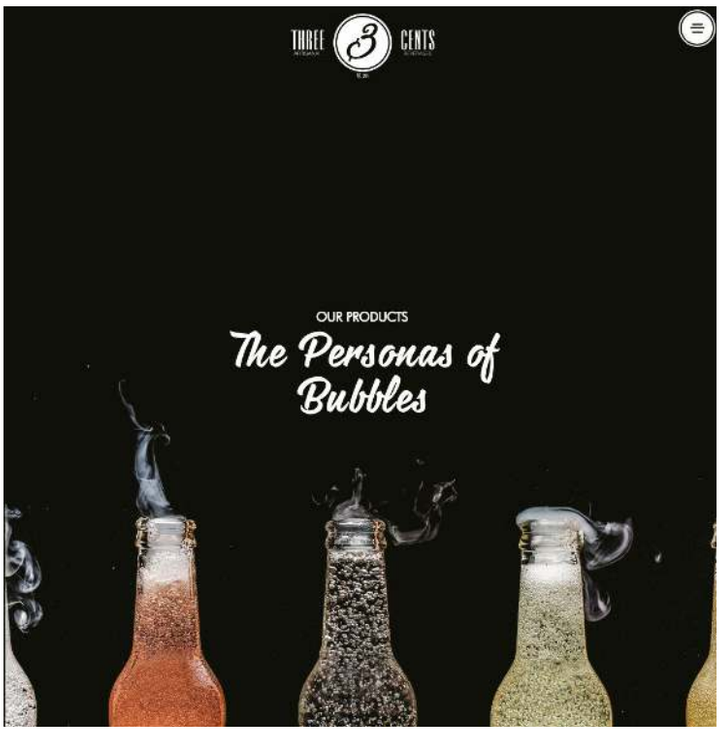
Above
Rukola font designed by Nikola Giacintová styles the distinctive script typeface that appears throughout the site

abcABC

Above
Futura BK Book by Neufville Digital and Linotype is used across all the site's remaining paragraph text

Above
Scrolling down the page brings into focus the individual flavours using subtle circular animations and random floating bubbles, which reveal elements of the background

Above
Smart and simple animations turn the bottle illustrations into real-life product photography



Above
The crisp, clean product photography provides a tempting insight into what the Three Cents product range has to offer

Create animated responses to navigation interactions

Use CSS transitions and target selector to present animation in response to user interactions

1. HTML document template

The first step is to initiate the project's HTML document. This contains the markup to define the general document structure that holds the visible content inside the body, along with the descriptive metadata and resource links inside the head section. The HTML content will be placed inside the body section in steps 2 and 3.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<meta http-equiv="X-UA-Compatible"
content="IE=edge">
<meta name="description"
content="description">
<title>Animate Into View</title>
<link rel="stylesheet" type="text/css"
media="screen" href="styles.css"/>
</head>
<body>
*** STEP 2
</body>
</html>
```

2. Body content: navigation

The first step for building the example content is the creation of the navigation links. These will be used to trigger the presentation of different content sections. In this case, the navigation links reference elements with ID 'p1' and 'p2' that will be defined in step 3. The ability to use the # symbol to link on page ID elements will enable us to create interactivity with CSS without the need for JavaScript.

3. Body content: articles

The page section contents are placed inside article containers that have ID attributes referenced by the links defined in step 2. Animated elements have a 'data-animate' attribute assigned to them, which will be referenced by CSS in later steps to apply their presentation and animation settings. The ability to specify data attributes like this allows for elements we want to animate to be given a unique name reference to use for their animation.

```
<article id="p1">
  <h1>Page 1</h1>
  <span data-animate="textItem">Text
item.</span>
  <span data-animate="pictureItem"></span>
</article>
<article id="p2">
  <h1>Page 2</h1>
  <span data-animate="textItem">Another
text item.</span>
  <span data-animate="pictureItem"></span>
</article>
```

4. CSS initiation

With the HTML now complete, create a new file called 'styles.css' and initiate its content with CSS to define the HTML document, content body and navigation containers. The main HTML document and its body is set to be fully visible with a black background and white text. The navigation needs to remain visible while visiting the different content sections, hence fixed positioning is used.

```
body,html {
  display: block;
  width: 100%;
  height: 100%;
  background: #000;
  color: #fff;
  padding: 0;
  margin: 0; }
nav{
  position: fixed;
  top: 0;
  width: 100%;
  z-index: 9999;
  background: #000; }
```

5. Articles & animated elements

The articles have a requirement for their animated elements to be animated relative to their position, so they use relative positioning. Child elements with the 'data-animate' attribute use absolute positioning, which becomes relative to their parent article. These elements become animated through the use of the transition rule applied. They are also placed out of view of the screen by default using the left attribute.

```
article{
  position: relative;
  height: 100vh;
  width: 100vw;
  padding-top: 1em;
  overflow: auto; }
article [data-animate]{
  position: absolute;
  transition: left 1s, top 1s, right 1s,
bottom 1s;
  left: -100vw;
}
```

6. Target page animation

The animated changes for article with ID 'p1' are defined in this step using the target selector. Individual elements are referenced using the 'data-animate' attribute, in which unique properties are applied. Left, top, right and bottom alterations appear animated due to the previously applied. The 'pictureItem' element is to be animated coming from the right instead of the left, hence its default left attribute is unset and replaced with a default right position.

```
#p1:target [data-animate="textItem"]{
  left: 10vw;
  top: 20vh;
  color: #fff; }
#p1 [data-animate="pictureItem"]{
  left: unset;
  right: -100vw; }
#p1:target [data-animate="pictureItem"]{
  right: 50vw;
  display: block;
  width: 20vw;
  height: 20vw;
  background: blue; }
```

7. More target animations

The same approach to animating the 'data-animate' elements is used for the article section with ID of 'p2'. These elements have different sizes, colours and positioning, which you could replace with your own settings. You can add as many elements as you want to the HTML and add their specific rules to the CSS.

```
#p2:target [data-animate="textItem"]{
  left: 20vw;
  top: 40vh;
  color: green; }
#p2 [data-animate="pictureItem"]{
  left: unset;
  right: -100vw; }
#p2:target [data-animate="pictureItem"]{
  right: 20vw;
  display: block;
  width: 20vw;
  height: 10vw;
  background: red; }
```

Enhance usability

Consider how the use of animated responses to navigation interactions can help improve the usability or understandability of the content.

Alcanyte

alcanyte.io

Designer: Hailiax – hailiax.io

Development technologies HTML5, jQuery UI, Modernizr, SoundCloud

"This dynamically animated call to arms hopes to connect non-profit organisations with digital designers seeking professional work experience"

#89FFFF

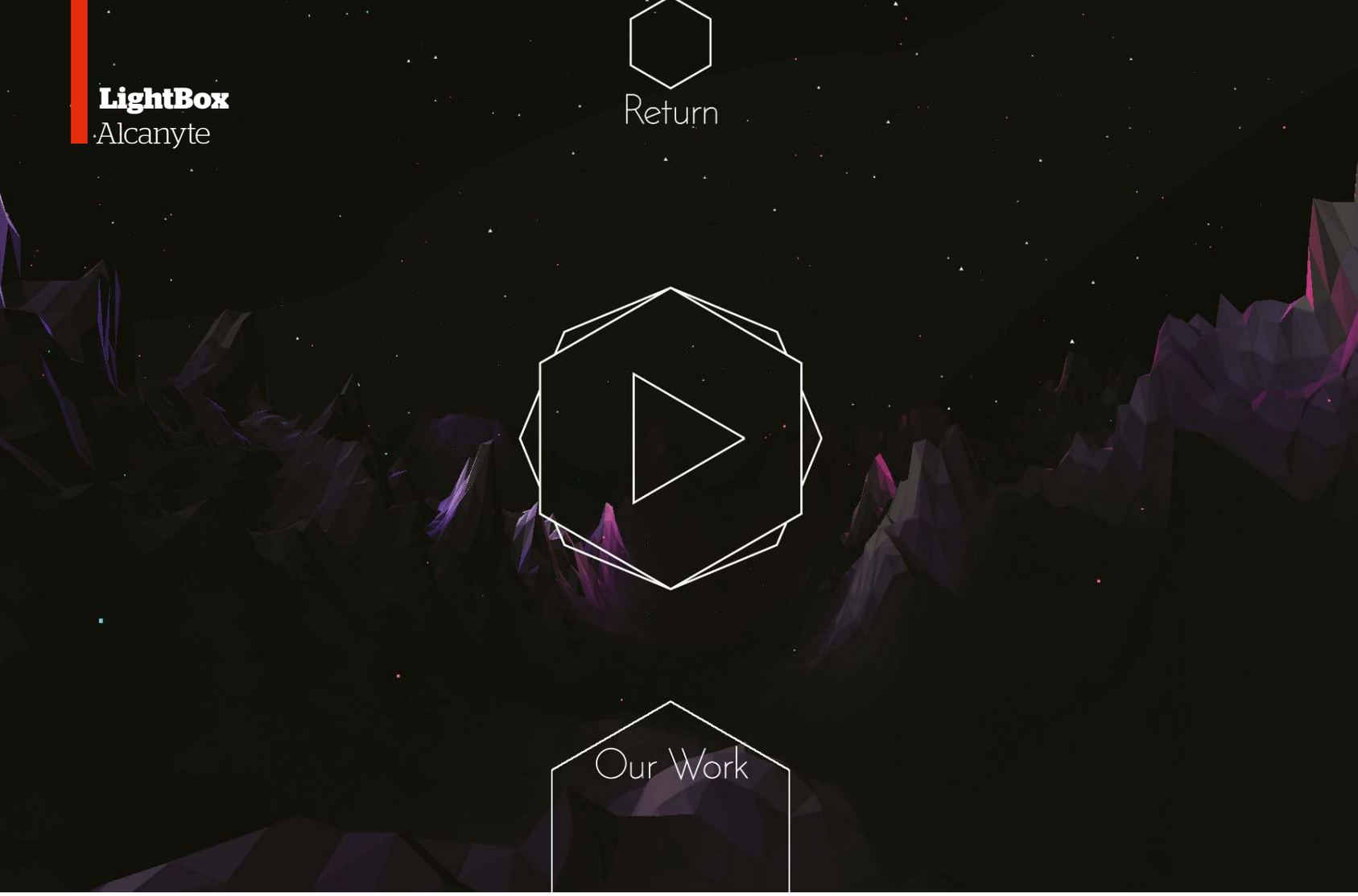
#4668B4

#F472C6

#2B2529

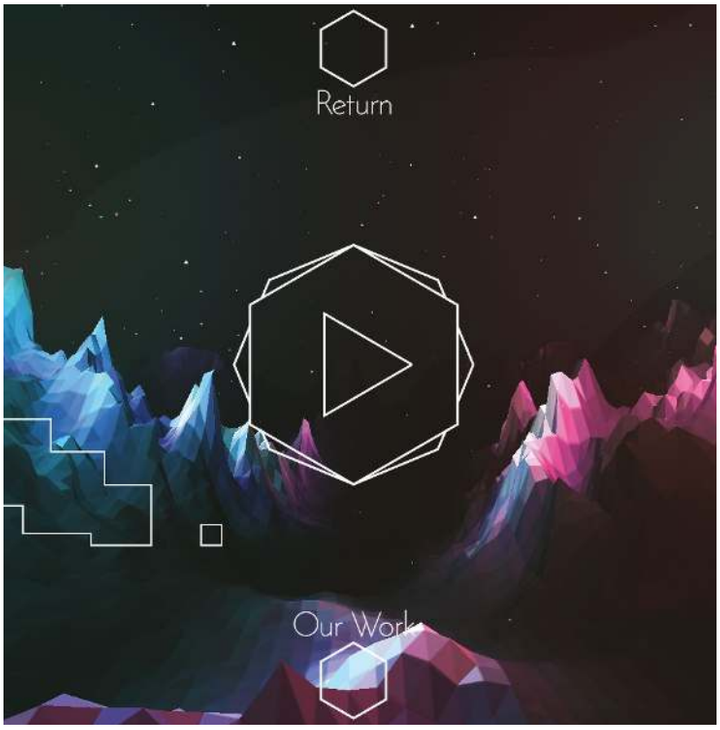
abcABC abcABC

Above
Josefin Sans font by Santiago Orozco for Google Fonts is the only typeface used, in Thin and Light weights



Above
The central line graphic comes to life on hover as a simple animation that encourages the user to play the background video

Above
What's that song? Head to SoundCloud to discover more about the soundtrack



Above
Click, hold and drag to induce a wobble effect to the selected background element

Create an article section play button for navigation

Use CSS to present a navigation link as a play button for linking to a specified content section

1. HTML document template

Start the project by defining the HTML document template. This is made from markup for defining the HTML document container, which is responsible for containing the head and body sections. The head section contains descriptions of the document such as the title, as well as links to external resources such as the CSS file. The page body is where the visible content is placed in step 2.

```
<!DOCTYPE html>
<html>
<head>
<title>Play Button</title>
<link rel="stylesheet" type="text/css"
href="styles.css" />
</head>
<body>
*** STEP 2
</body>
</html>
```

2. Visible content

The visible content is made from two articles – one for storing the play button, and the other for storing the content that the play button links to. The play triangle is made from the HTML symbol code &#9658, which allows for better scalability and faster loading than using an image. The button link references the corresponding article ID 'content'.

```
<article>
<a href="#content" class="big centered
button">Play &#9658;</a>
</article>
<article id="content">
  <h1>Content</h1>
</article>
```

3. CSS initiation

Now that the HTML is complete, a new file called 'styles.css' should be created to store the CSS presentation rules. Start this file with rules for the HTML document and body content containers.

These set the default size, fonts and colours to use to ensure that the content is fully visible without default border spacing.

```
html,body{
  display: block;
  width: 100%;
  height: 100%;
  margin: 0;
  padding: 0;
  font-family: Helvetica, sans-serif;
  background: #000;
  color: #fff;
}
```

4. Article containers

An important part of the presentation is how each content section uses an article to contain each separate part of the content. These articles are set to the full size of the screen using the vw and vh measurements. This is important to make sure that no presentation content is visible when the play button area is being displayed, and vice versa.

```
article{
  display: block;
  width: 100vw;
  height: 100vh;
  overflow: auto;
  transition: top 1s;
}
```

5. Button styles

The button is displayed as a padded block through the use of border and padding styling. A transition rule is also applied so that the hover selector triggers an animated change in colour when the button is hovered by the user, to indicate its interactivity. A display of inline-block is used to allow padding without forcing the element to stretch across the fully available space.

```
.button{
  display: inline-block;
  padding: .5em;
  text-decoration: none;
  border: 1px solid;
  color: #fff;
  transition: background 1s, color 1s;
}
.button:hover{
  background: #fff;
  color: #000;
}
```

6. Sizing and positioning

The button requires its text content to be presented at a bigger size than the main content. This can be guaranteed by using a font-size set using em, hence it will always be three times bigger than the defined content size. A margin is also used to automatically place the button in the horizontal middle of the screen.

```
.big{
  font-size: 3em;
}
.centered{
  margin: 0 auto 0 auto;
}
```

Above
Recreate cursor actions with fingers on mobile

Avoiding annoyances

As well as directing users to a content presentation, this is an alternative to the increasingly common yet annoying auto play of video and presentations.

MORE SMART SERIES

seedsandsupply.com/mss

Designer: MEFILAS Inc

Development technologies HTML, CSS, JavaScript, jQuery

"The background text keeps the user engaged while waiting for the main product image to slide in from the top of the page"

#BEA47D

#FFFFFF

#000000

#E0D4C2

abcABC 0123456789

Above
The animated background text is powered by images. The only font used throughout the page is Oswald by Vernon Adams

VOL.4

vol4.co

Designer: VOL.4 (in-house)

Development technologies HTML5, CSS3, JavaScript, Cloudflare

“This minimalist calling card for LA-based agency VOL.4 uses colour flipping and floating text animations to notable effect”

ncy based los

in

california.

#E73D1A

#FFFFFF

#111111

#2341BE

abcABC 123456

Above
The only font used on the modicum of text found throughout the site is Akzidenz-Grotesk BQ Medium Extended, by H. Berthold

Create an animated colour palette selector

Use JavaScript and CSS to change your webpage colour palette in response to user interactions

1. HTML document template

The first step is to initiate the HTML structure of the webpage. A head section is used to contain document description information and links to resources such as the CSS stylesheet. The body section will contain the visible content. The HTML content will be placed inside the body section in step 2. Take note how the body container also has a 'data-palette' attribute describing the default palette to use.

```
<!DOCTYPE html>
<html>
<head>
<title>Palette Change</title>
<link rel="stylesheet" type="text/css"
href="styles.css" />
<script type="text/javascript" src="code.
js"></script>
</head>
<body data-palette="1">
*** STEP 2
</body>
</html>
```

2. Control and content

The webpage consists of a drop-down list of available colour palette options, along with a visible content section. The drop-down list will be used as a control to trigger JavaScript functionality that will in turn trigger changes via the CSS style rules. This approach is highly useful for maintainability – hence keeping functionality to JavaScript and visual presentation within CSS.

```
<select name="palette">
  <option value="1">Palette 1</option>
  <option value="2">Palette 2</option>
  <option value="3">Palette 3</option>
</select>
<article>
  <h1>In a galaxy far away...</h1>
</article>
```

3. CSS initiate

With the page HTML now complete, the next step is to create a new file called 'styles.css'. This file is initiated with a set of rules to set the HTML document and body container to cover the full screen without border spacing. Default font and colour settings are also set in this stage.

```
html,body{
  display: block;
  width: 100%;
  height: 100%;
  margin: 0;
  padding: 0;
  font-family: Helvetica, sans-serif;
  background: #fff;
  color: #000;
}
```

4. Palette initiation

The changing colour feature is made possible through the 'data-palette' attribute applied to the document body in step 1. This attribute can be applied to any element to control a specific area of the page. The default palette settings make sure that a transition of one second is applied and the background and colour changes so that they appear animated.

```
[data-palette]{
  transition: background 1s, color 1s;
}
```

5. Palette definitions

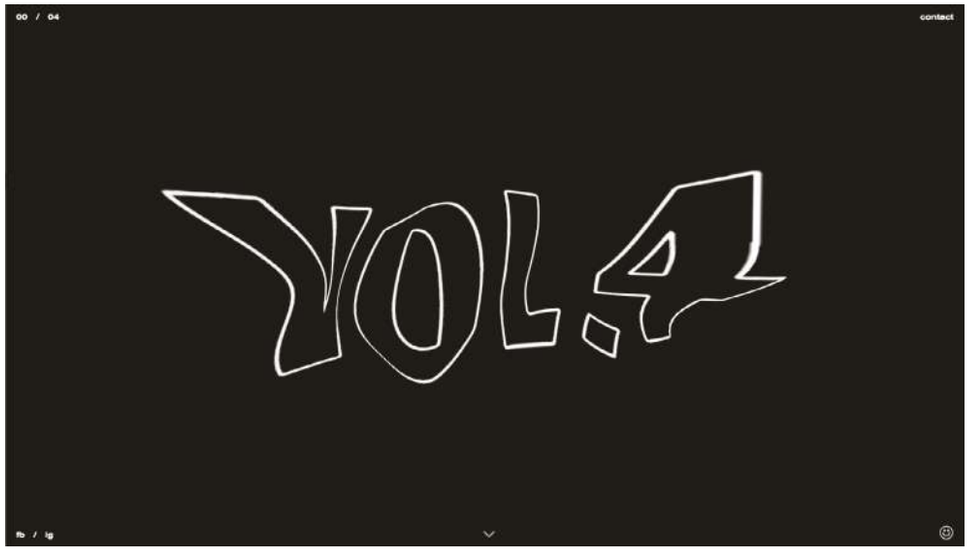
The final part of the CSS is to define the colours for each available palette option. The different palettes are identified by the 'data-attribute' value – options for 1, 2 and 3 are defined in this example, but you can use text names and other numbers too. Take note how these data values correspond to the options in the drop-down menu defined in step 2.

```
[data-palette="1"]{
  background: #fff;
  color: #000;
}
[data-palette="2"]{
  background: #9C27B0;
  color: #FFEB3B;
}
[data-palette="3"]{
  background: #9E9E9E;
  color: #E91E63 }
```

6. JavaScript event listener

Finally, create a new file called 'code.js' to store the JavaScript interaction code. First, the JavaScript waits for the page to load before applying an event listener to the drop-down list defined in step 2. This allows for the value of the drop-down menu to be applied to the 'data-palette' attribute applied to the body – hence triggering the CSS presentation when it is changed.

```
window.addEventListener("load", function()
{
  document.querySelector('[name="palet
te"]').addEventListener("change",
function(){
  document.body.setAttribute("data-
palette", this.value)
  });
});
```



UX and visuals

This type of feature can be used as both a visual effect and also to help visually impaired people – such as for colour blindness.

14 speakers, two tracks, one amazing day

TICKETS ON SALE NOW

www.generateconf.com

PROJECT

THE HIDDEN DANGERS PROJECT

WEB

HIDDENDANGERSPROJECT.COM

AGENCY
DEUTSCH

WEB
DEUTSCH.COM

PROJECT DURATION
7 MONTHS

PEOPLE INVOLVED
28

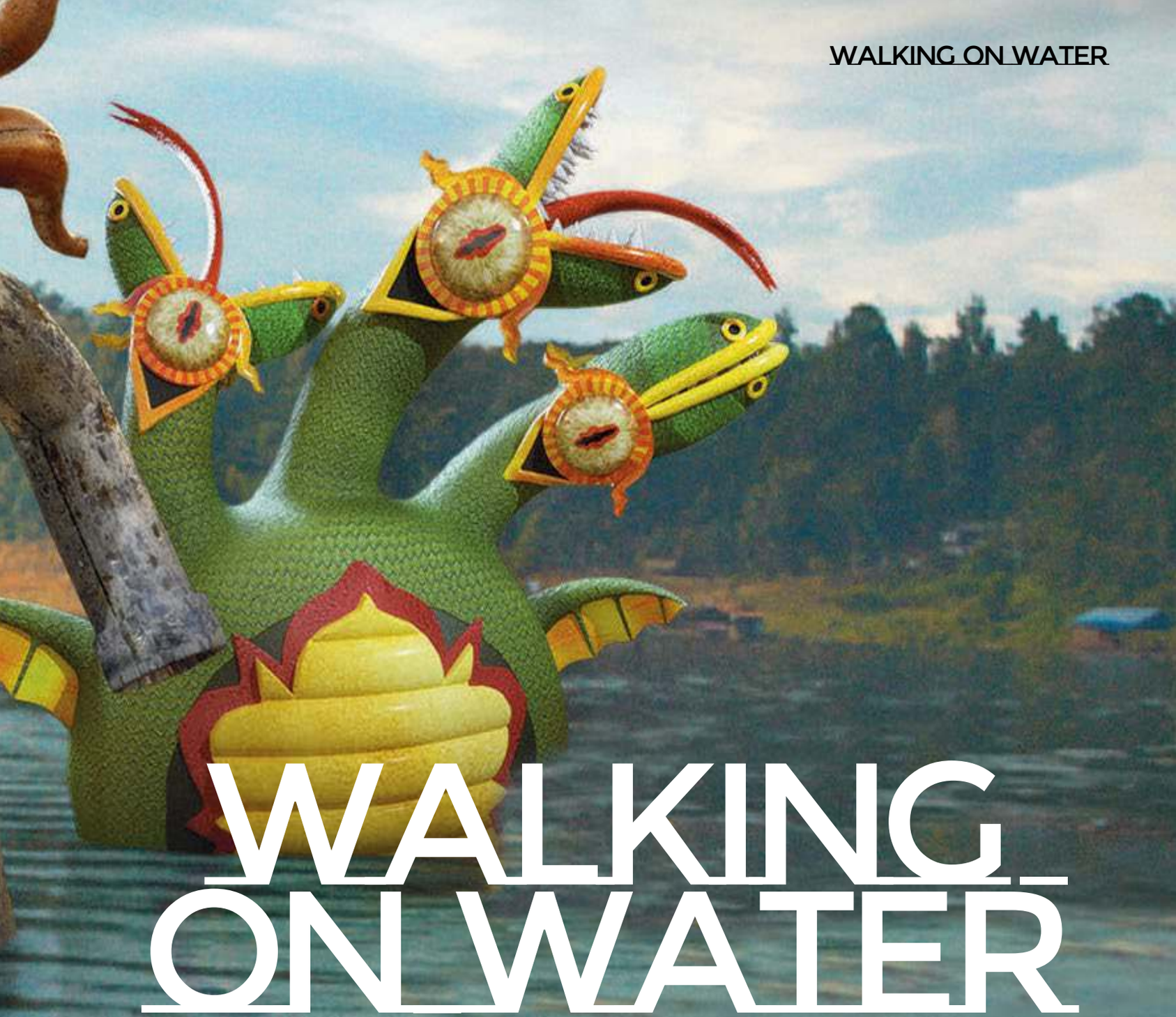
PERSONNEL

MENNO KLUIN
EXECUTIVE CREATIVE DIRECTOR

ALIZA ADAM
VP, INTERACTIVE DESIGN DIRECTOR

ALEX MILLER
SENIOR INTERACTIVE DESIGNER

LAUREN BROOKS
UX DESIGNER



WALKING ON WATER

Creative agency Deutsch describes how its talented team negotiated any hidden creative dangers when it came to delivering an engaging digital experience for WATERisLIFE

In this increasingly cynical world of fake news and inwardly looking social media, some might be forgiven for questioning certain causes. It seems that everywhere we look these days, the zeitgeist is one of gloom and doom, with a prevalence for dramatic, attention-grabbing headlines. So against that kind of sensationalist backdrop, how might charitable and humanitarian issues find a space to raise awareness and maybe, just maybe, write good news stories in the process? Well, thankfully, cutting-edge web and digital technologies can also be utilised to do just that, delivering engaging ways to educate on a global scale. Charities are developing innovative methods to not merely appeal to our wallets for handouts, but spread the word on quite complex issues and make us pause long enough to think. Of course, the most effective campaigns speak to the next generations, and often kids who are excited by advancements in stuff like virtual reality. That's really the thinking behind the Hidden Dangers Project and the terrific work that the

TRANSLATING A STORY

Hidden Dangers was, and remains, a remarkable campaign for attempting to communicate a serious environmental issue with real ingenuity. The idea of spanning multiple digital channels including the exciting advancements in virtual reality was ambitious and would demand a cohesive concept that the website would need to reinforce. So where did the central idea come from, what kind of considerations were made in the initial phases and how did they inform the subsequent design? "The initial idea for the Hidden Dangers Project – the creation of an educational VR game to make invisible monsters, the pollutants in the water, visible to children – was set so we focused a lot on how to translate that story and expand on it," explains senior interactive director Alex Miller. "The site needed to be easy to navigate and simple, but we also wanted to encourage users to follow the narrative, so we designed it in a way that would make it easy to just go with the flow of the story but skip around if necessary. We also made sure that the donate button was in focus wherever possible and even set the donation section up as an informational device to let users know exactly what their donations would be used for. We bounced around some rough wires that became a lo-fi prototype, and concurrent to that, we were developing the visual and UI elements."

Users are required to take on water-dwelling monsters, symbolising a variety of bacteria

organisation WATERisLIFE is doing to communicate the importance of clean water. Placing the lives of children in Southeast Asia at its heart, the campaign illustrates how polluted water in the rivers, so essential to local people in such communities, puts young lives at risk. The beautiful official site at **hiddendangersproject.com** highlights the unseen risks or 'monsters' lurking below the surface with a Pokémon-style narrative. Supplemented by a downloadable VR game for Oculus Rift, the experience can be deployed in classrooms everywhere to entertain and teach. To deliver such an ambitious vision, WATERisLIFE enlisted the services of Deutsch, a creative agency with offices in LA and New York. No strangers to devising compelling campaigns for high-profile commercial brands, the team would embark on a seven-month project with a weighty conscience to consider. We discover how Deutsch negotiated any 'hidden dangers' of its own to wow browsers and headsets alike.

CROSSING THE STREAMS

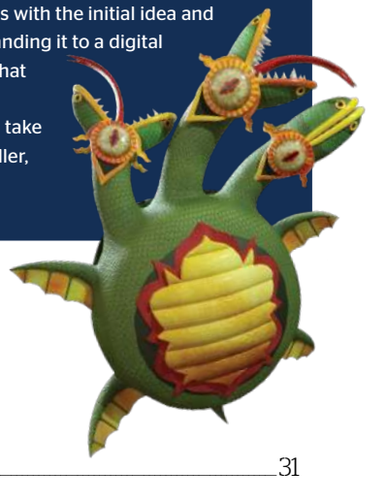
In terms of projects, from the outset there was nothing 'typical' about Hidden Dangers for client WATERisLIFE, or WiL for short. Previously the charity has employed traditional storytelling techniques such as the film *4 Year Old's Bucket List* to bring poignant stories to the fore. They 'hijacked' social memes on Twitter with *Hashtag Killer* and even constructed a book from water filter paper with *The Drinkable Book*. Point being, every project and campaign has never shied away from innovation and fresh thinking when spreading the word on the world's water crisis. But as Deutsch was aware of almost immediately, nothing stands still when it comes to effective communications and the challenge of capturing busy modern minds.

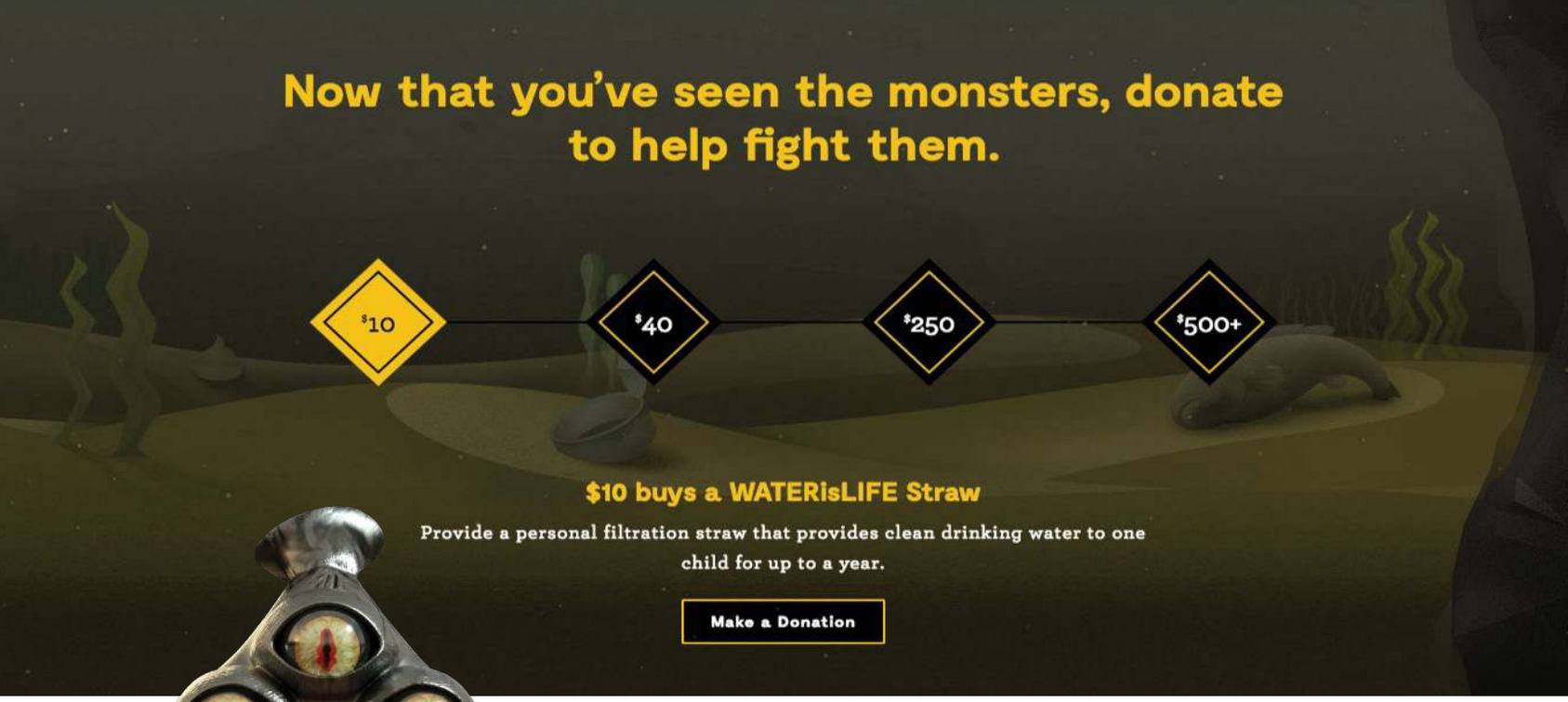
"One of our biggest hurdles in teaching future generations about proper water sanitation and hygiene is no longer just a language or awareness barrier," the team begins. "To truly engage with children on a personal level and capture their imaginations, we knew we had to create an educational experience that would mimic their surrounding environments and accurately teach

them about the dangers of the water they use daily; all while keeping them entertained." So how is such a thing achieved in 2017? Well, you might think the concept of a pioneering educational VR game set in the remote jungles of Thailand to be a tad ambitious. You'd be right, and Deutsch knew that to realise such a thing would require some backup. Production partners would be brought on board in the shape of studios NTROPIC (**ntropic.com**) and m ss ng p eces (**mssngpeces.com**), adding the invaluable VR directorial expertise of Tucker Walsh and Ray Tintori. "The Hidden Dangers campaign was always going to be a multifaceted effort, supporting the fundraising efforts of WiL that drive clean water initiatives and the distribution of clean water filtration straws around the world. In addition to the educational VR game, which continues to travel around schools in America and abroad, the campaign consists of a three-minute docu-short illustrating the charity's impact in a remote Thai village as well as the website itself."

FLUID FOUNDATIONS

As with many agency-client arrangements, this particular project had its roots in an already established working relationship. However, although we often hear that such closeness implies tight collaboration with constant consultation, Deutsch enjoyed more freedom to get the job done. "We've had a working relationship with WATERisLIFE for about eight years now and there is a lot of shared trust," admits executive creative director, Menno Kluin. "So only at key moments would we share updates with the client. The conversation with the client is mostly around if the messaging is correct, but other than that they place trust in us to do what is right by them and the project." Such a position makes sense when dipping into new technological waters of course, but it did offer useful creative fluidity when interpreting the earliest brief. "When the creative team approached us with the initial idea and tasked us with expanding it to a digital space, it was clear that there were a lot of directions we could take it," explains Alex Miller,





SITE HIGHLIGHT

The underwater scene is definitely a highlight. When we're in the Trash Monster world, there are cans floating around, and when we're in the Bacteria Monster world there is toilet paper, and so on and so forth. The CSS floating particles, the fish that swim in and out of frame... the whole thing is a design, illustration and coding achievement!

Aliza Adam - VP, Interactive Design Director

> **"We added CSS details that would bring our underwater environment to life"**

senior interactive designer on the project. "We started brainstorming ideas with the focus on building a relationship between the users and the villagers, to demonstrate just how much clean water can make an impact. We knew almost immediately that we wanted an underwater aspect to the user journey, and with so much of the story being about what's in the water, it only made sense that we explored that territory further."

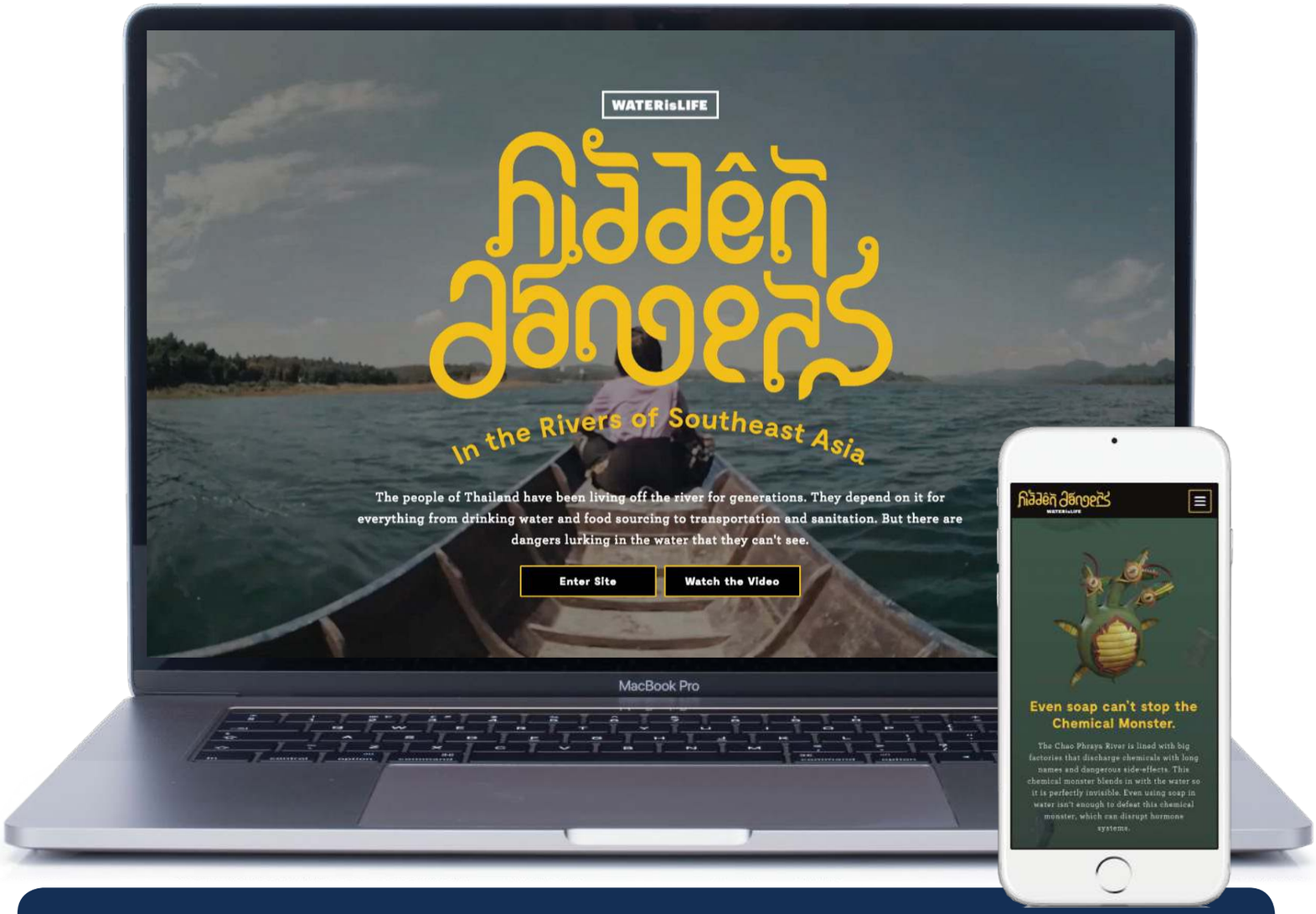
DIVING INTO THE DESIGN

Hardly surprising then that a website for a WATERisLIFE campaign would want to put water firmly up front. With a message so intertwined with the real-world environment brought to life in film and VR, the site would need to transport visitors with an inventive use of front-end standards. "We started to look at ways we could execute that idea and felt that the best way to create a rich experience for users would be to illustrate our own underwater environment and animate it using subtle CSS animation," Miller continues. "This would give users more context and visual reinforcement for the information we were presenting to them, in a more user-friendly way."

Visual reinforcements would be particularly crucial, not least in quickly capturing eyes and imaginations but also ensuring consistency of experience. Here the team could count on a cross-pollination of assets from the VR, print and film production happening alongside. "Our creative team had started working with Juan Carlos Paz, an illustrator who was helping create the monsters for the VR game and some print pieces. So we already had a great starting point for the site. We knew a dark interface with bright UI would give us the right mood to match the documentary video and convey that this idea revolved around a polluted river. We then designed and illustrated additional assets to bring the underwater world to life, simultaneously working with our developer to really bring out the details with motion. We added CSS details like floating particles, swimming fish and floating trash that would bring our underwater environment to life."

GO WITH THE FLOW

Such a rich visual journey, especially one involving so much water, would still need to be mindful of flow. Telling a meaningful story with these assets had to revolve around certain narrative rules to hang together, as UX designer Lauren Brooks describes. "We did a lot of sketching and user flows to understand a series of questions: How can we best guide the user through the experience of life on the river? What's the best way to introduce the Hidden Dangers monsters, as well as the VR experience?



There was a lot to say, but we wanted to keep it simple too, so there was quite a bit of back and forth before we settled on the high-level flow."

Those cognitive challenges and issues of usability, general navigation and so on would of course be joined by the practicalities of making it all just work. In terms of back-end development slog, the team admits that significant effort was minimal thanks to the emphasis on serving the front-end. Here it was much more a case of ensuring that the various visual effects, animations and transitions were synchronised for maximum impact. "The back-end architecture is composed of a Node.js web server with React's JavaScript framework built on top," starts Aliza Adam, VP and interactive design director. "The site is primarily front-end though, and we put a lot of focus into the finesse of the smallest details. Making transitions smooth between looped video to underwater worlds, creating motion on hovers, lighting effects when the user was at the top of the river, plunging the user into the next story when they were at the bottom of the river, the works... so that the site felt very integrated and cohesive."

KEEPING IT RUNNING

Anyone looking at and interacting with **hidden dangersproject.com** will admire the sense of cohesion achieved in just 31 weeks. The launched website does an enviable job of conveying the emotive message that purveys each channel of the campaign, particularly the short film. At the same time however, it manages to seamlessly handle such a weighty cause with the playfulness that children everywhere can identify with. Around launch time, it was therefore important to get the timing and marketing just right to capitalise on related initiatives and leave a lasting impression. "We mostly rely on PR and donated media to get the project out there," Menno Kluin explains. "We launched the Hidden Dangers Project prior to World Water Day and Discovery and *Time for Kids* have been very helpful in spreading the project. Also, combining resources with partners like m ss ng p eces, who directed the docu-short, and NTROPIC, who developed the VR game, is extremely important to ensure we can execute on the overall creative vision and deliver an experience people will never forget."

Certainly if the online industry reception is anything to go by, forgetting **hiddendangersproject.com** won't be happening anytime soon. The Deutsch team are understandably very proud of the recognition the work continues to garner since going live. More crucially, the URL's success is perhaps measured best as a fundraising tool for the client and one that must be kept afloat 24/7. "We're currently hosting **HiddenDangersProject.com** since we built it in-house, so there's no hand-off on this one. We've been really lucky in winning The FWA Of The Day, a CSS Design Awards Site Of The Day, and an Awwwards Site Of The Day, so our most recent upkeep is adding the win banners to our site. The site is also a portal for donations at any time for WATERisLIFE, so ensuring this component is constantly up and running is really important for us."

TRY 5 ISSUES FOR £5*

SAVE UP TO 83%

- ✓ **Never miss an issue**
  13 issues a year and you'll be sure to get every one
- ✓ **Delivered to your home**
  Free delivery of every issue, direct to your doorstep
- ✓ **Get the biggest savings**
  Get your favourite magazine for less by ordering direct

Simply visit **www.imaginesubs.co.uk/SPRING171**

Hotline 0844 856 0644+

Choose from our best-selling magazines

SAVE 82%

Games TM

SAVE 76%

How It Works

SAVE 80%

All About History

SAVE 80%

Digital Photographer

SAVE 80%

Retro Gamer

SAVE 80%

World of Animals

See our entire range online

www.imaginesubs.co.uk/SPRING171

OFFER ENDS **31 JULY 2017**

*Terms and conditions: This offer entitles new UK Direct Debit subscribers to receive their first 5 issues for £5. After these issues standard pricing will apply. New subscriptions will start with the next available issue. Offer code SPRING171 must be quoted to receive this special subscription price. Details of the Direct Debit guarantee available upon request. This offer expires 31th July 2017. + Calls will cost 7p per minute plus your telephone company's access charge

wild design

As a small group of designers, developers, producers and explorers, wild has become a preeminent design agency. Their insatiable appetite to discover the frontier of the digital landscape makes them pioneers that others follow

who wild

what Websites, digital content, virtual reality, e-Commerce, mobile, social media and design

where Zeitgasse 12/2/9, 1080, Vienna, Austria

web wild.as

key clients

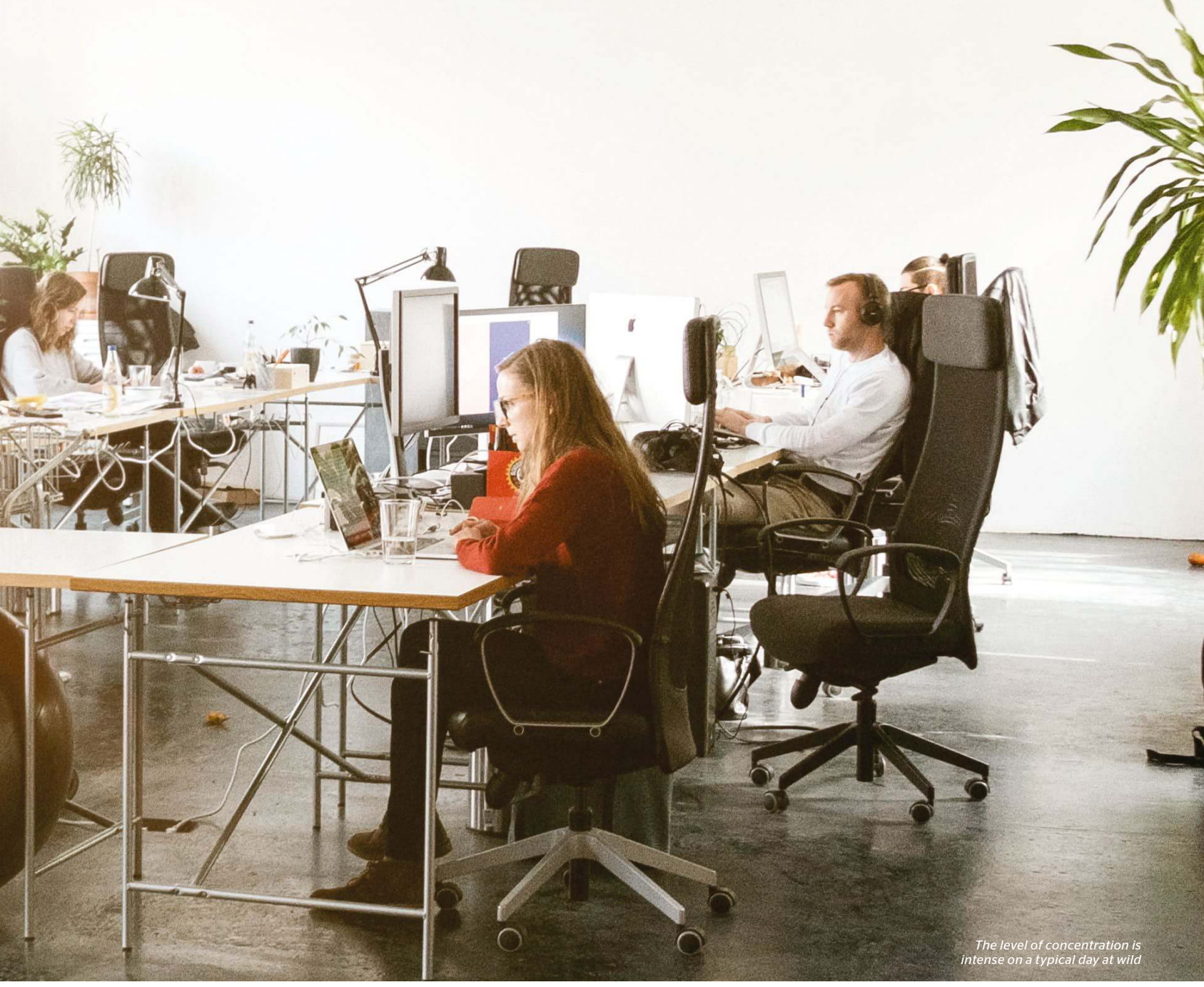
Google Creative Lab

Red Bull

Sid Lee

Wieden + Kennedy

Sherpa



The level of concentration is intense on a typical day at wild

Wild is Matthias Mentasti (creative director), Thomas Lichtblau (creative director), Thomas Strobl (tech director) and Thomas Ragger (tech director). Starting just two and a half years ago, the founders began their careers as freelancers. It quickly became clear that with the knowledge and skills that each founder possessed, it made sense that they should create their own design agency.

"We are all from a freelance background and decided that it would be a good idea to found a company together," explained Thomas Ragger.

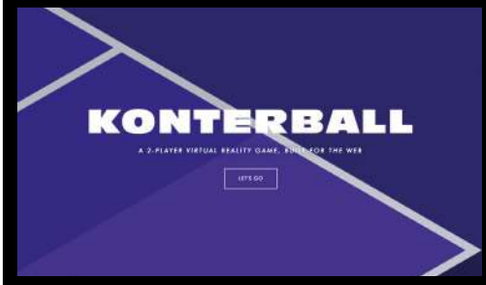
"We started with a lot of international clients such as Adidas, Intel, Acura or Al Gore. We were able to bring those projects into the company mainly through contacts we had gained while freelancing. After a while we were able to build a reputation inside Austria, where we continue to work with a growing roster of local clients." Thomas continued: "Our art directors both went to design school. Thomas Strobl is completely self-taught and I went to design school only to teach myself coding over the course of a lot of insightful internships and jobs at Firstborn NY, Hellohikimori Paris, and Stinkdigital (now Stinkstudios) in London and New York." The current iteration of wild's website is a minimal affair shunning the flashing graphics that can often be seen on other agency sites. Matthias outlined wild's approach to naming their company: "When we were still individual freelancers, we rented a fairly gnarly office space together which had a giant black board to draw onto. One day Matthias drew a big bear with the typo "where the wild things are" and we kept it for quite a while since it was the first thing you saw when you entered the space and it felt kinda right."

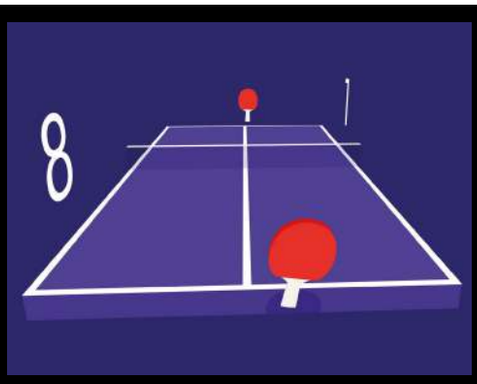
> "We immediately thought it captures our spirit and our way of tackling things"

Thomas Ragger also said: "When looking for a name we lunged for something short and concise while still not being used by a big company in our industry. After some failed attempts of settling on a name, one of us looked at the board and saw the word 'wild'. We immediately thought it captures our spirit and our way of tackling things while still being short and rather unused in the industry. Afterwards we tried to look for domains and obviously all major TLDs were taken, but then one of us came up with the clue of using the .as extension and playing with the first URL parameter afterwards. Thus, wild.as/hell was born." With Thomas Strobl also commenting: "I think our agency site is the main point of contact for anyone

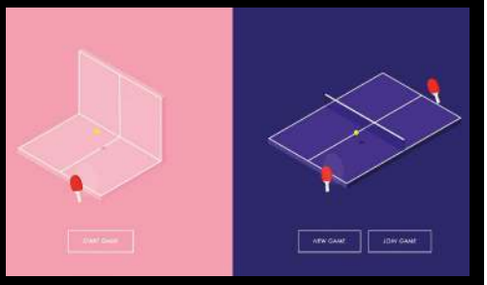
wild

KONTERBALL
konterball.com





Konterball allowed wild to flex their WebGL muscles with addictive effect



Konterball is a ping-pong game we developed in WebGL together with Google Creative Lab for the launch of the latest Google Chrome. Chrome is one of the first browsers to natively support WebVR, which massively lowers the barrier of entry for VR experiences.

Now you don't have to download a clunky app, as VR can be experienced directly inside of the browser just like the regular web. The game can be played with Google Cardboard, HTC Vive, Occulus Rift or just using your phone/desktop. We started developing simple prototypes in WebGL to find the best game mechanic.

At the same time, we explored various visual styles. After the game mechanic was working well, we went on to develop various game modes and integrate animations and transitions. After an extensive testing and fine-tuning phase, we launched the project. Konterball was well received and the new Google WebVR Experiments Platform was picked up by the likes of Techcrunch and Product Hunt.

We also open sourced all the code on GitHub for people to be able to browse through and learn from it. You can play the game by yourself or with friends in VR space in real-time by using a websocket connection, which is a lot of fun.

who has heard about us or has seen our work so it's pretty important. For us it's more important to keep it up-to-date to showcase the range of our projects rather than making it extremely fancy. We prefer a clean minimalistic agency site and would like for the projects to speak for themselves."

As a relatively new agency wild has continued to use an array of techniques to gain new clients. "We did a few pitches recently where we were invited to participate," said Matthias. "We want to make sure to only put time and effort into a pitch if we're convinced that it can be an amazing project. And if we do get started we want to deliver the best possible output. It often starts with just a random idea and develops into more than what the client asked for in a positive way. Another way to get work which can be a lot of fun is through collaborating with other people. We are not afraid to get somebody on a project who can do specific things better than ourselves in order to push the project to the next level."

> "We are not afraid to get somebody on a project who can do specific things better than ourselves in order to push the project to the next level"

Matthias continued: "We've had WebGL shader developers, amazing copywriters, illustrators, film makers or composers come in before and it was always an amazing experience. Not only have we that way created beautiful work together, we've also made new friends. It has happened more than once that one of these people would later reach out to us and in turn, we would work on a

INDUSTRY INSIGHT

Thomas Lichtblau
Creative Director and Co-founder

"I think JavaScript has developed far enough that mostly jQuery isn't really needed anymore. As for the development of the web in general, we have seen great progress over the last few years and really anything is possible on the web now – especially looking at WebGL and WebVR"

AGENCY BREAKDOWN

2x Tech Director

1x Creative Director

1x Art Director

1x Designer

2x Senior Developers

1x Junior Developer

1x Project Manager

1x Social Media Manager

project with them. We think collaboration is a key factor to creating ground-breaking projects, which often later result in new work. Additionally, we try to maintain a good relationship with agencies as well as direct clients and have ongoing conversations with them about potential projects."

wild, like all creative services, needs to balance the needs of their business with the desire and drive to create ground-breaking pieces of work for their clients. Thomas Ragger outlined their approach to the 'business' of being a growing design agency: "We try to only take on work where we have the freedom to participate starting from the initial concept phase, over design and through to development, sometimes also helping with the art direction of content production.

"For us this has proven to be the only way to keep a consistently high level of quality for all of our projects. Small projects can be fun as well and sometimes they offer more freedom and we're able to achieve a better outcome through that. We have already worked with quite a few start-ups where people came to us with just an idea and a small budget. We got hooked and wanted to help bring the idea to life. One of these projects recently was named a Webby Honoree even though it's still only in public beta (**trysherpa.com**). All our projects speak to the ethos of our agency, while still being very different at the same time. We always try to develop a distinctive style for each project that fits it best. But all our projects will include a high level of detail when it comes to design, animations and execution."

Thomas Ragger also commented: "I think Falter shows this quite well when you start playing around and notice more and more details over time such

The offices of wild have a distinctive minimalist approach to their design

The people that are wild (L-R): Thomas Ragger *(Partner/Technical Director),* Melissa Graf *(Producer),* Max Hermetter *(Senior Developer),* Thomas Lichtblau *(Partner/Creative Director),* Moriz Büsing *(Senior Developer),* Thomas Strobl *(Partner/Technical Director),* Manuel Haring *(Art Director) and* Matthias Mentasti *(Creative Director/Partner)*

as hover states in the navigation or particle reacting to your mouse. Send A Message (**sendamessage.to**) as an internal project, brings across our attitude and weirdness very well which often results in making something that is a lot of fun, a bit random but looks beautiful.

"People seem to like it, they sent out 300K messages already – not a bad result for a one-week project. For Konterball (**konterball.com**) we could have just made a fun, nice looking minimal ping-pong game – but then we added a rainbow mode (press R) and had a composer create a custom soundtrack which is now available on Spotify. I guess we do go overboard sometime. But that's the point; we want to have fun creatively and release something that's flawlessly programmed so as a user you can just lean back, enjoy the experience with all its little details." As a relatively compact team, developing efficient working processes has been vital to ensure deadlines are met. "Since we're still quite small, usually one or two of our founders (one tech, one art director) will take the lead on the project," said Thomas Strobl.

"That means doing the initial client communication, working out scope, budget and timeline. Once the project goes into production, one partner will always stay on as the main point of contact for the project. We will hand off day-to-day tasks to our project manager, do tech direction and oversee development and the same goes for art direction and design.

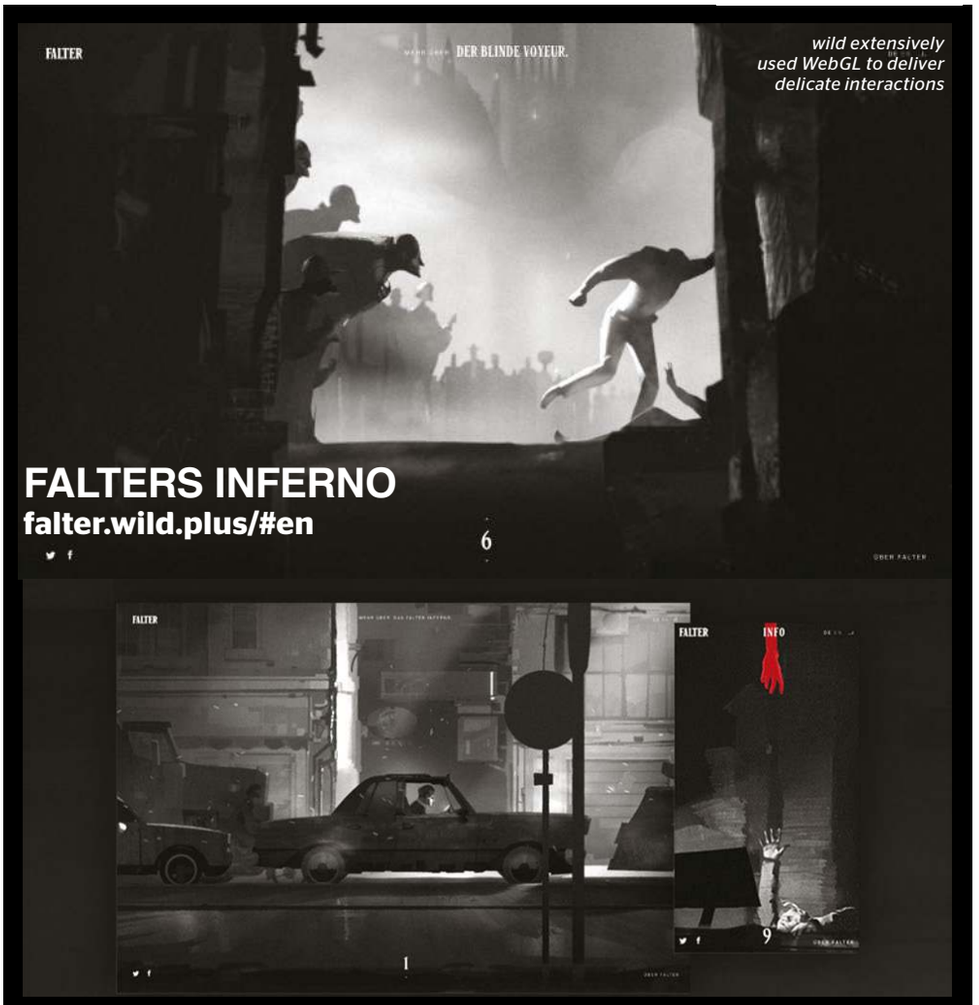
"A project team usually consists of one dev, one designer and a project manager plus additional specialists for things like WebGL shader development if needed. We have two types of projects: regular websites and microsites, which go anywhere from 6-12 weeks and longer ongoing projects like bigger e-commerce and platform solutions and collaborations with start-ups, as well as ongoing social media management.

"For shorter timelines, we usually have UX, design and development (alpha, beta) phases. Most of the time always goes into fine tuning all the details, animations, hovers and transitions in the end as well as testing on all platforms to make sure the quality of the site is up to our standards. For longer timelines, we work in continuous sprints between UX, design, development but also here fine-tuning should never be left out."

> "I guess we do go overboard sometime. But that's the point; we want to have fun creatively"

Each agency will have their own preferred toolset

wild



wild extensively used WebGL to deliver delicate interactions

FALTERS INFERNO
falter.wild.plus/#en

that enables them to achieve the vision they have for their clients. Thomas Lichtblau outlined what's in the wild toolset: "We mainly use Dropbox for file-sharing rather than bespoke server solutions. Adobe for anything design and animation related – although we are switching to Sketch now for app design in specific – and Webstorm/Sublime for development. To be honest we don't really focus on the tools so much since often they are mainly based on personal preference.

"I think JavaScript has developed far enough that mostly jQuery isn't really needed anymore. As for the development of the web in general, we have seen great progress over the last few years and really anything is possible on the web now – especially looking at WebGL and WebVR.

"We recently worked on a really interesting project using WebVR which the new version of Chrome supports natively. WebVR makes VR accessible to anyone with a browser and it allows your friends to

For Falters Inferno we worked with Salon Alpin – an Austrian animation studio – who had already developed an amazing short film for Austrian newspaper Falter. We wanted to bring their animation alive digitally and somehow transform the linear narrative into a non-linear interactive story.

To do that, we brought all different animation layers into WebGL and added custom WebGL Shaders and subtle interactions which you can trigger by moving your mouse or interacting with touch. We optimised the experience to perform especially well on mobile and focused on making it as seamless as possible. For this project, we spent about two full weeks just on fine-tuning but I think the effort has paid off. People really liked it and even though Falter is only available in some parts of Austria, over 200k people from around the web visited the experience which by the way wasn't advertised anywhere at all.

We also won three Gold Lions at Cannes and site of the year on Awwwards. Our favourite prize was the Webby. We went to the award ceremony in NYC and were amazed to meet Action Bronson and Jessica Alba in person. Taking a selfie with Action is my favourite memory from that trip.

TIMELINE

Year 2014	Year 2015	Year 2016	Year 2017
No. of employees 4	**No. of employees** 5	**No. of employees** 8	**No. of employees** 10
wild works on Al Gore's climate change site. He broadcasts live to the web for 24 hours straight.	We hire our first employee, talented coder and good friend Max.	Falter wins a Webby and SOTY at Awwwards. We build the Red Bull Mindgamers platform.	We launch our first WebVR game Konterball with Google Creative Lab.

join into the same virtual space and interact with you. We think that this technology can become really interesting."

"Also, we love working with React and React native. It's one framework we can use throughout mobile, web and desktop development, that all our developers know and use. It allows us to work more efficiently and focus on all the design details and refinements that are important to us. In general, JavaScript has gotten powerful recently. You can use it to do anything from frontend, backend, mobile and desktop development. It's awesome."

Mobile and social media have become the twin development hubs that all brands want to exploit. Does wild pay close attention to these channels when developing for their clients? Thomas Ragger explained: "Social media is a really important topic for us. We recently started developing bigger social media campaigns as an effort to communicate with the audience on a broader level. It must be treated as an equal digital touchpoint of a brand. We don't see it only as an opportunity to drive traffic to the client's site, but as an effort to increase visibility of the brand and interact with customers. Thomas continued: "Mobile has a really high priority for us and we usually work with a module based system where the modules we use in our site such as images, videos and custom animations will be able to adapt to any environment in their design and capabilities, whether it's tablet, mobile and desktop. In general, with mobile phones and their OS's continually getting faster and more performant desktop and mobile are growing closer together in a sense."

> "It's a really interesting time and we're learning something new each day"

Working for an agency like wild isn't all about technical skill. What is vital is the design sensibilities you bring to the team. Thomas Ragger explained: "The most important thing for us is personality – for people to fit in with wild. We don't mind if you're not the best coder or designer in the world yet. Most of these things can be taught or learned. But what we do ask from everyone is to not be afraid to ask questions and to be interested in learning new things all the time."

With the present being forward thinking, what does the future hold for wild? Thomas Strobl concluded: "Our definition of wild is actually constantly changing," he said. "We come from a background of microsites and now have developed to cover a much broader range of digital. From social media, content production and product websites to big e-commerce platforms. It's a really interesting time and we're learning something new each day which is awesome!"

ESKAPISMUS
bergluft.hervis.at/chapter/1

110% is an Austrian monthly magazine. They wanted to bring their print edition into digital. We developed a custom module system so they're able to show integrated video content, audio quotes, and 360-degree and panoramic images. The whole website was developed as a reusable framework which allows future editions to be released in a much shorter time frame. But what got us really excited about the project were the content shoots.

For the first edition, we had to hike to the top of the mountain together with a small film crew and a photographer which meant having to get up really early just to get that perfect morning sunrise. After a few hours of walking through the snow we had finally reached the peak and had an amazing ski run downhill. To finish off, we received an amazing meal at a local restaurant at the bottom of the mountain.

Experiencing this through the web is not quite the same, but it still gets close to getting across the vibe of this amazing tour. The next edition will be about running so be sure to look out for it. Funnily enough, our whole company is participating at the Vienna marathon and is equally afraid of it – so we think this topic might be one that you might prefer experiencing from the comfort of your living room couch!

INDUSTRY INSIGHT

Thomas Ragger
Tech Director and Co-founder

"The most important thing for us is personality – for people to fit in with wild. We don't mind if you're not the best coder or designer in the world yet. Most of these things can be taught or learned. But what we do ask from everyone is to not be afraid to ask questions and to be interested in learning new things all the time"

One of the many 'meetings' that wild hold to really get inside their next project

wild

wild.as

Founders
Matthias Mentasti, Thomas Lichtblau, Thomas Strobl, Thomas Ragger

Year founded
2014

Current employees
10

Locations
Vienna

Services
Strategy

Creative Art Direction

App, Web and VR Development

Photo and Video Content Production

E-commerce

NEW GENERATION CSS

WEAVE THE LATEST MUST-KNOW TOOLS & RULES INTO YOUR PROJECTS

CSS, just like every programming language, is always evolving as the web adapts and changes. CSS is maturing as it comes out of its awkward teenage years and into its twenties, with full spec implementation blunders out of the way and a bunch of exciting new features on the horizon.

CSS Grid has already started to gain traction. It has been the centre of attention lately, so some of the other new features have been left feeling a little neglected. One of these is CSS Variables or Custom Properties. Many of us have been using variables in CSS for a few years through the use of preprocessors like Sass and LESS, but now we can use them in native CSS, allowing these variables to be updated on the fly with JavaScript or inside media queries, aiding in responsive design and the implementation of custom user settings.

Speaking of media queries, we'll also be looking at the current state of Element or Container Queries, which allow elements to be targeted with queries instead of the viewport. This enables us, for example, to detect the width of an element and apply certain styles at different widths. This is an interesting topic with a couple of JavaScript libraries already offering support.

Animation is another area we'll be covering with the new Web Animations API, which aims to create a simple way to create animation in the browser, unifying all of the animation tools currently used to interact with elements in the DOM.

CSS GRID

CSS GRID HAS SOME POWERFUL POSITIONING TOOLS BUILT RIGHT INTO THE SPECIFICATION

To really harness the power of CSS Grid you need to understand how the grid lines are drawn. The number of grid lines is dependant on the number of rows and columns in the grid.

Numbering starts from the very top left of the grid with the number one – this is row one, column one. As a minimum with one cell in the grid, the coordinates (or start and end points) for the one cell would be row one/two and column one/two, with the cell spanning one single grid line on both axes, creating a single cell on a one-by-one celled grid.

The grid we'll be creating is a five by four, 20-celled grid. We'll be using the grid placement tools to position the last item in the grid in the centre while spanning three columns and two rows. To accomplish this we will only need to place 15 items into the grid – this is because the algorithm supporting grid will automatically place the other items around placed items. Since the centred item will span three columns and two rows, taking up six cells, the single cells equal 14 – including the last cell, which takes up the space of six cells, we have our total of 20 cells made up of 15 items.

- **5 * 4 = 20 cells (five columns, four rows)**
- **3 * 2 = 6 cells for our placed item (three columns, two rows)**
- **20 - 6 = 14 items without our placed item**
- **14 + 1 = 15 items when our placed item is re-added**

BROWSER SUPPORT

Browser support for CSS Grid is getting better and better, with the current versions of all major desktop browsers currently supporting it. Edge, however, only currently supports an older version of the specification requiring prefixes. The new specification is currently in development and will hopefully be in the new update, and you can keep up to date with the progress here on the Edge Development Roadmap (**srt.lt/b4P0T2**). It would seem mobile support has sneaked in almost undetected, with the latest updates to Chrome on Android, iOS Safari and Firefox for Android all adding support for CSS Grid.

57+ (Current version 57)

52+ (Current version 53)

12+ Uses an older version of the spec (Current version 15)

10.1+ (Current version 10.1)

HTML

The HTML consists of one div with the class of 'grid' and 15 divs inside each with a class of 'grid__item'. Each of the items also has a different background image.

POSITIONING THE LAST ITEM

Using the grid and column start and end properties we can position any item anywhere on the grid. Using the :last-child selector we can select the last item and position it in the centre of the grid.

```
.grid__item:last-child {
grid-column-start: 2;
grid-column-end: 5;
grid-row-start: 2;
grid-row-end: 4;
}
```

BASIC GRID

The CSS to create the basic grid requires the 'grid' display setting and then the row and column declarations:

```
display: grid;
grid-template-columns:
repeat(5, 1fr);
grid-template-rows:
repeat(4, 1fr);
min-height: 100vh;
min-width: 100vw;
```

SHORTHAND PROPERTIES

As with a lot of properties in CSS there are shorthand properties for grid positioning. These shorthand properties allow for the start and end positions to be grouped into one property. The example below, using two properties, is the same as the four used previously.

```
grid-column: 2/5;
grid-row: 2/4;
```

10 ESSENTIAL TIPS & TRICKS

1. GRID TERMINOLOGY

GRID CONTAINER: The element in which the 'display' property is set to 'grid', 'inline-grid' or 'subgrid'.

GRID ITEM: The direct children of the Grid Container become Grid Items.

GRID LINE: Either Column Lines (vertical) or Row Lines (horizontal). These lines are used for positioning elements onto the grid.

GRID TRACK: The space between two grid lines.

GRID CELL: The space between four grid lines and smallest unit in the grid.

GRID AREA: The space between four grid lines that can cover any amount of grid cells.

2. NEW MEASUREMENT UNITS

CSS Grid brings a new measurement unit: the 'fr' unit equaling a fraction. This new unit will account for margins in the grid meaning there is no need for calc() units.

3. GRID TEMPLATE ROWS AND COLUMNS

Set on the Grid Container, grid template rows and columns are used to define the number of cells that will be created on the grid. These properties will accept a number of different measurement units and will accept the new repeat, minmax and auto-fill functions.

```
grid-template-columns: 200px 20em auto 15rem 1fr;
grid-template-rows: repeat(4, 1fr);
grid-template-columns: repeat(auto-fill, minmax(15rem, 1fr));
```

4. GRID NAMED LINES

Lines on the grid can be named when being defined and then referenced for positioning.

```
grid-template-columns: [col1-start] 200px [col2-start] 20em  [col2-start] auto [col3-start] 15rem  [col4-start] 1fr [col4-end];
```

5. GRID POSITIONING

Using the Grid Line Numbers or Named Lines on the grid, Grid Items can be positioned anywhere on the grid regardless of their position in the DOM.

```
.grid__item:last-child {
  grid-column: 2/5;
  grid-row: 2/4;
  grid-column: col2-start / col5-start;
  grid-row: row2-start / row4-start;
}
```

6. GRID NAMED AREAS

Areas of the grid can be defined by giving child elements area names. These names are defined by the developer and can then be used when defining the grid template.

```
header {
  grid-area: header; }
```

7. GRID TEMPLATES

Once Grid Items have been given names grid templates can be created. Grid templates are written using an ASCII-style notation and can be redefined using media queries, allowing complex layouts to be easily maintained.

```
grid-template-areas:
  "header header header"
  "left-sidebar article right-sidebar"
  "footer footer footer"
;
```

8. REPEAT, MINMAX & AUTO-FILL

These powerful new functions allow for the browser to take suggestions you make about the layout and calculate the best way to fit all of the elements inside, especially when combined with the auto-flow properties.

```
grid-template-columns: repeat(auto-fill, minmax(15rem, 1fr));
grid-auto-flow: dense;
```

9. COMBINE WITH FLEXBOX

Grid Items can be turned into a flex-container allowing complex layouts to be created inside a single Grid Item, making CSS Grid perfect for page layout.

10. FIREFOX DEVTOOLS

Firefox Devtools has a CSS Grid Highlighter built in that will visually show you the lines of any grid. This highly useful tool can be quickly toggled on and off and can be a huge help when it comes to development.

CSS ELEMENT QUERIES

QUERY THE ELEMENT NOT THE VIEWPORT

Element containers have been a hot topic for a few years now. The idea is that we could query any individual element instead of the viewport. Using a similar syntax to media queries, EQCSS uses an '@element' query in the same way you would use a media query. There's also other syntax being considered that is more in line with current CSS:

```
@element .example and (min-width: 400px) {
.example/$this {...}
}
.example[min-width~="400px"] {...}
```

Tommy Hodgins (@innovati on Twitter and the author of EQCSS) is a huge advocate of this feature and has put forward one specification (**srt.lt/nE1X**) for CSS Element Queries as well as The Responsive Images Community Group (**srt.lt/kOJc8G**) who also have a specification written up.

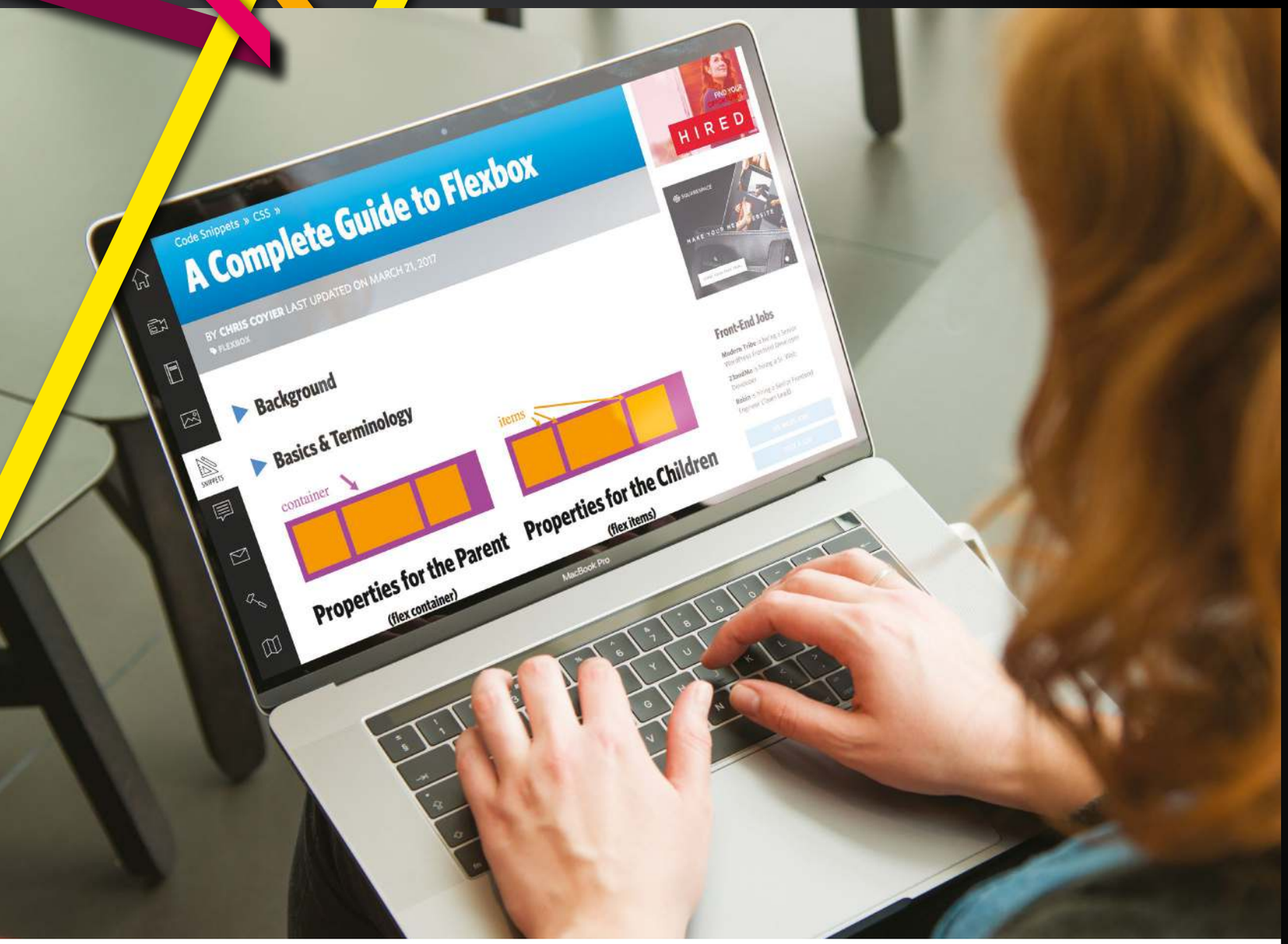
DEFINE A GRID LAYOUT WITH EQCSS

Once we have included the EQCSS script we can use Element Queries the same way we use media queries. By doing this we can check to see the width of the grid and change the layout by redefining the grid-template-areas when the grid is at a minimum width.

```
.grid {
  grid-template-areas:
  "header"
  "article"
  "left-sidebar"
  "right-sidebar"
  "footer"
  ;
}
@element .grid and (min-width: 500px) {
  .grid {
  grid-template-areas:
  "header header header"
  "left-sidebar article right-sidebar"
  "footer footer footer"
  ;
  }
}
```

DETECT ORIENTATION WITH EQCSS

EQCSS enables us to test for many different properties. One of particular interest is detecting the orientation of an element. If for example the element is displayed in the main content we can apply one set of styles, and others when placed in the sidebar or a smaller container. The same layout can be achieved using the Element



Queries library but requires a set width to be used.

```
/* Using EQCSS */
@element .box and (orientation: landscape)
{
  $this {
  display: inline-flex;
  }
  $this .image-container {
  margin-right: 1rem;
  margin-bottom: 0;
  }
}
/* Using Element Queries */
.box[min-width~=”400px”] {
  display: inline-flex;
}
.box[min-width~=”400px”] .image-container {
  margin-right: 1rem;
  margin-bottom: 0;
}
```

EQCSS PROS AND CONS

EQCSS (**elementqueries.com**) is a great library that adds a collection of new ways to approach responsive design. When combined with Flexbox and Grid new possibilities are created and truly dynamic, modular elements are written as a result. Browser support for the plugin is IE9+ but can support IE8 using an extra polyfill.

The downsides to using EQCSS is that this isn't something native to the browser, so using it requires an extra script being loaded into the page – and if you're a fan of squeaky clean markup EQCSS won't be your best friend.

Other libraries (CSS-Element-Queries for one – **srt.lt/Bz9f**) are trying to achieve the same thing using a different syntax and different extra markup, but still require the use of external resources.

PLANNED SUPPORT

When asked about native support for Element Queries, the CSS Working Group on Twitter (@csswg) recently said – "Not currently, no. Nobody's solved the architectural issue or attempted to re-architect their engine around such a dependency yet".

CUSTOM PROPERTIES

CUSTOM PROPERTIES (OR CSS VARIABLES) WILL SIMPLIFY DEVELOPMENT AND REFACTORING

Custom Properties are a powerful new tool that allow for a whole new level of flexibility when coding, with a few benefits over their Sass and LESS counterparts. Custom Properties are now aware of the cascade and can be inspected and updated using the browser's devtools at runtime, something not possible with preprocessors.

The new syntax is different from what we're used to, but this is not by mistake. This is to ensure they can be used with preprocessors and not conflict with the '$' or '@' used by existing languages.

Instead Custom Properties are defined or set beginning with a double dash --custom-property and then retrieved using var(--custom-property). It is best practice to set the Custom Properties inside the :root{...} selector – this is the same as 'html' but with a higher specificity. By declaring Custom Properties on the :root element using the normal CSS rules of specificity, they can be easily overwritten by a more specific selector.

Combining Custom Properties with existing CSS modules allows for some truly flexible code. For example, we can set a Custom Property for the font-size and then base all of our other spacing values and font-sizes on that single value, ensuring hierarchy and rhythm all using the calc() function.

CSS SCROLL SNAP POINTS

SCROLL SNAP POINTS AIM TO TAKE SCROLL MANIPULATION AWAY FROM JAVASCRIPT AND IMPLEMENT IT NATIVELY IN CSS

Anyone who has tried to create some kind of scroll locking, snapping or manipulating will know how tricky it can be. Until recently JavaScript was the only option but always seemed produce a mixture of success and choppy or questionable performance from device to device. Using this new property, carousels or sliders and other scroll manipulation techniques can be easily created and controlled using CSS.

Due to varying implementations of the specification multiple properties are required to experiment with this new CSS. The main new properties required to experiment with this new feature are:

```
scroll-snap-type: mandatory;
scroll-snap-destination: 100% 0%;
scroll-snap-points-y: repeat(100%);
```

The scroll-snap-type property currently accepts three values; none (the default value), mandatory and proximity. The mandatory value will force the scroll inside the snap container to the nearest snap point, proximity will snap but in a much more relaxed manner. Both the scroll-snap-destination and scroll-snap-points-y are applying the same value: both apply the scroll snap points to 100% of the height of the elements inside the scroll snap container (defined by the scroll-snap-type) property, along the Y (vertical) axis. Combined with the 'vh' and 'vw' properties fullscreen elements can easily be created.

```
.carousel {
  width: 100vw;
  height: 100vh;
  overflow-x: hidden;
  scroll-snap-type: mandatory;
  scroll-snap-destination: 100% 0%;
  scroll-snap-points-y: repeat(100%);
}
.carousel__item {
width: 100%;
height: 100%;
}
```

UPDATE CSS CUSTOM PROPERTIES WITH JAVASCRIPT

USING CUSTOM PROPERTIES AND CALC() WE CAN ENSURE HIERARCHY AND VERTICAL RHYTHM

In this short tutorial we'll be creating a simple page with input controls. The CSS will contain a number of CSS Custom Properties, which we can update with input from the user using JavaScript.
We'll be using the calc() function to set the font-sizes and spacing, allowing everything to be updated by changing a single Custom Property.

01. HTML SETUP

We'll start by building a simple page with a headline, paragraph and some basic form elements to control our Custom Properties with input from the user, using the 'onchange' property on the form elements to fire our JavaScript function.

```
<div class="page">
<h1>...</h1>
<p>...</p>
<input id="get-size" type="range"
step="0.005" min="1" max="1.3" onchange="upd
ateCustomProperties()" value="1.040">
</div>
```

02. OUR FIRST CUSTOM PROPERTY

In our CSS we'll create a :root selector and create our first Custom Property. Using the double dash ('--') prefix we can then add any name we want, so we'll start by creating a base font size.

```
:root {
--base-font-size: 1rem;
}
```

03. CALC() MAGIC

Once we've created a Custom Property for the font-size, we can create our spacing unit Custom Property and base that on a calculation of the font-size Custom Property using the CSS calc() function. Our calculation works like this; 1rem is equal to 16px (the default browser font-size), so our base-font-size is 16px. Our Spacing unit is then 16px multiplied by one and half. So by default our spacing-unit is equal to 24px.

```
--spacing-unit: calc(var(--base-font-size) *
1.5);
```

04. USING OUR CUSTOM PROPERTIES

Now we've set our Custom Properties on the :root element we can use them when styling elements in the page by using the new syntax var(). Inside this we recall our Custom Property name and optionally multiple fallback values, which can be other Custom Properties or values.

```
body {
font-size: var(--base-font-size, 1rem);
padding: calc(var(--spacing-unit, 1.5rem) *
2);
}
h1 {
font-size: calc(var(--base-font-size, 1rem) *
2.4);
margin: 0 auto calc(var(--spacing-unit,
1.5rem) * 1.5);
}
```

05. JAVASCRIPT FUNCTION

Using JavaScript we can take the input from the range slider to update the Custom Property controlling the font size, which will in turn update the spacing unit using calc().

```
function updateCustomProperties() {
var size = document.getElementById("get-
size").value;
document.body.style.setProperty('--base-font-
size', size + 'rem');
}
```

We can also use HTML5 colour inputs to change the background and text colours by adding them to the HTML and then adding new variables and setProperty values inside the JavaScript function.

BROWSER SUPPORT

CSS Custom Properties are now supported in the current version of all modern browsers, Edge included. Mobile support as of the most recent versions is good too with only IE, Blackberry and Opera Mini lacking support.

49+
(Current version 57)

31+
(Current version 53)

15+
Uses an older version of the spec (Current version 15)

9.1+
(Current version 10.1)

ESSENTIAL READING

- **CSS-Tricks**
  Covers everything you need to know about front-end development and blurring the lines into UI/UX, design and back-end development. CSS Tricks offers blogs, videos, forums and an impeccable CSS Almanac making it well worth adding to your bookmarks.
- **Smashing Magazine**
  Through their articles, digital books, job resources and industry conferences they have a lot to offer anyone in the web industry. Consistently provides quality resources for beginners and pros alike.

BROWSER SUPPORT

At the moment Scroll Snap Points have quite limited browser support, with Firefox offering the best support. However, the specification became a candidate recommendation in February this year so wider support should soon follow (**srt.lt/rT2xJr**). There's also a polyfill available to provide support for Chrome (**srt.lt/Er6Ri**).

In development

12+

39+

9+

Above: *When the user scrolling has stopped the browser will automatically scroll to the closest snap point*

WEB ANIMATIONS API

WHETHER YOU'RE ANIMATING IN CSS OR JAVASCRIPT, THEY WILL BOTH HARNESS THE POWER OF THE SAME UNDERLYING ENGINE

There are so many methods out there when it comes to animating on the web, from the old days of Flash through to JavaScript with jQuery, .animate(), then CSS3 and SVG and the animate tag, birthing libraries like Velocity and GreenSock.

The Web Animations API hopes to achieve a mixture of all of these libraries and tools by using the power and simplicity of CSS animations and the flexibility of JavaScript to interact with elements on the page, including SVG, leaving the browser to deal with performance.

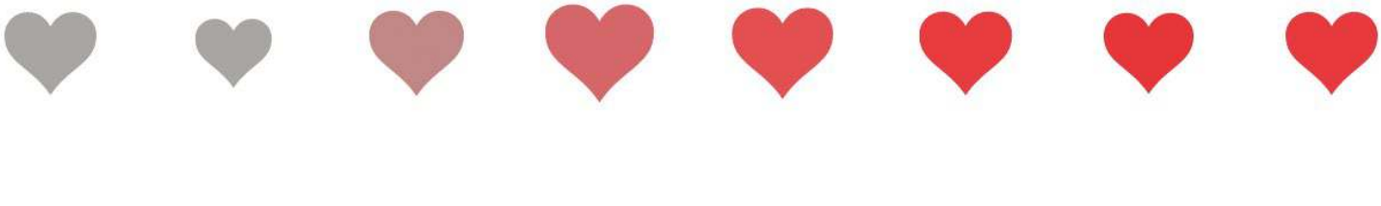
Basic usage of the API requires the use of '.animate' inside of JavaScript. The '.animate' call takes two parameters; an array of keyframes and the timing properties for the animation. If you've written CSS animation before this will look really familiar.

```
element.animate([
  { transform: 'rotate(0)' },
  { transform: 'rotate(360deg)'}
], {
  duration: 5000,
  iterations: Infinity,
  direction: 'normal',
  fill: 'forwards',
  delay: 100,
  easing: 'linear'
});
```

The new API keeps the benefits gained from CSS, like hardware acceleration, and adds more tools such as variables, finer controls and playback controls.

KEYFRAME EFFECTS

The new KeyframeEffect property in JavaScript adds more power to the API, allowing for animations to be modular and easily reused. The property takes three parameters; an element, an array of keyframes and timing properties. All of these can be mixed into as many KeyframeEffect properties as you want.



Above: *A selection of the frames that make up the animation*

CREATE A SIMPLE 'LIKE' ANIMATION

USE THE WEB ANIMATIONS API TO CREATE A SIMPLE SVG ANIMATION

The animation we'll be creating will consist of four keyframes, increasing and decreasing the scale and changing the colour of the SVG. We can also use JavaScript to retrieve the values of Custom Properties set in the CSS. Although this animation could be created in CSS we'll be using JavaScript to familiarise ourselves with the Web Animations API.

01. HTML AND CSS SETUP

The HTML and CSS is really simple. It's just a button containing an SVG and some CSS to apply some basic styling including some Custom Properties.

```
<a class="button">
<svg class="heart">
<path d="..."></path>
</svg>
</a>
:root {
--colour-inactive: #a4a4a4;
--colour-active: #ff4848; }
```

02. GET OUR CUSTOM PROPERTIES IN JS

By setting the colours we're going use in the animation as Custom Properties, we can then pull them out in the JavaScript and use them in our animation. To do this we need to set them as variables in JavaScript.

```
var getCustomProperties = window.
getComputedStyle(document.
querySelector("html"));
var colourInactive =
getCustomProperties.getPropertyValue('--
colour-inactive');
var colourActive =
getCustomProperties.getPropertyValue('--
colour-active');
```

03. SET THE KEYFRAMES

Next we'll create a variable containing the keyframes for our pop animation; we'll do this using the transform: scale() property. By shrinking the element slightly first and then pushing it over the initial size and back down, we're creating an animation that expresses joy and seems 'happy' when coupled with the animated colour change.

```
var keyframesPop = [
  { transform: 'scale(1)' },
  { transform: 'scale(0.8)', fill:
colourInactive },
  { transform: 'scale(1.2)' },
  { transform: 'scale(1)', fill: colourActive
}
];
```

04. SET UP THE TIMING PROPERTIES

Now we have our variable containing our keyframes we need to create another variable containing our timing properties. The duration is set in milliseconds, the number of iterations, the fill (which controls what happens when the animation is finished) and the easing, which we will set in cubic-bezier to give us greater control over the easing.

```
var timingPop = {
  duration: 750,
  iterations: 1,
  fill: 'forwards',
  easing: 'cubic-bezier(.61,-0.4,.69,1.32)'
}
```

5 KEY PROPERTIES

.ANIMATE()

The simplest way to create an animation is to call .animate() on an element.

TIMELINE

The player timeline allows for the current time of an animation to be either written or read using .currentTime(), allowing multiple animations to be synced or staged.

CONTROLS AND PLAY-STATES

Play-states can be detected and triggered using .pause(), .play(), .cancel() or .finish(), allowing animations to be user controlled or chained together.

PLAYBACK RATE

A separate control for the speed of playback allows for fine-tuning an animation without changing the easing or duration properties using animation. playbackRate = .25.

KEYFRAME, GROUP & SEQUENCE EFFECTS

These new properties provide a simple way to group keyframes, timing properties and animations into groups, which can be played in sequence.

CRTICAL CSS: WHAT & WHY?

SPEED UP YOUR WEBSITE WITH CRITICAL CSS

Critical CSS is a way of reducing the number of render blocking elements in order to render the page as quickly as possible.

Render blocking is caused when the browser has to wait until all of the CSS files in the <head> of the document have downloaded before rendering the page. This is so that the browser can accurately calculate the layout. So large CSS files can significantly slow down the page.

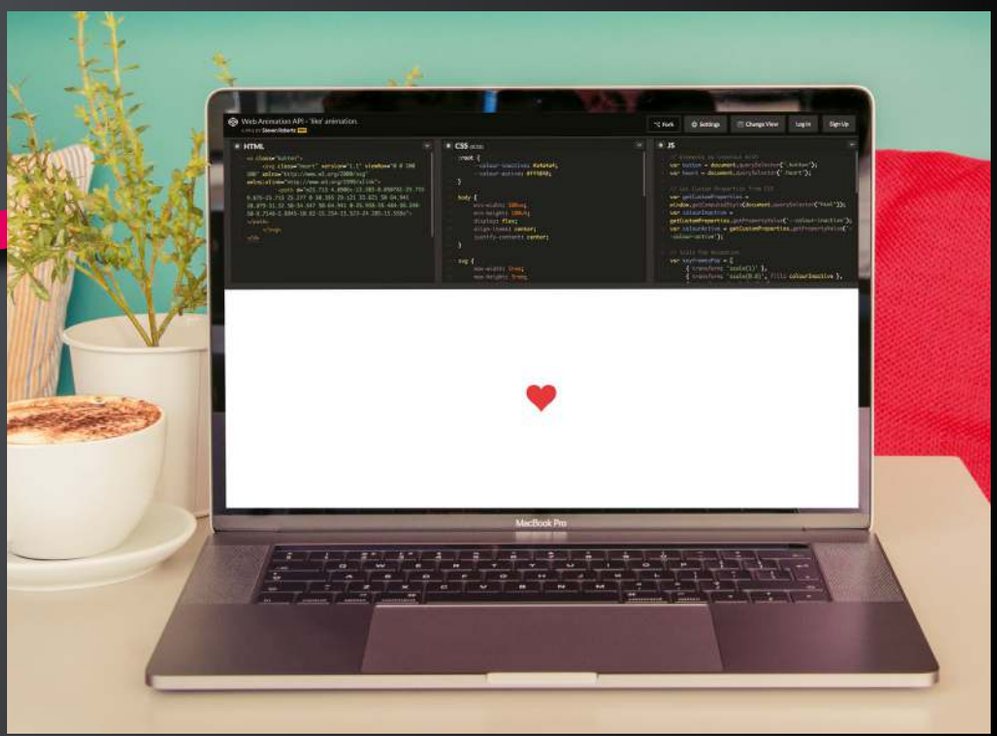
Critical CSS is a way of optimising the critical rendering path or, to put it another way, prioritising the display of elements. This is achieved by inlining the styles for elements that will be seen first (the above-the-fold content) into a <style> tag in the <head> of the document. The remaining CSS is then loaded asynchronously into the page with JavaScript.

There are a number of ways to achieve this: it can be done manually, or by using build tools such as Gulp and Grunt (**github.com/addyosmani/critical**).

ESSENTIAL READING

- **Critical Rendering Path**
  This free, in-depth course from Google covers every aspect of web performance, covering exactly how the browser renders the page and how to optimise your code.
  srt.lt/S8j6

- **Understanding Critical CSS**
  This article written by Dean Hume on Smashing is an in-depth look at Critical CSS, including tutorials to get you creating faster rendering pages in no time.
  srt.lt/WtE6a1



05. PLAY THE ANIMATION ON CLICK

With both our keyframes and timing properties created as variables we now need to combine them into an animation when the button is clicked. To do this we find the button and heart in the page and then listen for the button being clicked, running the '.animate' function and passing it to the two required parameters on the heart when clicked.

```
var button = document.querySelector('.
button');
var heart = document.querySelector('.heart');
button.addEventListener('click', function(e) {
  heart.animate(keyframesPop, timingPop);
});
```

06. POLYFILL AND FALLBACK

Due to limited browser support it's recommended to use a polyfill to provide support for a few more modern browsers (**github.com/web-animations**). For browsers with no support we can provide a fallback. Inside the button click function we can add a class of is-active to the heart and then in CSS we can change the colour when this class is applied.

```
heart.classList.add('is-active');
.heart.is-active {fill: var(--colour-active);}
```

BROWSER SUPPORT

Browser support is currently limited to Chrome, Firefox and Opera but both Edge and Safari have the feature 'Under Consideration', so hopefully we'll see it in the not-so-distant future.

36+

Under consideration

33+

Under consideration

Animate multiple elements simultaneously

As seen on **tram-house.com**

Side scrolling
The site's home page is side scrolling with the user still scrolling downwards to move the content on. Page elements animate onto the screen from the sides as the user moves through.

Home link
As with any project, using the logo takes the user back to the home page. As other content loads it will cover the existing design, so it's important to ensure users can get back to the main content.

Site music player
The old record player icon controls the music playing on the site, and it simply allows for it to be turned on and off.

Interactive exploration
Users have to interact with content to explore further, in this case dragging the key into the lock to load more content onto the page.

Loading animation
As the user puts the key in the lock, concentric circles animate outwards in bands of strong colour, taking over the screen until it loads the new content.

EXPERT ADVICE
Include or exclude music?
The inclusion of music on sites is a tricky bone of contention with web designers and users alike. Suddenly having music blasting out of your speakers, especially in a work environment, can be shocking. Try and keep the music subtle and unintrusive, but always let the user decide if it should be there.

Animate multiple elements simultaneously

⬇DOWNLOAD TUTORIAL FILES www.filesilo.co.uk/webdesigner

A tram-themed website for an upcoming apartment development might seem a little odd when you first visit this site, but the site's aim here is to take the essence of memories of trams and what they evoke, using that as a theme. The site claims that the tram taps into memories from our childhood, a hark back to halcyon days. The tram is about movement as well and the site is an "endless movement of our dreams about kind and joyful life".

The site asks the user to scroll down as normal, but as soon as they do the site starts to scroll horizontally instead of vertically. Animation of typography and images feature strongly as the user scrolls and elements are brought together into a well-designed site that entices the user to interact at different stages.

One of the best uses of interactions on the site is the act of getting the user to move a key into a lock. A strong metaphor for what the overall purpose of the site's goals are, selling apartments! Aesthetically this has a bold use of imagery with the site using a distinguished hipster gentleman throughout to add character to the many areas and later with some unexpected animation that's definitely worth checking out.

<comment>
What our experts think of the site

Bringing the design together

"The tram concept has been used extensively throughout the design and while the design is extremely contemporary, there is great use of imagery that invokes a bygone time that is forgotten. The consistency of this message is completely in keeping with the design of the new apartments."
Mark Shufflebottom

Technique

1. Random tram movements

The Tram House site features a pre-loader with some trams moving left and right in random movement, which will be explored here. The HTML structure for this is fairly simple with the container being the columns and the tram div containing the tram animation.

```
<div id="content">
<div class="container">
<div class="tram"></div>
</div>
<div class="container">
<div class="tram"></div>
</div>
<div class="container">
<div class="tram"></div>
</div>
</div>
```

2. CSS animation

There are a number of different classes used to control the columns and content on the screen. The main two that are of importance to us are the ones that contain the animation. Here the left animation is set up so that it calls keyframes, moving the background image.

```
.left {
animation: mleft 1s linear; }
@keyframes mleft {
0% { background-position: 500px 0px; }
100% { background-position: -700px 0px; } }
```

3. Randomised animation

The Tram House site features animation from the left and right so this is the CSS animation for the right. These will be called from JavaScript and by having two different animations will achieve the random look.

```
.right {
animation: mright 1s linear; }
@keyframes mright {
0% { background-position: -700px 0px;}
100% {background-position: 500px 0px;} }
```

4. Calling the page

The main functionality is found in the JavaScript, and the page content is stored in a variable so that it is cached. The trams are then put into an array called 'trams'. A timer variable will later control the repeat timing.

```
var content = document.
getElementById('content');
var timer;
function makeChange() {
var trams = content.querySelectorAll('.
tram');
```

5. Changing up

The trams need to be given a new animation so the left and right classes are removed if they have previously been applied. A random number generates a margin top to each of the tram div tags on the page.

```
function change() {
for (let i = 0; trams.length > i; i++) {
trams[i].classList.remove('left');
trams[i].classList.remove('right');
var random = Math.random();
var ht = Math.ceil(random * 400);
trams[i].style.marginTop = ht + "px";
```

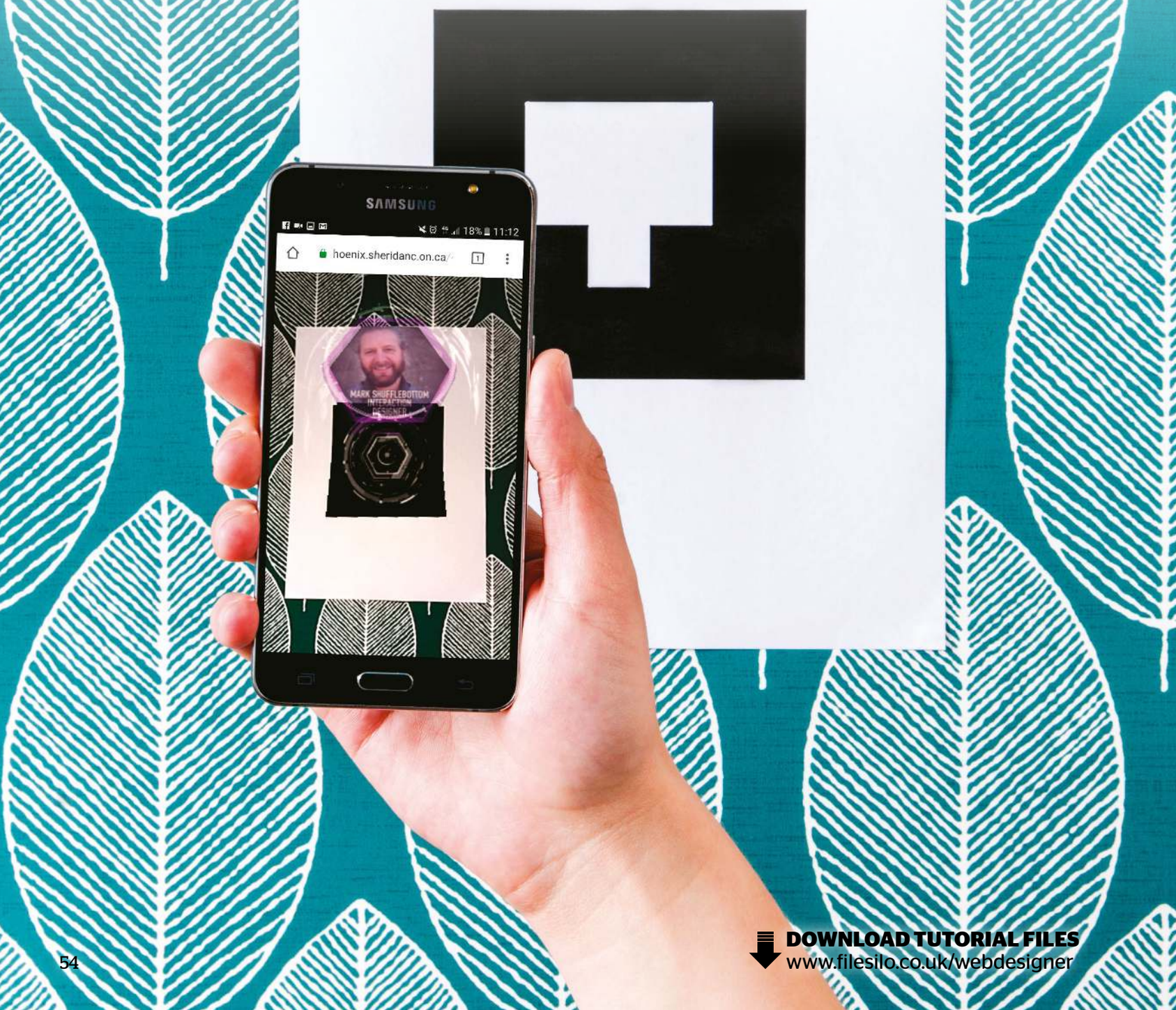
6. Applying the class

The final part is to apply either the left or right animation CSS class. The timer continues to call the change function so that it is applied to the trams.

```
if (Math.ceil(2 * random) === 2) {
trams[i].classList.add('left');
} else {
trams[i].classList.add('right'); } }
timer = setTimeout(change, 970);  }
change(); } makeChange();
```

Code augmented reality markers

Augmented reality can be achieved in the browser using native web technologies, detecting markers and displaying content



DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/webdesigner

Augmented reality has been around for a while now and it isn't anything new, however with the support of WebRTC (Real-Time Communication), access to the phone's camera can now be realised on Android devices and on desktop. At present, iOS can't support this as it hasn't been implemented in the WebKit browser that powers Safari, but it is in development and you can check the status of it here **bit.ly/2qadVFS**. If you do have an iOS device though you don't have to miss out, as you can still use the webcam on your desktop computer. An important note to get this working on the mobile Chrome browser is that the content must be served by a secure socket layer over https rather than standard http. Desktop currently works with regular http though.

In this tutorial an augmented reality marker is going to be placed in front of the camera. This will be picked up by the browser and AR.js, and content will be mapped over the top in 3D, sticking to the AR marker. The idea here is that you might want to create a simple 3D résumé, maybe even adding some of your core skills, and then the AR marker could be printed on your business card.

Because you can walk around the marker, this is great for content that you might want to see from different angles – think of a certain Swedish furniture manufacturer giving you animated steps that can be viewed from any angle! There are so many possibilities that this can be useful for.

1. Add the libraries

From the project folder, open the start folder in your code editor and then open up the 'index.html' file ready for editing. At this stage the libraries need to be linked up and as this is doing a lot, there are a lot of libraries! These are in three sections for Three.js, JSARToolKit, then the Three.js extension for the ARToolKit and marker.

```
<script src='js/three.js'></script>
<script src="js/ColladaLoader.js"></script>
<script src="vendor/jsartoolkit5/build/artoolkit.min.js"></script>
<script src="vendor/jsartoolkit5/js/artoolkit.api.js"></script>
<script src="threex-artoolkitsource.js"></script>
<script src="threex-artoolkitcontext.js"></script>
<script src="threex-armarkercontrols.js"></script>
<script>THREEx.ArToolkitContext.baseURL = '/'</script>
```

2. CSS styling

In the head section of the page, add some script tags and just drop in the style rules here for the body and the canvas element, to ensure they are placed correctly on the page without the default margins added by the browser.

```
body {
  margin: 0px;
  overflow: hidden;
}
canvas {
  position: absolute;
  top: 0;
  left: 0;
}
```

3. Global variables

In the body section of the page, add some script tags where the remaining JavaScript code for this tutorial will go. There are a number of variables that are needed; the first line is for Three.js, the second for the AR.js, the third for the model and then a variable to load the model.

```
var renderer, scene, camera;
var arToolkitContext, onRenderFcts,
arToolkitSource, markerRoot,
artoolkitMarker, lastTimeMsec;
var model, tube1, tube2, mid, details,
pulse;
var loader = new THREE.ColladaLoader();
```

4. Load the model

Before the scene is set up the model will be loaded so that it can be displayed when markers are detected. This is scaled down by 10 to fit exactly onto the AR marker. The model is 10cm for the width and height, so the marker is 1cm which translates to 1 increment in Three.js.

```
loader.load('model/scene.dae',
function(collada) {
  model = collada.scene;
  model.scale.x = model.scale.y = model.
scale.z = 0.1;
  details = model.getObjectByName("details",
true);
```

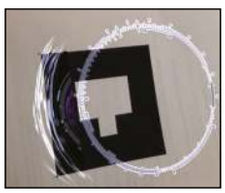
5. Fix some display issues

Still inside the Collada loading code, once the model is loaded there will be a couple of tubes that spin around so they are found in the Collada scene. The first tube is found and its material is grabbed. Here the material is set to just render on the inside of the model, not the outside.

```
tube1 = model.getObjectByName("tube1",
true);
var a = tube1.children[0].material;
a.transparent = true;
a.side = THREE["BackSide"];
a.blending = THREE["AdditiveBlending"];
```

Blend modes

Blending modes change the way that certain coloured pixels blend with pixels that are located below them. Additive blending makes all dark pixels transparent while adding light to the bright pixels.



Top left
The library AR.js has been downloaded and put into folders for you with all the dependencies, including the Three.js library

Top right
If the transparency and additive blending is not enabled, the model looks like this when loaded and displayed on top of the AR marker – not very exciting and barely visible!

Left
Experimenting with the lighting colours can give some different tints. Here a bright, light-blue colour has been applied to the direction light. Try experimenting with the colour of both lights

```
a.opacity = 0.9;
```

6. Repeat the fix

As in the last step, this same principle is repeated for the second tube and the blending mode, similar to those found in After Effects and Photoshop, is set to be an additive blend. This enables the outside of the pixels to have a softer transition to the camera image.

```
tube2 = model.getObjectByName("tube2", true);
c = tube2.children[0].material;
c.transparent = true;
c.side = THREE["BackSide"];
c.blending = THREE["AdditiveBlending"];
c.opacity = 0.9;
```

7. Final fix

The last model is a spinning circle just at the middle of the design. This follows the same rules as before but doesn't render the back of the object, just the front. The opacity of each of these materials has been set to 90% just to make it slightly softer. Once the model is loaded the init function is called.

```
  mid = model.getObjectByName("mid", true);
  b = mid.children[0].material;
  b.transparent = true;
  b.blending = THREE["AdditiveBlending"];
 b.opacity = 0.9;
  init();
});
```

8. Initialise the scene

The init function is set up and inside here the renderer settings are created. The renderer is using WebGL to give the fastest render speed to the content, and the background alpha value is set to transparent so that the camera image can be seen behind this.

```
function init() {
  renderer = new THREE.WebGLRenderer({
  alpha: true
  });
  renderer.setClearColor(new THREE.
Color('lightgrey'), 0);
  renderer.setSize(window.innerWidth,
window.innerHeight);
  document.body.appendChild(renderer.
domElement);
```

9. Create the scene display

The renderer is made to be the same size as the browser window and added to the Document Object Model of the page. Now an empty array is created that will store objects that must be rendered. A new scene is created so that content can be displayed inside of this.

```
onRenderFcts = [];
scene = new THREE.Scene();
```

10. Light up

To be able to see content in the scene, just like in the real world, lights are needed. One is an ambient grey light while the directional light is a muted blue colour just to give a slight tint to the 3D content on display in the scene.

```
var ambient = new THREE.
AmbientLight(0x666666);
scene.add(ambient);
var directionalLight = new THREE.
DirectionalLight(0x4e5ba0);
directionalLight.position.set(-1, 1,
1).normalize();
scene.add(directionalLight);
```

11. Lights, camera, action!

With the lights added to the scene, the next part to set up is the camera. As previously with the lights, once created it has to be added into the scene to be used. This camera will auto align with the position of the webcam or phone camera through AR.js.

```
camera = new THREE.Camera();
scene.add(camera);
```

12. Set up AR.js

Now AR.js is set up so that it takes the webcam as its input, it can also take an image or a prerecorded video. The AR toolkit is told to initialise and if it's resized it will match the same as the renderer on the HTML page.

```
arToolkitSource = new THREEx.
ArToolkitSource({
  sourceType: 'webcam',
});
arToolkitSource.init(function onReady() {
  arToolkitSource.onResize(renderer.
domElement)
});
```

13. Keep it together

Because resizing is something that happens a lot with mobile screens, as the device can easily rotate to the point that it re-orientates, the browser window is given an event listener to check for resizing. This resizes the AR toolkit.

```
window.addEventListener('resize', function()
{
  arToolkitSource.onResize(renderer.
domElement)
});
```

14. AR renderer

The AR.js needs a context set up, calling the Three.JS extension to do so. Here it takes the camera data file, which is included in the data folder, and detects at 30 frames per second with the canvas width and height set up for it.

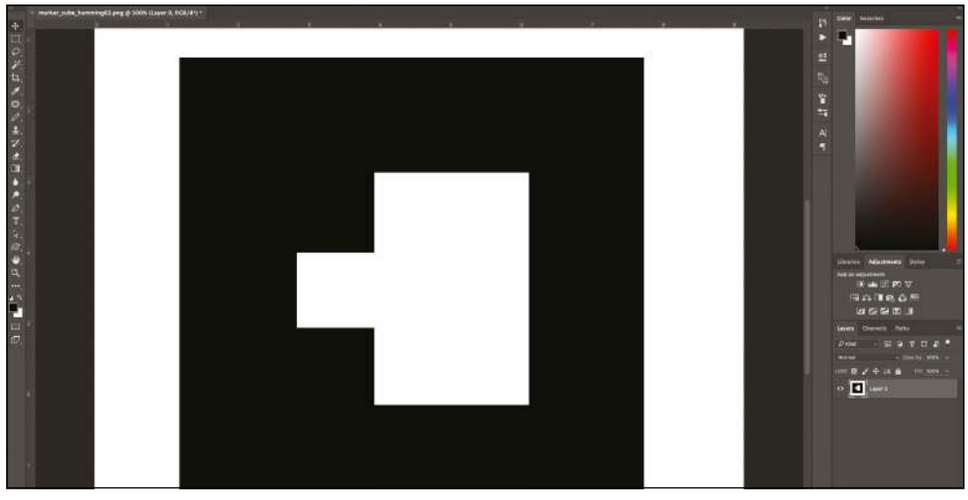
```
arToolkitContext = new THREEx.
ArToolkitContext({
```

Included marker

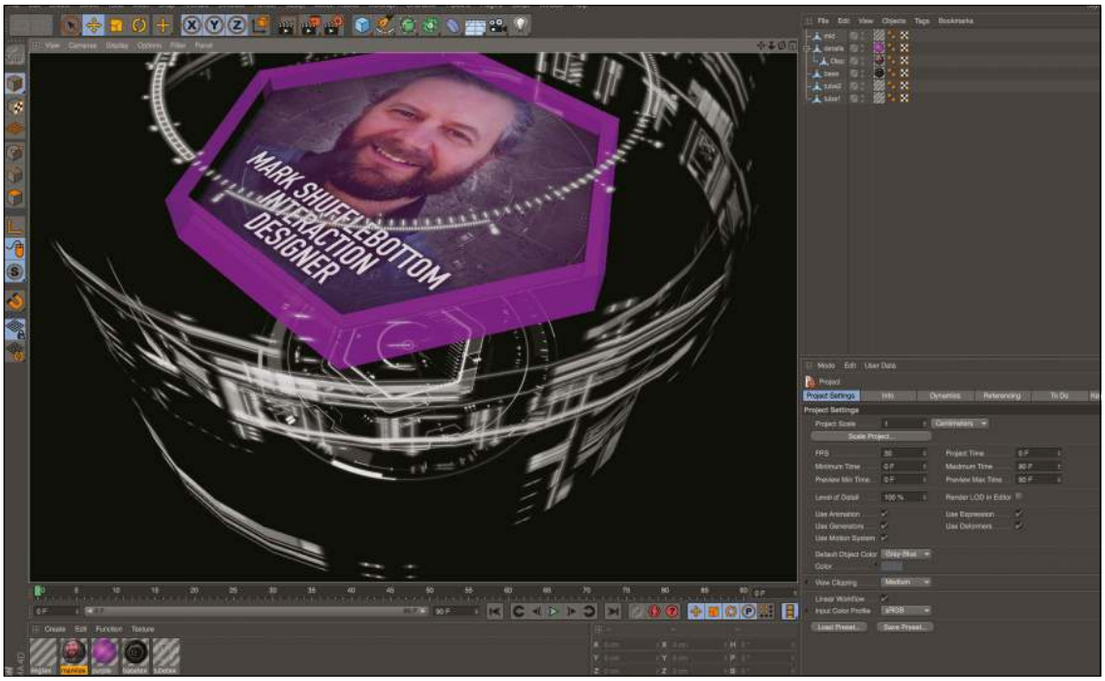
In the project files folder is the file 'marker.pdf'. If you print this, you can place either your webcam or a phone camera from the webpage to look at this so that it will display the 3D content.

Top left
Enabling the webcam in step 12 means that both desktop webcam and the phone's camera can be used to view the content

Top right
The different parts of the model are set to animate in step 17 and here the tubes and discs are spinning while the hexagon model in the centre moves up and down



Right
This is the image that will be detected by the camera as an AR marker; as you can see it shares some similarities with a QR marker, which you might be familiar with



Make the model

In this project the model was created in Cinema 4D, but most 3D modelling applications will be able to create the model. The main point you should look for is that you can export to Three.js, which does support a wide range of 3D model formats. One of the most popular is the Collada model format, which is XML based and has the file extension of DAE standing for Digital Asset Exchange. When the models are named in the modelling application, this ties up directly with the name of the models in steps 4 to 7 by using the getModeByName command. Once these are stored in a variable it's easy to get access to them and computationally animate these in the browser window.

```
  cameraParametersUrl: 'data/camera_para.
dat',
  detectionMode: 'mono',
  maxDetectionRate: 30,
  canvasWidth: 80 * 3,
  canvasHeight: 60 * 3,
});
```

15. Get the camera data

The AR toolkit is initialised now and the camera in the WebGL scene gets the same projection matrix as the input camera from the AR toolkit. The AR toolkit is pushed into the render queue so that it can be displayed on the screen every frame.

```
arToolkitContext.init(function onCompleted()
{
  camera.projectionMatrix.
copy(arToolkitContext.
getProjectionMatrix());
});
onRenderFcts.push(function() {
  if (arToolkitSource.ready === false)
return
  arToolkitContext.update(arToolkitSource.
domElement)
});
```

16. Match the marker

The markerRoot is a group that will be used to match the shape in augmented reality. It's first added to the scene, then this is used along with the AR toolkit to detect the pattern, which is also located in the data folder.

```
markerRoot = new THREE.Group
scene.add(markerRoot)
artoolkitMarker = new THREEx.ArMarkerControl
s(arToolkitContext, markerRoot, {
  type: 'pattern',
  patternUrl: 'data/patt.hiro'
});
```

17. Add the model

Back in the early steps a model was loaded and stored in the variable of the model. This is added to the markerRoot group from the previous frame. The model had some specific elements within it that are going to be animated every frame. They are also pushed into the render queue.

```
markerRoot.add(model);
onRenderFcts.push(function() {
  tube1.rotation.y -= 0.01;
  tube2.rotation.y += 0.005;
  mid.rotation.y -= 0.008;
  details.position.y = (5 + 3 * Math.
sin(1.2 * pulse));
});
```

18. Finish the init function

The renderer is told to render the scene with the camera every frame by adding it into the render queue, which is the array set up in step 9. The animate function is called, and this will render every frame to display content. The closing bracket finishes and closes the init function.

```
  onRenderFcts.push(function() {
  renderer.render(scene, camera)
  });
  lastTimeMsec = null;
  animate();
}
```

19. Just keep going

The animate function is created now and uses the browser's requestAnimationFrame, which is a call to repaint before the screen is drawn. This continues to call itself, and the browser attempts to call this function at 60 frames per second.

```
function animate(nowMsec) {
  // keep looping
  requestAnimationFrame(animate);
```

20. Timing issues

Mobile browsers sometimes find it difficult to reach 60 frames per second with different apps running. Here timing is worked out so that the screen is updated based on timing. This means if frames drop, it looks much smoother.

```
lastTimeMsec = lastTimeMsec || nowMsec -
1000 / 60;
var deltaMsec = Math.min(200, nowMsec -
lastTimeMsec);
lastTimeMsec = nowMsec;
pulse = Date.now() * 0.0009;
```

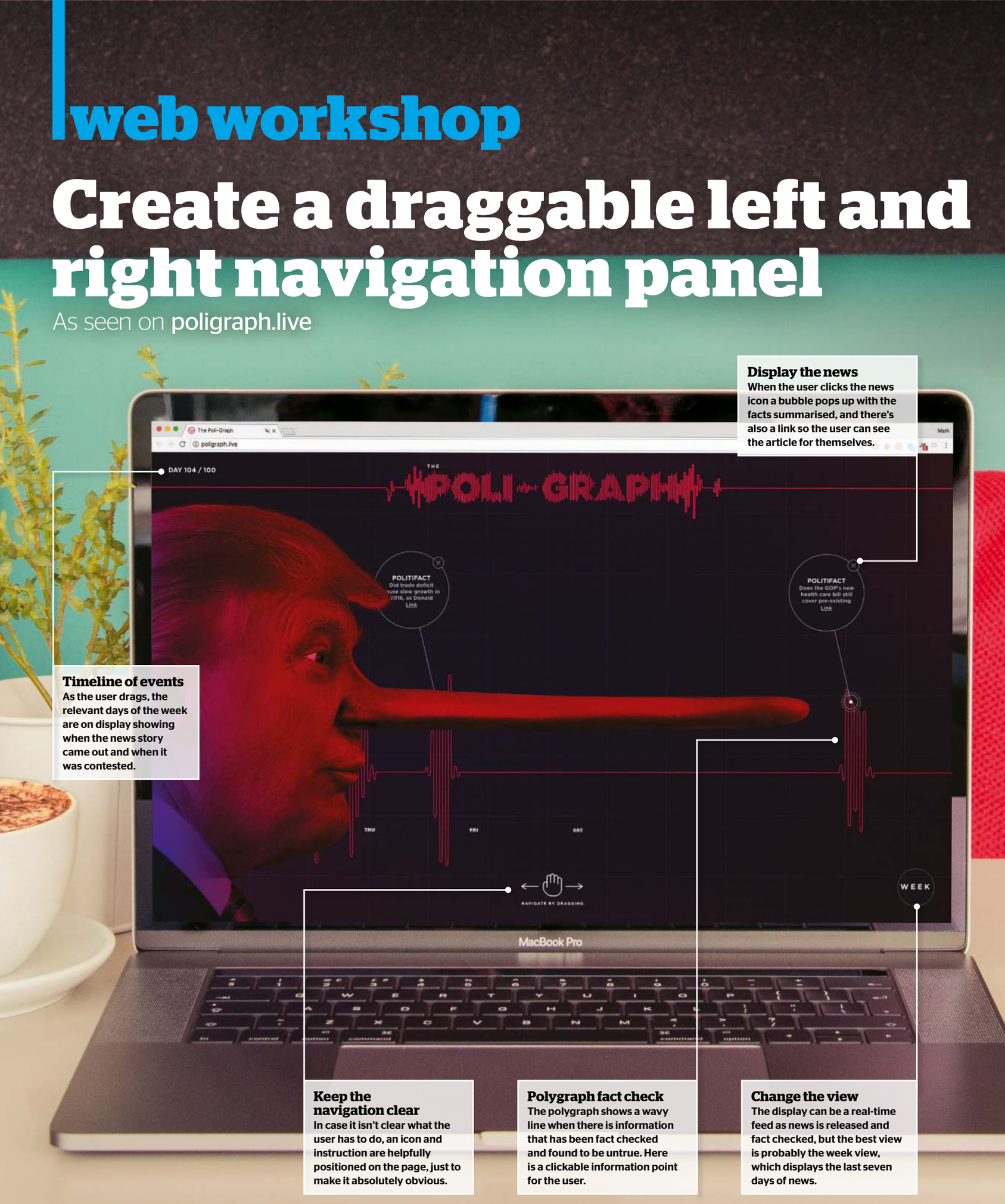
21. Finish it up

Finally each of the elements in the render queue are now rendered to the screen. Save the page and view this from a https server on mobile or a regular http server on desktop, print the supplied marker and hold it in front of the camera to see the augmented content.

```
onRenderFcts.forEach(function(onRenderFct) {
  onRenderFct(deltaMsec / 1000, nowMsec /
1000);
  });
}
```

Create a draggable left and right navigation panel

As seen on **poligraph.live**



Timeline of events
As the user drags, the relevant days of the week are on display showing when the news story came out and when it was contested.

Display the news
When the user clicks the news icon a bubble pops up with the facts summarised, and there's also a link so the user can see the article for themselves.

Keep the navigation clear
In case it isn't clear what the user has to do, an icon and instruction are helpfully positioned on the page, just to make it absolutely obvious.

Polygraph fact check
The polygraph shows a wavy line when there is information that has been fact checked and found to be untrue. Here is a clickable information point for the user.

Change the view
The display can be a real-time feed as news is released and fact checked, but the best view is probably the week view, which displays the last seven days of news.

EXPERT ADVICE
Function follows theme
As this site is named The Poli-Graph it sticks fairly rigidly to a polygraph for the site's theming. Using the polygraph idea for the news checking means that the graph itself becomes the navigation, and therefore the functionality follows out of the concept. This even dictates how the site content will look as the user browses the news.

Create a draggable left and right navigation panel

⬇DOWNLOAD TUTORIAL FILES www.filesilo.co.uk/webdesigner

There has been no end of controversy following the election of President Trump to the White House. One of the biggest areas of pontification has been around the subject of 'fake news', with even the President himself claiming that news outlets are making stories up. Simultaneously there have been a number of facts claimed by the President that have later been disputed, leading to a cat-and-mouse game of fact checking that seems to constantly pervade our news feeds.

Into this space comes the hilarious website, The Poli-Graph, which aggregates disputed President Trump facts by some of the largest global news outlets. The site was created by Camilla Ploug and Martin Furze who have day jobs as art directors, but they represent no organisation with this production.

Using the polygraph theme users get to browse the facts of President Trump in a timeline representing the graph line from a polygraph. Over the top of this is a fantastic PNG image with Trump's head, Photoshopped up with Pinocchio styling to show the extent of his lies. While this project is just a bit of fun, it does show some very disturbing realities about the state of the world in which we live.

<comment>
What our experts think of the site

Detecting porky pies

"The Poli-Graph is an online lie detector, hooking Trump up to it 24/7. Every false claim will be detected automatically via an AI and posted to our site the moment it is posted on the web. This was something we produced in our spare time, to highlight something we thought was important."
Camilla Ploug, Art director

Technique

1. Create the draggable timeline

The **poligraph.live** site has a draggable timeline that enables the user to drag a graph line back and forth. To recreate this effect, add the CSS on FileSilo to your page to hide the overflow content on the x axis to stop left to right scroll bars.

2. Draggable area

The draggable area is styled up now. It's been created much wider than the screen at 3,000 pixels wide. This is so that it can be dragged left and right. The background image is placed in to fill the background, and the cursor changes to a pointer when the mouse is over this.

```
#drag {
position: absolute;
width: 3000px;
height: 300px;
background: url(img/line.png) no-repeat;
cursor: pointer; }
```

3. HTML content

The HTML part of the code is very sparse, it really is just a nested div tag. The outer div is named 'container' and the inner element is 'drag'. There are mouse events on here for mousedown and mouseup to handle the mouse being pressed.

```
<div id="container">
<div id="drag" onmousedown='startMoving();'
onmouseup='stopMoving();'></div>
</div>
```

4. Add the functionality

Before the closing HTML tag on your page, add the script tags and enter the code here. The main part of this is to work out the screen's width and then take the 3,000-pixel width of the draggable div off this. This sets up how much we can then drag in each direction.

```
<script>
var drag = document.getElementById('drag');
var absRight = window.innerWidth - 3000;
drag.style.left = absRight + 'px';
function startMoving(evt) {
evt = evt || window.event;
var posX = evt.clientX;
var divLeft = drag.style.left;
divLeft = divLeft.replace('px', '');
var diffX = posX - divLeft;
```

5. Move the mouse

When the mouse is pressed on the draggable div, the function here is added to it. This works out the position of the div by taking its current position and taking off the current position of the mouse. The if statements stop it being dragged too far left and right.

```
document.onmousemove = function(evt) {
evt = evt || window.event;
var posX = evt.clientX;
var newX = posX - diffX;
if (newX > 0) newX = 0;
if (newX < absRight) newX = absRight;
drag.style.left = newX + 'px';
}
}
```

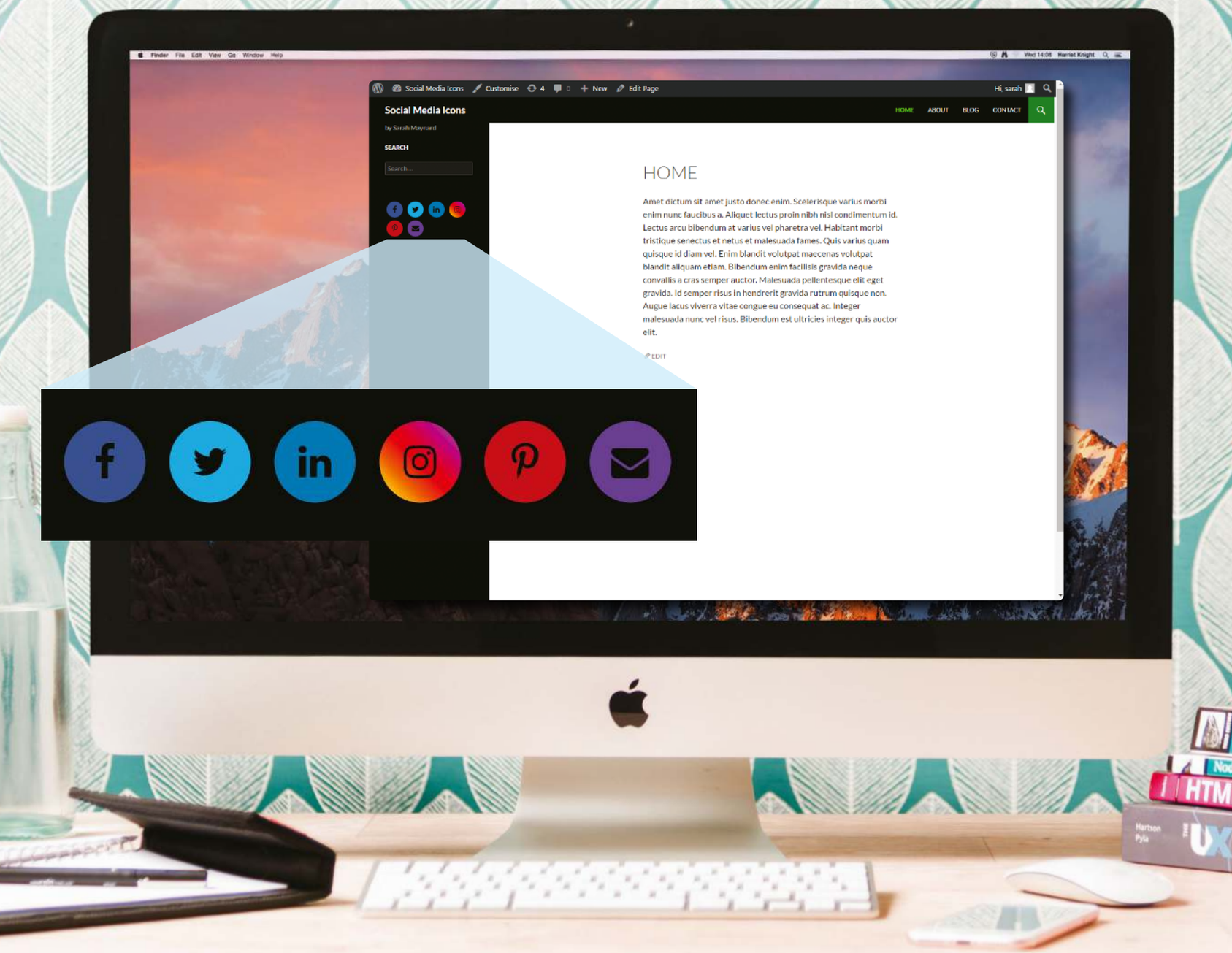
6. Release the mouse

Once the mouse is released the mousemove function is wiped out so that it doesn't keep firing. Save the document and try the page in your browser so that you can see the draggable content. To improve this you could add an event listener to check for the browser resizing and update the absRight variable with the new window width.

```
function stopMoving() {
document.onmousemove = function() {}
}
</script>
```

Code social media buttons for WordPress

Create great social media buttons using the WordPress menu tool, built-in icon fonts and a little CSS



DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/webdesigner

Social media is a huge part of our internet identity, so most websites have links to your sites of choice in a header, footer or sidebar. Engaging your audience and encouraging them to click on these links, let alone drawing attention to them in the first place, can be a tricky business. In this tutorial, we'll cover how to make icons that stand out, using just the tools that WordPress provides for us. You don't even need to know any coding, just be a copy-paste master!

We're opting for moderately sized buttons that will look great on mobile and desktop, with subtle hover states, coloured backgrounds, gradients, and even support for IE 9+ (with a graceful IE8 fallback). By using WordPress's powerful menu, a plugin and customisation tools, we can accomplish this all through the Admin panel without digging into background code. There's no need to crack open your theme, and all the changes made will be saved in the WordPress Admin panel.

This tutorial breaks down into three easy steps: set up a menu, install the FontAwesome icon font using a plugin, and add CSS to the customisation panel. If you're CSS-savvy, you can easily scale these buttons to be smaller, larger, and change the icon and background colours to match your theme.

1. Create menu in WordPress

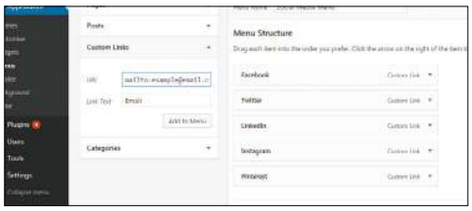
The first step is to set up the menu using the WordPress menu options. Go to your Admin panel, and open Appearance>Menus. Click 'Create a new menu' next to the Select dropdown. This will open a blank menu, now name your new menu and then click Create Menu.

2. Add menu items

Click the Custom Links dropdowns on the left-hand side. Add each social media link by inserting the URL and name into the fields provided and click Add to Menu after each one. You should end up with a list on the right-hand side. To reorder the list simply drag and drop.

3. Set email mailto link

For an email link, you can use the mailto link. Instead of a URL, in the URL box type mailto:example@email.com, with your contact address instead. When clicked on, this link will open a visitor's email client, with your email automatically in the To field.



4. Enable CSS Classes field

We also need to assign CSS classes to our menu items so, if it isn't already enabled, go to Screen Options in the top right and turn on the CSS Classes. Title Attribute and Link Target will also be useful, so turn those on too. Doing so will add more options to edit your menu items below.

5. Add CSS classes and titles

Add a title in Title Attribute, eg, 'Find us on Facebook'. Check the 'Open link in new tab' box. Assign two classes in CSS Classes, a shared and a unique class with spaces between, eg 'social-button facebook-button', 'social-button twitter-button'. Do this for each link and then click Save Menu.

6. Assign the menu to a location

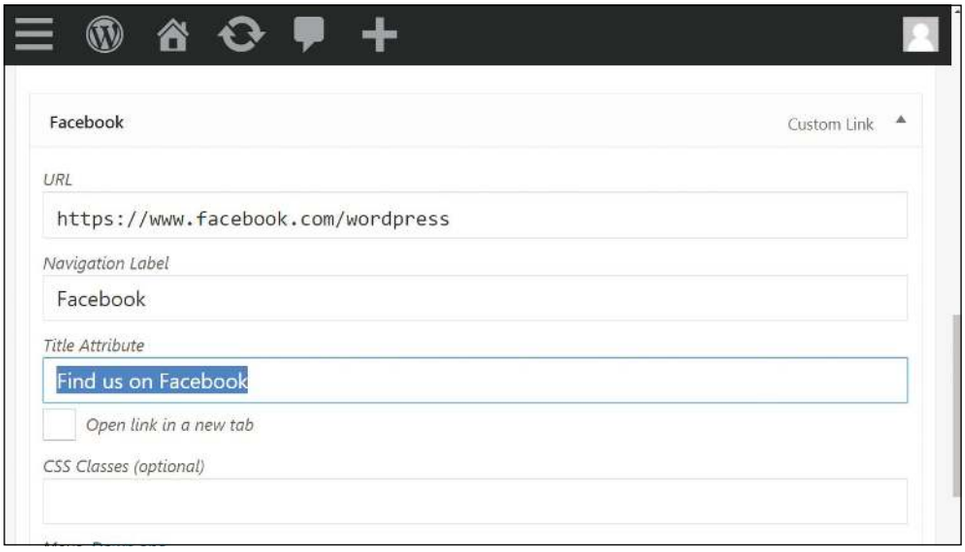
Assign the new menu to a theme-defined location at the bottom of the menu edit page by selecting one of the Display Location options. Alternately put the new menu into a widget by going to Appearance>Widgets, drag a Custom Menu widget into the sidebar and select the name of your new menu.

7. Install FontAwesome

WordPress has its own built-in font icon set which you can enable, but currently (WordPress Version 4.7.3) only packs Twitter and Facebook icons. For a more complete range, we can install FontAwesome, a free font icon set. For quick installation, get the Easy FontAwesome Plugin.

Edit link options

You can edit link configurations at any time. Expand link options with the dropdown arrow on the right of each bar. If you later need to change a link or remove one, you can do that here too!



Left
Adding a title will, by default, add a small hover state that shows the title so users know what to expect when clicking the link

Top left
Check the 'Open link in a new tab' option to turn on the target="_blank" attribute for links, so clicking them will not take users away from your main site

Top right
Adding classes will give us something easy to hook onto with CSS later. Make sure there's a space between each class, and use something memorable

8. Default menu display

Your new menu should now appear in your layout in the area you selected. It will currently be in the style of the other menus, with text and not icons. The generated markup, depending on your theme, will look similar to this (additional IDs and classes added by WordPress have been omitted for clarity).

```
<ul id="menu-social-media-menu"
class="menu">
<li class="social-button facebook-button"><a
title="Find us on Facebook" target="_blank"
href="URL">Facebook</a></li>
<li class="social-button twitter-button"><a
title="Follow us on Twitter" target="_blank"
href="URL">Twitter</a></li>
[...]
</ul>
```

9. Start styling

You can now add CSS to your theme files, or in WordPress 4.7 and above, you can add CSS via the Additional CSS theme options. Go to Appearance>Customise in the Admin panel and open the Additional CSS tab. CSS saved here will apply only to this theme, and will not edit your theme files.

FontAwesome is an icon font

They scale to any size, big or small! They are great for all screen types: desktop and mobile. They're also tiny in file size so they load faster than images and can be recoloured with CSS!

10. Align buttons side by side

Start by adding styles for all the buttons as a basis for customising each individual link. This code will start by aligning your buttons side by side. Depending on your theme you may need to make adjustments.

```
#menu-social-media-menu .social-button {
  display: inline-block;
  box-sizing: border-box;
  padding: 0;
  border: 0;
}
```

11. Turn text into button shapes

Add block and background to the anchor links with width and height settings to make the base shape. Use a tall line height and hide the overflow to hide the text out of the viewport while still making it available to screen-readers. Add a margin to put some space between buttons.

```
#menu-social-media-menu .social-button > a {
  display: block;
  background: #555;
  height: 30px;
  width: 30px;
  overflow: hidden;
  line-height: 200px;
  margin: 0px 3px;
  position: relative;
}
```

12. Add some style

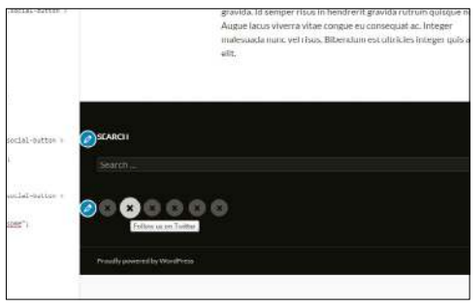
Give your buttons some panache! Set the border radius to 100% to make circles. Transition and transform can give your hover states a bit of a zoom and background colour change. This continues on from the last step's additions.

```
#menu-social-media-menu .social-button > a {
  [...]
  transition: all 0.2s;
  border-radius: 100%;
}
#menu-social-media-menu .social-button >
a:hover {
  transform: scale(1.2);
  background: #ccc;
}
```

13. Add icon placeholder

Now create the placeholder for the icon using the :after pseudo class. This will create a fake element using CSS that overlays your buttons, similarly to a div that we can style and add the icon to.

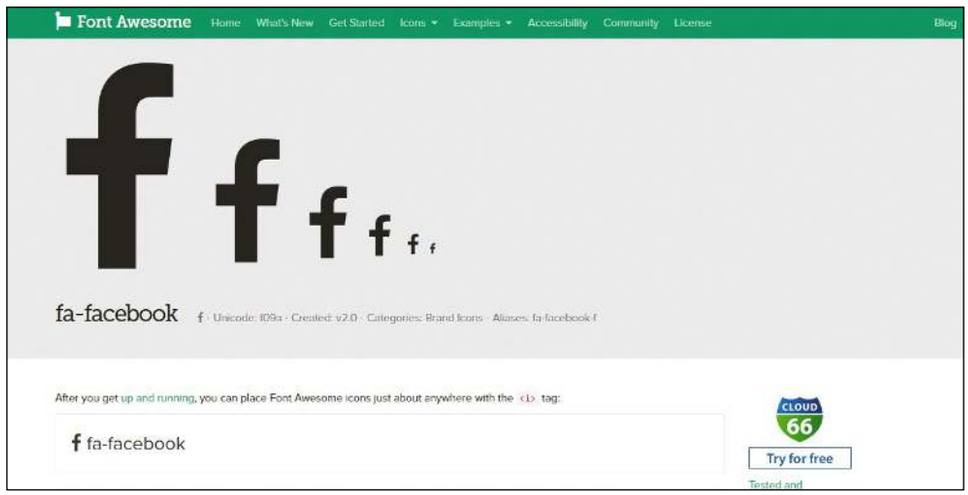
```
#menu-social-media-menu .social-button >
a:before {
  content: "\f00d";
  font-family: "FontAwesome";
  font-size: 14px;
  transform: rotate(0.001deg); /* IE Fix */
  color: #000;
  position: absolute;
  width: 30px;
  line-height: 30px;
  text-align: center;
}
```

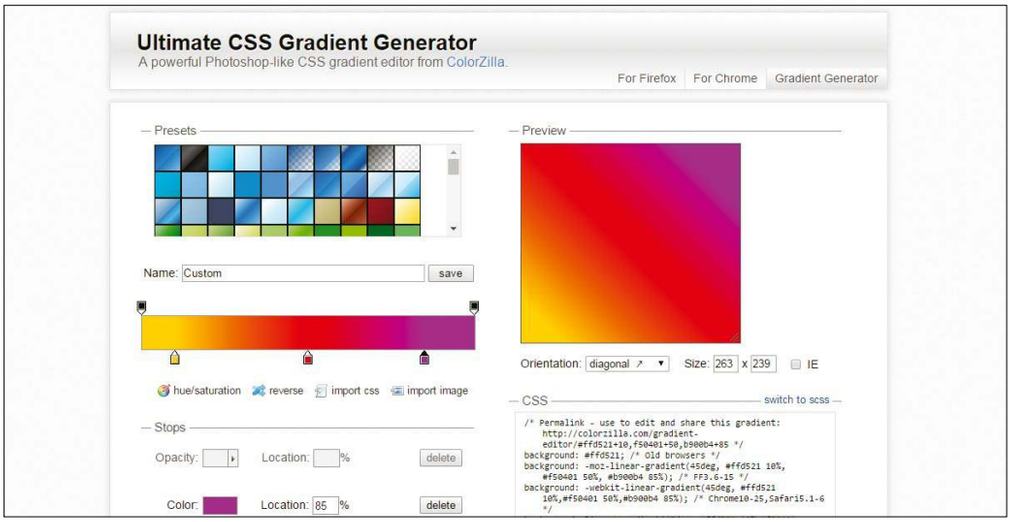


Top left
The list of FontAwesome icons is available at **fontawesome.io/icons**. There are many icon types available, not just social media logos

Top right
Use the search function to refine the view to a specific icon or scroll down to Brand Icons to see all site brand logos

Right
Click the icon to expand the full view. The unicode code that you need for CSS is in the grey bar just under the different sizes





Gradient backgrounds with CSS3

Creating gradient backgrounds can be time-consuming and require a lot of maths. Thankfully, tools exist to generate backgrounds for us! Google 'CSS gradient generators' for a selection. For this tutorial, we used the generator at **colorzilla.com/gradient-editor**. It provides an easy UI to choose your colours, lets you add and remove stops, and outputs code that's compatible with many recent browsers. It even provides IE support, for those versions that are capable (with some limitations). It's important to always provide a backup colour for browsers that do not support it, and you can always check **caniuse.com/#search=linear-gradient** to see which browsers it will work best on.

14. Get social icon unicode

Go to **fontawesome.io/icons** and search the name of each social network. Click the relevant icon and make a note of the 'Unicode' code. This is the character's position in the font file, and the CSS will use it to display the character as the icon.

15. Set up individual icon style

For each button copy the following code and create a rule for each, then change the class each time to match the unique ones previously set when the menu was created. For example, for this part there will be six versions of this rule, with the 'facebook-button' class different for each.

```
#menu-social-media-menu .social-button.
facebook-button a:before {
  /* icon here */
}
```

16. Add icons to CSS

Fill out all six button rules with the following. Use the unicode from **fontawesome.io/icons** and overwrite the content attribute for each button. Make sure to include the backslash before each unicode so CSS knows to escape the code and display the icon, otherwise it will display the code itself.

```
#menu-social-media-menu .social-button.
facebook-button a:before {
  content: "\f09a"; /* Facebook icon */
}
```

17. Icon display

All your icons should now have their unique icons. Being an icon font, all your icons should be lined up neatly, but if you need to change any of the positions, just add top or left values to the icon's individual :before: rules to bump the image. Use font-size to make the icons bigger or smaller.

```
#menu-social-media-menu .social-button.
facebook-button a:before {
  content: "\f09a"; /* Facebook icon */
top: 2px;
font-size: 12px;
}
```

18. Set up individual button style

Use each social media's brand colours to add some colour to your buttons. Start by making a new rule for each of the links, similarly to the way we added the icons to the :before element. Copy the following code for each button and amend the 'facebook button' class to the unique one for each.

```
#menu-social-media-menu .social-button.
facebook-button a {
  background: #HEX /* see next step */
}
```

19. Add colour to your buttons

Replace HEX inside each button's rule above, using the codes below. Each of these is a site's main brand colour (or a theme-fitting colour in email's case), place one into each of the rules you set up in the previous step, eg 'background: #3b5998;' for Facebook.

```
3b5998; /* Facebook*/
00aced; /* Twitter */
007bb6; /* LinkedIn */
d93175; /* Instagram */
cb2027; /* Pinterest */
6c469b; /* Email */
```

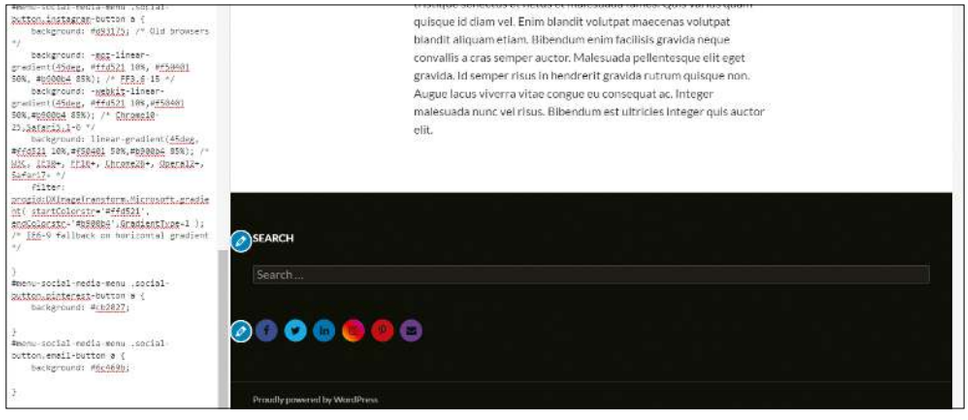
20. Instagram's gradient

Using CSS3 gradients, we can emulate the Instagram brand gradient. As gradients are not supported in all browsers, it's important to provide a backup colour. Try this code as your instagram background; the first line is your backup colour.

```
background: #d93175;
background: -moz-linear-gradient(45deg,
#ffd521 10%, #f50401 50%, #b900b4 85%);
background: -webkit-linear-gradient(45deg,
#ffd521 10%,#f50401 50%,#b900b4 85%);
background: linear-gradient(45deg, #ffd521
10%,#f50401 50%,#b900b4 85%);
```

21. Final word

For the finishing touches, make sure your :hover status is at the end of your CSS so it overwrites your button colours. If you feel like the left-hand correlation of blue is too much, use the menu tool to reorder your menu and alternate the blue buttons with other colours.



Create an interactive navigation display

As seen on **ivanaivazovsky.life**

Less essentials
Less essential features are placed in a location that doesn't obstruct the main content, but are still easily noticeable and accessible.

Navigation links
These are the page navigation links that alter the page background when the user hovers the mouse pointer over them.

Page background
The background image is set to cover the full area of the page. It is sized to cover the full area using background sizing.

Background positioning
Central positioning used for the background makes sure that the middle area of the image is always in view to avoid obvious clipping.

Logo and others
Like the less essential features, other navigation and the logo are placed in a way that doesn't obstruct the main page content.

⬇DOWNLOAD TUTORIAL FILES www.filesilo.co.uk/webdesigner

It's annoying for users to click on a link only to find that the webpage isn't of interest. Not only does this waste time, but also data – which is always going to be a concern for mobile users who are paying for their data usage.

With space often limited, especially on mobile devices, it is beneficial to provide additional information about content being linked to from page navigations in a way that doesn't require additional space. Using the page background is a good way to achieve this; using a picture that communicates the theme of the link provides the user with an opportunity to gain insight before committing to loading the page.

Another user experience design issue that using the page background can address are interpretation issues such as dyslexia. In these cases, the saying 'a picture speaks a thousand words' comes in very handy by allowing you to communicate additional information to the user without overloading them with too much text.

As with any effect that combines text with background images, make sure that your text remains readable at all times. This can be achieved by placing a semi-transparent colour on your navigation links. Also consider UX readability for issues such as colour blindness that affect how users are able to interpret coloured text.

<comment>
What our experts think of the site

As an accessibility asset

"The use of visual design to reduce written communication can help to make your website more accessible to people who have difficulty in reading text. Visual communication will benefit many people who have difficulties interpreting text."
Leon Brown, web developer and e-learning content creator at Nextpoint

Technique

1. HTML document

Create the main HTML document template, which consists of the HTML container storing the head and body section. The head section stores description information such as the document title, as well as links to external resources – ie CSS and JavaScript. The body stores the document content. A significant feature of the body is the 'data-theme' attribute it has applied – to be modified by JavaScript to trigger CSS presentation changes.

```
<!DOCTYPE html>
<html>
<head>
<title>Navigation Background</title>
<link rel="stylesheet" type="text/css"
href="styles.css" />
<script type="text/javascript" src="code.
js"></script>
</head>
<body data-theme>
*** STEP 2
</body>
</html>
```

2. Navigation HTML

The page content consists of a navigation container containing links. The navigation's ID and the titles of the child links will be used by JavaScript to listen for interactions and alter the 'data-theme' attribute defined as part of the body container in step 1.

```
<nav id="mainNav">
<a href="#" title="Page 1">Page 1</a>
<a href="#" title="Page 2">Page 2</a>
<a href="#" title="Page 3">Page 3</a>
</nav>
```

3. Initiate the CSS file

The HTML content is now complete, so create a new file called 'styles.css'. This file stores the CSS formatting rules that control the visual presentation. Start this file with rules to present the HTML document and its body to appear across the full screen using a white background and black text.

```
html,body{
display: block;
width: 100%;
height: 100%;
margin: 0;
padding: 0;
font-family: Helvetica, sans-serif;
color: #000; }
```

4. Theme definition

An important component for the effect is the use of the 'data-theme' attribute applied to the document body. A rule is set to define its default behaviour – which will be for background positioning and change transitioning. This avoids the need to repeat these rule definitions for each individual theme; this means it is also better for maintainability.

```
[data-theme]{
transition: background 1s;
background: no-repeat center center;
background-size: cover; }
```

5. Theme design

The design of each theme is set using the value of the data-theme attribute. To achieve the desired effect, we are setting a different background image for each version of the theme. The previous step takes care of all of the default settings that these themes will inherit.

```
[data-theme="Page 1"]{
background-image: url(image1.jpg);
}
[data-theme="Page 2"]{
background-image: url(image2.jpg);
}
[data-theme="Page 3"]{
background-image: url(image3.jpg);
}
```

6. Navigation container

The navigation container is used to make sure that links are presented with a consistent width placed in the middle of the screen. A width of 50% and auto calculation used for horizontal margining is applied. This approach provides a guarantee of consistency regardless of the user's screen resolution/size.

```
#mainNav{
display: block;
width: 50%;
margin: 0 auto 0 auto; }
```

7. Navigation links

The links inside the navigation container are to be displayed as blocks set to adapt to the width of the container. This means that their 100% width is defined by the container's width. Padding, borders and background colouring are also applied to make sure that they stand out from the background images.

```
#mainNav a{
display: block;
width: 100%;
background: rgba(255,255,255,.3);
color: #000;
padding: 1em;
margin-top: .5em;
border: 1px solid; }
#mainNav a:hover{
background: rgba(0,0,0,.5);
color: #fff; }
```

8. Link event listeners

The CSS is now complete, so create a new file called 'code.js' for the JavaScript. The effect requires each link inside the navigation to listen and react to a mouseover event where the user hovers over a link. This listener applies the 'title' value of the link to the data-theme attribute of the document body – hence triggering styles in the CSS. After the page window had loaded, the navigation link nodes are placed into an array, upon which a for loop is used to apply the event listener.

```
window.addEventListener("load", function(){
var nodes = document.
querySelectorAll("#mainNav a");
for(var i=0; i<nodes.length; i++){
nodes[i].addEventListener("mouseover",
function(){
document.body.setAttribute("data-theme",
this.getAttribute("title"));
});
}
});
```

WHICH FRONT-END FRAMEWORK?

FOR MANY WEB DEVELOPERS FRAMEWORKS ARE A MUST, YET THE EVER-GROWING LIST OF OPTIONS CAN BE MYSTIFYING. WE TAKE A LOOK AT THE FEATURES OF SOME POPULAR FRONT-END JAVASCRIPT FRAMEWORKS, AND HOW TO MAKE THE BEST CHOICE FOR YOUR PROJECT

FRAMEWORKS: SIMPLIFY YOUR WORKFLOW

Frameworks can help you develop better, faster code. But is it worth the effort to learn one?

When we talk about front-end frameworks, fundamentally we're thinking of collections of tools and libraries that aim to simplify web development practices. You've probably heard of some of these or used them already; Angular and React are currently the most popular.

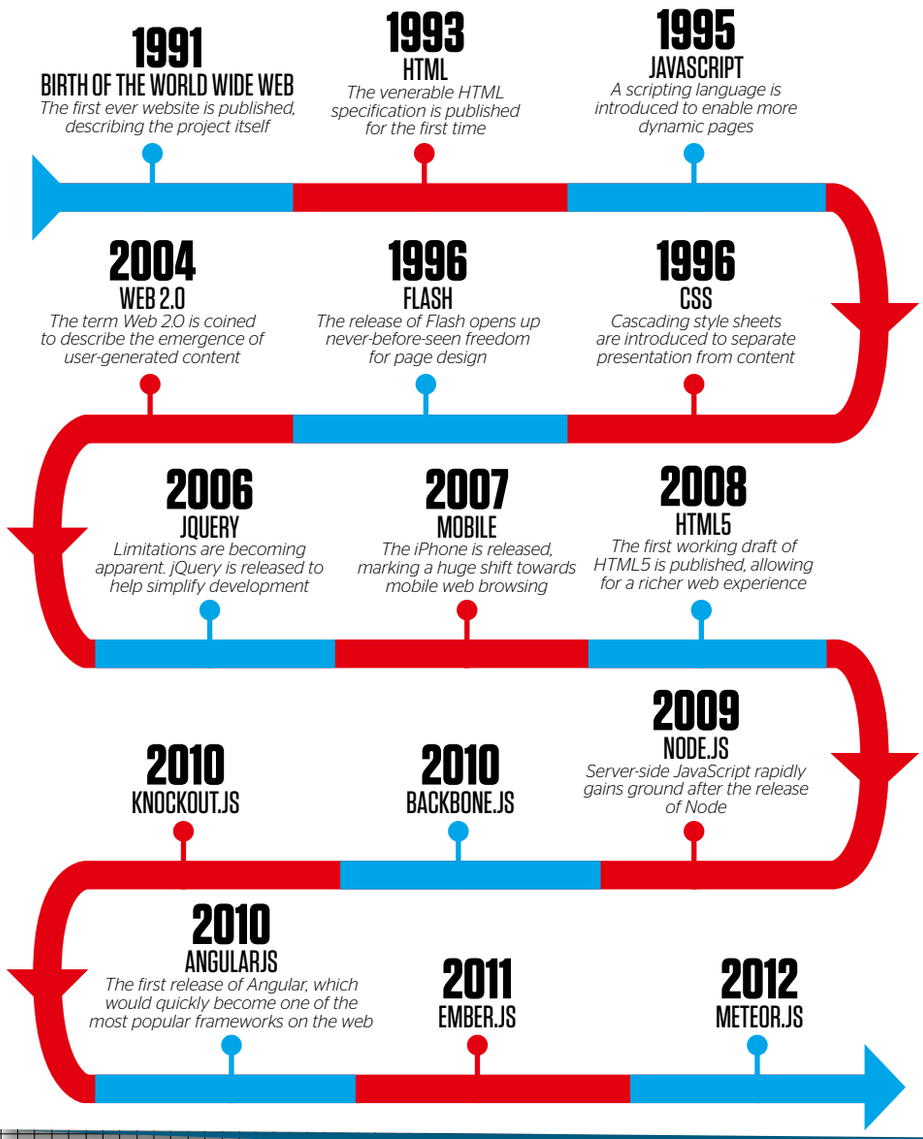
The strongest argument for the use of frameworks is looking at the problems that exist in 'vanilla' web development: convoluted code, lack of modularity and a great deal of time devoted to building basic functionality. Frameworks solve this by offering a layer of abstraction between the code written by a developer and that which is rendered by the browser, providing simplified ways of achieving common requirements. Essentially, you no longer have to worry about implementation details for simple things like manipulating the DOM. Frameworks also generally focus around component-driven development, which can make a huge difference to the maintainability of a page.

For many years, jQuery was the go-to abstraction solution for web developers. However, as app complexity has increased, the need has emerged for more extensive rethinks of how we develop for the web. As a result, there is now a proliferation of front-end frameworks, each with a different opinion on how things should be done. Along with Angular and React, we also have Backbone, Ember, Vue and more. In spite of their differences, ultimately each of these will offer improved code organisation and reuse, while helping to enforce good design patterns.

There are downsides to framework use, however. Each comes with its own learning curve and often an 'opinionated' design, which forces developers to adhere to design patterns that the framework author considers correct. For many the learning curve will be overcome quickly through improved productivity. But perhaps the biggest watch-out when adopting frameworks is how quickly the marketplace is changing. It's possible to learn a framework then find it discontinued or redesigned, as many found with AngularJS. You need to be prepared to keep up with the latest releases, updating code when needed.

A HISTORY OF THE WEB

Amazingly, the world wide web is only 26 years old. It's come a tremendous way, and a look at its history helps understand why we need frameworks to manage the increasing complexity of apps



IF YOU'RE SLIGHTLY BEWILDERED BY THE WIDE RANGE OF WEB FRAMEWORKS, TOOLS AND LIBRARIES, THE GOOD NEWS IS THAT IT ISN'T JUST YOU. THERE'S A HUGE AMOUNT OF CHOICE OUT THERE. RESEARCH AND EXPERIMENTATION WILL HELP YOU FIND ONE THAT SUITS YOUR STYLE THE BEST, AND YOU'LL QUICKLY SEE GREAT IMPROVEMENTS IN PRODUCTIVITY.

Simon Jones
Software engineering director

WHAT'S NEW IN ANGULAR 4?

After a bumpy ride moving from AngularJS to Angular 2, the latest release is less intimidating

Released in March 2017, Angular 4 introduces a relatively small set of changes compared to Angular 2. This will come as particularly good news for AngularJS developers who had difficulty with Angular 2's extensive and fundamental changes. If you're already developing in Angular 2, the modifications you'll need to make range from 'none' to 'very minor'.

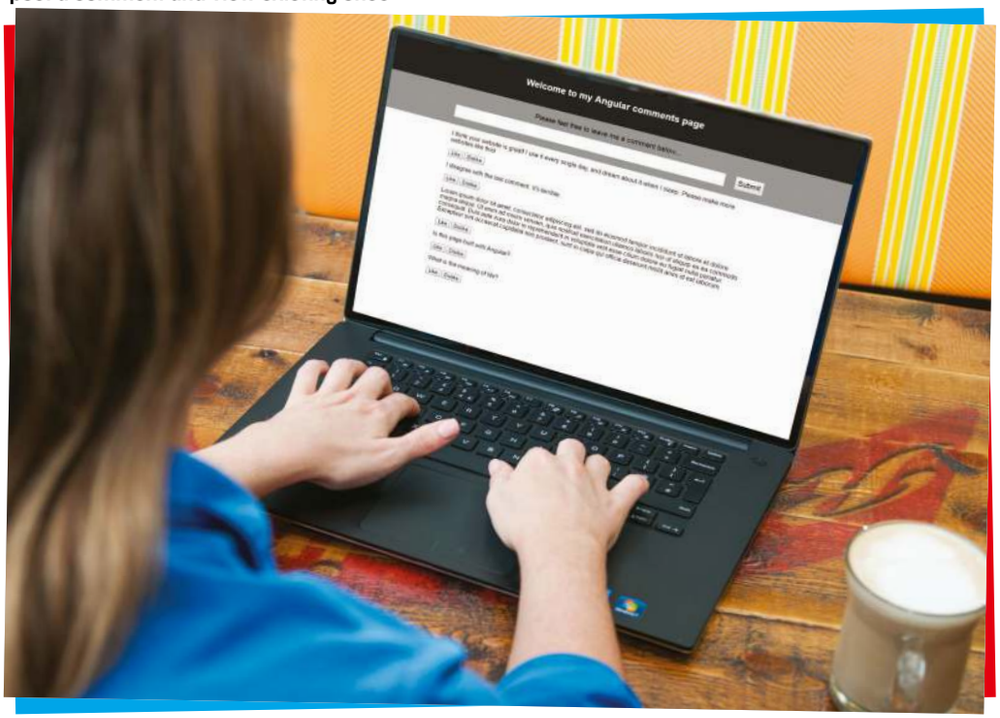
Some of the more notable changes introduced in version 4 include:

- Movement of animations to their own package in @angular/platform-browser/animations
- Replacement of the template tag with ng-template, since template is now an HTML5 tag for web components
- Introduces if/else syntax in templates, which will be very useful for conditional templates
- Numerous minor improvements, such as a titlecase pipe and email validator – plus, of course, bug fixes

On top of these changes, there's also significant optimisation to deliver more compact, faster code. In particular, Ahead-of-Time (AoT) compilation has undergone a major overhaul, which the Angular team have found to yield substantial reductions in the size of distributables. It's also worth mentioning that Angular now utilises semantic versioning, whereby major changes to core libraries increment the leading version number. Complete re-writes should be a thing of the past, and as a result, we should start calling it simply 'Angular', rather than 'Angular 2' or 'Angular 4'. Oh, and the Angular team skipped version 3.

CREATE A COMMENTS PAGE WITH ANGULAR

Let's create the outline of a simple comments page, which will enable us to post a comment and view existing ones



Angular is probably the best-known JavaScript front-end framework, and with good reason. It's designed to enable development of cross-platform, high-performance web applications, which are built from custom components. Component functionality is written in TypeScript, which is a powerful superset of JavaScript. It's typical to run Angular with a Node.js server, which we'll do here.

01. PLAN THE COMPONENTS

In a component-based world, before we start developing it helps to think about how we would break down our page. Within our comments page component (AppComponent), we'll have several other components: a new comment submission box (CommentBoxComponent), and a comment history section (CommentHistoryComponent), which will be a collection of individual comments (CommentComponent).

02. SET UP AN ANGULAR PROJECT

The easiest way to get Angular up and running is using the Angular CLI, a command-line interface that can create projects, files, deployments and so on. To do this, you'll need to install Node.js and npm (**nodejs.org/en** and **npmjs.com**). Once you've done this, in your console or command line, you can use:

```
npm install  g @angular/cli
```

This will install the Angular CLI globally. We can then create a new project using:

```
ng new angular-app
```

This will create a project folder and a set of files to get you started. From the project directory, we can then have the CLI build and launch our app:

```
ng serve --open
```

03. CREATE OUR COMPONENTS

We'll want to create several Angular components in line with our design, which the CLI can do for us:

```
ng generate component comment-box
ng generate component comment-history
ng generate component comment-history/comment
```

04. THE COMMENT COMPONENT

A component in Angular consists of TypeScript that defines its behaviour, with HTML and CSS defining its appearance. You can see each of these has been generated by the CLI for the Comment component. Let's start by modifying its appearance in the HTML file to allow for a text variable and some Like/Dislike buttons:

```
<div class="comment">
  <p>{{text}}</p>
  <button (click)="like()">Like</button>
  <button (click)="dislike()">Dislike</button>
</div>
```

To enable the text field to be populated by the parent component, we need to add an input property to the TypeScript file:

```
export class CommentComponent implements
OnInit {
  @Input() text: string;
      ...
```

This requires us to import Input from @angular/core. Note also the selector statement, which enables us to use this component with the tag <app-comment>.

05. THE COMMENTHISTORY COMPONENT

CommentHistory will be a collection of Comments. The HTML is simple and uses ngFor, which lets us iterate over a collection in our HTML:

```
<div class="comment-history">
  <div *ngFor="let comment of commentsList">
  <app-comment text={{comment}}></
app-comment>
  </div>
</div>
```

Note the text={{comment}} attribute, which passes the comment variable to the input property of each Comment component that we just created. In this component, we'll also hardcode the commentsList array with some historic comments for now (you'd want to read these from a database or JSON really). Finally, let's create an addComment method that can be called to add to this array.

```
addComment(comment) {
this.commentsList.push(comment);
}
```

06. THE COMMENTBOX COMPONENT

CommentBox is a little more involved. Let's look first at the HTML:

```
<div class="input-component">
  {{instruction}}
  <div class="text-entry">
    <form #form="ngForm" (ngSubmi
t)="handleSubmit(form)">
      <input class="text-
field" type="text"
[(ngModel)]="comment"
name="comment"/>
      <input class="text-
submit" type="submit"
value="Submit" />
    </form>
  </div>
</div>
```

We already know that {{instruction}} displays text from a variable. But we also have a form element that calls the handleSubmit function when the Submit button is clicked. We also have an input field with some unusual syntax. [(ngModel)] enables two-way data binding, which enables us to read and modify the text in the field by modifying the comment variable.

07. GENERATE EVENTS

We want our CommentBox to generate an event when a comment is submitted, which the parent component can use to update the history box. In the TypeScript, we'll define a component output:

```
@Output() submission: EventEmitter<string> =
new EventEmitter<string>();
```

This requires us to import Output and EventEmitter from @angular/core. We can then have the handleSubmit function to generate a submission event that includes the comment string:

```
handleSubmit(formData) {
if (formData.value.comment.trim() != "") {
this.submission.next(formData.value.comment);
}
this.comment="";
}
```

We must also define the instruction variable and set up the comment variable as an @Input to enable us to modify it.

08. BUILD THE ROOT COMPONENT

Now that we've created our individual components, it's time to bring them together. The AppComponent class is our 'root' component, which is added to index.html via <app-root>. We'll modify the AppComponent HTML as follows:

```
<div class="app">
  <div class="app-header">
  <h2>
  {{title}}
  </h2>
  </div>
  <app-comment-box (submission)="handleCommen
t($event)"></app-comment-box>
  <app-comment-history></app-comment-history>
</div>
```

TIP

STYLING

We've skipped details of the CSS styling in this tutorial, but you can modify each component's CSS file alongside its HTML to achieve the desired look.

This not only adds our comment box and history, but includes handlers for the submission event we created in the last step. When the comment box generates this event, we call the handleComment function.

09. THE FINISHING TOUCHES

The only thing left to do is create the handleComment function. This will call the addComment method of the CommentHistory component that we already created:

```
handleComment (comment) {
this.commentHistory.addComment(comment);
}
```

However, we must also explicitly allow the parent component access to the child's functions, which is done by adding an @ViewChild statement:

```
@ViewChild(CommentHistoryComponent)
private commentHistory:
CommentHistoryComponent;
```

We can then finish up by defining the title variable, and that's it! The basics of your page should now be working. If you're still running the Node server, you should have seen your changes refresh automatically as you went along.

5 ANGULAR PLUGINS, TOOLS & LIBRARIES

ANGULAR UNIVERSAL

universal.angular.io

Angular Universal allows server-side rendering for Angular apps, which means users will see a server-rendered view of your page on their first visit, greatly improving performance and SEO.

ANGULAR MATERIAL 2

material.angular.io

If you're looking for high-performance UI components designed for the web, mobile and desktop, then Angular Material may be the solution for you. You can also style them to your theme or brand!

NG-BOOTSTRAP

ng-bootstrap.github.io

For years, Bootstrap has been a staple of front-end web development. Now, ng-bootstrap brings the components of Bootstrap 4 to Angular, so you can have the best of both worlds.

SEMANTIC UI

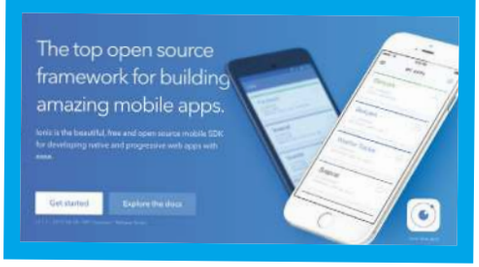
semantic-ui.com

Angular integration is still in development, but this will be exciting when ready. Semantic UI introduces some very interesting concepts to enable UI layouts to be created with natural-language HTML and JavaScript syntax.

IONIC FRAMEWORK

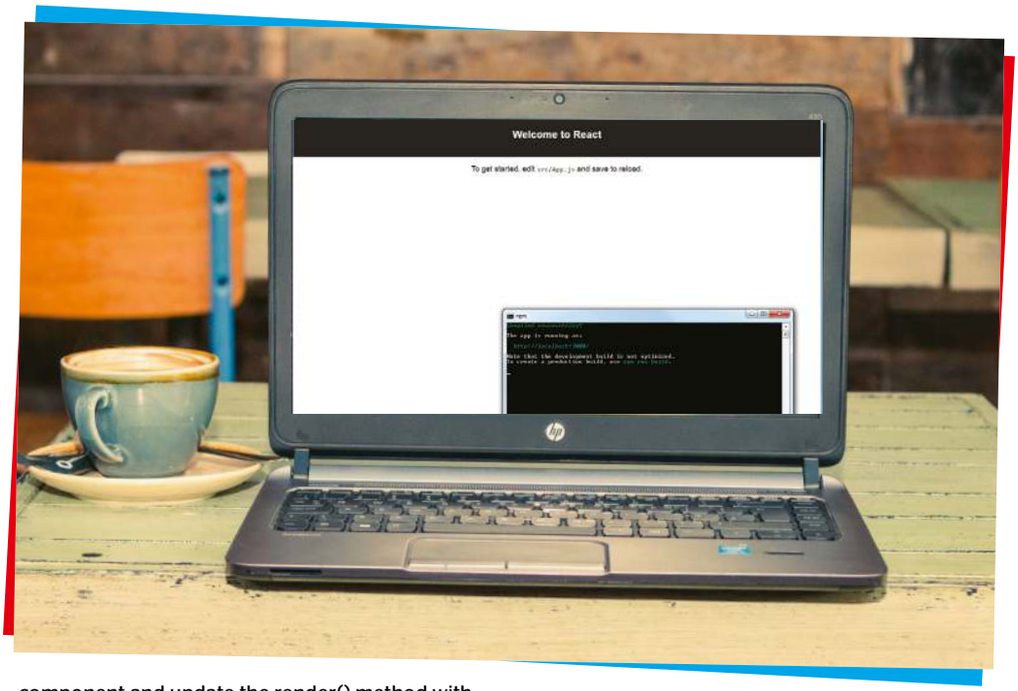
ionicframework.com

Ionic is focused on mobile app development, and can be used to create some truly impressive experiences. It also comes with a wide range of plugins to maximise the user experience on mobile devices.



CREATE A SIMPLE PAGE WITH REACT

Let's create the same comments page we built in Angular, but this time using React



React has seen an incredible surge in popularity since its release, and is focused first and foremost on creating high-performance, component-based user interfaces. The main purpose of React is to bind HTML elements to data, updating the elements when the data changes without reloading the page. Usually, React development is done with JSX syntax, an extension of JavaScript that enables HTML to be embedded directly in the script.

01. PLANNING AND INSTALLATION

Once again, the easiest way to get started is using npm. The create-react-app package will set up a build pipeline for you:

```
npm install  g create-react-app
create-react-app react-app
```

This will install the package and use it to create a react-app folder. Navigate to that folder, and you can serve and open the app:

```
npm start
```

02. REACT'S HANDLING OF COMPONENTS

Like most frameworks, React is focused on component development, and we'll use the same component structure for React that we designed for our Angular implementation. This time, we don't need to create a whole folder structure for our components. Instead, for simplicity, we'll build them all within a single JSX file.

If you open App.js, which was created automatically, you'll notice that what a component displays is controlled by the render() function, which JSX allows to encompass HTML-like syntax.

03. CREATE A COMMENT COMPONENT

Our comment component needs to display some text, plus Like and Dislike buttons. Let's define a component and update the render() method with JSX code to do this:

```
class Comment extends Component {
  render () {
  return (
  <div className="Comment">
  <p>{this.props.text}</p>
  <button onClick={this.like}>Like</button>
   
  <button onClick={this.dislike}>Dislike</
button>
  </div>
  );
  }
}
```

This goes inside the App class, which is already there. Note that React uses className where you might be used to using the class attribute to specify CSS classes. You can create stubs for the Like and Dislike functions for now, which are triggered when the buttons are clicked.

04. PASS DATA BETWEEN COMPONENTS

In the render() function we just created, {this.props.text} displays the value of a property of the component called text. This is accessible as an attribute of the element created by this component. In JSX, this would look like <Comment text="my comment text" />.

Let's use this to create a CommentHistory component that will create a set of Comment elements based on an array input:

```
class CommentHistory extends Component {
  render () {
  var comments=this.props.comments;
  return (
  <div className="Comment-history">
  {
  comments.map(function(comment, index) {
  return <Comment text={comment} />;
  })
  }
  </div>
  );
  }
}
```

The map function calls a specified function (in this case, generating a Comment element) for every element in the comments variable. We use {comment} to pass that specific comment string into the newly created component.

05. CREATE THE COMMENTBOX COMPONENT

Now let's try something a little more complex. The CommentBox component needs a text input and buttons, so we can create a render() function accordingly:

```
render () {
 return (
 <div className="Input-component">
```

SERVER SIDE OPTIONS

Do we have to use Node?

You'll notice that JavaScript frameworks are generally designed for a Node.js server environment. Node is understandably popular with front-end developers for the ability to code JavaScript on the server side. You don't strictly have to use Node, however. Both Angular and React will play just fine with other popular server environments such as Java, PHP or Rails. You may find documentation relatively thin on the ground, however. There's also arguably not a lot of reason to do this, unless you have a pre-existing backend to integrate with. Most web applications will be well-served by something like the MEAN stack (**mean.io**), which integrates MongoDB, Express.js, Angular and Node.js to provide a full-stack development framework.

However, even if you aren't using Node for your web server, you'll likely still need it for your development environment. Front-end development with any modern framework is highly dependent on tools such as npm and webpack, and you'll find almost all guides assume you're running Node. It's also a straightforward way to run a local development server.

FRAMEWORK...?

If we're talking technicality, React might not be considered a framework as such. It's fundamentally a library for front-end development, so other technologies are required for a complete development toolset. However, when developers refer to React, they're often referring to a stack of libraries and tools that together offer a distinct development experience. As such, a common decision for developers is between Angular, React or some other framework.

```
  Please feel free to leave me a comment below...
  <div className="Text-entry">
  <form onSubmit={this.handleSubmit} data-value={this.state.text}>
  <input className="Text-field" onChange={this.handleChange} type="text" value={this.state.text} />
  <input className="Text-submit" type="submit" value="Submit" />
  </form>
  </div>
  </div>
  );
  }
  handleChange(event) {
  this.setState({text: event.target.value});
  }
```

Most of this is self-explanatory, but the interesting parts are the data-value={this.state.text} and value={this.state.text} attributes, and the handleChange function. These enable us to link the state of the component to the input, and store it as a data attribute of the form. We'll use this later.

06. GENERATE A SUBMIT EVENT

Much as with Angular, we need our React component to generate an event when the Submit button is clicked. We've told the form to call handleSubmit when submitted, so let's create that function within the CommentBox class:

```
handleSubmit(event) {
 event.preventDefault();
 if (this.state.text.trim() != "") {
 this.props.onChange(event);
 }
 this.setState({text: ""}); }
```

With this we use the state of the component, which we talked about in the last step, to both read the comment that was entered, and reset it to empty. We also raise an onChange event that the parent component can react to.

07. BIND EVENT HANDLERS

There's a small nuance to the way JavaScript handles the 'this' object, which requires us to do some extra work here. Essentially, 'this' won't work in the event callback unless we explicitly bind it. There are a few ways to do this, but React's recommended approach is to do so in a constructor that is called when the component is created:

```
constructor (props) {
 super(props);
 this.state = {
 text: ""
 }
 this.handleChange = this.handleChange.bind(this);
 this.handleSubmit = this.handleSubmit.bind(this);
 }
```

08. BRING IT TOGETHER

Now our components are built, we can modify the main App component to display them:

```
render() {
 return (
 <div className="App">
 <div className="App-header">
 <h2>Welcome to my React comments page</h2>
 </div>
 <CommentBox onChange={(event) => this.submitComment(event)} />
 <CommentHistory comments={this.state.commentsList} />
 </div>
 );
 }
```

We also need a submitComment function. You'll notice that the CommentHistory component gets its list of comments from the state of the App component, so our submitComment function will update this state based on the onChange event it receives.

```
submitComment(event) {
 this.setState({
 commentsList : this.state.commentsList.concat(event.target.dataset.value)
 });
 }
```

09. THE FINISHING TOUCHES

Now our app is ready. All that's left to do is initialise the commentsList variable in the state using the App component's constructor:

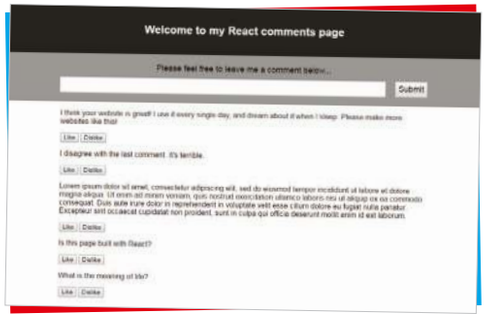
```
constructor(props) {
 super(props);
 this.state = {
 commentsList: ["...", "...", ...]
 }
 }
```

If you're still running Node, you should once again see your page refreshing live as you make changes. From here you can experiment – perhaps try to implement functional Like/Dislike buttons, comment deletion and more.

TIP

ES6

ES6 is the latest JavaScript implementation. Additions such as class syntax and arrow functions help make React code more concise and readable.



5 REACT PLUGINS, TOOLS & LIBRARIES

REACT-BOOTSTRAP
bit.ly/1kPFIFr

Yes, there's a Bootstrap for React too! This provides an opportunity to clean up the Bootstrap code and take advantage of JSX syntax. React-Bootstrap will reach a version 1.0.0 release soon.

REACTCSS
reactcss.com

So with JSX you can inline your HTML in the JavaScript. What if you want to do the same with your CSS? We're sure this won't be for everybody, but take a look and see if it's something you like the look of.

SEMANTIC UI
react.semantic-ui.com/introduction

Semantic UI is also available for React. Once again, it's under development, but much of the functionality is already available for you to experiment with. So why not start now?

REACT NATIVE
facebook.github.io/react-native

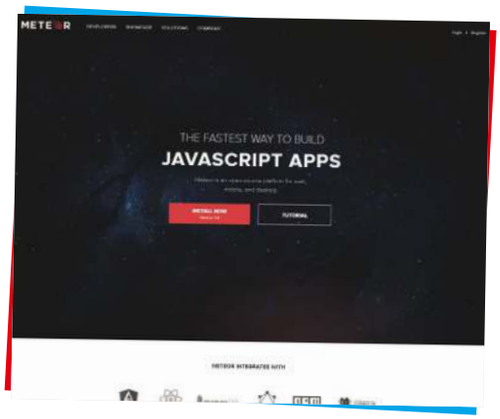
Okay, it isn't quite a library, but React Native is a mobile development framework that allows mobile apps to be built using only JavaScript, with the same design as React.

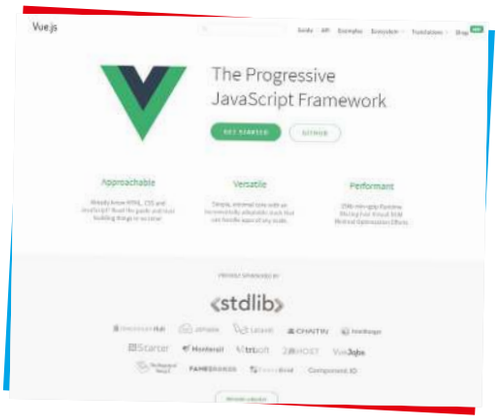
REDUX
redux.js.org

While you don't have to use it with React, it's particularly popular to do so. Redux manages application state for you, and since React allows your UI to respond to state, it makes a compelling pairing.

WHO ELSE TO CONSIDER

Angular and React aren't the only options on the table. So what else might we consider?





EMBER

emberjs.com

"A framework for creating ambitious web applications"

Ember aims to provide a complete solution for client-side development, and takes a strongly 'opinionated' approach in doing so. While it doesn't have the backing of a single major company like Google or Facebook, it's nonetheless used on many popular sites, including LinkedIn, PlayStation Now and Apple Music.

Ember uses the Handlebars templating engine, which provides the ability to embed expressions in the HTML to dynamically change what is displayed. Ember places heavy emphasis on components, using Handlebars to describe their appearance and JavaScript to implement behaviour.

Ember may be a particularly good choice for developers who like their frameworks prescriptive and all-inclusive, and who lack the inclination to piece together libraries from multiple sources. Many find its learning curve to be challenging, and it may be quicker to put together smaller pages in something like React. Ember has, however, found particular popularity with developers using Rails for their backend, perhaps because Rails itself offers a similarly opinionated design philosophy.

METEOR

meteor.com

"The fastest way to build JavaScript apps"

Meteor is a full-stack framework, and like Ember, is highly opinionated. It comes with a front-end rendering system called Blaze, which can be swapped out for Angular or React if you so desire.

Blaze uses a templating system, Spacebars, which is similar to Handlebars but takes advantage of some of Meteor's more powerful features. The templates are very close to HTML, which makes them immediately readable. Many find that this sits in a sweet spot, offering a gentler learning curve than Angular yet a more structured code style than React. Blaze also integrates very well with other JavaScript tools and libraries such as jQuery, and will not have issues with changes they make to the page.

Meteor and Blaze may be particularly good choices for full-stack development, especially where a database is required, since they offer particularly powerful integration between DOM and data sources. It's also become a highly popular choice to combine Meteor with React, and the developers behind Blaze indicate that closer coupling of the two is likely in the future.

VUE

vuejs.org

"The progressive JavaScript framework"

Vue was created by a single developer who had worked for Google on AngularJS. His objective was to build something that delivered its strengths while remaining lightweight and less opinionated. In a short space of time, its activity on GitHub had surpassed other popular frameworks and libraries.

One of the key attractions to Vue for many developers is its simplicity and straightforward learning curve. It's close enough to regular JavaScript that there is no need to learn complex syntax specific to the framework, and it also comes with very strong documentation.

Vue offers strong performance, outperforming both Angular and React in many benchmarks. This is in part because it implements a similar virtual DOM to React (and in fact Ember), enabling only the components that need changing to refresh.

It's apparent that Vue comes with a very fresh approach to the problems facing developers, and has learned from the journeys of both Angular and React. Overall, it may be a good choice for those new to developing with frameworks, who may not benefit from all the features of larger frameworks.

WHAT TO LOOK OUT FOR IN 2017

SVELTE

svelte.technology

Svelte is focused on being super lightweight, enabling faster loading and faster running apps. Could this be the beginning of a shift from heavyweight frameworks towards performance-optimised approaches?

AURELIA

aurelia.io

Developed by an ex-Angular team member, one of Aurelia's primary focuses is modularity. There are evident similarities with Angular, but the syntax is simpler and closer to standard JavaScript.

CYCLE.JS

cycle.js.org

Cycle has been around for a couple of years, offering a functional programming framework. There's growing interest in functional web development these days, so we could see a rise in popularity this year.

PREACT

preactjs.com

Preact positions itself as a small, fast alternative to React. And weighing only 3kB, it's hard to dispute this. With growing concern about large framework footprints, could it gain ground this year?

POLYMER

polymer-project.org

We're banking on web components really taking off this year, and Polymer gives some insight into how developers may solve for componentisation in the future without relying on frameworks at all.

MAKING THE RIGHT CHOICE

There are a lot of options out there, but how do we pick the right one?

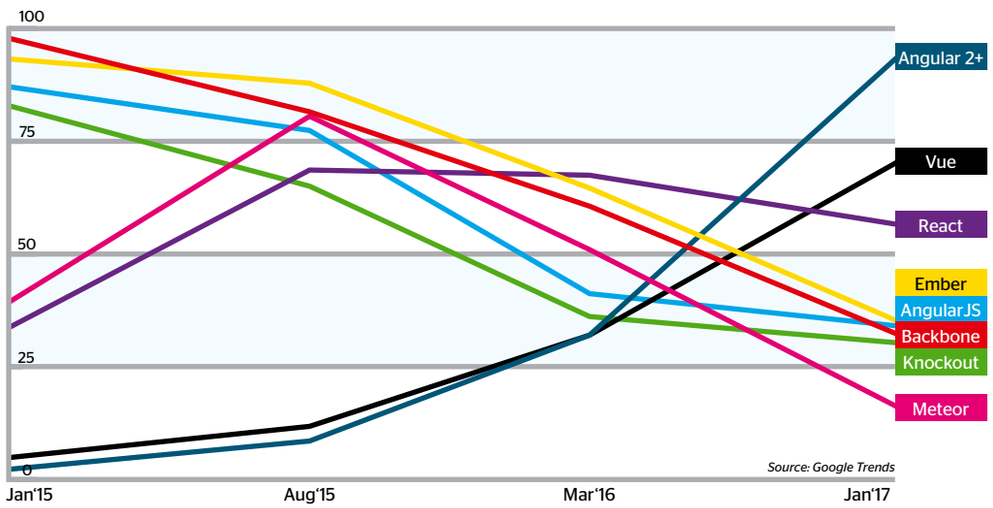
You might be able to get an initial feel for these frameworks by looking at the code samples and deciding for yourself whether their design philosophy is useful or absurd. However, it can be difficult to see all the benefits and pitfalls until you start spending time working with them.

Looking back at the pages we built in this article, the React code is concise and very readable for this simple scenario, thanks to JSX. Many feel that it has a simpler learning curve as a result. Angular, meanwhile, enforces more structure and design discipline, which can be powerful for large-scale, complex applications. The distinction is personal preference, and these are the two best-known frameworks for a reason. Vue, however, is a relative newcomer that has rapidly gained a significant following and is well regarded by its users.

A significant factor in considering a framework should also be longevity. React, Vue and Angular are all seeing an active increase in usage, indicating that they're likely to be around and improving for some time to come. If you pick one of these three, you're likely to be in safe hands. Conversely, older frameworks seeing a drop in interest, such as AngularJS, may be coming towards the end of their life.

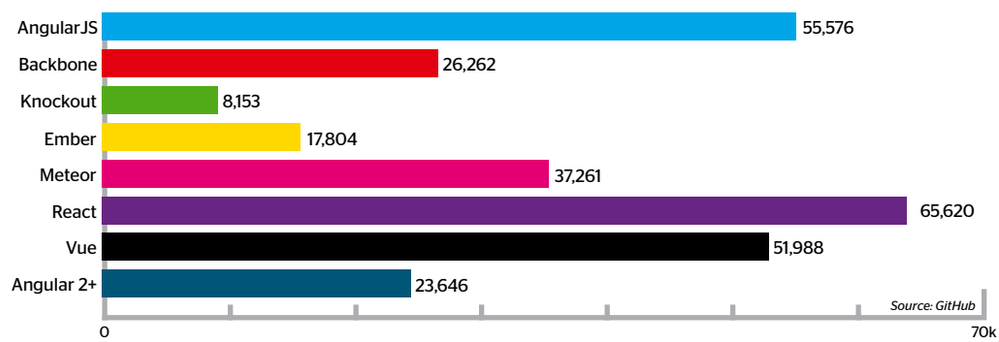
POPULARITY

Angular and Vue have been the popular searches over the last two years



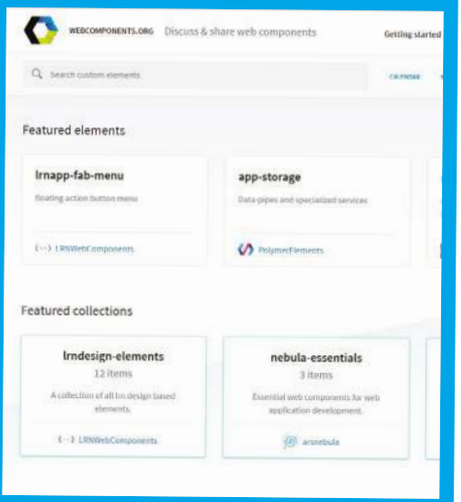
GITHUB STARS

Stars are effectively a like and demonstrate the popularity of the repository



WHAT DOES THE FUTURE HOLD?

Frameworks are certainly here to stay, but expect some changes in how we work with them



The challenges in web development that frameworks have emerged to solve aren't going away. Web apps continue to grow in complexity, especially with the proliferation of progressive web apps. Meanwhile, HTML and JavaScript continue to evolve but retain many of the same fundamental limitations that have existed for decades. In some places, there's an undertone of frustration emerging within the web community. Some argue that things have become too divergent, and that many frameworks introduce complexity unnecessary for the majority of users. It's likely that we'll start to see more done without frameworks, which may start to reverse this. Modularity and templating, which are some of the leading benefits of frameworks, are now part of the HTML specification. When the web components specification is implemented across all browsers, developers will find less reason to move to the likes of Angular.

Where frameworks remain in use, we're likely to see an increased focus on lightweight footprints and performance optimisation. There's growing realisation that these are important considerations, and many newer frameworks highlight them as key selling points relative to the heavyweights. We may also see some consolidation of frameworks (or at least convergence of styles and patterns), to bring more standardisation and help reduce the risk that particular frameworks fall out of favour over time.

GETTING STARTED WITH

Vue.js

MVVM frameworks can be found in abundance. Sadly, many – if not most – burden developers with a variety of custom syntaxes and DSLs. Vue.js takes a refreshingly approach...

WHAT IS VUE.JS AND WHY USE IT?

Web development will forever suffer from a fundamental conflict of interest because the web was once intended as a medium for sharing documents. Due to that, HTML/CSS/JavaScript are always at odds with the rest of the system

Libraries such as Angular add an additional abstraction layer on top, which causes overhead and binds developers to their custom syntax. Broadly speaking, one can claim Vue.js to take inspiration from classics like Knockout.js.

The simplest example to show off – proudly pilfered from the documentation – realises a basic data binding scenario.

```
<div id="app">
  {{ message }}
</div>
var app = new Vue({
  el: '#app',
  data: {
    message: 'Hello Vue!'
  }
})
```

Experienced HTML hands will soon realise that the code is fully compliant to ECMAScript 5 – this is, incidentally, the reason why Internet Explorer 8 is not among the supported browsers.

Running the example above in a page that includes Vue.js – the various deployment assistants allow you to reduce the additional file size to less than 20KB – reveals a website whose content is replaced with the value found in the message variable. This, however, it but a small subset of the total feature complement: the framework can also provide validation and other services to simplify the creation of attractive user interfaces.

COMPONENT CREATION

Considering Vue.js to be little more than a glorified data binding framework, however, is a grave mistake. The extremely flexible underlying architecture allows the framework to take in a variety of plugins that expand its capabilities beyond the already impressive ones found in the default distribution.

One particularly impressive feature is Vue.js's capability to create components: they can then be deployed as 'custom user interface elements'. The framework expands HTML5's component specification with a variety of comfort features – for example, the visibility of components can be limited.

PLAYS NICE WITH OTHERS

Finally, allow us to remark that Vue.js can be integrated along almost any other web framework known to man: if a list of 'un-opinionated' frameworks were to be created, Vue would definitely find itself at the top of the list.

HOW TO INSTALL

Vue.js can join your project in a variety of fashions. For sure, the simplest is the local inclusion of a JS file. A more involved process switches that for a JS file hosted on a CDN – in principle, the two approaches are much the same.

Recent advancements in the world of JavaScript development have led to the emergence of various 'package managers'. Vue.js, of course, integrates into NPM – however, it also comes with a CLI of its own.

WHICH BUILD?

MAKING SURE YOU USE THE RIGHT EDITION OF VUE.JS IS AN OBVIOUS FIRST STEP

Vue.js is made available in various editions: the most important distinction is between release and debug builds. The first ones omit error checking and protection logic for higher speed and lower bandwidth demands, while the latter are better suited for the actual development process. Distinction number two is concerned with the compiler feature.

The presence of this rather large and unwieldy module is required whenever strings are to be parsed at runtime – a classic would be the following snippet populating the template attribute via a string of markup.

```
new Vue({
  template: `<div>{{ hi }}</div>`
})
```

VUE KEY PROPERTIES AND COMPONENTS

THE INS AND OUTS OF WHAT MAKES THE VUE.JS FRAMEWORK TICK

CONTROL PROPERTIES

In some cases, a property of a control must be controlled from the Vue instance. A good example for that would be the changing of the visibility of the activeness of a control: in a to-do program, checkboxes must be checked and unchecked as the state of task fields changes.

A simple way to solve this is the v-bind attribute, which allows you to specify the target attribute comfortably:

```
<button
v-bind:disabled="someDynamicCondition"
>Button</button>
```

DIRECT HTML INJECTION WITH V-HTML

If the string generated in the back-end contains HTML, using a classic text binding will not go far. Instead, use v-html to inject code directly – be aware that this facility cannot be used to create new templates.

```
<div v-html="rawHtml"></div>
//code-ends
```

DO IT ONCE

By default, Vue.js will attempt to update the results of bindings as often as possible. Should you want to run a binding only once, assign it the v-once attribute:

```
<span v-once>This will never change: {{
msg }}</span>
```

However, be careful that v-once applies to all child bindings: if half of your web app stops working after you deploy it, finding the culprit should not be hard.

DO IT CONDITIONALLY

Vue bindings can also provide logic of their own. A nice example for this would be the v-if binding that affects visibility in relation to the state of a back-end object.

```
<p v-if="seen">Now you see me</p>
```

Another nice trick involves deploying JavaScript directly inside a binding – a good example would be this snippet, which performs a chain of string operations.

```
{{ message.split('').reverse().join('') }}
```

Please keep in mind that the JavaScript terms provided as binding targets must be evaluatable: while, if and co do not qualify, while the ternary operator does.

MODEL-A-GOGO!

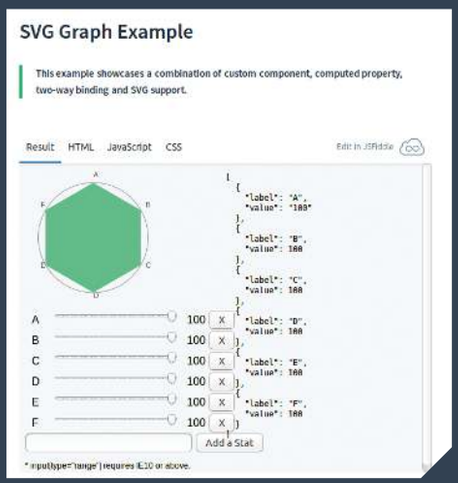
Simple bindings are limited in the support of data flow: if you want to realise advanced interactions, use models as binding targets. Further information on this topic can be found at **https://vuejs.org/v2/guide/forms.html**.

Finally, consider yourself made aware of the surprisingly well-written API reference found at **https://vuejs.org/v2/api**. It provides a detailed overview of the various elements of the framework, so it's well worth a read.

> **"If the string generated in the back-end contains HTML, using a classic text binding will not go far. Instead, use v-html to inject code directly"**

DEMOS AND EXAMPLES

A CLOSE LOOK AT VUE.JS BEING USED ON THE WEB



BIND TO SVG

vuejs.org/v2/examples/svg.html

Vue.js is one of the few data binding frameworks that does not limit itself to classic DOM elements. This is best demonstrated in this little example, which creates a diagram made up of an SVG polyline that has its control points connected to data stored in the back-end.

TODOMVC A LA VUE

vuejs.org/v2/examples/todomvc.html

A somewhat simplistic task-managing application specification called TodoMVC can be considered a benchmark for the performance of various frameworks – while its practical value is limited, the results obtained are interesting. Vue.js's preferred example takes up but 120 lines: a value that does not pale in comparison to its competitors.

GRIDDIE-GO-ROUND!

vuejs.org/v2/examples/grid-component.html

The above-mentioned component system can be used to create well-designed widgets. One of the most impressive ones involves the spawning of a simple Excel-like grid, whose design is modified via CSS. Be aware that the program is not interactive – adding click-handling logic would be a job for someone seeking to learn more about Vue.

UP CLOSE WITH VUE.JS CODE

The Vue.js component framework is really powerful. This example shows a small application...

DECLARE WORKSPACE AND TAG!
The HTML component specification lets developers specify custom tags, which – by and large – behave like in-built tags. In our case, the component will go by the name futurecomponent, and bears one parameter called message. This example provides two instances of futurecomponent with different values of message. When run, the two span elements will be shown above one another with their different textual contents.

USE SOME PROPS TO EXPOSE COMPONENTS
Vue treats the template string just as any other binding source: most of the bindings discussed in this article can also be used inside of the template.

Parameters that are to be populated from the tag invoking the component must be exposed via the props array: it can contain one or more tag names, which are then extracted from the DOM using a parser.

POPULATE THE COMPONENT
Creating templated objects must be announced to Vue outside of the declaration of an actual Vue instance: the component is provided with a string naming the component along with a JSON object containing one or more parameters which modify the behaviour of the element at large. In addition to that, the template string is required – it contains the HTML, plus any markup, which is to be shown inside of the declared tags.

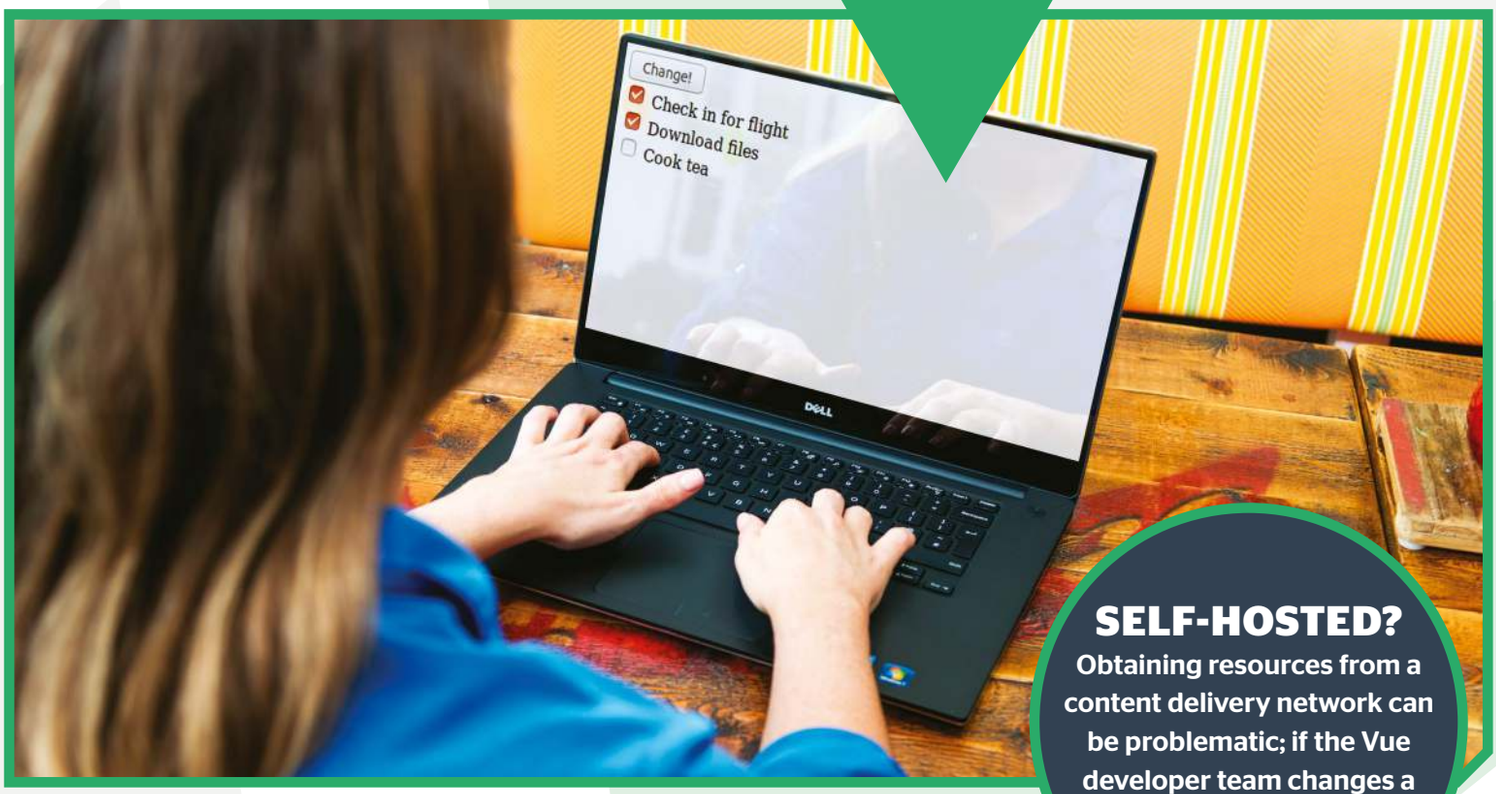
```
<html>
<body>
<div id="workSpace">
        <futurecomponent message="Alpha!"></futurecomponent><br>
        <futurecomponent message="Beta!"></futurecomponent><br>
</div>

<script src="vue.js"></script>
<script>
        Vue.component('futurecomponent', {
          template: '<span>{{ message }}</span>',
            props: ['message']
        })


        new Vue({
          el: '#workSpace'
        })

</script>
</body>
</html>
```

BEWARE OF THE DATA ATTRIBUTE
When using Vue.js's component system, the data parameter must always take the form of a function that returns a new instance for every element. This is important, as the function gets invoked once for each component instance: if a new instance of the backing store is not returned for every element, the widgets will share state and emit all kinds of odd program behaviour.

NORMAL INVOCATION REQUIRED
Keep in mind that the component framework of Vue.js is not working outside of designated working areas. Due to that, the last act in the life of our little example involves the creation of an empty Vue instance.

Its sole parameter contains a reference to the parent element that is to be targeted – in our app example, it is but a div containing the template invocations.

REDUCES REDUNDANCY!
When applied correctly, using components provides a significant reduction in code complexity: if you find yourself cutting and pasting large blocks of mark-up, keep this facility in mind!



SELF-HOSTED?
Obtaining resources from a content delivery network can be problematic; if the Vue developer team changes a few APIs, your application might break.

RUN THE VUE!

Getting started with Vue can be a daunting process. Let us take it step by step

The following steps guide you through the creation of a simple program which demonstrates some of the key aspects of the Vue.js framework!

1 LOAD VUE AND VUEX Using a framework requires it to be loaded. In addition to the basic Vue.js environment found in vue.js, we also load vuex.js. This file is part of an extension framework found at **https://vuex.vuejs.org/en/intro.html** – deploy it to make state management easier. Vuex.js is unproblematic – just be careful to make sure it gets included after the main Vue framework.

```
<html>
<head>
      <title>VueTodo</title>
</head>
<body>
      <script src="vue.js"></script>
      <script src="vuex.js"></script>
```

2 DECLARE THE WORK AREA Vue.js instances do their work inside a tag containing the various elements along with the directives intended to populate them. In the case of our to-do list, we furthermore create a template object which is bound to an element in the back-end via the v-for attribute.

```
<div id="workSpace">
      <template v-for="item in taskfield">
      <input type="checkbox" id="checkbox"
v-model="item.done"></input>
      <span>{{ item.text }}</span><br>
      </template>
</div>
```

3 CREATE THE VUE INSTANCE Our template binds itself to a field named taskfield, which is expected to contain one or more subitems that have two properties each. This is best accomplished by the code snippet here – using JSON syntax to create objects inline might not be a best practice, but makes life easier for small test programs.

```
window.app = new Vue({
      el: '#workSpace',
      data: {
      taskfield: [
{ text: 'Check in for flight', done: true
},
{ text: 'Download files', done: true },
{ text: 'Cook tea', done: false }
       ]
      }
});
```

4 PLACE THE CODE CORRECTLY Vue instances can be created only when the work area DOM elements have raised successfully. While this can also be ensured by using documentReady or jQuery, a simple way involves arranging the individual elements as shown in the skeletal outline accompanying this step.

```
<body>
<div id="workSpace">
      . . .
</div>
<script src="vue.js"></script>
<script src="vuex.js"></script>
<script>
window.app = new Vue({
el: '#workSpace',
data: {
```

5 RUN THE PROGRAM With this out of the way, open the file in a supported browser of choice. The three to-do items declared will show up. As an added challenge, wire up a button with the code accompanying this step – due to the reactivity, pushing an element into the taskfield will lead to the updating of the rendered view.

```
function runner(){
var an0=new Object();
an0.text = "Undone item";
an0.done = false;
app.taskfield.push(an0);
}
```

CREATE THE IMPOSSIBLE

www.photoshopcreative.co.uk

EVERY ISSUE • DOWNLOAD FREE RESOURCES

Photoshop creative

Available from all good newsagents and supermarkets

ON SALE NOW

• Striking imagery • Step-by-step guides • Essential tutorials

BUY YOUR ISSUE TODAY

Print edition available at www.myfavouritemagazines.co.uk

Digital edition available for iOS and Android

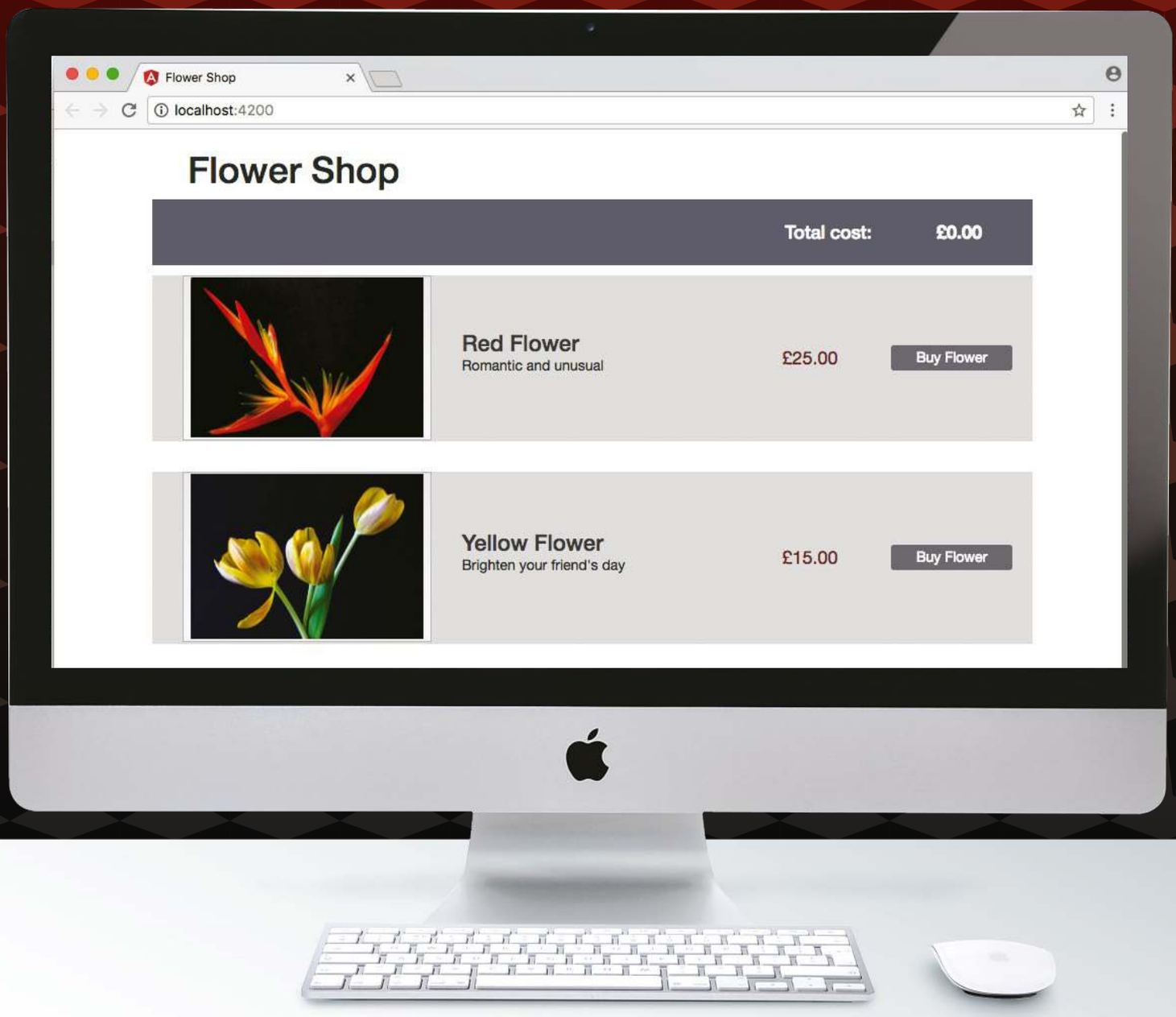
Available on the following platforms

Print Tablet Phone Desktop

facebook.com/PhotoshopCreative twitter.com/PshopCreative

What's new in Angular 4?

There's been some fundamental changes to how Angular works. See how they can benefit your application!



DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/webdesigner

With web frameworks in active development releasing increasingly frequent updates, it's important for any developer or team to be aware of the latest features and deprecations. This is the first major version release since the adoption of semantic versioning by the Angular team. So don't worry, you didn't miss 3.0! There were two major updates to be released and these were both rolled up to version 4.0.

The Angular team are calling it an 'invisible makeover', because most of the changes are in the background rather than with core coding functionality. An important change is the move to Ahead-of-Time compilation as standard, which has the potential to drastically improve performance when used correctly. TypeScript 2.1+ is also now supported, which gives us access to all the new features of ES2015.

That's not to say there aren't also any useful changes to the basics; some useful new template-binding syntax enables us to simplify our code by adding an 'else' option to ngIf and the ability to assign local variables within an ngFor. In this tutorial we'll show you how to utilise some of these new features as well as demonstrate some of the powerful performance-enhancing, under-the-hood changes.

1. Node and npm

Starting from scratch, we can use the Angular CLI to build our Hello World app in the latest version. If you want to update an existing app, skip to step 4. If you haven't already, get Node, which comes with npm and can be downloaded from **nodejs.org**. If you've got Node already, check you've got at least Node 6.9.x and npm 3.x.x from the command line.

```
$ node -v
v6.10.2
$ npm -v
3.10.10
```

2. New project setup

Now we have a package manager we can use it to install Angular and the Angular CLI (command-line interface). You can find the latest CLI documentation at **angular.io/docs/ts/latest/cli-quickstart.html**. Among other things, it gives you the ability to easily generate new projects and components.

```
$ npm install -g @angular/cli
$ ng new my-first-angular4-app # Angular CLI commands use 'ng'
$ ng serve -open # This command will bootstrap your app and launch it in the browser
```

3. Check versioning

If you've not seen an Angular project before, take time to familiarise yourself with the file structure generated by the CLI. You can get information on each of them in the Project File Review section of the CLI Quickstart page on **Angular.io**. For our new project, our package.json should list version 4.0.0 Angular packages. If so, skip to step 5.

```
"dependencies": {
"@angular/common": "^4.0.0",
"@angular/compiler": "^4.0.0",
"@angular/core": "^4.0.0",
"@angular/forms": "^4.0.0",
"@angular/http": "^4.0.0",
"@angular/platform-browser": "^4.0.0",
"@angular/platform-browser-dynamic": "^4.0.0",
"@angular/router": "^4.0.0" [...]}
```

4. Upgrade to Angular 4

If you have an existing Angular app with 2.x versions listed in your package.json, it's really easy to update to version 4 for most cases. We just need to install and update the relevant packages from the command line.

Mac:

```
$ npm install @angular/{common,compiler,compiler-cli,core,forms,http,platform-browser,platform-browser-dynamic,platform-server,router,animations}@latest typescript@latest --save
```

Windows:

```
> npm install @angular/common@latest @angular/compiler@latest @angular/compiler-cli@latest @angular/core@latest @angular/forms@latest @angular/http@latest @angular/platform-browser@latest @angular/platform-browser-dynamic@latest @angular/platform-server@latest @angular/router@latest @angular/animations@latest typescript@latest --save
```

5. Download tutorial app

From now on we'll be working with an Angular 2 example app that you can download from the tutorial file share. Note that this app has been created purely for illustrative purposes. Once you've downloaded the app Flower Shop into your root directory, install dependencies and launch in the browser.

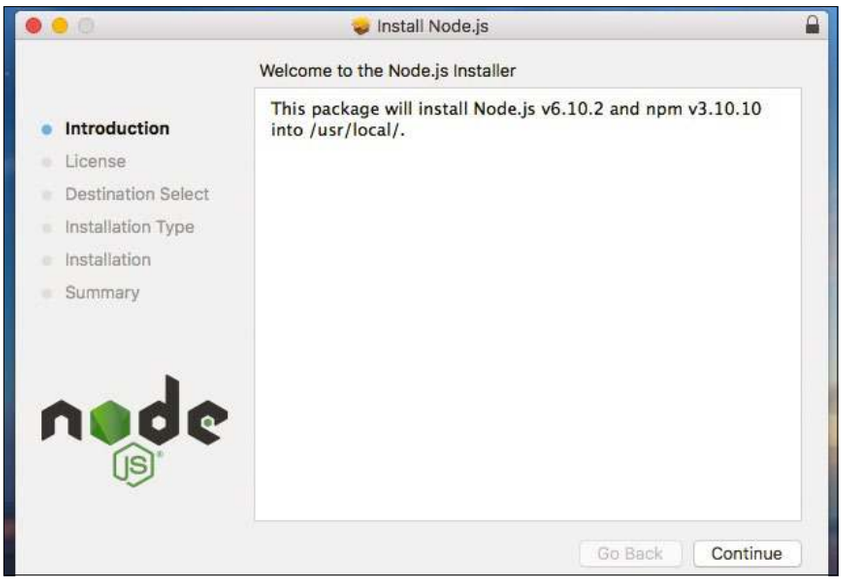
```
$ npm install
$ ng serve --open
```

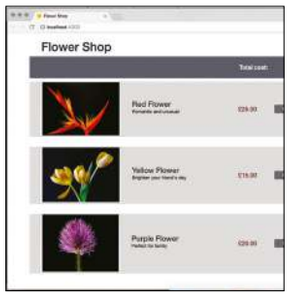
6. Upgrade Flower Shop to 4.0

Execute step 6 within the flower-shop directory. For our Flower Shop app, we have some UNMET PEER DEPENDENCY errors after upgrading.

Angular vs AngularJS

Angular 1.x is now known as AngularJS, and Angular 2+ is simply referred to as Angular. Be wary as some third-party tutorials and libraries may not have updated their use of these terms.





Left
Download and install Node.js by following the instructions in the wizard. Node comes pre-packaged with npm (node package manager)

Top left
Hello World app generated simply by using the Angular CLI command 'ng new test-app' and launched in the browser using 'ng serve --open'

Top right
A tutorial app created for illustrative purposes, launched in the browser using 'ng serve --open'

7. Unmet Peer Dependencies

Peer dependencies are for managing projects that have packages relying on different versions of the same dependencies. You have to add these manually to your package.json file. The peer dependencies that we need for Flower Shop include older versions of @angular/{core,http}, rxjs and zone.js. Review the errors in the terminal output and add each missing dependency to package.json.

```
"peerDependencies": {
  "@angular/core": ">= 2.0.0",
  "@angular/http": ">= 2.0.0",
  "rxjs": "^5.0.0",
  "zone.js": "0.7.8"
  },
```

package.json

For ongoing Angular projects, your package.json will need monitoring. If you update any part of your app, subsequent errors are likely to be solved by checking your dependency versions. The best place to check this is in terminal errors on npm install.

8. Watch out for custom component name changes

Our Flower Shop app is making use of an open-source custom component called ng2-modal. However, the author has since updated their naming convention to ngx-modal. We need to modify our dependency name and ng-module imports accordingly.

```
package.json:
"dependencies": {
  [...]
  "ngx-modal": "0.0.25",
     [...]
  },
app/app.module.ts:
import { ModalModule } from 'ngx-modal';
```

9. Check versions

Our custom modal component also needs updating to a more recent version, so be sure to update it with npm.

```
$ npm install ngx-modal@latest --save
```

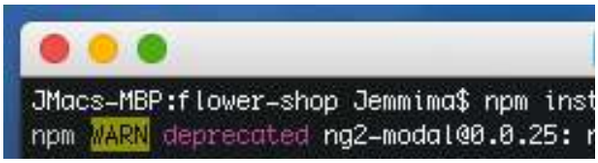
10. Check your work

Now that we've patched up our dependencies, it's a good idea to clear node_modules and build it again from our package.json. If all goes well you should have a clean build! If you don't, something may still be missing in your dependency versions. Take another look.

```
$ rm -rf node_modules/
$ npm cache clean
$ npm install
$ ng serve --open
```

11. View engine

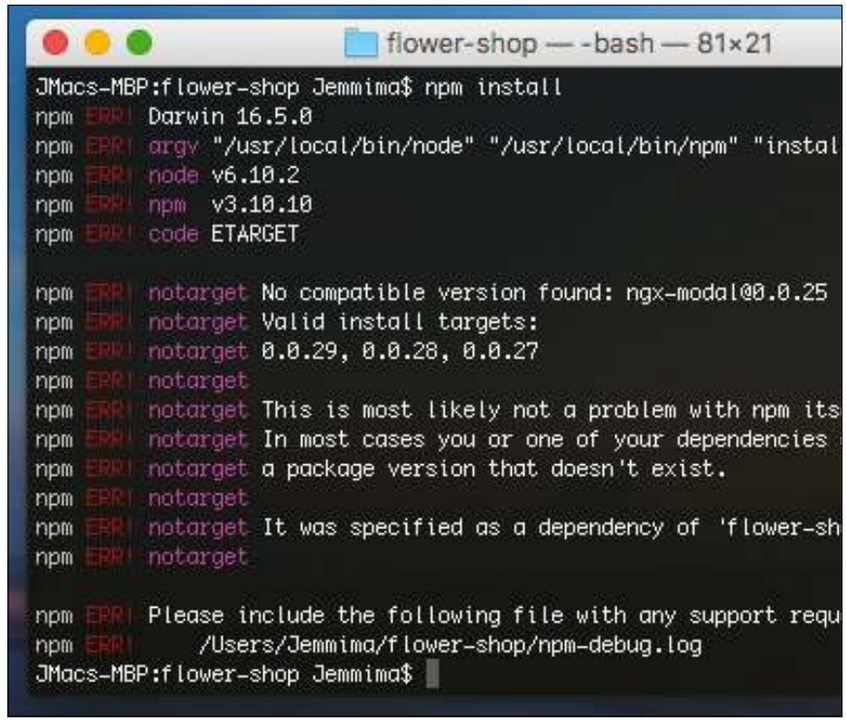
Let's see what new features we can use in 4.0. The new view engine compiles to a much smaller bundle, meaning

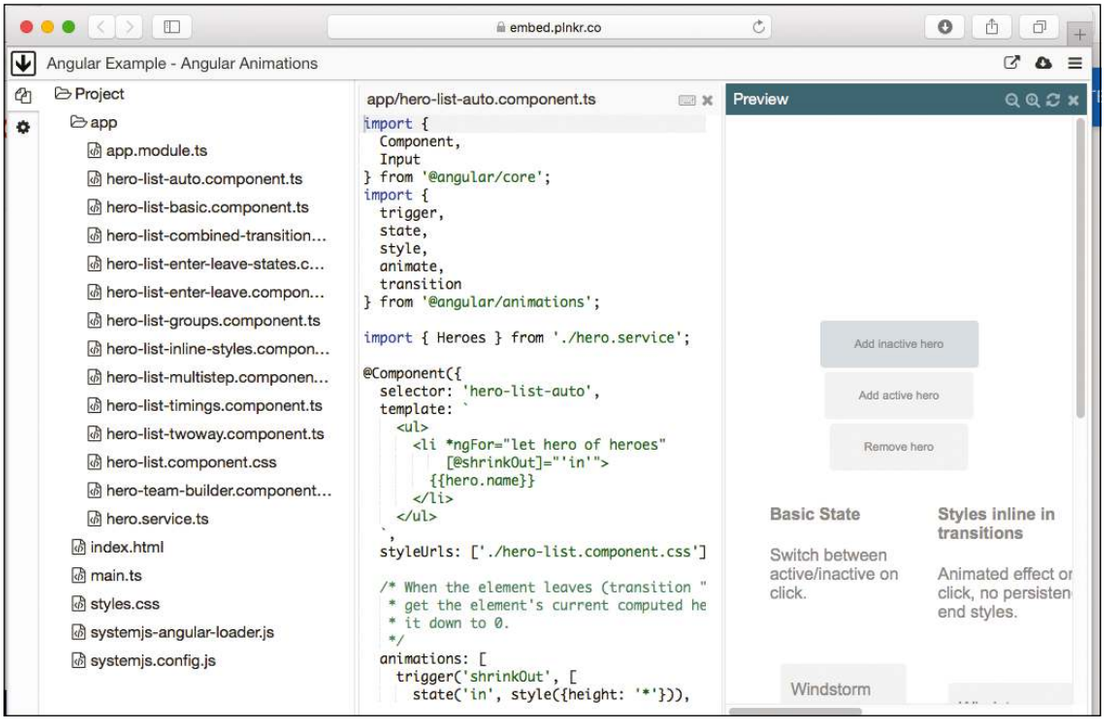


Top
Our custom modal module needs renaming to match new versions of angular naming conventions

Above
Our build should succeed now that we've corrected our dependency version discrepancies

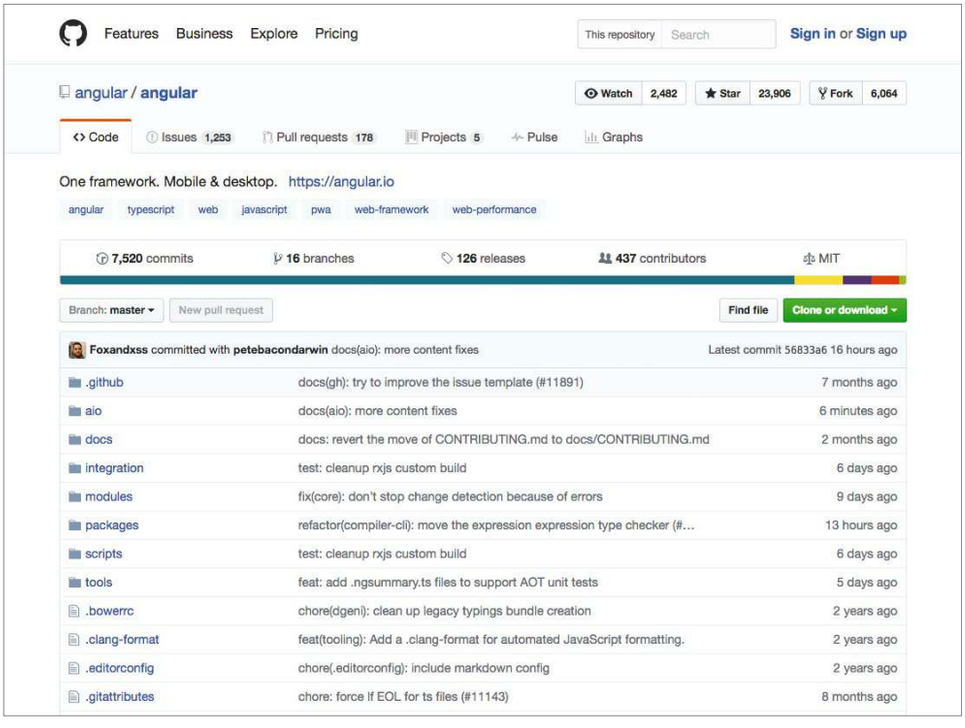
Right
Make sure all third-party and custom-dependency versions are up to date! Use @latest or one of the stated 'valid install target' version numbers





Animations

One of the changes in 4.0 is to strip out the animations module from @angular/core. It's large and therefore an unwelcome bulk in your build unless you're using it. It's easy to add back in if you do need it though, just import it like any other Angular module. Import the new BrowserAnimationsModule from @angular/platform-browser/animations into your root ngModule. The best place to learn how to use animations in Angular, and how to select the right one for your use case, is to use the live example feature at **bit.ly/2q2shvj**. A powerful feature of Angular's animations module is its slick handling of state transitions. It's certainly worth experimenting with – try adding an animation to our Flower Shop app!



extra performance benefits from using AoT compilation with the Angular compiler, ngc. This means that compilation happens at build time, once, rather than multiple times in the browser at runtime. Let's give it a try with our app.

```
$ npm install @angular/compiler-cli @angular/
platform-server --save
```

12. Create AoT config file

We need to create a new config file to tell our ngc compiler what's what. Create a new file in the project root and name it 'tsconfig-aot.json'. Enter the following configuration:

```
{
  "compilerOptions": {
    "target": "es5",
    "module": "es2015",
    "moduleResolution": "node",
    "sourceMap": true,
    "emitDecoratorMetadata": true,
    "experimentalDecorators": true,
    "lib": ["es2015", "dom"],
    "noImplicitAny": true,
    "suppressImplicitAnyIndexErrors": true
    },
    "files": [
    "src/app/app.module.ts",
    "src/main.ts"
    ],
  "angularCompilerOptions": {
    "genDir": "aot",
    "skipMetadataEmit" : true
  }
}
```

13. Initiate AoT compiler

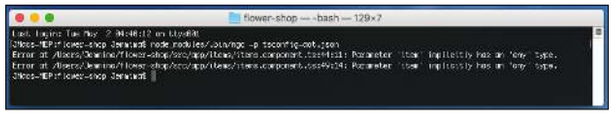
We can now run our ngc compiler, and there's an intentional bug in our code so we can see it working.

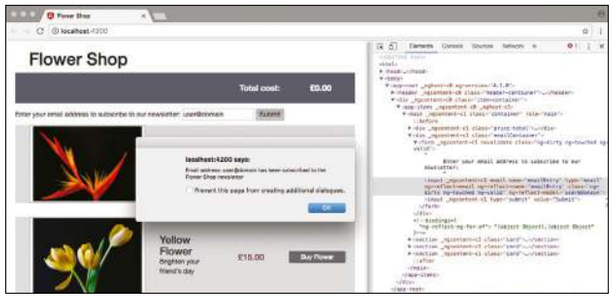
```
$ node_modules/.bin/ngc -p tsconfig-aot.json
```

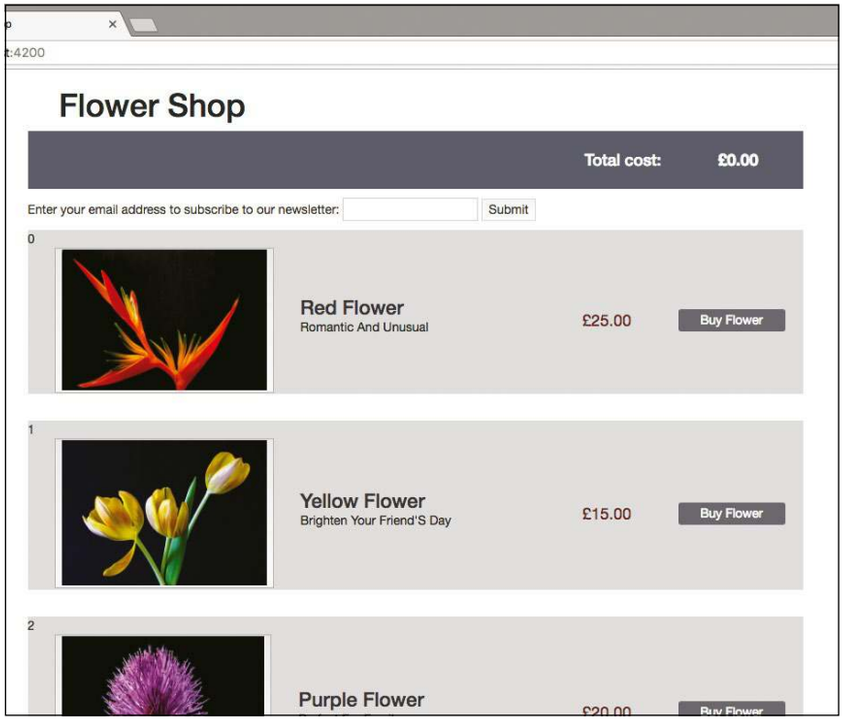
14. Fix the bug

In our items.component.ts file, we've forgotten to assert the type of our 'item' parameter. This doesn't show up as an error in the editor but might cause problems at runtime. Ngc enables us to catch these issues early. Correct buyItem and cancelItem.

```
buyItem(item: Item) {
item.inBasket = true;
```







Top
Using the AoT Angular compiler we can see errors before we ship our app to the browser. In this example we've forgotten to assert type for our 'item' parameter

Above
Our email validator easily recognises only valid email addresses and will only enable the Submit button when one is entered into the form

Right
Titlecase pipe applied to flower description text forces the first letter of each word to be capitalised. Unfortunately this has also been applied to the possessive 's' on Friend'S!

```
  }
  cancelItem(item: Item) {
  item.inBasket = false;
  }
```

15. Email validator

Another handy new feature in 4.0 is the new email validator, which makes it much easier to validate an ngForm input type of 'email'. Maybe our Flower Shop needs a newsletter. First, import the FormsModule to ngModule.

```
import { FormsModule } from '@angular/forms';
@NgModule({
  imports: [
  [...]
  FormsModule
  ],
```

16. Form template

Next, add an ngForm field in the item.component .html template.

```
  <div class="emailContainer">
  <form #form="ngForm"
(submit)="handleSubmit(form)">
  Enter your email address to subscribe to
our newsletter:
  <input type="email" [(ngModel)]="emailEntry"
name="emailEntry" email/>
  <input [disabled]="!form.valid"
type="submit" value="Submit" />
  </form>
  </div>
```

17. Handle @Input

Include handling for the email address input form field to item.component.ts.

```
@Input() emailEntry: string;
  handleSubmit(form: any)
  {
  alert("Email address: "+form.value.
emailEntry+" has been subscribed to the
Flower Shop newsletter");
  }
```

18. Test in the browser

We should find that we can only use the Submit button when we've entered a string that meets the basic criteria for an email address. Well that was easy!

19. Structural directive syntax

The final thing we're going to update is our structural directive syntax. ngFor and ngIf both have extended syntax options. ngIf now allows an 'else' conditional. Let's modify our ngIf in items.component.html to include an 'else'. The element selected by the 'else' condition must be within an ng-template.

```
<div *ngIf="item.inBasket; else buy">
  <p class="status">Item Reserved</p>
  <button class="button-cancel" (click)=
"cancelItem(item)">Cancel Purchase</button>
</div>
<ng-template #buy>
  <button class="button-buy"
(click)="buyItem(item)">Buy Flower</button>
</ng-template>
```

20. ngFor

We have updated syntax in ngFor iterators to enable the 'as' keyword to track an index. It's trivial and doesn't add much value to our Flower Shop app, but say we were receiving high volumes of item.json objects over HTTP and wanted to limit or count how many we are displaying on the page, this change makes that much easier.

```
section class="card" *ngFor="let item of
items; index as i">
  {{i}}
  <div class="panel-body">
  <div class="photo">
  [...]
  </div>
  <table class="item-info">
  [...]
  </table>
  </div>
</section>
```

21. New pipes

Finally, we have a new pipe! Titlecase can be used to force capitalisation of the first letter of each word in a string. This can be useful for consuming data from name and address form fields, for example. Let's test it out on the interpolated item descriptions. There's lots more to explore with Angular 4, with the most significant improvements in larger apps, but hopefully this tutorial has made updating seem a little less daunting!

```
<td>
  <h3 class="item-name">{{item.name}}</h3>
  <p class="description">{{item.
about|titlecase}}</p>
</td>
```

Special offer for readers in ***North America***

6 issues FREE

When you subscribe*

FREE resource downloads every issue

> ***"The only magazine you need to design and develop stunning websites"***

Order hotline **+44 (0)1795 592862**

Online at **www.imaginesubs.co.uk/wed**

Quote **USA3** for this exclusive offer!

***Terms and conditions** This is a US subscription offer. You will actually be charged £80 sterling for an annual subscription. This is equivalent to $105 at the time of writing, exchange rate may vary. 6 free issues refers to the USA newsstand price of $14.99 for 13 issues being $194.87, compared with $105 for a subscription. Your subscription starts from the next available issue and will run for 13 issues. This offer expires 31 July 2017.

Create a dashboard app with React

Make the most out of reusable components to create a highly flexible data visualisation screen

DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/webdesigner

After you have covered the basics of creating a web application with React, it can be hard to know what advantages it provides over updating the page using something like jQuery.

Component creation is a significant bonus. The ability to create self-contained, reusable components means you can keep code smaller and more organised. If components are set up well enough, they can be dropped in where necessary with no additional setup required at all.

In this tutorial, we will be making a dashboard application that keeps an eye on important support metrics. The screen is split up into a grid, which can be customised to show different visuals depending on the data it needs to show.

By making a generic widget component we can chop and change the display without affecting any of the underlying code. Wrapping these in a container component allows us to control the source of that data separate from its display.

We will be making use of CSS Grids – a new way of displaying content in defined blocks. While its syntax is not a focus of this tutorial, you may find it useful to have an understanding of how it works beforehand (see page 44). Browsers that do not support it will display in a single column, much like when using a smaller screen.

1. Download the dependencies

After getting the project files we need to pull down all the required packages we need for the project. These include files from 'create-react-app', which deals with the build process for us.

We also need to run two servers – one that provides hot reloading for the page and another that provides some fake data to test with.

Enter the following on the command line while inside the project directory:

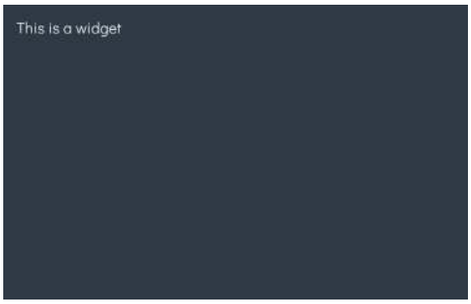
```
/* in one window */
> yarn install
> yarn start
/* in another window */
> yarn serve
```

2. Add the first widget

To start things off, we will render a simple component on the page. With Babel set up, we can write components using ES2015 classes. All we need to do is import them when we need to and Babel with Webpack will do the rest.

Open up /src/components/App.js and add the import to the top of the page. Then, inside the render function, place the component inside the container `<div>`.

```
import Widget from
  '../components/Widget';
[…]
<div className="App">
  <Widget />
</div>
```



3. Style a widget box

On this project, Webpack is set up to pick up CSS imports. This means we can import CSS files like we did with JavaScript in the previous step, which allows us to create separate files for each component. Add the following import to the top of Widget.js. This will link up with the 'Widget' className and will scope the styles to that component.

```
import '../styles/Widget.css';
```

4. Add heading and content

Each widget will need a short description of what data it is showing. To keep things customisable, we will pass this in as a property – or "prop" – to the component when we use it.

As for the content, React supplies a special 'children' prop, which will contain the content entered between a component's opening and closing tags.

Replace the placeholder <p> in the render function with the following. The Loading component will come into play later on.

```
<div className="header">
  <h2>{this.props.heading}</h2>
  {this.props.loading ? <Loading />:""}
</div>
<div className="content">
  {this.props.children}
</div>
```

5. Let the widget span the grid

In addition to the stylesheets we import, we can also create React styles dynamically based on the props passed through.

To span columns and rows with CSS Grid use the grid-column and grid-row properties. We can pass through 'colspan' and 'rowspan' props to alter this per

Use curly brackets

Curly brackets should surround any numbers being passed through in props directly, as otherwise they may be evaluated as strings by the recipient and fail any of its type checks.

Left
With the CSS Grid specification, layout elements such as gutters will be applied regardless of any styling applied to child elements

Top left
When importing CSS with Webpack, styles can be scoped to just the component in question to avoid them leaking to other parts of the page

Top right
In future versions of React, PropTypes will no longer be part of the React directly. Instead they will need to be imported as a separate package

component in a similar way to how table cells work in HTML.

Apply styles in the Widget constructor and link them to the container .

```
if (props.colspan !== 1) {
  this.spanStyles.gridColumn
  = `span ${props.colspan}`;
}
if (props.rowspan !== 1) {
  this.spanStyles.gridRow
  = `span ${props.rowspan}`;
}
[…]
<div style={ this.spanStyles }
  className="Widget">
```

6. Supply default props

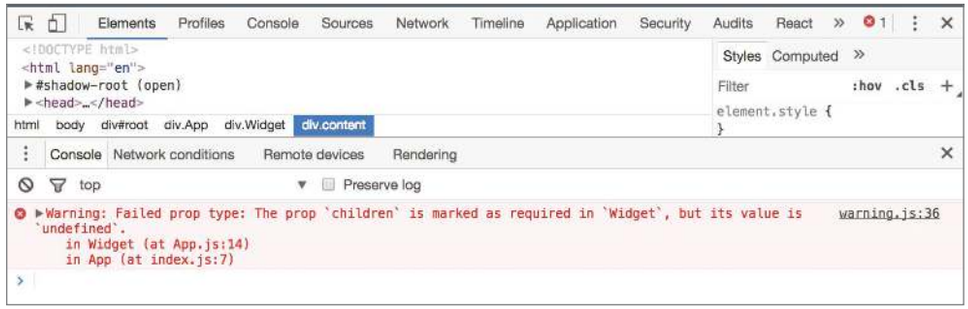
Right now our Widget is blank as we do not supply any props as yet. In these cases, we can supply default props to use instead.

Unless told otherwise, CSS Grids will default to taking up the smallest unit it can, which in this case is a 1x1 square. Just before we export the Widget, supply some default props to that effect.

```
Widget.defaultProps = {
  heading: "Unnamed Widget",
  colspan: 1,
  rowspan: 1
}
```

Slow down requests

To help test out loading states, you can delay responses from the data server by either running 'yarn serve-delayed', or by throttling the connection using the developer tools in the browser.



7. Enforce specific props

Components can provide hints as to what type of values should be sent as props. While developing an application, any incorrectly passed props will show up in the console as warnings to help avoid bugs further down the line.

Just underneath the default props, define what props should or need to be passed in, and what type they should be.

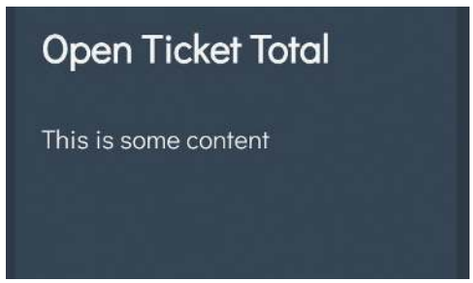
```
Widget.propTypes = {
  heading: React.PropTypes.string,
  colspan: React.PropTypes.number,
  rowspan: React.PropTypes.number,
  children:
    React.PropTypes.element.isRequired
}
```

8. Add props to the app

By defining the 'children' prop as required, you may notice the browser complaining it's currently undefined. While this will not break the app, it will keep bugging us until it's sorted.

Head back over to App.js and add a heading prop to the widget we created earlier. Instead of making the tag self-closing, open it up and add some placeholder content to show it's working.

```
<Widget heading="Open Ticket Total">
  <p>This is some content</p>
</Widget>
```



9. Display some data

The NumberDisplay component works much like the widget we just created – it renders some text based purely on the props we pass into it. We can drop it in where we need to and have a consistent display of numerical data.

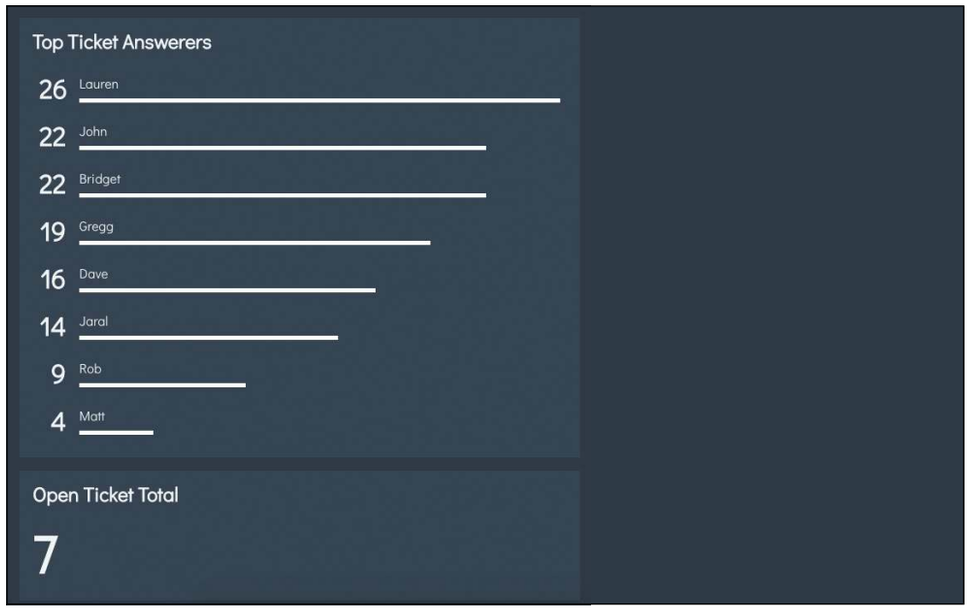
Import the NumberDisplay component at the top and use it to replace the placeholder content you just added within Widget.

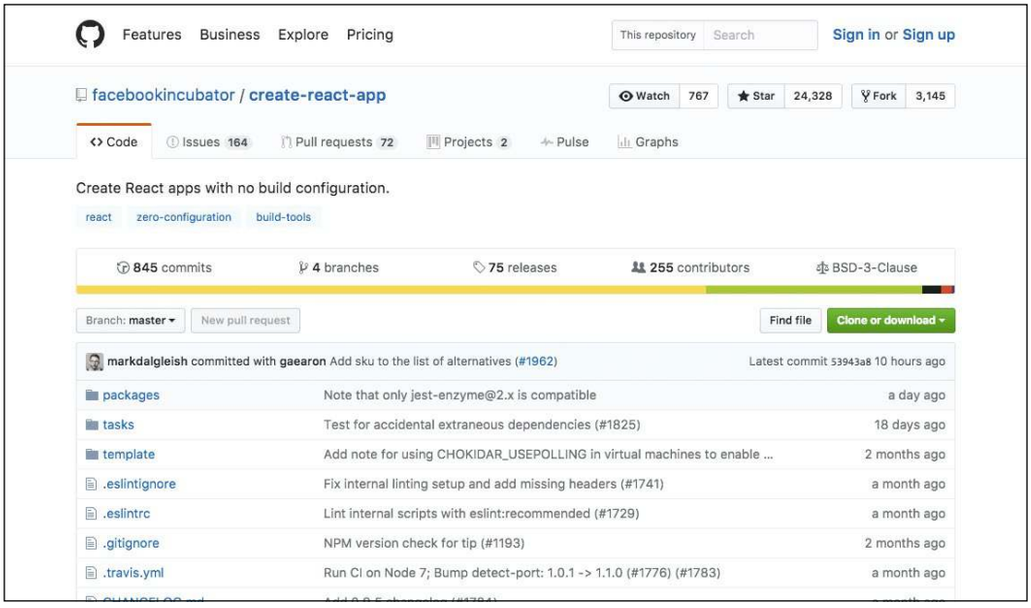
```
import NumberDisplay from '../components/
NumberDisplay';
[…]
<NumberDisplay max={9} value={5} />
```

Top left
The NumberDisplay component is a 'presentational' component, as it has no internal state and relies wholly on the props passed to it

Top right
ListWidget uses CSS Flexbox styling to make the most of the space available. When spanning multiple rows, this allows more items to be displayed

Right
We can use the @supports syntax in CSS to detect whether or not the browser supports grid layouts. If not, we default to the single-column view





Using 'create-react-app'

Getting development environments set up can be tricky to set up for every project. There are many React 'boilerplates' that have all the necessary tools in place to start your next project from.

Facebook has released 'create-react-app', which will instead generate a new project based on the latest best practices and includes Webpack, Babel and the testing tool Jest from the outset.

The advantage over traditional boilerplate projects is that it equates to a single dependency, which keeps things simple while a project is young. Once a more complicated setup is required, running the 'eject' command splits things out with helpful comments to know what each piece does.

To find out more, check out the GitHub project at: **github.com/facebookincubator/create-react-app**

10. Convert to NumberWidget

At the moment, there is quite a lot of code used to show something that will not change across widgets. We can make a special component to encapsulate all of it. That way we only need to import NumberWidget.

Replace the Widget and NumberDisplay imports at the top of App.js with NumberWidget. Make sure to also replace them in the render method.

```
import NumberWidget from
'../components/NumberWidget';
[…]
<NumberWidget
heading="Open Ticket Total"
max={9} value={5} />
```

11. Show a progress bar

Inside NumberWidget.js there are a couple of methods that help us to render things under certain conditions. We can call these in the render method and they can return what we need to output.

Inside showProgress() add some code that works out if we should render a progress bar. We need a minimum, maximum and a value at the very least.

```
if (this.props.min !== undefined &&
this.props.max !== undefined &&
this.props.value !== undefined) {
return <Progress min={this.props.min}
max={this.props.max}
value={this.props.value} />;
}
return null;
```

12. Add a Container component

Manually adding data isn't going to be much help to us. We can add a container around NumberWidget that will supply it with fresh data from our server. Inside App.js, replace what we added in step 10 with the NumberWidgetContainer and pass it a URL to load data from.

```
import NumberWidgetContainer from
  ../components/NumberWidgetContainer';
[…]
<NumberWidgetContainer href="
  http://localhost:3001/tickets/open"
  heading="Open Ticket Total" />
```

13. Fetch data on load

Over its lifetime, a React component will call many different methods depending on what point it's at. The componentDidMount method will be called when a component first renders, which makes it an ideal place to fetch data.

Add the following to componentDidMount in NumberWidgetContainer.js. This will call the fetchData method now and every minute after that. We will fix the TypeError in the next step.

```
this.getData().then(_ => {
  this.interval =
    setInterval(this.getData, 60000);
});
```

14. Update the state

Each component can have its own internal state, which holds information about itself at any given moment. We can pass this state as props for other components to use.

Make the request to the supplied 'href' prop URL using the AJAX module Axios and update the state with the values supplied. These will automatically be passed to NumberWidget, which will then update itself.

```
this.setState({ loading: true });
return axios.get(this.props.href)
.then(resp => {
  this.setState({
  loading: false,
  min: resp.data.min,
  max: resp.data.max,
  value: resp.data.value
  });
})
```

15. Add in a List Widget

The groundwork we have done for NumberWidget can be applied to other types of widgets too. ListWidgetContainer can be passed the same props as NumberWidgetContainer, but will render a list of values instead.

Import ListWidgetContainer alongside the rest, and place the component just above NumberWidgetContainer in the render method.

```
import NumberWidgetContainer from
  '../components/NumberWidgetContainer';
[…]
<ListWidgetContainer
  href="http://localhost:3001/stats/top"
  heading="Top Ticket Answerers"
  rowspan={3} />
```

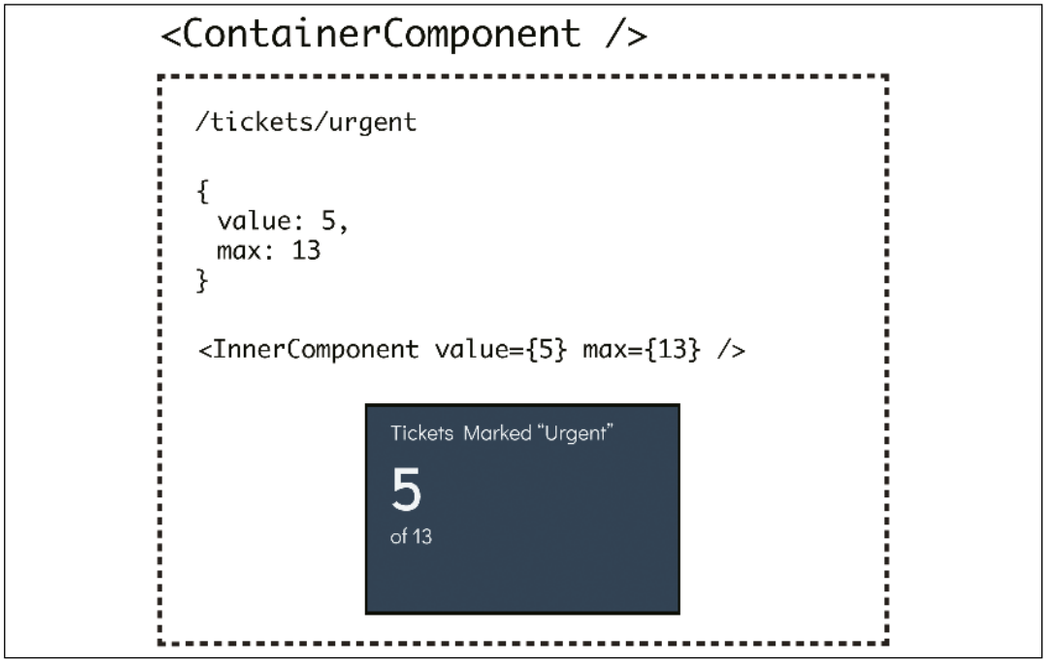
16. Display list items

With the data supplied in the sortedItems variable as an array, we can iterate over the results and render a separate component for each.

To do this we can use the 'map' method on the array prototype. This creates a new array of components that React will render.

Inside the ListDisplay component in showWidget, add the map function to create new ListItems.

```
{sortedItems.map((item, index) =>
  <ListItem key={item.label}
   label={item.label} value={item.value}
   min={min} max={max} />)}
```



Using Container components

There are many React patterns that if followed can help develop stable, more reusable components. One of the most common is the container, or 'smart' component pattern.

Instead of coupling a component with the logic of fetching its data, we can get the data elsewhere and pass it through props. The container component has no visuals and will only render the other component that makes use of the data.

This means if the format of the fetched data has changed, we can fix the container without having to touch the inner component, which could be in use in multiple places across the application.

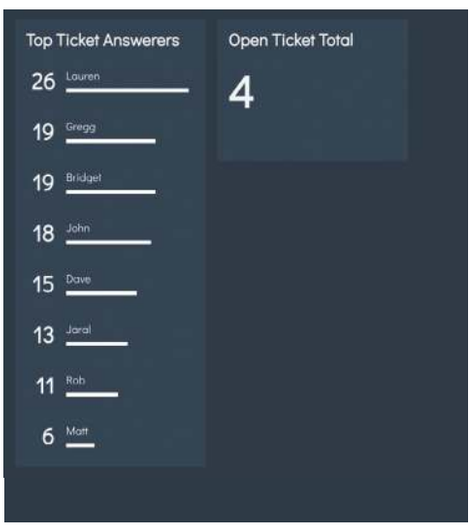
Separating the concerns of these components makes them easier to use elsewhere in a project, as well as more simple to unit test.

17. Sort data by value

We should avoid relying on data being sorted at the source to ensure we provide useful information to the user. We can use a sorting function to do this inside the sortListItems method.

Array sorting functions take two values and compare them. Apply this to sortedItems and return it to sort its content in descending order.

```
return sortedItems.sort((a, b) => {
  if (a.value > b.value) {
  return -1;
  } else if (a.value < b.value) {
  return 1;
  }
  return 0;
});
```



18. Add a Graph widget

Graph widgets can display large sets of data clearer than the number or list widgets. It uses a library called Chart.js with a React wrapper that updates as props change. Switch back to App.js, import GraphWidgetContainer and add it to the render function under the existing widgets.

```
import GraphWidgetContainer from
  '../components/GraphWidgetContainer';
[…]
<GraphWidgetContainer
href="http://localhost:3001
  /tickets/progression"
heading="Tickets Over Time"
colspan={2} rowspan={2} />
```

19. Prepare data for display

Chart.js takes data in a specific format. In this case, the 'datasets' array is a collection of points used to draw a line.

We will use state to hold all the configuration data for Chart.js and update it with the datasets fetched from the URL, which arrive in a different format.

Open up GraphWidget.js and loop over the props in generateDatasets to update the state.

```
props.data.forEach(function (data) {
  datasets.push({
  label: data.label,
  data: data.data,
  fill: false,
  borderColor: data.color,
  pointRadius: 0,
  pointHitRadius: 10
  });
}, this);
```

20. Regenerate when props change

The React wrapper for Chart.js will update when its props change, but as the state holds the set up, we do not update those values directly.

componentWillReceiveProps is another lifecycle method that will fire when a component's props will update. Sometimes this can fire when values have not changed, but a quick check for that can protect against unnecessary updates.

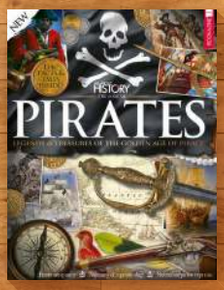
```
Call generateDatasets if the data has
changed.
if (this.props.data !== nextProps.data){
  this.generateDatasets(nextProps);
}
```

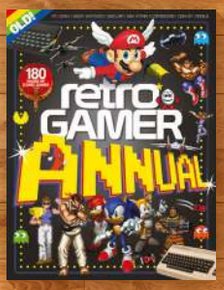
21. Add remaining widgets

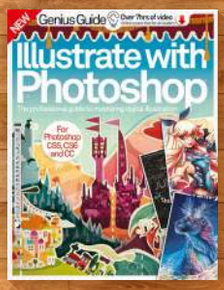
Finally, all that is left to do is add in some more data. With the flexibility we have created from the various widgets, we can add in whichever widgets are necessary.

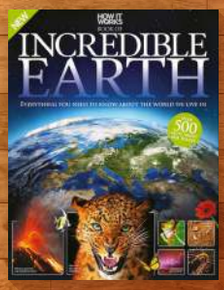
Fill in the remaining gaps on the page. Play around with the types, row and column spans, and positions of widgets to suit the data best.

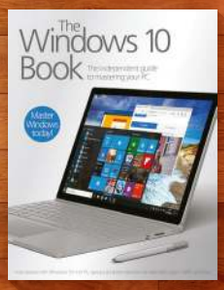
```
<NumberWidgetContainer href="
  http://localhost:3001/tickets/today"
  heading="Tickets Opened Today" />
<NumberWidgetContainer href="
  http://localhost:3001/tickets/urgent"
  heading="Tickets Marked 'Urgent'" />
<NumberWidgetContainer href="
  http://localhost:3001/stats/response"
  heading="4 Hour Response %"
  colspan={2}/>
```





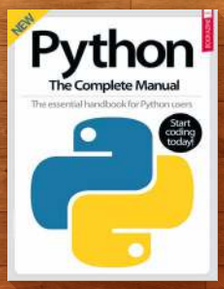


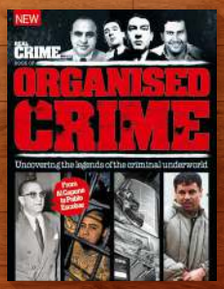


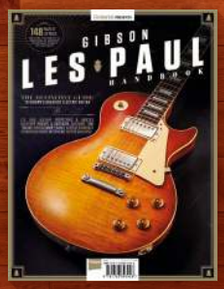


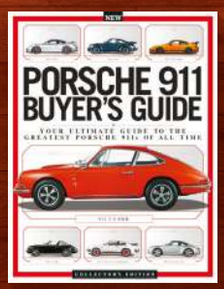
Discover another of our great bookazines

From science and history to technology and crafts, there are dozens of Future bookazines to suit all tastes









Get great savings when you buy direct from us

1000s of great titles, many not available anywhere else

World-wide delivery and super-safe ordering

Future **www.myfavouritemagazines.co.uk**

Magazines, back issues & bookazines.

Get your listing in our directory

To advertise here contact Chris

chris.mitchell@futurenet.com
+44 (0)1225 687832

COURSE LISTINGS

Featured: Makers Academy

www.makersacademy.com
Twitter: @makersacademy
Facebook: MakersAcademy

About us

Makers Academy is a fully immersive, 12-week computer programming boot camp. With their help, you will learn the principles of software craftsmanship and they'll also help you get your first job. They're Europe's number-one developer boot camp, running highly selective classes of the offline course every six weeks, and a remote course every 12 weeks. They take a 'learn by doing' approach, through project-based work. Students are encouraged to work in pairs on coding challenges, with weekly tests, culminating in a final project. They help set up job interviews via their network of hiring partners including ThoughtWorks and Deloitte Digital.

What we offer

- **On-site:** 12 week full-time coding course from the on-site campus in London
- **Remote:** 12 week full-time coding course remotely from home

5 tips from the pros

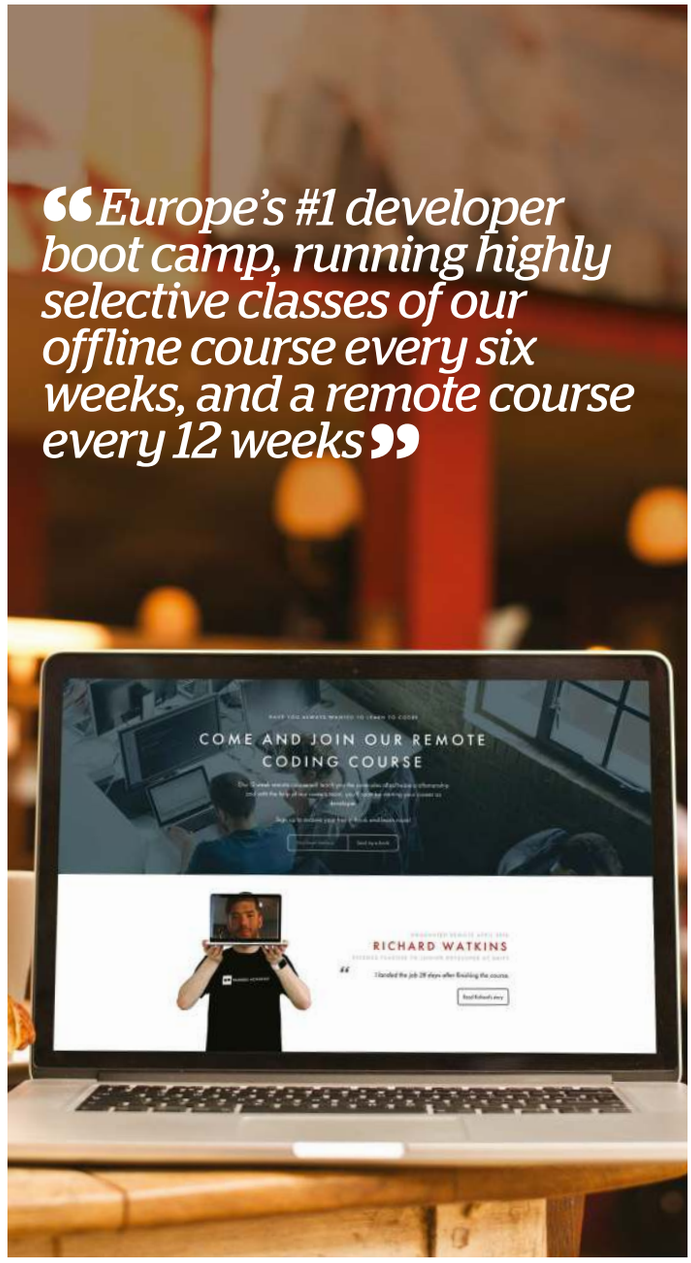
1.Research all your options
Do your research into lots of different boot camps, read the reviews, read the student blogs and reach out to previous graduates and speak to them.

2.Dabble in code
Although the course is for beginners, it's important that you've started to at least try to learn to code on your own.

3.Prepare for the interview
We send you everything to prepare for the interview. Make sure you go through all the resources and give yourself two weeks to prepare.

4.Budget
The course is full-time for three months and it can take up to three months after to secure a job. It's important to financially plan for the period you won't be working.

5.Visit us!
Book a visit and come visit us! Come see the Makers Academy HQ in person and learn more.



Richard Watkins
Science teacher to junior developer at Shift
Makers Academy was frustrating and daunting but amazing and I wouldn't change any of it. I landed the job 28 days after finishing the course.

Ina Tsetsova:
Email campaign manager to graduate software developer at ThoughtWorks
I found a really nice community and I've met really cool people. I got a job quicker than I expected.

Hannah Carney
3D designer to junior developer at Play Consulting
Makers Academy not only focuses on your learning for code, but they also focus on your well-being. Work feels like fun and I've finally found a job I love.

Get your listing in our directory

To advertise here contact Chris

chris.mitchell@futurenet.com
+44 (0)1225 687832

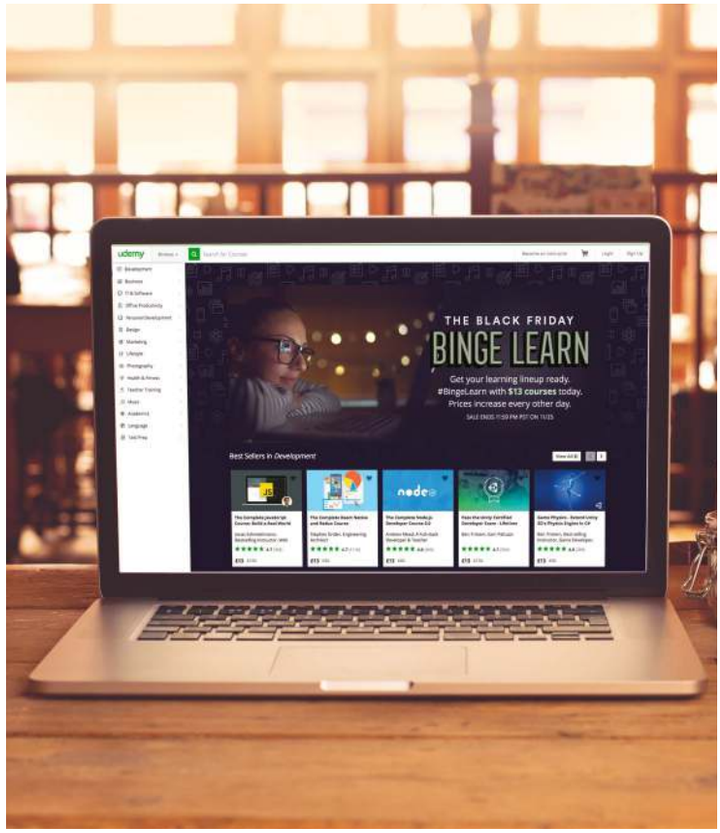
UDEMY

www.udemy.com

Twitter: @udemy
Facebook: udemy

The inspiration for Udemy began in a small village in Turkey, where founder Eren Bali grew up frustrated by the limitations of being taught in a one-room school house. Realising the potential of learning on the internet, he set out to make quality education more accessible. Udemy is now a global marketplace for learning and teaching online. Students can master new skills by choosing from an extensive library of over 40,000 courses including HTML, CSS, UX, JavaScript and web development.

40,000+ courses: There is a course for every designer and dev
Self-paced learning: Learn how to code at your own pace



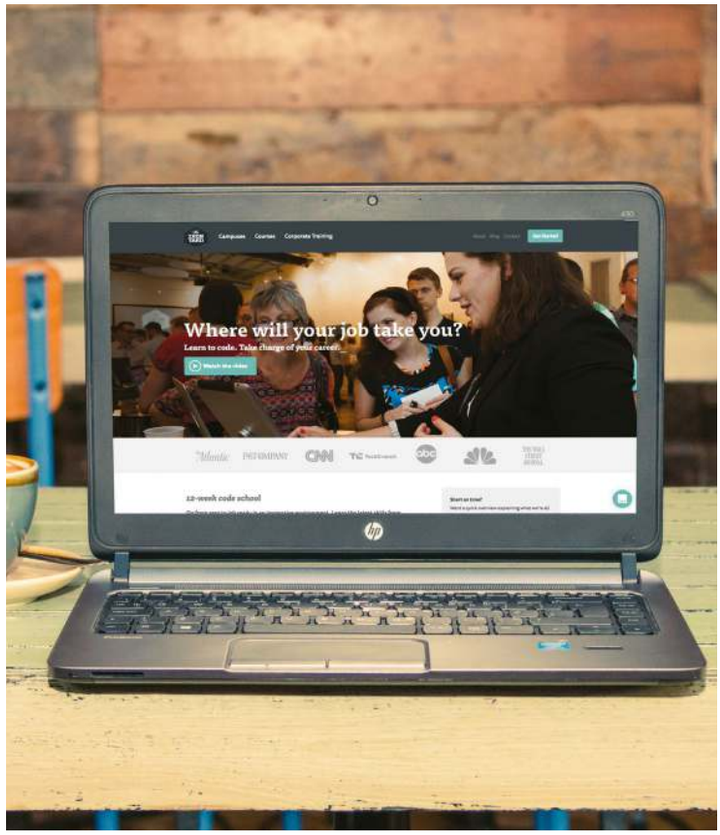
THE IRON YARD

www.theironyard.com

Twitter: @TheIronYard
Facebook: TheIronYard

The Iron Yard is one of the world's largest and fastest-growing in-person code schools. It offers full-time and part-time programs in Back-End Engineering, Front-End Engineering, Mobile Engineering and Design. The Iron Yard exists to create real, lasting change for people, their companies and communities through technology education. The in-person, immersive format of The Iron Yard's 12-week courses helps people learn to code and be prepared with the skills needed to start a career as junior-level software developers.

12-week code school: Learn the latest skills from industry pros
Free crash courses: One-night courses, the perfect way to learn



WE GOT CODERS

www.wegotcoders.com
hello@wegotcoders.com

We Got Coders is a consultancy that provides experts in agile web development, working with startups, agencies and government. Take one of their 12-week training course that covers all that is required to become a web developer, with highly marketable full-stack web development skills.

- Classroom-based training
- Real-world work experience
- Employment opportunities

FUTURELEARN

www.futurelearn.com
feedback@futurelearn.com

Choose from hundreds of free online courses: from Language & Culture to Business & Management; Science & Technology to Health & Psychology. Learn from the experts. Meet educators from top universities who'll share their experience through videos, articles, quizzes and discussions.

- Learn from experts
- Free courses
- All-device access

GYMNASIUM

www.thegymnasium.com
help@thegymnasium.com

Gymnasium offers free online courses designed to teach creative professionals in-demand skills. Courses are all self-paced and taught by experienced practitioners with a passion for sharing practical lessons from the design trenches.

- Gain real-world skills
- Get expert instruction
- Career opportunities

Get your listing in our directory

To advertise here contact Chris

chris.mitchell@futurenet.com
+44 (0)1225 687832

HOSTING LISTINGS

Featured host: **Netcetera**

www.netcetera.co.uk
0800 808 5450

About us

Formed in 1996, Netcetera is one of Europe's leading web hosting service providers, with customers in over 75 countries worldwide.

As the premier provider of data centre colocation, cloud hosting, dedicated servers and managed web hosting services in the UK, Netcetera offers an array of services designed to more effectively manage IT infrastructures. A state-of-the-art data centre environment enables Netcetera to offer your business enterprise-level colocation and hosted solutions. Providing an unmatched value for your budget is the driving force behind our customer and managed infrastructure services. From single server to fully customised data centre suites, we focus on the IT solutions you need.

What we offer

- **Managed hosting –** A full range of solutions for a cost-effective, reliable, secure host.
- **Cloud hosting –** Linux, Windows, Hybrid and Private Cloud Solutions with support and scalability features.
- **Data centre colocation –** Single server through to full racks with FREE setup and a generous bandwidth.
- **Dedicated servers –** From QuadCore up to Smart Servers with quick setup and fully customisable.

5 tips from the pros

1. Reliability, trust, support

Reliability is a major factor when it comes to choosing a hosting partner. Netcetera guarantees 100% uptime, multiple internet routes with the ability to handle DDOS attacks, ensuring your site doesn't go down when you need it.

2. Secure & dependable

Netcetera prides itself on offering its clients a secure environment. It is accredited with ISO 27001 for Security along with the options of configurable secure rackspace available in various configurations.

3. 24/7 Technical support

Netcetera has a committed team of knowledgeable staff available 24/7 to provide you with assistance when you need it most. Our people make sure you are happy and your problems are resolved as quickly as possible.

4. Value for money

We do not claim to be the cheapest service available, but we do claim to offer excellent value for money. We also provide a price match on a like-for-like basis, as well as a price guarantee for your length of service.

5. Ecofriendly

Netcetera's environmental commitment is backed by use of ecocooling and hydroelectric power. This makes Netcetera one of the greenest data centres in Europe.

Testimonials

Roy T

"I have always had great service from Netcetera. Their technical support is second to none. My issues have always been resolved very quickly."

Suzy B

"We have several servers from Netcetera and their network connectivity is top-notch, with great uptime and speed is never an issue. Tech support is knowledgeable and quick in replying. We would highly recommend Netcetera."

Steve B

"We put several racks into Netcetera, basically a complete corporate backend. They could not have been more professional, helpful, responsive or friendly. All the team were an absolute pleasure to deal with, and nothing was too much trouble, so they matched our requirements 100%."

Get your listing in our directory

To advertise here contact Chris

chris.mitchell@futurenet.com
+44 (0)1225 687832

Supreme hosting

CWCS Supreme Hosting. Supreme Support. Managed Hosting

www.cwcs.co.uk
0800 1777 000

CWCS Managed Hosting is the UK's leading hosting specialist. They offer a fully comprehensive range of hosting products, services and support. Their highly trained staff are not only hosting experts, they're also committed to delivering a great customer experience and are passionate about what they do.

- Colocation hosting
- VPS
- 100% Network uptime

UK-based hosting

CYBERHOSTPRO

www.cyberhostpro.com
0845 5279 345

Cyber Host Pro are committed to providing the best cloud server hosting in the UK; they are obsessed with automation. If you're looking for a hosting provider who will provide you with the quality you need to help your business grow, then look no further than Cyber Host Pro.

- Cloud VPS servers
- Reseller hosting
- Dedicated servers

Cluster web hosting

fasthosts

www.fasthosts.co.uk
0808 1686 777

UK based and operating 24/7 from dedicated UK data centres. Fasthosts keep over one million domains running smoothly and safely each day. Services can be self-managed through the Fasthosts Control Panel.

- Dedicated servers
- Cloud servers
- Hosted email

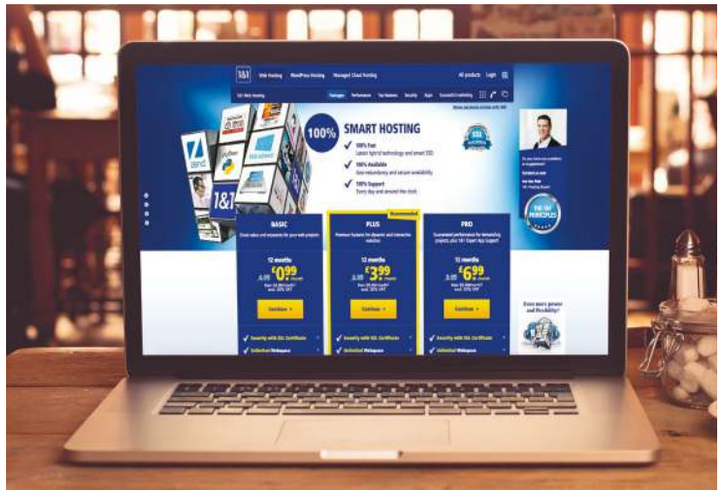
Budget hosting

HETZNER ONLINE

www.hetzner.com
+49 (0)9831 505-0

Hetzner Online is a professional web hosting provider and experienced data centre operator. Since 1997, the company has provided private and business clients with high-performance hosting products as well as the infrastructure for the efficient operation of sites. A combination of stable technology, attractive pricing, flexible support and services has enabled Hetzner Online to strengthen its market position both nationally & internationally.

- Dedicated/shared hosting
- Colocation racks
- SSL certificates



All-inclusive hosting

1&1

www.1and1.co.uk
0333 336 5509

1&1 Internet is a leading hosting provider that enables businesses, developers and IT pros to succeed online. Established in 1988, 1&1 now operates across ten countries. With a comprehensive range of high-performance and affordable products, 1&1 offers everything from simple domain registration to award-winning website building tools, eCommerce packages and powerful cloud servers.

- Easy domain registration
- Professional eShops
- High-performance servers

SSD web hosting

bargainhost

www.bargainhost.co.uk
0843 289 2681

Since 2001, Bargain Host have campaigned to offer the lowest possible priced hosting in the UK. They have achieved this goal successfully and built up a large client database which includes many repeat customers. They have also won several awards for providing an outstanding hosting service.

- Shared hosting
- Cloud servers
- Domain names

Value Linux hosting

PATCHMAN WEB HOSTING

patchman-hosting.co.uk
01642 424 237

Linux hosting is a great solution for home users, business users and web designers looking for cost-effective and powerful hosting. Whether you are building a single-page portfolio, or you are running a database-driven eCommerce website, there is a Linux hosting solution for you.

- Student hosting deals
- Site designer
- Domain names

Flexible cloud servers

elastichosts

elastichosts.co.uk
020 7183 8250

ElasticHosts offer simple, flexible and cost-effective cloud services with high performance, availability and scalability for businesses worldwide. Their team of engineers provide excellent support 24/7 over the phone, email and ticketing system.

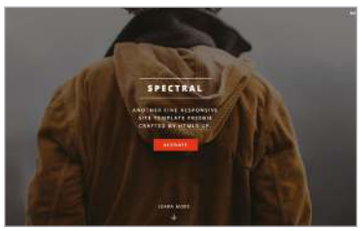
- Cloud servers with any OS
- Linux OS containers
- 24/7 expert support

Free with your magazine

Instant access to these creative resources…

Essential assets and templates

Get textures, fonts, backgrounds and more



Exclusive video tutorials

Learn to code/create with HTML, CSS, JS & PHP

Tutorial project files

All the assets you'll need to follow our tutorials

Plus, all of this is yours too…

- All-new tutorial files to help you master this issue's HTML, CSS and JavaScript techniques
- 72 minutes of expert HTML/CSS/JS troubleshooting videos from Pluralsight (www.pluralsight.com)
- 16 Orange and Teal Photoshop actions from SparkleStock (www.sparklestock.com)
- 70 vintage wood textures

Log in to www.filesilo.co.uk/webdesigner

Register to get **instant access** to this pack of must-have creative resources, how-to videos and tutorial assets

Free for digital readers too!
Read on your tablet, download on your computer

FileSilo

The home of great downloads – exclusive to your favourite magazines from Future!

- Secure and safe online access, from anywhere
- Free access for every reader, print and digital
- Download only the files you want, when you want
- All your gifts, from all your issues, in one place

Get started

Everything you need to know about accessing your FileSilo account

01 Follow the instructions on screen to create an account with our secure FileSilo system. Log in and unlock the issue by answering a simple question about the magazine.

02 You can access FileSilo on any computer, tablet or smartphone device using any popular browser. However, we recommend that you use a computer to download content, as you may not be able to download files to other devices.

03 If you have any problems with accessing content on FileSilo, take a look at the FAQs online or email our team at the address below.

filesilohelp@futurenet.com

An incredible gift for subscribers

Subscribe today & unlock the free gifts from more than 45 issues

Access our entire library of resources with a money-saving subscription to the magazine – that's more than 900 free resources

Over 53 hours of video guides

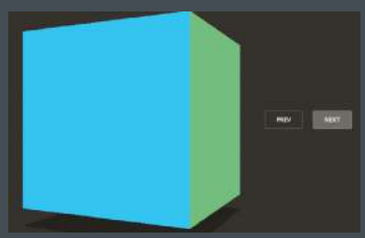
Let the experts teach you to create and code

More than 400 tutorials

Get the code you need to get creative

Over 210 creative assets

Templates, fonts, textures and backgrounds



Head to page 34 to subscribe now

Already a print subscriber?
Here's how to unlock FileSilo today…

Unlock the entire Web Designer FileSilo library with your unique Web ID – the eight-digit alphanumeric code printed above your address details on the mailing label of your subscription copies – also found on any renewal letters.

More than 900 reasons to subscribe

+ More added every issue

Essential projects to create menus, layouts, forms, touch sliders and animation optimised to work perfectly on the small screen

SAY HELLO TO REACT NATIVE

Discover how to code and create with the JavaScript framework

CSS BUTTONS & CHECKBOXES

Build active radio buttons and checkboxes with CSS and HTML

CODE A 3D SPHERE WITH WEBGL

Impress visitors with a browser-based, data-driven 3D sphere

UTILISE THE WEB AUDIO API

Employ the power of web audio to build an in-browser instrument

Visit the **WEB DESIGNER** online shop at

Future myfavouritemagazines

myfavouritemagazines.co.uk

for the latest issue, back issues and specials

ALL IN YOUR NEXT

WEB DESIGNER

Issue 263 on sale

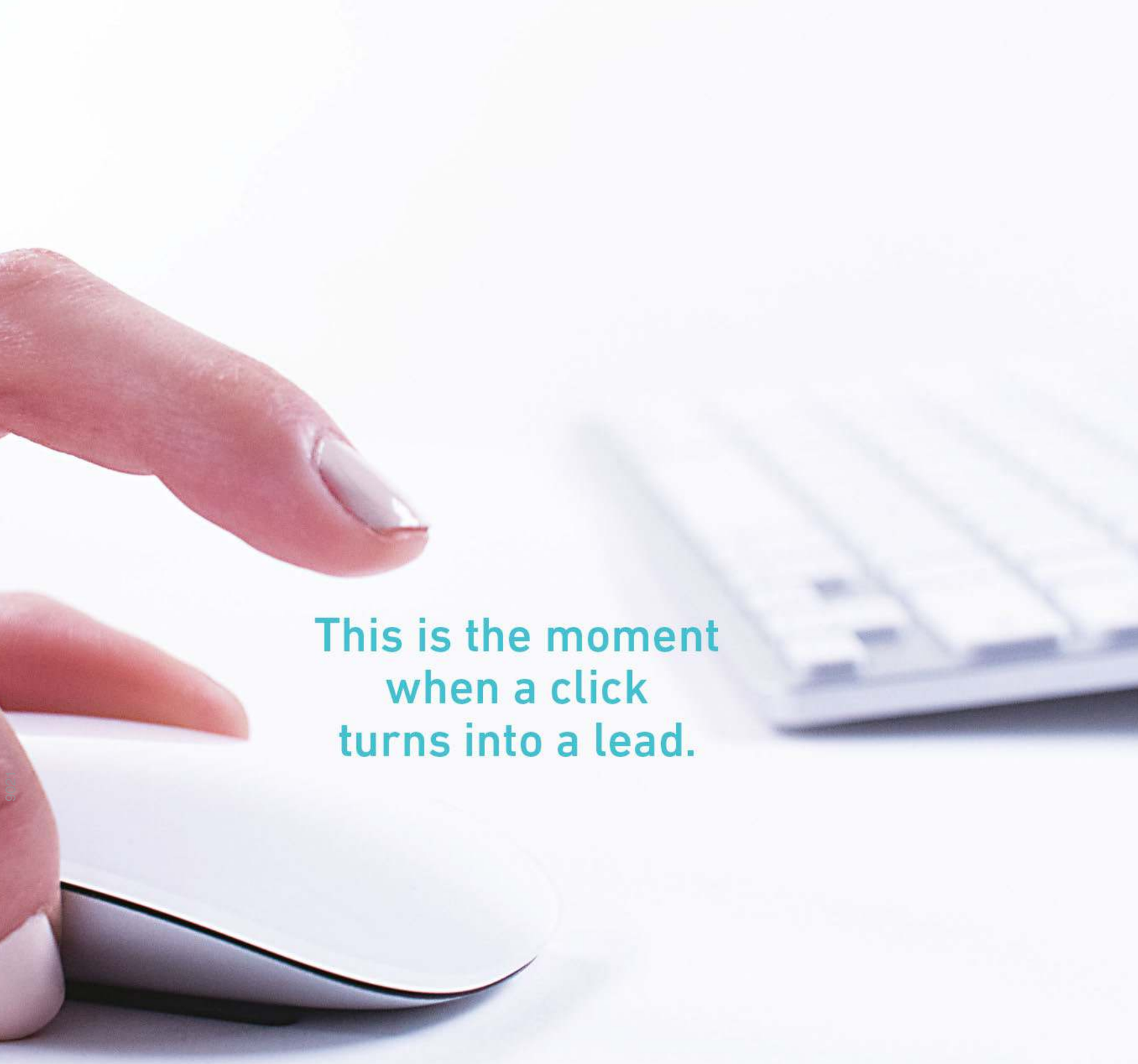
Thursday 22nd June 2017

(mt)

DRAW
WIRE
DESIGN
CODE

Media Temple focuses on delivering the world's best performance and support so that creative professionals and enterprises can focus on bold ideas, not web servers.

mediatemple.net



PRESS AHEAD

WP Engine's digital experience platform drives your business forward faster. **wpengine.co.uk**

WP engine®