31 PAGES OF EXPERT TUTORIALS

web designer

FREE 64 MINUTES OF JS VIDEOS

CSS
POWER UP WITH SASS & STYLUS

GOOGLE ANALYTICS
How to use the latest updates

AR
AUGMENTED REALITY · AUGMENTED REALITY · AUGMENTED REALITY · AUGMENTED REALITY

BUILD YOUR OWN APP
THE TOOLS & TECHNIQUES TO GET STARTED TODAY

SAY HELLO TO
THREE.JS
Learn to compose your first 3D scene

GENERATE
WEB COMPONENTS
Create standards-friendly custom elements

NATIVESCRIPT 4
Get a close-up view of seven of the best new features

STYLING REACT
Introduce dynamic design with the Fela framework

ANIMATE UI
Engage visitors with personalised profile cards

FUTURE
ISSUE 276

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

Build your own reality

Augmented Reality is no stranger to the web, but it's only now that it is really taking hold. There are no major standards to follow, but everything is evolving — and fast. Google are on-board with their Android and iOS browsers and Jerome Etienne is making fast and efficient Augmented Reality on the web a reality with his AR.js library.

With browser support helping to complete the WebAR picture, now is a great time to think of creating your very own AR app. And, this is exactly what our latest lead feature has to offer. Find out how to create custom markers and build your app with AR.js. Get ready to impress.

Elsewhere we head back to one of web designs staples, CSS. We love CSS. This time out it's CSS preprocessors. Sass is at the forefront, and we take a closer look at what it can do for you. But, it's not the only option for powering up your CSS. We weigh up the pros and cons of seven alternatives and help you decide which one is the best one for you.

Looking to create native iOS and Android apps with JavaScript? Then NativeScript 4 could be what you need. Take a look at the latest new features and find out what they can do for you. Plus, check how to use the latest Google Analytics updates, style React with Fela, and quickly create custom elements with Stencil.js. Enjoy.

> It's no surprise that AR has arrived on the web and this gives web designers some important new areas to consider for content creation

Highlight

> It can be very tricky to get what you want and keep performance high using off-the-shelf tools, so we made our own

makemepulse are building personalised environments that are unique, creative and new. **Web Designer** finds out more. **Page 32**

Follow us on Twitter for all the news & conversation @WebDesignerMag
Visit our blog for opinion, freebies & more www.creativebloq.com

This issue's panel of experts

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

Mark Shufflebottom

Mark is a Professor of Interaction Design at Sheridan College, an Adobe Education Leader and a veteran contributor to ***Web Designer***. In this issue he explores what is happening in the world of WebAR and takes you through the process of creating markers and building an app using the AR.js library. **Page 42**

> Whatever skill level you are, it won't make much difference if you don't have decent content. Think about appropriate use cases for AR before jumping in

Ashley Firth

Ashley is Head of Frontend development and Digital Accessibility at startup energy supplier Octopus Energy. In this issue, he weighs up the pros and cons of CSS reprocessors, focusing on the best of Sass and Stylus. **Page 66**

Tam Hanna

Tam started out wiggling command lines on process computers and is a veteran on many programming languages. This issue he takes a closer look at NativeScript 4 and seven of its best new features. **Page 74**

Richard Mattka

Richard is an award-winning Interactive director, designer and developer. In this first tutorial in an ongoing 3D programming series, he shows you how to create your first WebGL 3D scene, using the popular Three.js library. **Page 54**

Paul Betteridge

Paul has over 15 years' experience leading top digital organisations and brands across the UK. This issue he takes a look at the latest Google Analytics updates and how they can help you better understand a user's journey. **Page 58**

Simon Jones

Simon leads teams of frontend engineers working with different frameworks. Web components offer an appealing way to create standards-based reusable code. This issue he looks at how stencil.js makes it straightforward. **Page 86**

Neil Pearce

Neil is a frontend developer and designer who has worked in the industry for over ten years. In this issue he demonstrates how to add engagement to UI profile cards with a collection of CSS animation techniques. **Page 62**

Matt Crouch

Matt is a frontend developer at Vidsy in London. When styling React components, it's easy to end up juggling class names. In this issue, Matt takes a look at Fela – a JavaScript approach to slimming down your stylesheet. **Page 80**

Leon Brown

Leon is a freelance web developer and trainer who assists web developers in creating efficient code for projects. This issue he recreates a host of techniques as inspired by the top-class sites seen in Lightbox. **Page 14**

Follow us!

Facebook: *www.facebook.com/WebDesignerUK*
Twitter: *https://twitter.com/webdesignermag*

web designer

Future PLC Richmond House, 33 Richmond Hill, Bournemouth, Dorset, BH2 6EZ

Editorial
Editor **Steven Jenkins**
steve.jenkins@futurenet.com
01202 586233
Designer **Harriet Knight**
Editor in Chief **Amy Hennessey**
Senior Art Editor **Will Shum**

Contributors
Hilary Stephenson, Mark Billen, Leon Brown, David Howell, Mark Shufflebottom, Richard Mattka, Paul Betteridge, Neil Pearce, Ashley Firth, Tam Hanna, Matt Crouch, Simon Jones, Philip Morris, Rob Mead-Green, Ryan Wells

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

Subscriptions
Email enquiries **contact@myfavouritemagazines.co.uk**
UK orderline & enquiries **0344 848 2852**
Overseas order line and enquiries **+44 (0) 344 848 2852**
Online orders & enquiries **www.myfavouritemagazines.co.uk**
Head of subscriptions **Sharon Todd**

Circulation
Head of Newstrade **Tim Mathers**

Production
Head of Production **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**
Commercial Finance Director **Dan Jotcham**
Editorial Director **Paul Newman**
Head of Art & Design **Greg Whittaker**

Printed by William Gibbons, 28 Planetary Road, Willenhall, WV13, 3XT

Distributed by Marketforce, 5 Churchill Place, Canary Wharf, London, E14 5HU www.marketforce.co.uk Tel: 0203 787 9060

ISSN 1745-3534

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


PUT A PAUSE IN YOUR DAY
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

Meta

Inside issue 276

Contents

Cutting-edge features, techniques and inspiration for web creatives

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

webdesigner@futurenet.com • @WebDesignerMag • www.creativebloq.com

66
Power up your CSS
Discover the best in preprocessors

74
NativeScript 4
The new features that you need to know

FileSilo

- Part 3 of Beginner's JavaScript video guides from Killersites
- 27 Rogue PS presets (worth $14)
- 26 Orange Teal PS filters (worth $24)

"A particularly worrying trend we're seeing from some of the bigger brands is Dark UX patterns"

COMMENT — Hilary Stephenson – p11

Tutorials

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

Web Workshop

Web Developer

Header

The tools, trends and news to inspire your web projects

What are the digital trends in 2018?

Econsultancy recently released its 2018 Digital Trends for Creative and Design Leaders report. **Web Designer** takes a closer look

Trends in the world of design and creative are always changing. Designers and creatives are always to looking to see what the next trends are and, hopefully, jump on board to ensure that they, or their company, are ahead of the game, or at the very least on track with what is happening.

Econsultancy, in partnership with Adobe, recently released its 2018 Digital Trends for Creative and Design Leaders report. So what does the document reveal? The key findings of the report found that "The frontend experience is becoming increasingly important in a world of digital disruption and lower consumer tolerance of poorly designed products". Elsewhere the key findings state "that design-driven companies are almost twice as likely as other companies to be significantly outperforming their competitors (32 per cent vs 17 per cent)". While design is performing well, what are the opportunities that excite? According to the report, it is about delivering personalised experiences in real time. This is significantly ahead of the technology trends that are getting us at **Web Designer** excited. So what are they? Artificial Intelligence, Virtual Reality, Augmented Reality, Internet of Things, payment technologies and voice interaction. These are all technology trends that are already making an impact. While delivering personalised experiences may be the one that gets the juices flowing it seems AI is already benefitting companies. Maybe that's why they are not so excited about AI; it's already old news. The report states that "Artificial Intelligence is revolutionising the ability to deliver one-on-one marketing in real-time". We read a lot about how AI is infiltrating our everyday lives and taking over mundane tasks, but what are companies using AI for? Apparently it is the analysis of data, email marketing, programmatic advertising, on-site personalisation (there's that word again), content creation and automated campaigns. At least it's not taking over design jobs just yet.

So what else does the report reveal? This one may not come as any surprise. Poor processes are slowing down creative and design leaders. Perhaps they should be looking to put better processes in place. Simple, but effective.

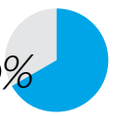
This stat may be the most interesting of all: creative and design is in short supply. The report states that "More than a third of in-house practitioners cite access to creative talent as a key challenge". It also goes on to say that the "retention of the right people with the right skills" is a major barrier. Start honing those design skills and start earning the big bucks. Your design skills are needed.

If you want to get a look at the full 25-page report and find out more you can download it from **www.adobe.com/uk/modal-offers/digitaltrends_creative.html.** You will need to sign up first.

STAT ATTACK

DESKTOP BROWSERS

Is Google Chrome getting even more popular worldwide?

Chrome
66.93%

Up 3% on the previous year

Firefox
11.55%

Down 3% on the previous year

Internet Explorer
6.97%

Down just over 2% on previous year

Safari
5.48%

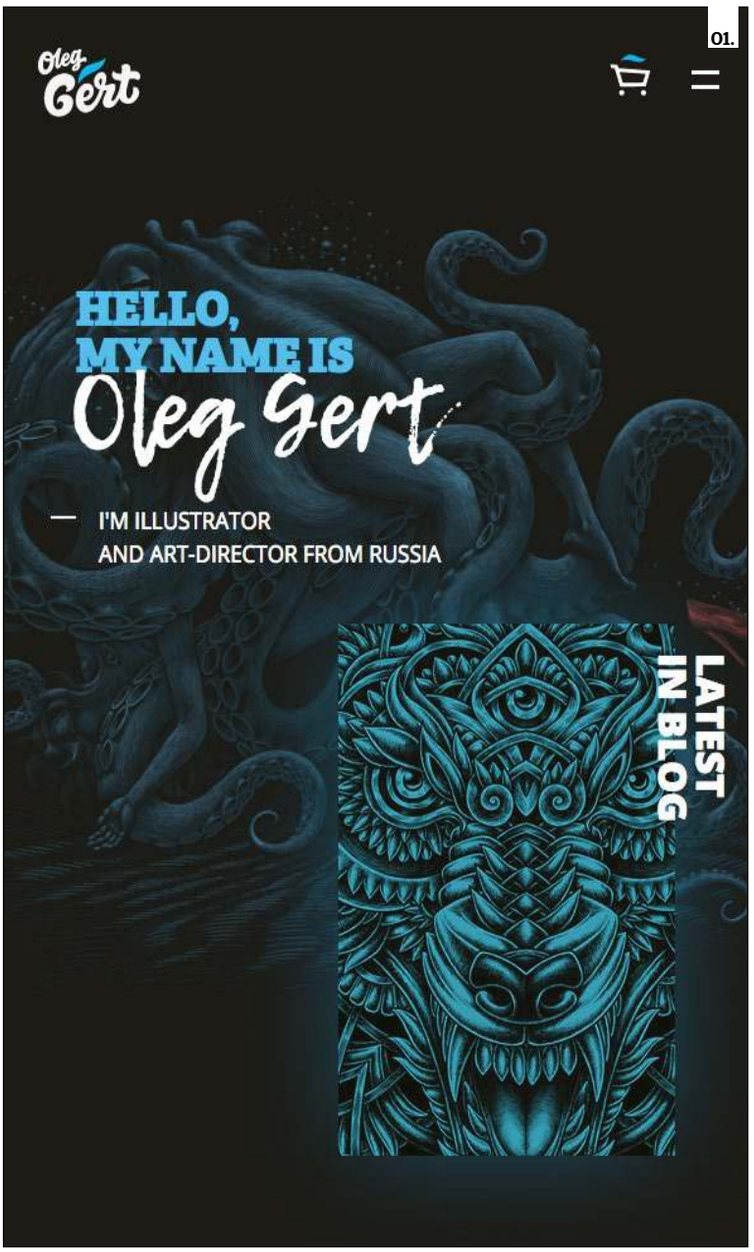
Up 0.2% on previous year

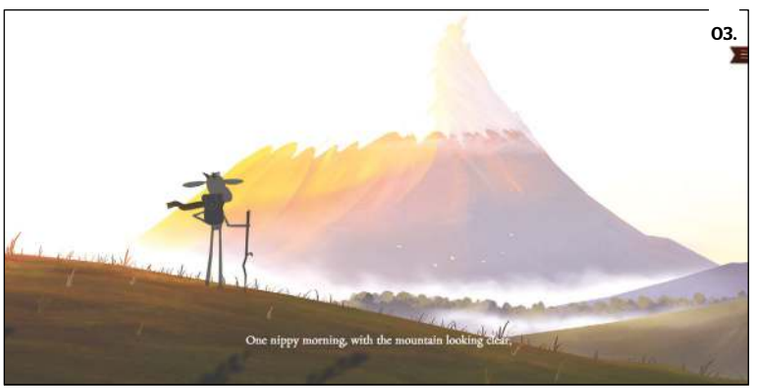
Edge
4.15%

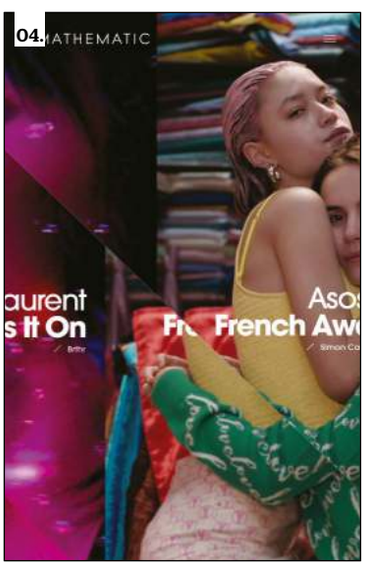
Up nearly 0.4% on previous year

Source: gs.statcounter.com (correct as of May 2018)

Sites of the month







01. Oleg Gert
oleggert.ru/en
A straightforward site that seduces the viewer with its gorgeous graphics.

02. Mathis Biabiany
www.mathis-biabiany.fr
These crazy particle effects and transitions are a must-see.

03. Oat the Goat
oatthegoat.co.nz/intl.html
A simple animated story, with a message about kindness, that draws the user in.

04. Mathematic
www.mathematic.tv
Smart split-diagonal transition immediately grab the viewer's attention.

Graphics
So Foot
bit.ly/2HlBraC
Gorgeous style and illustrations from Cristiano Siqueira celebrating the World Cup.

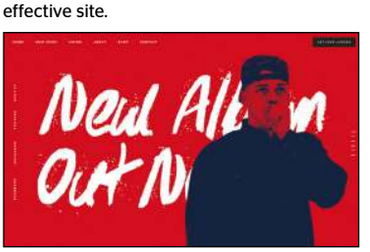
Colour picker
Balcony Vibe 2
bit.ly/2LnyuIJ

#B2B3AD
#B1D235
#5F4419
#EEE4CA
#37803E

Typesetter
Joane
bit.ly/2JjT3FF
An elegant serif with a warm unique style. Comes with engraved and deco versions.

ABCabc
ABCABC

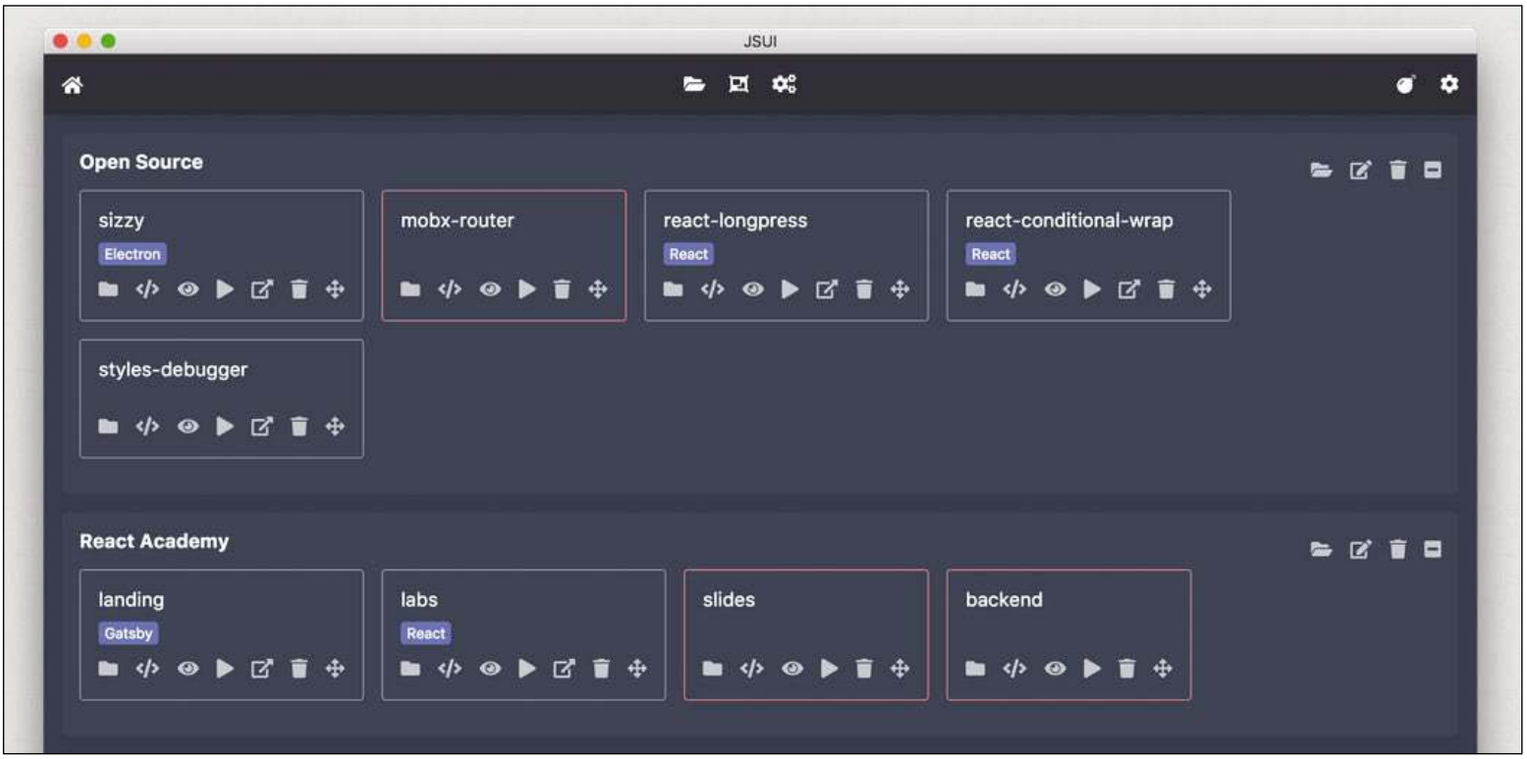
WordPress
Zample
bit.ly/2swqbDM
Full screen imagery coupled with bold, bright colours make for a simple but effective site.



CONTACT US AT: webdesigner@futurenet.com | @WebDesignerMag

webkit

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



JSUI

github.com/kitze/JSUI

Need a tool for visually organising, creating, and managing JavaScript projects? This tool from Generate London **(www.generateconf.com)** speaker Kitze can organise and group apps, generate new apps, search apps, has a project dashboard, and lets users search project files, run scripts, manage dependencies and apply plugins.

ml5js

ml5js.org

This library provides access to machine learning algorithms and models in the browser. It builds on top of the JavaScript library TensorFlow.js.

hybrids

github.com/hybridsjs/hybrids

Hybrids is a UI library for creating Web Components with a simple API. It uses plain objects with property descriptors and pure functions.

Paper.js

paperjs.org

Paper.js calls itself 'The Swiss Army Knife of Vector Graphics Scripting' It is an open-source scripting framework that runs on top of the HTML5 Canvas.

TOP 5 Web conferences – August 2018

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

TypeCon2018

typecon.com

Get to see some of the most influential names in type and design at this five-day conference in the US.

CoderCruise

codercruise.com

You'll need your sea legs for this one. Go on a cruise and get code insights while sipping a piña colada.

UX Week

uxweek.com

Get four days of community, inspiration and skills building from leading experts in UX design and beyond.

An Event Apart

bit.ly/2JsfSXC

This well-known event hits Chicago with 12 speakers, focusing on digital design, UX, content, code, and more.

Byteconf

byteconf.com

You won't need to leave your home for this one, and it's free. A two-day, single-track React conference streamed live.

The ethical minefield of design

Hilary Stephenson discusses how digital design may not be as ethical as we think

Hilary Stephenson
Managing Director at Sigma
wearesigma.com

> “A particularly worrying trend we’re seeing from some of the bigger brands is Dark UX patterns”

Design for digital platforms has the potential to influence the lives of billions of people across the globe. It is, therefore, our ethical imperative to ensure that digital products and services are designed to be as inclusive as possible, by ensuring they can be used easily by everyone. However, despite being well-intentioned, organisations of all shapes, sizes, and sectors, currently fall short in this regard.

Considering our increasing reliance on tech and digital services, a world in which we do not consider the ethical consequences of our design decisions is a world in which many people may be discriminated against, excluded, and potentially even harmed by the tech we increasingly rely upon.

The issue of ethics in digital design was a theme at our recent design conference Camp Digital, with internationally recognised design coach Per Axbom giving a thought-provoking talk on the topic. With Per’s talk in mind, it is clear that there are both intentionally and unintentionally unethical practices being adopted throughout the industry.

As the saying goes, “the road to hell is paved with good intentions” and unfortunately, many well-intentioned designers and businesses are increasingly trying to do the right thing, but letting bad design and development approaches get in the way. Take mindfulness apps as an everyday example. Designed to promote calm and emotional well-being, these apps send the user periodic notifications to encourage them to meditate. But, rather than being a helpful reminder, these can cause feelings of stress, anxiety and shame in the user; the exact opposite of their intended effect.

A more disturbing example of unintentional unethical design is machine bias. This occurs when errors are made in the machine learning processes and bear worrying similarities to human cognitive biases. While the study of machine bias is still in its infancy, there are already several prominent examples. Try entering the term “professional hair” into Google Images, and you’ll see the results are of predominantly Caucasian women. Now try entering the term “unprofessional hair” and see what comes up. Worrying, isn’t it? Perhaps even more concerning is that this bias also extends to the software used to profile and predict future criminals. This is technology that is supposed to make us safer, but in fact just furthers negative racial stereotypes. Investigative journalism newsroom ProPublica ran a Pulitzer-prize nominated study on this phenomenon, which found that the algorithms were not only spectacularly unreliable, but also biased in favour of white defendants, falsely flagging black defendants as future criminals at almost twice the rate of white defendants.

Unfortunately, not all bad design decisions occur by accident. Design can be an incredibly powerful means to influence user behaviour and many brands have woken up to this fact.

A particularly worrying trend we’re seeing from some of the bigger brands is Dark UX patterns. These are (for lack of a better term) “psychological tricks”, which brands deploy to encourage users to give up their money, data, or even simply to stay on their site longer than they otherwise would.

This issue is prevalent across all sectors, but particularly in retail, leisure and travel, where there is a clear financial incentive to keep users engaged and steer them towards purchasing certain products. To use a common example, Amazon “lures” customers into signing up for a free Prime trial by using a bright yellow ‘FREE one-day delivery’ button, while greying out the simple ‘proceed to checkout’ button. Customers’ eyes are drawn to the colourful option, which doesn’t clearly stipulate that a subscription charge would be taken monthly as soon as the 30-day trial period is over.

To change the pattern of unethical design, we must look beyond simple inputs, processes and outputs. It’s time for us to start thinking more long-term.

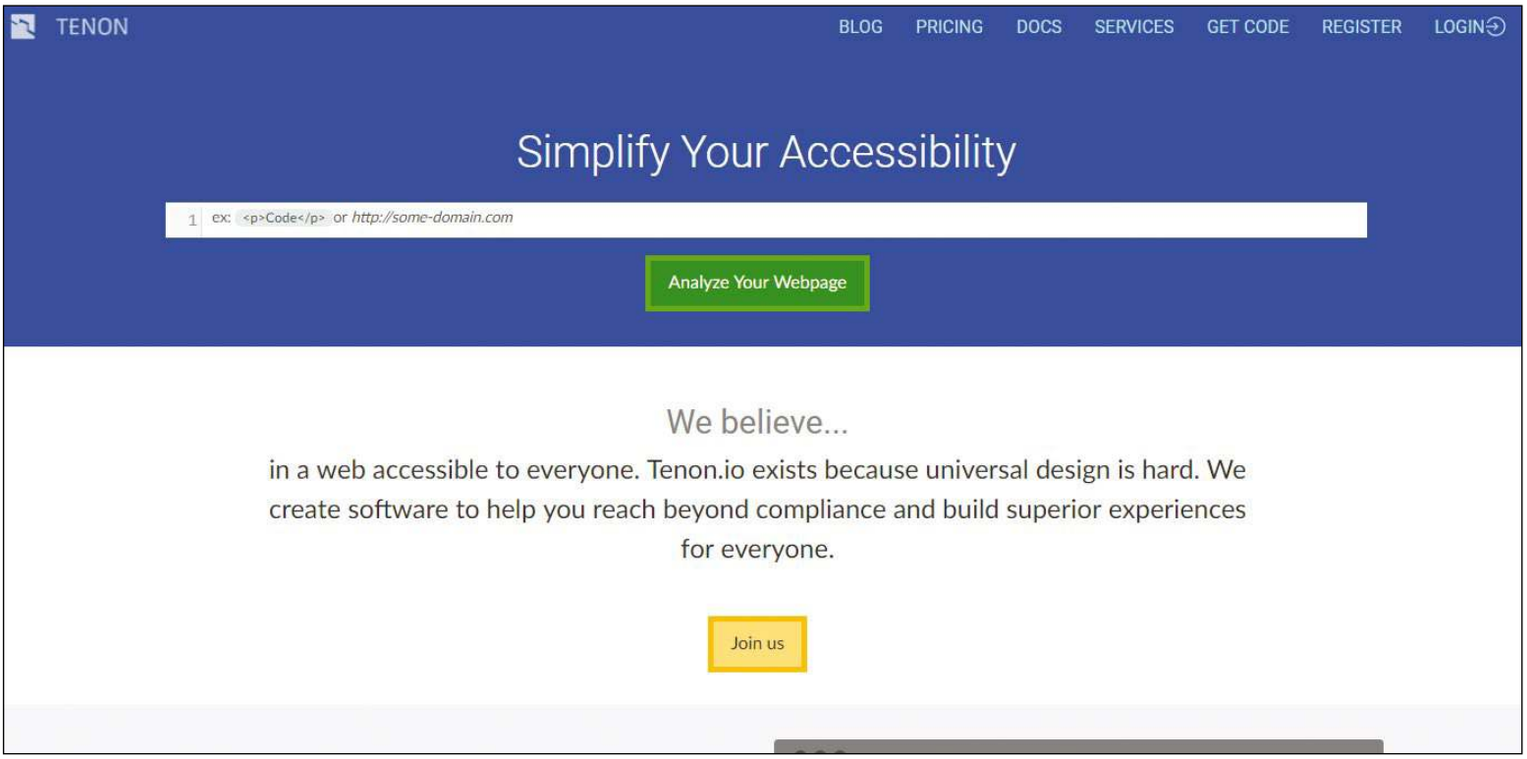
Every output has an outcome, and this outcome will have a longer-term impact which we need to begin taking into consideration if we are to become ethical, human-centric designers, and thus ensure that nobody is left behind or harmed by the technology that we design and build.

Remember, harm does not have to be something you do intentionally; sometimes it’s just the absence of good. If we have the opportunity to design more ethically and opt not to take it, we are contributing to harm in our own way. It’s time for this to change.

CONTACT US AT: webdesigner@futurenet.com | @WebDesignerMag

webkit

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



Tenon

tenon.io

How important is web accessibility? Extremely important. The web is for everyone so you need to make sure your site is up to scratch. Tenon is a paid-for tool that boasts a comprehensive set of features. There is a Free option, but this is limited. For a quick test, simply add the desired URL to be see a list of issues that the tool has found.

1.1.1.1

1.1.1.1

Everyone uses the web and wherever you go can be tracked. This neatly named tool will protect your privacy and speed up your browsing.

Wakamai Fondue

wakamaifondue.com

Need to know more about a font? Then simply drag the font onto the page for more info including characters, glyphs, size, format, designer and more.

Retro Hit Counter

joshwcomeau.github.io/react-retro-hit-counter

Who can resist a tool that uses the Comic Sans font? Recreate a GeoCities hit counter using the simple slider tools.

TOP 5 WordPress themes

Need to get a good-looking website up and running quickly? Try one of these themes

Redy

redy.axiomthemes.com

A sports theme that's big on images and numbers. Includes options to add reasons, schedules and running routes.

Tamarind

bit.ly/2sLIaqU

A multi-faceted theme with a host of home page designs that all use fullscreen imagery to show off the product.

Cyzen

hbit.ly/2sOQg0I

A straightforward, single-page theme with neat and clever touches. An animated opening window impresses.

Mifo

bit.ly/2JyerH6

A clean, contemporary theme with a full-width image slider as its focal point. Neat grids and simple animations.

AMA.ALI

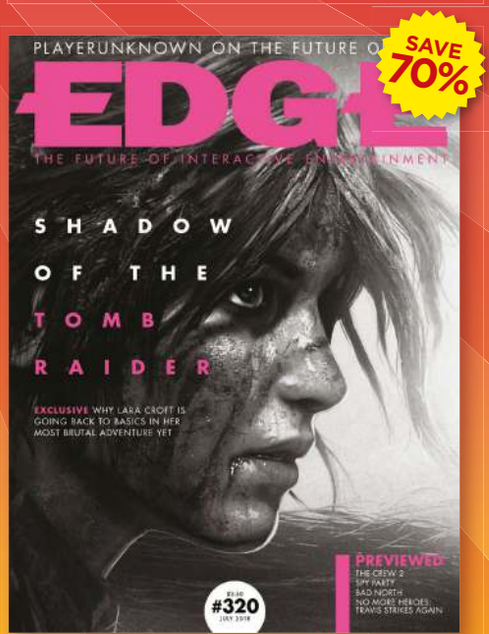
bit.ly/2I3nNA9

A modern, clean eCommerce theme that works on all screens. It boasts a host of diiferent home page designs.

SUMMER SALE!

TRY THREE ISSUES FOR £5*

BIG SAVINGS ON OUR BEST-SELLING MAGAZINES



For great savings on all of our magazines, see the entire range online

myfavouritemagazines.co.uk/summer182

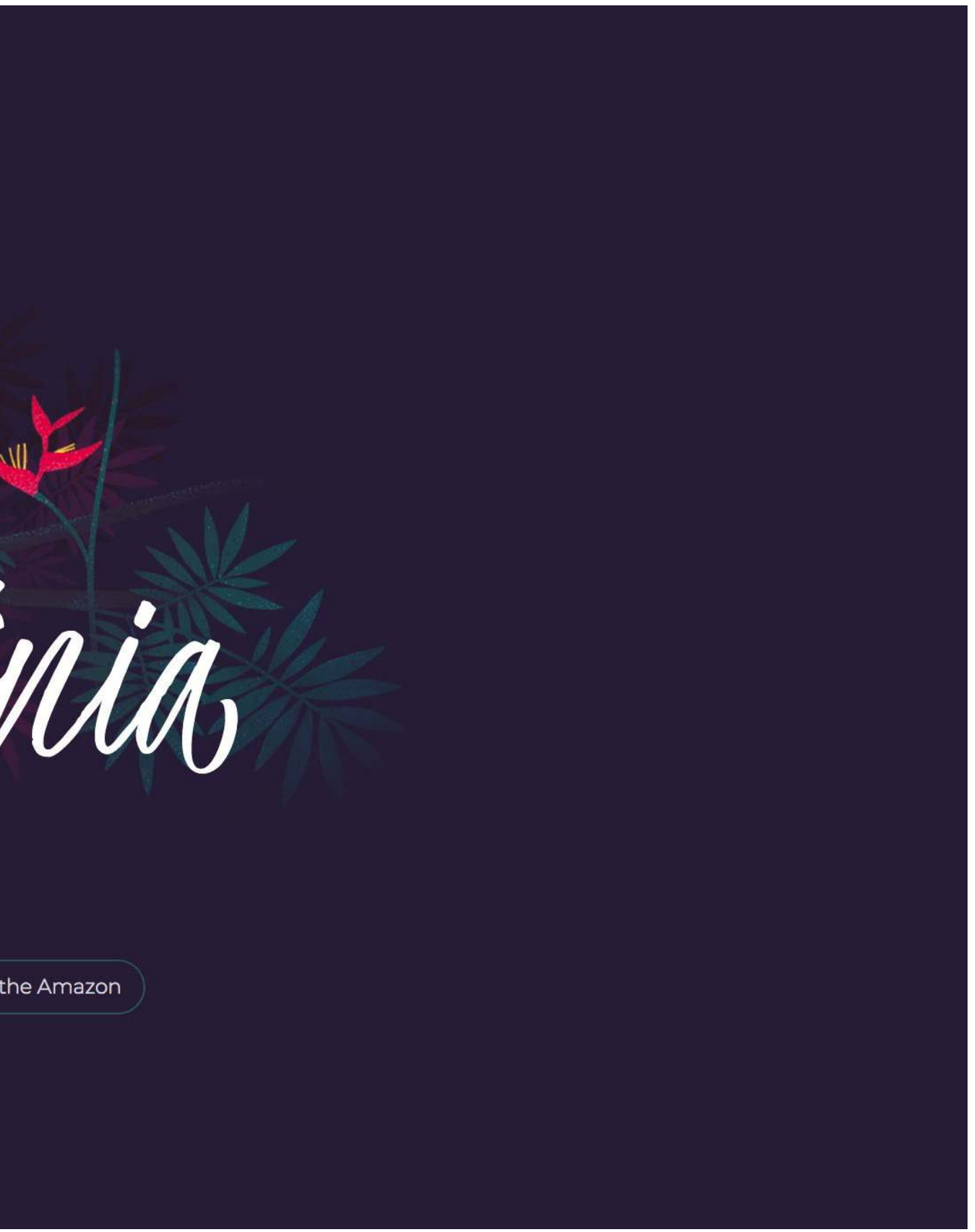
Hotline **0344 848 2852**

***TERMS AND CONDITIONS:** The trial offer is for new UK print subscribers paying by Direct Debit only. Savings are compared to buying full priced print issues. You can write to us or call us to cancel your subscription within 14 days of purchase. 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 upon request. UK calls will cost the same as other standard fixed line numbers (starting 01 or 02) are included as part of any inclusive or free minutes allowances (if offered by your phone tariff). For full terms and conditions please visit: bit.ly/magtandc . Offer ends 31st July 2018

Amazônia Font https://amazoniafont.com

Designer:
Trama http://tramastudio.net

“Interactive font specimen and eCommerce website that showcases Amazônia, a font design inspired by the Amazon rainforest”



Colours

#251B3E

#602C70

#E82152

#1F747E

Tools

Magento Enterprise, jQuery, WooCommerce, SVG

Fonts

Amazônia Script comprises of 763 glyphs, supports 85 languages and is naturally featured extensively throughout to promote the typeface.

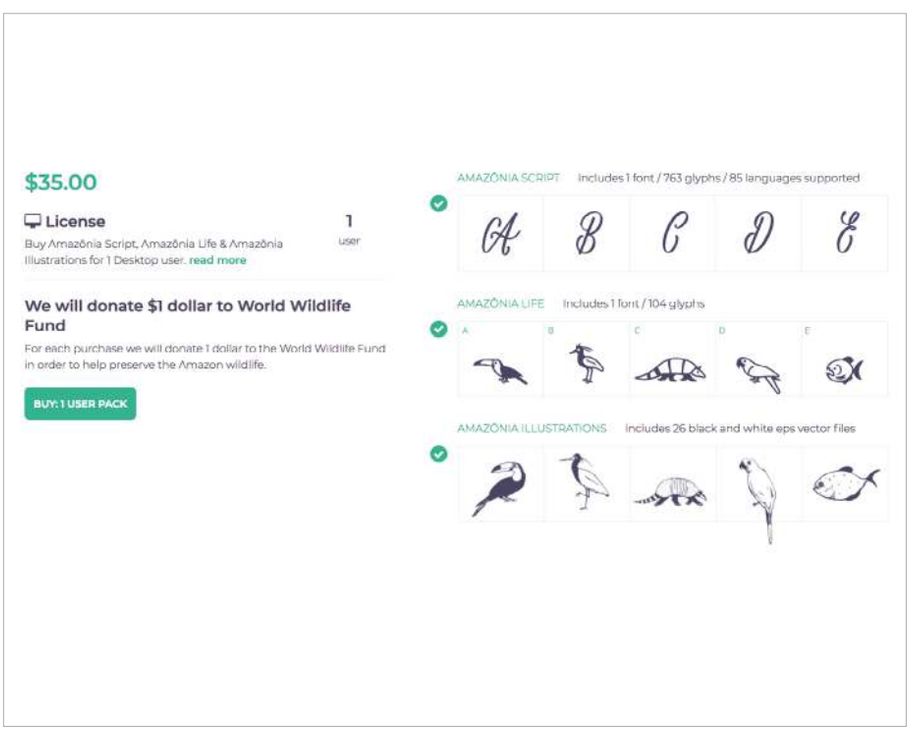
1234567890

1234567890

Amazônia Script Life and Life Alt provide a contrasting character set of 52 endangered animal dingbats.

Above
Colourful illustration work describes every featured animal with a combination of overlaid transparent PNG images

Above
An alphabetised circular wheel identifies each endangered species and plots the location of their habitat on a dynamically updated globe



Above
The site has an eCommerce element where licenses for the Amazônia Script and Life font sets can be purchased

Create a visually interactive expanding content element

Avoid content becoming cluttered by allowing click and tap interactions to open additional content

1. HTML document

The page document is defined using the HTML container tag. This is responsible for storing the head and body sections of the page. While the head section is used to load the external CSS and JavaScript, the body section is used for the page content created in step 2.

```
<!DOCTYPE html>
<html>
<head>
<title>Expanding Menu</title>
<link rel="stylesheet" type="text/css"
media="screen" href="styles.css"/>
<script src="code.js"></script>
</head>
<body>
  *** STEP 2 HERE
</body>
</html>
```

2. Page content

The HTML consists of an article container used to store the content components that will be displayed. A 'data-expand' attribute is applied to the article container, which will be used by JavaScript and CSS to apply the required functionality and styling. This approach keeps the HTML minimal and avoids unnecessary complexity.

```
<article data-expand>
  <span>Something 1</span>
  <h2>More details</h2>
  <p>...</p>
</article>
```

3. Event listeners

Create a file called 'code.js'. After the page has loaded, a search is performed for all elements with the 'data-expand' attribute. A for loop is used to reference each element found so that a 'click' event listener can be applied. This allows for a function to be executed when the 'data-expand' container is clicked.

```
window.addEventListener("load",function(){
  var nodes = document.
querySelectorAll('[data-expand]');
  for(var i=0; i<nodes.length; i++){
  nodes[i].addEventListener("click",
function(){
  *** STEP 3 HERE   });
  } });
```

4. Event functionality

The requirement to open and close the container is based around the value attached to the 'data-expand' attribute. Where the 'data-expand' attribute is set to 'open', the value is set to blank; the value is set to 'open' when not already this value. This allows the value to be switched between 'open' and blank when clicked.

```
if(this.getAttribute("data-expand") !=
"open")
  this.setAttribute("data-expand",
"open");
else
  this.setAttribute("data-expand", "");
```

5. CSS initiation

Create a new file called 'styles.css'. This first step applied to the CSS stylesheet sets the default settings for the page. A black background is applied, while the page document and body are set to cover the full window size. Default text alignment is set to centred for this example.

```
html, body{
  display: block;
  background: black;
  width: 100%;
  height: 100%;
  color: #fff;
  text-align: center;
}
```

6. Expand container

The default settings for the 'data-expand' container are defined with a thin red border and padding. The cm unit measurement guarantees consistency regardless of pixel size. For the default status of 'data-expand', the 'close' animation is applied to the container, with all items that are 'not' the first child being hidden.

```
[data-expand]{
  display: inline-block;
  border: .02cm solid red;
  border-radius: 2em;
  padding: 1em;
  overflow: hidden;
}
[data-expand=""]{
  animation: close 1s forwards;
}
[data-expand=""] > *:not(*:first-child){
  display: none;
}
```

7. Open status

This step detects the attribute changes applied via the JavaScript. When 'data-expand' is set to open, the 'open' animation is applied to display over a duration of one second. This status also switches the visibility of first level child elements. While the first child is hidden, elements that are 'not' the first child are made visible.

```
[data-expand="open"]{
  animation: open 1s forwards;
}
[data-expand="open"] > *:first-child{
  display: none;
}
[data-expand="open"] > *:not(*:first-
child){
  display: block;
}
```

8. Open animation

The 'open' animation is defined as starting with a red background highlight and one character height. This changes to animate to minimum size at 50% through the animation, before expanding to a height of 10 characters and 25% of the window's width. The background colour is also set to fade towards invisible.

```
@keyframes open {
  0% {background: red; height:
  1em; width: 0; padding: 0;}
  50%{ height: 0; width: 0; }
  100% {
  background: rgba(0,0,0,0);
  height: 10em;
  width: 25vw;
  padding: 1em;
  }
}
```

9. Close animation

The 'close' animation is mostly the reverse of the open animation. It starts with the same width and height that the other animation ends with. The middle animation point also sets the size to zero, before ending on an unset width and height. This allows the element to revert back to its originally defined size.

```
@keyframes close {
  0% {background: red; height: 10em;
  width: 25vw; padding: 0;}
  50%{ height: 0; }
  100% {
  background: rgba(0,0,0,0);
  height: unset;
  width: unset;
  padding: 1em;
  }
}
```

Xavier Cussó https://xaviercusso.com

Designers:

Xavier Cussó burundanga.studio/#/home **Christian MacMillan** cmacmillanmarin.com

"Webby-winning visual designer and art director Xavier presents a series of dynamic scrolling case studies of notable projects"

Colours

#130074 #14F9B4 #8552C9 #E9103A

Tools

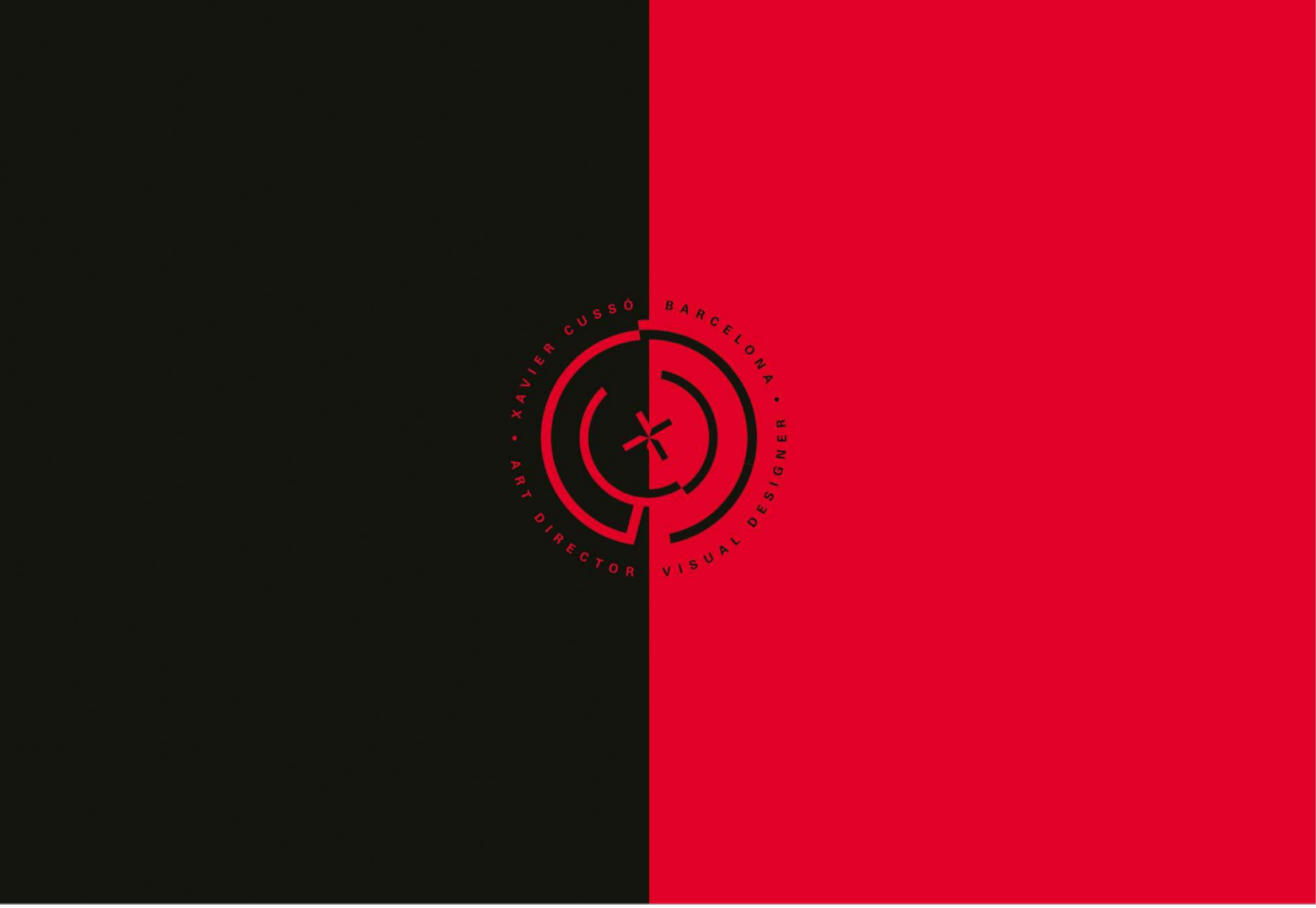
HTML5, GSAP TweenMax

Fonts

abcABC
1234567890

abcABC
1234567890

The Univers LT font by Adrian Frutiger for Linotype is the only font used across all text in either Standard or Black varieties.



Above
An opening pre-loader defines a split screen, black and red colour theme that runs throughout the main site sections

Above
The About section features a semi-transparent static image of Xavier sitting over a tilting background that uses CSS transforms

Above
Featured project shots pop with colour, floating numerals and parallax scrolling effects

Create an animated split screen loading intro

Use this split screen introduction effect to display your website content after loading has completed

1. Initiate HTML structure

The first step is to initiate the structure of the HTML document. This consists of the document container, which in turn contains the head and body sections. While the head section is used to load the external CSS and JavaScript, the body section is used to store the content created in step 2.

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

2. HTML content

The webpage content is defined as normal – in this case, a h1 title and paragraph has been placed. The loading screen is inserted as the last element to guarantee it has a z-index above all page content. This element has a 'data-loading' attribute, along with a series of inner span elements to be styled for presentation purposes in later steps.

```
<h1>Something</h1>
<p>Something more.</p>
<div data-loading>
  <span> <span></span> </span>
  <span> <span></span> </span>
</div>
```

3. JavaScript confirmation

Create a new file called 'code.js'. Having the HTML change to indicate the completion of the page loading is a handy way to trigger CSS presentation changes. JavaScript is used to apply an event listener to the page window for when loading has completed. This listener changes the value of the 'data-loading' container to 'completed' when the page has loaded.

```
window.addEventListener("load", function()
{
  var node = document.
querySelector("[data-loading]");
  node.setAttribute("data-
loading","complete");
});
```

4. Data loading container

Create a new file called 'styles.css'. This first step of the CSS file initiates the 'data-loading' container. Fixed positioning is used to guarantee that the loading screen is always visible. For the same reason, the position and size is set to cover the full screen visibility to hide any page content.

```
[data-loading]{
  position: fixed;
  display: block;
  width: 100vw;
  top: 0;
  left: 0;
  height: 100vh; }
```

5. First level elements

The first level span elements inside the 'data-loading' container are styled to fit the full height and half width of the browser window. Absolute positioning is applied to allow these elements to be placed with pixel co-ordinates. Overflow is set to hidden so that the closing animation hides its inner content.

```
[data-loading] > *{
  display: block;
  position: absolute;
  width: 50vw;
  height: 100vh;
  overflow: hidden;
}
```

6. Unique positioning

The left and right side of the loading screen require unique positioning for the effect to work. The first child span element inside the 'data-loading' container is set to be positioned to the top-left corner with a black background. The second span element is positioned from the bottom right with a red background.

```
[data-loading] > *:first-child{
  top: 0;
  left: 0;
  background: black;
}
[data-loading] > *:last-child{
  bottom: 0;
  right: 0;
  transform: rotate(180deg);
  background: red;
}
```

7. Second level children

The second level children of the 'data-loading' container are the span elements inside the span elements. These elements are placed for visual effect – appearing as blocks set as the reverse colours of their parent container. Absolute positioning along with the 'vh' measurement unit allow these elements to be positioned in relation to the size of the browser window.

```
[data-loading] > * > *{
  display: block;
  position: absolute;
  width: 10vh;
  height: 10vh;
  top: 45vh;
  right: 0;
  background: red;
}
[data-loading] > *:last-child > *{
  background: black;
}
```

8. Loading complete

The JavaScript 'complete' value applied to the 'data-loading' attribute indicates that the page has completed loading. CSS rules are defined to trigger the required animations when this occurs. The main 'open' animation is set to play over a one second duration and uses 'forwards' play to stop on the last animation frame.

```
[data-loading="complete"]{
  animation: close 0s forwards;
  animation-delay: 3s;
}
[data-loading="complete"] > *{
  animation: open 1s forwards;
  animation-delay: 2s;
}
```

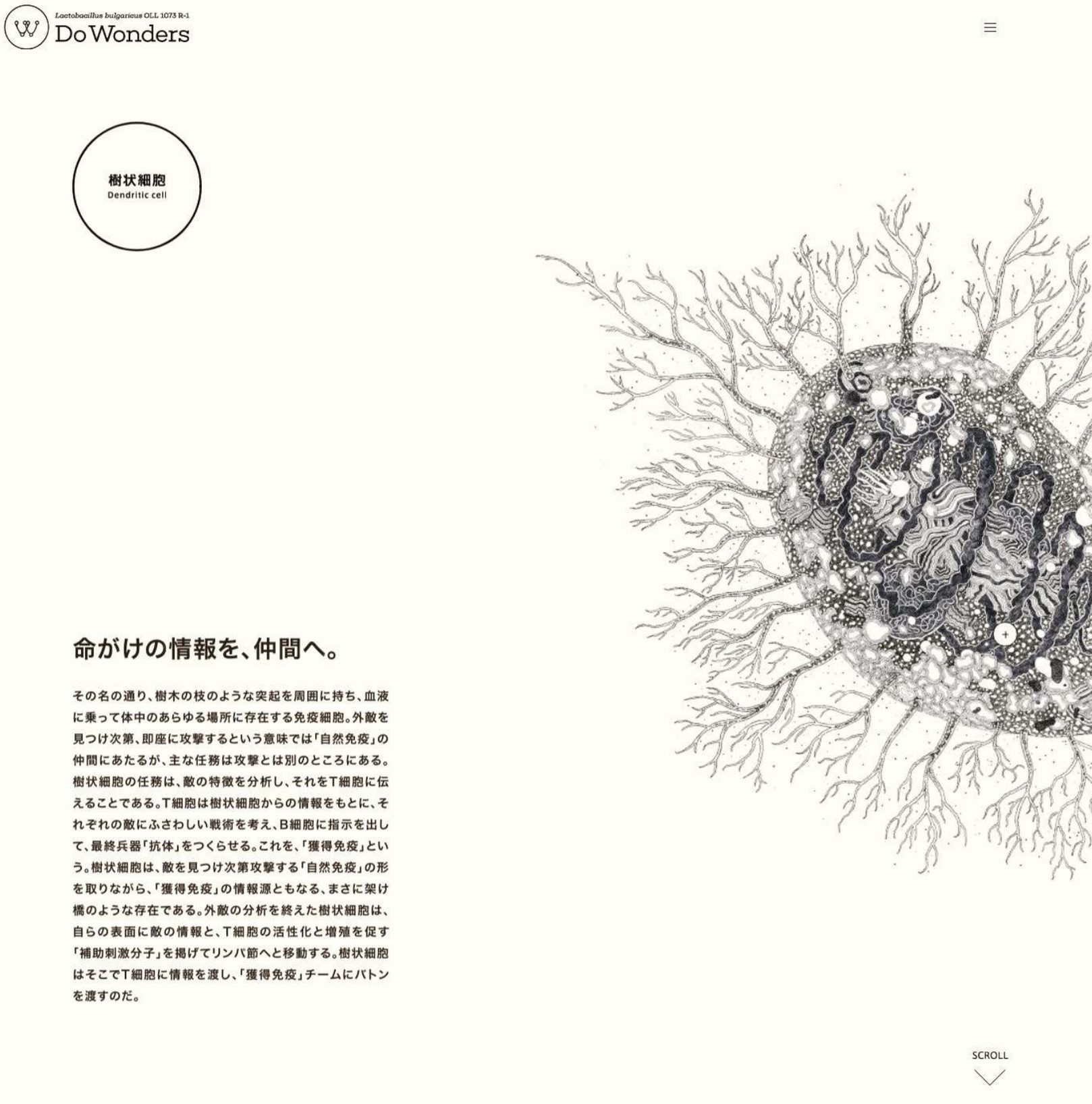
9. Animation definition

The open and close animations are defined using keyframes. The open animation is merely defined as starting from the full window height, then ending at zero height; resulting in elements animating to disappear. The close animation is used to place the 'data-loading' container below the page content – avoiding it becoming an obstruction to user interaction with page content.

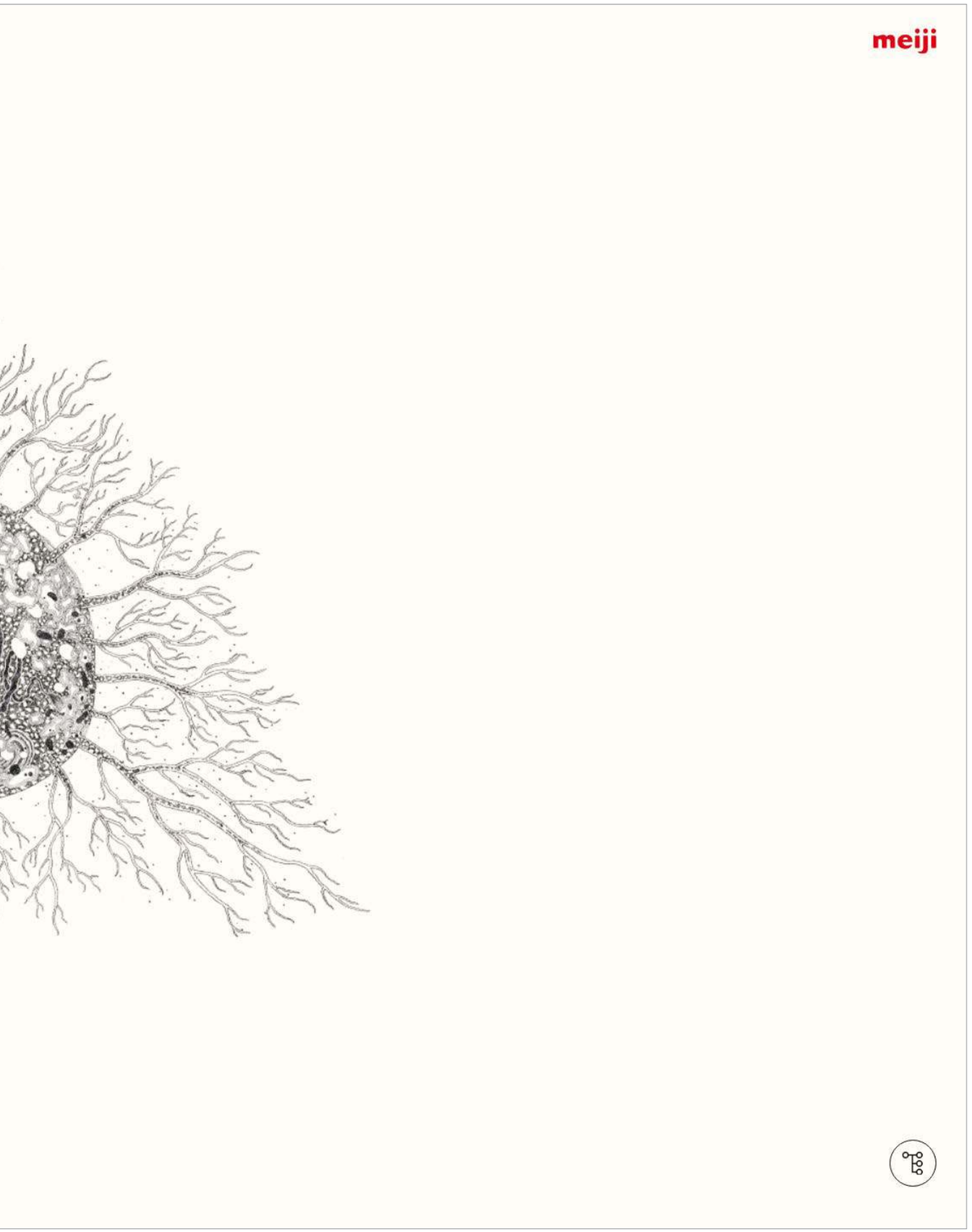
```
@keyframes open{
  from{ height: 100vh; }
  to{ height: 0;}
}
@keyframes close{
  to{ z-index: -100;}
}
```

Do Wonders meiji.co.jp/do-wonders

Designer:
Dentsu dentsu.co.jp



“A Japanese web project designed by Dentsu, Do Wonders illustrates the mechanics of the human immune system in an engaging way”



Colours

#3D424D

#FFFBF4

#ED252D

Tools

jQuery, WebGL, Contegro / Ektron (CMS)

Fonts

abcABC
1234567890

The standard choice for Japanese Kanji text display is Hiragino Gothic Pro, used here exclusively.

abcABC
1234567890

Meiryo font by Microsoft is also listed within the site's stylesheet, most likely as a fallback option.

Create an automatically animated steam text effect

Introduce text content with a blurry steam text effect that introduces content line by line

1. Initiate HTML document

The first step is to define the document structure that will store the HTML content. This consists of the document container, which stores the head and body sections. While the head section stores inks to the external CSS and JavaScript resources, the body stores the visible content created in Step 2.

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

2. HTML content

This step defines the visible HTML content. Take note of how the texts designated to have the blur effect are all contained within a container that has the 'blur' class. This class is used by JavaScript to reference the text items in Step 3, and by CSS in later steps.

```
<h2>
  Discipline comes from
  <ul class="blur">
  <li>commitment</li>
  <li>perseverance</li>
  <li>dedication</li>
  <li>training</li>
  <li>aspiration</li>
  <li>education</li>
  </ul>
</h2>
```

3. Automated animation delay

Create a new file called 'code.js'. Each item inside the blur container is to be presented three seconds after the previous item. JavaScript is used to automate the application of unique CSS attributes. The first step of this is to select all of the first-level items inside the blur container – after the page has loaded.

```
window.addEventListener("load", function()
{
  var nodes = document.querySelectorAll(".
blur > *");
  ***STEP 4 HERE
});
```

4. Number of seconds

A 'for' loop is used to reference each item returned to the 'nodes' variable in the previous step. The index counter of the 'for' loop is used to calculate the number of seconds to assign to the animation delay attribute. As a result, each item has a delay that is three seconds longer than the previous item.

```
for(var i=0; i<nodes.length; i++){
  nodes[i].style.animationDelay =
(i*3)+"s";
}
```

5. CSS blur

Create a new file called 'styles.css'. The first step for defining the CSS presentation rules initiates each of the text items as invisible. An animation called 'animationBlur' is also applied that will animate the item into view over a five-second duration. The animation mode must to be set to 'forwards' so that it stops on its last frame.

```
.blur > *{
  opacity: 0;
  animation: animateBlur 5s forwards;
}
```

6. Initiate animation

The animation applied to the 'blur' elements in Step 5 are defined in this step. The reference to 'animationBlur' is made as a keyframes animation. The first frame 'from' sets the elements as visible with a text shadow – but with a transparent text colour. This is what produces the blurred text effect.

```
@keyframes animateBlur {
  from {
  opacity: 1;
  text-shadow: 0 0 1em rgba(0,0,0,.5);
  color: rgba(0,0,0,0);
  }
    *** STEP 7 HERE
}
```

7. Final animation frame

The 'to' frame within the animation defines the final frame that the text will be animated to. This frame sets the text shadow to disappear, along with a text colour that's fully visible. Combined with Step 6, the animation frames between 'from' and 'to' will be automatically calculated by the browser.

```
to {
  opacity: 1;
  text-shadow: 0 0 0px rgba(0,0,0,0);
  color: #000;
}
***
```


generate
The conference for web designers
LONDON
19-21 SEPTEMBER 2018
3 DAYS OF INSIGHT AND INSPIRATION
TICKETS ON SALE NOW
www.generateconf.com
#generateconf
BROUGHT TO YOU BY
web designer
CB CREATIVE BLOQ
The voice of web design
net

Out of the Box

AFFINITY CREATIVE REVEAL HOW THEY ADDED SOME MARKETING SPARKLE TO REFRESHING THE BANDIT WINES BRAND

WHEN WE DARE TO CONSIDER what the term 'brand' really means in today's multi-channel world, it's a pretty abstract yet increasingly nuanced concept. We're constantly told we're all expected to be brands of ourselves, particularly online, when it comes to projecting the most engaging aspects of our personality, experience and abilities into the digital ether. For companies, however, things are more literal, and probably more vital, when the capture of audience attention is so pivotal to market success. The pursuit of effective corporate web branding could be argued as being the most important of all, and it's hardly surprising digital agencies often need that sixth sense for good marketing. Operating out of Mare Island, California, Affinity Creative Group are one example for whom the philosophy for the work they do has been firmly formed from over 40 years of branding experience. Establishing connections with audiences via beautifully crafted projects — an internal commitment to the 'meaningfully aesthetic' — typically results in a seamless brand experience across all major consumer touchpoints. "When we apply this philosophy to projects like Bandit Wines, we are able to form meaningful relationships with our clients that translates into the best possible brand experience for their consumers," Digital Director Justin Witt begins. "From concept, to design, to development and execution, we owned every step of the creative process, and we're only able to do so because of the mutual trust that is established with our clients from day one." Indeed, it is this collaboration with Bandit Wines under the Design Diary spotlight this month, and a client keen to follow a recent repackaging with a dotcom rich with millennial appeal. As a vendor of naturally produced Californian wines housed in eco-friendly boxes, the challenge would be hitting on a full-bodied branding message worthy of toasting.

PROJECT STATS

BANDIT WINES
banditwines.com

by

AFFINITY CREATIVE
affinitycreative.com
@affinitycreative

PROJECT DURATION
1 month

NO OF PEOPLE INVOLVED
5

NAMES & POSITIONS

JUSTIN WITT
Digital Director

JORDAN JOSEPH
Lead Digital Designer

TRENT GARDNER II
Lead Web Developer

TREVOR HOOPER
Photographer

CHRIS DACRUZ
Account Executive

UNCHARTED WATERS

It's worth noting that this project was not a redesign for the Bandit Wines website, but rather one to create a brand new digital experience from scratch. Given recent work the client had done to revamp its identity, the key challenge was to acknowledge this new brand positioning and packaging design online. "We wanted to emphasise our experience within the digital realm since this is uncharted waters for the Bandit brand," admits Digital Director Justin Witt. "We didn't have a whole lot of direction to rely on from them in terms of the look and feel, meaning that we had to rely more on giving them creative consultation to ensure that their vision for the look and feel of their site aligned with the look and feel of the physical brand that they've built."

That kind of 'creative consultation' would,of course, rely on strong lines of communication where give and take on both sides would prove beneficial.

"Great ideas can come from anywhere, and we want to find the best idea within our creative wheelhouse and champion that best idea, wherever that comes from. Neither the client or agency is always going to be totally comfortable, and that's a good thing, because we should be pushing them and looking into the future to make sure that we're responding to their target market first and foremost."

FOCUS SWITCHING

"This project actually got started when we were heading back from a client meeting," Witt recalls when quizzed on Affinity's initial perceptions. "We just had a great kickoff meeting for another project with our client when we got a call from Kelly Knight, the Trinchero Family Estates web manager, asking us about another website for an up-and-coming wine brand." Bandit Wines were in fact the brand in question, one of many under the Trinchero Family Estates umbrella, with Bandit attempting to corner a millennial demographic identifying increasingly as the top consumer of boxed wine in the United States. "She told us that she was going on vacation in three weeks and that we would need to have a site done in two. When we had our kickoff meeting a few days later we discussed the preliminary brand elements that needed to be included in the site, making sure that their envisioned future for the brand would not only align with our vision for the site, but also cater to their target demographic." From here the team's goal internally was to maximise the site's impact with the limited resources at hand, including the tight timeline. A process of sketching out initial reactions to Bandit's brand story and how this could be translated into the digital space. "Their primary design elements needed to be reworked in order to fit within the desired digital design space. This is what started our initial concepts and wireframing, and allowed us to come up with questions and proposed solutions to their original proposal."

PACKAGE DEAL

After the initial briefing, the communication between the two parties would be ramped up during the wireframing and mockup stages. Affinity's ethos is no different to most agencies, preferring an open line of communication with their

client in order to fix gaps in vision, ironing out inconsistencies sooner rather than later. “All of our content and design ideas stem from inspiration from our clients and from our own experiences as creative experts,” explains Lead Digital Designer Jordan Joseph. “For the Bandit site, we wanted to pay tribute to the sustainability aspect of the TetraPak box that the wine comes in, as well as the Northern California, San Francisco look and feel that people from all over the world would recognise. Getting in to the early design stages, starting with a mood board that encompassed the brand’s vision for their site was a must. We drew upon the natural colours of the landscapes associated with the wine, as well as an expanded colour palate that complements the nature and adventure themes that the brand wanted to use to draw user attention on screen.” This early concept work quickly centred then on themes of sustainability and ecological benefits that would resonate with younger, trendier drinkers. Affinity would therefore take those implied benefits and translate them into interactive modals for the site’s UX that would focus on how the brand is paving the way for a sustainable future within the boxed wine category, and how consumers can relate to and interact with their eco-conscious packaging and design. “When this part of the site was being created, we wanted to make sure that the colours and icons used to highlight the sustainable features complemented the packaging and overall look and feel of the brand.”

> “THEIR PRIMARY DESIGN ELEMENTS NEEDED TO BE REWORKED IN ORDER TO FIT WITHIN THE DESIRED DIGITAL DESIGN SPACE. THIS IS WHAT STARTED OUR INITIAL CONCEPTS AND WIREFRAMING

TRUE COLOURS

This awareness and acknowledgement of the existing brand’s look and feel was a big deal in tackling what was a largely open creative brief. Bandit’s revised packaging creates a direct graphical association with the various Californian national parks and locations synonymous with the region, so the desire was to reflect those colours and iconography in the early visual design work. “When working with a colour palette that’s very expansive like Bandit, it’s up to us to figure out which colours would look best on screen versus what colours are actually used in the

Custom photography reinforces the Bandit Wines’ brand aesthetic

Wireframing and proof-of-concept work for the Bandit Wines site

printing of the package. An example of this would be the Chardonnay, when we decided to use more of a grey-brown palette for most of the screen versus using the bright yellow, since this would be a very bold and aggressive colour on screen. The grey-brown definitely helps us blend out the page and balances out the bright yellow." Aside from the various texture pattern and topographical line work performed for the frontend, something Affinity also did for the client was offer custom photography duties. A service they are more than comfortable delivering where projects require it, in this instance Bandit simply didn't have imagery of their new packaging other than mockups. "We were given the challenge of shooting not only for print magazines but also shooting for their new site," explains Photographer Trevor Hooper. "We weren't given any art direction for the shoot, the brand managers just wanted us to go out and shoot the areas around San Francisco and capture the Northern California landscape that embodies what their target demographic resonates with. They also didn't have actors to model for their shots, so you'll see in some of those photos that Jordan (Joseph) and Chris (DaCruz) are shown in the beauty shots."

SCROLL JACKED

Moving to the backend development and many of the possible coding challenges were averted with the use of WordPress. Its choice as the content management system underpinning the site was actually settled in the first meetings with the client, with both parties agreeing it would be the easiest and most straightforward option in terms of the final handoff and future maintenance. Essentially this approach would enable the Bandit web team to make content updates without having to return to Affinity for minor changes to copy and imagery, etc. "One of the main features of the site for the

SUSTAINABLE FUTURE

We already know that development decisions were not only affected by the short timescale for this project, but also longer-term maintenance. Bandit Wines' eco-friendly packaging was therefore not the only future-conscious theme, when it came to the site giving value to its client after launch. "When our clients don't have any type of digital or IT support internally, we want to work with them on any form of maintenance to ensure that any and all design and copy changes are done with little or no difficulties," Digital Director Justin Witt explains. "Even working within a CMS there can be bugs that arise once copy is added or deleted, or when different photography and imagery sizes are changed. It's up to us as the creative agency to advise our clients on the best practices when it comes to keeping their site updated and free of errors or bugs. Most of the time our clients trust us to handle all maintenance within their site and any updates to their eCommerce platforms, which we offer to all clients with a post-launch maintenance retainer. In addition to the retainer, we will offer up a basic CMS training program with their internal IT team to ensure that they are aware of how the site functions and what changes they will be able to make on their own if they choose to do so."

Site Hightlight

eco and bottle pages are being loaded dynamically without a full page reload," begins Lead Web Developer Trent Gardner. "Also, when we're thinking of interesting ways of adding our own touch to sites built off of a WordPress CMS, we like to innovate the way in which the user navigates through the site using techniques like scroll jacking to grab the user's attention during key moments of their digital experience." All these added features would indeed bring extra challenges from the backend work standpoint, with those eco and bottle pages Trent mentions proving most tricky it seems. "When the user is looking through the different varietals on the site, they are not able to free scroll through the site and are given a visual cue to let them know that the page is forcing them to stop. At the same time if they continue to scroll, they will notice that they have to apply a certain amount of scroll action to continue on down the page. This technique becomes a bigger issue when designing and coding for a responsive layout. The scroll jacking brings special considerations and an approach that you wouldn't be experiencing in a free scrolling site."

DECISIONS VINDICATED

Given the tight timescale, this development work extended of course right up to what was a 'soft launch' for the site. In this instance, beyond that handover on completion, it was already agreed that the client would perform the analytics, tracking and marketing efforts themselves. "From browser and screen-size testing to mobile device testing we check every possible angle for any glitches or crashing from both the design and development standpoints," confirms Account Executive Chris DaCruz. "Once we've taken all the necessary precautions, we will pass off the marketing and promotion of the site to our client, unless otherwise outlined in our proposal." Happily now, since the launch in March, Bandit Wines have expressed their satisfaction with the site's success after a significant lift in traffic and engagement. Similarly, Affinity's decision to deploy a WordPress-based solution has been vindicated by the client's appreciation over ease of use, both internally and amongst their own clients and distributors. The added responsive nature of the site has also resonated highly with the Bandit target audience, while the work has also helped to promote Affinity Creative's talents throughout this industry. "This site has been hugely successful from the agency standpoint, culminating in a nomination for an Honourable Mention from the Awwwards. As we show this site to potential clients, we see how the sustainability and millennial vibe resonates with other wine brand managers, and has proven to generate positive results from our own sales team." In keeping with most of the projects we feature, a big part of this one's perceived success is due to good collaboration and effective communication. On both sides there was a clear, shared objective to do something true to a brand identity befitting of both parties, and within a limited space of time. "After we've tested and soft-launched the site, we have a final handoff meeting with the client to make sure that all sections outlined in the brief have been taken care of," concludes Justin Witt. "For Bandit, we were able to launch the site within their allotted timeline, which is something that we would have never been able to without the excellent communication between both us and our client."

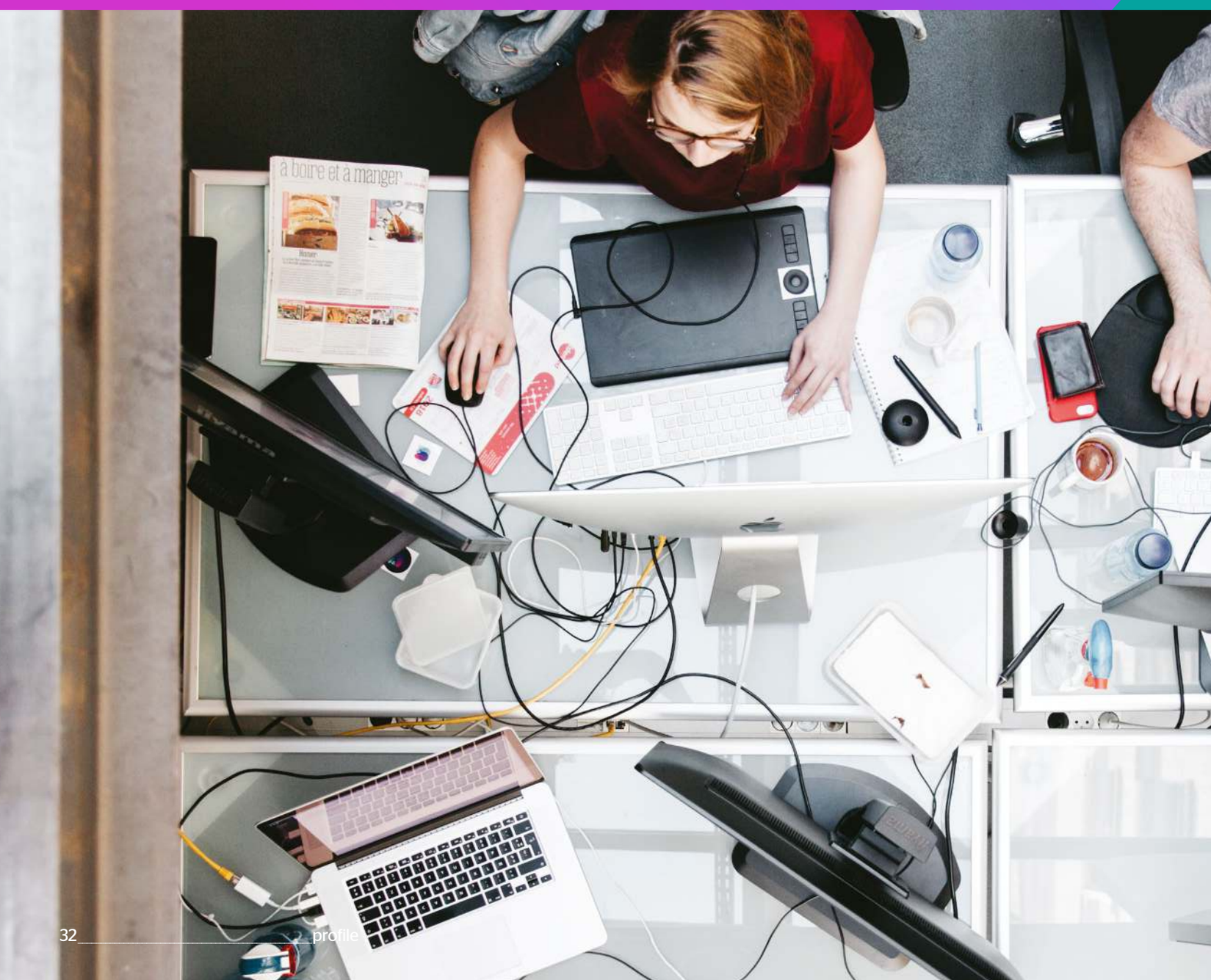
WE ASKED LEAD DIGITAL DESIGNER JORDAN JOSEPH TO IDENTIFY A PERSONAL HIGHLIGHT FOR THE SITE AND WHY IT DEMANDS CLOSER ATTENTION

"I THINK THE MOST STANDOUT FEATURE OF THIS SITE IS THE USE OF COLOUR AND SCROLL JACKING ON THE WINE VARIETAL PAGES. OUR CLIENT TRUSTED US TO BRING THEIR VIBRANT BRAND VISION TO LIFE, AND CREATING A USER EXPERIENCE THAT BECOMES SO PERSONAL AND INTERACTIVE ONSCREEN FEELS VERY REWARDING"

PULSING DESIGN

Crafting next-generation digital experiences is at the core of makemepulse. The emotional interactions they create speak to an audience that wants to see innovation and personalised environments that are unique, creative and new

makemepulse



Who **makemepulse**

What **Digital activation strategy, Digital Creation — Art Direction, Technology recommendations — Technical Direction, Development Front-End & Back-End, Websites, Mobile Apps, Interactive Displays, VR, AR, MR, Chatbots, AI, and DOOH**

Where **38 rue Legendre, 75017 Paris**

Web **makemepulse.com**

Key Clients

DDB / Ubisoft

BETC

We Are Social

Facebook

Google

The co-founders of makemepulse, Nicolas Rajabaly and Antoine Ughetto, met at Gobelins school in 2004 — a famous French school for design and technology. After school they both worked together at Publicis Net, and after spending two years there working on awarded projects for SFR, Diesel, Coca-Cola, Shu Uemura, Dior and Cartier, they decided to create their own company in June 2008.

The first job of business was of course to name their new agency. Antoine explains their approach: "Although the music service we initially created when we founded the company didn't actually happen in the end, we were thinking about our name in relation to artists, so subdomains would appear like madonna.makemepulse.com or u2.makemepulse.com, basically artist-name.makemepulse.com.

"What's great is the spirit of that founding idea endures: of music and rhythm; of everything in music being about the vibration, pulsation of the sound and the arrangement between all the instruments to create a beautiful melody. It's why we are still called makemepulse. We have this rhythm in production. We have always targeted innovative productions to create this rhythm in our work. The rhythm gives us the ability to craft amazing productions and tools (like our NanoGL rendering engine) and to offer real expertise in technological production."

The website that makemepuluse have created is minimalist, and unlike other agencies also utilises Medium — a resource of insightful content from a wide range of contributors. Antoine continued: "Our business is digital craft, so we treat our website as a showcase of our ability. It's absolutely critical that talent, prospects and clients understand what we can do even at a cursory glance.

"We create new content after every project, which is hosted on Medium. We choose to integrate with Medium rather than build case studies into the website for two reasons: firstly, because we honestly believe it's the best platform out there for people to find and share long-form content and secondly, because we want our content to be found and shared!"

How agencies attract their clients can be manifold. The cold pitch is still prevalent, but these days agencies and studios live by their reputation and word-of-mouth. For makemepulse their approach is multifaceted, as Antoine outlines:

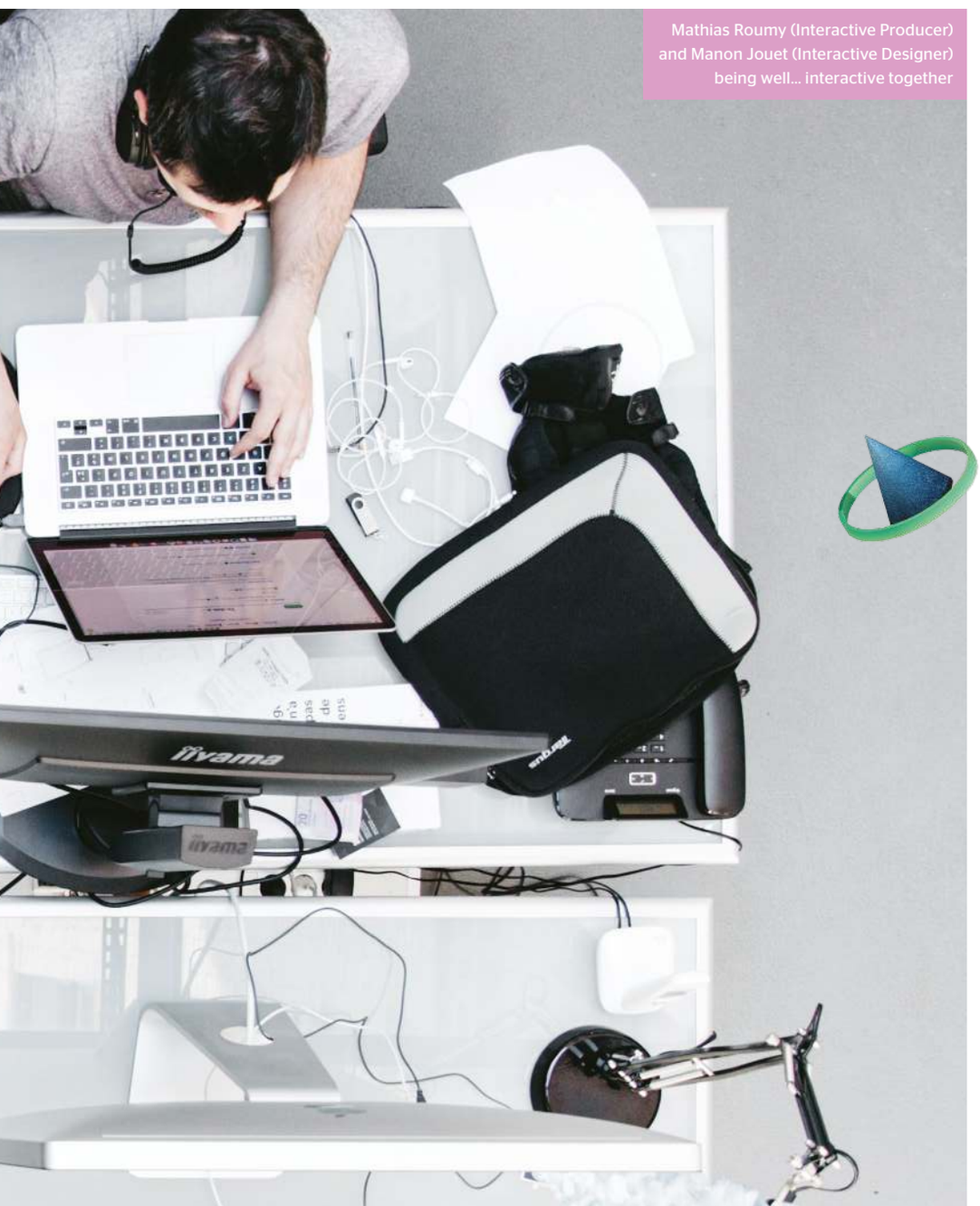
"Our HQ in Paris has now been open for ten years this year, and a new office in London that's just turned one-year old. We have an established value in the French market. Our clients and prospects understand what we do and how to work with us thanks to our reputation, so in 98 per cent of cases we are invited to the table before a pitch scenario."

"In London the culture and climate are very different," says Emma Willis, Managing Director, London. "As the new kid on the block we are working hard to earn the same trust. Right now, the tools we use are a bit of a mystery; brands and agencies alike know they need emerging tech ideas, but they don't always know what ideas they need! A lot of attracting new clients and briefs is about helping them on that journey of discovery and knowing where the tools fit into the idea."

Choosing the right clients to work with is also an important factor. The commercial viability of a project, of course, is a consideration, but so too is the work itself and how this could enrich the agency.

"We can be more selective now than when we started, but that comes more from discovering who we are as a family, the culture of our company, than about being an established agency *per se*" says Antoine. "Over the years we have realised what we like, what we are good at and how we can combine the two in business.

"It's our belief that this is why we attract the business we want to do. I am sure there are probably kinds of projects we would say no to, but we haven't ▷



Mathias Roumy (Interactive Producer) and Manon Jouet (Interactive Designer) being well... interactive together

been in that position just yet. We are, thankfully, lean enough to be able to consider the more modest productions, and organised enough to scale for big ones. And when it comes to delivering out of our wheelhouse we are not afraid to introduce partners that we believe in to our clients when we know they will add value."

The kind of work that makemepulse have created over the last decade clearly illustrates the diverse nature of their design sensibilities. Antoine explains the projects that define who they are and their approach to design:

"Our last two experiential campaign websites for Ubisoft — Ghost Recon Wildlands and For Honor Scars — were both big, ballsy productions with a lot of seemingly impossible challenges and next level attention to detail and craft.

"However, we are possibly best known for AiMEN, the Papal AI we designed to troll the internet using machine learning and IBM Watson. Our ethos is to use the right technology to tell a story and these are our finest examples of extremely technically complex productions where you don't notice the tech at all."

The diverse nature of the projects makemepulse takes on means being flexible when it comes to workflow and timelines. "The bigger projects can be anything from four months to whenever we finish," says Antoine. "We work in a Squad format, flat in terms of decision making, but guided by a holy trio of senior people: Production Lead, Creative Lead, Creative Technologist. The consuming parts are not consistently one thing or another, but the clients usually come to us because they have a big technical unknown. Our reputation is one of being able to solve seemingly impossible productions."

As the content makemepulse create is diverse, the toolset they employ has evolved over the years. Emma explains their current approach: "We use the best tools for the job. Choosing Medium for our case studies is one case in point, and building our own WebGL PRB rendering engine, NanoGL, is another.

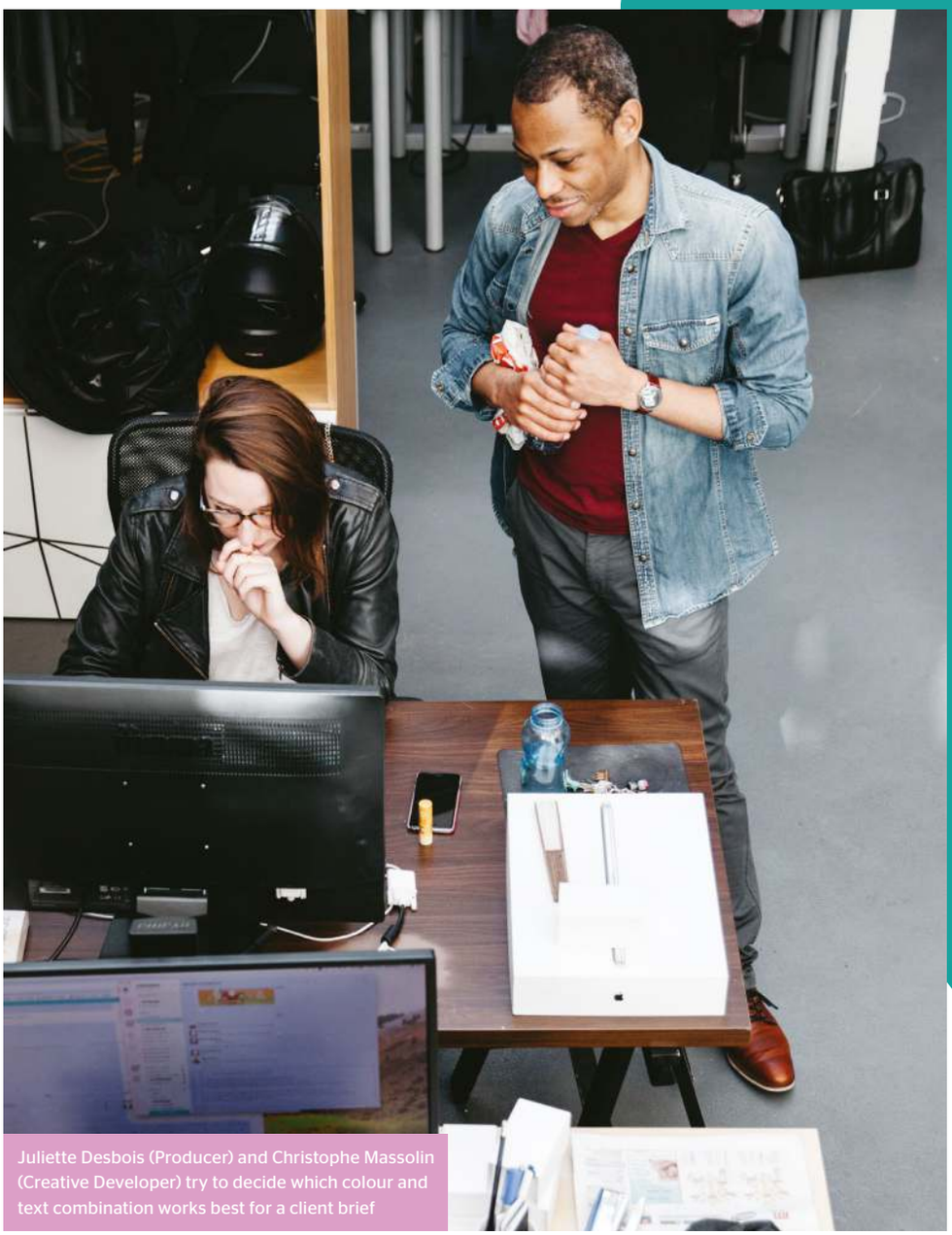
"We'd been using Three.js to make 3D for a long time, but as a very high-level library a lot of things are hidden to the developer. Sometimes if you want specific behaviours and effects, or more control over the render, it can be very tricky to get what you want and keep performance high using off-the-shelf tools, so we made our own, which we continue to evolve and maintain.

"JavaScript is definitively gaining ground in the web technologies. We love it simply because it enables a wide range of opportunities to use new technologies and create new experiences in a browser. Today we can use it for WebVR, tomorrow we will be able to do real markerless AR in a browser. We are so ready for that! Can't wait."

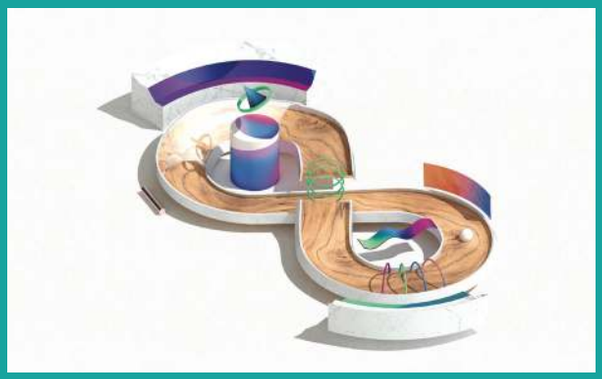
Antoine also comments: "We focus on the moment in the customer journey and crafting the right experience for that time, so our approach is less about building for a device and more about building for a person. Whatever you build for web ▷

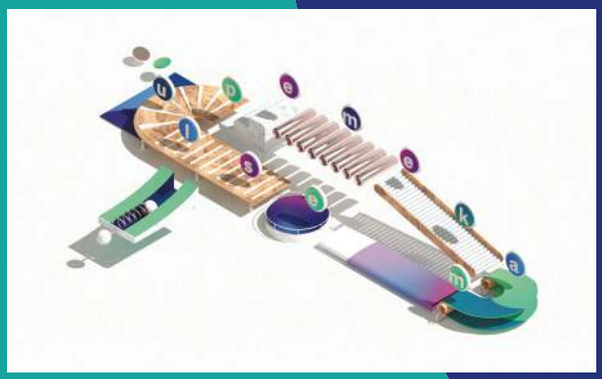
"We'd been using Three.js to make 3D for a long time, but as a very high-level library a lot of things are hidden to the developer. Sometimes if you want specific behaviours and effects, or more control over the render it can be very tricky to get what you want and keep performance high using off the shelf tools, so we made our own, which we continue to evolve and maintain"

Emma Willis, Managing Director, London



Juliette Desbois (Producer) and Christophe Massolin (Creative Developer) try to decide which colour and text combination works best for a client brief





Top *The annual in-house project challenge illustrates how makemepulse can use today's digital tools to engage an audience with astonishing techniques*

Far left *Sound and vision meld seamlessly together to deliver experiences visitors can immerse themselves within*

Left *As an exercise in animation prowess, the latest Wishes project is a masterclass*

makemepulse Wishes 2018

2018.makemepulse.com

Since 2016 — around Christmas time, we work on our 'Wishes'. The founders set the entire studio a creative challenge to solve, without the safety of the typical constraints of a client project to guide them. It's a makemepulse project and it's designed for two things: for the team to do dope shit, and to share our values and a bit of our culture with everybody else.

It started as a way to reconnect with our passion, which stems from a quote of André Malraux that "great ideas are made to be experienced." Last year we updated our logo and the company identity, which informed this year's brief: how can people experience our new identity? Brainstorming with 20 people is always a tricky thing, but after hundreds of graphic references shared, many sketches and 3D prototype rendered, we agreed on the concept and built a mini-36 Days Of Type in the form of four interactive doodles with studio, PBR, and mix material renders. Sound is always an important part of our productions, even more so in our annual Wishes. If you notice, when landing on the experience you have almost no ambient sound, but once you head to the second number you're invited to make some music. Clicking on various objects triggers an individual piece of a whole song. To make the whole sound experience harmonious, all sounds need to play at a certain tempo. This meant we needed to take into account that if the user's machine skips one or more frame per second, the timeline handling all the sounds must resynchronise everything. It's exactly this kind of conundrum that we live for. It's a bit of work, but worth it, isn't it?

The Young Pope

Last year we worked on an awesome campaign with BETC for the launch of the new Canal/HBO series *The Young Pope* starring Jude Law. The agency team had the fantastic idea to recreate his holiness Pius Thirteenth as an artificial intelligence that could spread God's word to the people via the internet.

We greatly admire these kinds of ideas, where you get to create a personalised and contextual interaction with an audience that's less advertising, more entertaining.

Their ambition was big — to monitor one million online comments and respond to as many as possible in a real-time and contextual way. Makemepulse used IBM Watson's natural language classifier, combined with machine-learning algorithms and a semantic analyser to understand the emotional context of what users were posting.

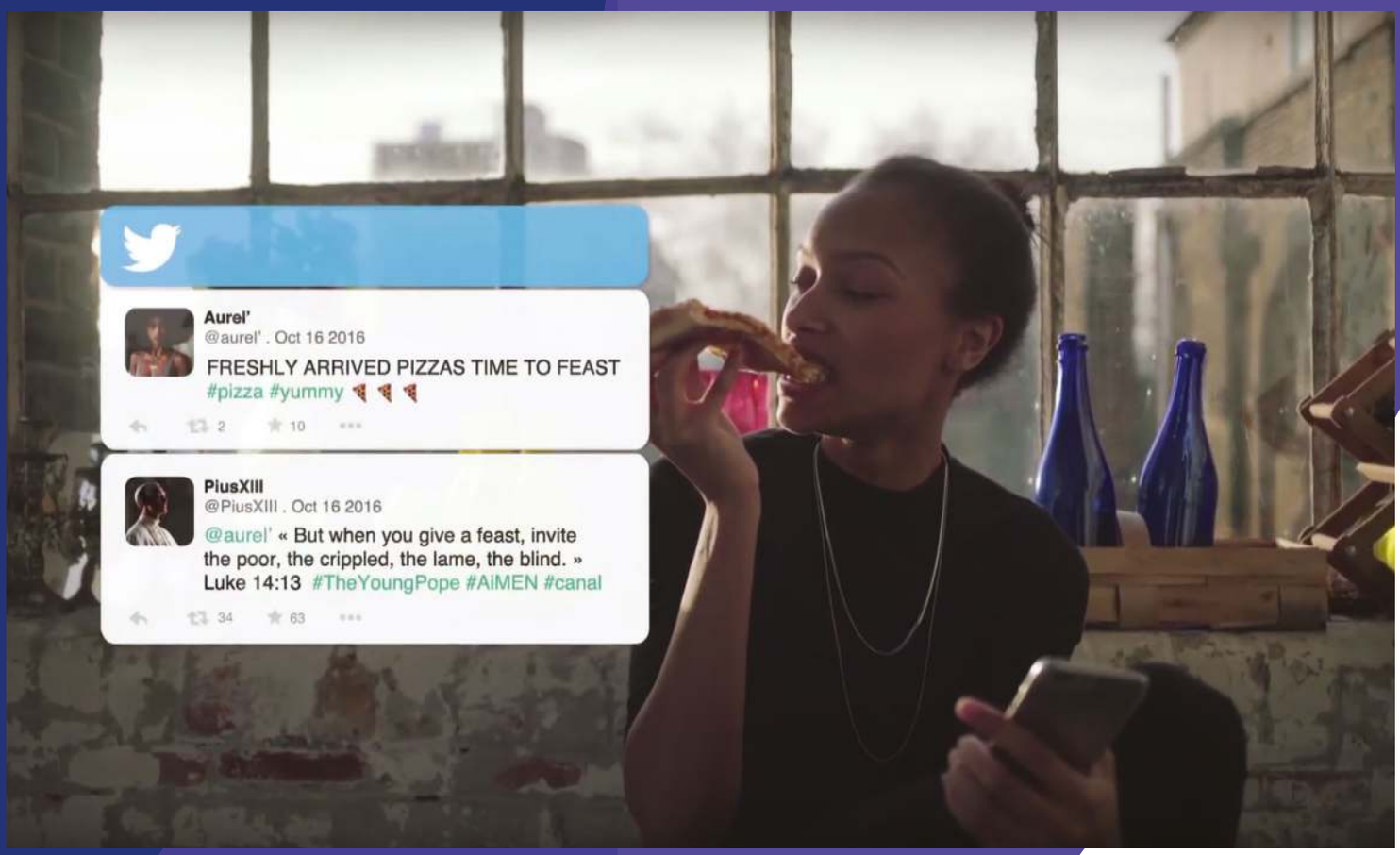
Then, drawing from a database of 39,000 Bible verses, responded to them with the most relevant thing to say. Because we love to use cutting-edge technologies, we also designed the application architecture based on the powerful serverless framework to prevent any scalability issues.

In just two weeks AiMEN analysed almost 4.5 million messages, 10GB of text data, invoked more than 15 million lambda, answered 900,000 users pulling responses from a database of 39,000 Bible verses and exposed 2.6 million people to the robot Pope.

Top *Creating an AI-based Pope was the challenge that makemepulse more than matched with their design and innovation*

Middle *For makemepulse data and design can work seamlessly together and deliver new digital experiences to everyone.*

Bottom *Connecting the processing power of IBM's Watson to their design enabled this project to personally reach millions of users*



now you need to be thinking about all of the things: market, mood, cultural usage, message, desired result, software and hardware. It's got so much bigger than mobile-first!

"We love to see how web technology is creating new tools and new frameworks every day. It's one of the reasons we created our own WebGL2 micro framework last year. It's performant, light, and works across all devices. We think it's a great example of the kind of tool that helps push the web to the next level, and it's why we always integrate new features as soon as they are available in the browser."

"Video games, architecture, movies and art all influence our design output, but not necessarily in that order. When it comes to influencing our design process, our philosophy stems from a famous quote of Andre Malraux that basically translates to: 'Great ideas are made to be experienced.' It's what motivates us to do, to try and to take action, to always be making it happen; because if you don't make it happen it will only ever be an idea and what good is that to you or anybody, really?"

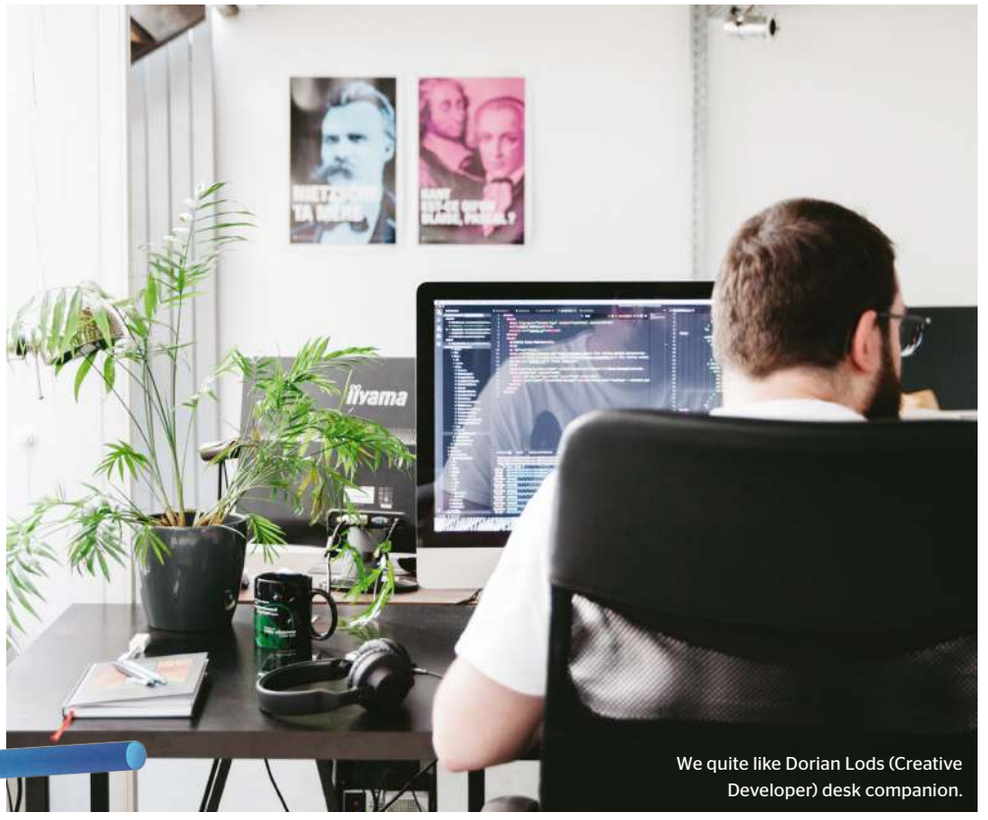
And where does makemepulse think the next innovation in digital design will come from? "It will come through voice, the founders agree. "It's a big topic in digital and in design. We have to better understand how people want to interact in the digital space with voice, but we already know that it's going to get bigger and bigger."

The current trend for one-page scrolling web pages has, for some, made the web a stale place for innovation. However, makemepulse have a different attitude, as Antoine explains: "Not at all boring, no, far from it! One-page scrollers might be the norm now, but it wasn't a few years ago. We see our job as trying to find new ways to create exciting experiences exactly because trends change all the time and we see exciting new stuff literally every day.

> You need to be thinking about all of the things: market, mood, cultural usage, message, desired result, software and hardware. It's got so much bigger than mobile-first

"In fact, this January Google announced that it is bringing AR technology to browsers to push 'Augmented Reality on the web, for everyone'. This means in the not-too-distant future people will be able to interact with 3D objects embedded into websites and place those objects into their environment straight from the browser (without launching an app). Once this functionality is available to the general public, we are anticipating entirely new levels of 3D craft and creativity on websites to support these kinds of immersive experiences. More filmic, much richer, just bigger really, in every way!"

What has made makemepulse go wow!? "The website for Kikk Festival by Dogstudio late last year ▷



We quite like Dorian Lods (Creative Developer) desk companion.

Timeline

2008
Antoine and Nicolas create makemepulse, the first production company in France with a huge focus on innovation.
Employees: 2

2011
makemepulse gets its first Cannes Lions win for the Greenpeace A New Warrior interactive fundraiser.
Employees: 6

2011
First award win for a real-time 3D experience in the browser for BMW.
Employees: 6

2013
First award win for an AR project on a mobile device for Issey Miyake.
Employees: 8

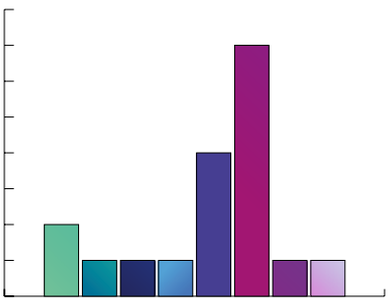
2014
First award win for a face lens project for Disneyland Paris.
Employees: 10

2017
makemepulse opens its new offices in Hoxton, North London
Employees: 18

2017
Ten Cannes Lions wins in one year.
Employees: 18

2017
Eurobest names makemepulse the best production company in Europe.
Employees: 18

Agency breakdown



- 2 Founder CEOs
- 1 General Manager
- 1 Head of Production
- 1 Head of Client Services
- 4 Producers
- 7 Creative Developers
- 1 Creative Director
- 1 Interactive Designer

Ghost Recon Wildlands

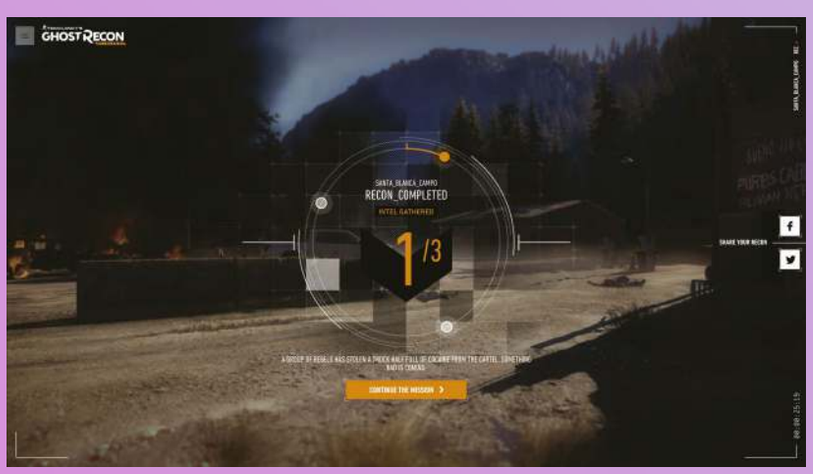
noheroes.ghostrecon.com

In February 2017 we released a campaign website to promote the launch of a new video game, Tom Clancy's Ghost Recon Wildlands. The agency, DDB° Paris, briefed us with a unique idea: they wanted to produce the first ever documentary on a virtual world using extensive game capture to reveal the lives and rituals of every character inside the World With No Heroes.

Using assets supplied by Ubisoft, first we conceived and built a 'living map' of Bolivia where the game is set and rendered it for web in real-time 3D using our in-house framework, NanoGL.

Then we developed a mini-game that put players at the heart of the documentary, where you follow the narrative of a cocaine heist gone wrong via surveillance cameras dispatched throughout the map. Players can also explore the site at their leisure through the same cameras.

This project was a behemoth and we worked on it for a whole year. Not only was it beset by issues — one involving the Bolivian government! — but there were more technical challenges than we could count: create a living map in 3D, in real-time by day and night; an autonomous world with systemic events and data visualisations? Why not! In fact, we wrote a lot about what went on behind the scenes on Medium (**https://goo.gl/1rkxtO**) for those who are interested, because it really was intense, but we learned so much. Suffice to say we were all very proud at the end of this. To see all our seemingly impossible ambitions achieved and polished when it went live.



Top *The rendering of a 'living' map challenged the team to enhance the graphics and animations they created so visitors could experience the story*

Middle *A tour de force of design that immerses the visitor in a 3D environment pushed the agency to their design limits*

Bottom *Incredible attention to detail coupled with an outstanding grasp of graphics and animation resulted in a project that showcases digital design at its best*

(**kikk.be/2017/**) is brilliant. Every time I load it I'm jealous we didn't do it," says Emma. "That Gorilla is crazy cool. In fact, we have a Slack Channel called Inspiration where the team will post awesome new stuff from around the web every day and it's hard to keep up! Around half the studio are judges for FWA too, which hugely helps when it comes to inspiration since the quality of work that's uploaded to that platform daily is pretty staggering."

Which technologies and environments to use with any given project will always depend on the demands of the client and how best to achieve their brief. "Our ethos is to use technology to tell stories and not the other way around, so we wouldn't advocate integration of any platform or tech just for the sake of using it," says Emma. "For us there has to be a reason. In AiMEN, for example, the brief was to preach God's word in a modern-day method, so the social soapbox made absolute sense. Platforms are only useful in an idea if they enable the right conversation at the right moment in the customer journey."

"As a brand, your customers aren't getting much of you if they only see six seconds of your TVC! There's been a significant number of users rejecting this kind of content recently and it's led to all the platforms, particularly Facebook and Snapchat, building back in some fantastic features for users to interact with brand content, instead of playing such a passive part in that communication. Hopefully this is an upward trend. I see the role of social in the future as a kind of infrastructure for brands wanting to act more like service providers than advertisers."

As the agency has grown over the past decade, how have they chosen the people to become part of their team? Emma explains: "For us, quality prospective candidates must show they are conscientiousness about self-improvement. Every one of our team participates in extracurricular activities related to their work, essentially because they all respect the pace of change in our industry. The more you know, the less you worry when you don't and that's an essential quality when you are faced with issues you don't know how to solve on a day-to-day basis."

Has it been difficult to recruit and does the agency believe design education is delivering the designers they want to hire? "It's a mix between training and self-teaching," Antoine continues. "You have to find a good balance. We also actively encourage and practice mentorship within the company. Younger people are mentored by senior staff and, of course, they learn faster as a result.

"We try to hire for potential and catch talent during their school years (starting with internships), then help them grow professionally to complement what they are doing with their studies. We try to systematise opportunities for feedback in our processes and approach to work too, which is also a great way to push learning and going beyond your comfort zone."

What of the future? "We have just opened the office in London, so this is a key focus for us right now," concludes Antoine. "Project-wise, it's a mixed but exciting bag of Augmented Reality, AI (chatbots, machine learning), voice and experiential campaign work. And a lot of what's on the horizon for us is about how all these kinds of experiences can exist inside a single idea, held together by a strong technical strategy. It's what we love to do."

A caption competition would read: "We really must stop watching cat videos and do some work"

"Our last two experiential campaign websites for Ubisoft — Ghost Recon Wildlands and For Honor Scars — were both big, ballsy productions with a lot of seemingly impossible challenges and next level attention to detail and craft"

Antoine Ughetto, Co-Founder, Head of Innovation

makemepulse

makemepulse.com

Founders
Nicolas Rajabaly, Antoine Ughetto

Year Founded
2008

Current Employees
18

Location
Paris, London

Services
Real time 3D, AR, VR

Technical strategy

Creative, experience design

Chatbots, AI and machine learning

Front and backend development

Free WordPress guide worth £12.99

Subscribe today and receive your free gift!

Every issue, delivered straight to your door

Never miss an issue
13 issues a year, and you'll be sure to get every single one

Delivered to your home
Free delivery of every issue, direct to your doorstep

Get the biggest savings
Get your favourite magazine for less by ordering direct

What our readers are saying about us…

"I've been a reader of Web Designer Magazine since the early days."
@sixrevisions via Twitter

"Love the latest magazine that I purchased today"
@navigation_web via Twitter

"My favourite magazine!!!"
@eduardomurillo via Twitter

Pick the subscription that's right for you

MOST FLEXIBLE

Subscribe and save 20%

✔ Automatic renewal – never miss an issue

✔ Pay by Direct Debit

Recurring payment of £34 every six months, saving 20% on the retail price

DIRECT Debit Instruction to your Bank or Building Society to pay by Direct Debit

Originator's reference 7 6 8 1 9 5

Name of bank

Address of bank

Postcode

Account Name

Sort Code

Account no

Please pay Future Publishing Ltd Direct Debits from the account detailed in this instruction subject to the safeguards assured by the Direct Debit guarantee. I understand that this instruction may remain with Future Publishing Ltd and, if so, details will be passed on electronically to my Bank/Building Society. Banks & Building Societies may not accept Direct Debit instructions for some types of account

Signature

Date

GREAT VALUE

One year subscription

✔ Great offers, available world-wide

✔ One payment, by card or cheque

A simple one-off payment ensures you never miss an issue for one full year. That's 13 issues, direct to your doorstep

☐ **UK** £69 (saving 20% on the retail price)

☐ **Europe** €81 ☐ **USA** $103

☐ **Rest of the world** $103

Pay by card or cheque

Pay by Credit or Debit card

☐ VISA Visa ☐ Mastercard ☐ AMERICAN EXPRESS Amex

Card number

Expiry date

Pay by Cheque
I enclose a cheque for £

Made payable to *Future Publishing LTD*

Signature

Date

Your information

Name

Address

Telephone number

Mobile number

Email address

Postcode

☐ Please tick if you want to receive any communications from Future and its group companies containing news, special offers and product information.

Please post this form to
Future Publishing Ltd, 3 Queensbridge, The Lakes, Northampton, NN4 7BF, United Kingdom

Order securely online to receive your free gift
myfavouritemagazines.co.uk/WBDPS18G

Speak to one of our friendly customer service team
Call 0344 848 2852

This offer will expire on
31 August 2018

*Terms and conditions: This offer entitles new UK Direct Debit subscribers to pay just £34 every 6 months plus receive a bookazine worth £12.99. Gift is only available for new UK subscribers. Gift is subject to availability. Please allow up to 60 days for the delivery of your gift. In the event of stocks being exhausted we reserve the right to replace with items of similar value. Prices and savings quoted are compared to buying full-priced print issues. You will receive 13 issues in a year. Your subscription is for the minimum term specified and will expire at the end of the current term. You can write to us or call us to cancel your subscription within 14 days of purchase. 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. . UK calls will cost the same as other standard fixed line numbers (starting 01 or 02) or are included as part of any inclusive or free minutes allowances (if offered by your phone tariff). For full terms and conditions please visit: www.bit.ly/magterms. Offer ends 31.08.18

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

The past two years has seen an explosion in the interest for VR and AR technologies, and this shows no signs of decline, with many major tech companies rumoured to be working on new top secret devices. It's also no surprise that AR has arrived on the web and this gives designers some important new areas to consider for content creation. With new technology, of course, comes new skills and right now it feels like the Wild West with no major standards to follow. The ability to display 3D on the web is nothing new, but if you've been avoiding it, then you need to jump into technologies like Three.js or A-Frame. The latter being the easiest if you are not familiar with JavaScript. You might also need to think about using a 3D app in much the same way you think of creating graphic content with a photo app, so upping those skills is going to be an advantage giving you the ability to craft your own models.

Whatever skill level you are, it won't make much difference if you don't have some decent content. Think about appropriate use cases for AR before jumping in. Because AR links physical space to the web, events can be enhanced. Further on, you'll read an article about creating a multi-marker AR experience, so that is good for anything that follows a step-by-step process. Perhaps the world needs an AR cooking app, for that matter any instruction or education content will work nicely as it can easily come alive with animation and has the advantage of being able to be viewed from any angle. How many arguments will be pacified with AR Ikea instructions animating the pieces into place?

You can of course tie AR to the geolocation API and detect if the user is near a certain physical place, such as a theme park and offer the user wayfinding options. By holding up their phone the user could get the direction to certain attractions, just by visiting the park's website.

All of these possibilities means the way we think about the web becomes more than just information on a screen but more about linking the physical world with the web. This offers some challenges but if you consider how major chain stores could utilise this, everyone could very easily have a personal shopper on their smartphone.

The web still has a way to go before being able to do everything that native apps can do, however, that is being overcome by Google's experimental browsers (WebARonARCore for Android and WebARonARKit for iOS). These browsers open the native ability to see ground planes through the camera and so markers are not needed for placing the content. Getting on board with these now, means you'll be ahead of where the web currently is at.

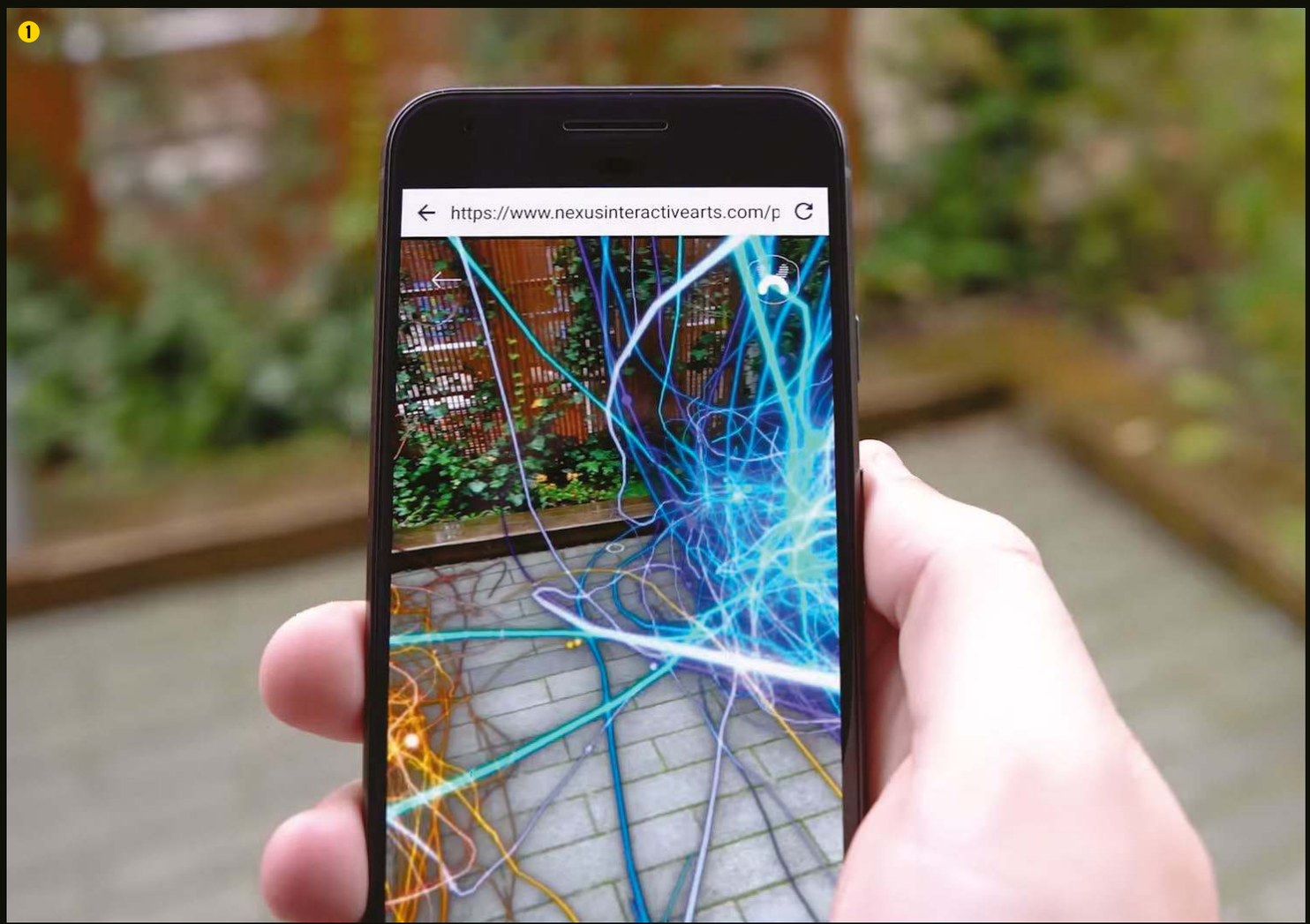
With all the hype surrounding AR for the web it's worth remembering that we have been here before. Back when Flash was big, it had access to the camera and could do simple 3D, so AR markers became something that was used. What makes it different this time around is that it runs well on mobile devices, WebGL has been around for quite some time and it has good support across different browsers and handsets. Plus, there is now dedicated hardware support with GPUs on phones and the ability to see ground planes on newer devices. The latter will take a little while to filter down to a wide scale user base, but all of this shows a road map ahead that will only get better. We may well look back on this period as a paradigm shift for the web, with it becoming more about augmenting our physical location as social, shopping and creative spaces enter our surroundings and web browsing is more about showing appropriate information as it becomes relevant.

MARK SHUFFLEBOTTOM
Professor of Interaction Design
www.webspaceinvader.com

BE INSPIRED

How is Augmented Reality being used today? Check out these exciting examples to give you insight and inspiration for your own projects

1



3

4

1. SUPER CHARGE

https://bit.ly/2xqABtI

Running on the prototype browsers WebARonARCore and WebARonARKit this is a great experiment to show the visual capabilities, giving you a super power to unleash a ball of energy in your surroundings.

2. WORLD HERITAGE SUNKEN VESSEL

https://bit.ly/2kyEbZ7

Built for the UNESCO World Heritage Site 'Prehistoric Stilt Buildings around the Alps', this gave visitors to the Vienna Ball of Science 2018, an insight into cultural heritage by revealing a sunken vessel.

3. ENARGY

https://bit.ly/2LGbCFo

This experimental project allows the phone's camera to be pointed at an appliance to see how much energy it was using by hooking up to data from smart plugs that power the appliance. This was built using A-Frame, showing the sophistication that can be achieved with the library.

4. A-FRAME CHESS

https://bit.ly/2xt9Cxg

Using AR.js for A-Frame, this is a nice demo of games creation. Even if this isn't fully functional at the moment, it wouldn't take a massive leap forward to make this into something that could be a lot of fun.

5. ARTE

https://bit.ly/2IYqvkM

Another great piece of work from Nexus Interactive Studios, this time it uses two AR markers – one to show 3D scans from The Louvre and the second to bring up the interface for browsing.

MAKE A MULTI-MARKER AR APPLICATION WITH AR.JS

By using multiple markers, it's possible to show different stages of a process or any unique content based on that marker

1. GETTING STARTED

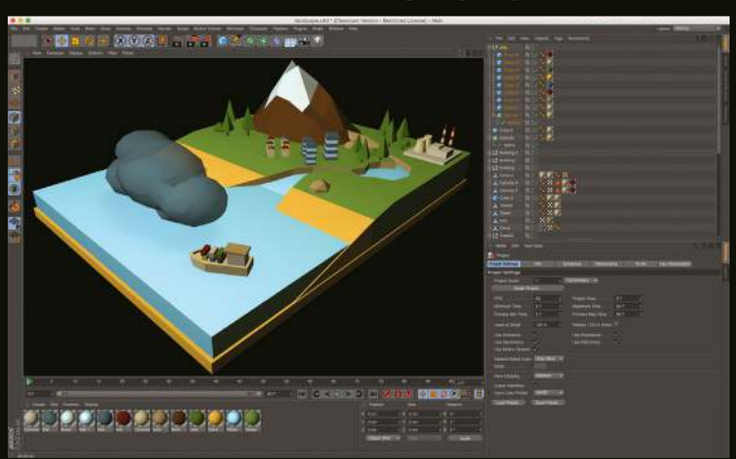
Open the 'start' folder in your IDE and inside the 'index.html' page find the script tags. All the code in the tutorial will go inside these. To test the app you will need to have a server and if you want to test on a phone you will need to host the files on a 'https' server. Add the initial variables in the script tags:

```
var model1, model2, model3,
count = 0,
  particles, particleCount,
particleSystem;
  var loader = new THREE.
ColladaLoader();
```

2. LOAD THE MODEL

To make the AR scene work, a model will be loaded. You will see that once loaded it is stored in the variable 'model1'. This is then scaled and cloned twice for the three steps. Rather than load three different models, all the adjustments to one model will be done in code to make it load quickly on mobile.

```
loader.load('landscape1.dae',
function(collada) {
  model1 = collada.scene;
  model1.scale.x = model1.
scale.y = model1.scale.z =
0.015;
  model2 = model1.clone();
  model3 = model1.clone();
```



3. SETTING UP THE TWEENING

On the first model the cloud is going to be found in the scene and this will be tweened to a new position so that the cloud rises out of the sea. This is set to repeat forever and it will take 8 seconds for the tween to animate up and show a cloud forming.

```
var cloud1 = model1.
getObjectByName("Cloud",
```

> RATHER THAN LOAD THREE DIFFERENT MODELS, ALL THE ADJUSTMENTS TO ONE MODEL WILL BE DONE IN CODE TO MAKE IT LOAD QUICKLY ON MOBILE

CREATE A CUSTOM MARKER

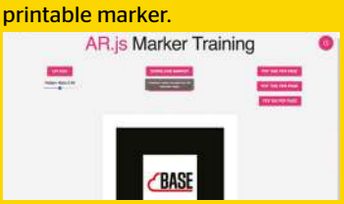
When creating your own AR experience, you will probably want the marker that you are using to either be branded to a client logo or customised in some way to reflect the nature of what your user is doing with the AR application. This is very straightforward and easy to do. The main part to make this work is downloading the pattern file, placing this within your project folder structure and referencing this in your JavaScript code.

1. MAKE THE PATTERN

In your image editing application make a square image and ensure the background is set to white. Now create a custom shape on the screen. Make sure this is black as this is the easiest to be read. Save the file as a PNG as this will be uploaded to a website to convert into the pattern code.

2. UPLOAD AND DOWNLOAD

Go to the pattern marker training page **https://jeromeetienne.github.io/AR.js/three.js/examples/marker-training/examples/generator.html** and click the upload button. Your marker will appear on the screen. Click the download marker button to get the marker file for your code. Click the appropriate download PDF button to get a printable marker.



3. PLACE IT IN THE CODE

Place the downloaded 'pattern-marker.patt' in the data folder. You can name it anything you like. Look in your code and change the line shown below to be the name of the pattern that you have downloaded. Now hold the PDF up to the camera to trigger the AR content to appear.

```
THREEx.ArMarkerControls(arToo
lkitContext, markerRoot, {
type: 'pattern',
patternUrl: THREEx.
ArToolkitContext.baseURL +
'data/pattern-marker.patt',
});
*********
```

```
true);
  cloud1 = cloud1.children[0];
  new TWEEN.Tween(cloud1.
position).to({
  x: 0,
  y: 30,
  z: -15
  }, 8000).repeat(Infinity).
easing(TWEEN.Easing.Quadratic.
InOut).start();
```

4. GROWING UP

The cloud is scaled down to be almost invisible. Another tween is added to the cloud and this scales the cloud up to its normal size. With the movement and the scaling, it will give the illusion that the cloud is rising and forming out of the sea as the first step in the process of the water cycle.

```
cloud1.scale.x = cloud1.
scale.y = cloud1.scale.z =
0.0;
  new TWEEN.Tween(cloud1.
scale).to({
  x: 1,
  y: 1,
  z: 1
  }, 8000).repeat(Infinity).
easing(TWEEN.Easing.Quadratic.
InOut).start();
```

5. SET UP THE SECOND CLOUD

The next cloud from the second model needs to be positioned where the first cloud finished its animation as a formed cloud in the sky. This is given a tweened movement to position itself over the land, rising slightly above the mountain. This will take 12 seconds to animate as it's a bigger move.

```
var cloud2 = model2.
getObjectByName("Cloud",
true);
  cloud2 = cloud2.children[0];
  cloud2.position.set(0, 30,
-15);
  new TWEEN.Tween(cloud2.
position).to({
  x: 0,
  y: 50,
  z: -145
  }, 12000).repeat(Infinity).
easing(TWEEN.Easing.Quadratic.
InOut).start();
```

6. MAKING IT RAIN

The key to making this illusion work is allowing the cloud to rain. The water cycle has the cloud rain as it moves higher over land. To get the effect, a particle system will be used. Here the amount of particles and the particle material is created, using a rain drop image.

```
var textureLoader = new
THREE.TextureLoader();
particleCount = 1500;
particles = new THREE.
Geometry();
var pMaterial = new THREE.
PointsMaterial({
  color: 0x3a4e5d,
  size: 0.05,
  map: textureLoader.
load("img/rain.png"),
  phaTest: 0.3,
  opacity: 0.9,
  transparent: true });
```

7. MAKING RAIN DROPS

Using a for loop, 1500 rain drops can be created with a random x, y, z position that will be between the cloud and the ground. Each rain drop is given its own random velocity to make the rain look more natural. The particle is pushed into the correct vertice of the geometry.

```
for (var i = 0; i <
particleCount; i++) {
  var pX = Math.random() * 60
- 30,
  pY = Math.random() * -10,
  pZ = Math.random() * 20 -
10;
  var particle = new THREE.
Vector3(pX, pY, pZ);
  particle.velocity = new
THREE.Vector3(0, -(Math.
random() * 0.9), 0);
  particles.vertices.
push(particle); }
```

8. RAIN SYSTEM

Now the particle system is created out of the geometry and the material. The particles are set to be sorted so that the z-order is correct and then the rain particles are made a child of the second cloud. Anywhere the cloud is tweened the rain also follows, so no need to animate the rain following the cloud!

```
particleSystem = new THREE.
Points(particles, pMaterial);
particleSystem.sortParticles =
true;
cloud2.add(particleSystem);
```

9. FINAL MODEL POSITIONS

In the final model the cloud is repositioned to the ending spot of the second cloud animation cycle. This new cloud is just going to sit in the sky and not animate. Instead the river is going to animate, so the river model is stored in a variable, ready to add the tween to it.

```
var cloud3 = model3.
getObjectByName("Cloud",
true);
  cloud3 = cloud3.children[0];
  cloud3.position.set(0, 50,
-145);
  var river = model3.
getObjectByName("river",
true);
  river = river.children[0];
```

10. RIVER RISING

In the third step of the water cycle the water runs off the hills, filling

FREE 3D APPS TO ADD CONTENT TO AR

SEA3D

http://sunag.github.io/sea3d

This is a completely free open source application that runs in the web browser or can be downloaded for Windows only. This has the wonderful Three.JS preview built right into it, so that you can explore your models and fine tune them for the web before exporting and bringing into your AR experience.

BLENDER

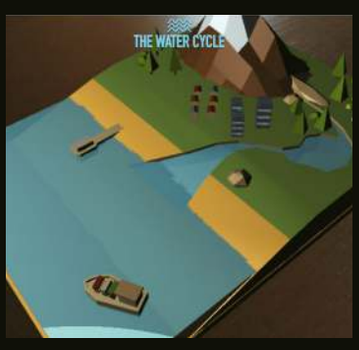
www.blender.org

The big thing going for Blender has always been that it is free and extremely powerful, that said it has a huge learning curve for people starting off. It does work very nicely with Three.JS as it can export models and scenes in raw JavaScript ready for loading right into your AR scenes, so some people use this just to convert models.

SCULPTRIS

http://pixologic.com/sculptris

A very interesting wildcard here is the free 3D app Sculptris from the makers of Z-Brush. This is a package that allows the height of objects to be painted up or sculpted, hence the name. It can export in OBJ, which isn't optimised for the web but can be easily loaded and displayed within Three.JS.



> THE INIT FUNCTION IS ABOUT SETTING UP THE THREE.JS SCENE AND THEN MAKING THAT SCENE RESPOND TO THE POSITION OF THE MARKER DETECTED IN AR

rivers and lakes as it returns to the sea. This is the most subtle movement as it will just entail moving the height of the river so that it appears to fill up. Everything is preloaded now, so the 'init' function is called.

```
  new TWEEN.Tween(river.
position).to({
  y: 3
  }, 8000).repeat(Infinity).
easing(TWEEN.Easing.Quadratic.
InOut).start();
  init();
});
```

11. INITIALISE THE AR SCENE

The init function is mainly about setting up the Three.JS scene and then making that scene respond to the position of the marker detected in AR. In this case the renderer is set up and the various attributes set. It's positioned in the top left of the browser screen.

```
function init() {
  var renderer = new THREE.
WebGLRenderer({
  antialias: true,
  alpha: true
  });
  renderer.setClearColor(new
THREE.Color('lightgrey'), 0);
  renderer.setSize(640, 480);
  renderer.domElement.style.
position = 'absolute';
  renderer.domElement.style.
top = '0px';
  renderer.domElement.style.
left = '0px';
```

12. ADDING THE CAMERA

The renderer is appended into the HTML document and then a new array is created that will contain all the elements that need to be updated in the scene for animation – more on this later. A 3D scene is created and a camera added to the scene which will be from the position of the user's camera.

```
document.body.
appendChild(renderer.
domElement);
var onRenderFcts = [];
var scene = new THREE.
Scene();
var camera = new THREE.
Camera();
scene.add(camera);
```

13. AR FUNCTIONALITY

The source to track for AR markers is set up now and you can see in this code that it is set to track the webcam. This will be a phone or tablet's camera if on those devices. If for any reason the window resizes, the 'onResize' function is called and this is also called at the start.

```
var arToolkitSource = new
THREEx.ArToolkitSource({
  sourceType: 'webcam' });
arToolkitSource.init(function
onReady() {
  onResize() });
window.
addEventListener('resize',
function() {
  onResize() });
```

14. RESIZING THE SCREEN

Now the code creates the 'onResize' function that is called from the previous step. This ensures the webcam image is set to be resized to fit inside the renderer. If you inspect your page the renderer is actually a HTML5 Canvas element.

```
function onResize() {
  arToolkitSource.onResize()
  arToolkitSource.
copySizeTo(renderer.
domElement)
  if (arToolkitContext.
arController !== null) {
  arToolkitSource.
copySizeTo(arToolkitContext.
arController.canvas)
  } }
```

15. ADDING THE FIRST MARKER

The next few steps all take a similar approach. A new variable is created and this becomes a group. Inside this group the marker is told to respond to a 'pattern' marker and the type of pattern is defined in an external file. For more information on how this was created see the separate tutorial.

```
var markerRoot = new THREE.
Group;
scene.add(markerRoot);
var artoolkitMarker = new
THREEx.ArMarkerControls(arTool
kitContext, markerRoot, {
  type: 'pattern',
  patternUrl: THREEx.
ArToolkitContext.baseURL +
'data/pattern-1.patt',    });
```

CONNECTING A GRAD SHOW TO THE WEB

Students on the Bachelor of Interaction Design degree at Sheridan College used AR to power their live grad show

After four years of studying Interaction Design, students wanted to show off their work to visitors to their grad show in a unique and interesting way that reflected their discipline. Lucas Di Monte came up with a way to do this, commenting: "The challenge was to allow guests at the grad show to survey a large number of thesis projects quickly and easily, while also facilitating conversations between graduates and guests. The solution was to use AR to allow guests to select postcards that can be used at kiosks to instantly get an overview of a graduate's work. After the show the postcards would work with the website to show the graduates work when holding up the AR marker".

This is an ingenious way of using AR to allow guests to take markers on a postcard. They could talk to the students then take the postcard away to revisit later. Lucas calculated that the AR markers are reduced down to a 16 by 16 pixel grid, so by designing for this grid he was able to create 48 unique markers that would bring the right work up.

16. SECOND MARKER

Now the second marker is created with a different variable name. This references a different pattern so it will respond to a different marker held in front of the camera. This way holding up different markers will trigger different responses.

```
var markerRoot2 = new THREE.
Group;
scene.add(markerRoot2);
var artoolkitMarker = new
THREEx.ArMarkerControls(arTool
kitContext, markerRoot2, {
  type: 'pattern',
  patternUrl: THREEx.
ArToolkitContext.baseURL +
'data/pattern-2.patt',
});
```

17. FINAL MARKER AND MODELS

Once more another variable is declared for the last marker and it references another pattern file. Now after this you will see that each marker variable group gets the correct model added to it that we set up in the first ten steps. These will display when the marker is placed in front of the camera.

```
var markerRoot3 = new THREE.
Group;
scene.add(markerRoot3);
var artoolkitMarker = new
THREEx.ArMarkerControls(arTool
kitContext, markerRoot3, {
  type: 'pattern',
  patternUrl: THREEx.
ArToolkitContext.baseURL +
'data/pattern-3.patt',
  });
markerRoot.add(model1);
markerRoot2.add(model2);
markerRoot3.add(model3);
```

18. UPDATING THE PARTICLES

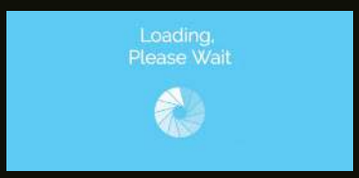
You may have forgotten about the rain particles created earlier, but they need updating every frame. So here a for loop moves through each particle in the array and updates its position, resetting it if it moves below the ground. This totally makes the rain effect work.

```
onRenderFcts.push(function()
{
  for (var i = 0; i <
particleCount; i++) {
  var ptcl = particles.
vertices[i];
  if (ptcl.y < -100) {
  ptcl.y = -10;
  ptcl.velocity.y = -(Math.
random() * 0.9);  }
  ptcl.velocity.y -= Math.
random() * 0.02;
  ptcl.add(ptcl.velocity);  }
  particles.
verticesNeedUpdate = true;
  });
```

19. REMOVING THE PRELOADER

While all of this set up has been going on, there has been a message over the screen that is asking the user to wait patiently for the content to load. Here that content is removed and the scene is rendered through the camera.

```
getElementById("preloader").
style.visibility = 'hidden';
onRenderFcts.push(function()
{
  renderer.render(scene,
camera); })
var lastTimeMsec = null;
```



20. CONTINUOUSLY UPDATING

The requestAnimationFrame is the browser's built-in loop that tries to run as close to 60 frames per second that it can. Here the frame rate is worked out so that the animation is divided by 60 frames per second to work out the interval, to create a delta of time that has passed. Mobile CPUs will run at a different speed so frame rate will not be an accurate time counter.

```
requestAnimationFrame
(function animate(nowMsec) {
requestAnimationFrame
(animate);
  lastTimeMsec = lastTimeMsec
|| nowMsec - 1000 / 60;
  var deltaMsec = Math.
min(200, nowMsec -
lastTimeMsec);
```

21. FINAL STEP

Now the tween engine is updated and every element added to the 'onRenderFct' array that will be updated at the correct speed. Save the file and make sure you look at this on a server, it must be 'https' if you want to serve to mobile devices. Place the markers supplied in the project files folder in front of the camera to see the different stages.

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

WHAT TO WATCH OUT FOR IN AR

The web is an ever-changing set of standards and there are always new things on the horizon that can suddenly change your whole workflow. Here we examine some of them relating to AR.

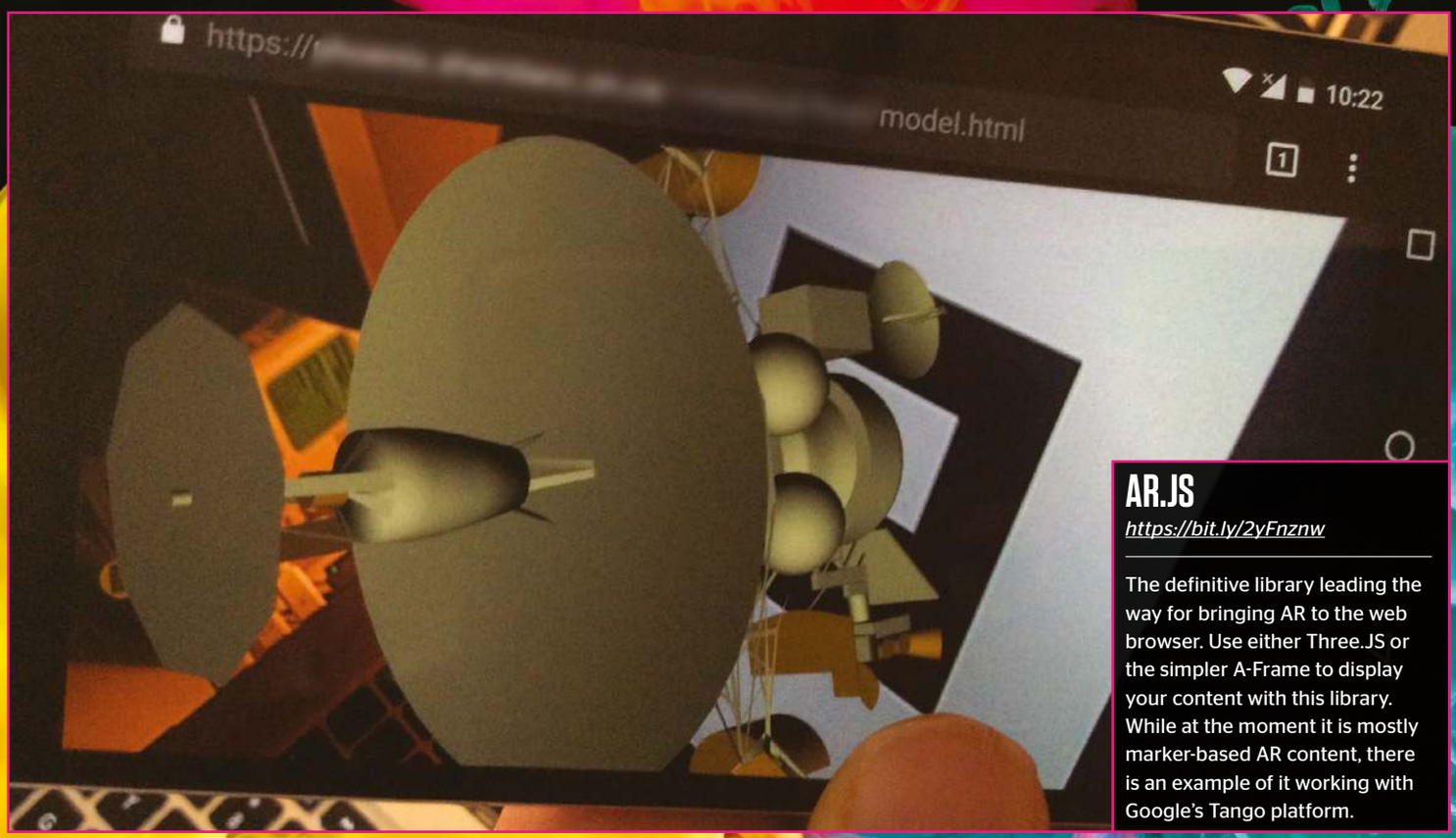
A relatively new model format for WebGL (the renderer of AR applications on the web) is glTF (GL Transmission Format). This is described as the "JPEG of 3D" as it reduces file size, making it perfect for the web. At present there are not many 3D apps that can export this format but there are plenty of conversion tools available **(https://github.com/KhronosGroup/glTF#converters)** for you to run models through.

You can also expect AR creation to become ever easier, and some of that might be tied to paid services. One such service is Amazon Sumerian, which claims to be the fastest way to create VR and AR, though we could only find support for WebVR scenes, so presumably AR is coming in the not too distant future. This is created through a standalone app to edit content. There is JavaScript support and the finished project is published to its own unique URL.

Chrome 68 will have support for Google's WebXR API, and this gives access to native phone capabilities that support both VR and AR. XR stands for Extended Reality and has become the catchall for combinations of VR and AR. We expect widescale roll-out in Chrome 68.

5 TOOLS TO BUILD AR

These are the essential tools that you need to get your hands on right now



AR.JS

https://bit.ly/2yFnznw

The definitive library leading the way for bringing AR to the web browser. Use either Three.JS or the simpler A-Frame to display your content with this library. While at the moment it is mostly marker-based AR content, there is an example of it working with Google's Tango platform.

GOOGLE AR

https://bit.ly/2LHMkqz

Google have developed two browsers for AR (WebARonARCore for Android and WebARonARKit for iOS). If your phone can support these browsers you can install them. Using Three.JS build 3D AR content using a phone's abilities to see flat surfaces and position your 3D content on them.

JEELIZ FACE FILTER

https://bit.ly/2LJN66o

This is still technically AR but focuses purely on detecting faces and augmenting 3D model content on top of the detected faces. Think of all those filters you see in social media applications that put animal ears and noses on selfies and this gives you a good idea of what you can expect to do with this library.

AWE.JS

https://bit.ly/2kAEOS3

Awe.JS has some sophisticated features that make this worth looking into. Notably the location based AR, however, this can be replicated with the Geolocation API with any other library. It isn't quite as well supported as other libraries as there isn't quite the same community around this.

ARGON.JS

www.argonjs.io

Argon has its own browser – the Argon4 web browser – but it also runs on browsers that are able to run AR natively. This makes a good fallback if you are targeting a demographic that might not have a blistering-fast phone, but you really want to be inclusive with your technical solution.

The number one destination for **web design** news, views and how-tos.

Get Creative Bloq **direct to your inbox** with our weekly web design newsletter

CB CREATIVE BLOQ

Graphic design | **Web design** | 3D | Digital art

www.creativebloq.com

Create an animated text wheel element

Inspired by www.this.design

Create an animated text wheel element

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

EXPERT ADVICE
Configuration
The best results in content design are achieved with design that's easy to make sense of and grabs attention. Avoid making the text spin too fast, as this will make it more difficult for some people to read. At the other extreme, making the text spin too slowly makes it difficult to grab attention.

<comment>
What our experts think of the site

Vector graphics: SVG

Modern web browsers support more than just the HTML content format – SVG being one of these. You may think of SVG as HTML because of its similarities, but it's an entirely different standard designed to define vector graphics. It looks similar to HTML because both are based on the XML standard.

Leon Brown, developer and author of e-learning content at nextpoint.co.uk

Technique

1. Initiate HTML document

The first step is to initiate the HTML document. This consists of the HTML container, which is responsible for storing the head and body sections. While the primary purpose of the head section is to load the external CSS stylesheet, the body section stores content elements created in step 2.

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

2. Content elements

Content uses a div element as a container for the text icon and an SVG element. HTML only draws text in straight lines, so an SVG element is used instead. Take note of how the text path uses href to associate itself with the 'path' element using the same ID. The path uses the 'd' attribute to define a circle.

```
<div class="spin">
<span>=</span>
<svg viewBox="0 0 110 110">
<path id="curve" fill="transparent"
d="M0,50a50,50 0 1,0 100,0a50,50 0 1,0
-100,0" />
<text>
<textPath xlink:href="#curve">
Some text put into a circle...
</textPath>
</text>
</svg>
</div>
```

3. CSS spin container

The spin class applied to the main container is used as a reference point to control the size and positioning of its children. Its relative positioning ensures that child elements are positioned relative to where the spin container is located. Its size is set to 7 characters wide and high.

```
.spin{
position: relative;
display: block;
width:7em;
height:7em;
}
```

4. First child

The first child inside the spin container is positioned in the centre-ish of the container – with positioning taking into account the character size. The font is set to be double the size of the default page font.

```
.spin > *:nth-child(1){
position: absolute;
top: 30%;
left: 40%;
font-size: 2em;
}
```

5. Spin SVG element

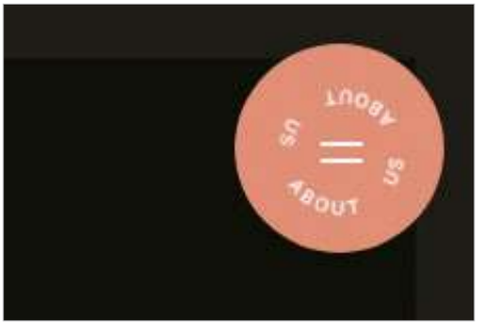
The SVG element inside the spin container is set to have an animation applied to it – also called 'spin'. This animation is set to loop infinitely, with each iteration lasting for five seconds. A font size is also applied to affect the SVG text display.

```
.spin > svg{
display: block;
animation: spin 5s forwards infinite;
animation-timing-function: linear;
font-size: 1.2em;
}
```

6. Spin animation

The spin animation referenced in step 5 is defined as a keyframes animation. Properties of the first and last frames in the animation are defined – animating the SVG from 0 to 360 degrees. The browser automatically calculates each required frame between these to key points of the animation.

```
@keyframes spin {
from { transform: rotate(0deg); }
to { transform: rotate(360deg); }
}
```



Get started with WebGL and Three.js

In this tutorial learn about WebGL 3D for websites and applications, and how to create your first animated scene

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

WebGL enables powerful programming, tapping into the graphics pipeline, for highly optimised and interactive experiences. Massive interest in 3D, WebVR/AR and mixed reality experiences, means high demand for web 3D skills. Broad browser and device support for WebGL makes it a viable approach for real-time rendering. No plugins are required and you can start learning these technologies right away.

Amazing real-time 3D graphics, powerful effects via shaders and immersive video are all possible. You can even create complex models with high level of detail, reflections, environment maps and shadows. Users now have instant access to beautiful experiences on their desktops or right in the palm of their hand.

Real-time rendering offers significant advantages over pre-rendered scenes when it comes to web design. Sites, games and apps can allow users to direct the action, move the camera and objects, navigate 3D world, and much, much more.

To get started you are going to create your first scene, add an object and get it moving. Other than having a JavaScript background, you can dive into this tutorial with no prior knowledge and get some great results. The goal is to demystify 3D web programming and get you inspired. We want to get you creating your own 3D experiences as soon as possible.

1. Create a file structure

To get started, it's a good practice to set up separate files for your HTML, CSS and JS. Create a CSS folder and a scripts folder, and create empty files for your style.css and main.js respectively in those two folders. It'll keep your HTML code clean and organised.

```
tutorial01
    index.html

    scripts
        main.js

    css
        styles.css

    libs
        three.min.js
```

2. Create a basic HTML file

To begin, you need to set up a basic HTML file and include your CSS and JavaScript files. You don't need to add a <canvas> element yet unless you prefer to. Three.Js's renderer class will create a DOM element for you. Add the following code to your index.html file.

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css"
href="css/style.css">
</head>
<body>
  <script src="scripts/main.js"></script>
</body> </html>
```

3. Include the Three.js library

Include a link to the Three.js library, either hosted externally or downloaded from the Three.js repository. You can find the library and minified JavaScript here: **https://github.com/mrdoob/Three.js**. Please note: the code in this tutorial has been tested on the latest release of Three.js – v91.

```
<script src="libs/three.min.js"></script>
```

4. Add basic styles in CSS

In your style.css file or in the head of your HTML file, set up a couple of basic styles. There are simple style rules to keep your canvas full screen, removing any margins or padding. We'll handle sizing of the renderer and handling window resize events later on.

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

5. Setting up your JavaScript

Create and open up your main.js file for coding. Add a few simple variables to keep things organised for global access when you need them later on. The first four variables are for our camera settings, and the last one is an object to hold our 3D cube we will be making.

```
// set up global vars
var fieldOfView= 45;
var aspect = window.innerWidth / window.
innerHeight;
var nearClippingPlane = .1;
var farClippingPlane = 1000;
var cube;
```

6. Create a 3D scene object

You're going to add a basic scene, which will be the container for your objects. The scene is the stage that will render with the camera. All 3D presentations will have a scene or stage of some form. What is in that stage and in view of the camera is what the user will see. Add the following code to add a scene:

```
// create a scene object
var scene = new THREE.Scene();
```

7. Add a perspective camera

Next, you need to add a virtual camera. The first attribute is the field of view of the camera. The second is the aspect ratio (width/height). Then you indicate the near clipping plane and the far clipping plane distances, which defines what is to be visible to the camera.

```
//create a camera
var camera = new THREE.PerspectiveCamera(
fieldOfView, aspect, nearClippingPlane,
farClippingPlane );
```

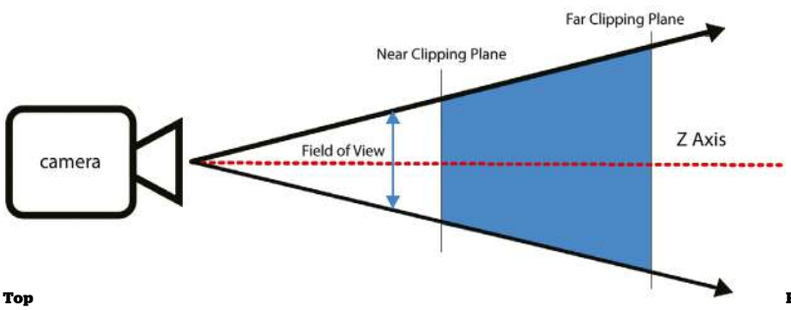
8. Add a renderer

The renderer handles the drawing of the objects in your scene that are visible to the camera you defined. You can set options via a JSON formatted set of values. Here we are setting the 'antialias' property to true, to get smooth edges on our object. You can also define the size of the draw area, as we have here, to full screen.

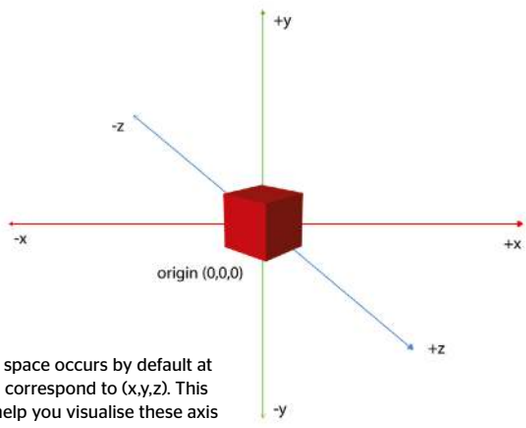
```
// create a renderer
var renderer = new THREE.WebGLRenderer({anti
```

Anatomy of a 3D scene

If you're just getting into 3D libraries and programming, understanding the terminology will make it much easier. 3D experiences are typically comprised of a common set of elements including: scenes, cameras, renderers, lights and objects.



Top
Setting up the camera allows you to define aspect, field of view and culling. This diagram helps illustrate these elements



Right
Adding objects in 3D space occurs by default at (0,0,0), where values correspond to (x,y,z). This handy diagram can help you visualise these axis

```
alias:true});
renderer.setSize( window.innerWidth, window.
innerHeight );
```

9. Add canvas element to HTML body

The renderer creates a domElement that is actually a HTML canvas element that you can then append to the body. You can optionally specify an existing canvas element to draw to if you prefer, via the 'canvas' attribute of the renderer.

```
document.body.appendChild( renderer.
domElement );
```

10. Create object geometry

Three.js includes lots of great primitive geometries to choose from including sphere, torus and cylinders. Try starting with a simple cube. The first three attributes are width, height and depth. Optionally, you can also define the segments of each of those sides of the cube using the next three attributes.

```
// create a 3D object
var geometry = new THREE.BoxGeometry( 4, 2,
.1 );
```

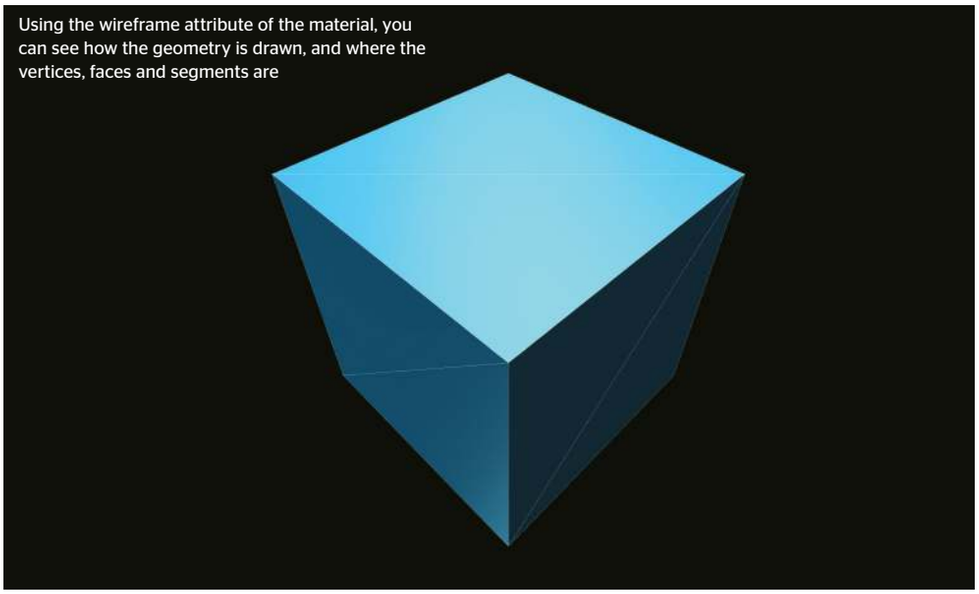
11. Create a material

Next, a material is required. Three.js includes a range of materials including physical shaders, lambert, phong and custom shaders. You can set textures such using video or images as well. Use a MeshNormalMaterial for now. This way you can see the object without needing to light it. We'll get into lighting and materials in the next tutorial.

```
var material = new THREE.
MeshNormalMaterial();
```

Defining 3D objects

3D objects are comprised of geometry and materials into a mesh. Geometry defines the shape through vertices, faces etc. Materials are the 'skin' that textures the geometry. A mesh is the result of combining geometry and materials together.



Using the wireframe attribute of the material, you can see how the geometry is drawn, and where the vertices, faces and segments are

12. Create a mesh

Next, you create a mesh, which is a combination of the geometry and material you just defined. Physical objects in 3D require a geometry that defines the faces, vertices and drawing of the shape. They also require a material or skin to cover that object so we can see it. Create the mesh object like this:

```
var cube = new THREE.Mesh( geometry,
material );
```

13. Add it to the scene

By default, when 'scene.add()' is called to add an object, it is placed at (0,0,0). To avoid the camera and the object being in the same location, we can move the camera back a little, (z= 5). You could also move your objects and leave the camera at 0. Use the following code, to add your cube:

```
scene.add( cube );
// move camera back
camera.position.z=5;
```

14. Use WireFrame to see segments

Segments are the number of triangles or faces used to draw the shape. The more segments the smoother the object. Flat sides of a cube like this can get away with the default of 1 or 2. To test out the wireframe option and see the segments, use this line of code instead of the previous one for defining material:

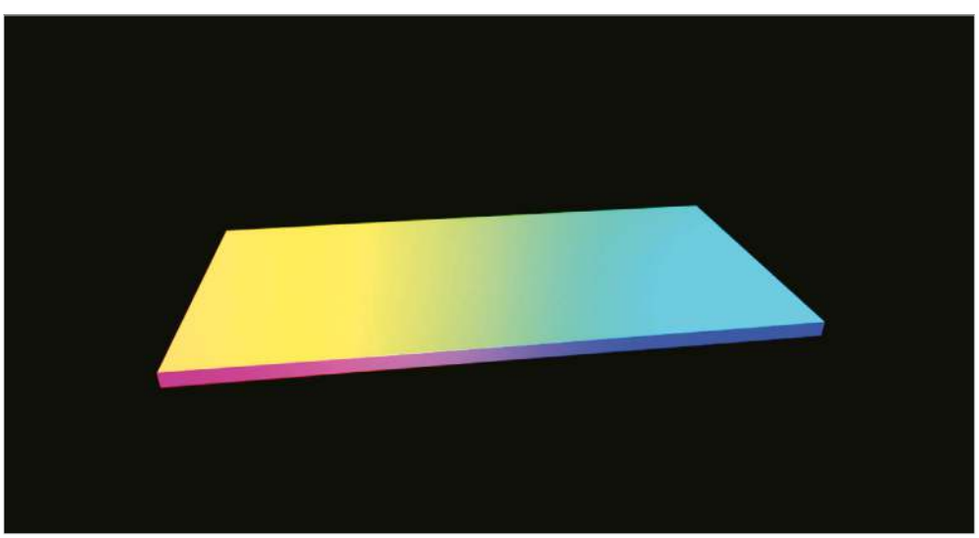
```
var material = new THREE.MeshNormalMaterial(
{wireframe:true});
```

15. Render the scene

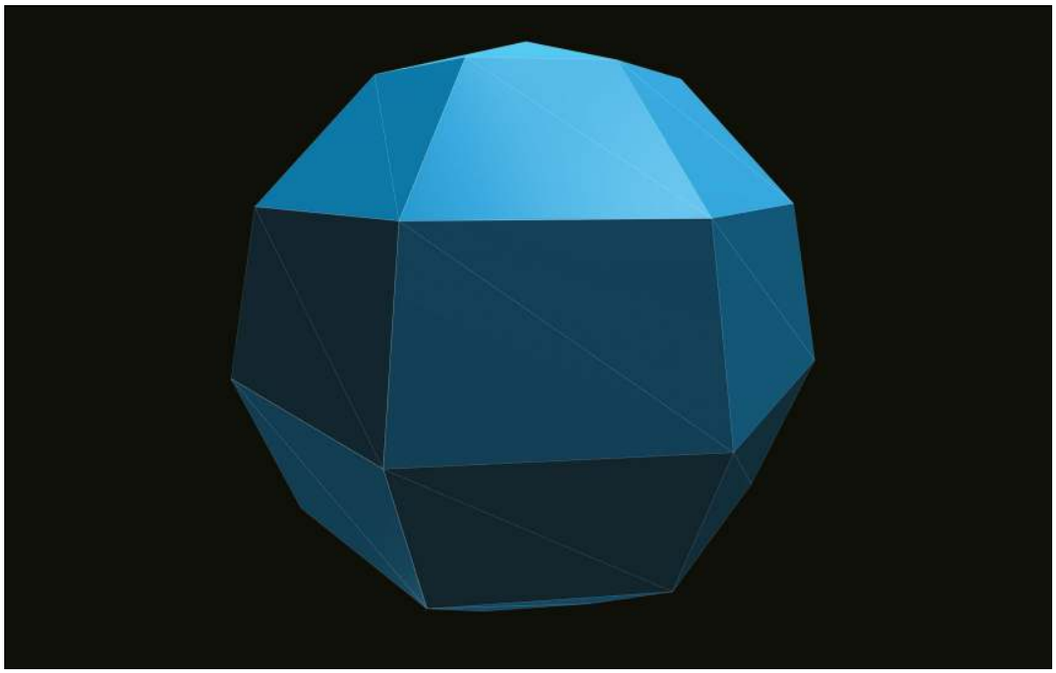
Now that you have a scene and a camera, along with a 3D object in your scene, you can render it. Rendering is the function of drawing the data to the canvas. It will draw to the canvas you indicated when you created it (either the generated one or using the 'canvas' attribute). Use this code now to see your cube!

```
renderer.render( scene, camera );
```

Top
You can now see your object rendered, although it's straight towards the camera. The normal material is nice because you don't need to light it to see it



Right
You can see how applying rotations to objects can let you manipulate them in 3D space. Radians are used here instead of degrees



Additional 3D primitive objects

Three.js and other 3D libraries come with a variety of primitive geometries you can use in your scenes. Spheres use the following form:

```
SphereGeometry(radius : Float, widthSegments : Integer, heightSegments : Integer, phiStart : Float, phiLength : Float, thetaStart : Float, thetaLength : Float)
```

Try out a sphere instead of your cube, using this code:

```
var geometry = new THREE.SphereGeometry( 5, 32, 32 );
```

Another useful shape is the cylinder:

```
CylinderGeometry(radiusTop : Float, radiusBottom : Float, height : Float, radialSegments : Integer, heightSegments : Integer, openEnded : Boolean, thetaStart : Float, thetaLength : Float)
```

Try swapping in this code to replace your cube:

```
var geometry = new THREE.CylinderGeometry( 5, 5, 20, 32 );
```

16. Render each requestAnimationFrame

To animate scenes smoothly you need to render at least 24 frames per second (ideally 60 fps). You will bind your 'render' function in a loop to the 'requestAnimationFrame' function. It will optimally run at 60 fps, and ensure the browser is ready to render the next frame. Replace your render line with this new code:

```
// render the scene
var render = function () {
  requestAnimationFrame( render );
  renderer.render( scene, camera );
};
render();
```

17. Add window resize handler

It's a good idea to add a window resize event handler to the code. It will keep it looking great, if the screen size changes or if the phone gets rotated. All you need to do is update the camera aspect, the 'projectMatrix' and the renderer size as resizing occurs. Add this code to the bottom of your JavaScript code:

```
// window resize handler
window.addEventListener( 'resize', function () {
  camera.aspect = window.innerWidth / window.innerHeight;
  camera.updateProjectionMatrix();
  renderer.setSize( window.innerWidth, window.innerHeight );
}, false );
```

18. Rotating objects

If you haven't already, remove the 'wireframe' attribute from the material so you can see the shaded cube. It will look nicer when you animate it. All objects have a 'rotation' property. In Three.js these are in radians not degrees. radians = degrees × (Math.PI /180). Try this code to rotate the cube:

```
cube.rotation.y=Math.PI/180 * 45;
cube.rotation.z=Math.PI/180 * -25;
// alternative format
cube.rotation.set(0, Math.PI/180 * 45, Math.PI/180 * -25);
```

19. Positioning objects

Similar to rotation, you can change an object's position in 3D space. The Z axis points towards the camera by default, the Y axis is up and down, and the X axis runs horizontally across the screen. Experiment with the position values to get the idea. Units are in metres, when you start working in VR/AR projects.

```
cube.position.x=1;
cube.position.z=.1;
cube.position.y=-.1;
// alternative format
cube.position.set(1,.1,-.1);
```

20. Animate the cube's rotation

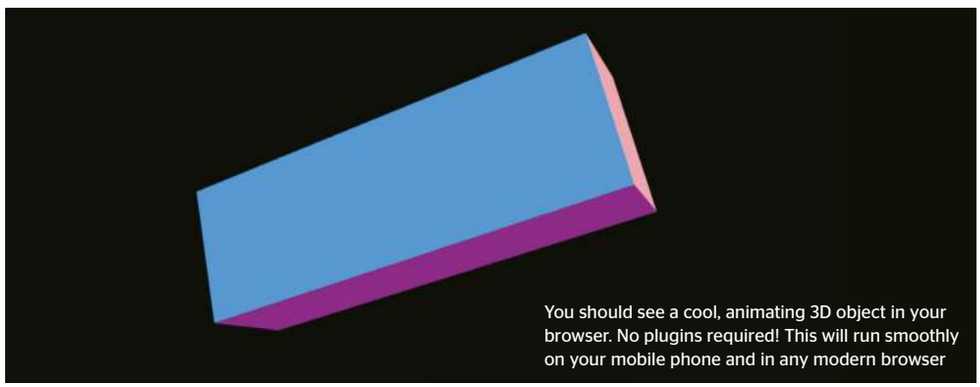
To animate the cube's rotation we can add the rotation code inside the render loop. This will update the rotation every frame. You can simply increment its rotation each frame. Update your render loop to look like this:

```
// render the scene
var render = function () {
  requestAnimationFrame( render );
  cube.rotation.x += 0.01;
  cube.rotation.z -= 0.01;
  renderer.render( scene, camera );
};
render();
```

21. Animate the cube's position

Incrementing in a linear way like we just did is good in some cases, but you can apply any form of easing or maths you like to your animations. Try using the JavaScript 'sine' or 'cosine' functions to give some easing to your position animations to finish off with a really smooth effect.

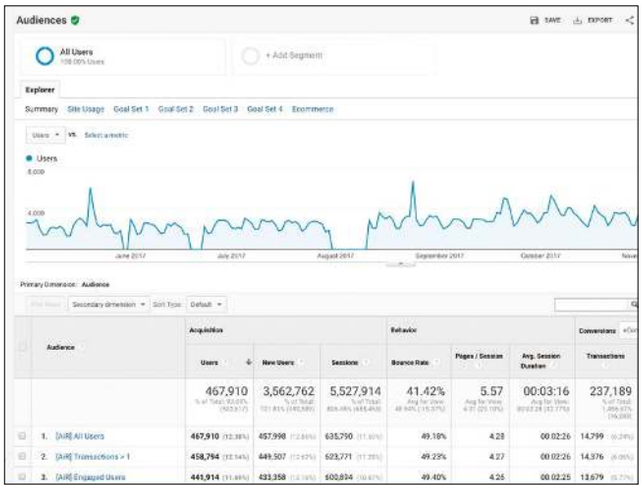
```
cube.position.x = 2 * Math.sin(cube.rotation.x);
cube.position.z = 2 * Math.sin(cube.rotation.z);
```

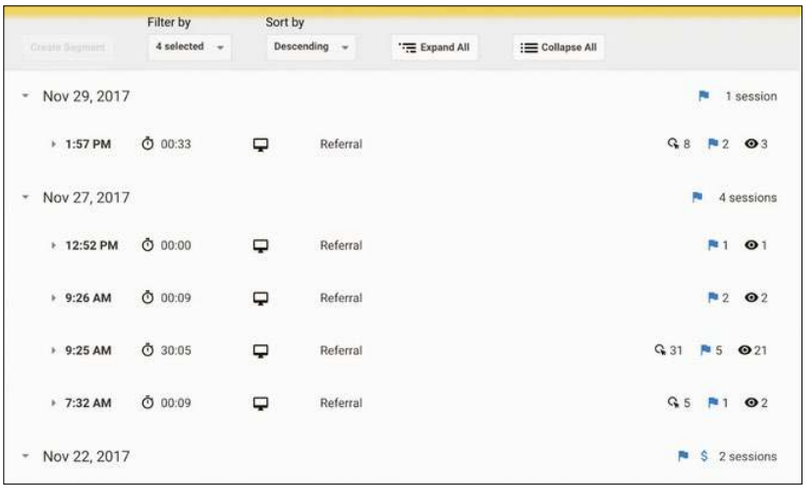


You should see a cool, animating 3D object in your browser. No plugins required! This will run smoothly on your mobile phone and in any modern browser

What's new in Google Analytics?

Discover how the latest Machine Learning updates will help marketers understand their user's journey better





Google has announced a range of significant updates to its Analytics reporting platform, utilising its machine-learning capabilities to create a deeper understanding of individual users. This is music to marketers and website owner's ears, since we are working hard to meet a resounding user demand for online experiences that are both relevant and personal. The more we know about our users' behaviour, the better equipped we are to make smart choices about our website, mobile app (iOS and Android) and application development investments.

These critical updates comprise four new features: all of which will help us to measure customer journeys at a deeper level, while accurately understanding individual user engagement across channels and devices.

This tutorial highlights the latest four enhancements, which include: Lifetime Metrics and Dimensions; Audience Reporting; Conversion Probability; and User-focused Reporting. These are on top of existing Analytics tools, which should help in your search marketing analysis of website users and their behaviour.

With Google continuing to improve the accuracy of user-level data and enabling more options for insights and targeting, we as marketers have an exciting future with more insights than ever before.

1. Enable User-focused Reporting

The standard reporting dashboard has been adapted, allowing 'user metrics' to be at the forefront when reviewing users' behaviour, as well as their journey to converting. This is a welcome shift from the historical focus on just sessions. Go to Admin > Property Settings > Enable Users in Reporting.

2. Audiences in Reporting

We now have the option to create audiences within Google Analytics and publish them to our analytics reports. This enables marketers to set 'audiences' as a primary and explore their behaviour and performance across different segments over time.

Previously you were only able to activate audiences and publish them on Google platforms such as AdWords and DoubleClick.

Being able to monitor the interactions of audience groups that matter to your business will provide insights and trends that you can actively take advantage of. Enable Demographics and Interests > Create Audiences > Publish.

3. Lifetime Metrics and Dimensions

Lifetime Metrics and Dimensions enables marketers to isolate user behaviour down to the session level. This allows marketers to view the lifetime metrics and dimensions at the individual user level (based on the lifetime of their cookie).

This provides a detailed, holistic overview of individual user behaviour, giving valuable insights on: how many times they have visited; the total time spent on a site; what pages they are looking at; the total number of transactions made, and so on.

We can effectively maximise the value of these insights, by delivering a better experience through tailored messaging and remarketing.

4. Demographics and Interests

As well as understanding your users' journeys better, you can also understand more about them by enabling Demographics and Interests Reports. These provide valuable insights on age, gender and interest categories that will help to shape your ongoing marketing activities. Go to Audience > Demographics.

5. Behaviour Flow

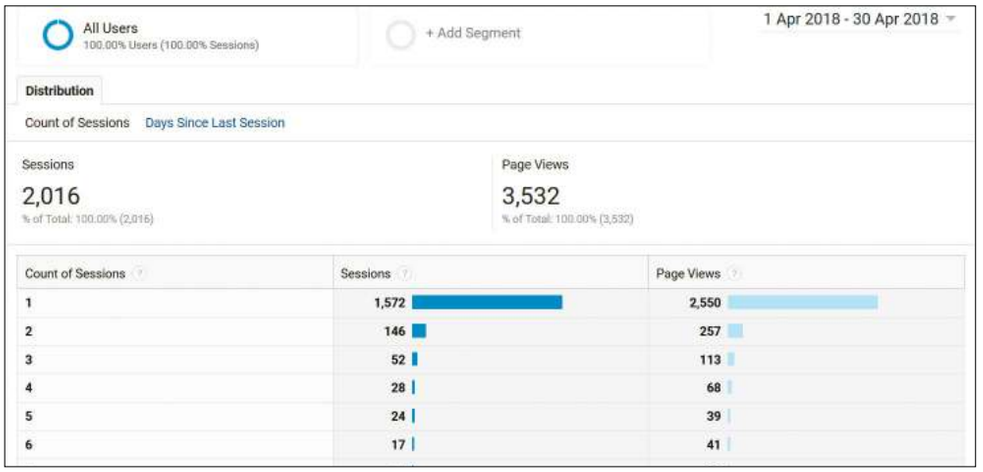
See how users interact with your website via the Behaviour Flow report, by visualising the path they take from one page or event to the next. This helps you to discover which content keeps users engaged, as well as identifying potential usability issues. Go to Behaviour > Behaviour Flow.

6. Frequency and Recency

Gain insights on how many times users visit your website, and how many days there are between visits. This is particularly useful in planning your marketing campaigns when combined with a conversion segment, since it will

Probability and Optimize

Publish conversion probability audiences in Google Optimize to gain further insights and understand exactly which refinements your website content needs to deliver the highest likelihood of conversion.

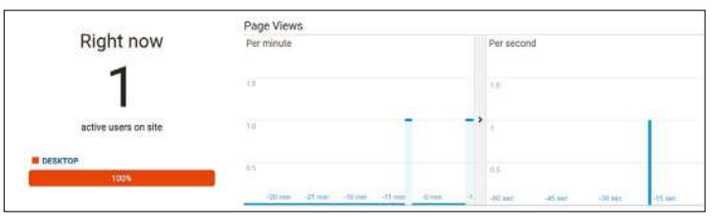


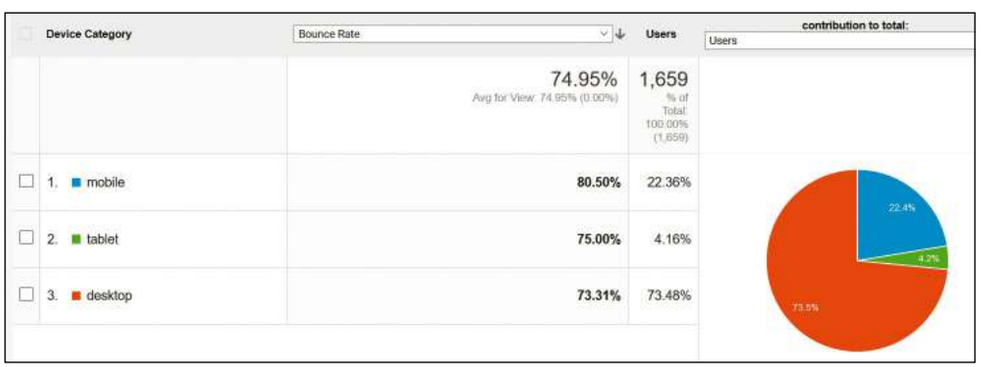
Above
Gain insights on how many times users visit your site

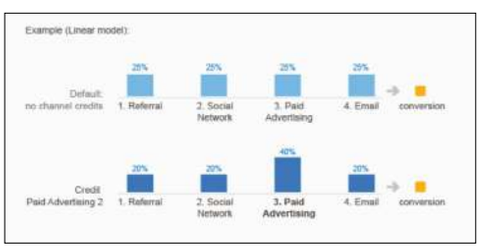
Right (top)
See who's on your site in real-time

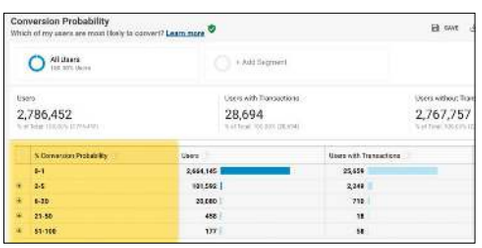
Below (right)
Conversion probablity is one of the most exciting new updates

Bottom
Find out how fast your site is loading and what can be done to improve it









Identify any content issues

Use the Behaviour Flow report to investigate how engaged your users are with your site's content: Is there an event that is always triggered first? Are there any patterns between desktop and mobile user pathways?

reveal data on how many visits it takes a user to convert over a set amount of time. With these segments you can identify patterns to help you schedule remarketing campaigns, email marketing and other promotions. Go to Audience > Behavior > Frequency & Recency.

7. Real-Time data

Real-Time is an invaluable resource within Analytics as you can review your user's interactions and behavior immediately... in real-time. This is particularly useful when launching a new website, design, or landing page. Go to Real-Time in your Analytics dashboard.

8. New vs Returning

Understanding what content encourages users to return to your website is a primary insight in driving increased user engagement, dwell time and conversion. Returning visitors are more likely to convert into customers or brand followers. Go to Audience > Behavior > New vs Returning.

9. Google's Mobile First Index

The mobile version of your website will become the baseline for how Google ranks and indexes your website. It is therefore vital to understand your mobile traffic, in particular the impact of mobile devices on user journeys, as well as conversions across your website. Go to Audience > Mobile.

10. Conversion Goals

Understanding how visitor segments are performing (across micro and macro conversion goals) can be extremely powerful. If you don't have the mechanisms in place to easily see how visitor segments perform, you won't have an understanding in how your campaigns perform. Go to Conversions > Goals.

11. Multi-channel funnels

Detail how your marketing channel traffic (direct, referral, organic, paid...) works together to create conversions. Understand what role prior website referrals, searches, and ads played in the final conversion. Without this, conversions are credited to the last touchpoint that referred the user.

12. Attribution models

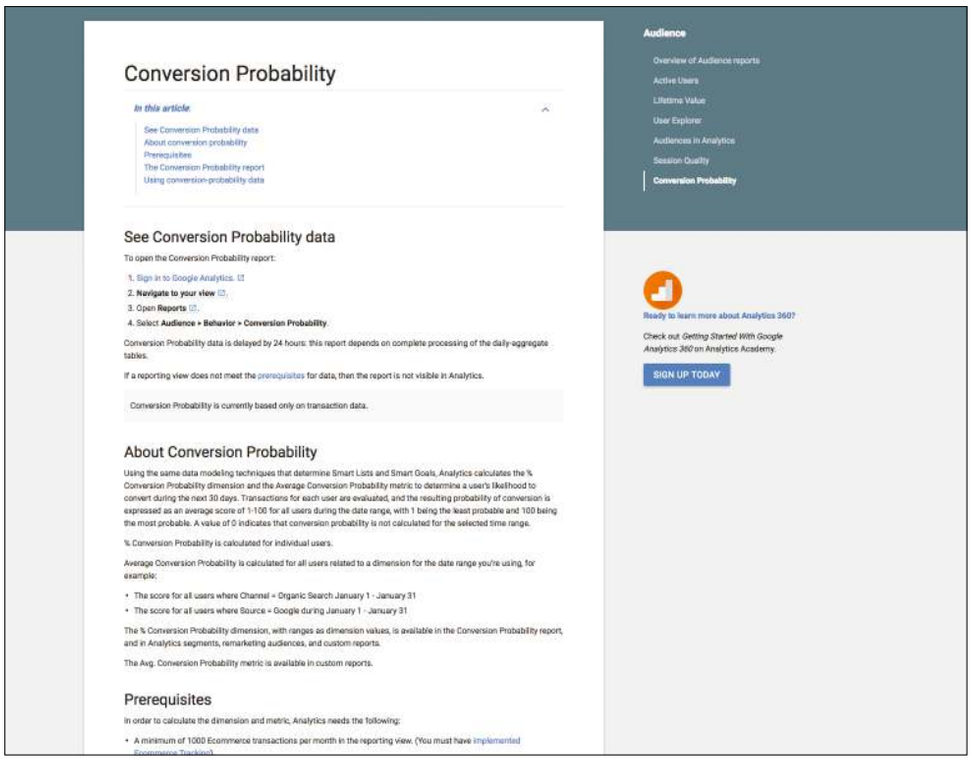
Assign credit for sales and conversions to touchpoints in conversion paths with the Model Comparison Tool. Compare how different Attribution models impact the valuation of your marketing channels, and what's the most effective use of your time. Go to Conversions > Attribution > Model Comparison Tool.

13. Conversion Probability

This is perhaps the most exciting of the four major updates from Google, since it has the highest potential impact on conversion. An exciting opportunity to take advantage of predictive analytics, and machine learning to identify future customers. Go to Audience > Behaviour > Conversion Probability.

14. Site Speed

The length of time your website takes to load not only



User's likelihood to convert

Using the same data modelling techniques that determine 'smart lists and goals', Analytics calculates the percentage conversion probability dimension, and the average conversion probability metric, to determine a user's likelihood to convert.

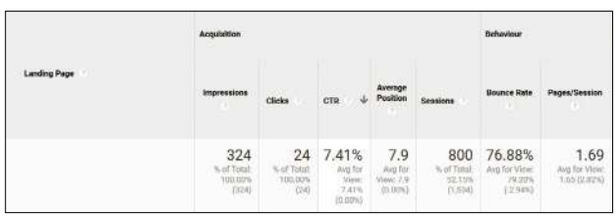
Transactions for each user are then evaluated, and the resulting probability of conversion is expressed as an average score of 1 (meaning least probable) to 100 (most probable) for all users during the date range.

By identifying recent website visitors with the highest probability of a future conversion, we can segment these potentially-converting users and create audiences for analysis and target them specifically in remarketing campaigns.

affects your user experience, it also influences your website's visibility in search. It is imperative to review your speed in Analytics and act upon the PageSpeed suggestions for each of your indexed pages. Go to Behaviour > Site Speed.

15. Utilise Search Console

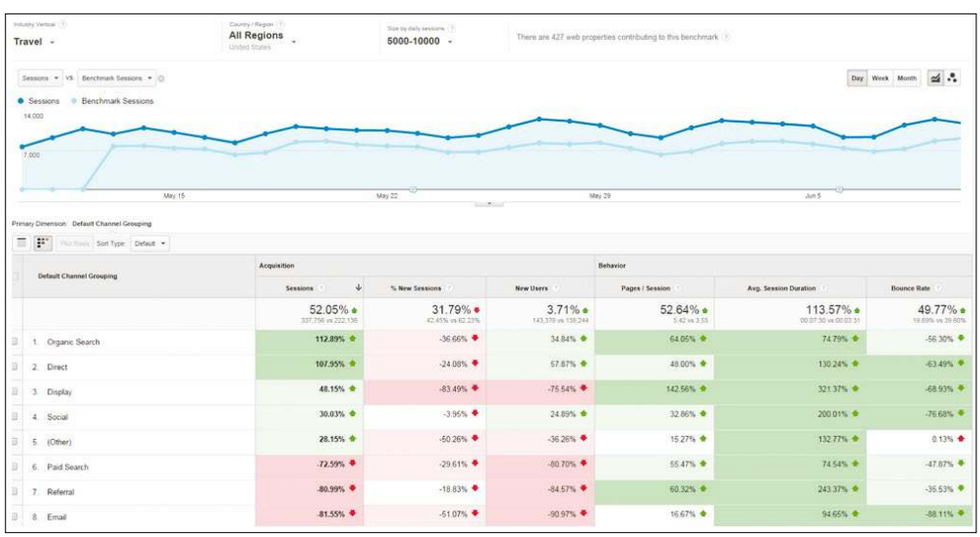
By configuring Search Console within your Analytics, you will see a combination of user data that will help to optimise your website for the most profitable traffic. This unlocks data, such as identifying landing pages which have good click-through rates (CTR) but poor average positions in the search results — these could be pages that people want to see, but are having trouble finding. This is an opportunity to improve both the content and metas to encourage more visitors. Go to Acquisition > Search Console.



Left
Optimise your site via the Search Console

16. Benchmark against the competition

Set meaningful targets, gain insights into your industry trends and benchmark yourself against the competition. This report provides valuable context by comparing website performance (channel, location, device and user flow) to previous results and to industry averages. Go to Audience > Benchmarking.



Above
Benchmark yourself against the competition to compare performance and more

17. Google Optimize

A great way to understand user journeys better is to perform testing. Google Optimize **(https://www.google.com/analytics/optimize)** is built on Analytics and enables you to discover the most engaging customer experiences by testing variations. You can then adapt your environment to deliver a personalised experience.

Animate UI profile cards with CSS

Add interest and engagement to profile cards with the help of contemporary CSS animations

CHECK THE CODEPEN
https://codepen.io/2975/pen/QrZpoa

When building web user interface elements, we've always been huge fans of transitions and transforms. Instead of just popping things in and out, everything that moves should fade or slide in and out of view on user interaction. This brings a super polished and professional look and feel to your websites or applications and should enhance the user experience, not hamper it.

jQuery had been for a long time, the best tool for the job, with its DOM manipulation being ideal. However, these days, to get our required look, all we need is CSS3. Throughout this tutorial we will build two profile card UIs that we would probably see within the team section of a agency website. Or these could also be used on smaller screens used for mobile devices. Whatever the case, building these just with HTML and CSS is what makes them so much fun to do.

To make things easier for us, we will be using the popular CodePen development tool so we can jump straight into it without the need to set up our local development environment – https://codepen.io. So open up CodePen and let's get started!

1. Getting started

Throughout this tutorial we will be using CodePen http://codepen.io. We won't need many dependencies for this, just font awesome and Sass. Once you're all set, go ahead and add a container div – which will be our main wrapper, and a div with a class name of 'card'.

```
<div class="container">
  <div class="card">
  </div>
</div>
```

2. Adding the profile image

Using your own image(s) from a local image folder would obviously be the norm, but we're using a free image hosting service to make things easier to use in CodePen. Your images should be around 350px wide to 450px in height and will go in between the 'card' div and will have a class called 'card-image'.

```
<div class="card-image">
  <img src="https://image.ibb.co/dUTfmJ/
profile_img.jpg"
  alt="profile one">
 </div>
```

3. Social icons

The social icons that we will be using will come from Font Awesome. If you haven't already done so, go ahead and add Font Awesome as a CodePen resource in the CSS settings. Then we will add the four most popular social icons within an unordered list which will go underneath the closing div tag of our profile image.

```
<ul class="social-icons">
  <li>
  <a href="">
  <i class="fab fa-facebook-f"></i>
  </a>
  </li>
  <li>
  <a href="">
  <i class="fab fa-instagram"></i>
  </a>
  </li>
  <li>
  <a href="">
  <i class="fab fa-twitter"></i>
  </a>
  </li>
  <li>
  <a href="">
  <i class="fab fa-dribbble"></i>
  </a>
  </li>
  </ul>
```

4. Profile details

At the bottom of each profile card there will be some details about the person. This will just be a name and job title, but feel free to use whatever tickles your fancy. These details will simply be given a class name of 'details' and sit just underneath our social icons.

```
  <div class="details">
  <h2>John Smith
  <br>
  <span class="job-title">UI Developer</
span>
  </h2>
  </div>
  </div>
</div><!-- end box wrapper -->
```

5. Setting up the CSS/Sass

The only Sass variable we will use is for the main body font. Then we will make sure that all elements are stripped of their default padding and margin, as well as the box model reset. However, if you don't want to strip all elements clean, you can include Normalize as a resource.

```
$bodyFont: 'Open Sans', sans-serif;
* {
  padding: 0;
  margin: 0;
  box-sizing: border-box;
}
```

6. Initial card styles

First thing we need to think about is how can we make this look like a card. Well, that's simply achieved by using border radius, setting the overflow to hidden and positioning it using absolute positioning and 'transform: translate'. With some width and height and a subtle drop shadow, things are starting to take shape.

```
body {
  font-family: $bodyFont;
  background: #333;
}
.card {
```

Immediate start

The default value for transition-delay is 0s, and as this suggests, no delay will take place. This means that the transition it is applied to will start immediately.

Left
All the main HTML has been added now and we can see our main image and unstyled social icons

Top
Now we have positioned our profile card and began to make it look like a card by adding rounded corners and a subtle drop shadow

```
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  width: 350px;
  height: 450px;
  border-radius: 16px;
  overflow: hidden;
  box-shadow: 0 5px 18px rgba(0, 0, 0,
0.4);
  cursor: pointer;
  transition: 0.5s;
}
```

7. Image hover

One of the things we want to happen when the card is hovered over is for our profile image to fade away slightly. We also need to make sure the z-index is set higher than '1' so that our details section will be hidden underneath until we're ready to reveal it. Also, make sure this 'card-image' CSS rule is nested within the 'card' rule.

```
.card-image {
  position: absolute;
  top: 0px;
  left: 0px;
  width: 100%;
  height: 100%;
  z-index: 2;
  background-color: #000;
  transition: .5s;
  }
  &:hover img {
  opacity: 0.4;
  transition: .5s;
  }
}
```

Centre with Flexbox

When you have two UI elements that you want to align centre and sit side-by-side, using Flexbox is the ideal solution.

8. Animating the image up

Not only do we want our image to lose some of its opacity when we hover over it, we also want the image to move up so we reveal the profile details underneath. We can easily achieve this by using the 'translateY' property and giving it a negative value.

```
.card:hover .card-image {
  transform: translateY(-100px);
  transition: all .9s; }
```

9. Social icons

The social icons can now be positioned. Even though these will be hidden to begin with and then animated on, we do need to set their final state. We will also want each icon to be next to each other inline, so we will use 'display: flex' to achieve that.

```
social-icons {
  display: flex;
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  z-index: 3;
  li {
  list-style: none;
  }
  // anchor styles goes here... }
```

10. Anchor styles

The social icons now need some styling added to them. Using the anchor tag we can create 50x50px white squares with a 6px margin left and right. We will set the icons to a dark grey but will end up giving these their own hover state in a later step.

```
a {
  position: relative;
  display: block;
  width: 50px;
  height: 50px;
  line-height: 50px;
  text-align: center;
  background: #fff;
  font-size: 23px;
  color: #333;
  font-weight: bold;
  margin: 0 6px;
  }
  }
}
```

11. Hide social icons

The social icons now need to be hidden and moved to the bottom of the card so we can animate them back into view. We also need to give this a transition so when we do reveal them, it's a lot smoother. So go ahead and add this additional CSS within the 'a' rule we just created.

```
transition: .4s;
transform: translateY(200px);
opacity: 0;
```

12. Revealing the social icons

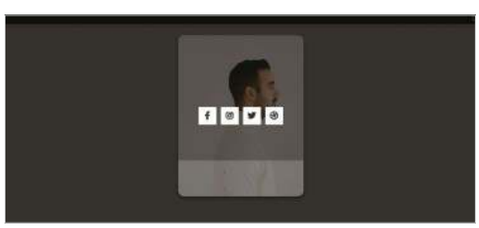
Now we can begin to have a little bit of fun with our social icons. With a hover state added to the card class, we can reveal our icons by setting the opacity to '1' and the 'translateY' value to zero. With that transition added in the last step, we should now have a nice animation happening on hover.

```
.card:hover .social-icons li a {
  transform: translateY(0px);
  opacity: 1;
}
```

13. Icon backgrounds

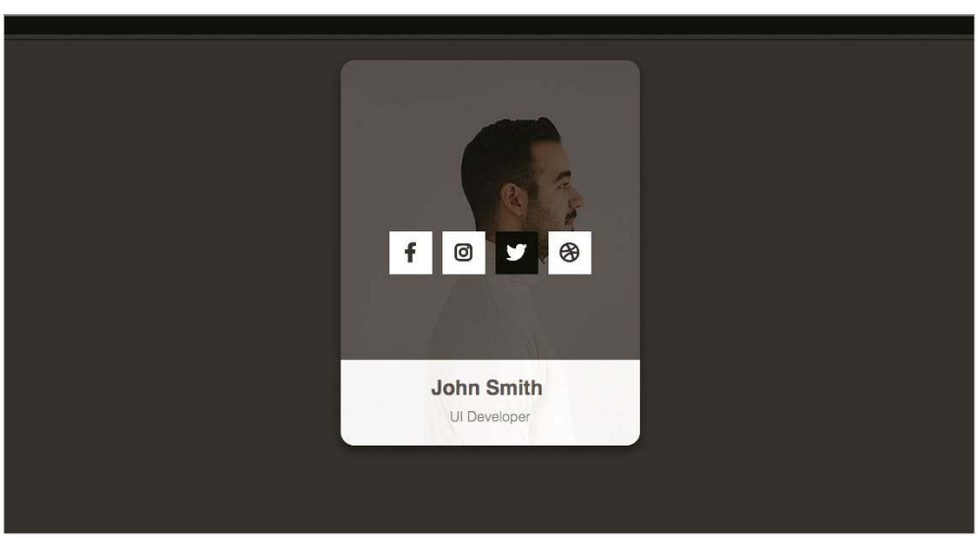
Hovering over our icons now doesn't really feel that nice without some sort of transition happening on their backgrounds. So adding a black background and setting the icon to white when we hover, gives us a much better experience.

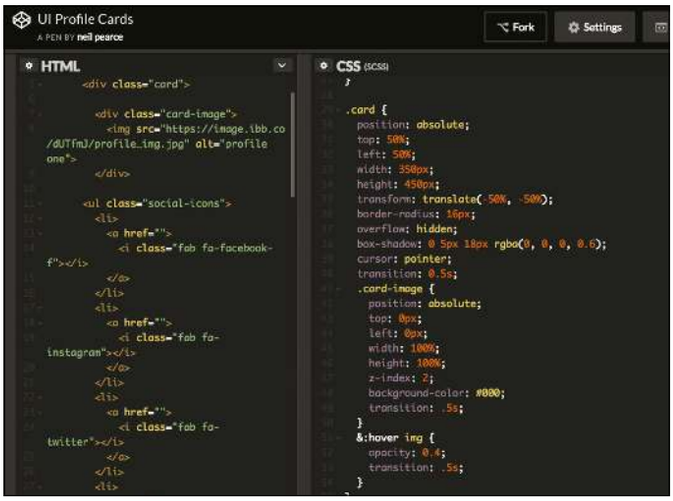
```
.social-icons li a:hover {
  background: #000;
  transition: .2s;
  .fab {
    color: #fff;
  } }
```



Top
We've given the social icons a square background and positioned them where we would like them to end up after they're animated

Right
Then we've given the social icons some more styling by adding a nice background transition





The transform property

Throughout this tutorial the transform property is used extensively. Can't remember, or not quite sure what the 'transform' property can do? We are going to take more detailed look here.

The 'transform' property allows you to manipulate elements by rotating, skewing, scaling and translating.

```
&:hover
{
transform: rotate(15deg);
color: #fff;
}
```

The above rule rotates an element clockwise from its original position. Conversely, using a negative value would rotate an element in the opposite direction, ie to the left. You can get even more specific by using either the 'X' or 'Y' axis as shown in the next code snippet.

```
&:hover {
transform: rotateY(360deg);
color: #fff;
}
```

You can also use 'translate' to position elements, using either pixels or percentages.

```
transform: translate(-50%,
-50%);
```

These values can be any length value, like 50px or 2.5em. One value will move the element to the right (negative values to the left). If a second value is provided, that second value will move it down (and negative values up). This is just a small insight into the transform property – if you would like to learn more, then recommended reading can be found on the MDN web docs website: **https://developer.mozilla.org**

14. Spinning the icons

Even though we now have a nice animation and hover effects happening, why stop there! Let's add some rotation on the icons so when we hover over them, they spin 360 degrees. The transition will be a bit longer than normal (0.8s) as we want the rotation to be nice and smooth.

```
.social-icons li a .fab {
  transition: .8s;
  &:hover {
      transform: rotateY(360deg);
      color: #fff;
  }  }
```

15. Animation delays

We're not quite finished with our social icons just yet as we now want to be more creative and animate each individual icon one after the other. We can do this by targeting each icon using the 'nth-child' psuedo selector, and then setting a transition delay on each one.

```
.card:hover li:nth-child(1) a {
    transition-delay: 0.1s; }
.card:hover li:nth-child(2) a {
  transition-delay: 0.2s; }
.card:hover li:nth-child(3) a {
  transition-delay: 0.3s; }
.card:hover li:nth-child(4) a {
  transition-delay: 0.4s; }
```

16. Styling the details

The details section, which gets revealed on hover, will need to have some styling added to it. For now we will just position it at the bottom, give it a white background and make sure the z-index is set to '1'.

```
.details {
  position: absolute;
  bottom: 0;
  left: 0;
  background: #fff;
  width: 100%;
  height: 120px;
  z-index: 1;
  padding: 10px; }
```

17. Finishing up the details

To keep things tidy, the CSS for both the header tag and job title for this details section can be nested in the details rule. Then we can give the 'h2' tag some margin at the top and bottom and finish off giving the job title a font size and weight.

```
h2 {
margin: 30px 0;
padding: 0;
text-align: center;
.job-title {
font-size: 1rem;
line-height: 2.5rem;
color: #333;
font-weight: 300;
}  }
}
```

18. Second profile card

To create another profile card, copy and paste all of the HTML except the main container div, and then add an additional div called 'card-wrapper' in between the container and card divs, and close that off underneath the details closing div. Then we can use Flexbox on the container class to position both profile cards on the page side-by-side.

```
.container {
max-width: 900px;
display: flex;
justify-content: space-between;
margin:  auto; }
.card-wrapper {
width: 400px;
height: 500px;
position: relative; }
```

19. Second profile classes

Now we have a new profile card, instead of keeping to the same animations, let's create something different. First make sure the new class 'profile-two' is added to the card div and the class 'profile-img-two' is also added to the image div. And lastly add a class called 'jane' on the details div.

```
<div class="card profile-two">
    ….
<div class="card-image profile-img--two">
     ….
<div class="details jane">
```

20. Second animation

So to make things a little more interesting, we will add a rotation of 360 degrees to the main image when hovered. Then we will turn the squared social icons into circles, but we will keep their transitions the same. And lastly we will give our details (now targeted by the class name 'jane') a slight delay in being revealed so that the image rotation has time to finish.

```
.jane {
  position: absolute;
  bottom: -120px;
  left: 0;
  opacity: 0;
  background: #fff;
  width: 100%;
  height: 120px;
  z-index: 3;
  padding: 10px;
  transition: .4s; }
.profile-two .social-icons li a {
  border-radius: 50%; }
.card:hover .profile-img--two {
  transform: rotateY(180deg); }
.card:hover .jane {
  bottom: 0;
  left: 0;
  transition-delay: 0.5s;
  opacity: 1;
}
```

21. Final thoughts

As you can see, you can get as creative as you want with user interface animations. However, you do need to be careful not to over do things, and the user experience should always be your first thought. So challenge yourself to create a slightly different one now that you've finished doing these two!

POWER UP YOUR CSS

We weigh up the pros and cons of the 8 best preprocessors to help you find the best one for your needs

Less

lesscss.org

Less is stylistically very similar to SASS in its feature set, and so anyone that has used one will feel right at home with the other. Its popularity boosted heavily after it was used in the source Twitter Bootstrap. It has since moved to SASS in Bootstrap 5, but it has left a lot of people comfortable using its syntax. The fact that it is so similar to Sass makes it difficult to advocate when Sass is more widely used, actively developed, and feature-rich – indeed this feature would be comparing the two (as many have before) if there were more differences and unique selling points about it. It still remains a popular and strong preprocessor though.

Pros

- Many web developers are familiar with JavaScript and because Less is written in JS, it can be processed client side making the set-up easy.
- GUI apps that can watch and compile code for you (Crunch, SimpLESS, WinLess, Koala, CodeKit, LiveReload or Prepros will watch and compile less for you).
- It has very detailed documentation, and a very active community which makes finding help or previous examples very easy.
- IDEs such as VS Code, Visual Studio and WebStorm support LESS either natively or through plugins.

Cons

- Less uses the '@' symbol to declare variables. However '@' already has meaning in CSS, as it is used to declare @media queries and @keyframes. This can result in some confusion when reading the code.
- Despite being similar to approaches like SASS, it's adoption is far less universal, and therefore getting used to its intricacies may not be the best time spent if you're not going to find it used in projects.
- Makes you rely entirely on mixins rather than allowing you to use functions that can return a value. Can result in slightly restricted use cases.

PostCSS

postcss.org

PostCSS is one of the biggest 'alternatives' to Sass, Less and Stylus when it comes to preprocessing based on its approach – it leverages a modular plugin system that allows the user to customise their feature set and compilation as much as they want. This means that rather than just adopting a library in its entirety, you get to choose what it's made of. Unlike many other preprocessors, PostCSS makes no assumptions about the features or stack that you're using. Instead, you simply add the plugins that you would like based on the features you want to use. You can add plugins to give it the exact same features as a library like Sass. Due to this modular approach, it means that you can even use plugins completely by themselves, such as auto-prefixing and minification, rather than building a full library.

Pros

- Un-opinionated, modular approach to preprocessing that can be heavily customised to your own preference.
- Speed can also be a big bonus from this approach depending on your setup. If you only make use of a few plugins, it can end up running and compiling quicker than its competitors.
- Uptake of the library is one of the biggest pros as well. PostCSS allows you to write in pure CSS, which means that developers don't need to learn a new syntax in order to use or understand it.

Cons

- Weirdly speed is a con as well as a pro. Other fully-fledged libraries are set up and optimised to accomplish the chosen features they have. When adding all the necessary plugins to allow PostCSS to mimic a setup like that, the speed is inferior.
- It often takes a lot more effort to install and maintain than a conventional preprocessor. This is because you are relying on a lot of different plugins maintained by a lot of different people. This can mean a plugin is modified or broken and no longer useful.

Pros

- Sass has the lowest barrier to entry – you can harness some of the most powerful features by simply learning a couple of new symbols. You can add it to your stack without fear that it will be too steep as a learning curve for new collaborators.
- Sass is written in Ruby, but there is also a C/C++ port of the Sass precompiler called Libsass that decouples Sass from Ruby. It's fast, portable and easy to build, and integrates with a variety of platforms and languages.
- Sass has by far the most engaged community, with forums, tutorials on every major teaching platform for code, and well-maintained sites dedicated to helping people get more from Sass. **thesassway.com** in particular breaks down posts into beginner, intermediate and advanced levels of knowledge.

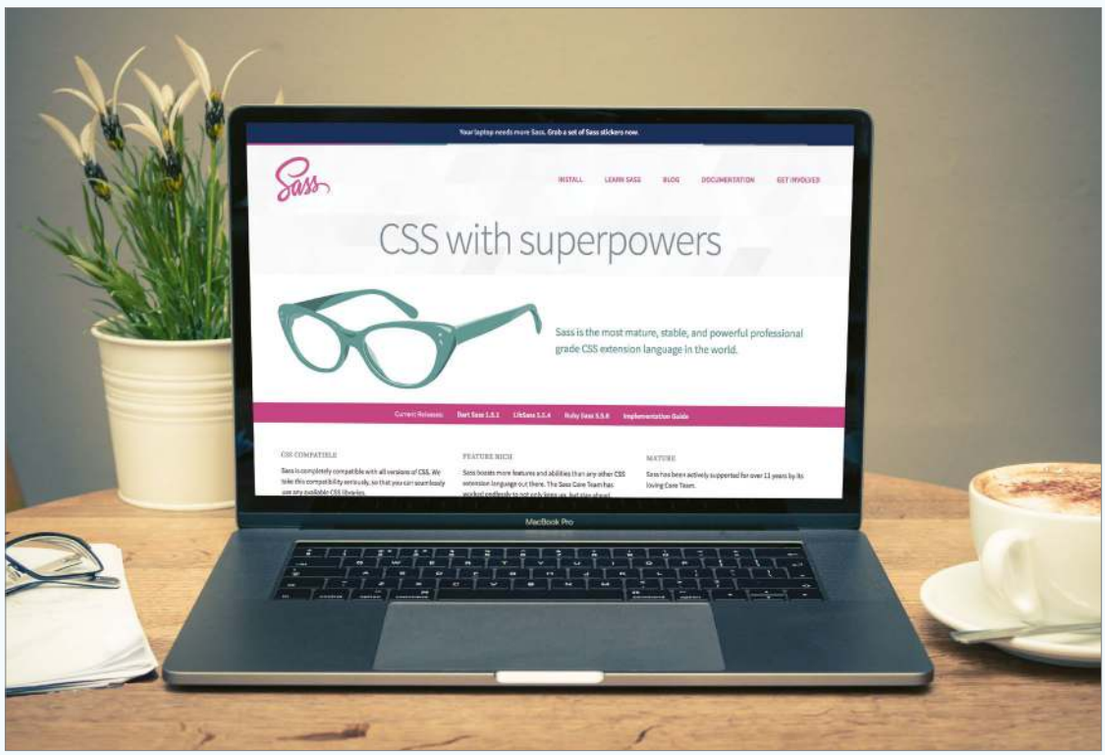
Cons

- "A big problem with frameworks is when up and coming developers attach themselves to a framework as opposed to the underlying code itself. The knowledge gained in this case surrounds a specific framework, which severely limits the developer." Jonathan Christopher said this, and it can apply to any language. Despite it's low barrier to entry, there is always a worry that you end up reliant on writing your code within a particular approach, which hinders your ability to understand the core language underneath. This is worth considering if you implement frameworks that others may have to adopt.

SASS *sass-lang.com*

"Sass is the most mature, stable, and powerful professional grade CSS extension language in the world". That comes from the makers, but it's pretty hard to contest. For over 11 years, Sass has been diligently worked on by its core team to build a preprocessor that is feature-rich, has a large community to support, and develop it and its users, and has been adopted by some of the biggest names in the industry. Sass is built on Ruby and offers two different syntax types depending on the user's preference. SASS doesn't use braces around selectors, or semi-colons at the end of rules, but feature-wise is identical to SCSS, which still uses both of those.

Despite being different from the name itself, SCSS is the most commonly used language, mainly because it means that there is nothing syntactically different from plain CSS – making adoption for the basic principles really easy. Additionally, every major task runner has a module to use with Sass.



THE BEST OF SASS

How to use the best and most powerful features of Sass

1. NESTING

The ability to mirror the nested hierarchy style of HTML in CSS is arguably the biggest improvement that Sass brings as a preprocessor to plain CSS. It stops you from repeating yourself constantly, makes project maintenance easier, and is far more pleasant to read as you can see exactly what rule relates to which selector:

```
// Plain CSS
.foo { .. }
.foo ul { .. }
.foo ul li { .. }
// Sass
.foo {
  ..
  ul {
  ..
  li { ..}
  }
}
```

2. REFERENCE SYMBOL

When nesting, you can use the '&' symbol to reference the parent element. This helps in two main ways. Firstly you can neatly nest pseudo-states or elements:

```
.foo {
  background-color: red;
  &:hover { background-color:
blue; }
  &:after { content: 'bar'; }
}
```

but also to combat pesky overwrite issues:

```
.foo {
  background-color: red;
  &.bar { background-color:
green; }
  .ie &.bar { background-color:
yellow; }
}
// =>
.foo { background-color: red;}
.foo.bar { background-color:
green; }
.ie .foo.bar { background-color:
yellow; }
```

3. VARIABLES

All variables in Sass begin with a '$' symbol followed by the name (without spaces):

```
$breakpoint-sm: 576px;
```

Which makes handling global values such as colours, fonts and breakpoints infinitely easier.

4. OBJECT VARIABLES

In Sass, variables can also be stored as an object for cleanliness. Here is a small object of variables for breakpoints:

```
$grid-breakpoints: (
  sm: 576px,
  md: 768px,
  lg: 992px
);
```

In order to retrieve a variable stored in this way, you can use a built-in Sass function called 'map-get' to return one of the variables based on a key passed in as an argument. Here we'll use it to set the max-width value of a '@media' query:

```
@function breakpoint-
var($breakpoint) {
    @return map-get($grid-
breakpoints, $breakpoint);
}
@media (max-width: breakpoint-
var(lg)) {
}
// => @media (max-width: 992px) {
//
// }
```

CSS does now have the 'var()' function to be able to create simple variables with '--' prepended, but the SASS preprocessor version is much more powerful.

5. MIXINS

Mixins bring the power of reusable code to CSS. Rather than having to go throughout a stylesheet and change a property multiple times, you can just handily change it inside a mixin:

```
@mixin foo() {
  border: 1px solid red;
}
.bar {
  @include foo();
}
// =>
// .bar { border: 1px solid red;
}
```

You can also set parameters for your mixins so that you can use the same styles but with slight adjustments based on your needs. You can even set default parameters for those arguments:

```
@mixin foo($width: 1px, $color:
red) {
  border: 1px solid red;
}
.bar {
  @include foo();
}
.bax {
  @include foo(5px,
blue);
}
// =>
// .bar {
border: 1px
solid red; }
// .baz { border:
5px solid blue; }
```

@ CONTENT IN MIXINS

Using this inside a mixin allows you to pass styles directly into the body of it on a case-by-case basis; great for breakpoint mixins!

6. FUNCTIONS

Functions and mixins can often be interchangeable and accomplish the same result, but their purposes are slightly different. As with many things in programming, the answer is use logic: mixins are used more for includes, and functions are more for returning values. For example, the 'breakpoint-var()' function we used earlier has a 'return' not seen in mixins.

7. @EXTENDS

This is brilliant for reducing duplication in your CSS, allowing classes to share a set of properties with one another.

```
.foo {
  color: black;
  border: 1px solid black;
}
.bar {
  @extend .foo;
  background-color: red;
}
// =>
.foo, .bar {
  color: black;
  border: 1px solid black;
}
.bar { background-color: red; }
```

You can even extend multiple selectors in the same rule using a comma-separated list

```
@extend .foo, .bar;
```

It can also be great for simplifying the way you name classes:

```
// Without @extend
.foo {
  background-color: red;
  border: 1px solid blue;
  color: black;
}
.bar { background-color: green; }
<div class="foo bar">...</div>
// With @extend
.foo {
  background-color: red;
      border: 1px solid blue;
          color: black;
          }
          .bar {
            @extend .foo;
            background-
          color: green;
          }
          <div
          class="bar">...</
          div>
```

The only downside to this we would raise is the drastic increase in the size of your stylesheet if you use it constantly.

8. @IMPORT

There's nothing more intimidating or headache-inducing than an overly large stylesheet, particularly if you're scouring through it for a particular part. Thankfully Sass allows you to separate your styles into multiple files, and then use '@import' to bring then in as and when you need them! The typical naming convention for a partial is prepending an underscore before the file name and then importing it with its name. For example, if we wanted to import separate styles for buttons, we would name the file '_buttons.scss' and then add: '@import buttons;'. Just like '@extend', you can import multiple imports using a comma-separated list:

```
`@import 'buttons', 'forms';`
```

You can import inside a selector (if it makes sense based on the contents of your partial:

```
.foo {
  @import 'bar';
}
```

And your partials don't even have to be on the same directory level:

```
@import 'utils/mixins';
.foo {
  @import 'components/buttons';
}
```

This gives you complete freedom in architecting the layout of your styles.

9. LOOPING

You can loop in Sass using three main rules:

`@for` – loop for a set amount of iterations, with access to the index on each loop

`@while` – loop until the check in place is no longer true

`@each` – loop through every item in a given list

If you have experience with JavaScript (or most programming languages) you will have been exposed on some level to these principles and they can be just as powerful in your styling as they are there. For example, Bootstrap's entire flex-based grid system is built in Sass using all of these heavily. A simpler example could be looping five times to quickly created a staggered fade-in animation for items using

```
`nth-child`:
.foo {
  @for $i from 1 through 5 {
  &:nth-child( #{i} ) {
  opacity: 1;
  transition: opacity .5s;
  transition-delay: #{($i * .05)
+ .05}s;
  }
  }
}
```

10. INTERPOLATION

The '@each' loop works really well with a powerful feature called interpolation; a way of using the content of the values you're looping (syntax is `#{VALUE}`) in the output itself. So, if we'd like to set up our heading rules using an object variable with keys and fonts, we could do the following:

```
$fonts: (
  h1: 24px,
  h2: 20px,
  h3: 16px
);
@each $element, $size in $fonts {
  #{$element} {
  font-size: $size;
  line-height: ($size * 1.5);
  }
}
```

Pros

- Hugely powerful built-in functions.
- **(http://stylus-lang.com/docs/bifs.html)**.
- Light() and dark() – is the colour light or dark?
- red(), blue(), green() gets the correct number for each of those in 'rgb'.
- push() and pop() – just like JS, add or remove from a variable with a list of values.
- Mathematical functions – abs(), percentage(), ceil(), floor() and round().
- blend() blends two colours.
- The ability to do much more computing and 'heavy-lifting' inside your styles.
- Written in Node.js which is fast and fits neatly with a 2018 JavaScript stack.
- Due to Stylus' 'pythonic' syntax, it looks a lot cleaner and involves you writing fewer characters.

Cons

- It's too forgiving. GitHub user declandewet had a great way of summarising the downside of such syntactical freedom: "It supplies the developer with no definitive direction... once you're ingrained in variables, mixins and functions not requiring a prepended dollar sign ($) or 'at' symbol (@), you'll soon start to realise that you can no longer tell the difference between them. This makes for very confusing code".
- It doesn't seem to be in very active development. A GitHub issue was raised last year as a form of discussion about the lack of ongoing development, where the primary maintainer of the project announced that they were stepping down, and the last update to the core code was three months ago.

Stylus *http://stylus-lang.com*

Stylus was created by TJ Holowaychuk who is a former programmer for Node.js and the creator of the Luna language. Its initial design was influenced by Sass and Less but it offers a wider range of features, a super fast Node.js system under the hood, and gives users the most freedom in how they write their CSS. It was used by Mozilla to redesign its developer network and has had David Walsh raving about it. Stylus offers most of what we just covered in the Sass section, but with a few minor syntax alterations. To avoid unnecessary duplication, we'll go through areas in which Stylus differs from Sass slightly in approach, and some features unique to Stylus that could be well worth knowing if you choose to use it.

THE BEST OF STYLUS

How to use it, and how it differs from Sass

1. SYNTAX

It's syntax is whitespace based, for nested as well as regular selectors:

```
body
  color white
compiles to:
body {
  color: #fff;
}
```

You can you can use braces and semi-colons if you would prefer, but we're not sure how you'd possibly be able to read your styles if you did. Both sit inside files with the '.styl' extension, instead of '.scss', '.sass' or Sass.

2. VARIABLES

Stylus variables don't require a '$' like Sass variables (although it does allow for them if you would like to follow that particular convention). You could argue that this is cleaner syntax, but also that it makes variables harder to spot when looking through a style file.

3. IMPORTS

For a start, there is no '_' before '.styl' files. '@import' works by iterating an array of directories, and checking if this file lives in any of them (similar to Node's require. paths). '@import' also supports index styles. This means when you '@import foo', it will resolve either 'foo.styl' or 'foo/index.styl'. This is really useful for libraries that want to expose all their features, while still allowing feature subsets to be imported.

4. EXTENDS

Stylus' '@extend' rule behaves the same way the Sass' does. However, Sass will throw a syntax error if you try to extend a nested selector. As shown here:

```
form
  button
    padding: 10px
a.button
 @extend form button
 Syntax error: Can't extend form
button: can't extend nested
selectors
  on line 6 of standard input
   Use --trace for backtrace.
```

Whereas in Stylus this will work:

```
form
  input[type=text]
    padding: 5px
    border: 1px solid #eee
    color: #ddd
textarea
  @extends form
input[type=text]
  padding: 10px
```

5. TRANSPARENT MIXINS

Transparent mixins allow you to pass argument(s) into a mixin like a normal selector, rather than having to call mixins and pass parameters individually:

```
border-radius()
  -webkit-border-radius:
arguments
  -moz-border-radius: arguments
  border-radius: arguments
a.button
  border-radius: 5px 8px
```

This certainly looks a lot neater from a cleanliness perspective. It is worth noting though that a lot of these features that are mentioned attempt to solve problems that

you may already have solved through your stack. In this particular example you may be covered through the use of autoprefixer or a similar library – you simply may not need your CSS preprocessor to handle this for you, and their addition to the library provides additional bloat that you don't need. However, the principle opens up a lot of possibilities.

6. FUNCTIONS – MULTIPLE RETURN VALUES

Functions in Stylus perform largely the same way as they do in Sass, but they have a couple of tiny quirks worth knowing:

- Functions can return multiple values, and you can reference them individually:

```
sizes()
    15px 10px
  sizes()[0]
  // => 15px
```

- You can alias functions so that one uses the name of another

```
add(a, b)
  a + b
plus = add
plus(1, 2)
// => 3
```

- You can also pass a function into another function which makes the ones you create even more dynamic

```
add(a, b)
  a + b
sub(a, b)
  a - b
calculate(a, b, fn)
  fn(a, b)
body
  padding calculate(5, 10, add)
  padding calculate(5, 10, sub)
returns:
body {
  padding: 15;
  padding: -5;
}
```

7. PARENT REFERENCE

Both Sass and Stylus use the '&' symbol as a way of targeting pseudo-selectors when nesting (eg '&:hover' for the hover state of the selector this rule is inside) however Stylus also exposes it so you can alter the selector you're styling when certain parent classes exist. For example, this mixin simultaneously styles the login ID, whilst adding a border to it only when the parent <html> tag has a class of 'ie' – all inside the same mixin:

```
box-shadow()
    -webkit-box-shadow arguments
    -moz-box-shadow arguments
    box-shadow arguments
    html.ie &
      border 2px solid
arguments[length(arguments) - 1]
  body
    #login
      box-shadow 1px 1px 3px
#eee
```

Becomes:

```
  body #login {
  -webkit-box-shadow: 1px 1px 3px
#eee;
  -moz-box-shadow: 1px 1px 3px
#eee;
    box-shadow: 1px 1px 3px
#eee;    }
  html.ie body #login {
    border: 2px solid #eee;    }
```

This can be really powerful in preventing the need to write duplicate rules. The only downside here is that this extra use of '&' inside the syntax means that you have to escape it when you need to use it in a selector itself, such as:

```
.foo[title*='\&']
// => .foo[title*='&']
```

8. PARTIAL REFERENCES

This is a really nice advantage that Stylus has over Sass. A partial reference gives you access to any level of a nested rule you have written, regardless of where you use it. In SASS you may need to do something like this:

```
.foo
  &__bar
  width: 10px
  &:hover
  .foo__bar
 width: 20px
```

To output the following:

```
.foo__bar {
  width: 10px;
}
.foo:hover .foo__bar {
  width: 20px; }
```

Whereas using a partial reference you can achieve the same result with less code, and prevent the need to re-write the selector name out (which would quickly become hard to maintain):

```
.foo
  &__bar
  width: 10px
  ^[0]:hover &
  width: 20px
```

9. PROPERTY LOOKUP

Another cool feature unique to Stylus is property lookup: the ability to use the value of an attribute by simply naming it with a '@' symbol before it. As a simple example, you could set the top and left margins of an absolutely positioned element to be minus half of the element's width and height, thereby centring it both vertically and horizontally:

```
  #logo
  position: absolute
  top: 50%
  left: 50%
  width: 150px
  height: 80px
  margin-left: -(@width / 2)
  margin-top: -(@height / 2)
```

When you pair this with some of Stylus' function operators, you start to be able to create some really powerful rules. For instance, setting an attribute on a selector through a mixin, but only if that attribute isn't already set, looks something like this:

```
position()
    position: arguments
    z-index: 1 unless @z-index
  #logo
    z-index: 20
    position: absolute
  #logo2
    position: absolute
```

To top it off, property lookups can also bubble up through the levels of a nested selector, and use the first attribute that matches the name. Here, the background colour for 'body ul li a' would be red:

```
  body
  color: red
  ul
  li
  background-color: blue
  a
  background-color: @color
```

10. KEYWORD ARGUMENTS (KWARGS)

More good news for programmers. Similar to a lot of languages, keyword arguments allow for parameters to be passed to functions without a particular order. These two calls to the 'rgba' function we mentioned earlier:

```
 body {
     color: rgba(red: 255, green:
200, blue: 100, alpha: 0.5);
     color: rgba(alpha: 0.5, blue:
100, red: 255, 200);
}
```

Returns the same values thanks to kwargs:

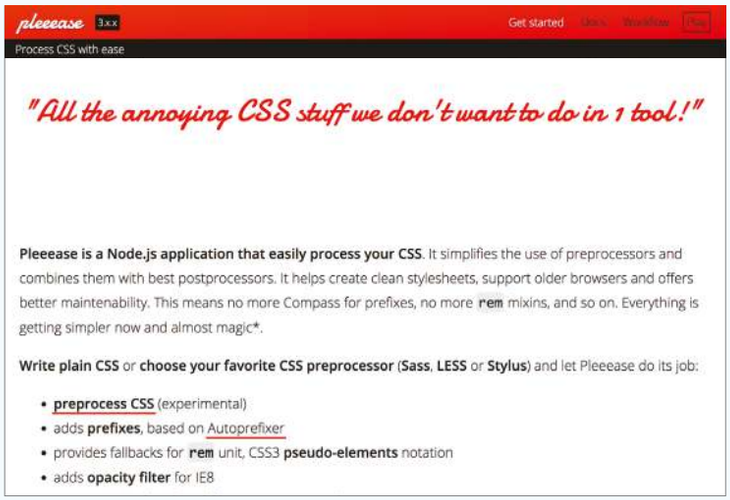
```
body {
  color: rgba(255,200,100,0.5);
  color: rgba(255,200,100,0.5);
}
```

Although it may be familiar and useful to some, it doesn't allow for a lot of consistency between function calls, and so may be confusing if you come back to the code, or if others are also working on it.

11. DEFINE()

One of our favourite features of Stylus is the ability to turn an object variable into a list of variables using the built-in 'define' function. This basically means that you can store all of the attributes of a certain type in one object variable, and then assign them to variables with a certain prefix. You can also decide, using the 'global' flag, whether the variables exist only in the current scope, or whether they apply globally (the layout is 'define(name, expr[, global])'). So let's say we have all of our attributes relating to borders specified in this object 'border = { color: #000, length: 1px, style: solid }'. Now to get them all out in a way that doesn't involve creating them manually, you can use 'define' to create the following (not just for this object but for any):

```
prefix = 'border'
border = { color: #000, length:
1px, style: solid }
for prop, val in border
  define(prefix + '-' + prop,
val)
body
  border: border-length border-
style border-color
// =>
// body {
  border: 1px solid #000;
//}
```



Pleeease *pleeease.io*

Pleeease is a slightly different approach to preprocessing in that it tries to tackle some of the more practical issues with CSS rather than focusing purely on its syntax or layout. According to its website, it is:

"A Node.js application that can easily process your CSS. It simplifies the use of preprocessors and combines them with the best of post-processors. It helps create clean stylesheets, support older browsers and offers better maintainability". It provides fallbacks to common issues with older browsers out of the box. These include pixel fallbacks for when you're using rem's as your measurement of choice, and filter fallbacks for IE8 when using opacity.

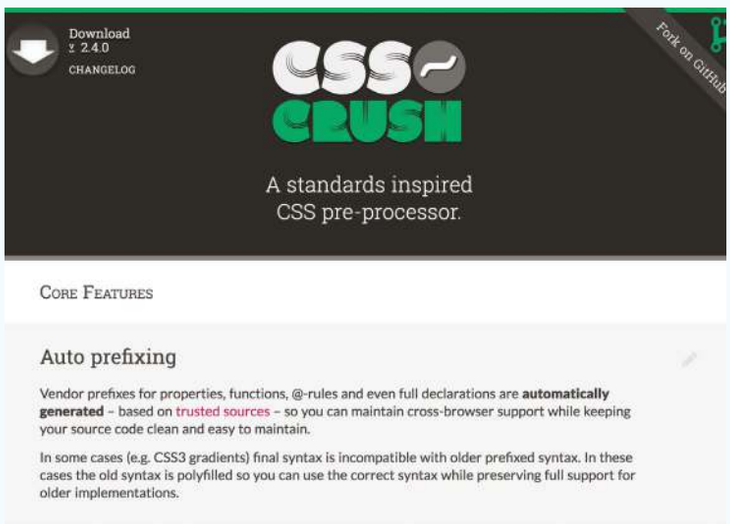
It even has a feature to allow you to use the syntax of your favourite preprocessor like Sass or Less within Pleeease's setup (as well as pure CSS), although this is experimental at this point.

Pros

- Pleeease uses autoprefixer to add vendor prefixes. It gets its data from **caniuse.com**, and is built in rather than having to add it.
- It generates source maps which allows you to see the initial written CSS (or Sass, Less etc) despite the output being minified CSS.
- Pleeease is a tool rather than a syntax library, so you can use it with pure CSS, or with another preprocessor.
- Clear, practical uses for dealing with older browser quirks.

Cons

- Pleeease isn't a very well-known tool. As a result, this may make finding guides, tutorials or resources more difficult than it would be for more popular preprocessors.
- The feature that allows for the inclusion of other preprocessors is completely experimental at this stage, which means that you may end up running into issues as you try to do more with it or use it regularly.



CSS-Crush *the-echoplex.net/csscrush*

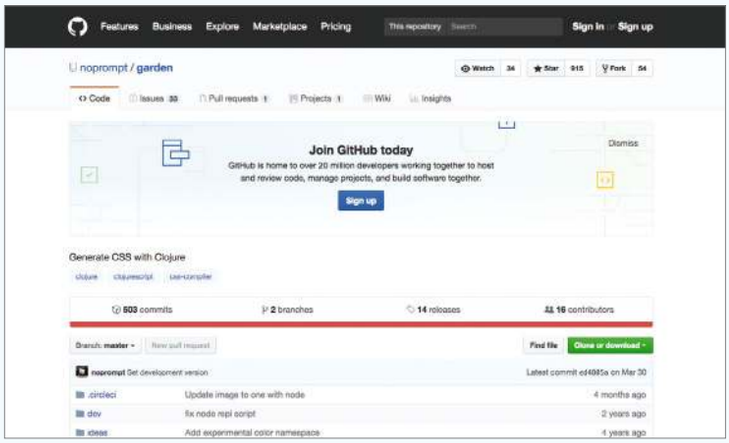
CSS-Crush is a standards-inspired preprocessor designed to enable a modern and uncluttered CSS workflow. It's written in PHP and features a combination of the standard features you would expect in one of the more popular preprocessors (variables, nesting, mixins) along with some of the more tool-based approaches we've covered like Pleeease (vendor prefixing, minification). It's PHP background means that it can be used neatly in conjunction with popular PHP content management systems such as WordPress or Drupal. This is probably its biggest pro because if you're confined to what you can do inside a CMS, you can install it as a plugin and still benefit from some of the advantages of having a preprocessor. One bonus is that its vendor prefixes for properties, functions, @-rules and even full declarations are automatically generated. This means that you can maintain cross-browser support while keeping your source code clean.

Pros

- Neat integration with popular PHP content management systems (Wordpress, Drupal etc).
- It's open source, which means you could technically go about fixing issues that you encountered yourself if you wanted to.
- It has a plugins section that is filled with useful bolt-ons such as working with aria roles and HTML canvas.
- Built-in autoprefixing for cross-browser styles.

Cons

- Despite its incredibly wide range of features, it has a real lack of popularity. CSS-Crush has just over 500 stars on GitHub, which means that its wider appeal beyond the regular PHP programming community is not large.
- Another factor is maintenance. At the time of writing this, there hasn't been a change made to its source in six months, with other parts remaining unchanged for years.



Garden *github.com/noprompt/garden*

This option is completely different to all the others because it pretty much does away with the conventional language of CSS as we know it. Garden is a library for rendering CSS in Clojure and ClojureScript. It uses vectors to represent rules and maps to represent declarations. It is designed for "stylesheet authors who are interested in what's possible when you trade a preprocessor for a programming language." As far as programming languages go, Clojure is known for its clean architecture and firm programming-language heritage. For CSS, this can mean great power. However, the syntax can be daunting. As an example, the below code is used to set no font-weight on h1, h2 tags.

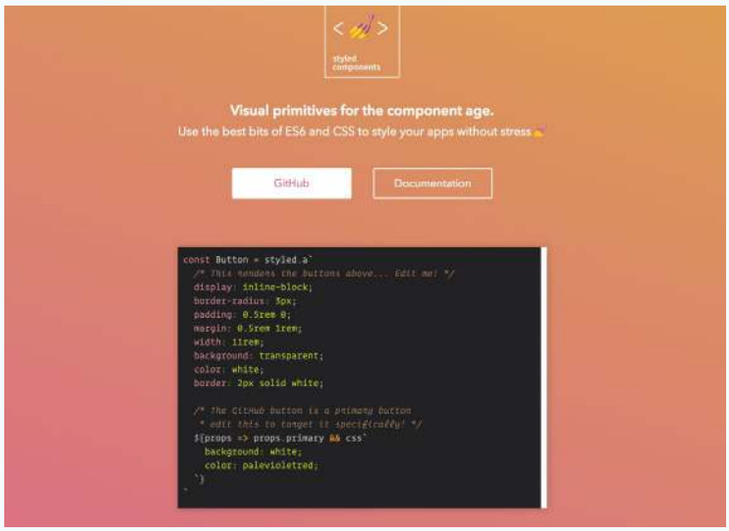
```
user=> (css [:h1 :h2 {:font-weight "none"}])
```

Pros

- Garden grants you access to the core features of a powerful programming language in Clojure, allowing you to build scripts using functions, variables, namespaces and data manipulation.
- The fact that it is built on Clojure means that you have a unique opportunity to code a project entirely in the same language front to back; Clojure for the back-end programming and Garden for the CSS. This creates a lot of consistency.
- Use the Garden Gnome plugin and you can pipe style changes directly to the browser without reloading.

Cons

- Because you are writing in a totally different language, you lose the ability to simply copy and paste CSS from elsewhere into your work; every snippet would need work to convert it to the correct format.
- Very different syntax to regular CSS or any other preprocessor, making it more difficult to read.
- Because of the syntax, the learning curve for Garden is steeper than the other options we're covering. It could be tough for other members of your team to hop straight on to a project using it.



Styled Components

styled-components.com

This last option is a pivot from conventional preprocessors. I only raise it in this context as it's certainly a way of writing your styles in a certain manner and layout and having it convert, alongside your component logic, as browser-ready CSS. Only the basics are covered here and its style handling. Styled Components is 'CSS-in-JS for the CSS folk' as Glen Maddern put it. It's the latest attempt to make truly modular CSS by interweaving it into the JS components you write. It has a big advantage over inline React styling as you don't have to camel case your attributes, and you declare each style block directly onto the name of the element you'll be using it on, like so:

Create a title component that'll render a <h1> tag with some styles like this:

```
const Title = styled.h1`
  font-size: 1.5em;
  text-align: center;
  color: palevioletred; `;
```

Pros

- You can achieve total encapsulation of your components. Every piece of markup, logic and styles are completely self-contained.
- Preventing 'append-only styles'. What are append-only styles? A team may work on the same site (or sites) and the CSS gets split into multiple files using Sass. To ensure there are no issues with the code, nothing is deleted. Developers may add to them but not take anything away.

Cons

- You are back to writing plain CSS, which has its drawbacks.
- This option is also quite opinionated, as it's built to work solely with the React JS library. It does, however, have a very large following, and active community and project owner, and represents a really fascinating shift in direction.
- You have to handle cross-browser issues and fallbacks using other methods.

WHAT'S NEW IN NATIVESCRIPT 4.0

The arrival of the latest update to the open source framework for building native mobile apps has brought with it a collection of new features. Find out what's on offer

New installation experience

There is life after the death of the TeleRik Platform

One of the most interesting aspects of TeleRik's NativeScript was the TeleRik platform: an integrated development environment which lived in your browser, and handled compiling and other processes. Sadly, the product was discontinued in May. Alternatively, NativeScript sidekick is offered at **https://www.nativescript.org/nativescript-sidekick**.

It is available for Windows, Linux and Mac OS. When the installation has been completed, you need to check the presence of the tns command – especially on Ubuntu, the installation routine sometimes forgets to deploy the necessary declaration to your bashrc file.

In the next step, start the product from the desktop and follow the instructions on the screen. The product will, among other things, require you to log in with the TeleRik or another account in order to manage the compile process quota assigned to you.

When logged in, click the blue Create button to start generating a new project. One of the interesting aspects of the new toolchain is the presence of various project templates which allow you to start out with predefined elements. While some of the templates, such as blank, are quite basic, others implement advanced functionality showing off some of the new features of NativeScript 4.0. When done, the NativeScript sidekick will start up completely.

Keep in mind that it is not a full-featured IDE: instead, click the button 'Show files' in order to open the file manager of your workstation in the project home directory. Files can then be edited there, using the various utilities in the NativeScript sidekick window to deploy them to devices, compile them (in the cloud) or prepare them for deployment to an app store of your choice.

More than one frame

In the past, <frame> was everything. This has changed!

One of NativeScript's key features is the creation of user interfaces using a custom markup syntax derived from XML. The developer creates the various widgets using language and abstraction classes specified by Progress, which are then interpreted by the phone at runtime.

So far, the entry point – conveniently located in the file app-root.xml – was always made up of a frame class, which was populated with one or more widgets.

This is problematic if developers seek to realise its structures, such as a tabbed view, a side drawer or similar complex navigation elements commonly seen in business applications. One of the changes in NativeScript 4.0 is that the product does away with this limitation. In the future, the entry point can also contain code like the following:

```
<TabView>
  <TabViewItem title="Teams">
  <Frame defaultPage="teams-main-page"></
Frame>
  </TabViewItem>
  <TabViewItem title="Players">
  <GridLayout>
  <!-- content here -->
  </GridLayout>
  </TabViewItem>
</TabView>
```

In fact, developers can even have more than one frame instance simultaneously, using the APIs outlined at **https://docs.nativescript.org/core-concepts/navigation#getting-frame-reference** to find the one most relevant to the current operating state.

Hard-hitting changes

Under the hood, developers get some neat new upgrades

TeleRik originally financed NativeScript by selling various ancillary services – one of them was a set of professional user interface components:

- AutoComplete
- Calendar
- Chart
- DataForm
- Gauge
- ListView
- SideDrawer

They resided in a package called nativescript-pro-ui. Sadly, including it was not exactly healthy for your application as it added quite a bit of girth to the compiled file.

Upgrading is not particularly difficult. First, remove the old pro-ui plugin using the tns plugin utility or the plug-in manager found in the NativeScript Sidekick:

```
tns plugin remove nativescript-pro-ui
```

In the next step, the modules containing the actual user interface elements need to be readied:

```
tns plugin add nativescript-ui-chart
tns plugin add nativescript-ui-calendar
tns plugin add nativescript-ui-autocomplete
tns plugin add nativescript-ui-listview
tns plugin add nativescript-ui-sidedrawer
tns plugin add nativescript-ui-gauge
tns plugin add nativescript-ui-dataform
```

With that, the package can be remodelled for deployment – simply fix any syntax errors which pop up to end up with a working file.

More authentication options

NativeScript was never intended to be used to create games: it, instead, is usually used for creating business applications. Synchronising these effectively requires advanced authentication – a feature which Progress covers via the Kinvey business division. In particular, authentication libraries are provided for the following three server types:

- SAML
- OAuth 2
- OpenID Connect

Fortunately, deploying this particular feature is not hard. When starting with a new application skeleton, simply pick the Enterprise Auth template, which is available for TypeScript, JavaScript and Angular.

The scaffolding contained in it can then be applied to other projects on hand. Alternatively, some of the routines can also be copied over – the main library resides in the kinvey namespace:

```
const Kinvey = require("kinvey-nativescript-sdk").Kinvey;
. . .
  var activeUser = Kinvey.User.getActiveUser();
  if (activeUser == null) {
    Kinvey.User.loginWithMIC('http://example.com')
```

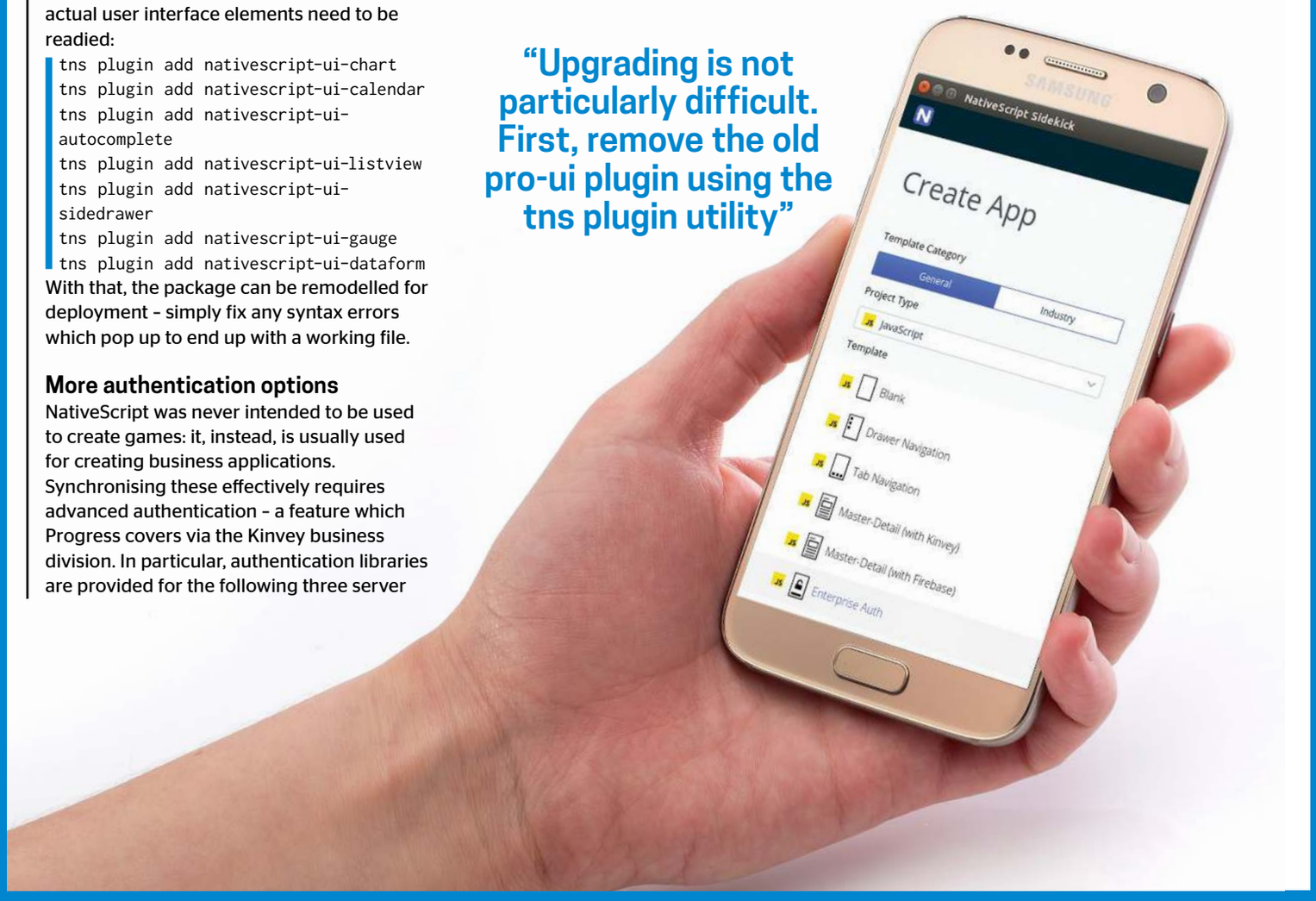
A question of need is a question of taste

Finally, the project structure can be modified using a file called 'nsconfig.json'. The parameters in it allow you to restructure the project by changing the paths leading to various resources if the plugins used are current:

```
{
  "appPath": "test/src",
  "appResourcesPath": "resources"
}
```

One especially interesting aspect of this reconstruction is that it is completely optional. If one or both of the parameters are not populated, this part is interpreted using the logic used in older versions of the product.

"Upgrading is not particularly difficult. First, remove the old pro-ui plugin using the tns plugin utility"



Graphical icon management

Resizing images and creatives by hand is painful and annoying

Diversity in the mobile space takes its toll due to the availability of different screen sizes: while iOS is somewhat constrained, Android screens literally range from tiny QVGA screenlets to the gigantic 4K AMOLED displays seen on high-end tablets.

One of the side effects is that icons and splash screen graphics need to be made available in a variety of different sizes. So far, developers have to create them by hand – using SVG traditionally proved unpopular. And, the lack of browser support did not help the cause.

In the new version of NativeScript, simply open the NativeScript Sidekick and click the Assets section to open the options shown in the figure accompanying this step. Then, simply select a bitmap with the size 1024x1024 – the Sidekick automatically rescales the images – and make sure that they get deployed to the right places for both Android and iOS.

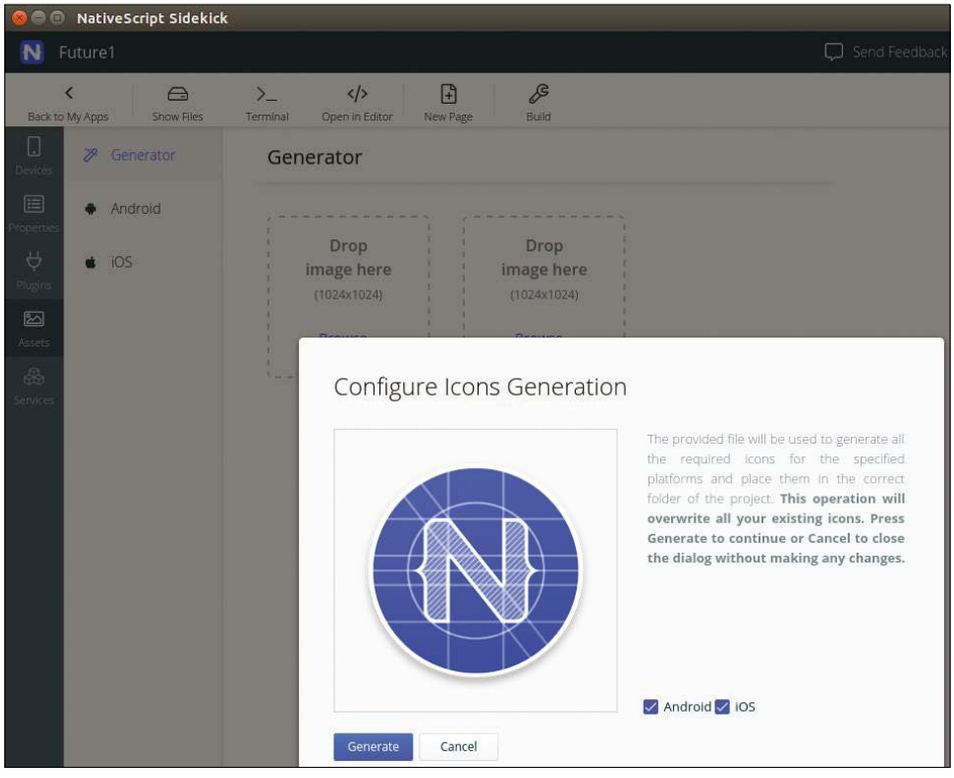
As with many other features of NativeScript, using the Sidekick is not strictly required. This feature can also be enabled from the command line by entering one of the following commands:

```
$ tns resources generate icons <path
to image>
$ tns resources generate splashes
<path to image> [--background <color>]
```

Do, however, keep in mind that automatic scaling of images tends to lead to imperfect results – as the image file gets smaller and smaller, tiny detail gets blurred, thereby leading to a less than satisfactory user experience, and no one wants that.

One straightforward way to work around this problem involves the use of the tool to generate the scaffolding, which can then be modified by hand to replace especially annoying files with better, hand-generated ones.

Developers who are currently using an SVG-based process will not need to abandon it either: simply export one version with the size of 1024x1024, scale it down and finally proceed to replacing problematic images with better versions.



Multiple modal windows

Sometimes, it takes more than one modal window to get the message across

While the back button makes users expect to be able to go back home from any screen, this is not always the case. If your application logic, for some reason, demands that a dialog must be filled out or interacted with, it can be declared modal using the following code:

```
  page.showModal(frame,
  "context",
  () => console.log("home-page modal
frame closed"),
  true);
}
```

A modal dialog is not a new feature: they belong to every good GUI stack, and were present in older versions of NativeScript. The main difference is that the showModal() function now no longer resides in page, but rather in view.

Furthermore, the GUI stack was redesigned to make it more flexible. Instead of only allowing the use of page root elements, you can now use arbitrary widgets. As if this weren't enough, one dialog can pop up after another one creating a cascade of modal dialogs.

Old hands might reject this idea, claiming that all the research from PalmSource indicated stacked modal dialogs to be inefficient. This is partially true – but while cascading was difficult given the lackadaisical hardware of the average Palm, today's smartphones will not buckle if they have to render more than one dialog at the same time.

"Instead of only allowing the use of page root elements, you can now use arbitrary widgets"

Android at the centre

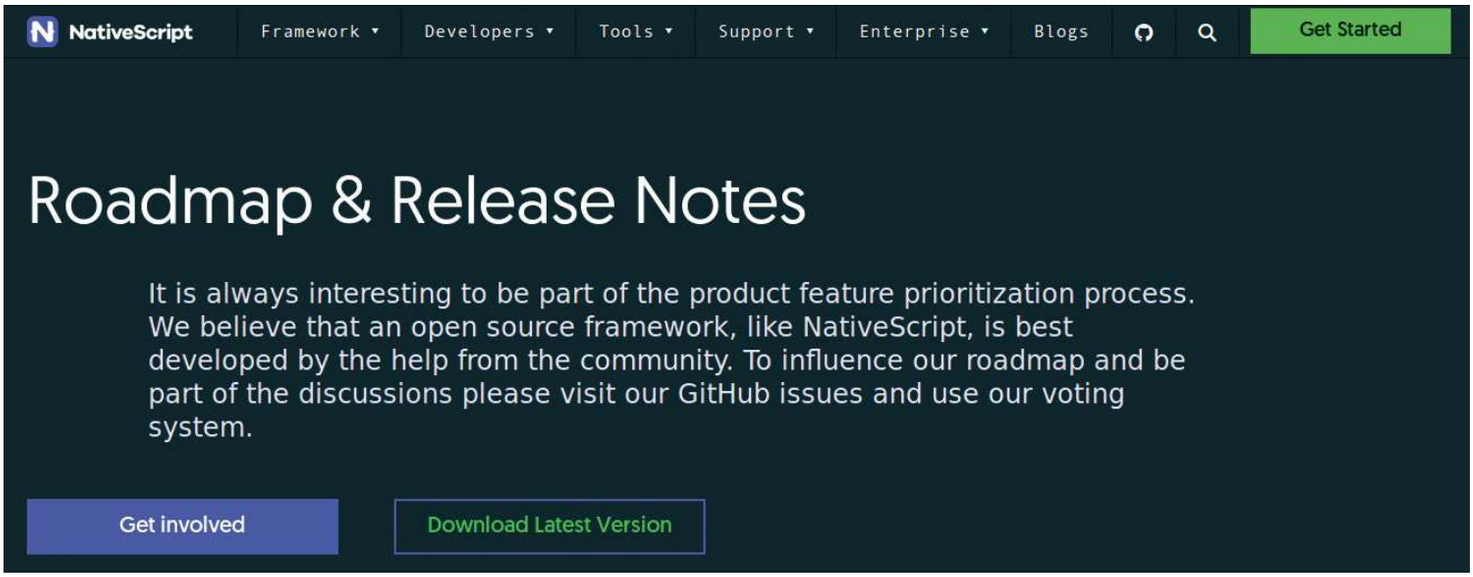
Developers working on applications for Google's mobile phone OS can look forward to changes

NativeScript always had pretty good support for plug-ins containing native code. The facilities were overhauled quite a bit in version 4.0, intending to make native development on Android less tedious.

Feature number one is the ability to open Android projects using the Android Studio IDE. That way, you can use various advanced features such as the profiler, and can furthermore place breakpoints in Android-specific code. This simplifies debugging, as the usually-used JavaScript debuggers tend to stop functioning the moment native Java islands are to be analysed. Trying this out is not particularly difficult.

In the first place, generate a project. Next, order the NativeScript sidekick or the tns client to perform a build – only when the build process has been completed for the first time, will the platform's directory be fully populated. Then, proceed to opening Android Studio, loading the build.gradle file.

An innovative feature (more detail at **https://bit.ly/2LicAXC**) is that Android plug-ins are now allowed to deploy a precompiled AAR file. In principle, an AAR file is a different version of a jar file, which has been provided with expansions allowing it to carry resources in additional to compiled code. This is beneficial for two reasons: first, programs compile faster when AAR files are already there. Second, taking work away from the client's workstation is beneficial – when implementing a new plug-in, developers should expect significantly less customer support effort when dealing with 'user' developers who, by virtue of NativeScript being based on JavaScript, often don't know particularly much about Android.



Get a detailed overview

One article cannot do justice to a whole release...

While NativeScript 4.0 is definitely a reliable and very mature development framework, Progress will not stop development. Firstly, all those who are deeply interested in the inner workings of NativeScript are advised to visit the website **https://docs.nativescript.org/releases/changes**.

It provides a detailed overview of the individual changes and bugs which were fixed in each of the four components which, together, make up NativeScript 4.0. In addition to that, the website found at **https://www.nativescript.org/roadmap-and-releases** provides a list of planned changes in the future. First and most importantly, the above-mentioned NativeScript sidekick is expected to undergo significant changes. The target is to make installs causes less hassle in terms of developing and deploying the various utilities needed for a successful build toolchain.

In addition to that, the development process will also be optimised. One interesting aspect is adding Vue.JS support and a variety of new interfaces. These will let developers who currently have native Android or iOS intellectual property add NativeScript islands with minimal effort.

In the future, NativeScript is furthermore intended to get significant enhancements in terms of both alternate and virtual reality. Progress explicitly states that they believe AR to play an increasingly significant role in the future, and want to provide NativeScript developers first-class access to the various features as they crop up.

CREATE THE IMPOSSIBLE

www.photoshopcreative.co.uk

EVERY ISSUE • DOWNLOAD FREE RESOURCES

Photoshop creative

Available from all good newsagents and supermarkets

ON SALE NOW

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

BUY YOUR ISSUE TODAY

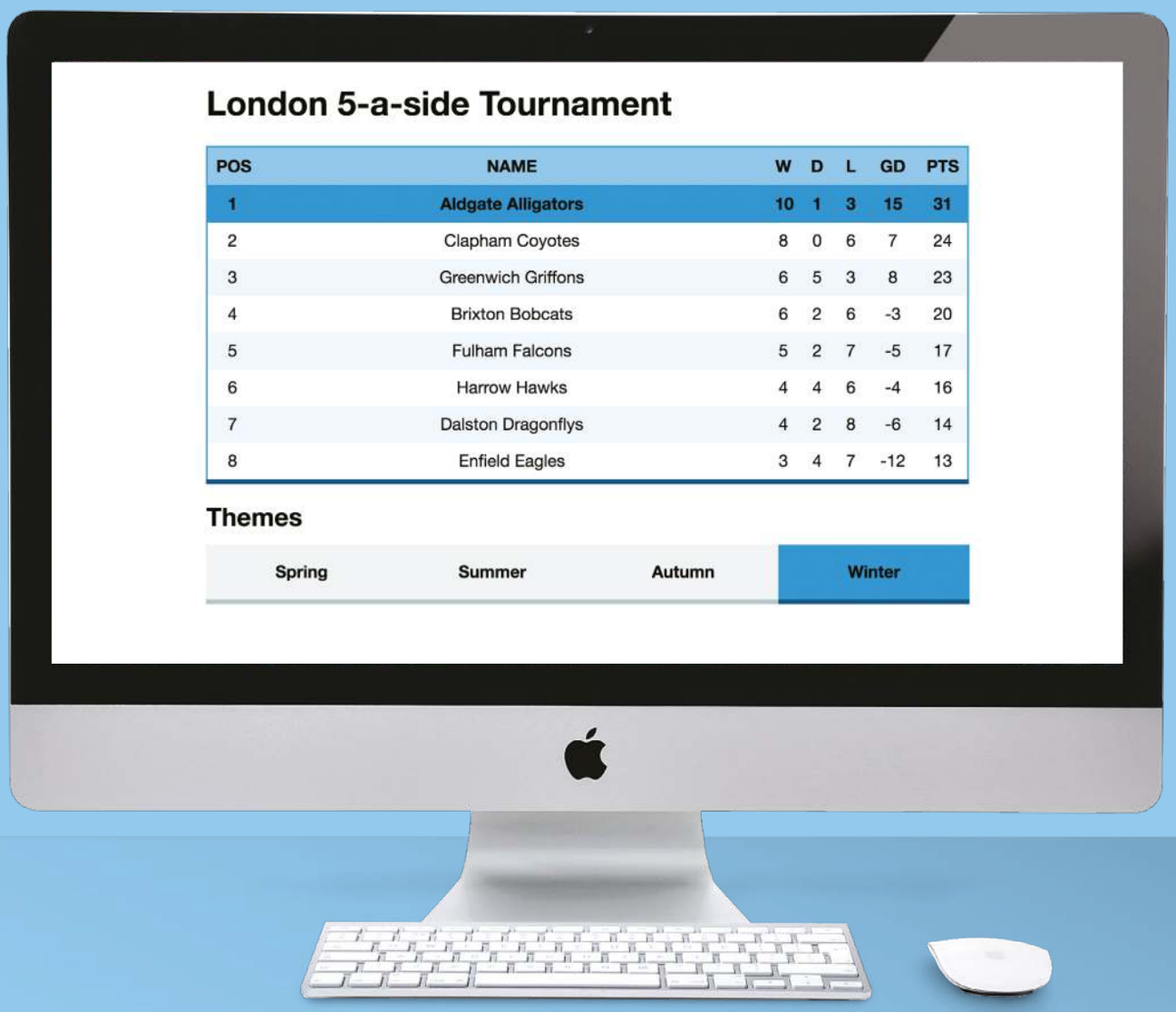
Print edition available at www.myfavouritemagazines.co.uk

Digital edition available for iOS and Android

facebook.com/PhotoshopCreative twitter.com/PshopCreative

Style React apps with Fela

Use the JavaScript-driven tool Fela to allow components to instantly update their look based on state using the latest styling techniques



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

Styling a component-based application can get difficult. While build tools such as Webpack can digest and create stylesheets made for each section, they remain unaware of the application as a whole. Changes in state have to be toggled with class names, which are prone to misuse and issues with specificity. The results are bloated classes, repeated properties and a set of styles that do not play well together.

For application components to be more easily reused, they should know about the bigger picture. Styles and logic should be aware and respond to one another. One way to tackle this is to use CSS-in-JS tools such as Fela.

On its own, Fela is a way of taking objects that define styles and convert them to atomic CSS classes. The styles it applies are based on values passed to functions known as 'rules' that create style objects. By combining with the 'react-fela' package, we can adjust styles based on React's state within an application.

In this tutorial, we will be using Fela to style a league table using some sample data. We can then use that data to highlight important information, such as the title-holders, and in response to a site-wide theme. By the end we will have lots of easily portable components that can be used across applications with no issues.

1. Install dependencies

This project makes use of a module called 'create-react-app', which stamps out an application that's already been set up. We need to install Fela alongside everything else 'create-react-app' requires. To do this we will use Yarn, which should be installed before the project gets underway.

Open a command line window, find the tutorial files and run the following. A window will open displaying our application:

```
> yarn install
> yarn start
```

2. Create a renderer

Fela generates styles by passing rules to a renderer. The job of the renderer is to compute the styles and generate the necessary classes. The 'react-fela' package supplies a <Provider> component, which enables us to use the renderer throughout the application.

Create a renderer inside 'App.js' and then pass it to the Provider as the outermost component in the render function.

```
const renderer = createRenderer({});
[…]
<Provider renderer={renderer}>
  […]
</Provider>
```

3. Supply some plugins

On its own, Fela can compute basic styles and apply them to components. We can enhance this behaviour by piping the data through plugins.

Update the renderer to include a couple of useful plugins. The first will automatically apply vendor prefixes, while the second takes those prefixes and makes them their own properties.

```
const renderer = createRenderer({
  plugins: [prefixer(), fallbackValue()]
});
```

4. Add a theme provider

While not required, it is likely an application will have some kind of theme across its components. Information about this theme can be passed to the <ThemeProvider> component, which uses React context to pass values to styled components, in 'react-fela'.

Inside the main <Provider>, add a <ThemeProvider> and then pass it some information about a theme. Also, add a <Layout> component that we will use in the next step.

```
<ThemeProvider theme={{
  mainColor: this.state.mainColor
}}>
  <Layout>
    […]
  </Layout>
</ThemeProvider>
```

5. Define a basic layout

Fela styles come from functions called 'rules' that return a style object, which are plain JavaScript objects with style properties to apply. The rules can calculate and produce that object in whatever way is required by the application.

Open up 'Layout.js' and style a layout for the application. Once that is done, pass the styles to 'createComponent' to make the <Layout> component in the previous step.

```
const Layout = () => ({
  fontFamily: "Helvetica,
  Arial, sans-serif",
  fontSize: "18px",
  padding: "0 1em"
});
export default createComponent(Layout);
```

6. Make the layout responsive

Much like regular CSS, Fela components can also be responsive. Nested objects define related styles such as media queries, but they are limited to styling only that component.

After the padding, add a media query object to limit the layout to 850px once the screen is large enough.

```
"@media (min-width: 600px)": {
  margin: "0 auto",
  maxWidth: "850px",
  width: "80%"
}
```

7. Show the league table

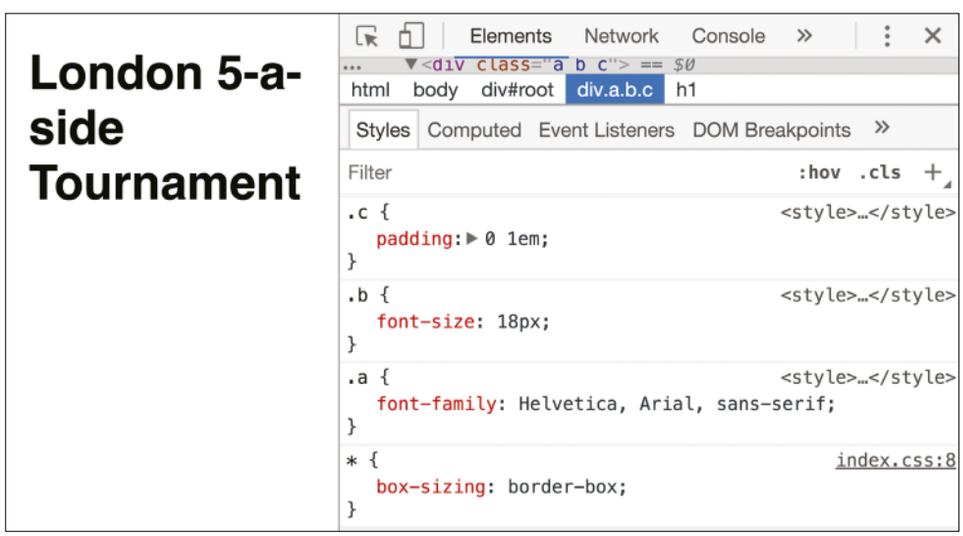
This application will show a league table for a football team. The basic structure is a <table> showing team names, their positions and some stats depending on the screen space available.

Passing 'undefined'

By setting the value of a property as 'undefined' – on its own or as a string – that property is removed. This can be useful when styling directly using optional props.

Top
The <Provider> component is required for Fela to be able to process classes. It needs to wrap all components that need to be styled

Right
Fela generates individual classes for each property defined. To save space, these are reused across components with the same properties



To kick things off, add a <LeagueTable> component inside <Layout> in 'App.js'. Pass it some generated <Team> components that we will style later.

```
<LeagueTable>
  {getTeams(this.state.teams)}
</LeagueTable>
```

8. Define table styles

Firstly, we need to define some base styles for the table. This will help clearly display the information when we add the teams later on.

Fela accepts all the usual CSS properties as 'camelCase' key names. The values can be numbers or strings. Functions like 'calc()' will still work as expected. Open up 'LeagueTable.js' and apply the following styles:

```
const LeagueTableStyles = () => ({
  background: "white",
  borderBottom: 0,
  borderCollapse: "collapse",
  marginBottom: "calc(5px + 1em)",
  tableLayout: "fixed",
  textAlign: "center",
  transition: "all 0.5s ease-out",
  width: "100%"
});
```

London 5-a-side Tournament

Pos	Name	W	D	L	GD	Pts
1	Fulham Falcons	8	5	1	17	29
2	Dalston Dragonflys	7	3	4	12	24
3	Greenwich Griffons	7	2	5	4	23
4	Clapham Coyotes	5	4	5	-1	19
5	Aldgate Alligators	5	3	6	-2	18
6	Harrow Hawks	5	2	7	-2	17
7	Brixton Bobcats	5	1	8	-7	16
8	Enfield Eagles	2	4	8	-21	10

9. Calculate border style

The main benefit of Fela is that it can respond to the state within an application. One of the state points is the theme we passed through in an earlier step.

Theme data is passed to each rule inside <ThemeProvider> as a prop. Use template literals to generate strings for the style object. We can use a package called Polished to alter the colours for us.

```
const LeagueTableStyles = ({
 theme: { mainColor } }) => ({
  […]
  border: `0.1em solid ${mainColor}`,
  boxShadow: `0 5px ${darken(0.2,
mainColor)}`,
});
```

10. Style the headings

The default stylings make the headings look like part of the main table. We need to colour the background to separate the header from the data.

Start creating a style object within 'HeadingStyles'. As we won't know the theme colour in advance, we need to calculate a lighter or darker colour for it before returning the style object.

```
const HeadingStyles = ({ hideSmall,
  theme: { mainColor }, width }) => {
  let background =
  lighten(0.2, mainColor);
  if (getLuminance(background) > 0.5) {
  background = darken(0.2, mainColor);
  }
});
```

11. Return a style object

With the background defined, we can start creating the style based on that value. As we are returning an object, we can use shorthand, function return values and ternary operators to dynamically set property values inline.

Return a style object from 'HeadingStyles'. A value for 'width' will be defined in the next step.

```
const style = {
  background,
  color: readableColor(background),
  textTransform: "uppercase",
  transition: "all 0.5s ease-out",
  width: width ? `${width}em` : undefined
};
return style;
```

12. Set width based on props

We used a 'width' prop in the previous step. The styled component gets passed a prop like any other component, which is then passed as a parameter to the rule to help calculate styles.

Edit the markup in 'LeagueTableComponent' and set widths for all heading cells apart from 'Name'.

```
<Heading width={3}>Pos</Heading>
<Heading width={2}>W</Heading>
<Heading width={2}>D</Heading>
<Heading width={2}>L</Heading>
<Heading width={3}>GD</Heading>
<Heading width={3}>Pts</Heading>
```

13. Increase cell padding

At the moment the cells are feeling flat. By adding some padding, we can open things up a little. With CSS, we could write a selector that covers both <th> and <td> elements. In Fela, we can combine rules instead.

Create and export a rule to add padding. We can then use the 'combineRules' method to replace the style for <Heading> to add the padding in.

The second argument to 'createComponent' defines which element to style.

```
export const cellPadding = () => ({
  padding: "0.5em"
});
[…]
const Heading = createComponent(
  combineRules(cellPadding,
  hideSmall, HeadingStyles), "th");
```

14. Hide columns on small screens

Similarly, we want to hide some statistics on smaller screens where space is a premium. Much like with <Layout> we can apply media queries within separate rules to combine as required.

Composing styles

The "combineRules" helper method can take multiple rule definitions and combine them into a larger one. Creating several smaller rules for common styles can save on repetition.

London 5-a-side Tournament

POS	NAME	W	D	L	GD	PTS
1	Clapham Coyotes	8	4	2	11	28
2	Aldgate Alligators	7	2	5	6	23
3	Enfield Eagles	6	5	3	4	23
4	Fulham Falcons	5	6	3	4	21
5	Brixton Bobcats	5	2	7	-1	17
6	Dalston Dragonflys	3	6	5	-4	15
7	Greenwich Griffons	4	3	7	-7	15
8	Harrow Hawks	2	4	8	-13	10

Top
As we define widths for every other one, the "Name" column will take up any remaining space. This keeps things flexible across all screen sizes.

Right
Make sure "hideSmall" is applied to the corresponding cells within <Team>, else the headings will not line up correctly.

London 5-a-side Tournament

POS	NAME	PTS				
1	Dalston…	10	1	3	11	31
2	Clapha…	8	4	2	11	28
3	Fulham…	7	3	4	6	24
4	Green…	5	3	6	4	18
5	Harrow…	4	6	4	4	18
6	Brixton …	4	4	6	-8	16
7	Enfield …	2	5	7	-13	11
8	Aldgate…	2	2	10	-15	8

Using Polished with Fela

Tools like SASS and PostCSS helped change the way we compose styles.

While writing CSS in JavaScript has its benefits for component-based applications, we lose access to the helper functions and mixins that these tools provided to streamline the process.

Polished is a set of helper methods designed to output a range of CSS properties and values. By taking dynamic values as arguments, it removes much of the string interpolation required when writing CSS-in-JS techniques.

There are multiple functions designed to manipulate colours, such as 'desaturate' and 'opacify'. These can take colour values in a variety of formats, understand them and perform alterations before passing them back. This means they can be curried together to perform the exact adjustments required.

You can find out more about Polished at: ***polished.js.org***

Create another rule that will hide any styled component that is passed a 'hideSmall' prop. Add that prop to the win, draw, loss and goal difference headings in the render method.

```
export const hideSmall =
  ({ hideSmall }) => ({
  display: hideSmall ? "none":undefined,
  "@media (min-width: 600px)": {
    display: "table-cell" }
});
[…]
<Heading hideSmall width={2}>W</Heading>
<Heading hideSmall width={2}>D</Heading>
<Heading hideSmall width={2}>L</Heading>
<Heading hideSmall width={3}>GD
 </Heading>
```

15. Alternate row colours

We need to separate each row from one another to make it easier to read. One way of doing that is to alternate the colour of rows. As we pass the position to the <Team> component, we can use this as a reference.

Open up 'Team.js' and add a base style object to TeamStyles. On every team in an odd position, we use a dimmed version of the theme colour as the background.

```
let style = {
  background: pos % 2 === 1 ?
   transparentize(0.9, mainColor):
   undefined,
  transition: "all 0.5s ease-out"
};
```

16. Highlight first place

The champions of the league should get a special highlight in our table. We can add extra properties to the returned style object just for the team in first place.

When styling the team in position 1, use object destructuring to copy the original style object and make some slight changes. Use 'readableColor' from Polished to make sure the text remains legible.

```
if (pos === 1) {
  style = {
    ...style,
    background: mainColor,
    color: readableColor(mainColor),
    fontWeight: "bold"
  };
}
```

17. Mix in cell styles

As we exported the 'cellPadding' and 'hideSmall' rules we made earlier, they can be imported inside 'Team.js' and made use of inside the main table cells. If any changes need to be made to cell padding, for example, it means that only needs to happen in one place.

Import these two rules and use them in the definition of the <Cell> component.

```
import { cellPadding, hideSmall }
 from "./LeagueTable";
[...]
const Cell = createComponent(
  combineRules(cellPadding, hideSmall,
   CellStyles), "td");
```

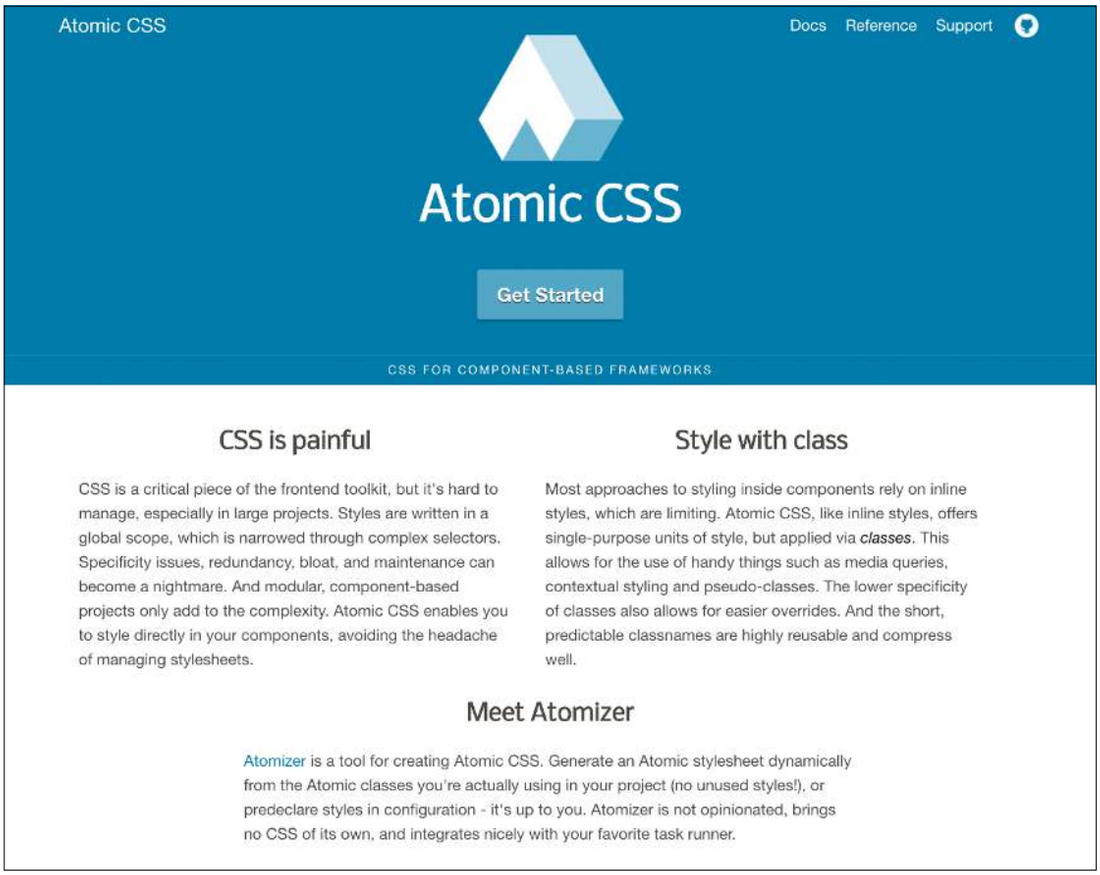
18. Truncate long team names

On very narrow displays, the comparatively long team names will start to break out of their cells. To avoid that, truncate these values to take up only the space available.

CSS provides the 'text-overflow' property for us to do that, which Polish provides a method for. Add the style for <Cell> components that have the 'truncate' prop.

```
if (truncate) {
  style = {
  ...ellipsis("100%")
  };
}
```

POS	NAME	PTS
1	Harrow Hawks	25
2	Fulham Falcons	23
3	Greenwich Griffons	21
4	Clapham Coyotes	18
5	Dalston Dragonflys	17
6	Enfield Eagles	17
7	Aldgate Alligators	16



What is Atomic CSS?

When a component is styled using Fela, the resulting class names are a seemingly random collection of letters rather than anything to do with the component itself. This is because Fela uses an approach called 'atomic CSS'.

Atomic (or 'functional') CSS creates classes for each different style and applies these to each element individually instead of using a specific ID or class name to define how each element should appear.

These are usually generated by a tool like Fela, but they can also be created by hand. The advantage of using a tool to generate them is that it guarantees all the classes it generates are being used on the site.

The end result is a smaller set of styles that are predictable, reusable and avoid common specificity issues that arise from deeply nested selectors. However, this is at the expense of semantic classes that can cascade styles to avoid repetition. It is best used with a component-based approach such as React.

19. Add theme buttons

We need a way to change the theme across the page. The <ThemeProvider> we added earlier takes 'mainColor' from application state and then passes it as a theme value. By changing state, we can change the theme.

Back in 'App.js', add some buttons to switch the theme to the end of the <Layout> component.

```
<ButtonGroup>
  {SEASONS.map(season => (
  <Button active={this.state.mainColor
  === season.color}
  key={season.name}
  value={season.color}
  onClick={this.setMainColor}>
  {season.name}
  </Button>
  ))}</ButtonGroup>
```

20. Add hover and focus styles

These buttons already have some styling applied, but do not yet react to being hovered or focused on.

Rules can generate style objects using regular JavaScript. This means we can apply a hover and a focus style in a loop to save duplication.

Add the following just before returning the style object inside 'Button.js':

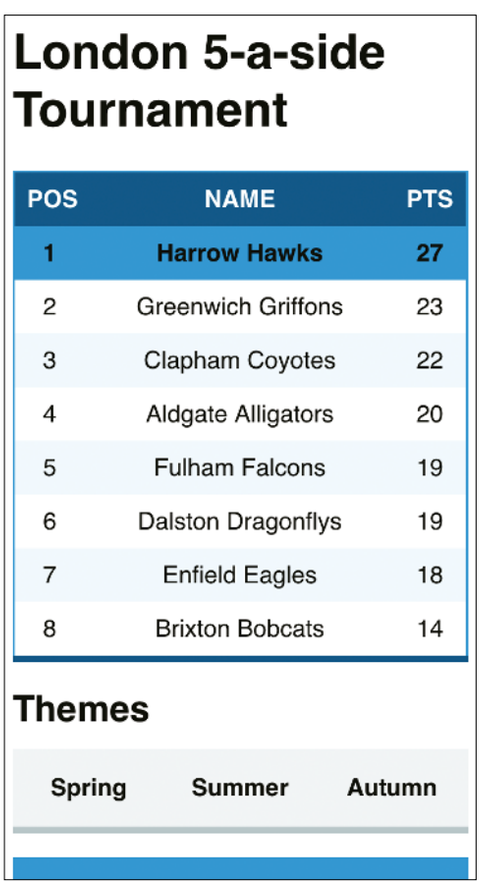
```
[":hover", ":focus"].forEach(
  value => (style[value] = {
  background: darken(0.1,
  buttonColor) }) );
```

21. Pass through props

Even though we passed an 'onClick' handler to the button, clicking it has no effect. This is because Fela will swallow all props by default to use for styling. A third argument to 'createComponent' will pass through any named props to the underlying component.

Update the exported Button component to allow the click event and the button value through.

```
export default createComponent(Button,
 "button", ["onClick", "value"]);
```



Special offer for readers in *North America*

6 issues FREE

When you subscribe*

FREE resource downloads every issue

"The only magazine you need to design and develop stunning websites"

Order hotline **+44 (0) 344 848 2852**

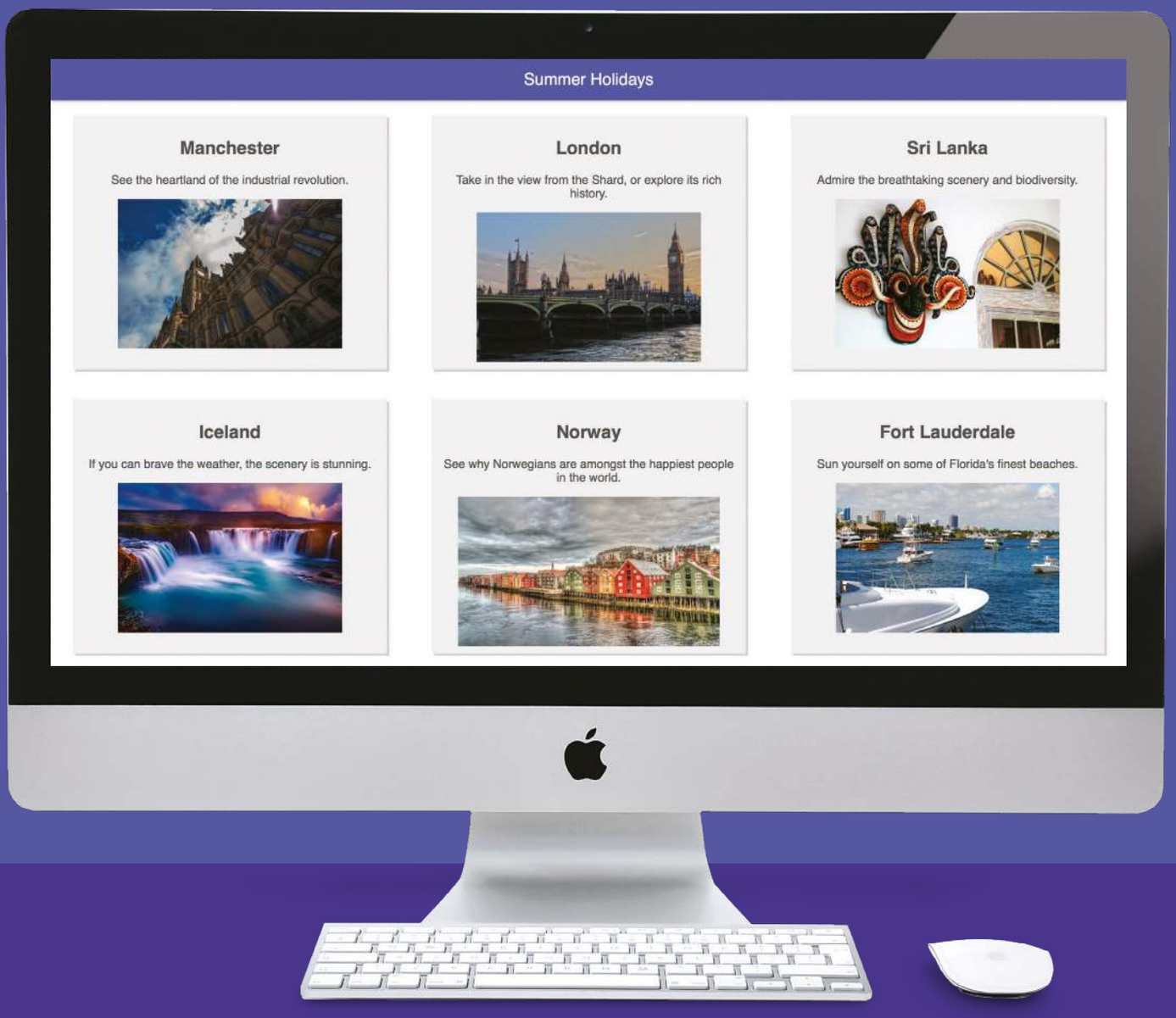
Online at **myfavouritemagazines.co.uk/WEDPQ17**

Expires 31 Aug 2018

***Terms and conditions** This is a US subscription offer. 6 free issues refers to the USA newsstand price of $14.49 for 13 issues at $188.37, compared with $103 for a subscription. You will receive 13 issues in a year. You can write to us or call us to cancel your subscription within 14 days of purchase. 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 upon request. UK calls will cost the same as other standard fixed line numbers (starting 01 or 02) are included as part of any inclusive or free minutes allowances (if offered by your phone tariff). For full terms and conditions please visit: bit.ly/magtandc Offer ends August 31 2018.

Build custom web components

Use the compiler stencil.js to produce vanilla web components that will work with any framework



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

The ability to create sites and apps using components makes modern web development a fairly slick experience, and is the backbone of almost every modern frontend framework. Web Components offer a standards-driven approach to doing this, meaning you're building vanilla code for the web and don't depend on any specific framework. You might already be familiar with the web component standards: shadow DOM, HTML templates, HTML imports and custom elements. It's the last of these which is the real crux of web components, since ultimately what you're trying to do is create your own HTML tags which encapsulate markup and behaviour you define.

While you could certainly get started building web components with nothing but a text editor, it can be a bit fiddly. Luckily, like everything on the web, there are various tools and libraries to help you. Google Polymer is one which has been around for a while. A relatively new kid on the block is Stencil.js, from the team behind the Ionic framework. Stencil is a web component compiler, which enables you to use TypeScript and JSX to create vanilla web components, which will work anywhere (along with apps that use them).

1. Get the Stencil starter project

To get started with Stencil.js, you'll need Node and npm. You probably have these by now, but if you don't, head over to **nodejs.org/** to grab them. Once you're done, the easiest way to get started is to clone Stencil's starter project Git Hub repository.

```
git clone https://github.com/ionic-team/
stencil-app-starter stencil-tutorial
cd stencil-tutorial
git remote rm origin
```

2. Install dependencies

Next, we'll use npm to grab all the dependencies for the starter project. This will grab the core Stencil files, plus the latest router, dev server, and some other utilities. Once this is done, fire up the starter project on a local server, which is accessible at http://localhost:3333/

```
npm install
npm start
```

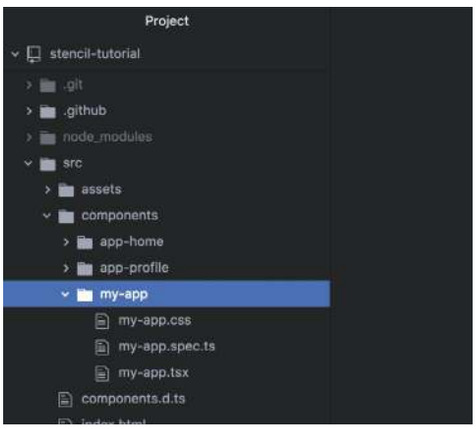
3. Live updates

If you take a look at the console, you will notice that Stencil is watching for changes to the source files. If you navigate to the 'src' directory and edit the <title> element in 'inded.html', you'll see the update straight away in the browser's title bar.

```
<title>Summer Holidays</title>
```

4. Components in Stencil

Now let's take a brief tour of how components are set up in Stencil. If you look in the 'src/components' folder, you'll see several different components shipping with the starter project. The core of each component is a TSX file. Stencil uses an amalgamation of TypeScript and JSX to define components. You'll also see a CSS file for each component describing its styling. The TS files that ship with each component are unit tests written in TypeScript. We won't worry too much about these for this tutorial (but you should be unit testing!)



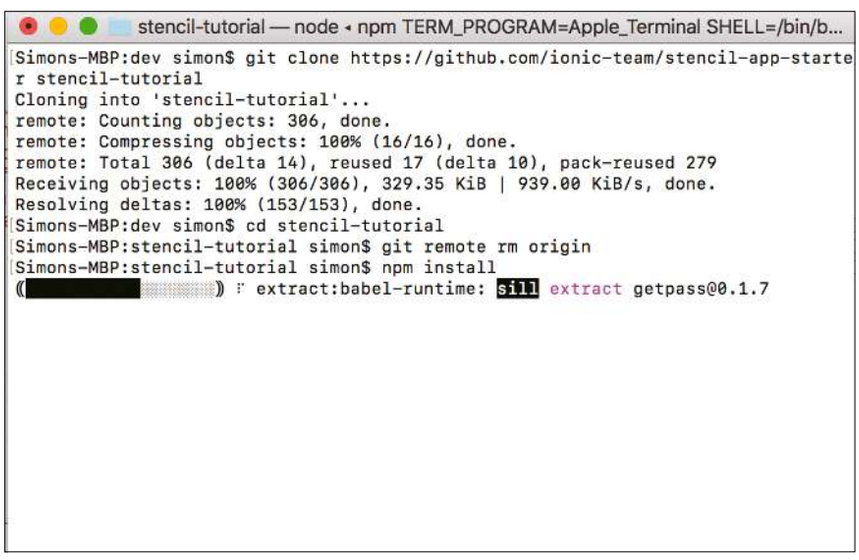
5. TypeScript and JSX

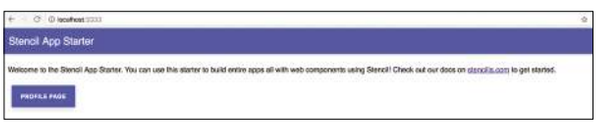
You might have used one or both of these before. TypeScript is probably most commonly associated with Angular, and JSX with React. TypeScript is a superset of JavaScript, which adds static typing. JSX is an extension of JavaScript which enables us to embed markup within it. If you take a look at the components, you'll notice that this is used in a 'render()' method in with TSX files, which tells Stencil what to display for that component. For example, in 'src/components/my-app/my-app.tsx':

```
export class MyApp {
  render() {
  return (
  <div>
  …
  </div>
  );
  }
}
```

Under the hood of web components

Stencil provides a layer of abstraction away from the web component specifications themselves, but it's useful to know what's going on behind the scenes. There are several parts to the web component specifications. HTML templates enable us to create reusable fragments of markup which can be instantiated when needed. HTML imports enable us to modularise markup and import one HTML file into another. Shadow DOM enables us to encapsulate the DOM tree for a component so that it's invisible to whatever is using it and acts just as a single element. And custom elements brings all of these together, introducing support for us to define whatever HTML tags we like. Browser support is improving, but isn't perfect, so Stencil fills in some of this for us when it builds for production.





Left
Clone the Stencil starter project and use npm to install dependencies

Top
The stencil starter project ships with some basic components and can be spun up on a development server

6. Custom elements in use

Remember that the reason we're doing all this is to use custom HTML tags, which encapsulate our own functionality. Take a look again at 'index.html'. Notice how it uses the <my-app> element. This obviously isn't a standard HTML element. It's actually created by Stencil from the 'my-app.tsx' file that we just looked at.

```
<my-app></my-app>
```

7. Modify components

The best way to get a feel for how JSX works is to play around with some of the existing components. Let's modify 'my-app.tsx' to display some different header text, and 'app-home.tsx' to display a different message. Also notice how 'my-app' is making use of 'app-home' and 'app-profile' as child components within its markup. When we make changes to components, we'll see them render instantly in the browser provided that the development server is running.

```
my-app.tsx:
  <h1>Summer Holidays</h1>
app-home.tsx:
  <p>
  Find your perfect summer getaway!
  </p>
```

8. Decorators

Okay, so you're probably now getting a feel for how the 'render()' function works. There's another part to these starter components, which you'll notice: decorators, which start with '@.' This '@Component' decorator provides metadata on the component to Stencil, such as which HTML tag to use and where to find its associated CSS. In 'app-profile.tsx', there's an '@Prop' decorator that defines properties, which can be passed to the component when it is used. We can use these properties as variables in curly brackets. Let's have 'my-app' accept a 'page-header' property and use it as its <h1> text.

```
import { Component, Prop } from '@stencil/
core';
…
export class MyApp {
  @Prop() pageHeader: string;
  render() {
  return (
  <div>
  <header>
  <h1>{this.pageHeader}</h1>
  </header>
  …
  </div>
  );
  }
}
```

9. Pass a property to a component

The 'my-app' component now accepts a 'pageHeader' property, which we've specified will be a string. Using it is easy. Note that when we use this variable as the property of the <my-app> element in an HTML file, camelCase is converted to hyphen-separated (but when using it in JSX, we'd stick with camelCase).

```
<my-app page-header="Summer Holidays"></
my-app>
```

10. Create a new component

Hopefully, you should now be getting a feel for how components, decorators, and the 'render()' function work with Stencil. Now let's think about creating our own components. Stencil doesn't currently have a CLI to generate component stubs, so we will have to add the files manually. Create a 'holiday-listing' folder, and in it, create 'holiday-listing.tsx' and 'holiday-listing.css'. The TypeScript file looks like this:

```
import { Component, Prop } from '@stencil/
core';
@Component({
  tag: 'holiday-listing',
  styleUrl: 'holiday-listing.css'
})
export class HolidayListing {
  render() {
  return (
  <div class='holiday-listing'>
 Coming soon!
  </div>
  );
  }
}
```

11. Properties for the new component

Let's set up our new component so that its parent can specify a title and text for it to display. To do this, we need to create two string properties. We'll also update the markup in the 'render()' function to use these properties.

```
export class HolidayListing {
  @Prop() title: string;
  @Prop() content: string;
  render() {
  return (
  <div class='holiday-listing'>
  <h2>{this.title}</h2>
  <p>
  {this.content}
  </p>
  </div>
  );
  }
}
```

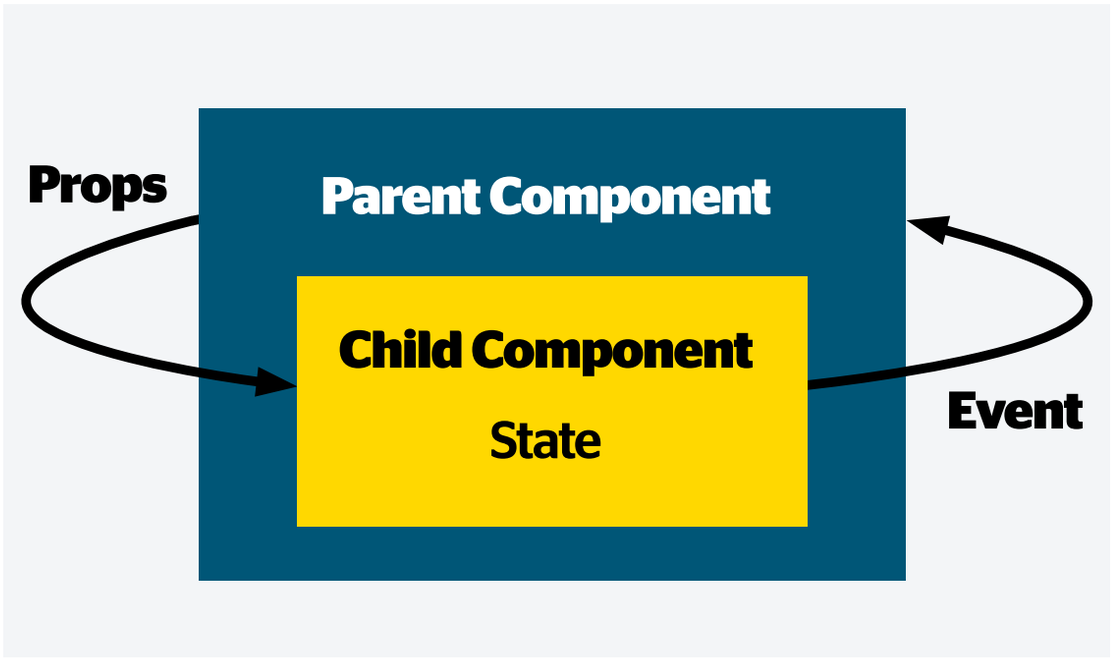
12. Style the component

You'll notice that earlier in the '@Component' decorator we specified 'holiday-listing.css' as the stylesheet for the component. Let's populate it to give our component a border and a background. Applying styling within a component is a common pattern these days, although

Component updates

Stencil components update when their props or state are changed. This will be very familiar if you've used React, and makes it relatively straightforward to have changes to data in your app reflect instantly (and efficiently) within every component.

Right
We've created our first Stencil component which can be instantiated via a custom element if we so wish



Events
We've looked at how to pass data from a parent component to a child via props, but a common scenario is to need to pass data back to the parent component - a button press, for example. Since props are in most cases immutable within a component, the approach you take to this is the same as you'd use in React. The solution is to use an event model. The actual implementation of this looks slightly different to React, though. With Stencil, you can use the @Emit decorator to specify and event that a component emits, then use the @Listen decorator in another component to respond to it.

we may also want to think about parametrising some of its appearance so that it can be themed by a parent site or component.

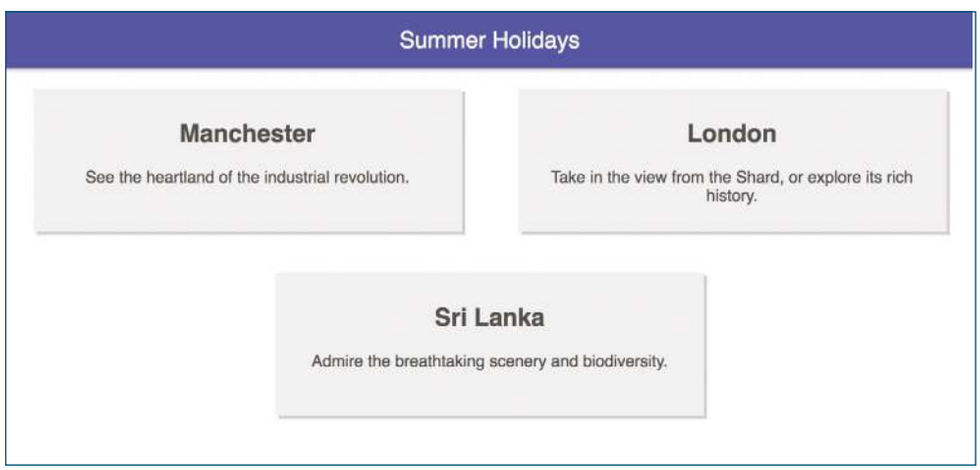
```
.holiday-listing {
  padding: 10px;
  width: 400px;
  height: 320px;
  margin: 20px;
  box-shadow: 3px 3px #cccccc;
  text-align: center;
  background: #ededed;
  color: #555555; }
```

13. Get ready to use the component

We can now use the component using the custom tag that we've defined. We could just code the properties in directly, but let's first discover some more of the power of TypeScript and JSX.

We'll create an array of holidays and use that to iteratively generate components. In this particular case, we'll use the '@State' decorator to store the array. If 'prop' is for data that's been passed down from a parent to a child component, then 'state' is for managing data internal to a component. This is the same method as employed in React, if you have used that.

```
@State() holidays: [string,string][];
  constructor()
  {
  this.holidays = [
  ["Manchester","See the heartland of the
industrial revolution."],
  ["Brighton","Enjoy a pint on the beach,
but watch out for seagulls!"],
  ["Sri Lanka","Admire the breathtaking
scenery and biodiversity."]
  ]   }
```



14. Use the component

So how do we use this array? Remember that within JSX markup we can evaluate expressions with curly brackets. So far we've used this to evaluate individual variables, but we can evaluate more complex expressions this way, too. Let's use the 'map()' function to return a <holiday-listing> element for every item in the 'holidays' array.

```
  <div class="flex">
  {
  this.holidays.map((o) => <holiday-listing
title={o[0]} content={o[1]}></holiday-
listing>)
  }
  </div>
```

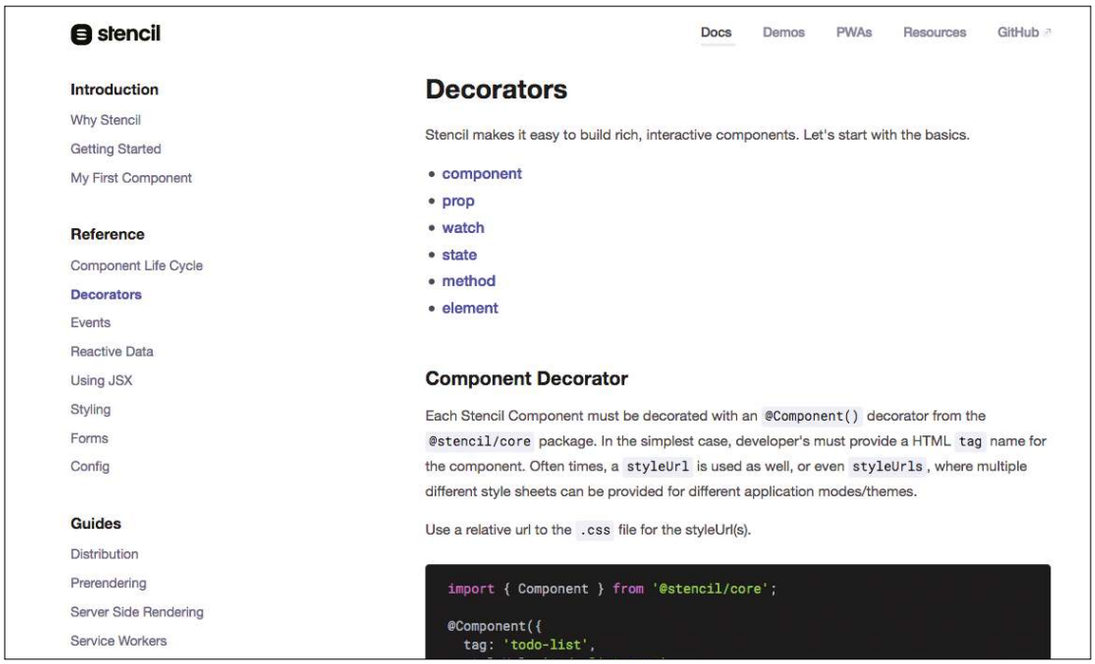
15. Place static assets

We can place static assets like images in the 'assets' folder of our projects. Let's extend the 'holidays' array to include an image filename for each holiday that we're featuring, add this as an additional property to the 'holiday-listing' TypeScript, and extend the 'map()' function to pass it to the custom element. This is also a good use case for default property values, since we can specify a placeholder image if one hasn't been provided.

```
  @State() holidays: [string,string,string]
[];
…
  <div class="flex">
  {
  this.holidays.map((o) =>
  <holiday-listing
  title={o[0]}
  content={o[1]}
  pic={o[2] != "" ? o[2] : "missing.jpg"}>
  </holiday-listing>)
  }
  </div>
```

16. Images in the components

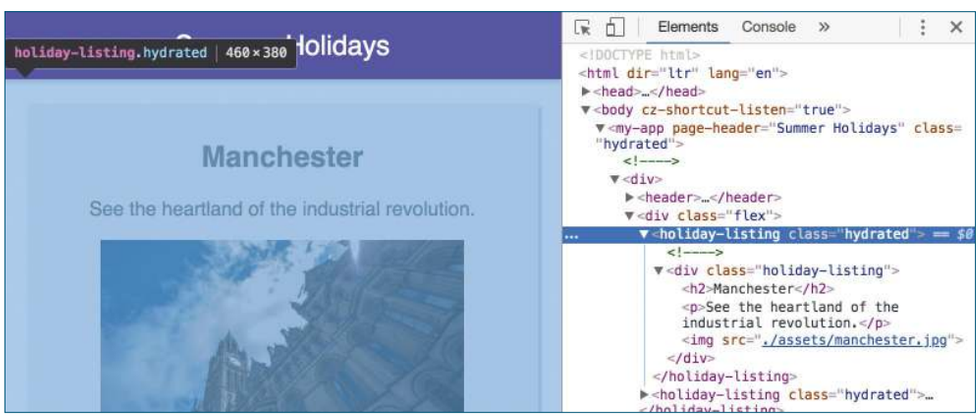
To take advantage of the new data we're passing, of



Decorators

We've used a couple of decorators, @Component, @Prop and @State, in our tutorial, but there are others which Stencil uses that are worth being aware of. We've already talked briefly about @Event and @Listen. Here are the others:

- **@Watch** listens for an update to a property, and passes its associated method to the old and new property value when triggered.
- **@Method** is used to expose methods as part of a public API. Methods annotated this way can be called directly from outside a component.
- **@Element** is used to refer to the host HTML element inside its own component, which can be useful to modify its properties on the fly.



course, we need to update the component to define it as a prop. We can then use it in an image tag. Notice that once again we can evaluate expressions in JSX to dynamically construct the relative path from the filename.

```
  @Prop() pic: string = "missing.jpg";
…
  <img src={"./assets/" + this.pic}/>
```

17. Configuring for distribution

So far we've only used our components within the Stencil starter project. But the real power of Stencil is being able to use them anywhere. We could be building an Angular, React, Vue, or other project, and still take advantage of them. Let's take a look at using one in a vanilla HTML file which isn't built and hosted on a local server. Add the following to the 'stencil.config.js' file:

```
exports.config = {
  namespace: 'holiday',
  outputTargets: [
  {
  type: 'dist'
  },
  {
  type: 'www'
  }
  ] };
```

18. Components to export

Now let's tell Stencil what we want to package for distribution. The 'holiday-listing' component looks useful, so we'll use that. Add the following within the curly brackets of your 'package.json' file:

```
  "main": "dist/holiday-listing.js",
  "types": "dist/types/components/holiday-listing/holiday-listing.d.ts",
  "collection": "dist/collection/collection-manifest.json",
  "files": [
    "dist/"
  ],
```

19. Building for production

Now we're ready to create a build. By default, when you're performing dev builds, Stencil won't include all the polyfills you might need for browser support. Therefore, we'll use a parameter to tell it to build as if it's for production. The other thing you could do at this point is publish to npm, but we won't do that for now.

```
npm run dev --prod
```

20. Use in a vanilla page

Finally, let's do something really simple. Create a straightforward HTML stub, and import the 'holiday.js' file Stencil generated. Then, we can use the <holiday-listing> element without any hassle! We'll need to create an assets folder and place the image(s) we want to use in it, since our component will be looking at a relative path.

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Vanilla Holidays</title>
    <script src="dist/holiday.js"></script>
    <holiday-listing title="Brighton"
content="Get the vanilla Brighton experience by importing a web component without using any framework." pic="brighton.jpg"></holiday-listing>
  </head>
  <body>
  </body>
</html>
```

21. Next steps

There you go! Hopefully you've seen some of the power of Stencil and web components. A good thing to try is conceptualising and creating a 'useful' component. See if you can get it working, allow it to inherit some (but not all) styling, and get it published to npm. This is where the specification really comes into its own, since it enables the creation of a truly framework-agnostic ecosystem.

HOSTING LISTINGS

Featured host: **Netcetera**

netcetera.co.uk
03330 439780

About us

Formed in 1996, Netcetera is one of Europe's leading web hosting service providers, with customers in over 75 countries worldwide.

As the premier provider of data centre colocation, cloud hosting, dedicated servers and managed web hosting services in the UK, Netcetera offers an array of services designed to more effectively manage IT infrastructures. A state-of-the-art data centre environment enables Netcetera to offer your business enterprise-level colocation and hosted solutions. Providing an unmatched value for your budget is the driving force behind our customer and managed infrastructure services. From single server to fully customised data centre suites, we focus on the IT solutions you need.

What we offer

- **Managed hosting –** A full range of solutions for a cost-effective, reliable, secure host.
- **Cloud hosting –** Linux, Windows, Hybrid and Private Cloud Solutions with support and scalability features.
- **Data centre colocation –** Single server through to full racks with FREE setup and a generous bandwidth.
- **Dedicated servers –** From QuadCore up to Smart Servers with quick setup and fully customisable.

5 tips from the pros

1. Reliability, trust & support

Reliability is a major factor when it comes to choosing a hosting partner. Netcetera guarantees 100 per cent uptime, multiple internet routes with the ability to handle DDOS attacks, ensuring your site doesn't go down when you need it.

2. Secure and dependable

Netcetera prides itself on offering its clients a secure environment. It is accredited with ISO 27001 for security along with the options of configurable secure rackspace available in various configurations.

3. 24/7 technical support

Netcetera has a committed team of knowledgeable staff available 24/7 to provide you with assistance when you need it most. Our people make sure you are happy and your problems are resolved as quickly as possible.

4. Value for money

We do not claim to be the cheapest service available, but we do claim to offer excellent value for money. We also provide a price match on a like-for-like basis, as well as a price guarantee for your length of service.

5. Eco-friendly

Netcetera's environmental commitment is backed by use of eco-cooling and hydroelectric power. This makes Netcetera one of the greenest data centres in Europe.

Testimonials

Roy T

"I have always had great service from Netcetera. Their technical support is second to none. My issues have always been resolved very quickly."

Suzy B

"We have several servers from Netcetera and their network connectivity is top-notch, with great uptime and speed is never an issue. Tech support is knowledgeable and quick in replying. We would highly recommend Netcetera."

Steve B

"We put several racks into Netcetera, basically a complete corporate backend. They could not have been more professional, helpful, responsive or friendly. All the team were an absolute pleasure to deal with, and nothing was too much trouble, so they matched our requirements 100 per cent."

Supreme hosting

CWCS Supreme Hosting. Supreme Support. Managed Hosting

cwcs.co.uk
0800 1777 000

CWCS Managed Hosting is the UK's leading hosting specialist. They offer a fully comprehensive range of hosting products, services and support. Their highly trained staff are not only hosting experts, they're also committed to delivering a great customer experience and are passionate about what they do.

- Colocation hosting
- VPS
- 100 per cent network uptime

UK-based hosting

CYBERHOSTPRO

cyberhostpro.com
0845 5279 345

Cyber Host Pro are committed to providing the best cloud server hosting in the UK; they are obsessed with automation. If you're looking for a hosting provider who will provide you with the quality you need to help your business grow, then look no further than Cyber Host Pro.

- Cloud VPS servers
- Reseller hosting
- Dedicated servers

Cluster web hosting

fasthosts

fasthosts.co.uk
0808 1686 777

UK-based and operating 24/7 from dedicated UK data centres. Fasthosts keep over one million domains running smoothly and safely each day. Services can be self-managed through the Fasthosts Control Panel.

- Dedicated servers
- Cloud servers
- Hosted email

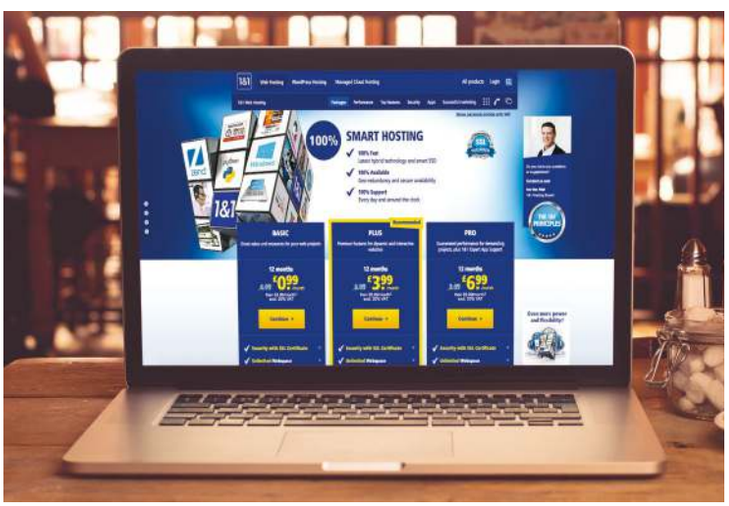
Budget hosting

HETZNER ONLINE

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 nationally and internationally.

- Dedicated/shared hosting
- Colocation racks
- SSL certificates



All-inclusive hosting

1&1

1and1.co.uk
0333 336 5509

1&1 Internet is a leading hosting provider that enables businesses, developers and IT pros to succeed online. Established in 1988, 1&1 now operates across ten countries. With a comprehensive range of high-performance and affordable products, 1&1 offers everything from simple domain registration to award-winning website building tools, eCommerce packages and powerful cloud servers.

- Easy domain registration
- Professional eShops
- High-performance servers

SSD web hosting

bargainhost

bargainhost.co.uk
0843 289 2681

Since 2001, Bargain Host have campaigned to offer the lowest possible priced hosting in the UK. They have achieved this goal successfully and built up a large client database, which includes many repeat customers. They have also won several awards for providing an outstanding hosting service.

- Shared hosting
- Cloud servers
- Domain names

Value Linux hosting

PATCHMAN WEB HOSTING

patchman-hosting.co.uk
01642 424 237

Linux hosting is a great solution for home users, business users and web designers looking for cost-effective and powerful hosting. Whether you are building a single-page portfolio, or you are running a database-driven eCommerce website, there is a Linux hosting solution for you.

- Student hosting deals
- Site designer
- Domain names

Flexible cloud servers

elastichosts

elastichosts.co.uk
020 7183 8250

ElasticHosts offer simple, flexible and cost-effective cloud services with high performance, availability and scalability for businesses worldwide. Their team of engineers provide excellent support 24/7 over the phone, by email and with a ticketing system.

- Cloud servers with any OS
- Linux OS containers
- 24/7 expert support

COURSE LISTINGS

Featured: Northcoders

northcoders.com
Twitter: @northcoders
Facebook: Northcoders

About us

Northcoders is the coding bootcamp for the north, based in the heart of Manchester and built upon northern values of grit, determination and community spirit. No matter what your background, you can fast-track your career and become a web or software developer in 12 weeks at their full-time bootcamp, or fit their course around your life with their 24-week part-time bootcamp. Their internal career support team will help find you work as a developer, setting up interviews with your choices of Northcoders Hiring Partners across the north of England.

What we offer

- **Full-time:** Fast-track your career in just 12 weeks.
- **Part-time:** Fit our curriculum around your life in 24 weeks.

5 tips from the pros

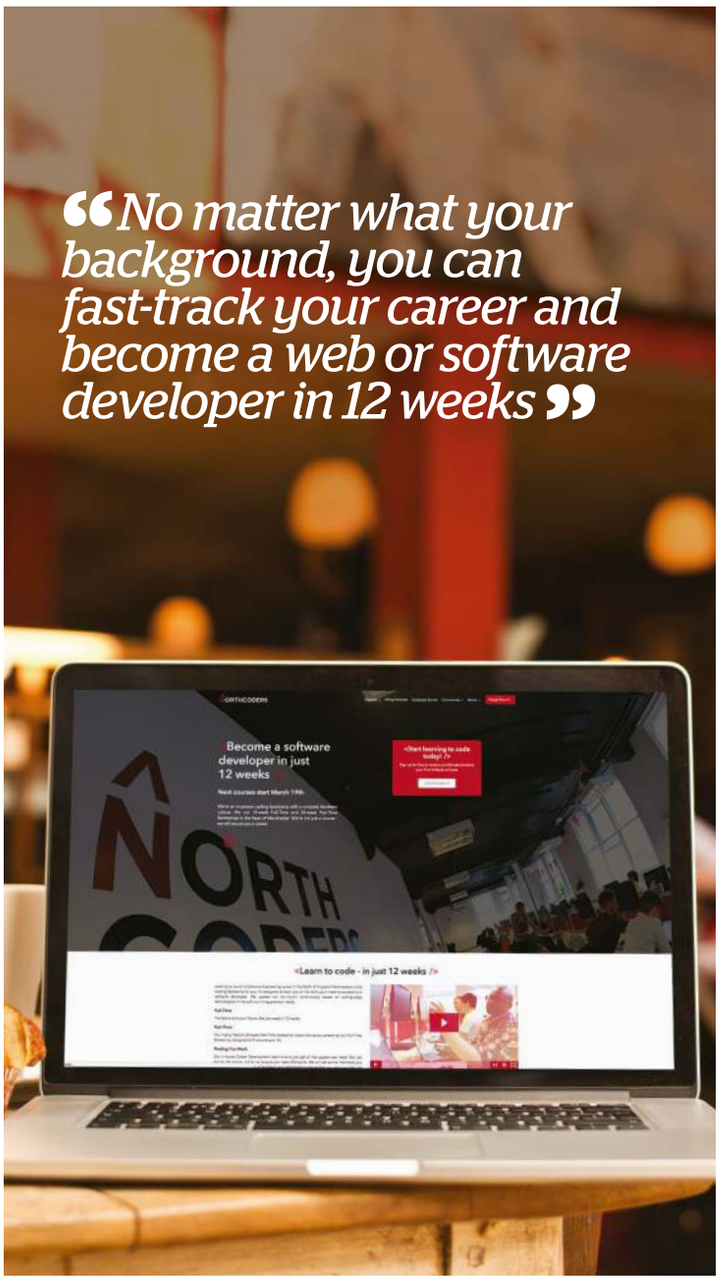
1. Get started with coding
The best way to know if coding is for you is to just try it! We recommend the free, online JavaScript track of Codecademy to get you started with the basics.

2. Do your research
Make sure you read plenty of student reviews to make sure you're applying somewhere reputable. Read their blog and have a look at their social channels.

3. Throw yourself in
Once you've decided it's right for you, set aside a few evenings each week to really start making progress! If coding is for you, this should be fun.

4. Be prepared
We'll be with you every step of the way when you apply. Make sure you go through all the materials we recommend and ask for help if you're stuck.

5. Get social
With Northcoders, you're not just on a course, you're part of a community that will stay with you long after you graduate. Make the most of it!



> Becoming part of this vibrant, caring community was something I hadn't expected before the course, but now I couldn't be without it. To be a Northcoder is to be enlightened, inspired and supported.
>
> **Joanne Imlay**
> ***Primary school teacher to software developer at Careicon***

> Northcoders delivered their part of the bargain in spades. They provided tremendous assistance in turning me into the full product – a well-rounded, capable, future tech employee – and they have the contacts to deliver the opportunities for such people.
>
> **Joe Mulvey**
> ***Maths teacher to software developer at Auto Trader***

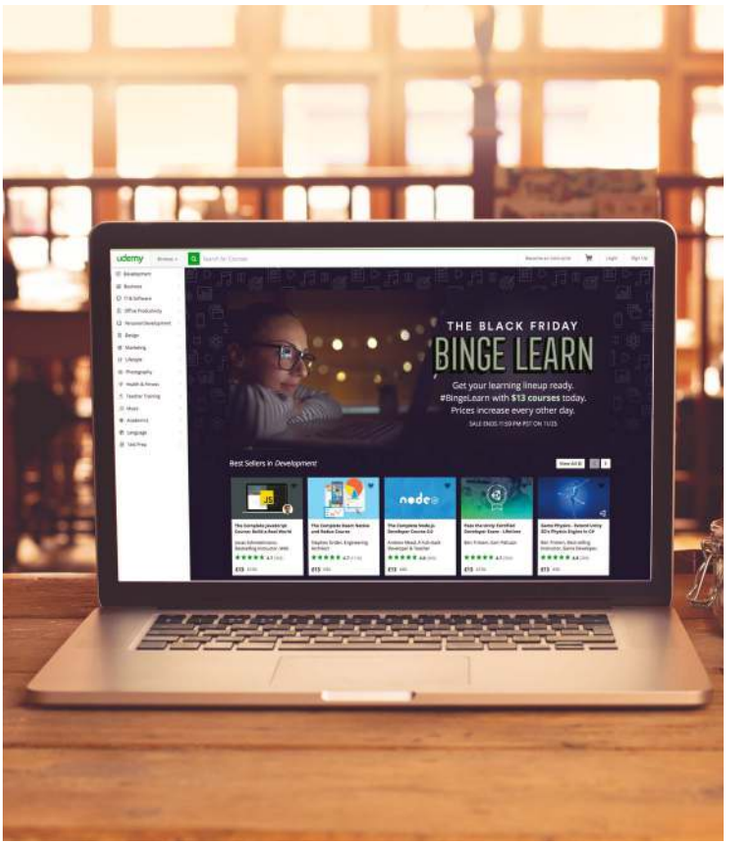
UDEMY

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.



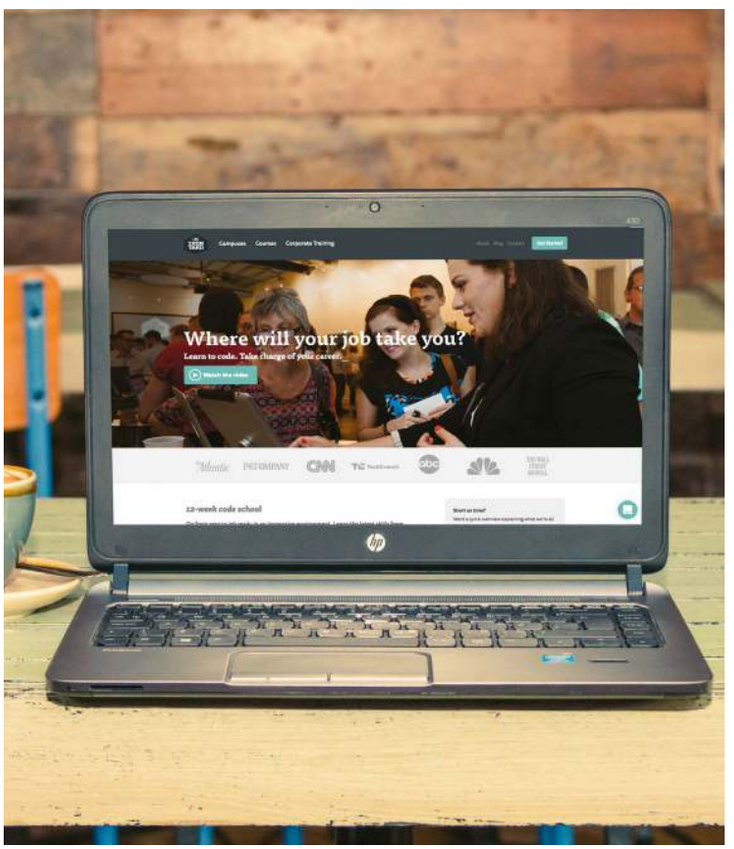
THE IRON YARD

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 backend engineering, frontend 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

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 courses 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

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

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.

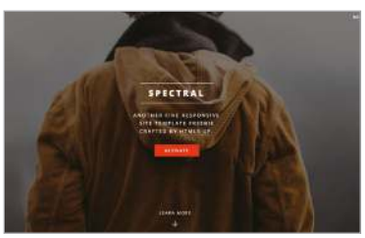
- Gain real-world skills
- Get expert instruction
- Career opportunities

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

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
- Two more chapters from the Beginners JavaScript video series from Killersites (**shop.killervideostore.com**)
- 26 Orange Teal PS filters and 27 Rogue presets from Sparklestock (**sparklestock.com**)

Log in to www.filesilo.co.uk/webdesigner

Register to get **instant access** to this pack of must-have creative resources, how-to videos and tutorial assets

Free for digital readers, too!
Read on your tablet, download on your computer

FileSilo

The home of great downloads – exclusive to your favourite magazines from Future!

- Secure and safe online access, from anywhere
- Free access for every reader, print and digital
- Download only the files you want, when you want
- All your gifts, from all your issues, in one place

Get started

Everything you need to know about accessing your FileSilo account

01 Follow the instructions on screen to create an account with our secure FileSilo system. Log in and unlock the issue by answering a simple question about the magazine.

02 You can access FileSilo on any computer, tablet or smartphone using any popular browser. However, we recommend that you use a computer to download content, as you may not be able to download files to other devices.

03 If you have any problems with accessing content on FileSilo, take a look at the FAQs online or email our team at the address below.

filesilohelp@futurenet.com

An incredible gift for subscribers

Subscribe today & unlock the free gifts from more than 50 issues

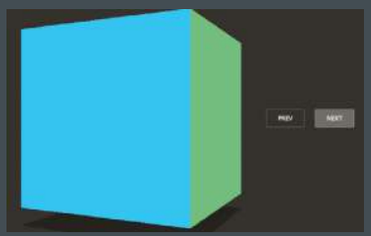
Access our entire library of resources with a money-saving subscription to the magazine – that's more than 900 free resources

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 250 creative assets

Templates, fonts, textures and backgrounds

Head to page 40 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 ten-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

HOT NEW CSS

Discover the latest and brightest properties and find out how to use them to style your projects

BUILD A CUSTOM CHAT APP

Find out how to use WebSockets and React to create an app that provides real-time feedback

GET STARTED WITH THREE.JS — PT2

In the second part of the series learn how to add lighting and shadow to a scene

DESIGN WORKFLOW: FROM START TO END

A practical guide to getting from wireframes to development handover

PRESS AHEAD

WP Engine's digital experience platform drives your business forward faster. **wpengine.co.uk**

WP engine®

Lease an Apple Mac!

Cheaper than your daily coffee

- Including accidental damage cover
- Upgrade when you want
- Own at the end
- 3 years service

iMac
with Retina display

21" FROM **£2.15** +VAT PER DAY

MacBook Pro

13" FROM **£2.20** +VAT PER DAY

Discover the benefits of leasing computers for business at
www.hardsoft.co.uk Email **info@hardsoft.co.uk**

HARDSOFT
Established for over 25 years

Authorised Reseller

9021

9000