FREE 75 MINUTES OF EXPERT PHP VIDEOS

web designer

Expert tutorials, techniques and inspiration

FIVE-STAR
SEO
Research your keywords and be number one

WHAT IS THE
WEB OF THINGS?
Find out how hardware and the web connect

30 BEST HTML CSS & JS APIS

THE ESSENTIAL ELEMENTS NEEDED TO BUILD BETTER WEB EXPERIENCES

GET STARTED WITH
THREE.JS
Learn how to load complex 3D models with WebGL

INTERACTIVE
JS CHARTS
Make your data dynamic and dazzle with ApexCharts

CODE VIDEO
TRANSITIONS
Introduce stripes and style with the curtains.js library

FUTURE
ISSUE 280

STORM

FASTER EVERYTHING

Introducing **STORM**, a hosting platform that helps your agency and websites run smoother. 82% of our customers have seen a 25% improvement in site speeds.

OVER 130

5★

GOOGLE REVIEWS

STORM HAS TRANSFORMED OUR CLIENT'S BUSINESSES

Nimbus Hosting are offering you a **30-DAY FREE TRIAL** of our revolutionary control panel STORM, with full access to explore just how much we can simplify how you manage your hosting. Don't just take it from us, take it from our clients.

Call, email or visit us at

0203 126 6782

sales@nimbushosting.co.uk

nim.host/clientstory

*Terms and conditions apply, please visit the website for more information

NimbusHosting

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

HTML is all around us

HTML has been a building block of the web ever since the day it began. Without HTML where would we be? HTML is a much different animal to the one that was knocking around in the Nineties. It has grown with the web and now offers so much more. But using it the right way is what's important.

In our latest lead feature, Matt Crouch looks at the essential elements and APIs that HTML has to offer, why you should be using them and how they should be used in everyday design.

Keeping HTML company are its two favourite friends, CSS and JavaScript. Discover a collection that will complement HTML and your builds.

Elsewhere we go beyond the traditional confines of the web and look at how the Web of Things is connecting hardware and the browser. Thanks to the internet you can take control of hardware without even having to touch it. Learn how to build a simple example.

In the final part of our Three.js series, Richard Mattka shows you how to load complex 3D models in WebGL. If you have the complete series you will have no problem creating some impressive 3D. If you want to get a back issue, head to **www.myfavouritemagazines.co.uk/web-designer-print-back-issues.**

Want to add some smart transition effects to video? Then make sure that you check out Mark Shufflebottom's Curtain.js tutorial on page 62.

As always, enjoy the issue.

> Semantic markup is vital for accessibility, search engines and much more – but it is so often misused. Build a strong foundation

Highlight

> It's a tricky thing to summarise ourselves in a single website. But it's super-important to us to present the best of who we are and what we do

Web Designer finds out what's going on at Helsinki-based Futurice. **Page 36**

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...

Matt Crouch

Matt is a software engineer who works for Vidsy in London. In this issue, he takes a look at the 30 best and latest HTML, CSS and JS APIs, as well as elements that make site development easier and standards-friendly. Think you know all that HTML has to offer? You might be surprised by what you find out! **Page 44**

> "Added in HTML 5.2, <dialog> is a new, semantic element designed to denote a supplementary, interactive component that displays out of the main flow of the document"

Paul Betteridge

Paul has over 15 years' experience leading digital organisations and pioneering digital marketing, across a range of competitive markets. In this issue he offers some essential pointers on licking the right keywords. **Page 66**

Mark Shufflebottom

Mark is a professor of Interaction Design at Sheridan College near Toronto. In this tutorial, Mark is showing the power of shaders to create advanced transition effects and interactions between videos with Curtain.js. **Page 62**

Richard Mattka

Richard is an award-winning interactive director, designer and developer. In this fifth and final tutorial in the 'Get started with Three.js' series, he shows you how to load complex 3D models in WebGL using the Three.js library. **Page 56**

Frank Kagumba

Frank is a frontend developer and tech writer based in Nairobi, Kenya. In this tutorial, he introduces you to the basics of the ApexCharts JavaScript library and shows you how to create a simple dashboard. **Page 80**

Mark Billen

Mark is a freelancer writer who has been writing about web design and technology for over 15 years. In this issue he get the inside scoop on how Helium Creative worked on the rebrand project for drip pop's stateside launch. **Page 28**

David Howell

David is a journalist with over 20 years' experience in publishing and runs his own business, Nexus Publishing. In this issue he heads to Finland to talk with Futurice and discover how they work, the tools they use and much more. **Page 36**

Tam Hanna

Tam has always found Node.js fascinating. This issue he looks at the Web of Things and how we can connect browser and hardware. Plus, he looks at Fastify, a way to create high-performance web apps without the overhead. **Page 86**

Leon Brown

Leon is a freelance web developer and trainer who assists web developers in creating efficient code for projects. This issue he recreates a host of techniques as inspired by the top-class sites seen in Lightbox. **Page 16**

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
Amit Bhaiya, Mark Billen, Leon Brown, Matt Crouch, David Howell, Richard Mattka, Mark Shufflebottom, Paul Betteridge, Frank Kagumba, Tam Hanna, Philip Morris, Rob Mead-Green

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 9001

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

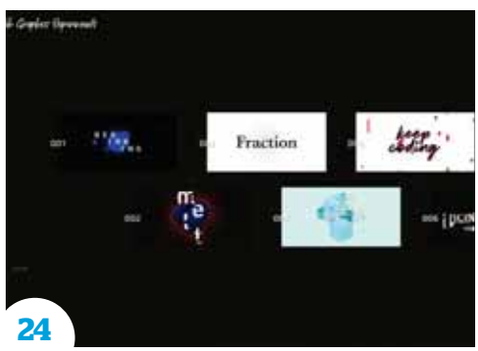
Inside issue 280

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

Cover focus

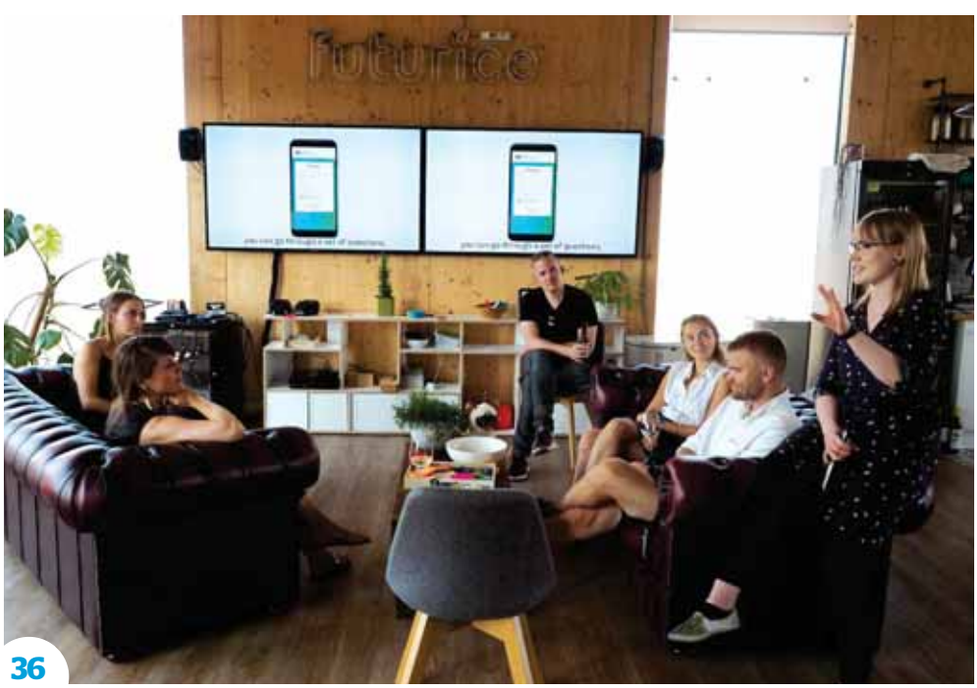


TUTORIALS

54

62


80



36

FILESILO

96 Get the latest must-have resources and videos

- 75 minutes of expert PHP video guides from Killersites.
- 100 essential brush strokes and 20 PS actions from Sparklestock.

DIGITAL EDITION

Do you want to get your hands on a digital edition? Head to your preferred app store, download, install and purchase the issue of choice.

Google Play — bit.ly/2wetvGp
iTunes — apple.co/2igtBYq

Header

The tools, trends and news to inspire your web projects

When will the web move from 2D to 3D?

Slowly but surely VR and AR are coming to the web. **Web Designer** finds out what Mozilla has been up to

Browsing the web brings home how users are still very much operating in a 2D space. There are some beautifully designed sites out there with clever little design tricks that raise them above the rest, but it's still all very flat. There is no doubt that 2D has its place on the web and it's here to stay for a good while yet. But, how do we push the web forward? How do designers make that same rectangular viewing space more exciting? Animation and sound have added a different dimension, CSS is looking to break out of the box, but it's still very much part of the 2D experience.

3D is the next obvious evolution and it has been happening for a while, yet it's still in its infancy. As with everything in web design and development, it's about browser support and the creation of new technologies. WebVR and AR are two great technologies that will make the web a more engaging and immersive place, when the art has finally been mastered. Augmented Reality (AR) is the technology that is currently pushing the web forward. With mobile being the preferred choice for web access, it provides the ideal platform with gyroscope, camera and more to bring the AR experience to life. WebVR is there as well, but it's not flavour of the month quite like AR currently is.

Mozilla have been giving the technologies a push for a few years now, with A-Frame, the web framework for building Virtual Reality (VR) experiences, making it much simpler than previously. And we can't forget Jerome Etienne's AR.js and the new AR libraries for iOS and Android. Late last year Mozilla released its experimental WebXR Viewer app. It was described by Mozilla as an app that "lets you view web pages created with the webxr-polyfill JavaScript library (**github.com/mozilla/webxr-polyfill**), an experimental library we created as part of our explorations. This app is not intended to be a full-fledged web browser, but rather a way to test, demonstrate and share AR experiments created with web technology". You can read more at **blog.mozvr.com/experimenting-with-ar-and-the-web-on-ios/**.

In a recent post Mozilla have been talking about the expansion of WebVR and Mixed Reality. They have "started the work of liberating VR and AR content from silos and headset stores, and making them accessible on the open web". The post (**hacks.mozilla.org/2018/09/webxr/**) talks about how Mozilla have entered a new phase of work on JavaScript APIs and what's new. Interestingly, it is looking to "establish a technical foundation for development of AR experiences, letting creators integrate real-world media with contextual overlays that elevate the experience". Check out the post to find out what's happening and see what we can expect in the near future. We can't wait.

STAT ATTACK

GOOGLE CHROME

How has Google grown over the last few years?

Aug 2018
59.67%
Nearly 60 percent of the global browser market

Aug 2017
54.89%
Five percent up on the previous year

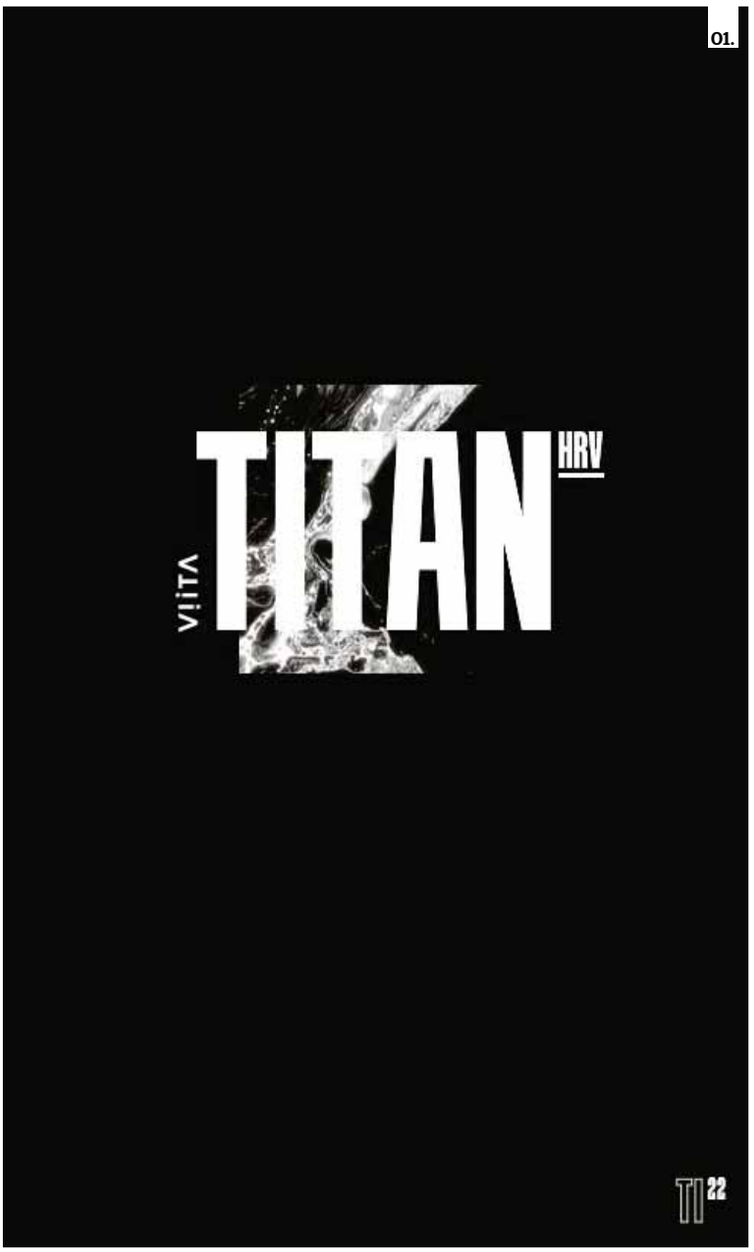
Aug 2016
49.82%
Still to break the halfway mark

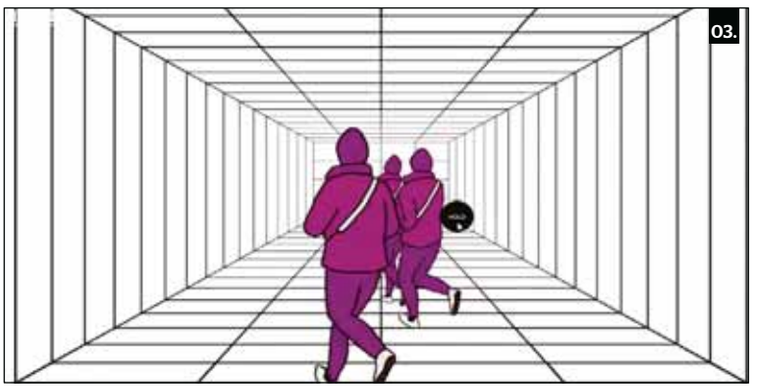
Aug 2015
45.26%
Slowest growth in the last few years

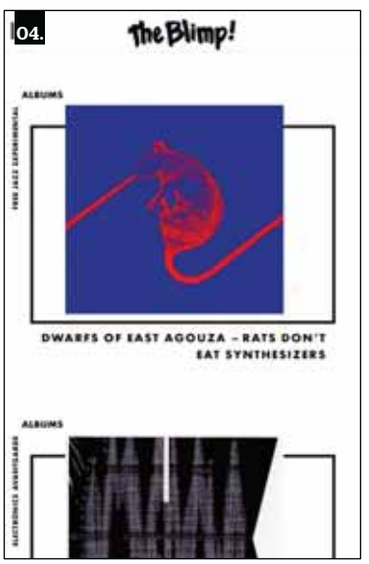
Aug 2014
39.53%
Getting very close to 40 percent market share

Source: http://gs.statcounter.com (correct as of August 2018)

Sites of the month







01. VIITA TITAN
titan.viita-watches.com/en
Surprisingly simple but smart text effects engage the user.

02. KIKK Festival
kikk.be/2018/en/home
A spotlight effect reveals what lies beneath the surface.

03. IN FOCUS
in-focus.co.jp
Click and hold to see new variations of onscreen video.

04. The Blimp
blimp.gr
Neat interactive 3D effects bring album art to life.

Graphics
ChocoToy cute
behance.net/chocotoy
A collection of cute and (cuddly?) characters from the very talented ChocoToyStudio.

Colour picker
KRTPOMix
bit.ly/2MOuuBo

#8BC63F
#F4CC2B
#F39732
#EF483A
#2BC2F1

Typesetter
Config
bit.ly/2xnJ5Pi
A condensed, geometric sans serif family consisting of 40 fonts in ten weights.

ABCabc
0123456789
ABCabc

WordPress
Tetsuo
tetsuo.edge-themes.com
A portfolio theme with attitude and glitch effects for a modern subversive feel.

The importance of visual experience

The visual experience is essential for earning conversions and loyal customers

Amit Bhaiya
Cofounder and CEO
dotcomweavers.com

> “For online shopping, sight is the strongest sense as messages and visuals are communicated through screens”

In recent years, the way people consume online content has become increasingly visual. Today, the visual experience on a website or app has a direct effect on the conversion rate. New Age consumers expect intelligent graphic design and they reward brands that offer high-quality visual experiences. Before getting into the ‘how’ of creating a great visual experience, it is important to understand the ‘why’.

People shop with their senses. They touch, feel, taste, smell and see the things around them, and then make purchase and brand trust/loyalty decisions based on the received information. For online shopping, sight is the strongest sense as messages and visuals are communicated through screens and evoke immediate emotional effects. Some eCommerce experts believe that people decide whether to buy a product or service within the first few seconds of seeing it. As the saying goes, ‘first impressions matter’. It pays (literally) to provide the best possible visual experience to prospective customers. How do companies build a conversion-focused visual experience into brand strategy? The answer is to focus on three important aspects of eCommerce visual content: branding; product content; and consistency.

The first step is to set the stage for customers to trust a brand — long enough for these individuals to stick around and explore the products/services. Digital natives can spot stock imagery, and they either disregard it (at best) or leave the website (at worst) since the lack of originality cheapens the brand experience. Brands should choose authentic images over stock photos. The style, layout and navigation of a website or app should also match the brand identity and story. Whether a brand sells edgy clothing or sophisticated financial products, visitors need to visually understand the brand and feel that they are in the right place. The same principle applies to textual content a website. Text and the corresponding layout should tell a unique story, communicate the benefits of a brand and guide user actions/decisions. Good visual content makes it easy — and sometimes even fun — for people to do what brands want them to do. Once visitors decide they are in the right place, brands should give them a total view of products/services. For example, nowadays, one image of a product is mandatory. To take things a step further, however, brands should embrace a curated collection of images that showcase a variety of angles and depictions (360-degree product views, for example). A great visual experience answers most, if not all, consumer questions before they even need to ask.

Additionally, original brand content that features real people wearing or using a product/service can help potential customers envision themselves doing the same. These visual context cues are more intriguing than standalone product shots. Incorporating interactive content such as instructional videos can also deepen the visual experience. The more value brands can demonstrate with content, the better. Beyond the latter, it is also important not to forget user-generated content (UGC). UGC includes reviews, ratings and images submitted by real customers. Today’s consumers make purchase decisions based on what others have to say. This type of ‘social proof’ is an essential aspect of eCommerce and must be included in the visual experience. Last but by no means not least, visual experiences need to be consistent — and provide consistent quality — across all channels. The visual experiences that customers buy into on a website must also translate on the corresponding app(s), marketing emails, social media channel(s), product packaging and in-store displays (where applicable). A seamless visual experience matters. It enables companies to offer true omni-channel experiences and solutions to customers.

Moreover, brands should remember that visual content weaves a carefully crafted experience for the end user to enjoy. At the end of the day, if the customer does not enjoy the experience or if it gets interrupted, brands risk conversion. On the other hand, with positive experiences, there is greater potential for a returning, loyal customer. Compelling visual experiences are the future of online business. Improving these experiences does not have to feel overwhelming. Brands should take things a step at a time. Start small with extra product photography and then go from there. A better visual experience can only help businesses stay competitive and continue to thrive in 2018 and beyond.

webkit

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

Batificity

batificity.com

Like superheroes and need a little help with your CSS specificity issues? Then this is the guide for you. The site uses a host of characters, scenarios and hardware from the Batman universe to explain the different rules. Simply scroll down the page, read the rules, note their strength, and then remember to apply them.

TOP 5
Web conferences
November 2018

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

Accessibility Club

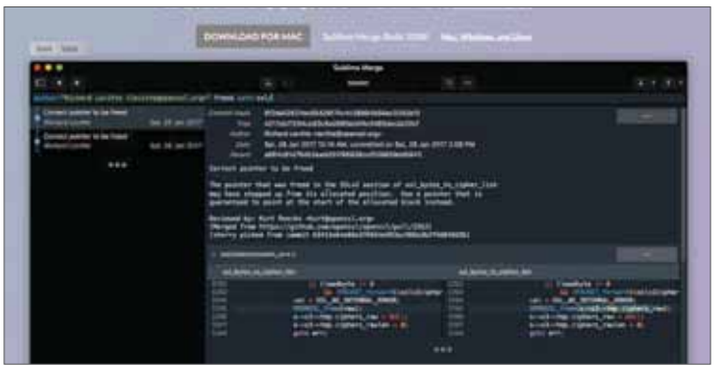
accessibility-club.org

A conference aimed at web accessibility. There is a lineup of speakers to educate and inform including Léonie Watson.

DevRelCon

london-2018.devrel.net

Are you a developer? Then this conference is for you. Get three days of developer relations, community, marketing and talks.



Sublime Merge

sublimemerge.com

Sublime Merge is from the same stable as Sublime Text. It provides great performance, a cross-platform UI toolkit, and a custom Git reading library.

Web Audio demos

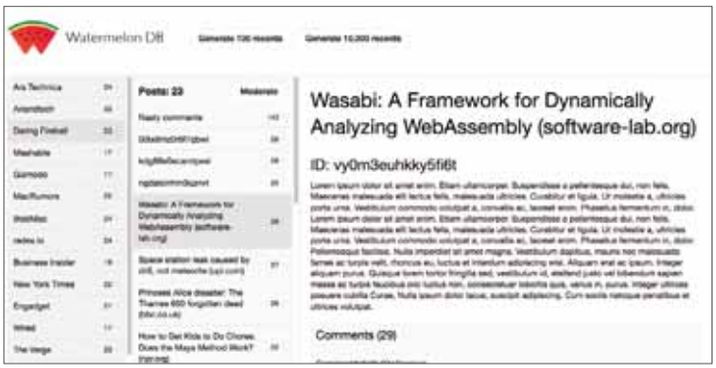
webaudiodemos.appspot.com

A collection of examples and demos demonstrating what can be done with the Web Audio API. Experiment and discover the power the API has to offer.

Pa11y

pa11y.org

Accessibility on the web is crucial. An all-inclusive experience is just around the corner with the help of these free and open source tools.



Watermelon DB

watermelondb.now.sh

Watermelon DB is described as a next-gen React database. Build React apps that scale from hundreds to tens of thousands of records and stays fast.

DotCSS

dotcss.io

It's all about CSS and boasts a host of speakers — including Sara Souiedan — to give you the lowdown on the king of styling.

Half Stack

halfstackconf.com

A one-day, single track, UI-centric, JS conf based in a pub in Shoreditch. 16 speakers and sessions, and lots of beer (we suspect).

React Day

reactday.berlin

'Build code, not walls' is the sentiment of this React conference. It boasts 24 speakers in the heart of Berlin.

webkit

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

Electron 3

electronjs.org

Electron has been building cross platform desktop apps with JavaScript, HTML, and CSS for a while now. Its third incarnation has just been released and continues the previous good work. The new release includes a selection of changes and new features. To see what's been happening with Electron take a visit to **electronjs.org/releases**.

TOP 5 CODEPENS

Be inpsired by this collection of smart and interesting codebases

Andy Warhol..ish

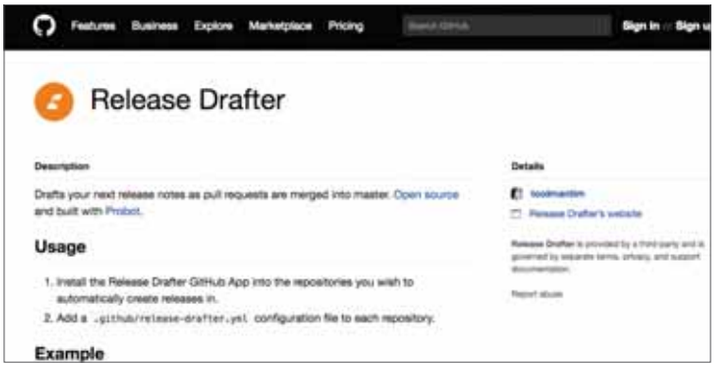
bit.ly/2PYTKHf

Divided into four quarters, drag from the middle to reveal new variations and effects. Add different images for more uniqueness.

CSS Rainbow Loader

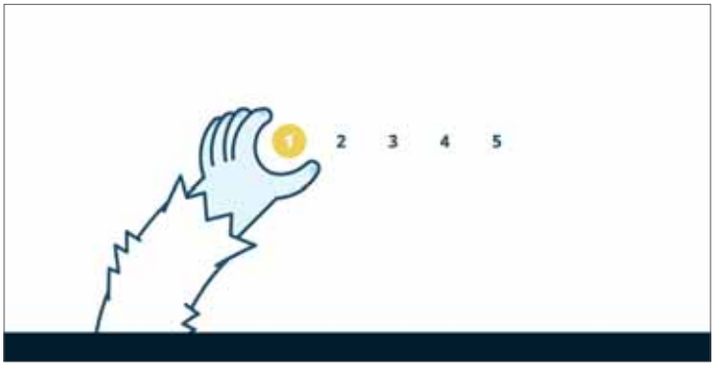
bit.ly/2QSGBkg

A simple repeating rainbow illustration that works well as colourful loader. Experiment with the speed and colours.



Release Drafter
bit.ly/2ONY9MR
Looking forward to drafting your next release notes? This neat GitHub app drafts your next release notes as pull requests are merged into a master.

T-Writer
bit.ly/2NE0mOz
The purpose of the T-Writer.js library is to make it fast and easy to create typewriter-effect text. Check the demos to see what's on offer.



Yeti Hand Pagination
bit.ly/2pyEOEn
A smart and engaging way to introduce pagination. Pick a page number and wait for the animation to spring into action.

Mauerwerk
bit.ly/2Octdc8
A neat react-spring driven masonry-like grid with impressive animated transitions. Click and see grid items transition to fullscreen and back.

Ripple Mouse
bit.ly/2zoLQBy
Crazy coloured swirling patterns burst into life with water-like ripple effects as you drag the cursor across the pattern.

Weird input
bit.ly/2ptMaJo
Introduce inputs with interest. Watch as animated letters jump into the input field as you type.

GLSL: Sinnoise 1
bit.ly/2zoFVMN
Use the cursor to send the spotlight across the screen with a soft swirl effect that goes in and out of focus.

Nuclear Dissent https://nucleardissent.com

Digital:
Jam3 www.jam3.com

Video:
Rogue Productions rogueproductions.co.nz

Sound:
Grayson Matthews graysonmatthews.com

3D production:
Aparato www.aparato.tv

"Interactive nuclear documentary combining compelling video footage with VR panoramas and interactive heat map x-rays"

Colours

#1F86AA #49909D #CDCOCO #9C99AA

Tools

HTML5, SVG, GSAP, Three.js

Fonts

abcABC
1234567890

The Druk font family is designed by Berton Hasebe and used in its distinctive Wide, squat bold style.

abcABC
1234567890

Romain BP Text by Ian Party appears as an accent typeface for the site's Credits section.

Above
More a hard-hitting experience than 'website' as such, Nuclear Dissent uses video chapters to tell a shocking story

Middle
VR-ready, 360-degree scenes provide an immersive backdrop to the narrative, bringing French Polynesia into desktop and mobile browsers

Bottom
Toggling into 'X-ray' view reveals red heat maps boasting rollover hotspots that trigger audio commentaries explaining the impact

Create a random text animation with video background

Use JavaScript and CSS to create an animated content overlay for a full-screen video background

1. Initiate the HTML document

The first step is to initiate the structure of the HTML document. This consists of the document container that stores the head and body sections. While the head section is used to load the external CSS and JavaScript resources, the body will contain the visible page content created in the next step.

2. HTML content

The foreground page content is placed inside the 'main' container to deliver the advantage of easy control of content flow. The text element has the 'overlay' class applied so that it can be referenced by the JavaScript and CSS for applying the text animation. Multiple elements can have the animation applied by using the 'overlay' class.

```
<main>
        <h2 class="overlay">
  This is a story all about how...
        </h2>
</main>
*** STEP 3 HERE
```

3. HTML video element

The final part of the HTML is to define the background video element. Not all browsers are able to support each video standard, hence the need to specify different sources. The browser will display the first source it is able to support. Take note of how the video element has the 'autoplay', 'muted' and 'loop' attributes applied so that it automatically plays and repeats without sound.

```
<video autoplay muted loop>
        <source src="http://techslides.com/
demos/sample-videos/small.webm" type="video/
webm" />
        <source src="http://techslides.com/
demos/sample-videos/small.mp4" type="video/
mp4" />
        <source src="http://techslides.com/
demos/sample-videos/small.ogv" type="video/
ogg" />
        <source src="http://techslides.com/
demos/sample-videos/small.3gp" type="video/3gp"
/>
</video>
```

4. CSS initiation

Create a new file called 'styles.css'. The first step in this file is to define the properties of the 'main' content container. Default settings for font and colour are applied for child content to inherit. Auto values are applied to the side margins so that child content appears centrally aligned.

```
main {
        font-family: Helvetica, sans-serif;
        color: #fff;
        padding: 2em;
        width: 75%;
        min-height: 100vh;
        margin: 0 auto 0 auto; }
```

5. Video element styling

The background element requires specific styling in order for the effect to work. Firstly, fixed positioning is important to guarantee that it stays in the same position if the user scrolls the page. Secondly, it must use a negative z-index that will guarantee its position underneath the main page content. Transform and size are also used to set the element's size and location to cover the full-page window.

```
video {
        position: fixed;
        top: 50%;
        left: 50%;
        min-width: 100%;
        min-height: 100%;
        z-index: -9999;
        transform: translateX(-50%)
translateY(-50%);
        background: #000; }
```

6. Overlay children

The 'overlay' element will be manipulated by JavaScript to split each letter of its text content to be wrapped by a span tag. This allows individual letters to be animated via CSS. Firstly, the default settings for the 'span' letters are defined to have relative positioning, invisible opacity and an applied 'animateOverlay' animation. Secondly, a delay to their animation is applied based on their child positioning.

```
.overlay span{
        position: relative;
        opacity: 0;
        top: 1em;
        animation: animateOverlay 1s ease-in-
out forwards;
}
.overlay span:nth-child(4n) { animation-delay:
0s; }
.overlay span:nth-child(4n+1) { animation-
delay: 1s; }
.overlay span:nth-child(4n+3) { animation-
delay: 2s; }
.overlay span:nth-child(4n+2) { animation-
delay: 3s; }
```

7. Overlay animation

The animation applied to each span element consists of just one frame that the elements will animate towards. This sets their opacity to become fully visible, along with their vertical positioning to animate towards their default text flow position. Take note of how step 6 sets each span element to be pushed down by 1em.

```
@keyframes animateOverlay {
        to {
                opacity: 1;
                top: 0;
        } }
```

8. Overlay search

Create a new file called 'code.js'. This first step will search for all of the elements using the 'overlay' class – of which a 'for' loop is used to apply the code in step 8. These elements are not available until after the page has loaded, so they need to be placed inside an event listener in the browser window that is triggered on its load completion.

```
window.addEventListener("load", function(){
        var nodes = document.
querySelectorAll(".overlay");
        for(var i=0; i<nodes.length; i++){
                *** STEP 9 HERE
        }
});
```

9. Text manipulation

Each element found in step 8 needs to have its HTML contents redefined to have each letter inside a span element. This is achieved by reading its plain text using 'innerText', and then using a second 'for' loop to individually add each letter to the new version of the HTML – complete within its span tag. After each letter has been read, the parent node's 'innerHTML' is updated with the new HTML.

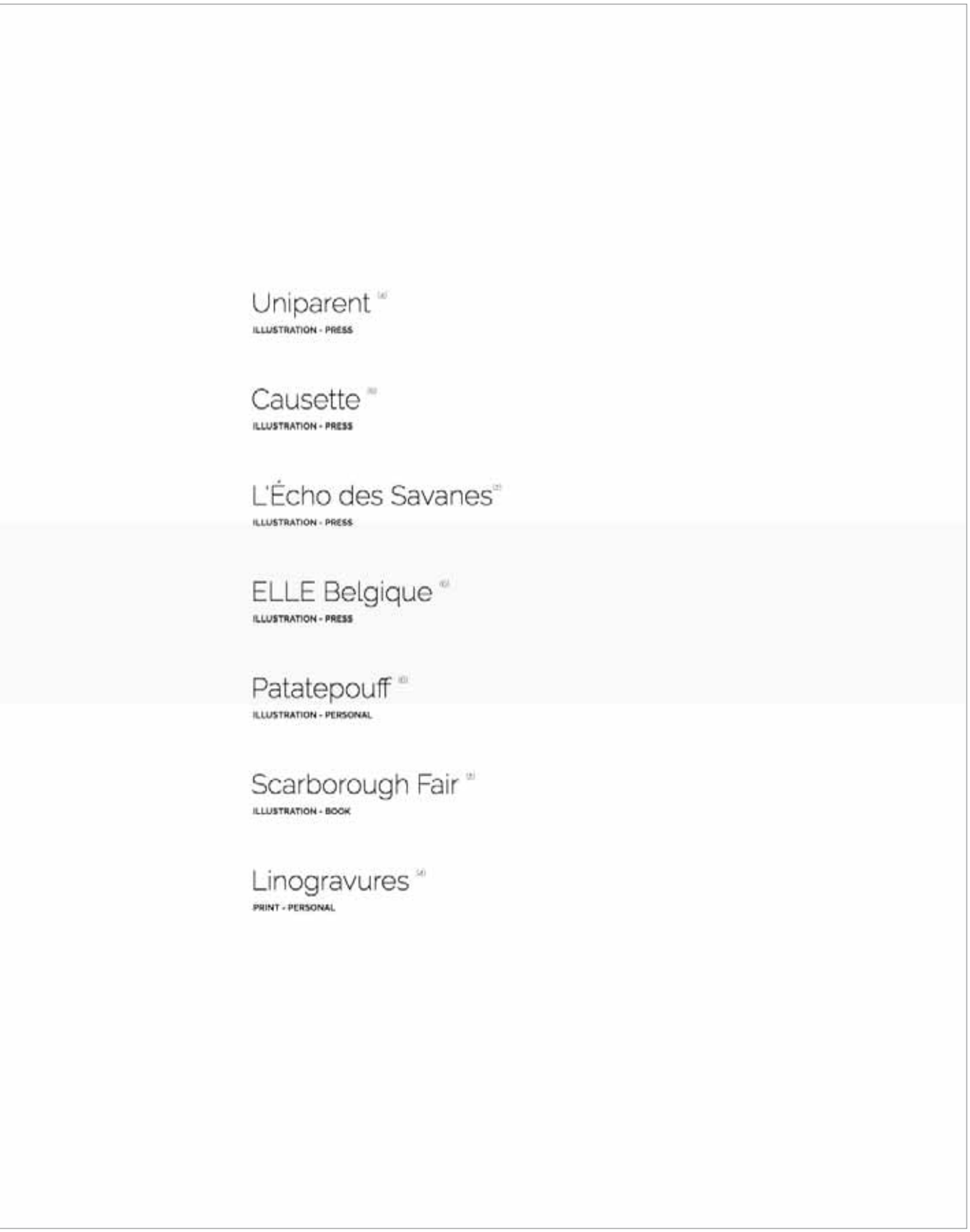
```
var words = nodes[i].innerText;
var html = "";
for(var i2=0; i2<words.length; i2++){
        if(words[i2] == " ")html +=
words[i2];
                else html +=
"<span>"+words[i2]+"</span>"
}
nodes[i].innerHTML = html;
```

Marie Morelle | Illustratrice

www.marie-morelle.com

Designer:

Vincent Saisset https://vincentsaisset.com



"Paris-based Illustrator and Art Director Marie Morelle serves up a whimsical and clean sketchbook site of collected works"

Colours

#FFFFFF

#F5F5F5

#E96547

#EFBE41

Tools

WordPress, Typekit, GSAP, jQuery

Fonts

abcABC
1234567890

The Raleway font, initially designed by Matt McInerney, is a sans-serif typeface used here via TypeKit but also available on Google Fonts.

abcABC
1234567890

Aside from the Regular weighting, Raleway is also used in Extra-Light typeface for the <h2> header links.

Top
An almost exclusively white colour palette is joined by superbly smooth page-turning effects to emphasise the sketchbook metaphor

Left
The site's simplicity and thoughtful use of animation technologies ensures it looks and behaves identically on smaller devices

Above
A custom mouse pointer suggests charming little areas of interactivity such as this rollover illustration within the About section

Create an interactive navigation image control

Make empty space more appealing with this eye-catching animated effect

1. Page initiation

The first step of creating the webpage is the definition of the HTML document. This consists of HTML to define the document container, which in turn stores the head and body sections. While the head section is used to load external JavaScript and CSS resources, the body section is used to contain the visible HTML content created in step 2.

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

2. HTML content

The navigation content is kept to a minimum, with a specific focus on defining the navigation content. This consists of the navigation container and its associated links. Each link has a 'title' attribute that will be used as a reference to influence visual styling in later steps, with the help of JavaScript and CSS.

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

3. Event management

Create a new file called 'code.js'. Upon completion of the page loading, this code will search for all of the 'a' links inside the 'mainNavigation' from step 2. A 'for' loop is used to apply a 'mouseover' event listener to each item found. This event listener will set an attribute called 'data-theme' on the parent 'mainNavigation' container, which is to be set as the title attribute of the 'mouseover' link.

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

4. CSS initiation

Create a new file called 'styles.css'. This step initiates the CSS with the general page styling. Specifically, all elements are set to use 'border-box' for including padding as part of width calculations. The page is set to have no visible border spacing through margin and padding, as well as the default font for content to inherit.

```
*{ box-sizing: border-box; }
html,body{
        display: block;
        margin: 0;
        padding: 0;
        font-family: Helvetica, sans-serif;
        color: #000;
}
```

5. Navigation container

The navigation container is set to use fixed positioning so that it always remains visible. Its width is set to half of the browser window, with settings to present any background image to cover 40% of the right side.

```
#mainNav{
        position: fixed;
        display: block;
        width: 50vw;
        transition: background 1s;
        background-repeat: no-repeat;
        background-position: right center;
        background-size: 40%;
}
```

6. Navigation items

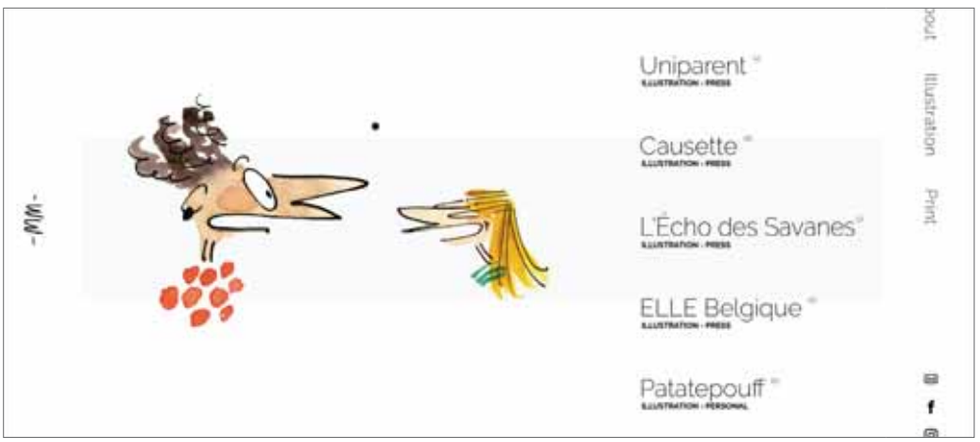
Each item inside the navigation container is set to display across 50% of the navigation – leaving space for its associated item. Each item is set with a border, padding and margin to appear distinct from each other. Alternative background and font colours are used when items are being hovered with the mouse pointer.

```
#mainNav a{
        display: block;
        width: 50%;
        color: #000;
        padding: 1em;
        margin-top: .5em;
        border: 1px solid;
}
#mainNav a:hover{
        background: rgba(0,0,0,.5);
        color: #fff; }
```

7. Image definition

The final step is to associate the images to display for each of the navigation links. With the JavaScript from step 3 setting the 'data-theme' attribute of the parent container to match the title of the latest hovered link, this step is used to specify the background image for each value that 'data-theme' could be set to.

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

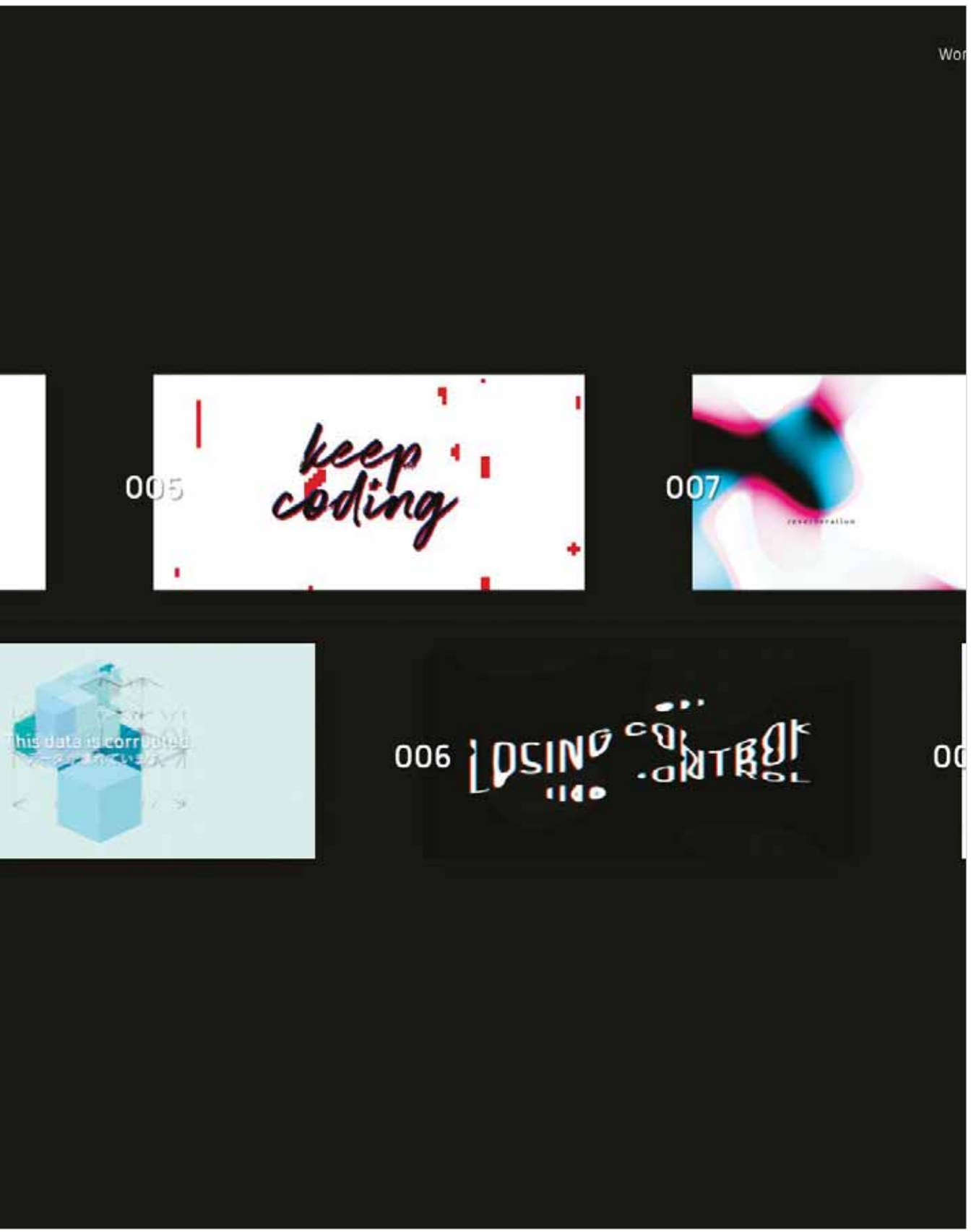


Web Graphics Experiments

experiments.p5aholic.me

Designer:

Keita Yamada p5aholic.me



"23-year-old Japanese developer Keita Yamada's collection of graphic art experiments in web technologies, with new works added weekly"

Colours

#101010 #003BFF #CEF6EF #F32D8E

Tools

WebGL, GSAP, GLSL, Three.js

Fonts

abcABC 1234567890

Moonstruck font by Make Media type foundry is used to provide the distinctive handwritten typeface on the title home link.

abcABC 1234567890
abcABC 1234567890

Anteb font designed by Chatnarong Jingsuphatada for Typesketchbook is employed in Light and Regular typefaces.

Create a fading previous/ next content navigation

Use the HTML article element to generate an animated previous/next content navigation feature

1. Initiate the document

The first step is to initiate the web page as an HTML document. This consists of defining the HTML container, which stores the <head> and <body> sections. While the <head> section references the external CSS and JavaScript resources, it is the <body> section which stores the page content – created in Step 2.

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

2. Page content

The page content consists of a <main> container that is used to store the primary page content, followed by two elements that will be used as navigation controls. Each article has a unique ID that can be used by CSS and JavaScript to present the individual content sections when they are required.

```
<main>
        <article id="p1">
                <h1>Section 1</h1>
        </article>
        <article id="p2">
                <h1>Section 2</h1>
        </article>
        <article id="p3">
                <h1>Section 3</h1>
        </article>
</main>
<span data-nav="left">&lt;</span>
<span data-nav="right">&gt;</span>
```

3. JavaScript: Navigation function

Create a new file called 'code.js'. The first part of this JavaScript code defines the 'nav' function that will be called in response to interactions with the navigation controls. Its parameter accepts details about the event, including a reference to the navigation control. The index value is calculated and used to update the window URL using the associated article ID – defined in Step 4.

```
function nav(e){
        e.preventDefault();
        if(e.target.getAttribute("data-nav")
== "left" && nav.index > 0){
                nav.index--;
        }else if(e.target.getAttribute("data-
nav") == "right"){
                if(nav.index < nav.nodes.
length-1){
                        nav.index++;
                }
        }
        window.location.href = "#"+nav.
nodes[nav.index].getAttribute("id");
}
nav.index = 0;
nav.max = 0;
```

4. JavaScript: Load setup

The JavaScript code must wait for the page to finish loading before it can reference the page content. We achieve this by placing the initiation code inside a 'load' event listener applied to the window. The 'nav' function is provided with a list of articles inside the <main> container, as well as the 'nav' function being applied in response to 'click' events on the navigation controls.

```
window.addEventListener("load", function(){
        var nodes = document.
querySelectorAll("[data-nav]");
        nav.nodes = document.
querySelectorAll("main > article");
  nodes[0].addEventListener("click", nav);
  nodes[1].addEventListener("click", nav);
});
```

5. CSS articles

Create a new file called 'styles.css'. This step defines the default presentation for the articles inside the main container. These elements are set to be placed in the top-left corner of the browser window and to cover the full page. The visibility are hidden using 'opacity' and 'z-index', with a transition on their opacity set to animate them into view when required.

```
main > article{
        position: absolute;
        display: block;
        height: 100%;
        width: 100%;
        top: 0;
        left: 0;
        opacity: 0;
        transition: opacity 1s;
        overflow: scroll;
        z-index: 0;
        color: #fff; }
```

6. Article activation

The 'nav' function defined in Step 3 will trigger changes in the page URI that references individual articles using their ID. CSS detects this using the 'target' selector, in which this step changes their default 'z-index' and 'opacity' to become visible. This step also sets individual background colours for the articles using their ID references.

```
main article:target{
        z-index: 2;
        opacity: 1;
}
main article[data-previous]{
        z-index: 1;
}
#p1{ background: red; }
#p2{ background: green; }
#p3{ background: blue; }
```

7. Navigation controls

The navigation controls are identified using their 'data-nav' attribute. They share a set of common presentation rules such as fixed positioning, vertical location and 'z-index'. These are important to remain visible in the same location above the page content regardless of scroll location. Unique horizontal styles are applied using the value of their 'data-nav' attribute.

```
[data-nav]{
        position: fixed;
        top: 45vh;
        font-size: 3em;
        width: 1em;
        color: #fff;
        z-index: 9999;
        background: rgba(0,0,0,.5);
}
[data-nav="left"]{ left: 0; }
[data-nav="right"]{ right: 0; }
```

lollipop gild

Florida's Helium Creative serves us the inside 'scoop' on how its flavourful rebrand project for drip pop's stateside launch featured a website build dusted with designer sprinkles

drip pop

drippop.com

AGENCY
Helium Creative
heliumcreative.com
@heliumcreative

PROJECT DURATION
7 months

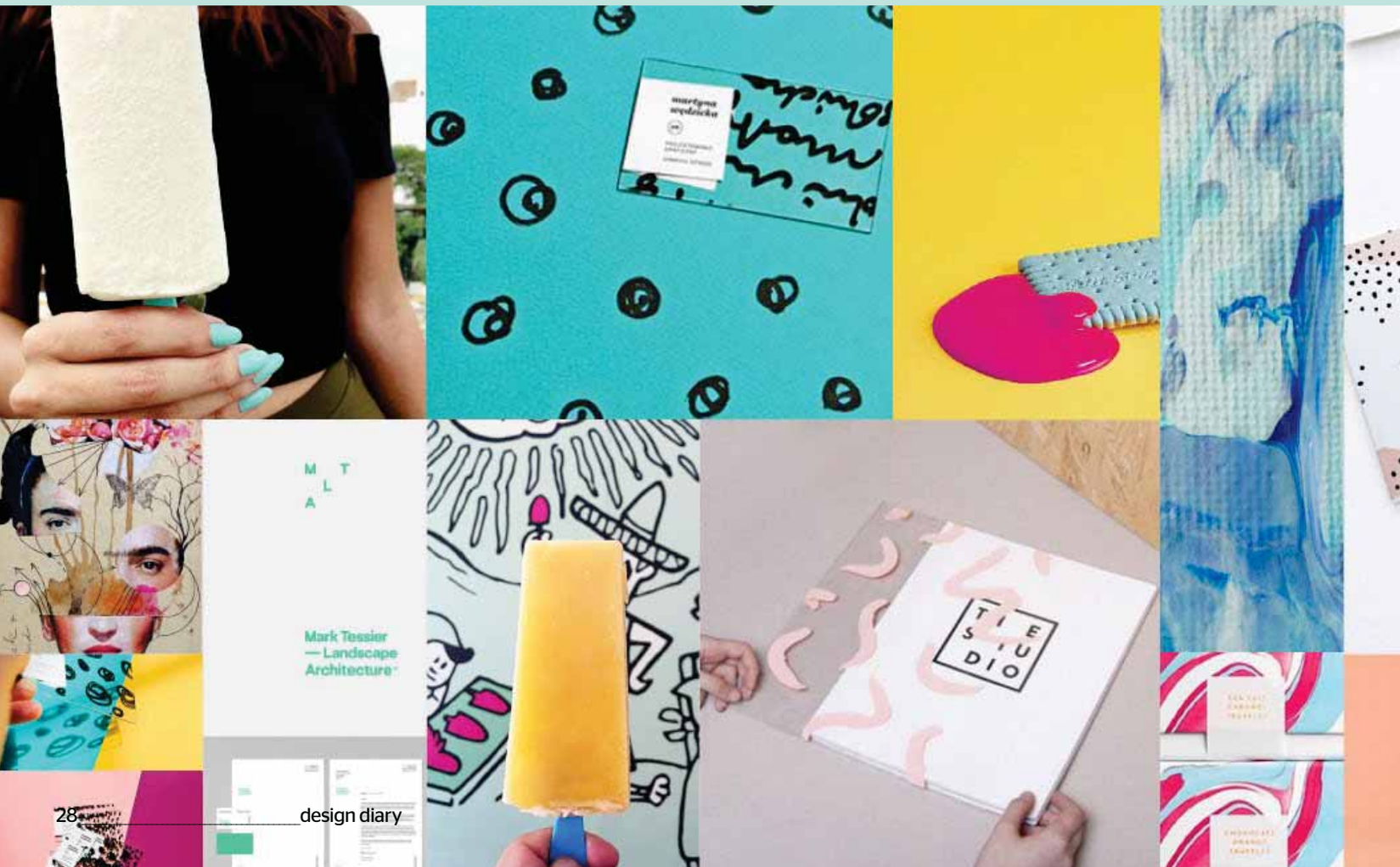
NAMES & POSITIONS

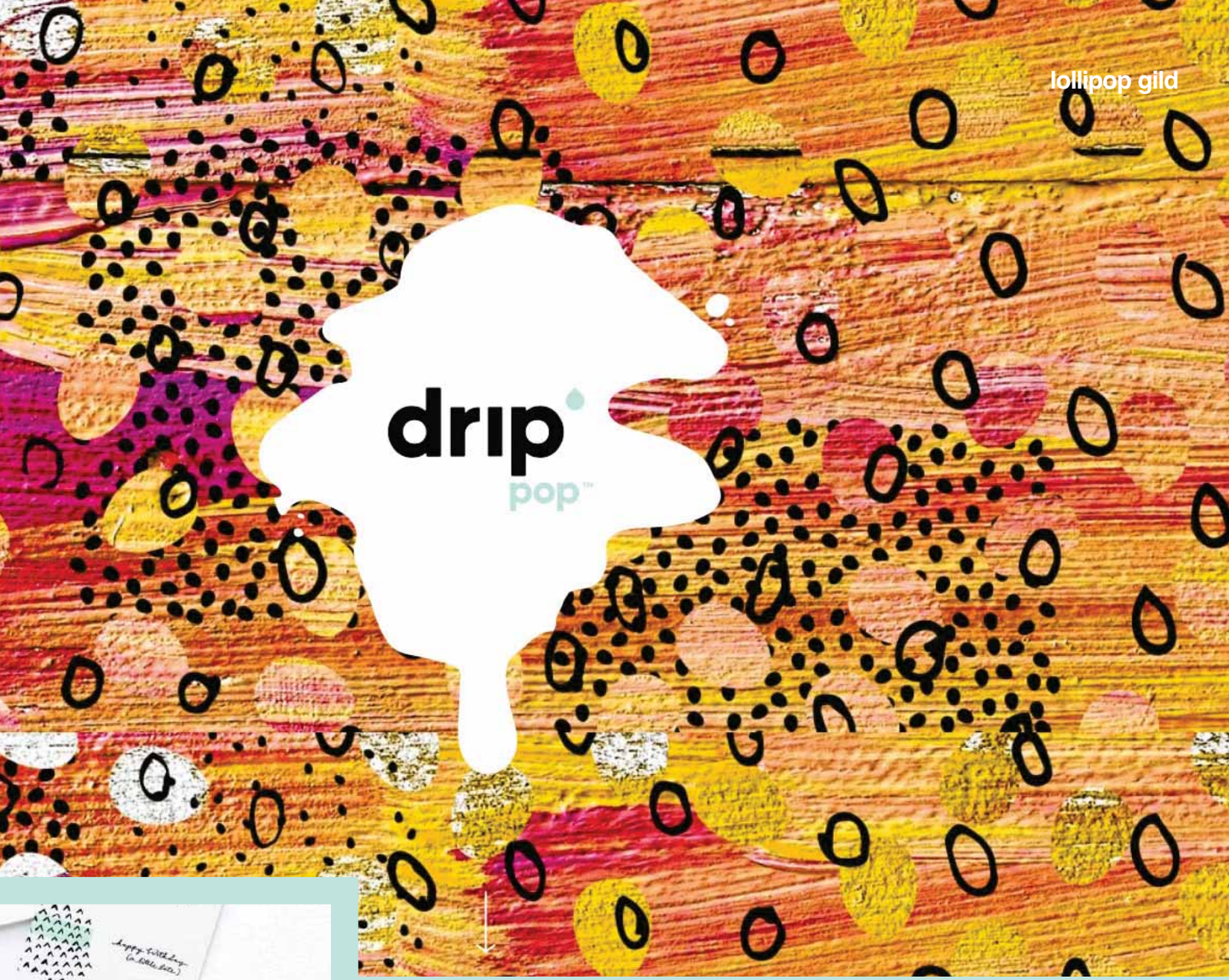
Christopher Heller
Creative Director

Ryan Heller
Art Director

Kelly Gedvilas
Experience Designer

Enid Nolasco
Creative Manager





IN THE SCIENCE OF MODERN MARKETING, THE IDEA OF 'BRAND' IS MORE CLOSELY ENTWINED with and intrinsically linked to an online presence than ever. Big, established and long-trusted household names feel a responsibility to push boundaries and exert their market prowess in the digital space, while for others it can really open the first window into consumer consciousness. Creative agencies have, of course, had to vastly evolve their remit to accommodate this seismic promotional shift and deliver 'out of the box' branding services to clients. Operating almost off the golden beaches of Fort Lauderdale, Florida, is one such award-winning agency that applies 'fine-arts' sensibilities to the seamless blend of brand with design. Helium Creative create dynamic, personified, branded experiences for clientele who value an in-depth collaborative approach around the delivery of 'unconventional solutions' for impactful brand experiences. Typically associated with luxury real estate web projects, things were to get decidedly more diverse when Fiji Pop approached them with a delicious dilemma. Keen to introduce the US market to their frozen popsicles, or lollipops to British tongues, these five young Bolivian entrepreneurs were seeking a complete brand overhaul including a new name. Taking direct inspiration from the flavours, sensations, colours, ingredients, and story behind the products, the Helium gang set off to create a solution full of vibrancy and playfulness. Rechristening them 'drip pop' instead, they would go on to develop new brand messaging, identity, pop packaging, supermarket box packaging, a splash page and, indeed, a full website. "They were looking for a creative studio that saw the brand as a piece of art," Helium begins. "Once we had the brand elements we needed to work with, the focus became building a unique online experience. With this in mind, we wanted to push the play on the different layers each flavour has, creating this dimensional playground and interaction with every user. We kept the design itself very minimal, so the focus remained on the message and the playful product." Allowing the project brief to bring out the kid in each of them, Helium would be uniquely placed to appreciate drip pop's appeal and spread the love.

MAPPING THINGS OUT

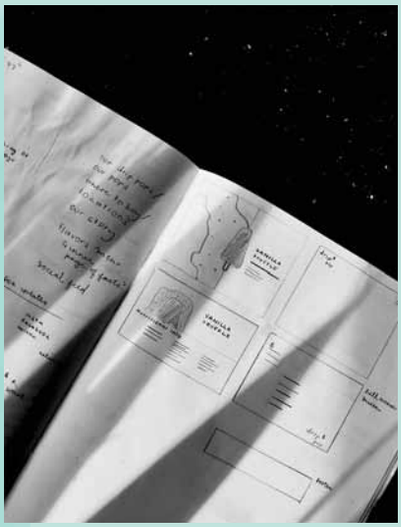
"Our process typically starts off with a basic sitemap for any web development project," begins Experience Designer Kelly Gedvilas when quizzed on the extent of client communications. "We gather all the necessities from the client; whether it's things they wish to see on their site, product shots or additional imagery. This ultimately helps us accommodate design for everything needed initially, as well as for the future of the brand. For example, while planning the design for the product page that showcases all the flavours of pops, we needed to create something that would make adding more flavours or categories easy, while also ensuring the user was able to interact with ease and engagement." With this initial sitemap established, Helium will design two main pages from the site as a working illustration. These are then shared with the client to stimulate deeper brainstorming and heighten the sense of collaboration, giving a client like drip pop enough confidence to let the team plough ahead. "We bounce different ideas that can be done through development to really elevate their site and based on budget we can play around with ideas that really push the design, like the value of adding video content or additional pages we see fit. When this stage is approved, we then move forward with completing the remainder of the design. There's communication when we finalise the flat design, but as for how everything comes together regarding animation and development, we are fortunate enough to have clients who trust us with that process and let us run full steam."

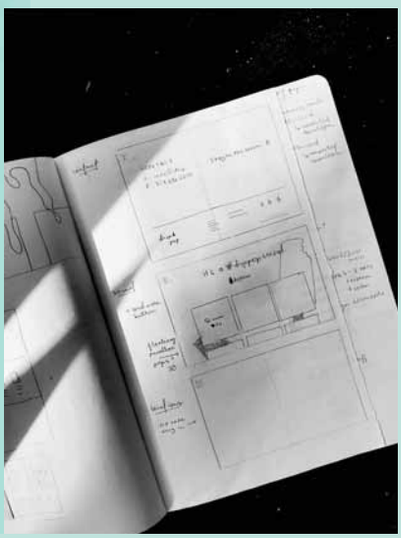
Artful branding

"Dessert is always fun, especially when it's a frozen dessert meant to be enjoyed on hot summer days," the team laughs. "For drip pop we had the opportunity to play with food both in real life and on the computer." Hardly a bad way to make a living, most would agree wholeheartedly. Plus that enthusiasm for the consumption of a food-based client product would, of course, reap dividends within the initial branding efforts. "From an actual melting lavapop we created a vector illustration used to further the drip identity. Hand-drawn patterns that portray ingredients are added to photographs of rich textures to represent the vibrant colours found within the lavapops themselves and help make the packaging unique pieces of art." This critical identity work needed to happen before even conceptualizing how the web side of the project could reflect it. The team knew that understanding the new brand inside and out was crucial, not least so they could be aware of all the brand elements to use in the design process. "To get started, we had this brand, all of these really cool textural design elements, patterns, and backgrounds, etc., and so the question became what can we do to make all of this content come alive while not becoming this crazy overload of design? Quickly the answer became all about creating that magical balance of clean, typographical design that the brand entails and contrasting it with organic and playful artistry."

Boundary pushing

In this respect, Helium had freer rein to design something without the normal boundaries their normal commissions dictated. That freedom can, of course, be a blessing and a curse in some respects and gave the designers some thinking to do over where they couldn't perhaps tread. "As much as we want free rein with anything, we always go back to needing those limits," laughs Experience Designer Kelly Gedvilas. "With this in mind and from an experience standpoint, I knew I wanted to showcase all these layers into moving parts and what would elevate these parts by how the user would interact with them. I suppose when it came to limits, I needed to create something that would be easily understood by an older demographic, while still





appealing to the younger millennial and what would be attractive to both." This formative direction was, guided by some sign-off approval from the client, with communications concentrated most at each end of the process. Helium tends to draw up a basic sitemap that signposts all the content assets required, which the client then green-lights before proceeding. "Once the client approves the content of the sitemap, we jump into designing two main pages of the site," Kelly continues. "The purpose of this is to show them our initial ideas without designing everything up front. We share these two main pages to open up collaboration between the client and designer. This enables everyone to discuss ideas and initiate conversation of the thought process behind the design, allowing the client to feel part of their own project."

Drool-worthy direction

From the very beginning, Helium was excited by the project's potential for doing something fresh. Diving into a marketplace markedly different from their usual work, they sensed an opportunity to buck a few trends. Early market research yielded few competitors in drip pop's space that had any leading concept, design or story within the industry. This would enable them to recognise what other companies were lacking, chiefly being any personality, life or over-arching concept. "Fortunately, we had done the initial brand development for drip pop so we were able to really tell a visual story with some great play and exploration. All of this background process set the stage for us to seamlessly create a website that mirrored the same energy as the brand foundation. Paint splatter, hand-drawing, custom artwork, quirky icons, are just a few of the brand elements that gave us the flexibility to concept a really engaging website." For Kelly, this insider brand development knowledge could be coupled with the wider market observations and kept in mind while foraging for design inspiration. This 'fun stuff' would include a broad sweep of all sorts of on and offline design sources, grabbing cool references like a creative magpie. "No matter the industry, no matter the platform, this is where I dive in the deep end of Pinterest feeds, Behance mini GIFs and trending sites on CSS Design Awards. I usually get my interaction

"We had the opportunity to play with food both in real life and on the computer"

TALKING THE WALK

"I think the biggest challenge during the process of development with this project was translating what was in my head for all the different layers in the design," laughs a reflective Kelly Gedvilas. "Then describing how I wanted them to interact with each other, which if you ask any designer to translate what's in their head into developer lingo and you'll often get something totally different." It's a valid point honestly made, with that bridging of the design and development divide proving problematic for most digital projects. Thankfully for the production of drippop.com, the team wasn't building a sprawling, complex, critical web app that would demand excessive coding. The development phase was much more about robustly bringing life to the frontend design elements, while ensuring consistency of performance across devices. "With that, I often thought, 'OK, how can I make this easier for both parties to understand and learn from, what exactly can't be done from a developer standpoint and what can we come up with instead so that the integrity of the design and user experience is still just as impactful?' The key is keeping the balance of what cannot be achieved through coding roadblocks and supplementing what will be the biggest and coolest impact for the customer."

ideas for a single site easily from a compiled list of over ten sites, liking bits and pieces from each one." Armed then with enough visual motivation, the challenge becomes focusing it to give drip pop's online presence enough distinction from the others. In a saturated market of sumptuous frozen treats, how could Helium's inspiration and branding work give a drool-worthy product a suitably mouth-watering website?

Practical challenges

"For websites, I usually sketch everything on paper beforehand," Gedvilas enthuses when asked about the visual development phase. "Everything. Navigation systems, footers, pop-ups, sort of in a Crazy-8's fashion just to get all of my ideas out of my head. Somehow, in some way, this process helps the transition onto the computer so much easier! I can quickly reiterate from my sketches to see what translates the best and just run with it." This project was also unique, being tied so tightly with a cohesive rebranding initiative that included product packaging. The graphics for all of the content were originally crafted for this purpose, while the team still wanted new renditions of the patterns to be implemented on the website. This would ensure that each area the customer sees, whether it's in a drip pop location, purchasing a box of pops, or visiting a kiosk, is a completely unique experience. "There was a lot of experimentation with the layering of each pattern and flavour because sometimes it worked and others simply didn't. All in all, however, this was a fun project to play with all of the brand elements with." Once this frontend design work is completed and approved by all the parties involved, Helium passes efforts over to its development team. Kelly concedes that this part of the process often takes the most time, typically to ensure everything is fully functioning properly on all devices, browsers, and responsive formats. "A lot of the time, we run into complications of what works on what device, and if there are issues, we need to come up with alternatives that won't compromise the user's experience with the site. The 'trial and error' period can then entail issues being fixed on one platform and then a problem where there was none before, so it's all about puzzle-piecing and problem-solving."

Launch phase licked

Fixing those issues is, of course, imperative before contemplating handover and launch, with the client enjoying a final 'sneaky peek' before that button gets pressed. "They don't receive the test link until the project acts and looks as a live functioning site. They test the website and once the site is approved as good to go from their perspective, it's just a matter of going live!" Naturally, a conscientious agency like Helium wants to set its clients up for success here, handling all of the backend development and setup this demands. This live launch and handover process is therefore pretty seamless to say the least, usually depending more on the circumstance the client is in.

lollipop gild

Site Highlight

Helium Creative's Experience Designer Kelly Gedvilas reveals a particular project achievement that represents a source of pride for the team after all their hard work.

"I think the standout feature for this project is the scrolling interactions. I think it makes all the pages and content flow cohesively. The shifts between parts on a single page are fun to go back and forth on, although I think that was the most challenging portion of the development phase. Getting those exact took quite a bit of time but totally worth it in the end!"

"For instance, if this is a client for a rebrand, we typically do a necessary introduction before launch and the same goes for a client that is relaunching under a new name. For drip pop, it was blank canvas and just a matter of putting the brand out there for recognition!" Thankfully, that attention has been a two-way street, with Helium enjoying an equally encouraging helping of recognition for the live site as the client. Honours including CSS Design Awards Website of the Day, as well as Best Innovation, Best UI Design, and Best UX Design prizes for the website, have been joined by numerous brand and packaging accolades. Such success vindicates the project's focus on this being the start of a lasting working relationship, with ongoing maintenance of web projects something Helium's clients can expect. In drip pop's case, this constitutes additional updates such as adding new flavours, stores and locations, fixing any possible bugs, and keeping the site up to date. "Right now, this company is still super-new and they are finalising a lot of the framework to flush out the rest of the brand to the world," Kelly concludes. "Drip pop has plans to sell in major supermarkets and opening kiosks in South Florida, but for the time being the website itself is serving as its marketing tool for creating initial brand awareness." ■

"Naturally, a conscientious agency like Helium wants to set it clients up for success"

SUBSCRIBE TODAY

and receive a REUSABLE COFFEE CUP & TYPOGRAPHY E-BOOK

Enjoy sipping on a cup of coffee while being inspired by the 100 Best Typefaces Ever. Each cup is environmentally friendly and can be taken wherever you go!

IT'S EASY TO SUBSCRIBE

myfavouritemagazines.co.uk/WCUEB

OR CALL: 0344 848 2852 AND QUOTE **WCUEB**

SUBSCRIBING FROM OUTSIDE THE UK?

HEAD TO **BIT.LY/2PSTN7N** TO REDEEM YOUR E-BOOK!

PLUS with your subscription you have access to **900+ FREE RESOURCES** from previous issues with FileSilo. **See page 96**

CHOOSE YOUR PACKAGE!*

PRINT EDITION ONLY

ONLY £17 QUARTERLY

3 months of **Web Designer** in print, plus **ecoffee cup**

DIGITAL EDITION

ONLY £15 QUARTERLY

3 months of **Web Designer** on your iOS or Android device

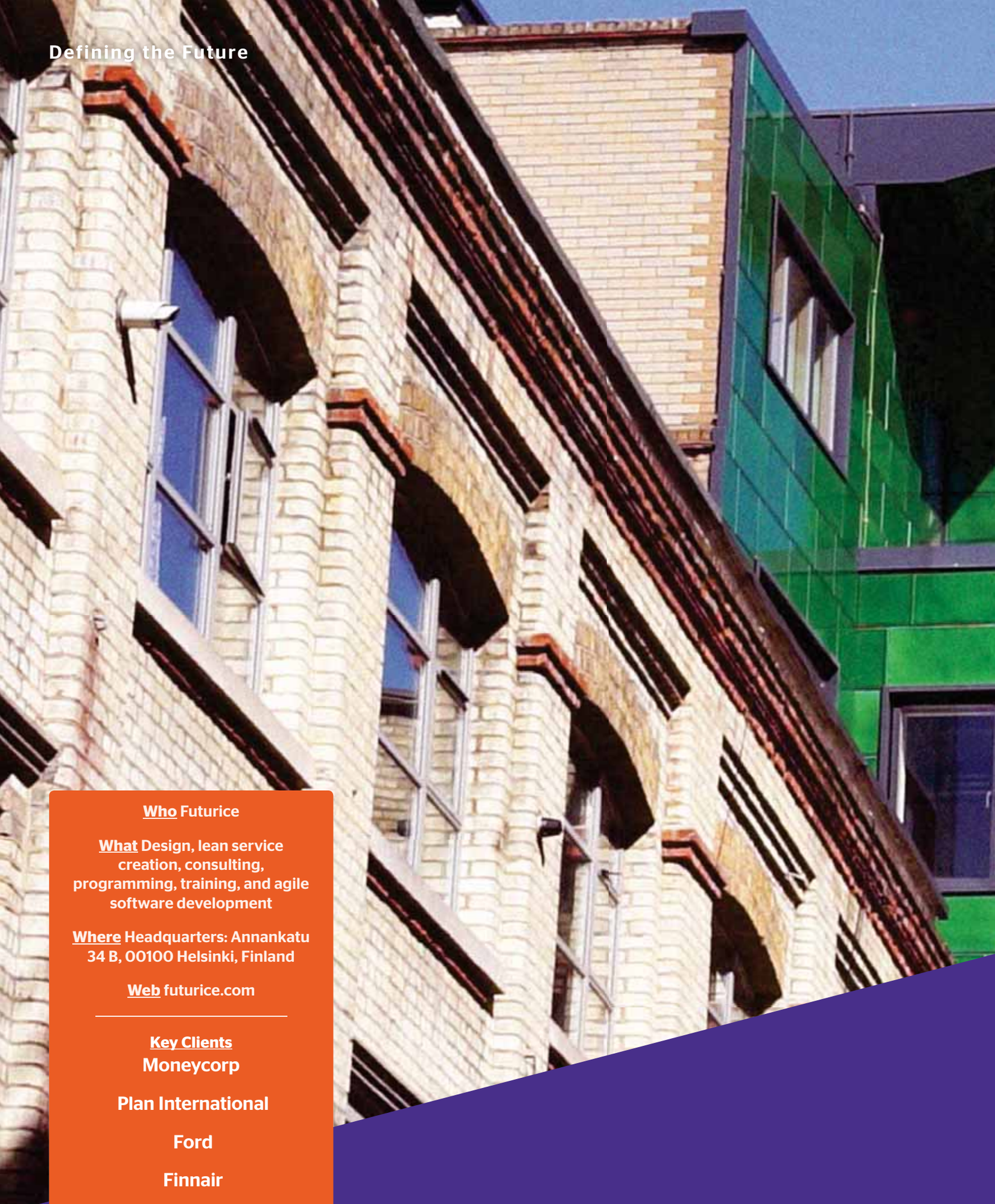
PRINT & DIGITAL BUNDLE

ONLY £24 QUARTERLY

3 months of **Web Designer** in print and digital, plus **ecoffee cup**

*** OFFER NOT AVAILABLE FOR DIGITAL EDITION ONLY PACKAGE**

Terms and conditions: This offer entitles new UK Direct Debit subscribers to pay just £17 every 3 months plus receive a reusable coffee cup and e-book. Gift is only available for new UK subscribers and is subject to availability. Please allow up to 60 days for the delivery of your gift and provide a valid email address to redeem your e-book. 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 December 2018.**



Who Futurice

What Design, lean service creation, consulting, programming, training, and agile software development

Where Headquarters: Annankatu 34 B, 00100 Helsinki, Finland

Web futurice.com

Key Clients

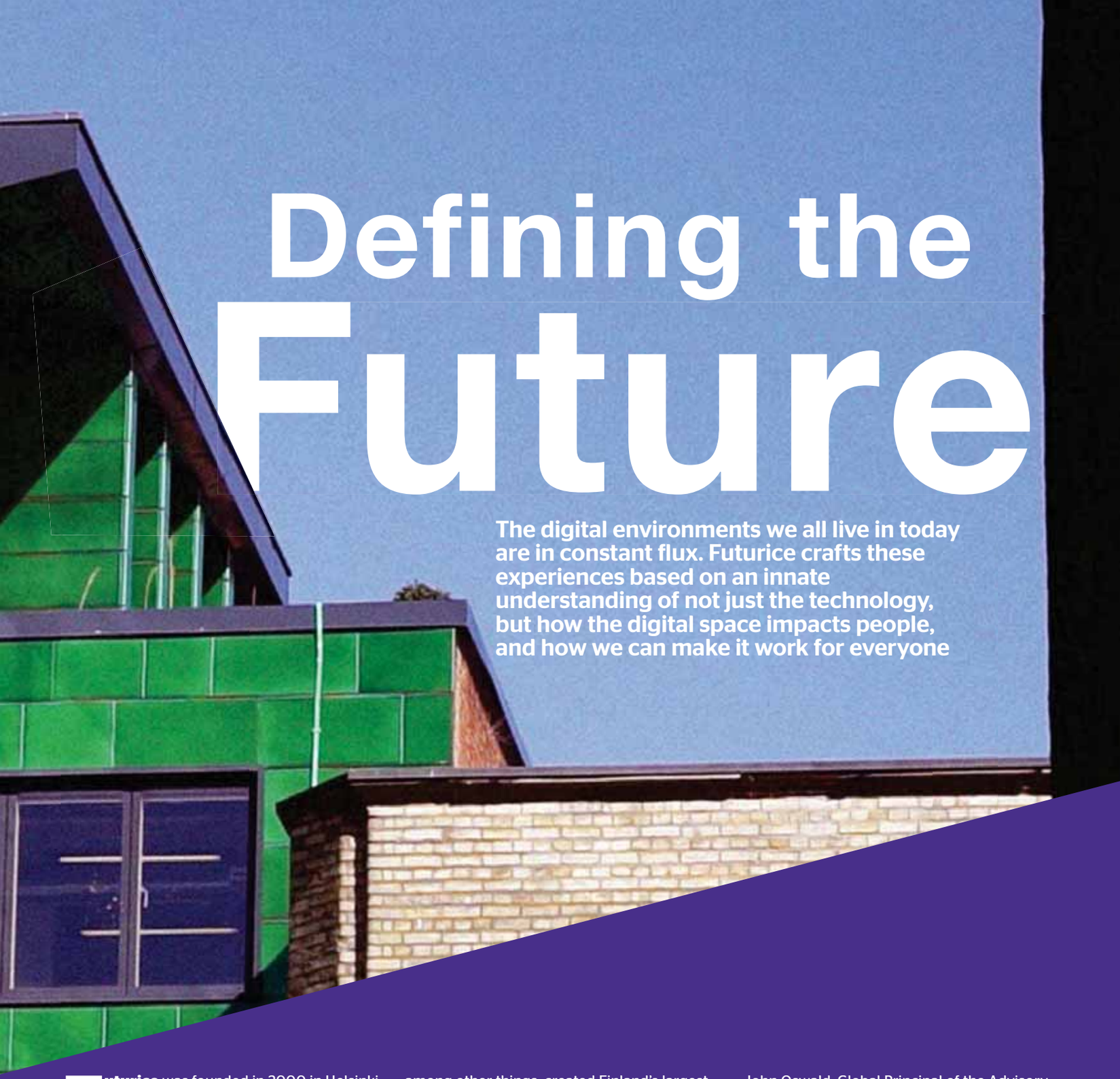
Moneycorp

Plan International

Ford

Finnair

E.ON



Defining the Future

The digital environments we all live in today are in constant flux. Futurice crafts these experiences based on an innate understanding of not just the technology, but how the digital space impacts people, and how we can make it work for everyone

Futurice was founded in 2000 in Helsinki, Finland by Tuomas Syrjänen, Hanno Nevanlinna, Mikko Viikari and Markku Taulamo while all four were still pursuing their academic degrees at what was then known as Helsinki University of Technology (currently Aalto University).

The founders majored in a variety of disciplines: computer science, electrical engineering, chemical engineering and shipbuilding. Three of them — Syrjänen, Nevanlinna and Viikari — are still with the company, and all four remain the company's majority shareholders.

Futurice initially focused on solutions and software development for early mobile platforms in the pre-smartphone era, and among other things, created Finland's largest photo-sharing service, Kuvaboxi, in the early '00s which was eventually sold to MTV3, a Finnish media company. Futurice has since evolved into an international digital consultancy that combines tech, design and advisory, with 500 professionals working in seven offices across five countries.

But that's just the business side of things. The founders also wanted to set up a company with a new set of values — transparency, trust and empowerment. And that's the part that stuck the most — the product and market fit may vary slightly between countries, but the culture is what helps Futurice stand out.

The founders like all new business startups needed to find a name for their new enterprise. John Oswald, Global Principal of the Advisory Team explains their approach: "Futurice registered the **futurice.com** domain name around the same time as the company was founded, in August 2000. But that's the simple part. The name came first — it signifies the 'Future' of 'ICE': Information, Communications and Entertainment. It's led to some internal in-jokes along the way. Pronunciation varies from country-to-country, but most settle on FuturISS."

A presence online is of course vital for all agencies. However, as John continues, for Futurice, their website is much more: "Our site is a big public statement," he states. "Given our focus on trust, transparency and empowerment, though it's tricky

getting our public-facing website exactly right for everyone!

"We recently went live with a new articulation of the company and our values, and we're pretty proud of it, although it was quite a journey getting there. In a company like ours where everyone's view matters, and is taken seriously, it's a tricky thing to summarise ourselves in a single website. But it's super-important to us to present the best of who we are and what we do, as well as our culture.

"The primary objectives of the site are to be interesting and insightful to our clients, attract new talent to the teams, and to give our people the platform to blog, connect and be noticed. It takes quite a bit of looking after — between our marketing team of seven, and a blogging team of 500, it's almost impossible to say how much time and effort goes in — but it's worth it!"

As the stated goals of Futurice are transparency, trust and empowerment, the work that the company has produced over the last 18 years has been diverse to say the least. However, **Web Designer** asked John if there are accounts that speak directly to the ethos of the company?

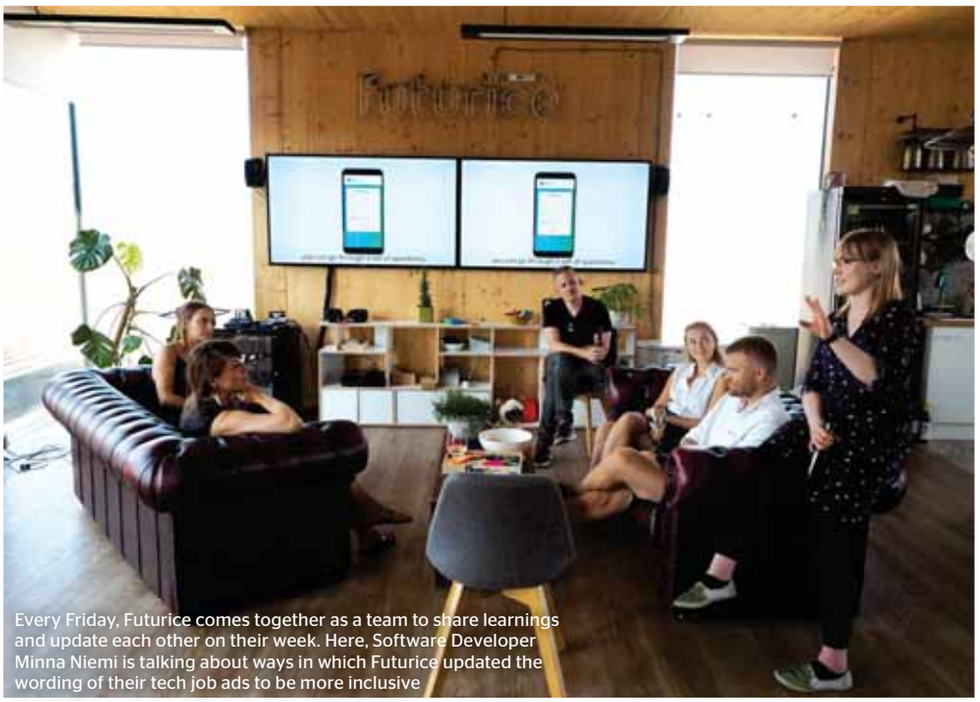
"We're particularly proud of the work we recently did with Finavia, Helsinki-Vantaa Airport's operating body, and Finland's flag carrier Finnair, in designing a prototype for the world's first face recognition check-in desk system," John explained. "The test run was conducted over a period of three weeks in May, to follow up on a preliminary trial last year.

"A thousand frequent flyers were invited to take part, with participants using a trial app to send their face profiles to the test software

> "We draw the line occasionally with clients who, for example, specialise in defence"

platform. The travellers were then forwarded to a designated check-in desk containing the facial recognition tech, which turns facial images into untraceable biometrics IDs, enabling Finavia to identify registered passengers on-the-go without having to store images."

John concludes: "The test will also provide useful information on the use of this solution for environments with large customer flows and tight security needs. This project represents Futurice best because it was a combination of all three of our passions: finding the right solution that works for people in their various contexts (the best of design); exploring emerging applications of technology (the best of tech); and establishing a business model that works (the best of our advisory people).



Every Friday, Futurice comes together as a team to share learnings and update each other on their week. Here, Software Developer Minna Niemi is talking about ways in which Futurice updated the wording of their tech job ads to be more inclusive

"It also shows how much we rely on our peoples' passions when developing our business. Biometrics — and facial recognition in particular — is an area that our Head of Innovation, Tuğberk Duman, is really into and started developing as a business independently and with minimal investment, which was a very smart move. Things that percolate up from our organisation are frequently awesome and we really rely on these signals in the development of both our business and culture."

The steps that agencies take to attract clients can be manifold. As a multi-country business, for Futurice a clear focus on the sectors they want to work within is a core driver, as John explains: "We're very well known in Finland with a lot of clients, so there's a healthy ongoing dialogue, which means that business flows in both directions.

"In other countries — particularly in Germany and the UK — we're constantly pitching for work and networking with clients and potential clients. But we're gradually getting better and better known wherever we work. We also like to have a point of view on what's happening in the world — hence the importance of blogging and having a voice."

In addition, John says: "We encourage literally everyone in our company to have that voice. And we're all empowered to go out and meet people, build trust and help solve problems. Finally, a lot of our work comes to us from our open-source efforts. We've designed open source ways of working, like Lean Service Creation, which generates a lot of buzz. Slightly hilariously a client recently asked us if we could demonstrate proficiency in Lean Service Creation. Naturally we could, given we designed the toolset."

The projects that an agency takes on need to fulfil the practical needs of the business, but they also need to feed the imaginations of everyone working with the client. Balancing business and creative needs is constant across Futurice. "As a relatively small business, we don't always have the luxury of picking and choosing, but we tend to build relationships with clients where our values align very nicely," says John. "We draw the line occasionally with clients who, for example, specialise in defence (and we have a healthy internal debate about it, too). We're basically here to help our clients, though, so we're more than happy to do something small like a bit of training or maybe a workshop, all the way through to getting involved with longer-term initiatives."

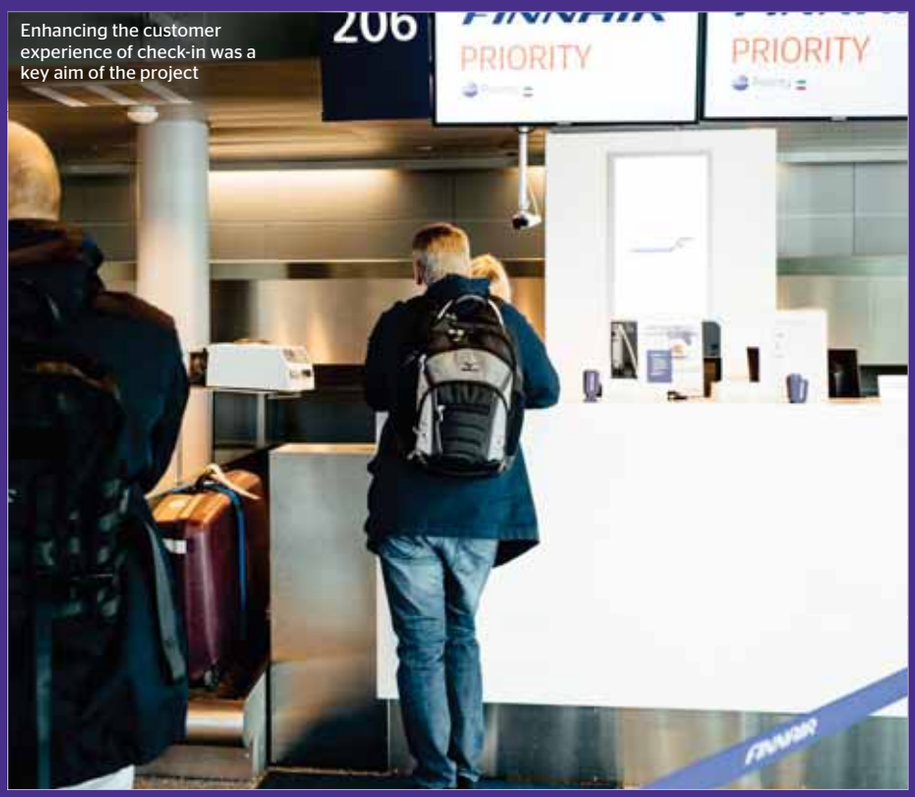
Approaching each new project means assembling the right talents to deliver on the client's brief. John explains their approach: "The whole ethos of the company is multi-disciplinary teams coming together with our clients to co-create a way to solve a business

“ We recently went live with a new articulation of the company and our values, and we’re pretty proud of it, although it was quite a journey getting there. In a company like ours where everyone’s view matters, and is taken seriously, it’s a tricky thing to summarise ourselves in a single website. But it’s super-important to us to present the best of who we are and what we do, as well as our culture ”

John Oswald
Global Principal of the Advisory Team

Using biometric IDs begins with a facial image that is then used to facilitate the passenger’s hands-free check-in.

Finnair, Finavia and Futurice experimented with ground-breaking technology which has the potential to rejuvenate the entire departure experience.



Enhancing the customer experience of check-in was a key aim of the project

Finnair and Finavia

finnair.com and finavia.fi/en

In an industry-first attempt to enhance the customer experience using biometrics, Futurice worked with Finnair and Finavia — Helsinki Airport’s operating body — to build and test a facial recognition system for ‘hands free’ check-in.

The team used feature-based facial recognition technology, which turns facial images into untraceable biometric IDs, enabling the system to identify registered passengers on-the-go without having to store their images.

In a competitive environment like aviation, safe travelling, comfortable aircraft and good onboard service are taken for granted by the passengers. To gain a competitive advantage, a service provider must further enhance the customer experience in unprecedented ways, through innovative interactions and data security.

The system was used for a time-boxed pilot project at Helsinki Airport. By testing the new system with real passengers, the team proved that it has the potential to rejuvenate the entire departure experience and provided information on the use of this solution for other environments with large customer flows and tight security needs.

Also, the pilot project positively impacts the brand image of Finnair and Finavia, helping them to be perceived as customer experience champions and pioneers of new technologies.

Nokian Tyres
nokiantyres.com/snapskan

Worn tyres end lives — Nokian Tyres and Vianor wanted to change that. The hypothesis was that improving drivers' awareness of the condition of their tyres, and their impact on the safety of their loved ones, should decrease road accidents, and increase brand awareness and tyre sales.

Through research, validation and testing, Futurice designed an IoT service — SnapSkan — that would automatically measure the condition of the tyres, then offer the driver the option of accessing the data, while supplying information about the impact of tyre condition on passenger safety and the option to purchase new tyres. Snapskan uses 3D scanning technology, machine vision and the newest web technologies to measure tread depth and tyre safety. A busy Q-Park garage in Helsinki served as the staging ground for the pilot. The results obtained proved that people are willing to buy tyres in conjunction with the SnapSkan measurement. SnapSkan also provided Nokian Tyres with a new channel for sales outside the market served by tyre storage facilities.

SnapSkan uses the latest web technologies to monitor tyre tread depth and safety

The SnapSkan service provides drivers with a new level of safety and Nokian Tyres with a new sales channel

" Adobe software is still used, but we use mostly the newer generation UX/UI focused tools such as Sketch, InVision, Zeplin, Principle, Sigma, Proto.io, to name a few. We try to complement the design work with useful collaboration tools such as Abstract or RealtimeBoard. We also have quite an extensive 'physical' toolkit that we always make use of in projects – anything from the Lean Service Creation canvases to the IoT Service Kit — all of which we have shared under open licenses "

Mikko Heikelä
Technology Director

problem. It's hard to pin down exactly what the flow or the process is, as it will adapt and shift, but you could characterise it as designers, technologists and advisors working in a small team from initial concept through to detailed design, build and integration, then through to launch.

"Not every project follows this pattern, of course: some will start straight from helping a team build a new feature or refactor an existing code base; others will start from the initial concept and simply provide recommendations. But all of our projects start with a relationship with a client who needs our help, and however small or large, we'll do that. Typically, our ethos rubs off and clients want to see more."

The diverse nature of the work Futurice completes for its clients means not only choosing the right people for the team, but also the right tools, which can be diverse. Mikko Heikelä, Technology Director outlined their current toolset: "When it comes to software choices, we want to empower teams and individuals. Every designer has the freedom to choose the tools they feel are the best choice for the project, as long as this doesn't affect team collaboration in a negative way.

Mikko continues: "Adobe software is still used, but we mostly use the newer generation UX/UI focused tools such as Sketch, InVision, Zeplin, Principle, Sigma, Proto.io, to name a few. We try to complement the design work with useful collaboration tools such as Abstract or RealtimeBoard. We also have quite an extensive 'physical' toolkit that we always make use of in projects — anything from the Lean Service Creation canvases to the IoT Service Kit — all of which we have shared under open licenses."

As technologies have evolved and diversified, how agencies create the assets and environments for their clients has also radically changed. "Growing browser support has made at least some CSS3 and HTML5 mainstream," Mikko comments. "While new technologies are applied to websites and pages without a lot of interactivity, most of our design and build projects focus on more complex web applications.

"In this area, React has been a very strong framework for years now, but our teams are also constantly looking into alternatives from the likes of VUE to more radical options like using Elm or even ReasonML. Complex web applications nowadays are rarely built with jQuery and JavaScript without higher level libraries. Within the React ecosystem, we have found styled-components to be a pleasant way to structure styles and keep the implementation of a design system or a pattern library manageable.

"Other technologies that are getting more and more interesting include the building

"Our teams are also constantly looking into alternatives from the likes of VUE"

blocks of Progressive Web Apps, such as service workers and web app manifests. For complex applications, using a strongly-typed language is becoming the most popular approach, and TypeScript is a strong choice right now. Longer term, WebAssembly holds the promise of allowing a significantly broader variety of programming languages to target the web browser."

Today the phrase 'mobile first' seems very overused. For Futurice, the platform will always come second to the needs of the target users, as Mikko explains: "Design always starts from the user and their needs, and the context in which we expect the app or service to be used. For simple services it is easier to find a common design approach that works for mobile, tablet and desktop use. Whereas, for more complex services it becomes increasingly important to make use of the capabilities of each platform in an optimal way.

"The delivery channel of the service or app matters in that it sets expectations. Mobile users don't expect every web page to follow their platform's guidelines and idioms, but the situation is different for apps installed from app stores, which has often been a problem for hybrid technologies. It will be interesting to see how these expectations evolve when Progressive Web Apps become more commonplace and harder to distinguish from native applications."

Mikko concludes: "The range of interfaces to consider has recently expanded beyond different screen sizes and input devices. Depending on the service, it may be relevant ▷

Timeline

2000
Tuomas Syrjänen, Hanno Nevanlinna, Mikko Viikari and Markku Taulamo start their business developing image products and small applications.
Employees: 5

2003
Futurice builds hit B2C imaging service Kuvaboxi and creates bespoke agile software development for clients.
Employees: 15

2008
Futurice focuses on consulting in tech and design. Also works with bigger clients including Nokia. A new office in Tampere opens.
Employees: 50

2010
Futurice expands internationally, beginning with Berlin. And Futurice celebrates its tenth birthday.
Employees: 115

2012
First of two consecutive #1s in Great Place to Work Europe's list for SMEs. London office opens.
Employees: 180

2013
CEO Tuomas Syrjänen chosen as leader of the year by Finnish business magazine *Fakta* for the unique culture Futurice has created across its business.
Employees: 225

2014
Futurice starts employee-initiated Spice Program: people are paid for open-source contributions in their free time.
Employees: 275

2015
Futurice enters the VR market by building a VR time machine for the Helsinki City Museum.
Employees: 325

2016
Employee-driven Chilicorn Fund launched, an open-source social responsibility initiative. New offices in Stockholm and Munich open.
Employees: 375

2017
Groundbreaking facial recognition pilot at Helsinki Airport with Finnair and Finavia created.
Employees: 450

2018
New AI initiative launched, headed by former CEO. New CEO Teemu Moisala takes the helm and a new office in Oslo opens.
Employees: 500

Futurice and E.ON built an end-to-end digital sales and delivery platform for renewable energy

E.ON

eon.com

E.ON challenged Futurice to help them develop and build an end-to-end digital sales and delivery platform for renewable energy. Together, they created a seamless and transparent customer experience for buying solar panels and home batteries, called 'Project Silicon', via a set of digital touchpoints. These touchpoints help customers calculate the potential benefits and investments, keep customers fully informed of their order status and provide a timetable for installation.

On the trade side, 'Project Silicon' makes the sales process smoother for its delivery partners, by engaging new customers and converting leads into sales, while a quality assurance app — designed specifically for installers — takes care of their needs.

Futurice and E.ON digitised the entire sales pipeline from lead generation to delivery, reducing the cost of acquiring new customers, while increasing the number of leads and sales, and slashing installation times for solar panels. The team ensured the propositions could be scaled, including frontend and backend development, and new features and functions could be added easily.

Key to E.ON's transformation was shifting the business from its traditional focus, towards building a culture in which digital services can be prototyped and deployed faster than at conventional service providers, with increased customer-centricity.

> “We look for people who thrive on transparency, who are able to trust themselves, each other and our clients”

to figure out how it should feature in a broader variety of channels, from chat apps and social media posts to smart watches, TVs, virtual reality glasses and voice assistants.”

And what technologies are exciting Futurice at the moment? “In addition to the many frontend technologies discussed above, there is an explosion of different libraries for managing state in frontend applications and simultaneously trying to offer good support for type checkers, integrations with GraphQL, etc. Io-ts, mobx-state-tree, apollo-link-state are just a few examples from different parts of this space.

“GraphQL itself is interesting in how it allows a different division of responsibilities between a frontend app and an API, which is more flexible and potentially more convenient than, for example, a traditional REST API. Within React, the asynchronous rendering and updated context API should allow the simplifying and unifying of some patterns that have made the ecosystem complex.

“For startups and small teams, hosting options that don’t require operating system and server-level maintenance are great enablers. Heroku was a pioneer for this approach for apps that require bespoke backend logic, but Netlify has made a strong entry as a more modern challenger.

“Moving further from frontend tech, we expect data-driven personalisation, recommendations, and other machine learning-based features to become a big part of future web applications. These can, in principle, be achieved with very different technical approaches, from relatively low level open-source libraries to features in existing product platforms such as CMSs or eCommerce products, to cloud-based machine learning services. Each approach places different requirements for the skillset in the development team and the overall architecture, and it is not yet clear which approach will be the most fruitful.”

An agency is effectively only as good as the people it employs. What qualities do you look for in a prospective employee and what advice would you give to anyone looking to take a step into the industry?

John explains: “This is pretty much the heart of our business — the people who make up our family. We look for people who thrive on transparency, who are able to trust themselves, each other and our clients, people who want to collaborate and who aren’t afraid to share, iterate and improve what they do. And, almost above all, we look for people who are naturally curious, who care, who want to make the world a better place and who take the time to have hobbies they are passionate about.”

As an agency with clear ambition, what does the future look like for Futurice? “Projects first,” John concludes: “We’re embarking on an engagement with Plan International: they’re currently leading an amazing initiative to design and build an open-source civil registration Global Good, which we’re helping to design and build in Bangladesh as a first reference implementation. It involves our advisory, design and technology skills to uncover the human needs and frictions around birth and death registration and how to respond to them in a technology platform that, once complete, could help ensure that millions of children are properly registered and avoid future exploitation and harm.

“We’re always on the lookout for opportunities to expand, having recently opened our newest site in Oslo. And, most importantly, we’re always keen to provide a platform for our people to try out something new, which we can either turn into a new competence area to enrich the core of our business, or to spin off and launch as a new venture within the family.

“Examples include Futurice XR, a growing initiative that focuses on virtual, augmented and mixed reality technology, as well as our subsidiaries, growth consultancy Columbia Road, and the recently launched **Aito.ai**, which is an AI startup aiming to provide a simple and effortless way for any business to rapidly implement and experiment with machine learning.”

As Futurice states, you can’t predict the future so this agency creates it. The breadth of their understanding of the digital spaces they define is masterful. Coupled with an innate sense of the social, economic and emotional connections that digital environments can contain, Futurice innovates to build our future. □

futurice

futurice.com

Founders
Tuomas Syrjänen, Hanno Nevanlinna, Mikko Viikari and Markku Taulamo

Year Founded
2000

Current Employees
500

Location
Helsinki, Berlin, London, Munich, Stockholm, Tampere, Oslo

Services
Digital visioning

Human-centred design and build of experiences, services and products

Business application of emerging technology

Lean and agile ways of working

Organisational transformation

Data science and engineering

Agile Software development

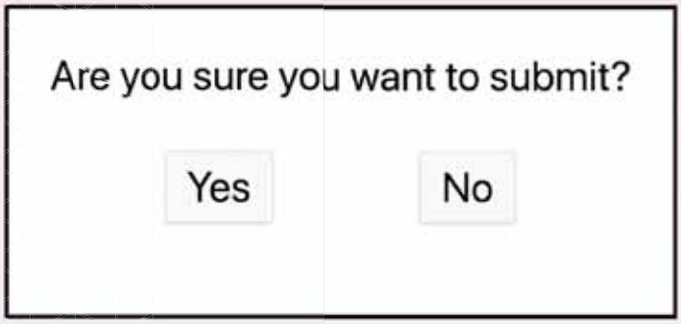
30 BEST

SEMANTIC MARKUP IS VITAL FOR ACCESSIBILITY, SEARCH ENGINES AND MUCH MORE - BUT IT IS SO OFTEN MISUSED. BUILD A STRONG FOUNDATION AND TREAT CSS AND JAVASCRIPT LIKE AN ENHANCEMENT.

Matt Crouch
Software Engineer at Vidsy
@mattcrouchuk, *www.mattcrouch.net*

CSS & JS APIS

HTML IS MORE CAPABLE THAN EVER BEFORE. MAKE SURE THAT YOUR SITE IS MAKING THE MOST OUT OF THE ELEMENTS AVAILABLE TODAY AND DON'T FORGET TO ENHANCE WITH CSS AND JS



Forms inside a dialog can submit with a 'form' method. The 'returnValue' of the modal is the value of the button used to submit

<DIALOG> https://developer.mozilla.org/en-US/docs/Web/HTML/Element/dialog

Display a popup or modal window without the overheads

A common design pattern on the web is to have an extra window display to provide more information or options for a complicated interaction. Adding a confirmation as an extra step is also a good way to make a user aware their action will have consequences.

These windows have been around for years through the help of libraries such as Bootstrap or jQuery UI. While they can be easy to implement, they often require heavy scripting and extra styling to match the look and feel of the site they are implemented into.

One of the few elements added in HTML5.2 is <dialog>. This new, semantic element is designed to denote a supplementary, interactive component that displays out of the main flow of the document.

```
<dialog open>
  <p>Dialog Content</p>
</dialog>
```

The <dialog> element itself is designed to be as simple to use as possible. Any content within the tags will become part of the window and do not appear on the page by default. When the element has the 'open' attribute applied, it then appears centred based on where it appears in the DOM.

While dialog boxes can request a response, a user can still navigate the page without having to interact with it. A modal window is similar to a dialog, but has the aim of requiring an action from the user before proceeding. This makes it useful for potentially destructive actions, such as deleting an account.

```
const dialog = document.
getElementById("modal");
dialog.showModal();
```

Modal windows can only be triggered using JavaScript. Once opened, the window appears completely centred within the user's screen and dims the rest of the page. The only way to close it will be by either pressing the escape key or by calling the 'close' method on the element.

Non-native modal implementations often forget about the content behind the modal. For keyboard users, their browser can start focusing and interacting with elements underneath the modal that cannot be seen. Native modals make the rest of the content inert, making sure the focus stays within the window before being dismissed.

Browser support is currently limited to Chrome, Opera and Samsung Internet browsers. As unsupported browsers treat unknown elements like a , it makes it possible to add in this behaviour when needed. The Chrome team have put together a polyfill, which can be found at **https://github.com/GoogleChrome/dialog-polyfill**

8 ESSENTIAL SECTIONING ELEMENTS

<MAIN>

https://mzl.la/1KLTqoe

This marks up the core content of the document. In contrast to any header, footer or navigation elements, its content will vary from page to page. There can only be one <main> element visible at any time.

<NAV>

https://mzl.la/2cfIHsH

This represents any area of a document that is responsible for navigation. This can be a site's main navigation or a grouping of internal links such as a table of contents. Not all links need to be inside a <nav>.

<HEADER>

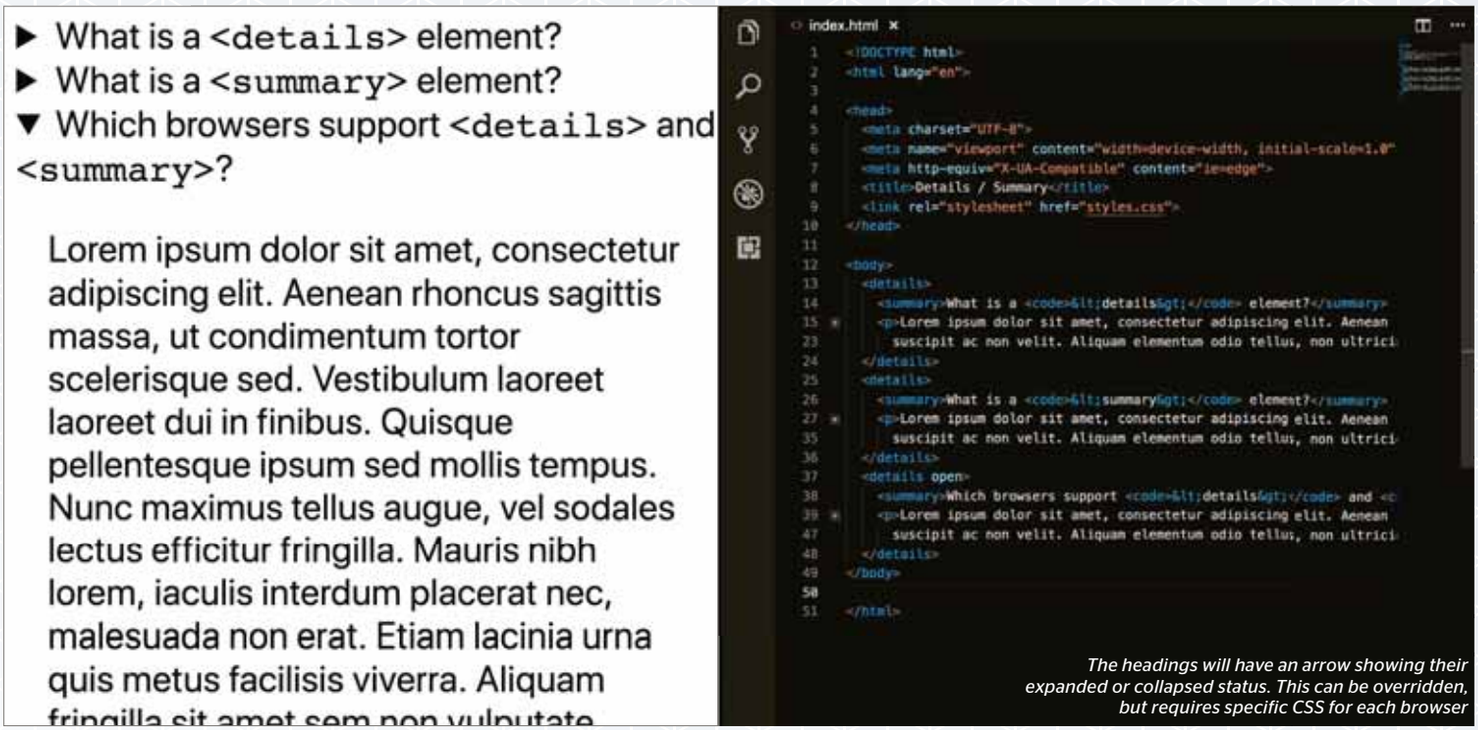
https://mzl.la/2FIMeyS

Use a <header> to separate any kind of introductory content from the rest of the document. It is commonly used to define page headers, but alternatively it can also be used to add headers to sub-sections.

<FOOTER>

https://mzl.la/2PIYUXC

In contrast to <header>, the <footer> element marks the final content of a page or section. This would typically hold extra information like author or copyright information, along with any related navigation.



The headings will have an arrow showing their expanded or collapsed status. This can be overridden, but requires specific CSS for each browser

<DETAILS> & <SUMMARY> https://developer.mozilla.org/en-US/docs/Web/HTML/Element/details

Show and hide extra content under a collapsible heading without using JavaScript

The accordion is a common user interface pattern. It serves as a way to fit potentially lengthy content in a small space, providing only some of it needs to be immediately visible. These work best for a table of contents, a set of frequently asked questions or any other accompanying information such as directions to a location.

While many libraries exist to help solve this problem, it can also be achieved without any JavaScript at all. Content can be hidden inside a <details> element and therefore be toggled visible when clicked.

```
<details open>
  <summary>Heading</summary>
  <p>Some extremely long content…</p>
</details>
```

Each <details> element denotes a single collapsible area. Any content inside that block will be hidden by default until the 'Details' heading is clicked.

Adding an 'open' attribute will expand the block. This can also be triggered through JavaScript as a way to reveal certain information to the user, such as an answer to a specific question. Adding a <summary> to the top of the block will replace the default heading to the contents of that element. That heading then becomes interactive, which makes the contents inside keyboard accessible. While it can have almost any value, adding other interactive or clickable elements such as <label> or <button> will override the collapsing behaviour and break the element.

Multiple <details> elements can also be nested without issue. This could make it useful for nesting sub-links or hiding supplementary information within a block.

According to the specification, <details> should be limited to additional information or controls rather than anything considered important to read. In some instances, a more semantic element such as <dl> for key-value pairs may be more suitable.

Both <details> and <summary> are available in all browsers apart from Edge and IE. For these, information will display expanded by default, or a JS fallback can be used.

Accurately define page structure and each element's role by using these semantic tags

<ASIDE>

https://mzl.la/2NI2vOY

Designate an area of a document that contains supplementary information about the main content. While this is often styled to look like a sidebar, it can also be used to define accompanying content such as a related fact.

<ARTICLE>

https://mzl.la/1KgymGv

Use <article> when the content it will display is self-containing, for example, a blog post or even a news story. There can be multiple <article> elements on a page and there are no limits on where they can appear.

<SECTION>

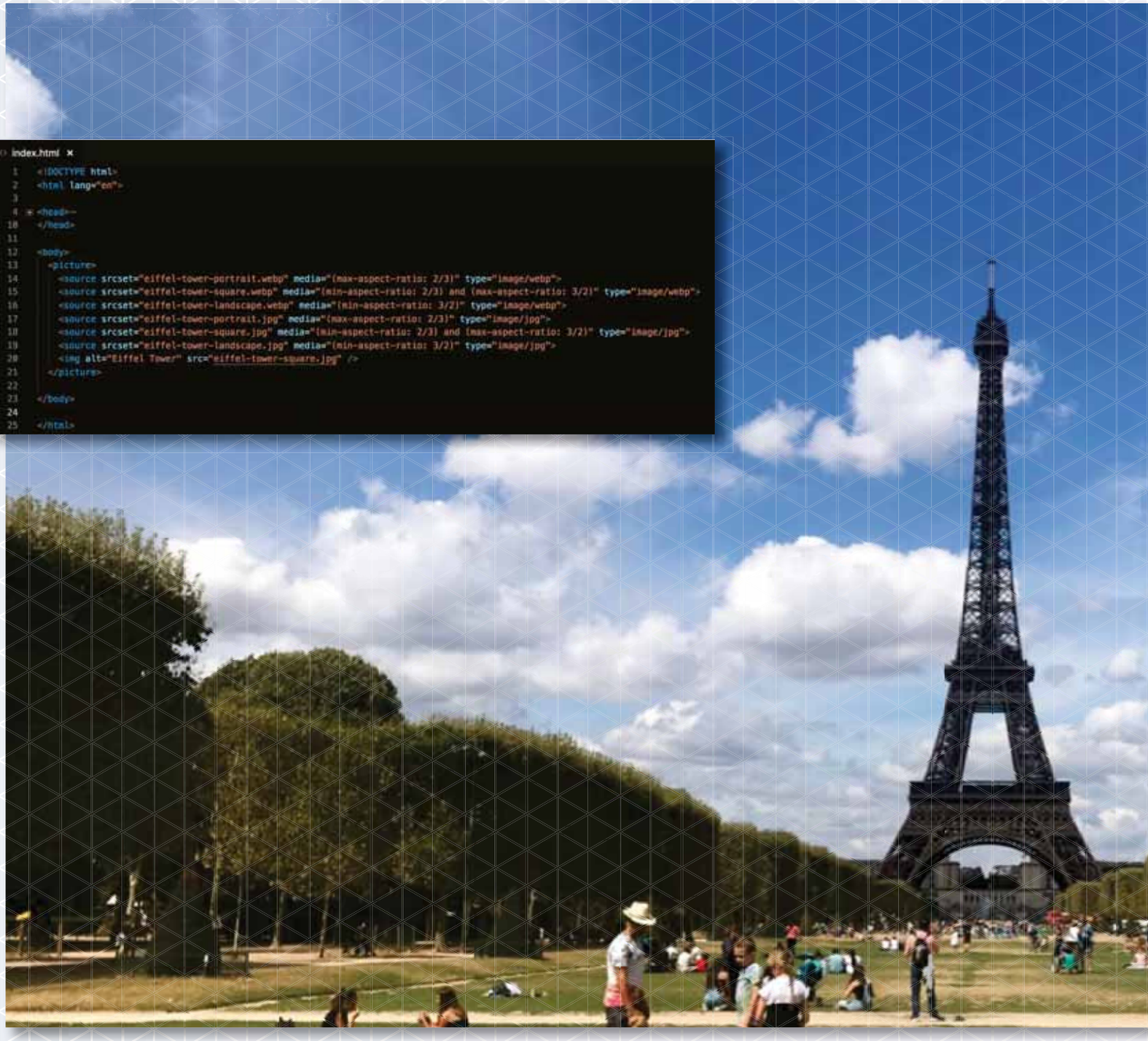
https://mzl.la/2j561ii

This represents a grouping of content within a document where no other element is suitable. This provides more meaning than as it is specifically defining an area as content over simply added markup for styling.

<H1> to <H6>

https://mzl.la/2ISYvHj

Heading elements have been around for a long time, but make sure they are being used correctly inside sectioning elements. Each section can have its own heading hierarchy, meaning multiple <h1> tags can now appear on a page.



<PICTURE>

https://developer.mozilla.org/en-US/docs/Web/HTML/Element/picture

Respond to different viewports and serve specialised content to serve them better

When images need to be responsive, sometimes it isn't enough simply to resize an image at certain breakpoints. On larger screens, the image can show as distorted and blocky but on smaller screens it could result in downloading a much larger image than is required.

Where images are used for informational purposes, it may make more sense to show an image adapted especially for a certain screen size or type. For example, larger screens may benefit from a fully annotated diagram, while smaller screens can get away with using coloured labels instead.

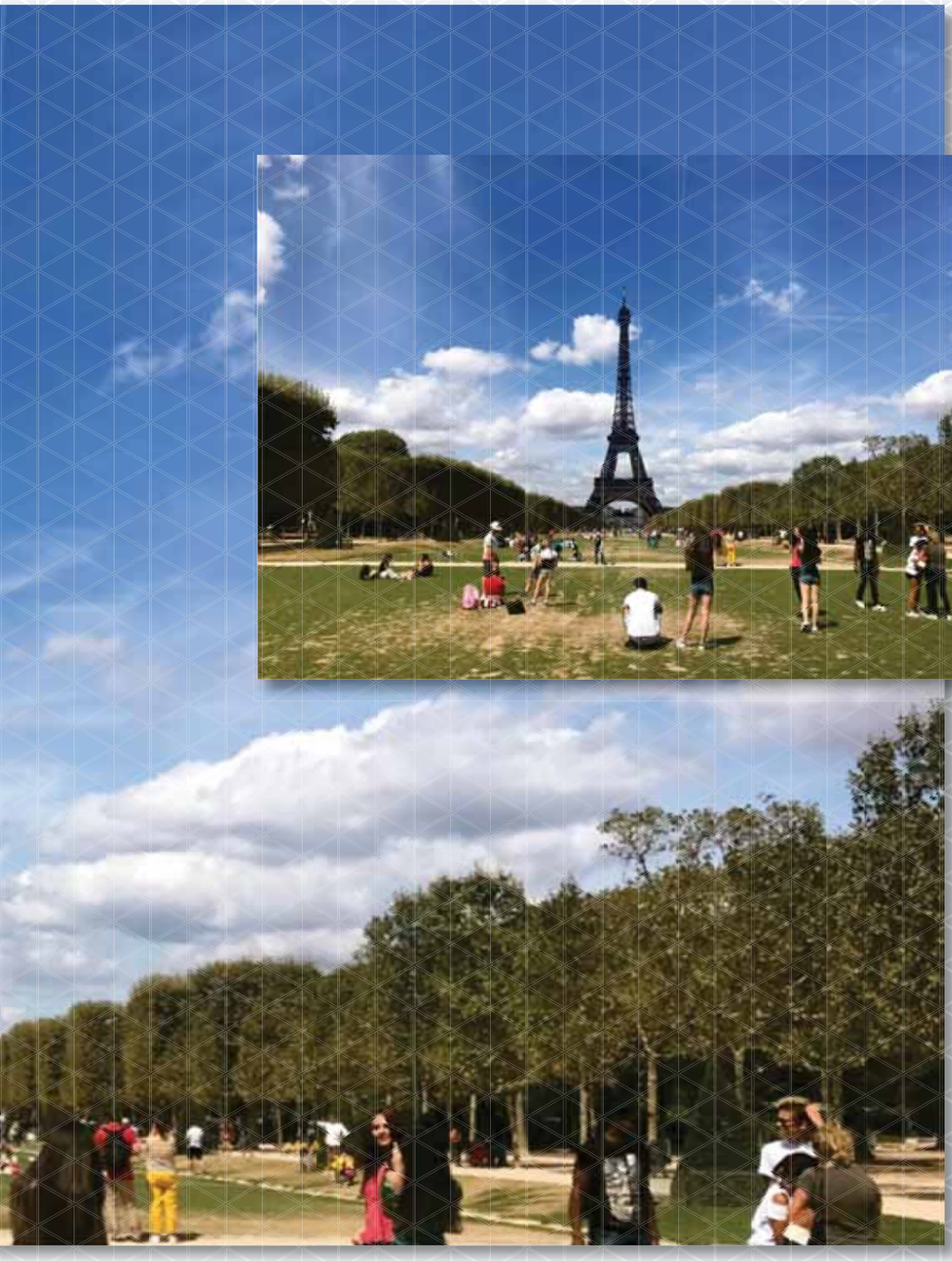
```
<picture>
  <source srcset="paris.webp" media="(max-
aspect-ratio: 2/3)" type="image/webp">
  <img src="paris.png" alt="Eiffel Tower" />
</picture>
```

The <picture> element allows developers to define different sources for the same image. Based on the attributes passed to those sources, the browser determines which image to download and use.

Each source defines a potential image to display. These can then optionally have different attributes to define when to display that image. The 'media' attribute works much like a media query in CSS, whereas 'type' defines the MIME type of the file. If a browser doesn't meet the media query, or does not understand the file type, it moves on to the next <source> in the list.

The block ends on a regular <img> element. This will display if none of the other images can be displayed based on the conditions, or for browsers that do not understand the <picture> element at all.

The element itself acts more as a container for the elements inside. By itself it has no visual appearance and will be distorted when the fallback image is used. Be sure to style this



element for this use case.

```
<img srcset="paris-160.jpg 160w, paris-320.jpg
320w" sizes="80w" src="paris-80.jpg"/>
```

The <picture> element is best used in diagrams and informational images rather than simply supplying different resolution photos based on width. For that, use a regular img with a 'srcset' attribute. Using these hints, the browser can then decide which image to display.

For more information on 'srcset' and how to combine its power with the <picture> element, make sure you pay a visit to **https://www.html5rocks.com/en/tutorials/responsive/picture-element**

THE PICTURE ELEMENT IS BEST USED IN DIAGRAMS AND INFORMATIONAL IMAGES RATHER THAN SIMPLY SUPPLYING DIFFERENT RESOLUTION PHOTOS BASED ON WIDTH

MUST-TRY CSS ELEMENTS & APIS

CLIP-PATH

https://mzl.la/2Gxrzil

Define part of an element that should be displayed. Use it to create shapes without having to fake it with images.

CSS SHAPES

https://mzl.la/2Q1yFfh

Create more exciting, print-inspired layouts. Mark out a shape around an element for the content to flow around.

CUSTOM PROPERTIES

https://www.w3.org/TR/css-variables

Pass a single value to multiple selectors. They can read and update through JavaScript and can be useful for site-wide themes.

CSS FILTERS

https://bit.ly/2OOvSWx

Alter the appearance of elements without having to alter the file itself. For example, blur an image to make a background.

SCROLL SNAPPING

https://developers.google.com/web/updates/2018/07/css-scroll-snap

Create scrollable elements that have defined regions that should snap into view. Those larger than the viewport are handled automatically.

INLINE ELEMENTS

Avoid if you can and try elements designed for the job

<TIME>

https://mzl.la/2v46JSw

Dates and times are formatted differently across the world and so cannot be reliably parsed by a search engine or email client. Specify what parts of a sentence are a time and allow programs to extract and use that information.

<MARK>

https://mzl.la/2MI8Ej8

When wanting to highlight a few words of a sentence, it may seem best to use . While denotes importance, <mark> denotes relevance in the current situation. An example of this would be matched terms in a search result.

<ABBR>

https://mzl.la/2NR3aqF

Language is full of abbreviations that readers potentially may not be familiar with – there's plenty in this article! The <abbr> element provides a way to define potentially unfamiliar abbreviations with an accompanying 'title' attribute.

<Q>

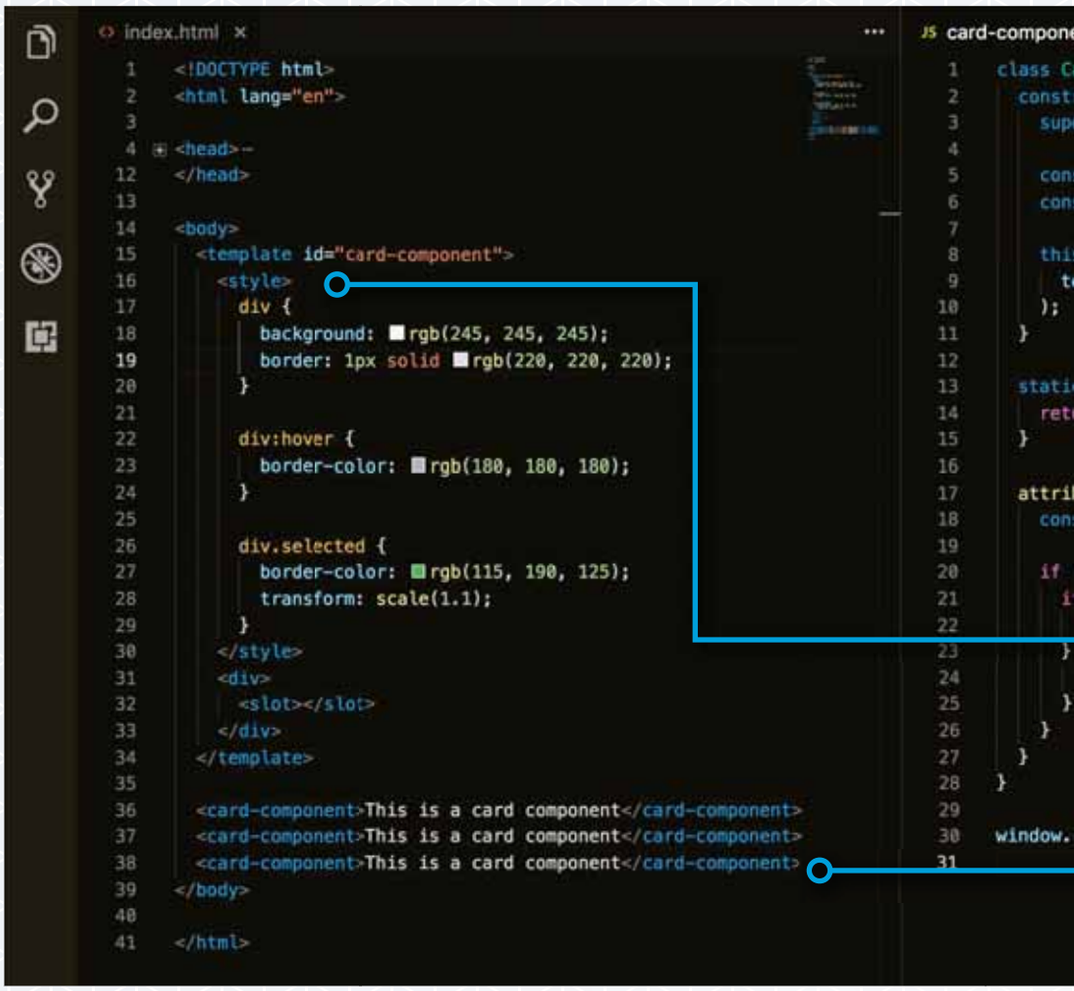
https://mzl.la/2xqiOze

When quoting a small section of text, surround it with <q>. The optional 'cite' attribute can provide a name, reference or link to the original source. Browsers will add quote marks by default.

<KBD>

https://mzl.la/2D9ojbB

When providing instructions that should be input by a user, <kbd> should surround that command. While typically used for keyboard inputs it can be used for any kind of text entry, including voice.



WEB COMPONENTS *https://mzl.la/2D5BOnZ*

Struggling to find an element that fits? Make your own with JavaScript

The introduction of HTML5 in 2014 brought with it lots of useful elements to browsers. Special components like progress bars used to mean misusing elements in order to achieve the right visuals. These features are now native to the browser and elements like <progress> can be added where needed without having to worry too much about the inner workings.

Thankfully, new web standards have been in development for some time to allow developers to make their own elements as part of the platform. Web components are a set of specifications that let developers create their own custom HTML tags and use them anywhere on a webpage. As long as they are registered using JavaScript, they are just as capable as any other element.

Building a web component uses three different specifications in the browser to construct, configure and generate their inner workings. Let's take a look at them.

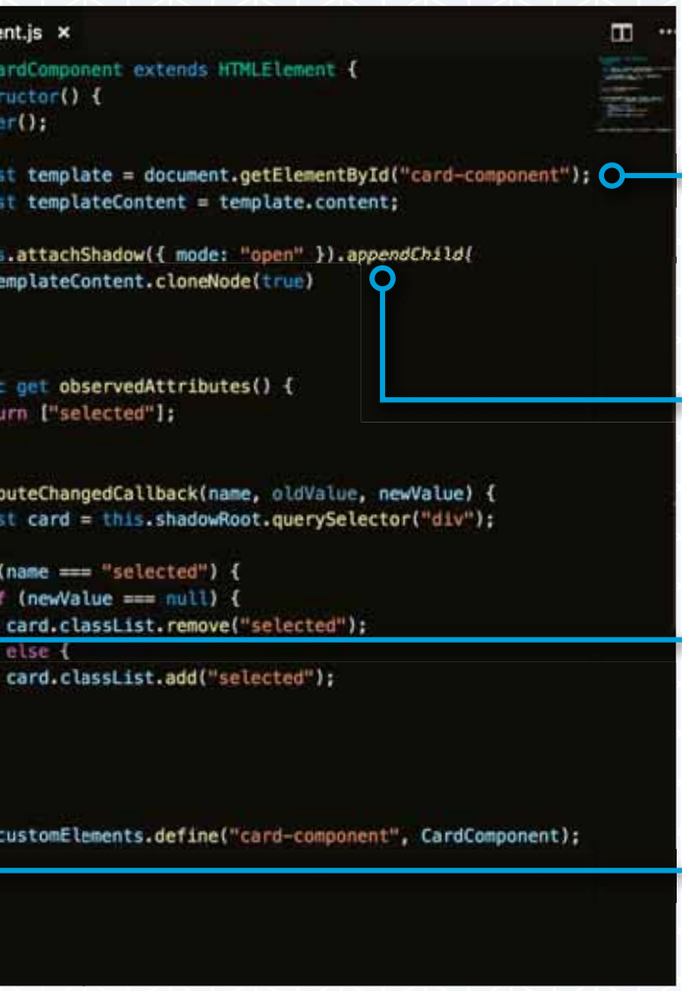
Custom elements

The most important feature of web components are the use of custom elements. These allow developers to use their own tag to render something specific to the browser wherever it appears on a page.

Each one is an ES2015 class that defines its behaviour, extending from an existing HTML element. These can contain any methods required for the operation of the component, but must use a constructor in order to set up any visuals or interactive elements such as event handlers.

In order to behave like a native element, it is important that elements react to external changes. Every custom element can tap into callbacks, such as 'connectedCallback' or 'attributeChangedCallback', to detect when an element needs to update.

The 'customElements' window property will allow an element to be registered with the browser. Until it is registered, a custom element



Selecting templates
Even inside a component class it is still possible to select elements within the page. Here the component is using the content from the 'card-component' template in the main page.

Open or closed?
Shadow DOM can be in one of two modes. Being open allows it to be accessed externally using a 'shadowRoot' property, while being closed does not.

Scoped styles
Styles are encapsulated within the shadow DOM, which means that any selectors written in this <style> element will not leak out to the rest of the page.

Slot content
By default, the content inside the component will be added to the first <slot> element within the template. In order to add content to specific slots, use the 'name' attribute.

will be treated like any other unknown tag so it is important to design its behaviour with progressive enhancement in mind.

Shadow DOM
The Document Object Model – or DOM – represents each page as a set of connected elements. The shadow DOM is a hidden subset of further connections within a specific element of that DOM. Nothing inside the shadow DOM can affect anything outside.

For example, a page may have a <video> element in its DOM, but the shadow DOM inside <video> houses the internal controls such as the play button and volume slider.

While this behaviour has been in browsers for a while, the shadow DOM API allows developers to create their own. When used together with custom elements, they allow a full range of visuals to be displayed without worrying about affecting other parts of the page.

HTML templates
Page structure elements are often repeated to make sure each one works the same as the last. To save time and reduce errors, developers can opt to make a function to generate HTML for an element, adjusting the contents as they go.

HTML templates bring that ability natively to browsers through use of the <template> element. The contents of a template stay inert and invisible, but JavaScript can access it like regular content without issue.

Extracting the contents of a template is as simple as selecting the template and getting its 'content' property. It can then be used wherever needed and acts just like any other HTML content.

All three specifications are designed to work together. A template can be used with the shadow DOM to produce the visuals for a custom element.

Cross-browser support is good
The best part about web components is that the support is almost there in every major browser.

At the time of writing the latest versions of Chrome, Safari and Opera all support all web component specifications.

As it stands, Firefox and Edge are both lacking full support for custom elements and shadow DOM, but polyfills are available to work in these browsers while these features are being developed.

Mobile browsers have a wider level of support, with all features being available in the latest versions of Chrome for Android, Samsung Internet and iOS Safari.

For updates on support and any new features, make sure to visit **https://www.webcomponents.org**

HTML TEMPLATES BRING THAT ABILITY NATIVELY TO BROWSERS THROUGH USE OF THE TEMPLATE ELEMENT. THE CONTENTS OF A TEMPLATE STAY INERT AND INVISIBLE, BUT JAVASCRIPT CAN ACCESS IT LIKE REGULAR CONTENT WITHOUT ISSUE

<INPUT>

The most common of elements can be more flexible than you think

Even the simplest of forms need some kind of validation. A contact form, for example, needs to check if a name and email have been filled out. Users now expect instant feedback on what has been entered.

This used to be the role of JavaScript. If a website needed a date entry JavaScript would need to check a text value looked like a date. This required the developer knowing all the different formats that could be valid and implementing that check correctly.

All of that changed with HTML5. The new specification introduced a host of new input types that could be added as needed without the need for scripted validation.

When given a certain type, a browser renders an appropriate interface geared towards that input. On touchscreen devices, this also meant a contextual keyboard that merited itself to the input in question.

If used within a form, the validation occurs on submission. Browsers will show an inline error message for each field that is wrong, so users know what is happening. An invalid form will not submit.

Colour inputs

```
<input type="color" />
```

In forms where colour input is required, the 'color' type provides either a colour picker or a hexadecimal text input to provide a value.

The display of the colour picker can vary depending on the browser, which may only supply a limited subset of colours by default. A site using this input should provide an alternative method of input in this case.

A default colour can be applied by supplying a 'value' to the field, but this must be a valid hex colour and not a CSS colour name or function like 'linear-gradient'. If an invalid colour is added, it will default to black.

Number inputs

```
<input type="number" min="0" max="99" />
```

When a form requires a number, it makes sense to use an input specialised for that purpose. A number field get automatic validation to be sure the value is numerical.

Most modern browsers will also render stepper buttons, which allow the user to increment or decrement the value inline. On a touchscreen device, the keyboard will update to show numbers by default and may even allow users to swipe to increase and decrease the value.

Extra attributes on the input can limit the value being entered. For example, an age field may have a 'min' attribute to stop a negative value being added.

```
<input type="range" min="0" max="10" />
```

When a specific value is not necessary, a range input may be more useful. In this case, browsers can render a more appropriate slider to allow the user to select a value more easily. These are useful for simple components such as volume sliders.

Modes on any input

While these type inputs are great for their typical use case, it can become a hindrance for others. For example, a credit card number is numeric but does not need the increment and decrement behaviour that the 'number' type provides.

```
<input inputmode="numeric" pattern="[0-9]*" />
```

Most forms would fall back to regular text input in this case and use a 'pattern' attribute to define what is required. This requires the developer to know the right pattern for their input, which can cause a similar issue to before.

An 'inputmode' attribute is coming to browsers that aims to tackle this issue. It hints to the browser what kind of interface would help the user add data to this field. Having a value of none will ensure no virtual keyboard is displayed, allowing for a custom keyboard within the page itself.

Always server validate

As great as these input types are, they are not without flaws. Browsers may not support a specific type or can have unexpected valid inputs, such as 'e' for number inputs. At the very least, a user can still edit the HTML on a page and remove validation altogether.

It is important that any data submitted to a server is also validated on that side to avoid any issues. Include client-side validation to help.

When using the correct input type, browsers will validate the input without the need for any added checks with JavaScript

ESSENTIAL JS APIS & ELEMENTS

INTERSECTION OBSERVER

https://bit.ly/2NO3ejx

Fire a callback whenever an element enters and exits a certain area. Best for lazy loading or infinite scroll effects.

REPORTING OBSERVER

https://bit.ly/2OTO17n

Get notified when the browser has to intervene, for example having to provide a fallback for a slow-loading font.

OFFSCREENCANVAS

https://developers.google.com/web/updates/2018/08/offscreen-canvas

Drawing to a <canvas> is an expensive task for a browser. This new API can render without hitting the DOM.

INTERNATIONALISATION API

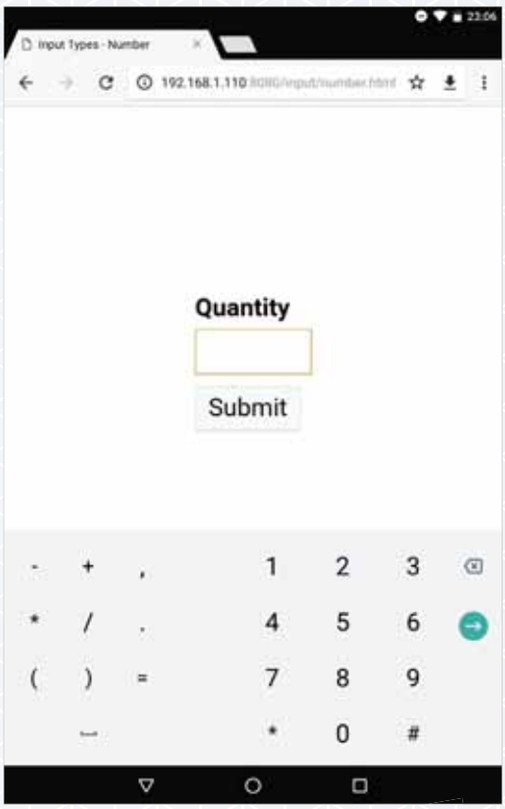
https://mzl.la/1dsIHLk

Avoid sending heavy libraries and show dates, numbers and strings in a format that makes sense in the user's location.

CSS PAINT API

https://bit.ly/2IcGL1P

Use JavaScript to generate dynamic backgrounds, borders and image masks and apply them in CSS without a heavy performance hit.



Depending on the device, virtual keyboards can show vastly different input methods. For example, the number keyboard on iOS still allows text input

Browsers will update the input methods based on screen space available. For mobile devices, colour inputs are often chosen through sliders

BLOCK-LEVEL ELEMENTS

For larger elements always use the right element for the job

<MENU>

https://mzl.la/2xln6st

Use <menu> to define a set of controls within an interface, with each child being an <li> element. It behaves similar to <ul>, with the distinct difference between them being that <menu> contents are expected to be interactive rather than a simple list of items.

This component was previously defined in HTML5 as a way to provide context menu items alongside <menuitem>. However as of HTML5.2 this behaviour has been removed and the element repurposed. While it is now not provided by the browser directly, it is still encouraged to use <menu> for context and drop-down menus created as part of an application.

<PRE>

https://mzl.la/2g7x8Vo

HTML is a fault-tolerant language, which means it can detect mistakes and formatting quirks within a page easily and infer intended meaning. This includes cleaning up spacing, which is sometimes intended for content such as code samples.

The <pre> element will render content exactly as it appears within the HTML, typically in a monospace font to keep alignment intact.

As this could potentially be unlimited in width, make sure to set an overflow value to make sure the rest of the layout stays intact. Also, be sure to provide an accessible description as screen readers will struggle to read its contents.

Create glitchy, blinking text effects

Inspired by **betamatters.com**

Blinking text
The most noticeable element on the home page is the flashing, blinking, text letters that flash in and out and each one turns on like a fluorescent tube light.

Opening animation
All the green elements on the screen animate onto the screen one at a time to build up the highly visual illustrated scene.

Ticker side bar
The green bar on the left is reminiscent of TV news tickers with the content continuously scrolling up.

Contextual content
Letting your visitors know exactly what site they have landed on and its purpose is always a good way to orientate the uninitiated.

Navigation menu
Clicking the navigation menu gives one of the best animated transitions that the *Web Designer* team have ever seen, you must see this for yourself!

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

EXPERT ADVICE
Making the design work
The theme for this site works so well because the contrast between technology is offset by having organic characters that float around the screen. The organic shapes work well with the geometry of the tech industry, giving a friendly and approachable feel to the design. This lets the user know that they are the important part of the working relationship.

<comment> What our experts think of the site

Bringing the design together

The betamatters website stands out so well because of it's bold use of colour and imagery. The designers have thought through the kind of look that they want to achieve and brought together illustration with design and code to make the aesthetic effect stand out and have maximum impact for the user.
Mark Shufflebottom, Professor of interaction Design

Technique

1. Blinking letters

To make the blinking letter effect found on betamatters, add some images to the page that will act as the text to make blink on and off. The structure shown below is a minimal way to achieve this effect:

```
<div id="wrap">
<img src="images/b.png" class="letter b">
<img src="images/e.png" class="letter e">
<img src="images/t.png" class="letter t">
<img src="images/a.png" class="letter a">
</div>
```

2. Style the content

Now the content of the page will be styled with the background getting the right colour and the letter being given the right size. The opacity is turned off so that with JavaScript the 'anim' class can be added.

```
body{
    background: #5700c8;
}
.letter{
    width: 150px;
    opacity: 0;
}
.anim{
    animation: glitch 1s forwards;
}
```

3. Blinking with keyframes

Using keyframes, the 'anim' class in the previous step uses these to change the opacity and make the letters blink on and off as they appear onscreen.

```
@keyframes glitch {
    0% {opacity: 0;}
    40% {opacity: 1;}
    45% {opacity: 0;}
    50%  {opacity: 1;}
    55% {opacity: 0;}
    85% {opacity: 1;}
    90% {opacity: 0;}
    100% {opacity: 1;}
}
```

4. Making it work

To make this effect work, first of all, the letters – which are child elements of the 'wrap' div – are placed into an array called 'letters'. The selected letter will be stored in the 'sel' variable.

```
var elem = document.getElementById('wrap');
var letters = [];
var sel;
for(i=0; i<elem.children.length; i++){
    letters.push(elem.children[i]);
}
```

5. Generate a letter

The 'generate' function will randomly select one of the letters in the array and run a 'setTimout' to call the 'animate' function at a random time so that the letters appear more glitchy.

```
function generate(){
    sel = Math.floor(Math.random()*letters.
length);
    setTimeout(animate, (Math.
random()*500)+100);
}
```

6. Animating the letter

The 'animate' function adds the 'anim' class to the letter and then removes this letter from the array so that it can't be called again. The 'generate' function is called to run which starts the whole process working.

```
function animate(){
    letters[sel].classList.add("anim");
    letters.splice(sel, 1);
    if( letters.length > 0 ){
        generate();
    }
}
generate();
```

Get started with Three.js – Part 5

In this fifth tutorial, you'll learn how to load complex 3D models in WebGL using Three.js

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

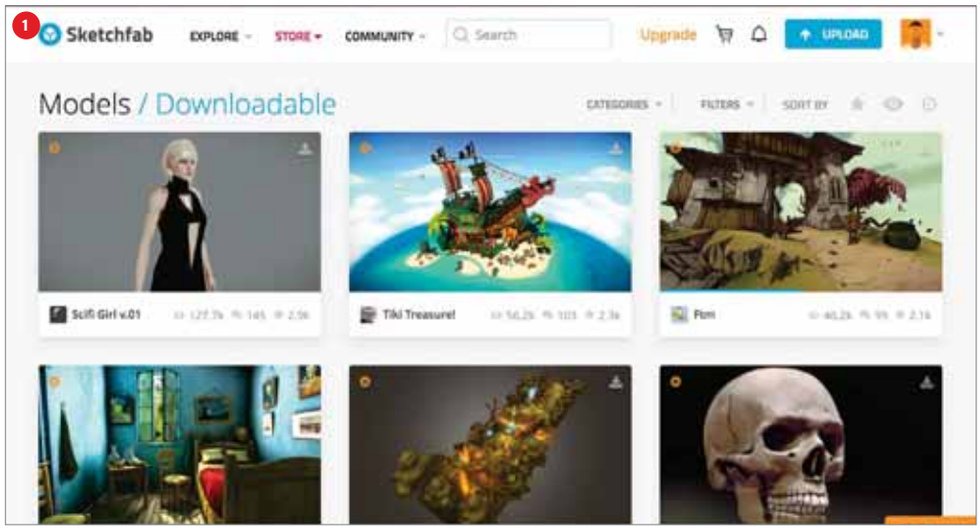
WebGL enables developers to create rich, console-quality experiences, that render in real time on mobile devices and desktop browsers.

Nearly universal browser and device support makes it a perfect approach for web developers wanting to create incredible experiences. No plugins are required and you can start learning these technologies right away.

One of the questions that comes up the most often about 3D development is about importing models. It's often confusing trying to choose the right format and loading it correctly. You can accomplish a lot with primitives, but very often you want to integrate models created in 3D modeling software.

You'll be using the popular 3D library Three.js in this series. It's free, open source and lightweight, and countless award-winning websites have used it.

Building on previous tutorials, you'll move onto learning about loading complex models in your 3D scenes: What formats are supported, which are the best to use, and why. Plus, where to get models and how to get them into your scenes to start using them in 3D. 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 get you started in 3D web programming and to get you inspired.



1. Choose your models

To follow along, you'll need an OBJ format model and it's corresponding texture(s). You'll also need a glTF model as well. Here are two great sources for downloading models:

- turbosquid.com/
- sketchfab.com/models?features=downloadable&sort_by=-likeCount&type=models

We've used the Pony Cartoon, by Slava Z, which you can buy or try out here: **sketchfab.com/models/885d9f60b3a9429bb4077cfac5653cf9**

2. Create a basic HTML file

Next, you need to set up a basic HTML file. You can setup external CSS and JavaScript files or include inline for simplicity. Three.js's 'renderer' class will create a <canvas> element for you. Add the following code to your 'index.html' file.

```
<!DOCTYPE html>
<html>
<head>
        <style>
        html, body { margin: 0;
        padding:0; overflow: hidden; }
        </style>
</head>
<body>
        <script>
                // Code will go here
        </script>
</body>
</html>
```

3. Include Three.js classes

Include a link to the Three.js library in the <head> of your file, either hosted externally or download it from the Three.js repository. You will also need the 'OrbitControls', 'OBJLoader' and 'GLTFLoader' classes for this tutorial. You can find both the library and the supporting classes at **github.com/mrdoob/Three.js/**. Note: The code in this tutorial has been tested on the latest release of Three.js v95.

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

4. Add global variables

Between your <script> tags for your code, add the following global variables to globally access the camera, scene, renders, loaded object and the Orbit controls. Instead of writing your own interactivity, you'll use the built in Orbit controls in this tutorial:

```
// global vars
var camera, scene, renderer, object, controls;
```

5. Create a 3D scene

You're going to add a basic 3D 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:

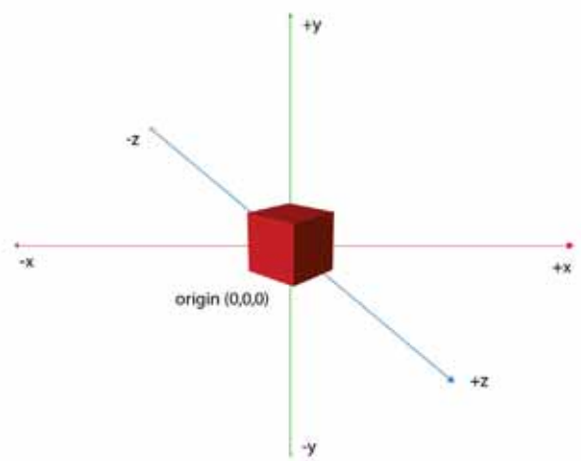
```
// create a scene object
var scene = new THREE.Scene();
```

6. Add a perspective camera

Next, you need to add a camera. You'll use the perspective camera, meant for 3D scenes. 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 far clipping plane distances, which define what is to be visible to the camera. You'll also push the camera back in Z space a little to make things things easier to see.

```
// create camera
camera = new THREE.PerspectiveCamera( 75,
window.innerWidth / window.innerHeight, 1,
```

6



3D Model formats

There are hundreds of file formats available for 3D models. They range widely in complexity, features and purpose. The Three.js library has many loaders available to handle these formats, including OBJ, FBX and glTF.


```
2000 );
camera.position.z = 15;
scene.add( camera );
```

7. Add a renderer and canvas

The 'renderer' handles the drawing of the objects in your scene that are visible to the camera. Set the 'antialias' property to 'true' to get smooth edges on our object. The renderer creates a 'domElement', which is actually an HTML <canvas> element, you can then append to the body. Try setting your renderer's 'gammaOutput' to 'true' for optimal support of glTF models.

```
// create renderer
renderer = new THREE.WebGLRenderer({
antialias:true});
renderer.setPixelRatio( window.
devicePixelRatio );
renderer.setSize( window.innerWidth, window.
innerHeight );
renderer.gammaOutput = true;
document.body.appendChild( renderer.domElement
);
```

Testing 3D models

To save yourself stress and lost hours, it's a good idea to test out your models in one of the following online viewers. You can verify the model data is good, and tweak some settings before importing:

- **glTF Viewer**: gltf-viewer.donmccurdy.com/
- **Three.js Editor**: threejs.org/editor/

8. Add orbit controls

To keep things simple, you can use Three.js's built-in 'orbit controls' class. This enables you to drag the camera around in an 'orbit' around the target. You attach the controls to the camera and then set a target for it to orbit.

```
// add orbit controls
controls = new THREE.OrbitControls( camera );
controls.target.set( 0, 0, 0 );
controls.update();
```

9. Add lights to the scene

To light your loaded models, a balance of ambient and point or directional lights can work well. OBJ models do not have lights, but glTF scenes can include them. If your chosen model includes lights then you can either omit them later on, or use them as additional lighting.

```
// add ambient light
var ambientLight = new THREE.AmbientLight(
0xffffff, .2 );
scene.add( ambientLight );
// add point light
var pointLight = new THREE.PointLight(
0xffcc66, 0.6 );
camera.add( pointLight );
```

10. Load environment map

Models often benefit from environment maps. These maps are 'skyboxes' that surround the object so it can affect it from all directions accurately, impacting the colour and intensity of the colour on the surface texture. A great Cube Maps resource can be found on the Humus site (**humus.name/index.php?page=Textures**). Add the following code to load your map:

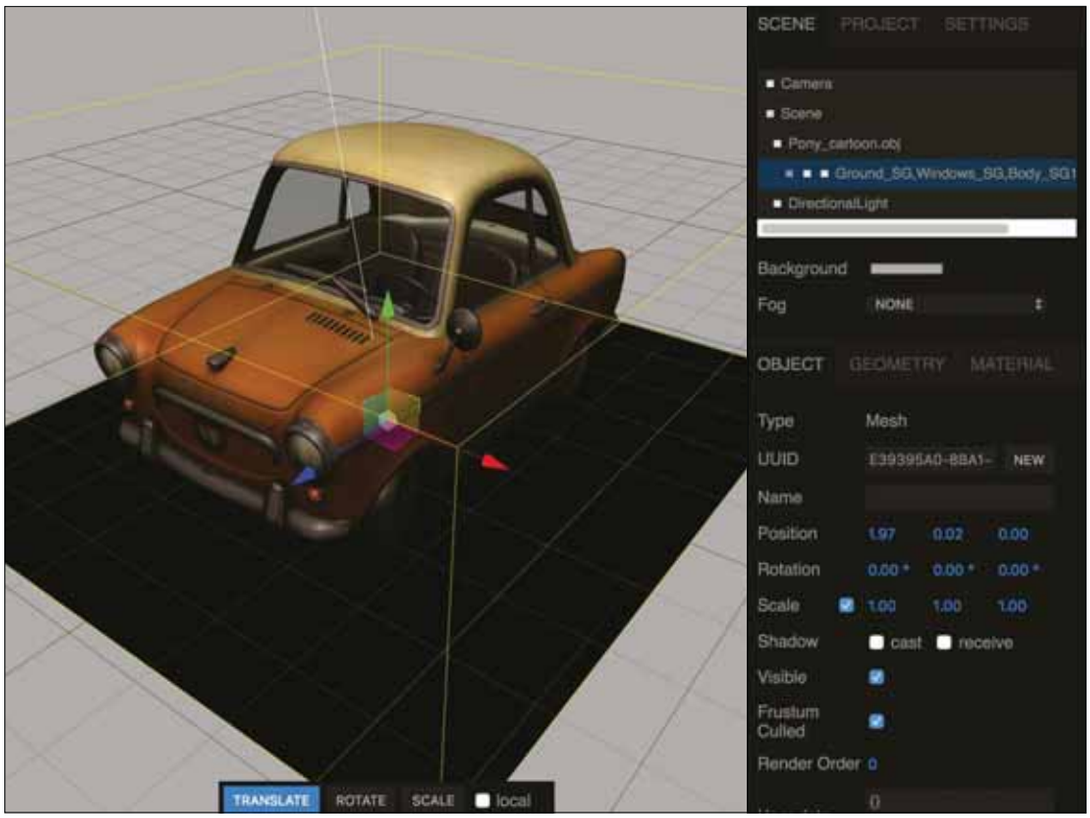
```
// create environment map
var envMap = new THREE.CubeTextureLoader()
.setPath( 'assets/')
.load( [ 'posx.jpg', 'negx.jpg', 'posy.jpg',
'negy.jpg', 'posz.jpg', 'negz.jpg' ] );
// set as skybox
scene.background = envMap;
```

Note: the order of the cube map images is important, so be sure to follow the above pattern when setting yours.

11. Create a loading manager and handlers

You will be loading the OBJ format model first. It is a much simpler model, but needs more steps than a glTF model. Start by adding a 'LoadingManager' class and handlers for progress and completion. Use these to time the next step and when to show your model. For now we're going to keep things simple.

```
// loading manager
var manager = new THREE.LoadingManager();
manager.onProgress = function ( item, loaded,
total ) {
        console.log( item, loaded, total );
};
// load complete
manager.onLoad = function ( ) {
        console.log( "finished loading models
& textures" );
};
```



glTF (GL Transmission Format)

For years, there has been a large variety of 3D model formats with varying levels of performance and support. Now glTF has emerged as a viable standard format to use. There are exporters or convertors available for many of the major 3D applications. glTF reduces the size of 3D assets, and the real-time processing needed to use those assets. Both GLB and GLTF versions of the format are well supported by the Three.js library. glTF is compact to transmit and fast to load. Features include meshes, materials, textures, skins, skeletons, morph targets, animations, lights, and cameras. You'll also see that loading glTF scene data is extremely easy inside Three.js making it a great choice whenever possible You can read more about the format here: **khronos.org/gltf/**

12. Set up textures array

For some models, such as OBJ, you may want to manually assign the textures to the loaded model. Set up an array of textures for as many as you have for your model. Not all will have these, but you'll want at least one primary texture to use as the default 'map'.

```
// textures
textures = [];
textureAssets = [
        {“file”:”assets/sword/German_Bastard_
Sword_DIFFUSE.jpg”,”name”:”diffuse”},
        {“file”:”assets/sword/German_Bastard_
Sword_SPECULAR.jpg”,”name”:”specular”},
        {“file”:”assets/sword/German_Bastard_
Sword_NORMALMAP.jpg”,”name”:”normal”}
];
```

13. Load texture files

You can use the 'ImageLoader' class to load in the image files and then assign them to your textures array. This gives you an easy way to access all your loaded images, so you can assign them to your loaded OBJ model. It also connects to the loading manager you set up. Add the following code next:

```
// assign each loaded texture to a texture
object
textureAssets.forEach(function(t){
        textures[t.name] = new THREE.
Texture();
        var loader = new THREE.ImageLoader(
manager );
        loader.load( t.file, function ( image
) {
                textures[t.name].image =
image;
                 textures[t.name].
needsUpdate = true;
        }); });
```

14. Load OBJ Model

Next, load in the OBJ file you want to use. You can use the 'OBJLoader' class to do this and connect it to your loading manager, like you did for the 'ImageLoader'. Once it's loaded you will be able to work with the resultant 3D object that is passed to the function. Add the following code:

```
// load OBJ Model
var loader = new THREE.OBJLoader( manager );
loader.load( ‘assets/sword/German_Bastard_
Sword.obj’, function ( obj ) {
        // handle loaded file here
        // adjust position and scale });
```

15. Handle the loaded OBJ

Inside your 'handler' function, you can assign the textures to the loaded OBJ. Your loaded object may be comprised of multiple meshes, depending on the model. You need to 'traverse' the object and find the mesh you wish to assign the textures to. You can check if a child is an instance of a mesh. You could also check against its 'name' property.

```
// handle loaded file here
console.log(object);
object=obj;
object.traverse( function ( child ) {
        if ( child instanceof THREE.Mesh  ) {
                // assign textures here
        } });
```

16. Assign material to target mesh

Once you have the mesh located, you can assign its 'material' property as usual, using the textures in your textures array. If your OBJ also has an associated MTL file you could use the Three.js MTLLoader and assign textures using that. It sometimes needs adjustments to work nicely, so we'll use this more straightforward method here. Add the following code next:

```
// assign textures here
child.material = new THREE.
MeshPhysicalMaterial({ map:textures[“diffuse”],
specularMap: textures[“specular”], normalMap:t
extures[“normal”],envMap: envMap});
```

17. Set object scale and position

Once you have your OBJ loaded and the materials assigned, you are ready to add it to the 3D scene. You'll most likely need to scale your model and position it where you can see it. This is going to vary depending on the model you use. You'll need to update 'scale' and 'position', and 'rotation'. Add the following code after your 'traverse' function:

```
// adjust position and scale
object.position.set(5,0,0);
object.rotation.z=Math.PI/180*70;
object.scale.set(.25,.25,.25);
scene.add( object );
```

18. Animation render loop and object rotation

Next, you need to render your scene. You learned about the renderer in previous tutorials. It renders a frame inside the animation loop running at a target of 60 frames per second. Add the following code and run it to see your loaded model:

```
var animate = function () {
        requestAnimationFrame( animate );
        if(object){ object.rotation.x+=.01; }
        renderer.render(scene, camera); }
animate();
```

19. Remove textures and texture loading

Now that you know how to load OBJ and assign textures, you can use these complex models in your scenes. However, there is a much better way to load models when you have the format available – and that is to use the glTF models. To do this, remove the code (approximately 25 lines) from 'loading manager' all the way down to the texture loading lines, including the texture assets for each section.

```
// loading manager
…
// assign each loaded texture to a texture
object
….
```

20. Replace the OBJ loader with GLTF loader

Next, replace the OBJ loader you were using with the glTF loader using the following code. Notice that the format is the same and the steps you follow are similar, but it's just much simpler; the glTF loader handles all the texture assignments for you, and much more:

```
// load glTF model/scene
var loader = new THREE.GLTFLoader();
loader.load( 'assets/pony/scene.gltf',
function ( gltf ) {
        // handle loaded file here
        // adjust position and scale });
```

21. Handle glTF model loaded

Now that your glTF scene is loaded, you can use the 'traverse' function you used before. This time it's used to assign the 'envMap' you created. You could also manually adjust textures further using this method as well. Run this new code and you'll see your loaded glTF model! This is a great first step in getting complex models into your 3D scenes!

```
// handle loaded file here
gltf.scene.traverse( function ( child ) {
        if ( child.isMesh ) {
                child.material.envMap =
envMap;
}
});
// adjust position and scale
gltf.scene.scale.set(.005,.005,.005);
gltf.scene.position.y = -1;
scene.add( gltf.scene );
```

Code smart WebGL transition effects

Transition between videos in a slideshow using Curtain.js to power the transition with WebGL and Shaders

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

Some of the most advanced effects on the web are found on sites which win various web awards for their respective 'site of the day'. These are always great inspiration pieces to look at and get ideas from, and a recent trend has been to use something known as a shader in WebGL to create very advanced graphical effects. These can be quite difficult to use and require advanced programming skills to master.

Curtain.js allows you to easily create transitions from one object to another using WebGL and shaders. In this tutorial, Curtain.js will be used to transition from one piece of video to another using a prebuilt shader that will use a displacement image to power the transition. What that means is instead of just fading from one to another, the different shades of the displacement image will affect the way it transitions and so some really advanced effects can be created. Because the shader is prebuilt the tutorial will focus on integrating the effect into a regular HTML page, adding the tags, the CSS and the JavaScript to power the way Curtain.js works. Shaders do offer the ability to create advanced interactions for your users and as Curtain.js takes some of the hassle out of this, it can help elevate your site above others.

1. Adding the textures

To get started, drag the start folder onto your code editor and open 'index.html'. There is already some code in here for the design of the page. At the top of the body before all other content, add the code shown. It will add the 'canvas' element and the container of textures to make the video transition.

```
<div id="vid-wrap">
                <div id="canvas"></div>
                <div class="vid-textures-
wrapper">
                        <div class="vid-
textures" data-vs-id="vid-textures-vs"
data-fs-id="vid-textures-fs">
```

2. Adding the video

Now the image for the displacement is added onto the screen with two videos that will be used as textures in the WebGL. The shader will be able to move between these videos using the displacement as a mask to create the transition.

```
                    <img src="images/
displacement.jpg" data-sampler="displacement"
/>
                <video src="videos/v_tex1.
mp4" data-sampler="firstTexture"></video>
                 <video src="videos/v_tex2.
mp4" data-sampler="secondTexture"></video>
        </div>
      </div>
</div>
```

3. Link up the library

Now move to the end of the body tag and add the following script tags. The first links up the curtains library to the page so that the code from this can power the interactions. The second is the JavaScript file that will be used to add the interactions in this tutorial.

```
<script src="js/curtains.min.js" type="text/
javascript"></script>
<script src="js/setup.js" type="text/
javascript"></script>
```

4. Adding the CSS

After adding the HTML in the first two steps, it's now time to style up that content. Switch over to the 'style.css' file and here the body is defined to remove the margins, increase the line height and to stop content overflowing on the x-axis. The 'vid-wrap' will hold the video that will transition so it's made to fit 90% of the browser height.

```
body {
    margin: 0;
    background: #ffffff;
    line-height: 1.6;
    overflow-x: hidden;
}
#vid-wrap {
    width: 100vw;
    height: 90vh;
    overflow: hidden;
    background: url(images/bg.jpg) no-repeat
center center;
    background-size: cover;
}
```

5. Painting on canvas

The canvas element is where the WebGL content is rendered, so inside here the width is set to the full width of the screen and the height to 90% of the viewport height. The opacity for this is turned off and it will transition in once the video content has loaded.

```
#canvas {
    height: 90vh;
    width: 100vw;
    z-index: 10;
    opacity: 0;
    transition: opacity 0.5s ease-in;
}
```

6. Triggering changes

When the page has loaded and the video has started playing, the JavaScript file will add the 'video-started' class to the page. This then fades the video in by making it fully opaque. The video textures wrapper is set to fill the screen and is positioned behind other content.

```
.video-started #canvas {
    opacity: 1;
}
.vid-textures-wrapper {
    position: absolute;
    left: 50%;
    top: 50%;
    transform: translate(-50%, -50%);
    z-index: 15;
}
```

7. The video textures

The video textures are stretched to fill the screen and then temporarily made invisible to keep them out of public view for the time being. These will be used by the code to be placed in the webGL content as textures on planes.

```
.vid-textures {
    position: absolute;
    top: 0;
    right: 0;
```

Curtain, what?

Curtain.js solves the problem of positioning WebGL planes relative to your page as you position your elements with CSS, and then the code converts into WebGL and shader content.


```
    bottom: 0;
    left: 0;
    cursor: pointer;
    font-size: 3em;
    color: white;
    opacity: 0;
    transition: opacity 0.5s ease-in;
}
```

8. Inner content

Inside the video textures the video and image are turned completely off so that they cannot be seen at all. They are not needed for the tutorial, but they will be used inside the canvas element as their content is passed in via code to be used in the scene.

```
.video-started .vid-textures {
    opacity: 1;
}
.vid-textures img, .vid-textures video {
    display: none;
}
```

9. A message for mobile

If the website is being displayed on mobile then a 'click to play' message will be added because mobile can't auto-play video. The 'enter-site-wrapper' class holds that message. It will only be displayed if a mobile device is detected in the JavaScript later.

```
#enter-site-wrapper {
    z-index: 230;
    opacity: 0;
    transition: opacity 0.5s ease-in;
}
.curtains-ready #enter-site-wrapper {
    opacity: 1;
}
```

10. Turning off the message

Here the message for 'clicking to play' is turned off when the video has started; if the video has started there is no need to click to play. The 'enter-site' is the message so it has a dark grey background to make the content legible on the screen, over the placeholder image.

```
.curtains-ready.video-started #enter-site-
wrapper {
    opacity: 0;
    pointer-events: none;
}
#enter-site {
    padding: 20px;
    color: white;
    background: rgba(0, 0, 0, 0.5);
    max-width: 200px;
    text-align: center;
    cursor: pointer;
}
```

11. Open the JavaScript

Save and close the 'styles.css' now as it is completed. Open the 'setup.js' from the js folder and it should be empty. The code here gets access to the canvas element after the page has finished loading.
The active texture is used to switch between textures, and the timer will measure how long has passed in the transition.

```
window.onload = function() {
    var canvasContainer = document.
getElementById("canvas");
    var activeTexture = 1;
    var transitionTimer = 0;
```

12. Variable essentials

These variables hold the reference to instantiate a new Curtains object, then the textures are placed into a variable named 'planeElements'. As there is more than one, it becomes an array of the three textures. The element that will change the video is referenced and the pixel ratio stored.

```
var webGLCurtain = new Curtains("canvas");
var planeElements = document.
getElementsByClassName("vid-textures");
var clicker = document.
getElementById("featured");
var pixelRatio = window.devicePixelRatio ?
window.devicePixelRatio : 1.0;
```

13. Parameters for the shader

The shader needs some parameters adding in that allow the shader to run. The resolution of the curtains element is updated so that the dimensions are passed in, which is width and height multiplied by the pixel ratio of the screen from the previous step.

```
var params = {
        uniforms: {
            resolution: {
                name: "uResolution",
                type: "2f",
                value: [pixelRatio *
planeElements[0].clientWidth, pixelRatio *
planeElements[0].clientHeight],
            },
```

14. Timing is everything

The timing of the transition needs to be passed into the shader. These values will be updated later on so that it can time the change from one of the video textures to the other. This completes the information that is required to be passed into the shader.

```
transitionTimer: {
                name: "uTransitionTimer",
                type: "1f",
                value: 0,
            },
        }, }
```

15. Checking the device

A plane is added into the WebGL scene with the first texture, which just happens to be the displacement map. The function 'isMobile' is defined here and it simply checks if the browser can support orientation changes. If it's 'IEMobile' then this is being displayed on a mobile device.

```
var multiTexturesPlane = webGLCurtain.
addPlane(planeElements[0], params);
function isMobileDevice() {
        return (typeof window.orientation !==
"undefined") || (navigator.userAgent.
indexOf('IEMobile') !== -1);
};
```

16. Changing video

When the multi-textures plane is ready this adds a class to the body that causes transitions to occur in fading

What is a shader?

A shader is a program that is written in a C-like language. It is a very specific program that controls the graphics card to do advanced graphic features.

Setting displacement images

You can completely change the effect of the transition by simply changing the image that is used as the displacement image. You need to make sure the image is in greyscale values but then just save it as a regular RGB JPEG image. In the code change the line:

```
<img src="images/displ2.jpg"
data-sampler="displacement" />
```

Here, another image has been used, which is a rippled swimming pool image. Instead of the geometric lines in the transition there are wavy ripples that transform the video images together. It's best to use a greyscale image because it uses the different shades of grey to transform the amount of movement. The lighter areas transform faster than the darker areas, so keep that in mind when selecting your own image to use.

the video in. The clicker object which will change the videos is given an event listener and this will switch out the textures in the scene causing the video transition.

```
multiTexturesPlane.onReady(function() {
        document.body.classList.add("curtains-
ready");
        clicker.addEventListener("click",
function() {
            if(activeTexture == 1) {
                activeTexture = 2;
            }
            else {
                activeTexture = 1;
            }
        });
```

17. Updating the screen size

If the browser window is changed in its size, which can happen by changing orientation on mobile, the screen size is updated for rendering. The variable 'mobile' calls the function to check if this is a mobile device or not, and stores the result as true or false.

```
window.onresize = function() {
            multiTexturesPlane.uniforms.
resolution.value = [pixelRatio *
planeElements[0].clientWidth, pixelRatio *
planeElements[0].clientHeight];
 }
mobile = isMobileDevice();
```

18. What to do?

Once we know if this is a mobile device or not, some decisions can be made. If it is, a button is placed on the screen that will allow the user to start the video as it needs a click to play on mobile. Otherwise if it's desktop, the video just starts to play as it is.

```
if(mobile){
            document.getElementById("enter-
site").addEventListener("click", function() {
                start();
            }, false);
}else{
            start(); }
```

19. Start playing

The start function from the previous code is defined here and this adds a class to the body that tells the screen that the video has started playing. The videos themselves are called to play so that the effect can work, and the webGL content will update with the video image.

```
function start(){
            document.body.classList.
add("video-started");
            multiTexturesPlane.playVideos();
}
```

20. Changing the textures

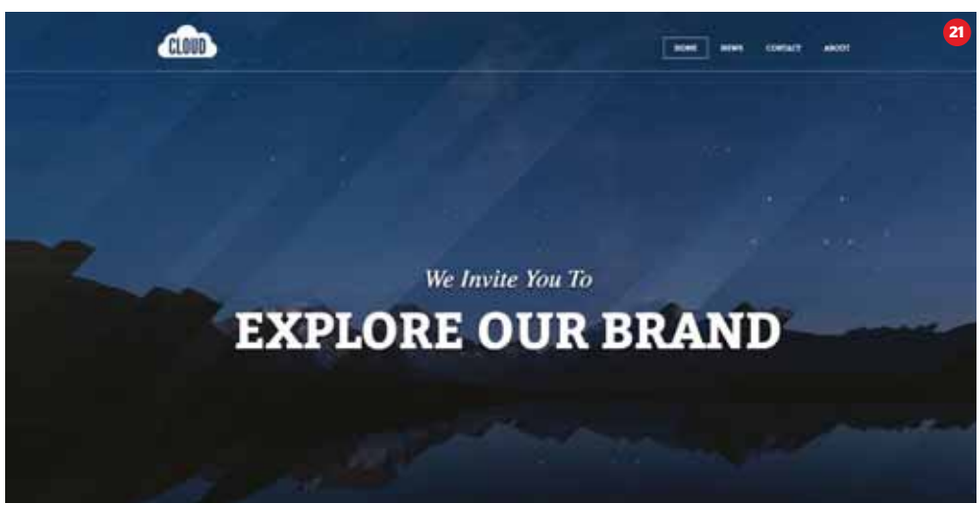
When the activeTexture variable is changed this function deals with that by creating the transition from one screen to the other and timing the transition over. In the final step these values are passed into the shader and that updates the display on the screen.

```
}).onRender(function() {
        if(activeTexture == 2) {
            transitionTimer = Math.min(90,
transitionTimer + 1);
}
```

21. Finishing up

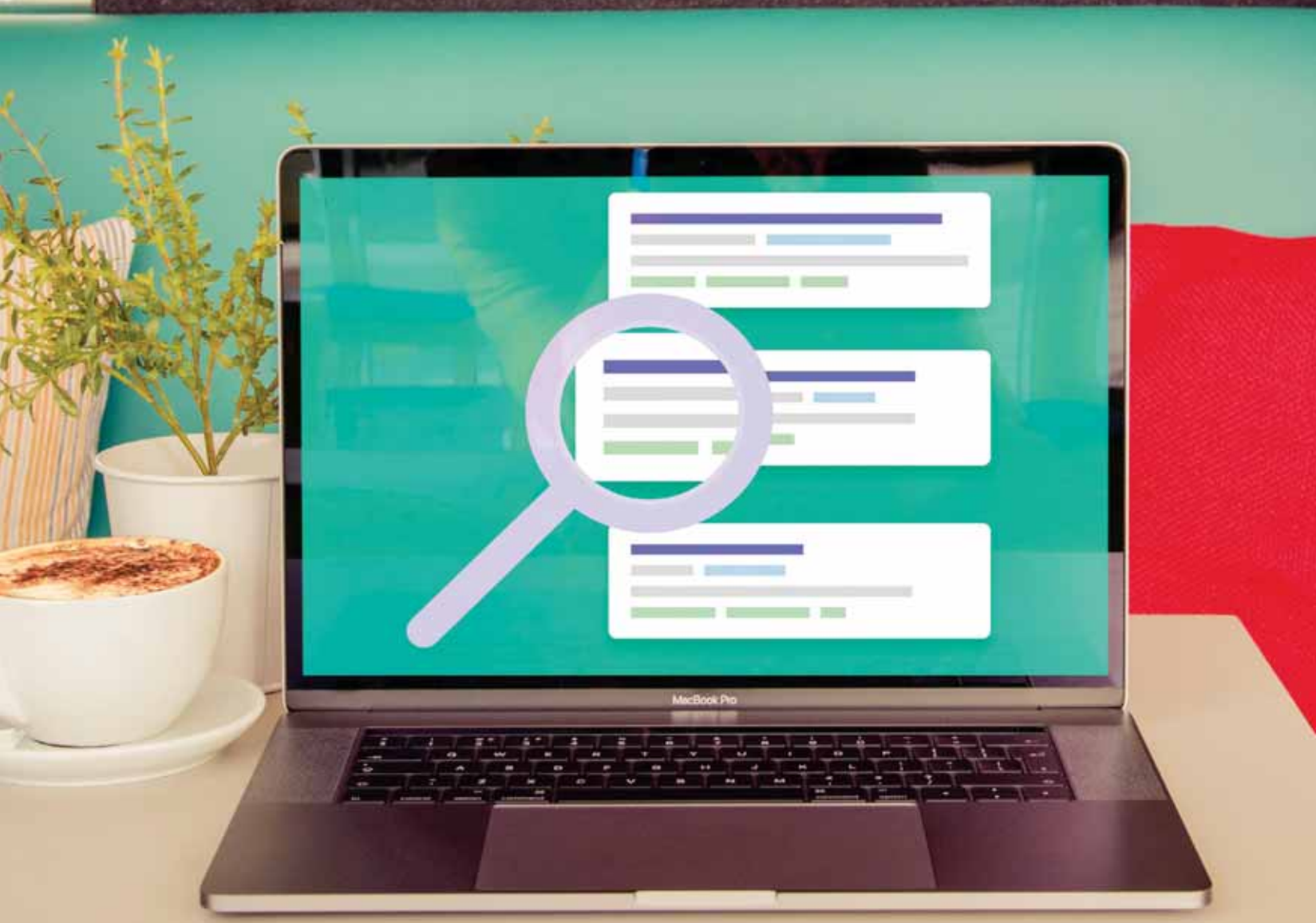
The final step just monitors the transition back to the other video. Save this file now and test in the browser. The video will start playing automatically on desktop but requires a click on mobile devices. The video transition takes place every time the user clicks on the screen.

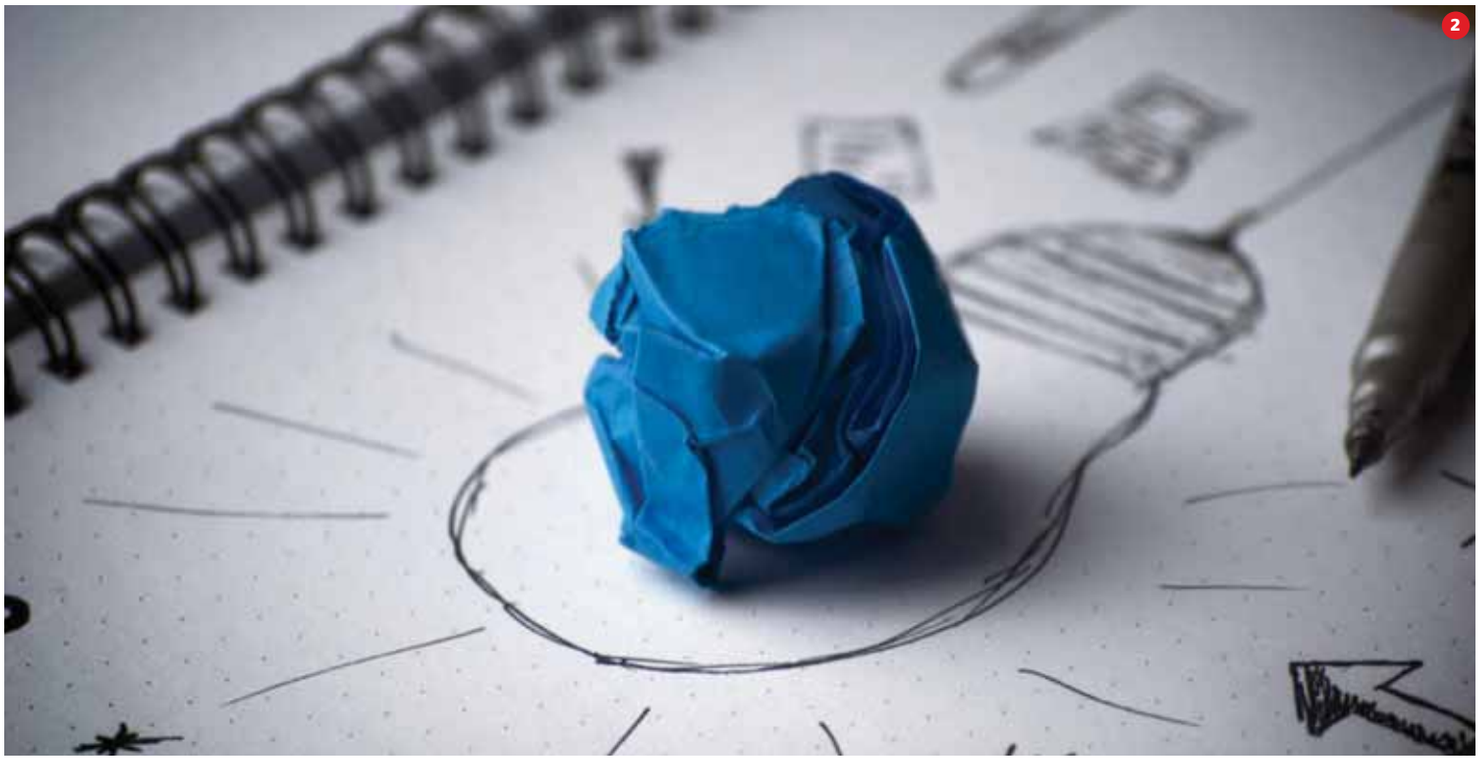
```
else {
            transitionTimer = Math.max(0,
transitionTimer - 1);
        }
        multiTexturesPlane.uniforms.
transitionTimer.value = transitionTimer;
    });
}
```



How to research your keywords

Ranking for the right keywords can make or break your website, meaning research is absolutely vital





By researching your market's keyword demand you can not only learn which terms and phrases to target, but also learn more about your audience.

Your primary objective shouldn't be ranking for just one single keyword but increasing the total domain authority through content quality and relevancy to your target audience.

Keywords have long played a central role in SEO and driving traffic to websites. Originally the mechanics were straightforward, although today the use of keywords is much more complicated. Google's algorithms have become much more complex (Rank Brain AI, in particular), and they now evaluate the intention behind your query and then search for a 'best fit' candidate in its network of indexed sites.

Clarifying the intentions and purpose of your business to Google is therefore paramount in achieving good quality traffic. The modern customer journey is also complex, so it's important to focus on the key moments that can help inspire people to interact and engage with your business.

With the evolution of SERP features and a more personalised approach to search results, you cannot solely rely on a page 1 keyword ranking to get you as much traffic as possible. Here's how you can master keyword research...

1. Understand short and long tail

Researching the terminology that inspires your audience and encourages them to convert is crucial. These high intent keywords are an excellent opportunity to connect with qualified, conversion-ready audiences. These keywords can be broken down into:

- Short tail (shorter, more generalised with greater search volume).
- Long tail (longer, more specific with less search volume, which also has an impact with voice search).

These variants will have an impact on impressions, click through rates and competition.

2. Start with ideation and research

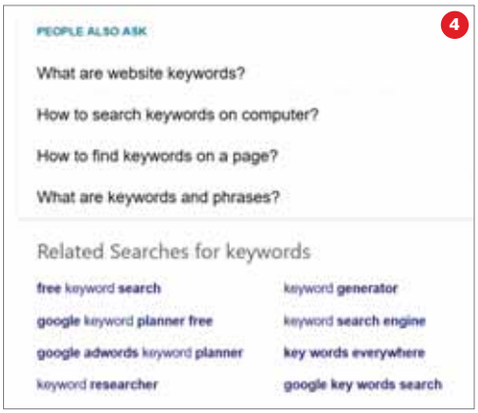
A good place to start is with Ideation. Brainstorm your ideal customer's keyword search terms and establish a baseline list. These seed keywords define your niche and describe your service. Get into the mindset of your audience: who they are and what are their pain points; get to know them better by studying their terminology.

If you can, utilise your 'search box' result terminology to outline what your target customer is thinking during their buyer journey. Brainstorm their Awareness stages (Inspiration-pain points), Consideration stages (research-comparison) and Decision stages (purchase-advocacy).

3. Google your initial ideas

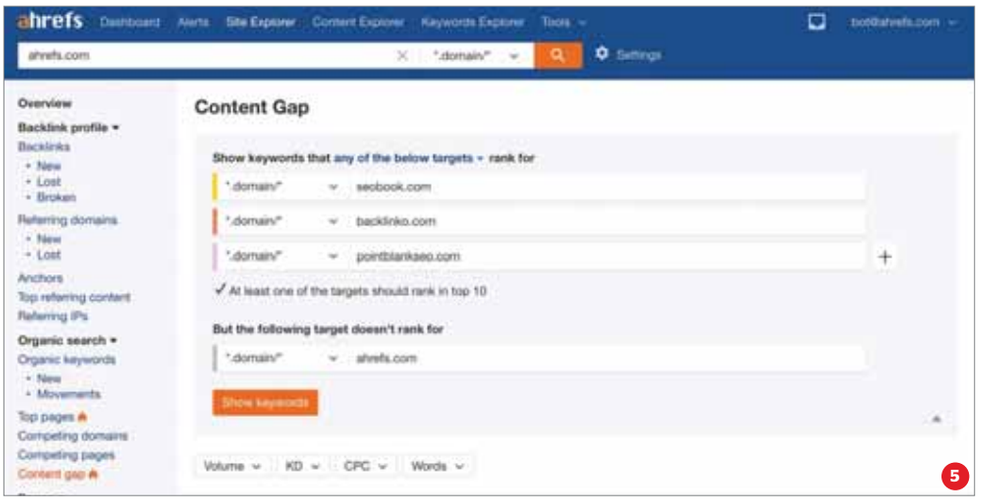
The next step is to understand what keyword terminology, in line with your ideation list, is used by your audiences in the search engines. Investigate the SERP and review what Google believes is the most relevant to your keyword.

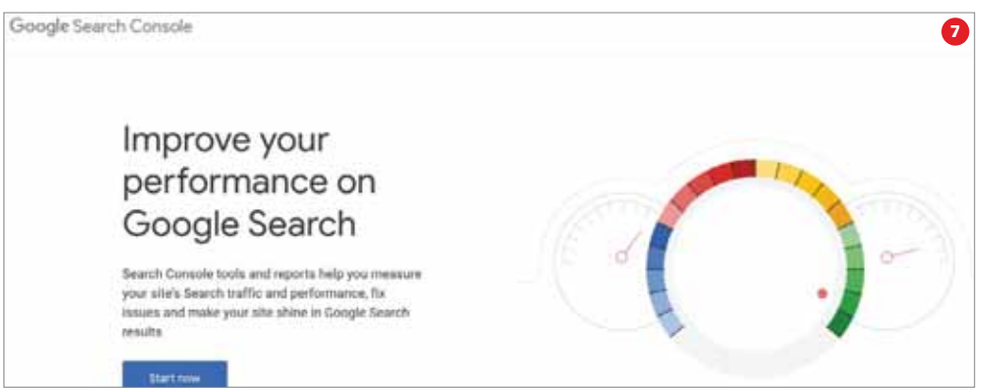
There is also an opportunity to review the 'searches related to' field in Google (bottom of the page). This will showcase a wealth of related keyword searches aligned with your initial thoughts. You can also head over to Google Trends (https://trends.google.com/trends) and review how your keywords have evolved. This will give you a good indication of interest and related topics due to seasonality, geography/location and media coverage to add further weight and terms to your list.



Advantageous analysis

'Site Explorer' in AHRefs (**https://ahrefs.com**) enables you to browse the keywords your competitors rank for, closing the gap between those terms you are not taking advantage of and ultimately increasing traffic to your own domain. The 'Content Gap Tool' also highlights keywords that all your competitors rank for, but you don't.





Leverage featured snippets

A rich answer is a snippet that contains a brief answer to a search query. It appears above other organic search results and thus enjoys more exposure. Identify 'keyword questions' you might answer (such as FAQs) and create great answers, then amplify reach.

4. Don't forget about Bing

Bing is increasing its share of the search engine market, with the mass release and rollout of Windows 10. The savvy marketers will capitalise on this and understand that Bing may have a certain demographic to target.

Keyword terminology will therefore be a key factor with this channel and nuisances between this and Google will be evident. Research keyword terminology around this older demographic and the opportunities that come from it.

5. Analyse your competitors' organic keywords

Having an insight into what your competition is doing well – and what keywords they rank for – can give you a huge advantage.

Review the websites appearing on the first page naturally alongside your ideation list. Dig deeper into these websites, understanding the keyword terminology used within their URLs, header hierarchy, meta titles and descriptions.

6. Analyse your competitors' PPC keywords

Look at what keyword terms the competition is bidding on. You can do this as per your initial research process utilising your ideation list within the search engines. As well as utilising third-party tools to save time, a useful tool that can help with this is SpyFu (**https://www.spyfu.com**). Simply enter the competitor's website domain and press enter.

7. Understand what you already rank for

As well as building your keyword list, knowing what you already rank for can increase opportunities to target keyword placements outside of page 1 positioning. This can highlight 'easy wins' with Page 2 positioning's that need a simple push!

Go to Google Search Console, click Search Traffic> Search Analytics, filter by Queries and click 'Clicks/Impressions/CTR/Position'. Then select Dates and Filter by Position to view the positioning and terminology.

8. Set primary and secondary keywords

Now you have a list, select a primary keyword and a set of related secondary keywords that share your searcher's intent (understanding what their motivation is crucial). The intent behind these keyword terms and phrases should be the same, so the same landing page content can ultimately serve it. Employ these primary, secondary and related keywords in the page's content, metas and links.

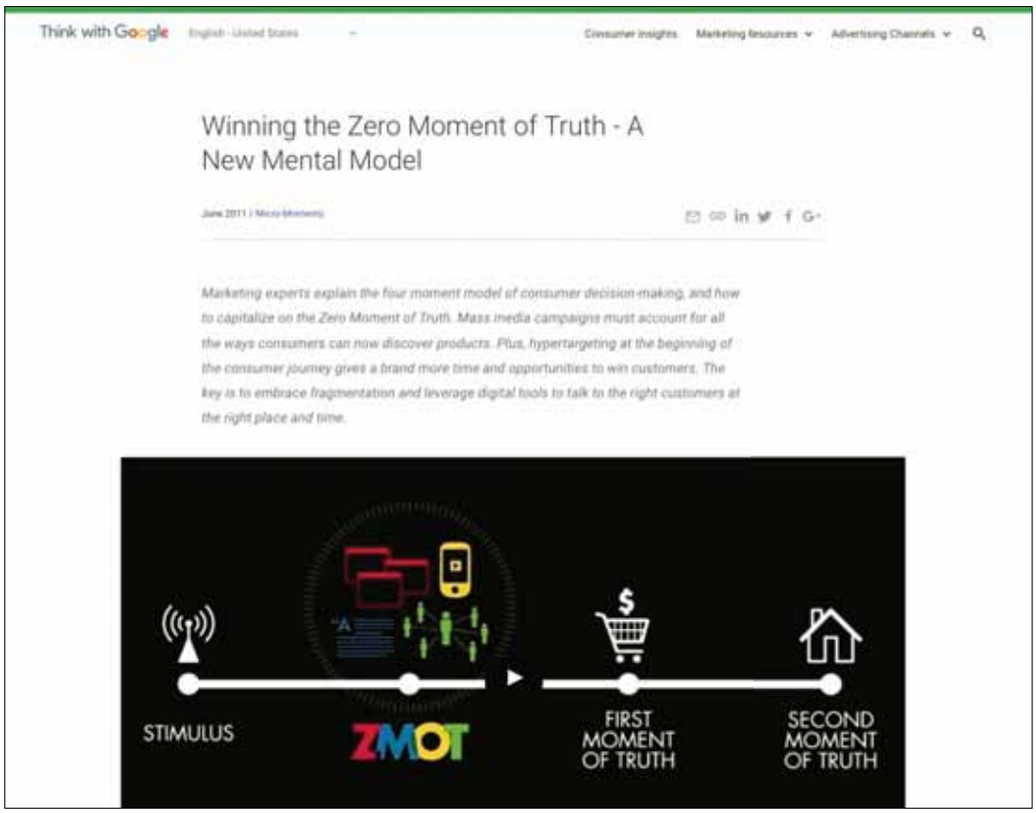
9. Gain some metrics

Use your keyword list inside the free 'Google Keyword Tool' (**https://adwords.google.com/home/tools/keyword-planner**). Here you can review these metrics:

- **Search Volume:** understand the search demand for a keyword and utilise this alongside 'Google Trends' for insights on seasonality.
- **Keyword Difficulty**: is a balance between the business value of the keyword and its ranking difficulty. Invest in where you will receive the best return and be patient for the results.
- **Clicks:** Having volume is great, but PPC adverts, localised map listings and rich snippets can immediately answer or steal clicks. Be mindful of this metric when selecting keywords.

10. Structure your keyword list

Having generated a list and used the metrics to identify the very best keywords, it's now time to add some structure to your list. Ideally this process is whatever makes the most sense to you. For example, group by keyword topic and landing page, or group by user intent, or group by business value.



Prioritise actions based upon Groupings

Group by Topic and Page: find which keywords (primary and secondary) are semantically and contextually related, and group them under a 'highest volume keyword' to target within a single page. Highlight the keywords and the page within your spreadsheet.
Group by Intent: at what stage is your audience in the buyer's journey (ZMOT – UMOT); what is their intent;what are their expectations? This moves us into the implementation of the design and content of the page, building the page to match expectations and keyword relevancy.
Group by business value/ commercial intent: be mindful of vanity metric keywords and those which will drive a clear ROI.
Prioritise: The aim with any keyword research is to utilise the findings across your website content and back-end metas. The best keyword strategy is a diverse one that understands each page will have its own challenges. Review each page individually and embrace the flexibility of topic-based, location, short tail and long tail keywords.

7 useful research tools

Ubersuggest
neilpatel.com/ubersuggest
Generate more suggestions of your keyword terms with this free tool, plus data on volume, seasonality, competition and cost per click.

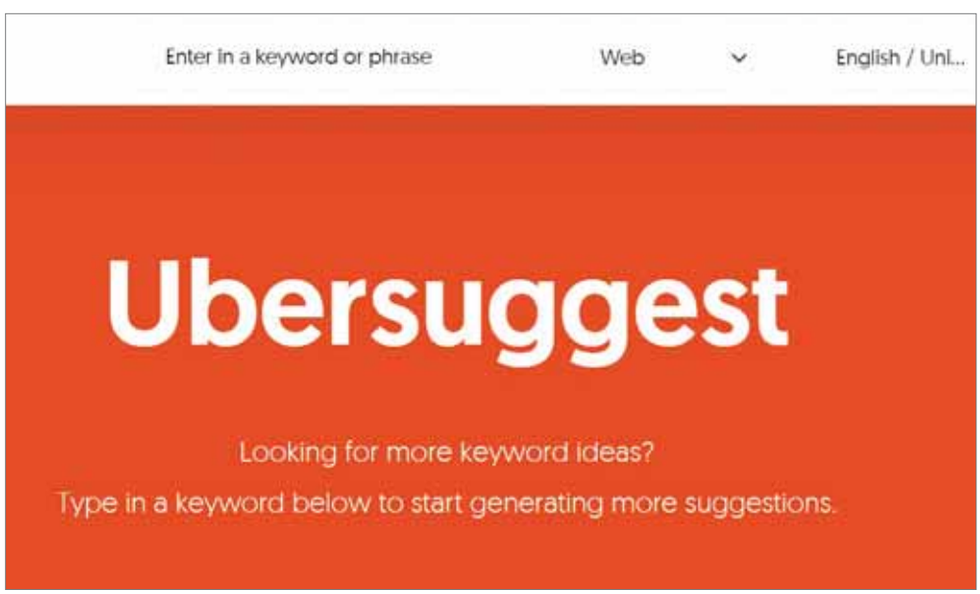
Keyword Tool
keywordtool.io/google
A free online keyword research tool that uses Google Autocomplete to generate hundreds of relevant long-tail keywords for any topic.

Wikipedia
www.wikipedia.org
Gain new keyword ideas from 'Wikipedia' articles around your existing keyword list (a wealth of succinct content that explains concepts in terms people understand and search for).

Quora
www.quora.com
Insights into the keywords used by your audience in their niche communities, notably when their questions were not answered properly with a Google search.

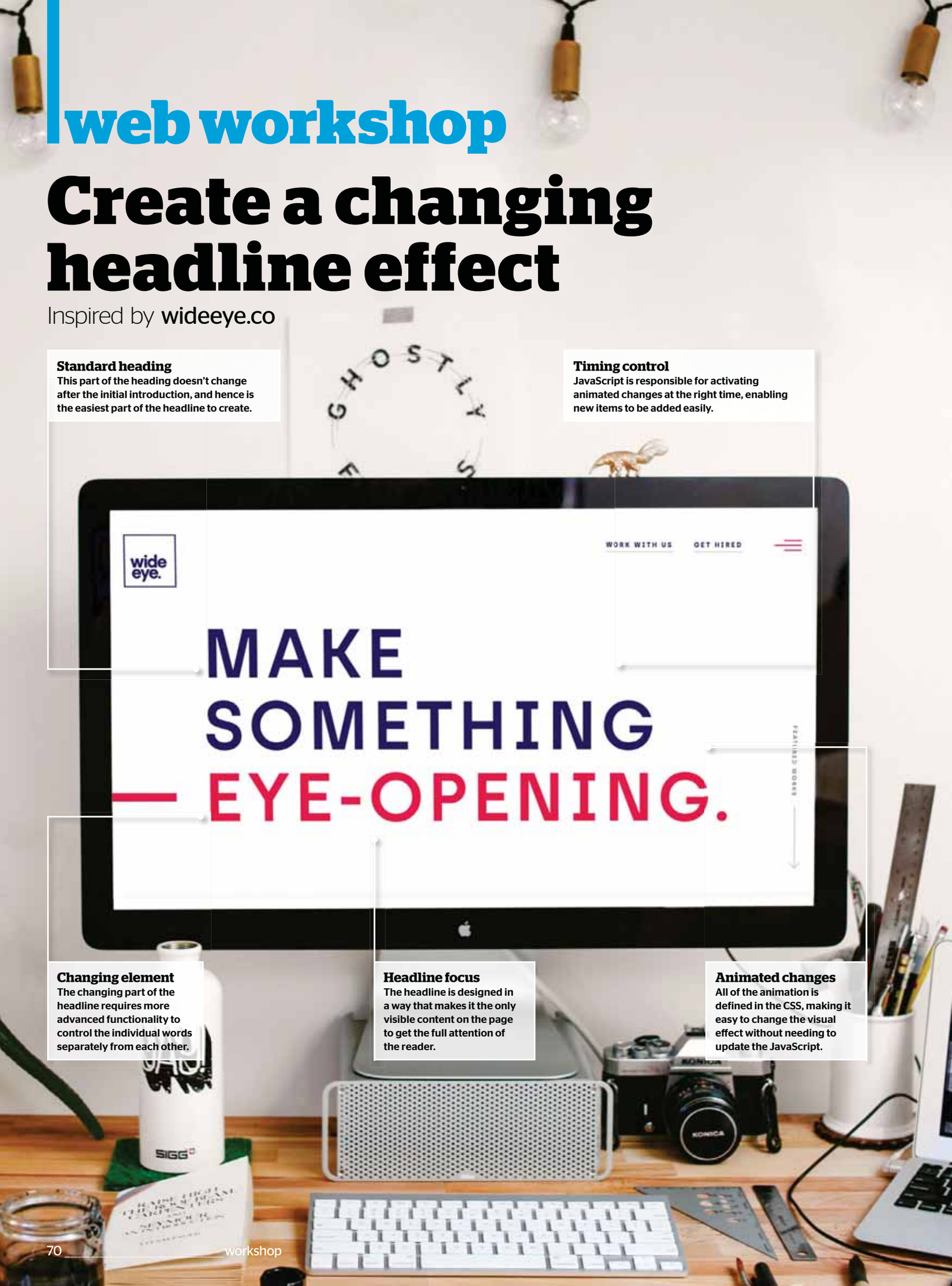
Thesaurus.com
www.thesaurus.com
Substitute in synonyms of your keyword list.

Moz & SEMrush
www.moz.com
www.semrush.com
Both operate a keyword database of their own, providing you with further keyword ideas and metrics.



Create a changing headline effect

Inspired by **wideeye.co**



Standard heading
This part of the heading doesn't change after the initial introduction, and hence is the easiest part of the headline to create.

Timing control
JavaScript is responsible for activating animated changes at the right time, enabling new items to be added easily.

Changing element
The changing part of the headline requires more advanced functionality to control the individual words separately from each other.

Headline focus
The headline is designed in a way that makes it the only visible content on the page to get the full attention of the reader.

Animated changes
All of the animation is defined in the CSS, making it easy to change the visual effect without needing to update the JavaScript.

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

<comment>
What our experts think of the site

Keep it simple and separate

The content, styling and functional code are created in a way that keep everything separate. While JavaScript is used to control the animation timings, it only alters the page with the required class to trigger visual changes from the CSS. There is no dependency between the HTML, JavaScript and CSS.

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

Technique

1. Document initiation

The first step is to initiate the web page document. This consists of the document container that stores the <body> and <head> sections. While the <head> section is used to load the external JavaScript and CSS resources, the <body> section is used in Step 2 to store the web page's content elements.

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

2. Page body content

The content is made from an <h1> heading that is used to store a collection of tags. Each tag stores one row of content – with the last tag storing the items that will be repeatedly animated.

```
<h1>
        <span>Testing</span>
        <span>Testing</span>
        <span>
                <span>One</span>
                <span>Two</span>
                <span>Three</span>
        </span>
</h1>
```

3. CSS: General span styles

Create a new file called 'styles.css'. The stylesheet sets presentation rules applied to all elements inside the <h1> tag. 'Width' and 'overflow' are set for invisibility by default, with 'transition' set to animate change on width. These elements appear at full width when the 'open' class is applied.

```
h1 span{
        position: relative;
        display: block;
        width: 0;
        min-height: 1em;
        transition: width 3s;
        overflow: hidden;
}
h1 span.open{
        width: 100%;
}
```

4. CSS: last item spans

Child spans for the repeating animation require additional settings for their position and initial visibility. Using their parent's relative positioning, these elements are placed in the top-left corner of their parent's starting position. Visibility is activated when the 'open' class is applied. The 'span' items within the first level container require specific styling for their animation.

```
h1 span > span{
        position: absolute;
        top: 0;
        left: 0;
        opacity: 0;
        color: red;
}
h1 span.open > span.open{
        opacity: 1;
}
```

5. JavaScript Event function

Create a new file called 'code.js'. This step initiates the 'effect' function, which will repeatedly call itself every two seconds (2000 milliseconds) to update the animation. This function is kickstarted by an immediate call after the web page has loaded.

```
var effect = function(){
        *** STEP 6 HERE
        setTimeout(function(){
                effect()
        },2000);
}
window.addEventListener("load", function(){
        effect();
});
```

6. Opening first level spans

A search is performed to find the first child 'span' inside the <h1> parent that doesn't have the 'open' class. The 'open' class is applied If its parent is the 'H1' element, triggering the CSS animation. If not, the code found in Step 7 is executed instead.

```
var item = document.querySelector("h1
span:not(.open)");
if(item.parentNode.tagName == "H1"){
        item.classList.add("open");
}else{
        *** STEP 7 HERE
}
```

7. Find last span children

Children of the last 'span' element require different treatment for their animation. A variable, 'n', references the index of the item to update. Siblings to the item are identified for application of Step 8. The sibling at position 'n' is updated with the 'open' class – triggering the CSS animation.

```
var n = 0;
var nodes = item.parentNode.children;
for(var i=0; i<nodes.length; i++){
        *** STEP 8 HERE
}
nodes[n].classList.add("open");
```

8. Node class check

Inside the 'for' loop from Step 7, each item is checked to see if it contains the 'open' class. If it does, the 'n' variable is updated with the next available index position – reset to '0' if beyond the last item. This step also attempts to remove the 'open' class if it exists.

```
if(nodes[i].classList.contains("open")){
        var n = i+1;
        if(n >= nodes.length)n = 0;
}
nodes[i].classList.remove("open");
```

NEVER MISS AN ISSUE!

Issue 279

Pro tips to build animation for the web, 23 ways to make your app shine, work with variable fonts and what's new in Node 10

Issue 278

What's new in JavaScript, say hello to Google Flutter, build with CSS Grid, work with WebRTC and create patterns with CSS

Issue 273

SEO Today, 20 hot new libraries and frameworks, ten pages of ReactJS tutorials, get started with Rust and build a WebGL racing game

GOT AN APPLE DEVICE?

Download **Web Designer's** digital edition for your iPad, iPhone or iPod touch

https://apple.co/1hsGYgl Available on the App Store

CATCH UP ON ANY ISSUES YOU'VE MISSED BY DOWNLOADING OUR DIGITAL EDITIONS https://bit.ly/2rh73Xk

Issue 277

Hot new CSS properties for today, build interactive 3D, streamline your design workflow with Sketch and code a real-time React app

Issue 276

Build your own augmented reality app, get started with Three.js, power up with Sass and Stylus and generate web components

Issue 275

50 must-try tools for designers and developers, how to add awesome audio, Progressive Web Apps and interactive images

Issue 274

The new rules and top tools for UX, 5 hot new CSS properties, use the CSS Paint API, add particle effects and discover JSX

Issue 272

Build with Web Components, 8 WordPress security secrets, Web Accessibility, convert visitors to customers and data visualisations

Issue 271

28-page design special. Create the perfect colour palette, top type trends, design and build grid layouts and automate your workflow

Issue 270

Get into creative coding, 16 of the best free fonts, prototype with Facebook Origami and how to power up your portfolio

Issue 269

13 must-know JavaScript techniques for React, Angular, Vue, Polymer and Aurelia, 21 killer Sketch design tips, optimise SEO and animate SVG

PREFER TO READ ON ANDROID, PC OR MAC?

Web Designer magazine is also available on Google Play and Zinio

https://tinyurl.com/yalm3wul GET THE MAGAZINE ON Google play https://bit.ly/2xPbv4p zinio

WHERE JAVASCRIPT MEETS REAL HARDWARE

JavaScript-based protocols made little ingress into the embedded space. The Web Things API working group wants to shake up the status quo

What is the Web Thing API?

Embedded systems are more than code. Mozilla's Web of Things concept takes a few interesting decisions which we analyze before diving in

The Internet of Things usually isn't heterogeneous. Developers working on hardware pick protocols without much afterthought. The approach might annoy web developers but makes good sense. Keep in mind that many, if not most electronics goods in the market run on simple microcontrollers.

Providers of cloud or integration services tend to work with a multi-tier approach. Powerful smart devices such as a process computer can connect directly to cloud infrastructures – keep in mind that real-time operating systems usually are not happy about additional jobs.

INTRODUCING MR. GATEWAY

Power-constrained devices cannot handle the (pretty large) burden of a TCP-IP stack. They tend to use embedded wireless protocols distinct from the ones used on the internet at large.

The WoT working group addresses this problem via a gateway. Think of it as a transparent translation engine between things and the cloud, taking care that message exchange takes place in a format both sides understand.

A QUESTION OF PRESENTATION!

While MQTT has largely done away with custom binary protocols, the HTTP protocol and its friends have not made significant inroads. At first glance, this is understandable - overhead is extremely high, parsing JSON on an eight-bit microcontroller with less than 1 kB of RAM is not particularly funny.

Due to this lack of presence, the Web of Things workgroups started with a clean sheet design – it, sadly, led to two completely different approaches. API number one is the Web of Things (WoT) Scripting API, documented at **https://www.w3.org/TR/wot-scripting-api**. It specifies things and an accompanying JavaScript API, developers can use it to interact with hardware.

On the Mozilla side, the ecosystem offers the Web Thing API at **https://iot.mozilla.org/wot**. It forgoes JavaScript for the use of REST and WebSocket APIs. One extremely interesting aspect involves the use of vendor-independent properties, actions and events.

The idea involves the use of abstractions – a smart light behaves much the same, independent from which vendor created it. As long as vendors implement the property as described, lights behave the same. These units can then be combined into clusters - for example, a smart fridge with an LED might expose the smart light interface to make complication with its decorative light easier.

What can the WoT API do?

Here's five things that WoT can do fo you

1 Enable hardware access from the web

Handling protocols like MQTT from JavaScript directly is difficult. The Web Things API provides standardised interfaces for hardware interaction.

2 Provide an abstraction layer

Sending specified messages to each device becomes tedious. Thanks to the predefined properties, behavior can be standardised across device classes.

3 Describe hardware in a standardised way

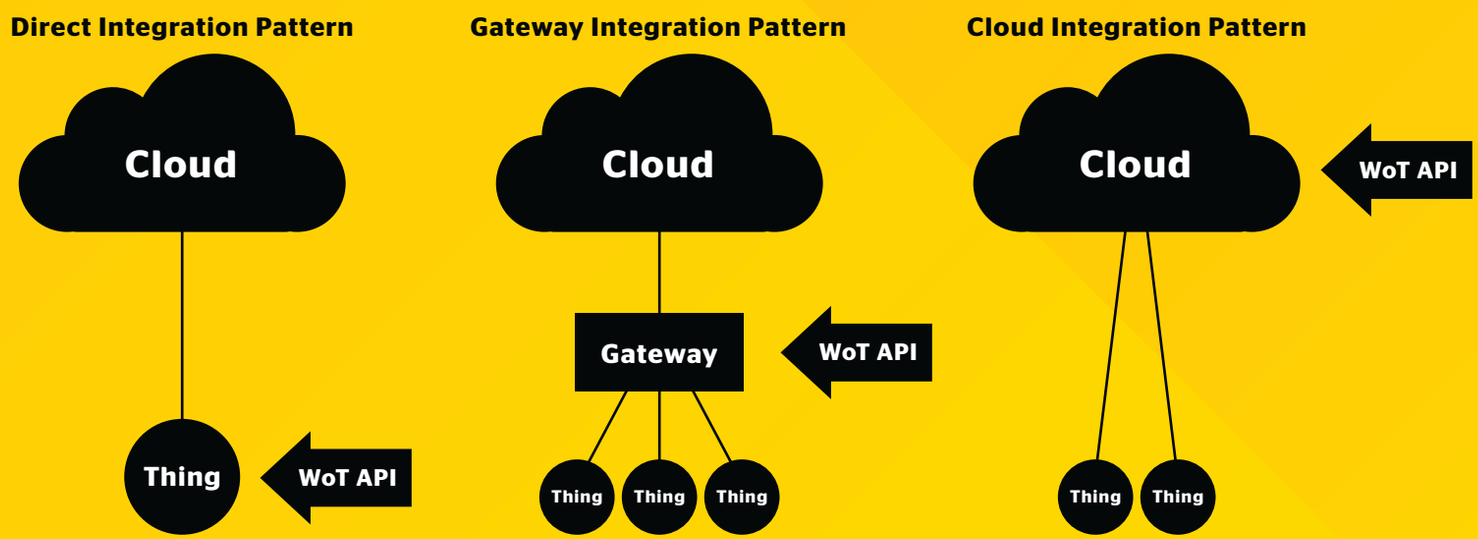
Once a JSON file is specified, your hardware is fully described. Other programmers can create use this standardised description.

4 Create smart home systems

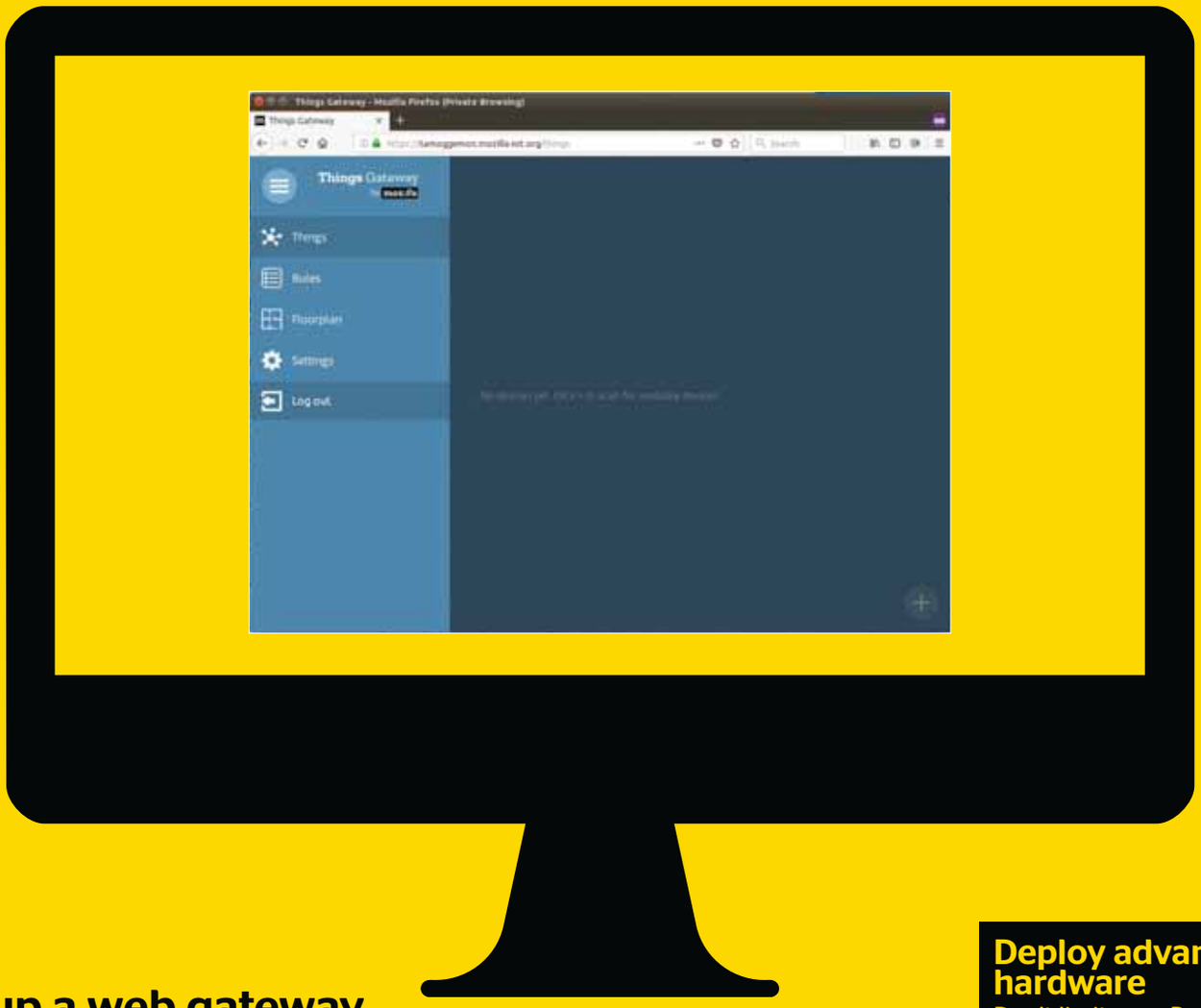
Smart homes are a hodge-podge mix of hardware components from different vendors. The Web of Things API makes things play along nicely.

5 Enhance machine usability

Adding rich user interfaces makes goods appear more valuable. Using standardised APIs greatly reduces the amount of code needed.



The gateway acts as a translator between things and the cloud



Fire up a web gateway

Experimenting with Web Things becomes easy once a Raspberry Pi acts as a gateway

As discussed on the preceding page, connecting small Internet of Things devices to the web requires the presence of the gateway. The simplest way to create one is a Raspberry Pi.

The process computer currently is available in multiple versions. We recommend the Raspberry Pi 3 Model B.

Getting started involves visiting the website **https://iot.mozilla.org/gateway/**. Scroll down to find the green download button, and click it to start the download. As of this writing, the latest version of the protocol is 0.5.3. When downloaded, extract it and burn it to an SD card just like any other operating system.

The actual image file is less than 4 GB as of this writing - yours truly installed on one of his trusty 8 GB memory cards. Finally, connect an ethernet cable and a sufficiently powerful power supply to initialize the startup process. Using cheap power supplies with Raspberry Pi three is not recommended – Upton's engineers provide various forms of toggling which allow the process computer to slow itself down when it considers its power supply to be insufficient.

After connecting power, a start-up time or between ten and twenty minutes is normal. Do not wonder if a command prompt shows up – the Web Gateway is not intended to be controlled via mouse and keyboard. Instead, an IP address is emitted – open it in a browser of choice to reveal a WiFi setup assistant. Our gateway can connect itself to a WiFi – Ethernet can also do the trick. After the obligatory reboot, a domain and a username can be assigned.

When setup is complete, a screen similar to the one shown in the screenshot accompanying this step will pop up. The menu on the left-hand side of the screen switches between configuration and a graphical map of things and their properties. Settings permit the loading of various plug-ins such as a driver which transforms the GPIO pins of the Raspberry Pi into addressable Things. The plus sign at the bottom of the Things window permits the adding of new hardware – keep its position in mind, for we will use it at a later stage of this tutorial.

End users will be satisfied with the Floorplan and Rules screens. A gateway can provide a geospatial overview of the location of things, and can even create IFTT-like rules. These, however, are not of relevance for us.

Deploy advanced hardware

Don't limit your Raspberry Pi to WiFI!

While Wi-Fi is commonly used when networking computers, the IoT space is relatively hostile to it - Wi-Fi transmitters can easily use more than 100mA of current when transmitting information.

Mozilla's IoT Gateway can include support for various wireless protocols which are better suited to the Internet of Things – think about systems like Z-Wave or the well-known Zigbee protocol stack.

Firing them up is as easy as connecting an adapter, which usually plugs into one of the four USB ports at the back of your Raspberry Pi. More information about compatible modules can be found by visiting the supported hardware list, which the Mozilla team kindly provides at **https://github.com/mozilla-iot/wiki/wiki/Supported-Hardware**.

Fire up the ESP32

Emulating web things is boring. Espressif's ESP32 lets us build something real

Espressif does not offer its core libraries via the Arduino deployment system – instead, code must be downloaded using GitHub via the command git clone **https://github.com/me-no-dev/ESPAsyncWebServer**.

Compress the contents of the received repository to create a file called ESPAsyncWebServer.zip, and install it using the Manage Libraries option of the Arduino IDE. Repeat the process with the second library, which resides at **https://github.com/me-no-dev/AsyncTCP**. A full archive can also be found at **http://www.tamoggemon.com/test/2018/esp32-webthings-libs.zip**.

Finally, click Sketch → Include Library → Manage Libraries, look for "ArduinoJson" and select version 5.13.2 for installation – beta versions of V6 are not supported. The final missing library is called webthing, as of this writing, version 0.4.1 is current.

CREATE A SKETCH

Our program starts off by including a set of headers providing access to the hardware. We connect an LED to GPIO 12 in order to let our thing emit information to the outside world.

```
#include <Arduino.h>
#include "Thing.h"
#include "WebThingAdapter.h"
const int ledPin = 12;
```

Things expose a set of properties. In the case of the Arduino library, create a few global variables to hold status and other information:

```
WebThingAdapter* adapter;
const char* ledTypes[] =
{"OnOffSwitch", "Light",
nullptr};
ThingDevice led("led", "A LED",
ledTypes);
ThingProperty ledOn("on", "",
BOOLEAN, "OnOffProperty");
```

The bulk of the code is responsible for the establishment of a WiFi connection. Be sure to replace ssid and password with values applicable to your local network.

```
bool lastOn = false;
void setup(void){
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, HIGH);
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  bool blink = true;
  while (WiFi.status() != WL_
CONNECTED) {
    delay(500);
  }
  digitalWrite(ledPin, HIGH);
  Serial.println("");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  adapter = new
WebThingAdapter("w25", WiFi.
localIP());
```

Once the WiFi stack is connected to a network, information about the IP address can be emitted to the serial monitor of the IDE:

```
  led.addProperty(&ledOn);
  adapter->addDevice(&led);
  adapter->begin();
  Serial.println("HTTP server
started");
  Serial.print("http://");
  Serial.print(WiFi.localIP());
  Serial.print("/things/");
  Serial.println(led.id);
}
```

The last part of the program provides CPU time to the adapter during invocations of the loop() function:

```
void loop(void){
  adapter->update();
  bool on = ledOn.getValue().
boolean;
  digitalWrite(ledPin, on ? LOW :
HIGH);
  if (on != lastOn) {
    Serial.print(led.id);
    Serial.print(": ");
   Serial.println(on);
  }
  lastOn = on;
}
```

When done, install the program on your ESP32 and note down the URL emitted in the serial monitor – in the following steps, we assume http://192.168.1.105/things/led

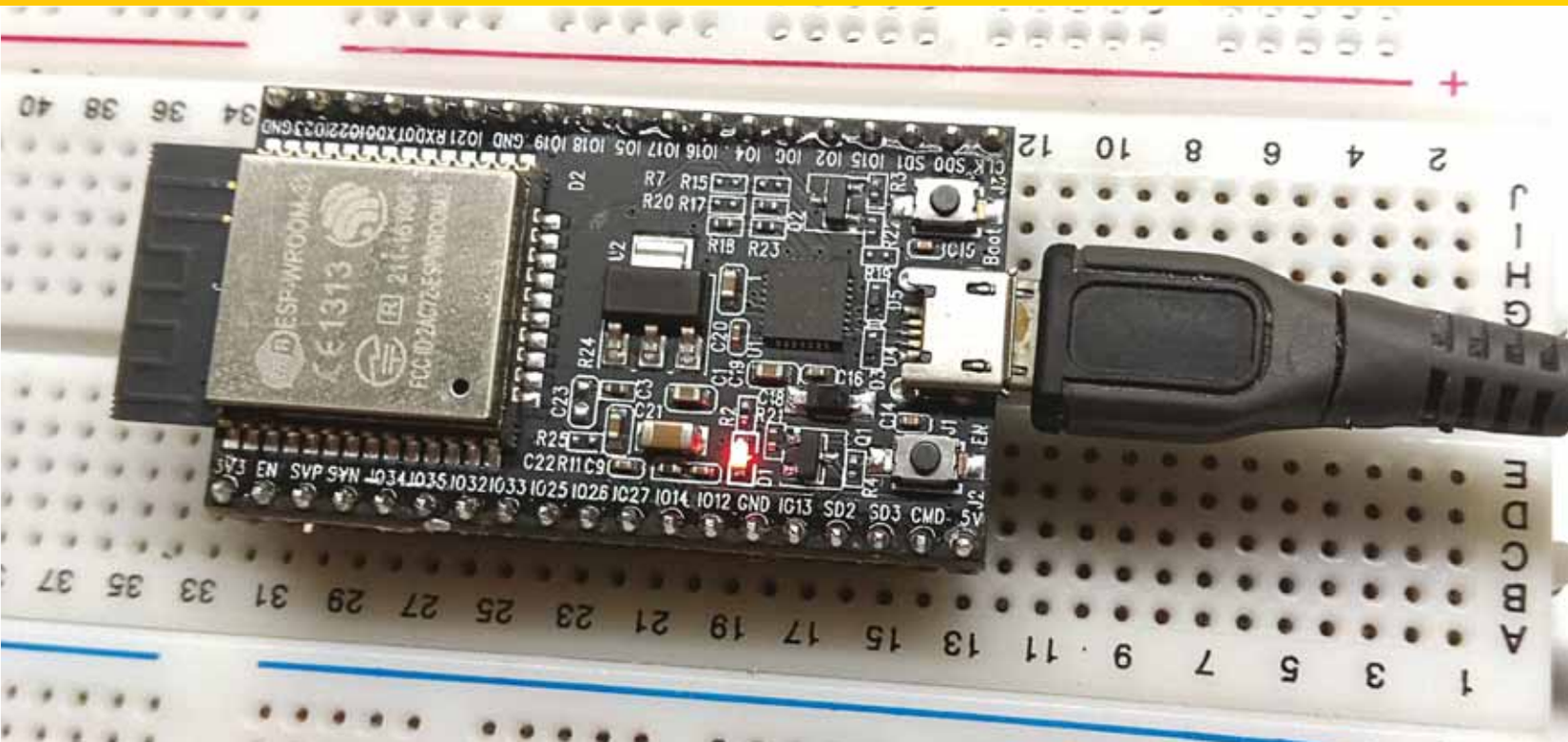
Set things up

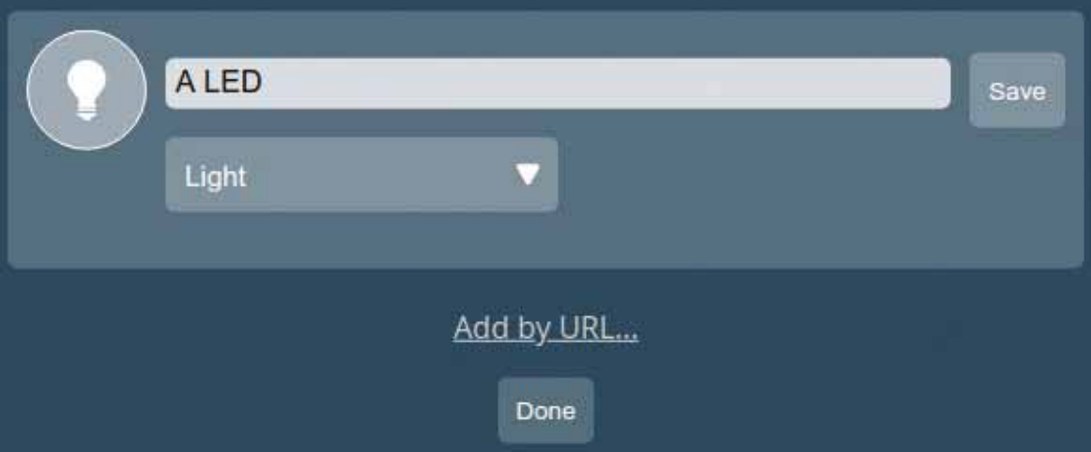
Let us guide you through the ESP32 world

ESP32 planars come dime a dozen. We will use the DEVKIT C, which is conveniently available from **https://www.elektor.com/esp32-devkitc**. First, make sure that all relevant drivers are installed and that the serial port becomes visible. As we work on Linux, we expect it to mount at /dev/ttyUSB0.

Open a recent Arduino IDE (>=1.8), and use the board manager to install an ESP32 core according to the steps outlined at **https://github.com/espressif/arduino-esp32/blob/master/docs/arduino-ide/boards_manager.md**. Next, open the Board choice tool and select "ESP32 Dev Module".

Finally, use the Tools menu to select the correct serial port. When setup is complete, deploy a sketch of choice – if the programming process succeeds, all is well.





Interact with “things”

Creating a thing is but half of the problem

First, ensure that the Raspberry Pi and the ESP32 are connected to the same network. Open the Things view of the Gateway, and click the plus symbol at the bottom-right hand side of the screen. The gateway will usually be able to find the LED as shown in the figure – if it cannot, click Add by URL and enter the address emitted into the serial monitor.

When the adding process is completed, the lamp can be turned on and off from the web browser. As the example expects the LED to be active low, switching it on “digitally” may lead to the LED turning off.

“When the adding process is completed, the lamp can be turned on and off from the web browser”

GETTING PERSONAL WITH THE THING

Open http://192.168.1.105/things/led in a browser of choice.

```
{“name”:”A LED”,
“href”:”/things/led”,
“@context”:” https://iot.mozilla.org/schemas”,
“@type”:
    [“OnOffSwitch”,”Light”],
“properties”:
    {“on”:
        {“type”:”boolean”,
        “@type”:”OnOffProperty”,
        “href”:”/things/led/properties/on”}}}
```

As our lamp has but one attribute, open http://192.168.1.105/things/led/properties/on/. The thing will respond by returning a simple JSON object:

```
{“on”:true}
```

CHANGE THE VALUE

While more complex objects tend to specify actions, our property limits itself to receiving GET and PUT requests in accordance to the HTTP specification. On unixoid operating systems, CURL provides a convenient and fast way to send PUT requests. Simply open a command line and try one of the two commands:

```
tamhan@TAMHAN14:~/$ curl -X PUT -d “{“on”: true}” http://192.168.1.105/things/led/properties/on/ --header “Content-Type:application/json”
tamhan@TAMHAN14:~/$ curl -X PUT -d “{“on”: false}” http://192.168.1.105/things/led/properties/on/ --header “Content-Type:application/json”
```

Both methods are, by and large, similar. Make sure to pass a content header along with the data field – other than that, the commands are vanilla.

Web Things from cyberspace

Working with hardware isn’t everybody’s thing. If you feel like using a virtual thing instead of the real deal, Mozilla also has you covered. Developers working with Android can find a Java library at **https://github.com/mozilla-iot/webthing-java** – it transforms any JVM-based device into a Web Thing. Friends of Node.JS find a complete implementation at **https://github.com/mozilla-iot/webthing-node**. It provides a fully event-driven implementation of the Web Things API and comes complete with samples.

Should none of these strike your fancy, visit **https://github.com/mozilla-iot/webthing-python**. It provides a Python 3.5 library which does essentially the same thing – keep in mind that most programming languages can somehow embed Python code. Finally, the code can also be rewritten – TCP/IP support and a JSON parser are almost universally available.

Web Things without ESP32

Mozilla’s predefined libraries aren’t limited to Espressif’s processors. As of this writing, the Arduino sketch can also be run on any Arduino connected to a MKR-series WiFi shield. If this does not suffice, developers can fall back to the serial adapter library hosted at **https://github.com/mozilla-iot/serial-adapter**.

It transforms a USB port into a primitive serial port. Ultra-low-end MCUs can parse the information emitted, and can thus connect themselves to the gateway without the burden of implementing the whole TCP/IP protocol from scratch. Sadly, its memory consumption still isn’t low – some incoming messages are more than 130 bytes long.

ESSENTIAL ART RESOURCES
Exclusive videos and custom brushes are available with your digital editions!

Try out a digital edition for **FREE** today!

Just search for 'ImagineFX' on these selected platforms…

Available on the App Store · nook by Barnes & Noble · zinio · Available on amazon apps kindle fire · Google play

Build interactive visual charts

Learn how to create a simple dynamic dashboard using the Apex Charts JavaScript library

We are wired for visuals. About half of our brain is dedicated for visual functions, and 90% of the information transmitted is visual. What this implies is that users are more likely to obtain clear insights from data that is visually and aesthetically presented as opposed to a slew of numbers.

Enter data visualization. In simple terms, this involves representing data graphically through forms such as charts, infographics and dashboards. While software applications ship with in-built data analytic features that provide the sought-after visualization, representing data visually in websites and web-based projects can be challenging, especially where interactive interpretation is needed. A modern JavaScript library that is tackling this problem is Apex Charts. The library serves two key roles; creating beautiful JavaScript charts and providing an effective visual and interactive representation of data. Apex Charts is open-source, easy to use and highly customisable. It also provides diverse chart types that can be combined to provide a rich and visual representation of data.

In this tutorial, the fundamentals of the library are discussed and a practical application of the library demonstrated, whereby a simple dashboard with several charts is created. Here's how...

1. Getting started

Begin by creating a folder, apex, on your desktop to store the tutorial files. Create two additional folders within it: CSS to store the styling files, and JS to store JavaScript files. HTML files will be stored in the root folder (apex). Create a file 'styles.css' in the CSS folder and 'index.js' in the JS folder. These will be used in the later section of the tutorial.

2. Creating the main HTML page structure

Open any code editor and create an 'index.html' document to contain mark up for the main webpage. Begin by creating the basic structure and give a suitable title to the page.

6 Simple Dashboard using Apex Charts

```
<! DOCTYPE html>
<html lang=”en”>
    <head>
        <meta charset=”utf-8”>
        <title>Simple Dashboard using Apex
Charts </title>
 </head>
<body> </body>
</html>
```

DESIGNING THE DASHBOARD LAYOUT

3. Adding rows

The aim of the tutorial is to create a simple dashboard layout using the library. As such, the main page will be structured into three sections, each with a different chart type using the Bootstrap grid layout. To do this, create a main container class div in the body section, and within it, add two Bootstrap rows.

```
<body>
        <div class=”container”>
                <h1> Simple Dashboard using
Apex Charts </h1>
                <div class=”row”></div>
                <div class=”row”></div>
        </div>
</body>
```

4. Adding columns

Essentially, the dashboard being created in this tutorial will contain a bar graph in the first row and two pie charts in the second row. Subsequently, the first row contains only 1 column while the second is divided into two columns.

Add the following code in the second row in order to divide it into two columns while ensuring that the first row remains unchanged.

```
<body>
        <div class=”container”>
                <h1> Simple Dashboard using
Apex Charts </h1>
                    <div class=”row”></div>
    <div class=”row”>
   <div class=”col-lg-6”></div>
   <div class=”col-lg-6”></div>
                </div>
        </div>
</body>
```

5. Specifying chart positions

Now that columns have been specified in the webpage, create new divs within the different sections where chart functionality will be added using JavaScript. Differentiate these sections by assigning a unique id to each section ('chart', 'chart1', 'chart2') respectively. Similarly, assign a unique CSS class for each section in order to facilitate styling.

```
<div class=”row”>
<div id=”chart” class=”bar-graph”></div>
</div>
 <div class=”row”>
     <div class=”col-lg-6”>
        <div id=”chart1” class=”radial”></
div>
      </div>
     <div class=”col-lg-6”>
<div id=”chart2” class=”circle”></div>
     </div>
</div>
```

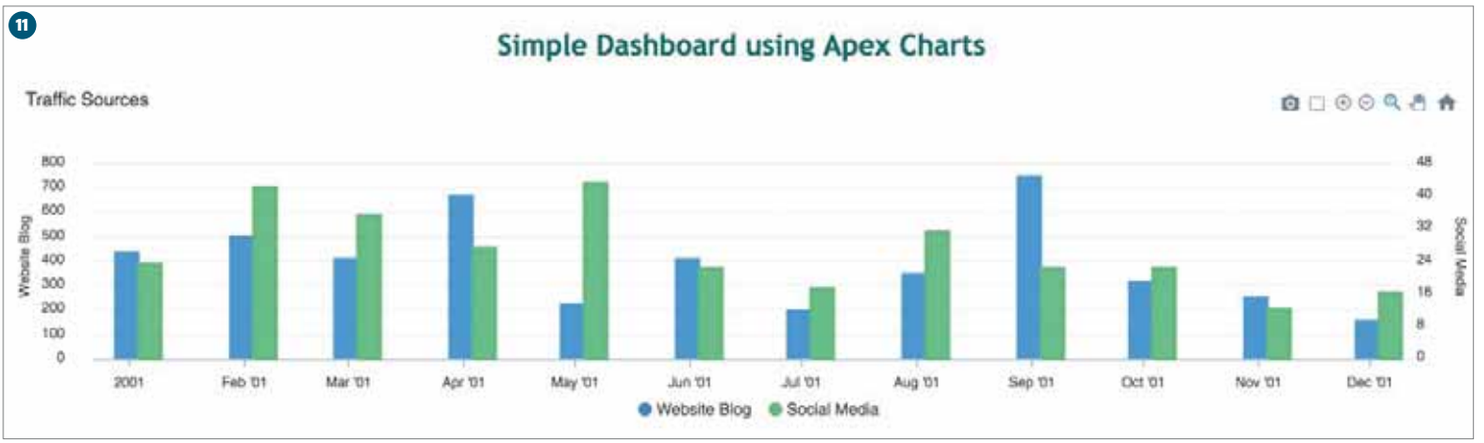
STYLING THE WEBPAGE

6. Header and page background

Next, open the 'styles.css' file created earlier and specify styling to centralise the heading in the webpage and add

Apex Charts or Chart.js?

Chart.js is also a JavaScript library that facilitates the creation of visual charts. The library's demo page **http://www.chartjs.org/samples/latest** offers diverse examples that can be studied and utilised in projects. Likewise, Apex Charts has collated several demos for review (**https://apexcharts.com/javascript-chart-demos**). Take a look at Chart.js if you cannot find what you want in the Apex Charts library.



a background colour. Copy the code below which specifies the header's font, size, colour and positioning, and specifies a white background for the dashboard.

```
h1{
    text-align: center;
    position: relative;
    margin:20px;
    top:15%;
    font-family: Trebuchet MS, Arial,
Helvetica, sans-serif;
    font-size: 25px;
    color: #167676;
}
html, body {
  background-color: #fff;
}
```

With the header centralised and background set to white, link the stylesheet in the header section by adding the following code:

```
<link rel="stylesheet" href="css/styles.css" >
```

Currently, the landing page is very basic as it only contains coloured text on a white background.

7. Preparing to add JavaScript functionality

Begin by linking the Apex Charts JavaScript library to the webpage. Two alternatives exist: either downloading the file from the official website **https://apexcharts.com/downloads/apexcharts-bundle.zip**) or directly linking it from a CDN source such as unpkg or jsdelivr.

In the former's case, first download and extract the bundle in an easily accessible section. Next, copy the 'apexcharts.min.js' file contained within it, and paste it in the JS folder created in step 1. Link the file as follows in the body section of the page:

```
<script type="text/javascript" src="path/to/
apexcharts.min.js"></script>
```

In the latter's case, simply add the link below in the head section.

```
<script src=" https://unpkg.com/apexcharts/
dist/apexcharts.min.js"></script>
```

Finally, link the 'index.js' file created earlier, by adding the code below in the body section:

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

CREATING THE DASHBOARD

8. An overview

Before proceeding to create different charts using the library, it is important to understand the flow of logic adhered to in creating them. In essence, creating the library involves only three steps.

Step 1: declaring the options variable and specifying the chart functionality.

Step 2: declaring the chart variable and passing two key arguments: the document's section where the chart is targeted and the option specifications created in step 1.

Step 3: rendering the chart.

As such, we've followed this process in creating the three different charts in the tutorial.

9. The bar graph

Open the 'index.js' file created in step 1. Begin by declaring the options variable where the functionality of the Apex Chart will be defined. Creating a bar graph involves five steps: specifying the graph type, adding series data, specifying stroke, adding chart title and data labels and finally adding axis details.

Begin by specifying the graph type as line and adding the data to be visually represented as shown below.

```
var options = {
chart: {
    height: 300,
    type: "line"
  },
  series: [
    {
      name: "Website Blog",
      type: "column",
      data: [440, 505, 414, 671, 227, 413,
201, 352, 752, 320, 257, 160]
    },
    {
      name: "Social Media",
      type: "column",
      data: [23, 42, 35, 27, 43, 22, 17, 31,
22, 22, 12, 16]
    }
  ],
    };
```

The code above specifies that a line chart of height 300 will be created. Further, it will visually represent two types of data in a column format: website blog and social media.

10. Chart title

Next, add the chart title and data labels for the series added in step 10. Copy the code below which specifies a stroke width of size 4 and a smooth animation of the column data. Likewise, the chart title is added and labels specified for the different series' data points.

Add the data below.

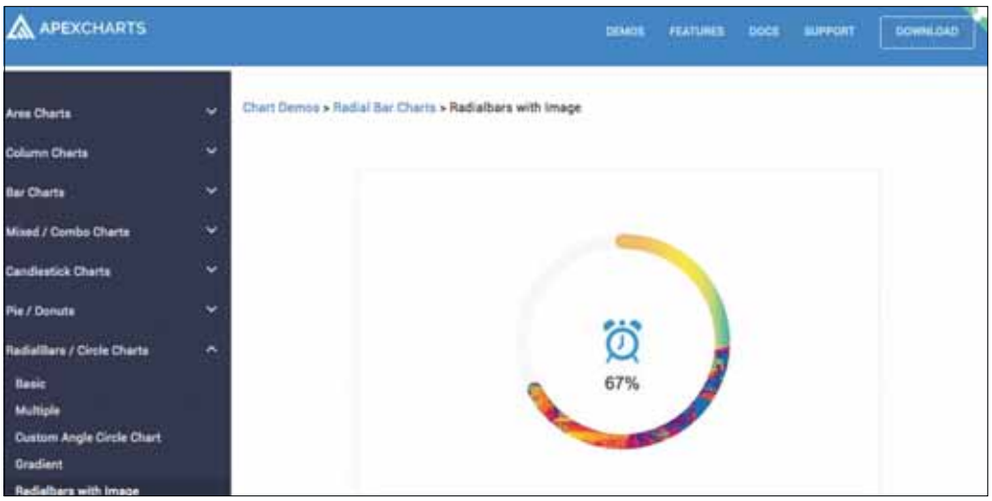
```
stroke: {
    width: [0, 4],
    curve: 'smooth'
  },
  title: {
    text: "Traffic Sources"
  },
labels: [
    "01 Jan 2001",
    "01 Feb 2001",
    "01 Mar 2001",
    "01 Apr 2001",
    "01 May 2001",
    "01 Jun 2001",
    "01 Jul 2001",
    "01 Aug 2001",
    "01 Sep 2001",
    "01 Oct 2001",
    "01 Nov 2001",
    "01 Dec 2001"
  ],
```

11. Axis details

Next, specify the axis details by indicating the data type and specific labels for each axis. Copy the code below which assigns a type of date-time to the x axis and

Using in Vue and React

React-ApexCharts and Vue-ApexCharts are wrapper components for Apex Charts that are easily integrated with applications utilising either React.js or Vue.js respectively. Refer to **https://apexcharts.com/docs/vue-charts** and **https://apexcharts.com/docs/react-charts** to review the steps involved.



Get creative with Apex Charts – design, methods and features

Once the foundational concepts of creating with Apex Charts have been understood, enhance the different charts by tweaking the design and utilising methods that are maintained in the repository (**https://apexcharts.com/docs/installation**). In terms of the chart design, the library offers the capability to use diverse backgrounds ranging from patterns and image fills to solid colours and gradients. Additional methods are also provided to make it easier to update charts once they are created. For instance, adding annotations at the x and y axes, appending data, adding text etc. Finally, various chart features can also be tweaked such as the animation, interactivity, zoom, legends and tooltips among others. For example, observe the creative use of image backgrounds in creating the radial bar.

likewise assigns labels for each axes.

```
xaxis: {
    type: “datetime”
  },
  yaxis: [
    {
      title: {
        text: “Website Blog”
      }
    },
    {
      opposite: true,
      title: {
        text: “Social Media”
      }
    }
  ]
};
```

12. Rendering the bar graph

Now that the details of the chart have been specified, a new chart variable is created and the options variable passed as an argument to the chart div created in step 5. The chart is then rendered. Copy the code below.

```
var chart = new ApexCharts (document.
querySelector (“#chart”), options);
chart.render ();
```

The bar graph should now render.

13. The pie chart (simple donut)

Next, a pie chart is created in the left column of the second row, beneath the bar graph. To do this, create a new JavaScript file ‘chart1.js’ and save it in the JS folder. As previously discussed, three steps are involved in creating the pie chart: declaring the options variable, creating the charts variable and rendering the chart.

However, unlike the bar graph, creating the pie chart is slightly different: specify the chart type and size, add a title and add the series data. Copy the code below into ‘chart1.js’, which specifies the chart type of donut, a title and the series data.

```
chart: {
                width: 300,
                type: ‘donut’,
            },

        title: {

        text: “Respondent Age”

        },
            series: [44, 55, 41, 17, 15],
            labels: [“18-30”, “31-40”,
“41-50”, “51-60”, “61-70”]          }
```

14. Rendering the pie chart

Next, render the chart by adding the two parameters to the chart variable. Note that the div ‘chart1’ specified in step 5 is referenced in order to ensure the chart renders in the appropriate column of the dashboard.

```
var chart = new ApexCharts (
            document.
querySelector(“#chart1”), options);
chart.render();
```

However, we need to style them using absolute positioning. Paste the following CSS code in ‘styles.css’.

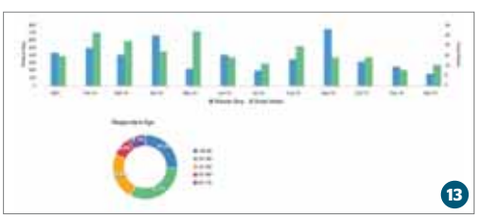
```
.circle{
    position: absolute;
    top: 55%;
    right: 20%;
    padding: 10px; }
.radial{
    position: absolute;
    top: 55%;
    left: 20%;
    padding: 10px; }
```

Both charts should now render.

15. The multiple radial bar charts

At this stage, finalise development of the dashboard by creating a multiple radial bar chart in the right column, next to the donut pie chart. Begin by creating a new file ‘chart2.js’ in the JS folder.

Four steps are involved in creating a multiple radial bar chart: specifying the type of chart, declaring the plot options, adding chart title and the series data. Copy the code over the page that specifies the chart type and declares the plot options.



```
chart: {
                height: 280,
                type: “radialBar”
        },
        plotOptions: {
                circle: {
                        dataLabels: {
                        showOn: “hover”
                        }
                }
        },
    title: {
    text: “Survey Responses”
  },
        series: [44, 55, 67, 83],
        labels: [“Disagree”, “Strongly
Disagree”, “Neutral”, “Strongly Agree”]
};
```

16. Rendering the multiple radial bar charts

Finally, render the chart by passing the appropriate parameters similar to steps 11 and 13. However, ensure that ‘chart2’ is referenced as the page section where the Apex Chart is rendered. Add the code below to chart2.js.

```
var chart = new ApexCharts (
            document.querySelector
(“#chart2”), options);
chart.render ();
```

The three charts should now render.

TWEAKING THE DASHBOARD

17. Adding annotations to the bar graph y axis

Annotations enable you to write custom text on specific values or on axes values. We can create an annotation to indicate the value of the highest web traffic (September 2001). To do this, simply paste the following code after chart.render() .

```
chart.addYaxisAnnotation({
  y: 752,
  yAxisIndex: 0,
  label: {
    text: ‘Highest Web traffic’
  },
})
```

The render should appear as shown. Notice the dotted line rendered at September.

18

18. And to the x axis

Annotations can also be added to the x axis if required. If we wanted to indicate that in May 2001 there was a system problem that led to poor performance, we can do this by simply pasting the code below after chart.render(). The render should appear smoothly. Check the code on the FileSilo download site.

19. Adding point annotations to the bar graph

Apart from adding annotations to the x and y axes, point annotations can also be added at specific points in the graph. Suppose you would like to indicate that the website was updated in September 2001 leading to the improved traffic. Simply paste the code below after chart.render ( )

```
chart.addPointAnnotation ({
  x: new Date (‘01 Sep 2001’).getTime(),
  y: 752,
  label: {
    text: ‘New Web Update’
  },
})
```

Render the chart again. Observe the annotation present at September without the dotted line as was the case in step 17.

EXPLORING THE APEX CHARTS LIBRARY

20. Robust features

Unfortunately, the tutorial cannot cover all the different unique features associated with the library. However, the resource that can be found at **https://apexcharts.com/features** provides elaborate examples of how to style the charts, utilise different colour palettes, add smooth animations, annotations and interactivity. We’d recommend exploring this section of the Apex Charts world to advance your projects further. Have fun and let us know how you get on!

17

Special offer for readers in *North America*

6 issues FREE

When you subscribe*

PLUS

you'll receive **The 100 Best Typefaces Ever** e-book!

FREE resource downloads every issue

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

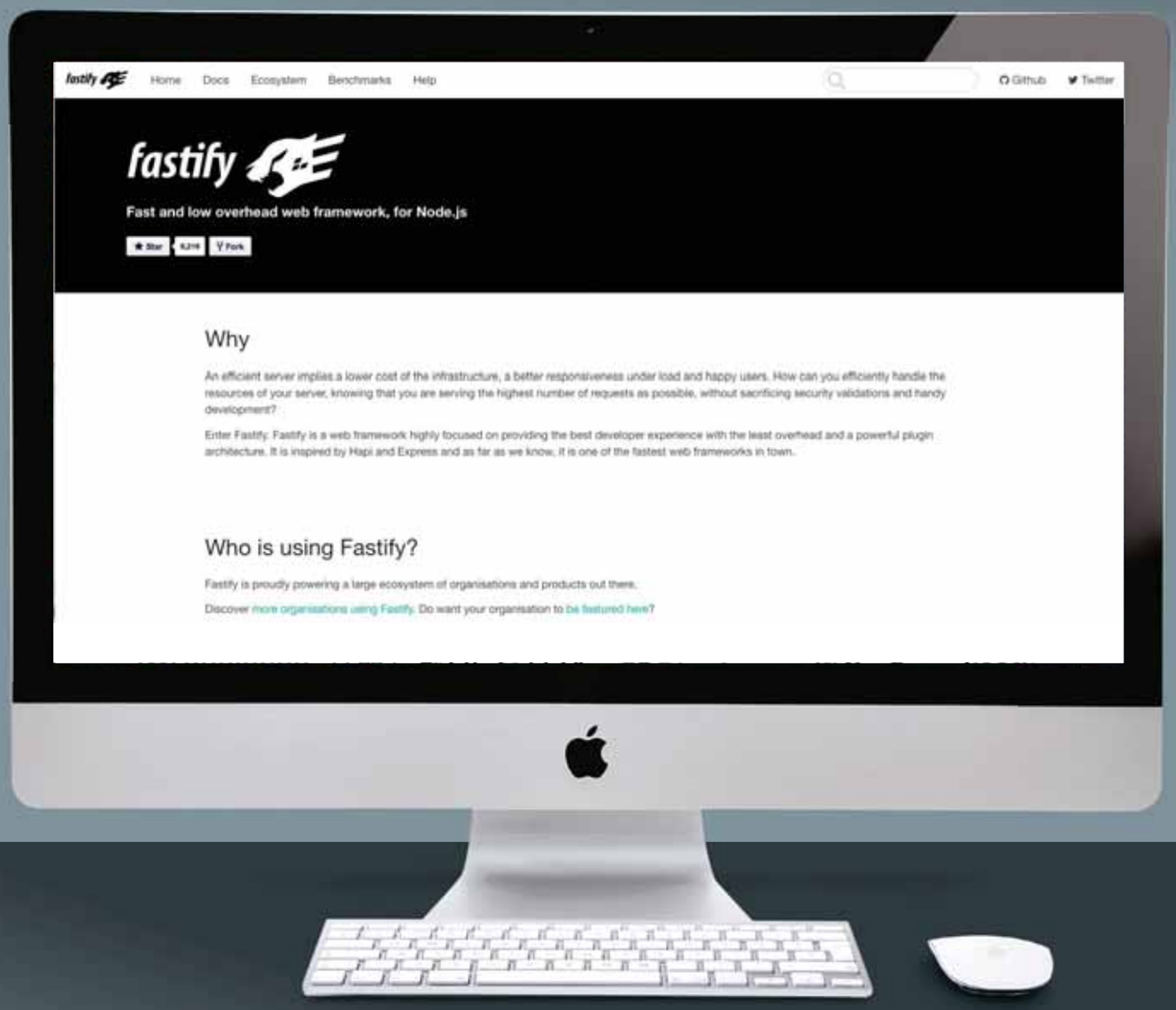
Online at **myfavouritemagazines.co.uk/WCUEBW**

Expires December 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. Please provide a valid email address to receive your e-book. 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 December 2018.

Combine Node.js, speed and HTTP

By default, Node.JS is a JavaScript runtime. Introducing the fast and low overhead web framework Fastify can make serving HTML really simple



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

Entering a URL into a web browser's address field may look innocent to end users – but in the background, a set of complex processes kick off. Implementing HTTP handshakes by hand is a tedious task, which developers should outsource to routing frameworks.

TJ Holowaychuk's seminal Express.js product is the best-known candidate. Its creator had a stroke of genius by adapting the event-handling framework known from operating systems such as Palm OS to the serving of web pages. Developers committed one or more route objects to the base framework, which the system queried as URL requests came in – the detection of a match triggered the relevant element's processing function.

As time went by, the growth of the JavaScript and Node.js ecosystems propelled Express.js into ever-larger projects. These placed stringent demands on performance – if thousands of requests are to be handled each second, even minimal overheads take a significant toll.

A team around the Italian JavaScript expert Matteo Collina set out to create a slimmed-down version of Express.js. Developers who seek to implement REST APIs or basic HTTP interactions are well-advised to take a look at the product – we're going to introduce it in more detail in the following steps.

1. Ready for liftoff

Fastify is a Node.js package like any other – installing it is best accomplished by creating a new NPM project and executing the 'npm install' command as shown in the code accompanying this step. As of this writing, version 1.11.2 is current, but breaking changes have been rare in the product's recent history.

```
tamhan@tamhan-thinkpad:~/nodespace$ mkdir
fastifytest
tamhan@tamhan-thinkpad:~/nodespace$ npm init
tamhan@tamhan-thinkpad:~/nodespace$ cd
fastifytest/
tamhan@tamhan-thinkpad:~/nodespace/
fastifytest$ npm install fastify
```

2. Open a server

Create a file called 'index.js', and start out by adding the code shown below. It uses the 'require()' command to get a reference to the framework's mother object, and proceeds to invoke its constructor. Fastify's entry point can also take various parameters, which modify program behavior as whole:

```
const fastify = require('fastify')()
fastify.get('/', (request, reply) => {
  reply.send({ hello: 'world' })
})
```

3. Get cracking

Now that the server instance is ready to rumble, invoke the 'listen' method to fire it off. The number passed in describes the network port, which Fastify will use to receive HTTP requests from clients.

```
fastify.listen(3000, (err) => {
  if (err) {
    fastify.log.error(err)
    process.exit(1)
  }
  fastify.log.info(`server listening on
${fastify.server.address().port}`)
})
```

4. Async considered possible

Fastify is a non-dogmatic framework. Developers can decide if they prefer the traditional coding style shown in the article, or if the 'async/await' pattern introduced in ECMAScript 5 is more interesting. Both programming styles are considered equal – an 'async' version of our example is shown here, while the documentation usually provides both 'sync' and 'async' versions.

```
const fastify = require('fastify')()
fastify.get('/', async (request, reply) => {
  return { hello: 'world' }
})
const start = async () => {
  try {
    await fastify.listen(3000)
    fastify.log.info(`server listening on
${fastify.server.address().port}`)
  } catch (err) {
    fastify.log.error(err)
    process.exit(1)
  }}
start()
```

5. Start it up

As with most other Node.js-based programs, our Fastify-based server starts up after you enter 'node index' into a Terminal window. Our configuration makes the server listen for input on port 3000. Harvest the JSON output with a browser of choice, as seen in the figure accompanying this step.

```
tamhan@tamhan-thinkpad:~/nodespace/fastifytest$
node index
```

6. Disable local security

Fastify's developers see the framework's main use case in the creation of APIs, which usually do not need to be accessed from the outside. As limited access scopes is a classic security design pattern, allowing 'global' access to the Fastify server requires an additional parameter for the 'listen' function.

Mind user rights

When working on Linux boxes, opening a server on ports <1024 usually requires super-user rights. Keep this in mind before invoking 'npm run' if your project uses Fastify to serve HTTP on the protocol's well-known port 80.

4

Request/Response validation and hooks

Of course, Fastify can do much more than this.

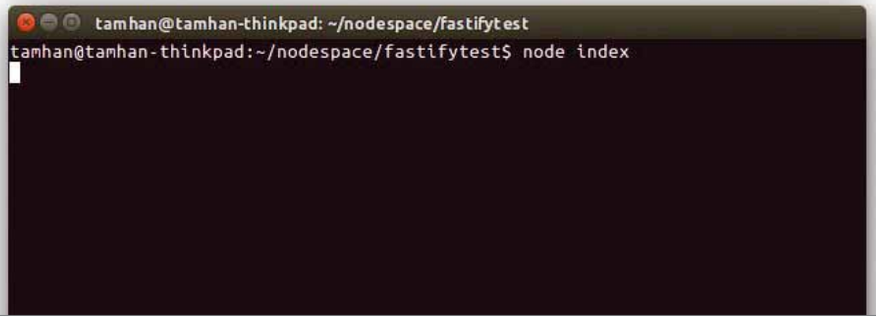
For example, you can easily provide input and output validation using JSON Schema and perform specific operation before the handler is executed:

async/await

```
const fastify = require('fastify')()

fastify.route({
  method: 'GET',
  url: '/',
  schema: {
    // request needs to have a querystring with a `name` parameter
    querystring: {
      name: { type: 'string' }
```

5



6


```
fastify.listen(3000, '0.0.0.0', function (err, address) {
  if (err) {
    fastify.log.error(err)
    process.exit(1)
  }
```

7. Handle additional HTTP verbs

Fastify is not limited to 'get' commands. The framework handles the entire verb assortment found in the HTTP protocol. Let us demonstrate this by taking a look at the dedicated functions shown in the code box accompanying this step. If you want to register a handler for 'post', simply use its method instead of the 'get' command shown above.

```
fastify.get(path, [options], handler)
fastify.head(path, [options], handler)
fastify.post(path, [options], handler)
fastify.put(path, [options], handler)
fastify.delete(path, [options], handler)
fastify.options(path, [options], handler)
fastify.patch(path, [options], handler)
```

8. Add a parametric route...

Routes are not limited to static matcher strings. If your API needs the presence of one or more parameters, they can be declared using the colon character. If a route is able to match the string, parameters arrive in the 'requests.params' object.

```
const fastify = require('fastify')()
fastify.get('/tamsapi/:userId', (request, reply) => {
        reply.send({ whatisit: request.params.userId })
})
```

9. Add verification support

JSON and XML are superior to proprietary data formats as an entire ecosystem of verification tools exist. In the case of Fastify, developers may submit both incoming and outgoing information to a JSON scheme verification process based on the venerable (and very fast) Ajv plugin found at **npmjs.com/package/ajv**.

10. Set up a scheme

Fastify schemes take the form of JSON objects containing descriptive information about the way the element should look. In the case of our simple API, we must verify two incoming parameters in regards to their type.

```
const schema = {
  params: {
    type: 'object',
    properties: {
      userId: { type: 'number' },
      aName: { type: 'string' }
    }
  }
}
```

11. Deploy the input scheme

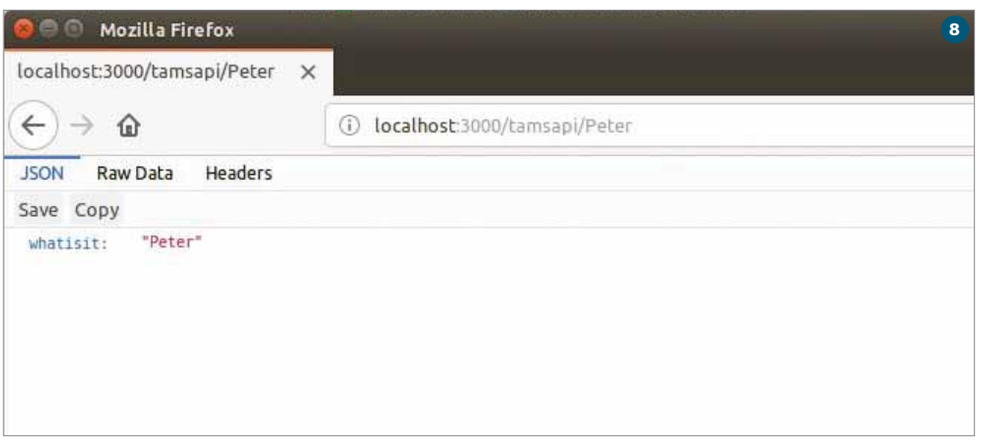
Creating the JSON object is just half of the trick. Verb methods such as the 'get' call used for registering a new route take an optional parameter of the 'schema' type. In principle, passing in the parameter could not be easier – the main issue involves the obligatory use of winged parentheses around it.

```
fastify.get('/tamsapi/:userId/:aName', { schema }, (request, reply) => {
        reply.send({ whatisit: request.params.userId })
})
```

12. Cause a verification problem

Now that the verifier is set up, invoke a non-complying URL such as http://localhost:3000/tamsapi/AAA/AAA. In addition to a '400' error, the error message returned also contains the string 'params.userId should be number'. This can be a problem for security-critical applications – telling your attacker more about the syntax of an unknown API tends to be a bad idea.

8



Support policies

Fastify releases will get at least six months of support after their initial launch. Further information on the policies can be found by visiting **fastify.io/docs/latest/LTS/** in a browser of choice.

13. Deep verification

Verifying URL schemes is but a subset of the capabilities of Fastify. The example accompanying this step shows how the body of an entire 'post' request can be checked to ensure the presence of a set of keys. Do bear in mind that body verification does not work with bodyless 'get' requests.

```
const opts = {
  schema: {
    body: {
      type: 'object',
      properties: {
        someKey: { type: 'string' },
        someOtherKey: { type: 'number' }
      }
    }
  }
}
```

14. Scheme on steroids

Serialising objects for transmission consumes immense amounts of processor time. Fastify's 'fast-json-stringify' significantly accelerates the process, thereby saving time when returning information to the consumer. Sadly, using 'fast-json-stringify' requires the declaration of an output schema.

```
const schema = {
  params: {
    . . .
  },
  response: {
    200: {
      type: 'object',
      properties: {
        whatisit: { type: 'string' },
      }
    }
  } }
```

	Version	Router	Requests/s	Latency	Throughput/Mb
bare	8.11.2	✗	30156.8	3.23	4.36
connect-router	1.3.2	✓	32603.8	2.98	4.62
connect	3.6.6	✗	38119.2	2.54	5.37
express-route-prefix	4.16.2	✓	18524.95	5.45	6.46
express-with-middlewares	4.16.2	✓	14403.4	6.83	5.21
express	4.16.2	✓	22385.2	4.38	3.50
fastify-big-json	1.5.0	✓	5273.8	18.83	60.69
fastify	1.5.0	✓	35912	2.7	5.62
hapi	17.5.0	✓	21789.2	4.53	3.42
koa-router	7.4.0	✓	23983.6	4.1	3.74
koa	2.5.0	✗	24987.2	3.92	3.89
micro	9.1.0	✗	35254.4	2.75	5.54
microrouter	3.1.1	✓	19473.6	5.04	3.03
polka	0.4.0	✓	36824	2.63	5.23
rayo	1.0.0	✓	35299.2	2.74	5.96
restify	7.1.0	✓	20796	4.36	3.30
spirit-router	0.5.0	✓	27592.4	3.2	4.34
spirit	0.6.1	✗	36883.2	2.39	5.78
take-five	1.3.5	✓	29164.8	3.04	9.74
total.js	2.0.30	✓	22850.8	4.29	6.51
trek-engine	1.0.5	✗	31837.2	3.06	4.58
trek-router	1.2.0	✓	28472.8	3.43	4.11

A question of microbenchmarks

German computing experts love to quote Klaus Gims' excellent work on microbenchmarks frequently – and one of his core messages was that getting reliable results is all but easy.

The Fastify team has endeared itself to developers by providing a series of benchmarks whose code can be inspected at **github.com/fastify/benchmarks** and shows excellent performance in both latency, throughput and request handling capability. Classic frameworks such as Express.js get pummelled, with lesser-known competitors faring better.

One thing that makes the data believable is that the Fastify team's own product is not at the very top – frameworks such as Spirit and Connect manage to be slightly faster by ditching the router infrastructure.

15. Go generic!

As response codes trigger via their numerical IDs, developers can associate them with more than one return code. As an example, look at the declaration accompanying this step – it handles all responses, which have a code in the 200 range.

```
const schema = {
  response: {
    '2xx': {
      type: 'object',
      properties: {
      . . .
      }
    },
```

16. Prevent information leakage

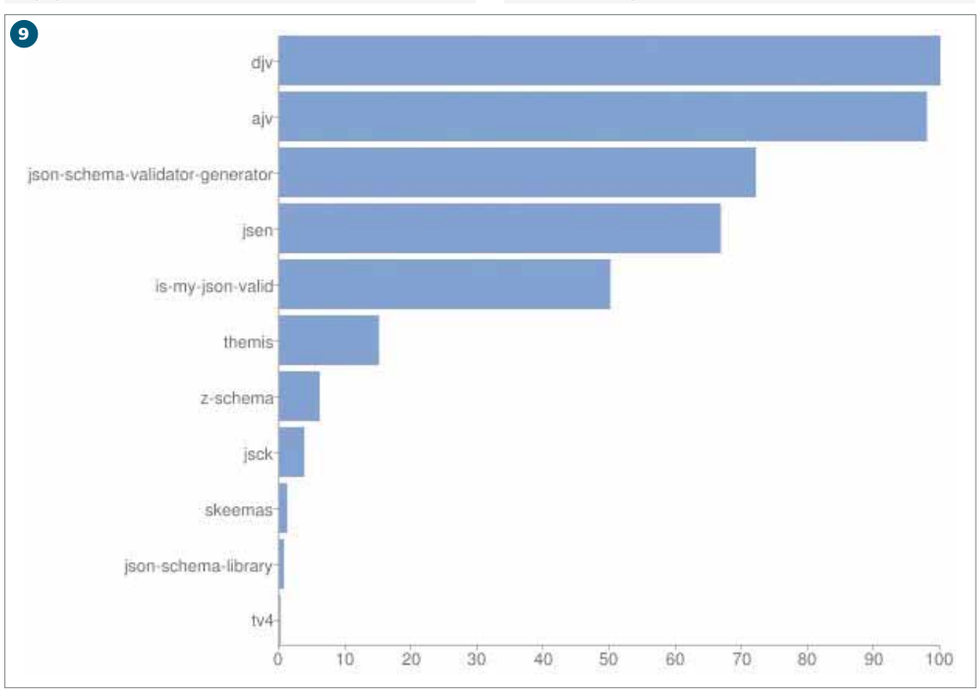
Setting up schemes does more than simply increase performance. Fastify discards all information not found in the declaration of the scheme, thereby preventing data leaks. A good example would be adding an additional parameter to 'reply.send' – as long as the route contains an output schema, the value of 'howmany' will not arrive at the client.

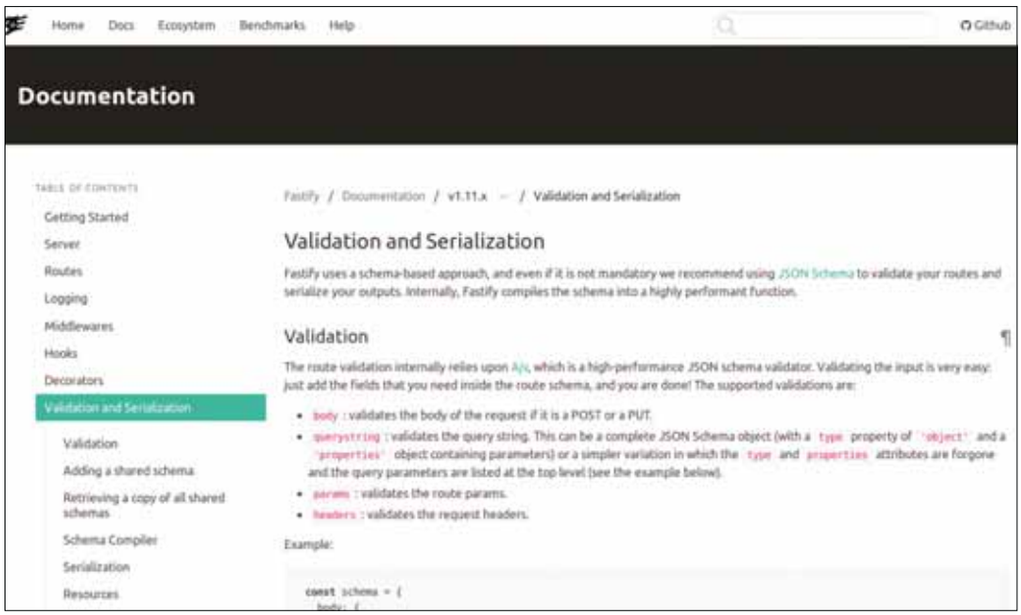
```
fastify.get('/tamsapi/:userId/:aName', {
schema }, (request, reply) => {
        reply.send({ whatisit: request.params.
userId,
                        howmany:"202" })
})
```

17. Enable logging

Mistakenly enabling debugger functions is a common anti-pattern, leading to reduced performance. Fastify tackles this problem by forcing developers to enable the logger in an act of will during program initialisation – you can not turn it on once the startup process of the framework is complete.

```
const fastify = require('fastify')({
  logger: true
})
```





Deep verification

As shown in the documentation screenshot accompanying this boxout, Fastify can apply a total of four different verifiers to incoming requests. In addition to the parameter and the body checks shown in the tutorial, validation can be performed against the entire query string or the contents of the request header – activating multiple matchers at the same time is permitted too. The somewhat unwieldy schemes can also be registered with the framework via its addSchema method – once a verification template has its ID, you can add it to multiple routes in a more compact fashion. Finally, keep in mind that you can change the settings of the schema compiler: simply use the setSchemaCompiler function to add a custom worker of choice.

18. Emit information

Google's introduction of LogCat changed the world of logging forever: instead of simply emitting a series of textual messages, users now expect 'ranking' information, colour-coding the messages. Fastify solves this problem by integrating Pino, thereby enabling developers to access various priority levels comfortably.

```
fastify.listen(3000, (err) => {
  if (err) {
    fastify.log.error(err)
    process.exit(1)
  }
  fastify.log.info(`server listening on
${fastify.server.address().port}`)
  fastify.log.error('Time to say Goodbye')
}
```

19. Install Pretty Printer...

By default, Pino's logging output is relatively bland. This problem can be addressed via Pretty Printer — it is a small command-line utility that takes in textual input and adds some formatting characters. The most comfortable way to use Pino involves installing it globally.

```
tamhan@tamhan-thinkpad:~$ sudo npm install -g
pino-pretty
[sudo] password for tamhan:
/usr/bin/pino-pretty -> /usr/lib/node_modules/
pino-pretty/bin.js
+ pino-pretty@2.2.0
```

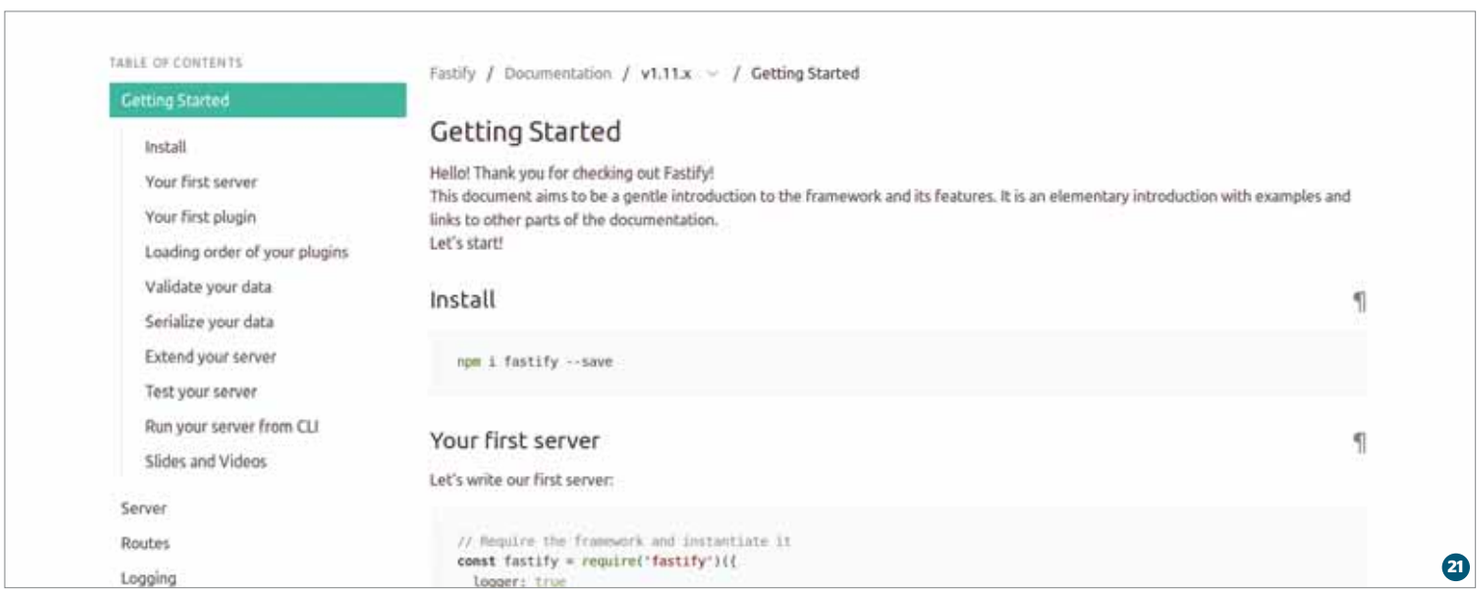
20. ...and beautify your logs

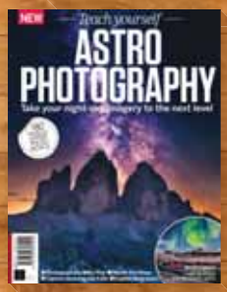
Once the global installation of 'pino-pretty' has succeeded, simply pipe the unformatted Node.js output into 'pino-pretty'. From that moment onward, logging messages will be colour-coded and will furthermore show up without the various JSON elements present in 'normal' output.

```
tamhan@tamhan-thinkpad:~/nodespace/
fastifytest$ node index | pino-pretty
[1537424752877] INFO (11100 on tamhan-
thinkpad): Server listening at
http://127.0.0.1:3000
[1537424752878] INFO (11100 on tamhan-
thinkpad): server listening on 3000
[1537424752878] ERROR (11100 on tamhan-
thinkpad): Time to say Goodbye
```

21. Learn more

Like most projects maintained by Matteo Collina, Fastify comes with excellent documentation. Visit **fastify.io/docs/latest/Getting-Started/** on a PC or a similar device with a wide screen to make the table of contents show up — and dig in as deep as you feel like it.





Discover another of our great bookazines

From science and history to technology and crafts, there are dozens of Future bookazines to suit all tastes

Get great savings when you buy direct from us

1000s of great titles, many not available anywhere else

World-wide delivery and super-safe ordering

FUTURE

www.myfavouritemagazines.co.uk

Magazines, back issues & bookazines.

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 MANAGED HOSTING — SUPREME HOSTING. SUPREME SUPPORT.

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

Agency hosting specialist

STORM

nimbushosting.co.uk
02031266781

Nimbus Hosting have partnered with agencies to develop our revolutionary platform STORM. With a team dedicated to outstanding support our 5 star Google reviews truly speak for themselves. Join the thousands of agencies and freelancers who are benefitting from a control panel that speeds up your website developement as well as your client's websites. Super charge your digital projects today with STORM.

- 30 second WordPress install
- Deploy directly from GitHub
- Easy team management

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
- 75 minutes of expert PHP video courses from Killersites (**shop.killervideostore.com**)
- 20 Vibrancy Photoshop actions and 100 essential brush strokes from Sparklestock (**www.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 device using any popular browser. However, we recommend that you use a computer to download content, as you may not be able to download files to other devices.

03 If you have any problems with accessing content on FileSilo, take a look at the FAQs online or email our team at the address below:

filesilohelp@futurenet.com

An incredible gift for subscribers

Subscribe today & unlock the free gifts from more than 60 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

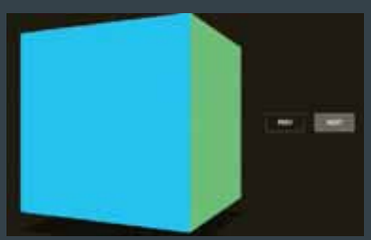
Let the experts teach you to create and code

More than 400 tutorials

Get the code you need to get creative

Over 250 creative assets

Templates, fonts, textures and backgrounds



Head to page 34 to subscribe now

Already a print subscriber?
Here's how to unlock FileSilo today…

Unlock the entire Web Designer FileSilo library with your unique Web ID – the 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

DESIGN, DEVELOP AND CREATE WITH GOOGLE

Discover the best pro tools, frameworks and libraries to power up your projects

UNIQUE LAYOUTS WITH CSS SHAPES
Introduce circles, squares and more to build print-style layouts for the web

GO INTERNATIONAL WITH ANGULAR
Learn how to make applications available and user-friendly to a worldwide audience

GSAP ANIMATION MASTERCLASS
Discover easing, control animations, work with timelines and extend with plugins

PRESS AHEAD

WP Engine's digital experience platform drives your business forward faster. **wpengine.co.uk**

WP engine

Take delivery of **more** than just a computer...

Get so much more with **Flexi-Lease**

Business users only

HARDSOFT
ONE STOP COMPUTER LEASING

Authorised Reseller

Discover the benefits of leasing computers for business
at **www.hardsoft.co.uk** Email **info@hardsoft.co.uk**

9021

9000