

33
PAGES OF
EXPERT TUTORIALS

FREE 225 MINUTES
OF JS VIDEOS

web designer

HTML5 CSS3 jQuery JavaScript

@WebDesignerMag

/WebDesignerUK

10 HTML APIs YOU NEED TO BE USING NOW

GET 60 FRAMES
PER SECOND

Stop the stutter with CSS & JS

TUTORIAL

SEE THE
COVER IN
ACTION
Page 3

ANIMATION

20 KILLER TECHNIQUES! RECREATE THE WEB'S BEST EXAMPLES TODAY

BUILD A VOICE
ACTIVATED APP

Code a to-do list using Node.js

CREATE DYNAMIC
3D TEXT EFFECTS

Add excitement & interaction to typography

WIN
PRO GUIDE GIVEAWAY!

BETTER
WEB
TYPOGRA-
PHY FOR
A BETTER
WEB

Future
ISSUE 268

BROWSER
CONSOLES
MASTERCLASS

Tools to dig deep into your code

(mt)

DRAW
WIRED
DESIGN
CODE

Media Temple web hosting delivers premium performance and support so that creative professionals and enterprises can focus on bold ideas, not web servers.

mediatemple.net

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
Editor

Recreate animations with CSS



SS and animation have grown to become good friends on the web. Need to add a little dynamism to a project? Then give CSS a call. There is no denying that animation can take a project or design to a new level. But, you need to think very carefully about how it's used.

In our latest lead feature (starting on page 42) we have scoured the web looking for the best-in-class animations to dissect and show you the techniques needed to recreate. You will find 20 killer techniques that range from the inspirational to the practical. Check out the flying birds, animated text drawing, mesmerising mandalas and much more.

All code is included, making it even easier to recreate. Plus, try out five of the best CSS animation libraries on the web to add even more smart animations to your projects. Start experimenting now. As an added bonus, see the cover in action and check out the code (bit.ly/2zmoY72).

APIs are not just the domain of software. As browsers have evolved, so have HTML, web and third-party APIs. Simon Jones chooses a selection of the best to help you start building for the next-generation web. Learn how to use Service Workers and get creative with WebGL.

What else do we have? Get a practical guide to Google and Firefox's browser consoles, learn how to get 60fps on animation and make interactive 3D text.

• The `<g>` element in SVG can be used much like a `div` in HTML; we need to wrap each of our bubbles (which are already in a group) in a group tag •

Highlight



• We try not to be led by the constraints or possibilities of a tool, but instead, ask how this tool can contribute in the vision we have for the end result •

51North is a studio that keenly understands how to communicate messages to multiple audiences. Page 34

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

Visit our blog for opinion, freebies & more www.creativebloq.com



FREE - exclusive with this issue
59 **Designer resources**

Video Tuition - 225 minutes of expert video guides on Node.js from Pluralsight (www.pluralsight.com)

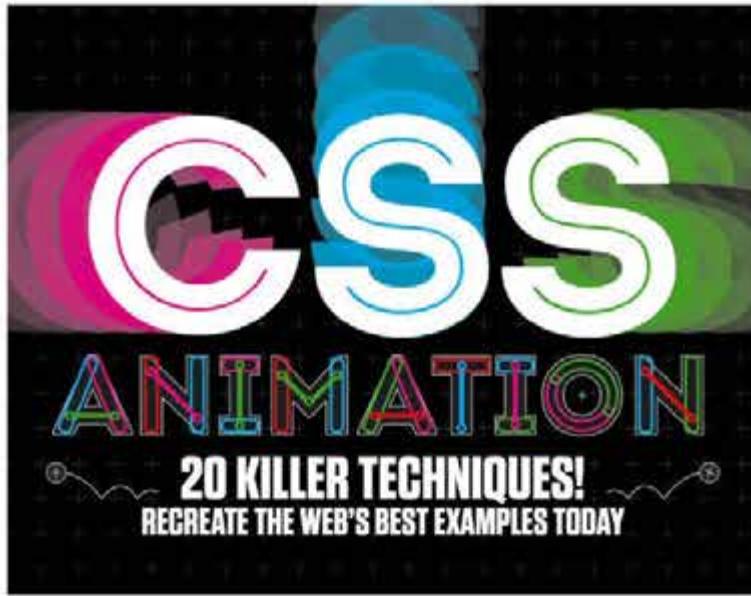
Assets - 50 Seamless asphalt textures and 8 Dystopian Photoshop actions from Sparklestock (sparklestock.com)
- Tutorial files and assets



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...



Steven Roberts

Steven is the lead front-end developer at Better Brand Agency in Middlesbrough. This time out, he has been scouring the web looking for inspirational and best-in-class animations to dissect and recreate. Get the code and learn the secrets of building beautiful and practical animations for today's web.

Page 42

• CSS has a powerful animation engine and really simple code in order to produce complex animations •

Simon Jones

 You can do more on the web now than ever before, thanks to numerous APIs which extend the functionality available through the browser. This month, Simon takes a look at some web APIs you really can't afford to miss. **Page 66**

Mark Shufflebottom

 Mark is a Professor of Interaction Design at Sheridan College of Advanced Learning near Toronto. In this issue's tutorial, Mark is creating animated text effects for headers using flowing, swirling particles. **Page 52**

Tam Hanna

 Tam started out wiggling command lines on process computers. So, he is happy to see old and trusted concepts revived for the world of Web 2.0. This issue he takes a look at the benefits of Google and Firefox's consoles. **Page 74**

Ashley Firth

 Ashley is lead front-end developer at startup energy supplier Octopus Energy and has worked with major clients including Sky. In this issue, he shows the power of creating documentation through code commenting. **Page 80**

Matt Crouch

 Matt is a front-end developer from London. Without careful consideration, interface animations can stutter and ruin the user experience. In this issue, learn about the latest techniques to keep things running smooth. **Page 56**

Luke Harrison

 Luke is a web developer from Sheffield, who is all about scalable and efficient front-end architecture. In this issue, he explores how to implement the WordPress REST API into a Vue.js-powered blog using Vuex. **Page 60**

Lily Madar

 Lily is a creative technologist with a tendency to tinker with both hardware and web-based projects. This issue, she explores VUIs (voice user interfaces) and shows how to create applications for Google Home. **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 on the top-class websites featured in our Lightbox. **Page 14**

Future Publishing Limited
Richmond House, 33 Richmond Hill
Bournemouth, Dorset, BH2 6EZ

Editorial

Editor Steven Jenkins
steve.jenkins@futurenet.com
01202 586233

Designer Harriet Knight
Senior Art Editor **Will Shum**
Editor in Chief **Amy Hennessey**

Contributors

Kim Jones, Steven Roberts, Simon Jones, David Howell, Mark Shuttlebottom, Luke Harrison, Matt Crouch, Ashley Firth, Lily Madar, Tam Hanna, Mark Billen, Leon Brown, Phil King, Philip Morris, Jen Neal

Photography

James Sheppard

All copyrights and trademarks are recognised and respected

Advertising

Media packs are available on request
Commercial Director Clare Dove
clare.dove@futurenet.com
Senior Advertising Manager Mike Pyatt
michael.pyatt@futurenet.com
01225 687538
Account Director George Lucas
george.lucas@futurenet.com
Account Manager Chris Mitchell
chris.mitchell@futurenet.com

International

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

Print subscriptions & back issues

Web www.myfavouritemagazines.co.uk
Email contact@myfavouritemagazines.co.uk
Tel 0344 848 2852
International +44 (0) 344 848 2852

Circulation

Head of Newtrade **Tim Mathers**
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

Managing Director **Aaron Asadi**
Editorial Director **Paul Newman**
Art & Design Director **Ross Andrews**
Head of Art & Design **Greg Whittaker**
Commercial Finance Director **Dan Jotcham**

Printed by

William Gibbons, 28 Planetary Road,
Willesden, NW13, 3XT

Distributed by

Marketforce, 5 Churchill Place, Canary Wharf, London, E14 5HU
www.marketforce.co.uk Tel: 0203 787 9001

We are committed to only using magazine paper which is derived from responsibly managed, certified forestry and chlorine-free manufacture. The paper in this magazine was sourced and produced from sustainable managed forests, conforming to strict environmental and socioeconomic standards. The manufacturing paper mill holds full FSC (Forest Stewardship Council) certification and accreditation

Disclaimer

All contents © 2017 Future Publishing Limited or published under licence. All rights reserved. No part of this magazine may be used, stored, transmitted or reproduced in any way without the prior written permission of the publisher. Future Publishing Limited (company number 2008885) is registered in England and Wales. Registered office: Quay House, The Ambury, Bath BA1 1UA. All information contained in this publication is for information only and is, as far as we are aware, correct at the time of going to press. Future cannot accept any responsibility for errors or inaccuracies in such information. You are advised to contact manufacturers and retailers directly with regard to the price of products/services referred to in this publication. Apps and websites mentioned in this publication are not under our control. We are not responsible for their contents or any other changes or updates to them. This magazine is fully independent and not affiliated in any way with the companies mentioned herein.

If you submit material to us, you warrant that you own the material and/or have the necessary rights/permissions to supply the material and you automatically grant Future and its licensees a licence to publish your submission in whole or in part in any/all issues and/or editions of publications, in any format published worldwide and on associated websites, social media channels and associated products. Any material you submit is sent at your own risk and, although every care is taken, neither Future nor its employees, agents, subcontractors or licensees shall be liable for loss or damage. We assume all unsolicited material is for publication unless otherwise stated, and reserve the right to edit, amend, adapt all submissions.

ISSN 1745-3534



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: FUTPL). www.futureplc.com

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

Tel +44 (0)1225 442 244

Follow us!

Facebook: www.facebook.com/WebDesignerUK

Twitter: <https://twitter.com/webdesignermag>

1,500
FOOTBALL PITCHES
EVERY DAY!

Did you know that European forests, which provide wood for making paper and many other products, have grown by 44,000km² over the past 10 years? That's more than 1,500 football pitches every day![†]

Love magazines? You'll love them even more knowing they're made from natural, renewable and recyclable wood



[†]UNFAO, Global Forest Resources Assessment 2005-2015.

Two Sides is a global initiative promoting the responsible use of print and paper which, when sourced from certified or sustainably managed forests, is a uniquely powerful and natural communications medium.

There are some great reasons to #LovePaper
Discover them now,
twosides.info



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...

steve.jenkins@futurenet.com • [@WebDesignerMag](https://twitter.com/WebDesignerMag) • www.creativebloq.com

08 VR + AR equals MR

Mozilla looks to create a new standard for web reality. **Web Designer** investigates

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 WordPress: time of change

MakeDo founder Kimb Jones extols the virtues of Gutenberg, the new editor for the CMS

14 Lightbox

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

26 Platform for shoes

Sneaker creator SWEAR needed a new online rebrand. Matter of Form were keen to oblige

34 A latitude of design

51North is a studio that keenly understands how to communicate messages to multiple audiences. **Web Designer** finds out more

42 Killer CSS animation techniques

Learn how to recreate some of the best-in-class animations on the web today

66 HTML APIs you need now

Discover the HTML and third-party APIs that are the building blocks of the next-generation web and find out how to use them

74 Browser console masterclass

Get a closer look at your site code with Google and Firefox's advanced consoles

92 Hosting listings

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

94 Course listings

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

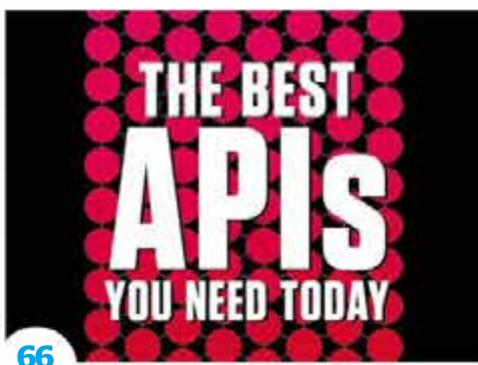
98 Next month

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

Cover focus

42

20 KILLER TECHNIQUES
RECREATE THE WEB'S BEST EXAMPLES



FileSilo

96 Get the latest must-have resources and videos

- 225 minutes of expert Node.js video guides from Pluralsight
- 50 Seamless asphalt textures
- 8 Dystopian Photoshop actions



34
Profile: 51North
Communicating the message

Never miss
an issue
Subscribe
Turn to page 32 now

USA readers turn to page 85
for an exclusive offer



66 The new Gutenberg editor aims to bring everything together in a more modern and easy-to-use layout system 99

COMMENT - Kimb Jones - p11



14
Lightbox: Name That Place
Take the quiz and test your knowledge

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



52 Make interactive 3D type effects

Learn how to use JavaScript to create flowing lines and reactive mouse movement site headers

56 Get 60 frames per second

Code silky-smooth UIs with the help of the latest CSS properties and the Web Animations API

60 Power a blog with WordPress API

Discover how to implement the WordPress API into a Vue.js-powered blog app using Vuex

Web Workshop

50 Add an instructional video

worldbrush.net

Introduce a demo to promote an app

64 Create a text knockout effect

badassfilms.tv/real/dante-ariola

Bring bold typography into focus

Web Developer

74 Browser console masterclass

Get a closer look at your site code with Google and Firefox's advanced consoles

80 Comment your code

Add code comments to styles and JavaScript to generate and maintain comprehensive documentation

86 Build a voice-activated app

Take your first steps with voice-activated devices and build a to-do app using API.ai and Node.js

Header

The tools, trends and news to inspire your web projects

Mozilla proposes Mixed Reality API

To bring WebAR in line with WebVR Mozilla is proposing the WebXR API. Web Designer investigates...

The web is changing. Nothing new there, as it's constantly evolving, but VR and AR are both new technologies that the web is slowly adopting. The possibilities that virtual reality and augmented reality could bring to the web has got Web Designer the most excited it has been in a while. But the technologies are in their infancy and have a long to go before they can truly fulfil their potential. And this applies even more so on the web.

Mozilla's A-Frame (aframe.io) and Jerome Etienne's AR.js library (bit.ly/GH-ARjs) start the WebVR and WebAR revolution. Both offer easy ways to start creating and experimenting with the possibilities that the technologies can bring to the web without the obstacles of third-party plug-ins. For WebVR/AR to take off, developers need an easy gateway to the end product. We don't want to be saddled with the next version of Flash when the browser can do the hard work for us.

Up steps Mozilla to help. It recently announced "a new development program for Mixed Reality that will significantly expand its work in Virtual Reality (VR) and Augmented Reality (AR) for the web. Our initial focus will be on how to get devices, headsets, frameworks and toolsets to work together, so web developers can choose

technologies evolve together. Mozilla has said that it's going to "work on the full continuum of specifications, browser implementations, and services required to create open VR and AR web experiences".

The WebVR API has already been shipped in Firefox and Microsoft Edge, but as you may have guessed by the title, this

“For WebVR/AR to take off, developers need an easy gateway to the end product”

from a variety of tools and publishing methods to bring new immersive experiences online – and have them work together in a fully functional way."

As Web Designer has mentioned before, WebVR is ahead of WebAR and this is something that Mozilla is looking to put right. It is going to create a Mixed Reality program to help ensure that both

concentrates on virtual reality. Mozilla is proposing a WebXR API, which will look to include both virtual and augmented reality devices. You can read more about the proposal at github.com/mozilla/webxr-api. Some of the concepts in the draft include "Supporting the potential for multiple simultaneous AR pages" and "control the render of reality inside the browser".



In-depth tutorials, expert tips, cutting-edge features, industry interviews, inspiration and opinion. Make sure to get your daily dose of creativity, design and development.

WEB DESIGNER DIGITAL EDITION

Do you want to get your hands on a digital edition of your favourite web design magazine? Head to your preferred app store - Google Play (bit.ly/2wetvGp) or iTunes (apple.co/2igtBYq) - then download, install and purchase the issue of choice from within the app.



STAT ATTACK

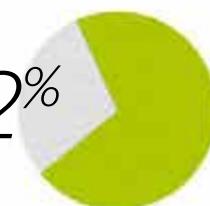
SOCIAL MEDIA - UK

Who are the big (and little) players? Are there any surprises?

Facebook

71.02%

Number one - as expected



Twitter

11.13%

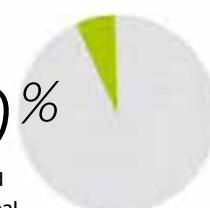
Do you have a Twitter account?



Pinterest

6.60%

The digital pinboard tops the visual appeal



Reddit

2.58%

The online community offers custom news



Instagram

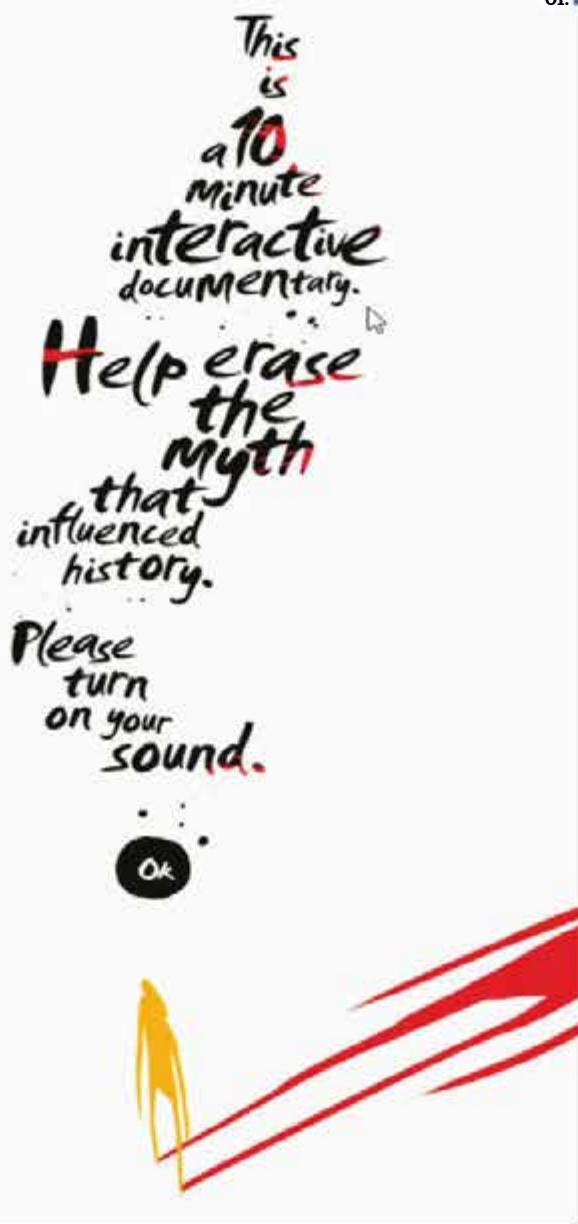
2.49%

Doubled its share in a single month

Source: <http://gs.statcounter.com>

Sites of the month

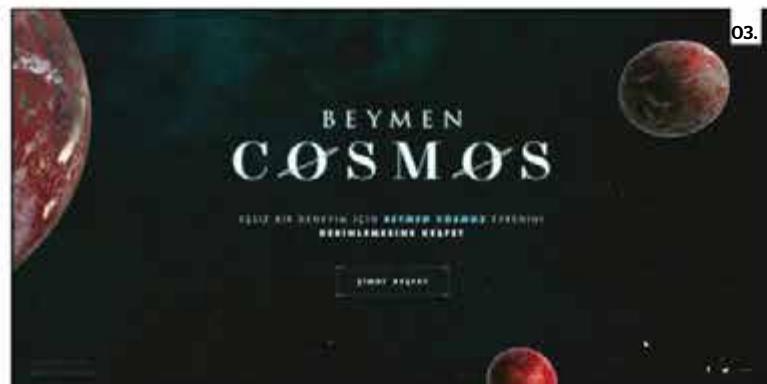
01.



02.



03.



04.



01. K'gari

www.sbs.com.au/kgari

A simple canvas interspersed with neat typography and animation

02. GUS-TRANS

gus-trans.com

This a perfect example of taking a less traditional subject and making it interesting

03. Beymen Cosmos

cosmos.beymen.com

Watch worlds go around and admire the zooming animations and more

04. Inculerate

inculerate.com

Clever user interaction using real-life graphics to enhance the story

Graphics

Nic Cester - Psycho

bit.ly/2A1vYmR

Simple but beautifully crafted illustrations from the hand of T Wei.



Colour picker Comments are back

bit.ly/2iPgbU6

#E1DFEO

#3B3A35

#95D8ED

#C1E6E9

#FFFFFF

Typesetter Heading Pro

bit.ly/2zUoJMo

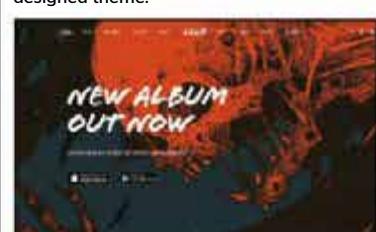
A geometric condensed typeface with high legibility. Created to optimise space on the printed page and on the screen.

ABC abc
0123456

WordPress Band

bit.ly/2lAM43y

In a band? Need a stylish site quickly? Check out the variations of this well-designed theme.



Header

Resources

CONTACT US AT: steve.jenkins@futurenet.com | @WebDesignerMag

webkit

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

DOCS BLOG GITHUB SLACK .js LIVE DEMO

TOUCHARTS

Flexible javascript charting library for data exploration.

Designed with passion and taste :)

Download latest release (0.9.1)

Star 1,533 GitHub Twitter Slack LinkedIn

Taucharts

taucharts.com

Taucharts describes itself in seven simple words: 'Flexible JavaScript charting library for data exploration'. But more tellingly, there is a subline that says 'Designed with passion and taste'. It has a clear API that can map seamlessly dataset columns to visual properties. This ensures that users will get something attractive.



ProseMirror

prosemirror.net

ProseMirror is a tool that tries to bridge the gap between Markdown text editing and classical WYSIWYG editors. It allows users to build text editors on the web.



handorgel

oncode.github.io/handorgel

It's an accordion that briefly describes itself as 'Accessible W3C conform accordion written in ES6.' It offers a host of configurations.



date-fns

date-fns.org

Date-fns is a comprehensive, yet simple and consistent toolset for manipulating JavaScript dates in a browser and Node.js. It has over 35 locales available.

TOP 5 Web conferences - Dec 2017 - Jan 2018

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



dotJS

www.dotjs.io

Tagged as The largest JavaScript conference in Europe. Come and see Brendan Eich speak.



Decompress

bit.ly/2znThdp

Decompress is a community day for designers and developers. It is run by people that run CSSConf and JSConf.



Front

www.frontutah.com

Inspiration and training for UX designers and product managers. In January there's a two-day product bootcamp.



Agent Conf

www.agent.sh

Experts and industry leaders come together to showcase their work in ReactJS, React Native and more.



Agile content conf

bit.ly/2huQ8xP

Two days: one for a conference, one for workshops. It's all about finding content solutions.

WordPress: time for change

MakeDo founder extols the virtues of Gutenberg, the new editor for the CMS



Kimb Jones

WordPress speaker and founder of Make Do
www.makedo.net

“The new Gutenberg editor aims to bring everything together in a more modern and easy-to-use layout system”

The thing about open source development is that it's much like natural evolution: it evolves and changes in the environment. Change is never easy, but change is necessary and while the current WordPress Editor is simple, relatively easy and painless - it's also basic and a bit boring. It's become established... or in other words, it's become really old.

Little has changed in a decade. We've got used to its idiosyncrasies and we're now blind to its shortcomings. Anyone who has edited content on Squarespace or Medium will have found that the user experience is so much more smooth and intuitive. Thankfully, the WordPress Community has caught onto this and while it's taken many months of development, we're now seeing WordPress roll out its editor changes. The new Gutenberg editor aims to bring everything together in a more modern and easy-to-use layout system. It promises to be "effortless" says Matt Mullenweg, who's not only the project lead but also the co-founder of WordPress.

Blocks are central to the new editor and will make it easy to create rich posts, bypassing things like shortcodes and custom HTML that we've all become accustomed to but were always a bit of a 'hack' anyway. There will be blocks for images and texts and these will be central to the new editing workflow, so getting your head around blocks is perhaps the first place to start.

WordPress knew it needed to add new UIs and revisit existing UIs. It also knew that it had to free up as much space as it could and minimise where it could. In January, Mullenweg and the core team drew up a wish-list and began important R&D.

If you're keen to see what it looks like, you can actually download a plugin for any self-hosted WordPress and install Gutenberg and have a play around. The 'framework' for the new editor went live in June's 4.8 WordPress update, but a word of warning - only use it for testing, not on a live site, until all the glitches are ironed out.

What I really love about the change is how you can do advanced mock-ups with a cleaner look and how the enhancements to the UI make it a much more enjoyable experience. Controls appear as you hover over them and the UI is now somewhat like the Medium.com experience for creating content, but with more control and expandability. Minimal controls appear as you edit and it's pretty easy to convert content into different types of text blocks such as headings and lists.

The other big plus is that there's cleaner code at other end. No more shortcodes or proprietary formatting. The aforementioned Blocks system ensures that code is totally 'portable' and free of any dependencies. We also like how videos auto-embed and can be dragged and dropped around. You can upload images and move them and create multi-column layouts. It's incredibly simple to add a gallery or set of product images.

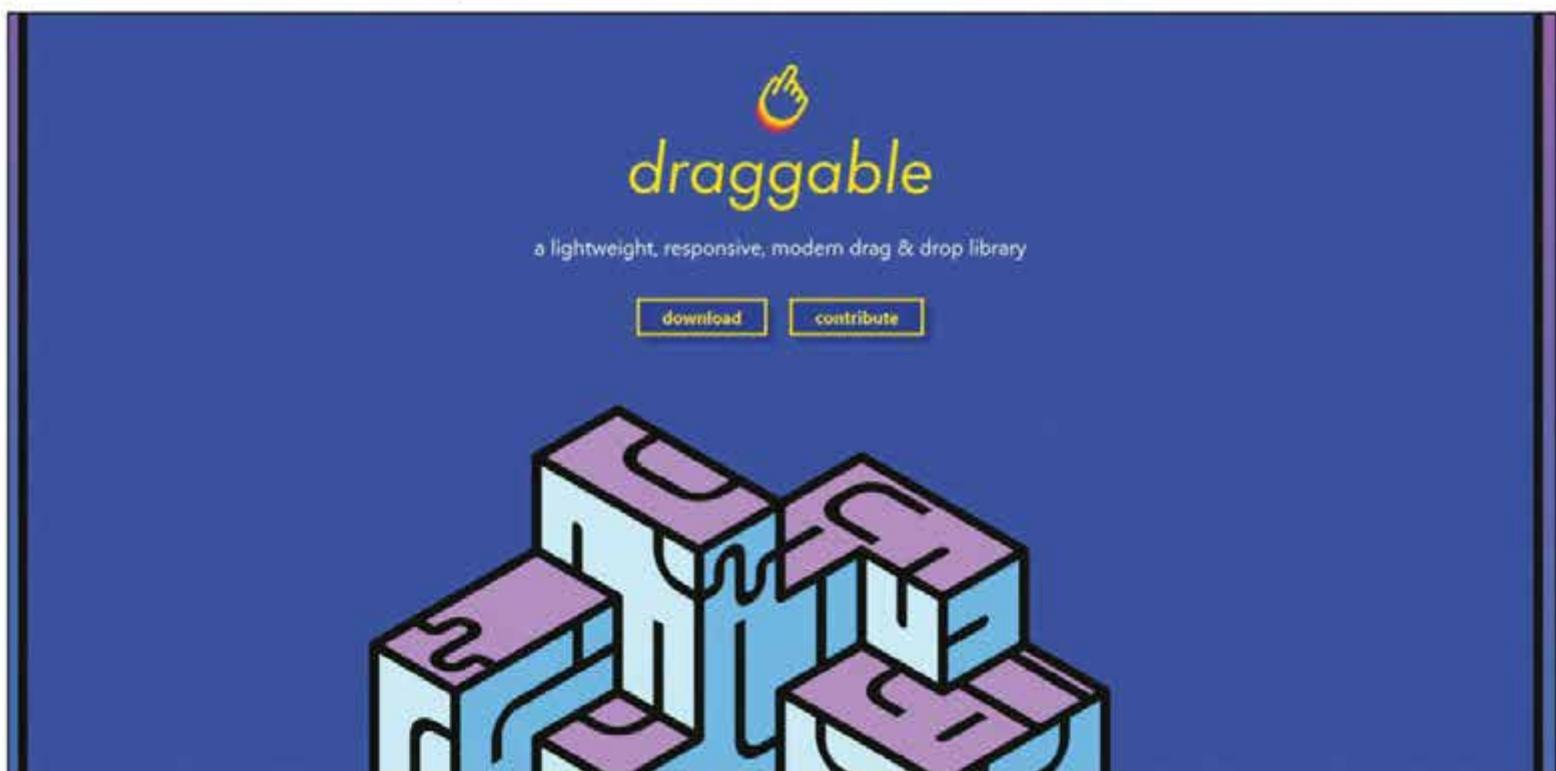
Widgets can also be dropped in for extra flexibility and although we're trying to get away from shortcodes, it's going to be possible to drop these in just like before.

As Gutenberg develops further, these built-in blocks will come 'out of the box'. The idea is that developers will be able to create their own and build a new editing experience that uses 'Blocks' rather than relying on plugins like ACF or Visual Composer.

There's been a lot of scaremongering about Gutenberg, but Editor has needed to change for a long time. We can't continue using it the way we are. Years of hacking the editing experience has gotten messy and means there's a hundreds ways to do very similar things. It might take five or even ten years to root out the bad practice we've introduced, but we must embrace change and keep up with evolution, documentation, discussion and testing. Gutenberg is a work in progress - it's not going anywhere, it's just evolving and adapting.

webkit

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



draggable

shopify.github.io/draggable

There is a clue in the name: draggable is a lightweight, responsive, modern drag-and-drop library brought to you by the good people at Shopify. The library is modular and allows users to start small and build in features as needed. So what does it do? Users can create draggable components, swap elements, drag items to sort and employ collision detection.



Khroma

khroma.co

Currently in beta, this is 'The AI color tool for designers'. Users choose a set of 50 colours and 'train' an algorithm to generate the shades they like.



Wallaby

wallabyjs.com

This is a no-nonsense real-time testing tool for JavaScript. It runs tests immediately as you type and displays execution results in your code editor.



Vuera

github.com/akxcv/vuera

Do you have an irresistible urge, or reason, to combine Vue and React? Then head over to this GitHub repository to help you seamlessly integrate the two.

TOP 5 WordPress themes for December 2017

Need to get yourself a good-looking site fast? Check out these smart themes



LawKing

bit.ly/2z32CXo

This theme is aimed at lawyers, but the layout is smooth and well designed and makes it ideal for any type of company.



Venda

vendastudio.info/venda

Smart, sassy and clean theme that focuses on the personal. Big images, contemporary fonts and great hierarchy.



Apress

apressthemes.com

Not sure what you want? This theme offers endless customisation options, giving what you want when you want it.



Rigid

bit.ly/2z4Hqjl

Into fashion and got something to sell? This eCommerce theme has multiple landing pages, all with class and on trend.



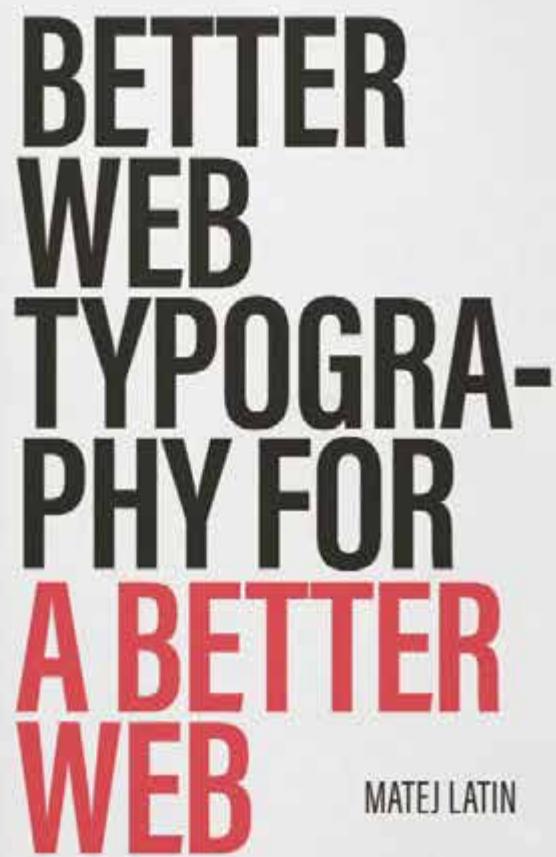
Paragon Lite

bit.ly/2gZTJH4

This theme is for creatives/designers looking to get a portfolio together quick. Neat and straight to the point.

WIN

A COPY OF BETTER WEB
TYPOGRAPHY FOR A BETTER WEB



5 BOOK GIVEAWAY

Better Web Typography for a Better Web is a book based on a top-rated online course explaining typography to people who build websites – web designers and developers. The author, Matej Latin, takes complex concepts such as vertical rhythm, modular scale and page composition, and explains them in an easy-to-understand way.

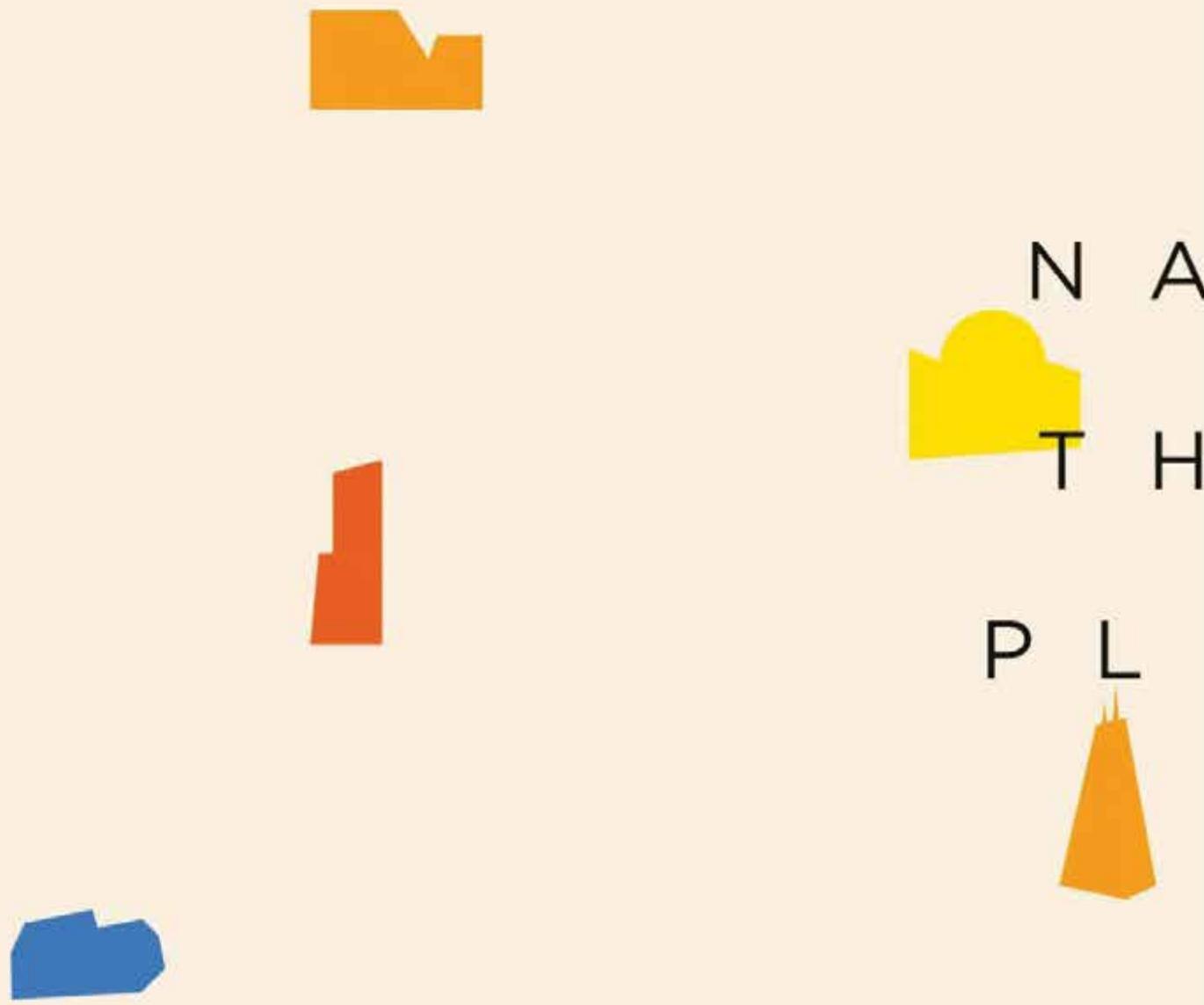
This is a new typography book for a new medium; the rules haven't changed much – everything else has. We have one printed copy and four digital copies to give away. So, get entering now.



HOW TO ENTER

All you need to do is answer one simple question.
Head over to bit.ly/2xCUMjf and choose the correct answer.
It really is that easy.

*TERMS AND CONDITIONS: The competition is open to UK entrants only. Under 18s must obtain parental consent to enter this competition and be able to demonstrate this to Web Designer's reasonable satisfaction. Answers must be received by 14/12/2017. The winners will be selected at random from all correct entries received and will be sent the prize free of charge. For full terms and conditions, please go <http://www.futureplc.com/competition-rules/>.



Scroll ↑

Name That Place Quiz

namethatplace.us

Designer: Zero Studios - zero.nyc

Development technologies jQuery, Google Maps API, MorphSVGPlugin (GSAP), Typography

“Promoting the Foursquare Swarm mobile app, this interactive site serves up multiple-choice quizzes on six prominent US cities”



#F9ECDB



#F16C1B



#FEE32E



#2182EB

abcABC abcABC
1234567890 1234567890

Above

Gotham font in both Light and Regular typefaces is used extensively across the site

LightBox

Name That Place Quiz

swarm.



Let's get started. Select a City.

New York

Los Angeles

Chicago

San Francisco

Atlanta

National

Above

Flatly coloured, animated SVG shapes are used as motifs for masking landmarks associated with each city

QUESTION 1
This Art Deco icon was the world's tallest building for a mere 11 months, in 1930-31.

Empire State Building New York Life Building
Chrysler Building White Tower

Take the Quiz
Download the App

Share Share

Privacy Terms Cookies
Copyright Facebook 2012 • See by Law

Above

Featuring a split-screen layout throughout, the timed quiz sections give participants four answer buttons to click

Above

With the embedding of Facebook SDK and Twitter Platform, visitors can follow links to the app download or share the experience socially

How to implement a scrolling jigsaw effect

Add an effect that animates individual elements in response to the user scrolling the webpage

1. Initiate the HTML document

The first step of the project is to initiate the HTML document. This consists of definitions of the HTML document container, which contains a head and body section. While the head section is primarily used to reference external CSS and JavaScript resources, the body section is used to store the visible HTML content.

```
<!DOCTYPE html>
<html>
<head>
  <title>Animated Scrolling Shapes</title>
  <link rel="stylesheet" type="text/css"
  href="styles.css" />
  <script src="code.js" type="text/
  javascript"></script>
</head>
<body>
  *** STEP 2 HERE
</body>
</html>
```

2. Body content

The main content consists of a container that stores the elements that are to be animated. For this example, these are span elements that present individual numbers, with styling being defined later in the CSS. You can change them to another element type such as img elements for your own project.

```
<article class="jigsawAnim">
  <span>1</span>
  <span>2</span>
  <span>3</span>
  <span>4</span>
  <span>5</span>
  <span>6</span>
  <span>7</span>
  <span>8</span>
</article>
```

3. JavaScript styling

With the HTML now complete, create a new file called **code.js**. The first part of this code is responsible for attaching some style properties to each element inside the container defined in step 2. Take note of how this is all executed inside a function applied to a 'load' event listener of the window; the code will fail if it executes before the page has fully loaded.

```
window.addEventListener("load",function(){
  var cssRef = ".jigsawAnim > *";
  var nodes = document.
  querySelectorAll(cssRef);
```

```
    for(var i=0; i<nodes.length; i++){
      nodes[i].style.top = "0";
      nodes[i].setAttribute("data-speed",Math.
      floor(Math.random()*10)+2);
    }
    *** STEP 4 HERE
});
```

4. Scroll interactions

The explosion effect occurs in response to the user scrolling the page, hence the requirement to define these visual changes via JavaScript. This step applies a 'scroll' event listener to the window, upon which will update the rotation, position and opacity of elements referenced in step 3 with new calculations based on the page window scroll position.

```
window.addEventListener("scroll",functi
on(){
  var nodes = document.
  querySelectorAll(cssRef);
  for(var i=0; i<nodes.length; i++){
    var speed = window.scrollY/
    parseInt(nodes[i].getAttribute("data-
    speed"));
    nodes[i].style.transform =
    "rotate("+speed+"deg)";
    nodes[i].style.top = speed+"px";
    nodes[i].style.opacity = 1-(speed/100);
    if(i >= (nodes.length-2)/2)nodes[i].
    style.left = speed+"px";
    else nodes[i].style.left =
    "-"+speed+"px";
  }
});
```

5. CSS initiation

With the JavaScript code now complete, the next step is to initiate the CSS stylesheet; create a new file called **styles.css**. The first rule sets the page size to a minimum of four times the height of the browser screen to guarantee scrolling - which is required for the effect to work.

```
html,body{
  min-height: 400vh;
}
```

6. General jigsaw style

The jigsaw consists of two types of element: the parent container and its children. The parent is set with fixed positioning in relation to the browser window now, not the page document. The children use relative positioning,

meaning that their positioning alterations are made in relation to their parent. The result is that jigsaw elements don't move out of view as the page scrolls.

```
.jigsawAnim{
  position: fixed;
  text-align: center;
}
.jigsawAnim > *{
  position: relative;
  display: inline-block;
  top: 0;
  font-size: 5em;
  padding: .5em;
}
```

7. Unique jigsaw styles

The final step is to define unique styles of the individual children of the jigsaw container. These can be individually referenced using the nth-child selector. This example references every 4th item +1, 2, 3 or 4 so that the rules repeat for every fourth item; you could also change these to a specific child number. Consider how the background attribute could be changed - for example, to a bitmap or SVG image.

```
.jigsawAnim > *:nth-child(4n+1){
  background: red;
}
.jigsawAnim > *:nth-child(4n+2){
  background: green;
}
.jigsawAnim > *:nth-child(4n+3){
  background: blue;
}
.jigsawAnim > *:nth-child(4n+4){
  background: purple;
}
***
```



COCONUT,
CRANBERRIES, WHITE
CHOCOLATE AND
DARK CHOCOLATE
60%

19 DKK

ADD TO BOX



Simply Chocolate

simplychocolate.dk

Designer: Spring/Summer - springsummer.dk

Development technologies WordPress, WooCommerce, jQuery, Google Fonts

“Blending eCommerce capabilities with a stylish scrolling brochure, this home for Copenhagen’s Simply Chocolate is a confidently tasteful experience”



NEXT PREV

#8FD5CO

#F8F8F4

#832F39

#545E66

ABCABC 1234

Above
Social Gothic designed by Patrick Griffin for Canada Type provides the bold DemiBold text styling

abcABC 1234

Above
Libre Baskerville from Impallari Type and available via Google Fonts is also used across paragraph text

**Above**

Doubling as an eShop, the site encourages purchases with a button for adding selected bars to the 'Your Box' shopping cart

WE MAKE
ALL NATURAL
CHOCOLATE
BARS.

HEAD ABOUT COPENHAGEN CHOCOLATE FACTORY

SALTY FRED

Above

Product colours are used to theme each colourful section, while big bold typography provides an immediate impact

**Above**

The bold but minimalistic design reforms responsively to look and perform identically on smaller screens

Code a hovering image with matching background colour

Create designs with complementary colours for the foreground and background.

1. HTML document template

The first step is to create the HTML definition of the webpage, which consists of the HTML document container that stores the head and body sections. While the head section is used to load external JavaScript and CSS files, the body section is used to store the visible HTML content created in step 2.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="description" content="description">
<title>Image Flip</title>
<link rel="stylesheet" type="text/css" href="styles.css"/>
<script src="code.js"></script>
</head>
<body>
    *** STEP 2 HERE
</body>
</html>
```

2. Visible content

The visible content consists of a navigation and an article container that stores the individual sections. Each section has a unique ID attribute and a child h1 element for its title display. The navigation container stores visible icons for the user to select - their interactivity will be defined via JavaScript, hence no href attribute.

```
<nav>
    <a>#9650;</a>
    <a>#9660;</a>
</nav>
<article>
    <section id="partA">
        <h1>Section A</h1>
    </section>
    <section id="partB">
        <h1>Section B</h1>
    </section>
    <section id="partC">
        <h1>Section C</h1>
    </section>
</article>
```

3. JavaScript initiation

With the HTML now complete, create a new file called **code.js** to store the JavaScript. This step waits for the

page to load before finding each of the content sections. Upon this, the section ID is stored in an array called 'sectionList' and a 'span' element is added to the section - for use later by CSS.

```
var sectionIndex = 0;
var sectionList = [];
window.addEventListener("load", function(){
    var nodes = document.querySelectorAll("section[id]");
    for(var i=0; i<nodes.length; i++){
        sectionList.push(nodes[i].getAttribute("id"));
        nodes[i].appendChild( document.createElement("span") );
    }
    *** STEP 4 HERE
});
```

4. Navigation control

This step applies event listeners to each of the navigation icons created in step 2. These event listeners wait for the icons to be clicked - upon which, the 'sectionList' array created in step 3 is checked for the next section ID to navigate to. This also sets the requirement for sections to have a unique ID attribute if they are to be navigated to from the navigation.

```
var node = document.querySelector("nav a:nth-child(1)").addEventListener("click", function(){
    if(sectionIndex > 0)sectionIndex--;
    window.location = "#" + sectionList[sectionIndex];
});
document.querySelector("nav a:nth-child(2)").addEventListener("click", function(){
    if(sectionIndex < sectionList.length-1)
    sectionIndex++;
    window.location = "#" + sectionList[sectionIndex];
});
```

5. Container styling

With the JavaScript complete, create a new file called **styles.css**. The first rules of the stylesheet define the default presentation for all types of containers. All containers are set to cover the full size of the browser screen, with no overflow or padding. Sections are set to use absolute positioning with a covering background size and positioned out of view by default; a transition is applied to animate all changes over one second.

```
html, body, article, section{
    display: block;
    width: 100%;
    height: 100%;
    padding: 0;
    margin: 0;
    overflow: hidden; }

section{
    position: absolute;
    background-size: cover;
    top: -100%;
    opacity: 0;
    transition: all 1s; }
```

6. Last child and target

The targeted section is set to appear within full view and full opacity. Each of the 'span' elements created by the JavaScript in step 3 are also set to appear at a specified size and position. These elements are referenced using the 'last-child' selector - with these span elements always being guaranteed to be last due to them being added after the page content has loaded.

```
section:last-child,
section:target{
    top: 0;
    z-index: 9000;
    opacity: 1; }

section:last-child{
    z-index: 0; }

section > *:last-child{
    display: block;
    position: fixed;
    width: 50vw;
    height: 25vh;
    top: 35vh;
    left: 25vw; }
```

7. Navigation style

The navigation is set to appear in the top right corner of the screen. This is achieved using fixed positioning to guarantee that the navigation is always visible regardless of page scrolling. A width is applied that is the same size as the font size, to guarantee that the icons appear stacked vertically.

```
nav{
    position: fixed;
    z-index: 9001;
    top: 0;
    right: 0;
    width: 2em;
    font-size: 2em; }
```



Iceland
a magical land

Discover
the island

Places
of interest

Culture
and lifestyle

More
info and advice

Points of interest

Credits

Iceland, the land fire and

Iceland is a country of great contrasts, light and darkness, fire and ice.

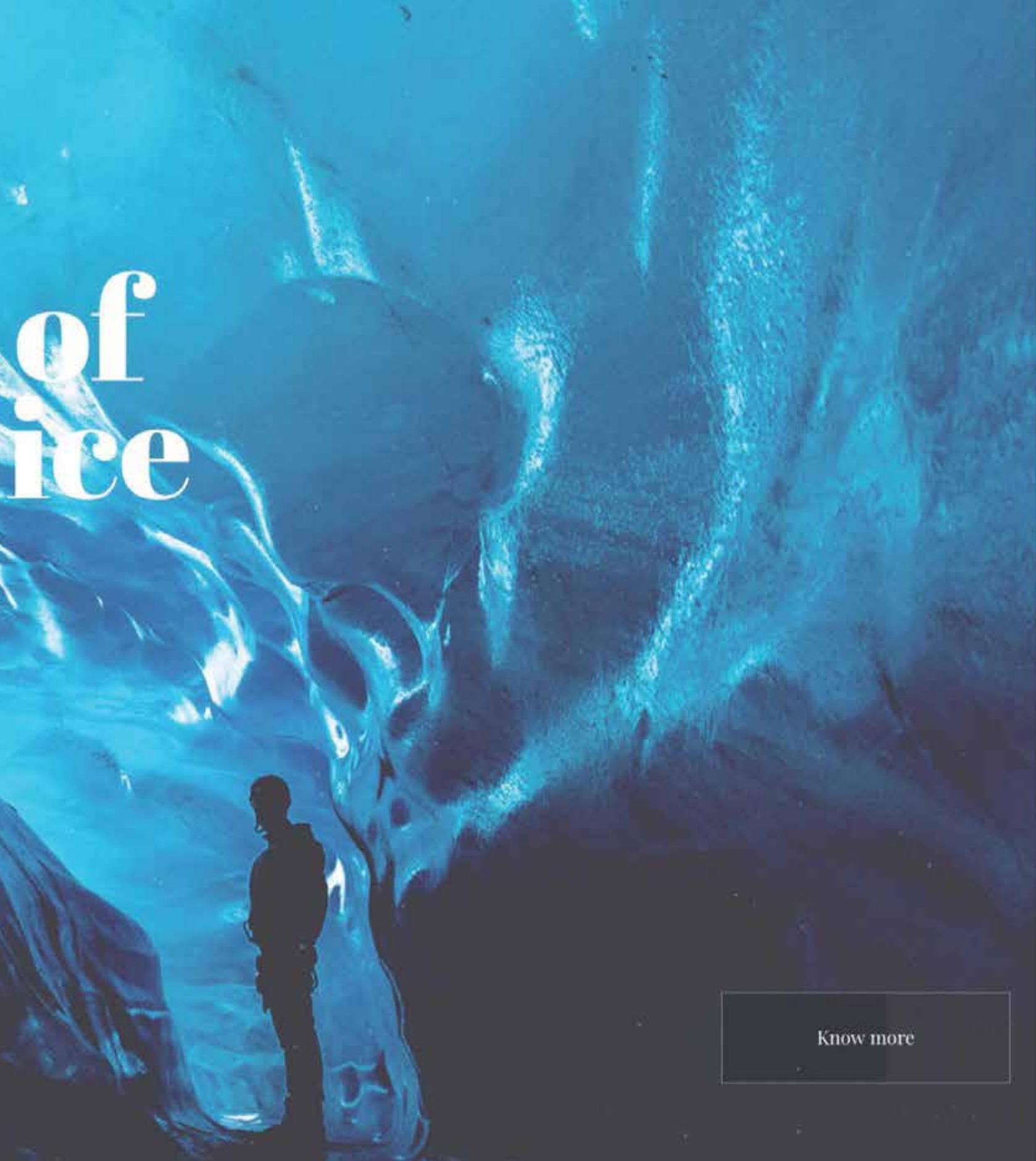
Love for Iceland

loveforiceland.com

Designer: Veintido Grados - veintidosgrados.com

Development technologies Google Maps API, WordPress, RawGit, WebGL

“Designers Veintido Grados describe this dynamic Icelandic celebration as “an elegant and simple design based on modules and lateral navigation”



of ice

[KNOW MORE](#)



#D9E7EA



#454F6B



#49DCEE



#434148

**abcABC
1234567890**

Above
Abril Fatface by TypeTogether is used in Regular form to style the animated headings

**abcABC
1234567890**

Above
Playfair Display in Regular by Claus Eggers Sørensen is employed across subsequent paragraphs of page text

Create an animated title with fading background

Allow visual elements to become the page background in response to user actions

1. Define the document

The first step is to define the HTML document. This consists of the HTML container that represents the page document, which stores the head and body sections. The head section is used to load the external CSS stylesheet, while the body section will store the visible HTML content created in steps 2 and 3.

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

2. HTML navigation

The first piece of HTML defines the page navigation. This consists of a nav container that stores a series of links to sections created in step 2. Due to these links referencing elements on the same page, it's important to make sure that href attribute values begin with a hash symbol.

```
<nav>
  <a href="#">&times;</a>
  <a href="#one">One</a>
  <a href="#two">Two</a>
  <a href="#three">Three</a>
</nav>
```

3. HTML article sections

The final part of the HTML places the individual content sections on the page. All sections are placed within an article container and have a H1 element for displaying their title. Additionally, each section has an ID attribute that corresponds to an associated link in the navigation created in step 2.

```
<article>
  <section id="one">
    <h1>Title One</h1>
  </section>
  <section id="two">
    <h1>Title Two</h1>
  </section>
  <section id="three">
    <h1>Title Three</h1>
  </section>
</article>
```

4. Stylesheet initiation

With the HTML now complete, create a new file called "styles.css". The first rule to define in this file will set defaults for the HTML document and body containers to have a set of consistent font, colours and padding. This allows child elements to cover the full width and height where required.

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

5. Section styling

Each of the sections require default styling for when they are not selected. As we don't want them to be visible, their opacity is set to zero and top position set out of view. Sizing is also set to match the browser window, while a transition is set to trigger animation on any changes made to the opacity.

```
section{
  position: absolute;
  opacity: 0;
  width: 100vw;
  height: 100vh;
  top: -100vh;
  left: 0;
  padding-top: 3em;
  background: no-repeat left center;
  background-size: contain;
  transition: opacity .5s;
}
```

6. Section titles

Each H1 element also requires a set of default rules for when its parent section is not selected. H1 elements are not visible by default; achieved by setting their height to zero with no overflow. A transition is also applied to allow changes to appear animated over a duration of one second, with a transition delay of half a second.

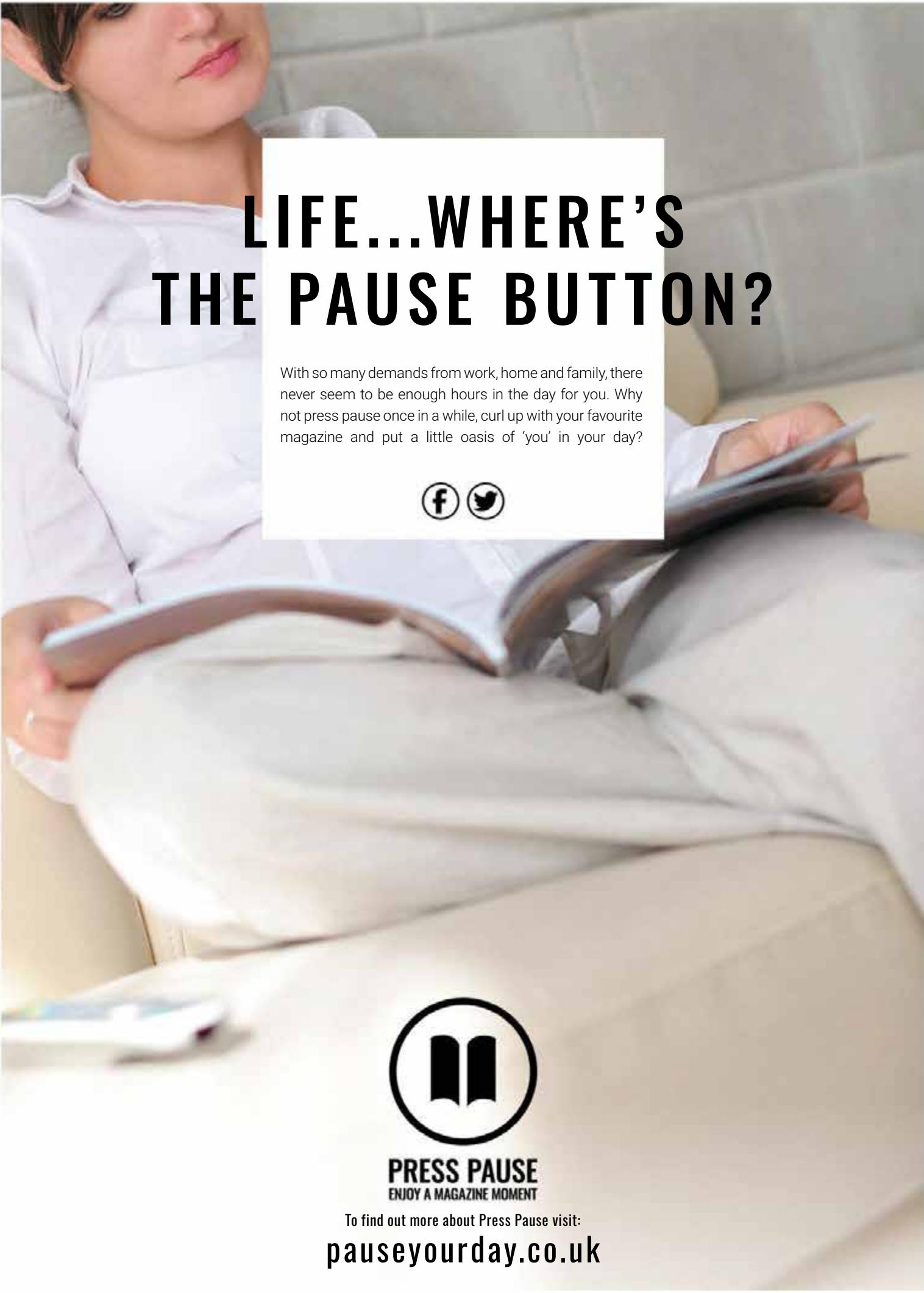
```
section h1{
  position: absolute;
  display: inline-block;
  clear: both;
  height: 0;
  overflow: hidden;
  font-size: 4em;
  transition: all 1s;
  transition-delay: .5s;
  background: rgba(0,0,0,.5); }
```

7. On target

Changes to the section containers and their H1 titles are required to be applied when they are targeted via the navigation links. The section container becomes fully visible with its opacity and positioning, while the height of the H1 title changes to the height of one character. This step also sets the background image for each section.

```
section:target{
  opacity: 1;
  z-index: 9000;
  top: 0; }
section:target h1{
  height: 1em; }
#one{ background-image: url(image1.jpg); }
#two{ background-image: url(image2.jpg); }
#three{ background-image: url(image3.jpg); }
```





LIFE...WHERE'S THE PAUSE BUTTON?

With so many demands from work, home and family, there never seem to be enough hours in the day for you. Why not press pause once in a while, curl up with your favourite magazine and put a little oasis of 'you' in your day?



PRESS PAUSE
ENJOY A MAGAZINE MOMENT

To find out more about Press Pause visit:
pauseyourday.co.uk



PLATFORM FOR SHOES

WHEN FOOTWEAR MAKER SWEAR DEMANDED AN ONLINE REBRAND COMPLETE WITH INNOVATIVE CUSTOMISATION FEATURES, THE ECOMMERCE EXPERTS TAKING IT ON WOULD REALISE IT WAS MORE THAN JUST A MATTER OF FORM...

SWEAR London

www.swear-london.com

by

Matter of Form (MOF)

<http://matterofform.com>

@matterofform

PROJECT STATS

PROJECT DURATION

4.5 months

TOTAL PROJECT HOURS

511

PEOPLE

UX Lead

- Melanie Menard

Design Director

- Anna Jehan

CEO

- Anant Sharma

Account and Project Management Lead

- Simone Strahle

One of the biggest revolutions that the modern internet age has bestowed upon us all is the enthusiasm for 'on demand' services. The two-way, quick and convenient connectivity that the web provides for eCommerce is fundamental for delivering a much more personal approach. Entranced by the rapid-fire nature of purchasing digital wares including music, films, apps and certainly television, customers increasingly strive for that experience elsewhere. Our featured website project this month exemplifies this by making customisation central to 'on demand' selling.

not of digital, but designer footwear. As a brand, London's SWEAR is a forward-thinking maker of 'sneakers' or trainers on these shores, boasting two decades within the business. It's new online shop front, built by neighbouring London agency Matter of Form, stylishly encourages buyers to personalise luxury shoes from scratch. Via the CUSTOMIZE 360 service, bespoke sneakers can be created online using innovative 3D modelling technology, then handmade ready for shipping. Existing designs from SWEAR's range can also be initialised before ordering, while the site itself oozes with a graphical energy so indicative of urban fashions. "For the creative department it was a really exciting project; to get involved with a long standing brand that was open to being bold and trying a new approach was a fantastic opportunity," begins Matter of Form's Design Director Anna Jehan. "Not only did we have a lot of freedom to experiment with visual styles, the configurator part to the project served up

"LONDON'S SWEAR IS A FORWARD-THINKING MAKER OF "SNEAKERS" OR TRAINERS ON THESE SHORES"



London footwear firm SWEAR called upon city neighbours Matter of Form to create a seamless eCommerce experience





some interesting challenges." Known as a specialist for helping luxury brands with eCommerce design work, Matter of Form (MOF) were approached to relaunch SWEAR's corporate identity without alienating an original audience going back some 20 years. "The products are a luxury item and giving the user confidence to not only design their own, but purchase online without seeing them in real life, meant we had to make it as seamless and easy as possible to build trust and show the quality of the product."

CHAMPION THE PEOPLE

The first engagement with SWEAR's rebrand challenge to MOF was very much on a 'digital first' basis. Strategically, the team would begin by examining millennial and Gen X demographics to understand their buying behaviours. This groundwork revealed some important information from prospective target customers about the trust they placed in other customisation platforms. It showed that a lack of confidence in the personalisation process often proved a barrier to purchase, but would inspire greater brand loyalty for those who did. "Our solution needed to champion the people, injecting confidence into customisation and make sure we 'handhold' customers through the journey with comforting messaging and design features through to sale," explains CEO Anant Sharma. "It would celebrate imperfection, moving the focus from the product to the notion that there is no incorrect answer, every product is a unique reflection of 'you', as is fashion itself." This last point would lead on to editorialising the facility, adding a community story with influential brand ambassadors and curator creations conveying how engaging it could be. Creatively, the two parties kicked off with meetings to also recognise the client's vision, introduce the brand properly and scope out the requirements versus any impediments. "Throughout the creative process we always try to ensure the client is involved as much as we can,"

adds Jehan. "So that there are never any unexpected curveballs, so much so our clients tend to make our office their home."

STRENGTH IN AGILITY

On a more formal level, MOF's team employed an agile approach with the client, its brand and its marketing teams. The very same methodology was also applied to dealing with Farfetch, the back-office Platforme development studio based in Porto that partnered the efforts. "All project management was sprint based, allowing us to test flows, learn and iterate fast," Sharma explains. "Our design team went to Portugal and worked closely with the team, and our strategy and design team worked through concepts with brand, marketing and eComms stakeholders." Daily meetings with

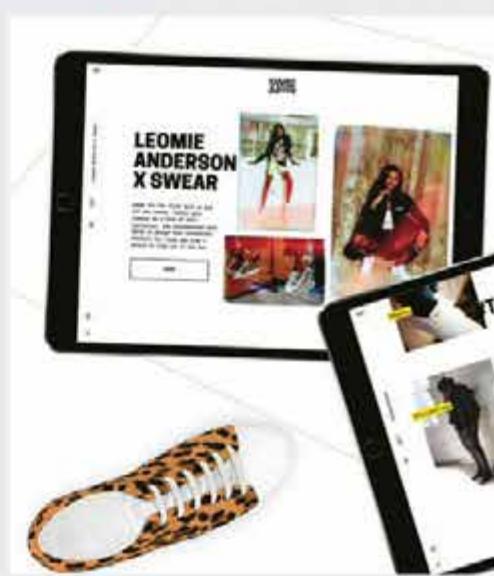
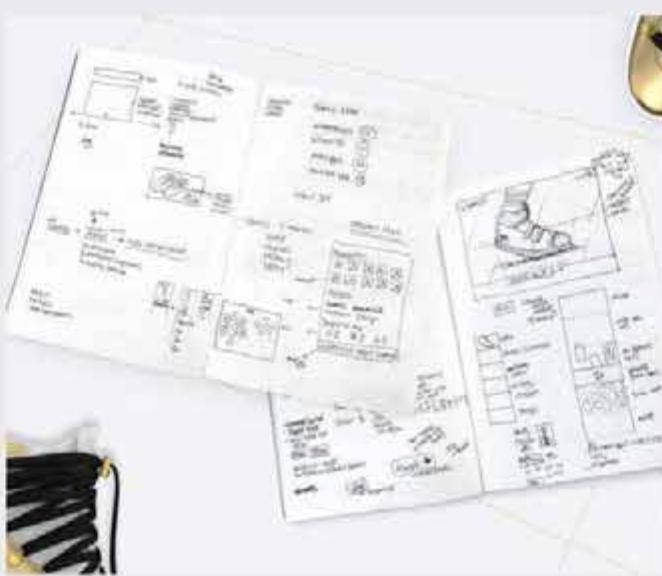
production team behind the brand which is an important aspect in understanding the foundation of the brand," elaborates Senior Account Manager Simone Strahle. "The detailed design phase was presented in a sprint phase fashion, with page designs delivered every two weeks followed by iteration and back-end integration. This allowed the client to be able to feed back in tandem with new designs being created, which allowed a flexible and agile way of working. Weekly catch-ups with the development team also ensured that functionalities were properly briefed in and that QA could be applied for each sprint."

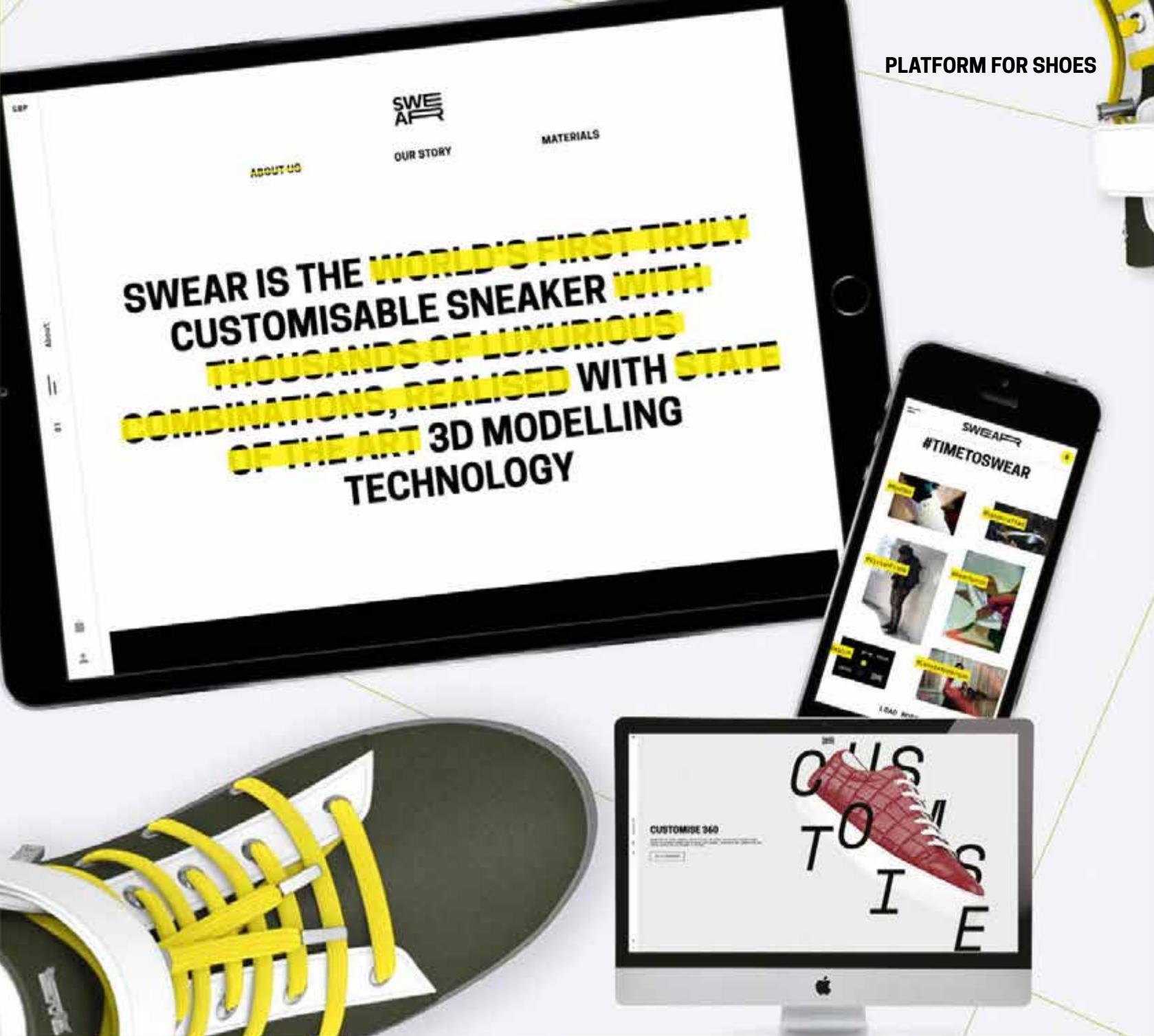
IMAGE MATTERS

When it came to dreaming up a visual direction to move on, the project benefitted from a fairly clear idea from SWEAR themselves. While harking back to its older days, the imaginative aesthetic would provide enough exciting license to throw the shackles off. "The client had a high-level initial style in mind and a direction it wanted us to explore, which stemmed from the retro visuals that came from original glitchy VHS movies," Jehan describes. "We loved the idea because it connected back to SWEAR's roots and original designs in a fun way and also

"STRATEGICALLY, THE TEAM WOULD BEGIN BY EXAMINING MILLENNIAL AND GEN X DEMOGRAPHICS"

stakeholders during project inception fleshed out the requirements gathering and early concept creation, using a kick-off trip to Porto to get a closer look. "We were taken through the 3D creation of the shoes and met the creative and





HONING INTERACTIVITY

Due to the nature of the SWEAR project, the guys at Matter of Form were conscious of the fact that robust interactivity was critical. The configurator element is central to the brand message and identity, inspiring customers to create and purchase the sneakers they desire. This made the visual narrative and interface crucial, bringing its own challenges as Director Anna Jehan explains: "There are hundreds of configurators out there online - all with varying degrees

of success - and so we created various prototypes through principle to ensure the UI and animations, from a visual point of view, would be translated correctly into build but also to ensure that the interface was clear and easy to use. This is a really useful approach and one we instil in all our projects because it also enables the client to get a good understanding of the final outcome whilst we are still experimenting and allows us to be sure when an experiment is going to

work." Additionally this approach enabled the mobile approach to the UI to be "honed" and provide an equally effective experience alongside the desktop while also differentiating it enough. Jehan added: "Our focus was to ensure that at all times the visual of the shoe could be seen as the elements were selected. This meant we broke it down into bite size chunks, guiding the user through the steps while not forcing them to change every element if they didn't want to."



SWEAR gives customers the opportunity to customise and get a unique shoe



allowed loads of room for us to play with vibrant colours, compositions and animations." The team then immersed themselves in ephemera from the 80s and 90s, referencing the best design styles from magazines, posters and film artwork too. Jehan continues: "We took the most interesting elements and played with them to see how we could modernise them by offsetting with bold modern fonts. These ideas then fed into two initial moodboard routes, one being more editorial and the other being punchier

with quirky layout twists. The route we jointly preferred with the client was the stronger bold route using yellow as a primary colour, balanced out with plenty of white space to showcase the products so it still had a clean aesthetic."

This feedback then went into homepage concept designs and the integration of layouts and product assets supplied by the SWEAR, including the CGI imagery of the shoes. The high quality of these materials was pivotal when forging the customisation animations, while some time could then be spent on realising the stylised effects. "We also spent a lot of time playing around with visual ways to present the 'glitch' style from overlaying colours on imagery, using staggered multiple versions of the image, slight animations of the glitch appearing to interrupt the flow of the product and breaking the imagery up with typography. We even fed this idea through to the way the navigation opens out in an irregular shape and the way on rollover the text is sliced through with bold brand colour overlay," Jehan revealed.

DIRECTING DEVELOPMENT

The back-end and front-end development was executed by the Platforme team in Porto, which required a tight communication strategy between teams to ensure that the designs were translated impeccably in regards to animations and transitions. This required weekly developer catch-ups and the supply of annotations and videos for directing the desired behaviours. "Seeing as the designs were slightly unconventional and in some ways pushing the boundaries on standard eCommerce features, the front-end team had to adapt and work around some of the features to enable these to be translated correctly with the same great impact as envisioned with the designs," Strahle explains. "Some of the modules and pages on the site had to be translated down to smaller breakpoints which resulted in our designers working very closely with the developers to iterate and amend even through to final stages of the build. One of these elements was, for example, the split screen on mobile that changes photos reversibly. This was hard to achieve along with the navigation as the screen real estate is limited in regard to animated features." The customisation engine itself would then present certain challenges due to back-end limitations that prevented the design from taking full-flex levels. "This meant that some of the initial wireframes had to be reworked in order to match the functionality available." Of course, all this work had to be performed within a fairly modest project time, which always brings its own problems when coordinating everyone's

POST-LAUNCH PARTNERSHIP

As is so often the case, a digital project doesn't really stop dead on completion. Matter of Form continue to work post-launch with client SWEAR on materials and promotional wares. Duties here have included the formulation of creative ads, banners, the production of artwork for any upcoming media campaigns and indeed any site revisions. "We have also developed a strong strategic partnership which involves us revisiting aspects of the site for

improvements," begins Senior Account Manager Simone Strahle. "Add to this the conducting of user-testing and analytics etc, means we will work to continuously improve the flow of the customisation journey to iron out any friction points that may be experienced by the audience." Such devotion to refining a delivered experience isn't a rare phenomenon for the MOF team, far from it, usually using such fruitful collaborations as a

springboard to broader brand relationships. A project such as SWEAR's also offers valuable opportunity to learn vital lessons in usability and marketing that can only inform an agency specialising in effective eCommerce. "Our engagements are rarely one-off pieces of project work, rather long-term journeys that look at the brand experience and corresponding digital ecosystems," admits CEO Anant Sharma. "This is especially so in the

case of a rich content and service platform like SWEAR. We can honestly learn so much about users from how they interact with the customisation engine, what materials and colour combinations resonate and which influencers drive sales. SWEAR is the ultimate example of a brand where we can use these digital insights to inform the top level marketing, merchandising and manufacturing strategy to create a truly user-centric brand."



Design Director ANNA JEHAN picks out a personal standout feature for the website and explains why it represents such a creative highlight for the project

"FOR ME THE HIGHLIGHT IS THE CONFIGURATOR ITSELF, THE CORE BRAND OFFERING. PURELY BECAUSE IT'S A PRODUCT WITH SO MANY INTRICATE ELEMENTS THAT COULD BE CHANGED BY THE USER, AND WE HAVE MANAGED TO KEEP THE INTERFACE INCREDIBLY CLEAN, ELEGANT AND REALLY SHOWCASE THE PRODUCT. THE QUALITY OF THE CGI THEN SUPPORTS THIS SO THAT THE USER CAN SEE A REALLY CLEAR REPRESENTATION OF WHAT WILL ARRIVE AT THEIR DOORSTEP."

efforts. Matter of Form were directing development based on design concepts agreed with SWEAR and would lean on certain methods to translate effectively between the phases leading up to completion. "It was a fairly tight timeline from concept designs, through to detail design and handover to the developers," admits Jehan. "However, using the prototypes we had initially created to show the developers how the designs should be translated, and using great plugins like Zeplin, allowed us to have the tools quite quickly to get into build from design and begin seeing the real output."

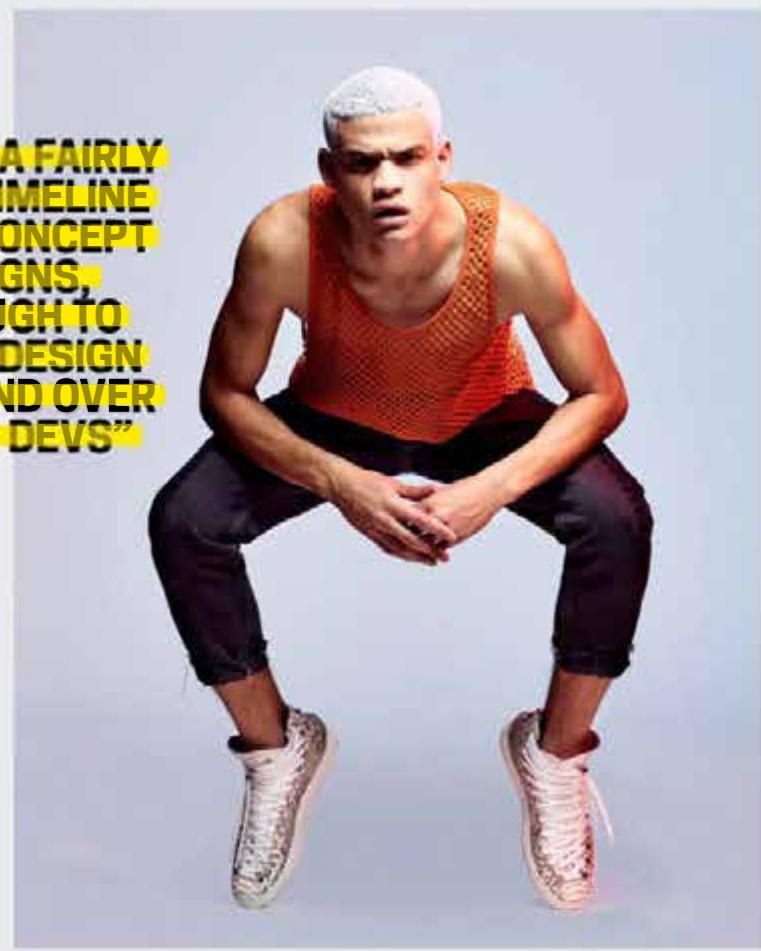
TESTING TIMES

Before anyone could think about completion and launch, a big effort went into testing the core feature. The customisation tool required extensive and thorough testing in a bid to prove a more streamlined and effective user journey with adequate visual cueing that would increase conversions and prevent choice paralysis. "This was tested with user testing groups and internal UX staff to ensure that the flow presented clear and concise steps for easier decision-making," says Strahle. "Prior to launch, the MOF team also ran a QA session on the test site that ensured that all the features were

translated and behaved in accordance to initial concept and vision. This involved looking at some of the imagery and campaign shots to ensure that the art-direction was in line with the overall message and that the right images were placed accordingly." Making sure that the site and customisation element functioned correctly was one thing, but equally important for a retail brand would be a consistent message – spreading the word online and tying the new site into how SWEAR appeared elsewhere. "To ensure the online marketing was connected visually to the site, the client also asked us to produce a series of different digital banners that showed the core message of customisation," Design Director Jehan concludes. "They were short little animations that showed the combinations of colours and materials while inviting the user to get stuck in. We kept it simple but effective." That they certainly did, which SWEAR will testify is no mean 'feet' when it comes to selling sneakers on demand. ■



"IT WAS A FAIRLY TIGHT TIMELINE FROM CONCEPT DESIGNS, THROUGH TO DETAIL DESIGN AND HAND OVER TO THE DEVS"





SAVE UP TO 49% ON A SUBSCRIPTION THIS CHRISTMAS



FROM
JUST
£13.75

FROM £13.75 EVERY 3 MONTHS (£73.40 / \$92.88 PER YR)

Biggest savings when you buy direct

Choose from a huge range of titles

Delivery included in the price

ORDER HOTLINE: 0344 848 2852

PLEASE QUOTE XMAS17 WHEN ORDERING BY PHONE

LINES ARE OPEN MONDAY - FRIDAY 8AM TO 7PM AND SATURDAY 10AM TO 2PM (GMT)

**OFFER ENDS
31 DECEMBER 2017**

You might also like...



FROM £14.75 EVERY 3 MONTHS
(€99.03 / \$121.26 PER YR)



FROM £12.65 EVERY 3 MONTHS
(€104 / \$137 PER YR)



FROM £12.70 EVERY 3 MONTHS
(€104 / \$137 PER YR)



FROM £13.70 EVERY 3 MONTHS
(€79.22 / \$100.62 PER YR)



FROM £12.65 EVERY 3 MONTHS
(€104 / \$137 PER YR)



**THE PERFECT
PRESENT FOR
EVERYONE**

SEE THE FULL RANGE AND ORDER ONLINE

myfavouritemagazines.co.uk/xmas17

***Terms and conditions:** Savings calculated against the full RRP (single issue price x frequency). Dollar prices quoted are for the United States, other global territory dollar pricing may vary. This offer is for new subscribers only. You can write to us or call us to cancel your subscription within 14 days of purchase. Your subscription is for the minimum term specified and will expire at the end of the current term. Payment is non-refundable after the 14 day cancellation period unless exceptional circumstances apply. All gift subscriptions will start with the first issue in January 2018. Your statutory rights are not affected. Prices correct at point of print and subject to change. Full details of the Direct Debit guarantee are available on request. For full terms and conditions please visit: bit.ly/magtdc. Offer ends 31st December 2017.



A LATITUDE OF DESIGN

Immersed in every digital channel, 51North are a studio that keenly understands how to communicate messages to multiple audiences via astonishingly engaging design. Able to craft each digital space, this streamlined studio are world-class exponents of next-generation experiences

Founded in 2008, Marcel Bold, Kamiel Meijers and Daan Kusters all studied Communication and Multimedia Design at Zuyd University of Applied Sciences in Maastricht. During those years' it became clear that they all wanted to start their own company. Without any experience of working for a web agency Kamiel and Marcel teamed up and started a web agency called Webspun.

Daan also started his own company, DPM Webdesign, which was more development orientated. After a few years of making Flash and HTML websites for small companies, they decided to merge into one studio, bundling their skills together to form 51North.

Soon after the merger they started working for bigger local brands and companies and the larger ad agencies in

Amsterdam. These relationships started with designing banners and EDM's (electronic direct mail) for brands like as Mercedes-Benz, Snickers and Pepsi. Over time, the jobs 51North were completing continued to grow and expand until they were developing complete marketing campaigns and corporate websites.

Kamiel Meijers, Founder and Design Director explained how they named their new studio: "The agency name definitely came first (instead of the domain name). After a lot of brainstorming and days where we would spend hours writing down catchy and not so catchy names, we eventually came back to the name we first wrote down, which was 51North.

"The name originated from the fact that the parallel latitude line that marks 51 degrees north, runs exactly through the city of Sittard, where we founded

51North and where our HQ is still situated today," Kamiel continued. "We felt it was a name that sounded good without being too pretentious or trendy. Besides that, we liked the fact that it feels like a physical place, although in reality, it is a complete parallel line running across the Earth, so it also feels really global. And it's pretty short, so that helps when it comes to remembering it and using it as a URL."

After a lot of brainstorming and days where we would spend hours writing down catchy and not so catchy names, we eventually came back to the name we first wrote down, which was 51North

Kamiel, then explained their approach to developing their own website: "We think it really is important that your website shows that you understand your game, but to be honest, it also takes a lot of time and effort to keep it up-to-date. We tend to put the effort into creating great content for our clients, which means our own site is often neglected. We do think our site should act as a calling card that illustrates who we are as a



51NORTH

WHO

51North

WHAT

Interactive design, digital concepts and experiences for mobile, web and social networks, branding, UI and UX development

WHERE

Holleweg 31A, 6131AA Sittard,
The Netherlands

WEB

<http://51north.nl>

KEY CLIENTS

- McDonald's
- TomTom International
- DDB & Tribal
- Dynamo Marketing AG
- INDIE Amsterdam

TIMELINE

2008

Marcel Bold, Kamiel Meijers and Daan Kusters found 51North in Sittard, The Netherlands.

No of employees: 3

2010

51North develop campaign sites for companies like Mercedes-Benz and Pepsi.

No of employees: 5

2013

In need of a makeover, 51North give their offices an upgrade, which turns into a major reconstruction.

No of employees: 5

team. We do need a website that is less time consuming to maintain and more efficient in showcasing our latest work! In short... we need a new website!"

51North have been steadily building their client roster, which now includes some of the world's most recognisable brands. "We don't consider our studio as established yet," said Jaap Kraan, Partner and Creative Director. "We believe that we should keep pushing ourselves and expanding the work we do before we consider 51North established in the marketplace.

"That being said, in recent years we have been in the fortunate position to pick a certain project over another. In a 51North-utopian world this would of course be our dream: Cherry picking projects all day long! But in the real world we also have our financial targets and competition from other agencies

A big project that would require a lot of programming and technical development without fun, creativity and design is not something we would take on

making great work, so it's kind of a good balance between cherry picking sometimes and proving ourselves a lot of times."

Jaap continued: "The best projects are challenging and fun to make. We have affinity with the product or subject, or is it the type of project that will give us lots of exposure, financial gain and enough time to design and develop it all. Of course, we do have projects that we will not take on. 51North focuses on projects that are creatively challenging and visually stunning. We have a strong passion for design and technical craftsmanship, combined with a strong sense for user experience and usability."

Marcel Bold, Founder and Managing Partner gave an example: "A big project that would require a lot of programming and technical development without fun, creativity and design is not something

we would take on, because even if it would be a big fish financially, it wouldn't do anything for us in terms of self-cultivation and exposure. We wouldn't be satisfied without being able to put our passion into a project and in the end, we want to be known for high-quality creative work, that inspires people and touches them."

The drive to create unique pieces of work is embodied in the Woodwork website. Marcel explained why this project encapsulates what 51North is as a design studio: "This site was one of the latest projects we had loads of fun working on. This project is a fine example of collaboration between motion studio Woodwork Amsterdam and 51North. With their expertise in motion design and our knowledge of online productions, we pushed this responsive website to the max. Up

WOODWORK

<http://woodwork.nl>

We recently teamed up with Woodwork, a Motion Design Studio, to build their new website. Woodwork has a lot of awesome content to show off, so we had to come up with a design that put the content first. We made a simple clean design that made use of different types of background colours for each piece of work which really gave it an identity.

Secondly, the flow and animations had to come close to a smooth movie experience. This proved quite a challenge with plain HTML/CSS/JS and GSAP. The motion elements were first created in After Effects and then translated onto the screen by mostly using just CSS and GSAP JS animations.

The logo animation is the only exception, because we used the bodymovin' plugin to render the animation. For the developers, timing page transitions was important to provide smooth animations and prevent lag and layout thrashing. We had to find that perfect balance between what a motion studio wants to see and what a web browser can do without lagging.



Above
The clean and simple design of the Woodwork site hides the technical prowess 51North applied to this project

2014

Agencies like BBDO, JWT, TBWA and DDB added to 51North's client list.

No of employees: 5

2015

Feeling the need to expand, 51North opens new office in Amsterdam. Jaap Kraan joins as Partner of 51North

No of employees: 7

2017

Moving into new markets, 51North becomes a partner of Adnight, Amsterdam's largest ad festival

No of employees: 9

front we agreed that it would be a showcase project, so our designers, and developers, wouldn't be too pressured by the project manager and time schedules. Everybody put a lot of love and passion in it, and we think it shows. Getting recognition in the media and from award websites always makes us proud."

In addition, Kamiel also said: "We are in an online business so we see a lot of design work from around the world. Everything we see has an influence on us. I did however, grow-up in the time just before the Internet arrived. Back then I was already intrigued by Dutch designers like De Stijl, MC Escher, Dumbar and Wim Crouwel, but also their German counterparts like Bauhaus, Mies Van Der Rohe and Dieter Rams. So those are the things that first caught my attention. After that I got a lot of



Below
Kamiel Meijers uses perhaps the most important tool any designer possesses - a pencil and paper



KAMIEL MEIJERS

Founder and Creative Director

"My personal view on creation tools is pretty literally, they are tools, just like a brush is a tool for a painter. We try not to be led by the constraints or possibilities of a tool, but instead, ask how this tool can contribute in the vision we have for the end result, whether it be design, motion, code, project management or whatever other discipline."



► inspiration from movies and later the Internet, of course, which opened up a whole new world of influential stuff and made our influences far more global."

Creating world-class content means having a workflow that results in astonishing work. Jaap outlined 51North's approach to each project: "Of course all projects have their own approach, but normally a project team consists of a Project Manager, a Creative Director, one or two designers, a technical lead and one or two supporting developers. The project manager or producer also

acts as the account manager in most of the cases.

"When the projects are bigger and more complex, we see that the UX phase is the most time consuming. UX, UI designers and developers work closely together in this phase. It's a mix of wireframing, mood-boarding, prototyping and motion designing. It's also really important for the rest of the project to get this part right. After that we continue with design, motion design, development and QA. If everything is right we go live!"

The toolsets of creative studios are often diverse. Kamil explained the

51North approach: "My personal view on creation tools is pretty literal – they are tools, just like a brush is a tool for a painter. We try not to be led by the constraints or possibilities of a tool, but instead ask how this tool can contribute in the vision we have for the end result, whether it be design, motion, code, project management or whatever other discipline."

"We are pretty Adobe centric when it comes to the creative processes like conceiving, wireframing, visual design and motion design. Our developers keep it simple and work with Sublime. Other tools include GIT and Gitflow (a breeze thanks to Sourcetree), NPM, Composer, Envoyer, Harvest and forecast, Asana. And the techniques we use include: Laravel, CraftCMS, Vanilla JS (ES2015) powered by VueJS and Laravel Mix for processing SCSS, ES2015."

Martijn Raap Developer at 51North said: "I can remember when IE6 was the standard everyone designed for. Today we can write according to the actual standards and, low and behold, it works in every browser most of the time! Of course, we keep an eye on how these standards are developing. Technologies like Canvas, SVG and WebGL allow us to do pretty much anything that would require Flash in the old days, but their implementation can be tricky and a tedious business."

"Luckily there are plenty of libraries nowadays that will help managing your canvas and its redraws, SVG XML manipulation and browser support. A good example of a library is PixiJS, which offers a single interface to draw in either WebGL or Canvas, preferring whichever technology works best for the end user's browser."

Martijn concluded: "Even though jQuery has been another great help in overcoming browser discrepancies and animation support we expect to see a decline in its use over the coming years. Thanks to JavaScript pre-processors, the ease of writing cross-browser compatible vanilla JavaScript has greatly increased. ECMAScript has also seen some awesome improvements over the last few years making it much less tedious to use and much easier to leave out the jQuery."

"We have been trying to phase it out for a year or two, and have succeeded in doing so on several projects, though not all. Its main strength lies in a great

Below
51North are a close knit team and work closely together



AGENCY BREAKDOWN

-  **UX Designer**
-  **Designer**
-  **Back-End Developer**
-  **Full Stack Developer**
-  **Front-End Developer**
-  **Project / Account manager**
-  **General Manager**



treasure trove of very useful and convenient libraries that it has acquired over time. Yet many of such libraries - or viable replacements for them - have been appearing in vanilla JavaScript, which we expect they will continue to do in the future.

"Our most recent change has been a transition to the Laravel framework in combination with Vue.js. The latter does take some getting used to, but once mastered it allows for a pretty lean front-end structure. Another noteworthy improvement would be Laravel's integration with webpack through Laravel-mix. It has greatly improved the ease of compilation and deployment of projects which also turned out to be quite a time saver. The combination of Vue.js and webpack together with GSAP for animation handling has allowed us to move away from jQuery which, in turn, allows us to write more according to ECMA standards. On the subject of GSAP it should also be said that its animation performance is truly amazing."

Creating compelling innovative digital content of course means paying attention to the social media networks. "Many of our clients use social media to generate traffic to their website," said Jaap. "A lot of the time the social media content is integrated in their websites, as well, to achieve a cross-pollinating effect. We believe that 'content is king' applies here. We think you should only use social media if you have a story to tell, not just for shouting out. We personally prefer Instagram as it is very simple, concentrates on visuals, and is capable of telling stories in pictures and movies. But we should never forget the enormous user-base Facebook has. Social media can be very effective, if you take the time to understand each network to ensure the content you are creating speaks to their users."

Building for mobile can take several directions, native, web or hybrid. What's 51North's stance on the different options and which do you think works best? Kyle Adams, Technical Lead commented: "When the smartphone as we know it was in its infancy, Google already said 'web apps are the future'. Even Steve Jobs was a web app advocate. However, these devices weren't powerful enough to really provide a good experience in their browsers. Apps did deliver. And when Android got big, devices still weren't powerful enough to provide performance and experiences on their

MENZIS - MAN, YOU'RE SO BEAUTIFUL

<https://watbenijmooi.nl>

Together with DDB and Tribal Amsterdam we created a special website where Menzis gave people the chance to personalize the 'Mens, wat ben je mooi'-video for a friend, mum, dad, neighbour or other beloved one. The 'Mens, wat ben je mooi' ('Man, You're So Beautiful') video had already been shot, created and used in an advertising campaign for Dutch

health insurance company, Menzis.

We had to come up with an efficient way to use the existing material and create personalised videos for thousands of different names. We created a editing-file which contained the movie and the original voiceover with the 'Mens' part cut out. Then we inserted an audio file where the 'Mens' part used to be, replacing it

with a name recorded by the same voiceover as the original text. To automate this process for all the custom voiceovers we created a smart AppleScript task, which finally resulted in thousands of personalised 'Man You're So Beautiful' videos.

The creatives wanted the icons to feel organic. So we had to come up with an animation technique that didn't

look like the animation itself had a beginning or ending. We created four different states for each button border. Then we morphed one state into the next by using SVG animation techniques, creating a beautiful illusion of a button that seems to be moving on its own. We used the same SVG animation techniques to draw and hide the icons inside the buttons.

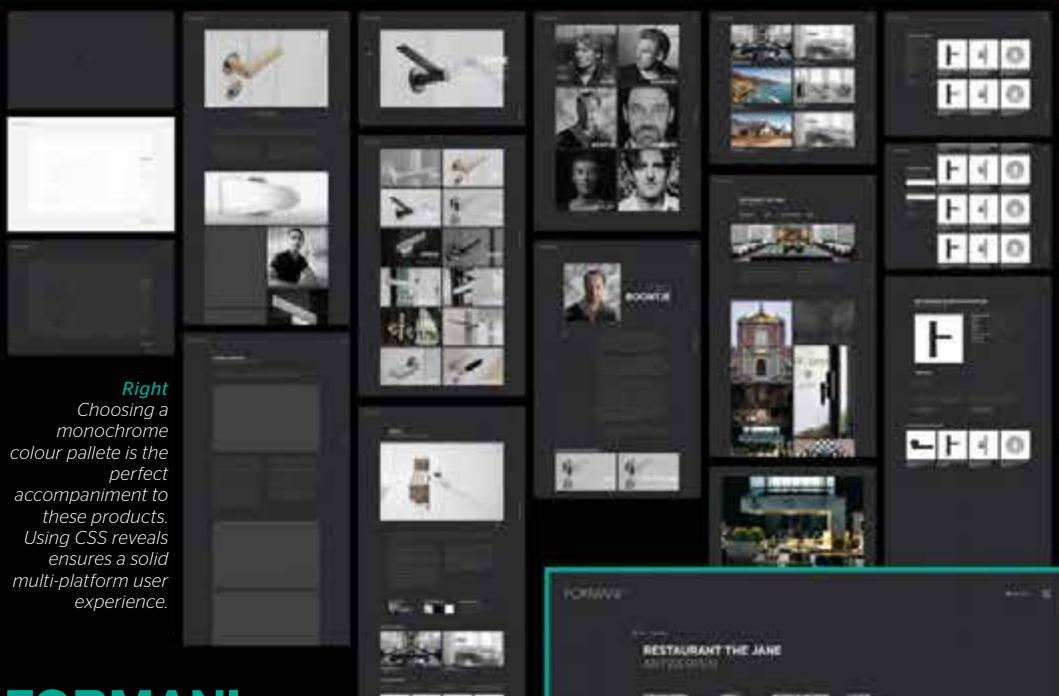


Above
Personalising video content meant 51North had to use all their technical skill to deliver a great user experience

Right + below
This site is testament to the skill that 51North possesses. Emotion is communicated with invisible code

We had to come up with an efficient way to use the existing material and create personalised videos for thousands of different names





Right
Choosing a monochrome colour palette is the perfect accompaniment to these products. Using CSS reveals ensures a solid multi-platform user experience.

FORMANI

<http://formani.nl/>

Formani is company we have been working with for years. This spring we launched their new website. Formani makes high end door fittings, which they create in collaboration with famous designers from all over the world.

Although a door fitting seems like a detail, Charles Eames ones said: 'the details are not the details, they make the design'. This quote was the starting point for creating this website. We

created a clean design with small design details that showcase these door fittings as beautiful pieces of design. With the use of clean and fluid animations we got to create a great seamless experience for users on all types of devices.

Most of the techniques used are simple CSS reveal animations triggered by scroll events. Revealing beautiful pictures against a clean minimal frame, really gives the



user a 'showcase' feeling. The page transitions are simple fade animations, became leading, instead of manipulation the DOM in multiple areas. Using the Vue transition logic with CSS animations, multiple parts of the slider can easily react to a change of slide by using the keyboard mouse, swipe gesture or pagination on both sides of the slider.

Our creatives designed a collection slider which challenged the developers to create fluid css transitions. They came up with a Vue.js powered structure, where the collection datasource



browsers that could match the native apps. And this gave hybrid application building solutions the chance to grow.

"Currently, however, we think the mobile landscape has grown to a point where mobile browsers can do a lot, because the devices have become powerful enough to provide a performance experience in a browser. This has enabled developers to build high-quality web-applications that can match the quality provided by apps built using a hybrid approach.

"So, we believe the mobile application landscape will move away from hybrid development towards the web. And that's a good thing. Native will always provide the best performance and experience. And if there is enough budget to build and maintain a native application, go for it. Just make sure you really need an app."

We believe the mobile application landscape will move away from hybrid development towards the web. And that's a good thing

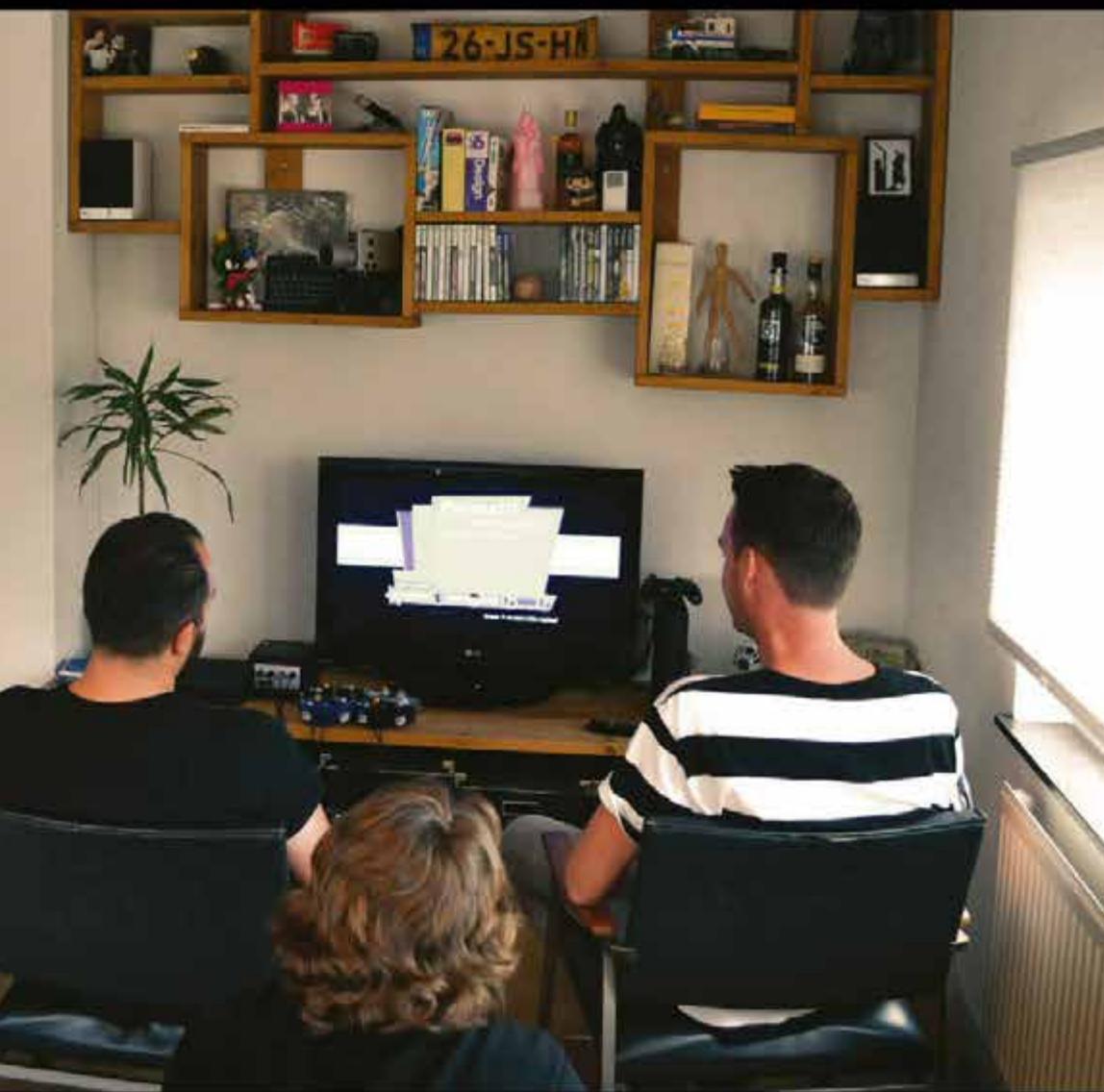
And what of Progressive Web Apps? "It's more of an approach/philosophy than a technology really," Kyle continued. "I think it can really close the gap between the native mobile application and web application in some use cases. The approach generally embodies some of the more 'best practices' when it comes to web development. Keep the page load as small as possible. Compress your assets. Load heavier things later. Keep everything as responsive and ensure it has great performance. Use HTTPS. Use our image compression algorithm that doesn't work on Firefox and Edge. And so on."

"Since we already focus on high performance, small page loads and the latest web technologies, a lot of the Progressive Web App is already present in our development approach. It's a good thing that Google is advocating these practices, but I don't think we are

going to aim for perfect scores on their PWA audit. For us, it's more about finding that balance where we can have good looks and smooth performance."

As 51North is a relatively small agency, how do they choose a new team member? Kamil outlined what they are looking for: "Fit in the team! This is the most important aspect of anyone new joining the agency. Since we're a small agency, we are not a factory. We strive to be a digital playground! It must be fun to work at 51North. Happy and curious people make great work. Ambition is also very important. We love digital explorers. Passionate about the work they do. Being able to adapt. And it helps if you like dogs, gaming and an occasional beer with your colleagues."

It's clear that 51North are on an upward trajectory. Jaap concluded by outlining what's next for the studio: "We regularly sit together to discuss where



KYLE ADAMS

Technical Lead

"Since we already focus on high performance, small page loads and the latest web technologies, a lot of the Progressive Web App is already present in our development approach. It's a good thing that Google is advocating these practices, but I don't think we are going to aim for perfect scores on their PWA audit. For us, it's more about finding that perfect balance where we can have good looks and smooth performance."

we think the agency should be going in the future. Right now, our focus is on improving our quality level. Craftsmanship, high-end, cutting edge are some of the keywords we want to be known for.

"Combining those keywords with great client service and a flexible way of working and you have a pretty awesome company in our opinion. To get there we will surely need some future additions to our team. We are working for international clients from area offices in Amsterdam and Sittard but opening up a location beyond the Dutch borders is something we would love to do."

For 51North, digital channels are their playground. With a canny understanding of current trends, they have the ability to craft new experiences that their clients need to communicate. Always pushing the boundaries, this studio is at the forefront of digital design.

51 NORTH

51north.nl

FOUNDERS

Kamiel Meijers,
Marcel Bold,
Daan Kusters,
Jaap Kraan (partner)

YEAR FOUNDED

2008

CURRENT EMPLOYEES

9

LOCATION

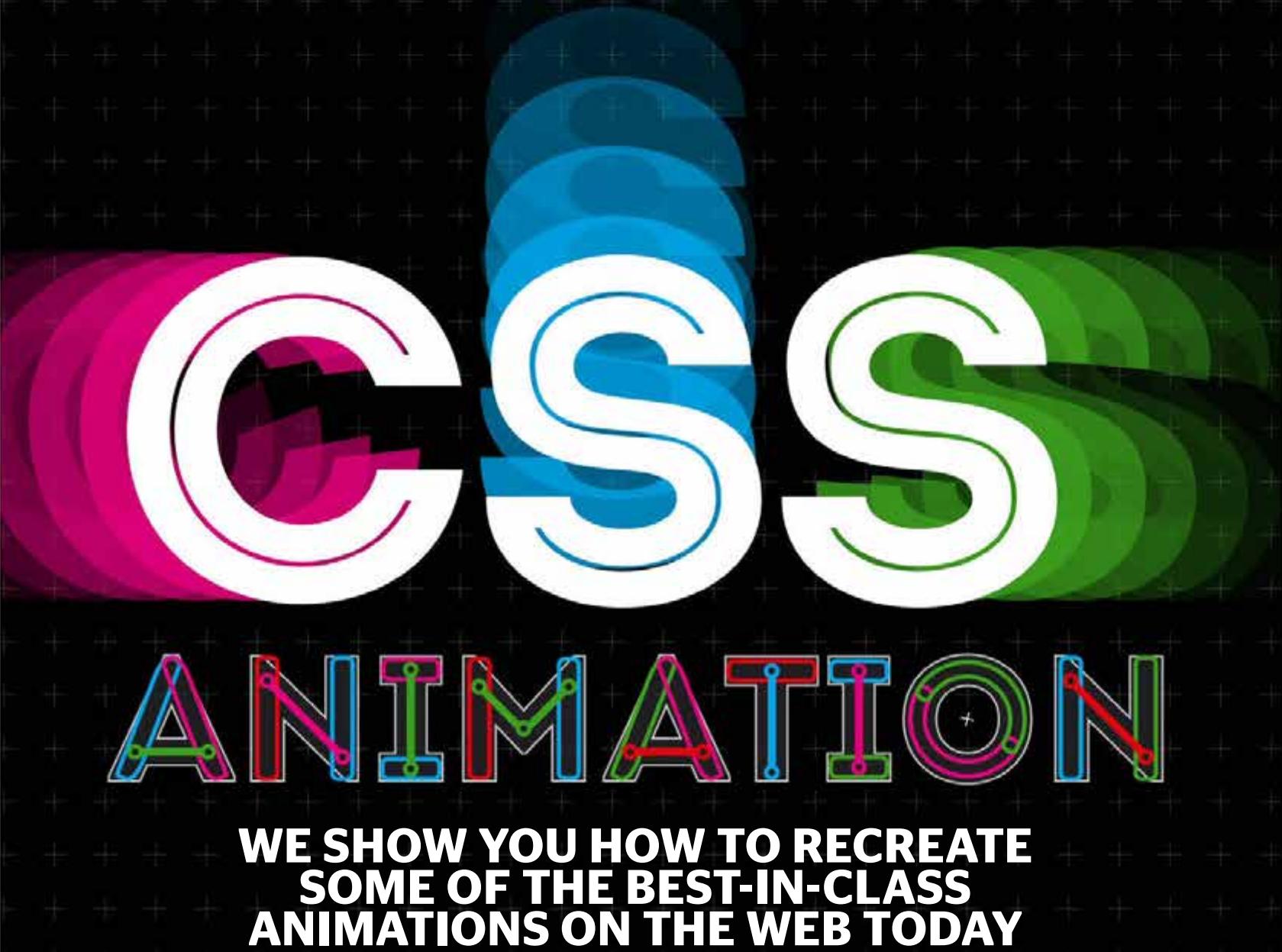
Amsterdam, Sittard,
The Netherlands

SERVICES

Digital Experiences

- Campaign and Product websites
- Digital Design
- Interactive installations
- Platform websites





**WE SHOW YOU HOW TO RECREATE
SOME OF THE BEST-IN-CLASS
ANIMATIONS ON THE WEB TODAY**

BLOWING BUBBLES

BEAUTIFULLY CRAFTED PLAYFUL BUBBLE ANIMATIONS

The 7UP (Netherlands) website is a beautiful example of carrying the brand to the design of the website. It features a number of beautifully crafted animations, but the bubbles are the most appealing. The animation consists of a few elements: the SVG 'drawing' of the bubbles and then two animations applied to each bubble. The first animation changes the opacity of the bubble and moves it vertically in the view box; the second creates the wobbling effect for added realism. The offsets are handled by targeting each bubble and applying a different animation duration and delay.

1. SVG

In order to create our bubbles we'll be

using SVG. In our SVG we create two layers of bubbles: one for the larger bubbles and one for the smaller bubbles. Inside the SVG we position all of our bubbles at the bottom of the view box.

```
<g class="bubbles-large"
stroke-width="7">
  <g transform="translate(10
940)">
    <circle cx="35" cy="35"
r="35"/>
    ...
  </g>
<g class="bubbles-small"
stroke-width="4">
  <g transform="translate(147
984)">
```

7UP NETHERLANDS www.7up.nl/7up

```
<circle cx="15" cy="15"
r="15"/>
</g>
</g>
...
</g>
```

```
<circle cx="35" cy="35"
r="35"/>
</g>
</g>
```

2. WRAP THE BUBBLES

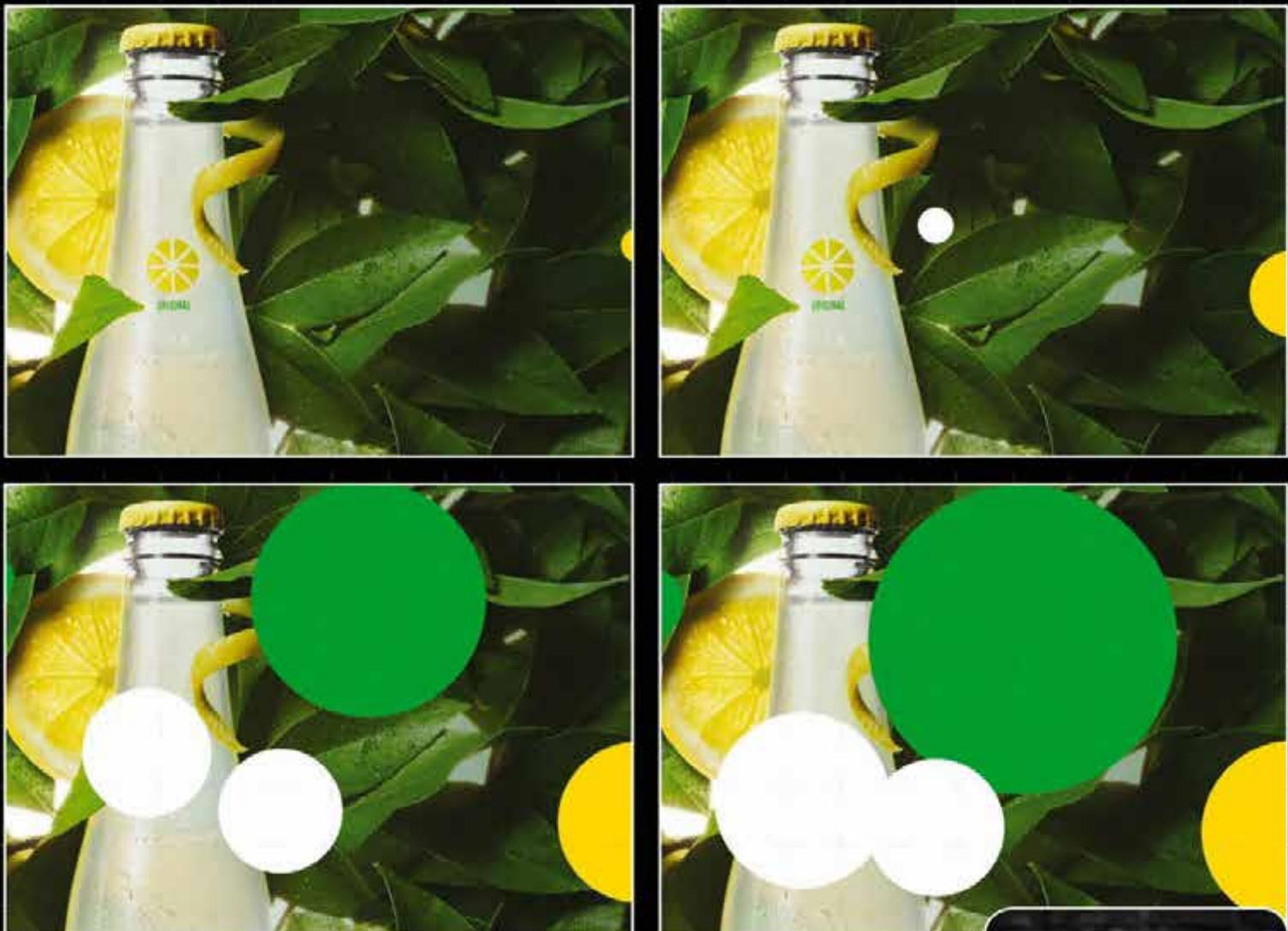
In order to apply two separate animations to our SVGs, both utilising the transform property, we need to apply the animations to separate elements. The `<g>` element in SVG can be used much like a div in HTML; we need to wrap each of our bubbles (which are already in a group) in a group tag.

```
<g>
  <g transform="translate(10
940)">
```

3. ANIMATION SETUP

CSS has a powerful animation engine and really simple code in order to produce complex animations. We'll start with moving the bubbles up the screen and changing their opacity in order to fade them in and out at the beginning and end of the animation.

```
@keyframes up {
  0% {
    opacity: 0;
  }
  10%, 90% {
    opacity: 1;
  }
```



```
100% {  
  opacity: 0;  
  transform: translateY(-1024px);  
}  
}
```

4. WIBBLE-WOBBLE

In order to create a wobbling effect, we simply need to move (or translate) the bubble left and right, by just the right amount - too much will cause the animation to look too jaunting and disconnected, while too little will go mostly unnoticed. Experimentation is key with when working with animation.

```
@keyframes wobble {  
  33% {  
    transform: translateX(-50px);  
  }  
  66% {  
    transform: translateX(50px);  
  }  
}
```

5. READYING ELEMENTS FOR THE ANIMATION

In order to apply the animation to our bubbles, we'll be using the groups we used earlier and the help of 'nth-of-type' to identify each bubble group individually. We start by applying an opacity value to the bubbles and the 'will-change' property in order to utilise hardware acceleration.

```
.bubbles-large > g {  
  opacity: 0;  
  will-change: transform, opacity;}  
.bubbles-large g:nth-of-type(1)  
{...}  
...  
.bubbles-small g:nth-of-type(10)  
{...}
```

6. APPLYING THE ANIMATIONS

We want to keep all the animation

GET THE CODE

srt.lt/PxF2R

times and delays within a couple of seconds of each other and set them to repeat infinitely. Lastly, we apply the 'ease-in-out' timing function to our wobble animation to make it look a little more natural.

```
.bubbles-large g:nth-of-type(1) {  
  animation: up 6.5s infinite; }  
.bubbles-large g:nth-of-type(1)  
circle {  
  animation: wobble 3s infinite  
  ease-in-out; }  
...  
.bubbles-small g:nth-of-type(9)  
circle {  
  animation: wobble 3s 275ms  
  infinite ease-in-out; }  
.bubbles-small g:nth-of-type(10)  
{  
  animation: up 6s 900ms infinite; }
```



STEVEN ROBERTS

Steven is the lead front-end developer at Better Brand Agency in Middlesbrough. He's been scouring the web looking for the best-in-class animations to dissect and recreate for you to learn the secrets of beautiful animation for the web.

BETTER BRAND AGENCY

www.betterbrandagency.com

EARN MORE

Apprentices can earn almost £4,000 more than graduates in their first job

[FIND OUT MORE](#)

BALTIC TRAINING baltictraining.com

Leave a message

SCROLLING MOUSE

SUBTLE ANIMATION CAN GIVE DIRECTION TO THE USER

1. CREATE THE SVG

Although this can be accomplished using HTML elements and properties, SVG is more suited to drawing. Inside our SVG we need a rectangle with rounded corners and a circle for the element we're going to animate, by using SVG we can scale the icon to any size we need.

```
<svg class="mouse" xmlns="..." viewBox="0 0 76 130" preserveAspectRatio="xMidYmid meet">
  <g fill="none" fill-rule="evenodd">
    <rect width="70" height="118" x="1.5" y="1.5" stroke="#FFF" stroke-width="3" rx="36"/>
    <circle cx="36.5" cy="31.5" r="4.5" fill="#FFF"/>
  </g>
</svg>
```

2. STYLE OUR SVG

Now we've created our SVG, we need to apply some simple styles in order to control the size and position of the icon within our container. We've wrapped a link around the mouse SVG and positioned it to the bottom of the screen.

```
.scroll-link {
  position: absolute;
  bottom: 1rem;
  left: 50%;
```

```
  transform: translateX(-50%);
}
.mouse {
  max-width: 2.5rem;
  width: 100%;
  height: auto;
}
```

3. CREATE ANIMATION

Next we'll create our animation. At 0 and 20 per cent of the way through our animation. We want to set the state of our element as it begins. By setting it to 20% of the way through, it will stay still for part of the time when repeated infinitely.

```
@keyframes scroll {
  0%, 20% {
    transform: translateY(0)
    scaleY(1);
  }
}
```

4. FINISH ANIMATION

We need to add in the opacity start point and then transform both the Y position and the vertical scale at the 100% mark, the end of our animation. The last thing we need to do is drop the opacity in order to fade out our circle.

```
@keyframes scroll {
  ...
  10% {
    opacity: 1;
  }
}
```

```
  }
  100% {
    transform: translateY(36px)
    scaleY(2);
    opacity: 0.01;
  }
}
```

5. APPLY ANIMATION

Lastly we apply the animation to the circle, along with the 'transform-origin' property and the 'will-change' property to allow hardware acceleration. The animation properties are fairly self-explanatory. The 'cubic-bezier' timing function is used to first pull the circle back before dropping it to the bottom of our mouse shape; this adds a playful feel to the animation.

```
.scroll {
  animation-name: scroll;
  animation-duration: 1.5s;
  animation-timing-function: cubic-bezier(0.650, -0.550, 0.250, 1.500);
  animation-iteration-count: infinite;
  transform-origin: 50% 20.5px;
  will-change: transform;
}
```

ALTHOUGH THIS CAN BE ACCOMPLISHED USING HTML ELEMENTS AND PROPERTIES, SVG IS MORE SUITED TO DRAWING

GET THE CODE
srt.lt/q4N8gO

MESMERISING MANDALAS



ADIDAS
adidas.co.uk/claimfreedom

PLAYFUL POP

The circular buttons on the Adidas website – when interacted with – have a playful popping animation whereby another circle covers the button growing from the centre.

This animated transition is overlapped by a fading circle around the edge, the same colour as the button's original state. This adds a sense of fun and engagement.

To achieve this we use both the ':before' and ':after' pseudo elements – one for each of our animated circles.

We absolutely position both of the elements, fixing them to the original button. In order to animate from the centre, we're also setting the transform origin to the centre of the element.

We've set the initial scale of our ':before' element to 0 so we can transition the value when interacted with. The 'transition' property performs the animation.

When we interact with our element, we simply increase the scale. However, since we've also changed the translation (another value of the 'transform' property), we need to carry that value to the new 'transform' property's value.

The outer circle is achieved using a very simple keyframe animation – transitioning the opacity.

GET THE CODE
srt.lt/Ccm5K

The original state of the mandala consists of a few simple masked elements.



The second step in the animation has a second darker circle, masked on top of the pattern beginning to expand.



As the animation progresses, the circle expands to cover the entire pattern.



At the end of the animation, as the masked circle reaches the edge of the mandala, the opacity is reduced, returning the pattern to its original colour – readying the animation to repeat.



GARDEN EIGHT
garden-eight.com

ANIMATED DRAWING

The Garden Eight website utilises a common animation technique whereby text appears to be written out. To achieve the effect, we turn to SVG. To begin with, we'll create the SVG. There are two approaches here: convert the text to paths in order to animate them or use SVG text. Both approaches have their pros and cons. We start by creating our keyframe animation. The only function we need it to perform is to change the 'stroke-dashoffset'. Now we've created our animation, we need to apply the

values we want to animate from. We set the 'stroke-dasharray', which will create gaps in the stroke. We want to set our stroke to be a large enough value to cover the entire element, finally offsetting the dash by the length of the stroke.

The magic happens when we apply our animation. By animating the offset, we're bringing the stroke into view – creating a drawing effect. We want the elements to draw one at a time, with some overlap between the end of drawing one element and beginning to draw the next. To achieve this we turn to Sass/SCSS and 'nth-of-type' to delay each letter by half the length of the animation, multiplied by the position of that particular letter.

GET THE CODE
Paths - srt.lt/Zq6m
Text - srt.lt/Sz3jB9

GOD OF WAR
godofwar.playstation.com/

Once again, to achieve this we turn to SVG creating our shapes, patterns and masks. Once we have our SVG, we target the masked darker circle with CSS and prepare it for animation:

```
.masking {
    transform: scale(0);
    transform-origin: 50% 50%;
    will-change: transform;
}
```

Next, we create our keyframe animation. We only need to change two properties here: the opacity and the scale.

```
@keyframes scale {
  0%, 50% {
    opacity: 1;
```

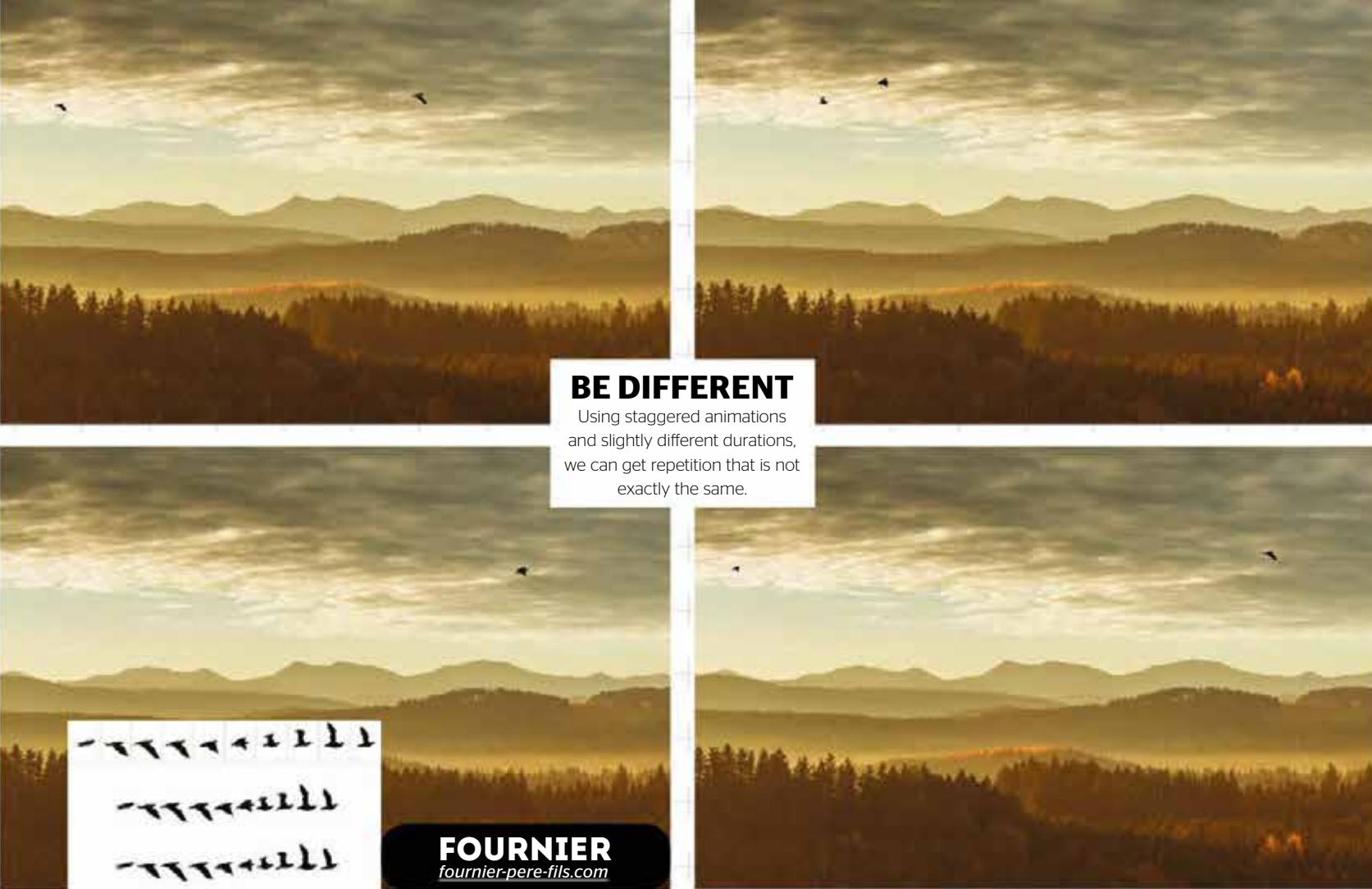
```
}
```

- 80% { opacity: 0; }
- 100% { transform: scale(1.4); opacity: 0; }

Lastly we need to apply the animation to the mask:

```
.masking {
  ...
  animation: scale 4s linear infinite;
```

GET THE CODE
One - srt.lt/pF4KOL
Two - srt.lt/yBzHj9



BE DIFFERENT

Using staggered animations and slightly different durations, we can get repetition that is not exactly the same.

FOURNIER
fournier-pere-fils.com

Above
This image shows how the bird SVG was created

ANIMATION WILL MOVE THE BIRD ACROSS THE SCREEN HORIZONTALLY WHILE ALSO CHANGING THE VERTICAL POSITION

FLYING BIRDS

RECREATE REAL-LIFE ACTIONS

1. DRAW THE BIRD

We start with completely straight vector lines, drawing each frame of our animation, depicting the bird in a different state of flight. We then manipulate the vector points and round the lines and edges. Finally, we put each frame into an equally sized box and place them side-by-side. Export the file as an SVG.

2. HTML SETUP

The HTML setup is really simple. We just need to wrap each bird in a container in order to apply multiple animations - one to make the bird fly and the other to move it across the screen.

```
<div class="bird-container">
  <div class="bird"></div>
</div>
```

3. CREATE THE FLY-CYCLE

We apply our bird SVG as the background to our bird div and choose the size we want each frame to be. We use the width to roughly calculate the new background position. The SVG has ten cells, so we multiply our width by ten and then alter the number slightly until it

looks correct.

```
.bird {
  background-image: url('bird.svg');
  background-size: auto 100%;
  width: 88px;
  height: 125px;
  will-change: background-position;
}

@keyframes fly-cycle {
  100% {
    background-position: -900px 0;
  }
}
```

4. APPLY ANIMATION

CSS animation has a couple of tricks you may not be aware of. We can use the 'animation-timing-function' to show the image in steps - much like flicking through pages in a notebook to allude to animation.

```
animation-name: fly-cycle;
animation-timing-function:
steps(10);
animation-iteration-count:
infinite;
animation-duration: 1s;
animation-delay: -0.5s;
}

Singular Bird Animation - srt.lt/nOSOq
```

5. FLY ACROSS THE SCREEN

Now we've created our fly cycle, our bird is currently flapping her wings but isn't going anywhere. In order to move her across the screen, we create another keyframe animation. This animation will move the bird across the screen horizontally while also changing the vertical position and the scale to allow the bird to meander across more realistically.

6. FINISHING TOUCHES

Once we've created our animations, we simply need to apply them. We can create multiple copies of our bird and apply different animation times and delays.

```
.bird--one {
  animation-duration: 1s;
  animation-delay: -0.5s;
}

.bird--two {
  animation-duration: 0.9s;
  animation-delay: -0.75s;
}
```

GET THE CODE
srt.lt/wQ8jEc

CSS ANIMATION LIBRARIES

GREENWICH LIBRARY

greenwichlibrary.org

SLIDING ARROW

The Greenwich Library has a really interesting transition on its buttons. When interacting with the button, two things happen: the text part of the button is covered and the arrow is then animated off the right-hand side of the button and back in from the left. The colour transition is accomplished with the transition property and the arrow using a simple keyframe animation. Both the transition and the animation use the same duration in order to synchronise the movements.

CODE - srt.lt/D5GsB

PRINCESS ALEXANDRA AUDITORIUM

thepaaonline.org

EXPANDING CORNERS

The Princess Alexandra Auditorium website has a visual way to show the categories of its shows. Each of the show cards has a triangular corner set in a colour which represents the category and then, on hover, the name of the category is displayed. The effect is accomplished using the ':before' and ':after' pseudo elements, transitioning the size of the triangle and fading the name in when the element is interacted with.

CODE - srt.lt/W1N9Mo

COULEE CREATIVE

couleecreative.com

SPINNING MENU ICON

The animated menu button is created using an SVG. The animation occurs when the user interacts with the menu button. Two transitions take place: the circular group around the menu icon in the centre changes colour. The most complicated part is the timing-function. Utilising 'cubic-bezier' to gain complete control, we're able to start the animation slowly, race through the middle part and slow it down again at the end.

CODE - srt.lt/S3Y6X

DA-INK

da-ink.com

COLOURFUL TRANSITIONS

The Da-ink website utilises a really effective technique to transition between pages. The transition is simple and consists of an SVG containing a number of different-sized rectangles of different colours positioned on top of one another. The animation consists of transforming the X position by the width of the SVG. Then, using 'nth-of-type', we apply delays, offsetting each by 75ms from the last to create a smooth transition.

CODE - srt.lt/RaEo

HEARTBEAT

heartbeat.ua

EXPANDING HIGHLIGHT

This is a very simple, yet really effective technique. The transition is accomplished using the ':before' pseudo element. To begin with, the pseudo element is placed at the bottom while spanning the full width, but only a few pixels in height. When the element is interacted with, the width and height of the pseudo element are both transitioned to 105% of the parent's size (the change is much more dramatic vertically), as well as transitioning the colour of the text.

CODE - srt.lt/cBg6e

USA TODAY

usatoday.geex-arts.com

FALLING SNOW

The snow is created using an SVG and the technique is very similar to the way we created the bubbles earlier. To start, we create two layers of circles inside an SVG, then we animate those two layers by translating the Y value with a keyframe animation. We apply the animation to each layer instead of individual elements and reuse the same animation for both layers. By simply giving them different durations, we can add some depth to our scene.

CODE - srt.lt/YjRdLO



CSSSHAKE

bit.ly/1cvZWMY

CSSShake is a library of CSS animations dedicated to making elements shake. The library has a number of classes you can add to elements to apply one of many different shake animations. The animations use keyframes and utilise the transform property to create these un-ignorable, sometimes crazy or violent-looking animations.

ANIMATE.CSS

bit.ly/1qTVdJA

This library contains no fewer than 76 animations, all created in CSS and ready to use by simply adding a class name to any element. These animations cover almost every simple animation you could need and the library is an impressively small file size when minified and gzipped.



HOVER.CSS

bit.ly/1ezWUbI

Hover.css is another CSS animation library. This library was created specifically for button interaction animations. The library can be downloaded in vanilla CSS, Sass and LESS, allowing these animations to fit into any project. The library has almost every animation you might need for interaction.



OBNOXIOUS.CSS

bit.ly/2q57Tsl

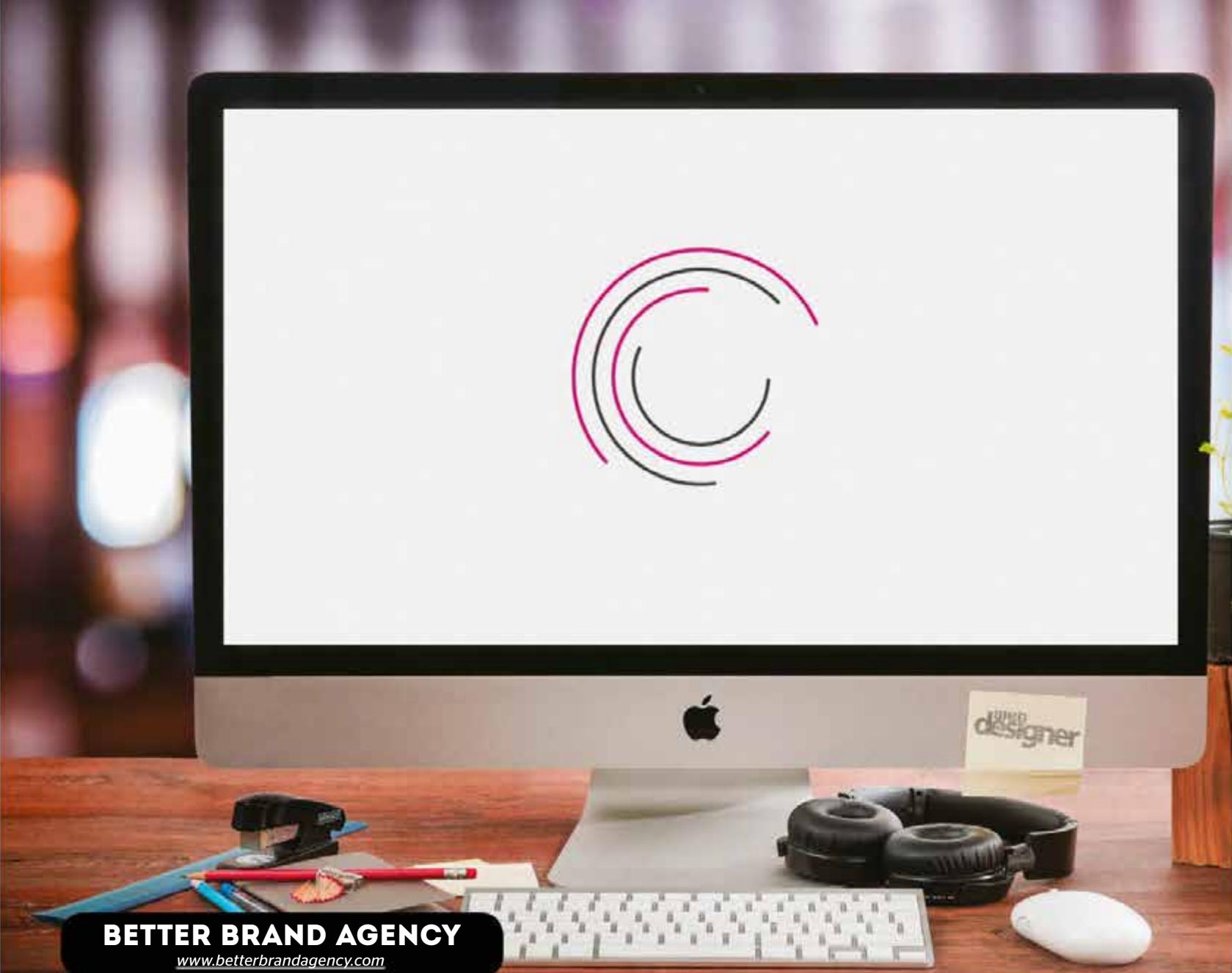
Obnoxious.css was created by Tim Holman and is exactly as the name suggests. The library explores what is possible with CSS animation, but illustrates perfectly what you should not be doing with CSS animations. While presented as a product-ready animation library (and it is), it is intended to be tongue-in-cheek.



REPAINTLESS.CSS

bit.ly/2gO7Prh

Replainless.css is a small and lightweight CSS animation library with a focus on creating animations that do not cause a repaint from the browser (so long as they're used correctly). By not requiring a reflow or repaint, these animations are super-fast and performant!



BETTER BRAND AGENCY

www.betterbrandagency.com



CHASING CIRCLES

1. CREATE THE CIRCLES

The animated loading icon is made up of four circles. The circles have no fill, but have alternating stroke-colours.

```
<svg class="loader"
  xmlns="http://www.w3.org/2000/svg" viewBox="0 0 340 340">
  <circle cx="170" cy="170" r="160" stroke="#E2007C"/>
  <circle cx="170" cy="170" r="135" stroke="#404041"/>
  <circle cx="170" cy="170" r="110" stroke="#E2007C"/>
  <circle cx="170" cy="170" r="85" stroke="#404041"/>
</svg>
```

2. SET THE DASH-ARRAYS

In our CSS, we can set some basic properties to all of our circles and then use the ':nth-of-type' selector to apply a different 'stroke-dasharray' to each circle.

```
circle:nth-of-type(1) {
  stroke-dasharray: 550;
}
circle:nth-of-type(2) {
  stroke-dasharray: 500;
}
circle:nth-of-type(3) {
  stroke-dasharray: 450;
}
circle:nth-of-type(4) {
  stroke-dasharray: 300;
}
```

3. CREATE THE ANIMATION

Next, we need to create our keyframe animation. Our animation is really simple: all we need to do is to rotate the circle by 360 degrees. By placing our transformation at the 50% mark of the animation, the circle will also rotate back to its original position.

```
@keyframes preloader {
  50% {
    transform: rotate(360deg);
  }
}
```

THIS ANIMATION IS CREATED USING SVG CIRCLE STROKE DASH-ARRAYS AND ANIMATED ROTATION

```
}
```

4. APPLYING THE ANIMATION

With our animation created, we now just need to apply it to our circles. We set the animation name, duration, iteration count and timing function. The 'ease-in-out' will give the animation a more playful feel.

```
animation-name: preloader;
animation-duration: 3s;
animation-iteration-count: infinite;
animation-timing-function: ease-in-out;
```

5. DELAY THE ANIMATIONS

At the moment, we have our loader, but all of the elements are rotating together at the same time. To fix this, we'll apply some delays. We'll create

our delays using a Sass for loop.

```
@for $i from 1 through 4 {
  &:nth-of-type(#{$i}) {
    animation-delay: #{$i * 0.15}s;
  }
}
```

6. NEGATIVE DELAY

Due to the delays, our circle now animates in turn, creating the illusion of the circles chasing each other. The only problem with this is that when the page first loads, the circles are static, then they start to move, one at a time. We can achieve the same offset effect, but stop the unwanted pause in our animation by simply setting the delays to a negative value.

```
animation-delay: -#{$i * 0.15}s;
```

GET THE CODE
srt.lt/S6YrMm



BETTER BRAND AGENCY

bit.ly/2IKSsnl

CROSS MY HAMBURGER

This animation is used all over the web, turning three lines into a cross or close icon. Until fairly recently, the majority of implementations have been achieved using HTML elements, but actually SVG is much more suited to this kind of animation – there's no longer a need to bloat your buttons code with multiple spans. Due to the animatable nature and SVG and its navigable DOM, the code to accomplish the animation or transition changes very little – the technique is the same.

We start by creating four elements, be it spans inside of a div or paths inside of an SVG. If we're using spans, we need to use CSS to position them inside the div; if we're using SVG, this is already taken care of. We want to position lines 2 and 3 in the centre – one on top of another – while spacing lines 1 and 4 evenly above and below, making sure to centre the transform origin.

We're going to rely on transitioning two properties: opacity and rotation. First of all, we want to fade out lines 1 and 4, which we can target using the ':nth-child' selector.

```
.menu-icon.is-active {element-type}:nth-child(1),  
.menu-icon.is-active {element-type}:nth-child(4) {  
    opacity: 0; }
```

The only thing left to do is target the two middle lines and rotate them 45 degrees in opposite directions.

```
.menu-icon.is-active {element-type}:nth-child(2) {  
    transform: rotate(45deg); }  
.menu-icon.is-active {element-type}:nth-child(3) {  
    transform: rotate(-45deg); }
```

GET THE CODE

Spans - srt.lt/zTdOEo

SVG - srt.lt/iOz8



**DUE TO DELAYS,
OUR CIRCLE
NOW ANIMATES
IN TURN,
CREATING THE
ILLUSION OF THE
CIRCLES
CHASING EACH
OTHER**



A VIOLENT ACT

bit.ly/2hlcGyf

MOVING BACKGROUND

The 'A Violent Act' website utilises masking and subtle movement to grab the attention of the user. The majority of the work here is in the setup and creating the SVG.

CODE - srt.lt/E5fJ



PEEK-A-BEAT

peekabeat.com

PULSING CIRCLES

The pulse animation used on the Peek-a-Beat website is simple yet effective and not difficult to reproduce – consisting of three circles inside an SVG in which we animate their scale and opacity.

CODE - srt.lt/Zc8S5c



ENSEMBLE CORRESPONDANCES

bit.ly/2nSwkbB

ELEVATED TITLE

Ensemble Correspondances utilises simple animation to convey movement in music. The design loosely represents sheet music.

CODE - srt.lt/cL4o6



GOD OF WAR

godofwar.playstation.com

UNDERLINE FROM THE CENTRE

The animation consists of positioning the '::after' pseudo element to the bottom and then scaling it when the button is interacted with.

CODE - srt.lt/E2t7K



COLOUR TRANSITION

codepen.io

SIMPLY EFFECTIVE

This effect is used on the majority of websites in some way or another. The effect is transitioning between two text colours and is done using one simple line of CSS.

CODE - srt.lt/g2Q8X



FADE AND EXPAND

This movie credits effect is simple and animates a few properties at once to achieve the effect. We increase the 'letter-spacing' and 'scale' while reducing the 'opacity'.

CODE - srt.lt/sPp3x3

web workshop

Add an instructional video to demo an app

Inspired by <https://worldbrush.net>



Add an instructional video to demo an app

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

EXPERT ADVICE

Creative background

As the app is a painting app using augmented reality, the background of this site gives a glimpse into how users can paint in the real world. As easy as it is to use the brush here, it is just as simple to create drawings with the app. This is a great taster of the real thing, which helps promote the app.



<comment>
What our
experts think
of the site

Collaborative augmented reality painting

"World Brush is an AR experience that lets users paint on the world for others to discover. A simple mechanic of drawing on your screen tied to the real world allows for unlimited possibilities that transcend language barriers and fit within the context of each person's unique experience."

Active Theory

Technique

1. Designing the video demo

Having a video demonstration on the screen to help promote your app is crucial, so here the code structure is added to create a similar layout to World Brush, with the phone, video and text on the screen.

```
<div class="container">
<div class="phone">
<video src="assets/screen.mp4" class="vid"
loop preload="auto" muted="true"
autoplay="true"></video>
</div>
<div class="text">
<p>Add your own text</p>
</div>
</div>
```

2. Creating the CSS

Most of the power for this layout is all in the build of the CSS. The first step to achieving the right result is to set the body to fill the height of the browser and turn off the default browser margin and padding.

```
body {
width: 100%;
height: 100%;
margin: 0;
padding: 0;
}
```

3. Centred container

Now the container is centred on the page by setting its width and height, then positioning this 50% from the top and left. Half of the width and height is removed to make this centred.

```
.container {
position: absolute;
width: 880px;
height: 780px;
background-size: 440.7px 780px;
left: 50%;
top: 50%;
margin-left: -440px;
margin-top: -390px;
}
```

4. Placing the phone

The phone section is going to be on the left so the code really just needs the correct width and height applying to it so that the image can fit onto the screen. The phone image is placed into the background and fits into the size of the div.

```
.phone {
width: 440.7px;
height: 780px;
background-size: 440.7px 780px;
background-image: url(assets/device.png);
}
```

5. Adding the video

The video is a child of the phone so it just needs the right height and width applying to it and moving into position for where the screen is. The video is set to autoplay and loop from the HTML side of the code.

```
.vid {
position: relative;
width: 288px;
height: 511.2px;
background-size: 288px 511.2px;
left: 50%;
top: 50%;
margin-left: -144px;
margin-top: -292.5px;
}
```

6. Position the text

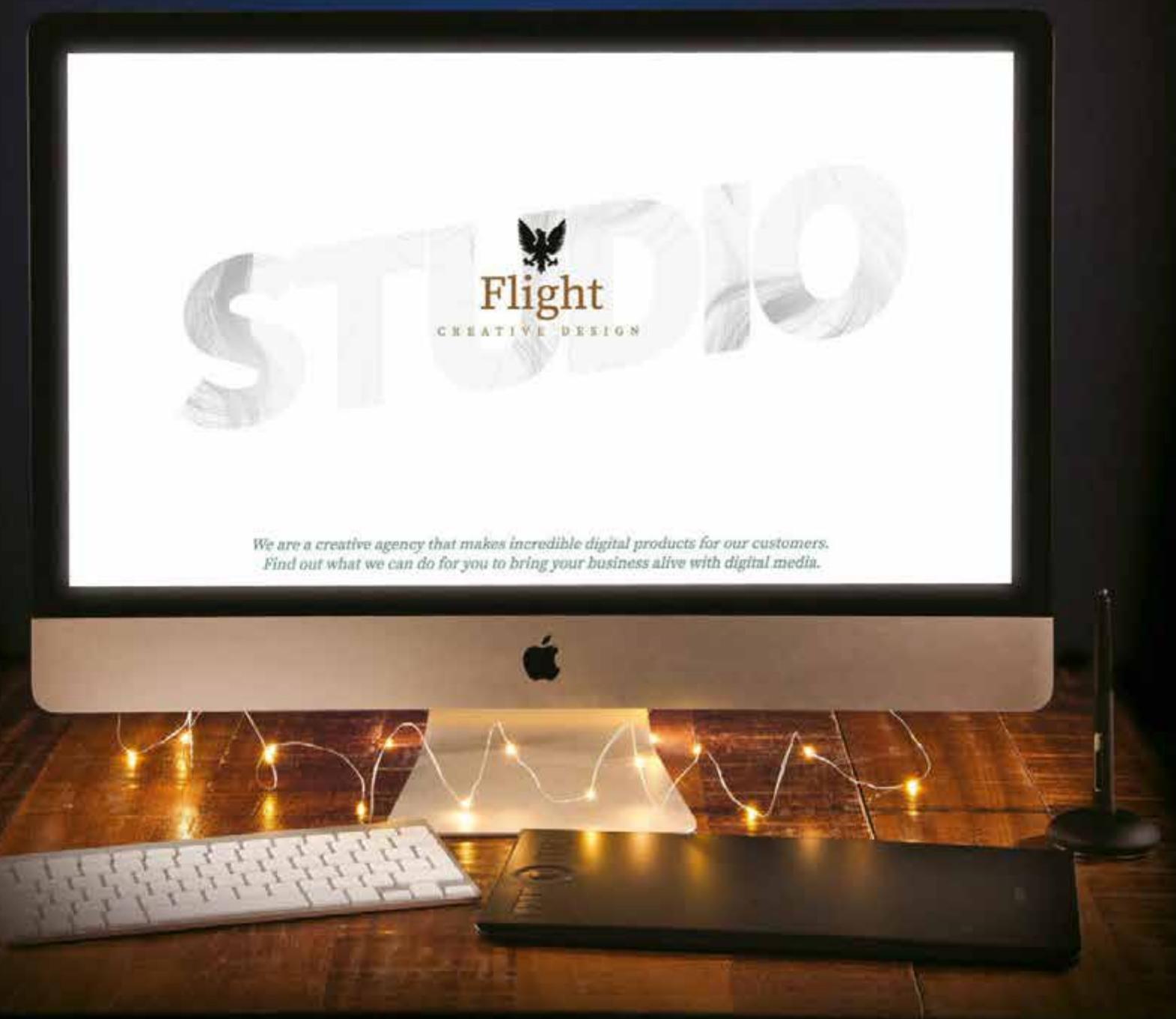
The final step is just to position the text over to the right-hand side of the content so here this is completed. Save the page and preview this in your browser and you will see the video and phone perfectly align to give a demo for any app you're marketing.

```
.text {
position: absolute;
width: 320px;
height: 780px;
top: 30%;
right: 0px;
}
```



Make interactive 3D typography effects

Create exciting text effects with flowing lines and reactive mouse movement, perfect for your site's headers





Typography has always played a major part in any designer's arsenal of tools as they select the right typeface that will enhance the message and present the right context for what is being communicated. Over the past eight years, the web designer has had the ability to bring in custom typefaces to their design and have similar typographical control to those enjoyed by print designers. Taking a look at many of the sites that are featured as award winning or receiving 'site of the day' titles and you will soon notice that their use of typography becomes central to the design, allowing them to rise above their competition. This can range from animated letter forms, reactive movement to the user interactions, to bold use of type forms taking centre stage. In this tutorial, the type effect will use the shapes of the letters as a mask to some fast, free-flowing particles trails that will dynamically swirl and move through the letters. Not only will there be this beautiful animation, but as this will be rendered onto the HTML5 canvas element, this will be transformed in 3D to rotate towards the mouse as it moves around the screen. This is perfect for site headers or just when you need to grab the user's attention for a call to action.

1. Starting the process

Open the 'start' folder from the project files in your code IDE. The project is going to start by creating the particle object class. This will be used to create the flowing imagery within the text in the project. Open the 'sketch.js' file and add the following variable to start creating the base particle.

```
function Particle() {
  this.pos = createVector(random(width),
    random((height - 100)));
  this.vel = createVector(0, 0);
  this.acc = createVector(0, 0);
  this.maxspeed = maxSpeed;
  this.prevPos = this.pos.copy();
```

2. Updating the particle

In order to move the particle, an update function will be run each frame, this will work out the velocity of the particle and the acceleration to the velocity. The velocity will eventually be limited by a global variable which will be added later. The velocity is added to the position of the individual particle. By creating one particle, several thousand will be created on the screen at any one time.

```
this.update = function () {
  this.vel.add(this.acc);
  this.vel.limit(this.maxspeed);
  this.pos.add(this.vel);
  this.acc.mult(0);
}
```

3. Going with the flow

To give the particles their flowing movement, a flow field generated by noise will be followed. The function created here enables the vector of flow to be passed in and it will then be followed, hence the name of this function. The force of the vector direction will be applied to the particle.

```
this.follow = function (vectors) {
  var x = floor(this.pos.x / scl);
  var y = floor(this.pos.y / scl);
  var index = x + y * cols;
  var force = vectors[index];
  this.applyForce(force);
}
```

4. Follow but not too closely

In order to stop all the particles bunching up together, which can easily happen with this kind of movement, the particles will have a very small amount of randomness added to their position. This will cause a slight amount of scattering to occur.

```
this.scatter = function (vectors) {
  this.pos.x += random(-0.9, 0.9);
  this.pos.y += random(-0.9, 0.9);
}
this.applyForce = function (force) {
```

```
this.acc.add(force);
}
```

5. Displaying the particle

The show function here displays the particle. The first thing that it does is add a one pixel stroke of a light grey colour to create the line. The line is drawn from its current position to its last position on the previous frame. The previous position is stored for next time through the loop.

```
this.show = function () {
  stroke(180);
  strokeWeight(1);
  line(this.pos.x, this.pos.y, this.
  prevPos.x, this.prevPos.y);
  this.updatePrev();
}
this.updatePrev = function () {
  this.prevPos.x = this.pos.x;
  this.prevPos.y = this.pos.y;
}
```

6. Wrap around

The edges function works out if the particle reaches the edge of the screen and, if so, wraps it around to come on the opposite side. This section deals with the x position so it is detecting if it is greater than the width of the screen then sending it to left edge and vice versa.

```
this.edges = function () {
  if (this.pos.x > width) {
    this.pos.x = 0;
    this.updatePrev();
```

Defining a particle

A particle can be thought of as a dust particle that can be seen through a shaft of sunlight. Digitally it can represent anything from snow and rainfall, to explosions or the way that fluids move. Generally, many particles are used to make a particle system.

p5.js

[Download](#) • [Start](#) • [Reference](#) • [Libraries](#) • [Learn](#) • [Community](#)

Hello! p5.js is a JavaScript library that starts with the original goal of Processing, to make coding accessible for artists, designers, educators, and beginners, and reinterprets this for today's web.

Using the original metaphor of a software sketchbook, p5.js has a full set of drawing functionality. However, you're not limited to your drawing canvas, you can think of your whole browser page as your sketch! For this, p5.js has added libraries that make it easy to interact with other HTML5 objects, including text, input, video, webcam, and sound.

p5.js is a new interpretation, not an emulation or port, and it is in active development. An official editing environment is coming soon, as well as many more features!

p5.js was created by Daniel Shiffman and is developed by a community of contributors, with support from the Processing Foundation and NYU ITP. Thanks and grants always to [these awesome folks](#).



Left

The effect that is being created is helped extensively by the p5.js library that enables a number of helpers for drawing to the HTML5 canvas element

Top left

The basic HTML5 layout and CSS design has been created in advance so that you can focus on the integration of the flowing lines of the text effect in JavaScript

Top right

Once the particle object class is created, a number of particles are added to the page. The flowing lines can be seen without the addition of the text effect

Tutorials

Make interactive 3D typography effects

```
    }
    if (this.pos.x < 0) {
        this.pos.x = width;
        this.updatePrev();
    }
```

7. Wrapper's delight

This code is the remainder of the edge detection and it detects the particle on the y axis for the top and bottom of the screen. The brackets here wrap up the entire particle class. This means by using this class many particles can be created.

```
if (this.pos.y > height) {
    this.pos.y = 0;
    this.updatePrev();
}
if (this.pos.y < 0) {
    this.pos.y = height;
    this.updatePrev();
}
}
```

8. Making many

Now as the particle is created it's time to think about making many particles. To do this all of our code can go above the Particle function class. Here a number of global variables are declared to enable the system to run. They'll be called at various times during the code, so they can then be explored.

```
var inc = 0.1;
var scl = 100, zoff = 0;
var cols, rows, movement = 0;
var particles = [];
```

CSS 3D transforms

CSS 3D transforms allow elements to be transformed on all three axis meaning that flat elements can be twisted and spun around. Using this technique, the mouse position will slightly rotate the canvas element.



Top left

The text is now present and it's possible to see the flowing lines, swirling inside the text of the design.

Top right

If the design is loaded on smaller size screens, the number of particles is reduced as there is less screen and it also helps by having less particles to process

Right

The final section of code takes the mouse position and applies a CSS transform to the canvas element. This rotates the canvas towards the mouse in 3D space

```
var flowfield;
var img;
var maxSpeed;
var t, calcX = 0, calcY = 0, currX = 0,
currY = 0, targetX = 0, targetY = 0;
```

9. Setting it all up

The setup function, declared here, sets how the screen will look at the start. The first detection being done is to see what the width of the screen is. If it's relatively large, a large image is loaded, the canvas is created and this is scaled via CSS to fit within the display.

```
function setup() {
    if (windowWidth > 1200) {
        img = loadImage("assets/studio.png");
        var canvas = createCanvas(1920, 630);
        maxSpeed = 10.5;
    }
}
```

10. Other screens

The rest of the if statement checks different screen resolutions and loads an image that is most appropriate for that screen size. Similarly different-sized canvas elements are created. This is mainly to stop a mobile dealing with more pixels than it has to.

```
else if (windowWidth > 900) {
    img = loadImage("assets/studio-tablet-wide.png");
    var canvas = createCanvas(1200, 394);
    scl = 60;
    maxSpeed = 7;
} else {
    img = loadImage("assets/studio-tablet-tall.png");
    var canvas = createCanvas(700, 230);
    scl = 40;
    maxSpeed = 5;
}
```

11. Making a grid

Once the screen size is worked out the canvas is placed inside the header div tag in the index.html page. A

number of columns and rows are worked out based on the width and height; it's a little like an invisible grid. Finally, an array is set for the flow field.

```
canvas.parent('header');
cols = floor(width / scl);
rows = floor(height / scl);
flowfield = new Array(cols);
```

12. Making particles

The number of particles is set up based on the width of the screen - if the screen is 1920 pixels wide then 2500 particles will be created and it moves downwards from there. A for loop creates the new particles. The background colour of the screen is set to almost full white.

```
var numParticles = Math.floor((2500 / 1920) * width);
for (var i = 0; i < numParticles; i++) {
    particles[i] = new Particle();
}
background(245);
```

13. Drawing the screen

The results of all the calculations are drawn on screen every frame in the draw function. Firstly, a light grey rectangle with a very low opacity fills the screen to fade what has been drawn previously. After this is drawn, the fill is turned off as the particles will be made up of strokes not fills.

```
function draw() {
    noStroke();
    fill(245, 10);
    rect(0, 0, width, height);
    noFill();
    var yoff = 0;
```

14. Creating a flow effect

To get the flow effect there are two 'for' loops moving through the rows and columns to update the noise values. These are then changed into angles from the noise value ready to update the particles for each of the positions on the screen.





Changing the behaviour

Whenever there is a particle system, there is normally a series of numbers which control the behaviour of what is happening on screen. In this tutorial the scaling is important and this is set to 100 for a 1200 pixel and larger monitor. This means that the way the particles will flow is in 100 pixel blocks and being able to turn. Increase this to 200 and you will see the turning be far less because it's spread out further. On smaller devices this is reduced slightly so that the overall effect still remains true to the original desktop experience. Playing with the scl variable as defined in steps 8 through 10 will change the behaviour as will changing the maxSpeed. Enjoy experimenting!

```
for (var y = 0; y < rows; y++) {
  var xoff = 0;
  for (var x = 0; x < cols; x++) {
    var index = (x + y * cols);
    var angle = noise(xoff, yoff, zoff) *
    TWO_PI * 4;
    var v = p5.Vector.fromAngle(angle);
```

15. Update the array

The array of flow is updated with the angle and the values are increased so that the offset of each position is increased each time it goes up. This might seem complicated but it really just creates random flowing motion for the particles to follow on the screen.

```
v.setMag(1);
flowfield[index] = v;
xoff += inc;
}
yoff += inc;
zoff += 0.001;
}
```

16. Update the particles

Now the particles are all looped through in their array. Each individual particle is told to follow the flow field, to update, check the edges of the screen, scatter slightly and finally be drawn on the screen using the show function. Save the file and test the 'index.html' to see the particles moving about.

```
for (var i = 0; i < particles.length; i++)
{
  particles[i].follow(flowfield);
  particles[i].update();
  particles[i].edges();
  particles[i].scatter();
  particles[i].show();
}
```

17. Adding the text

The text is a mask that is placed over the top. To do this, the correct image is placed over the top of the particles. Add this code before the closing brace of the draw function. Save and check the browser to see the effect working with the text now.

```
image(img, 0, 0);
```

18. Detecting the mouse position

The mouse position is referenced and the x and y values are mapped onto degree angles that can be moved. On the Y axis this will be -25 to 25 and vice versa for the x axis. The remaining code should be placed after the last code was added, before the end of the draw function.

```
targetY = Math.round(map(mouseX, 0, width,
-25, 25));
targetX = Math.round(map(mouseY, 0, height,
25, -25));
```

19. Easing into place

The target position is now given a little easing so that the degrees slowly reach their target. This is created using a classic easing algorithm of taking off the current position from the destination and multiplying by a low number.

```
var vx = (targetX - currX) * 0.05;
var vy = (targetY - currY) * 0.05;
calcX += vx;
calcY += vy;
```

20. Writing the CSS

The 't' variable here takes the calculated values and places them into a CSS string using the transform values of rotateX and rotateY. The current position is calculated from the position the canvas is currently rotated to.

```
t = 'rotateX(' + calcX + 'deg) rotateY(' +
calcY + 'deg)';
currX = calcX;
currY = calcY;
```

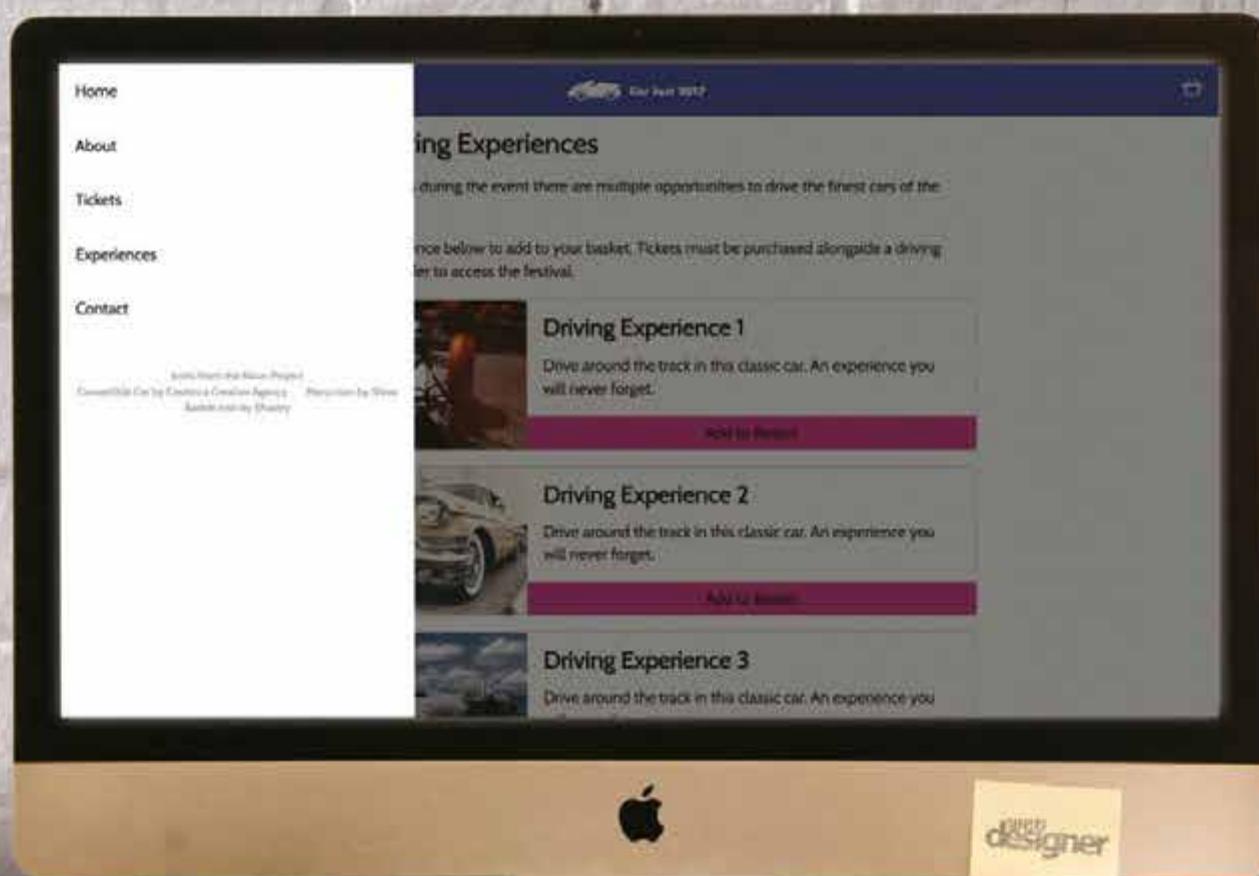
21. Finishing off

Now the CSS is applied to the canvas element in this code. Save the page and preview this in the browser. Now the mouse fully updates the rotation of the canvas so that it turns as the mouse moves. Of course all of the particles in that space move with it on the screen.

```
canvas.style.WebkitTransform = t;
canvas.style.msTransform = t;
canvas.style.transform = t;
```

Create silky smooth UIs with CSS and JS

Create animations that run at 60fps using the latest CSS properties and the Web Animations API



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



As a website becomes more complex, issues around site performance soon arise. While a lot of attention goes towards the initial page load, less emphasis is put on how it performs while in use.

Users expect pages to work smoothly all the time. While browsers optimise what they can, it is not always possible to know what the developer's intentions are soon enough to see any benefit.

Great rendering performance is a delicate balancing act between HTML, CSS and JavaScript. They all play a part in how well a page performs at different stages. Too much work in any of these areas can grind everything to a halt.

If we can take as much pressure from the main thread as possible, that will give it more time to respond to user input. By using some new features in the browser, we can let it know what can be done elsewhere to keep the frame rate high.

In this tutorial we will be building a couple of small interface animations to bring a fairly standard website to life. The first will be a smooth slide-in sidebar, which just uses CSS to create a stutter-free animation. The second is a small animation when adding a product to the basket that reacts to the user's input.

We will be making use of the Web Animations API, which means we should be developing in a browser that supports it. Chrome or Firefox will work just fine.

1. Add sidebar structure

Before we get started, it's important that the HTML structure is right. We want to avoid adding any unnecessary markup to complicate the animation.

The main sidebar `<div>` will act as an overlay when the navigation slides out. It will have one child `<nav>` element that becomes the actual sidebar and is the only element being transformed.

Open up `index.html` and add the following to the sidebar `<div>`:

```
<nav class="sidebar__contents">
```

```
<ul class="sidebar__nav">
<li class="sidebar__nav-item">
<a href="#home" class="sidebar__nav-item-link">Home</a>
</li>
</ul>
</nav>
```

2. Move to an overlay

Right now there is no overlay and our sidebar is sitting in the middle of the page. The overlay should take over the entire screen to focus on the sidebar. We need to fix its position and take over 100% of the viewport.

In `Sidebar.css`, add some CSS to make the overlay take over the screen. Be sure to adjust the z-index to keep it on top of the page contents.

```
.sidebar {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0,0,0,0.5);
  z-index: 2; }
```

3. Hide by default

The sidebar now looks right, but is constantly on the screen. It needs to be hidden when we load the page and only show up when requested.

Normally this would be done by setting its 'display' property to 'none'. But to get a smooth animation in right from the start, the element needs to be ready to go.

Add a couple of properties to the sidebar style to both make it invisible and make it non-interactive, and reverse them on the 'visible' class.

```
.sidebar {
  ...
  opacity: 0;
  pointer-events: none;
}
```

```
.sidebar.sidebar--visible {
  pointer-events: auto;
  opacity: 1; }
```

4. Toggle sidebar visibility

The button in the top left will trigger the reveal of the sidebar. We need some JavaScript to show and hide the sidebar when that button is clicked.

Open up `Sidebar.js` and add a listener to perform the toggle. Remember to keep a note of the sidebar state within the module.

```
let visible = false;
toggleButton.addEventListener('click', toggleSidebar);
function toggleSidebar() {
  if (visible) {
    hideSidebar();
  } else {
    showSidebar();
  }
}
```

5. Add show and hide logic

Rather than animate using JavaScript, we will let CSS do the hard work. Browsers can perform CSS animations a lot more easily than adjusting top and left positions with JavaScript.

Create the show and hide functions that add and remove a special class to the sidebar to show it's visible, as well as update the internal state.

```
function showSidebar() {
  container.classList.add('sidebar--visible');
  visible = true;
```

Visibility vs opacity

While 'visibility: hidden' and 'opacity: 0' will both hide an element, changing visibility can cause a repaint of the page, which can be computationally expensive and should be avoided.

Top

Be sure to revert the 'pointer-events' property, otherwise users will only be able to interact with the sidebar through keyboard input

Left

Changing CSS properties will cause different rendering behaviour in different browsers. The csstriggers.com site groups these behaviours by property

Tutorials

Create silky smooth UIs with CSS and JS

```
}

function hideSidebar() {
  container.classList.remove('
    sidebar--visible');
  visible = false;
}
```

6. Hide on overlay click

When the user either clicks the overlay or a sidebar item, we need to hide everything again.

We can apply a listener to the entire sidebar container to preserve some memory using a method called 'event delegation'. If we directly clicked the container or an anchor tag, then we close the sidebar.

```
container.addEventListener('click',
  hideSidebarCheck);
function hideSidebarCheck(e) {
  if (e.target.id === 'sidebar' ||
    e.target.nodeName === "A") {
    hideSidebar();
  }
}
```

7. Fade sidebar contents in

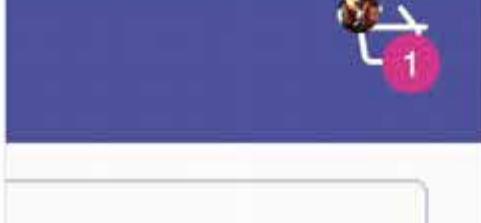
The CSS 'will-change' property highlights what our intentions are to the browser in order for it to make certain optimisations. In this case, it will promote the element to its own layer and use additional resources to improve composition and overall performance.

Add the following to the sidebar class in **Sidebar.css**. We should then see the sidebar and overlay nicely fade in and out.

```
will-change: opacity;
transition: opacity 0.2s;
```

The User Timing API

This new API is making its way to browsers and can help report on performance issues through JavaScript. For example, `performance.mark` can be used to measure interactions specific to your application.



Above

As images zoom towards the basket, they are offset slightly from the top right of the window. Without that, they look like they're disappearing off the screen

Right

Firefox's animation tools also work with the Web Animations API. Green lightning bolts indicate when an animation is only using composites

8. Slide sidebar in and out

While the fade makes a nice transition, it makes sense for the sidebar to slide in from the side of the screen as well.

Rather than slide in when visible, we will do the opposite and slide it out when it is not visible. This allows us to use the default positioning of the element and write less code.

Apply a similar technique from the last step to a transform of the sidebar contents to move it off screen by default. Reset it when visible by removing the transform.

```
.sidebar .sidebar__contents {
  [...]
  will-change: transform;
  transition: transform 0.2s;
  transform: translateX(-100%);
}
.sidebar.sidebar--visible
.sidebar__contents {
  transform: none;
}
```

9. Add event listeners

With our sidebar complete, it's time to turn our attention to the basket animation.

Before we go any further, we need to add some event listeners. As this product list could get very long, we want to add listeners to the container to avoid the performance overhead that comes with adding lots of listeners individually.

Open up **Products.js** and add listeners for click and touch events.

```
container.addEventListener('click',
  checkAddToBasket);
container.addEventListener('touch',
  checkAddToBasket);
```

10. Check the correct target

As we are using event delegation, we need to check first if what was clicked was an 'Add to Basket' button. As these buttons have a data attribute associated with them linking them to a product, we need to check it has that as well.

Inside the `checkAddToBasket` function, add these checks to an if statement to make sure we get the behaviour we are expecting.

```
if(e.target.className == 'product__add'
  && parseInt(e.target.dataset.id,
  10) >= 0) {
  // next step goes here }
```

11. Locate the product by ID

The data attribute provides an ID, so we can use it to locate the product from our list.

To do this we can use the 'find' method, introduced in ES2015, to flick through the array and locate the related object. If it isn't found for any reason, exit the function here to avoid any nasty side effects.

```
const product = productList.find(
  listItem => listItem.id ===
  parseInt(e.target.dataset.id, 10));
if (!product) {
  return false; }
```

12. Add product to basket

In short, our code will be in two steps – first adding the product to the basket, then the animation comes afterwards.

A normal application would have some server logic here, but here we are just simulating the process with a function. Add a call to it, followed by a call to the animation phase. Wrap it in a browser support check, which will be dealt with in the next step.

```
addToBasket(product);
if (supportsAnimationsApi) {
  animateToBasket(e, product); }
```

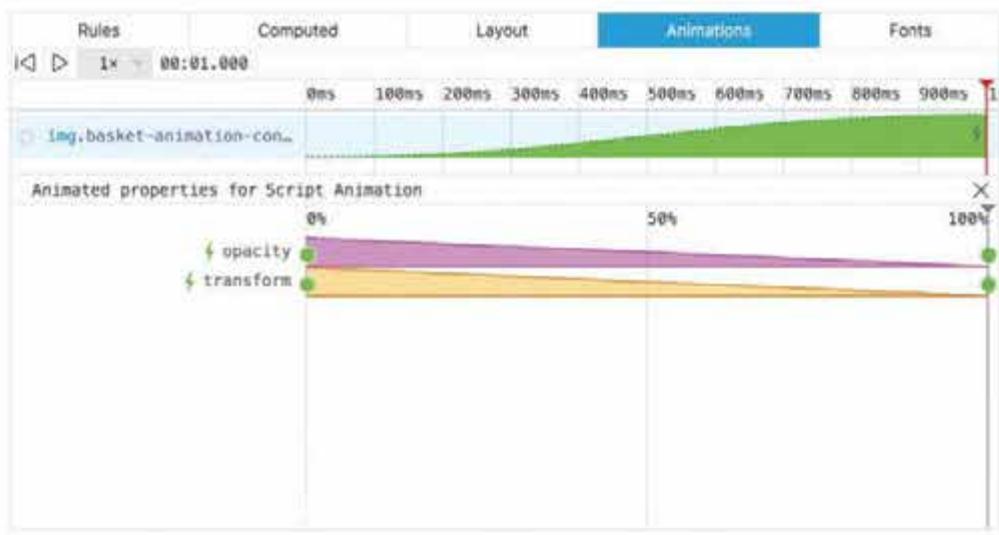
13. Add feature detection

The Web Animations API is a fairly new addition to browsers. At the moment, only Firefox, Chrome and Opera support it.

Our basket animation is added flair and as such it is not critical if a browser doesn't support it. But to avoid any crashes, we need to check it exists before using it.

Add a variable to check for support above the event listeners from earlier.

```
const supportsAnimationsApi =
  browserSupportsWebAnimationsApi();
```



How browsers render pages

A lot of work goes into setting up a page, but each web browser follows the same pattern in order to get pixels on the screen.

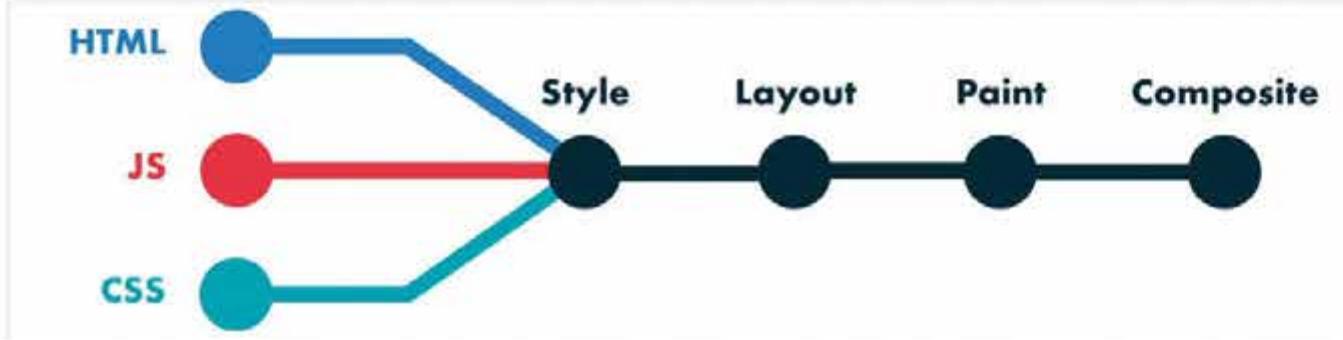
First, it checks if JavaScript has caused any visual changes. This

could be anything that changed the document. Next, it looks at styles: it first checks the page structure and which style rules apply. The next step then works out how to lay all these elements together on the page.

These elements are then painted on the page. Complex effects can be put on a separate layer, which means they can be updated quicker and easier.

Finally, all these painted layers are composited together to create the

webpage. This is the cheapest step in terms of computation, which is why any changes should target this part of the pipeline. At the moment, 'transform' and 'opacity' are the sole properties that only trigger composites.



14. Check for Web Animations API

Unfortunately, the check for the Web Animations API is a bit cumbersome. It involves creating an element and checking if a method exists on it. To avoid repeating work unnecessarily, we are caching the result of this function to reference in future.

Add the following lines of code to the `browserSupportsWebAnimationApi` function.

```
const test = document.createElement('div');
if ('animate' in test) {
  return true;
}
return false;
```

15. Capture event data

As we can only perform transform animations with any efficiency, we will need to do some calculations in order to work out how the image moves from its starting position to the basket in the top right.

All this happens inside the `animateToBasket` function. For starters, store the values of the point the button was clicked, as well as the current screen size.

```
const x = e.clientX;
const y = e.clientY;
const pageWidth = window.innerWidth;
const pageHeight = window.innerHeight;
```



16. Create a product icon

Now we need something to animate. We are going to be using a thumbnail of the product as an indicator of what is heading to the basket.

Create an `img` element and add it to the animation context, which is a transparent box over the screen to make sure these animations happen in the same space.

```
const image = document.createElement('img');
image.className = 'basket-animation-context__icon';
image.src = product.image;
animationContext.appendChild(image);
```

17. Avoid `getBoundingClientRect()`

We need to get the dimensions and position of the newly created icon, but using `getBoundingClientRect()` would trigger a paint on the page. Triggering too many paints can be bad for performance.

The styled class on the icon means it is 50x50px and located in the top left. Use these values for later calculations instead.

```
const coordinates = {
  left: 0,
  top: 0,
  width: 50,
  height: 50, }
```

18. Calculate start and end

The animation starts from the point that was clicked or touched and ends in the top right, where the basket icon is. As we are transforming the icon rather than directly positioning it, we need to calculate the offset by using the dimensions of the window.

While using `window.innerWidth` does also trigger a paint on the page, unfortunately it is unavoidable in this case.

```
const startX = x - coordinates.left -
  (coordinates.width / 2);
const startY = y - coordinates.top -
  (coordinates.height / 2);
const endX = window.innerWidth -
  coordinates.left -
  (coordinates.width / 2) - 20;
const endY = 0 - coordinates.top -
  (coordinates.height / 2) + 40;
```

19. Provide animation keyframes

The Web Animations API works much like CSS

animations, which means they are just as performant. They also share similar syntax.

The API needs two parts - an array of keyframes and a set of options to apply to them. For the first part, create some keyframe objects to move from our start to end positions.

```
const keyframes = [
  {
    transform: 'translateX(${startX}px)'
      'translateY(${startY}px) scale(1)',
    opacity: 1,
  },
  {
    transform: 'translateX(${endX}px)'
      'translateY(${endY}px) scale(0.25)',
    opacity: 0,
  }];
]
```

20. Set up icon animation

The second part works in a similar way to the properties of regular CSS animations. These define how the animation plays out.

In this setup we need the animation to stop where it finishes, have easing to make the movement feel smooth, and last for 1 second.

Lastly, kick off the animation by calling the `animate` method.

```
const options = {
  fill: 'forwards',
  easing: 'ease-in-out',
  duration: 1000,
};

const animation = image.animate(
  keyframes, options);
```

21. Remove on complete

The last step involves a little tidying up. To avoid having multiple invisible elements clogging up the page, we should remove them when the animation finishes.

The Web Animations API provides a simple event system to handle situations like this. Create an `'onfinish'` callback function to remove the icon.

```
animation.onfinish = function() {
  animationContext.removeChild(image); }
```

Use the WordPress API to power a blog

Discover how to Implement the WordPress API into a Vue.js powered blog app using Vuex, a state management library





ver the last few years, the development of a REST API for WordPress has opened new doors for developers. Previously, developers who had been limited to

writing a WordPress-powered project in PHP, now have more flexibility and control in how they can approach the technology stack of their website-to-be. This means it's possible to retain all the advantages of the brilliant WordPress control panel, which is made extremely flexible by popular plugins such as Advanced Custom Fields, and have a completely custom built front-end which only interacts with WordPress when it needs to.

In this tutorial, we'll be exploring how to implement the WordPress REST API into a simple blog app, which is currently using a local JSON file as its data source, and is built as a single page application (SPA) using the popular JavaScript framework Vue.js. This will involve the implementation of Vuex, a state management pattern + library for Vue.js applications, which we'll use to store the data we request from a WordPress using a combination of its action and mutation methods.

Once completed, you should have created a lean, simple single page application, which has all the reactivity of Vue.js, while displaying posts retrieved from and managed by WordPress.

1. Setup workspace and dependencies

First things first, you should download the project's files from FileSilo and open them in your preferred editor. In the console, cd into 'website-template' and run the command below to install the project's node dependencies (If you don't have Node installed, visit nodejs.org). We'll be working purely in the 'src' directory from here on out.

```
npm install
```

2. Install Vuex

Next, using the command below, we'll install Vuex, which is a state management pattern + library for Vue.js

applications. This will act as a central information store for all Vue components which depend on the data we receive from WordPress API. For developers familiar with React, Vuex is heavily inspired by Flux.

```
npm install vuex --save
```

3. Start development server

In the console, run the command below to start the development server. This will compile the Vue.js project as it currently stands and assign it to a URL so you can access it. This is usually localhost:8080. One big advantage this brings is live reloading, so once you make changes to the app and save, the page in your browser will update itself without the need to manually reload.

```
npm run dev
```

4. Create Vuex store

In 'src', create a directory called 'store' and within it a new file called 'index.js'. This will be where our Vuex store will be defined. Though before we get to that, we need to first make sure our Vue.js app is aware of its existence. To do this, open 'main.js' and import the store and include it as a dependency, as in the snippet below.

```
import Vue from 'vue'
import App from './App'
import router from './router'
import store from './store'
Vue.config.productionTip = false
/* eslint-disable no-new */
new Vue({
  el: '#app',
  router,
  store,
  template: '<App/>',
  components: { App }
})
```

5. Create store scaffolding and install Axios

To help us simplify the AJAX requests our store will be

making to WordPress API, we'll install Axios, which is a Promise based HTTP client. To do this, open a separate console window, navigate to the 'website-template' directory and run 'npm install axios -- save'. Next, let's start to scaffold the store by instantiating a new empty Vuex store object. At the moment, it isn't doing anything, but at least Vue should be aware of it.

```
import axios from 'axios'
import Vue from 'vue'
import Vuex from 'vuex'
Vue.use(Vuex)
const store = new Vuex.Store({
  state: {},
  actions: {},
  mutations: {}
})
export default store
```

6. Create posts state

In Vuex, the state object holds application info, which in this case will be the WordPress post data we'll grab using the API. Within this object, create a new property called posts and assign it a value of null.

```
state: {
  posts: null
}
```

7. Create getPosts action

In Vuex, actions are the main way in which asynchronous requests are handled. These are typically methods defined in the store, which can then be dispatched as required by the app. In the empty actions object, let's

WordPress API docs

The WordPress REST API provides endpoints which allow developers to send and receive data from a WordPress installation. For more uses of the API beyond this tutorial, visit <https://developer.wordpress.org/rest-api/>



Left

If you visit www.lukeharrison.net/webdesigner you can access the WordPress installation, which this tutorial is using as its data source

Top

With the username 'demo' and the password 'd3mo' you will be able to login and view some of the test posts which this Vue.js app will be displaying

Tutorials

Use the WordPress API to power a blog

define one where if our posts state is still null, axios is used to perform an AJAX request to the WordPress API and return a list of posts. Once we've received a positive response, we'll then resolve the promise and commit the posts using the 'storePosts' mutation.

```
getPosts: function(context) {
  return new Promise((resolve, reject) => {
    if(context.state.posts) {
      resolve()
    } else {
      axios.get('http://lukeharrison.net/webdesigner/wp-json/wp/v2/posts')
        .then((response) => {
          context.commit('storePosts', response.data)
          resolve()
        }).catch((error) => {
          reject(error);
        });
    }
  })
}
```

8. Create storePosts mutation

Another concept introduced by Vuex is mutations, which are also typically methods defined in the store. Mutations are the only way to change state in a Vuex store, which allows state to be easily tracked when debugging. In the empty mutations object, let's define the 'storePosts' method we referenced in the previous step and make it override the post property in the state object with any data the mutation receives as payload.

```
storePosts(state, response) {
  state.posts = response }
```

Vue Linting

The vue-cli template used for the app comes bundled with eslint. Linting analyses your code using a set of rules and reports failures. It's a great way of making code more consistent and avoiding bugs.

Right

When developing with Vue.js, it's recommended that you also download the Vue devTools extension, which is available at the time of writing for Chrome and Firefox.

Top

The extension gives you the power to extensively debug your Vue.js application, and also supports Vuex, allowing you to visualise the stores current state and mutation history.

9. Trigger getPosts action on load

We've created the action and mutation methods which grab posts from WordPress API and commit them to Vuex state, but now we need to actually trigger this process somewhere. In the top level Vue.js component 'App.vue', add the snippet below. 'created()' is a lifecycle hook which triggers code as soon as the Vue component is created on load, while the use of the global '\$store' variable allows us to access the contents of our Vuex store and dispatch the 'getPosts' action from step 7.

```
created() {
  this.$store.dispatch('getPosts') }
```

10. Update attribute paths

If you're working in Chrome or Firefox and have the Vue.js DevTools extension (if not, I recommend you do as it's very useful), you should now be able to see the loaded WordPress posts in 'Base State' under the 'Vuex' tab. Back to the app, in '/components/posts/posts.vue', the template HTML needs to point to this data, so let's tweak a few of the attribute paths.

1. Switch 'post.title' to 'post.title.rendered'
2. Switch 'post.preview' to 'post.acf.preview'
3. Switch 'post.previewImage' to 'post.acf.header_image_small'

11. Update v-for loop

In the posts component, there's a Vue.js directive in use called 'v-for'. This loops through all posts and for each one prints an instance of the post component, displaying them in a list. We need to update the path passed to this directive as it's still using the local dummy test data.

Update the v-for directive to the snippet below in order to point to our stored posts in the Vuex store.

```
v-for="post in this.$store.state.posts"
```

12. Housekeeping

A list of the WordPress posts should now be displaying. As we're no longer using the local post data, let's delete 'src/data'. Then in the posts component, remove the 'posts: postData.data' property in the components local data store and then the postData import.

13. Fixing post author

You may notice that for each post the author is showing up as a number. This is because the WordPress API returns an author id, rather than a name. We need to use this id to query WordPress for the full author information. Let's start by creating a place to store this in our Vuex store, alongside our post info, in 'store/index.js'.

```
state: {
  authors: null,
  posts: null }
```

14. Create getAuthors action

As with posts, we'll create an action in '/store/index.js' to trigger an AJAX request to query WordPress API. Once a successful response is received, the promise will then resolve and trigger the 'storeAuthors' mutation, which we'll create next.

```
getAuthors: function(context) {
  axios.get('http://lukeharrison.net/webdesigner/wp-json/wp/v2/users')
    .then(function(response) {
      context.commit('storeAuthors', response.data)
    })
}
```

15. Create storeAuthors mutation

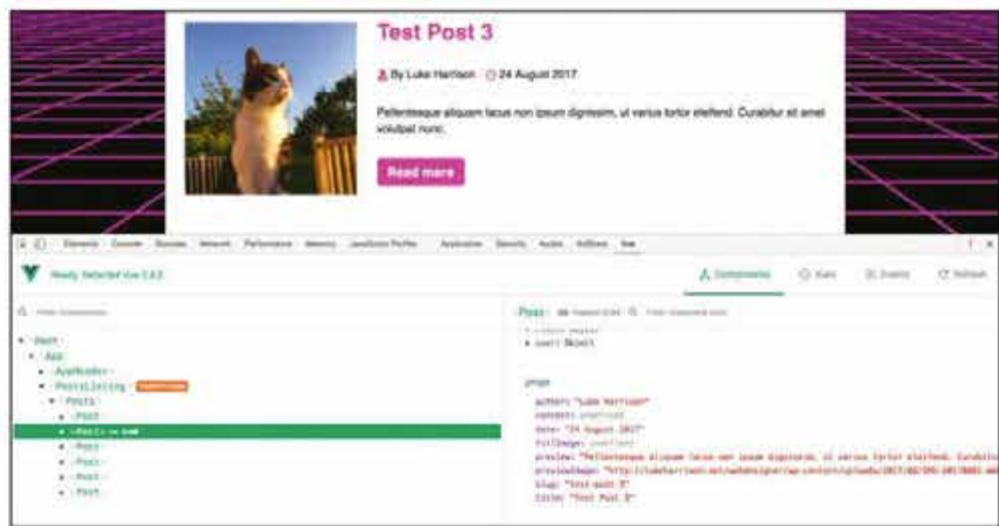
Within the mutations object of the Vuex store, create the 'storeAuthors' mutation using the snippet below. Like with 'storePosts' from step 8, this takes the payload its passed and sets it as the value of the authors property in our store's state.

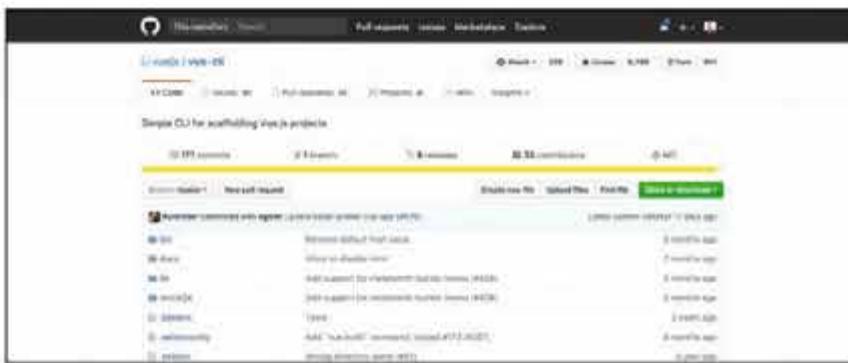
```
storeAuthors(state, response) {
  state.authors = response }
```

16. Trigger getAuthors on load

We need to get the author info from WordPress as soon as the app begins to load. Let's amend the top level component 'App.vue' again and dispatch the 'getAuthors' action in the same 'created()' lifecycle hook as the 'getPosts' action.

```
created() {
```





Introducing vue-cli

This Vue.js powered blog is built upon a boilerplate template provided by vue-cli, a simple CLI (command line interface) for scaffolding vue.js projects. There's a variety of official templates to choose from, of varying sizes, each suited to a particular type of project.

I recommend checking out the webpack template, which is in use here, as it provides a great insight

into how the application should be structured, and also takes care of webpack configuration, which has a steep learning curve for the uninitiated. You also get some invaluable extras built into the template such as vue-loader setup with hot reload, linting, testing and CSS extraction.

You can see the full documentation at github.com/vuejs/vue-cli.

```
this.$store.dispatch('getAuthors')
this.$store.dispatch('getPosts' ) }
```

17. Create getUsername method

Now we're querying WordPress for author information on load, all we need to do is define a method in our posts component which lets us pass an author id and get a name in return. Copy the snippet below into the posts component's methods object, below the existing 'getSinglePost' method.

```
getUserName: function(userId) {
  var username = 'unknown';
  this.$store.state;
  authors.filter(function(user) {
    if(user.id === userId) {
      username = user.name
    }
  });
  return username;
}
```

18. Call getUsername method

Now we just need to call 'getUserName'. Still in the posts component, in the template, replace the author attribute's reference to 'post.author' so it reflects the snippet below. The author's name should now be correctly displaying for each post.

```
:author="getUserName(post.author)"
```

19. Blog loading

As we're loading the post data asynchronously, there is a moment before the request completes where the application is empty. To counter this, we'll implement a loading state which is active until the blog is fully populated. In the posts component, paste the snippet below just after the opening <script> tag to import the icons we'll be using.

```
import icon from '@/components/icons/icons'
import loading from './../../../assets/img/loading.svg'
```

20. Add icon to components list

Next, still within posts, add a reference to icon in the components objects. This makes the posts component aware of our recently imported icon component.

```
components: {
  icon,
  post }
```

21. Create loading elements

We now just need to add the loading elements to the posts template so it shows up on the page. Firstly, wrap the second div in the snippet around the two divs with the 'v-if' directives to make sure no posts show up until loading is complete. Then add the first div from the snippet above it. This contains the loading icon and a 'v-if' directive which means it's only going to be visible until the point where the app is fully loaded. Once done, loading should now be implemented.

```
<div v-if="!this.$store.state.posts"
class="u-align-center">
  <icon class="c-icon-loading"
  use="loading"></icon>
</div>
<div v-if="this.$store.state.posts">
  [...]
</div>
```

22. Update single post attribute paths

The only thing left to do is to make sure single posts are correctly setup so they are using the WordPress post data in the Vuex store. The first step is to update the attribute paths in the posts component template within the 'v-if="this.type === 'single''' div, which handles the display of single posts.

1. Switch 'singlePost.title' to 'singlePost.title.rendered'
2. Switch 'singlePost.author' to 'getUserName(singlePost.author)'
3. Switch 'singlePost.fullImage' to 'singlePost.acf.header_image'
4. Switch 'singlePost.content' to 'singlePost.content.rendered'

23. Refactor getSinglePost method

We also need to refactor the posts component's 'getSinglePost' method. It needs to return a promise which dispatches the 'getPosts' action. In the follow up 'then' function, we'll search the Vuex store's posts for an entry with a slug matching the one passed in the URL. If found, we'll copy the data to our component's local state and resolve the promise. If it isn't found, the promise will be rejected.

```
getSinglePost: function() {
  return new Promise
  ((resolve, reject) => {
```

```
this.$store.dispatch('getPosts')
```

```
.then(() => {
  var foundPost = false;
  this.$store.state.posts.
  filter((post) => {
    if(post.slug ===
    this.$route.params.slug) {
      this.singlePost = post;
      foundPost = true;
    }
  });
  foundPost ? resolve() : reject();
})
```

24. Refactor posts created0 hook

Next, we need to refactor the 'created()' lifecycle hook in the posts component. If we're needing to display a single post, the hook should call the 'getSinglePost' method from the previous step, and if its promise is rejected, send the user to the 404 'page not found' page. This is to account for scenarios where users enter a non-existent post slug in the URL.

```
created() {
  if(this.type === 'single') {
    this.getSinglePost().then(null, () => {
      this.$router.push({name: 'pageNotFound'});
    });
  }
}
```

25. Add v-if directive

The final step is to add the snippet below to the post component within the 'v-if="this.type === 'single''' div in the template. This directive means the post will only display when the local post data made available by the 'getSinglePost' method is populated. This is to stop Vue from prematurely rendering the component and thus causing errors.

```
<v-if="singlePost"
```

26. Build the app

Now with everything working in the console, cancel the 'npm run dev' command or open a new console and run the below command to generate a production ready version to upload to your own server. This will appear in the 'dist' directory.

```
npm run build
```

web workshop

Create a highlighted text knockout effect

As seen on badassfilms.tv/real/dante-ariola

Logo presentation

The logo is clearly presented in the top left corner with a colour that avoids conflict with the page content background imagery.

Unfocused text

Text items that are not the current focus of attention are still clearly readable through presentation with an identifiable outline.

The text highlight

Text appearing in the middle of the screen is gradually covered with the highlight colour as the user scrolls up or down towards it.



Call to Action text

Similar to the logo, the "call to action" text uses a colour that clearly distinguishes itself from the content background imagery.

Background imagery

All of the background images make use of darker colours to allow the lighter coloured foreground text content to be clearly visible.

Technique

1. Document initiation

The first step is to initiate the HTML document. This consists of the HTML container for storing the head and body sections. While the head section is responsible for loading the external CSS stylesheet, the body section will store the visible HTML content to be created in step 2.

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

2. Knockout SVG

SVG is used to create a "mask" for the knockout effect. A "rect" rectangle references the mask group by its ID. The mask has a rectangle to define its size, position and fill transparency - with white representing no transparency. The following text items appear masked out from the rectangle.

```
<svg class="knockout" width="100%"
      height="100%">
<rect class="knockout-bg" fill-opacity="1"
      mask="url(#knockout-text)" width="100%"
      height="100%" />
<mask id="knockout-text">
<rect width="100%" height="100%" fill="white"
      x="0" y="0"/>
<text x="50%" y="30%" fill="#000" text-
      anchor="middle">One</text>
<text x="50%" y="50%" fill="#000" text-
      anchor="middle">Two</text>
<text x="50%" y="70%" fill="#000" text-
      anchor="middle">Three</text>
</mask>
</svg>
```

3. Regular elements

The remaining requirements of the HTML is to define the highlight bar and an area for the main content to be placed. A span element with a "bar" class reference is used for the highlighter, while an article container is used for the main content area.

```
<span class="bar"></span>
<article>
<h1>title</h1>
</article>
```

4. Initiate stylesheet

With the HTML now complete, the next step is to initiate the CSS stylesheet, so create a new file called "styles.css". This step ensures that no border spacing appears in the HTML document container or its body section by setting their padding and margin to zero.



<comment>
What our experts think of the site

Less is more

"It can be tempting to fit as much content as possible into each frame, but this will lead to reduced impact with users. Similar to how you would approach creating content on Twitter, keep your content short and direct. Also, keep a focus on what type of user response you want each tile to achieve."

Leon Brown

```
html, body{
padding: 0;
margin: 0;
}
```

5. Highlight bar

The highlight is required to display statically behind all content; achieved with fixed positioning and a negative z-index. The top and left position remain fixed to the screen regardless of where the user scrolls, giving the illusion of text changing colour when the SVG mask is scrolled over it.

```
.bar{
display: block;
position: fixed;
width: 100vw;
height: 25vh;
top: 10vh;
left: 0;
z-index: -1;
background: red;
}
```

6. Knockout container

The knockout SVG element needs to be set to a specific size. In this case, we want it to cover the full width of the available screen. We also want to avoid any margin being placed on the element that may make it appear out of place.

```
.knockout {
height: 100vh;
width: 100%;
margin: 0;
}
```

7. Knockout text

Text elements inside the knockout container are required

to have a font size. This step sets this size to five times bigger than the default document font. The size specified in this step will be used where no other font size are directly applied to text elements.

```
.knockout text{
font-size: 5em;
}
```

8. Knockout background

The background cover of the SVG that the mask is applied to requires settings for its colour, size and position. With this being an SVG element, the CSS attributes used are those specific to SVG and not the regular HTML style attributes for regular web page elements.

```
.knockout-bg{
fill: black;
width: 100%;
height: 100%;
x: 0;
y: 0;
}
```

9. Article styling

The final requirement is to make sure that the highlight bar used for the effect is not visible when the SVG mask has scrolled out of view. The easiest way to achieve this is to set the article content container to have a solid background colour that will cover the highlight bar.

```
article{
display: block;
min-height: 100vh;
background: #fff;
margin: 0;
}
article h1:first-child{
margin-top: 0; }
```





THE BEST APIs YOU NEED TODAY

**DISCOVER THE HTML AND THIRD-PARTY APIs THAT ARE
THE BUILDING BLOCKS OF THE NEXT-GENERATION WEB**

WHAT MAKES SOMETHING AN API?

WHAT IS THE CRITERIA TO BE AN API

When you're writing code in any language, you'll almost certainly be using a range of APIs without knowing. In its simplest form, an API (Application Programming Interface) is a clearly defined specification for communication between two software components. APIs can take various forms, but will typically include some combination of functions or methods, and data structures. Essentially they provide pre-fabricated functionality which you as a programmer can utilise to build your own application. Java ships with a core set of APIs within its `java.*` packages, which provide the basic functionality of the

language which all Java programmers are familiar with.

It's these APIs which enable us to create meaningful functionality within software we write. Of course, software components have needed to communicate since the dawn of computing, but what leads us to describe something as an API is the concepts of standardisation, openness, and re-use, where the underlying functionality is exposed along with some documentation which allows anyone to come along and use it. If you work in tech, it's quite likely that your company will be developing its own ecosystem of APIs.

WHAT IS A BROWSER API?

DISCOVER THE DIFFERENT

Web development is a rapidly-changing environment, and there's a continuous effort by standards bodies and browser makers to introduce new APIs to the web which make building sites and apps easier, or bridge the gap between the browser and native software. Following the latest progress and adopting new APIs when well-enough supported can make your own life easier, and ensure a modern experience for your users.

SIMON JONES

Software engineering director

In the context of the web, you might already have thought about APIs, in the form of services made available by third parties. Google, for example, expose a Google Maps API which you can consume as a developer to integrate Google's mapping, street view, navigation and more into your app. This involves importing scripts hosted on Google servers, then utilising objects and functions from those scripts in line with the API documentation which they provide. APIs like this are generally what we'd refer to as "server-side APIs".

However, what you might never have considered is that the basic JS

functionality built into the browser is also composed of a set of APIs, which are constantly being updated. These "client-side APIs" provide many of the basic features taken for granted within JS, such as manipulating the DOM, embedding graphics/sound, or interfacing with the device you're running on. More likely than not you'll have read documentation for many of them online when building a site/app. These APIs are typically based on a single specification which ensures that all browser makers implement the functionality consistently, thus ensuring that your code works across any browser.

APIs IN COMMON USE TODAY

CANVAS & SVG

<http://bit.ly/2h4za9q>

<http://bit.ly/2wxqWfK>

Canvas and SVG are one of the most obvious examples of widely-used browser APIs. Both canvas and SVG expose a straightforward set of entry points and JavaScript functions to fulfil their respective objectives. These days we can add the more recent WebGL to the mix for 3D graphics, which we'll look at later on.

XMLHTTPREQUEST

<https://xhr.spec.whatwg.org>

XMLHttpRequest has for some time been the JavaScript API used to send asynchronous requests to a server. This can allow you to, for example, retrieve data from the server and update a page without having to reload the full page. Its name is an anachronism since it works with JSON as well as XML. It's likely to be superseded by the more recent Fetch API over time.

THE ENTIRE JAVASCRIPT ECOSYSTEM IS CONSTRUCTED FROM APIs, MANY OF WHICH YOU'VE LIKELY ALREADY ENCOUNTERED

WEB SOCKETS

<http://bit.ly/2lebN1p>

Web sockets enable a two-way communication session to be opened between a browser and server. This enables the server to provide updates to the user without the browser needing to poll the server at an interval. Web sockets are in use on a number of sites today; on Stack Overflow they're used to feed live notifications of responses to questions.

DOM FUNCTIONS

<http://html5index.org>

This may seem like a cop-out, but it's important to think about. Even core JavaScript functions do things like manipulate the DOM. Think of the document object as an entry point. This means while they may be defined in separate specifications, there's little fundamental difference between the new APIs we'll look at below and core JavaScript functions that have been around for years.

HTTP CACHING

We've used the `-c-1` argument to tell the web server to prevent the browser caching content. This makes it easier to see the effect of our service worker; otherwise, the HTTP cache can override the expected service worker behaviour.



Welcome to the service worker demo!

On this page, we'll demonstrate the use of service workers to serve up content while offline. Our site is designed to show users a picture of a cat.



APIs USED

SERVICE WORKER API

<https://mzl.la/1nA8Zr3>

OLD OR NEW? TEST

As you modify your service worker, you'll find that you can't always rely on refreshing the page or restarting the server to update it. This is because the old service worker remains registered in the browser in the background. Wouldn't it be useful to be able to see a list of active service workers, and terminate them when needed? Fortunately, Chrome has a couple of pages you can visit which let you do just this. As you test, it's generally worth closing down existing workers each time you make changes.

```
chrome://serviceworker-internals/  
chrome://inspect/#service-workers
```

BUILD APPS WHICH WORK OFFLINE

EMPLOY THE SERVICE WORKER API TO BRING WEB APP DEVELOPMENT CLOSER TO NATIVE

For a long time, offline functionality, background synchronisation and push notifications has differentiated native apps from their web counterparts. The Service Worker API is a game-changing technology which evens the playing field. In this tutorial, we'll use it to build a page which can serve up content even while there's no Internet connection.

1. AN HTTPS SERVER

The easiest way to think about service workers is as a piece of code, which is installed by a site on a client machine, runs in the background, and subsequently enables requests sent to that site to be intercepted and manipulated. Because this is such a powerful capability, to work with service workers in a live environment you need to be running over https. This ensures that they can't be exploited, by making sure the service worker the browser receives from a page is genuine. For development purposes, however, we can run

without https since http://localhost/ is permitted as an exception to this rule. The simplest way to get started is with the npm http-server package.

```
npm install http-server -g  
http-server -p 8000 -c-1
```

2. BASIC PAGE

There's nothing on the server right now, so let's make a basic page to serve up. We'll create a new index.html file, and when we run the server it will now be accessible at http://localhost:8000

At this stage, you'll find that if you terminate the http server and refresh the page in the browser, you'll get an error page since the site can't be reached. This is entirely expected since we haven't cached any offline content yet.

```
<!DOCTYPE html>  
<html>  
  <head>  
    <meta charset="UTF-8" />  
    <title>Service Worker</title>  
    <script src="site.js"></script>
```

```
  <script>  
    <link rel="stylesheet" type="text/css" href="custom.css"/>  
  </head>  
  <body>  
    <header>  
      <h1>Welcome</h1>  
    </header>  
    <div id="content">  
      <p>Content here</p>  
        
    </div>  
  </body>  
</html>
```

3. REGISTER SERVICE WORKER

We've now got a fairly unremarkable page running, and it's time to start thinking about implementing a service worker. Before we get coding, it's worth taking a moment to understand the lifecycle of service workers. The process kicks off with the 'registration' of a service worker in your JavaScript, which tells the browser to start

installing the worker – the first step of its lifecycle. Throughout their lifecycle, a service worker will be in one of the following states:

installing: once a service worker has been registered, its installation is typically used to download and cache static content

installed: once the worker is installed, it is theoretically ready for use but does not immediately activate

activating: an installed service worker will activate itself if either there is no existing service worker, or certain conditions lead the existing one to expire; activation is typically used to clear old files from cached offline content

activated: the service worker now has control over the document, and can handle requests

redundant: a service worker may be redundant if it failed to install or activate, or if it is replaced by a newer service worker.

4. ARE YOU REGISTERED?

Let's register a service worker. This effectively points the browser to the JavaScript file which defines the service worker's behaviour.

Registration is done using the serviceWorker object which is the entry point to the API. We'll also check the API is actually present in the browser before trying to do so. The register() function can safely be called every time the page loads, and the browser will determine whether the service worker has already been registered.

```
if ('serviceWorker' in navigator) {
  window.addEventListener('load', function() {
    navigator.serviceWorker.register('./serviceworker.js', {scope: './'}).then(function(registration) {
      console.log("Service worker registered successfully.");
    }, function(error) {
      console.log("Error registering service worker: " + error);
    });
  });
}
```

5. IMPLEMENT SERVICE WORKER

Next we need to implement the service worker itself. Service workers can listen for a range of events related to their own lifecycle and activity on the page. The most important ones are 'install', 'activate' and 'fetch'. Let's start by creating a listener for the 'install' event, which triggers once the worker's installation is completed. This enables us to instruct the service worker to add some offline content

in the current folder to a cache. We also need to name our cache – since old caches can persist, updating/versioning this cache name enables you to serve up newer versions of content later on.

```
var currentCache = 'demo-cache';
self.addEventListener('install', event => {
  event.waitUntil(
    caches.open(currentCache).then(function(cache) {
      console.log("Adding content to cache.");
      return cache.addAll([
        './index_offline.html',
        './kitty_offline.jpg',
      ]);
    }));
});
```

EXPERIMENTAL TECHNOLOGY

Service workers are not yet supported by all browsers, so you'll need to use Firefox, Chrome or Opera to test them out. This also means they aren't quite ready for production usage yet.

that your page works both online and offline. It's likely, however, that you'll want more intelligent offline behaviour, with different content or functionality available when the server is unavailable. To achieve this,

we can extend our 'fetch' event response further to check specifically for navigation requests and respond with a different offline page when one is detected. This index_offline.html file can be a variation of your online page, or something completely different, and can also use other resources you've cached such as custom.css.

```
self.addEventListener('fetch', event => {
```

```
  './custom.css'
]);}
});}
});
```

6. FETCH EVENT

Our page will now cache content when loaded, but we need some mechanism to intercept requests and redirect them to this cache. To do this, we need to listen for 'fetch' events, which are triggered when a request such as obtaining our index.html file is made across the network. We then match the request against the cache, and serve up the cached resource if found. Otherwise, we fall back to a Fetch API request to the server. It's worth at this point noting that we have a heavy dependency on JavaScript Promises to work. These can be a little tricky, so are worth familiarising with if you haven't used them before.

```
self.addEventListener('fetch', event => {
  event.respondWith(
    caches.match(event.request).then(response => {
      return response || fetch(event.request);
    })
  );
});
```

7. EXTEND FETCH EVENT

If you test it out now (terminate the http server and refresh the page), you should find

```
if (event.request.mode ===
  'navigate') {
  event.respondWith(
    fetch(event.request));
  catch(error => {
    console.log("Page unavailable. Returning offline content.");
    return caches.match('./index_offline.html');
  })
} else {
  event.respondWith(
    caches.match(event.request).then(response => {
      return response || fetch(event.request);
    })
  );
}
});
```

8. DELETE CACHE

There's one more thing we need. If you now try modifying your offline content, you'll find it doesn't update when you test out your page – you still get the old version! This is because the older files are still cached.

You need to implement something to clean out outdated files from the cache to prevent them being served up. This is done by responding to a 'activate' event and deleting all caches which do not match the name specified in currentCache. You can then add a version number to currentCache each time you modify your offline content, to ensure it is refreshed.

```
this.addEventListener('activate', event => {
  var activeCaches =
  [currentCache];
  console.log("Service worker activated. Checking cache is up-to-date.");
  event.waitUntil(
    caches.keys().then(keyList => {
      return Promise.all(keyList.map(key => {
        if (activeCaches.indexOf(key) === -1) {
          console.log("Deleting old cache " + key);
          return caches.delete(key);
        }
      }));
    })
  );
});
```

APIs USED

WEBGL

<https://www.khronos.org/webgl/>

THREE.JS

<https://threejs.org>



7 BROWSER APIs TO WATCH

NOT ALL OF THESE ARE READY FOR PRIME-TIME YET, BUT THEY'RE WORTH WATCHING FOR USE IN THE FUTURE

GEOLOCATION API

Let's start with one that is well supported. You might have seen pages request permission to use your location already. The Geolocation API, accessed through the `navigator.geolocation` object, enables a site or app to determine the location of a user based on a combination of information such as GPS, wifi and mobile signal. It's supported in all major browsers, although only usable with a connection over https.

PUSH API

We've touched briefly on the Push API already. Not to be confused with mobile push notifications, it enables web apps to receive messages pushed from a server, even if the application is not in focus or even if the application is not loaded. To do this, it depends very much on a service worker to be running in the background. There are a range of interesting applications here for providing real-time updates within web apps.

NOTIFICATIONS API

The Notifications API allows pages to display notifications outside the context of the browser. The user will need to grant explicit permission, but once done, notifications can be generated by creating a new `Notification()` object. This is widely supported on desktop, though not on mobile, but particularly interesting are the possibilities that could emerge from combining this with the Push API to enable background notifications to a user.

WEBVR API

WebVR is, as you might expect, designed to add support for virtual reality devices such as the Oculus Rift and HTC Vive within the browser. This would allow web developers to directly utilise position and motion data from the device for use on a page (most likely in tandem with WebGL). This could be very exciting for future projects, but is experimental at this time, with limited support (by device) in Edge, Firefox and Chrome.

GET STARTED WITH WEBGL USING THREE.JS

THE WEBGL API BRINGS 3D GRAPHICS TO THE BROWSER, NO NEED FOR PLUGINS. GET THE CODE FROM FILESILO.CO.UK



WebGL, which is widely supported on all modern browsers, enables you to work with hardware accelerated 3D graphics in JavaScript, which opens up a huge range of possibilities for browser-based apps and games. WebGL itself is rather low-level, and it's unlikely you'll want to work with it in its vanilla form. There are a range of libraries and even game engines available to provide higher-level functionality, but for this tutorial we're going to use three.js, which is a free open-source library providing abstraction of WebGL.

1. GET THREE.JS

You'll need to start by getting hold of three.js, which is available at <https://threejs.org>. Once you have the latest build, place three.js into your project folder. We'll then add it as a script to our page like any other JavaScript library. We'll place our own code into a separate JavaScript file.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset=utf-8>
    <title>Getting started with
    three.js</title>
  </head>
  <body style="margin: 0;">
    <script src="three.js"></script>
    <script src="demo.js"></script>
  </body>
```

</html>

2. GET A SCENE

We need three things to get started with WebGL: a scene, a camera and a renderer. A scene is where we place objects to be displayed by three.js. A camera is the point of view from which we will see them. The renderer brings the two together and draws the screen accordingly. Once we have these set up, we add the renderer to the document.

```
var width = window.innerWidth;
var height = window.innerHeight;
var viewAngle = 45;
var nearClipping = 0.1;
var farClipping = 9999;
var scene = new THREE.Scene();
var camera = new THREE.PerspectiveCamera( viewAngle,
  width / height, nearClipping,
  farClipping );
var renderer = new THREE.WebGLRenderer();
renderer.setSize( width, height );
document.body.appendChild(
  renderer.domElement );
```

3. CREATE A LOOP

Next, we need to create a loop to actually render our scene. We do this using the renderer.render() function, but the key part is to recursively use requestAnimationFrame(), which is a built-in function which enables the browser to request another frame when it is ready to draw one.

Using requestAnimationFrame() is easier and results in smoother animation than trying to time drawing of frames yourself.

```
function animate() {
  requestAnimationFrame( animate );
  renderer.render( scene, camera );
}
animate();
```

4. BASIC OBJECTS

Now we can start adding some objects to our scene. We can do this by creating Mesh objects and adding them to it. A mesh is composed of geometry (the shape of the object), and a material (the colour or texture used to paint it). We'll create some basic objects by defining three different geometries, and assigning different colour materials to them.

```
var cubeGeometry = new THREE.BoxGeometry( 1, 1, 1 );
var cubeMaterial = new THREE.MeshLambertMaterial( { color:
  0xffff00 } );
var cube = new THREE.Mesh(
  cubeGeometry, cubeMaterial );
var coneGeometry = new THREE.ConeGeometry( 0.5, 1, 4 );
var coneMaterial = new THREE.MeshLambertMaterial( { color:
  0x00ff00 } );
var cone = new THREE.Mesh(
  coneGeometry, coneMaterial );
var sphereGeometry = new THREE.
```

VIBRATION API

We may be into slightly dubious territory now. Invoked via the navigator.vibrate() method, the vibration API is arguably a necessity on the quest to bring web apps closer to native. However, it's also quite obviously prone to abuse, and in its early days there were sporadic reports of ads using it without consent to attract the user's attention. It's currently supported in Edge and Safari, so iPhone users are out of luck.

BATTERY STATUS API

Here's an example of when APIs don't make it. This API was designed to do more or less what you might expect by its name. It introduces a navigator.getBattery() method which returns a promise with a battery object, which can then be used to listen for various events related to battery status. It's currently supported in Chrome and Opera only, and isn't likely to make it much further due to privacy concerns.

ENCRYPTED MEDIA EXTENSIONS

If you want to see what controversy looks like in the world of web APIs, look no further than Encrypted Media Extensions. EME allows HTML5 video playback of DRM protected content. Objections have been raised to this on the basis that it introduces proprietary components into the predominantly open and free ecosystem of the web. Regardless of your perspective, the specification is here to stay.

GET MORE INFO

GEOLOCATION API

<http://bit.ly/2iyrYG2>

PUSH API

<http://bit.ly/2uXFnYM>

NOTIFICATION API

<http://bit.ly/2yNk2XI>

WEBVR

<https://webvr.info>

VIBRATION API

<http://bit.ly/2iyzElz>

BATTERY STATUS API

<http://bit.ly/2zBRTjf>

ENCRYPTED MEDIA EXTENSIONS

<http://bit.ly/1TiR9q2>

10 BEST APIs



```
SphereGeometry( 0.5, 8, 8 );
var sphereMaterial = new THREE.
MeshLambertMaterial( { color:
0x0000ff } );
var sphere = new THREE.Mesh(
sphereGeometry, sphereMaterial
);
```

5. SPECIFY A POSITION

By default objects are added at the origin ($x=0, y=0, z=0$) of the scene, which is also where our camera is, so we'll need to specify a position for them as well. We're then ready to add the meshes to our scene, which will mean they are automatically rendered in the `animate()` loop.

```
cube.position.x = -2;
cube.position.z = -5;
cone.position.z = -5;
sphere.position.z = -5;
sphere.position.x = 2;
```

```
cube.position.z = -5;
scene.add(cube);
scene.add(cone);
scene.add(sphere);
```

6. ADD A LIGHT SOURCE

If you view your page now, you'll find things are looking a little flat. There's no lighting applied to the objects, so they are solid primary colours and look two-dimensional. To fix this, we need to switch from `MeshBasicMaterial` to a material which supports lighting; we'll use `MeshLambertMaterial`. We also need to add a light source to the scene.

```
var light = new THREE.
PointLight(0xFFFFFF);
light.position.x = 0;
light.position.y = 10;
light.position.z = 0;
scene.add(light);
```

7. MOVE THE SOURCE

Now we're getting there! You should see what are quite evidently three-dimensional objects on your page. What we haven't yet done is taken full advantage of the `animate()` function. Let's make a few changes to have our light source move around in a circular motion above the objects.

```
var lightAngle = 0;
function animate() {
  lightAngle += 5;
  if (lightAngle > 360) {
    lightAngle = 0;};
  light.position.x = 5 * Math.
cos(lightAngle * Math.PI / 180);
  light.position.z = 5 * Math.
sin(lightAngle * Math.PI / 180);
  requestAnimationFrame( animate
);
  renderer.render( scene, camera
);}
```

8. ADD TEXTURE

In practice, we probably don't want our objects to be flat colours. It would be more typical to apply some texturing to them from a file. We can do this using `THREE.TextureLoader()`. Let's change how our cone object is created to utilise a texture we've loaded from a file. The function is called when the file is loaded.

```
var textureLoader = new THREE.
TextureLoader();
textureLoader.load("./grass_
texture.jpg", texture => {
  var coneGeometry = new THREE.
ConeGeometry( 0.5, 1, 4 );
  var coneMaterial = new THREE.
MeshLambertMaterial( { map:
texture } );
  var cone = new THREE.Mesh(
coneGeometry, coneMaterial);
  cone.position.z = -5;
  scene.add(cone);
},
```

9. MAKE IT NATURAL

Things are looking better, but something still isn't quite natural. The texture looks very flat and doesn't respond to the lighting. We can solve this through the use of bump mapping, which lets us use light and dark parts of an image to simulate texturing on the surface of an object. Let's try this out with a different texture on the sphere. We'll switch the material to `MeshPhongMaterial` which allows us to specify a `bumpMap` attribute.

```
var textureLoader = new THREE.
TextureLoader();
textureLoader.load("./bump_map.
```

5 POPULAR 3RD PARTY APIs

THESE ARE SOME OF THE MOST HIGHLY USED APIs OUT THERE. YOU'LL TYPICALLY NEED AN API KEY FROM THE CREATOR TO UTILISE THEM

GOOGLE MAPS

Google Maps offers an immensely popular set of APIs which bring mapping capabilities, street view, navigation and more to millions of sites. Google offers a multitude of APIs by platform and function. The simplest are basic page embeds to display a map on a page, but there are also numerous JSON web services which enable you to, for example, receive a set of directions for navigation between locations.

FACEBOOK

Facebook offers something called the "Graph API" which enables website and apps to get data from and write data to Facebook. Working with the Graph API involves the representation of Facebook's content as a set of nodes (and associated data fields) and edges in a graph structure. They also offer a Graph API Explorer which let authenticated users test out the results of various calls to the API.

TWITTER FOR WEBSITES

Twitter offer several APIs for different use cases. The most straightforward is the Search Tweets API, which allows searching of the vast volume of historic tweets over a specified period of time and set of criteria. However, web developers may be most interested in Twitter for Websites, which provides a simple way to embed Twitter functionality into the front-end. Such as a simple Tweet button.

YOUTUBE

The YouTube API enables you to embed YouTube functionality in your page, and like others on this list, allows you to retrieve content from and post content to the platform. Because of the high bandwidth required by its content, the YouTube API in particular is an interesting study in how developers limit the usage quota of APIs. - Google use a current of "units" to price various operations.

RESOURCES

SOME OF THE BEST PLACES TO LOOK TO KEEP UP-TO-DATE AS NEW APIs EMERGE

MOZILLA DEVELOPER NETWORK

<https://developer.mozilla.org/en-US/>

The Mozilla Developer Network (MDN) provides one of the most comprehensive sources of reference on web technology available. In addition to documentation of all the latest APIs, it also includes tutorials and code samples illustrating their usage. This is almost always a good starting point when learning something new.

W3C

<https://www.w3.org/standards/webdesign/script>

The W3C specifications aren't always easy going, and probably aren't for everyday usage. However, if you're interested in some of the detail on the specifications, want to follow what might come in future but hasn't yet been implemented in browsers, or are just curious about how the specs come about, it can be worth a browse.

HTML5.ORG WEB PLATFORM

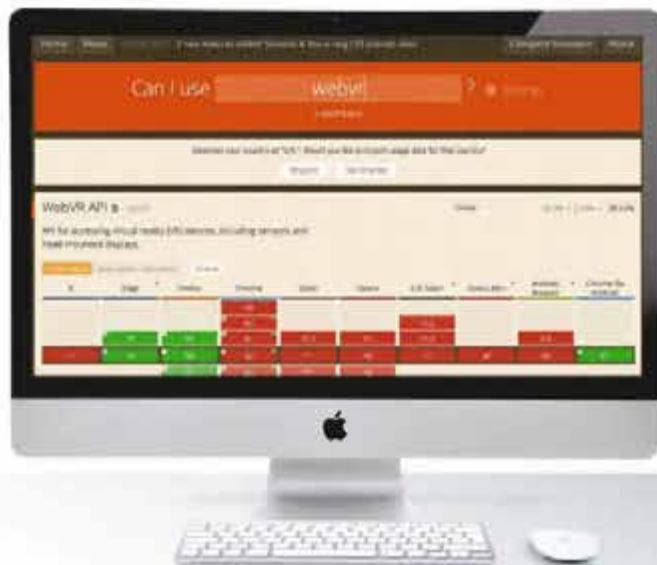
<https://platform.html5.org>

If you've ever found yourself confused by the growing list of technologies and APIs which make up the modern web, this page could be a great reference source. It lists all the components of the modern HTML, CSS and JavaScript platforms, including APIs, for your convenience. If nothing else, it's an admirable effort to bring a sense of order to an occasionally chaotic ecosystem.

CAN I USE...

<https://caniuse.com>

As we've noted, browser support for the latest APIs can be a challenge, and it's hard to know whether something is really ready for use. Can I Use provides an excellent source of reference on the state of browser support for APIs as well as other web features.



The Mozilla Developer Network (MDN) provides one of the most comprehensive sources of reference on web technology available

GET MORE API INFO

GOOGLE MAPS

<http://bit.ly/1xoVpqB>

FACEBOOK

<http://bit.ly/2gCSj1C>

TWITTER FOR WEBSITES

<http://bit.ly/14LSwat>

YOUTUBE

<http://bit.ly/1uJlOzO>

AMAZON S3

<http://amzn.to/2yLkdAN>

```
jpg", texture => {
  var sphereGeometry = new THREE.SphereGeometry( 0.5, 8, 8 );
  var sphereMaterial = new THREE.MeshPhongMaterial( { color: 0x0000ff, bumpMap: texture, bumpScale: 1.0 } );
  var sphere = new THREE.Mesh( sphereGeometry, sphereMaterial );
  sphere.position.z = -5;
  sphere.position.x = 2;
  scene.add(sphere);
},
);
```

10. ADD CONTROLS

As a final touch, let's also give the user a little control over the scene. If we want to add the ability to pan around, there's an extra library that ships with three.js which makes it incredibly easy to do just that. Make sure to extract OrbitControls.js from the three.js package to your project directory, and include it in your page. This is one of

several control libraries which ship with three.js to fulfil common styles of camera control.

```
<script src="OrbitControls.js"></script>
```

11. APPLY TO CAMERA

Now all we need to do is create a THREE.OrbitControls object and apply it to our camera. The library will take care of the rest for you: you won't need to listen for clicks, mouse movements, and so on. You might also want to move your objects back to the origin point, and offset the camera instead so that it can pan neatly around the objects. With that, we're done with our introduction. You should have three objects with varying styles of texture, some dynamic lighting, and a user-controlled camera to play with.

```
camera.position.z = 10;
var controls = new THREE.OrbitControls( camera );
```

AMAZON S3

Let's finish with one which may be less familiar. Amazon S3 (Simple Storage Service) is part of the Amazon Web Services offering, and offers REST APIs (and client-side SDK to simplify their usage) which enable you to place, modify and delete content on the storage. Many use cases for this are on the server side, but Amazon's tutorials cover uses like uploading photos to the cloud from the browser.



BROWSER CONSOLES

MOZILLA AND GOOGLE HAVE RECENTLY UPPED THEIR GAME BY OFFERING ADVANCED CONSOLES IN THEIR WEB BROWSERS. HERE WE SHOW YOU THE TWO DIFFERENT APPROACHES, AND HOW EACH CAN BENEFIT YOUR PROJECTS



WHAT DOES IT DO?

GOOGLE CHROME HAS HAD A CONSOLE TAB KICKING AROUND FOR AGES. HERE'S A CLOSER LOOK

Consoles can be used for different things: traditionally, web developers used them to output textual information which can be used for debugging or logging. Google's approach to web consoledom centres on improving the existing features - due to the fact that console tabs have been around for ages, the concepts behind them can not be so bad.

In fact, Google even went so far as to provide its mobile operating system with one of the most advanced logging features. If you ever used LogCat and have to go back to a lesser implementation, the pain stings.

Due to this, Google focuses the work of its team on features that improve the existing experience. For example, information can now be grouped to keep on top of an ever-growing amount of log data.

In short, Google Chrome's web console is the go-to point when data needs to be provided for further analysis. Chrome even provides a comfortable way to measure execution time - in short, a more than worthy featureset totally deserving of your attention.

The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The output area contains the following text:

```

oi!

✖ This is an error!
runMe      @ chromedemo1.htm:8
(anonymous) @ chromedemo1.htm:12

⚠ Some call it a warning...
runMe      @ chromedemo1.htm:9
(anonymous) @ chromedemo1.htm:12

Bye!
> |

```

LET'S GET STARTED

OPENING THE CONSOLE BY PRESSING CTRL+SHIFT+J IS ALMOST ALL YOU NEED

Once the console window is on the screen, it is time to start outputting log information. Instead of simply using `console.log`, you can now use a group of different functions.

Execute the following snippet of JavaScript code to generate the output shown in the figure:

```

<html>
<body>
<script>
function runMe(){
  console.log("oi!");
}
runMe()

```

```

console.error("This is an
error!");
console.warn("Some call it a
warning...");
console.log("Bye!");
}
runMe()

```

One look at the figure reveals that warn and error create differently-coloured lines in the log output: this is helpful, in that it allows you to mark relevant errors. Furthermore, the little error displayed on the left side of

warning and error lines provides a call stack with further information.

A really handy feature allows you to modify the display by passing in either CSS or printf-like parameters:

```

console.log("%cLarge, blue
text", "color: blue; font-size:
x-large");
console.log("%s has %d points",
"Sam", 100);

```

Keep in mind that console output can be saved any time. Simply right-click the console window and select Save.

KEEP AN EYE ON SYSTEM BEHAVIOUR

HUNT DOWN PERFORMANCE ISSUES WITH HELPER APIs

Performing runtime analysis requires us to create an inefficient system whose performance is to be logged. The fibonacci algorithm's complexity provides us with ample ways to create stupid and low-performance implementations such as this one:

```

function fib(num){
  var a = 1, b = 0, accu;
  while (num >= 0){
    accu = a;
    a = a + b;
    b = accu;
    num--;
  }
  return b;
}

```

In the next step, `fib()`'s execution time can be analysed with the

Web Console:

```

function runner(){
  console.time("Starting the
Fib!");
  fib(20000000);
  fib(40000000);
  fib(60000000);
  fib(80000000);
  console.timeEnd("Starting the
Fib!");
}
runner()

```

time and `timeEnd` take a string parameter that identifies the counting process triggered. Invoking `timeEnd` causes the console to emit a message containing both the key string and the time elapsed from start to end.

Please keep in mind that all microbenchmarks - such as the one above - are, by definition, inaccurate. Getting meaningful and comparable data out of them requires significant scientific rigour, which is a point for another day.

MARK ME UP!

Simply collecting execution times often is not enough. The Chrome Web console, furthermore, allows you to log when specific events occur. Let us try this out by creating the following new function:

```

function runner(){
  console.timeStamp("Go!");
  fib(20000000);
  console.timeStamp("fib8");
}

```

```

fib(80000000);
}

```

Careful onlookers might have seen that the timeline of the browser is populated when the primitive `fib` example is run. The figure accompanying this step shows the zoomed-in view: we can discern which `fib` invocation takes how much time. Sadly, this can not be combined with `normal time()` recording due to oddities in Chrome.

FASTER BY NON-EXISTENCE

Logic tells us that the fastest code is one which never runs. The `count()` function allows you to find out exactly which method gets invoked and how often.

IN-DEPTH FEATURES

CHROME'S FEATURES CAN RIVAL A DEBUGGER. HERE'S THE BEST OF THEM...

Traditionally, analysing objects and DOM trees in depth is a job for a debugger. Sadly, this requires you to place breakpoints etc - in many cases, interesting situations cannot be recreated at will.

Google Chrome's take on the problem gives developers a set of tools that allow them to log information in a compact, yet all-encompassing fashion. First of all, groups can be created - these can then be collapsed and opened at will:

```
<html> <body> <script>
  console.group("Authentication
phase");
  console.log("Authenticating");
  console.groupEnd();
  console.group("Analysis
phase");
  console.log("Analyzing");
  console.groupEnd();
</script></body></html>
```

When this program is run, the various sections can be folded in and out. Next, let us proceed to generating an array of a bunch of objects:

```
var feld=new Array[];
feld[0]=new Dummy("Alpha",
"ALPHA", 2);
feld[0]=new Dummy("Beta",
"BETA", 3);
feld[0]=new Dummy("Gamma",
"GAMMA", 4);
```



```
> 1+1
< 2
> feld[2]
< * Dummy {a: "Gamma", b: "GAMMA", c: 4}
  a: "Gamma"
  b: "GAMMA"
  c: 4
  > proto_ : Object
> |
```

The Web Console allows you to analyse variables

ESSENTIAL ADVICE

A FEW HINTS TO GET THE MOST OUT OF YOUR CONSOLE EXPERIENCE

PRINTF SUBSTITUTION

<http://bit.ly/2zJFHOV>

Just like its great C-based ancestor, Chrome's log call takes substitution patterns. %s outputs a string, while both %i and %d will output integers. Using %f allows you to access the maximum accuracy of the underlying floating-point runtime. Using %o and %O - mind the capitalisation, please - modifies the output of objects. The lower-case O is responsible for a DOM object, while the upper-case O outputs a JSON element with clearly-visible properties. Finally, %c can be used to pass in even more advanced CSS formatting commands.

1

LIMITED COMPLAINING

<http://bit.ly/2iEP7pv>

Flooding the console window with useless information telling you that "everything is OK" is not sensible. Fortunately, an assert function not dissimilar to the one from C is implemented in the console object:

```
function greaterThan(a,b) {
  console.assert(a > b, {"message": "a is not
greater than b", "a":a,"b":b}); }
```

Chrome will log a message only if the provided condition turns out to be false. Please be aware that failed assertions will be shown as errors - which can be annoying at times.

2

\$

<http://bit.ly/2zvX0Id>

Web Console will 'pick up' elements selected in the JavaScript and DOM tree analysis utilities. These are then provided via a set of five variables: \$_ provides the very latest element, while \$0 to \$4 provide a historic reference. If jQuery is not loaded, you can even use \$(selector) to run a query against the currently-loaded page element. These results can, then, of course be passed on to other functions. \$\$(selector) can even be used to generate an array of all results, which is ideally suited to table() and the likes.

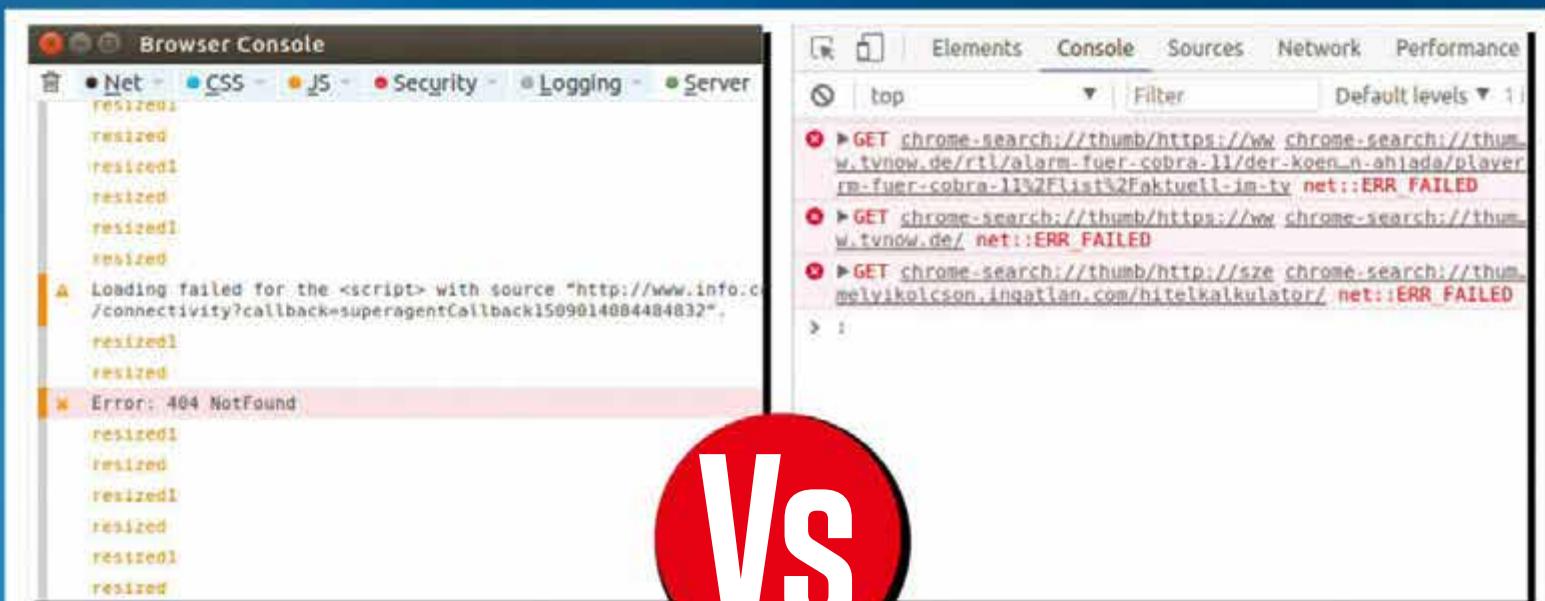
3

MONITORING

<http://bit.ly/2pxFUOJ>

Finally, both monitor and monitorEvents() can be used to set up monitors that trigger log updates whenever the element of concern is invoked or activated. Be careful that monitor can not be accessed directly from your application code. Monitors must, instead, be set up at runtime by entering the relevant command into an open console window. For example, key events on the window object could be analysed by entering monitorEvents(\$0, "key");.

4



BROWSER CONSOLE

While Google's Chrome Web Console definitely also provides some dynamic features such as the analysing of expressions, the primary focus of the product lays firmly in the collection of profiling data that can be analysed at a later date. In short, Chrome Web Console is ideally suited if you want to grab the log file, a cup of tea and some shortbread - the dynamic analysis features, which are described at <https://developers.google.com/web/tools/chrome-devtools/console/expressions> and in some of the API boxes, are not the main focus of the product. This does not mean that they

are not useful - if you run an all Chrome shop and don't have Firefox installed on any of your machines, by all means do take a look at the analysis features and do integrate them into the product at hand.

When integrated into a product intelligently, users can run the instrumented version almost as if it were a deployment build. If problems arise, the console can be opened and a screenshot/log file can be collected on the fly. This means that test results are always collected, and that no issues are encountered when reproducing errors.

WEB CONSOLE

The prevalence and the speed of the development of web technologies ensure that Mozilla's Web Console offers many, if not most of the extensions found in Chrome. While code using Google's extension methods will usually run without issues, the display does not offer the same level of comfort. For example, warnings are provided with less additional syntactical sugar. This means that dumping large objects or fields leads to less visually pleasing output - with a bit of elbow grease, relevant elements can still be found. Web Console can truly shine

whenever a developer needs to interact with a website's content. Powerful queries can be run against various elements, allowing you to limit the results shown to material significant for the application at hand. Due to the tight integration with the browser's debugger engine, you can, furthermore, use the console to limit the scope of the analysis feature. One classic application would be using a targeted query to sniff out an interesting part of the DOM, which is then analysed further using the traditional inspector tools found in the Firefox OS developer tools.

INSIDE THE DOM TREE THE CONCEPT OF THE CURRENT WORKING DIRECTORY IS OF FUNDAMENTAL IMPORTANCE TO CONSOLES

Traversing a DOM requires the presence of one. The following steps will use a predefined one consisting of an iframe and some additional syntactic sugar. In addition to that, the web developer edition will be used in lieu of the normal one. After loading the file, press Ctrl+Shift+K to open the console window - one nice aspect of the developer edition is that the developer tools pane is open only in the one tab where it was requested.

Just like in Web Console, the double caret at the bottom can be used for entering various commands. They will be evaluated in the context that the browser is currently running. This is especially interesting as the console remains available during a

debugger halt - you can even interact with local variables.

MASTER OF IFRAMES

Our example includes an iframe, which is aptly assigned an ID value of 'frame1'. Mozilla assumes that websites containing iframes are complex - due to that, a cd0 function is provided. It can be used to set the context of the page - the following command first changes into the iframe, and then out of it again:

```
cd("#frame1");
cd()
```

Sadly, this feature currently is limited to iframes; it is not possible to step into a normal DOM node. However, the DOM can be queried using a jQuery-like

syntax and the \$0 function. Be aware that this is valid only if no framework occupies \$0 - should this be the case, the console will always invoke the framework-provided method over your custom one. A simple example would be using the # operator to look for controls whose ID matches:

```
$("#knob2")
<button id="knob2">
$("#knob1")
<button id="knob1">
```

RESET TO NEW

Recent versions of Firefox store the command history between browser invocations. Should you ever want to clear the console's memory, simply enter the function clearHistory().



"ONE NICE ASPECT OF THE DEVELOPER EDITION IS THAT THE DEVELOPER TOOLS PANE IS OPEN ONLY IN THE ONE TAB WHERE IT WAS REQUESTED"

ACCESS DOM ELEMENTS QUICKER

USE THE VARIOUS SELECTORS TO GET ACCESS TO DOM ELEMENTS

The first step to modifying an element involves getting a reference to it. Ideally, store it in a local variable to make further accesses easier. The following snippet first stores the button knob2 in a local variable named zknob2, and proceeds to changing its text:

```
zknob=$("#knob1")
<button id="knob1">
zknob.innerText="New Text"
"New Text"
zknob.innerText="New Text 2"
"New Text 2"
```

When entering the commands one by one, keep an eye on the button; its text will be changed as you modify the value of innerText. Modifying innerText, of course, is one of the

possibilities – all attributes are fair.

GROUP OPERATIONS!

document.querySelectorAll() also made the switch to the console. The main difference is that its alter ego \$\$() returns an array instead of a node list. As an example, look at the following command sequence, which obtains a list of all buttons currently in the DOM:

```
$$([type="button"])
Array [ button#knob1,
button#knob2 ]
```

These values can then also be stored in a variable for further usage.

Alternatively, pass them to a function which processes them – in short, you can use them as if they were an array created by normal JS code.

BECOME AN EXPERT

MOZILLA HAS IMPLEMENTED SOME NIFTY LITTLE FEATURES. HERE'S A FEW

Having a DOM element in an array or in a variable does you little good. In most cases, you want to see where it appears on-screen. This question can be answered by moving the mouse over it – the highlight will automatically be adjusted to target the specific element (see figure). In addition to that, take note of the little rectangle after the name – one click of it, and you find yourself looking at the element in the inspector.

ADVANCED MARK-UP GENERATION

The web consoles of old were limited to outputting the name and the memory address of the element in question. When working with Firefox,

many items get enhanced during the logging process.

In particular, take a look at the table lifted from MDN – it provides a list of all objects which get marked up as of this writing.

DEBUGGER INTEGRATION

Finally, keep in mind that the Inspector view used in the debugger is never more than a few keystrokes away. Simply pass an element to inspect() in order to open it in the viewer.



MUST-KNOW TIPS & TRICKS

MOZILLA'S WEB CONSOLE ALSO HAS A FEW TRICKS UP ITS SLEEVE...

1

CONTENT IS KING

Should you ever find the need to quickly copy the contents of a variable to the clipboard, feel free to enlist the services of the copy() command. It takes whatever information is passed in, and copies a textual representation of it into the command line of the workstation. When a DOM node is passed in, its outerHTML will be provided – for other elements, stringify() is called to obtain a textual representation.

2

THE MOZILLA WAY

<https://mzl.la/2zJdXcR>

While Mozilla also supports the table() command, using it tends to be overkill if but one or two objects are to be handled. In this case, why not take a stab at the pprint() function – its name comes from pretty print, and describes its role in life pretty well. When invoked on an object, prepare yourself for a sveltely-formatted list of properties and functions:

```
pprint($("#knob1"))
  " checkValidity: function ()
  reportValidity: function ()
  setCustomValidity: function ()
  autofocus:
    get: function ()
    set: function ()
    . . .
```

3

DEEP NAVIGATION WITH XPATH

<https://mzl.la/2zJdXcR>

Developers who dislike the jQueryesque syntax used in the dollar sign functions can find refuge by using \$x(). It takes an XPATH query, and returns an array of all nodes that match its definition. The main issue with the method is that Mozilla is not particularly interested in supporting XPATH further – the current state of the parser by and large is all what will ever be available. Furthermore, keep in mind that XPATH is not particularly widely used in the Web 2.0 industry.

4

KEYBOARD SHORTCUTS

<https://mzl.la/2lct5fA>

While not an API in the strictest sense, do take a look at the keyboard shortcut list. It provides a set of quick-access features that allow you to keep your hands away from the mouse (or the touchpoint). For example, did you know that the message output box can be searched by pressing Ctrl+F, and proceeding to entering a (partial) query? Alternatively, press Ctrl+Shift+K at any time to return focus to the command line to fire off a quick order or two.

CREATE THE IMPOSSIBLE

www.photoshopcreative.co.uk

FREE! OVER 290 RESOURCES
✓ BRUSHES ✓ TEXTURES ✓ BACKGROUNDS ✓ STOCK IMAGES ✓ ACTIONS AND MORE



Photoshop creative

Available from all good newsagents and supermarkets

ON SALE NOW

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

PHOTO EDITING



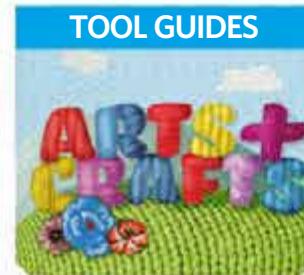
DIGITAL PAINTING



PHOTO ART



TOOL GUIDES



BEGINNER TIPS



BUY YOUR ISSUE TODAY

Print edition available at www.myfavouritemagazines.co.uk

Digital edition available for iOS and Android

Available on the following platforms



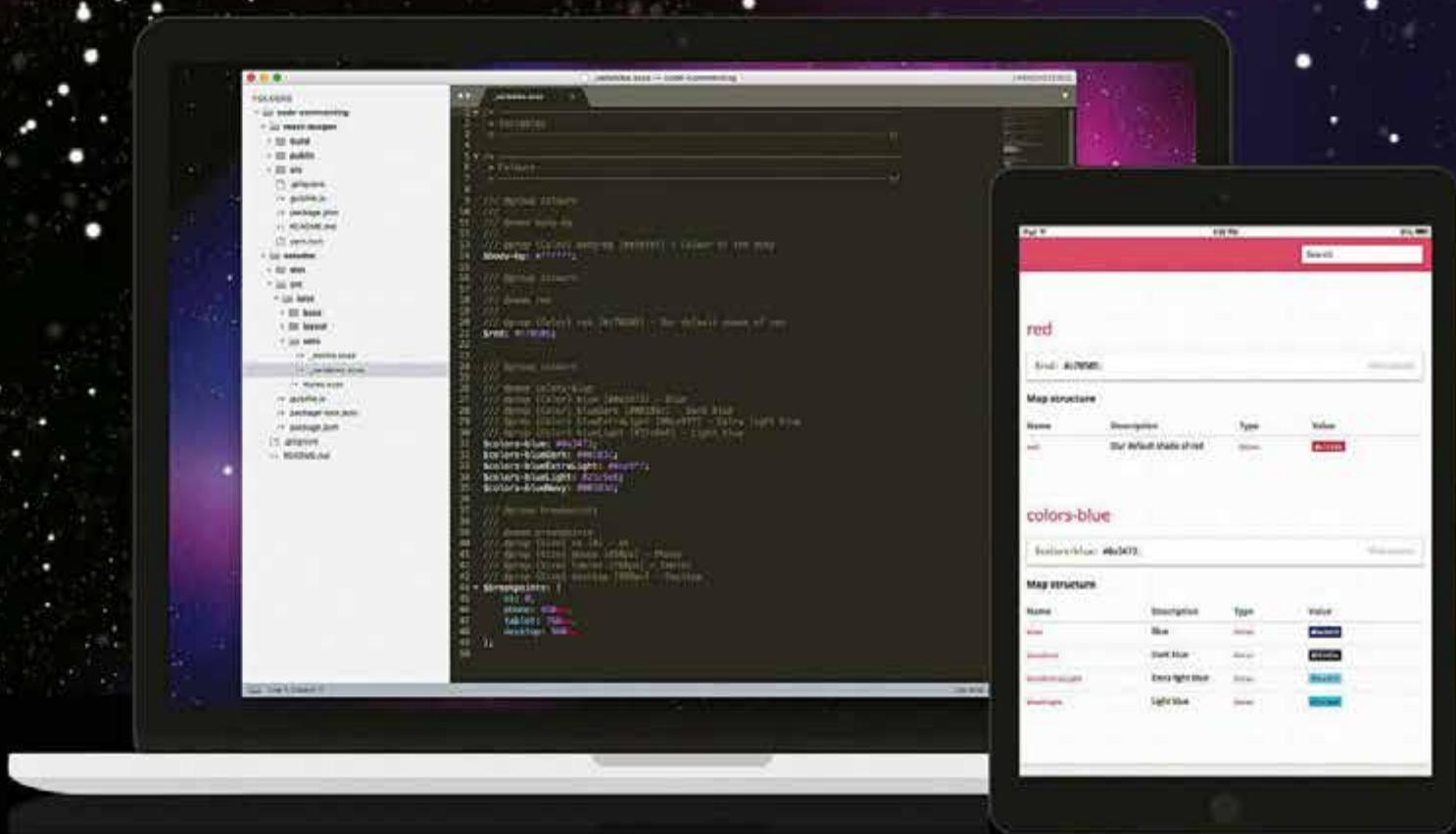
facebook.com/PhotoshopCreative



twitter.com/PshopCreative

Comment your code for healthy docs

Add code comments to styles and JavaScript components to generate and maintain comprehensive codebase documentation





hen done right it can be the key to a successful project handover, or a new developer getting up to speed. But documentation must be updated manually. It's only useful if every change to the codebase is followed by an update to the documentation. If that doesn't happen, you can end up with something worse than having no guide at all: a guide that tells developers the wrong thing.

Luckily, there is a way to create healthy and hassle-free docs through simply adding comments above your code. There are two main areas that benefit from this, so this tutorial has been split into aesthetics and functionality.

Aesthetics

Designers benefit from documentation within a codebase as much as developers. It's an all-too-common problem for a developer to receive a design that differs slightly from the rest of the site. Wouldn't it be great if designers and developers could see what styles are in place?

Enter sassdoc - a tool to create a style guide via inline commenting. Its setup is minimal, and by adding comments directly above the styles you're documenting, it generates an entire standalone project complete with CSS & JS.

Once this is in place, and by continuing this as your project grows, you can maintain (with minimal effort) colour and measurement detail for designers, and practical pieces of reusable code for developers. Everyone wins.

1. Install sassdoc

To get started, install sassdoc like you would any other npm module. This gives you access to the sassdoc command on the command line. At this point you're already ready to generate the documentation for your project, given the correct commenting and a directory to look in (you'll get to that shortly).

```
npm install sassdoc
```



2. Add your wrapper of choice (if applicable)

Here you set up how you'll call the sassdoc command to generate the output. As sassdoc works by using code commenting in SASS. Any project you intend to use this in will need a build process in place to convert it. For this example we'll use gulp - as of version 2.0, sassdoc supports direct use within gulp without the need of a wrapper. If you're a grunt user, there's a wrapper called grunt-sassdoc (<http://bit.ly/2zfxYHB>).

3. Create the basic task

The code below sets out a simple gulp task to check every .sass and .scss file inside the src/sass directory for the correct commenting syntax, and then generate the sassdoc files in dist/sassdoc.

```
gulp.task('sassdoc', function() {
  let options = {
    dest: 'dist/sassdoc',
    display: {
      watermark: false
    },
    exclude: ['src/sass/vendor'],
    verbose: true
  };
  return gulp.src('src/sass/**/*.{sa,cs}ss')
    .pipe($.sassdoc(options));
});
```

4. Customise implementation (optional)

There are lots of useful sassdoc options to explore (which are well documented on their website), so for now you'll add some basic configuration. Here you add a verbose option to allow the command line to give you a full output, exclude a particular folder from being checked, and alter the display properties to remove the sassdoc watermark.

```
let options = {
  dest: 'dist/sassdoc',
  display: {
    watermark: false
  },
  exclude: ['src/sass/vendor'],
  verbose: true
};
```

Exploring sassdoc

There are so many possible options for sassdoc that this tutorial cannot cover them all; I aim only to set you loose on it. Therefore, I would highly recommend heading to <http://sassdoc.com/configuration> and the surrounding docs to help craft the setup that's perfect for you.

FUNCTIONS

breakpoint-var

Description

Return the value in pixels of the breakpoint key provided

Parameters

Name	Description	Type	Default value
breakpointName	The variable above used to find the right size	text	-
sizeKey	The key to reference the variable object for	text	-

Example

```
placeholder: 100;
@rule: breakpoint-var($breakpoint-var);
width: $rule;
```

Requires

Top

Once you add a basePath option to your configuration, you can click 'view source' to go to its corresponding line

Middle

The output in sassdoc of commenting a simple variable with the correct syntax, showing attributes alongside an example

Left

Commenting a function in the correct way can show its parameters with type, description, requirements and an example of usage

Developer tutorials

Comment your code for healthy docs

5. Add a basePath for context

The final piece of config you'll add is very useful for group projects. When pointed at the root of the styles folder, the basePath attribute allows you to specify a link to the code (commonly GitHub) which then creates a 'view source' link next to code examples. This link takes the user to the exact line and file where the styles are created.

```
basePath: 'https://github.com/your-repo/tree/master/src/sass'
```

6. Add a basic code comment

The sassdoc syntax begins with three forward slashes, and requires only a name to get started. Let's start with commenting a simple variable with a name (@name), its group (@group), and its property (@prop)

```
/// @group colours
///
/// @name red
///
/// @prop {Color} red [#c70505] - Our
default shade of red
$red: #c70505;
```

7. Commenting a function

You can go into much greater detail for a function or mixin, due to their extra intricacies. Here we add a description under the name (which requires no @ tag), an @param to display a parameters type, reference, and purpose, and an @example to display the proper use of it.

Implementing react-docgen

To focus on the output alone, this tutorial is using the basic command line approach to react-docgen. However, there are many other integration options available that might fit better with your setup. If you would like to use a dedicated gulp wrapper, try react-docgen-ui or gulp-react-docs, and for webpack lovers, docgen-loader and react-docgen-loader are both capable options.



The screenshot shows the official React.js website. At the top, there's a navigation bar with links for 'Docs', 'Tutorial', 'Community', and 'Blog'. A search bar says 'Search docs' with a magnifying glass icon. To the right, there are buttons for 'VIDEO' and 'GITHUB'. Below the navigation, the word 'React' is prominently displayed in a large blue font, followed by the subtitle 'A JavaScript library for building user interfaces'. Underneath, there are two buttons: 'Get Started' and 'Take the Tutorial'. The main content area is divided into three columns: 'Declarative', 'Component-Based', and 'Learn Once, Write Anywhere'. Each column contains a brief description and a small image.

Declarative	Component-Based	Learn Once, Write Anywhere
React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.	Build encapsulated components that manage their own state, then compose them to make complex UIs.	We don't make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.
Declarative views make your code more predictable and easier to debug.	Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM.	React can also render on the server using Node and power mobile apps using React Native.

```
/// @name add
/// Add the two provided units together
///
/// @param {Size} $a - The first value to be added
/// @param {Size} $b - The second value to be added
///
/// @example scss - placeholder ` .foo {
///   .foo {
///     width: add(10px, 25px);
///   }
/// }
```

8. Allow sassdoc to watch for changes

The final step here is to set your gulpfile to update your docs at the same time your sass is changed. You might only want to generate the docs during a production or first development build to make your build process lighter, but this approach means you're always up to date.

```
gulp.watch('src/sass/**/*.{a,c}ss',
['sass', 'sassdoc']);
```

Functionality

Documentation of functionality has grown with the soaring popularity of component-focused frameworks, such as React JS. As developers strive to make these mini modules independent and reusable, it's essential to understand their individual purpose (and restrictions). In that vein, React JS is the perfect framework to showcase the power of this concept.

Meet react-docgen - a library that uses conditional commenting on components and their prop types. Unlike sassdoc, it doesn't generate the finished article. Instead, it returns a JSON object to display however you like. You need to be in a React project to use it, so we'll create a small React component to display the documentation of our React components.

1. Install react-docgen

Much like the sassdoc example, to get started you need to install react-docgen for our project. You will then have access to the react-docgen command in the command line. You can then run this with either a desired



Topleft

An example of the JSON object generated having run react-docgen on a project

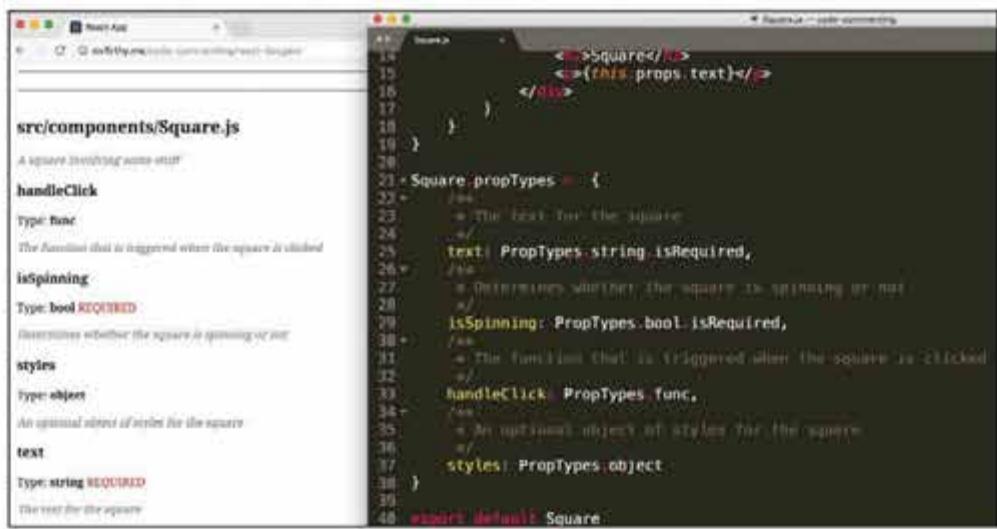


Topright

An example of the final output; displaying documentation for all components within the chose directory and each of their prop types

Right

Here we can see the comments placed above prop types in a component (right) and the eventual output we will create (left)



The screenshot shows a code editor with two panes. The left pane displays the code for a component named 'Square.js'. It includes comments explaining the purpose of the component, its state, and its props. The right pane shows the JSON output generated by react-docgen, which maps the component's structure and prop types to the JSON object shown in the first screenshot.

```
src/components/Square.js
A square that rotates when spun
handleClick
Type: func
The function that is triggered when the square is clicked
isSpinning
Type: bool REQUIRED
Determines whether the square is spinning or not
styles
Type: object
An optional object of styles for the square
text
Type: string REQUIRED
The text for the square
```

```
44 export default Square
```

```

const sassdoc = require('sassdoc');
const gulp = require('gulp');
const config = require('config');
const sass = require('sass');

module.exports = function() {
  options = {
    dest: 'basePaths.dist + "sassdoc"',
    verbose: true,
    display: 'list',
    access: ['public', 'private'],
    alias: false,
    watermark: false,
  },
  groups: {
    colours: 'Colours',
    breakpoints: 'Breakpoints'
  },
  basePath: 'https://github.com/MrFirthy/'
};

gulp.task('config', sass.glob);
gulp.task('options', () => {
  const options = {
    dest: 'basePaths.dist + "sassdoc"',
    verbose: true,
    display: 'list',
    access: ['public', 'private'],
    alias: false,
    watermark: false,
  },
  groups: {
    colours: 'Colours',
    breakpoints: 'Breakpoints'
  },
  basePath: 'https://github.com/MrFirthy/'
};

gulp.task('sassdoc', sassdoc);
}
  
```

Sassdoc-example - 1.0.0

Open all

Breakpoints

Colours

General

Organising your styles

It's really important that the output of your styles are organised in a way that's easy to understand. This will likely change from project to project, and so sassdoc offers a 'groups' option. This allows you to group styles and functions together under certain headers, rather than having sassdoc dictate the layout. Each group requires a slug and an optional 'human-friendly' group name for display purposes. For example, if you added a 'groups' object in your sassdoc options with colours: 'Colours' you could then add a selector to it by adding the `/// @group colours` comment.

destination directory/file name, or left without to display the returned object inline.

```

npm install react-docgen
// Write the output to a file
react-docgen target/src/dir --out output/
dir/components.json
// Display the contents of the object inline
react-docgen target/src/dir
  
```

2. Create basic React app (optional)

As this tutorial focuses solely on integrating documentation into applications, we won't spend time covering the approaches to creating a React application. If you already have a project that you'd like to add documentation to, feel free to continue without this step. If you don't, I recommend using Facebook's own `create-react-app`. It works out of the box and is ready to go in four lines:

```

npm install -g create-react-app
create-react-app my-app
cd my-app/
npm start
  
```

3. Commenting components

Commenting a component itself is very simple. The path and name of the component is generated automatically, and you can add a description by placing react-docgen's specific commenting just above a component definition, like so:

```

/**
 * A square designed to do square things!
 */
class Square extends React.Component {
  ...
}
  
```

4. Commenting prop types

To add documentation for a component's prop types, you simply use the same commenting style directly above each property declaration. Doing so will generate the prop type's name, type, description, and if it's required. For example, the below comment would return a text prop type, a type of string, a description, and that it is required:

```

Square.propTypes = {
  /**
   * The text for the square
   */
  text: PropTypes.string.isRequired,
}
  
```

5. Retrieving output

As I mentioned earlier, react-docgen outputs a JSON file instead of a fully-fledged site like sassdoc. As you're focusing on handling the output, use the simple command line interface in your gulpfile to generate it. You achieve this through `gulp-shell`, a small package that allows you to run shell commands within your gulp file:

```

npm install gulp-shell
gulpfile.js
var shell = require('gulp-shell');
gulp.task('generateDocs', shell.task('react-
docgen src/components --out src/docs/
components.json'));
  
```

6. Pass data into a component

Now that we have our JSON, you can import it into your app and pass it directly into a component for display. You will make this `<Docs>` component soon, but to give an overview we can start with something as simple as this: import data from `'./docs/components.json'`:

```

class App extends Component {
  render() {
    return (
      <Docs data={data} />
    );
  }
}
  
```

7. Split the data into components

With this data passed into your `Docs` component as a prop, you now need to iterate over it to render each piece of documentation individually within a separate `<Doc>` component (which you'll make for cleanliness).

```

class Docs extends React.Component {
  render() {
    const comps = [];
    for (const component in this.props.data) {
      const obj = this.props.data[component];
      comps.push(<Doc component={component}
key={component} obj={obj} />);
    }
    return (
      <div>
        {comps}
      </div>
    )
  }
}
  
```

8. Organise and sort the components

Next, you make the `<Doc>` component that will render each component reference individually. To start, you make a simple `keys` function in your `render()` method that

Developer tutorials

Comment your code for healthy docs

The screenshot shows the GitHub repository page for reactjs/react-docgen. The README.md file contains the following content:

```
react-docgen is a CLI and toolbox to help extracting information from React components, and generate documentation from it.

Usage: react-docgen [path]... [options]

path      A component file or directory. If no path is provided it reads from stdin.

Options:
  -o FILE, --out FILE  store extracted information in FILE
  --pretty            pretty print JSON
  -x, --extension    File extensions to consider. Repeat to define multiple extensions. Default: .js,.jsx
  -e, --exclude       Filename pattern to exclude. Default: []
  -i, --ignore        Folders to ignore. Default: {node_modules,__tests__,__mocks__}
  --resolver RESOLVER Resolver name (findAllComponentDefinitions, findExportedComponentDefinition) or
                        path to a module that exports a resolver. (findExportedComponentDefinition)

Extract meta information from React components.
If a directory is passed, it is recursively traversed.
```

react-docgen configuration

Again it is not the focus of this tutorial, but there are many configuration options to affect that output you receive from react-docgen. The only option we are using is the `--out` flag that allows us to write the JSON object to a file in a directory of our choosing, as by default it will output this directly to the command line.

There are also a few ways you can filter the scraped files. To limit what gets checked, you can use the `--ignore` flag and provide a list of directories that would then be overlooked, or `--exclude` to provide a list of filename patterns to exclude. Alternatively, if you would like it to look for more than just `.js` or `.jsx` files (the default), you can add `--extension` with a list of extensions to consider. There is also a `--pretty` flag to return the object with proper indentation.

will take the component object passed down by props in the step before, iterate over each key within it, and return them for easy reference. You finish by running `.sort()` on the returned keys, and they're ready to be used.

```
function keys(obj) {
  const keys = [];
  for (const key in obj) {
    if (obj.hasOwnProperty(key)) { keys.push(key); }
  }
  return keys;
}

const sortedProps = keys(this.props.obj).sort();
```

9. Render the props

Now the props are sorted, you need to retrieve the content and create an array of each property individually to be rendered within a component. In the same file:

```
const props = [];
for (const content in sortedProps) {
  const propName = sortedProps[content];
  propObj = this.props.obj.props[propName];
  const requiredText = propObj.required ?
```

```
<span>REQUIRED</span> : null;
propNames.push(
<div key={content}>
<h3>{propName}</h3>
<p>Type: <strong>{propObj.type.name}</strong> {requiredText}</p>
{propObj.description ? <p>{propObj.description}</p> : null}
</div>
);
```

10. Bringing it all together

With your props array created, you simply pass that object into the return method of your `<Doc>` component, and each prop type found will be displayed with either its comments, or with a message stating that there are no prop types for this component.

```
return (
<div>
<h2>{this.props.component}</h2>
<p>{this.props.obj.description}</p>
{props.length ? null : <p>This component has no properties</p>}
{props}
</div>
)
```

11. Watching for future changes

Just like in the aesthetics section, your last step is to set up a watch task in gulp to have react-docgen re-generate the JSON when a component is changed.

```
gulp.watch('src/components/**/*.{js,jsx}', ['generateDocs']);
```

Conclusion

If you have any issues, there is a full copy of this example available at <https://github.com/MrFirthy/code-commenting>, and a working example of the output at <http://mrfirthy.me/code-commenting>. This approach doesn't completely negate the potential for out of date docs, but makes it far easier for developers and designers to mirror codebase changes in colour, size, or function. By having a simple code comments directly above any type of declaration like this, it's borderline unmissable. This gives a project the best possible chance of having its documentation up-to-date, and hopefully avoid some of the stress usually involved in doing so.

The screenshot shows the GitHub repository page for code-commenting. The README.md file contains the following content:

```
Simple example of creating self-explaining docs for your component and SASS styles through code-commenting

Setup
=====
1. cd comment
2. npm install
3. npm start

This will then generate the SASS along with an instance of component.js and serve them up at http://localhost:3000. You can view the code at https://github.com/MrFirthy/code-commenting and the compiled documentation here in action at https://mrfirthy.me/code-commenting

Building the docs
=====
The browser sync instance will compile a watch file. As you make changes to the component, the changes will be reflected live on the page, and the comments will be regenerated. You can also use the browser sync feature to see how your changes affect the page.

Readme generator
=====
1. cd read-gen
2. npm install
3. npm start

The code in this app is made using code-commenting, and is used to simply display the use of code-commenting. When you run this, it will build up the code synchronously with live reload, and to quickly fire it from a build or generator. This means that if you change any file, it will automatically rebuild the app and then update the page with the updated code.
```

Example

This tutorial is using the basic command line approach to react-docgen, which you can find here:
Live: mrfirthy.me/code-commenting/sassdoc
Live: bit.ly/2yeChEg
Code: bit.ly/2fZnjZk

OFFER ENDS
31 DECEMBER 2017

SAVE UP TO 49% ON THE PERFECT GIFT THIS CHRISTMAS



FROM £21 EVERY 6 MONTHS
(\$69.90 / \$77.40 PER YEAR)



FROM £21 EVERY 6 MONTHS
(\$85 / \$85 PER YEAR)



FROM £23 EVERY 6 MONTHS
(\$103 / \$105 PER YEAR)



FROM £24 EVERY 6 MONTHS
(\$81 / \$113 PER YEAR)



FROM £16.80 EVERY 6 MONTHS
(\$78 / \$108 PER YEAR)



FROM £25 EVERY 6 MONTHS
(\$89 / \$89 PER YEAR)



GUIDES & SPECIALS, BACK ISSUES AND GIFT VOUCHERS WILL MAKE GREAT GIFTS TOO!

Delivery included in the price

Free personalised e-card when buying for someone else

Buy as a gift or treat yourself!

Choose from a huge range of titles

SEE THE FULL RANGE AND ORDER ONLINE

www.myfavouritemagazines.co.uk/xmas17

ORDER HOTLINE: 0344 848 2852

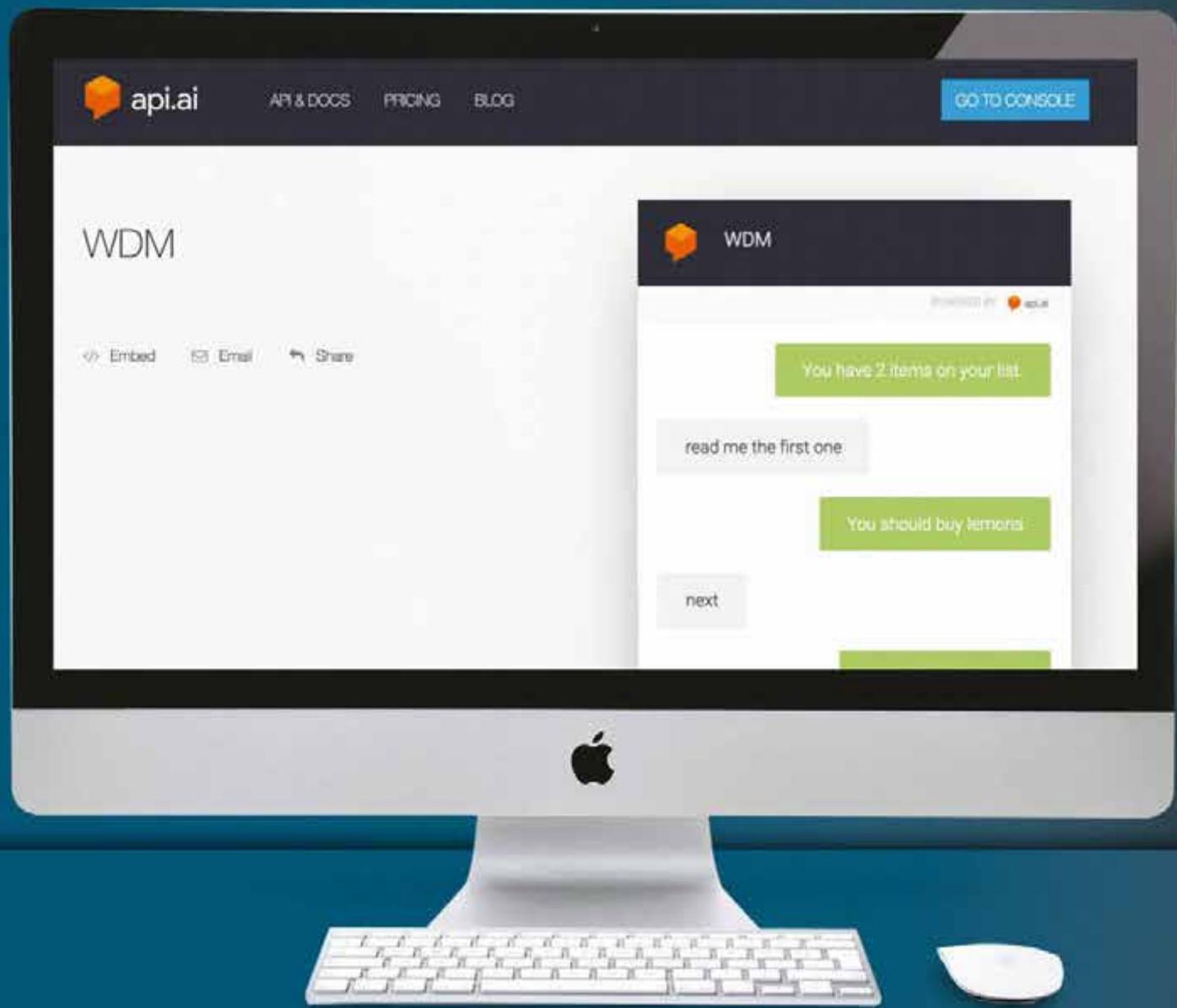
PLEASE QUOTE XMAS17 WHEN ORDERING BY PHONE

LINES ARE OPEN MONDAY - FRIDAY 8AM TO 7PM AND SATURDAY 10AM TO 2PM (GMT)

Terms and conditions: Savings calculated against the full RRP (single issue price x frequency). Dollar prices quoted are for the United States, other global territory dollar pricing may vary. This offer is for new subscribers only. You can write to us or call us to cancel your subscription within 14 days of purchase. Your subscription is for the minimum specified and will expire at the end of the current term. Payment is non-refundable after the 14 day cancellation period unless exceptional circumstances apply. Your statutory rights are not affected. Prices correct at point of print and subject to change. Full details of the Direct Debit guarantee are available on request. For full term and conditions please visit: bit.ly/magtdc. Offer ends 31st December 2017.

How to build a voice-activated app

Take your first steps with voice-activated devices and build a to-do app using API.ai and Node.js



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



UIs (voice user interfaces) are the new domain to explore for experience designers and developers. The big players (Google, Amazon, Apple etc) are multiplying devices, moving the assistant that used to come with your smartphone to a separate unit. While speaking to your phone in public might be embarrassing, they are betting on the fact that you will be in the privacy of your home and therefore more comfortable interacting with a machine that can help your efficiency. Unlike phones, these devices are not mobile and meant to stay in one room, so you could have a number of them all over the house.

A to-do app is the default application a developer builds when learning a new programming language. Although this tutorial uses JavaScript, programming for voice feels like a new language. Of course, you could use your phone to set reminders and send text messages to yourself, but where's the fun in that? So, let's embark on our exploration of VUIs with a simple to-do list that can be added to, amended, completed and read out to you, or displayed on a webpage.

1. Initial setup

Before we start, download the assets from FileSilo and run `npm install`. We won't be using these assets right away, but you'll be ready for step 7. Later, we will use Node.js and an Express server to create routes for our app; we will also use the 'actions-on-google' library to interact with Google Home.

Note: you will need a hosting platform such as Heroku or AWS EC2 to deploy your code, as testing through localhost triggers a timeout.

2. Defining user interactions

Before getting stuck in, it is important to define the available interactions and commands for our app, so we won't have to retrofit them. You are welcome to use a visual diagram builder like Twine or Lucidchart; or simply use a text file, like this:

Q: I need to \$action
A: I've added \$action to your list

Q: What's on my list?
A: You have \$number items to do
A: You have no items on your list

Q: Read me the \$ordinal item
A: You should \$action1

Q: I've done that/Delete that/Next
A: OK, I've completed \$action1/I've deleted \$action1/\$action2

3. API.ai

Next, sign up to API.ai. It is Google's interface to create applications for its Home device. You could create a basic application with intents and entities, without having to write any code. We'll start with that and go further in later steps. Once signed up, create a new 'Agent'.



4. Your first intent

If you go to the 'Intents' section of your Agent, you'll see two default intents already there; we'll come back to those. Go ahead and create a new one. This one will be triggered by the sentence "I need to", and the answer can be the one we've defined previously. Here is an abstract of the JSON that can be exported from API.ai:

```
"data": [
  {
    "text": "I need to ",
```

```
  "userDefined": false
},
{
  "text": "actions",
  "alias": "actions",
  "meta": "@actions",
  "userDefined": true
}]
```

5. Entities complement the intent

As you can see from the above example, the intents need an '@actions' entity to be completed. To get the example working, you can go to Settings (the cogwheel next to your Agent name) > Export and Import, and import **first-integration.zip** from your FileSilo assets. You will notice the Entities being populated and you can add more to the list.

```
//Example entity values
[
  {
    "value": "call",
    "synonyms": [ "call" ]
  },
  {
    "value": "buy",
    "synonyms": [ "buy" ]
  }
]
```

6. Testing integration

Once you've imported the example and amended as you

Help and instructions

Voice interfaces are too new to be anchored in people's habits. It is recommended to implement a 'help' menu in your application, where available commands are listed. Beware, though: the device cannot be interrupted while it speaks, so it can become frustrating if it goes on for too long.

The screenshot shows the 'Intents' section of the API.ai web interface. A new intent named 'I need to' has been created. The 'Events' field contains the sentence 'I need to \$action'. The 'Action' field is set to 'with just'. Below this, there is a table for defining entities. The table has columns for 'Entity', 'Value', 'Type', and 'Parameter'. There are two rows: one for 'actions' with value 'actions' and type 'actions'; and another for 'action name' with value 'action name' and type 'actions'. At the bottom, there is a 'Text response' section with a 'DEFAULT' tab selected.

The screenshot shows the 'Intents' section of the API.ai web interface. An intent named 'Actions' has been created. The 'Events' field contains the sentence 'Actions'. The 'Action' field is set to 'with just'. Below this, there is a table for defining entities. The table has columns for 'Entity', 'Value', 'Type', and 'Parameter'. There are two rows: one for 'actions' with value 'actions' and type 'actions'; and another for 'action name' with value 'action name' and type 'actions'. At the bottom, there is a 'Text response' section with a 'DEFAULT' tab selected.

Left

To complete intents, in this case actions, an entity needs to be defined

Above

This shows the web demo interface responding to user input, ie voice commands

Developer tutorials

How to build a voice-activated app

wanted, you will be able to go to Integrations > Web Demo and try out various responses, when you visit the URL that API.ai gives you. It will be in the format <https://bot.api.ai/<token>>. You will notice that it only works for entities that have been defined.

7. Interact with API.ai

Now we can have a look at the code base. You will have noticed in the 'I need to' intent, an action name of 'wdm.add'. This is the action we are going to use to intercept requests coming through, and then display a custom response. When detecting an input, Google will post a request to our /vui endpoint, as detailed here, in `routes/vui.js`.

```
const Actions = {  
  ADD: 'wdm.add',  
};  
const addItem = google => {/*response  
happens here*/ };  
const actionMap = new Map();  
actionMap.set(Actions.ADD, addItem);  
app.post('/vui', (request, response) => {  
  const app = new ApiAiApp({ request,  
    response});  
  app.handleRequest(actionMap);  
});
```

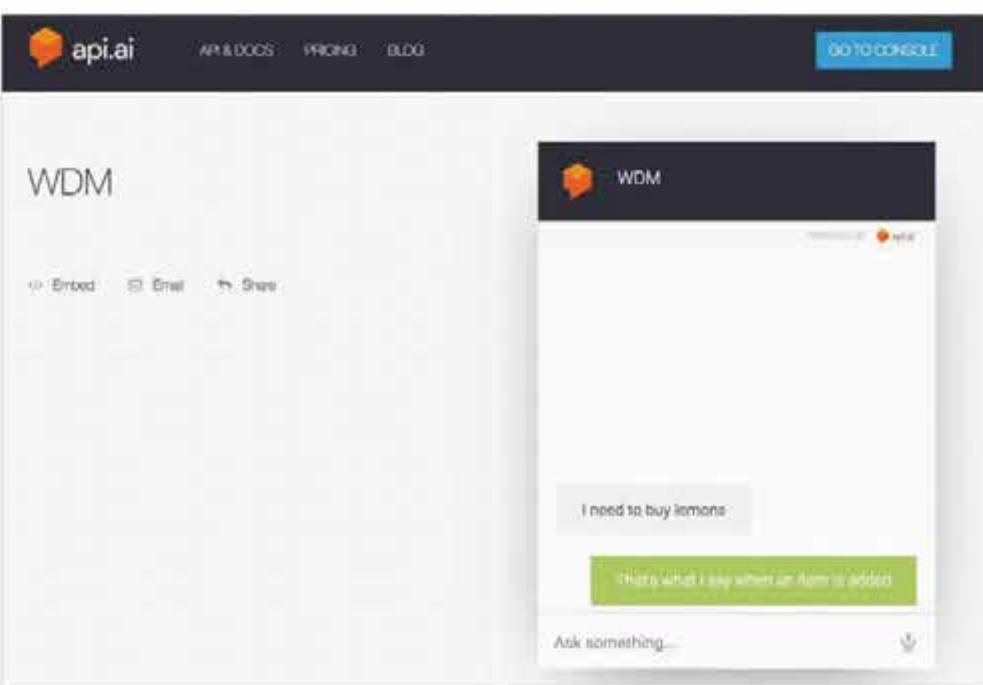
8. Enabling the webhook

In order for the interaction in the previous step to work, the code will have to be deployed to a live URL. Then, in API.ai's 'Fulfillment' section, enable the webhook, add your URL, and don't forget the /vui endpoint.

Next, open the 'I need to' intent, scroll to the bottom, and in Fulfillment, check 'Use webhook'.

What is SSML?

SSML provides markers, similar to XML, that indicate phonetic pronunciation of unknown words or give an indication of how big a pause there should be between sentences. To use SSML, enclose your text in `<speak>` tags.



9. Sending a response

At the moment, we are not sending a response to Google when the intent is active; let's change that in the 'addItem' function. Now, if you launch a new version of the Web Demo, you should see 'That's what I say when an item is added' when asking to buy lemons. Of course, that response is a bit static, so let's develop our logic to add items.

```
const addItem = google => {  
  google.ask('That's what I say when an  
  item is added');  
};
```

10. Storing data

For this example, we will use a JSON file to store our items. For a more scalable and robust backend, you could integrate services like Contentful or Trello; or even a database like DynamoDB. In `/project/data` you will find

the JSON file with the following structure:

```
{  
  "todo": [  
    {  
      "id": 0, "action": "buy lemons"  
    },  
    ],  
  "done": [] }
```

11. Adding an item

Now we can amend 'addItem' to read the user input in the form of a task. We can then read the data file ('DB_PATH') and append a new task to the 'todo' array before overwriting the file. Now if you deploy, and go to the Web Demo, the bot should state the item you wanted to add. Caution: as we still add features to the app, every time you do a new deployment, you will reset your JSON file to the original.



Above

Adding and listing items

Right

Intent setup for ordinal numbers

The screenshot shows the API.ai interface for creating intents. On the left, a sidebar lists various sections: WDM, Entities, Training, Integrations, Analytics, Fulfillment, Prebuilt Agents, and Small Talk. The main area is titled 'Read' and contains a configuration for an intent named 'wdm'. Under 'User says:', there are several examples of user inputs: 'Read me the [ordinal] book', 'Read me the [ordinal] one', 'Read', and 'Read me the [ordinal] item'. Below this, there are sections for 'Events', 'Action', and a preview of the bot's response. The preview shows a message: 'User says: Read me the 1 book' and 'Bot says: That's what I say when an item is added'.

TABLE OF CONTENTS

1.	Introduction
2.	Design Patterns and Widgets
2.1	Generally Applicable Keyboard Recommendations
2.2	Accordion (Sections With Show/Hide Functionality)
2.3	Alert
2.4	Alert and Message Dialogs
2.5	Breadcrumb
2.6	Button
2.7	Checkbox
2.8	Combo Box
2.9	Dialog (Modal)
2.10	Dialog (Non-Modal)
2.11	Disclosure (Show/Hide)
2.12	Feed
2.13	Grids : Interactive Tabular Data and Layout Containers
2.14	Link
2.15	Listbox
2.16	Menu or Menu bar
2.17	Menu Button
2.18	Radio Group
2.19	Slider
2.20	Slider (Multi-Thumb)
2.21	Spinbutton

2.9 Dialog (Modal)

A dialog is a window overlayed on either the primary window or another dialog window. Windows in a dialog are inert. That is, users cannot interact with content outside an active dialog window. Inert content in an active dialog is typically visually obscured or dimmed so it is difficult to discern, and in some implementations attempts to interact with the inert content cause the dialog to close.

Like non-modal dialogs, modal dialogs contain their tab sequence. That is, Tab and Shift + Tab focus outside the dialog. However, unlike most non-modal dialogs, modal dialogs do not provide means for gaining keyboard focus outside the dialog window without closing the dialog.

The alertdialog role is a special-case dialog role designed specifically for dialogs that divert users' brief, important message. Its usage is described in the alert dialog design pattern.

Example

Modal Dialog Example

Keyboard Interaction

In the following description, the term "tabbable element" refers to any element with a `tabIndex` value greater. Note that values greater than 0 are strongly discouraged.

- When a dialog opens, focus moves to an element inside the dialog. See notes below regarding placement.
- Tab:
 - Moves focus to the next tabbable element inside the dialog.
 - If focus is on the last tabbable element inside the dialog, moves focus to the first tabbable element inside the dialog.
- Shift + Tab:
 - Moves focus to the previous tabbable element inside the dialog.
 - If focus is on the first tabbable element inside the dialog, moves focus to the last tabbable element inside the dialog.

A word about authentication

In this tutorial, we cut a few corners and assume that you will be the only user. However, if you wish to publish this application, you will need a stronger backend, and a way to identify and authenticate various users, to return the correct data. For example, you won't be able to use the global variables ('currentIndex', 'tempList', etc) as they are now - they will have to be tied to a user token. Maybe a random UUID or JWT (JSON Web Token). Another protection you can (and should) set on your application is Basic Auth, to avoid repeated calls to your service. API.ai lets you add a username and password for authentication headers when you set up the webhook URL. On the application side, you can then check that those headers match the ones you've set or reject the request, if not.

```
const task = google.getRawInput().replace('I need to ', '');
fs.readFile(DB_PATH, 'utf-8', (err, file) => {
  if(err) throw err;
  const list = JSON.parse(file);
  list.todo.push({id: list.todo.length, "action": task});
  fs.writeFile(DB_PATH, JSON.stringify(list), (err) => {
    if (err) throw err;
    google.ask('I\'ve added ${task} to your list.');
  });
});
```

12. What's on my list?

Next, we want to check how many items have been added to the list. To do this, we need to create a new API.ai intent, with action 'wdm.check', and enable the fulfillment. On our app side, we will have to add a new action and map it to another function, like so:

```
'actionMap.set(Actions.CHECK, checkItems);'
let tempData;
let tempList = [];
const checkItems = google => {
  fs.readFile(DB_PATH, 'utf-8', (err, file) => {
    if(err) throw err;
    tempData = JSON.parse(file);
    tempList = tempData.todo;
    google.ask('You have ${tempList.length} item${(tempList.length === 1)?':':''} on your list.');
  });
};
```

13. Selecting an item to read out

After that we create a new intent, 'Read', with the user trigger 'Read me the @sys.ordinal item'. This will get filled in when the user says "first", "second", etc. and will send the corresponding number. We then save the index globally, so the command can be "Next", without an ordinal input, in which case we increment 'currentIndex'.

```
let currentIndex = null;
const readItem = google => {
  const chooseIndex = (google.getArgument('ordinal') !== null)?parseInt(google.getArgument('ordinal')):"";
  if(chooseIndex !== "") {
    currentIndex = chooseIndex - 1;
  } else {
    ++currentIndex;
  }
}
```

14. Reading the item

Then, we complete 'readItem' by reading out the task at the desired index. Note that currently this only works if the user has already asked "What's on my list?", otherwise tempList is empty and the JSON file needs to be read again.

```
if(tempList.length > 0) {
  if(tempList[currentIndex]) {
    google.ask('You should ${tempList[currentIndex].action}');
  } else {
    google.ask('You don\'t have any more items on your list.');
    currentIndex = null;
  }
} else {
```

```
//TODO:: get list again.
google.ask('I can\'t find items on your list.')
}
```

15. Completing an item

Now that we can add items to the list, we should be able to mark them as completed. In order to do that we should set up a new intent with action 'wdm.complete' and map the action to the 'completeItem' function below. This will remove the latest item that has been read out from the 'todo' array, store it in the 'done' array in the JSON file, and send confirmation.

```
const completeItem = google => {
  if(currentIndex !== null) {
    const task = tempList[currentIndex];
    action;
    tempData.done;
    push(tempList[currentIndex]);
    tempList.splice(currentIndex, 1);
    tempData.todo = tempList;
    fs.writeFile(DB_PATH, JSON.stringify(tempData), (err) => {
      if (err) throw err;
      google.ask('I\'ve marked "${task}" as completed.');
    });
  }
}
```

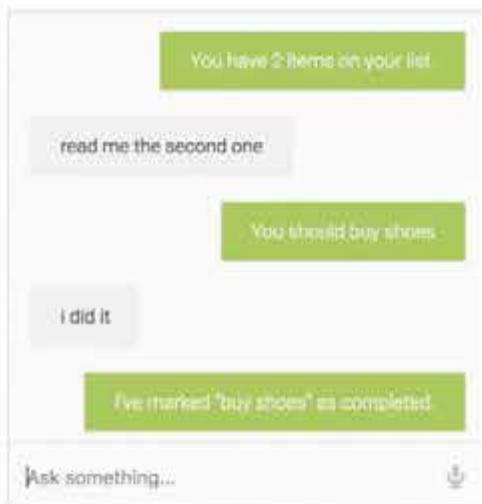
16. Deleting an item

Similarly, to delete the latest mentioned task, we can create a 'wdm.delete' intent and map it to 'deleteItem'. In this case, we don't save the task in the 'done' array - we just remove it from 'todo' and send confirmation.

```
const deleteItem = google => {
```

Developer tutorials

How to build a voice-activated app

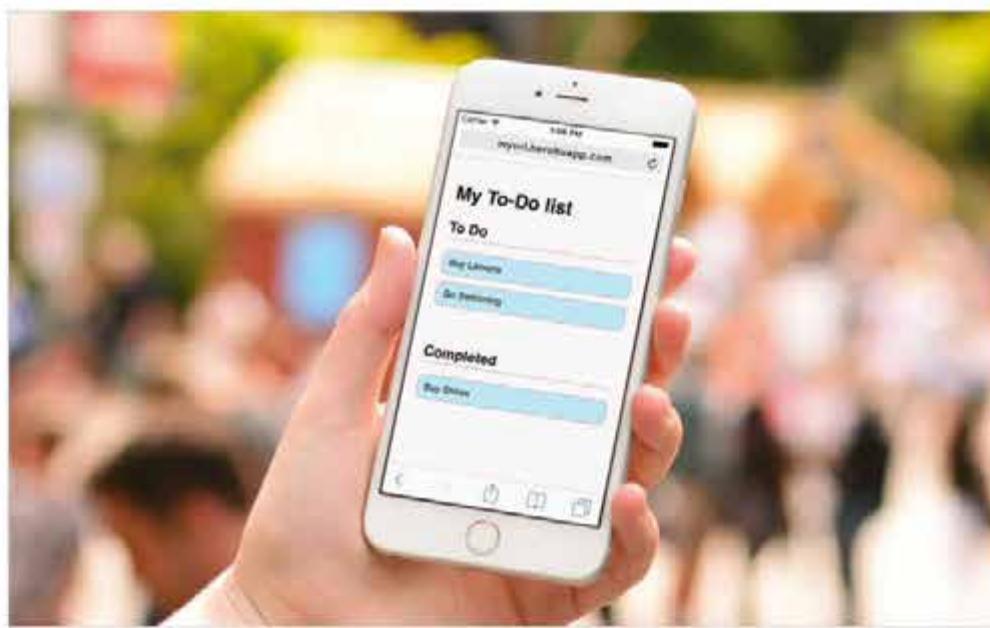


Above

When a task has been completed, let the app know

Right

A live example of the completed app in action



```
if(currentIndex !== null) {  
  const task = tempList[currentIndex].  
  action;  
  tempList.splice(currentIndex, 1);  
  tempData.todo = tempList;  
  fs.writeFile(DB_PATH, JSON.  
  stringify(tempData), (err) => {  
    if (err) throw err;  
    google.ask('I\'ve deleted "${task}".');  
  });  
}  
}
```

17. Visual route

You can also have a visual feedback for your to-do list. There is already a file for the 'index.js' route; we need to read the items from the JSON file and send them to the client. The app uses handlebars as a templating engine.

```
router.get('/', (req, res) => {  
  getList(data => res.render('index',  
  data));  
});  
  
function getList(callback) {  
  fs.readFile(DB_PATH, 'utf-8', (err, file)  
=> {  
  if(err) throw err;  
  const data = JSON.parse(file);  
  callback(data);  
});  
}
```

18. Display the list on a screen

Then, in the template, we can loop over the 'todo' array and display each item; the same can be done for the 'done' array. Now when you update the completed items through the web demo, you should see an updated list after a page refresh. You can then implement the update and delete actions on the front-end too.

```
<section id="todo">
```

```
<h2>To Do</h2>  
<{{#each todo}}>{{action}}</div>  
<{{/each}}>  
</section>
```

19. Next steps: synchronisation

To go further, you might want to implement WebSockets to keep the voice and visual parts up-to-date whenever a change is made. You would need to trigger a refresh on the client when the list is changed through voice, and update the 'tempData' in the voice route when a change is made through the client, to avoid discrepancies.

20. Default Welcome & Fallback Intents

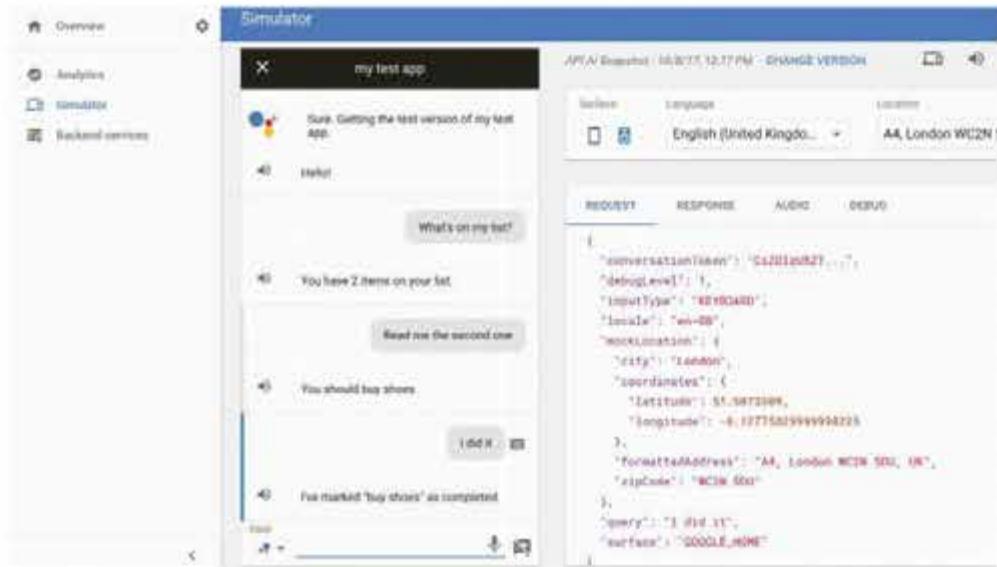
If you recall, when we created the agent in API.ai, there were two default intents present. The Default Welcome Intent is the text that will be read out when the app is

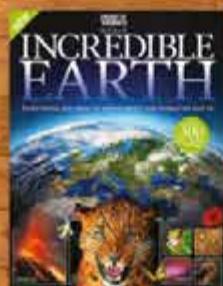
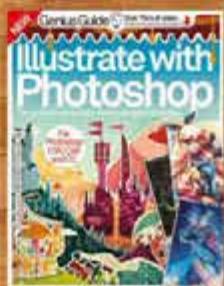
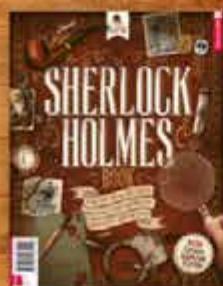
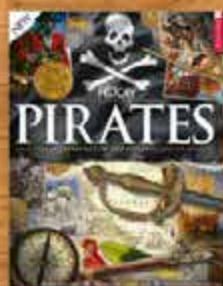
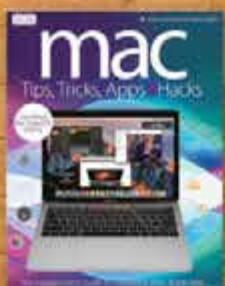
launched ('invoked') on a Google Home device. It should contain brief instructions, but can also use the webhook to read the list straight away when connected.

The Default Fallback intent casts the "I did not understand" responses when the user inputs don't match any of the intents. It can offer solutions or direct the users to the "Help" menu.

21. Publishing your action

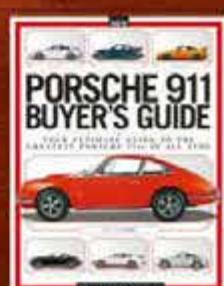
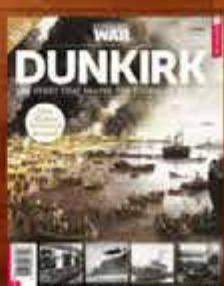
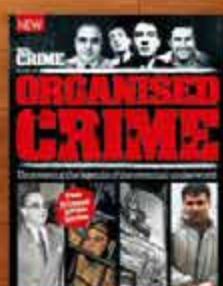
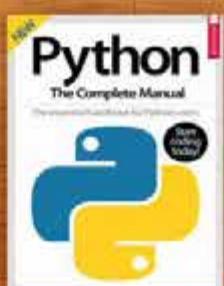
To publish your action for Google Home or even try it on the device, you will first need to download the app and pair the Home with your phone. Then, in 'Integrations', you can select Google Assistant and use the web simulator to test the voice responses. When you're ready, instead of 'Test', select 'Update draft'. Fill in the details for your application: Name, Invocation, Privacy Policy, Logo, etc. When done submit it to Google for review (you can also test it on your linked device before submitting once you've filled those parameters in).





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

HOSTING LISTINGS



Featured host: tidyhosts

www.tidyhosts.com / 05603 674610

About us

Tidyhosts has become a leading cloud hosting provider throughout the world. It boasts a feature-rich selection of services including domain registration, shared web hosting, WordPress hosting, hosted exchange email, cloud virtual servers and media streaming. Its passion and drive for

success has gained it a highly reliable and trusted reputation from its customers, making it the number one for hosting services. Founded in 2004 by three developers who wanted to build hosting infrastructure for developers - years on, the company has expanded to offer much more.

What we offer

- **Domain names** - Simple domain registration with a large choice of TLDs
- **Shared web hosting**
 - Includes one-click application installers and a choice of Windows or Linux
- **Cloud virtual servers** - Take full control over your hosting, and install the software you need on your server
- **SHOUTcast hosting** - Start your own radio station with our easy to set up and use SHOUTcast hosting service

5 Tips from the pros

1. Ensure that you choose the right domain

When choosing your domain, make sure it's easy to remember and resembles what you are offering. Customers are more likely to come back if they can remember your URL.

2. Create clear, concise website content

Keep the content on your website clear, informative and, more importantly, relative! Avoid any duplication of content on different pages as this can affect your rankings on search engines.

3. Utilise SSL certificates to stay secure

More sites are moving to SSL security

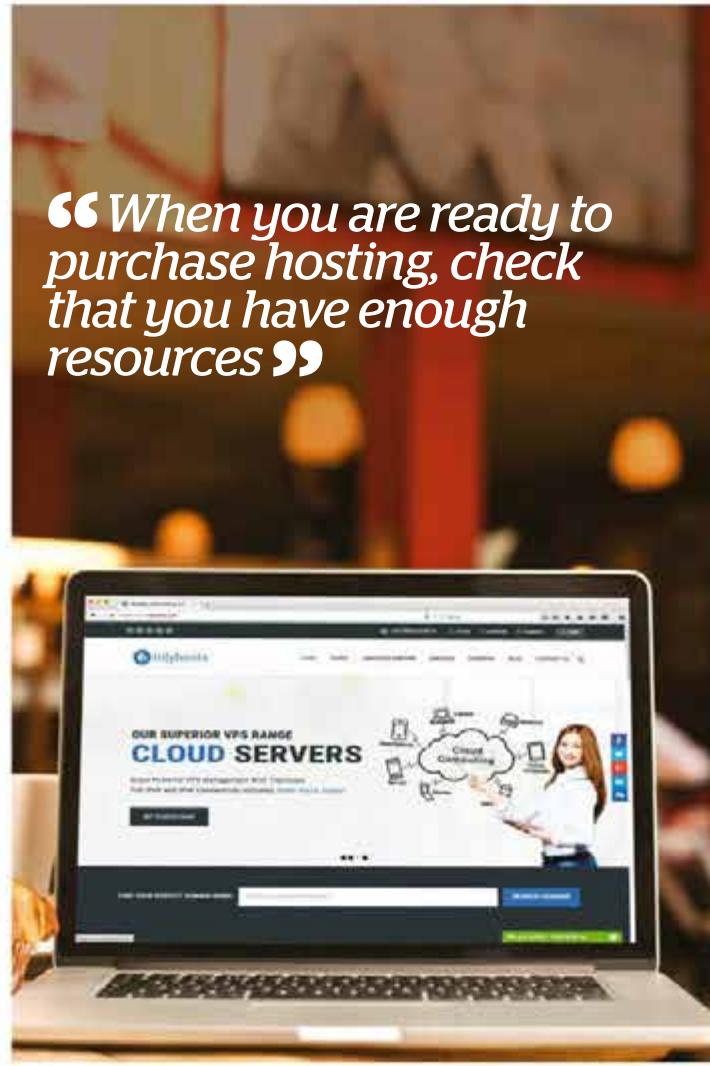
to protect their customers from the ever-increasing threats on the web. Search engines are now ranking sites with an SSL higher than those without.

4. Make sure you choose the right plan

When you are ready to purchase hosting, check that you have enough resources, especially if you expect your website to grow quickly. Seek advice from tidyhosts if you are unsure.

5. Use one-click installers when building a website

If you are new to building a website then we have a number of useful one-click application installers. These help you get up and running, including the popular WordPress system.



Testimonials

Kelly Underwood

“ I've been with a number of web host providers in the past which have at some point let me down. I now realise the importance of using a host that is well known ”

John Corey

“ We have our email hosted with tidyhosts, which serves a small number of users in our office. We have found this solution is much more cost-effective ”

Jenny Brice

“ A great host with fantastic knowledge. I have only had to use the support channels a few times, but the replies I have had helped me instantly, so well worth it ”



Get your listing in our directory

To advertise here contact *Chris*

chris.mitchell@futurenet.com

+44 (0)1225 687832

Supreme hosting



www.cwcs.co.uk
08001777000

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.

- Co-location hosting
- VPS
- 100% Network uptime

UK-based hosting



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



SSD web hosting



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

Budget hosting



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



Cluster web hosting



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

All-inclusive hosting



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

Flexible cloud servers



[elastichosts.co.uk](http://www.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



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.

sure you go through all the resources and give yourself two weeks to prepare.

4. Budget

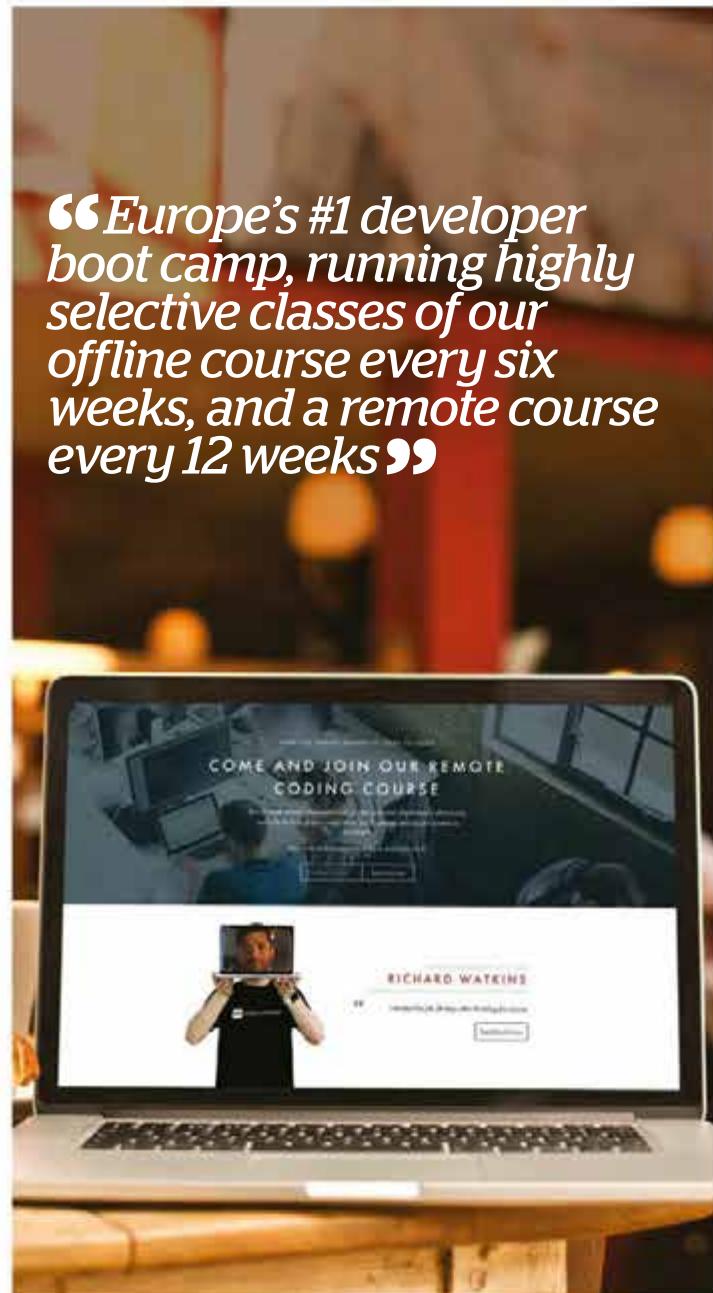
The course is full-time for three months and it can take up to three months after to secure a job. It's important to financially plan for the period you won't be working.

5. Visit us!

Book a visit and come visit us! Come see the Makers Academy HQ in person and learn more.

3. Prepare for the interview

We send you everything to prepare for the interview. Make



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

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

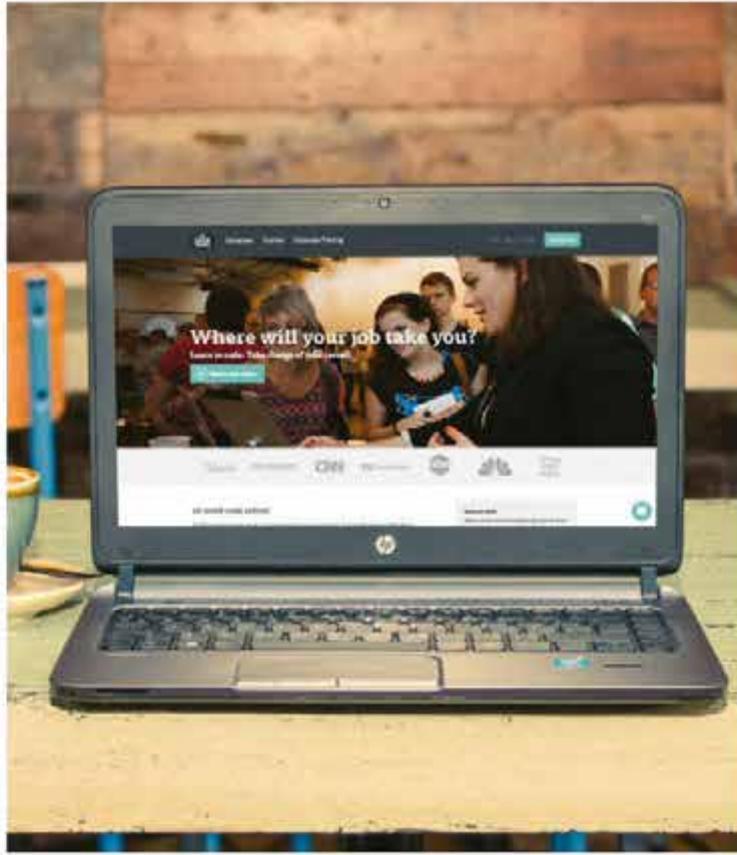
Twitter: [@TheIronYard](#)

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



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



Future
Learn

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

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

Free with your magazine

Instant access to these creative resources...

Essential assets and resources

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
- 225 minutes of Advanced NodeJS videos from Pluralsight (www.pluralsight.com)
- 50 Seamless asphalt textures from Sparklestock (www.sparklestock.com)
- Bespoke False Nine font

→ 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





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



Unlock
every
issue

Subscribe today & unlock the free gifts from more than 50 issues

Access our entire library of resources with a money-saving subscription to the magazine – that's more than 900 free resources

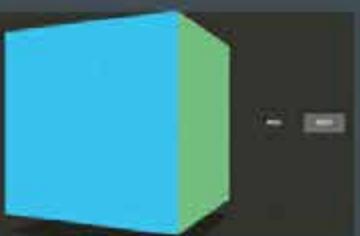
Over 60 hours of video guides

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 32 to subscribe now



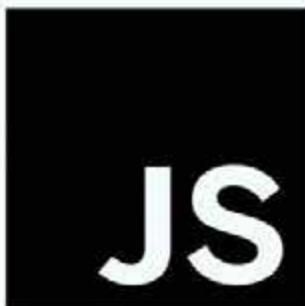
Already a print subscriber?
Here's how to unlock FileSilo today...

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

NEXT MONTH



THE NEW KINGS OF JAVASCRIPT

TOP TECHNIQUES TO USE THE LATEST
GENERATION OF LIBRARIES & FRAMEWORKS

ANIMATE SVG WITH GREENSOCK

Discover the art of creating crisp, clean, dynamic designs with JavaScript

BUILD PAGE LAYOUTS WITH CSS GRID

Create magazine style pages with the assistance of CSS's layout wizard

FACEBOOK DESIGN PROTOTYPING

Find out how to use Origami Studio to design modern interfaces

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 269 on sale
Thursday 14th December 2017

LOSE YOURSELF IN A WORLD OF

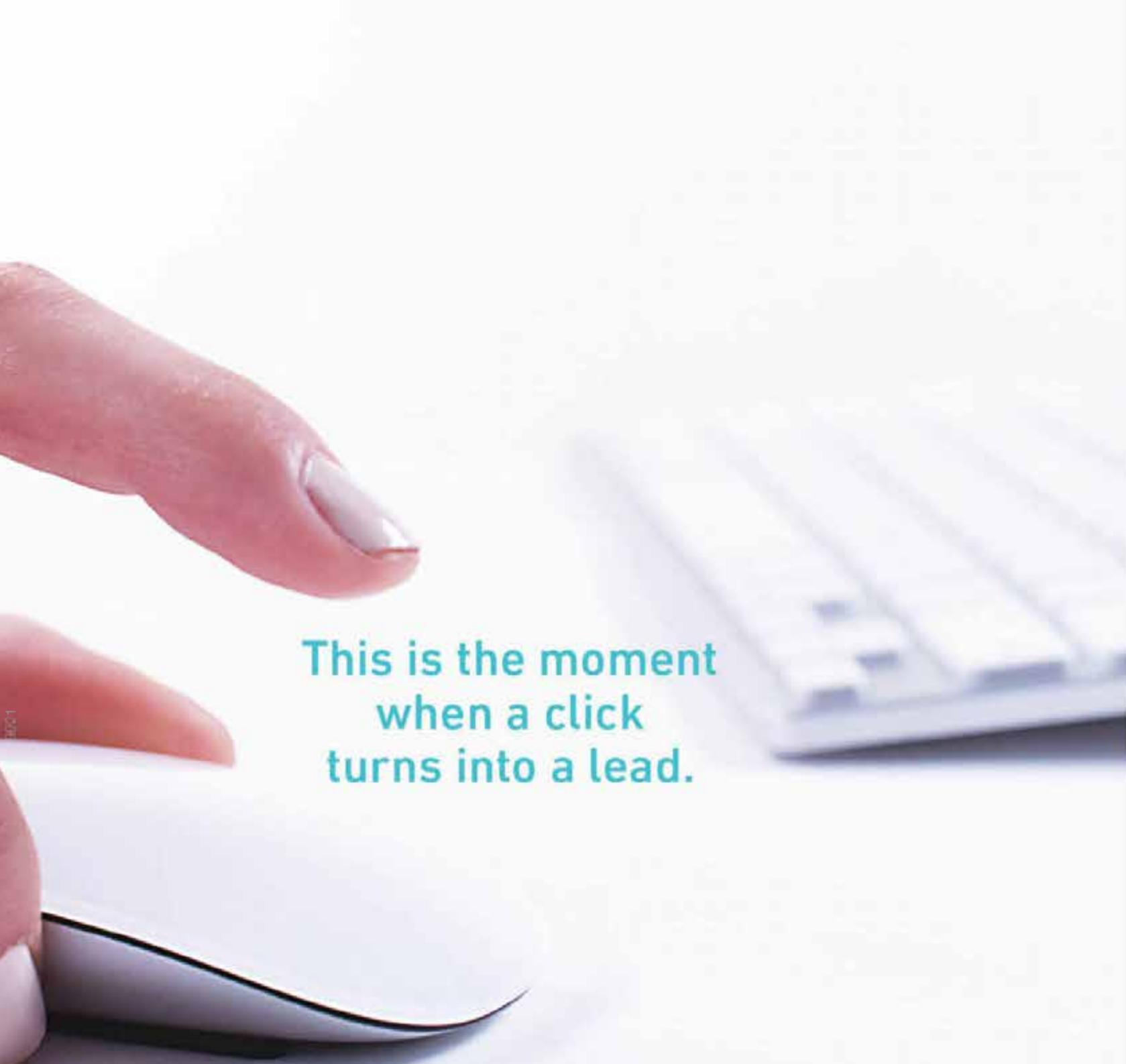
Vinyl

FIND YOURSELF IN
OXFAM'S ONLINE SHOP

oxfam.org.uk/shop



OXFAM



This is the moment
when a click
turns into a lead.

PRESS AHEAD

WP Engine's digital experience platform drives your business forward faster. wpengine.co.uk

WP engine*