NEW!
Download 3+ hours of free video from FileSilo
The Web Design Book
Create
Expert advice for CSS, HTML, jQuery & more
WORK HARD! BE NICE!
The complete guide to desktop and mobile web design

The Web Design Book

A web designer's job is as versatile as it is ever-changing. As new software hits the industry and new mobile platforms gain popularity, a web designer needs to consistently up his or her game and take on new challenges to brush up their skills. Whether it's a new language or design element, no update can remain unexplored. That's why in the new Web Design Book we provide you with not only the tutorials that will put your skills to the test, but we also showcase the amazing ways they can be put to use. Whether it's HTML, CSS, jQuery or good old JavaScript, you will find your vice within these pages and discover new ways to use them. We also check out the latest in web development and mobile apps, before taking you on a tour through the world's most renowned design studios.

The Web Design Book

Imagine Publishing Ltd
Richmond House
33 Richmond Hill
Bournemouth
Dorset BH2 6EZ
☎ +44 (0) 1202 586200

Website: www.imagine-publishing.co.uk
Twitter: @Books_Imagine
Facebook: www.facebook.com/ImagineBookazines

Publishing Director
Aaron Asadi

Head of Design
Ross Andrews

Production Editor
Sanne de Boer

Senior Art Editor
Greg Whitaker

Designer
Abbi Castle

Printed by
William Gibbons, 26 Planetary Road, Willenhall, West Midlands, WV13 3XT

Distributed in the UK, Eire & the Rest of the World by:
Marketforce, Blue Fin Building, 110 Southwark Street, London, SE1 0SU
Tel 0203 148 3300 www.marketforce.co.uk

Distributed in Australia by:
Network Services (a division of Bauer Media Group), Level 21 Civic Tower, 66-68 Goulburn Street,
Sydney, New South Wales 2000, Australia Tel +61 2 8667 5288

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

The Web Design Book Volume 6 © 2016 Imagine Publishing Ltd

ISBN 978 1785 462 696

Part of the web designer™ bookazine series

IMAGINE PUBLISHING

Contents

HTML & CSS

018 25 pro plugins
028 Inspiration: Guggenheim Zero
032 3D experiences with PlayCanvas
036 Inspiration: brightly digital agency
040 Create a 3D navigation menu
044 Responsive image techniques
048 Inspiration: Namale

> Use 3D cards to present basic text content & links

jQuery & JavaScript

054 Be a jQuery code master
062 Image-based pop-up menu
064 Build your own jQuery plugin
070 Inspiration: WeGet design
074 Manage asynchronous tasks
080 Robust JS code with TypeScript

Development

088 15 Google developer secrets
094 Get free web hosting with GitHub

8 Your best CSS ever

098 Make a resizable sliding panel
100 Inspiration: Project Schoolkrant
104 Split-screen sliding panel effects

Apps

108 Testing, testing
114 Animate with TweenMax
118 Build with PhoneGap
124 Desktop apps with NW.js and JS
130 Create a mobile 3D interface (P1)
134 Create a mobile 3D interface (P2)
138 Develop a web app with Lucee
144 Use Ionic to create a native app
150 Desktop apps with Electron

Studio

156 Ice & Sky
162 DPDK
168 Cartelle
174 Phoenix the Creative Studio
180 All of Us
186 Amaze

YOUR BEST CSS EVER

THE TOP TOOLS, ESSENTIAL TECHNIQUES AND THE BEST ADVICE FOR STYLING PERFECTION

Analyse your CSS

ESSENTIAL TOOLS & TECHNIQUES TO GET PERFECT CODE EVERY TIME

CSSLinting is just like any other type of linting, in that it is "a tool that flags suspicious usage in software written in any computer language" (from the Wikipedia page on programmatic linting). Front-end developers are likely aware of JSONLint and JSLint. Well now there's CSSLint, which analyses CSS and helps developers write CSS that conforms to a set of performance and syntax of best practices.

'Helps' is understating it really, CSSLint throws errors and if used as part of a build server – say, TeamCity for example – it will cause builds to fail.

This is both a good and a bad thing. It means the rules are enforced with harsh penalties, but bad rules will create an upsetting development environment.

That's the rub really – some of the out-of-the-box defaults of CSSLint are controversial, so it pays the project team to take time customising the rule set. Here are some of the ones that draw controversy for many.

FIVE RULES TO FOLLOW:

Don't use IDs

It's not that there is anything inherently wrong with IDs, just that the Object Oriented ideology CSSLint was born of rejects them. OO-CSS says all CSS should be reuseable, so the concept of writing an ID into a CSS selector is very foreign. OO-CSS takes the stance that there should be no unique CSS.

For the everyday project team, this is likely an extreme view, and it isn't relevant if OO-CSS methodology isn't being employed.

Style headings once, globally

What CSSLint is getting at here is when developers write styles for heading elements and then redefine those with more specificity later in the stylesheet, heading styles should be declared once and not need to change.

Again, this is a bit extreme for the everyday project team. It's likely that the complexity and number of heading styles that can be seen on any website is defined by the design team.

Don't use too many floats

Lots of floated elements on an individual page cause longer paint times because of the extra layout calculations that browsers have to go through. However, unless you organise your CSS on a per-page basis, this rule is completely arbitrary.

Don't use adjoining classes

This is blacklisted because of browser support. IE6 doesn't support selectors that look like this:

```
class-one.class-two {
/* … */
}
```

Turn this rule off if the project offers legacy support towards IE6 users.

Better code hygiene

All of these bad defaults make CSSLint look like a bad tool. It really isn't – there are real advantages to rolling this tool into a project team's build processes.

CSSLint enforces syntax and code standardisation across the project team on a project-by-project basis. This code hygiene should make it easier to developers working on the project and make the codebase easier to work with as a whole. Accessibility can be enforced with styles like outline:none being outlawed. CSSLint also enforces the bulletproof font-face syntax – the best way to implement @font-face.

Style secret

CSSLint has a Wiki over at GitHub that details all of the default rule options. Each rule has its own page which goes into depth of what the rule covers and the idealogy behind it. Check it out at **github.com/CSSLint/csslint/wiki/Rules**.

To follow

CSS-Tricks
@Real_CSS_Tricks
The official account for **CSS-Tricks.com**, a web design community curated by CSS legend @chriscoyier.

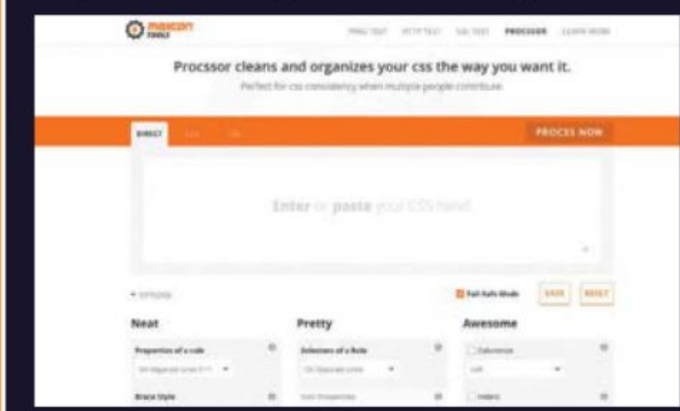
> *"CSSLint enforces syntax and code standardisation across the project team on a project-by-project basis"*

5 top tools

GET YOUR CSS IN TIP-TOP SHAPE

CSS Beautifier
html.fwpolice.com/css
A quick and simple option for creating easy-to-read code.

ProCSSor
tools.maxcdn.com/procssor
Helps clean and organise CSS the way you want it.



W3C CSS Validation service
jigsaw.w3.org/css-validator
Built by the community with three options for validating code.

Format CSS code
cssportal.com/format-css/index.php
Pick your presets, add code and format to get the code you want and need.

CSScomb
csscomb.com
Build your own configuration to get the exact code you want.

Bring design and build together

CSS PREPROCESSORS DON'T JUST BEEF UP YOUR CSS WORKFLOW, THEY CAN BE USED TO UNITE DESIGN AND BUILD TEAMS!

At this point it's fair to say that CSS preprocessors have captured the imagination of the industry and have become prevalent in production – the most well known being: Sass and SCSS, Less and Stylus. At this point in time there isn't much to tell these preprocessors apart. They all have fairly similar capabilities, just with each going about it a different way and with a slightly differing syntax.

The basic premise of CSS preprocessing is to provide more powerful tooling to developers and also enable more expressive code. This has its pros and cons – the more advanced the tool is, the more skill it will take to use it effectively. In line with this, some of the most basic and easiest to use features of CSS preprocessing are also the most powerful.

These basic features can be used to bring design and build teams closer together, creating codebases that are more flexible, easier to understand and, on the whole, much better to work with.

Organise components

Split your components into folders. Widgets belong in a widgets folder – same for pods, components (think header, footer and so on) and global modules (grid systems and the like).

Name colours

Design teams are usually in possession of brand guidelines from the client or have created their own by themselves. In it, the colours are probably named so simply name variables after these colours and maintain them in a file that is specific to the purpose. We do this because it will make it easy to ensure colours are consistent, and this is so that even if the codebase is very large, designers will be able to look at the preprocessor CSS and immediately understand the colour scheme of a component. They'll easily be able to tweak colours of concepts at later stages in the project, which will no longer create headaches for the build team.

Name font stacks

Most websites have more than one font stack and they're often used in combination with each other, for example headings and body text.

Monitoring the number of different stacks in regular CSS can become a bit of a chore. Find and replace can also be a bit haphazard should two similar, but different, font stacks contain the same font family.

Naming font stacks has similar benefits to naming colours. It enforces consistency and makes recognising incorrect font assignment easier for both designers as well as developers.

Componentisation

Take advantage of a preprocessor and split the styles up in a logical manner that helps introduce and describe the codebase. No longer does the 'biography pod' (contains a person's biography) have to sit in the middle of a massive stylesheet.

At a glance, the build team and designers alike should be able to see all the components in the website. This will give everyone a feel for the complexity of the site and should make code reuse and time estimation easier.

Flexibility

CSS preprocessors allow you to craft a more flexible codebase. Within a project, changes introduced late shouldn't cause as many headaches for the build team as before. Within a team, resources should be easier to scale with the demands of a sprint or iteration because the separation of concerns is more apparent. Within a company, the front-end discipline should become one that can deliver realistic prototypes with the design team without having a negative impact to the project timeline.

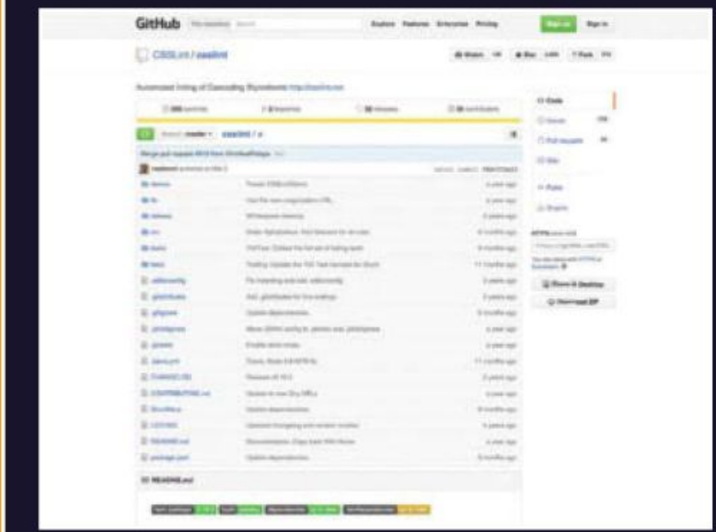
Points to consider

- Maintaining components in separate files so they are ready to be combined in different ways.
- Sensible use of variables, enabling for changes to be made later on in projects without costing time, for example accessibility issues caused by poor colour contrast can be easily remedied.

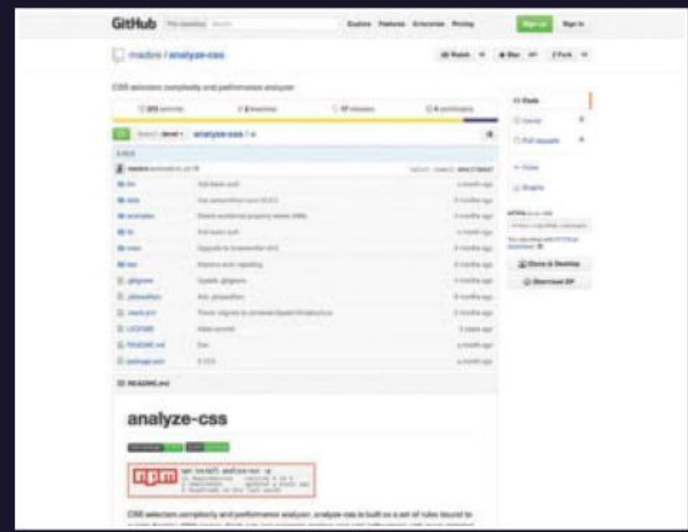
Style secret

With Sass, treat @extend like classes in other languages. It should be immediately clear that a style inherits other rules from somewhere else, so list your @ extends first. This also helps with overriding styles later.

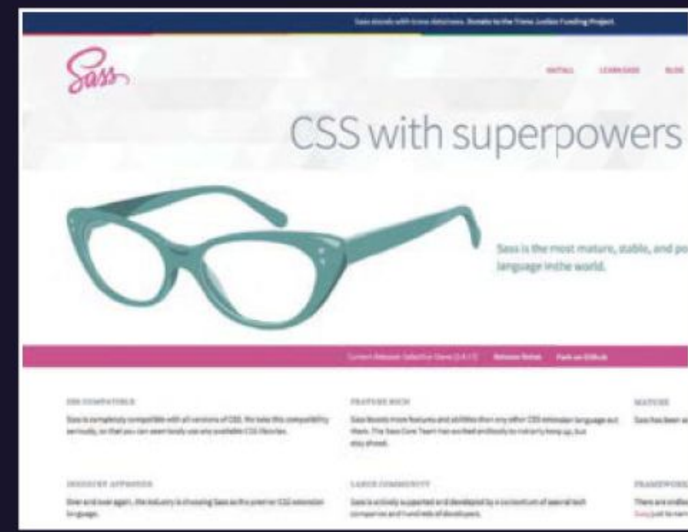
6 tools WRITE BETTER CSS TODAY



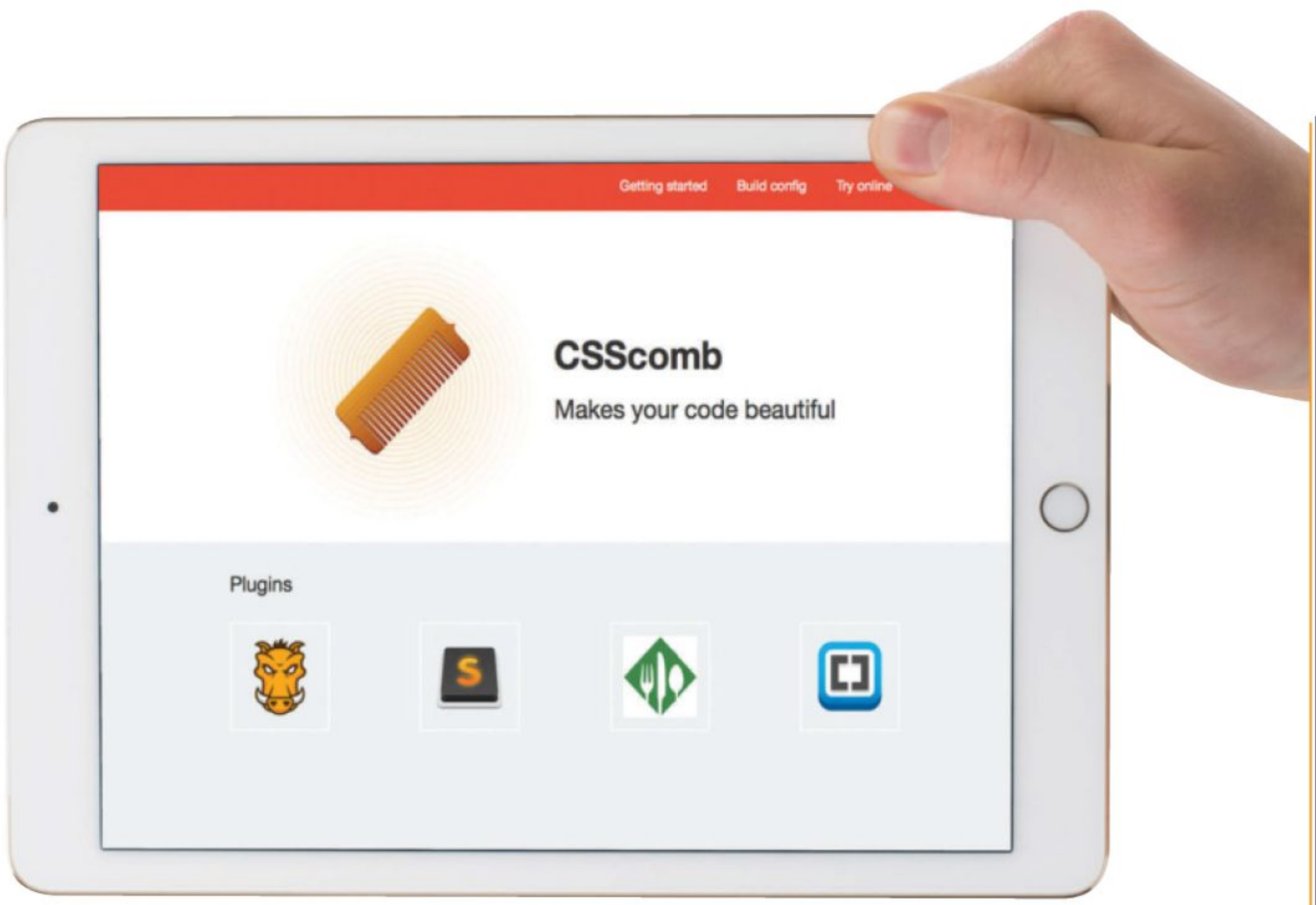
CSSLint
bit.ly/1IfwYPn
CSSLint is a tool to help point out problems with your CSS code.



analyze-css
bit.ly/1MMTh52
Analyze-css generates reports on selector complexity and performance of stylesheets.



Sass
bit.ly/1nTLuD5
Sass is the most stable and powerful professional-grade CSS extension language in the world.



- The ability to retire code quickly and easily when crafting components. If the client doesn't need a specific component anymore, finding and deleting it is now easy.
- Architecting preprocessor files to allow for variations of the website to be easily generated. This can be easily achieved by importing selected components into new parent preprocessor files.
- Using a preprocessor to quickly make colour scheme/typography changes site-wide, in other words prototyping. No longer does the design team have to slave away producing variations in Photoshop.

The advance of CSS tooling

Perhaps the single biggest benefit of using one of the big CSS preprocessors is the tools that feed into them. There is a vast range of workflow tooling that can be used to analyse performance, automate tedious tasks and hint at code improvement – among other things.

It must be said that the most valuable of these aren't the 'run once then throw away the results' type, but instead tools that run on 'build'. A build team that has a build server, which creates release builds automatically, could benefit from having a performance report generated for example.

To follow

CSS Author
@cssauthor
A web design and development blog that tweets the best resources and tools for designers and developers alike

Beware of bad code

WHAT TO AVOID AND WHY

Excessive nesting

CSS preprocessors make it very easy to write long selectors by enabling nested code. Long selectors come at a performance cost, so consider that when nesting code.

Vendor prefixing

Most preprocessors have options or third-party middleware that will add vendor prefixes automatically. Double-check the output of this behaviour – old prefixes for tools like Flexbox can cause weird behaviours.

Length of output

When dealing with many small files it's easy for a build team to create a rapidly growing 'main' stylesheet. Look at the output regularly and minify as part of a build process.

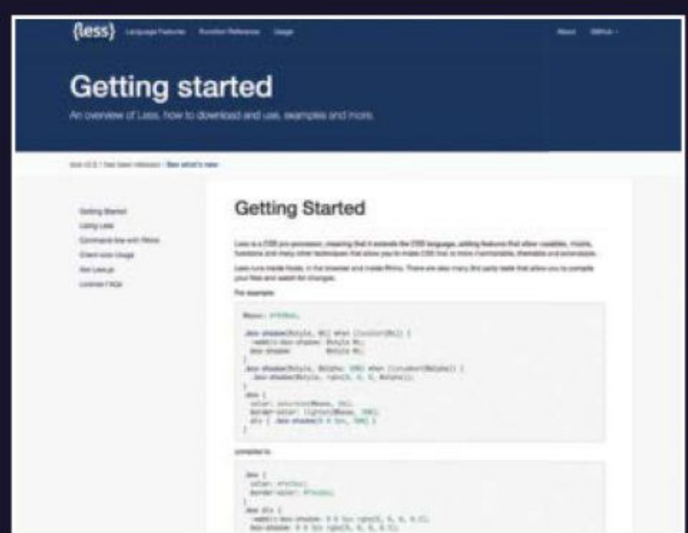
Build errors

Developers not previously exposed to preprocessors may be horrified to find that writing erroneous syntax now results in build errors. But this is still better than weird browser behaviour, however.

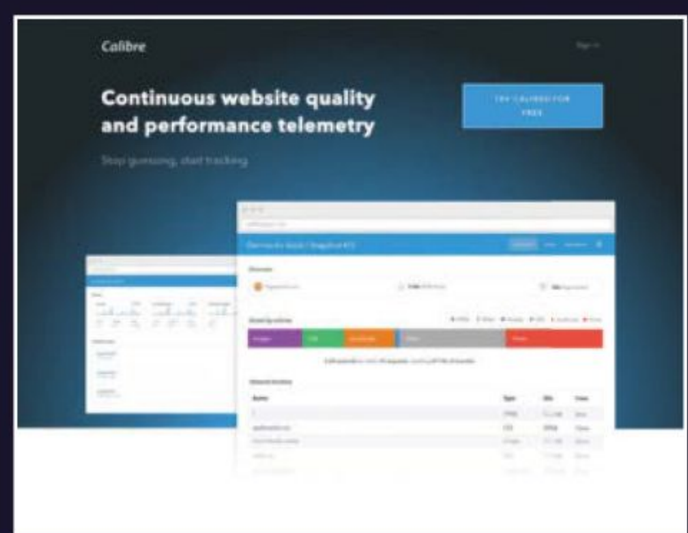
Lack of comments

It's rare in vanilla stylesheets to need comments, so preprocessors present a bit of a mindset change when it comes to functions and mixins. Good comments can explain what the developer is thinking.

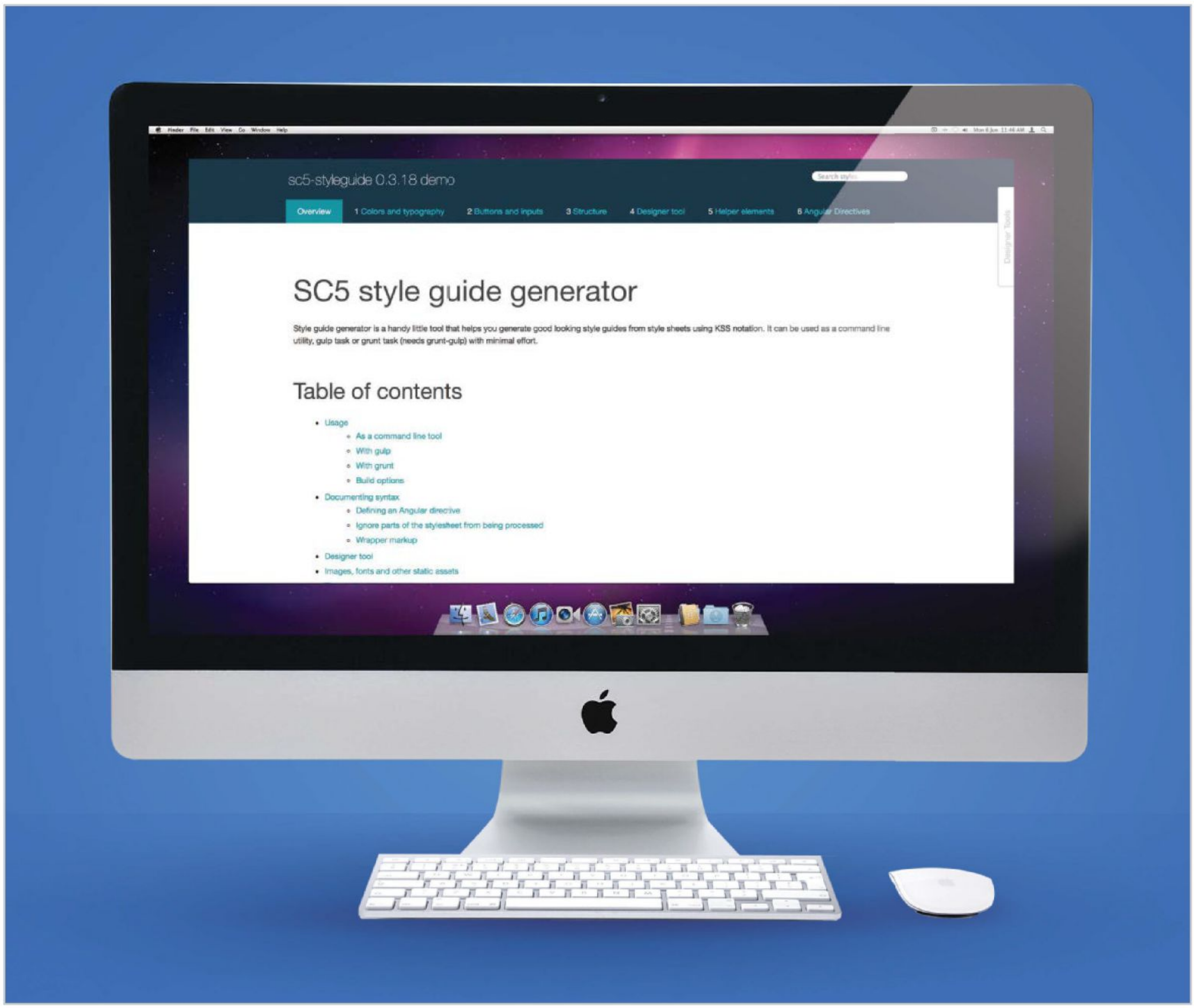
Stylus
bit.ly/1paZXwg
Stylus is a revolutionary new language, providing an efficient and expressive way to generate CSS.



{less}
lesscss.org
Less runs inside Node, in the browser and inside Rhino as well.



Calibre
calibreapp.com
Continuous website quality and performance telemetry. Think Google Analytics for the frontend.



Style guides: why you need them

HOW TO BUILD WORKING DOCUMENTS FOR CONSISTENT DESIGNS

Style guides are a brilliant idea, however they are usually woefully executed in practice. The idea is that at any point in time there is a working document visible to the project team that captures the look and feel of core components – and this includes typography, buttons and form fields. What actually happens though is that the working document becomes stale and discarded by most of the project team, and that's because it captures the look and feel of the project from two months ago.

An out-of-date working document is pretty useless because the project team can't use it for anything they're working on at that moment – it simply gathers dust, becoming a relic.

There are probably many reasons for stale style guides, but the biggest standout is the associated maintenance cost. The cost to design and build a style guide once is mostly absorbed, as it should mirror the initial stages of the project. It's only when it takes extra time to keep this document in sync with the working project that it gets bumped in favour of any higher priority tasks.

The style guide idea is brilliant so long as the style guide maintains itself after the initial incarnation. Thankfully, modern tooling can help improve this situation somewhat.

Build tasks

Create a build task to distribute postprocessed files to all relevant locations instead of just to the website. This

should ensure that the style guide is up to date with any CSS or JavaScript as it's committed to the project.

Team visible

Make the style guide available internally by hosting it on an internally available URL. This gives everyone within the project an easily accessible and now up-to-date aesthetic reference.

It's also worth considering authoring a 'making contributions' document, which provides guidelines on how to contribute to the style guide project, all without actually being a developer.

Seek design team contributions

Once a style guide is up to date and team visible, it's time to turn the style guide into the working document it was born to be. It's time to ask the design team to learn some code... sort of.

Following on from the CSS preprocessors section, the advice was to create a 'colours.scss' or similar. This file should contain all of the colour definitions used within a project and be trivial to update.

It's worth giving training on this process and basics on the preprocessor to the project team. This will enable the design team to have input on the look and feel of the site through the actual code, rather than being one step removed with concepts.

The colours example is very easy, but this could be applied to typography, buttons, form inputs and many of the other simpler aspects of build. The limit here is going to be the design team's knowledge of CSS, but fortunately though, these are the easiest aspects of CSS that a person can pick up.

The design team picks up basic aesthetic changes, giving more control over the result of the build. They're also free to adjust margin and padding on elements.

This should then free up the build team to focus on more fundamental and specialist problems, such as box model layout, code architecture and also flexboxes.

> *"A 'making contributions' document provides guidelines on how to contribute to the style guide project"*

Style secret

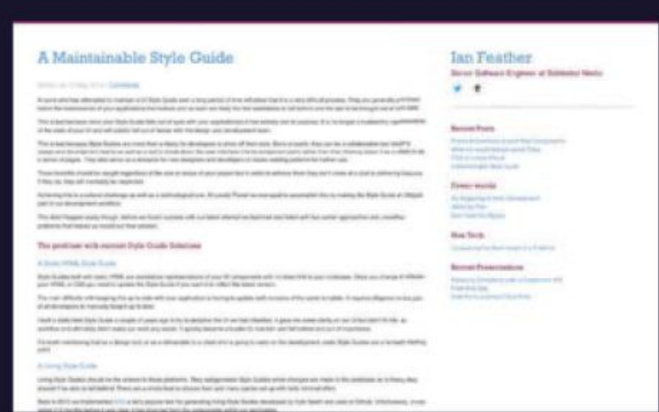
Consider choosing a styleguide generator that has a GUI which allows for non-technical users to make adjustments without having to write code. Sc5- styleguide has a demo for just that via **demo.styleguide.sc5.io**.

To follow

CSS3
@CSS3
An account that is dedicated to CSS. It offer news, previews, tutorials and more for the discerning designer.

Get better style guides

A Maintainable Style Guide
bit.ly/1glFg13
Craft a maintainable style guide, using it as a tool.

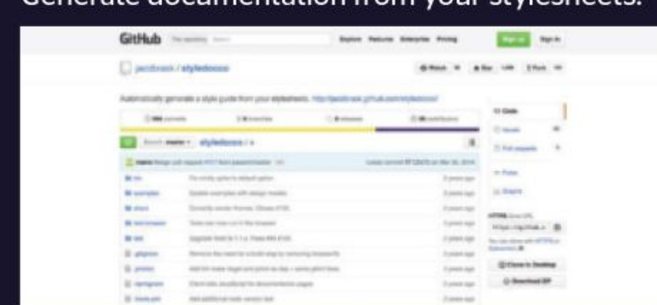


Style Guide Generator Roundup
bit.ly/1zKLmQm
A look at style guides that run on platforms.

gulp-kss
bit.ly/1LA8BDy
A gulp extension to help create of a style guide.

sassdown
bit.ly/1JiMITr
Build living style guides with Handlebars.

StyleDocco
bit.ly/1NAY3Ep
Generate documentation from your stylesheets.



5 Photoshop extensions

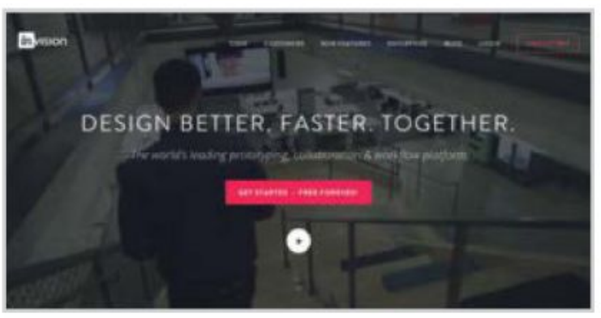
HOW TO ADD MORE POWER FOR FRONT-END DEVELOPERS

CSS Hat
csshat.com

Including any plugin that "writes code for the developer" is always going to be controversial, but CSS Hat does an okay job. When it comes to trying to scratch something together in minutes or unpicking several font styles at once – it can be a handy pallette to have, especially for those who don't have a mental CSS encyclopedia.

render.ly
render.ly

If something needs exporting, **render.ly** is the right tool for the job. The basic premise is that each top-level group in a PSD is a 'screen'. and child groups with names beginning with "&" get exported as separate states of that screen. **Render.ly** also generates webfonts for icons in vector format. No more manual image exporting from PSDs.



InVision
invisionapp.com

InVision takes designs and provides an interface for adding interactions – enabling designers to craft interactive prototypes complete with gestures, transitions, and animations. InVision also provides real-time presentation, collaboration and project management features. It's a cheaper route to designing effective UIs.

Renamy
klaia.com/Renamy

Renamy is a layer renaming power tool. It allows for renaming of multiple layers the right way. It can find and replace and even handles regular expressions. Those who grew up with CS2 would be jealous of this tool. An alternative use is to rename a colleague's PSD layers from 'New Layer 102' to 'Old Layer 102' just to upset them.

Ink
bit.ly/1KOSPIO

Ink is a free Photoshop plugin that generates documentation with detailed information on Photoshop documents and their layers/styles. For typefaces this is information on font family, weight, leading, size and colour. For images and other layer assets, dimensions are output. It spots minor inconsistencies at a glance.

Built-in function

There are lots of Photoshop extensions that do amazing things, but Photoshop also comes with a built-in batch file processing function (File>Automate>Batch). Now, tasks like image resizing become kind of trivial.

Improve performance

TAKE THE PAIN OUT OF CSS SPRITES. AUTOMATE WITH... CSS-SPRITE!

CSS sprites can really help cut down on the number of HTTP requests made on each page load. When done effectively, sprites can have a big impact on performance, especially on mobile, where the number of simultaneous connections is limited or perhaps the data connection is just plain slow.

Unfortunately, they're also quite costly to create and maintain when done manually – especially if icons need to change sizes late in the build. This is where css-sprite (an npm package) comes in: give it a bunch of images and it returns a stylesheet (in your preprocessor of choice) and a single image.

What is particularly good about css-sprite is that it is capable of laying the sprites out in multiple ways. Vertical and horizontal are options, but the best is binary-tree. This will find the most optimal layout for all the sprites, creating a smaller final image – though this is slower because more calculation is required. This is unlikely to affect most everyday projects however, and for those it does it is unlikely for the file size to need optimising.

The best thing about automating this process is that the design team doesn't have to commit to final versions of icons until later in the project. They're free to change dimensions in response to client/user feedback.

> *"What is particularly good about css-sprite is that it is capable of laying the sprites out in multiple ways"*

Sprites that have been generated from that gulp task

Style secret

File size still matters for CSS sprites even when the process for creating them is automated. It's worth adding a compression task to your workflow for the final output. Also consider grouping images into different sprites where it makes sense.

Complete your CSS toolkit

TEN MORE TOOLS THAT WILL MAKE YOUR CODE BETTER THAN EVER

SpaceBase
spacebase.space150.com
A Sass-based CSS framework that provides the launchpad for builds. Boasts a boilerplate layer.

Skyline
skyline.is

Critical Path CSS generator
bit.ly/1ygfAsw

Editr
lab.idered.pl/editr

DeSVG
benhowdle.im/deSVG

Pleeease
pleeease.io

Reuze
derny.github.io/reuze

CSS MenuMaker
cssmenumaker.com

CSS Sans
bit.ly/1Fn86Ys

CSSColorEditor
bit.ly/1NXTP9I

HTML & CSS

018 25 pro plugins
Discover the vital plugins that will transform your future projects

028 Inspiration: Guggenheim Zero
Learn from design studio Hello Monday and make your own CSS shapes

032 3D experiences with PlayCanvas
This 3D HTML5 game engine produces beautiful interactive experiences

036 Inspiration: brightly digital agency
Get inspired and recreate brightly's digital Swiss pocket knife

> Ever since responsive web design became a thing, developers have wanted better tools to manage imagery

040 Create a 3D navigation menu

Use HTML to build a navigation menu that uses 3D page elements

044 Responsive image techniques

Imagery and responsive web design don't always mix, but here's a way to work around it

048 Inspiration: Namale - Joyeux Uniques

A brochure site with a twist that can teach you all about timing your animations

25 PRO PLUGINS

DISCOVER THE TOP JQUERY ADD-ONS FOR MAKING YOUR WEBSITES AND APPS A BETTER EXPERIENCE

Our expert

Tam Hanna works with Symbian software products and a variety of coding languages. This issue he reveals the best plugins for powering up jQuery

TWEAK PHOTOS

Plugin: Image Editor
bit.ly/1hIv6c1

Category: Photo editing

Users love to edit uploaded pictures on the web. Image Editor provides a svelte little GUI wrapper around the Camanjs filter library: it saves you from the hassle of coding a set of sliders, buttons and symbols which come together to make up a little editor.

Getting started with the Image Editor plugin requires the embedding of four libraries and a picture box – our test harness starts out like this:

```
<head>
<script src="dependencies/jquery.min.js"></
script>
<script src="dependencies/jquery-ui.js"></
script>
<script src="dependencies/bootstrap.min.js"></
script>
<script src="dependencies/caman.
full.js"></script>
<script src="js/boostrap-image-
editor.min.js"></script>
<link href="dependencies/
jquery-ui.min.css"
rel="stylesheet">
<link href="dependencies/css/
bootstrap.css"
rel="stylesheet">
<link href="css/boostrap-image-
editor.css" rel="stylesheet">
</head>
```

Keep in mind that the download is not complete and that's because each of the libraries also expects its resources to be in place for the GUI to work. This can be accomplished by extracting the individual archive's contents into the /dependencies folder.

Image Editor does its magic in a tag, which is populated at runtime via the imageEditor function. Our example loads the editing tools during page load – in practical applications, starting the editor on demand might be more efficient:

```
<body>
<div id="editor-window"></div>
<script type="text/JavaScript">
$("#editor-window").imageEditor({
'source': '/img/demo.jpg',
"maxWidth": 500,
"onClose": function () {
}
});
</script>
</body>
```

OnClose is invoked if the user clicks the close button below the image: you can use the various methods found in Canvas in order to obtain the results of the editing operation. Image Editor's Save button compresses the picture into a file which is then served via the download utility.

Other initialisation options include maxHeight, for the maximum height of the image; maxWidth, for the maximum width of the image; remoteSave, where the default save method is browser; remoteURL, for where you want to send the image to; blockEnable, which should only be enabled if jQuery BLOCKUI is enabled; blockMessage, a message that will be displayed in the blockUI; and saveCallBack, the callback after the save event.

The actual editing is handled by the plugin – no callbacks are provided to inform you about the individual steps taken. Image Editor should not be used to process large files: most browsers pop up an alert dialog about unresponsive scripts if an operation takes too long. In practice, images smaller than XGA tend to work well – keep in mind that Firefox is not multithreaded, which leads to resource competition between tabs.

Required dependencies

Image Editor will need these libraries: the latest jQuery, jQuery UI, Camanjs and Bootstrap. Camanjs is very easy to extend, with a constantly growing number of image editing capabilities.

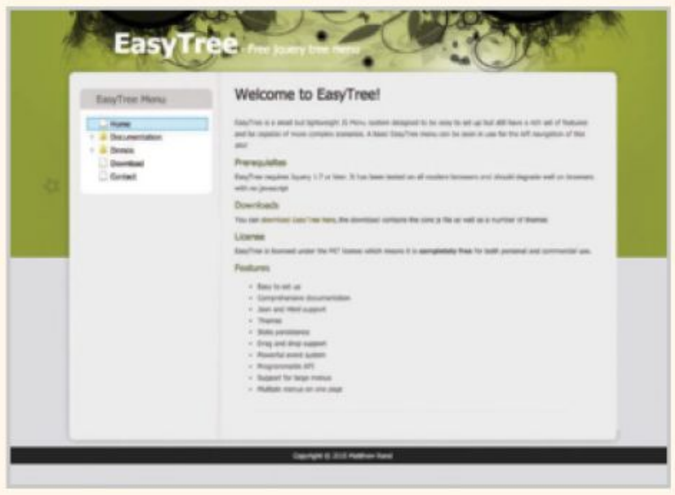
OTHER AUTHOR PLUGINS:
Ritchy Chen
rithychhen.com

Plugin 1: Scrollbar Indicator
bit.ly/1LxgHeR
Plugin 2: Birthday Picker
bit.ly/1hIvnvy

> *"Image Editor does its magic in a tag, which is populated at runtime via the imageEditor function"*

Manage UIs
bit.ly/1GK1Ub7

Tree views provide a way to manage complex user interfaces: if options are grouped up, the UI appears less daunting at first glance. EasyJSTree's tree structures start out as a group of list tags, which are then spruced up by invoking the easytree() method on the mother element.



Pop-up dialogs
bit.ly/1MWNFGr

Palm OS graced developers with the popup: a complex modal dialog which could be invoked to get a result. Prompt21 obviously was inspired by the golden oldie. Invoke the popup after calling the var p = $(".popup").prompt21(); method on it, and feast your eyes on a JSON object containing the values entered in the dialog's widgets.

Tab bars
bit.ly/1OHWpjG

Tab bars are a classic: if there's little screen real estate, multiply it by introducing some tabbing. TurboTabs expects you to feed it with a structure of <ul> and elements alongside a configuration object that can be set up via a tool on the plugin's website. Your input will then be transformed into a svelte-looking and nicely animated tab bar.

Sticky cursors
bit.ly/1RMbaSs

Making a widget stick to the cursor is helpful when a drag-and-drop-like interface is to be implemented. ObeyCursor transforms this task into a no-brainer: simply configure a tag of choice, and configure it with a call to the obeyCursor method:

```
<script type="text/JavaScript">
$(function() {
$("#slave").obeyCursor(opts);
});
</script>
```

Our web app has been rendered in a little pop-up window. You can spruce up its looks by deploying your chosen GUI framework

Users are free to change the desktop background to an image as they see fit

Deskible's start menu will remind users of classic versions of Windows 95 to 7. The 'default' items in our screenshot can be removed if desired

A small clock located in the bottom-right corner of the screen can help your users to keep track of time

WEB WINDOWS MANAGER

Plugin: Deskible
bit.ly/1PwPvyu
Category: Window manager
Ever found yourself suffering from MDI envy while working on a web app? This proof of concept creates a relatively full-featured Window Manager based on your favourite web technologies. It helps developers to create and manage a number of dynamic windows inside their web applications.

First download Deskible from GitHub. This must be done via the command line to satisfy the dependencies. Next, open index.html for the correct operation. As of this writing, the log-in window is not implemented – just click Login to move into the desktop environment. Start menu content is configured by startMenu array content declared in app.js. One example looks like this:

```
startMenu: [{
icon: 'book',
label: 'Documentation',
sub: [{
icon: 'wrench',
id: 'docu-setup',
label: 'Setup and Layout',
options: {
tabs: false,
content: {
url: 'apps/documentation/setup.html'
} } }, {
```

Higher-level folders are created as simple array elements, while their subelement contains the actual content. Looking at /apps/examples/irc.html reveals that the IRC client consists of but the following line:

```
<iframe type="text/html" width="100%"
height="100%" src="https://kiwiirc.com/
client" frameborder="0"/>
```

Complex content is best created via an iframe: Deskible will load it as if it were found in a website, displaying content in the area decorated by the window manager's widgets. A custom installation of Deskible starts by removing unneeded elements: for most applications, a YouTube player and an IRC client are overkill. Proceed to an iFrame-hostable web apps in the apps folder:

```
<p>
<script>
function tmgnBtnClicked()
{alert ("Hello World!");}
</script>
R: <input type="text" name="r" ><br>
U: <input type="text" name="u" ><br>
I: <input type="text" name="i" ><br>
<button onclick="tmgnBtnClicked()">Calculate</
button>
</p>
```

Next, add the relevant entry to the start menu. Our resistance calculating widget will sit in Tamoggemon -> OhmCalc, accomplished via the following snippet:

```
startMenu: [{
icon: 'book',
label: 'Tamoggemon',
sub: [{
icon: 'wrench',
id: 'ohmcalc',
label: 'OhmCalc',
options: {
tabs: false,
content: {
url: 'apps/ohmcalc.html'
} } },]}
```

> *"A custom installation of Deskible starts by removing unneeded elements"*

Telephonic enforcer
bit.ly/1VVXt3V

Users hate sharing their phone number with random websites. This library contains a set of validation routines aimed at catching miscreants red-handed. Please keep in mind that MobilePhoneNumber does not check whether the user really owns the phone in question – it merely keeps honest people honest by not allowing them to enter malformed numbers.

Protect your images
bit.ly/1LdEGy9

Image theft happens more than you think. This little utility uses the HTML5 Canvas editing capabilities to stamp images with a watermark on the client's browser – a feature which can also be used to frame images, thereby strenghtening your website's corporate identity. As with all other client-side security systems, be aware that sophisticated attackers are able to work around it with the most minimal of effort.

MANAGE COOKIES

Plugin: CookieBar and js-cookie
bit.ly/1LdDLxR and bit.ly/1av6HUb
Category: Cookie management

EU regulations have made cookie management difficult: as Google recently decided to require EU compliance in its AdSense program, developers need to conform to the regulations. Fortunately, a combination of these two cookie plugins makes compliance and data handling really, really easy!

Achieving compliance in itself is really easy: cookieBar is shipped with a stylesheet that contains the logic for a basic cookie bar that is easily dismissable by the user. Best of all, you don't have to use any markup either! Start off by including the JS and the CSS file along with the main jQuery library:

```
<html>
<head>
<script src="jquery-2.1.4.js"></
script>
<script src="jquery.cookieBar.
js"></script>
<link rel="stylesheet"
type="text/css"
href="cookieBar.css">
</head>
```

Displaying the actual bar can then be accomplished by invoking the cookieBar() method. The plugin contains the necessary logic to keep state: the user will not see the bar twice if he does not delete the cookie:

```
<body>
<script type="text/JavaScript">
$(document).ready(function() {
$.cookieBar();
});
</script>
</body>
</html>
```

CookieBar provides a variety of methods to aid in the customisation process: you can replace the default text with one provided by your legal department, and can also modify the styling to make the cookie bar blend in better or stand out more.

In the next step, it is now time for us to actually persist some data. First of all, the cookies library must be included in the header:

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

JQuery's js-cookie plugin provides a bunch of helper methods for implementing a basic KV store – reading data from it can be accomplished like this:

```
$(document).ready(function() {
if(Cookies.get('imagineCookie')===undefined)
{
alert("No cookie there yet");
}
else
{
alert("Found: " + Cookies.
get('imagineCookie'));
}
});
```

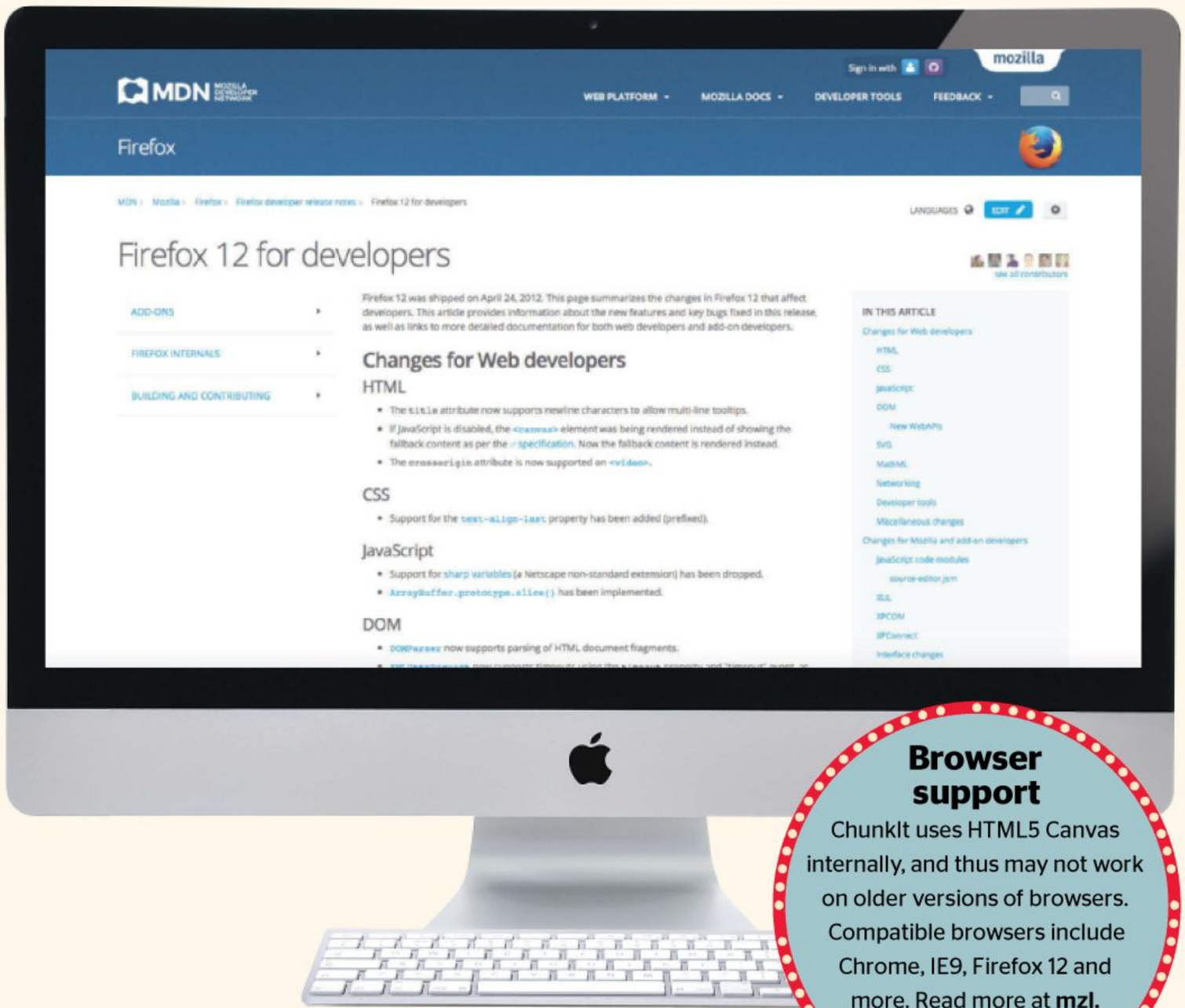
Js-Cookie
For some quickfire information on js-cookie, make sure you check out the Wiki **github.com/js-cookie/js-cookie**. It is written by one of the authors, Fagner Brack, and includes details on integration with Angular.

Saving data is then done via the set method. Cookies do not limit themselves to basic types: you can also persist JSON objects by passing them in.

```
$(document).ready(function() {
if(Cookies.get('imagineCookie')==
=undefined)
{
alert("No cookie there yet");
Cookies.set('imagineCookie', "Hello Cookie");
}
```

AUTHOR BOX:
Carl Woodhouse
github.com/carlwoodhouse
Klaus Hartl
github.com/carhartl
Fagner Brack
github.com/FagnerMartinsBrack

"JQuery's js-cookie plugin provides a bunch of helper methods for implementing a basic KV store"



Browser support

ChunkIt uses HTML5 Canvas internally, and thus may not work on older versions of browsers. Compatible browsers include Chrome, IE9, Firefox 12 and more. Read more at **mzl.la/1W8UMkX**.

BREAK AND MERGE IMAGES

Plugin: ChunkIt
bit.ly/1GK5ims
Category: Image manager

Breaking images down along rectangular patterns has always been popular – the Pet Shop Boys' teaser for *Love Etc* in 2009 brought the design pattern to public attention. This plugin takes an image, renders it into a Canvas and then chops it down into tiny pieces which you can arrange to create psychedelic wallpaper designs.

ChunkIt does its magic by creating a structure of li tags contained in a ul tag: each one of them contains an <img> tag which, in turn, contains its picture data as an inline image.

Since browsers decorate list tags with all kinds of objects by default, we start out by declaring an embedded stylesheet which aims to calm these tendencies.

```
<!DocType html>
<html>
<head>
<style>
ul.grid {
padding: 0px;
}
ul.grid li {
list-style: none;
display: block;
border: 0px solid #CCC;
float: left;
line-height: 0px;
}
</style>
```

In the next step, the actual instantiation of the plugin takes place. The invocation of chunkIt() does the actual work – expect the browser to stall for a bit of time:

```
<body>
<script src="jquery-2.1.4.js"></script>
<script src="jquery.chunkIt.js"></script>
<script>
$( document ).ready(function() {
$('#gridHouse').chunkIt({ imgURL : 'flowers.
jpg' , cellsInRow:5, cellsInColum:5,
shuffle:true});
});
</script>
```

The chunkIt() method takes a JSON object which configures the actual behaviour of the tiling engine. ImgURL is a URL to the image, while cellsInRow and CellsInColum decide the size of the result. Setting shuffle to false arranges the <li> tags in line – you can always perform a correlation between their ID and the part of the image contained within.

In our case, the structure generated looks like this:

```
<div id="gridHouse">
<ul class="grid" style="width: 1200px;">
<li id="cell_25" class="cell">
<img src="" style="width:
240px; height: auto;"></img>
</li>
<li id="cell_22" class="cell">
<img src=""
style="width: 240px; height: auto;"></img>
</li>
. . .
```

Finally, a tag must be erected. ChunkIt will use it as the 'mother tag' for the generated structure:

```
<div id="gridHouse"></div>
```

ChunkIt should not be combined with very large images: sizes that are any larger than about 1600 x 1200 leads to long processing times. This is especially harmful on single-threaded browser such as Firefox: the entire browser window shuts down when the tesselation process is running.

> *"The chunkIt() method takes a JSON object which configures the actual behaviour of the tiling engine"*

RESPONSIVE CHARTS

Plugin: chartist-js
bit.ly/1GhA775
Category: Data visualisation
Libraries that make diagrams come a dime a dozen: ultrasimple Canvas-based ones stand next to behemoths such as the fearfully complex D3.js. Chartist differs from its competitors in that it aims to combine the no-frills approach of Canvas-based renderers with the print-ready quality provided by SVG.

Even though Chartist can – in theory – also be used without jQuery, most developers will settle with an include structure similar to the following. At the time this issue went to press, the recommended CDN contained broken code – our example contains a copy of the code which works on Firefox:

```
<head>
<link rel="stylesheet" href="chartist.min.
css"/>
<script src="chartist.js"></script>
<script src="jquery-2.1.4.js"></script>
</head>
<body>
<div class="ct-chart"></div>
</body>
```

Our example contains a group of buttons which populate the tag with various diagrams. First up is a humble line chart generated by makeLineChart:

```
function makeLineChart()
{
new Chartist.Line('.ct-chart', {
labels:[1,2,3,4,5],
series: [
[12, 9, 7, 8, 5],
[2, 1, 3.5, 7, 3]
]
}, {
fullwidth:true

});
}
```

Chartist's diagram methods tend to take a pair of parameters: the first variable contains the data to be displayed, while the second variable can provide you with a variety of display settings.

Changing the display to create a pie chart doesn't require much extra effort:

```
function makePieChart()
{
var data = {
series: [20, 30, 55]
};
var sum = function(a, b) { return a + b };
new Chartist.Pie('.ct-chart', data, {
labelInterpolationFnc: function(value) {
return Math.round(value / data.series.
reduce(sum) * 100) + '%';
}
});
}
```

This example extends the diagram with custom labels created by an interpolation function. It uses data.series.reduce to obtain the sum of the entire dataset, which is then applied to a simple percentage calculation.

Developers can use the events exposed by Chartist to modify the look of the resulting diagram. The following version draws a significantly larger circle on top of each data point of our line chart:

```
function makeFancyChart()
{
    var chartRef=new Chartist.Line('.
      ct-chart', {
        . . .
      chartRef.on('draw', function
       (data)
       {if(data.type===="point"){
       data.group.append(new
       Chartist.Svg('circle', {
       cx: data.x,
       cy: data.y,

       r: 20
      }, 'ct-slice-pie'));
    }});
}
```

SMIL support
Chartist.js supports all kinds of animations: charts can be enriched with marching ants and similar niceities. Sadly, this requires support for SMIL – a feature sorely missing in all versions of Internet Explorer.

Draw gets emitted every time a new element is created: its type subattribute provides information about the type of element at hand.

Some features – axis labels are a common example – might be required in some jurisdictions like Austria, but are not considered important enough by the developers who pride themselves on the tiny 10KB total file size of their product.

OTHER AUTHOR PLUGINS:
Gion Kunz
github.com/gionkunz
Plugin 1: Accessibility for Chartist
bit.ly/1GhD6MG
Plugin 2: Point Labels for Chartist
bit.ly/1NfSeKe

To follow
John Resig
@jeresig
Two words: jQuery.daddy==John. Follow him to enrich your Twitter timeline with content more or less related to the big jQ!

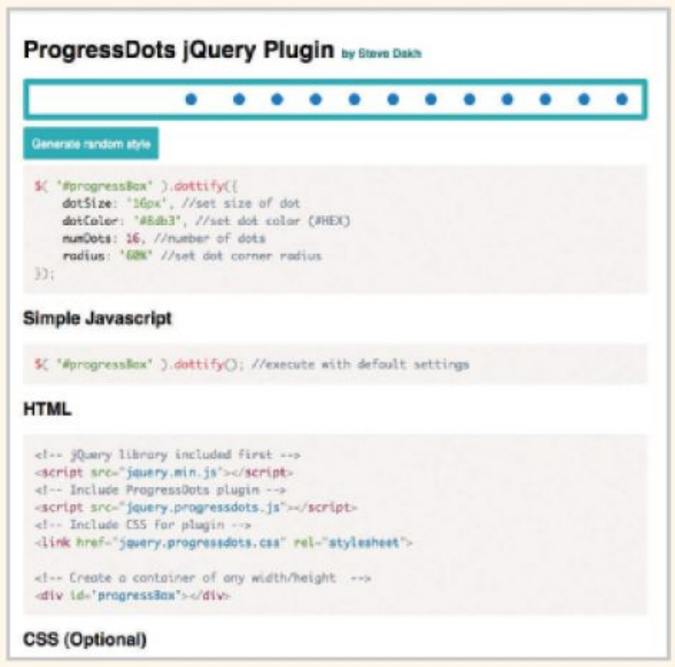
To follow
JQuery Foundation
@jquery
The jQuery Foundation is the body which governs the jQuery product family. Follow the account for all the latest, official announcements.

To follow
JQuery Plugins
@jqueryplugins_
This Twitter account provides a never-ending stream of interesting, weird and/or useful jQuery plugins for your perusal.

Progress dots
bit.ly/1jGTzzS

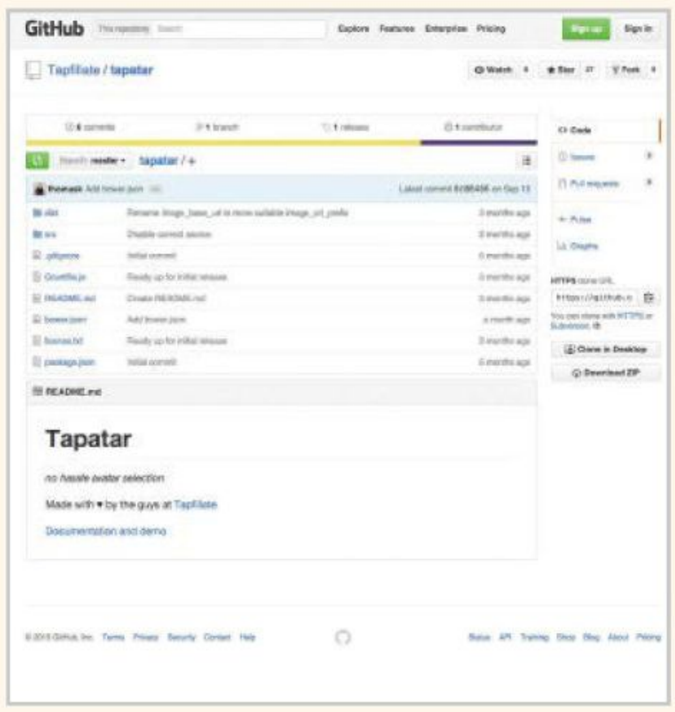
Waiting sucks. If the duration of a process is not known in advance, displaying a Windows Phone-inspired line of 'marching ants' is but a small acknowledgment of the user's patience. ProgressDots lets you deploy a dotted line with but five lines of code:

```
$( '#progressBox' ).dottify({
dotSize: '28px', //set size of dot
randomColors: true, //use random colors
numDots: 7, //number of dots
radius: '20%' //set dot corner radius
});
```

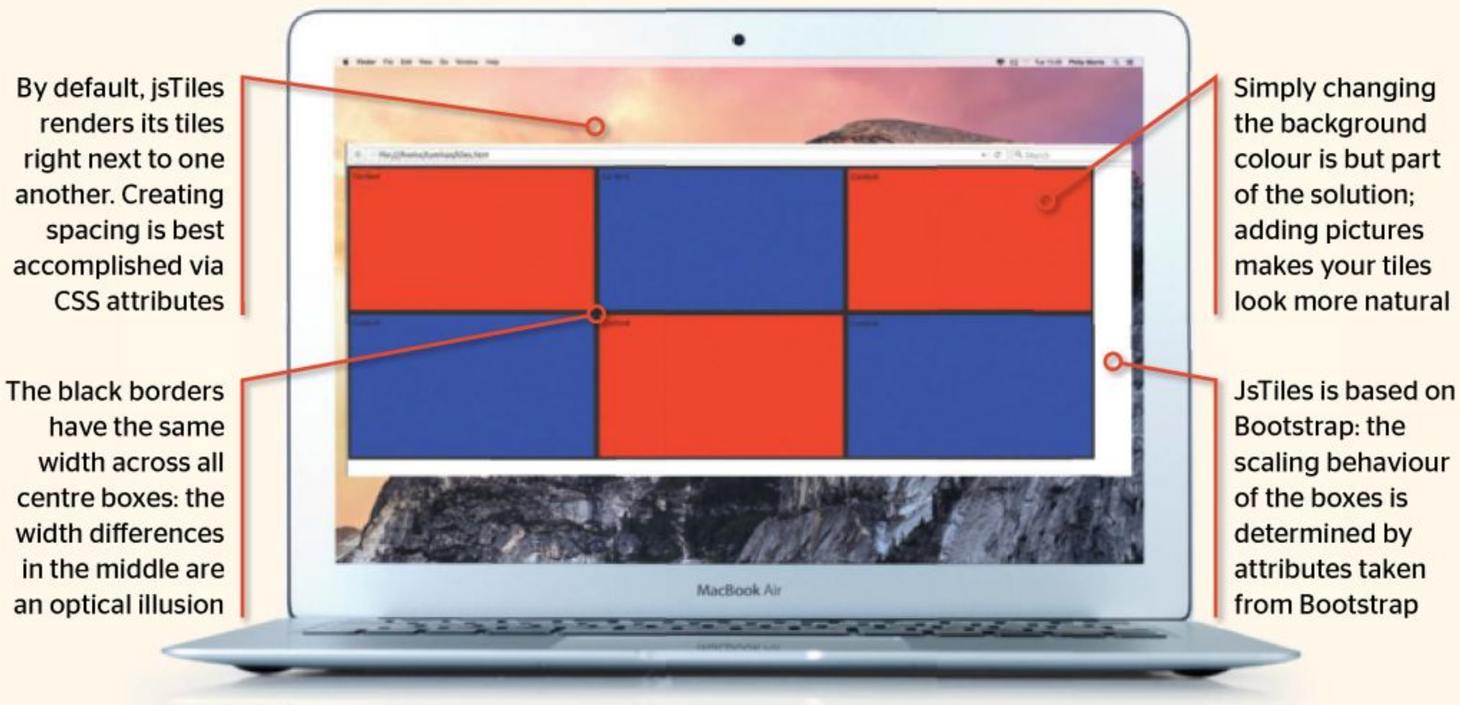


User avatar selection
bit.ly/1VWCIKO

Displaying faces next to comments adds to the perception of an active site or application. Tapatar simplifies the avatar procurement process: it provides a small widget where customers can select their avatar of choice by uploading a file from their desktop, logging in on Facebook or using the Gravatar database.



TILE INTERFACE



Plugin: jsTiles
bit.ly/1GhDhro

Category: Tile manager

Microsoft's Windows Phone 7 introduced the world to the concept of the tile: ever since, the design paradigma got more and more attention. Implementing tiles looks simple at first glance, but tends to become tedious once attention is paid to details.

Getting started with jsTiles requires the presence of three support libraries: jQuery, jQuery Easing and Bootstrap. As you can see in the full code on FileSilo, Style.css and jstiles.js come from the jstiles distribution, while the remaining references come from support libraries. The following container is to be created:

```
<div id="tiles-container">
<div class="tl-page" data-tl-
template="myTemplate">
<div style="background-color: #f00; border-
style: solid; border-width: 5px;">Content</div>
…
</div>
</div>
```

A superficial inspection of the code on FileSilo reveals that the tile management framework limits itself to the arranging of the individual elements; their design, layout and content lie in the responsibility of the developer. With that out of the way, the template must be configured:

```
var myTemplateObject = {
myTemplate: {
tilesNum: 6,
rows: {
0: {
rowClass: 'tl-row col-xs-12',
start: '3',
end: '5'
}
},
tiles: {
0: 'col-xs-6 col-md-3',
1: 'col-xs-6 col-md-3',
2: 'col-xs-6 col-md-3',
3: 'col-xs-6 col-md-3',
4: 'col-xs-6 col-md-3',
5: 'col-xs-6 col-md-3'
}
}
}
```

The most important part of the options array is the tilesNum property: it must match the number of the tags in the container. Rows sets up the size of the individual rows, while tiles configure the individual tile widths. Js-tiles will need to work with Bootstrap: the class names used are taken from the template framework. Next up we'll apply the generated configuration. In our example, the following bit of code shall suffice:

```
var opt = {
templateObj: myTemplateObject,
}
$('#tiles-container').jstiles(opt);
});
```

With that, our example is done. Js-tiles provides a variety of additional features for animation and card management: further information on these topics can be found in the documentation.

KEEP HEADERS ON TOP

Plugin: Headtacular
bit.ly/1Lon5Id
Category: GUI stack
Keeping a header on top of scrolling content can turn out to be a daunting task. This plugin is a representative of the BYOS genre: the acronym stands for Bring Your Own Style, which means that it limits itself to the provision of display logic.

As always, our code example starts out by including both jQuery and the headtacular file. In the next step, shown below, you can now see that a little stylesheet has been generated for us:

```
<!DocType html>
<html>
<head>
<script src="jquery-2.1.4.js"></script>
<script src="jquery.headtacular.js"></script>
<style>
.header {
transition: all 300ms;
background-color: #f00;
position: relative;
z-index: 9999;
}
.header.is-stuck {
position: fixed;
top: 0;
width: 100%;
background-color: #f0f;
}
</style>
</head>
```

Headtacular permits you to use two styles: header is used when the user has not touched the scroll point, while header.is-stuck's turn comes once scrolling has begun. In the next step, a small header – a b tag along with some text shall suffice – is created:

```
<body>
<div id="myHead" class="header"><b> My Header</
b> and some text</div>
<div id="myFill"></div>
</body>
</html>
```

Finally, the headtacular method must be invoked after the DOM tree has been initialised. Our snippet passes in an object showing one of the supported options – as the plugin contains sensible defaults for all options, you can also call the method with an empty JSON object:

```
<script type="text/JavaScript">
$(document).ready(function() {
var innerHtml="Lorem Ipsum sic . . .<br>";
for(i=0;i<12;i++){innerHtml+=innerHtml;}
$( "#myFill" ).html(innerHtml);
$('#myHead').headtacular({ scrollPoint: 1 });
});
</script>
```

The lorem ipsum code unfortunately bloats the page: an empty website does not flow over the screen and thus does not require scrolling.

Colour picker
bit.ly/1ReHcpt

The creation of colour pickers have always turned out to be a nuisance, but you can easily rectify this problem in our example by making an open source colour picker for Windows Mobile. JPipette is a similar extension which lets users pick colours from an image on non-IE browsers.

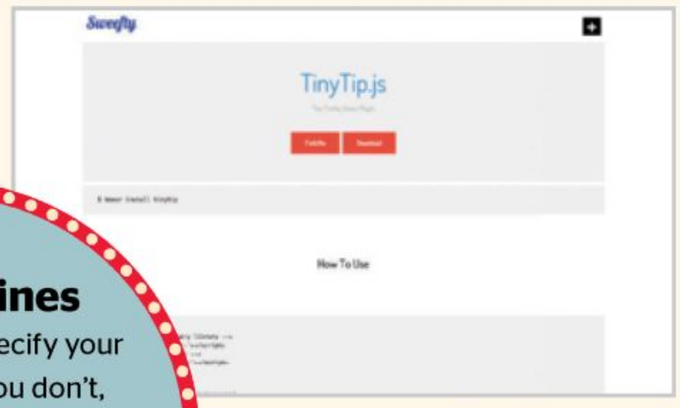
Integrating the colour picker is as easy as adding an input element to the site. A call to the jPipette() method then makes sure that everything is set up correctly:

```
<input type="text" name="myInput">
$(document).ready(function(){
$('input').jPipette();
)}
```

Tooltip windows
sweefty.com/tinytip

Forcing users to consult a manual is a sure-fire way to drive them straight to the arms of competitors. This plugin strives to increase discoverability by displaying tooltip windows with further information at strategic locations.

TinyTip differs from other products due to its flexibility. Tooltips can be styled via CSS; their appearance can furthermore be triggered by mouse-over, click and/or custom gestures. One classic example involves complex tooltip windows which get spruced up visually via CSS-generated arrow shapes and/or custom background colours.



Timeline display
bit.ly/1NLSX96

Log file viewers benefit from a timeline view. This timeline plugin takes a dataset, which is displayed in a zoomable, double-clickable widget. The dots in the timeline are presented horizontally and they can help users to visualise any graph patterns, as part of their log analysis. Its data structure looks like this:

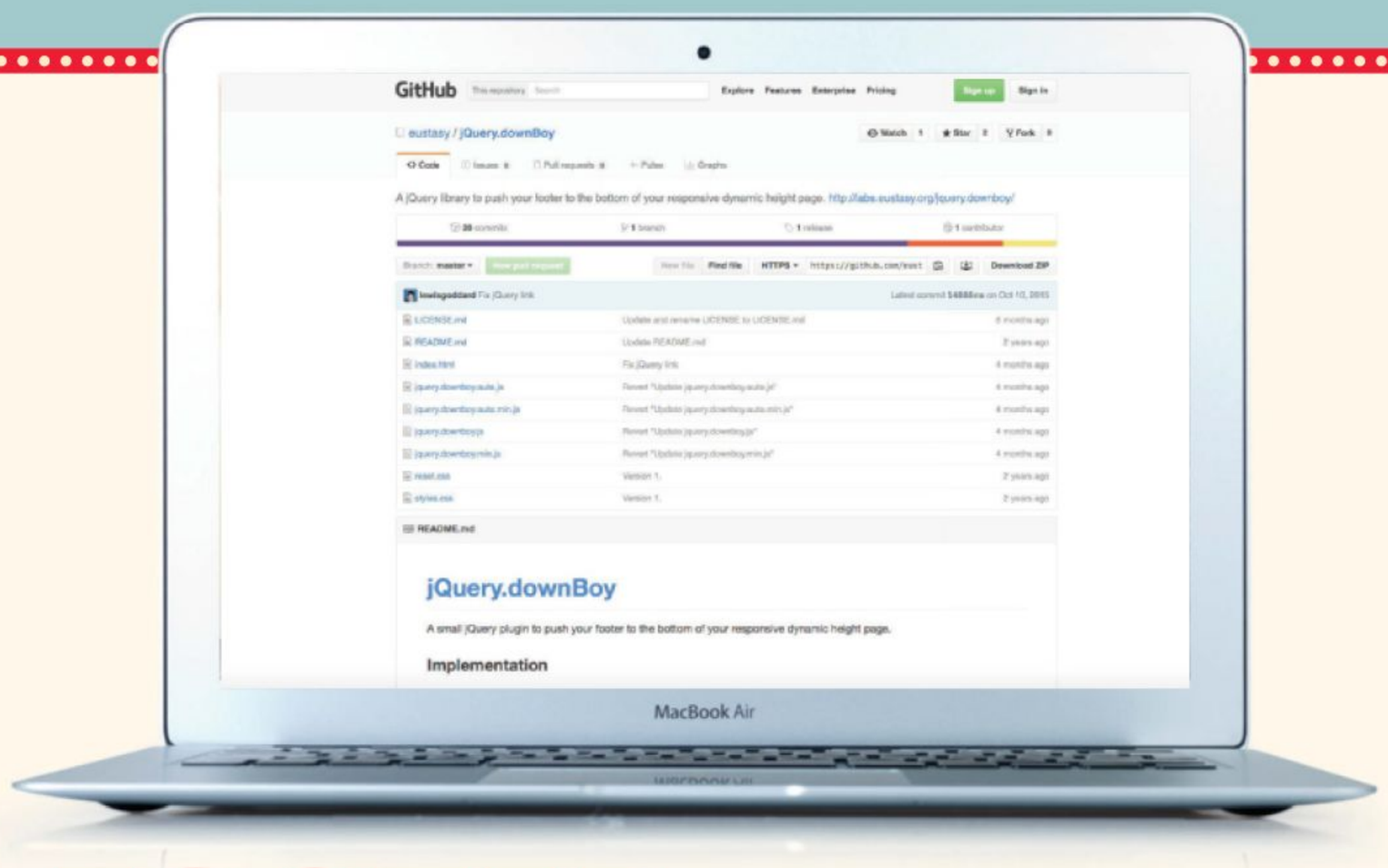
```
var categories = {
'alarms': {'color': "rgba(255,
0, 0, 0.7)"},
};
var events = [
{'time': 1398706972000, 'category':
'alarms', 'line': 12345, 'label': "one"},
{'time': 1398706771300, 'category': 'pings',
'line': "fred", 'label':
"two"},
];
var options = {'categories':
categories};
</script>
```

> *This timeline plugin takes a data set, which is displayed in a zoomable, double-clickable widget*

Timeline lines
It's important to specify your lines because if you don't, Timeline will automatically identify your lines and plot them in the order in which the plugin finds them (likely to be from bottom to top).

Each event is made up of three tuples: a Unix timestamp, a category (like colour for example) and a line identifier which describes the timeline housing it.

The Timeline plot uses the options object that is passed in for configuration. The options object currently supports attributes such as categories, lines, lineLabels and sortLines.



FOOTER

Plugin: jQuery.downBoy
bit.ly/1PlZesf
Category: Footer management

A footer can makes web apps look much more professional. Sadly, keeping content at the bottom of a page can be a bit of a daunting task for people who are not 100 per cent confident at working with CSS. This plugin wants to help you out here.

Testing a footer management plugin requires the presence of a footer, which you'll have to create first. You can start out by creating the following simple example page which self-populates with a bit of lorem ipsum as well as containing a small footer with some buttons:

```
<html>
<body>
<script src="jquery-2.1.4.js"></script>
<script src="jquery.downboy.js"></script>
<script>
$( document ).ready(function() {
var innerHtml="Lorem Ipsum sic . . .<br>";
for(i=0;i<7;i++){innerHtml+=innerHtml;}
$( "#myFill" ).html(innerHtml);
});
</script>
<div id="myFill"></div>
<div id="myFooter" style="background-color:
yellow;"><h1>This is a footer</h1></div>
</body>
</html>
```

DownBoy is offered in two different versions. If your footer goes by the name #footer, simply embed downboy.auto.*.js in order to put the element in its place. Developers working with a custom footer must instead deploy downboy.*.js, which requires the following bit of extra code:

```
<script src="jquery.downboy.min.js"></script>
<script>
$(function() {
downBoy('#myFooter'); // Run on load
window.onresize = function() { // On Resize
downBoy('#myFooter'); // Run Again
};
});
</script>
```

Keep in mind that DownBoy can handle scrolling only if the DocType is set like this:

```
<!DocType html>
```

"A footer can makes web apps look much more professional"

Scrolling indicator
bit.ly/1MHijDH

This plugin treats your users to a progress indicator informing them about how far they have already scrolled down the page – coders working with Sublime Text will find the feature familiar right away. The viewport differs from classic IDEs in that you need to provide the overview image.

Rain effect
bit.ly/1NLRfVa

Less can be more. This little plugin uses wave theory to render a line, which is then animated to simulate raindrop impact seen from the side. Daniela Harel and Gred McAusland's plugin is difficult to describe – take a look at the examples by going to the URL above, which really do the effect justice

Substitute with icons

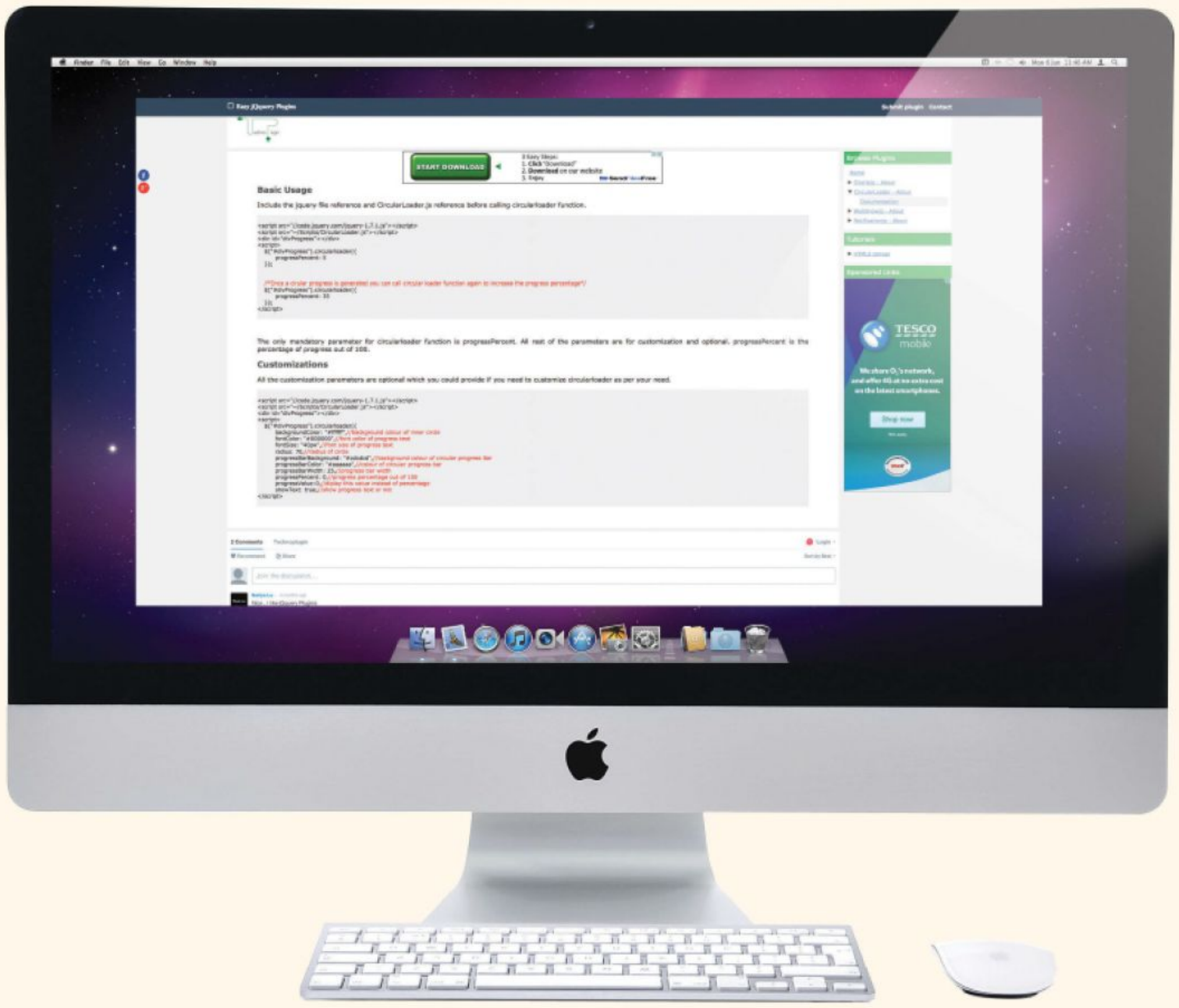
For the scrolling indicator, it's possible that you may encounter roadblocks with the overview image. Should this ever become a problem, feel free to substitute the image with icons.

Superfast parallax
bit.ly/1MtzLXU

Research by the Purdue University (**docs.lib.purdue.edu/cgttheses/27/**) has shown that parallax scrolling significantly increases user satisfaction. Bring a few svelte panoramic shots and this little plugin by Kaspars Jaudzems of Latvia can add the impressive scrolling effect to your web projects with minimal effort.

Virtual terminal
bit.ly/1GK5Qso

This plugin simulates a Ubuntu-styled terminal window by displaying a set of span elements according to a timing schedule. It is best suited to providing a visual guide for unsophisticated users working on a command-line system – be aware that the plugin cannot handle user interaction.

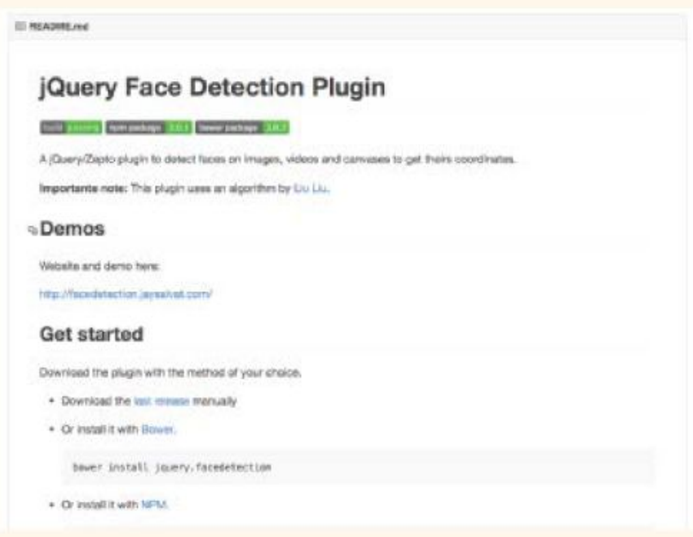


Circle calculations
bit.ly/1MtzZye

Waiting for Godot is normal nowadays: why not ease the wait by presenting your users with a snazzy little circular progress bar? This plugin saves you from having to deal with the sine, cosine and tangent as found in circles. Creating and/or updating the widget is as easy as calling the circularloader function like this:

```
$("#divProgress").circularloader({
progressPercent: 35
})
```

“Face detection is relatively ‘accepting’ of older jQuery versions – the plugin’s author specifies it with jQuery 1.11.1. We’ve used a recent edition for brevity reasons”



FIND FACES

Plugin: Face detection
bit.ly/1jGT3BR
Category: Image analysis
Finding faces in pictures is important for multiple reasons: when digital cameras with the feature were first introduced, getting perfectly exposed portraits became so much easier. This plugin allows you to look for faces in image files provided by your users or a server of choice.

Face detection is relatively ‘accepting’ of older jQuery versions – the plugin’s author specifies it with jQuery 1.11.1. We’ve used a recent edition for brevity reasons:

```
<!DocType html>
<html>
<head>
<script src="jquery-2.1.4.js"></script>
<script src="ccv.js"></script>
<script src="cascade.js"></script>
<script src="jquery.facedetection.js"></
script>
</head>
```

Face Detection expects to find its source picture in a <img> tag. We have used an image provided by the developer here – feel free to swap out its URL to point to a photograph of choice:

```
<img id="picture" src="picture.jpg">
```

Now let’s invoke the actual processing function. As with most other jQuery plugins, Face Detection takes an array of elements which permit you to customise plugin behaviour. Complete gets invoked once faces have been detected. The method is provided with an array of face objects, each of which exposes a group of properties containing further information about the detected face. Our code example restricts itself to outputting some data to the debugger’s console:

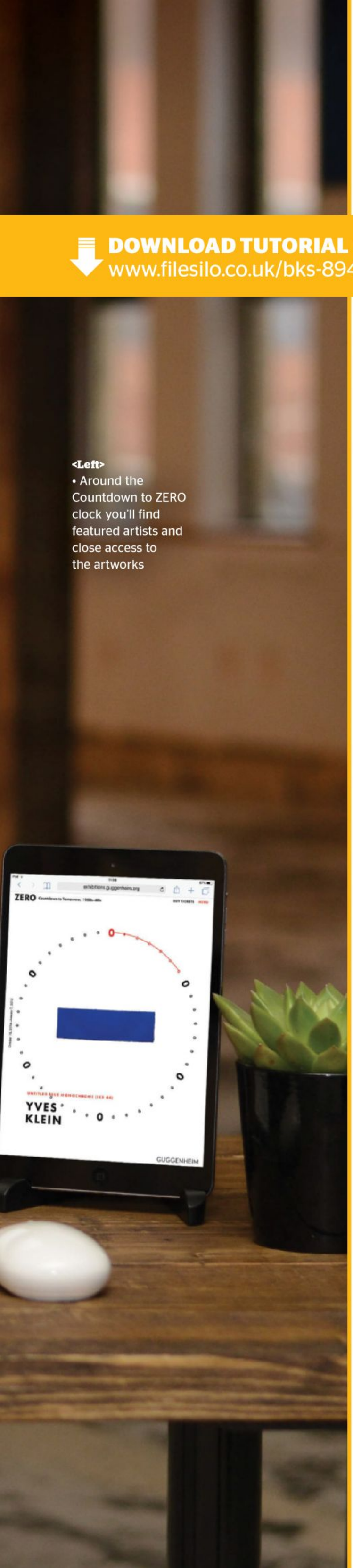
```
$('#picture').faceDetection({
complete: function (faces) {
for(i=0;i<faces.length;i++)
{
console.log(faces[i].x + " " + faces[i].
width + " " + faces[i].y + " " + faces[i].
height);
}
}
});
```

Inspiration

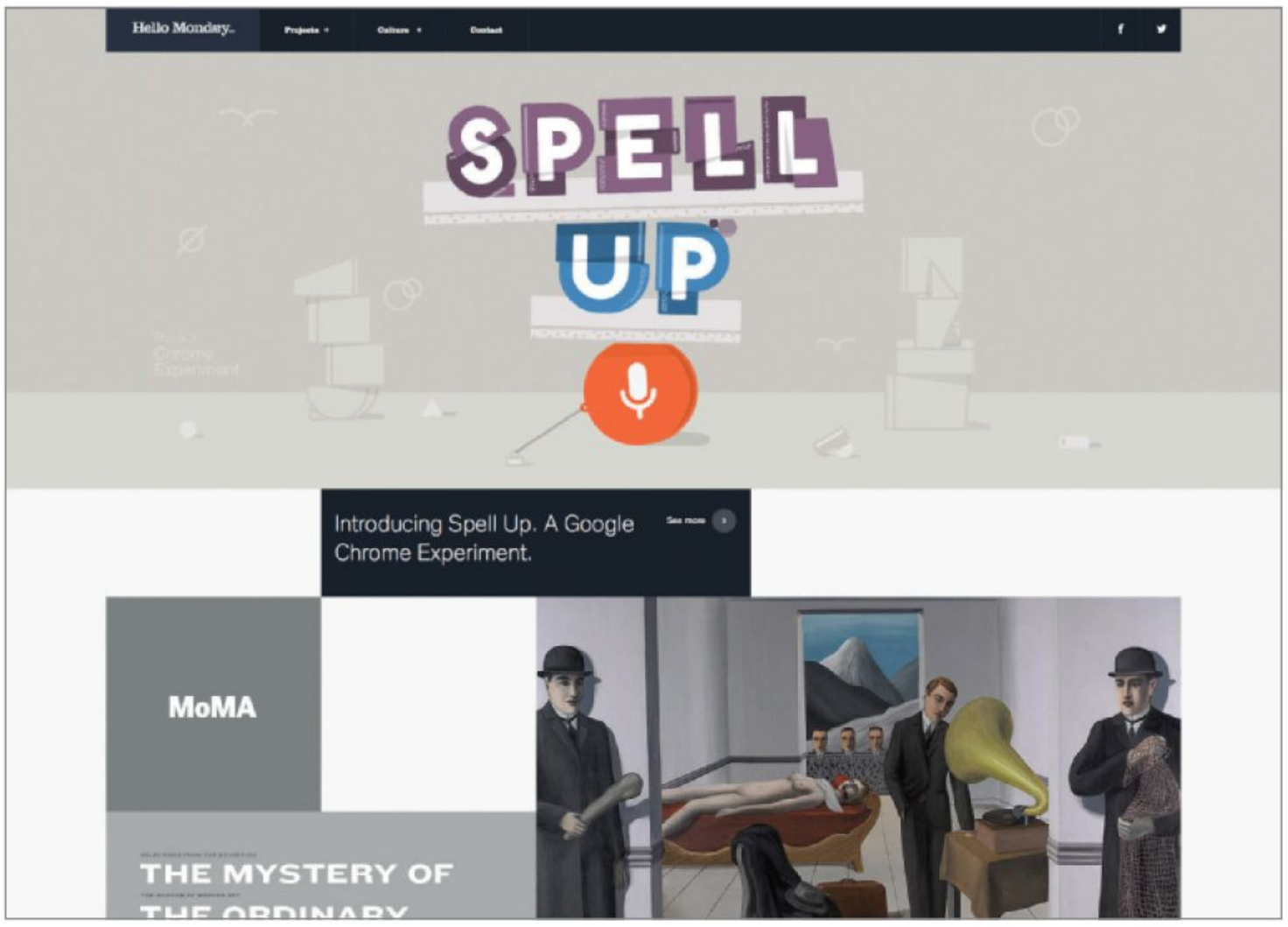
Guggenheim ZERO

This virtual catalogue for an art retrospective is vibrant and fun – learn how to copy its interactivity

DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/bks-894



<Left>
• Around the Countdown to ZERO clock you'll find featured artists and close access to the artworks



Designer Hello Monday
Site exhibitions.guggenheim.org/zero

The Solomon R. Guggenheim Museum on Fifth Avenue New York is one of the most iconic art galleries in the world. The cylindrical, shell-like building, designed by Frank Lloyd-Wright in 1959, typically plays host to art exhibitions – with ZERO currently running until January 2015. Titled ZERO: Countdown to Tomorrow, this dedicated website provides a rich, virtual preview of the featured exhibits and links to booking viewings. Opening on an intriguing abstract video sequence and landing within a clean white template, artworks are arranged within a clock-like rollover wheel. Clicking on a node reveals each example, moving to neighbouring pieces left and right or zooming in for closer inspection.

Everything is kept very gallery-like and true to Guggenheim's identity, using a palette that seldom strays from white, black and red. Delve deeper and the story of the ZERO movement is explained with longer poster-like pages where subtle image effects, fading backgrounds and Futura PT styled typography maintain a stylish experience across multiple devices.

It is a combination of CSS transitions and the jQuery-like Minified.js that provide the slick moments of dynamism, while atmospheric audio design recommends headphones for added immersion. Celebrated agency Hello Monday, again based in New York, take credit for a project that has already scooped honours such as FWA's Site of the Day.

<comment>
What our experts think of the site

User interaction and IoT

"Multiuser sites and applications are on the rise, and will become much more widely used, creating new layers of interaction. The Internet of Things will flourish even more – going from experimental uses to useful applications that people can interact with, taking user interactions on a website into the real world."

Anders S. Jessen – Technical director at Hello Monday

> Everything is kept very gallery-like and true to Guggenheim's identity, using a palette that seldom strays from white, black and red

HTML & CSS

abcABC
1234567890
abcABC
1234567890

#D72229 #101010 #FFFFFF #A1A1A1 #B8B8B8

<Above>
• Futura PT by ParaType, and available via Typekit, handles the text in Book, Medium, Heavy and Heavy Oblique (700). These clean fonts suit the website's modern style perfectly.

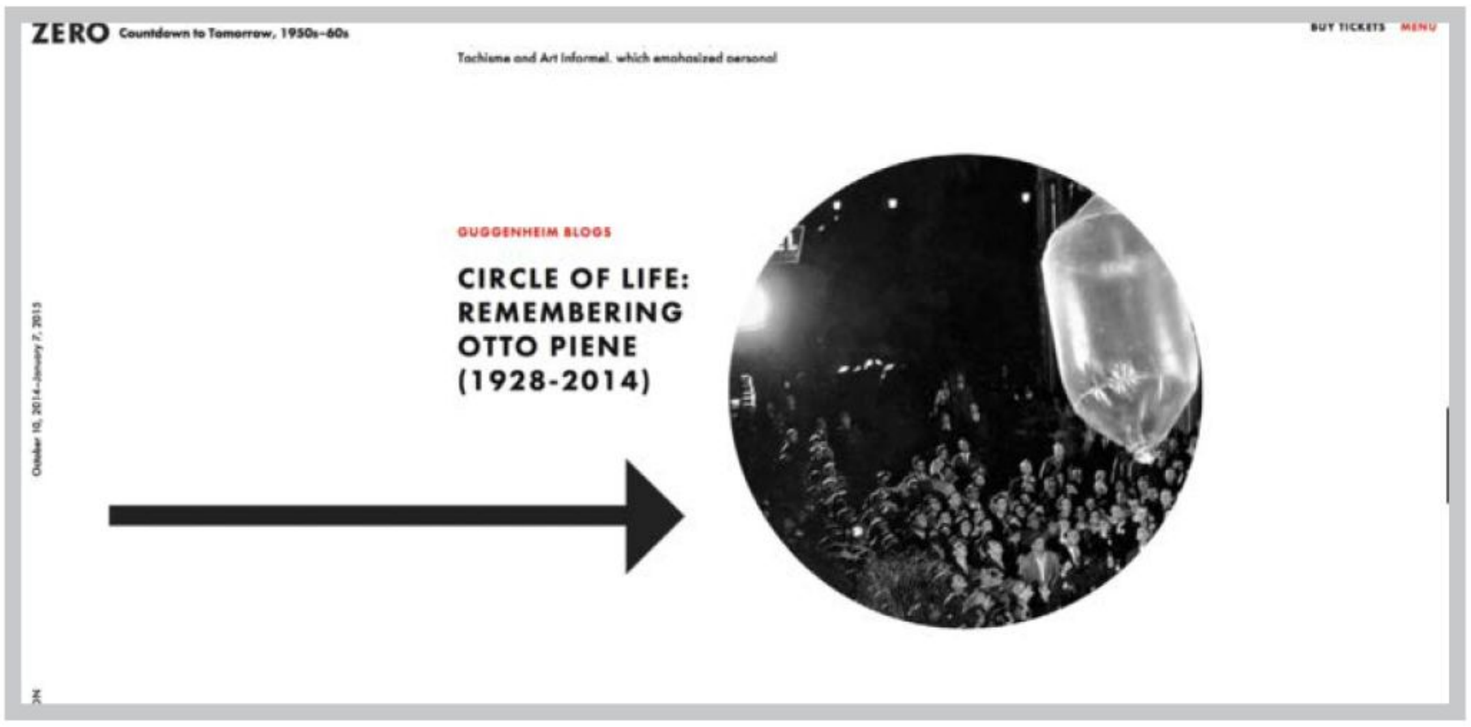
<Above>
• A tasteful arrangement of images and text gives the exhibition plenty of vacant space for content

<Top right>
• An injection of red catches the eye and this fullscreen navigation menu lets you jump between sections

<Bottom right>
• The Network area plots where and when landmark exhibits within the ZERO movement took place

Draw CSS arrow shapes and style circular images

Create arrow and circle background frames, inspired by the custom post styling of Guggenheim ZERO



1. The arrow elements

For the first step in this tutorial, let's start with this idea of creating an arrow shape purely with CSS. On Guggenheim ZERO, the arrows are images but in fact these arrows can also be made more elegantly. It begins with two elements, one to form the arrow stem and the other to be the pointed head. We give these an id to suit and nest them inside another we'll call 'arrow-wrapper' like so:

```
<div id="arrow-wrapper">
<div id="arrow-stem"></div>
<div id="arrow-head"></div>
</div>
```

2. Arrow wrapper styling

We want the two parts of our arrow to be joined so when we move them around the page they remain as one. Later we will also ensure the head element floats to the head of the stem on the right. In our CSS class we'll set the desired width and height, while the top margin pushes it down inside a main container:

```
#arrow-wrapper {
width: 700px;
height: 100px;
margin-top: 400px;
}
```

3. Style the arrow stem

The arrow stem is styled so that we actually only see the element's bottom border, set to black. Setting the background to transparent and the height as required means the 34px solid border line is pushed down to meet the centre point of what will be the arrow 'head' sitting directly to the right:

```
#arrow-stem {
background: transparent;
width: 600px;
height: 33px;
border-bottom: 34px solid  black;
}
```

4. Create the arrow head

Now it's time for us to continue with creating the arrow head. The arrow head will append to the right of the stem to form our completed arrow. This is formed by adding the following CSS class and by rendering a triangle shape. This technique is derived from **http://css-tricks.com/examples/ShapesOfCSS** – a page with some truly excellent solutions for drawing various different CSS-based shapes. Essentially, our again becomes merely a left border by making the top and bottom ones transparent:

```
#arrow-head {
border-top: 50px solid transparent;
border-left: 100px solid black;
border-bottom: 50px solid transparent;
}
```

5. Add image and float left

The next vital part in our CSS is floating all our elements to the left, so we not only force our arrow into line but also line up the wrapper to our new circular image. So in your HTML just after the arrow-wrapper add a fresh called 'circleImg' as shown. The container is merely a box for positioning everything centrally on the page:

```
CSS:
#arrow-stem, #arrow-head, #arrow-  wrapper,
#circleImg {
float: left;
}

HTML:
<body>
<div id="container">
<div id="arrow-wrapper">
<div id="arrow-stem"></div>
<div id="arrow-head"></div>
</div>
<div id="circleImg"></div>
</div>
```

6. Style the circle image

Our example image is added as a background for circleImg in the associated CSS class setting the desired width and height dimensions squarely to 600 by 600 pixels. The property background-size can then be set to 'cover' if you want the source image in a better size within the frame. We then simply use the border-radius property and set it to a value exactly half of our edges, so in this case it will be 300px. The margin-left of 70px is merely a spacer between the arrow point and image. And now you're all done!

```
#circleImg {
background-image: url('myImage. jpg');
width: 600px;
height: 600px;
border-radius: 300px;
margin-left: 70px;
}
```

DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/bks-894

Build 3D experiences with PlayCanvas

Learn how to use the PlayCanvas library, a 3D HTML5 game engine delivering beautiful interactive experiences

HTML5 Canvas has been a fantastic addition to the web's feature set for 2D experiences, allowing for easy adoption among developers with straight forward APIs, documentation, and generally quick development times. HTML5 canvas has made the web the go-to platform, should someone want to learn what's involved in creating a 2D experience quickly and with minimal fuss.

Despite 3D experiences with native web technologies being a reality for the same amount of time as 2D ones, there hasn't been as much progress. Even drawing something as simple as a triangle in WebGL means learning about things called shaders and understanding what a buffer is. This complexity really whittled down the number of people making engaging experiences for the platform. Until libraries like three.js came about, a lot of developers simply wouldn't bother as it was just too much of a learning curve.

This is something that PlayCanvas has obviously been aware of and tackled cleverly. There is an open source game engine on github (**github.com/playcanvas/engine**) for people who want to write code, but they've also produced a fantastic game editor as a web app. This tutorial covers using the editor to create a basic scene with a first-person viewpoint.

1. Get started with PlayCanvas

Visit **playcanvas.com** and create a free account, which makes most of the features available. The playable demo for this tutorial can be found at **playcanvas.com/project/364550/overview/web-designer-243** Logged in users can also view and edit the project in the editor.

2. Create a new project

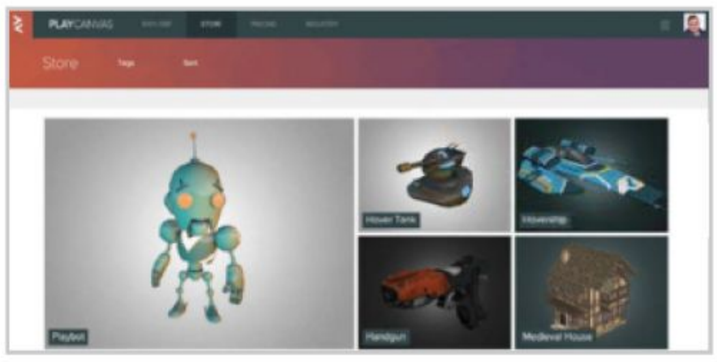
The big orange "+ New" button will bring up some options for creating a new project. The default platform game and model viewer projects are both good introductions into what's possible with PlayCanvas. For this tutorial, create a blank project.

3. Check out the editor

As can be seen above, settings and tools are on the far left. The hierarchy panel lists all the entities available in the scene. The 3D area is the scene. The assets panel is a library of all assets available to the project. The inspector displays information and tools on selected entities.

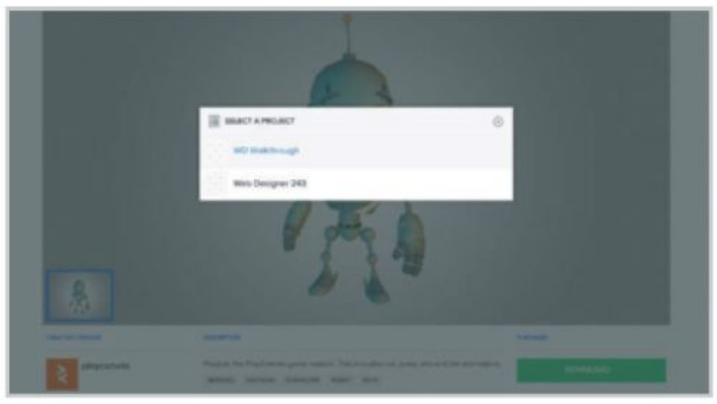
4. Grab some assets

PlayCanvas has a "store" of 3D assets that the community has produced. It seems that in future some of these may be paid for, but at the moment they all seem to be free. Find a model that has some animations as well as an environment to put it in.



5. Download assets to projects

Clicking the green download button will bring up a list of projects the asset can be added to. Once added to a project, this asset will appear in that project's asset library panel, ready for use in a scene.

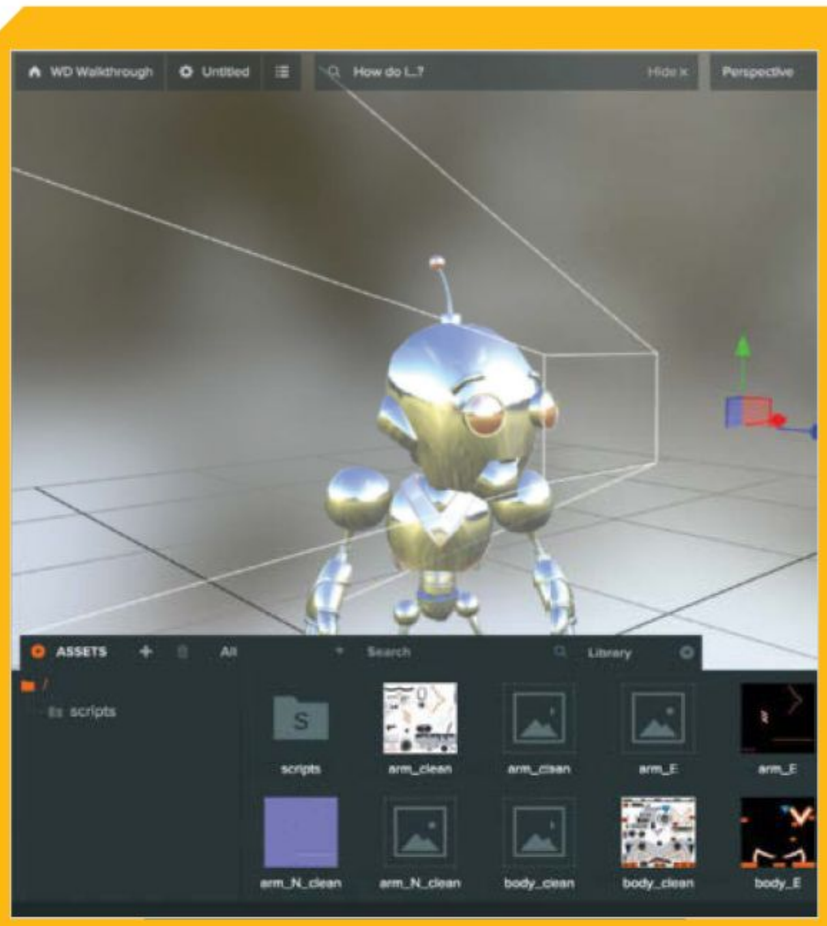


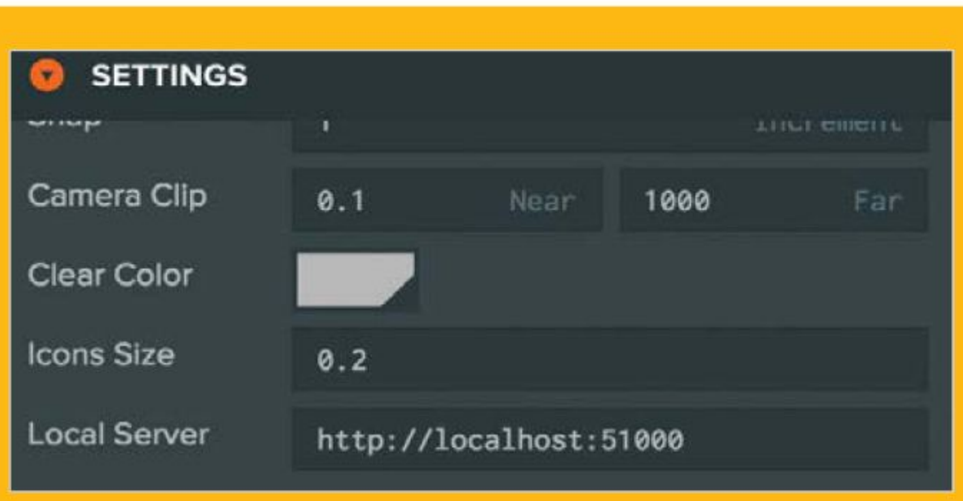
6. Add an asset to a scene

Back in the editor, the model will be in the assets panel.

Play with Mip

In the scene settings the environment texture has a Mip dropdown. The demo scene looked odd with a Mip of 1, but a Mip of 3 blurred everything and really created a sense of space.



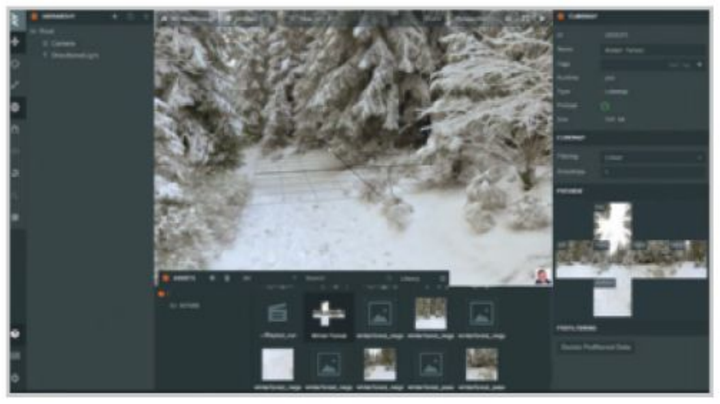


<Top left>
• Detailed inspector view of the scene and environment settings. Camera and environment are the main focus here, but experimenting with fog can be interesting

<Bottom left>
• The scene with just the Playbot model and the 3D environment assets downloaded and then added to the scene, blurred using the Mip dropdown

<Left>
• Shows the framing of the camera and model in the scene. The wide angle on the camera will make the model appear closer in-game than it looks here

The model will have a 3D preview as an icon and also have meshes in the inspector panel. The environment will look like a cube net. Click and drag both into the 3D area to add them to the scene.



7. Scene & environment settings

Click the cog at the bottom left of the editor to see the options that are available to you appear in the inspector window. The tutorial scene took the camera exposure down, and then it bumped up the intensity of the environment. This was because the highly reflective Playbot model was too bright.

8. Moving the camera

To make the scene interactive, add a fly camera script component to the camera entity in the hierarchy using right-click. Fly camera creates a first-person-view camera that can be moved with WASD and Arrow keys, with the mouse controlling the swivel.

9. Tweak the fly camera

Test playing the game will reveal the camera moves very quickly. To create a scene where the player can slowly explore the model and examine it up close, take the speed down, reduce the near clip, and finally increase the field of view.

10. Compose your scene

Next, let's look at using the translate tool on the far left of the editor and then put the entities in their default positions. The demo for this tutorial starts off with the player looking Playbot directly into the reflection in his soulless eyes.

11. Assign animations

Selecting the model entity in the hierarchy will present you with a few options inside of the inspector panel. Click to add a component and then choose an animation. From there you will now be able to add the animations that came with the model. If there aren't any such animations, then you will need to work on a different model from the store.

12. Controlling the model

To control which animation is playing, a script component will be required. Select the model in the hierarchy, and over in the inspector panel add a new script component. Call this script 'model_name_control.js'.

13. Anatomy of a script

All new PlayCanvas scripts look similar to this one. The code registers a script called playbot_control and makes the entity object (representing the 3D model in this case) publicly available. This entity is then returned for the PlayCanvas (pc) engine to consume.

```
pc.script.create('playbot_control', function
(app) {
var Playbot_control = function (entity) {
this.entity = entity;
};
Playbot_control.prototype =
{
};
return Playbot_control;
}
);
```

14. The Initial function

Extend the first function to setup some event listeners and the default state animation to use on the model. blendTime float will be used to blend different animations together, avoiding a "snap" between the two. Think of this like a "main" function.

```
var Playbot_control = function (entity) {
this.entity = entity;
this.blendTime = 0.2;
this.setState('idle');
app.keyboard.on(pc.EVENT_KEYDOWN, this.
keyDown, this);
```

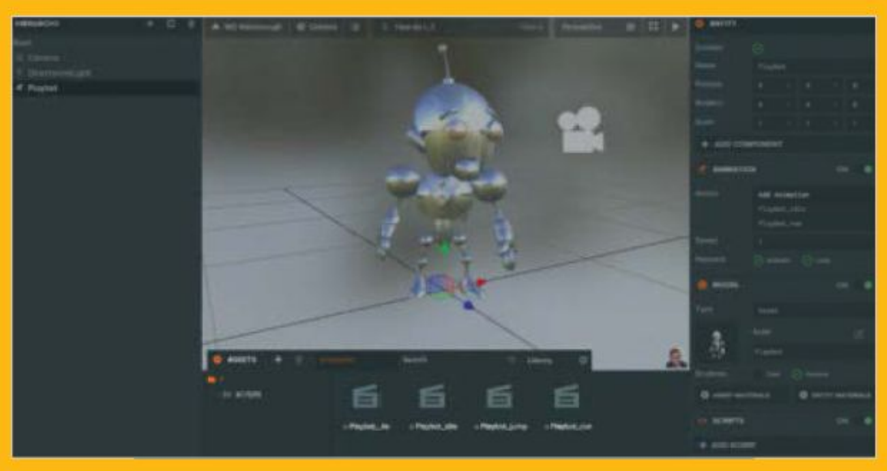
Animation setup

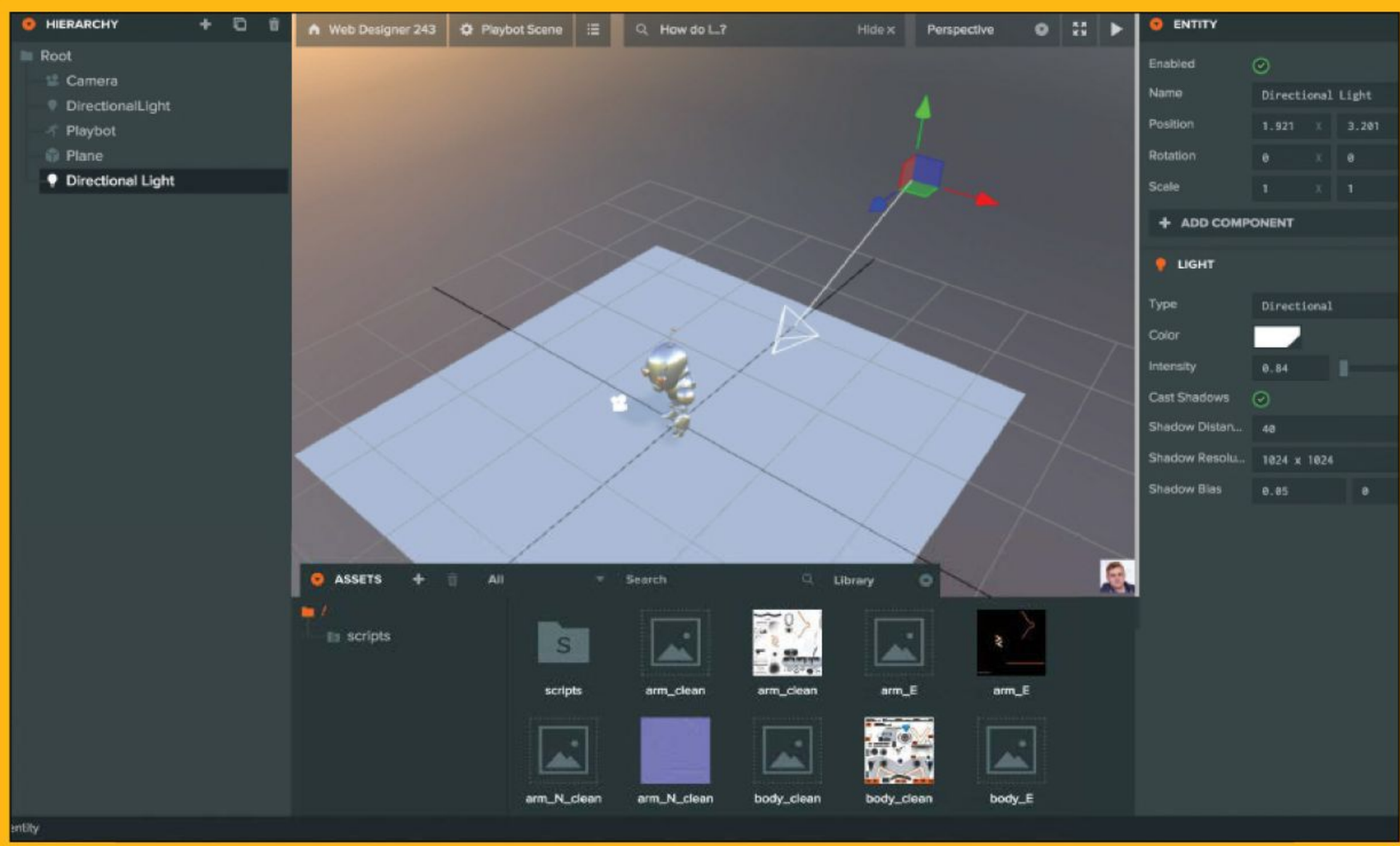
Animations should be listed under the animation component as assets. "Activate" is required for animations to play, and "loop" is required for animations to loop.

<Top left>
• When adding an animation from the animation assets area, the PlayCanvas editor will highlight all animation assets in the project's library

<Bottom left>
• Adding a script from the scripts area will cause the PlayCanvas editor to highlight them in the library. New scripts can also be added here

<Right>
• The mini metal titan Playbot in midstride, running across the featureless landscape, the sun shining behind him and infinity in front





Try adding lights and shadows

To really take the scene to the next level, try adding in a light source as well as a surface, so that the model can cast shadows. A plane works well for the surface, as it doesn't need a material, but under the Shadows section in the inspector, the "Receive" checkbox will need to be checked for this.

It's worth playing with the different types of lights that PlayCanvas has to offer as each have different options that change the look of light and shadow.

The demo scene uses a directional light with the intensity taken down slightly. Upping the shadow resolution will give a better quality shadow, and reducing the shadow distance will make it less blurry. Don't forget, the "Cast Shadows" checkbox must be ticked.

```
app.keyboard.on(pc.EVENT_KEYUP, this.keyUp,
this);
};
```

15. Extend with prototype

After defining the initial function, the prototype should be used to set up further variables and define further functions. This way long functions and other complicated logic is visually separated from the initial function, or "main" function.

```
Playbot_control.prototype = {
};
```

16. The setState function

In the initial function there was a call to a function called setState which was going to set the animation the entity was using. The code calls part of the PlayCanvas API which plays animation on entities and passes it the name of the animation, with the blendTime defined before.

```
setState: function(state) {
this.state = state;
// Set the current animation, taking 0.2
seconds to blend from
// the current animation state to the start
of the target animation.
this.entity.animation.play(states[state].
animation, this.blendTime);
},
```

17. The state object

The previous step passed the name of an animation using this object. Abstracting animations to states like this allows for simpler names and concepts to be used in the codebase and also means less typing should your model come with long animation names.

```
var states = {
idle: {
animation: 'Playbot_idle'
},
run:
{
animation: 'Playbot_run'
}
};
```

18. Key binding

The final part of the puzzle is to bind the state changes to key presses so players can interact with the scene. The code will play the run animation while a user holds down the spacebar key, going back to its idle state upon its release.

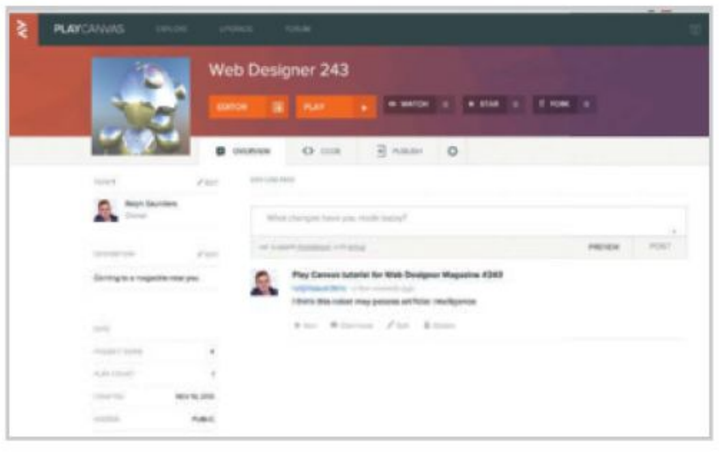
```
keyDown: function(e) {
if(this.state != 'run' && e.key == pc.KEY_
SPACE) {
this.setState('run');
}
},
keyUp: function(e) {
if(e.key == pc.KEY_SPACE)
{
this.setState('idle');
}
}
```

19. Play testing time

Finally, the scene is ready for play testing. Clicking the play icon will bring up a launch menu. Launch will launch the game in a new tab. This is the full experience, and what players would get if the game were published.

20. Project Page

Back on the main PlayCanvas website, there will be a project page for the game that has just been created. Here a preview image can be added, along with developer and community comments, and other information about the project.



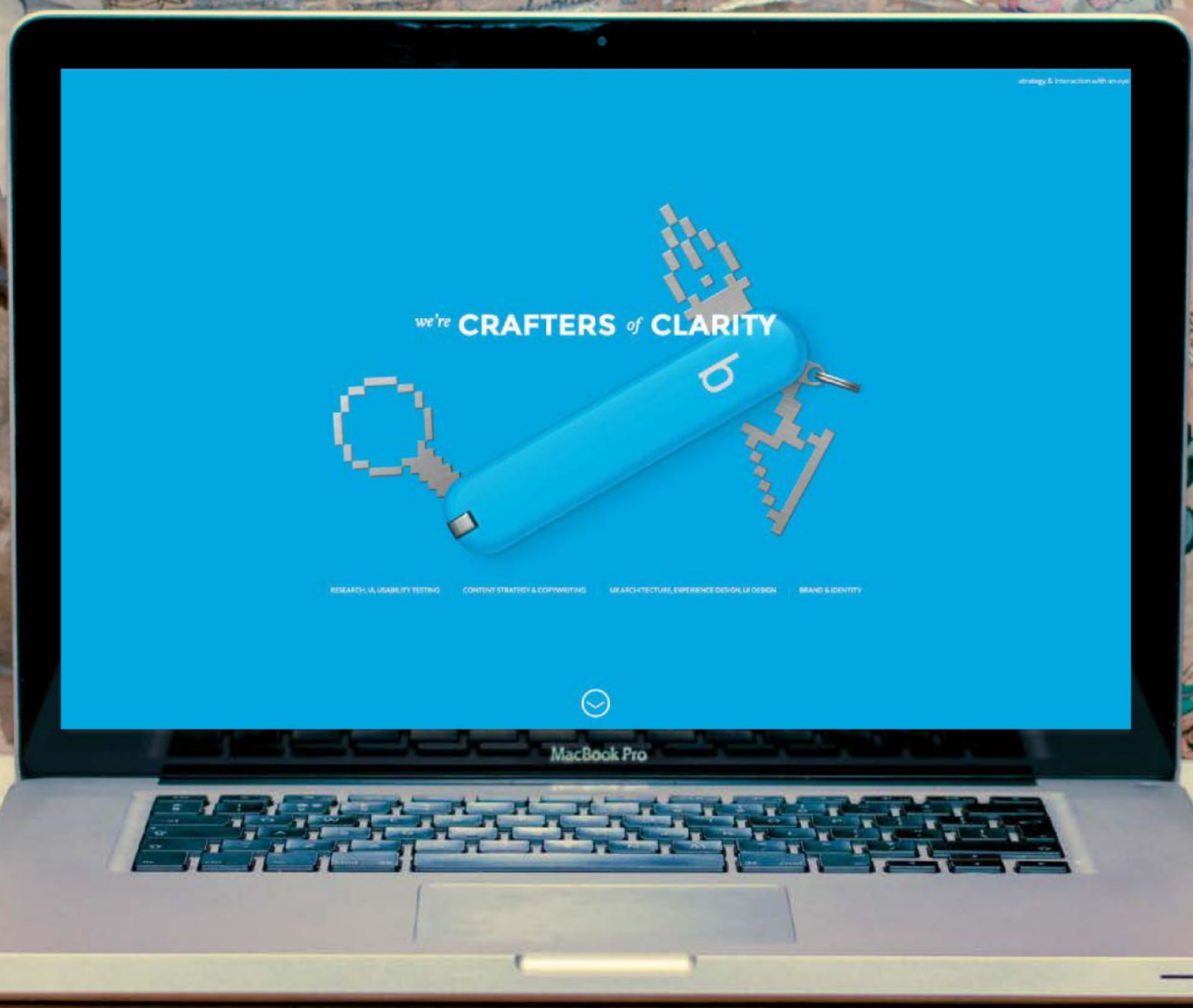
21. Publish the game

On the publish tab there's an option to publish the game, which will make a build of the game public through the PlayCanvas website. Publishing a build allows for selection of particular scenes. When publishing is complete, PlayCanvas will craft a unique short URL for the game.

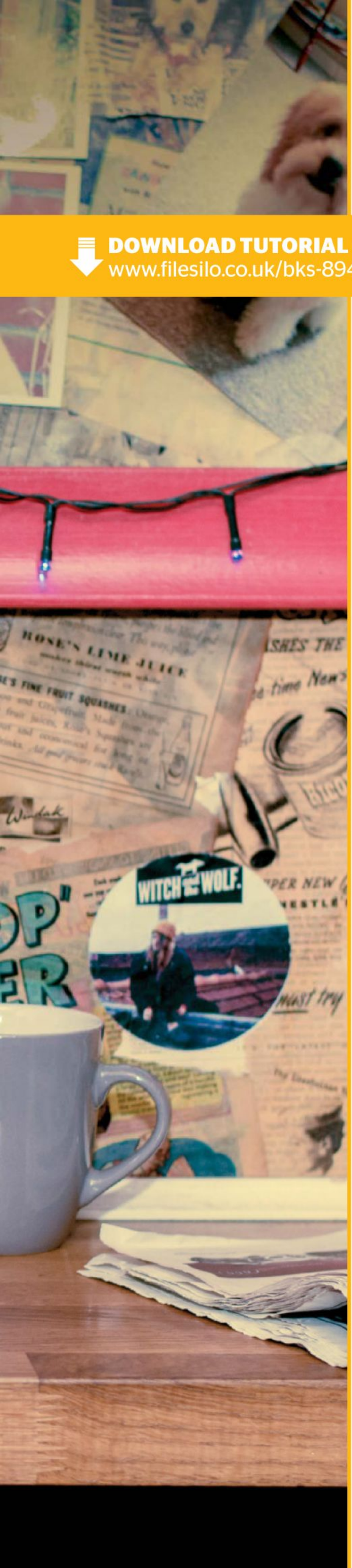
Inspiration

brightly digital agency

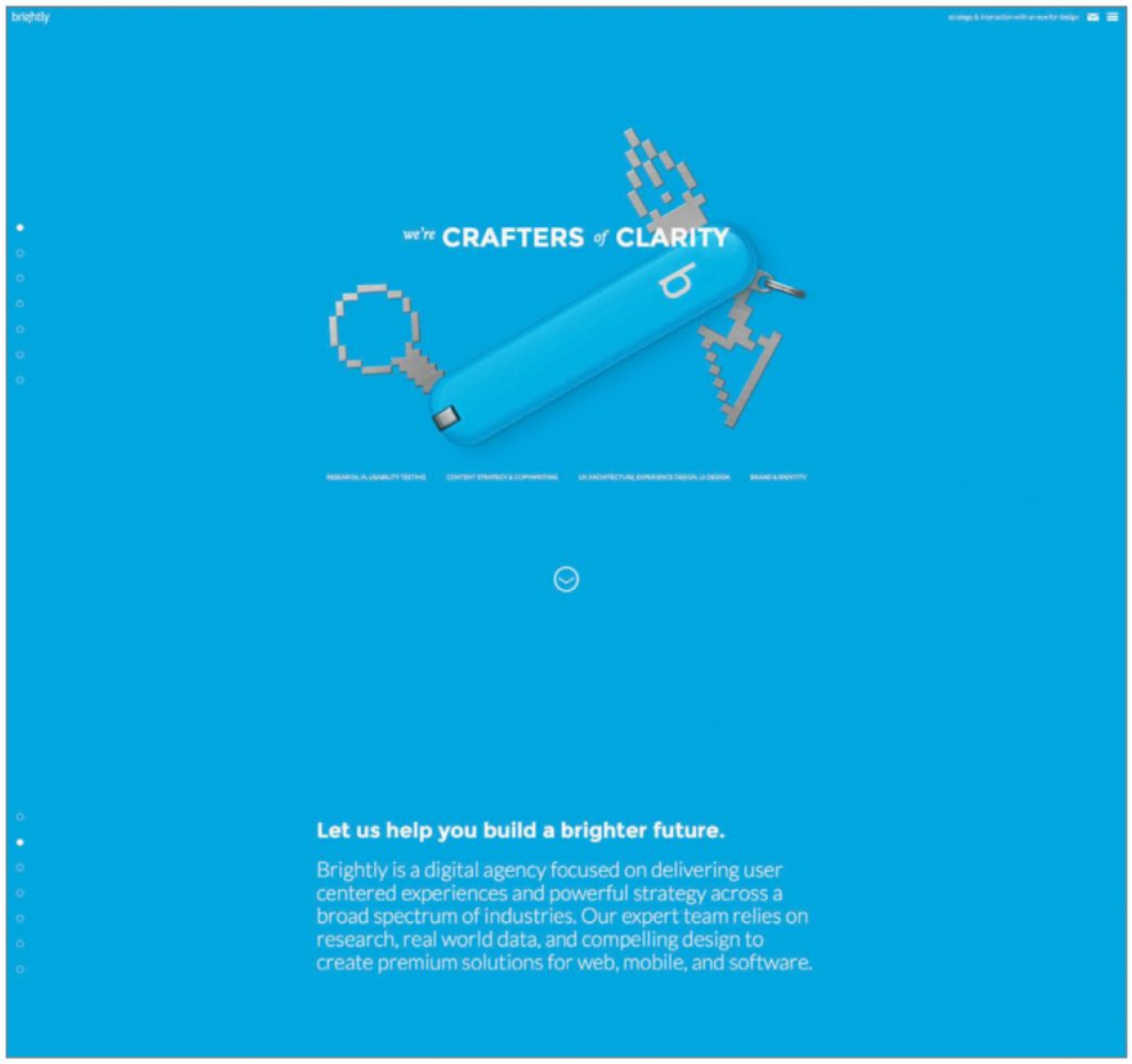
The creative process is celebrated within agency brightly's latest designer dotcom, and you can recreate its slick style



<Right>
• The site's main page thread is long and vertical, with sections also prompting horizontal navigation to show further content



DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/bks-894



Designer brightly
Site teambrightly.com

The latest in a long line of noteworthy agency websites so far this year, this stellar effort from Michigan-based studio brightly is a shining example. Operating out of Grand Rapids, the website exhibits a rather apt, tangible sense of "team" that floods out from this cascading WordPress-powered template.

Opening on an animated PNG of an unfurling Swiss Army knife, suitably reconfigured for digital deployment, the message is multidisciplinary. These "Crafters of Clarity", who are known for delivering projects across a broad industry spectrum, have a creative story to tell. Built around a long central 'page' that behaves more like a vertical slider, sections branch off to screen right producing a waterfall-like feel to the navigation. Horizontal threads detailing team biographies, working process and featured work within web, mobile and software are all colourfully represented. "For our new site, we wanted to convey that we're more than designers; we're problem solvers," explains Christian Faragalli, associate creative director at brightly. "We've built a strong team of researchers and strategists, and those forces drive our process. It has allowed us to tackle projects of all size and complexity, and our site needed to reflect that.

We also wanted to have a little fun. The idea was to make a cutting-edge site with clear messaging, easy navigation, and bits of visual storytelling throughout. We used bold colours and clear typography to achieve a classic look with a modern twist."

> The idea was to make a cutting-edge site with clear messaging, easy navigation, and little bits of visual storytelling throughout

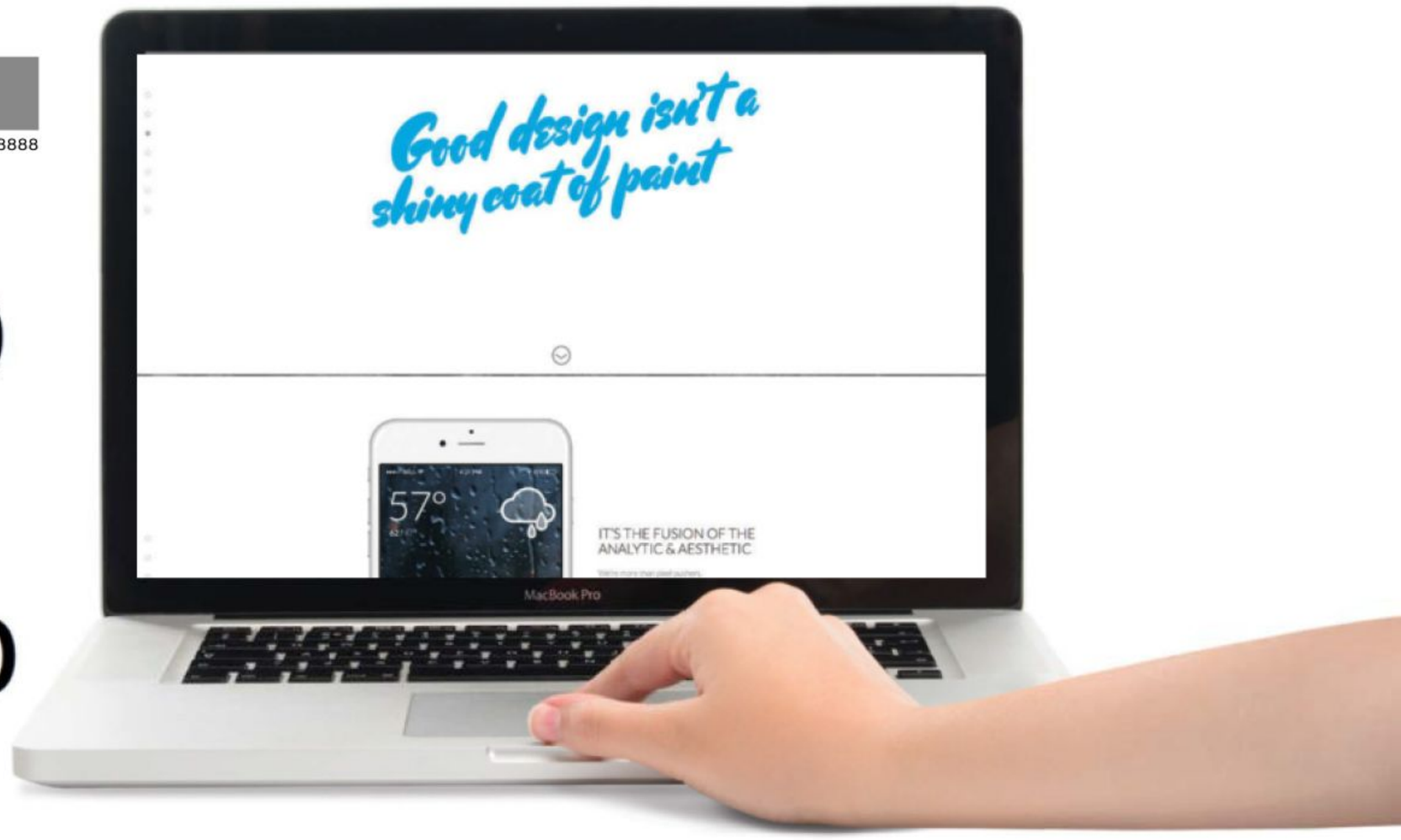
#00A8E1 #333333 #81B9E9 #555555 #C02327 #00BC70 #888888

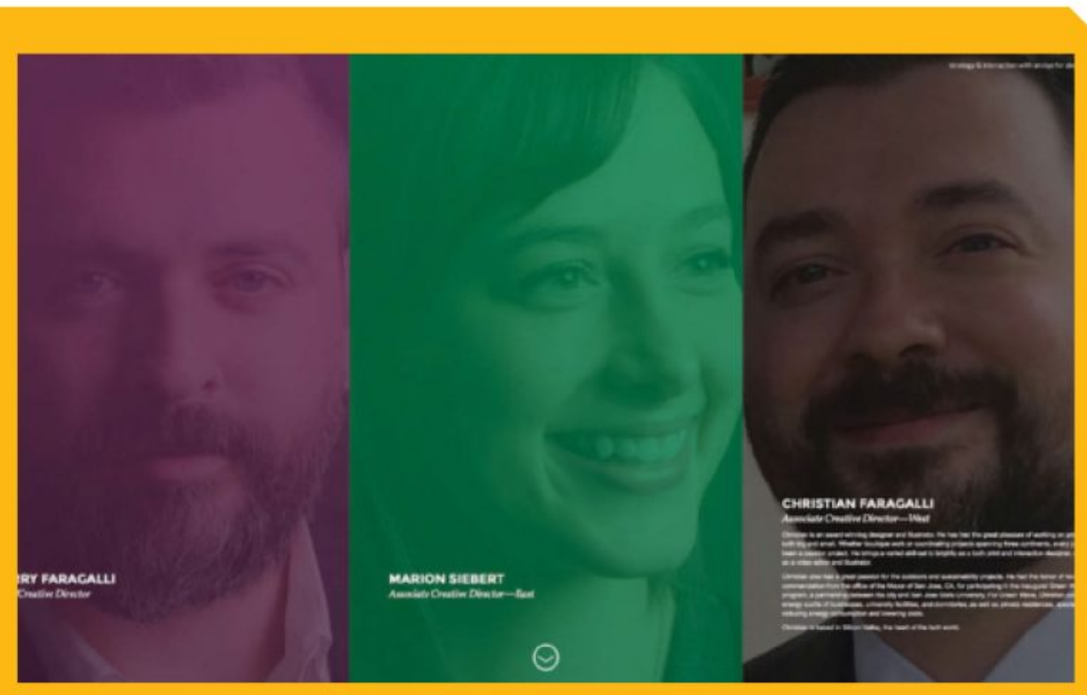
abcABC
1234567890

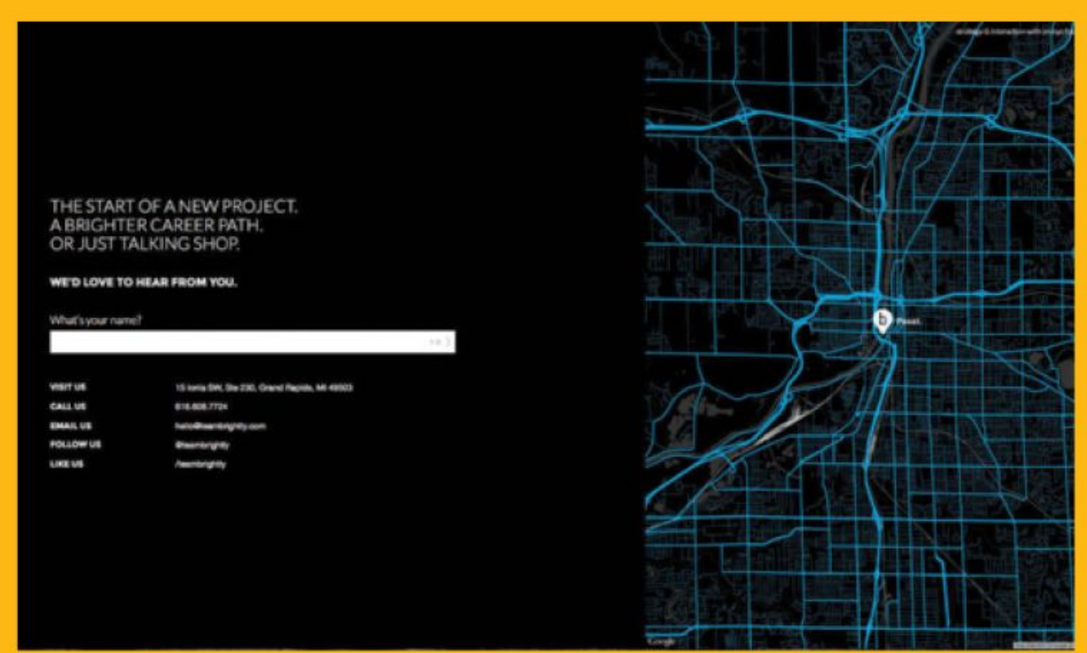
<Above>
• Abril Fatface by TypeTogether, available on Google Fonts is used for the down arrow icons

abcABC
1234567890

<Above>
• Montserrat by Julieta Ulanovsky available via Google Fonts kicks off the typography on brightly







<Top left>
• Short looping video clips of the team at work provide full-motion backdrops for describing brightly's work process

<Bottom left>
• Project case studies are frequently illustrated with bold, crisp device shots showing websites and apps in action

<Top right>
• Each member of the brightly team is introduced using colourful, oversized portraits with expandable biographies

<Bottom right>
• The site boasts one of the coolest reskins of an embedded Google Map on its stylishly minimal contact page

Swing open a structured animation with CSS

Justin Gauthier, UX designer at brightly explains how the Swiss Army knife on teambrightly.com was "crafted" using a combination of JavaScript and CSS

1. Exported assets

The technique we will be exploring in this tutorial, based on the team brightly website, starts with exporting the three, semitransparent knife blade PNG images in their final animation states. You can find these, and everything else you need, on the aforementioned filesilo page. Exporting the PNG images will mean less code is required to correctly place the images in the browser at the final stage. The knife handle should be exported in its starting state.

2. Write a specific structure

Each blade is absolutely positioned relative to the handle. Since the knife as a whole rotates, we use a knife-container for the whole element. CSS transformations for rotating the knife are applied here, while supporting heading elements fall outside this rotating . Then wrap everything in a "Stage" to provide scope to the animation and specifically style elements without affecting other site styles. For the best final result, each blade will need a unique class name since there are multiple positions, with each blade moving independently:

```
HTML:
<div id="Stage">
<h1><span>we're</span> Crafters
<span>of</span> Clarity</h1>
<div class="animation-container">
<div class="knife-container">
<img src="img/swiss-army-knife.png" alt="We're
Crafters of Clarity" class="knife">
<img src="img/pen-blade.png" alt="Content
Strategy" class="pen">
<img src="img/glass-blade.png" alt="Research"
class="glass">
<img src="img/arrow-blade.png" alt="UX
Strategy" class="arrow">
</div>
<h2>We have the tools to solve any problem</
h2>
</div>
</div>
```

3. Position the elements

Next we add CSS transformations to position the knife handle in its final state. Once the handle is in its final state, you can position the blades where they need to be by using -transform rotate and -transform-origin. After this is applied, you can see if your HTML structure is working by seeing whether the blades are rotating with the knife handle. Then apply absolute positioning to the blades and place them in their desired position.

```
CSS:
.knife-container {
position: relative;
z-index: 2;
width: 100%;
height: 190px;
/* Vendor prefixes here */
transform-origin: 71% -2%;
/* Vendor prefixes here */
transform: rotate(-30deg);
}
.knife-container img {
position: absolute;
top: 0;
left: 0;
z-index: 1;
}
.knife-container .knife { z-index:2; }
```

4. Define the starting point

For animation start points we add closed CSS classes for each individual blade. For the next step, you'll have to move the rotate transform from .knife-container into a separate .knife-rotate class. These classes will then be used in the JavaScript function to launch and restart the animation. Use the same CSS -transform rotate and -transform-origin to place the knife blades into closed states, and -transform scale to reduce the size and hide the blades behind the handle.

```
CSS:
.knife-container .glass {
top: -279px;
left: -114px;
/* Vendor prefixes here */
transform-origin: 100% 100%;
}
.knife-container .glass-closed {
/* Vendor prefixes here */
transform: rotate(123deg) scale(0.72);

}
.knife-rotate {
/* Vendor prefixes here */
transform: rotate(-30deg);
}
```

5. Apply CSS transitions

After you have defined the start and end points of the animation, you can then continue by adding the transition effects to the blades and the knife-container to smoothly animate from beginning to end. Since the blades flip out sequentially, you should apply a CSS -transition-delay in addition to the CSS -transition to create the effect. Use the following code for this step in the tutorial:

```
CSS:
.knife-container .arrow, .knife- container .
glass, .knife-container .pen {
/* Vendor prefixes here */
transition: 0.55s ease-in;
}
.knife-container .glass {
top: -279px;
left: -114px;
/* Vendor prefixes here */
transform-origin: 100% 100%;
/* Vendor prefixes here */
transition-delay: 0.2s;
}
```

6. Set the stage

Once the transition effects are added to the knife handle and all of its individual blades, you can now continue by adding the closed CSS class to the blades in the HTML. This class is necessary because on page load, the knife should be closed:

```
<img src="img/pen-blade.png" alt="Content
Strategy" class="pen pen-closed">
<img src="img/glass-blade.png" alt="Research"
class="glass glass- closed">
<img src="img/arrow-blade.png" alt="UX
Strategy" class="arrow arrow- closed">
```

7. Switch the classes

Now it's time for our final step in this tutorial. Once the stage is set, you can then write a JavaScript function to be called on page load. The function simply removes the closed class on all of the blades and this then adds the knife-rotate class to the knife container, which rotates the knife. Once you have completed this step, you're all done! You can try inserting this particular technique in your own website or projects, or simply use it to tinker and experiment with your coding. We hope you enjoy using it in your future projects!

```
JavaScript:
function openKnife() {
$(".knife-container").
addClass("knife-rotate");
$(".animation-container h2"). addClass("hide-
caption");
$(".pen").removeClass("pen-closed");
$(".glass").removeClass("glass-closed");
$(".arrow").removeClass("arrow-closed");
}
```

DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/bks-894

Create a 3D navigation menu with HTML

Learn how to use 3D page elements to reveal content on hover

The amount of space available for websites can often be limited, especially where the website is being used on a small screen. With the different information components in your design competing for screen space, it is important to make sure that you give each element the most appropriate placement in the design of the page or the screen.

A more recent addition to the CSS specification is the use of 3D elements. We are all familiar with the default 2D presentation of webpage elements, in which we can position them on the page by using their horizontal and vertical coordinates; the 3D presentation features takes this one step further by allowing elements to be presented with depth perspective, which can then be used for rotation and making the elements appear further into the distance.

This tutorial will look at how the use of 3D presentation can be used to the advantage of user experience design of websites and web apps by allowing additional information to be expanded from the same space without the need to scroll or load a new page. Our example uses 3D page elements that allow users to hover or tap to access additional information and navigation links – similar to how game cards are presented.

To keep this tutorial uncluttered, we will try to avoid using browser vendor tags such as -webkit. If you require compatibility with older versions of Safari and Firefox, you can simply add these for various different CSS commands such as transform, perspective and backface-visibility.

1. Initiate HTML page

As with any webpage you make, there is a requirement to define the page's HTML, head and body structure. You can use the head section to contain any meta data such as the page title for the sake of SEO and accessibility that you may need.

```
<html>
<head>
<title>3D Card Navigation</title>
** Step 2 goes here
</head>
<body>
** Step 3 goes here
</body>
</html>
```

2. Load resources

The <head> section of your HTML is used to contain links to resources you are using. Our example will contain a CSS file for styling and a JavaScript file for detecting when the user taps on the navigation cards.

```
<link rel="stylesheet" type="text/css"
href="styles.css" />
<script src="code.js"></script>
```

3. Card container

The easiest way to ensure consistency and simplicity to control all elements of the 3D navigation card is to place them inside a container. Two types of container are used – the main container will control the origin for 3D rotation while the card container will contain the content elements that will display relative to the card.

```
<div class="container">
<div class="card">
** Step 4 goes here
** Step 5 goes here
</div>
</div>
```

4. Front content

The card will have two sides – we place a container for each of these inside the card container, with each container having their content placed within them. The front card will have a style added to the main element that sets the background image to use.

```
<div class="front" style="background-
image:url(img/burger.jpg)">
<h3>Burger</h3>
</div>
```

5. Back content

This element will appear when the card is flipped. Like the front element, it is a container for any content that is to appear at the back. Using a container like this allows CSS to rotate everything from the same place.

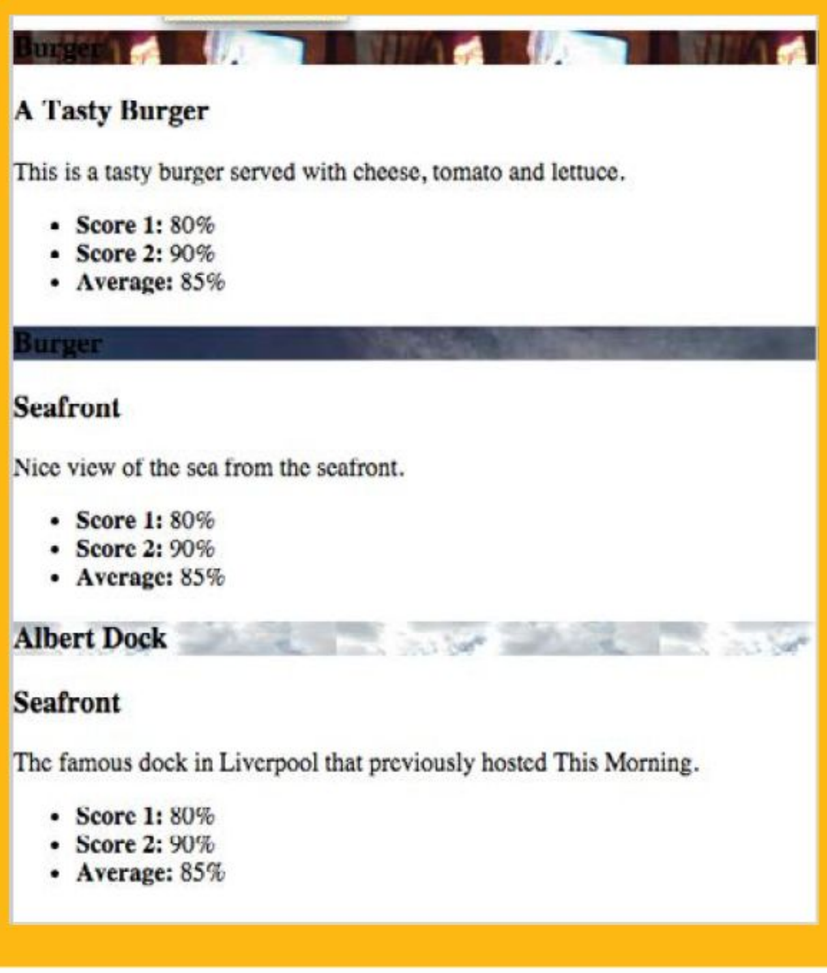
```
<div class="back">
 *** Step 6 here
</div>
```

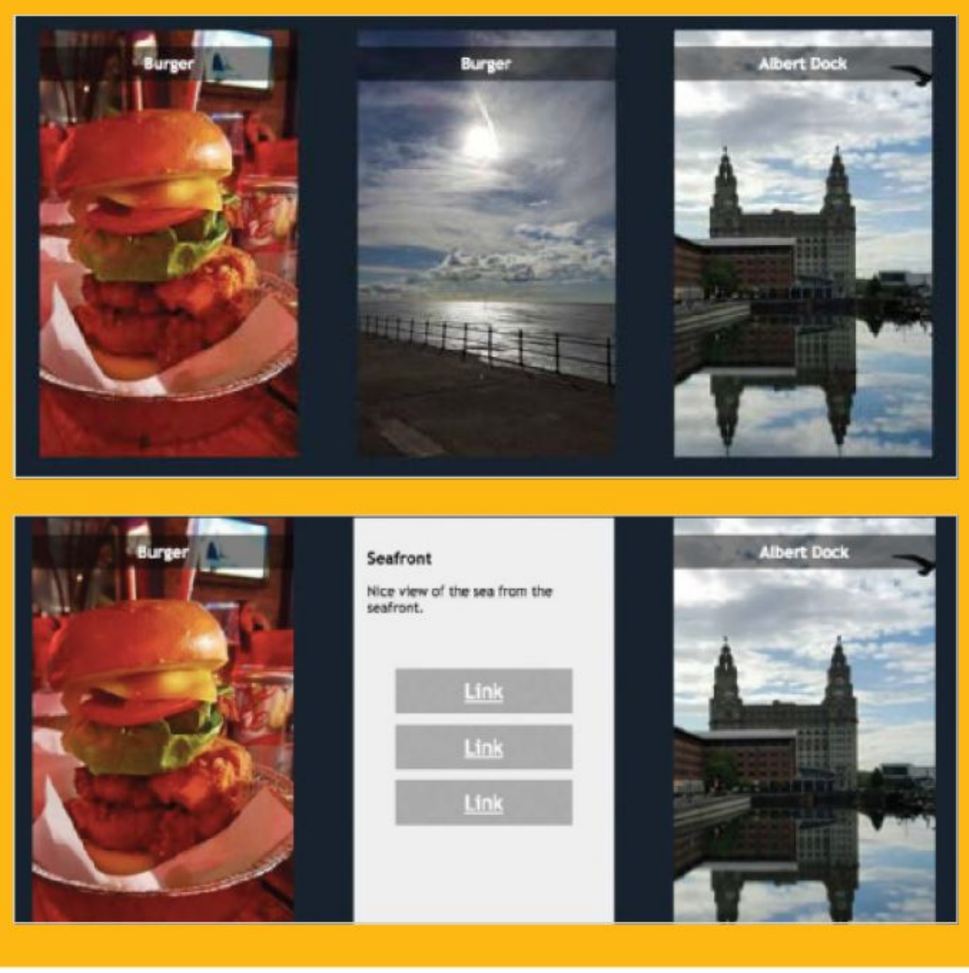
6. Content inside

The content to be placed inside the front and back is just regular HTML. We will add some text descriptions and navigation links to the back content container inside the card. Upon completion of this step, you can now repeat Steps 3 to 6 to add new navigation cards.

Better use of space

Using 3D elements for user experience design can allow you to include more information in limited space without disorienting the user by taking them to a different part of the page.





<Top left>
• Here, the basic page and element styling in place, with front of card elements showing by default

<Bottom left>
• Presentation of how one of the cards look when it has been flipped to reveal the reverse side content

<Left>
• The full HTML content is now ready for styling with CSS and interaction listeners via JavaScript to be applied

```
<h3>A Tasty Burger</h3>
<p>This is a tasty burger served with
cheese, tomato and lettuce.</p>
<nav>
<a href="#">Link 1</a>
<a href="#">Link 2</a>
<a href="#">Link 3</a>
</nav>
```

7. CSS initiation

With all of your HTML set up, the next step is to initiate the CSS. The page is already expecting to load a file called 'styles.css', so create this file and add the following initial CSS for the standard page styling:

```
var img = $(document.createElement("img"));
$(img).attr({
"src":this.url,
"data-index":count,
"title":this.title,
"alt":this.description
});
//.. code from next step goes here
```

8. Container definition

The first of the containers is used to set the foundations from which any 3D calculations are performed from.

Different effects

Other effects can be achieved with 3D rotation by changing settings such as the transform-origin and perspective. Add and change these values to see what effects you can produce.

This includes setting the position, display mode, initial rotation, margin and also the view perspective – ie the distance from the viewer. We will use a display of inline-block to allow multiple cards to be placed next to each other.

```
.container {
display: inline-block;
transform: rotateY(0deg);
perspective: 500px;
margin: 2em;
}
```

9. Container interactions

Interactions will be triggered whether either a :hover is detected on the main container or a hover class has been applied to the container. The reaction to these will be a transformation that rotates the inner contents by 180 degrees around the y axis.

```
.container:hover .card,
.container.hover .card{
transform: rotateY(180deg);
}
```

10. Set card size

We can set the size of the navigation card by simply setting the size of the foundation container and the inner content containers. In our case, we used pixel-based measurements but if you wish you can choose to experiment with percentages.

```
.container,
.front,
.back{
width: 300px;
height: 500px;
}
```

11. Card container

The next container to define rules for is the card container, which is used to activate 3D transformation rules, animation transition time and also to set positioning to relative so that any absolute positioning is relative to the card – and not the main screen.

```
.card{
position: relative;
display: block;
transition: 1s;
transform-style: preserve-3d;
}
```

12. Front and back containers

The front and back containers will be used to contain content that are to be shown specifically when the front or backside of the card is visible. We need to make sure that the backside of these elements are not visible when they are in reverse. We also need to position them at the top left of the card.

```
.front, .back {
backface-visibility: hidden;
position: absolute;
top: 0;
left: 0;
}
```

13. Content box sizing

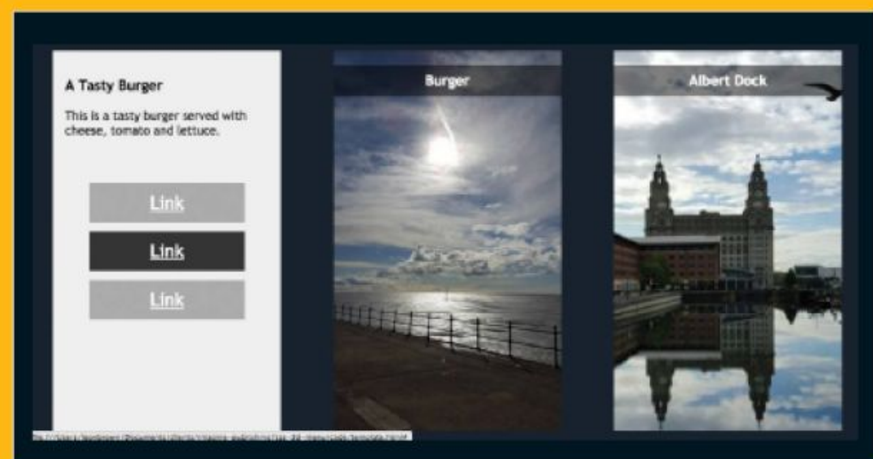
In addition to the previous content container settings, we also want to ensure that any future padding or border additions don't make their sizing out of sync with what we've already set up. We can achieve this by telling them to include padding and border size as part of their width and height settings.

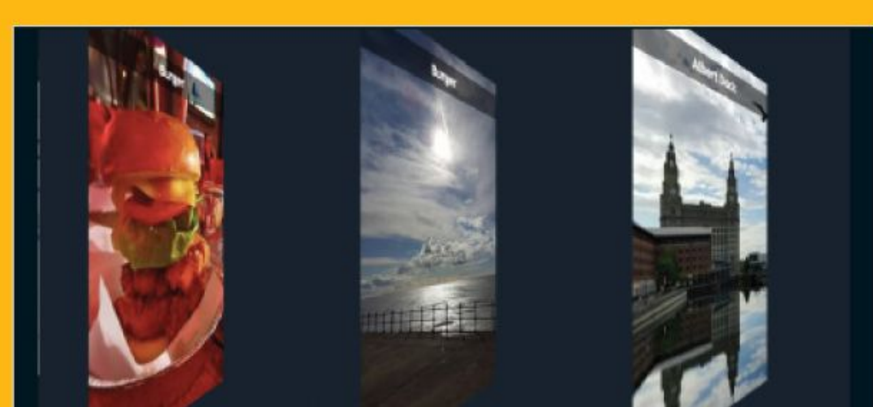
```
.front, .back {
```

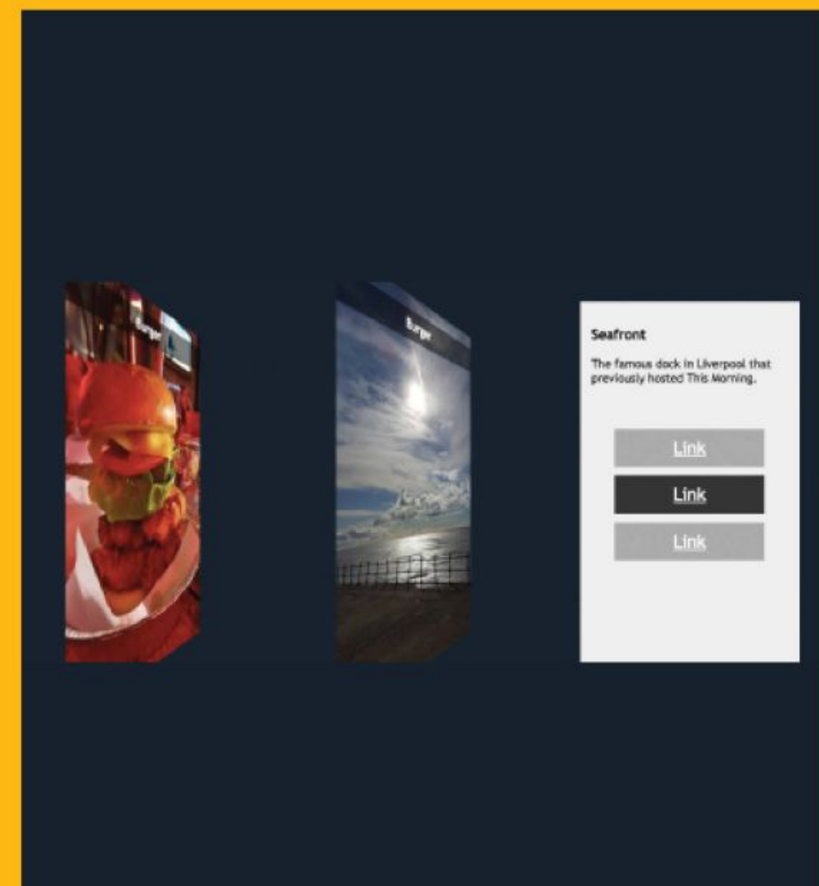
<Top left>
• Navigation elements change colour when the mouse pointer is placed over them – a way to indicate that they can be clicked on

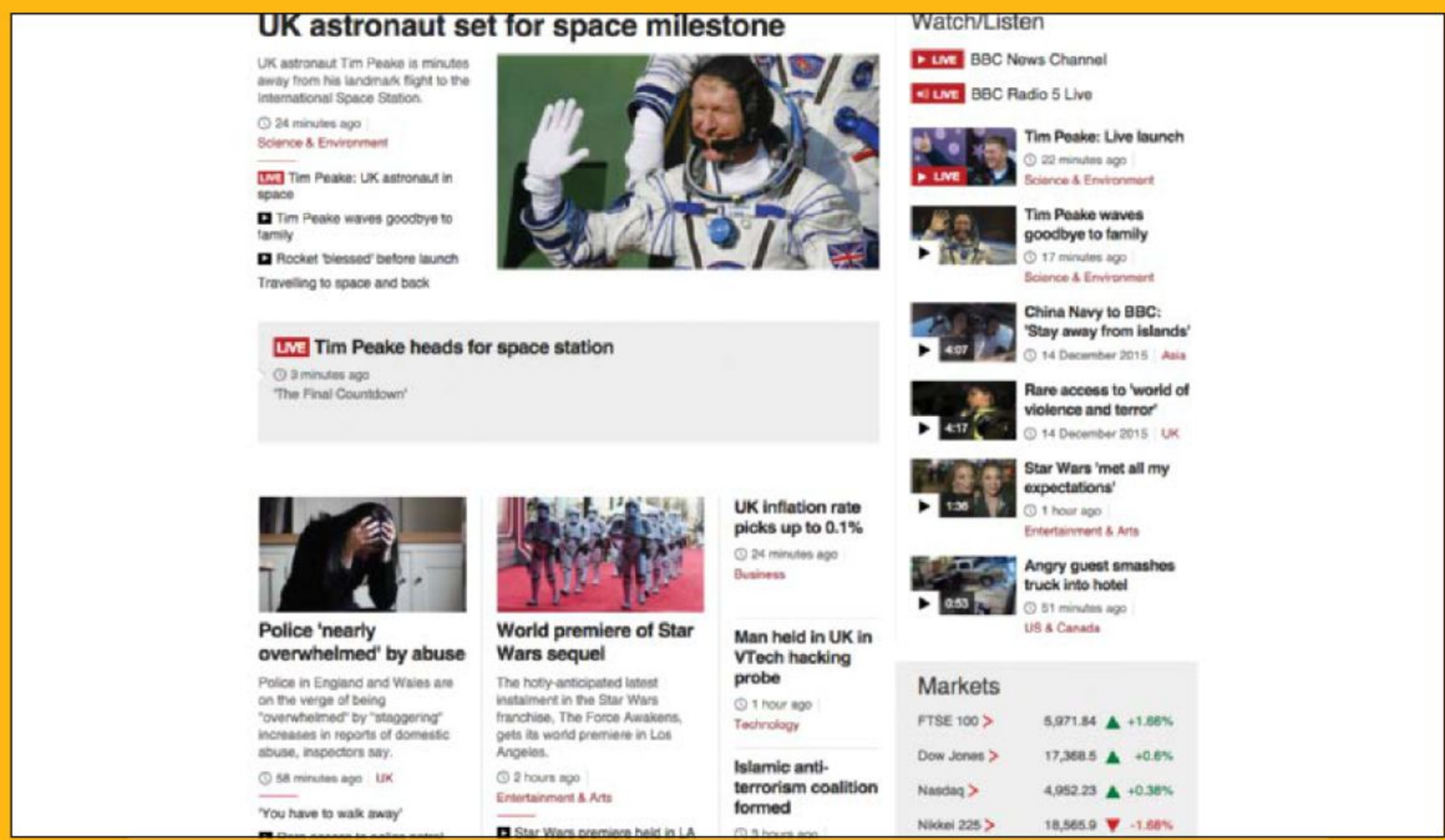
<Bottom left>
• Elements appearing with 3D rotation applied to them. No fancy calculation code – just CSS formatting

<Right>
• An example of card elements rotated with selected card showing full reverse with navigation and content









Use of space

Our example shows how 3D cards can be used to present basic text content and navigation links, but the concept could be used for more than just these types of content. Other content that cards can be used to reveal include video, charts, and interactive applications. Links can also be used to activate the presentation of a light box, allowing you to make your website more convenient for any features where users will not want to leave your webpage when accessing information from links, preventing any early exits.BBC News is an example of a news website that contains a lot of information, making it difficult for users to identify whether they want to read further without loading a new page – not good for users on mobile data plans.

```
box-sizing: border-box;
}
```

14. Front image settings

The front of the card will have its background image set to cover the full size of the element. It will also need to guarantee its placement on top of the back element, hence a z-index set to 2. An initial rotation of 0 degrees is also set for displaying content at the front.

```
.front {
z-index: 2;
transform: rotateY(0deg);
background-size: cover;
}
```

15. Front title

The front of the card makes use of a <h3> title tag to present the introduction title of the card. The title will have a semitransparent black background with white text. We want to make sure that the styling for this is specific to <h3> tags inside of the front class.

```
.front h3{
background: rgba(0,0,0,0.5);
color: #fff;
text-align: center;
padding: .5em;
}
```

16. Back card

The presentation of the back of the card will have a plain grey colour background and will require some padding to ensure that there is some spacing between text and the edges of the card. The element will need to be initially rotated by 180 degrees.

```
.back {
padding: 1em;
transform: rotateY(180deg);
background: #eee;
}
```

17. Navigation container

The navigation links of each card are placed inside a <nav> container. This means we can apply styling to their container – in this case to set the font size, padding and the display mode to ensure that links appear separate to the main content.

```
nav{
display: block;
padding: 2em;
margin: 0;
}
```

18. Navigation links

Navigation links will appear in darker grey boxes with clear white text. Each element will use padding to make the boxes appear more prominent amongst everything else and margin to make them appear more separate to each other.

```
nav a{
display: block;
list-style: none;
background: #aaa;
color: #fff;
font-size: 1.5em;
margin-top: .5em;
padding: .5em;
text-align: center;
}
```

19. Navigation hover

Changing the colour of navigation options that are being hovered over is a good way to indicate to the user that they can select the item. This is achieved by using the :hover property on the navigation item.

```
nav a:hover{
background: #333;
}
```

20. JavaScript get cards

We now need to add some JavaScript to activate detection of taps on the cards when used on mobile devices. Create a file called 'code.js' and add the following code to fetch all elements with a class name of 'container' when the page loads:

```
window.addEventListener("load", function(){
var containers = document.
querySelectorAll(".container");
** Step 21 code here
});
```

21. Touch start listener

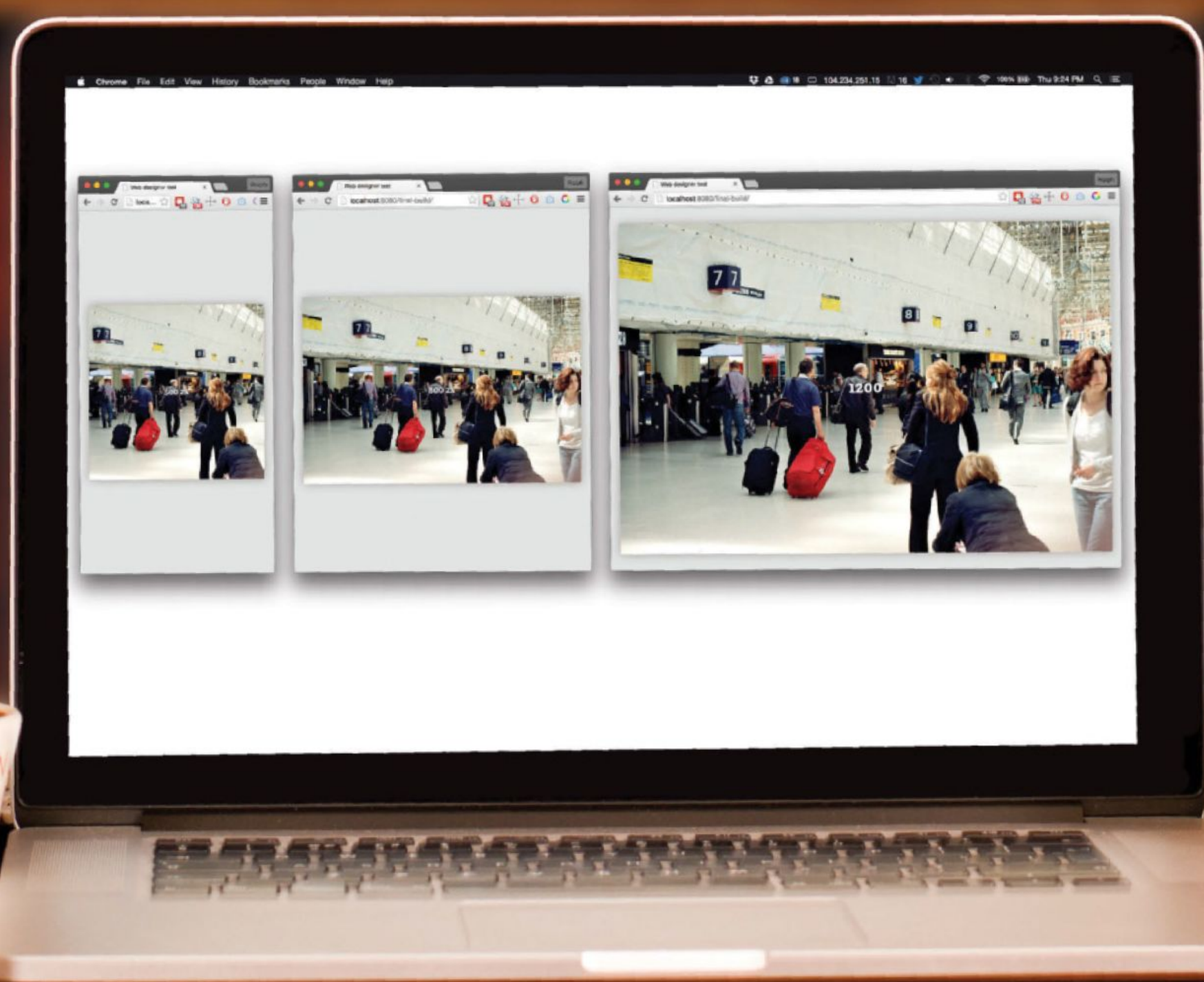
With JavaScript now having all of the container elements, it can now loop through each of the element and apply a listener for each time one of these elements are touched. Our code will toggle the addition or removal of a "hover" class – of which the presentation rules were defined in Step 9.

```
for(var i=0; i<containers.length; i++){
containers[i].addEventListener("touchstart",
function(){
this.classList.toggle('hover');
})
}
```

DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/bks-894

Master responsive image techniques

Learn how to use <picture> and srcset, and bring art direction to responsive images



Ever since responsive web design became a thing designers and developers have wanted better tools to manage imagery across many different clients and viewports. With the advent of the srcset attribute and picture elements, this is now achievable in modern browsers and responsive images are even starting to become possible on mobile devices, with Android 4.4.4 offering good native support. For aging browsers, Scott Jehl's picturefill JavaScript polyfill can be used to plaster over the cracks – although network performance will still be an issue.

The pinnacle of this new feature set from HTML5 is the picture element, enabling creative teams finite control of how images are presented to their web browser – creative teams are finally able to give art direction on a per-client basis. This can be something that works as simple as serving different aspect ratios to mobile devices, to serving entirely different image compositions for different viewports.

This tutorial walks through the process of crafting flexible images the old way, image swapping using srcset for different viewports and display capabilities, fully responsive images using picture, using picturefill the right way, and finally bringing all of these components together to produce an interactive demo that exploits the best that a picture has to offer.

1. Basic image markup

Marking up a standard image hasn't changed for a long time. The src attribute specifies the location of the image. The alt attribute is for supporting text that should be useful if the image can't be found, or for the visually impaired who use screen readers.

```
<img src="image.jpg"
alt="text for screen readers" />
```

2. Basic responsive images

Responsive images will need to shrink with their containers. The code that we're using below is a bit counterintuitive because it suggests that it just limits the width of the image, but in effect it implies the image should be flexible in its width. These images are really flexible rather than responsive.

```
img
{
max-width: 100%;
}
```

3. Change resolutions with srcset

Modern browsers can make use of the srcset attribute that enables the browser to make a decision with regards to what image to actually display. The width specified after each additional image path corresponds to the width of the image.

```
<img src="small.jpg"
alt="text for screen readers"
srcset="medium.jpg 600w,
large.jpg 1200w" />
```

4. See srcset in action

Here's a basic implementation of srcset on jsfiddle as a live demo (**jsfiddle.net/aqx9aoat**). The browser has chosen to display the 500px width image rather than the initial 1000px one. The browser is choosing which image to pick by itself here.

5. Control srcset behaviour

The sizes attribute can enable developers to control the amount of screen real estate the image will take up. The example code shown specifies that the image will take up 66 per cent of the viewport width as opposed to the preset value of 100vw.

```
<img src="small.jpg"
alt="text for screen readers"
srcset="medium.jpg 500w,
large.jpg 800w"
sizes="66vw" />
```

6. Responsive srcset behavior

The sizes attribute also takes media queries, which allows developers to change the behaviour of images depending on the viewport. Here, 66vw is used whenever the viewport is wider than 499px, otherwise the image will display with 100vw.

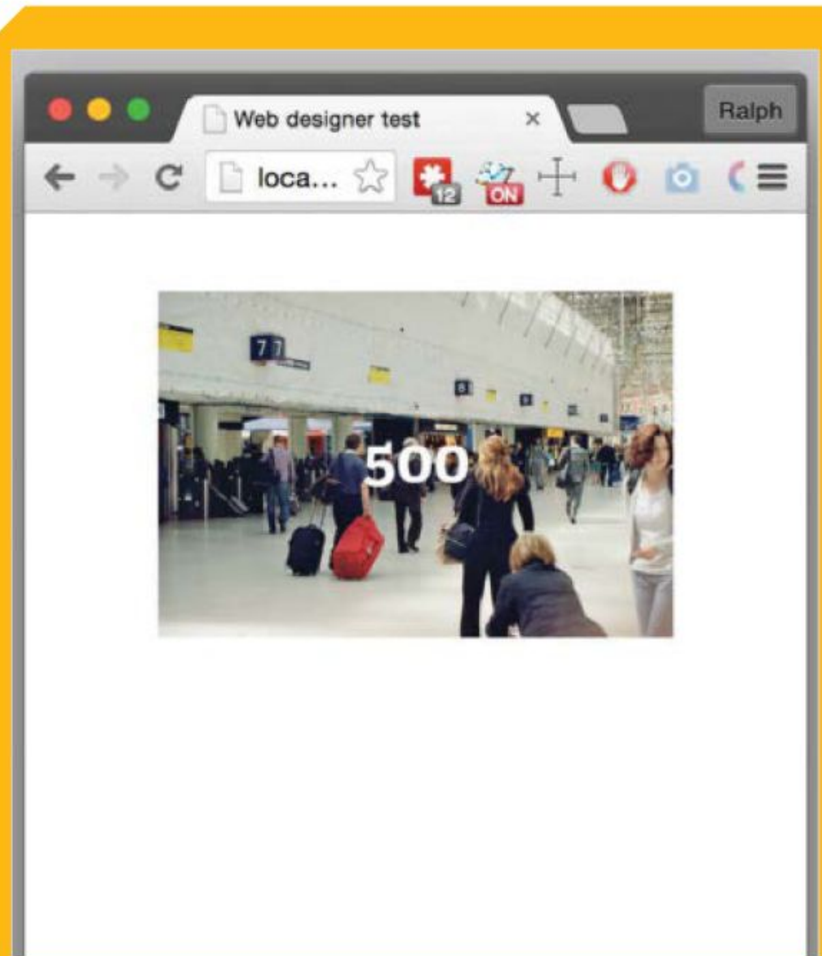
```
<img src="small.jpg"
alt="text for screen readers"
srcset="medium.jpg 600w,
large.jpg 1200w"
sizes="(min-width: 500px) 66vw,
100vw"
/>
```

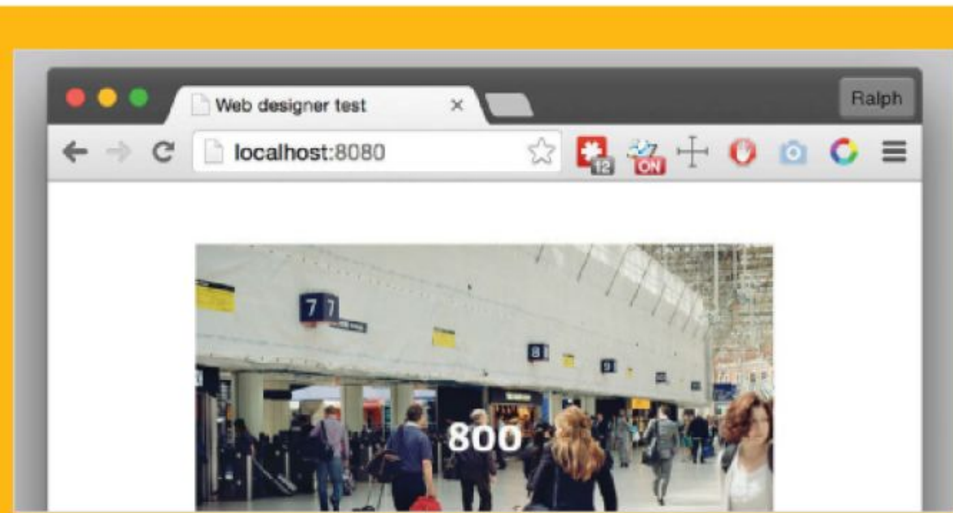
7. Retina support with srcset

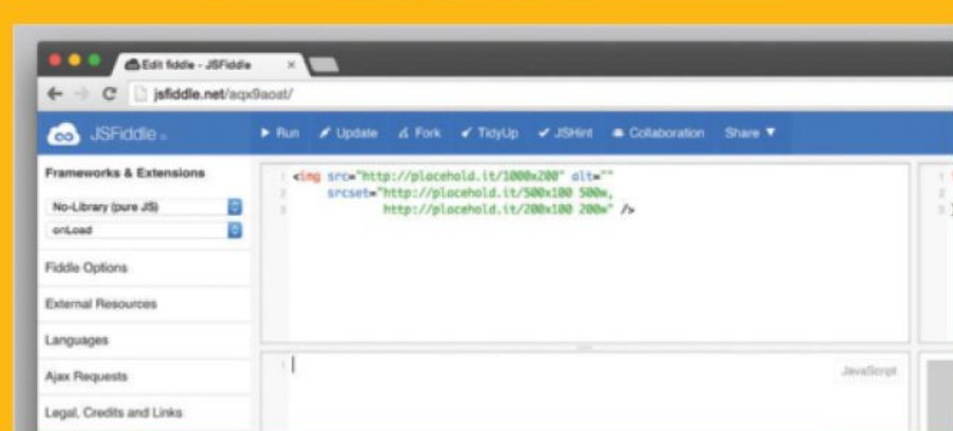
Srcset enables developers to specify the use of hi-DPI

Use calc in sizes

The sizes attribute allows for using calc() just like CSS. This enables us to have finite control that wouldn't be possible otherwise, like calc(100vw – 40px). It's handy for keeping gutters on images.







<Top left>
• When the user increases the width of their viewport, the browser will load the 800px version as it will appear less stretched

<Bottom left>
• The browser will have to decide which image to display based on the information provided which, in this example, is just the width descriptor

<Left>
• Due to the small viewport on page load the browser has chosen to load the 500px wide image, as it's smaller in dimensions and file size

imagery via the x descriptor. Unfortunately though, images with the x descriptor are fixed-width only (not flexible), which limits their usefulness. This is something that prove to be handy for making hi-res cropped background effects, however.

```
<img src="image.jpg"
alt="alt text"
srcset="imageRetina.jpg 2x,
hiDPI.jpg 3x"
/>
```

8. Responsive art direction

The picture element enables developers to specify multiple source elements and a fallback img, enabling media queries to be applied to each. The code shown below swaps out widescreen images for square ones – as this will be a better aspect ratio for mobile – at a 499px wide viewport.

```
<picture>
<source
media="(min-width: 500px)"
srcset="wideLarge.jpg 800w,
wideMedium.jpg 500w" />
<source
srcset="squareLarge.jpg 800w,
squareMedium.jpg 500w" />
```

Comment sources

With so many different sources required for each responsive image being implemented, it pays dividends to write HTML comments to help keep track of the images being sourced.

```
<img src="fallback.jpg" alt="" />
</picture>
```

9. Style picture elements

This may also appear to be counterintuitive, but the picture element can't exactly be styled in the same way as an image. The picture simply wraps an image, passing the right image path for any given media to the image element. Flexible images are also possible within picture elements, but they will first need to be marked up like the code as shown below:

```
picture img
{
max-width: 100%;
}
```

10. Starting markup

Earlier on, this tutorial briefly covered Retina image support using srcset and explained that Retina support only worked for fixed-width images. Picture can provide us with a way around that problem by making use of the media attribute. As you can see in the code below: catering for wide devices first, the fallback.jpg file will now be replaced with wide1200.png.

```
<picture class="album">
<source
media="(min-width: 801px)"
srcset="wide1200.png 1200w" />
<!-- ... -->
<img src="fallback.jpg" alt="" />
</picture>
```

11. Responsive Retina images

In this additional source the media attribute specifies that these images are only to apply to viewports between 501px and 800px. Srcset will then load the image that is found to be the most suitable to the client with either the 1x or 2x versions of wide800.

```
<source
media="(max-width: 800px) and (min-
width:501px)"
srcset="wide800.png 800w,
wide800-2x.jpg 2x" />
```

12. Markup for responsive

This final source caters for the smallest viewports and implements the same technique as the one in Step 11. The downside to using this approach is that a lot of markup is needed for one responsive image – each version will require its own source tag as well as a set of media queries.

```
<source
media="(max-width: 500px)"
srcset="square500.png 500w,
square500-2x.jpg 2x" />
```

13. Style picture elements

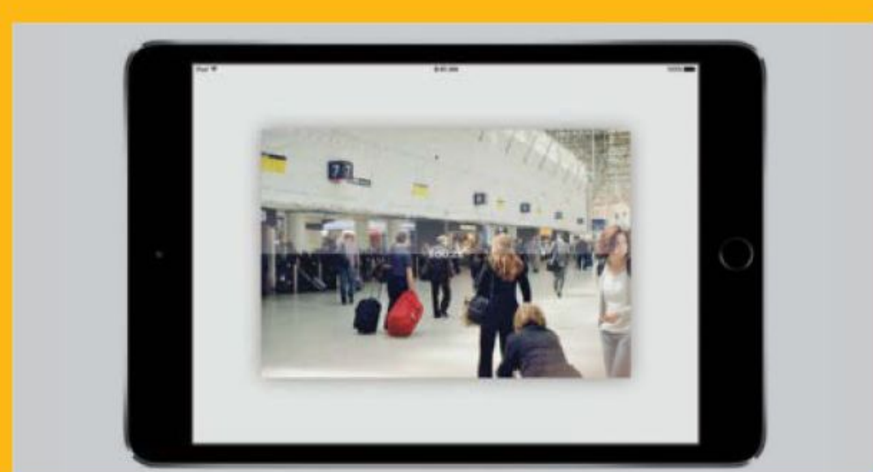
Ultimately, when it comes down to it, picture elements are essentially glorified spans – a display:block later and then you will be able to add whatever styles that you want. The code shown below uses calc and margin to add gutters to the picture, as well as show a nice, attractive box shadow.

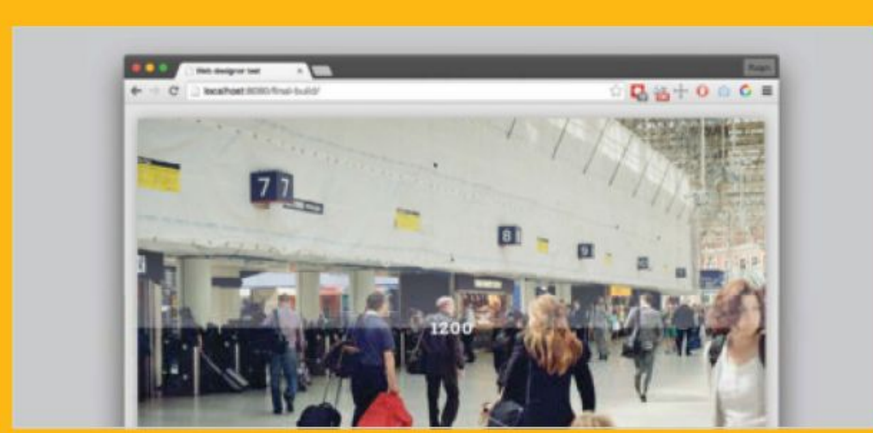
```
.album {
display: block;
width: calc(100%   40px);
margin: 0 auto;
box-shadow: #A39A9A 0 0 30px;
position: relative;
}
```

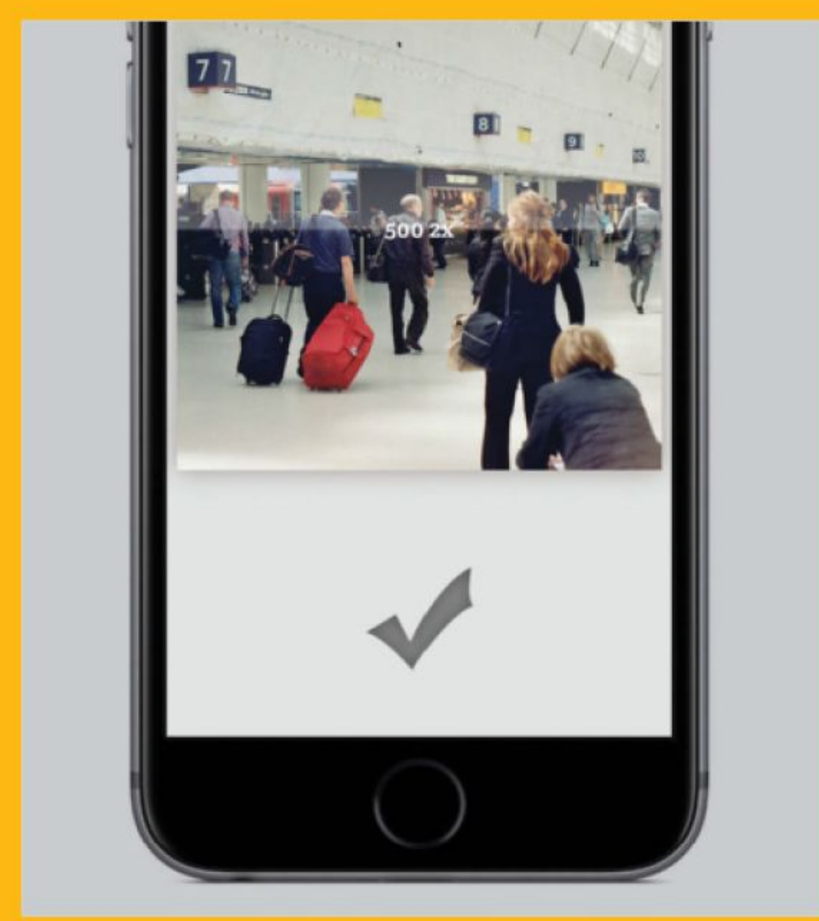
<Top left>
• The medium Retina image should be served to higher-end tablet devices with Retina or hi-DPI displays, like Apple's iPad, when they eventually support it

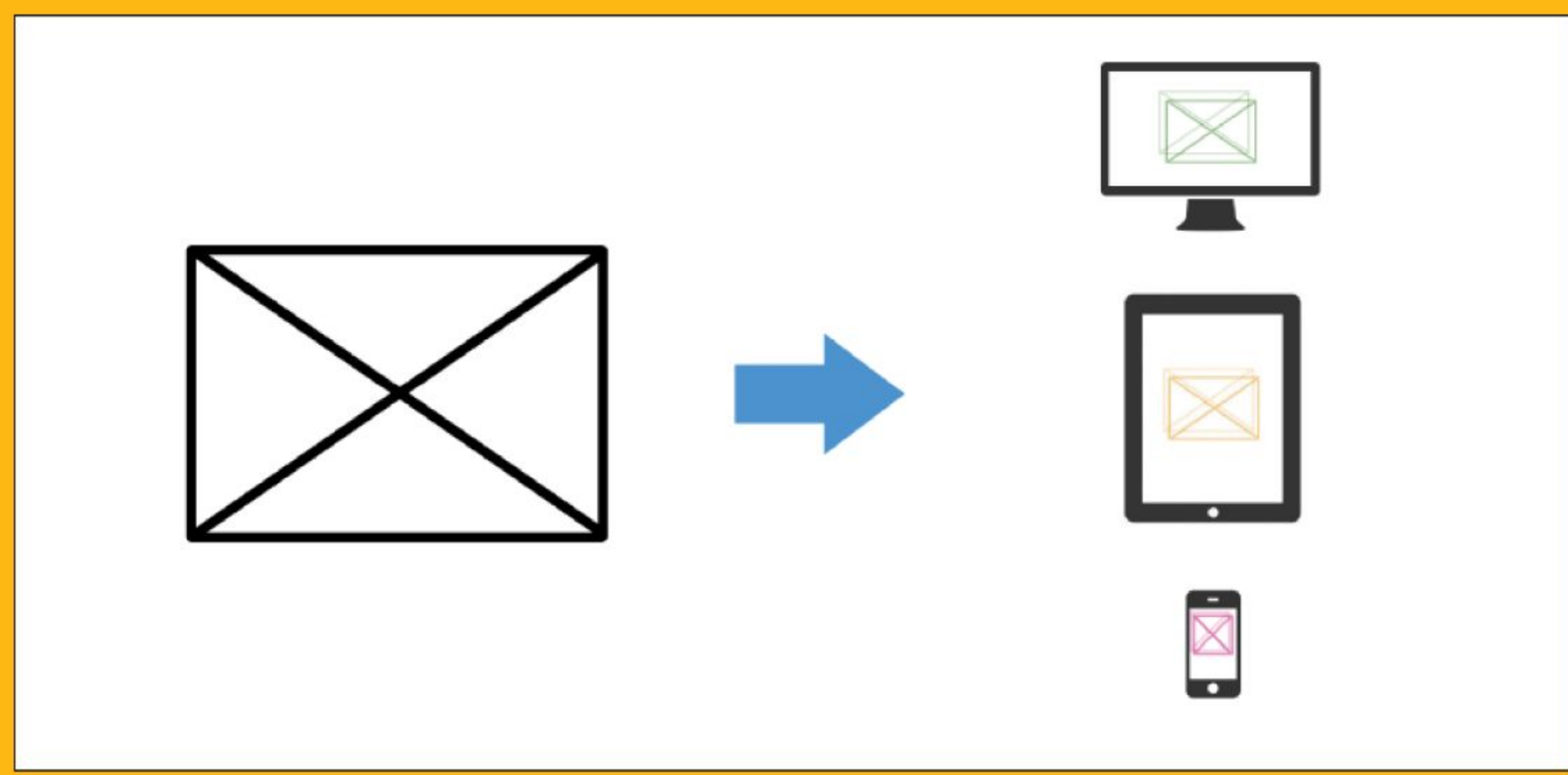
<Bottom left>
• This client gets the wide image. We could have supplied a 2x variant as part of this step as a lot of modern clients are on 2x plus displays

<Right>
• Widescreen images on mobile viewports can get very small when scaled proportionately. The picture element allows us to serve a cropped image









Using the device's hardware

Ionic uses Cordova to hook into the operating system's native functionality. Ionic created ngCordova, a collection of over 70 directives which can be included to make this easier to work with Angular. It includes everything from Touch ID integration to accessing the camera. http://ngcordova.com/docs/plugins/ has a list of plugins and how to use them. For example, if we wanted to include vibration feedback to an action we would add the module via the command-line and simply inject the service into our controller. Some of these APIs are platform-specific but there is general support for iOS, Android, and Windows Phone.

14. Extended picture styling

Because the picture element behaves like a regular span, the element can be extended with :before and :after in CSS to apply effects over the top of images. Below this, a gradient is overlaid the image. View the full source of this demo on FileSilo.

```
.album:before
{
content: '';
position: absolute;
width: 100%;
height: 100%;
background:linear-gradient(...);
}
```

15. Image styling

Picture essentially progressively enhances img elements. CSS can still target the image, just as if it's inside a regular container. Images inside picture elements are available for styling just like any other image.

```
picture img {
border-radius: 3px;
}
picture {
border-radius: 3px;
}
```

16. Add interactions

The process of adding interactions to picture elements is slightly different. Binding hover to the picture element ensures the hover will work as intended. However, the visual feedback still needs to happen on the image, so that's where styles should be applied.

```
picture:hover img
{
opacity: 0.5;
cursor: pointer;
}
```

> Picture allows the build team finite control over what, how, and when any given variant of an image displays

17. Add filters

With filters coming to CSS, developers are able to apply filter effects to DOM elements – this includes picture element images. The contrast of the image is slightly bumped with this filter. Naturally these filters can be applied on a per-media-query basis.

```
picture img {
-webkit-filter: contrast(1.25);
}
```

18. Centre everything

The first part of centring the image is to have the container (in this case, the page body) fill the entire viewport. This is easily achieved using absolute positioning and resetting any margins/padding to 0.

```
body
{
margin: 0;
padding:0;
position: absolute;
height: 100%;
width: 100%;
background-color: #E1E4E2;
}
```

19. Vertical centre

Flexbox can easily be used to vertically centre content of variable height. Align-items will vertically centre child elements, and justify-content horizontally will centre them. Try playing around with these in dev tools to see how they behave.

```
body
{
display: flex;
align-items: center;
justify-content: center;
}
```

20. Picture polyfill

Scott Jehl has led the development of a polyfill for the Picture element called picturefill. Simply head over to GitHub (**github.com/scottjehl/picturefill/releases**) to easily download the latest release. It was recently updated to fix some bugs, and the latest version is sure to satisfy.

21. Use picturefill

Like most polyfills, using picturefill is very straightforward. First, make sure that you reference the script in the head of the page – the async attribute is optional but it's beneficial for performance. Also you should create the element for any browsers that don't support HTML5 – but bare in mind that this isn't a required step if HTML5 shiv is already present.

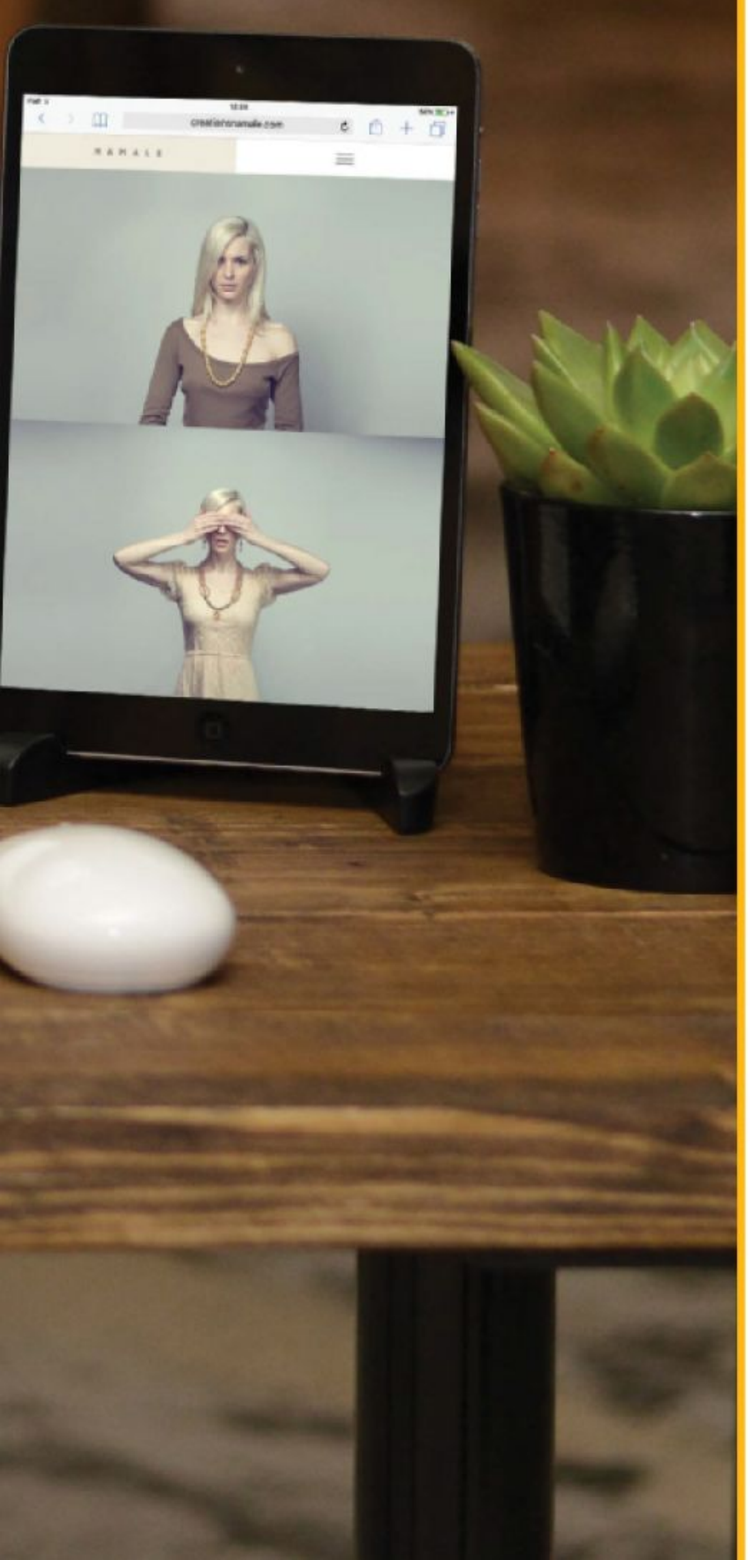
```
<head>
<script>
// Picture element HTML5 shiv
document.createElement( "picture" );
</script>
<script src="picturefill.js" async>
</script>
</head>
```

Inspiration

Namale – Joyaux Uniques

PHOENIX designed this distinctive brochure site – learn the secret behind their smooth transitions

DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/bks-894



<Left>
• Créations Namale is built around scrolling effects, that navigates through a cascade of fashion photography

PHOENIX

Designer PHOENIX
Site creationsnamale.com

If there's one UI trend that has come to characterise the desktop web over the past couple years, it is scrolling effects. Content animated and given a sense of depth on scroll, often provides an illusion of long pages when in fact only browser height is occupied. This new design for jewellery brand Créations Namale pulls a similar jQuery-driven trick that leaves you confused and scratching your head. The main page feeds a succession of fashion photography through a bordered area, almost like a viewing pane for focusing the eye. Gorgeously shot imagery then performs 50/50 splits to cleverly reassemble and invite full-width zooms on click. It's purposefully puzzling but achieved with such elegance from Montreal studio PHOENIX, thanks to a painstakingly coordinated colour palette and an almost complete dispense of superfluous text. Ideas here on creationsnamale.com have centred on the very heart of the site and the product showcased itself – the jewellery. "What inspired us for this project was the name of the company 'Namale' which means 'unique jewel' in Fijian," explains creative director Louis Paquet. "So we really wanted to create something unique to showcase the jewellery as best as possible. To do so, we opted for a very clean and minimalist look, some smooth animations and a pastel palette of colours. The website really focuses on the pictures associated within the lookbook and products in an original yet functional way."

> We really wanted to create something unique to showcase the jewellery as best as possible

HTML & CSS

abcABC
1234567890

<Above>
• Merriweather font by Eben Sorkin and available via Google Fonts styles the elegant main <h2> text

abcABC
123456789

<Above>
• CP Company by Fabrizio Schiavi and FSD styles <h3> and subsequent vertically aligned page links

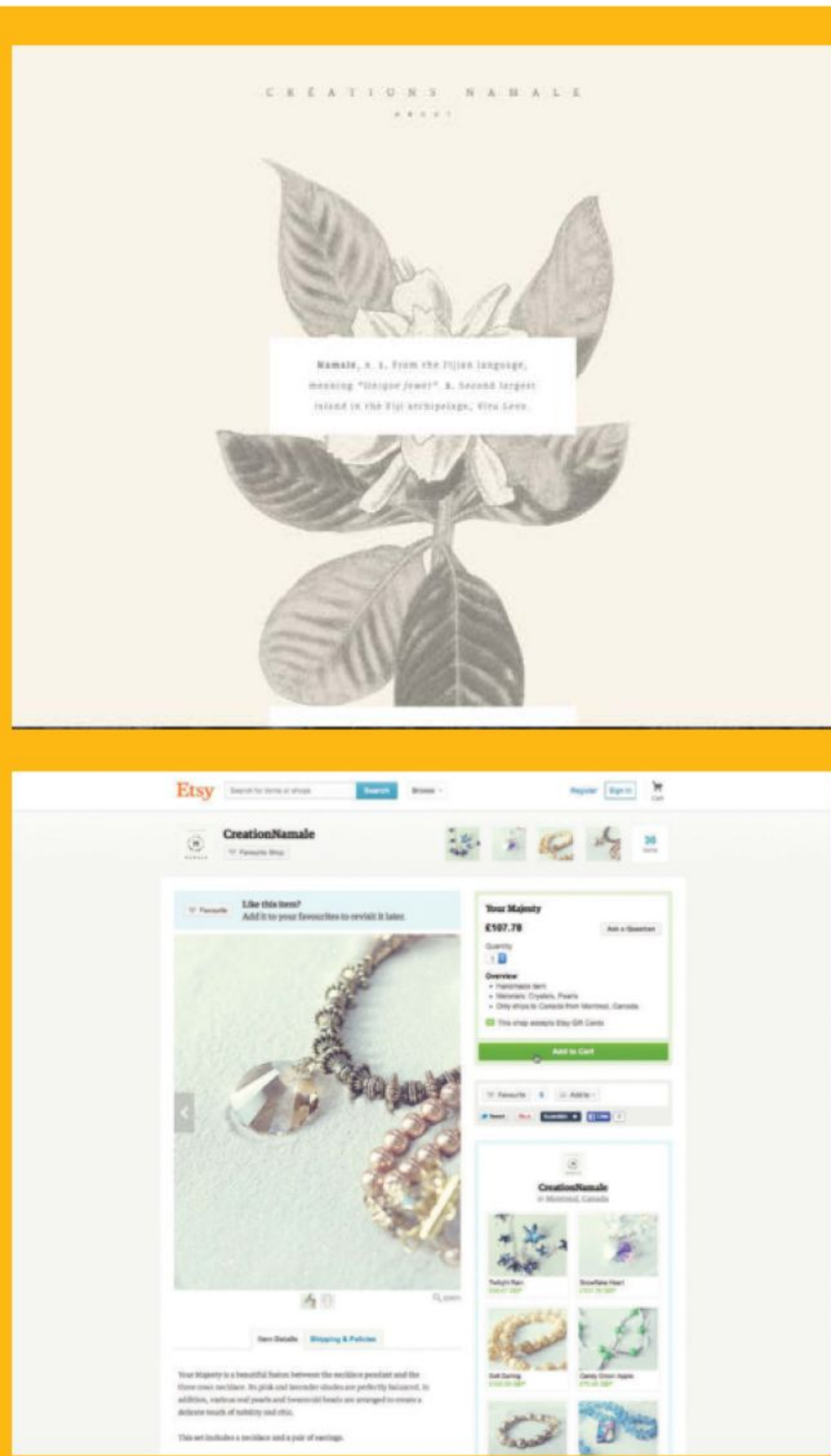
#787772 #F5F2E1 #C9D8CE #FFFFFF #F7F3E7

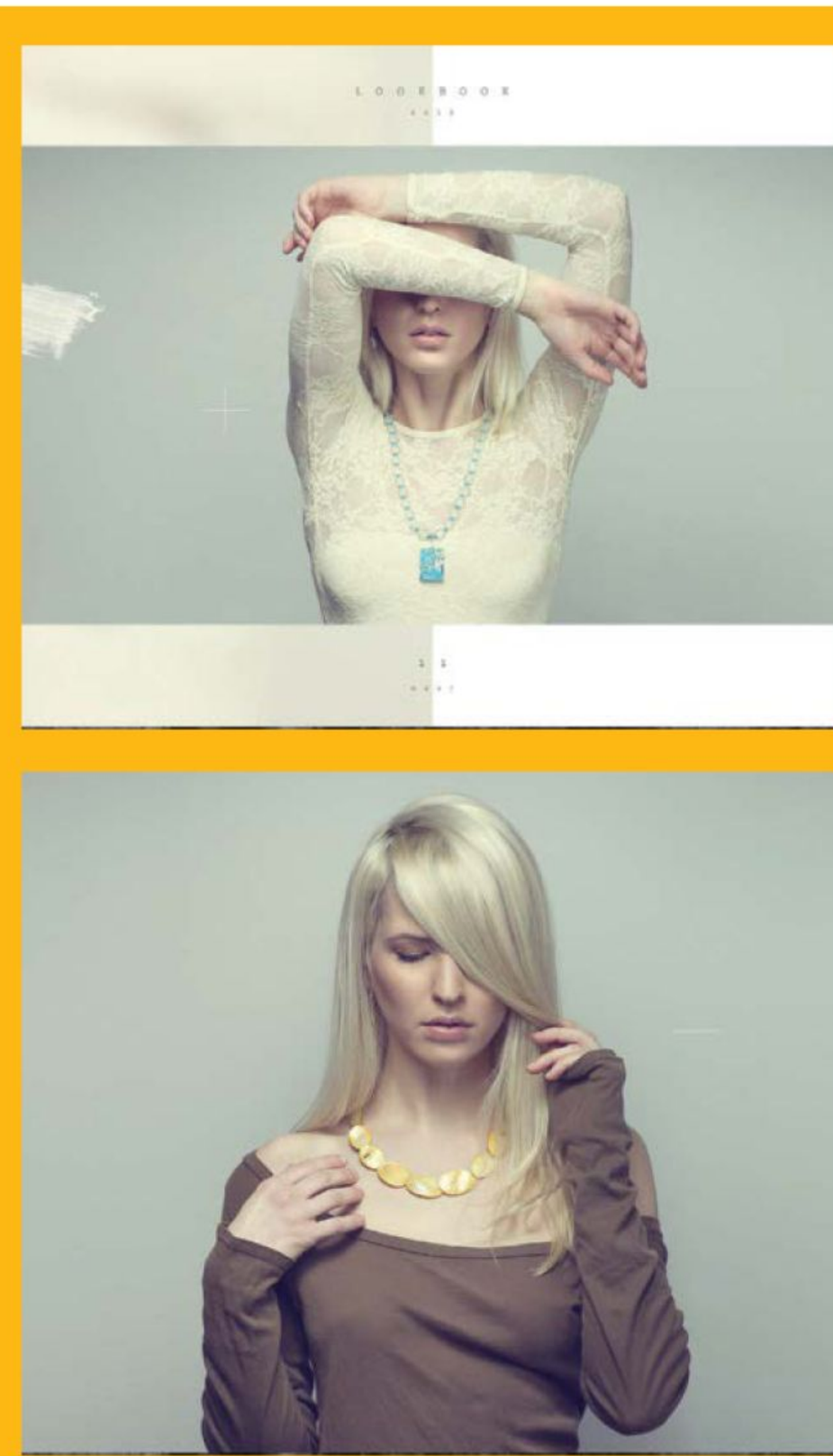
<Top left>
• The about section remains consistent to the scrolling effects and subtle typographic work employed throughout the site

<Top right>
• Hover over the main image viewer panel and a custom plus sign cursor suggests the ability to click and zoom in

<Bottom left>
• Links to products on marketplace site etsy.com enables visitors to swiftly add jewellery to their bag and buy

<Bottom right>
• Sure enough the zoom blows the current image up to fill the browser window, pulling back upon a click of the now negatively signed pointer – cool huh?





Time your CSS animations just right

The PHOENIX developers talk CSS animations and how to use custom speed curves to create spotlights

1. Timing is key

Adding motion to certain key elements in your design can make the difference between an ordinary website and a really interesting one. The interface doesn't need to be loaded with images and graphics to catch the eye, the elements present only sneed to be properly highlighted and that's where CSS animations come in play. But take note, as with a artful project like Créations Namale, timing is key.

2. The basic CSS transition

Animating an element in CSS is usually pretty simple for an experienced developer. For example, if we would want to animate the width of our targetElement when we put the mouse over it, we just give it a basic width and modify it when it is in hover state. If there is no transition, the element's width changes instantaneously, but with CSS3 we know you can control how the transition happens so that it fits your design better and adds a slick touch to the experience:

```
targetElement {
/* Sets the parameters of the transition */
transition-delay: 0s;
transition-duration: 0.25s;
transition-property: width;
transition-timing-function: ease-in-out;
/* Sets the default width of the element */
width: 200px;
}
targetElement:hover {
/* Sets the width of the element when in
hover state */
width: 450px;
}
```

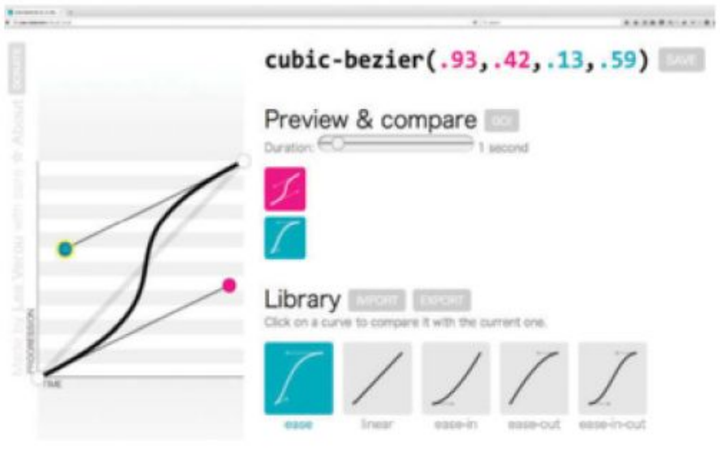
3. The timing-function parameter

The timing-function parameter already has a couple of premade speed curves that you can use, as you can see listed below. Those are good in most cases and come in exceptionally handy if you don't want to break your head creating custom effects that may even not be worth the bother in the end. But for the effect achieved with Créations Namale, we had to go a little bit further and actually create our own custom speed curves to really put specific features under the spotlight.

```
/* Use one of the following to revert to more
basic transition effect */
transition-timing-function: linear;transition-
timing-function: ease;
transition-timing-function: ease-in;
transition-timing-function: ease-out;
transition-timing-function: ease-in-out;
/* Use this one for a more customized effect
*/
transition-timing-function: cubic-
bezier(0.770, 0.000, 0.175, 1.000);
```

4. Work on your curves

It may look alien and complicated at first sight, but don't worry, you don't need to have a degree in maths to be able to tweak Bézier curves. A lot of sites are there to help you create your own timing function by playing with a visual representation of the animation. You can test your new curve directly on the site and compare it to the basic timing functions we saw in the previous step. A good prime example of this would be Lea Verou's **cubic-bezier.com.**



Testing your timings

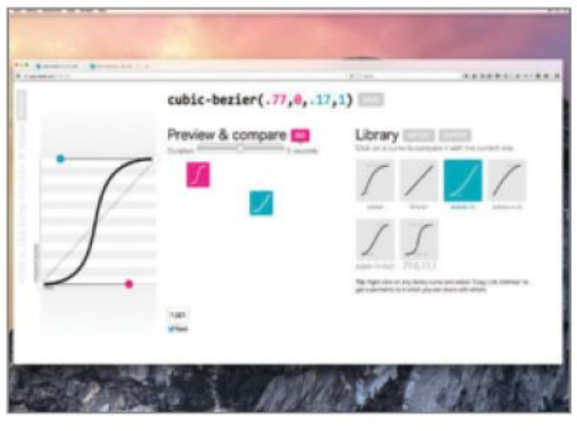
So with good timing so crucial to CSS Animation, we take a closer look at how **cubic-bezier.com** can help here

1. Set a cubic curve

This useful online tool is built for testing and comparing timing-function curves. For starters, you will find a cubic-bezier() that has its time and progression parameters set using the curve to the left. You can simply drag the blue and red handles into position to try it out.

2. Use the library

Clicking the Save button and naming adds your custom curve to the Library on the right-hand side of the screen. Here you'll also find some standard, default ease, linear, ease-in and other curves to select for comparison. These curves can then be imported or exported as desired.



3. Preview and compare

A slider selects an overall duration in milliseconds, before the GO! button previews. The two curve blocks below animate using the cubic-bezier() in pink and the library curve in blue. You can even then send your favourite as a link to friends!

> jQuery's developers established some 'best practices' which need to be respected if a plugin is to feel right to users

054 Be a jQuery code master

Learn the tricks from the experts and dominate your jQuery code

062 Image-based pop-up menu

An intuitive menu is the cornerstone of good navigation – now let's make it look good too

064 Build your own jQuery plugin

Make your own custom plugin and submit it to a repository

070 Inspiration: WeGet design

Recreate the technique behind this website's slick slide-in menu panels - we won't tell!

jQuery & JavaScript

074 Manage asynchronous tasks

Remove redundant callback functions from background tasks in JavaScript

080 Robust JS code with TypeScript

Improve your JavaScript code with a little help from Microsoft's Typescript library

be a jQuery code master

We reveal the core techniques for the library, integration with ES6 and cheat sheets for easy referencing

The current state of jQuery

IT'S BEEN TEN YEARS SINCE INITIAL RELEASE, BUT WEB DEVELOPERS STILL NEED TO KEEP AN EYE ON JQUERY

Falling victim of its own success in recent years, jQuery can sometimes be described as 'not-cool' – especially when used in the same sentence as libraries like Angular or React. JQuery doesn't include a templating language or follow a MV* pattern, yet it's still an incredibly useful library in the modern developer's toolset.

JQuery's gentle learning curve can be a great introduction to JavaScript and since conception it's become the go-to for a first 'taste' of JavaScript – a language that is leaving its browser only roots behind. The library enables complex topics to be introduced sooner and in bite-size chunks. The more experienced developer should remind themselves of jQuery's own slogan which is 'write less, do more'. This ease of use can, in the 'real' world of fast-paced deadline-driven development, be a lifesaver. How many of us can remember how to effectively traverse the DOM in vanilla JavaScript?

Simplicity shouldn't be a negative, the library is in use in over 65 per cent of the top one million websites (**libscore.com/#jQuery**) and this isn't a coincidence. The library's core is well-thought-out and the jQuery team has an incredibly methodical process to implementing new features – features that are introduced to improve the library with each release.

Being designed from day one with extensibility in mind has meant that in the last ten or so years, thousands of plugins have been developed by a huge open source-based community; there is a jQuery plugin for almost all use cases now. Unfortunately though, some of these plugins that have been created may also generate some bad press due to the poor-performing and underdeveloped code involved. The saying 'A poor workman blames his tools' is true for all of these terribly written plugins and the developers that decide to actually use them. Knowledge within the open source community allows the cream to rise to the top and the jQuery community has published guidelines on extending the library.

We have to thank jQuery's popularity for kick-starting the vibrant JavaScript community we have today. There have been off-spin projects such as 'jQuery Mobile', which is an extension of the library to produce mobile and tablet-ready interfaces. Web developers can use an existing skillset to make advanced interfaces for the newest devices – devices that didn't exist when it was initially released.

The library has enabled developers to work with the DOM API to build a far richer web than we had ten years ago – improving the end user's experience, which should ultimately be the most important goal of any web project.

The library will be ten years old in August 2016 making it the great grandfather of JavaScript frameworks – a proven and stable platform that has enhanced millions of websites. There was a time when people believed Facebook would start losing users as its popularity increased – this never happened because the platform worked whilst serving a purpose, like jQuery.

Understanding, when enhancing an interface to improve the end user's experience, is key and jQuery has been successfully doing this for nearly ten years. Newer libraries may introduce new features and paradigms; it may not be faster than its native counterpart functionality, yet jQuery's dependable core and widespread knowledge base ensure that it stays production ready.

Daniel Jenkins

Development lead

"JQuery's easy-to-understand syntax allows me to introduce experience-enhancing features on websites quickly and efficiently that work across multiple web browsers and devices."

"The library is well-thought-out and the jQuery team has an incredibly methodical process to implementing new features"

What's up with v3?

LEARN MORE ABOUT THE FIRST MAJOR RELEASE IN TWO YEARS

Version 3 alpha is now out in the wild and comes in two flavours; much like the release of version 2 it drops support for Internet Explorer below version 9. There is however a 'compat' version if you want to support older browsers and utilise new features for your more up-to-date users. The only caveat here being that the compat version will be of a larger file size as well as being slower in real-use performance.

When developers release a major version of any framework there is usually a queue of people complaining about breaking changes, and the jQuery team has worked hard to ensure this isn't the case – almost all code should continue to work with zero changes.

There are, however, changes that are a step in the right direction. In previous releases any jQuery methods that hide/show content (.hide(),.show(), .slideIn()) applied display properties directly to the element, sometimes adding display:block to elements and overriding display properties that should have been display:inline – this can cause a lot of layout issues. This is no longer the case and jQuery recommends to hide/show content using CSS classes with methods such as .toggleClass(). This increases performance by ensuring the change is rendered by the CSS render engine.

Performance seems to be high on the agenda and whilst reworking the internals of custom attribute selectors (:visible, :hidden) they've noted speed increases of up to 17 times faster in execution. The underlying animation engine now utilises requestAnimationFrame meaning smoother animations, less CPU time and reduced battery drain on mobile devices – great if you're using jQuery in PhoneGap or a Cordova application.

Version 3.0 goes some way to futureproofing the library, ensuring that it's keeping up with the latest standards (.defferred() is now Promises/A+ compliant for example). Along with the increase in performance, and when combined with the removal of deprecated functionality, jQuery v3 is another step forwards.

What's in the core API?

REMEMBERING THE CORE METHODS IS A BIG UNDERTAKING, SO HERE'S A LIST OF WHAT YOU SHOULD KNOW

selectors $('selector')

*****
api.jquery.com/all-selector/
Selects all elements.

.class
api.jquery.com/class-selector/
Selects all elements with the given class.

#id
api.jquery.com/id-selector/
Selects a single element with the given id.

selector1, selector2, selector3
bit.ly/1Cs2xEX
Selects the combined results of all the specified selectors.

element
bit.ly/207g9md
Selects all elements with the given tag name.

parent > child
bit.ly/207gbdS
Selects all direct 'child' elements specified by 'parent'

ancestor descendant
bit.ly/1PPT039
Selects all of the elements that are the descendants of the ancestor element.

:first
api.jquery.com/first-selector/
Selects the first matched element.

:last
api.jquery.com/last-selector/
Selects the last matched element.

:first-child
bit.ly/1LXfv4A
Selects all elements that are the first child of their parent.

Attributes & CSS

.attr()
api.jquery.com/attr/
Gets the attribute for the first selected element.

.data()
api.jquery.com/data/
Store data associated with matched element or return the first matched elements.

.prop()
api.jquery.com/prop/
Gets the value of a property for the first matched element or sets a property on matched elements.

.val()
api.jquery.com/val/
Gets the value of the first matched element or sets a value on the matched elements.

.addClass()
api.jquery.com/addClass/
Adds specified class to the matched elements.

.css()
api.jquery.com/css/
Adds specified style property to the matched elements.

.hasClass()
api.jquery.com/hasClass/
Determines if any matched elements are assigned to a specific class.

.removeClass()
api.jquery.com/removeClass/
Removes specified class from matched elements.

.toggleClass()
api.jquery.com/toggleClass/
Adds or removes a specific class from matched elements.

.innerHeight()
api.jquery.com/innerHeight/
Gets the height of first matched element, This will include the padding as well but does not include the border.

.innerWidth()
api.jquery.com/innerWidth/
Gets the width of first matched element. This will include the padding as well but does not include the border.

.outerHeight()
api.jquery.com/outerHeight/
Gets the height of first matched element. This will include the padding, border and optionally the margin.

.outerWidth()
api.jquery.com/outerWidth/
Gets the width of the first matched element. This includes padding, border and optionally the margin.

.width()
api.jquery.com/width/
Gets the current computed width (in px) of the first matched element, or it will set the width on all of the matched elements.

.offsetParent()
api.jquery.com/offsetParent/
Gets the closest ancestor element that is positioned.

.position()
api.jquery.com/position/
Gets the current coordinates of the first matched element, relative to the offset parent.

.scrollLeft()
api.jquery.com/scrollLeft/
The current horizontal position of the scrollbar for the first matched element or sets horizontal position of the scrollbar for elements.

Browser Events

.resize()
api.jquery.com/resize/
Binds event handler on resizing.

event preventDefault()
api.jquery.com/event.preventDefault/
If this method is called, then the default action of the event will not be triggered.

.scroll()
api.jquery.com/scroll/
Binds event handler upon element scrolling.

.load()
api.jquery.com/load/
Binds event handler to JavaScript 'load' event.

.ready()
api.jquery/ready/
Specify function to run when DOM load is complete

.on()
api.jquery.com/on/
Attaches an event handler for one or more events.

.trigger()
api.jquery.com/trigger/
Executes matched events.

.blur()
api.jquery.com/blur/
Binds to the form 'Blur' event.

.change()
api.jquery.com/change/
Binds to the 'change' event handler.

.focus()
api.jquery.com/focus/
Binds or triggers to the form 'Focus' event.

jQuery and ES6

TAKE A LOOK AT THESE ES6 FEATURES AND HOW JQUERY CAN USE THEM

ECMAScript 6 or ES6 is the name for the latest version of JavaScript to be released, officially launched in June 2015 it's going to take browsers a considerable amount of time to implement all of the new features. Developers wishing to implement the code now will need to use a compiler from ES6 into ES5 such as babel (**babeljs.io**) to ensure backwards compatibility. The current version of JavaScript (ES5) can turn developers with a back-end mindset blue in the face with its lack of certain language features. The release of ES6 is designed to move JavaScript as a language towards a platform for building more complex applications; the creation of platforms like Node.js and PhoneGap have proven that JavaScript is an extensive language no longer existing in just the traditional web browser context. Smart TVs, mobile applications, websites, servers and many other platforms now depend on JavaScript, thus updating the language needs to be controlled and ES6 goes a long way to becoming backwards compatible.

ES6 may not be production-ready today – unless the exact platform specifications are known, yet it contains features the modern web developer will want to be aware of. Languages evolve and the technology that supports them does too, jQuery has a proven track record of helping developers build interfaces for the widest level of support available. The majority of JavaScript wrote by developers utilising v1 of the library works with the latest version of today's browsers, and vice versa with code written in v2 and older browsers. This is no mean feat – it's one of the main reasons the jQuery library still exists and has a massive user base, a lot of logic happens behind the scenes. As newer features are introduced, jQuery takes a logical approach to supporting them with a good forewarning on sunsetting older functionality.

Asynchronous functionality is notoriously hard to understand and recent versions have introduced the latest standard of Promises. Many developers rely on the library to send/receive data to the server, from the browser using the $.ajax() wrapper.

```
var request = $.ajax({
url: "script.php",
method: "POST",
});
request.done(function( msg ) {
console.log(msg);
});
request.fail(function( jqXHR, textStatus ) {
console.log('error');
});
```

JQuery's recent implementation of Promises enables developers to effectively ensure that the JavaScript runtime can continue whilst waiting on asynchronous tasks to complete. ES6 introduces the fetch() API, an extension of the XHR standard and it look similar to jQuery's syntax:

```
fetch('/script.php')
.then(function(data){
console.log(data)
}).catch(function(){
console.log('error')
})
```

There is the potential for future releases of jQuery to enable the fetch() API in browsers that support it.

ES6 allows developers to work with JavaScript files as modules. On a website where jQuery is only required on one webpage ES6 modules can help. Libraries such as RequireJS help to do this but incur browser load and performance penalties for the end user. Theoretically jQuery could be extended in ES6 when you do it like this:

```
//script.js
<module>
import * as $ from 'jquery';
</module>
//jquery.js
module.exports = {
$ : jQuery()
}
```

This module pattern would enable jQuery to remove the method $.fn.extend() and allow plugin developers to still hook into the jQuery core.

ES6 controversially introduces classes, constructors and extend into the language – something back-end developers will rejoice about. Regardless of opinion the new inheritance model could be put to good use with jQuery becoming class-based, with plugins extending the base jQuery class.

```
class pictureSlider extends jQuery {
constructor() {
this.slideSpeed = 5;
this.touchEnabled = true;
}
//plugin code
}
```

There isn't currently an ES6 implementation of the jQuery library, but as popularity increases the library will look to support it. A large community built on plugins using the current $.fn.extend() prototype makes picking one of the previous two examples for extending the library a logical move once browser support is ready. ES6 introduces many native APIs that jQuery has included workarounds for in previous releases such as new iterators (for + of) and common data-structures (.map). You can read more at **ecma-international.org/ecma-262/6.0/**) for a full list of all the new ES6 features. ES6 will make a lighter and faster jQuery, ensuring that developers using the library can focus on writing code that is doing less behind the scenes and more for the end user.

Extending jQuery

You can make use of jQuery.fn.extend() ($.fn) to extend the jQuery prototype and provide yourself with new methods that can be chained off the jQuery() function.

"JQuery provides a multitude of helpful functions, that should have been included in JavaScript from day one"

A pro's guide to the benefits of jQuery

JQUERY HAS THE POTENTIAL TO TURN A GOOD DEVELOPER INTO AN EXPERT: FIND OUT HOW

Matt Leach

Head of development, Thinking Juice

"Using jQuery was the first time I saw the web as more than a flat interface and a place to create experiences for users and customers, every time I use the library I learn something new."

When looking to use web technologies for a project there are many factors involved in the decision-making process to consider, such as how long until the project needs to be delivered. Does the project need to support the latest browsers? Or even worse support the oldest of browsers? Does the project have to withstand being production-ready, bug-free and dependable? Multiple factors play a part, some that may not even be considered by the developer when embarking on a new project.

Working with HTML and CSS can, for many, be an easy choice when it comes to building interfaces, yet the sheer number of JavaScript frameworks to choose from can be bewildering. Syntax aside, choosing the right framework takes a large amount of consideration. It's the wider thought process of the entire project and the impact of decisions made early on that makes a good developer, great. Knowledge like this isn't gained over night but practising, experimenting, making mistakes, learning and talking to peers can reduce the headaches along the way of mastering development.

Progressive enhancements

JQuery was initially introduced to enhance the world of web development and, like all open source efforts, to enhance these framework user's experience. In the real world these enhancements can be measured by trivial elements such as allowing new features to be added to a website within tight timescales or ensuring a student can complete a task they set out to do. Take for example the blue chip world, a world that has inherently slow IT upgrade schedules with the majority of these users using much older versions of Internet Explorer, picking the right library helps to ease the pain of building features that are cross-browser compliant.

Trying to write vanilla JavaScript for even basic use cases can be painful, taking the blue chip world for example try adding a class in jQuery:

```
$('#contact').addClass('sent');
```

The jQuery method has enabled this functionality in minimal code for working across multiple browsers. Now try writing that in native JavaScript:

```
document.getElementById("contact").
classList.add("sent");
```

Now this is nearly double the length and works in modern browsers, looking back to older versions of IE this would cause an error. To ensure cross-browser compliance look at this code (and its length):

```
var d = document.getElementById("contact");
d.className = d.className + " sent";
```

.promise()
You can use this method to return a Promise object to observe when all actions of a certain type that are bound to the collection, regardless of whether it is queued or not, have finished.

One line of code may suffice if the project only uses JavaScript for this single action – loading in a library would then be overkill. JQuery gives developers a set of tools that keeps all levels focused on the end goal.

Deadline-driven development

Deadlines, much like taxes, are unavoidable in most walks of life, and a day-to-day occurrence. Known for causing many a grey hair, working smart can make the approach much more bearable. It's all too common to see more junior developers looking at a current project or application and then announce 'Hey, we should rewrite that in this cool framework'. This doesn't mean they are incorrect and their enthusiasm shouldn't be rewarded, but taking on board a new library has to be carefully considered. It needs to ensure a return on investment whether that's personal or financial. There will be companies pushing products live with a new library, built by a freelance resource that their internal developers can't support. In a development team, at crunch time, everyone will be expected to help and chances are nine out of ten developers in the team will be able to help iterate on any code wrote in jQuery. If written in the 'cool' framework the resource pool may be slightly smaller. When starting any project be aware that at some point in the future, other developers will have to support it.

How do the pros really use it?

A skilful developer is someone that has not only mastered the art of writing code and logical thinking but the understanding that if something has to be repeated more than once there is probably a way of automating that process.

DRY (don't repeat yourself) principles are abundant when it comes to writing efficient, human-readable code. Look at this code for changing an element's state once interacted with:

```
$('.btn').addClass('in-use');
$('.btn').attr('disabled', true);
$('.btn').data('used', true);
```

JQuery provides a smart developer the ability to chain these methods, using the prototype inheritance model (a paradigm widely used JavaScript):

```
$('.btn').addClass('in-use').
attr('disabled', true).data('used',true);
```

JQuery provides a multitude of helpful functions, that should have been included in JavaScript from day one. Replicating functionality that other languages contain from the outset, a lot of very similar libraries exist purely to fulfil these tasks. Many JQuery utility methods are a secret toolset that a lot of developers don't mention frequently.

The jQuery library may not be the fastest or the lightest but a community of thousands can continue producing interfaces in an ever-changing world of technology. Being a good developer isn't about being an early adopter it's about writing well-formed code that works when required, whilst knowing the tools to get the best out of the platform.

> *"Taking on board a new library has to be carefully considered. It needs to ensure a return on investment"*

5 jQuery features you need to know

METHODS WORTH USING

.ajaxSetup()
api.jquery.com/jQuery.ajaxSetup/
This little-known method allows variables to be predefined for all AJAX requests. For example using a defined URL route for all requests in an application could be set via:

```
$.ajaxSetup({
url : '/requestMe'
})
```

.grep()
api.jquery.com/jQuery.grep/
Grep is a Unix command and jQuery provides this as a helper that is great for searching through arrays for specific values.

```
var data = [0,1,2];
var filteredData = $.grep( [ 0, 1, 2
], function( n, i ) {
return n > 0;
}, true );
```

The variable filteredData is now an array of any values greater than 0;.

.closest()
api.jquery.com/closest/
This method traverses upwards from the current DOM element to a document's root element. It accepts a jQuery object as an argument and if included when traversing, jQuery returns the first matching element.

.holdReady()
api.jquery.com/jQuery.holdReady/
This is a relatively unknown method that holds or releases the .ready() function. Use cases for this may seem far-fetched, but there can be times where the DOM is ready but other functionality should happen first.

```
$.holdReady( true );
$.getScript( "myplugin.js",
function() {
$.holdReady( false );
});
```

$.merge()
api.jquery.com/jQuery.merge
Developers used to PHP (array_merge) will recognise this as a fast way of merging two arrays – this neat helper provides functionality that native JavaScript doesn't provide.

```
var new_array = $.merge( [ 0, 1, 2 ],
[ 2, 3, 4 ] )
```

Advanced jQuery

REGARDLESS OF EXPERIENCE THESE ADVANCED METHODS OF ARE DEFINITELY WORTH CHECKING OUT

Filters

.eq()
api.jquery.com/eq/
Reduces the set of matched elements to the one at the specified index.

.filter()
api.jquery.com/filter/
Reduces the set of matched elements to those that have matched the selector or pass the function's test.

.first()
api.jquery.com/first/
Reduces the set of matched elements to the first one that is in the set.

.has()
api.jquery.com/has/
Reduces the set of matched elements to those that have a descendant that matches the selector or DOM element.

.is()
api.jquery.com/is/
Checks against a selector and return true if at least one of these elements matches the selector.

.last()
api.jquery.com/last/
Reduces the set of matched elements to the final one in the set.

.map()
api.jquery.com/map/
Parses each element in the current matched set, producing a new jQuery object containing the return values.

.slice()
api.jquery.com/slice/
Reduces the set of matched elements to a subset specified by a range of indices.

Utilities

inArray()
bit.ly/1MgB8c6
Searches for a specified value in an array and returns its index.

jQuery.isArray()
bit.ly/1GvAzz8
Determines whether the argument is an array.

.jQuery.isEmptyObject()
bit.ly/1Nyv0zh
Checks to see if an object is empty.

jQuery.isFunction()
api.jquery.com/jQuery.isFunction/
Checks if the argument passed is a JavaScript function object.

jQuery.isNumeric()
api.jquery.com/jQuery.isNumeric/
Determines whether its argument is a number.

jQuery.makeArray()
api.jquery.com/jQuery.makeArray/
Converts an array-like object into a true JavaScript array.

jQuery.merge()
api.jquery.com/jQuery.merge/
Merges the contents of two arrays together into the first array.

jQuery.now()
api.jquery.com/jQuery.now/
Returns a number representing the current time.

jQuery.parseHTML()
api.jquery.com/jQuery.parseHTML/
Parses a string into an array of DOM nodes.

Deferred Object

deferred.always()
api.jquery.com/deferred.always/
Adds handlers to be called when the deferred object is either resolved or rejected.

deferred.done()
api.jquery.com/deferred.done/
Adds handlers to be called when the Deferred object is resolved.

deferred.notify()
api.jquery.com/deferred.notify/
Calls the progressCallbacks on a deferred object along with the given arguments.

deferred.progress()
api.jquery.com/deferred.progress/
Adds the handlers that will be called when the deferred object goes on to generate the progress notifications.

deferred.reject()
bit.ly/1GIgIMT
Rejects a deferred object and calls any failCallbacks with specified arguments.

deferred.resolve()
api.jquery.com/deferred.resolve/
Rejects a deferred object and calls any failCallbacks with the given arguments.

deferred.resolveWith()
api.jquery.com/deferred.resolveWith/
Resolves a deferred object and calls any doneCallbacks with the given context and args.

AJAX

jQuery.ajax()
api.jquery.com/jQuery.ajax/
Performs an asynchronous HTTP (AJAX) request.

jQuery.ajaxSetup()
api.jquery.com/jQuery.ajaxSetup/
Sets default values for any future AJAX requests.

jQuery.get()
api.jquery.com/jQuery.get/
Loads data from the server using a HTTP GET request.

jQuery.getJSON()
api.jquery.com/jQuery.getJSON/
Loads JSON data from the server with a GET HTTP request.

jQuery.getScript()
bit.ly/1RdOJVI
Specifies a function to run when DOM load is complete.

.on()
api.jquery.com/on/
Attaches an event handler for one or more events.

.trigger()
api.jquery.com/jQuery.getScript/
Loads a JavaScript file from the server using a GET HTTP request, then executes it.

.load()
api.jquery.com/jQuery.load/
Loads data from the server and places the returned HTML into the matched element.

jQuery.post()
api.jquery.com/jQuery.post/
Loads data from the server using a HTTP POST request.

5 common mistakes to avoid

A LITTLE KNOWLEDGE OF THE COMMON MISTAKES EVERYONE MAKES CAN HELP TO PRODUCE BETTER CODE

Relying on $(document).ready();

Including JavaScript in the <head></head> tags and then waiting for jQuery to run when the DOM is fully loaded can cause the end user a headache-inducing wait. Executing JavaScript code when the browser has loaded the complete DOM is in itself inherently difficult and troublesome. Each browser tells JavaScript when the DOM is ready at a different time and jQuery has a helpful method – .ready() – that does a multitude of things in the background to ensure the DOM is loaded before firing. JavaScript is a blocking script – meaning when the browser finds the <script> tag it won't do anything else until that code has loaded in. Any assets, photos, CSS and other JavaScript files, won't execute until this file is loaded. It's considered best practice to place all JavaScript assets in the footer as JavaScript should enhance a page, not build the UI. If code is required to execute when all assets in the webpage are ready, it would be convenient to do this:

```
The code doesn't get executed until the
whole DOM is ready – large images slow down
the time from page load to the JavaScript
firing.
//code//
(function($) {
//fire code
})(jQuery);
```

Initialising any functions this way enables JavaScript to execute when possible. The end user will see the page being useable sooner – perceiving a faster load time. If specific code needs to ensure an element is completely loaded before being executed, use .load():

```
$(img).load(function(){
//execute code
})
```

The $var syntax

Use the dollar $var syntax to let others know it's a jQuery object. If the variable is in the correct scope, then jQuery will be able to access the element without having to search for the DOM again.

Hosting jQuery yourself

When building a website that utilises jQuery it's all too easy to download a copy of the library, place it in a local directory then load it in to the pages required.

```
<head>
<script src='/jQuery.js'></script>
</head>
```

For every new visitor to a website they have to download this file from the server, which slows down the initial page load and introduces an extra drain on bandwidth. Reference jQuery from a CDN and the chances are that when someone visits the website, they will previously have visited a website that hosts jQuery with that CDN (Google for example) and already have jQuery stored in the browser's cache, reducing bandwidth and page-load time.

Faster selectors

With the introduction of CSS preprocessors, it's become a bit of an issue with overly specific CSS declarations yet with jQuery – we find the opposite happening. Look at this navigation code:

```
<ul id="navigation">
<li><a class="nav-item"
href='#home'>Home</a></li>
<li><a class="nav-item"
href='#about'>about</a></li>
<li><a class="nav-item"
href='#contact'>contact</a></
li>
</ul>
<script>
$('.nav-item').click(function(){
//code to scroll to location
})
</script>
```

The lack of specificity here means that jQuery has to traverse the whole DOM to find .nav-item – on big pages this can reduce performance greatly. It's like going shopping and having to look at every item in the shop to find a specific item. You can increase performance by increasing selector specificity:

```
<ul id="navigation">
<li><a class="nav-item"
href='#home'>Home</a></li>
<li><a class="nav-item"
href='#about'>about</a></li>
<li><a class="nav-item"
href='#contact'>contact</
a></li>
</ul>
<script>
$('#navigation .nav-item').
click(function(){
//code to scroll to location
})
</script>
```

Now when traversing the DOM, jQuery looks for the parent id (#navigation) and then looks for its child element with the class declared. Going back to the shop analogy, the aisle would now be known as (#id), thus searching for the specific item would take much less time than before.

Not caching DOM elements

Here is a simple script for adding a class to a website's navigation, then removing that class after executing other logic:

```
$('#navigation').addClass('visible');
//execute other code
$('#navigation').removeClass('visible');
```

Every time jQuery wants to alter the navigations class attributes, it has to traverse the whole DOM and find the element with the id #navigation, alter its class then continue processing the next line of JavaScript. Each time the $('selector') is used, jQuery has to look for this element in the DOM. Assign the jQuery $('selector') object to a variable increases performance, as a reference to that element is now available in memory.

```
var $navigation = $('#navigation');
$navigation.addClass('visible');
//execute other code
$navigation.removeClass('visible');
```

Animating with jQuery

When jQuery was initially released its internal animation engine was one of the most ground-breaking features – since then browsers and CSS have moved considerably forward. Take this example of animating an element's height for example:

```
$('.changeHeight').animate({
'height' : 250
}, 650);
```

JQuery internally calculates the element's height, then changes the height up to 60 times a second, causing the browser to redraw the DOM every single time it's changed (60fps). CSS animations in modern browsers are handled using hardware acceleration and are processed by the GPU ensuring a faster experience over CPU tasks like rerendering the DOM for example.

Smooth auto heights

Adding the class .open to the element will cause the GPU to redraw the elements properties without having to do any of the the complex JavaScript calculations that will be running.

```
<style>
.changeHeight{
-webkit-transition: all 0.65s
ease-in-out;
-moz-transition: all 0.65s
ease-in-out;
-o-transition: all 0.65s ease-in-out;
transition: all 0.65s ease-in-out;
overflow:hidden;
height:auto;
max-height:0;
display:block;
}
.open{
max-height:250px;
}
</style>
<script>
$('.changeHeight').addClass('open');
</script>
```

Make an image-based pop-up animation menu

As seen on **mediadistribution.espn.com**

Above the slider
Just above our elements you can find a neat parallax slider showcasing some ESPN facts, by individual sport.

Within a slider
You may also notice that the hoverable elements are contained within a slider, enabling a larger selection of elements.

On hover
Hovering on the elements leads to simultaneous transitions including opacity, margin-top and hidden element changes.

Video backgrounds
The focus effect is not the only treat on this page, or within the site. Note the great use of video backgrounds here.

The pop-up effect
Scroll down the page until you find the the elements which utilise the pop-up effect we are re-creating in this workshop.

DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/bks-894

Hover effects – they never go out of fashion. No matter how much web design progresses away from the desktop and further into the touch-screen world, where hover effects become moot, designers still love to come up with new and entertaining rewards for placing the cursor on a relevant area of the page. Not only does it deliver the doorway to the next step in the user experience, but it also make it more entertaining to knock on that door in the first place.

Take sports channel ESPN's website: a veritable feast of background video, parallax animations, responsive layouts and, of course, impressive hover effects. In this Web Workshop, we're going to create our own version of the pop-out menu items that appear about halfway down the ESPN page. Hover on any of the images in the strip, each corresponding to a sport on the ESPN line-up, and that image will pop out of the strip in full colour, presenting a link to the relevant page. It's a relatively simple effect, and fairly easy to re-create, but there's a small thrill that comes from hovering over each item to see the same effect. Combining Bootstrap CSS with some CSS3 animations, we will build our own pictorial menu and re-create the hover effects.

<comment>
What our experts think of the site

Don't be a purist all the time

"It's a common criticism that too many effects and animations are superfluous and do nothing to enhance the flow of information. While a lean path through your site is a must, it's also important to remember how many of your users are not web designers and appreciate a bit of animated magic for its own sake."
Richard Lamb, freelance web designer at Inspired Lamb Design

Technique

1. Initial HTML

Our starting point is a container and three menu-item divs within. These divs will contain all the elements, visible or hidden in static and hover states. Inclusion of Bootstrap's CSS will style the container and responsive classes. Also add a header and title.

2. Initial CSS

When styling the h1 tag, make sure you are giving generous margins above and below, to accommodate raising elements. Add transition to the menu-item class, covering all possible animations. Don't forget the vendor prefixes. To align our elements snugly, counter the Bootstrap padding either side.

3. Insert images

Insert your images into each of the three menu-items. You may want to include a max-width of 100 per cent in the CSS. Then give each image an opacity of 0.3. Combined with the dark background of the div, this gives each image a darkened appearance.

```
<div class="menu-item col-sm-4"><img
src="images/bread.jpg" alt=""/></div>
.menu-item img{
opacity:0.3;
}
```

4. Link box HTML

Underneath the image, but still within the menu-item, insert a linkbox div which contains an a href link. Repeat this for all three menu items. This will be hidden initially, so move on to the CSS for these elements.

```
<div class="menu-item col-sm-4">
<img src="images/bread.jpg" alt=""/>
<div class="linkbox">
<a href="#">SERVE ME</a>
</div>
</div>
```

5. Style boxes

As mentioned, set an initial display of none on the linkboxes. Style as you see fit, but a generous padding around both the box and the link, plus a subtle border, looks good and corresponds with the look of the ESPN site that we are trying to emulate.

```
.linkbox {
display:none;
adding: 25px;
background: #03A678;
txt-align: center;
}
.linkbox a {
border: 2px solid #4DAF7C;
color: #ffffff;
text-decoration:none;
padding: 25px;
}
```

6. Hover effects

Three distinct hover animations bring the effect to life. First, what you have to do is just set a minus value to the margin of the menu item. Then set the opacity of the image to 1. Finally, change the display value of the linkbox, bringing it into view.

```
.menu-item:hover {
margin-top:-50px;
transition: all 0.2s;
}
.menu-item:hover img {
opacity:1;
}
.menu-item:hover .linkbox {
display:block;
}
```

EXPERT ADVICE
Watch the browsers
Browser capabilities is still a constant headache for many web designers and developers. Including vendor prefixes for CSS3 transitions and animations is as vital as it is easy to overlook. While Chrome remains top of the pile, our job is easier, but IE9 and Opera Mini will still be the ones that will miss out.

DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/bks-894

Build your own jQuery plugin

Add custom logic to jQuery with your own custom plugin and submit it to a repository

jQuery is everywhere: very few web designers can make do without it. For quite some time, developers have proceeded to create expansion modules intended to add extra logic to the product.

While the logic contained in these so-called plugins can reign from benign to supercomplex, the actual steps involved in coding and uploading tend to be similar regardless of product complexity. Yes, you are reading right: interfacing a charting plugin to jQuery is just as easy as creating a little demo toolkit.

Due to JavaScript's unique habits, hacking together a plugin is a sure-fire way to creating hordes of angry users: jQuery's developers established a series of 'best practices' which need to be respected if a plugin is to feel right. But fret not: we are ready and waiting to lead your code so that you're caught between a rock and hard place. During that process, we will also learn a bit about fitting into heavily populated global namespaces as well as a chainable method groups.

Finally, we will also take a look at getting your plugin listed in jQuery's shiny new NPM-based plugin registry. This is really useful: not only does it give you an ego boost, but it also greatly simplifies development.

1. Create a harness

Testing code requires the presence of a suitable environment. For our plugin, we will start out by creating the markup structure shown in the code on FileSilo that accompanies this step. The jQuery version is not particularily sensitive so just grab one that suits you.

2. Protect intellectual property

JQuery is not the only library which occupies $ – the safest way to keep our code from harm's way involves the use of an immediately invoked function expression. Create a file called 'imaginejqp.js', now and add the following code:

```
(function ( $ ) {
//Our work area
//. . .
//ends here
}( jQuery ));
```

3. Add function and variable

The closure provides us with a work area where member variables, methods and other code can be hidden. We will, for now, create a method which opens an alert dialog and a variable which contains textual content.

```
(function ( $ ) {
var aGreeting="Hello World!";
function showAnAlert()
{
alert("Hello from your plugin!");
}
}( jQuery ));
```

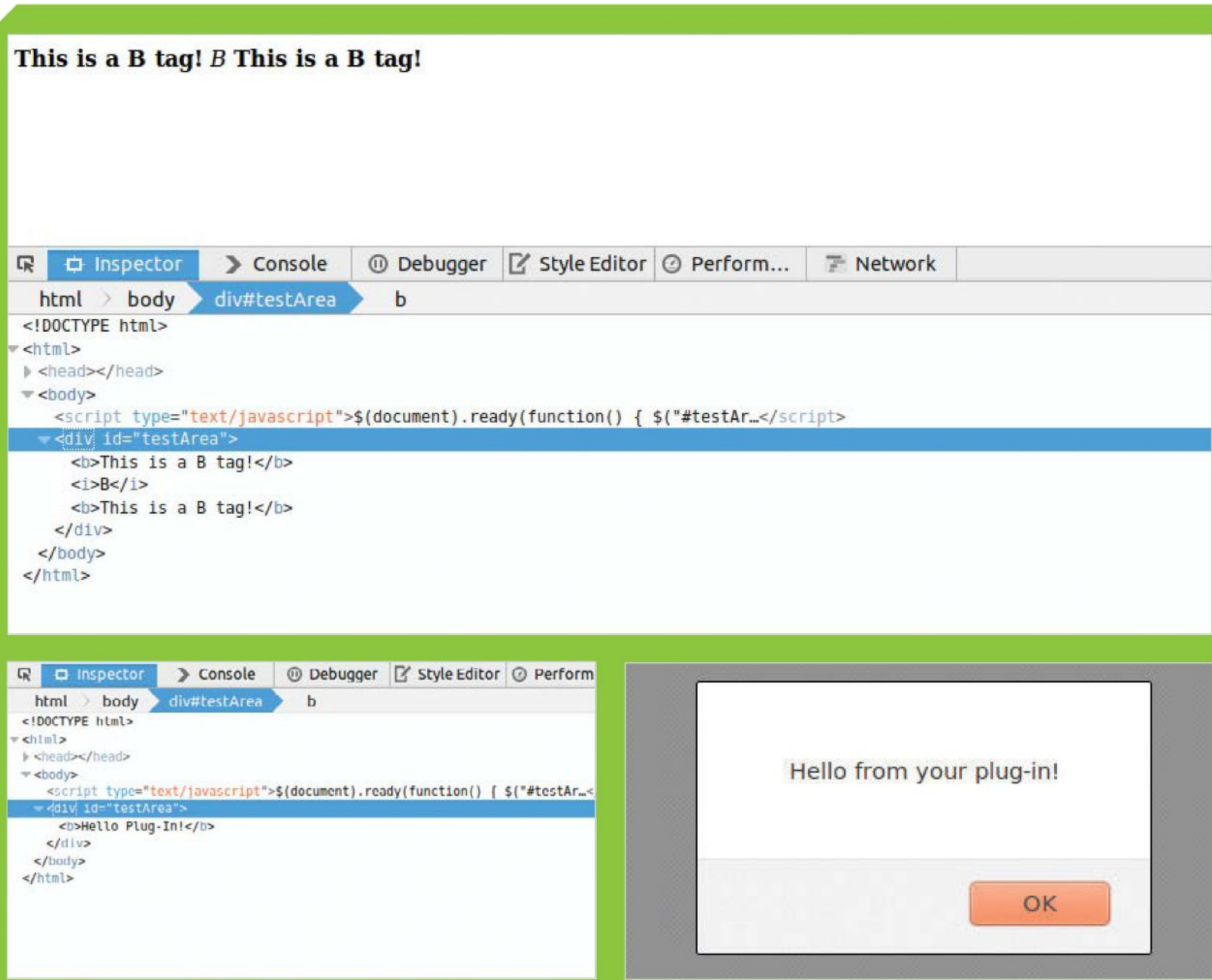
4. Expose some content

With that, it's time to expose some members to the caller. This is accomplished via the fn object of jQuery, which is exposed to the users of the framework during initialisation. Fn behaves just like any other object: you can add methods and variables, as shown in the code snippet below.

```
(function ( $ ) {
```

Pick a unique name

Avoid name collisions: add a two- or three-letter prefix to all members exposed in the $.fn object to make your product much more friendly.



<Top left>
• Thanks to the invocation of filter(), the contents of the lone <i> tag remain untouched

<Bottom left>
• Our newly added content can be seen both in the inspector and in the rendered view

<Bottom right>
• The message box is quite clear: code execution is passed successfully to the plugin code

```
var aGreeting="Hello World!";
function showAnAlert()
{
alert("Hello from your plugin!");
}
$.fn.imaginejqpnum=42;
$.fn.imaginejqp = function() {
showAnAlert();
};
}( jQuery ));
```

5. Keep the frequency clear

Even though the IIFE provides our code with a modicum of privacy, this does not protect us from congestion of the main $ object. Due to this, it is highly advisable to limit the population of the object. One way to achieve this involves the returning of an object like the one shown in our example:

```
(function ( $ ) {
var aGreeting="Hello World!";
function showAnAlert()
{
alert("Hello from your plugin!");
}
var retval=new Object();
retval.func=function() {
showAnAlert();
};
retval.num=42;
$.fn.imaginejqp = function()
{
return retval;
}
}( jQuery ));
```

Don't go overboard!

Your plugin is but a small part of the entire system. Web designers do take size into consideration when selecting modules: trying to do everything is a sure-fire way to making your plugin unpopular.

6. Get invoking

Let's now do a small test to determine whether our plugin works. Expand our harness with the code shown in the step – it calls the func method which we exposed via our JSON object. Run the code in a browser:

```
<!DocType html>
<html>
<head>
<script src="jquery-2.1.4.js"></script>
<script src="imaginejqp.js"></script>
</head>
<body>
<script type="text/JavaScript">
$("").imaginejqp().func();
</script>
</body>
</html>
```

7. Chained up

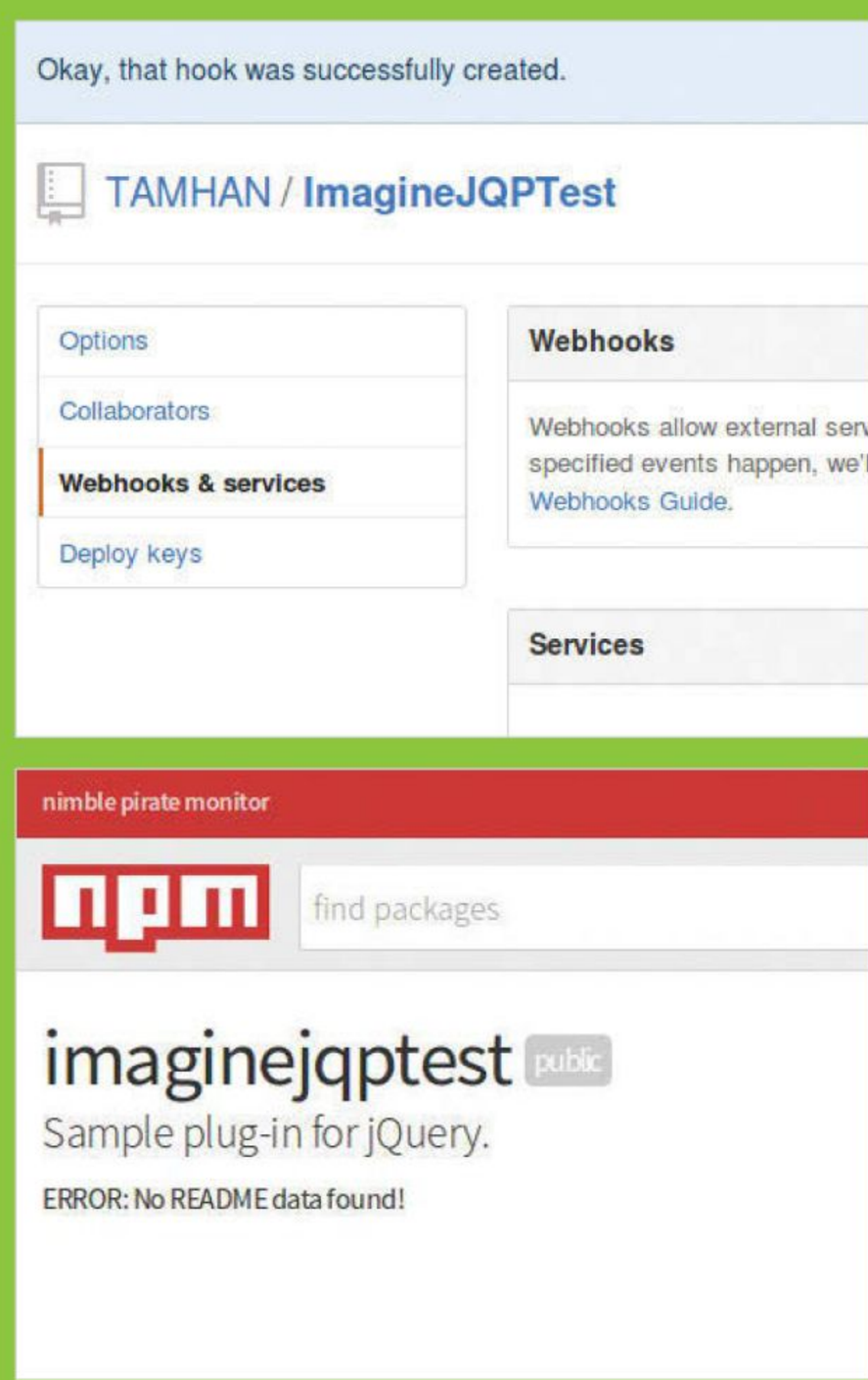
JQuery is known for its 'chained invocations': developers invoke functions on one another in a chained fashion like a().b().c().d(). Custom plugins should support this pattern by returning this instance from every method which does not return a value:

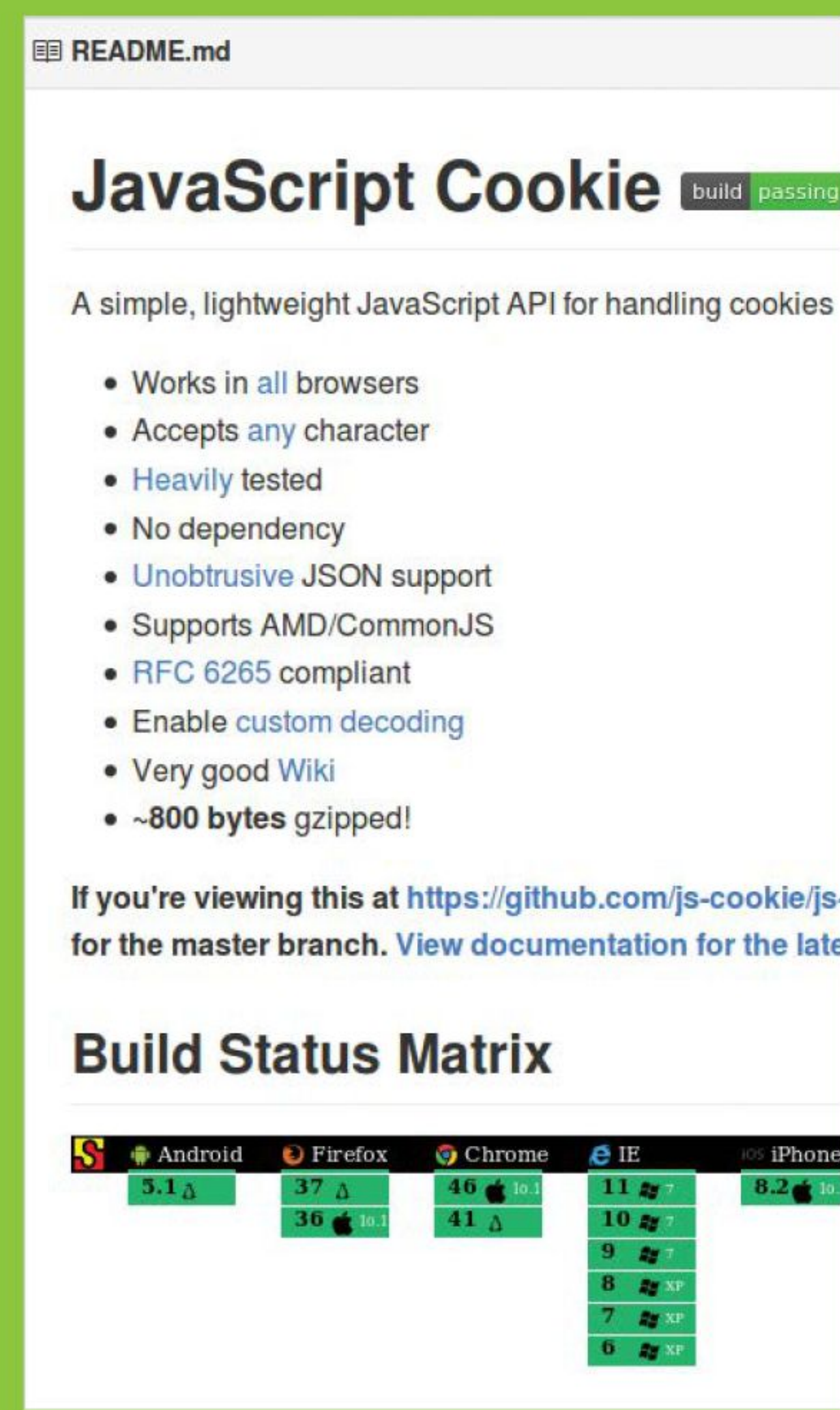
```
(function ( $ ) {
. . .
var retval=new Object();
retval.func=function() {
```

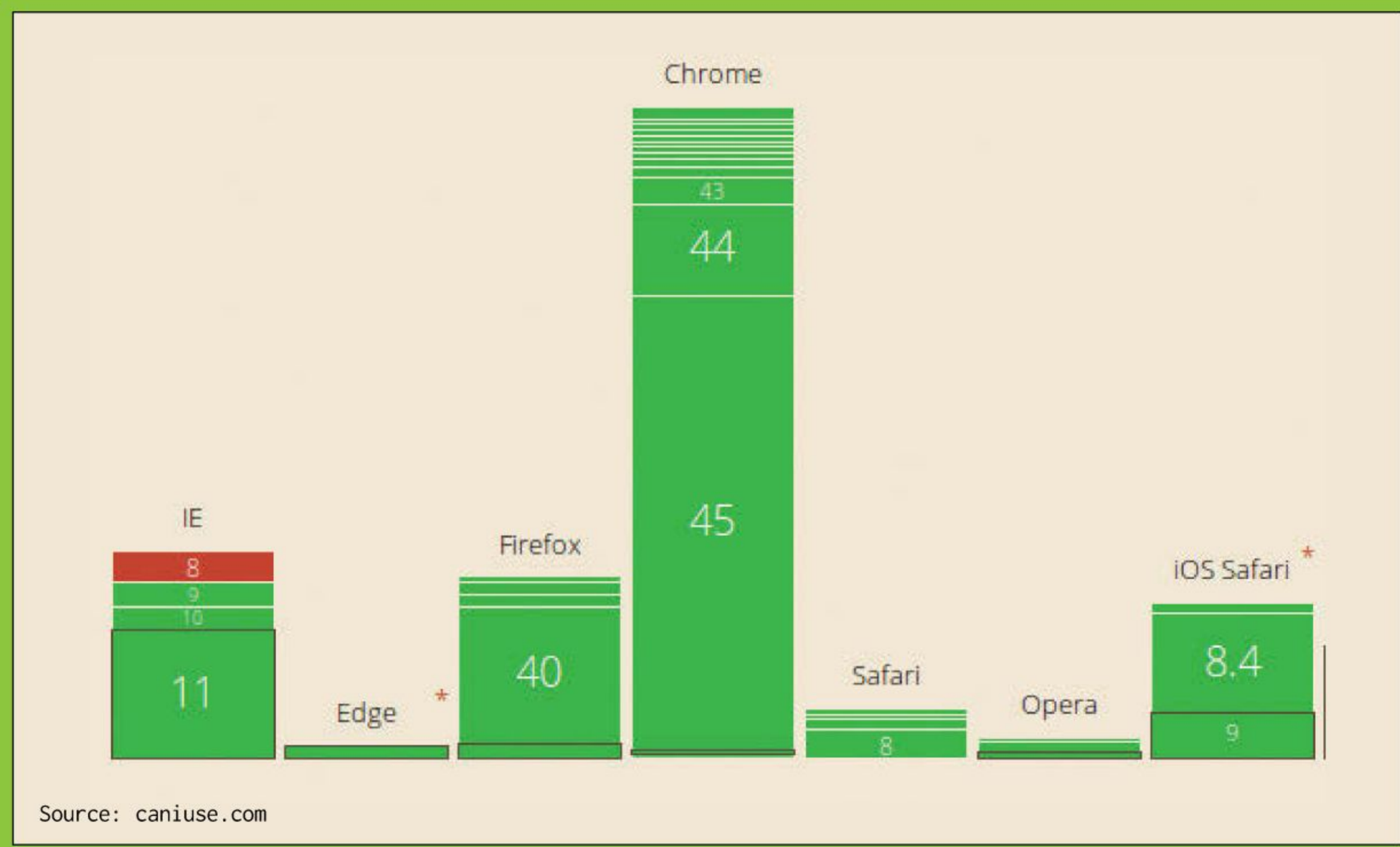
<Top left>
• Even though GitHub still permits you to create hooks for the jQuery plugin repository, be aware though that it is no longer supported

<Bottom left>
• Once the upload has been completed successfully, the finished product can be observed via a web browser of your choice

<Right>
• A well-written bit of documentation is a sure-fire way to raise developer interest







Test thoroughly

Many web designers will release jQuery plugins in order to draw attention to themselves and their work. You should be aware though that the release of one badly tested product can be enough to tarnish your reputation for many years to come.

Due to this, taking a variety of precautions against unreliable releases is recommended. First of all, you can deploy a system like Jasmine or QUnit in order to keep an eye on the code quality as you improve your product's design. Secondarily, it makes sense for you to prepare to perform your tests on a wide variety of browsers. If your product is supported on just one or two popular browsers, a lot of developers out there will be unlikely to deploy it for use in mass-market production systems.

```
showAnAlert();
return this;
};
retval.num=42;
$.fn.imaginejqp = function()
{
return retval;
}
}( jQuery ));
```

8. Access the stack

JQuery plugins are often used to modify elements of the DOM tree. An example for that would be the adjusting of the content of a tag, which can be accomplished via the method that is shown in the code that accompanies this step. Due to us returning a JSON object, our code does not live inside the jQuery object – if the function is a a direct child of fn, then myContext would not be needed.

```
//imaginejqp.js
(function ( $ ) {
. . .
retval.num=42;
retval.modify=function()
{
this.myContext.html("<b>Hello Plugin!</b>")
return this.myContext;
}
$.fn.imaginejqp = function()
{
retval.myContext=this;
return retval;
}
}( jQuery ));
//index.htm
<!DocType html>
. . .
<body>
<script type="text/JavaScript">
$(document).ready(function() {
$("#testArea").imaginejqp().modify();
});
</script>
<div id="testArea"></div>
</body>
</html>
```

9. Handle the gang

When dealing with a group of elements, jQuery's each method can provide you with a convenient way to execute a payload on each member of the set. Its return value should furthermore be returned in order to improve chainability.

```
(function ( $ ) {
. . .
retval.eachify=function()
{
return this.myContext.children().
filter("b").each(function(){
var elem = $( this );
elem.html("This is a B tag!");
});
}
//Return Retval
$.fn.imaginejqp = function()
{
retval.myContext=this;
return retval;
}
}( jQuery ));
```

10. Let's play!

Next, configure the test harness according to the settings shown in the code accompanying this step. Open the file in a browser: the <b> tags inside the structure are provided with some new content automagically.

```
<body>
<script type="text/JavaScript">
$(document).ready(function() {
$("#testArea").imaginejqp().eachify();
});
</script>
<div id="testArea">
<b>A</b>
<i>B</i>
<b>C</b>
</div>
</body>
```

11. Expose some properties

Plugins should be flexible. One way to ensure this is the creation of stateful products which contain methods intended for setting and/or getting the attributes stored in the privately visible global property store.

```
(function ( $ ) {
//Private elements
```

```
var aGreeting="Hello World!";
function showAnAlert()
{
alert(aGreeting);
}
//Exposed members
var retval=new Object();
retval.changeGreeting=function(newG)
{
aGreeting=newG;
}
//index.htm
<script type="text/JavaScript">
$(document).ready(function() {
$("#testArea").imaginejqp().
changeGreeting("Oi, World!");
$("#testArea").imaginejqp().func();
});
</script>
```

12. Expose a callback

Plugins should not block. One way to ensure reactivity involves the use of event-driven design: your clients pass in one or more callback methods, which get invoked as events occur. Our snippet leads the way.

```
(function ( $ ) {
//Private elements
var aGreeting="Hello World!";
function showAnAlert()
{
alert(aGreeting);
}
//Exposed members
var retval=new Object();
retval.doSomethingCB=null;
retval.doLongRunning=function()
{
setTimeout(retval.doSomethingCB, 10000);
}
```

13. Protect your user

Invoking an undefined callback is risky behaviour: on some browsers, the JavaScript runtime punishes it by immediately aborting code execution. One way to prevent the problem involves the declaration of a 'dummy', which makes sure that there always is something to call.

14. Handle the JSON

Most jQuery plugins expect to be provided with a JSON object containing further parameters. The structure shown in the code below shows a way to unburden the caller from providing a complete JSON object. Extend combines the default object with the parameters that we passed in.

```
(function ( $ ) {
$.fn.funcName = function( options ) {
var settings = $.extend({
// Defaults go here
}, options );
//Do work via contents of settings variable
};
}( jQuery ));
```

15. Advanced optioneering

Getting forked is the worst possible outcome for an open source product. This problem can be solved by making the logic highly modular. As an example for that, let us create a fictive measurement device handler. It communicates via a GPIB interface, which is replaced with a simple call to alert().

```
var measuringTools=new Array();
//Exposed members
var retval=new Object();
retval.measureSomething=function(what,
withWhom)
{
for(i=0;i<measuringTools.lenth;i++)
{
if(measuringTools[i].myName==withWhom)
{
measuringTools[i].measureGPIB(what);
}
}
}
```

16. Even more optioneering

If the system is set up as shown above, your users can easily add another measuring instrument to the plugin via the addMeasuringInstrument array. As long as the new measuring instrument adheres to the constraints set forth, no forks are needed to create new equipment.

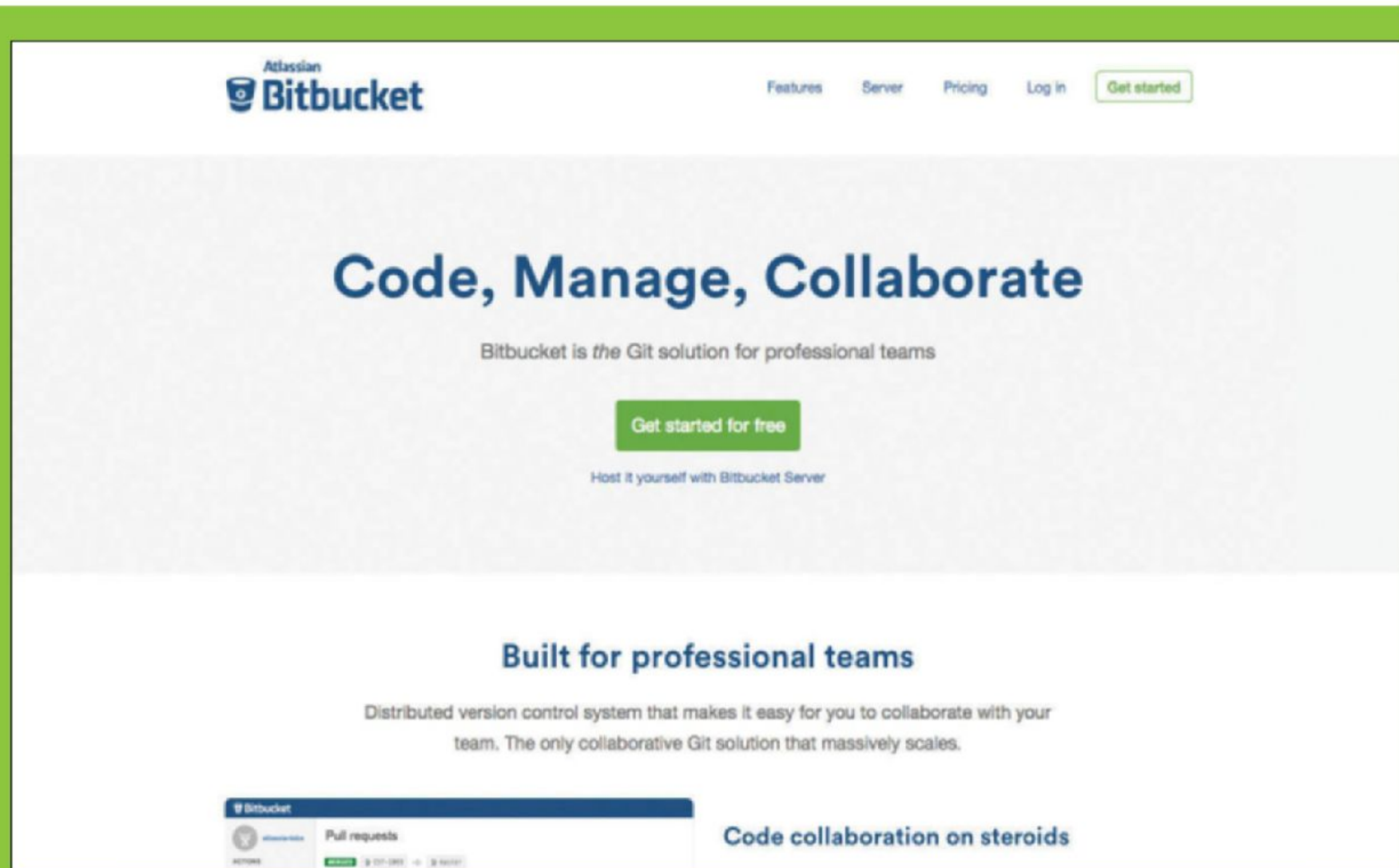
17. Document well

Making plugins accessible for third-party developers is best accomplished by providing detailed documentation.

Promote your plugin!

The web is full of blogs and websites related to jQuery, web design, CSS and more. Simply releasing a product via NPM is unlikely to get you the attention that you so desire as there may be overcrowding from a barrage of newly released plugins from a variety of sources. First of all, look to set up a plugin website alongside an issue tracker at a Git hoster of your choice – like Bitbucket for example. If developers see an ability to report issues with your plugin easily, then they will also be much more inclined to give your code a chance.

Secondarily, you should think about ways for you to attract attention to your new plugin. Sending an email to a bunch of bloggers and journalists, for example, might prove to be helpful.



One really helpful add-on is a minimal example, which shows the plugin operation and does not bog the reader down with loads of stylesheets and other niceties.

18. Log in to NPM

JQuery recently switched from its GitHub-based plugin repository to a more modern NPM-derived architecture. Install NPM, and proceed to signing yourself in via the adduser. Also be aware that the password is saved on the NPM server, and that it can be reset via **https://NPMjs.org/forgot**.

```
tamhan@TAMHAN14:~$ NPM adduser
Username: tamhan
Password:
tamhan@TAMHAN14:~$
```

19. Create a manifest

Hosting a product in NPM requires the presence of a manifest file. It describes the contents of the plugin (but note that all similarities to J2ME are unintentional). In the case of ImagineJQPTest, the file is called package.json and has the contents shown in the code.

```
{
"name": "imaginejqptest",
"title": "jQuery Plugin Test",
"description": "Sample plugin for jQuery.",
"version": "1.0.0",
"author": {
"name": "Tamoggemon Holding k.s."
},
"licenses": [
{
"type": "MIT",
"url": "https://github.com/jquery/jquery-color/blob/2.1.2/MIT-LICENSE.txt"
}
],
"main": "./imaginejqptest.js",
"keywords": [
"jquery-plugin",
"ecosystem:jquery"
],
"dependencies": {
"jquery": ">=1.5"
}
}
```

> Making plugins accessible for third-party developers is best accomplished by providing detailed documentation

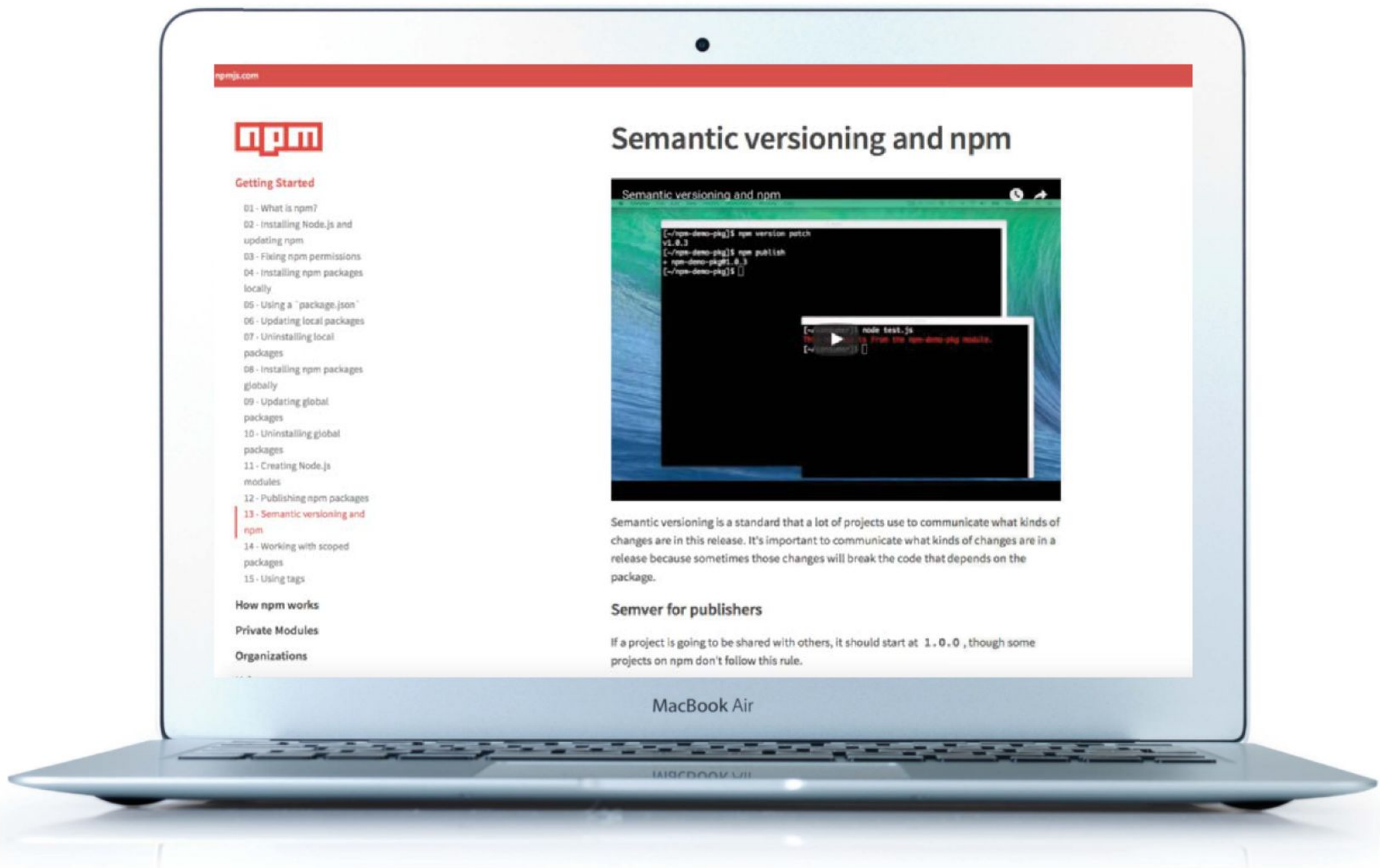
20. Push the code

Uploading the contents of the folder can then be accomplished via the corresponding NPM commands. NPM version lets you set up a version, while the publish command takes care of the actual uploading. Finally, visit the package's website at **NPMjs.com/package/<package_id>**.

```
tamhan@TAMHAN14:~/Desktop/imaginejqp$ NPM version 0.1.0
v0.1.0
tamhan@TAMHAN14:~/Desktop/imaginejqp$ NPM publish
+ imaginejqptest@0.1.0
```

21. Understand versioning

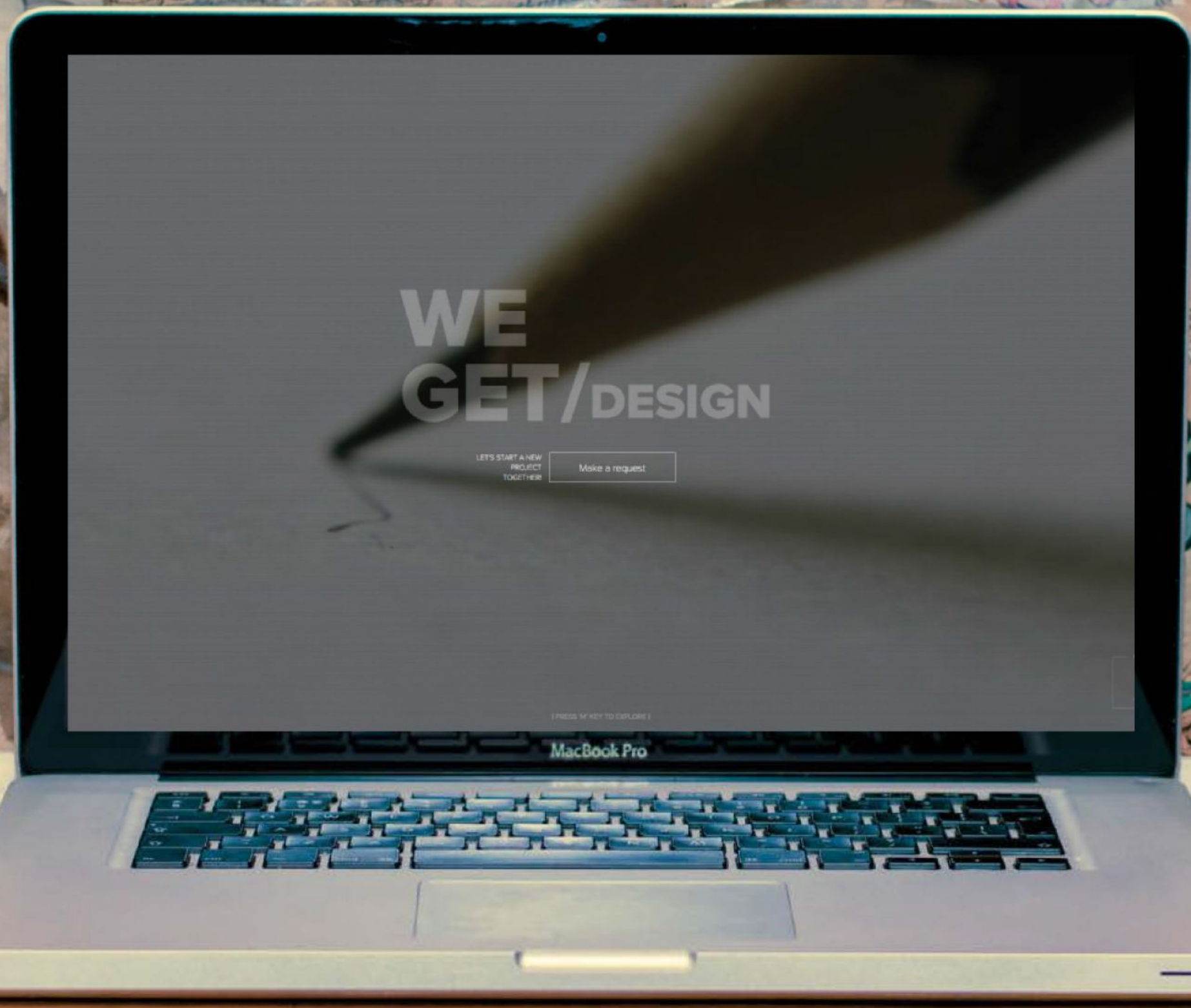
NPM's way of versioning – professionals refer to it as semantic versioning – can look quite odd. In principle, versions must be serialed according to the scheme.. Further information on the topic can be found via **docs.NPMjs.com/getting-started/semantic-versioning**.



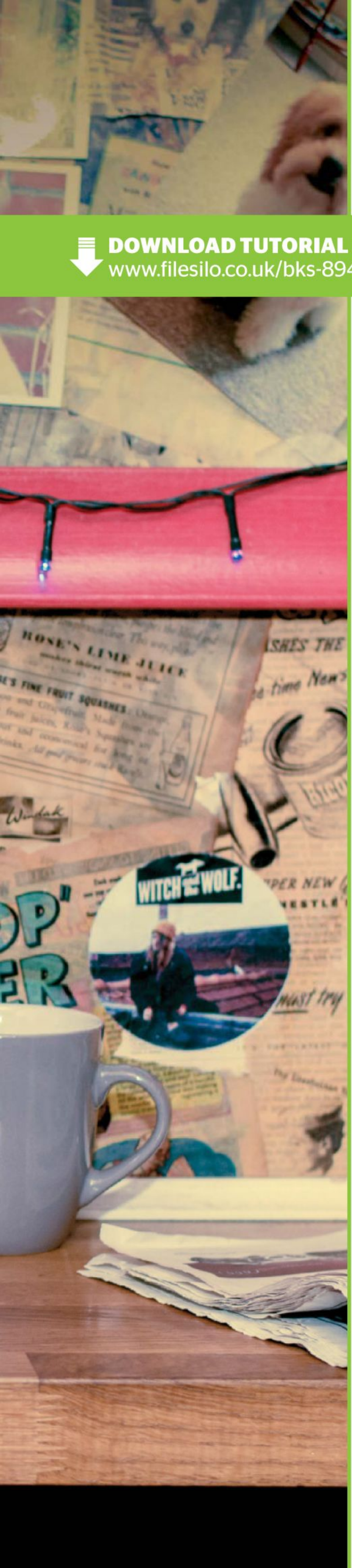
Inspiration

Weget

A cascading collection of HTML5 video clips forms a distinctive portfolio website for Weget, and you can learn how to reproduce their slide-in menu panels



<Right>
• Weget Enterprise's site is produced by Dan Ghimpu, Valentin Baranetchi (developers) and Alexandr Mocreac (art)



DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/bks-894

Designer Weget Enterprise
Site weget.nl/en

Weget is a digital and design studio based in the city of Groningen, Netherlands, offering a range of services. Whether you require essential web products for big or small business startups, eCommerce, web applications or hosting and domains – you'll find it here. Recognised for notable design work across numerous CSS award nominations, this unique portfolio experience details Weget's work in all languages. Content is arranged in a lengthy structure more akin to a header slider component than a traditional web page. These seven screen-width panels are defined by HTML5 <video> backgrounds, looping relevant clips (WEBM) of around 25 seconds long. Sparse snippets of text provide hints towards further information and a section of featured works. Most distinctive though is the navigational mode, prompting the use of the M key to pop out the main menu and the arrow keys to cycle down. Neat transitions then roll or flip the next panel into view, while on tablets and mobile a simple swipe gesture controls flow instead. "The goal for the website was to represent best our strongest points," explains Dan Ghimpu, CEO and project manager. "In order to achieve this, we chose to use video backgrounds for the most powerful visual impact, combined with a full-screen design, built [with] a block strategy. The gallery of the website had been designed and programmed separately, and then integrated. This was needed to complete the concept of the gallery, which didn't fit the main style of the website."

> "We chose to use video backgrounds for the most powerful visual impact"

abcABC
1234567890

<Above>
• Lato by Łukasz Dziedzic, found on Google Fonts, is used on header links within the footer section

abcABC
1234567890

<Above>
• Proxima Nova by Mark Simonson Studio is available via Typekit and is used in Bold across page headings

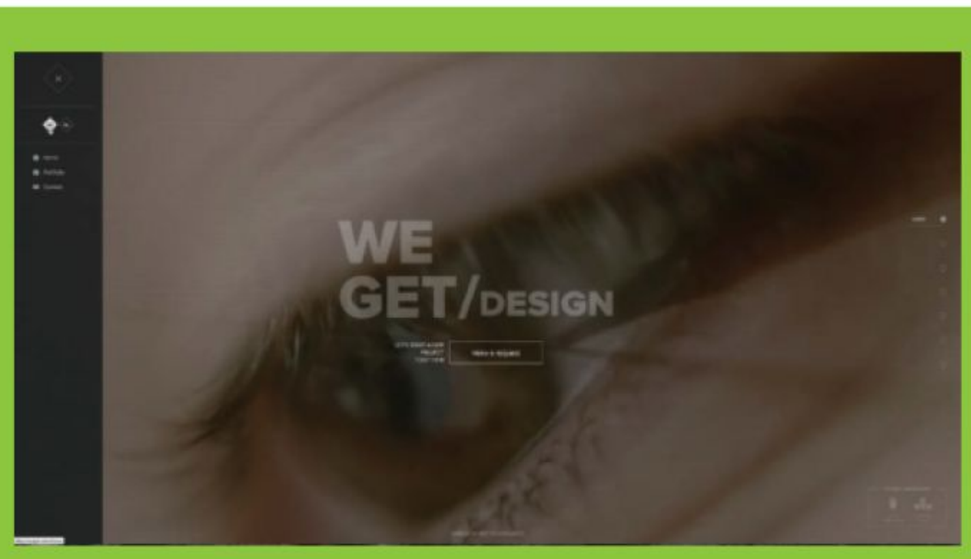
#42413B #285258 #34495E #280000 #006064

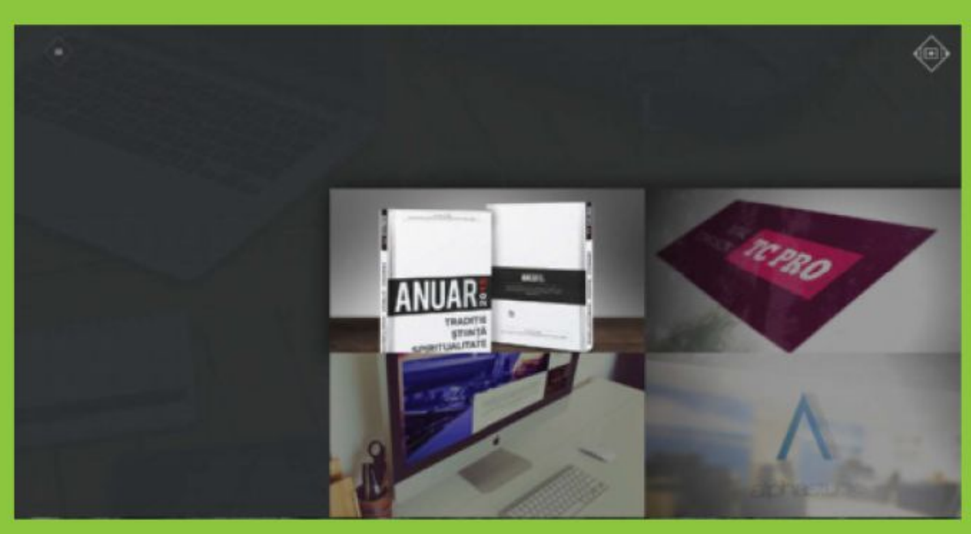
<Far right>
• The site's eight main sections (shown stitched together here), uses mouse scrolls or arrow keys to shuffle through

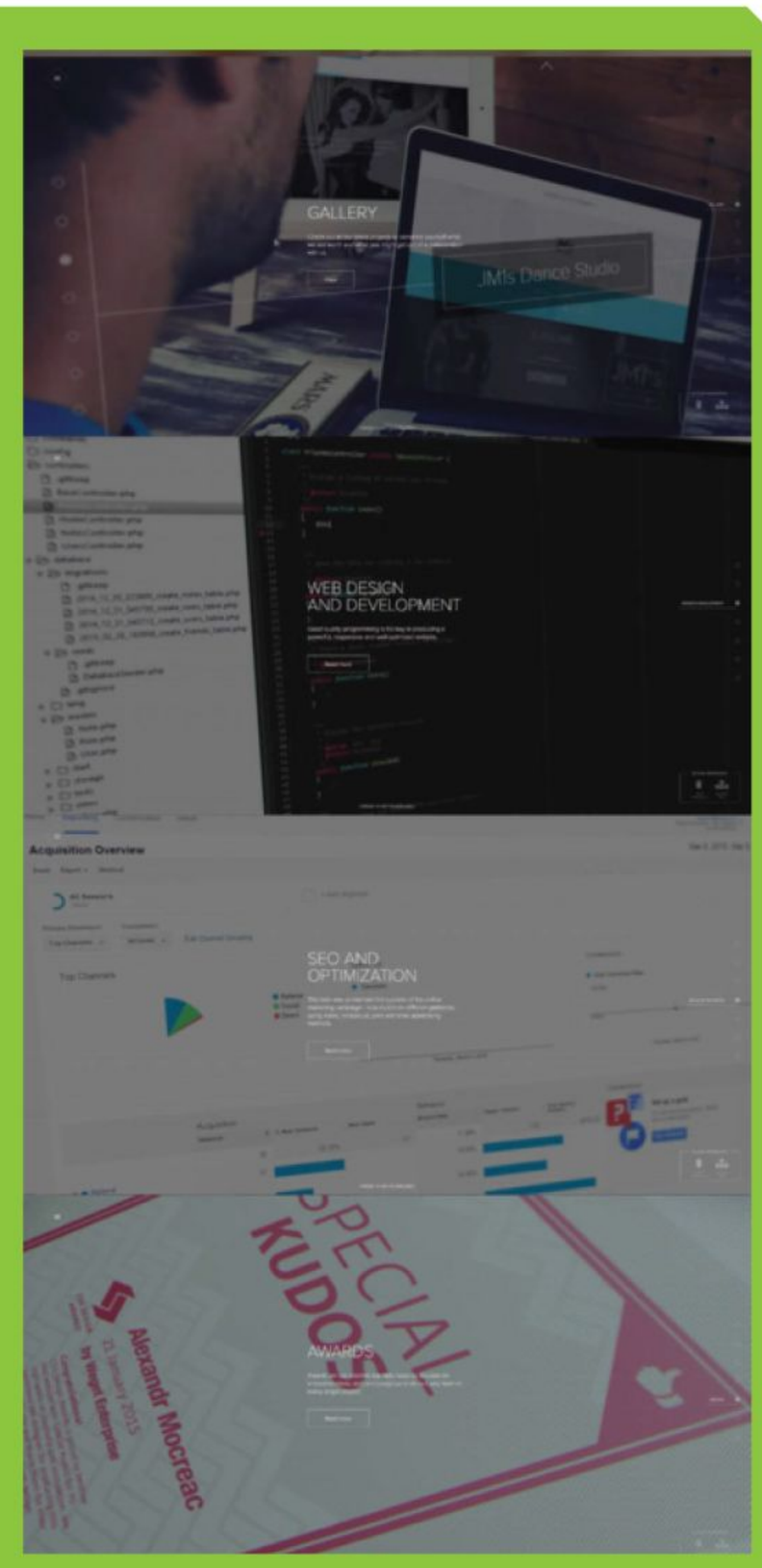
<Top right>
• A pop-out side menu on the left provides links to key sections but also the option to toggle languages

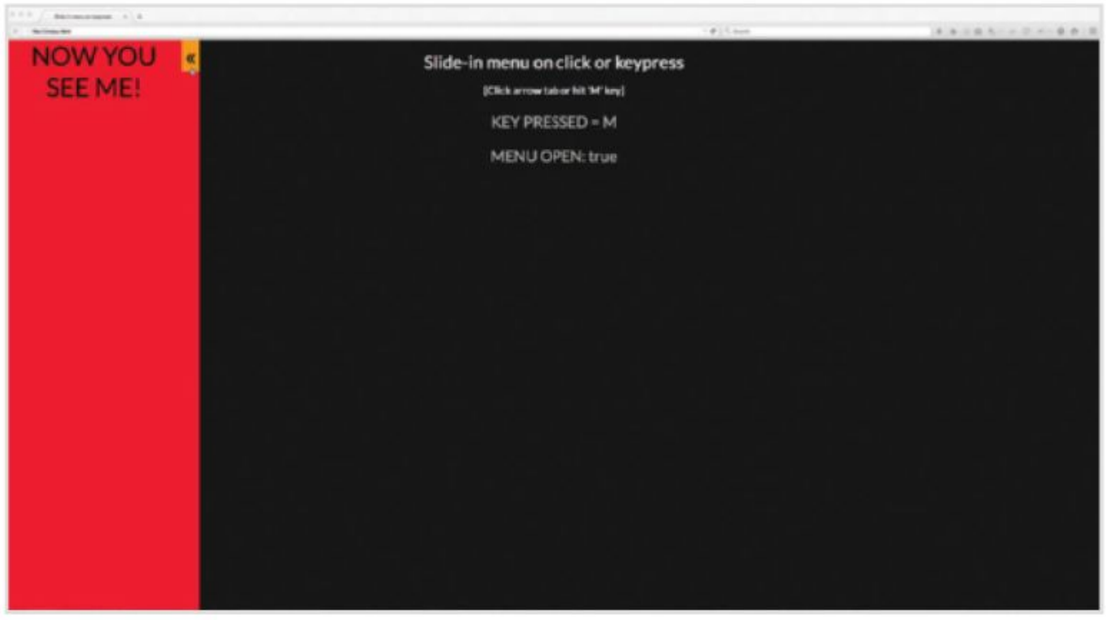
<Middle right>
• The backgrounds for each section use HTML5 video and looping clips in WEBM to add life and motion

<Bottom right>
• A Portfolio section provides access to a range of featured projects, with a range of rollover previews









Create slide-in menu panels with CSS3 and jQuery

Inspired by Weget.nl, we describe a solution for adding slide-in menu panels activated with both click and keypress events

1. Get started

In this example we will emulate Weget's slide-in navigation menu panel, using CSS3 transitions and jQuery to trigger open and closed states. This tutorial should be easy to follow for an experienced developer. The panel will be activated both on click and also via pressing the M key. Start with a new HTML document and attach the latest jQuery library to the <head> of the page. In addition we'll be styling our page text with Lato via Google Fonts:

```
001 <script src="http://code.jquery. com/
jquery-1.11.2.min.js"></script>      002 <link
href='http://fonts.
googleapis.com/css?family=Lato'
rel='stylesheet' type='text/css'>
```

2. Define the HTML elements

For page elements, we will set up a menu containing a pair of nested divs – a main panel and a small tabbed edge. We'll style these in Step 4 with the subsequent <h1>, <h2> and <p> tags (for clarification, these are labels for instruction and outputting event or variable states).

```
<body>
<div id="menu">
<div id="panel">NOW YOU<br>SEE ME!</div>
<div id="tab">&raquo;</div>
</div>
<h1>Slide-in menu on click or keypress</h1>
<h2>[Click arrow tab or hit 'M' key]</h1>
<p id="label1">KEY PRESSED: None</p>
<p id="label2">MENU OPEN: false</p>
</body>
```

3. Basic page styling

Moving on to the CSS, we willstart by styling optional body styles. Here we've opted for Weget-inspired background and text colours, along with the familiar use of the Lato font. Alongside this are sizing preferences for our various page labels. All of this is optional and included for illustration, as it has no bearing on the menu itself.

```
<style>
body {
font-family: 'Lato', sans-serif;
text-align: center;
background-color: #161616;
color: rgba(224, 224, 224, 0.8);
}
h1 {
font-size: 40px;
}
p {
font-size: 36px;
}
```

4. CSS menu element

The menu element contains two parts, the main panel which is hidden on close as well as the tabbed edge which remains visible. So in the CSS we will define styles for those divs, crucially setting the left property negatively to hide it in the left page. By setting a transition we effectively dictate how it will animate into view when we dynamically toggle on the open class and set left to zero:

```
#menu{
position: absolute;
top: 0;
bottom: 0;
left: -410px;
background-color:#F03;;
color: #000000;
font-size:60px;
-webkit-transition: all 0.4s ease;
transition: all 0.4s ease;
}
#menu.open{
left: 0;
}
#panel{
width: 410px;
float: left;
}
#tab{
width: 40px;
background-color: #F90;
cursor: pointer;
float: left;
}
</style>
```

5. Bind the key events

We will add our script just before the </body> tag and start by defining a Boolean variable as an open or closed flag. By default the menu will be closed so we will have to set it false. For our next step, we have a small function called on load which very simply binds keyup and click events to the page body and the menu's tab element. Upon either of these events we will then call an activated() function that will be defined next.

```
<script>
var menuOpen=false;
$(function(){
$("body").bind("keyup", activated);
$("#tab").bind("click", activated);
});
```

6. Invoke the menu transitions

The main activated() function then performs our menu actions. Additionally, it outputs variable states to our page labels, which is purely cosmetic to track the events. However, the main part checks the passed event object for an M press keyCode (77) or our bound click. If true we merely have to toggle the #menu.open class from our CSS before moving to flipping our Boolean flag. This will then be used to switch the tab arrow icon to either in or out. Now that that is all finished, you are all done! Enjoy your very own slide-in menu panels and don't forget to show it off on your newest project.

```
function activated(event){
$("#label1").text("KEY PRESSED = " + String.
fromCharCode(event.keyCode));
if(event.keyCode==77 || event.type=="click"){
$("#menu").toggleClass("open");
menuOpen=!menuOpen;
if(menuOpen){ $("#tab").html("&laquo;");} else
{
$("#tab").html("&raquo;");
};
$("#label2").text("MENU OPEN: " + menuOpen.
toString());
};
};
</script>
```

DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/bks-894

Manage JS with asynchronous tasks

Promise objects aim to remove redundant and/or excessive callback functions from background tasks

Using callback functions is a sure-fire way to induce JavaScript bloat: we know of some code examples where the accomplishment of a single task requires the presence of a dozen callbacks. Furthermore, callbacks are not a catch-all solution for event handling. As the use of exceptions spreads over to JavaScript, keeping an eye on the various error sources becomes tedious pretty quickly.

Even though Promises have not been fully formalised at the time of writing, they nevertheless provide a fascinating alternative to callbacks. In principle, a Promise is but a state machine object whose state reflects the situation of the task at hand.

Deploying Promises significantly reduces the efforts involved in callback and event handling. When done right, core functions can be observed at a glance. Furthermore, time becomes insignificant: if a Promise is invoked after the event it is intended to analyse has taken place, it will automatically transition through the necessary states.

As with all concurrent code, care must be taken to avoid race conditions and similar problems. So, let us be your guide and come along for a fascinating ride!

1. Set the stage

As Promises are not fully standardised at the time of writing, care must be taken to ensure a consistent working environment. The examples presented in this tutorial worked well on a Firefox 42.0 hosted on a 64-bit Ubuntu workstation.

2. Create a model

A Promise class is a wrapper around an asynchronous payload. To keep things 'simple', we will start out by creating a simple model of a good friend of ours. Matz will greets us with a friendly "Oi" message once a day – sadly, the time when the actual Oi takes place is not predictable – so it could occur at any time.

```
<html>
<body>
<script>
function Matz()
{
setTimeout(function(){ alert("Oi!"); },
(Math.random()*10000));
}
Matz();
</script>
</body>
</html>
```

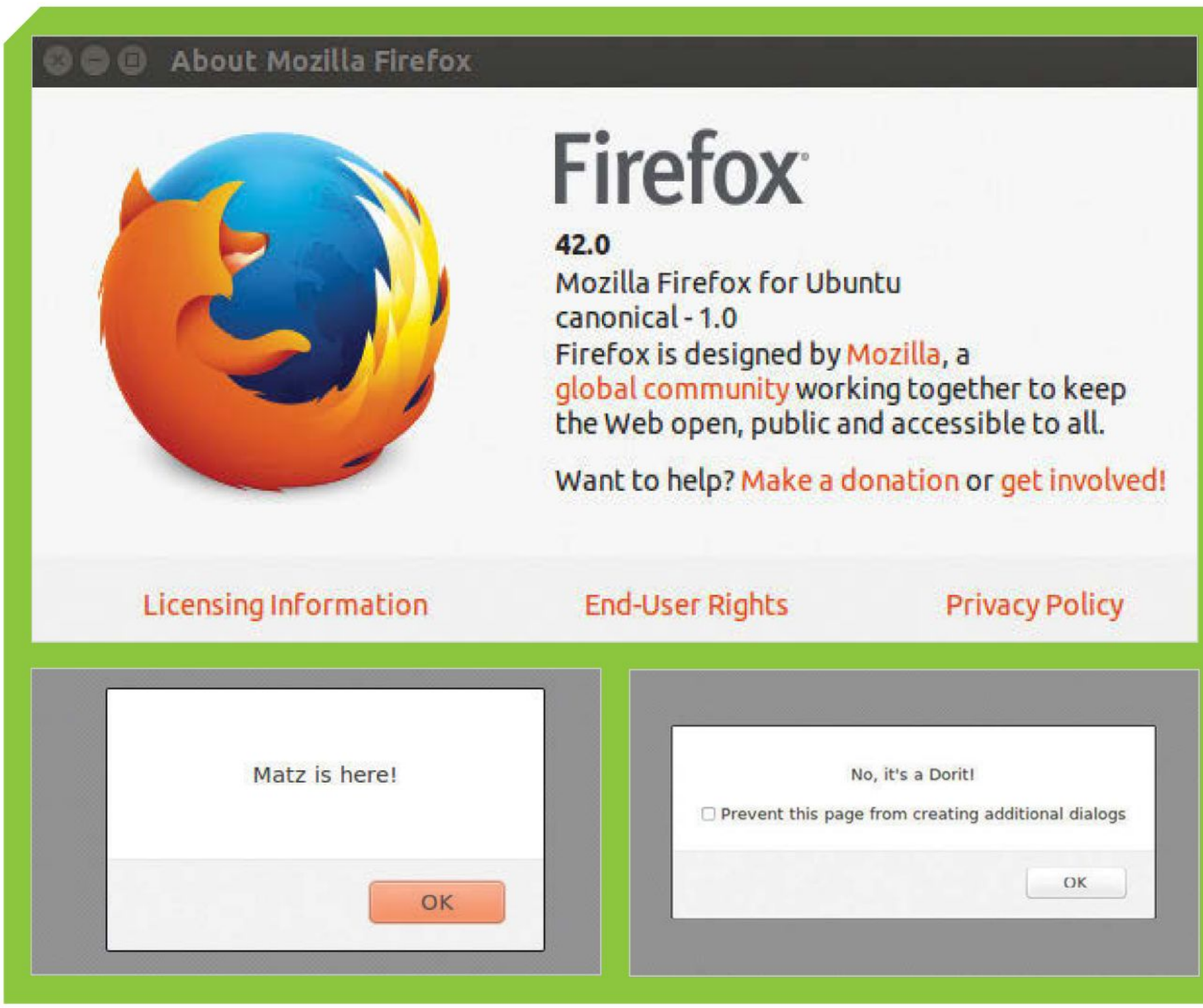
3. Adapt to Promise

Adapting our Matz payload to a Promise involves the creation of a method with two parameters. Both reject and resolve can be invoked as normal members, along with a value containing further information about the state of the event at hand.

```
<body>
<script>
var aMatz=new Promise(
function Matz(resolve, reject)
```

Don't block

Promises are not a way to perform multithreading. If your Promise contains blocking code, the JavaScript runtime will stall just as if it were found in a normal function or member.



<Top left>
• When push comes to shove, knowing the exact version of the browser where a program is being run tends to be really helpful

<Bottom left>
• The timing of the two alerts will be dependent on the "mood" of the random number generator

<Bottom right>
• By setting doritIsHere to true you will make Matz think twice about greeting passers-by – a situation modelled with reject()

```
{
setTimeout(function(){ resolve("Oi!"); },
(Math.random()*10000));
});
</script>
</body>
```

4. Announce events

Our Promise only actually becomes useful once we can ensure that our code reacts on its completion. The snippet accompanying this tutorial step announces the Greeting event by displaying a message box once the Oi has been emitted.

```
<body>
<script>
var aMatz=new Promise(
function Matz(resolve, reject)
{
setTimeout(function(){ resolve("Oi!"); },
(Math.random()*10000));
});
aMatz.then(function(){alert("Matz is
here!")});
</script>
</body>
```

Advanced functionality

Check out the Promise RSVP library offered at github.com/tildeio/rsvp.js

5. Transcendental Oi!

Classic event handlers suffer from a chicken/egg problem: if the interesting event occurs before the registration of the handler, information will be lost. As Promises act as state machines, this problem can not occur. Once Matz is done, it's done forever, which means that the late alert will always pops up.

6. Truly asynchronous

Then does not interrupt program execution. Instead, the method stores the payload in an execution queue and returns afterwards. Running the example accompanying Step 6 a few times will lead to different alert dialog sequences depending on the speed of your system and the random timeout value.

```
<body>
<script>
var aMatz=new Promise(
function Matz(resolve, reject)
{
setTimeout(function(){ resolve("Oi!"); },
(Math.random()*10000));
});
aMatz.then(function(){alert("Matz is
here!")});
alert("We're done here");
</script>
</body>
```

7. Dorit objects

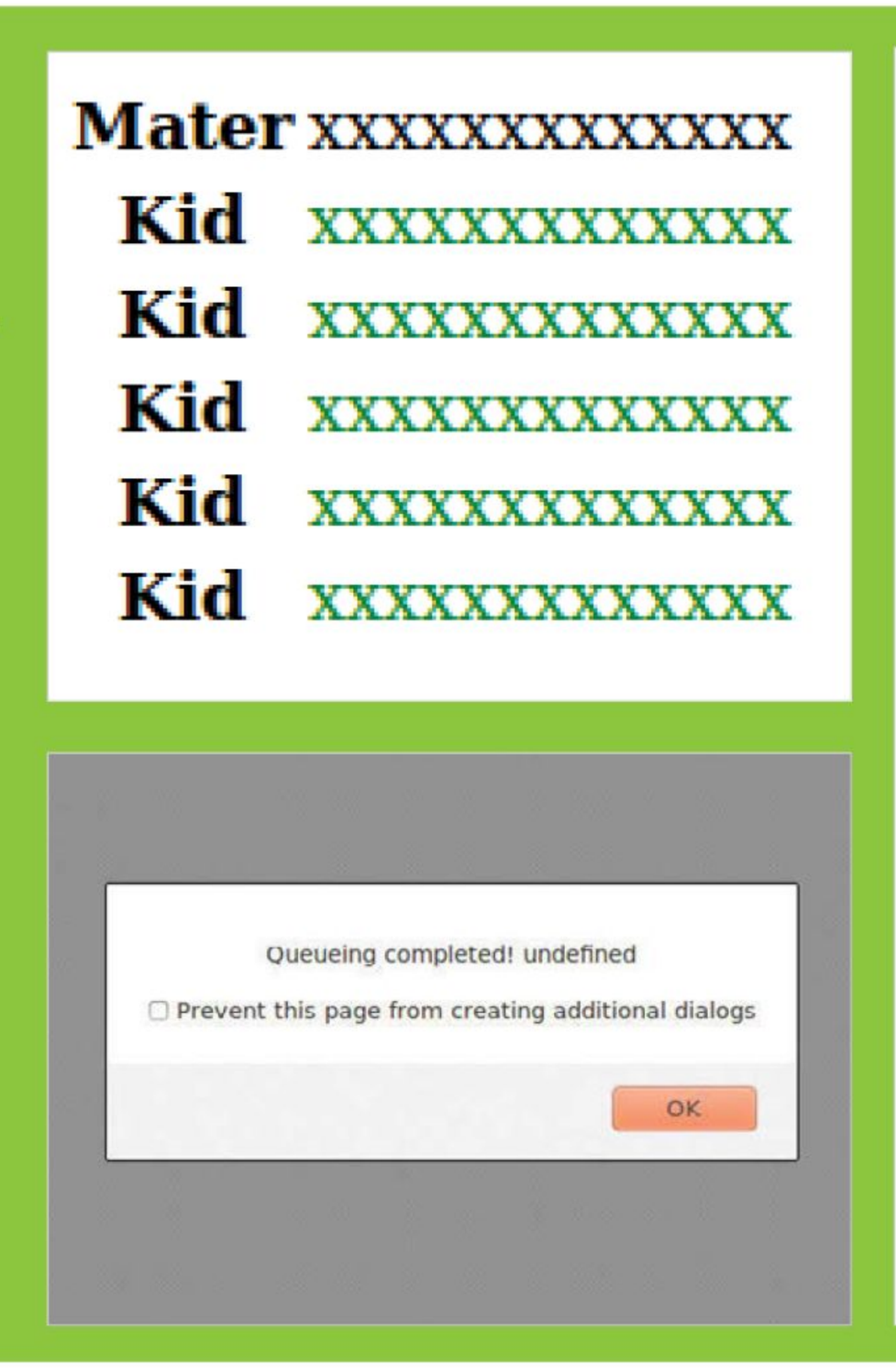
Real-world processes have the nasty habit of failing from time to time. We will model this via the presence of a Dorit object: if she is around, Matz gets intimidated and no longer greets passers-by. This is modelled via the reject method, which gets invoked instead of resolve:

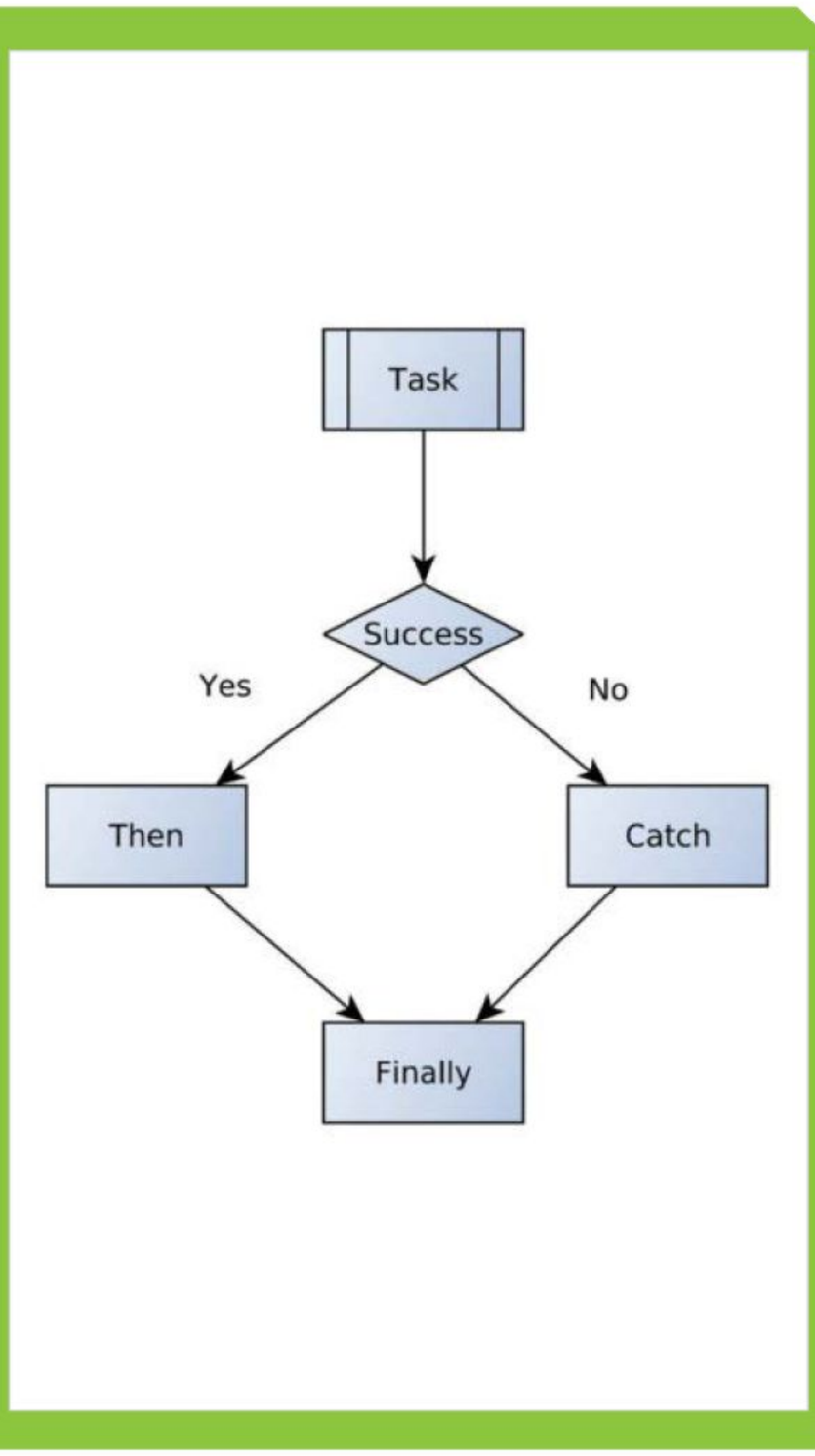
```
<script>
window.doritIsHere=true;
```

<Top left>
• The colours of the XXX-laden cells are changed as the individual Promises fire off their payloads

<Bottom left>
• Failed invocations of the then() payload are indicated by passing of undefined for val

<Right>
• A finaliser will be run regardless of the success of the monitored operation





```
~/Desktop/deadstuff/2015Oct/JenkinsMatz/17/index.htm - Sublime Text (UNREGISTERED)
OPEN FILES
index.htm
3       <script src="jquery-2.1.4.js"></script>
4       <script>
5
6
7
8       function AddClosure(what)
9       {
10          return function (resolve, reject)
11          {
12              setTimeout(function(){
13                  what.css( "color", "green" );
14                  resolve();
15              }, (Math.random()*10000));
16          };
17      }
18
19      $( window ).load(function() {
20          var e1=new Promise(AddClosure($("#k1")));
21          e1.then();
22          var e2=new Promise(AddClosure($("#k2")));
23          e2.then();
24          var e3=new Promise(AddClosure($("#k3")));
25          e3.then();
26          var e4=new Promise(AddClosure($("#k4")));
27          e4.then();
28          var e5=new Promise(AddClosure($("#k5")));
29          e5.then();
30
31          window.myArray=[e1,e2,e3,e4,e5];
32          Promise.race(window.myArray).then(function(){
33              $("#master").css( "color", "blue" );
34          });
35      });
36
37      </script>
38      <table>
39          <tr><th>Mater</th><td id="master">xxxxxxxxxxxxx</td></tr>
40          <tr><th>Kid</th><td id="k1">xxxxxxxxxxxxx</td></tr>
41          <tr><th>Kid</th><td id="k2">xxxxxxxxxxxxx</td></tr>
42          <tr><th>Kid</th><td id="k3">xxxxxxxxxxxxx</td></tr>
43          <tr><th>Kid</th><td id="k4">xxxxxxxxxxxxx</td></tr>
44          <tr><th>Kid</th><td id="k5">xxxxxxxxxxxxx</td></tr>
45      </table>
46  </body>
47  </html>
Line 31, Column 1
```

Do the closure

Setting parameters on the Promise object returned by the constructor is not a valid way to pass them into the function responsible for the actual handling of the payload. Instead, parameters must be passed using a closure which returns the function body. In the case of the code shown in Step 15, the element at hand is stored as a parameter passed to AddClosure. The inner function can access it as if it were a global variable, as a closure is erected around it during the returning of the function body. It will remain valid for the entire life cycle of that reference: as long as someone can envoke the function, its closure remains alive. JavaScript newbies should be aware of this behaviour, as it is the prime source of excessive memory consumption.

```
var aMatz=new Promise(
function Matz(resolve, reject)
{
setTimeout(function(){
if(window.doritIsHere==false)
{resolve("Oi!");}
else
{reject ("Eeek!")}
}, (Math.random()*10000));
});
aMatz.then(function(){alert("Matz is
here!")});
alert("We're done here");
</script>
```

8. Chain it up

Step 6 established that then() does not wait for the completion of the Promise. Thus, its returned context can be used to chain things up. If you look at our example in this tutorial, you will see that the payload passed to catch() is executed only if Dorit's presence will cause Matz to bolt.

```
<html>
<body>
<script>
window.doritIsHere=true;
var aMatz=new Promise(
. . .
aMatz.then(function(){alert("Matz is
here!")}).catch(function(){alert("No, it's a
Dorit!")});
alert("We're done here");
</script>
</body>
</html>
```

9. Clean me up

Developers who are already familiar with object-oriented programming languages (also known as OOP to most developers) will recognise the basic concept of a finaliser: it is a method that gets called on both success and failure of the underlying operation. Its position in program execution is represented in the flowchart accompanying this step in the tutorial (see the boxout on page 80, right image).

10. Implement the finaliser

With that, it is now time to get started with your coding. Adding a finaliser is as quick and easy as invoking the final method at the end of the call queue: the execution engine will ensure that its payload gets invoked once the then or the catch payloads were given an opportunity to run. Unfortunately, this feature isn't supported on the Firefox browser yet, and thus will require the use of external libraries.

```
<script>
window.doritIsHere=true;
var aMatz=new Promise(
. . .
aMatz.then(function(){alert("Matz is
here!")}).catch(function(){alert("No, it's a
Dorit!")}).finally(function(){alert("It's
done!")});
alert("We're done here");
```

11. Cascade the processes

In many cases, one asynchronous process kicks off another one. This is one of the main issues in classic JavaScript, as the number of callbacks raises with a growth rate of O(n)=2*n. Fortunately, a Promise can be made to return another one:

```
{
setTimeout(function(){
if(window.doritIsHere==false)
{resolve("Oi!");}
else
{reject ("Eeek!")}
}, (Math.random()*10000));
}
var matzBuilder=new Promise(function
MatzBuilder(resolve, reject)
{
setTimeout(function(){
alert("Spawning new Matz!");
resolve(new Promise(Matz));
},1500);
});
```

12. Gang them up, part one

Let's show our MatzRaiser in action. The following test fixture demonstrates how multiple then commands are chained. By the way: the 'linear' indentation of the individual methods is recommended in the standard.

```
var matzBuilder=new Promise(function
MatzBuilder(resolve, reject)
{
setTimeout(function(){
alert("Spawning new Matz!");
resolve(new Promise(Matz));
},1500);
});
matzBuilder.then().then(function(val)
{alert("Queueing completed! " + val);})
```

13. Gang them up, part two

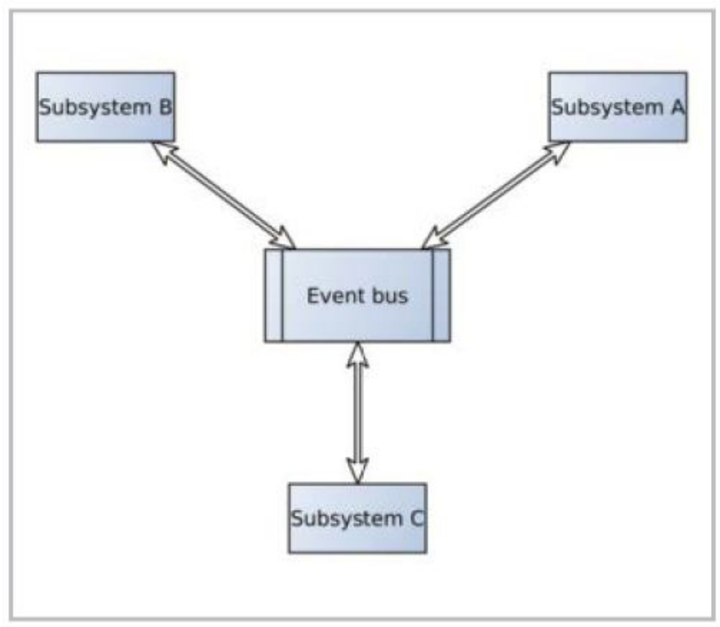
If an error occurs in the "inner" part of the Promise and is not handled in the first part of the then() chain, the second part handles it automatically. This little snippet handles the Matz error in the instantiation of the mage.

```
<script>
window.doritIsHere=true;
matzBuilder.then(undefined,function(){
alert("Unhandled Inner Error")
}).then(function(val){
alert("Queueing completed! " + val);
});
</script>
```

14. Promise arrays

Pictures must be downloaded in bulk. We will 'simulate' this problem via a Promise which – eventually – changes the colour of a table cell assigned to it:

```
<body>
<script src="jquery-2.1.4.js">
</script>
<script>
```




```
function DeltaColor(resolve, reject)
{
setTimeout(function(){
this.myElem.css( "color", "green" );
}, (Math.random()*10000));
}
</script>
</body>
```

15. Set it up!

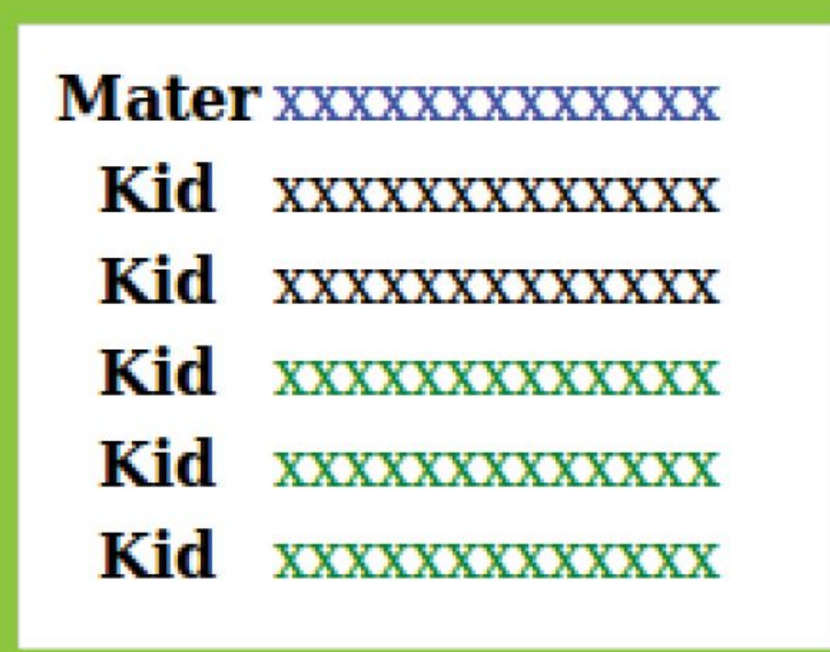
Demonstrating our code in action requires the presence of a table containing a batch of red cells. Furthermore, one "master" cell is created. Finally, each cell is assigned with one instance of the Promise generated in step 14.

```
function AddClosure(what)
{
return function (resolve, reject)
{
setTimeout(function(){
what.css( "color", "green" );
resolve();
}, (Math.random()*10000));
};
```

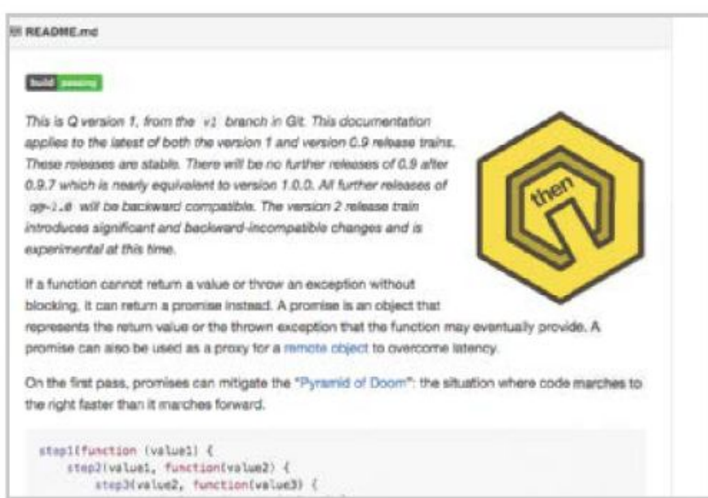
<Bottom right>
• If race() is used, one colour change is enough to trigger the modification of the mater element

<Bottom left>
• Radio, available from radio.uxder.com, allows you to create event chain systems quickly and efficiently

<Top>
• Promises are not – and probably will not be – supported in Internet Explorer (Credit: caniuse.com)



```
}
$( window ).load(function() {
var e1=new Promise(AddClosure($("#k1")));
e1.then();
var e2=new Promise(AddClosure($("#k2")));
e2.then();
var e3=new Promise(AddClosure($("#k3")));
e3.then();
var e4=new Promise(AddClosure($("#k4")));
e4.then();
var e5=new Promise(AddClosure($("#k5")));
e5.then();
window.myArray=[e1,e2,e3,e4,e5];
});
</script>
<table>
<tr><th>Mater</th><td
id="master">xxxxxxxxxxxxx</td></tr>
<tr><th>Kid</th><td id="k1">xxxxxxxxxxxxx</
td></tr>
<tr><th>Kid</th><td id="k2">xxxxxxxxxxxxx</
td></tr>
<tr><th>Kid</th><td id="k3">xxxxxxxxxxxxx</
td></tr>
<tr><th>Kid</th><td id="k4">xxxxxxxxxxxxx</
td></tr>
<tr><th>Kid</th><td id="k5">xxxxxxxxxxxxx</
td></tr>
</table>
```



16. Wait for all

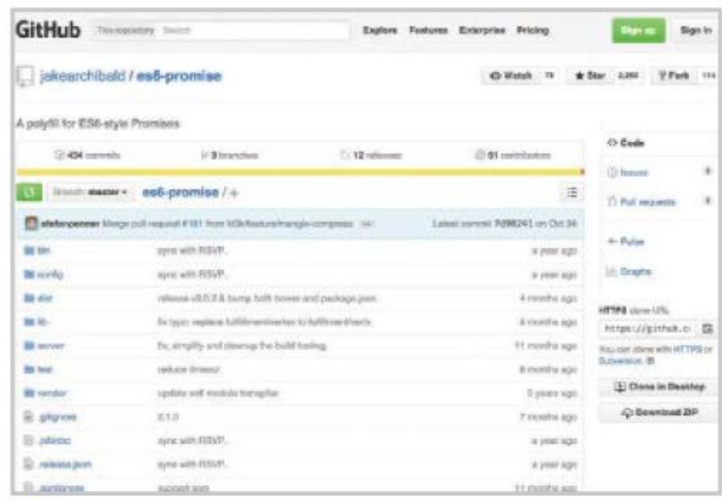
One especially interesting convenience function found in the standard involves awaiting the completion of an entire array of Promises. In our example, the "master" cell gets coloured only after all the children have achieved readiness by changing their colours.

```
. . .
window.myArray=[e1,e2,e3,e4,e5];
Promise.all(window.myArray).then(function(){
$("#master").css( "color", "blue" );
});
});
```

17. Wait for one

If work should proceed after one of the Promise objects contained in the array is completed, the test harnish must be modified along the lines shown in the code accompanying this step.

```
window.myArray=[e1,e2,e3,e4,e5];
Promise.race(window.myArray).then(function(){
$("#master").css( "color", "blue" );
});
});
```



18. Abbreviated then

Most then() invocations will contain a success and a failure clause. The Promises standard contains a special method that takes two parameters – this will then prevent you from having to painstakingly invoke the failure method separately.

```
p.then(onFulfilled, onRejected);
p.then(function(value) {
// fulfillment
}, function(reason) {
// rejection
});
```

19. Antipattern: recurring events

Promises change their state but once: in the example shown, our Matz could say Oi once. Trying to shoe-horn Promises into the role of a classic event bus as the one shown in the image is not recommended – other tools – like Radio.js – provide a faster, slimmer and more effective solution.

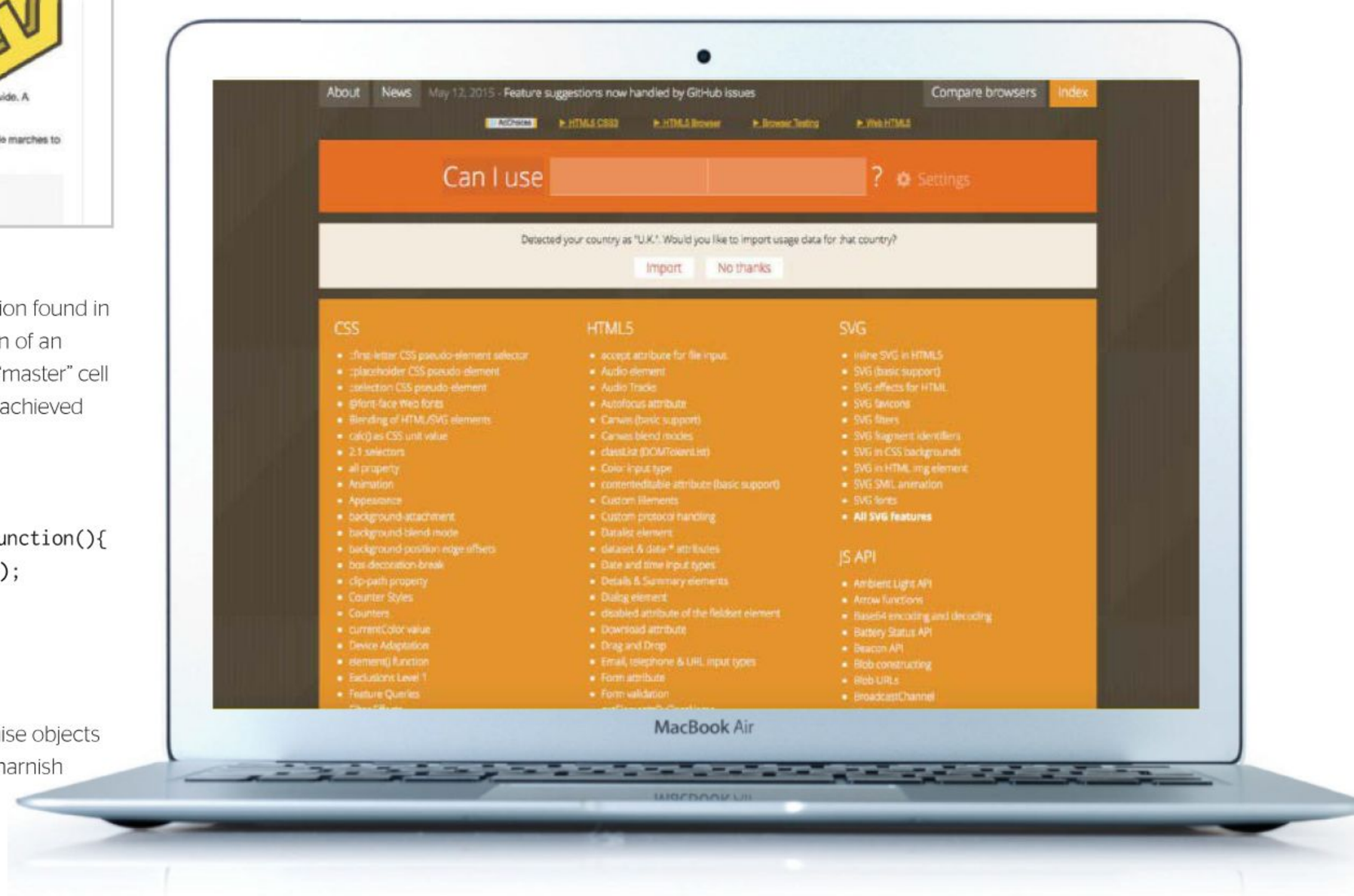
20. Cancel and Query

The basics of UI design dictate that users should be provided with progress information to 'shorten' the wait. Sadly, this feature is not implemented in the A+ standard. Instead, developers are recommended to use the Q library which, incidentally, also supports the termination of Promise payloads.

21. Promises for IE

Analysing the compatibility of Promises on CanIUse demonstrates that all versions of Internet Explorer do not support it. Fortunately, a large variety of alternative libraries are available. ES6-Promise has become a bit of a quasi-standard due to its small size – more information on this awaits you at **github.com/jakearchibald/es6-Promise**.

> The Promises standard contains a special method taking two parameters



DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/bks-894

Robust JS code with TypeScript library

Structure your JS code more rigidly than ever before with the help of Microsoft's TypeScript

JavaScript's liberal syntax declaration lets developers do all kinds of weird and wonderful things. Programmers used to restrictive class-based systems will need to aquaint themselves to the new working environment.

Google engineer Addy Osmani collected design patterns which helps this process – when, and only when, applied correctly. Duck typing is even worse though: it permits the creation of bugs which occur only when the code path in question is executed. Murphy's law then ensures that this will happen on the computer of an important client. TypeScript solves these problems by creating a statically typed dialect of JavaScript which is verified via a more stringent execution environment. Transgressions are punished mercilessly, defective code never makes it to the browser.

Microsoft does its magic via a process called transpilation. A special compiler transforms the TypeScript code to vanilla JavaScript, which can then be run in a JavaScript VM of choice. State-of-the-art JavaScript runtimes are so fast that they can be used as execution environment for arbitrary code – some developers even go as far as to transpile C++ to a JavaScript-based VM.

1. Install Visual Studio

Even though TypeScript can also be run from the command line, we will use Microsoft's free version of Visual Studio 2015. Download it by visiting **visualstudio.com/en-us/products/visual-studio-community-vs.aspx** and install it like any other application.

2. Create a project

Click New Project in order to open the new project wizard. Then proceed to opening the TypeScript tab, where a new solution based on the HTML application with TypeScript template is to be created. It will contain an HTML file and an accompanying TS file containing the actual TypeScript code.

3. Wire a button

The following steps will demonstrate various features of the runtime via an ever-expanding method. It must be triggered from a button – open the HTML file and append the markup shown in the source code accompanying this step. Then, replace the window.onload block in the TS file with the function specified.

```
<body>
<h1>TypeScript HTML App</h1>
<button onclick="btnClicked()">Button clicked</button>
</body>
//snip
function btnClicked()
{
alert("Hello TS");
}
```

4. A question of typing

TypeScript takes its name from its incredible typing capabilities. The snippet accompanying this step in the tutorial shows a few variable declarations. Be aware though, that variables don't necessarily need to have a

Vetoing Microsoft

Developers harbouring a distaste for Visual Studio can use a Node-based transpiler to transform their TypeScript code into JavaScript. Sadly, doing so means missing out on various comfort features.

```
<!DOCTYPE html>

<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>TypeScript HTML App</title>
    <link rel="stylesheet" href="app.css" type="text/css" />
    <script src="app.js"></script>
</head>
<body>
    <h1>TypeScript HTML App</h1>
    <button onclick="btnClicked()">Button clicked</button>
</body>
</html>
```

```
function btnClicked()
{
    var myString: String;
    var myNumber: Number;
    var myBool: Boolean;
    var myAny: any;
    myBool = "Hello World";
    myString = 22;
}
```

(local var) myString: String

Type 'number' is not assignable to type 'String'.
Property 'charAt' is missing in type 'Number'.

<Top left>
• Visual Studio Community combines the formerly separate web, Windows RT and desktop versions of Microsoft's popular IDE

<Bottom left>
• TypeScript code resides in TS files. Visual Studio will do the job of transforming them into temporary JavaScript during the deployment of the web app

<Bottom right>
• The transpilation process mercilessly weeds out all kinds of errors which traditional JavaScript developers can find only at runtime

type – if any is specified, the variable in question will not be subject to input validation.

```
function btnClicked()
{
var myString: String;
var myNumber: Number;
var myBool: Boolean;
var myAny: any;
}
```

5. Validation

TypeScript's capabilities can be validated by assigning invalid elements to the variables created in the previous step. Visual Studio will then flag them down the very moment the file has successfully been saved. It will, furthermore, prevent deployment until any and all objectionable passages have been remedied.

6. Create an enum

If variables are to contain but a few predefined values, using an enum is the way to go. The snippet accompanying this step demonstrates the definition of an enum handling aircraft types. After that, a new instance of the enum is created and a value is assigned in order to demonstrate its handling.

```
enum Aircraft {
MIG21,
MIG25,
MIG29,
MIG31
}
function btnClicked()
{
var plane: Aircraft;
plane = Aircraft.MIG25;
}
```

7. Create a class

For this next step, we will now start working on real classes. Remove the declaration of the greeter class, and then subsequently replace it with the following bit of code. Our AircraftManager class contains a member variable, a member function as well as a constructor which is used to set its value during the initialisation of the object instance.

```
class AircraftManager {
myWhatAmI: Aircraft;
constructor(aWhat:Aircraft) {
this.myWhatAmI=aWhat;
}
sayHello() {
alert("Hello");
}
}
```

8. Spawn an instance

The process of pawning an instance is as easy as invoking the new operator on the class name, which can furthermore be used as a variable type. By and large, any objects that are spawned from TypeScript classes behave just like normal JavaScript objects – accessing their members and member functions via the . operator is a non-brainer.

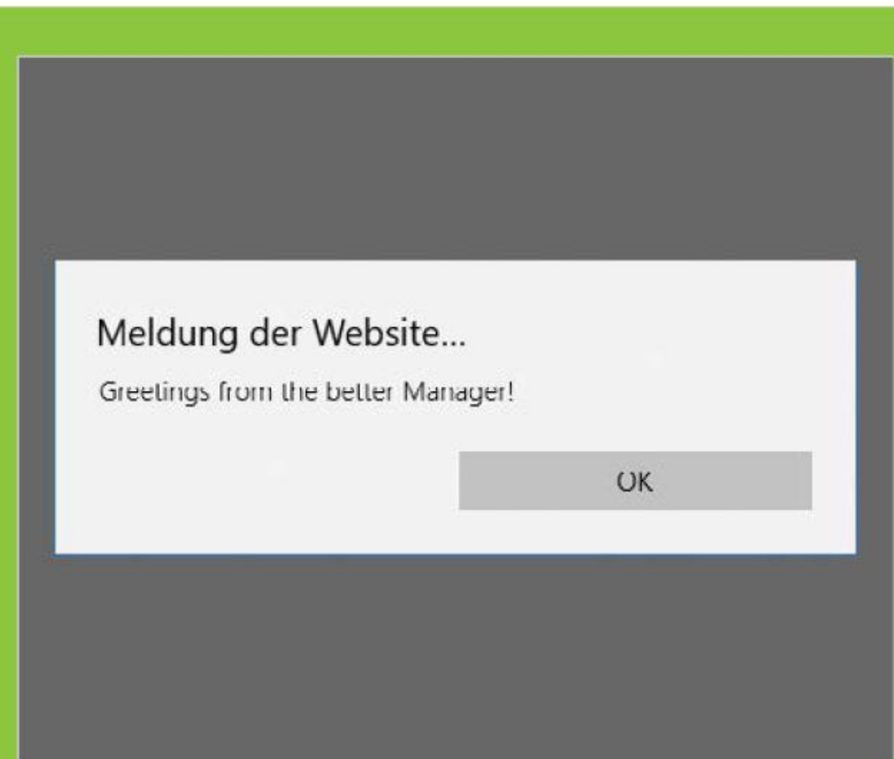
The playground

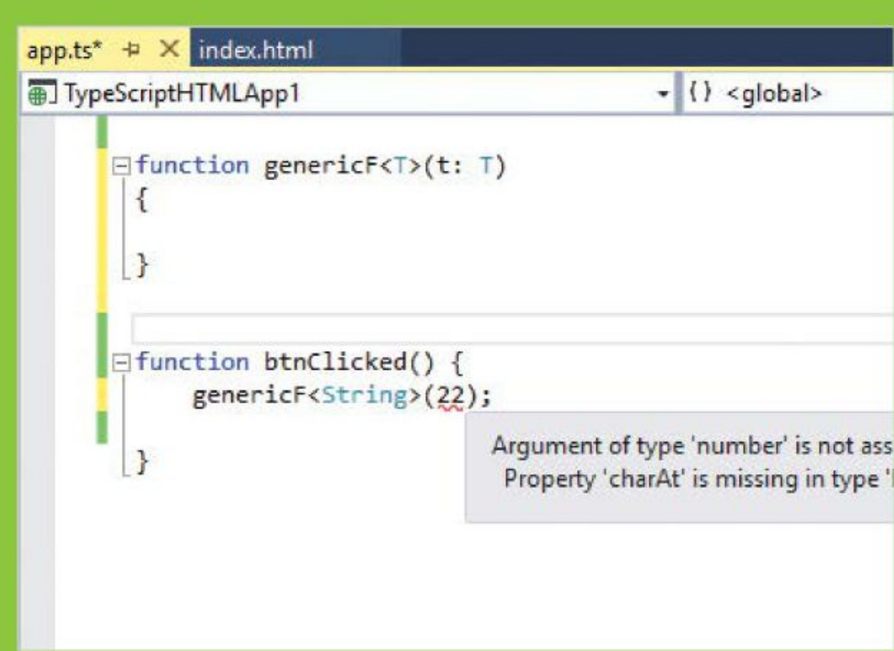
Individuals adverse to both Visual Studio and Node.js can visit the TypeScript Playground (**typescriptlang.org/Playground**). It provides a hosted version of the TypeScript transpiler.

<Top left>
• Thanks to inheritance, the BetterManager can easily change the message that is displayed in its instance's sayHello() function

<Bottom left>
• Invoking a generic function with an invalid parameter is punished with yet another transpiler warning

<Right>
• Using the private keyword permits developers to hide implementation details from pesky callers that want to wreak havoc in their code





Type	Contains what
Any	Accepts any JavaScript variable, thereby disabling TypeScript type checking
Array	An Array of elements
Boolean	True or False
Enum	An enumeration of numeric values, adressible via their assigned names
Number	A numeric value
String	Text
Void	Designates that a method returns nothing, and thus can not stand on the right side of an =

To type where no one typed before

The picture accompanying this step provides a rough overview of the types found in TypeScript. Developers experienced with Java or C will note the lack of specific numeric types: this is due to the reliance on the common JavaScript interpretation core. It cannot keep integers and floats apart, the weakness propagates to TypeScript at runtime. Strings share a similar fate – they are but a bit of syntactic sugar attached to a normal JavaScript var.

Void is interesting from a psychological standpoint: it informs the compiler that the element at hand will return nothing. This bit of metadata is helpful as it ensures that such methods never find themselves on the right-hand sign of an equals operator.

> TypeScript treats all members of a class as public by default. This can be modified by making use of a private keyword, which can be appended to both member functions and variables

```
function btnClicked()
{
var planeManager: AircraftManager;
planeManager = new AircraftManager(Aircraft.
MIG21);
planeManager.sayHello();
}
```

9. Hide a member

TypeScript treats all members of a class as public by default. This behaviour can be modified by making use of the private keyword. This keyword can be appended to both member functions and variables – in both cases, external access is no longer permitted by the transpiler.

```
class AircraftManager {
private myWhatAmI: Aircraft;
constructor(aWhat:Aircraft) {
this.myWhatAmI=aWhat;
}
private sayHello() {
alert("Hello");
}
}
```

10. Yield results

TypeScript's zealotic quality control algorithms, of course, can also make amends for all kinds of class-related oddness. We can try this out with two methods: by attempting to invoke the constructor with a wrongly typed parameter or via an access attempt addressed at a private variable.

11. Check values

Accessors are among some of the oldest structuring aides known to programmers. Our snippet that can be seen below demonstrates the adding of a property which checks the values passed into it before committing them to the data store found inside the class. The getter could, in theory, also be expanded to modify the values returned.

```
class AircraftManager {
private _WhatAmI: Aircraft;
get myWhatAmI(): string
{
return "An Aircraft";
}
set myWhatAmI(what: string)
{
if (what == "MIG21")
{
this._WhatAmI = Aircraft.MIG21;
```

```
}
}
```

12. Function cut short

Typing time is one of the most important 'time wasters' that can be encountered during software development. TypeScript permits you to set sensible defaults for parameters, which can then be invoked with a shorter parameter list. Please be aware though that all non-optional parameters will need to be in front of the optional ones.

```
class AircraftManager {
private _WhatAmI: Aircraft;
. . .
constructor(aWhat: Aircraft = Aircraft.
MIG21) {
this._WhatAmI=aWhat;
}
. . .
```

13. Creation of parameters

Functions with a variable number of parameters can be extraordinarily helpful at solving some rarely encountered problems. TypeScript provides a facility which makes the creation of variable-parameter functions really easy – all you have to do is take a look at the code accompanying this step below.

```
{
return first + " " + rest.join(" ");
}
```

14. Shared static elements

Static elements are shared between all instances of a class: although this commonly maligned pattern could potentially lead to some brittle code if it is overused. However, static elements can also be really helpful when it is actually applied correctly. TypeScript delights its users with the presence of a static keyword, and this behaves as expected:

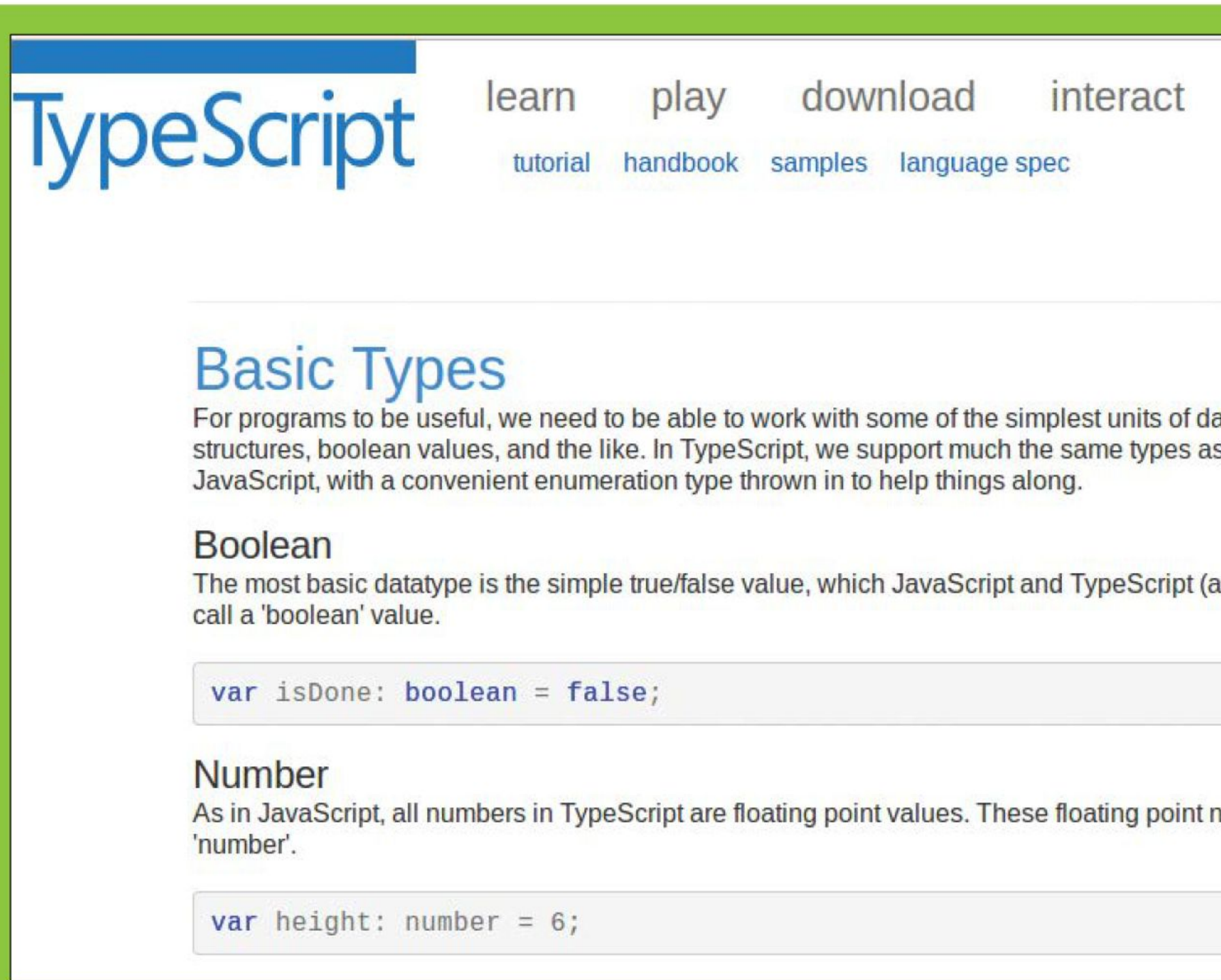
```
class AircraftManager
{
. . .
static sayHello() {
alert("Hello");
}
}
function btnClicked()
{
AircraftManager.sayHello();
}
```

> Static elements can be really helpful when applied correctly. TypeScript delights its users with the presence of a static keyword, and this behaves as expected

Learn more about TypeScript

Even though TypeScript is not particularly complex, it cannot be covered in five pages worth of tutorial. Developers interested in finding out more about their working environment should visit the TypeScript tutorial found at **typescriptlang.org/Handbook**.

Microsoft's endorsement is helpful in that it ensures the presence of a lively and active community: many if not most questions can be answered via Google. Newbies and professionals facing doubts should visit StackOverflow in order to find help and consultation from their peers.



15. Inheritance

Making classes inherit from one another simplifies the modelling of real-world relationships. Derived classes can, furthermore, overwrite the behaviour of their hosts – for example, the BetterManager replaces the message that is shown by the normal sayHello() function if invoked,

```
class AircraftManager {
. . .
sayHello() {
alert("Hello");
}
}
class BetterManager extends AircraftManager
{
sayHello()
{
super.sayHello();
alert("Greetings from the better Manager!");
}
}
function btnClicked()
{
var aManager: BetterManager;
aManager = new BetterManager();
aManager.sayHello();
}
```

16. Access mother object

TypeScript does not hide the mother object instance from view. It can, instead, be accessed via the super keyword. Our snippet below demonstrates the usage of super in a constructor and in a member function – it could not be easier.

```
class BetterManager extends AircraftManager
{
sayHello()
{
super.sayHello();
alert("Greetings from the better Manager!");
}
}
```

17. Enforce presence via interfaces

JavaScript's duck typing is a never-ending source of pain and this is because everything is valid for everything, that is until a NullReferenceException occurs though. Using an interface enables developers to specify the presence of member variables and/or functions – and any elements that are not confirming are therefore not allowed to pass.

```
interface AnObject
{
myName: String;
myNumber: Number;
}
function worker(a: AnObject) { }
function btnClicked()
{
var anObject = { myName: "AnElement",
myNumber: 22 };
worker(anObject);
}
```

18. Make parts optional

An old adage states that exceptions can prove the validity of rules. Interfaces can be configured to contain optional members and of course these don't necessarily need to be implemented. This feature, unfortunately, is not particularly helpful for us and that's because crashes can occur if the callee forgets to check the presence of the implementation.

```
interface AnObject
{
myName: String;
myNumber?: Number;
}
function worker(a: AnObject) { }
function btnClicked()
{
var anObject = { myName: "AnElement"};
worker(anObject);
}
```

19. Implement interfaces

Classes can be designated as implementations of particular interfaces. This is accomplished thanks to use of the implements keyword: if 'implements' is present and the class misses declarations that are required in the interface then what happens is that a compiler error will be raised in order to notify the developer about his unforgiveable omission.

```
interface AnObject {
myName: String;
myNumber?: Number;
}
class SomeClass implements AnObject
{
myName: String;
myNumber: Number;
}
```

20. Modularise

The TypeScript transpiler does not merge the individual parts of a library into one file – instead, each TS file gets transformed into an individual JS file. Do be aware though that each JS file must be included into the website separately in order to be able to use the module's content.

21. Go generic

Developers can side-step the type-checking process and all they have to do to do this is make use of the Any keyword. The use of generics provide a safer way to creating type-agnostic classes and/or functions: the parameter informs the compiler about the type that is going to be used by a particular instance, and this can then be enforced with zealotism.

```
function genericF<T>(myVal: T)
{
return myVal;
}
function btnClicked()
{
genericF<Number>(22);
genericF<String>(22);
}
```

22. Mixin on the loose

Keep in mind that creating complex inheritance structures is not always the best solution. Mixins are 'building block classes' which provide a small set of functionality, and these mixins can then be integrated into larger classes. As an example, let us take a bit of code from the documentation which creates activity management logic.

```
class Activatable {
isActive: boolean;
activate() {
this.isActive = true;
}
deactivate() {
this.isActive = false;
}
}
```

23. Use applyMixing

Mixins are instantiated via the implements keyword. The host class must contain a stub implementation, and this is then overwritten at runtime by making use of the applyMixing method.

```
class SmartObject implements Disposable,
Activatable {
. . .
// Activatable
isActive: boolean = false;
activate: () => void;
deactivate: () => void;
}
applyMixins(SmartObject, [Disposable,
Activatable])
var smartObj = new SmartObject();
setTimeout(() => smartObj.interact(), 1000);
```

24. Mix it up

Now it is time for our very last step in this tutorial. The actual deployment of the mixin must be done at runtime via the function shown in the code snippet shown below. Due to the complexity of the mixins, it's very important for you to try and consult documentation before proceeding further.

```
function applyMixins(derivedCtor: any,
baseCtors: any[]) {
baseCtors.forEach(baseCtor => {
Object.getOwnPropertyNames(baseCtor.
prototype).forEach(name => {
derivedCtor.prototype[name] = baseCtor.
prototype[name];
})
});
}
```

Development

088 15 Google developer secrets

Discover the hidden features and tools that can make your life so much easier

094 Get free web hosting with GitHub

We love GitHub almost as much as we love the word "free", so this was bound to be a hit

098 Make a resizable sliding panel

Insert this effect into your projects for a little extra oomph

100 Inspiration: Project Schoolkrant

This interactive website for Dutch school children has a few tricks up its sleeve

> Software truly is an unique product in that it can easily be multiplied with minimal cost

104 Create suberb split-screen sliding panel effects

The perfect marriage between form, design and usability, these sliding panels are ideal for any website

15
Google
developer secrets
revealed

Master Google's best APIS

THE SEARCH GIANT IS A TREASURE TROVE OF APIS, READ ON FOR THE BEST ONES

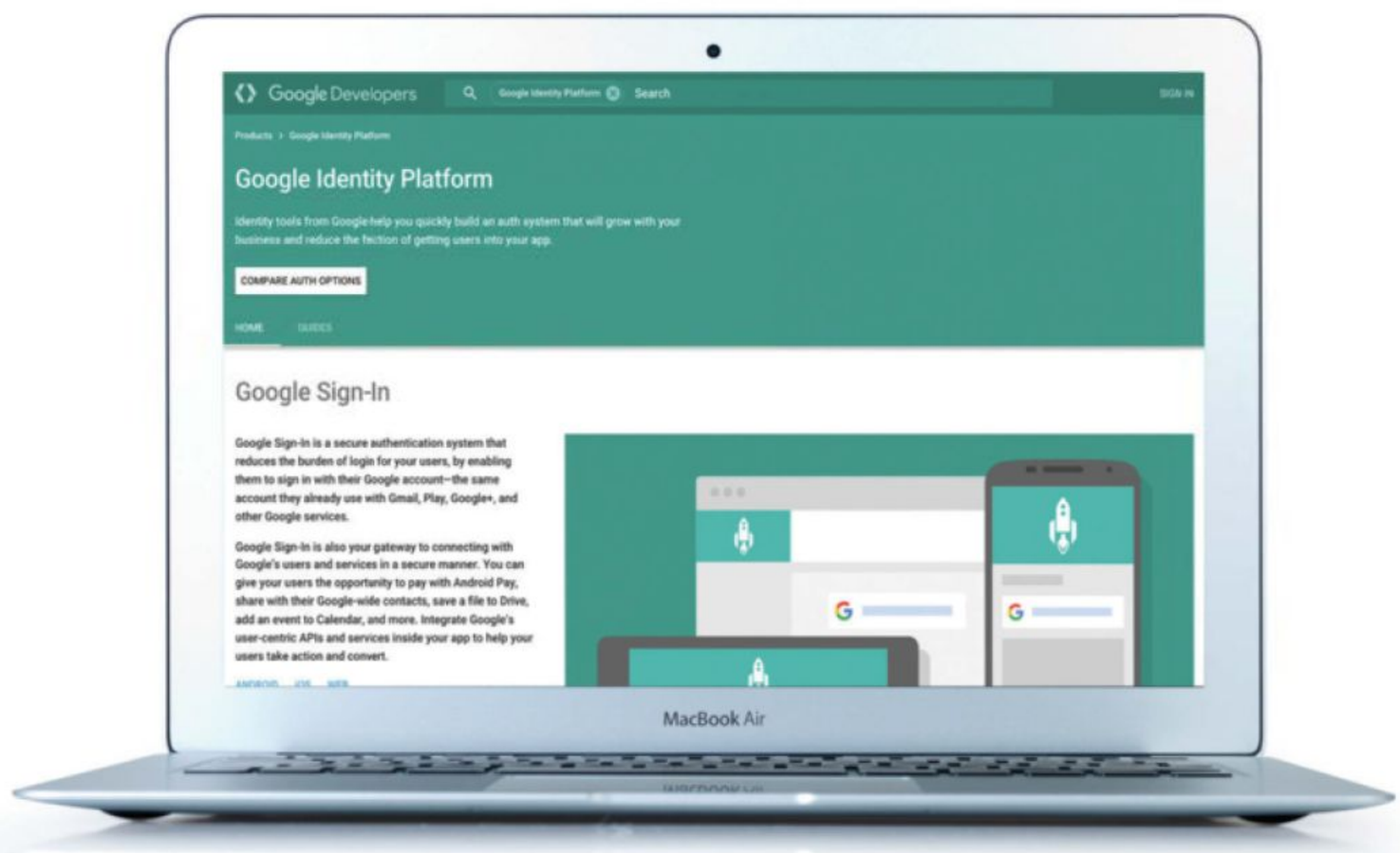
Software is an unique product in that it can be multiplied with minimal cost. Once the development has been footed, adding new installations tends to be an issue of purchasing extra hardware. Google has used this very approach to produce a variety of APIs, providing web designers access to building blocks used for the realisation of its own services. Some of them are, fortunately for us, made available for free: Google willingly foots the bill in order to tie developers closer to its own ecosystem.

In addition to the eventual introduction of monetisation (well-established APIs tend to stop being free when enough people use them), Google furthermore profits from the ability to get its APIs stress-tested by third parties without needing to pay them for their work.

Another, possibly more sinister, benefit though involves additional access to customer data. If a website generates graphs and other features using Google's services, then the search engine can track user movements and site retention times more efficiently. It would not surprise us if this data somehow found its way into search engine rankings. For example, a scammer with a well-placed but content-poor website could give himself away via the very short visit duration.

But let us throw caution to the wind for now though – in all fairness, the possibilities offered are far too great for us to simply pass them up. Look forward to effortlessly creating diagrams and graphs, handling signins automatically, renting databases on an as-needed basis over the next few pages in our list of top APIs. If you can name it, it's likely that Google already has you covered.

> "Google willingly foots the bill in order to tie developers closer to its own ecosystem"



Registering for accounts

API: Google Sign In

Cost: Free

URL: developers.google.com/identity/sign-in/

Making your users sign in with an account is a sure-fire way to lose them: creating a new account is an extraordinary level of hassle for many people. You'll find that many users will simply switch off or close the browser tab at the sight of such a page – they will want to get content there and then, without having to wade through the tiresome and overdone username and password boxes. But by using Google's sign-in system, we can easily mitigate the problem by permitting your users to log in with their already-available Google account.

Getting started with most Google APIs requires you to sign up for an account in the Google Developer's console (**console.developers.google.com**). Next, create a project and open the subsection APIs&auth and click the Credentials subtab. Then, create an Auth ID. If you created a new project, the Developer Console will now ask you to configure the Consent screen: the information entered here will be displayed to the users as your app authenticates. Then, you just have to create a client ID for a web application – don't forget to fill out the authorised URL fields with your testing and deployment URLs in order to prevent Google from blocking the app. Google will then kindly reward your efforts by generating a client ID and a client secret for you.

Don't abuse data

Misuse of Google-sourced data always gets media attention. Be sure that their legal team will set out to punish the culprit harshly.

In the next step, a test must be hacked up. Its structure is simple – it loads a script, and runs an onSignIn-Method when the login process is completed:

```
<html>
<head>
<script src="https://apis.google.com/js/
platform.js" async defer></script>
<meta name="google-signin-client_id"
content="YOUR_CLIENT_ID.apps.
googleusercontent.com">
</head>
<body>
<script type="text/
JavaScript">
function
onSignIn(googleUser) {
var profile = googleUser.
getBasicProfile();
console.log('ID: ' +
profile.getId()); // Do not
send to your backend!
console.log('Name: ' +
profile.getName());
console.log('Image URL: ' +
profile.getImageUrl());
console.log('Email: ' + profile.getEmail());
}
</script>
<div class="g-signin2" data-
onsuccess="onSignIn"></div>
</body>
</html>
```

The log-in button is generated via a predefined class, which takes its properties from the header. It will work only if the HTML file is served from one of the URLs specified – set up a local web server to test this file.

Get your games in store

API: Google Game Services **Cost:** Free **URL:** bit.ly/1iMdSvO

Keeping players loyal has become more and more difficult. As time has passed by, botch achievements and leaderboards established themselves as 'tools of the loyalty trade'.

Implementing both of these by hand is not complex: in today's fast-turning game market, the time needed should better be invested into more marketing. Thus, Google's offerings come at the right time: Game Services is available to native, Android, iOS and generic C++ apps.

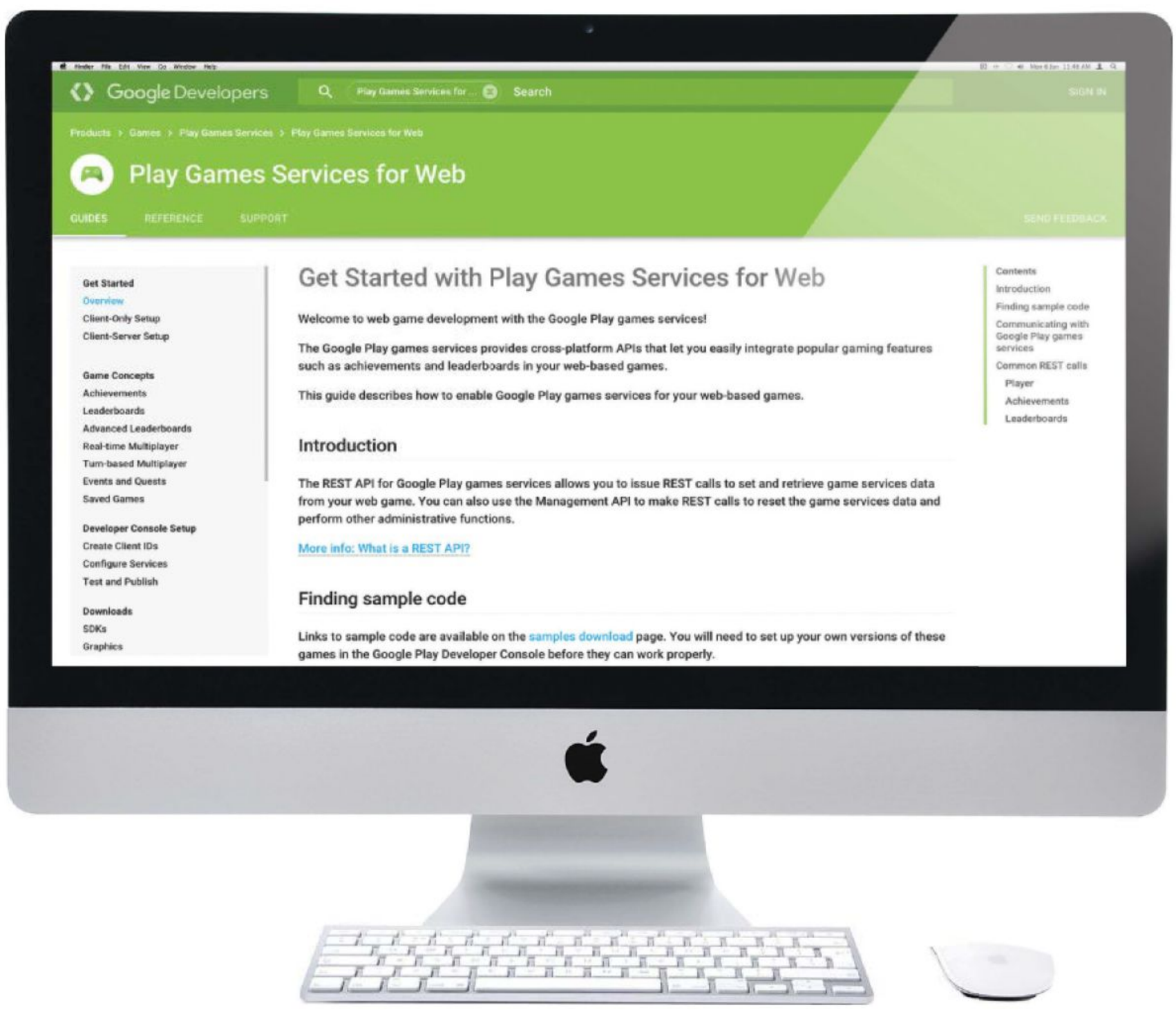
In all cases, the journey to a Game Services-enabled product begins in the Google Play Developer Console found at **play.google.com/apps/publish/signup/**. Developers who have not signed up for the Play Store yet will find themselves coughing up $25 – Google insists on account verification. In the next step, create a new game in the Game Services tab. Finally, create an Auth ID and proceed to setting up leaderboards and/or achievements as needed. The actual implementation begins with an Auth 2.0 login process according to the steps outlined in **bit.ly/1L8CiMo**. Once this is out of the way, registering success on an achievement can be accomplished by firing off a POST request based on the scheme (**bit.ly/1L8CjQn**).

By and large, leaderboards will behave in the same way: it will specify criteria and make sure that the player's competitive sense is activated. Google provides a list of interesting REST calls, such as for displaying a leaderboard or submitting a finished game score, at **bit.ly/1KBP7It**. Play Services aren't limited to serving 'static' content. The system can also be used to handle the communication between players: real-time synchronisation is offered only to native apps, while turn-based game management is offered to web and native apps. In essence, synchronising a game involves the frequent transmission of JSON objects with state information. A game's players are systematically grouped into matches by the system, and once a game has been established, turns are taken until one of the game instances reports a victory.

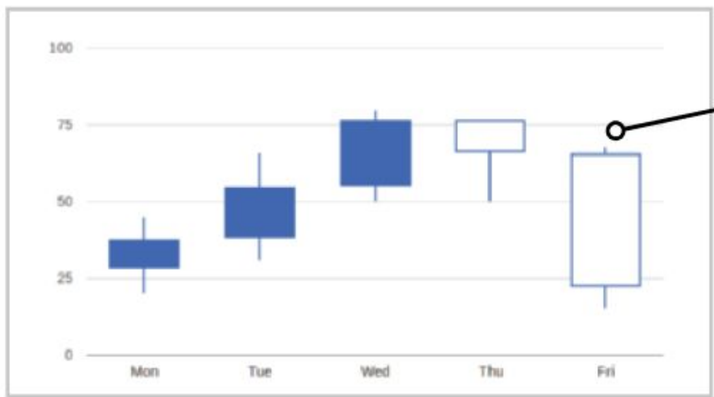
Professional advice

In-app-purchase optimisation is now a science of its own – there's no shame in getting professional help for that IAP game addiction.

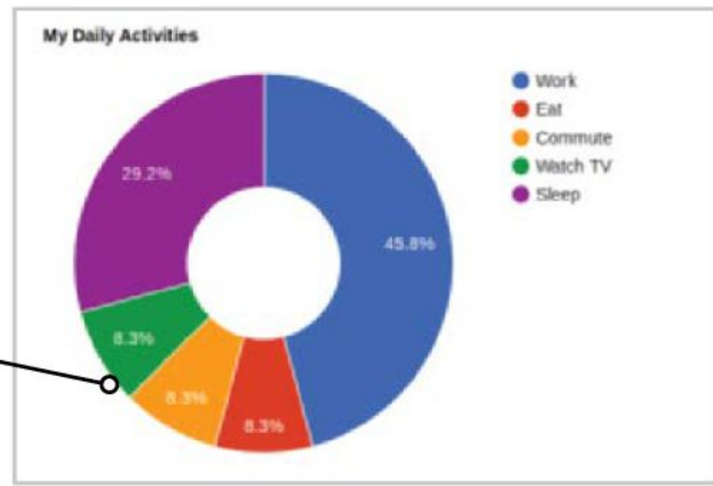
Cross-platform games have recently picked up a lot of steam in the gaming industry, which has been great for gaming fans who love to see their platform of choice take the lead over a passionately hated competitor. By default, Google Play Services will automatically mingle players independent of the operating system they are playing on – an almost ideal situation for everyone.



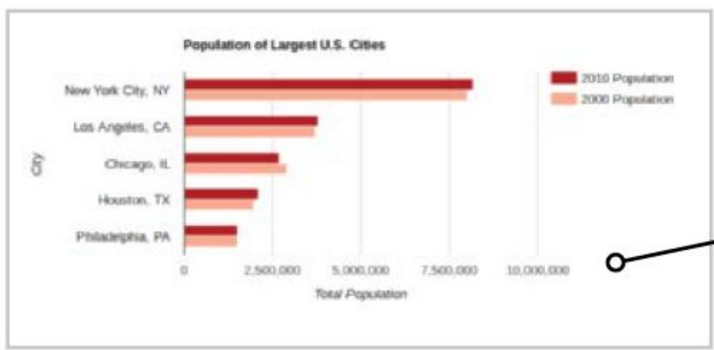
Fast custom charts



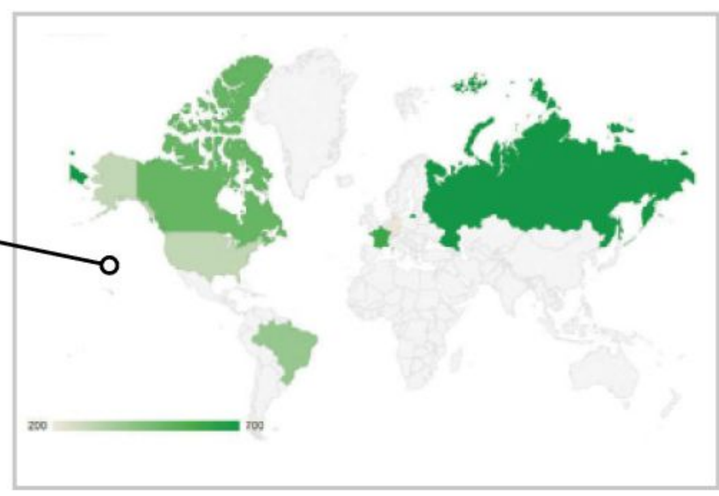
Candlestick charts are especially useful for technical analysis: they display open, high, low and close in one compact diagram



Doughnut charts are but a small modification of the pie chart: the hole in the middle saves ink when large diagrams are to be printed



Bar charts can be considered the classic chart. It's no surprise that they're still so widespread with its easy-to-read information and a key to distinguish data



Rendering regional diagrams is considered a black art. Google's library makes it really easy – it even provides the map outline for you

API: Diagrams for everybody

Cost: Free

URL: developers.google.com/chart

A chart says more than a thousand words. The widespread implementation of the <canvas> tag has permitted the creation of chart-rendering libraries which do their magic in the browser of the customer's workstation. Copy the file to your workstation and load it in a browser of choice in order to feast your eyes on the bar chart contained within:

```
<html>
<head>
<script type="text/JavaScript" src="https://
www.google.com/jsapi"></script>
<script type="text/JavaScript">
google.load('visualization', '1.0',
{'packages':['corechart']});
google.setOnLoadCallback(drawChart);
function drawChart() {
var data = new google.visualization.
DataTable();
data.addColumn('string', 'Good');
data.addColumn('number', 'Amount');
data.addRows([
['Silver', 300],
['Gold', 100],
['Platinum', 12],
]);
var options = {'title':'Wealth allocation
plan',
'width':640,
'height':480};
var chart = new google.visualization.
PieChart(document.getElementById('chart_
div'));
chart.draw(data, options);
}
</script>
</head>
<body>
</body>
</html>
```

Google insists that the packages constituting the Charts API must be downloaded freshly on each page-load via the Google JS loader. In the next step, a data object is created. Google provides a ready-made DataTable class with convenience functions making entering data easy.

The actual rendering process is then fired up by invoking the draw method of the chart object. Charts implements about two dozen different chart types – they differ in data format, invocation and load request. Start out by visiting **bit.ly/1VcR8AZ**: Google provides an instructive example for most diagrams. Be aware that the usage of Google Charts requires an active internet connection on your client's workstations.

Maps, fonts and design

API: Material Design Lite

Cost: Free

URL: getmdl.io

JavaScript GUI stacks tend to be hit and miss. Google's Material Design Lite is a lightweight implementation of Android's Material Design theme, based on HTML5, CSS and a bit of JavaScript. It comes with a set of widgets including templates, components and themes. These are easy to integrate, customise and download once you understand the structure used. Since Material Design Lite can be self-hosted, individuals paranoid about hosting will not be dissuaded from using it.

API: Google Maps

Cost: Depends on usage

URL: developers.google.com/maps/web/

Rendering maps by hand is not for the faint-hearted. If you don't have a solid understanding of data management and matrix math though, it might be best to read up on this one carefully before giving it a shot. Google permits you to access its mapping system's features and functionality via a variety of APIs. Initially usage is free however, once you exceed 25,000 map loads per day for 90 days, the use of some APIs will require payment.

API: Fonts API

Cost: Depends on usage

URL: developers.google.com/fonts/

Font wars extended to the web the moment browsers started to be able to parse custom fonts. Google's font repository provides developers with a large selection of free-to-use fonts. Deploying them is as easy as embedding the requested CSS file:

```
<link rel="stylesheet" type="text/
css" href="http://fonts.googleapis.
com/css?family=Tangerine">
<style>
body {
font-family: 'Tangerine', serif;
font-size: 48px;
}
</style>
```

High-performance data

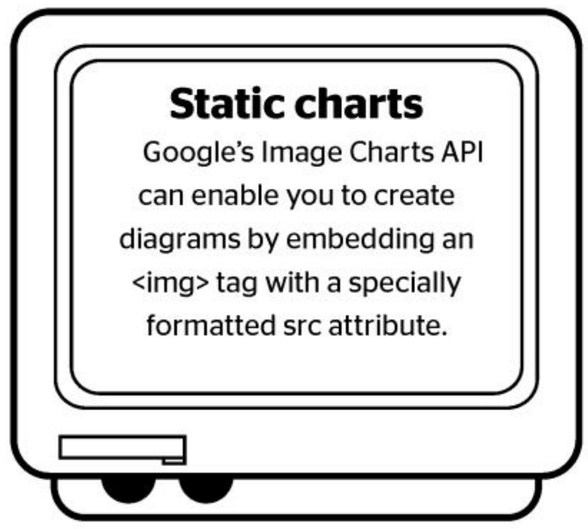
API: Google Cloud SQL **Cost:** Billed **URL:** cloud.google.com/sql/

As systems become larger, databases transform themselves into bottlenecks. Running large, high-performance server systems is a science of its own: if you don't have an expert on databases in-house, outsourcing it is more sensible.

Google is without a shadow of a doubt the undefeated master of big data: managing its search index is an exercise in high-performance databasing. Getting started with a Cloud SQL database is incredibly easy: log into the developer's console and then create a new Cloud SQL instance for yourself in the project. In the next step, you need to set up which IP ranges are permitted to interact with the system.

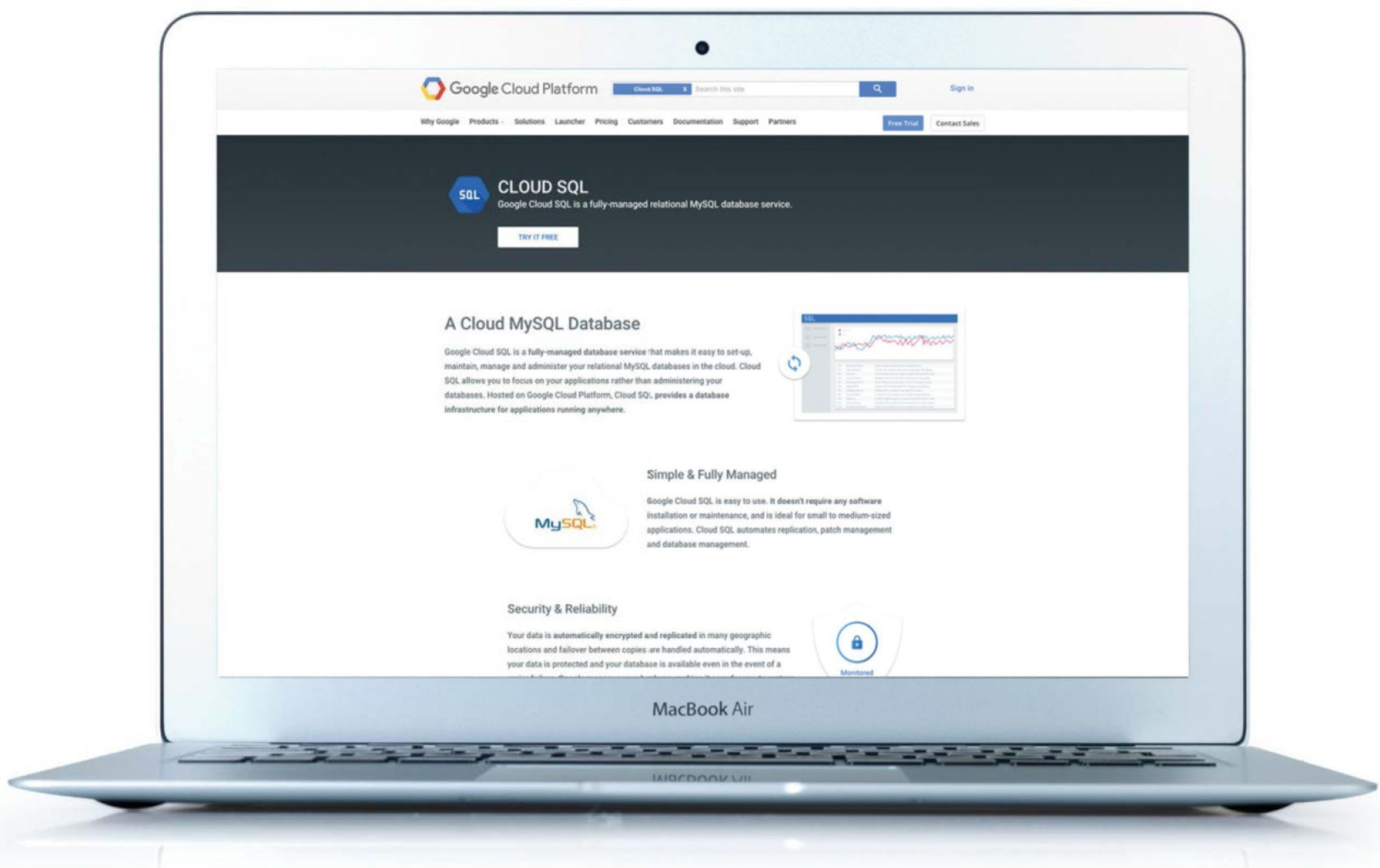
From a user's point of view, Google Cloud SQL-hosted databases show next to no real differences to a normal database that they would otherwise be used to: you can connect to them like you usually can to any other SQL instance. Administrators will also be very happy to find out that many commonly used DB tasks can be performed from the graphical backend in the Google Developer Console. Setting up replication to a local database host currently no longer requires the use of arcane SQL commands – even though SQL wizards can, of course, work their magic on the database via the MySQL client.

Google provides two different billing models, which also simultaneously affect availability and uptime. An instance billed on a per-use basis is shut down when clients leave it alone for over 15 minutes. Fixed packets, on the other hand, tend to be active during their entire lifetime, shutting down only after more than ten hours of user inactivity.



Static charts

Google's Image Charts API can enable you to create diagrams by embedding an <img> tag with a specially formatted src attribute.



Get the best out of YouTube

API: Power to YouTube! **Cost:** Free **URL:** developers.google.com/youtube/

Streaming video is a sure-fire way to miff your hosting company. Google provides a variety of APIs which permit you to use YouTube as a storage and management system. Embedding a player into a website can be accomplished with a bit of JavaScript:

```
var player;
function onYouTubeIframeAPIReady() {
player = new YT.Player('player', {
height: '390',
width: '640',
videoId: 'M7lc1UVf-VE',
events: {
'onReady': onPlayerReady,
'onStateChange': onPlayerStateChange
}
});
}
```

This is a part of the features and you can also use this API to permit users to sign in to YouTube. Live streaming events can also be set up via YouTube in a relatively, more complex process.

Pay, date & find APIs

API: Calender **Cost:** Free **URL:** bit.ly/1MJbfaE

Google Calendar initially got little attention, but stepped into the limelight as Android smartphones and their syncing capability became widespread. The JavaScript API lets applications interact with the customer's calendar data. Events can be added, removed and edited – a stocks app could use this to add dividend information to a smartphone. Google implements the Calendar API as a REST service. Interactions with the server take place by sending predefined JSON strings: JavaScript developers must make do without the abstraction classes found in SDKs for other languages.

API: Android Pay **Cost:** N/A **URL:** bit.ly/1L6kz8b

The Android Pay service takes Google Wallet to the real world. It is intended to streamline payment processing both in apps and in physical high-street stores. As users get accustomed to the way the system works, conversion rates should increase significantly. Sadly, all rumoured release dates for Android Pay have, so far, slipped – one can not be sure when the service will finally launch.

API: Geocoding made easy **Cost:** Free with usage limits **URL:** bit.ly/1FbcYDa

Real-world addresses are sensible to us mere humans: geodesic applications tend to work better with coordinates. Google's Geocoding API is dedicated to this conversion process: users can enter a street name, and then your app will generate a pair of latitude and longitude data in return. Be aware though, as Google always limits apps to ten requests per second.

> Interactions with the server take place by sending predefined JSON strings: you must make do without abstraction classes found in SDKs for other languages

Manage & Drive APIs

API: Task manager

Cost: Free

URL: bit.ly/1KA6uJE

Google provides an advanced to-do list management system for each and every owner of a Google account. Third-party applications can access this to-do list in order to add, remove and manage tasks at hand. Developers should take note of this API due to Google not offering Android and/or iOS clients of their own.

API: Accessing Drive

Cost: Depends on usage

URL: bit.ly/1YBP6yH

Google Drive is among the widest-spread cloud storage services, which can be accessed via a dedicated API. Developers used to normal file systems need to rethink their approach; Drive uses a file-ID-based storage scheme. Both files and folders are represented by a globally unique ID, which must be traversed to find eventual children.

API: Google Analytics

Cost: Depends on usage

URL: google.com/analytics/

Integrating analytics into a website and/or an app is very useful. You can also prepare yourself for a never-ending flow of all kinds of useful data about your customers and their behaviour in the web property when under testing. Google Analytics also collects system data which can be used to improve targeting, resource allocation and feature deployment.

API: Google Picker

Cost: Free

URL: developers.google.com/picker

The above-mentioned ID-based file system makes entered paths all but worthless. Fortunately, this API provides your users with a CommonDialog-like experience. You can tailor the displayed content with special properties: for example, a photo management application could drop the user off in his webcam picture folder.

DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/bks-894

Get free web hosting with GitHub Pages

GitHub is commonly used to host portfolios of code. Showing the code in action increases its effect ten-fold

Traditionally, developers solved the problem of hosting by adding a link to content hosted at an external website. Few coders know that each and every GitHub repository comes with a dedicated area for files which are served via a normal web server.

In fact, GitHub supports three different repository types. A Project site acts as a deployment vector for the above-mentioned cide examples. User and/or organisation sites differ from project sites in that they are instead intended to represent a firm or an individual's web presence.

From a technical point of view, taking GitHub's offering is interesting as long as websites remain static. GitHub does not provide databases or other "active" content – if you plan to host a forum or a WordPress-managed blog, this is not the host of choice.

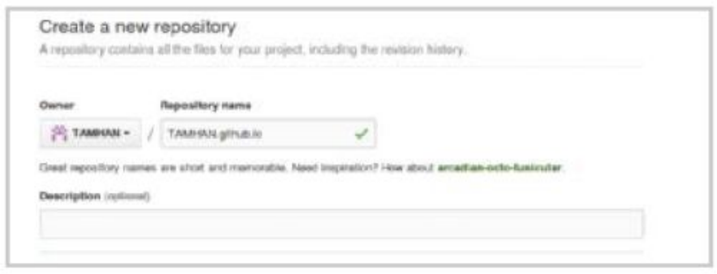
Please be aware that GitHub Pages is not a replacement for a classic web host. It is a hosting option for placing GitHub-related content, which enables its users to test code.

1. Sign up!

Using GitHub Pages requires a GitHub account. If you have not signed up yet, hit **GitHub.com** and click the big green Sign Up button at the top right. For GitHub Pages, purchasing a premium account is not necessary as there are no privacy options provided either way (see steps 20 and 21 for more information).

2. Create a page repository

Personal pages must be created in a repository where the name must match your username in a 1:1 fashion along the rest of the string shown in the window. If you have not created a repository before, then set one up according to the parameters in the image that follows.



3. Initialise the local copy

The following steps are based on an Ubuntu workstation. Open a Terminal window, and create a new and empty folder. In the next step, enter the following commands to connect the folder to the GitHub server. During the process, a group of metadata files are spawned.

```
echo "# TAMHAN.github.io" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/
TAMHAN/TAMHAN.github.io.git
git push -u origin master
```

4. Position your web content

Your Terminal window's working directory now contains an empty file called README.md. Put a second file called index.html in the same folder: the location of README.md shall be the "root" of the web host, while subfolders below readme.md will go to a subfolder of the newly created website.

5. Upload the data

From the point of view of the git client, a GitHub Pages-hosted website is little more than an exposed Git repository. Due to that, the newly created page and any contained images must be pushed to the server before they can become visible to the World Wide Web.

```
git add --all
git commit -m "Initial commit"
git push -u origin master
```

6. Check if working

After this upload, our website is ready for testing. Open the URL **<username>.github.io** in a browser of your choice. Be aware that all GitHub user names are lower case by force: even if you log in as TAMHAN, the correct URL will still be **tamhan.github.io**.

7. Create a 404 message

All GitHub repositories can be equipped with a "default 404 page" which is displayed to users who attempt to access invalid or non-existing parts of the GitHub repository. In the case of our example, the file 404.html contains a friendly error message.

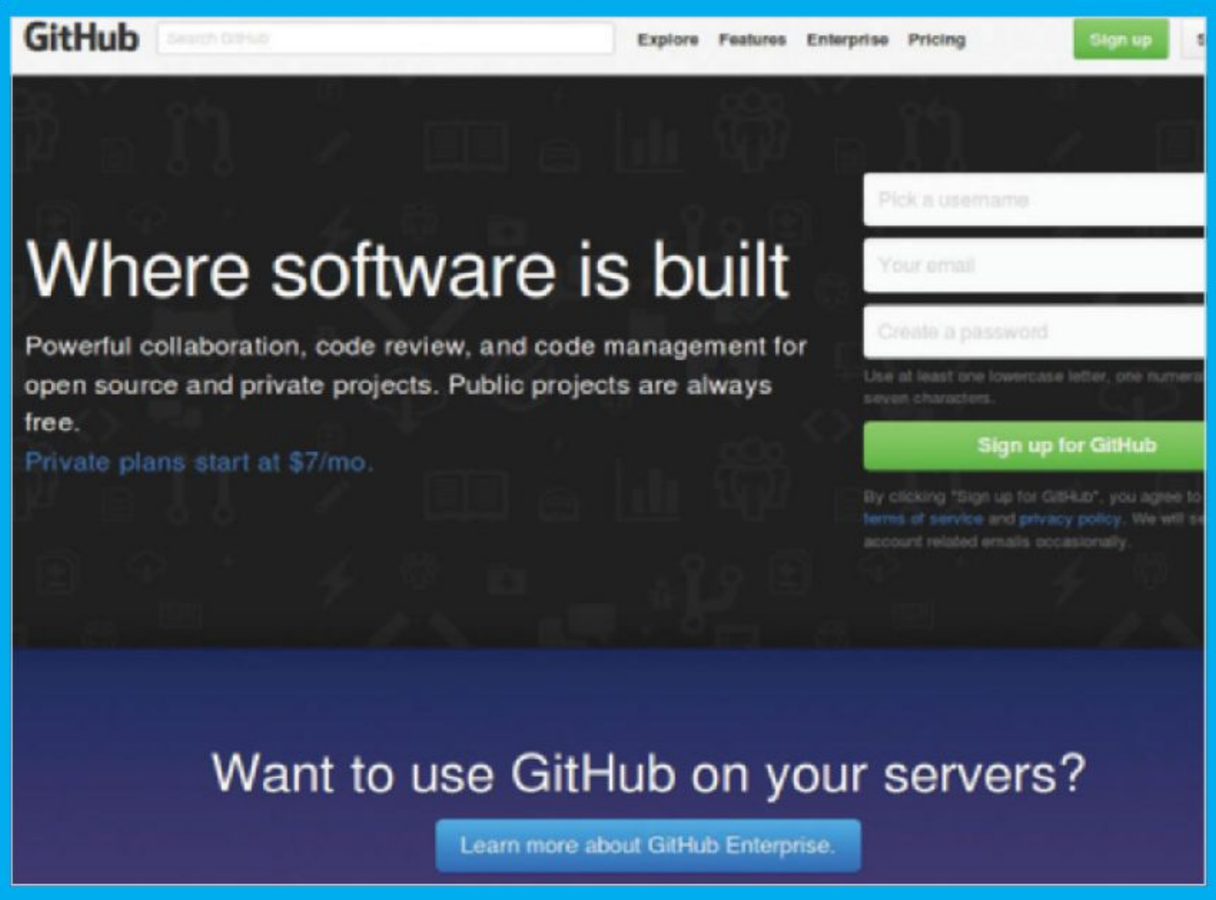
```
<html>
<body>
<h1>An error has occurred</h1>
</body>
</html>
```

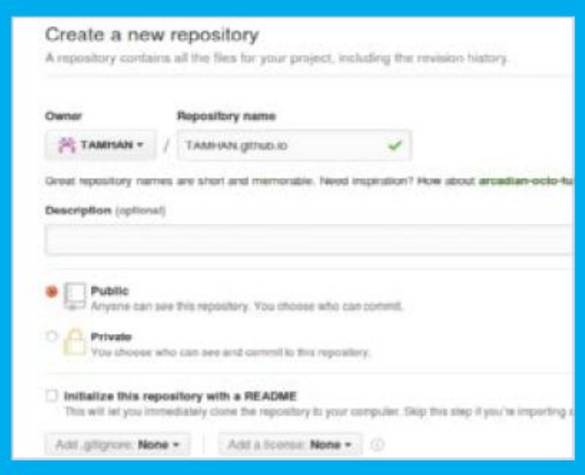
8. Deploy, the second

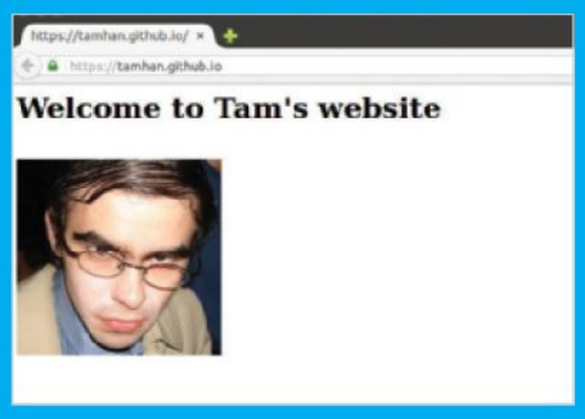
As GitHub Pages are hosted in a Git repository, committing changes should be accompanied with a descriptive message such as the one shown in the code

Attention: Unix!

GitHub Pages is hosted on a Unix operating system. This means that paths and file names are case sensitive – a nasty trap for people switching from Windows Server.







<Top left>
• GitHub Pages requires a specific repository name based on the scheme <NAME>.github.io – picking any other name leads to issues

<Bottom left>
• Our GitHub Pages-hosted website is being served as a normal HTTP page when its "access URL" is entered into a browser of choice

<Left>
• Using GitHub Pages will require you to sign up at GitHub's site – an act which may not always appeal to the taste of more libertarian individuals

featured in this step. Do not wonder if the client will ask for username and password from time to time as a security precaution.

```
git add --all
git commit -m "Add 404 page"
git push -u origin master
```

9. It's really just GIT

As we have stated earlier on, a GitHub Page is really little more than a publicly exposed Git repository. In the case of our website, you can just open up **github.com/TAMHAN/TAMHAN.github.io** in order to see the commit history along with a stub file left over from an editor crash.

10. Let's start a project

While personal pages on GitHub might be nice, the real value lies in the creation of websites for projects. Let us set out by creating a new project containing a bit of C code – simply upload it in order to make GitHub show the project contents view when the repository is opened in the WebView.

11. Automagic generation

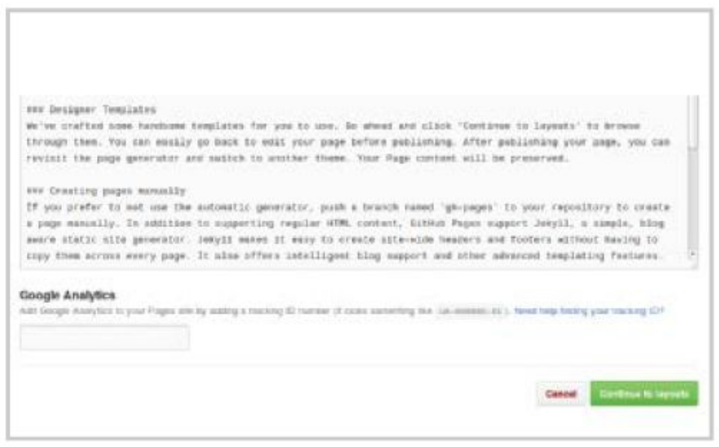
In the next step, click the wrench icon on the right-hand side of the screen. Next, move down towards the "Automatic page generator" segment and click the "Launch automatic page generator" button in order to start the modification process.

12. Do the Markdown!

In the next step, GitHub will open an editor where the contents of the project's homepage can be edited. The product uses a Markdown-like syntax which is well-known from a variety of CMSs and a few Mac OS-based editors.

13. Add Google Analytics

Tracking the successes of a GitHub repository is a process that is best accomplished via the use of Google Analytics. In the first step, you should work on following the comprehensive instructions that have been outlined on the Analytics documentation at **support.google.com/analytics/answer/1032385?hl=en** in order to obtain the Google ID. Then, once you have the Google ID to hand, you can now enter it into the box at the bottom of the Markdown editor.



14. Choose your layout

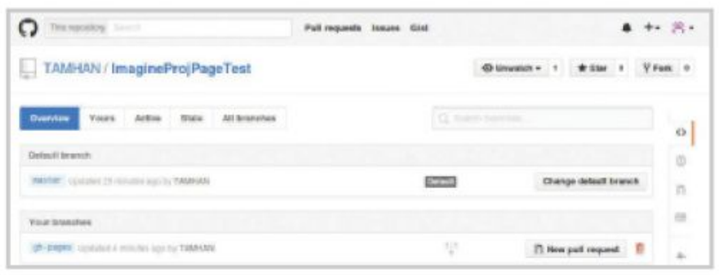
Presenting a bland website is no fun. Thus, click Continue to Layouts in order to open a preview alongside with a ribbon permitting you to select the style. GitHub provides a total of twelve different themes; you can switch between the two groups with the arrow symbols pointing to the left and right.

15. Go live

Finally, click the Publish button in order to go live with the changes. GitHub will confirm the successful generation via a small banner at the top of the screen. You can go ahead and open **tamhan.github.io/ImagineProjPageTest** in order to take a look at the results of our labour.

16. Do the branching

The files intended for the GitHub page of a project are not stored in the main branch. Instead, GitHub's generator automatically generates a branch called gh-pages which acts as the "host" containing the files intended for the project's homepage.



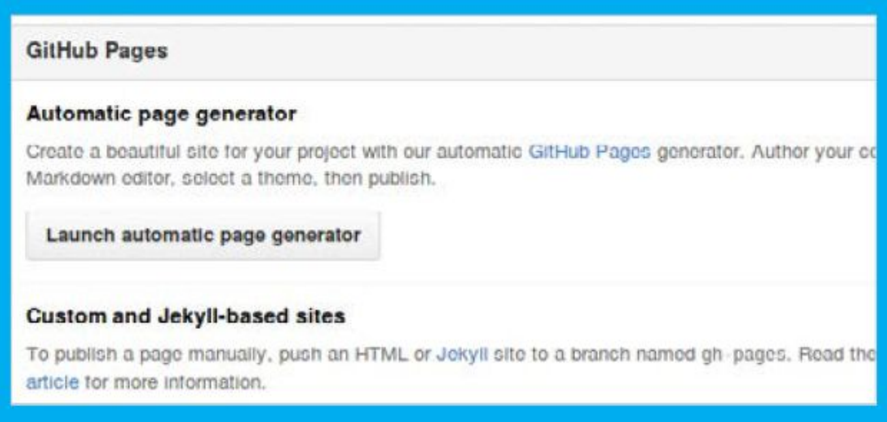
Analytics caveat

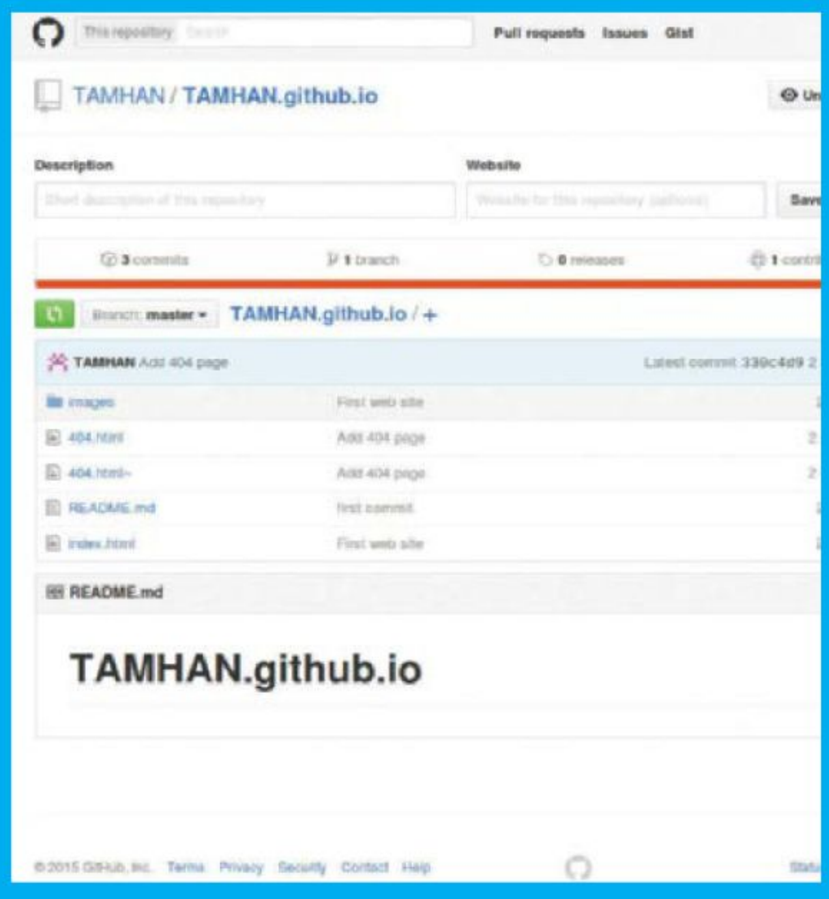
Deploying Google Analytics is complex from a legal point of view: in many countries, users need to be notified that their data is collected and/or forwarded to Google's servers.

<Top left>
• Even though the Editor embedded into GitHub Pages might look a whole lot like the familiar WordPress, it works with a completely different markup language

<Bottom left>
• GitHub's automatic page generator is well-hidden: you can find it in the settings dialog of a project

<Right>
• Forgetting to remove temporary files before starting the commit process leads to weird and – sometimes – embarrassing results





How GitHub Pages sites use custom domains

Type of GitHub Pages site	Host location on GitHub	How the page is redirected	Example custom domain
User Pages site	`username.github.io`	Automatically redirected to the custom domain that has been set for it	`user.example.com`
Organization Pages site	`orgname.github.io`	Automatically redirected to the custom domain that has been set for it	`org.example.com`
Project Pages site owned by a **user account**	`username.github.io/projectname`	Automatically redirected to a subdirectory of a User Pages site custom domain (`user.example.com/projectname`)	`project.example.com`
Project Pages site owned by an **organization**	`orgname.github.io/projectname`	Automatically redirected to a subdirectory of an Organization Pages site custom domain (`org.example.com/projectname`)	`project.example.com`

Warning: If the URL for your Pages site contains a username or organization name that begins or ends in a dash, or contains consecutive dashes, then people browsing with Linux will receive a server error when they visit the site. To fix this, change your GitHub username to remove non-alphanumeric characters. For instructions on how to do this, see "Changing your GitHub username."

To troubleshoot problems with your custom domain, see "My custom domain isn't working."

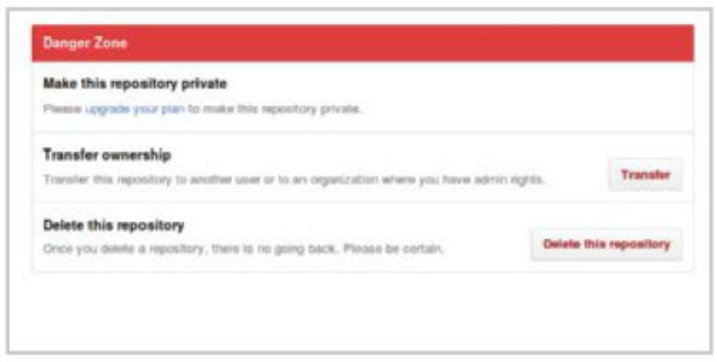
Do the domain!

One way to make GitHub Pages a little more professional is through the use of subdomains. You can add a CNAME record at your web or domain provider in order to redirect traffic to GitHub Pages – as the system is based on DNS names, your site will continue to benefit from GitHub's optimisation and CDN service. For example, **orgname.github.io/projectname** could be redirected to **project.tamoggemon.com**.

If your domain provider does not offer the CNAME option, you can alternatively use an apex domain which forwards to the IP adress of GitHub. GitHub officially advises against the use of apex domains as they do not permit the use of a CDN – further information on the topic can be found at **bit.ly/1NpCaZX**.

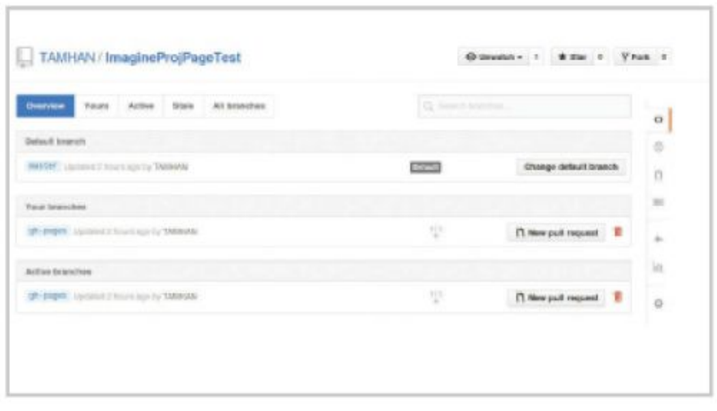
17. Delete a personal page

Getting rid of a personal page is not difficult. Open the repository containing its files, and switch into the Settings pane. Then, move into the Danger Zone tab and click the "Delete this repository" button.



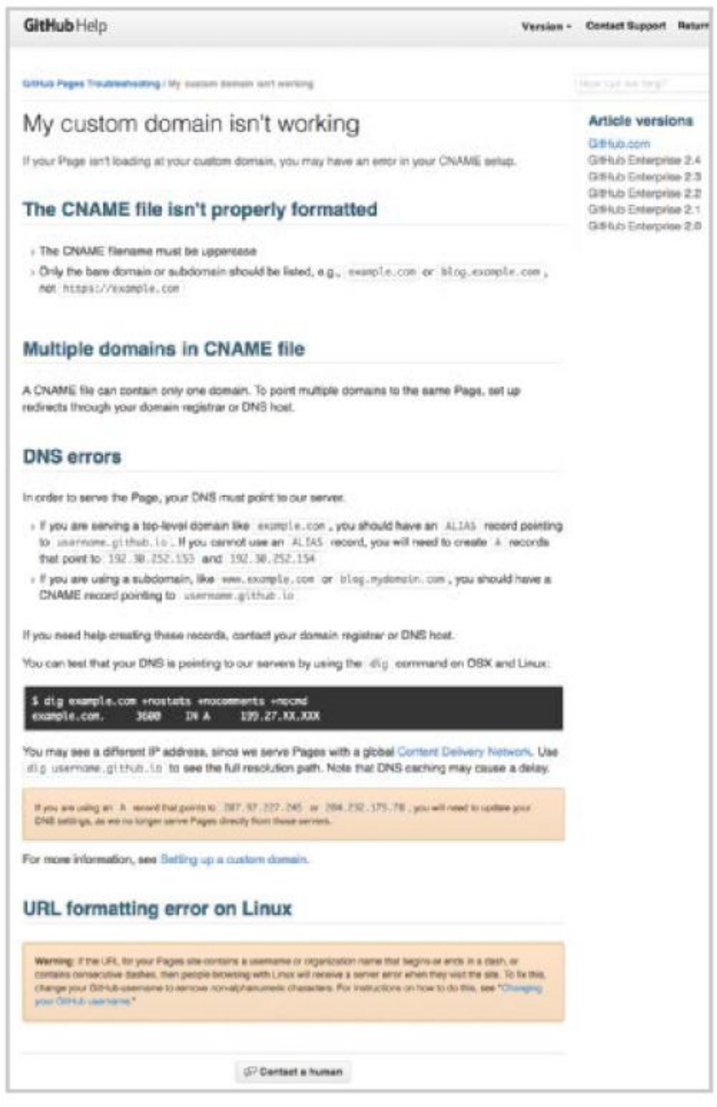
18. Delete a project page

Should you ever decide to eliminate your project page, simply remove the gh-pages branch. This can be accomplished by opening the branches tab of the repository in the web interface. Next, click the little garbage can next to the gh-pages branch in order to banish it forever.



19. Troubleshooting

The automatic page generator can be a bit troublesome at times. Fortunately, the GitHub team provides a selection of frequent issues which can be accessed by opening the URL **help.github.com/categories/github-pages-troubleshooting/**.

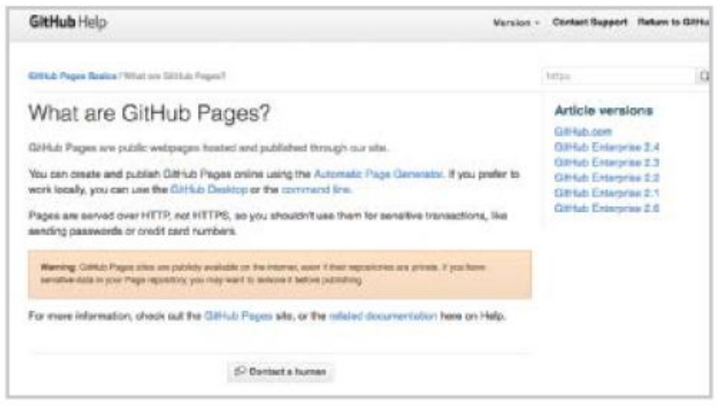


20. The NSA's best friend

While GitHub repositories can be declared private, this flag does not apply to any Pages content found within. Be aware that any content uploaded to the Pages system is made publicly available – if someone finds it, consider yourself officially out of luck.

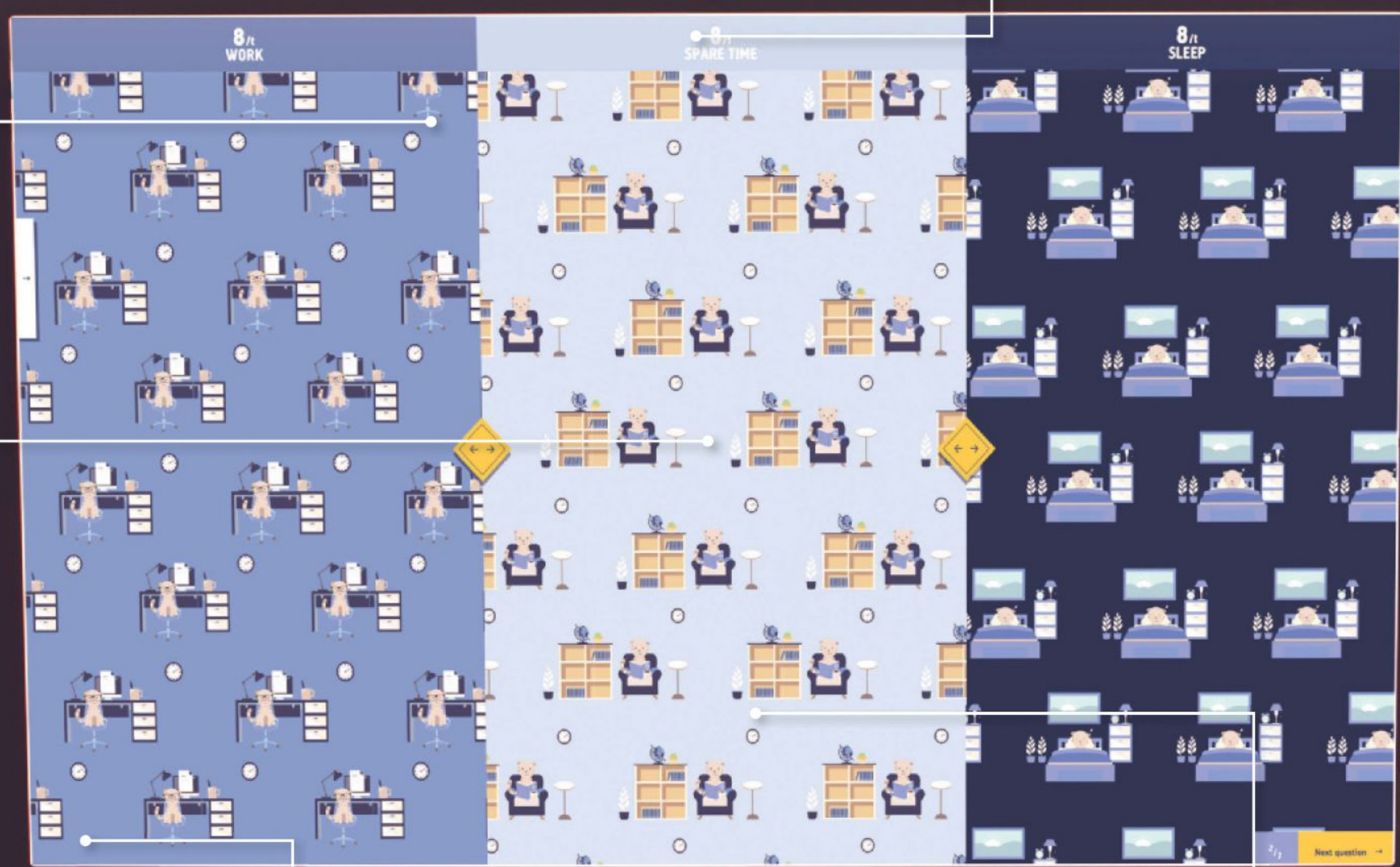
21. HTTPs: missing in action

Another significant limitation of GitHub Pages involves the fact that HTTPs is not supported. Even though secure HTTP might be easy to crack for dedicated attackers, it nevertheless provides an urgently-needed extra layer of security that is absent here.



Make a resizable sliding panel for your content

As seen on **sparetime.arkivert.no/en**



jQuery UI-powered
Our drag and drop engine comes from the jQuery UI framework – it's well known for its almost-universal compatibility.

Minimal width
Our example does not enforce a minimum width for convenience. But adding this feature does not require much code.

Touch suggestion
Tilting the movement gems to the side provides the users with a visual cue to motivate them to move the widgets around.

Static backgrounds
Arkivert uses cute illustrations but you can replace the backgrounds with any content you consider to be interesting.

Another drawer
Adding another drawer is not particularily complicated: simply duplicate the s and add a bit of resizing logic.

DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/bks-894

Research has repeatedly proven that humans have their fair share of problems interpreting numbers. If data is spruced up visually, keeping tabs on information is much easier.

This instalment of our never-ending series on web techniques was inspired by a website (**sparetime.arkivert.no/en**) which makes use of draggable columns to represent an amount of information visually. Users arranged the width of multiple columns on the screen in order to map their behavioural patterns to the data model of the application.

This, however, is but one application for movable columns. As websites strive to replace classic desktop applications, multicolumn views gain importance: for example, an email client could display three panes containing the folder's content, an email's content and a folder structure next to one another.

Providing draggable windows is a sure-fire way to endear such an application to its users: as of this writing, most web-based programs lack capabilities for the flexible positioning of their contents.

Fortunately, getting started is really easy and you can find all the code for this Web Workshop on FileSilo

<comment>
What our experts think of the site

Putting a value on time

"Figuring out how much one hour of your work day is worth is relatively easy. Performing a similar computation for spare time tends to be more involved. Arkivert's Sparetime calculator attempts to value your free time by asking a group of questions, which are then algorithmically parsed into a surprisingly accurate estimate." **Tam Hanna**

Technique

1. Div tags

Let's get started by erecting the scaffolding of our website. It consists of three tags, which are arranged next to one another via the width and padding-left properties. Describing the spaces with relative values ensures that the page remains fullscreen even if the user changes his window size.

2. Add a draggable handle

Offering resizable panels is fun only if the user can actually change their size. This is best accomplished via two handles displayed in the middle of the borders. For now, adding simple boxes via a tag and some CSS magic shall suffice.

3. Add draggability

JQuery UI contains a robust drag-and-drop capability which works better than the one found in HTML5. Our example loads jQuery UI from the official CDN, and it adds event handlers which dumps the supplied information into the browser's debugger console.

```
<script type="text/JavaScript">
$( init );
function init() {
$('#box1').draggable({
start:handleDragStart,
drag:handleDrag});
$('#box2').draggable({
start:handleDragStart2,
drag:handleDrag2});
}
function handleDragStart( event, ui ) {
console.log( "Drag starts!" + event);
}
function handleDrag( event, ui ) {
console.log( "Drag !" + event);
}
function handleDragStart2( event, ui ) {
console.log( "Drag starts!" + event);
}
function handleDrag2( event, ui ) {
console.log( "Drag !" + event);
}
</script>
```

4. Do some maths

The next step involves the dynamic recalculation of the column widths as the position of the element is modified. This is relatively easy: the element's initial location is stored as onDragStart fires. After that, the CSS properties are recalculated dynamically.

5. Provide a helper object

As of this writing, the user can move the handles up and down. This is unsatisfactory, but can be solved by providing a helper object, which gets dragged across the screen in place of the original object.

6. Tilt the handles

Now transform the handles to an oblong shape in order to give our users a visual cue that they can use them to drag around. This is easily accomplished via a transform order, which gets added to the stylesheet of the boxes.

```
<style type="text/css">
div span.myBox{
position: absolute;
top: 50%;
right:-25px;
width: 50px;
height: 50px;
background-color: lavender;
border: solid 1px silver;
transform: rotate(45deg);
}
```

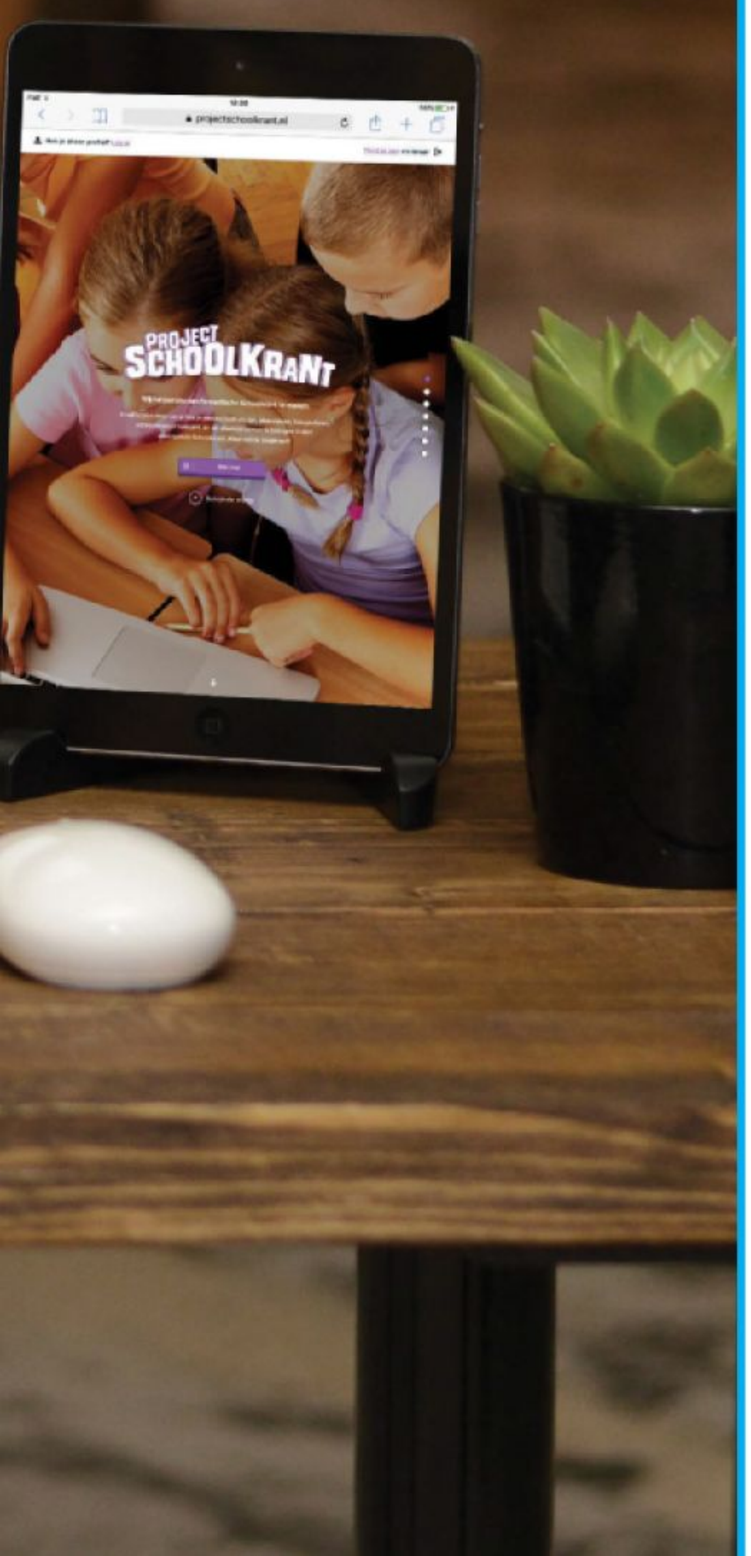
EXPERT INSIGHT
Not for mobile
Even though our slider looks awesome on a desktop, be aware that it is not particularily well suited to the needs of mobile phone users. Most smartphones are used in portrait mode, where horizontal space is at a premium. Drag and drop also requires a level of accuracy difficult to achieve on touchscreens.

Inspiration

Project Schoolkrant

It's top marks for this charming site encouraging Dutch children into newspaper journalism. Test your skills by copying their animations

DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/bks-894



<Left>
• A large header background image immediately speaks to kids and the virtual classroom context works well for the Project Schoolkrant experience

Designers J. Walter Thompson (JWT) Amsterdam, Superhero Cheesecake
Site projectschoolkrant.nl

This rich educational site produced by JWT Amsterdam and commissioned by De Persgroep Publishing's daily Algemeen Dagblad, a major national Dutch newspaper, acts as a curriculum tool for teachers and pupils alike. Built around five colourful lesson sections spanning news gathering, interview, photography, writing and design skills, the content culminates in compiling a school newspaper. Video tutorials from virtual tutors and a collection of animated characters combine beautifully to form an engaging experience for eager young eyes and minds.

A three-pillar approach marries colourful environments, playful learning assets and game-like features throughout. Completing tests unlocks badges as rewards to mark progress so visitors can return, log in and continue at leisure. The keenest kids can even submit their efforts in the hope of winning valuable publishing work experience in the real world. "We wanted to make an entertaining and easy-to-use platform using appealing colourful illustrations and animations when building the lessons and quizzes," explains David Navarro, head of design & digital creative director at J. Walter Thompson. "The best way to bring that universe of journalism to life was to use Flash to animate the various characters and use Adobe tools and the CreateJS suite to convert these animations to HTML5.

Digital production studio Superhero Cheesecake built the platform in vanilla JavaScript on a Backbone base and [used] the GSAP animation framework in creative ways in order to make it work properly across platforms and devices." The result is an engaging and successful website that has an appeal to a young audience as well as offering a truly useful experience.

> "We wanted to make an entertaining and easy-to-use platform"

Development

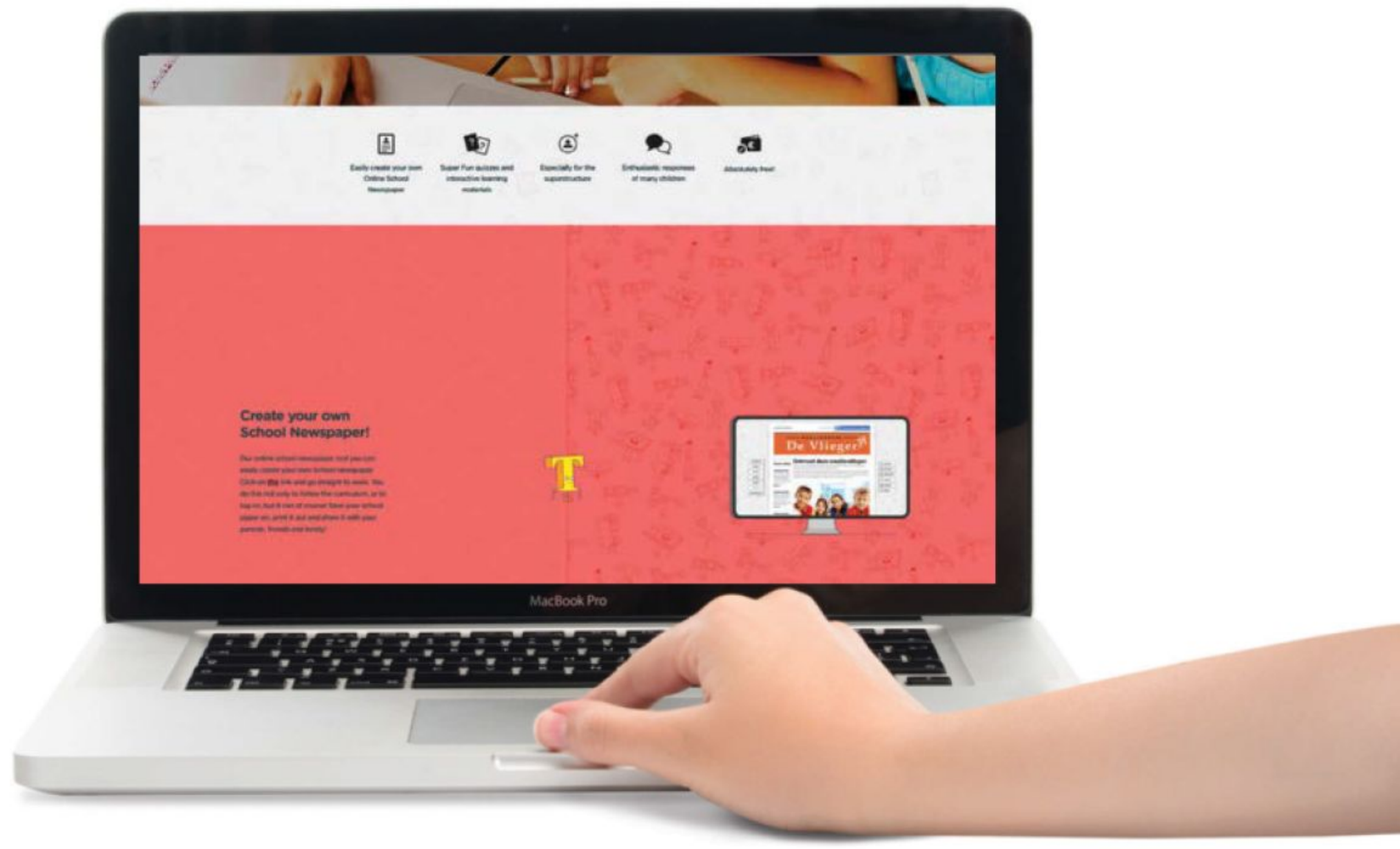
abcABC
1234567890

<Above>
• Besides Times Roman listed as a system font, Roboto Regular in normal weight is listed amongst the site styles

ABCABC
1234567890

<Above>
• Monstro Solid by PintassilgoPrints is the uppercase font used across headings in main tabbed sections

#9C6534 #89579F #0A2D32 #898989



<Top left>
• The site content revolves around 24 lessons, passing on core journalistic skills for creating a school newspaper to children

<Top right>
• Each section has its own bright, recognisable colour scheme to provide a modular feel that remains stimulating to kids

<Bottom left>
• Cute illustrations and subtle animations are peppered throughout, while navigation keeps access to specific lessons quick and intuitive

<Bottom right>
• A cast of virtual tutors host a series of embedded videos. They also appear as their own animated avatars to guide and inform

Add wiggling text headings to your pages with GSAP

Jordi Romkema, managing partner at Superhero Cheesecake, reveals the animation technique behind the site's wiggly quiz-time headings

1. Download and attach GSAP

Achieving the same effects as in the example on the previous pages isn't too hard. For this technique we'll show how you can re-create the 'wiggly' text headings found within Project Schoolkrant whenever a quiz begins. To get started, download the latest version of GSAP (1.18.0 or higher) from **greensock.com/gsap**. Unzip the minified JS file and place within your chosen project path. Start with a clean HTML document in the site root and reference the JS file inside the <head> tag or link to the CDN instead as shown:

```
<html>
<head>
<!-- Download GSAP link to page as  follows
-->
<script src="js/vendor/TweenMax.min.js"></
script>
<!-- Or via the latest CDN link as follows
-->
<script src="https://cdnjs.cloudflare.com/
ajax/libs/gsap/1.18.0/TweenMax.min.js"></
script>
</head>
</body></body>
</html>
```

2. Add your heading text

So in the very simplest terms, what our page needs is a heading to apply an animation to. To create an example heading, what you will need to do is add a <h1> tag to the <body> tag in your HTML document. You can of course use <h2> to <h6> headings as desired just as long as you specify a class name (and stick to it), such as 'title', so that we can reference it within our JavaScript as well as also apply additional CSS styling within the <head> tag.

```
<body>
<h1 class="title">Quiztime!</h1>
</body>
```

3. Create a script tag

Before you can get started with creating the desired wiggle function for the heading, you will first need to create a <script> tag for our JavaScript and add it to the bottom of the <body> tag in your HTML document. Start by creating a variable within the <script> tag that we will call 'chars'. This variable will store all the references to the individual characters found within the text of your heading.

```
<script>
var chars = [];
</script>
```

4. Split up the characters

For the next step we need to write a function within the <script> tag named 'textSplit'. This starts by finding our heading element based on the class name before reading the text contents of the heading. It then iterates over all its text characters and splits them into the 'chars' variable. By also replacing the contents of the heading with a separate tag per character, we can give each individual text character its own animation effect:

```
function textSplit() {
var title = document.querySelector('.title');
var splitText = title.textContent.split('');
title.textContent = '';
var character;
for(var i=0, len=splitText.length;   i<len;
i++) {
character = document.   createElement('span');
character.textContent = splitText[i];
character.style.display = 'inline- block';
chars.push(character);
title.appendChild(character);
}
}
```

5. Create the wiggle effect

Next we need another function within the <script> tag. Name this one 'textWiggle'. Add a GSAP Timeline to this function that will repeat itself when done, and for it to go back and forth. Within the textWiggle function, create a loop to create multiple random positions for each character within the heading. Now use the new cycle property within GSAP so that each character will get a different random position and thus a different animation as well:

```
function textWiggle() {
var timelineWiggle = new
TimelineMax({repeat:-1, yoyo:true});
for(var j=0; j < 10; j++) {
timelineWiggle.staggerTo(chars, 0.1, {
cycle: {
x: function() { return Math.random()*4   2;
},
y: function() { return Math.random()*4   2;
},
rotation: function() { return Math.
random()*10   5; }
},
ease: Linear.easeNone
}, 0);
}
}
```

6. Trigger the wiggling

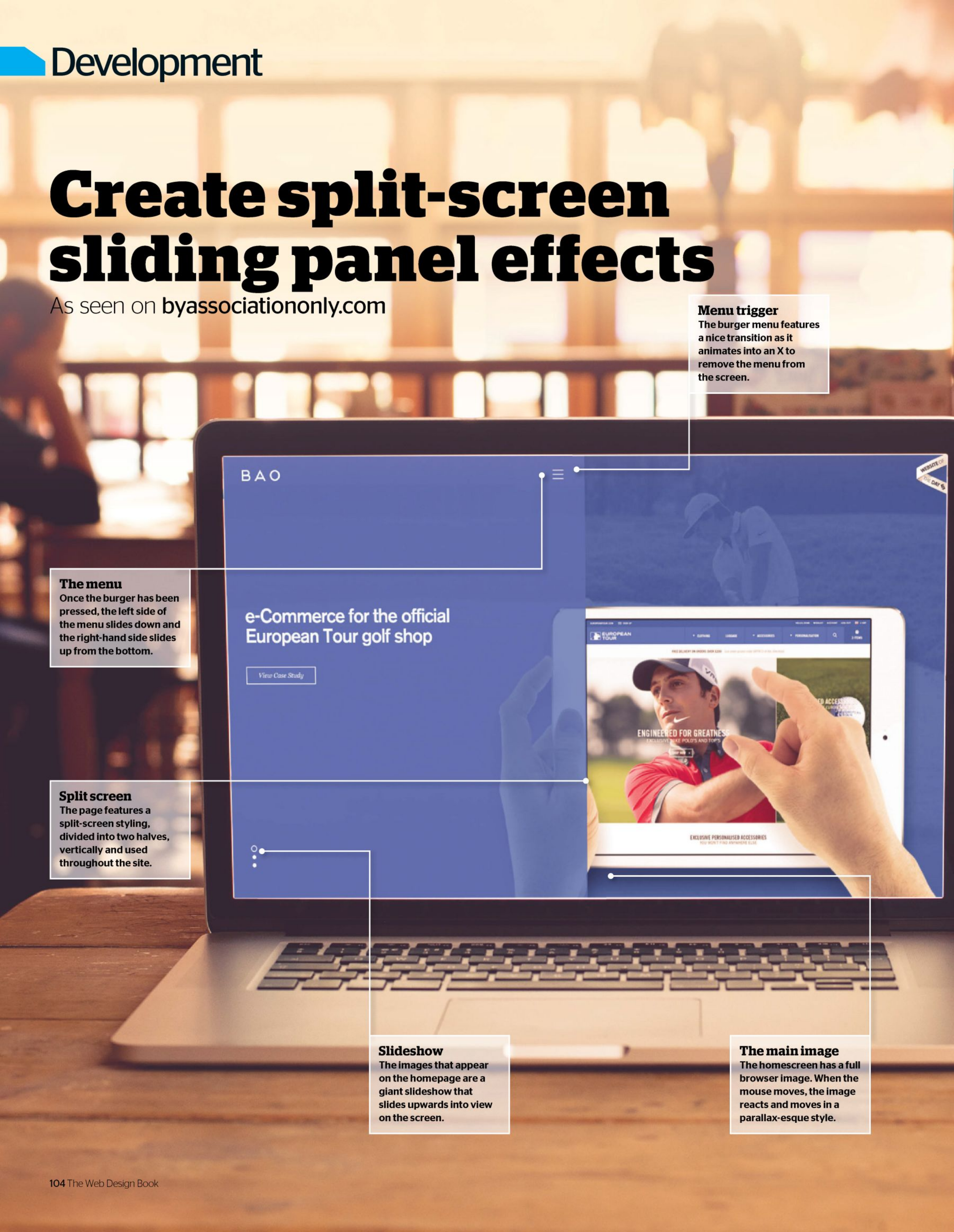
For our final step, all that is left to do within our code is to finally call the textSplit and textWiggle functions at the very end of the <script> tag to initiate the effect. Simply save your page and test in your browser to preview. You can then use this technique in various ways, so try changing the various parameters or come up with new animations to suit your own projects. Experiments with the techniques in this tutorial and see what you can come up with.

```
textSplit();
textWiggle();
</script>
```

> With a separate tag per text character, we give each its own animation effect

Create split-screen sliding panel effects

As seen on **byassociationonly.com**



Menu trigger
The burger menu features a nice transition as it animates into an X to remove the menu from the screen.

The menu
Once the burger has been pressed, the left side of the menu slides down and the right-hand side slides up from the bottom.

Split screen
The page features a split-screen styling, divided into two halves, vertically and used throughout the site.

Slideshow
The images that appear on the homepage are a giant slideshow that slides upwards into view on the screen.

The main image
The homescreen has a full browser image. When the mouse moves, the image reacts and moves in a parallax-esque style.

DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/bks-894

EXPERT INSIGHT
The rise of the preloader
Many websites now feature preloaders, all this does is present a holding screen until all the content is loaded. When using large images, it is essential to wait until they have loaded before presenting them to the user, otherwise they would think that the site isn't working properly with content missing.

The creators of the new By Association Only website state that the "hardest website to build is your own site". This is a statement that would certainly ring true with many web designers. Client work always takes priority so it can be difficult finding the time to create your own website, but at the same time you are always going to be judged by your own site, so its importance is crucial when looking at self-promotion. By Association Only were keen to create something innovative and different, but keeping a minimalist approach that emphasises usability, and so the process for creating their site had begun. They were actually close to finishing a site, but because it had been so long in the pipeline, they decided it would be better to start afresh and reimagine their brand. Realising that their own name was a bit of a mouthful, they rebranded their logo to an easier BAO letterform. The finished site features a consistent two-panel approach that splits the screen in half vertically but this isn't just for aesthetics though, as this style is also used as part of the menu design. This brings the form and functionality together into a coherent whole that makes sense for the user rather than simply satisfying a design aesthetic.

<comment>
What our experts think of the site

Finding visual inspiration

"As designers we find inspiration in every visual medium: movies, TV, magazines and posters. We knew we wanted to mix things up with our new website and it turned out that one old Seventies movie poster just happened to be that spark that informed the 50/50 design route."
Joe Trippett – cofounder/creative at By Association Only

Technique

1. Create the sliding panels

In your HTML page add the tags that are shown here. There are two panels for the left and right side of the screen and these will be hidden to start with, then animated when the burger menu icon is pressed in the third div. Another div is included just to show where the page content would go.

```
<div id = "left" class="hidden animated"> </
div>
<div id = "right" class="hidden animated"></
div>
<div id = "icon"><a href="#"
onclick="menu();">&#9776;</a></div>
<div id = "content">Regular page here</div>
```

2. Style the content

Move to the head section of the site and link to the animate.css file from **daneden.github.io/animate.css** as they will be used to trigger the movement. Then some of our own CSS is added in so that we can style up the background of the page.

3. Look left and right

Let's now add the style for the left and right panels, these are positioned on the page appropriately to the left and right side of the screen. They are also positioned absolutely so that they can appear above other content on the screen with a higher z-index. The sizing is then handled by percentage to fit the different screens.

4. Finish the styling

Now let's add in the remaining CSS, which will style the burger icon, link and the content that would appear on the page. The final CSS style will hide any content that shouldn't be seen until it is animated.

```
#icon{
position: absolute;
top: 0; left: 40%;
font-size: 3em;
z-index: 250;
}
a {
text-decoration: none;
color: #fff;
}
#content{ padding: 30px; }
.hidden{ display: none; }
</style>
```

5. Set the styles

When the user clicks the burger icon the menu function is called. This code swaps out the hidden property of the panels so that they become visible. The CSS styles that are applied come from the animate.css file that was linked up in Step 2.

```
<script>
var over = false;
var left = document.getElementById( 'left' );
var right = document.getElementById( 'right'
);
function menu(){
if (over==false){
left.classList.remove('hidden');
left.classList.remove('fadeOutDownBig');
left.classList.add('fadeInDownBig');
right.classList.remove('hidden');
```

6. Switch out

The rest of the class swapping code is added. Notice that a variable is also used to hold for whether the menu is over or not. This is then used to apply the correct classes to the code. This is all done without the need for jQuery, through vanilla JavaScript.

> Ever wanted to develop a native application but have been put off by learning a new language and ecosystem?

108 Testing, testing

If you want your apps to ever succeed, you need to master the art of testing

114 Animate with TweenMax

Control your interface animations with this powerful animation library

118 Build with PhoneGap

Learn everything you need to know about this essential design solution

124 Desktop apps with NW.js and JS

Build JavaScript-powered desktop apps that run on almost every operating system

Apps

130 Create a mobile 3D interface (P1)
Build an augmented menu that hooks into a mobile's device orientation controls

138 Develop a web app with Lucee
The Lucee language will be a breath of fresh air for many developers

144 Use Ionic to create a native app
Using Angular and web technologies you'll create a hybrid app for iOS and the web

150 Desktop apps with Electron
HTML, CSS and JavaScript come together for this cross-platform desktop application

TESTING TESTING

Take a look at our top-ten list to find out why you should test your apps and stick around to learn more about the tools of the trade

Unit testing started to become fashionable more than ten years ago. Its core idea is simple: since testing is a repetitive manual activity, it could also be handled by computers. Frameworks permitted the creation of small routines called unit tests, which were run automatically on request. Results could then be compared against predefined good and bad constants – the concept caught on like wildfire.

Unit tests are but one of the many automatic testing tools. As computational resources and research into artificial intelligence proceeded, static analysis became possible. It is based on a set of commonly encountered mistakes. Incoming product code is then compared against these known errors.

Since then, even more advanced systems saw the light of day. User interface tests manipulate the GUI stack of the platform, thereby simulating clicks and other user input. When set up correctly, most manual acceptance testing can be eliminated.

Fuzzers and Gremlins use random number generators and permutation logic to create pseudo-random input. They excel at sniffing out problems in input validation and/or parsing. Finally, load tests determine how your system behaves when bombarded with hundreds and/or thousands of requests at the same time.

You don't necessarily need to use all of the concepts that we've outlined – try using just one or two to give you significant boost in productivity.

Kacper Nowicki
CTO at GoEuro

"Testing code rigorously is an essential part of our production cycle. Because GoEuro has needed to scale so quickly, it's tempting to skip complete testing to release faster – but we've learned the hard way that there are no shortcuts."

10 REASONS WHY YOU NEED TO TEST

1. Reduction of legal liability

In many jurisdictions, vendors are required to ensure their products are of marketable quality. Having a documented testing process can help you if sued due to software bugs. Worst comes to worst, you can prove that there wasn't any malpractice, thereby reducing punitive damages or personal responsibility.

2. It's good practice

Professionals enjoy working with other professionals. Any vendor which does not have a dedicated testing process reeks of unprofessionalism. Not having a test system complicates the hiring of experienced staff. Software developers can just quit their job due to lack of testing and documentation processes.

3. It increases motivation

Performing repetitive tasks demotivates people. Eastern European outsourcing vendors punish their customers for turning in unsorted piles of bills due to the detrimental effect on staff morale. Automating your tests frees up developer and tester time for more important tasks.

4. It anticipates errors

The theory of falsification won a Nobel Prize. It states that a scientist should try to prove that his idea is wrong. It can be considered valid only if this process fails. Testing motivates developers to think about ways to break their code, thereby leading to more resilient applications.

5. It wins contracts

Some software-certification systems require the presence of a testing cycle. Not having one in place leads to your company failing inspections. When working for clients like the governmental or military, not having ISO certification is a deal-breaker and makes winning lucrative contracts difficult.

6. It reduces release anxiety

Some products suffer from slow release cycles: pushing out a product requires mental energy. Having a test cycle removes the QA step. This reduction in total effort tends to help anxiety situations as the total amount of effort is lowered by the reduction in the number of steps.

7. It simplifies fulfilling customer expectations

Ever had to turn down a feature request due to not being sure about how its integration will affect the rest of the product? If your code contains an ample amount of unit tests, adding a feature is as easy as coding it. Your computer will then take care of integration tests.

8. Tests are documentation

Ever suffered from not knowing how a newly acquired and quite complex system works? Well-written unit tests show how the individual classes work together to achieve one or more business tasks. Analysing that code permitts new developers figure out the internal workings of the system quickly.

9. Unit tests help you get zoned

Programmers and HR managers dream of the concept of the zone – a mental state where effectiveness reaches the highest possible level. Getting into it tends to require some introductory work, which is subject related but not particularly complex. Coding unit tests is an ideal example for such 'menial tasks'.

10. It prevents catastrophic errors

If you release an application where a crucial feature doesn't work it can result in huge monetal losses. A reliable set of tests mitigates this by ensuring that no products get shipped with significant defects.

6 TYPES OF TESTING

1. A/B testing

Design meetings get bogged down by never-ending squabbles. A/B testing remobilises teams which have come to a deadlock. It can also optimise conversion. An A/B test involves the creation of two variations of your product. Customers are served one of two versions at random, with metrics being captured on the fly.

2. Beta testing

If a developer can't find willing testers, the product's concept usually is deeply flawed. The value of a beta test lies in setting customers loose on the product. While they are not able to replace the work done by 'normal testers', a well-run beta test tends to unveil all kinds of issues related to functionality, usability and discoverability.

3. Acceptance test

An acceptance test is a group of tests which are to be run before the product is released to customers. Since the usefulness of the test tends to be directly related to the time needed for its execution, the daily run should be compact. Since acceptance tests are run rarely, they can test uncommon cases.

4. Integration test

Integration tests look at the large picture. They analyse the cooperation between individual modules: in the case of a database and transaction system, it could model a transfer of money or precious metals from one account to another. Due to their increased complexity, integration tests are not run after each commit.

5. Fuzzing

Fuzzing uses mathematical models to create valid input invalidated by adding or removing elements at random. Fuzzing has some security problems but can be effective when dealing with an API or a service: user-facing applications tend not to benefit much if they don't read files.

6. Unit testing

A unit test is a test framework which runs one or more simple tasks and compares results to the provided values. If your test case spans multiple pages, it is too complex. Code coverage is associated with unit tests: it specifies the amount of code touched during test suite execution.

BROWSER TESTING

CODE VALIDATION

JavaScript's amazing evolution permits developers to create ever-more complex web apps. Unit tests have established themselves as a classic pattern: not implementing them leads to increased fragility and makes maintenance difficult. A large selection of unit testing tools for JavaScript has cropped up in the last few years. Their concept is simple: developers create small routines commonly known as test cases. Each case tests one or more workflows and compares the computed results with good values provided in advance. Since most frameworks provide dedicated test runners, verifying the correctness of your application is as easy as invoking a single website. Some products even go as far as to integrate itself into node.js and similar test runners.

A SMALL TEST

Since the concept of unit testing is not commonly taught in web design courses, we will take a small look at an

4 ESSENTIAL VALIDATION TOOLS

JSLint
jslint.com
Crockford's JSLint checks JavaScript for all kinds of common programming problems. However, the product uses a very rigid approach, so be prepared for a steep learning curve.

JSHint
jshint.com
JSLint's pedantism gave rise to JSHint, which has many options for tailoring its validation process to your personal coding style or your company's own coding conventions.

Scan.js
bit.ly/1SPIpFI
Scan.js transforms its input into a syntax tree, which is then compared against a database of known bugs and inconsistencies. This lets Scan.JS find common and obscure bugs.

Jasmine
jasmine.github.io
Jasmine recently managed to gain a cult following due to its English-like domain specific language used for specifying the individual test cases:

Thousands of browsers compete for attention in today's world. While websites might not always have to support each and every superobscure HTML rendering engine, covering all major browsers is helpful when it comes to maximising reach.

In an ideal world, every browser would behave the same: rendering a piece of content would lead to the same bitmap on each engine. Sadly, browser vendors tend to take significant liberties when interpreting the specifications declared by the W3C.

Ensuring that a website runs on all browsers is best accomplished by using a well-known strategy to J2ME developers. If you ever heard them ranting about 'write once debug anywhere', you know just too well what's in store for you.

In the past, developers solved this problem by installing a large variety of browsers on their workstation. Since this bogged down their operating system, hardware-accelerated virtual machines were used to keep the browser installations at bay.

Since browser testing has become a very common problem, vendors started offering virtual machines containing various browsers. The features offered by the individual vendors differ. While some limit themselves to screenshots displaying your website in action on various browsers, other companies permit you to access the individual virtual machines which usually come preloaded with a variety of helper tools.

Watch your audience!

Even though browser market share stats tend to provide a reliable 'first glance overview' of usage, keep in mind that some customer demographics may have varying usage behaviours.

PAY MORE, GET MORE

While most vendors provide some kind of basic service at no cost, paying customers tend to be treated to extra features. The most important one is the automatic execution of unit tests. This means that the virtual machines will be spun up in order to be set loose upon a predefined set of Selenium, QUnit or other similar tests. Their results are then collated by the vendor, providing you with an accurate report of the health and sanity of your web application across various browser implementations.

A QUESTION OF PRIVACY

When deploying vendor-provided testing solutions, you must always keep data security in mind. Executing unit tests requires you to give your source code to the vendor: this might not be acceptable in proprietary or highly classified applications. In this case, setting up a stack of virtual machines remains the best approach: products such as VirtualBox and VMware can simplify the process for you.

> *"When deploying vendor-provided testing solutions, always keep data security in mind"*

4 ESSENTIAL TOOLS

SauceLabs
saucelabs.com/home

This company is one of the oldest vendors in the browser testing market. You can pick from a manual free version and a variety of paid automatisation options which run sets of unit tests for you automagically. Sadly, the latter service starts out at $16 (£8) a month for two hours worth of automatic testing.

Spoon
spoon.net/browsers

Spoon provides a set of virtual machines which can be run from their website. Their offering is ideally suited to quick explorative testing: if you are not sure about how a web app looks in an obscure browser, simply visit them from a Windows box and find out more.

Microsoft DevEdge
dev.modern.ie/tools/vms

Being the source of the most problematic browser, Microsoft is well aware of its obligation to web designers. **Modern.ie** provides a variety of fully configured virtual machines with various versions of IE – just pick the one best suited to your testing situation!

VirtualBox
virtualbox.org

VirtualBox is a highly reliable open source VM host which can be considered the standard on Unix workstations. A VM setup requires specifying hardware resources and installing the OS and any required tools.

example. QUnit, being developed as part of the jQuery product family, makes for a good starting point.
First of all, a test fixture must be created. The official documentation recommends the use of the following structure – but make sure that you adjust the file paths before deployment:

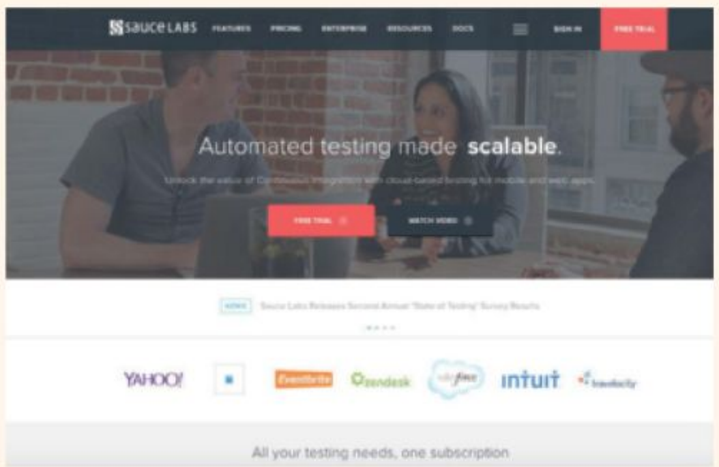
```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>QUnit Example</title>
<link rel="stylesheet" href="//code.jquery.
com/qunit/qunit-1.18.0.css">
</head>
<body>
<div id="qunit"></div>
<div id="qunit-fixture"></div>
<script src="//code.jquery.com/qunit/
qunit-1.18.0.js"></script>
<script src="tests.js"></script>
</body>
</html>
```

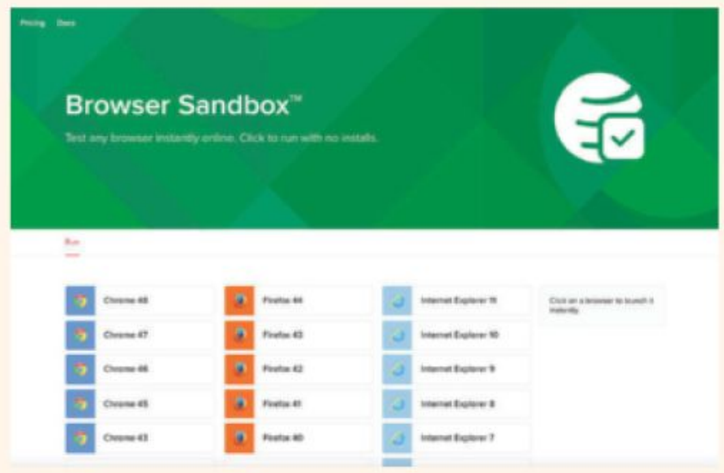
Test cases are stored in one or more JavaScript files. The test function takes a name and a function pointer to a method containing the actual code to be run:

```
QUnit.test( "hello test", function( assert
) {
assert.ok( 1 == "1", "Passed!" );
});
```

The actual comparison between returned and ideal results is then performed via assert commands. 'Ok' is the most basic of the batch, ensuring that the parameter passed in evaluates to true:

```
QUnit.test( "ok test", function( assert ) {
assert.ok( true, "true succeeds" );
```





EMULATED BLICKS

Unit tests reach their limit of usability when testing very complex workflows. In this case, it is more sensible to use a product like Selenium WebDriver. Its test cases are created by interacting with your web app and how it does this is by tracking the buttons and characters that are entered. Results are compared against known-good results using bitmap and text content analysis.

Well-done WebDriver tests will work by analysing both user interface and logic at the same time, thereby leading to a higher quality of test coverage as a result. Sadly, such tests also tend to be more brittle. They can be caused to break both due to changes in the user-interface, in the HTML rendering engine as well as in the actual code.

LET'S SCAN ONLINE

Google Webmaster Tools

google.com/webmasters/tools

The famous search giant's Webmaster Tools alerts you to a variety of problems encounted by the crawler. Fixing them will increase search visibility significantly.

PageSpeed

developers.google.com/speed/pagespeed

Google's PageSpeed tool analyses your website and provides an array of impressive fix suggestions: from caching optimisations to the bundling of requests.

Nibbler

nibbler.silktide.com

SilkTights Nibbler service performs a complete analysis of your website and grades it in five disciplines. Various improvement and remedy suggestions are shown.

SEO Centro

seocentro.com

If there is a possibility of an SEO problem, SEO Centro is likely to have a diagnosis tool for you. Spending a few minutes with their various utilities is recommended.

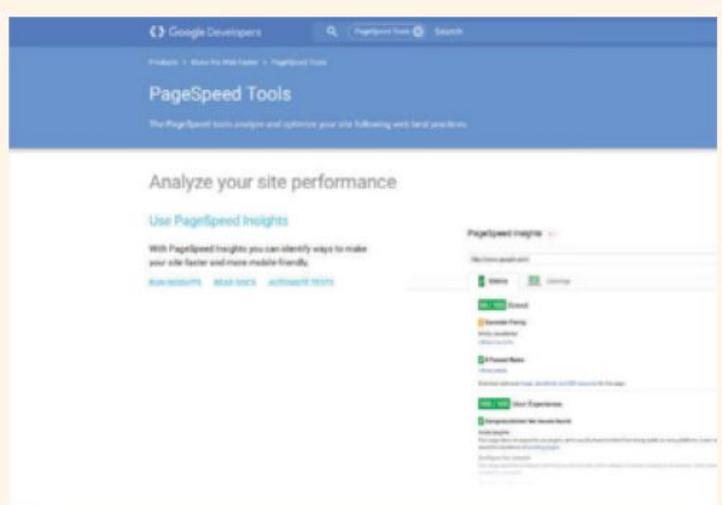
> *"SilkTights Nibbler service performs a complete analysis of your website"*

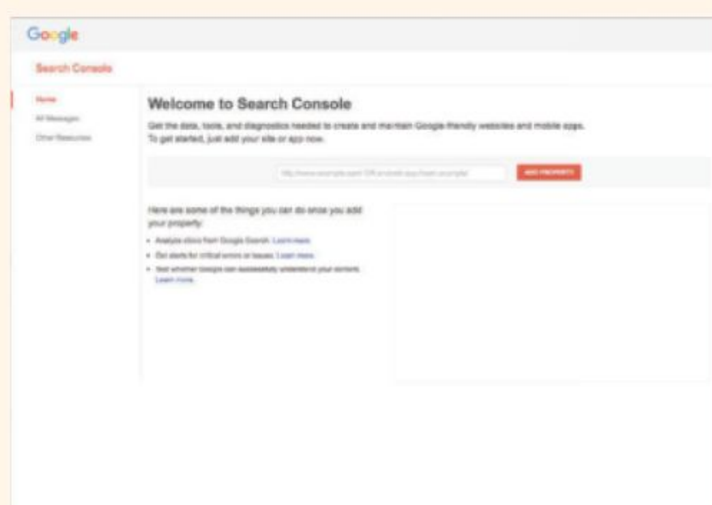
BETTER PERFORMANCE

Keeping an eye on website performance has never been more important. Back when broadband was introduced, sites seemed fast, but designs soon caught up and added a whole host of bandwidth-hogging features. Google's decision to penalise slow sites was the straw that broke the camel's back and performance frenzy has taken hold of web masters and their managers ever since.

Tests performed by various research bodies have determined that slow websites actively lose customers: making your websites load faster is a sure-fire way to improve conversion rates. Ookla Speedtest is just one nifty tool which is worth its weight in gold when it comes to ensuring the reliability of your internet connection.

When it comes to figuring out a website's speed, a large variety of different factors must be kept in mind before you start. The actual loading time of a particular website is made up of three components: First of all, the server must generate the content that needs to be displayed. Second, this content must be deployed to the user. Finally, rendering must take place on the customer's workstation. Optimising the content generation process can be accomplished in a variety of fashions. First of all, increasing the amount of performance available to your server directly reduces the amount of time needed to handle requests. Many web application frameworks provide a large variety of utilities to optimise performance: in the case of WordPress for example,





caching plugins store generated content in order to accelerate its redeployment.

Cutting down actual transfer times can be accomplished by reducing the size of images, minifying CSS and JavaScript and performing a variety of similar tricks. Setting cache headers instructs clients browsers to keep images and similar files on the local disk: this cuts back loading time and also frees up bandwidth.

Finally, some markup tweaks can be done if you wish, in order to help accelerate rendering. Setting image widths, for example, will permit browsers to start laying out the page even if the actual picture files in question have not yet been successfully transferred to the client's computer.

Testing the speed of a website or webpage is a skilful art in its own right, and should be treated as such. The sheer size of the earth has the unfortunate repercussion that the time to transfer data significantly effects performance: an American downloading from a server in Europe is likely to see significantly slower transfer rates. Fortunately, this problem can be properly addressed by the deployment of content delivery networks.

Performance testing tools come in two groups. Utilities such as PageSpeed or YSlow perform a static analysis of the delivered content, providing you hints and advices on how to increase the performance of your page.

Down and out

Sometimes it happens that a webmaster may call a vendor to complain about downtime. They then investigate the issue, only to find out that the problem was actually located at their ISP.

Load testing tools such as LoadImpact have a large array of worker-drone workstations, which can be used to determine the impact of heavy loads. The level of performance needed, of course, differs from site to site: if you are unlikely to attract large amounts of customers and/or DDOS attacks, being able to survive seventy thousand customers per second might not be as important a step.

3 ESSENTIAL TOOLS

Microsoft Azure

visualstudio.com/en-us/get-started/test/load-test-your-app-vs

Microsofts offers you 20,000 free users every month. Just log into Visual Studio Online to configure load tests; owners of the Ultimate Edition of Microsoft's IDE can kick the test off from their workstation.

Pingdom

tools.pingdom.com/fpt

Provide Pingdom with a URL to be rewarded with a detailed breakdown of loading times and optimisation hints. The DNS and Ping tests let you check your server's response times from a variety of locations.

LoadImpact

loadimpact.com

If you need a lot of users, Load Impact are your go-to place. Hit them up to get access to hordes of drone systems. The free plan provides up to five minutes of usage – enough for a quick check.

ACCESSIBILTY

When working on government contracts, accessibility for disabled and/or visually impaired users is of significant importance. Both American and European governments have passed a variety of standards specifying parameters which are to be met by 'conforming websites'.

Fortunately, a selection of testing utilities are available. They compare your markup to a list of problems known to bring screenreaders out of sync. Another classic involves loading a website with the Lynx browser found in a command line of choice: its rendering engine is said to be very similar to the one used by screenreaders.

USABILTY

An old adage states that some children are so ugly that only their mother can love them. This is often the case when it comes to dealing with user interfaces. The history of IT is full of cases where great GUI ideas simply did not work out when implemented in real life.

Usability problems are best snuffed out by beta testing: get some customers to try your product before it goes live, and observe their behaviour to find out areas which may trip them up. Keep in mind, though that beta testers get accustomed to your GUI as time goes by – so frequently adding new testers and fresh eyes can be helpful.

SECURITY

Getting hacked is probably one of the worst experiences in the life of a webmaster. Keeping a web app safe is an art of its own: security professionals must be brought in if a product is intended to handle high-value data.

For smaller web apps though, following the guidelines in the OWASP web application security testing cheat sheet is a good first (bit.ly/1T5eyJI). Secondarily, try to keep all components of your web server up to date. Vulnerability scanners analyse your web server for known weaknesses, while port scanners inform you about places where attackers can connect to your system.

DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/bks-894

Animate an interface with TweenMax

Control your interface animations with TweenMax, a powerful animation library that can animate anything through DOM elements, Canvas and SVGs

Animations are now an integral part of **the front-end designer's skillset.** Having an understanding of how to create and add animations can really enhance the user experience of a product and makes the interface come alive. A well-timed shake to show the user has input the wrong values will link with the user's mental model of a negative shake of the head. Sliding the interface to some new part can be a compelling and satisfying resolution to clicking the interface.

Understanding CSS animation and transitions is crucial to becoming good with these, but sometimes the front-end designer needs a helping hand. They need a consistent way to write animation that works in all instances whether animating DOM elements, Canvas illustrations, SVGs or WebGL 3D objects. In this case you will need to use GSAP's TweenMax. TweenMax runs 20 times faster than jQuery so no need to worry about performance on devices. Libraries such as jQuery, Zepto, Velocity and Transit can only animate CSS properties, but TweenMax can handle anything. Add to that its ability to sequence and handle browser inconsistencies, then you have a serious contender for the most advanced animation platform that there is. It doesn't try to interfere with how you structure parts of your frontend, it simply looks after animation alone, making it incredibly useful.

1. Link to TweenMax

The TweenMax library by GSAP can be downloaded from the website (**greensock.com**). If you click the Download GSAP button there is an option to link to the CDN version of the library; copy and paste that line into the head section of the index.html file.

```
<script src="https://cdnjs.cloudflare.com/
ajax/libs/gsap/1.18.0/TweenMax.min.js"></
script>
```

2. Link up the style sheet

A lot of the content has already been styled up ready to go. The tutorial will take you through one or two CSS classes that are important in both the design and animation of the project. As before in the head section, just link up to the stylesheet as shown.

```
<link rel="stylesheet" type="text/css"
href="style.css">
```

3. Define the page

Open the style.css file and style up the body. This sets the background image to cover the entire screen. The font family is set to Helvetica and the overflow of the page is hidden in case any of the elements hang over the bottom of the page as it stops the scrollbar showing.

```
body {
width: 100%;
height: 100%;
background: url(img/bg.jpg) no-repeat center
center fixed;
background-size: cover;
font-family: Helvetica, Arial, sans-serif;
overflow: hidden;
}
```

4. First column

The design is being split into two columns, the first column is on the left of the design. The padding is to ensure the inner content to this starts 40 pixels in from the top and left. The logo and menu appear in this column so it doesn't need to be too wide.

```
#left {
padding: 40px 0 0 40px;
width: 135px;
float: left;
}
```

5. Second column

The next column is the main column that the majority of content will go into, especially if this was a real site. However this will contain several elements that will be animated onto the page so that attention is drawn to them as the page shows.

```
main {
float: left;
margin: 245px 0 0 150px;
}
```

6. Create the logo

At the moment the logo looks lost as it is dark against the dark blue sky. Here, the code styles the outer container that holds the logo. Later in the tutorial, TweenMax will be used to animate this element onto the screen and fade into view.

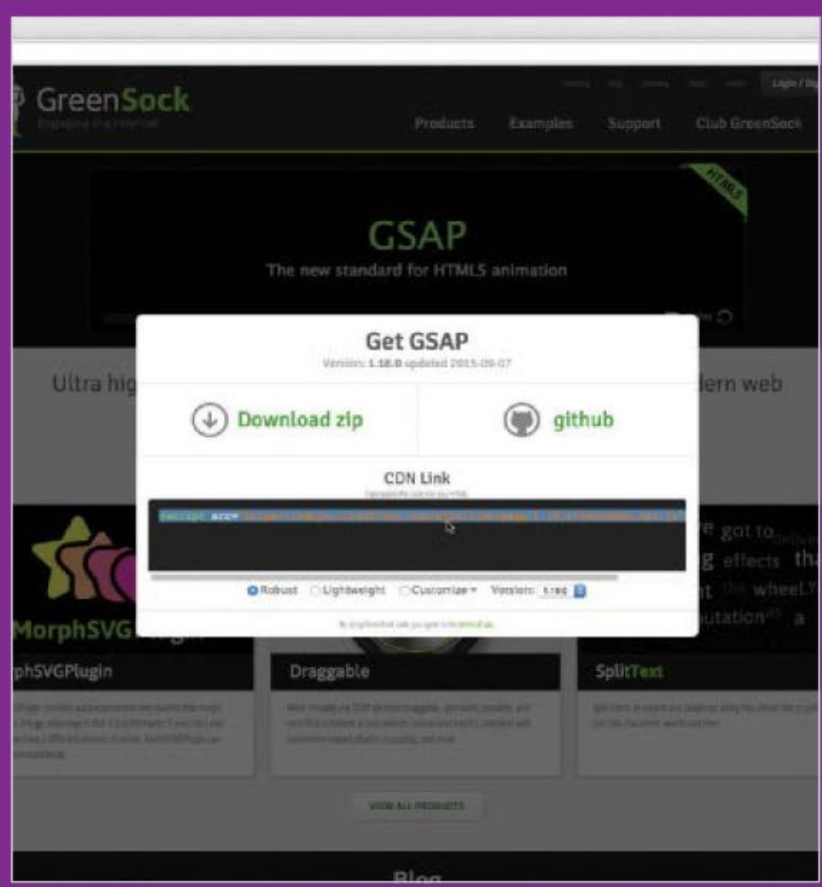
```
#logo {
position: relative;
width: 135px;
height: 135px;
background: #eddaeb;
}
```

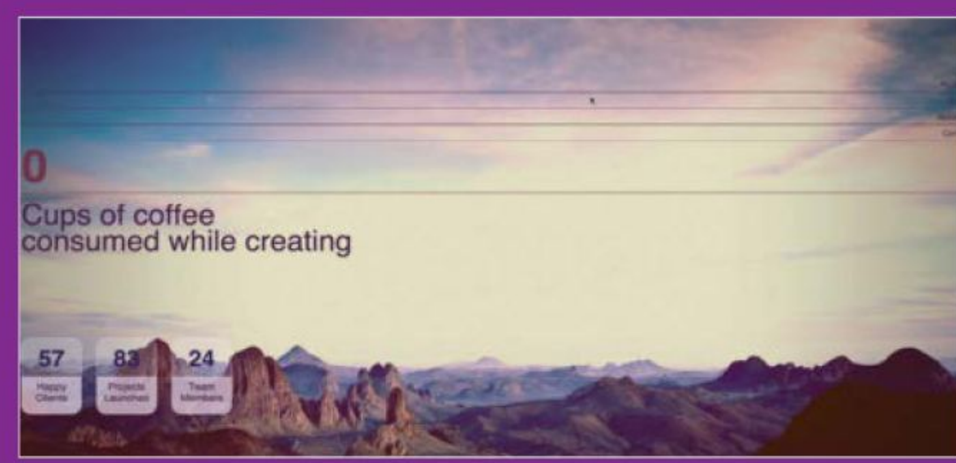
7. Position the logo

Now as there is a light block of colour placed behind the logo, the logo text should be moved downwards slightly

DOM elements

TweenMax can find your DOM elements by tag, ID or class, in much the same way as you would write CSS for each of these elements, with just the tag, the hash or period, respectively.





<Top left>
• The basic styling of the page is added so that there is a background image and a colour palette used that will enhance the animation on the page

<Bottom left>
• Left and right columns are added to hold the page content, this shows how TweenMax animation can be used with a web design

<Left>
• From the main GSAP site, the CDN for the TweenMax library can be linked to so that it can be cached by the web browser

so that it can be positioned in the centre of the square on both the horizontal and vertical axis. You can save the CSS file now.

```
#logo img {
position: relative;
display: block;
margin: 0 auto;
padding-top: 38px;
}
```

8. Link up the JavaScript

Back in the index.html page, scroll down to the bottom of the page and insert the link to the main.js file. This file is where all of the TweenMax JavaScript commands will go to power the animation. You can open that file and you will see that it is empty at the moment.

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

9. Animate the logo

Add the first animation code as shown to animate the logo. The command tells TweenMax to animate from these values given. It animates the logo for one second, fading in the opacity, scaling it up and rotating it from off the screen to the left. It's given some easing so that it eases back into position.

```
TweenMax.from("#logo", 1, {
opacity: 0,
rotation: 360,
scale: 0,
left: -200,
ease: Back.easeOut
});
```

10. Delay animation

Now we tell gulp about the plugins, directories and files to monitor. Gulp uses a gulpfile.js file in the root folder of our project to configure tasks. Starting with the gulp-jscs plugin, we will tell gulp to watch the src folder for changes and then to run all the files contained through the JSCS tool.

```
TweenMax.to("#logo", 1, {
borderRadius: "50%",
delay: 0.9
});
```

11. Fade the logo in

Instead of the logo being in the square as it moves onto the screen, a fade-in of the logo will happen. This is delayed by 1.2 seconds and takes place over the duration of one second. Save the file and test the animation in your browser to see it working.

```
TweenMax.from("#logoTxt", 1, {
opacity: 0,
delay: 1.2
});
```

12. Powerful staggering

All of the links need to be animated in. It would be good if these could be staggered so that each link comes in one at a time. Fortunately there is a staggerFrom command that is used here. It is applied to any link class on the page. There is a 0.2 second stagger between each element.

```
TweenMax.staggerFrom(".link", 0.5, {
opacity: 0,
left: -200,
delay: 1.2,
ease: Back.easeOut
},
0.2);
```

13. Animate the text

There is a large zero on the page, this element is going to be targeted so that it can have its text value increased and animated. This will allow the text to whizz up through the number of coffees drank. The first step in this is to set some variables.

```
var myText = document.
getElementById('lgNum'),
endVal = 2372,
currVal = 0,
num = {};
```

14. Get the value

The num object, short for number, that was created in the previous step now has a function attached to it. This allows TweenMax to get the value of the text as an integer, which is a whole number so that there are no decimal places displayed on the page.

```
num.getTextVal = function () {
return parseInt(currVal, 10);
};
```

15. Set the value

Similar to the previous step, another function is added to the num object this time to set the value of the variable and this adds the new text value to the page. The page has to be explicitly updated so that the new value is visible, hence the num object will be tweened.

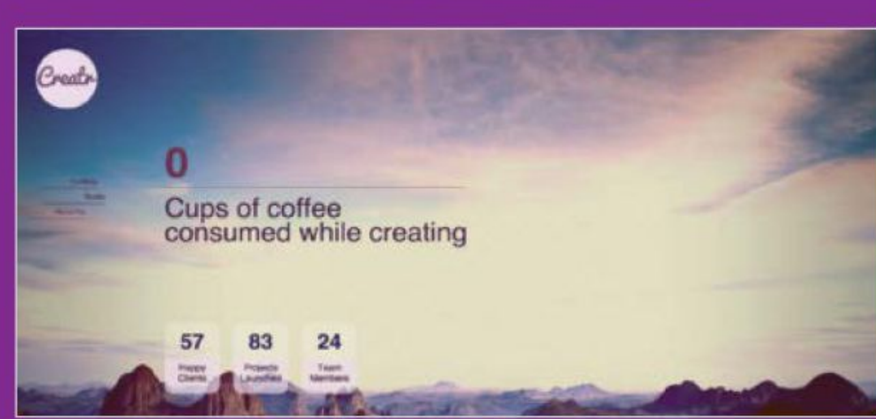
JavaScript objects

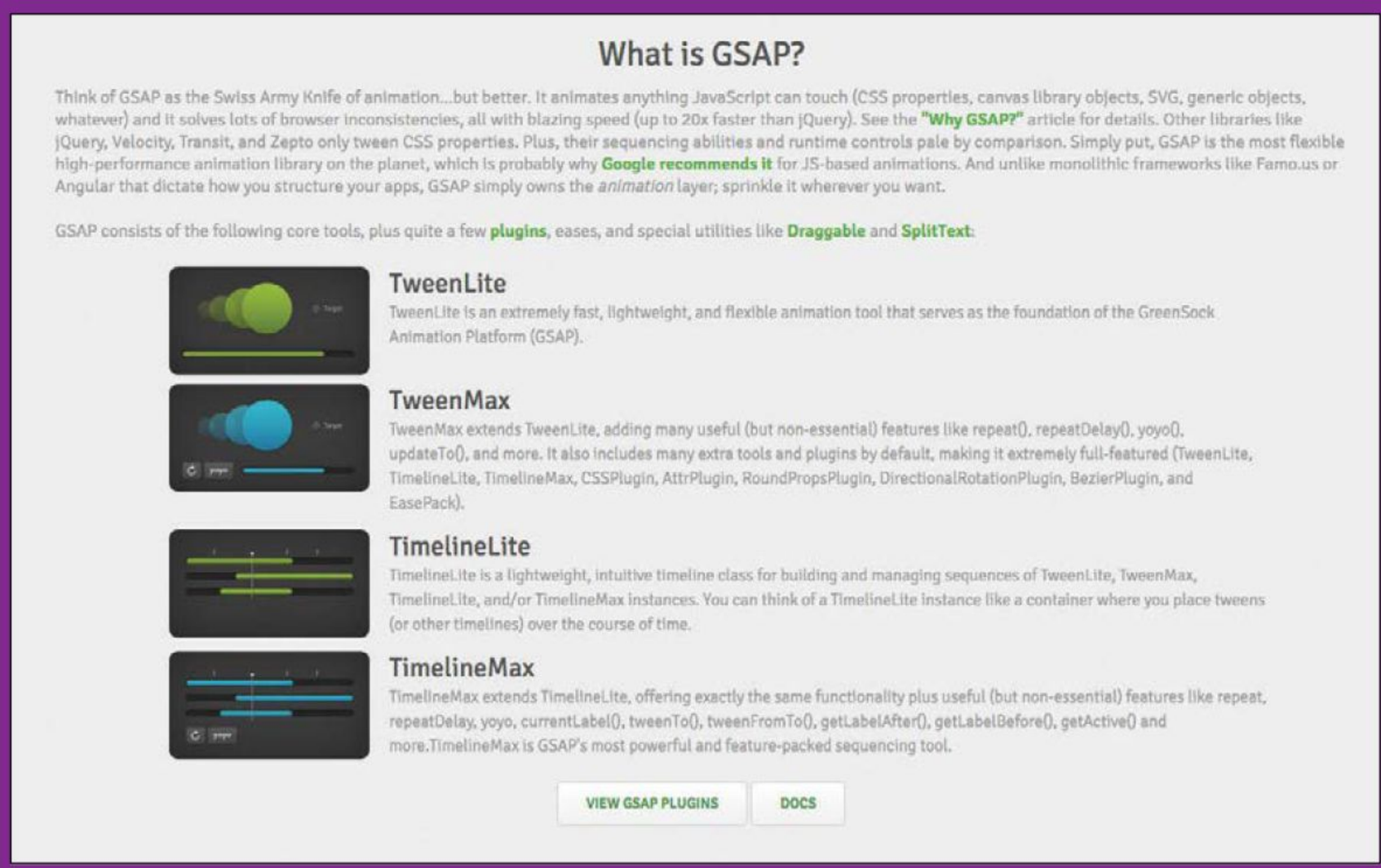
Creating an empty object in JavaScript is as simple as assigning curly braces to a variable, hence num = {}; in the tutorial. Functions can be added to these objects.

<Top left>
• The logo is animated into place using the 'back' easing so it slightly overshoots the target and then smoothly eases into place

<Bottom left>
• Each of the links animates into place using the powerful 'stagger' feature of TweenMax, which allows every instance of a class to be animated into place separately

<Right>
• Not only can positions be animated with TweenMax but content in the div elements can be animated. Here the large number is animating up from zero to 2372





TweenMax and the GSAP platform

TweenMax is just one product on offer from GSAP. The four main ones are TweenLite, TweenMax, TimelineLite and TimeLineMax.

The difference between TweenLite and TweenMax are that TweenLite has less features than TweenMax so it is quicker to load and as it doesn't have the ability to repeat or yo-yo, it should execute slightly faster. But GSAP claims that as both are highly optimised, the differences should be virtually imperceptible. Use TweenLite purely for shorter loading time.

TimelineLite contains all the essentials of a controlled timeline-based animation system. TimelineMax also has AS3 like event dispatching. Both have the ability to pause animation and reverse animation.

```
num.setTextVal = function (val) {
currVal = parseInt(val, 10);
myText.innerHTML = currVal;
};
```

16. Initialise zero

The num objects's setter function is called to set the text value to 0, while it is on the screen it's the first step that must be done in order for the tween to start moving up through the numbers and the coffee consumption can be displayed on the webpage.

```
num.setTextVal(0);
```

17. Tween that value

After setting up the background work for the num object the tween animation is fed into TweenMax. It's set to run over three seconds but delayed to only start after one second. SetTextVal is called to tween to the end value.

```
TweenMax.to(num, 3, {
setTextVal: endVal,
delay: 1,
ease: Circ.easeInOut
});
```

18. Fade in the number

As the number is spinning up, it would be good if this faded in so it wasn't just sat at 0 for a second. Adding the following tween will do that and move it in from 200 pixels above its current position. The tween for the fade-in lasts just less than a second.

```
TweenMax.from("#lgNum", 0.8, {
opacity: 0,
top: -200,
delay: 1
});
```

19. Fade in the text

Below the number is some text and this code sets that to tween in just after the number is on the screen. It will fade in from below and move upwards into place on the screen. Save the page and reload the browser to watch the effect of the different elements animating.

```
TweenMax.from("#mainStrap", 0.8, {
opacity: 0,
bottom: -200,
delay: 1.8,
ease: Cubic.easeInOut
});
```

20. Stagger around

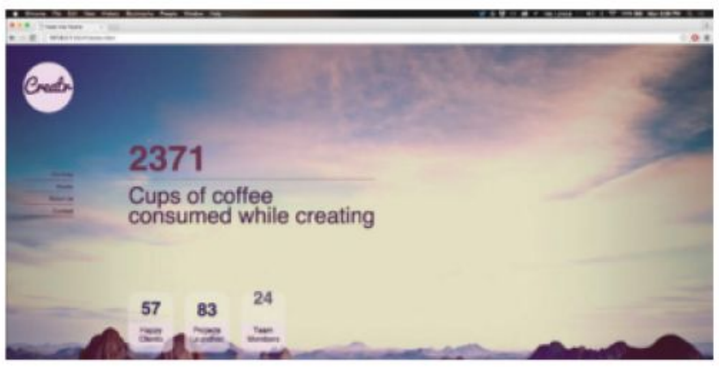
At the bottom of the page are three panels and as in Step 12, they can be staggered in so that each one animates in turn. Here the panel class is located and each of those is animated in from the right, with a slight spin into place. The animation starts after two seconds and each is given a half-second pause between animating.

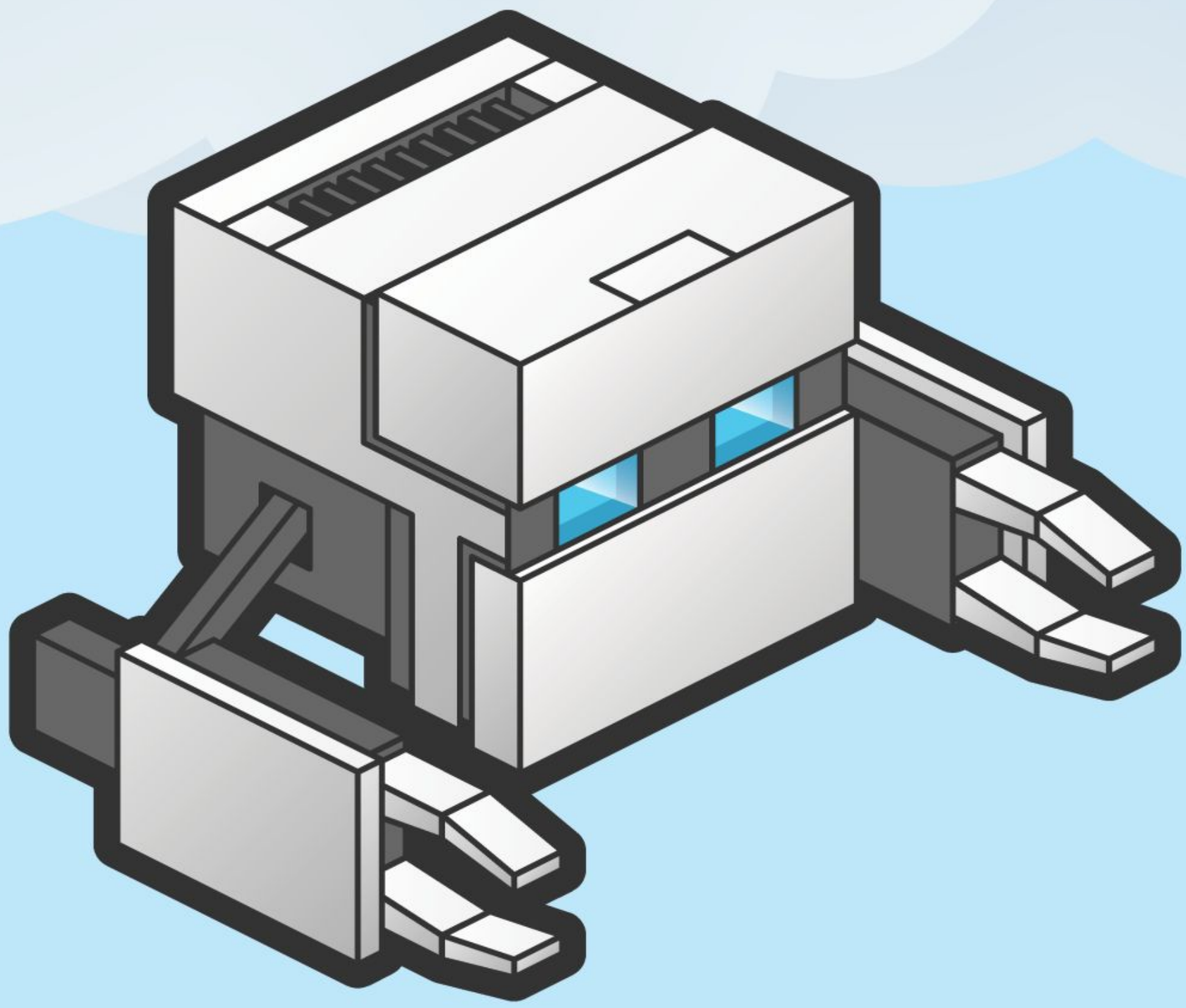
```
TweenMax.staggerFrom(".panel", 0.8, {
opacity: 0,
rotation: 45,
left: 100,
delay: 2,
ease: Back.easeOut
},
0.5);
```

21. Drop in the numbers

In the panels there are numbers and those are now going to be dropped in one at a time so that they appear inside the panels. Again this is a stagger animation. Save the page for the final time and refresh the browser to see all user interface elements now animating into place.

```
TweenMax.staggerFrom(".mdNum", 0.8, {
opacity: 0,
top: -100,
delay: 2.8,
ease: Back.easeOut
},
0.5);
```





Build with PhoneGap

Design and build apps with PhoneGap by using your existing web knowledge to publish for multiple device app stores

As a web designer you've cut your chops learning how to create beautiful designs on the web with HTML, CSS and probably even JavaScript. The beautiful part about designing for the web is that it is ubiquitous. Provided you're sticking to web standards, your design should look consistent bar the odd browser quirk.

When the app scene arrived in the latter half of the Noughties, development became split into silos with development teams needing to know Objective C and the Cocoa Touch framework for iOS, Java for Android or C# for Windows phone. Yet the web community was probably the most able to design for these, given their ability to work across multiple platforms already.

This is where PhoneGap came into being, in order to fill that hole. PhoneGap was a series of templates for each device that upon starting them as an app, they opened up a full-screen web view inside the app. This meant that anyone who could write HTML and CSS, could create their own interfaces for apps. But PhoneGap wasn't just a design solution, it was a full development solution. PhoneGap opened up the native features of the phone such as geolocation, the accelerometer, the contacts, the camera and the network status, so that developers could create apps but just call those features using JavaScript rather than learning the ins and outs of the specific SDKs. What's more, the JavaScript commands are the same regardless of which platform you choose.

These apps became known as a hybrid solution. They were installed as native apps on the device and could access native features of the device, but used web technology to build them. In the early days they operated slow. It was easy to tell that they were PhoneGap apps because they would act like webpages instead of native apps. As JavaScript has become much faster and the web views much better at rendering, a well-coded and put-together PhoneGap app can be difficult to tell apart from a full native app.

PhoneGap applications are generally single-page apps. That doesn't mean a single page in the app, but rather it has a single HTML page. This seems strange at first, but it enables JavaScript variables to stay in memory. When a new HTML page loads, everything from the previous page is inaccessible – so having one HTML page means that preferences and functionality work from screen to screen on the app.

Though the designer is using web technologies, the design process follows a different methodology. A network connection can't be taken for granted – this impacts the use of fonts in the design, as they have to be downloaded to the app, and not linked to Google Fonts or a similar service. Despite these various differences, the pay-off for using PhoneGap nonetheless makes for a faster end-to-end build service for developers.

Install and set up PhoneGap

Setting up PhoneGap is a breeze with the new Desktop app, learn how to sync this with your phone

The PhoneGap Desktop app provides an easy way for you to create and sync the app you are working on, from the desktop to your phone. The PhoneGap app must be installed on your phone for this first though, and once this is done you can then easily pair it up with your devices.

1. Desktop app

Download the latest version of the PhoneGap Desktop app from **docs.phonegap.com/getting-started/1-install-phonegap/desktop**.

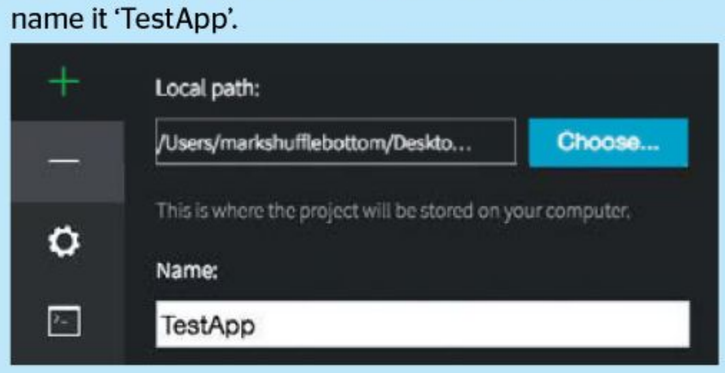
The Desktop app is an alternative to using PhoneGap CLI (command line interface) and contains various packages such as Node.js within it. Once it downloads, simply double-click to start installing it on your computer.

2. Install on your computer

Follow the on-screen prompts to install the software for your operating system. The Windows installer requires you to select a location to install to. For the Mac, you simply drag the PhoneGap icon onto the applications alias to install.

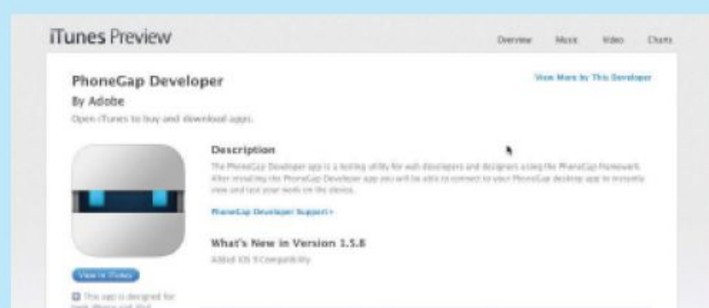
3. Start the software

Double-click the PhoneGap icon to start the software. This will start Node.js running in the background, which will allow the user to pair their desktop with their phone. Click the + icon to create a new project now. Then choose a location to save the app and then name it 'TestApp'.



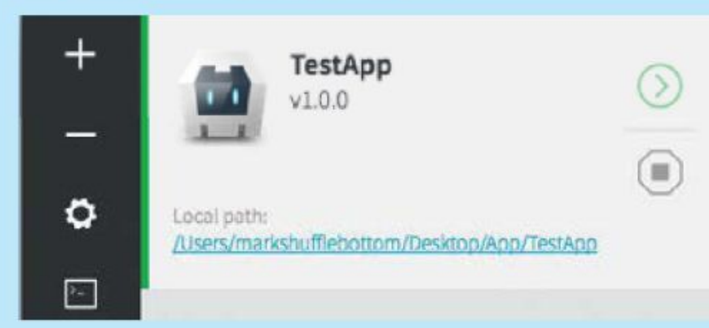
4. Install the phone developer app

Go to your app store for the phone that you have; the developer app is available for iOS, Android and Windows phones. Install the software onto your phone or tablet and start the software running on the device. Make sure your device is on the same Wi-Fi network as your computer.



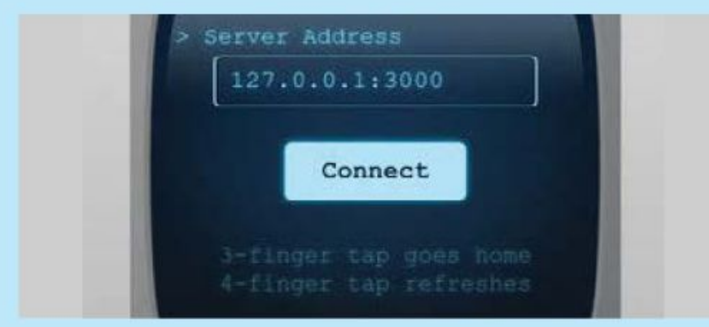
5. Start the server

Back in your desktop app click the 'Create Project' button and your new app will be created. There is a sample app created to kick-start your development. You will see that at the bottom of the screen is the text 'Server is running on' and then your network address, keep this in mind for the next step.



6. Connect the device

Back on your device will be a space to add the server address. Add the address that is being displayed on the desktop app. It runs over port 3000 so add the ':3000' at the end of the address and click the Connect button to see the app transferred from the computer onto the device.



The PhoneGap philosophy

Get the most out of PhoneGap by understanding the goals and ambitions of the project

Like all good design solutions, PhoneGap solves a very real problem for developers. If you need to make an app and you need to release it across different mobile platforms, then you have to set aside a huge amount of time, to understand those platforms and to take full advantage of what they have to offer. If only there could be an easy way for you to use well-known development skills to publish your app across many different platforms!

PhoneGap was first conceived at Nitobi Software, a small web consultancy studio in Vancouver, Canada, in 2008. One of their core beliefs was that the web had solved cross-platform development. Yes it has its quirks with different browser implementations, but generally a webpage looks and operates pretty consistently across all platforms and browsers. Take a look at one of the biggest success stories in software development in the last 20 years, Google. They operate by releasing software on the web, whether that's a search engine or a collaborative document authoring tool. There's no doubt that web technologies offer the easiest way to reach everything from internet TVs down to your Grandma's £45 Android phone.

The second belief Nitobi had was that all technology deprecates with time. This was a simple observation from looking at different platforms historically. At

some point or another a platform will die and if you've spent time exclusively with one that has died, then the consequences for you can, unfortunately, be disastrous. Stranded on an island of outdated technology and skills, you must quickly cobble together a raft and get back to civilisation.

The smart readers may have noticed that the second belief (all technology deprecates) kind of clashes with the first belief (the web solved cross-platform). Of course you would be right in drawing the conclusion that the web could deprecate like any platform. But, this is exactly what drives the PhoneGap team, they realise the limitation of the web platform and want to see that corrected. As such PhoneGap gets updates almost every two weeks to keep up with native platforms.

The console is your best friend

Just like with desktop development, PhoneGap can log variables and message to the console built into it, to help debug any issues during the app-building process.

PhoneGap's goals

Having those two beliefs made Niotobi have two high-level goals for PhoneGap. Firstly they wanted to see the web as a first-class development platform, which to put it bluntly, it isn't. It has opaque introspection tools, poor API surface, limited GUI elements and a sandbox that was missing many APIs to allow communication with hardware. So PhoneGap was set up to work specifically on the mobile platform and polyfill those gaps.

The second goal was to make the purpose of PhoneGap simply cease to exist! While this sounds like some kind of self-destructive journey into oblivion, it's actually a good thing. The team is committed to the standardisation of the web as a platform, so that eventually the gaps don't need to be plugged anymore. The team is influenced directly by what the W3C and WHATWG are doing, wanting to see standards take precedent.

Armed with these goals, the team first created PhoneGap back in 2008 for an iPhoneDevCamp event in San Francisco. PhoneGap later went on to win the People's Choice Award at O'Reilly Media's 2009 Web 2.0 Conference. At a time when other technologies were being banned from the iTunes store, PhoneGap had, and has, the approval of Apple.

What's in a name?

As PhoneGap gained attention it caught the eye of Adobe, who were not having so much success with cross-platform mobile publishing using Flash. In 2011 Adobe announced it had acquired Nitobi Software and simultaneously announced it would continue to remain an open source project as part of the Apache Software Foundation. Its name would be Apache Cordova and this has left some people wondering what the difference between Cordova and PhoneGap are. The easiest way to describe this is that Cordova is the technology that powers PhoneGap and anyone can use Cordova, as it's open source, to power their own software. This is exactly what has happened with Cordova powering Intel's XDK, Telerik Platform, Ionic framework, Asail's Monaca, IBM's MobileFirst platform and Convertigo. These companies can use Cordova and can also contribute to its open source project, so code contributions have been made from Adobe, IBM, Google, Microsoft, Intel, Blackberry and Mozilla. This way Cordova doesn't get pulled in as proprietary software, but the benefit shared around.

PhoneGap is like the Adobe brand for Apache Cordova. It's powered by Cordova and the tools that Adobe release are built with it, it's just that most people know it as PhoneGap. Just to complicate matters more, there is also something called PhoneGap Build. This is a cloud-based service run by Adobe that takes the code created by you on the desktop and builds it into a finished app ready for testing on your device or for submitting to the app store. In the early days of PhoneGap when this service didn't exist, the only way to build an iOS app was to have a Mac, similarly you would need Windows to publish an app to a Windows phone. By putting this in the cloud it saves the app creator from having to install the different development environments on their computer and anyone can publish to any mobile platform, regardless of computer choice. PhoneGap Build is offered as a free service that limits the developer to one private app

PhoneGap showcase

Look at these apps to see what phonegap is capable of

Mallzee
mallzee.com

Mallzee is a personal stylist in your pocket. It helps to pull in over 200 million products for people who love fashion but are too busy to spend time shopping. The style feed enables its users to get specific content that they maybe interested in. The app was featured on the iOS store, giving them great exposure.

ViceVersa
app.viceversa.io

This app is to help people make tricky decisions. For example, if you were thinking of spending a holiday in New York or San Francisco but are not sure which one to choose, then the app lets you poll other people to get a general consensus about which is the best option. You can set up some common categories to follow such as technology.

Untappd
untappd.com

Untappd is a social drinking experience. It shows you popular bars near to your location and what's available there. Your friends let you know recommendations for what and where to drink. In turn, you share where, and what you are currently drinking to help your friends. When you drink new beers you unlock new badges.

Pacifica
thinkpacifica.com

Pacifica is a tool to help calm your mood, relax your thoughts and drive away anxiety. It helps you manage your feelings so that you can cope with the everyday stresses of life. It provides the user with lasting treatment called cognitive behavioural therapy. The app has received rave reviews and been featured on a number of sites such as BuzzFeed and Upworthy.

SchoolLink
apple.co/1P6xSUW

This particular app was actually created specifically for the Surrey school district in Canada. SchoolLink was created to help keep parents in touch with the schools that matter the most to them by allowing them to subscribe to news, announcements, alerts and stories from those schools. If you're a parent yourself, you can appreciate this kind of service, and it sure makes it a whole lot easier to stay involved in your children's education experience. Alerts are pushed to the phone so information reaches parents immediately and they stay in the loop without missing a thing. Contact information is also available for each school followed, so users are never without the vital phone numbers and e-mail addresses of their children's teachers, school nurses and principals. A great example of an app providing a vital service.

and unlimited public source code apps, hosted on GitHub. If you need a number of private apps then there is a paid service. The cloud build takes a few minutes to compile the app and download to either the user's device or computer, and it simplifies the process of installing and setting up the specific SDKs.

There's a plugin for that

As PhoneGap developed it decided to split all of the device native features of the framework into plugins. The reason for this is because they don't want to have such a large codebase for the different platforms such as Android, iOS, Windows and so on. By moving these into plugins it stops unnecessary duplication of code and centralises their JavaScript writing efforts.

Third-party plugins have always been strongly supported with PhoneGap and if you feel inclined, Holly Schinsky, an Adobe web development evangelist, has produced a guide to creating your own plugins on her blog (**bit.ly/21UnF4W**).

At this point you may be ready to start browsing the third-party plugins available for PhoneGap. There are several sites that can help you with this, plugreg (**plugreg.com/plugins**) has a list of all plugins and state that there are currently 1,570, however the Cordova (**cordova.apache.org/plugins**) plugin registry puts this number at almost half that, with the smaller figure of 792. Finally if you are using PhoneGap Build to build out the final app then you need to make sure the build process can support your plugin, find out more at **build.phonegap.com/plugins**.

The kind of plugins that are available can offer your app a great deal of extra functionality, for example if you need a barcode or QR code scanner there's a plugin for that. Similarly push notifactions, geofencing or the use of real device transitions rather than CSS emulations, then these are all covered and enable you to elevate your PhoneGap app beyond what can be done with native web technology, plugging any gaps that can occur between your app and the device.

Turning it up to 11

When you start building your app, whether you choose to use PhoneGap CLI or the PhoneGap Desktop app, you will get a basic app setup, ready for testing. All this app does is register a message on the phone screen that the device is ready.

At this point you are ready to create your own app. You will need to either choose a mobile UI framework which we've picked the best out for you or create your own, which actually is surprisingly easy depending on what your apps goal is. Remember even though you use HTML, CSS and JS, you must build an app that warrants being installed on the device. To do this, have an idea that needs someone to come back to it again and again, have it perform a task that is otherwise complicated to do, like avoiding traffic jams. As the user is on a small screen they are likely to be out and about somewhere, so think about what else that user might be doing while they are in the throws of navigating your app. If you are new to making apps then it's a good idea to have one very clear goal for your app. Make it do that one thing really well, for example if you are at a beer festival, the app could guide you to the nearest toilet facility, or better still get you home safely on public transport. Once you have the app working it is always much easier to add functionality to an app that has a base of users than offer a poorly thought through jack-of-all-trades app.

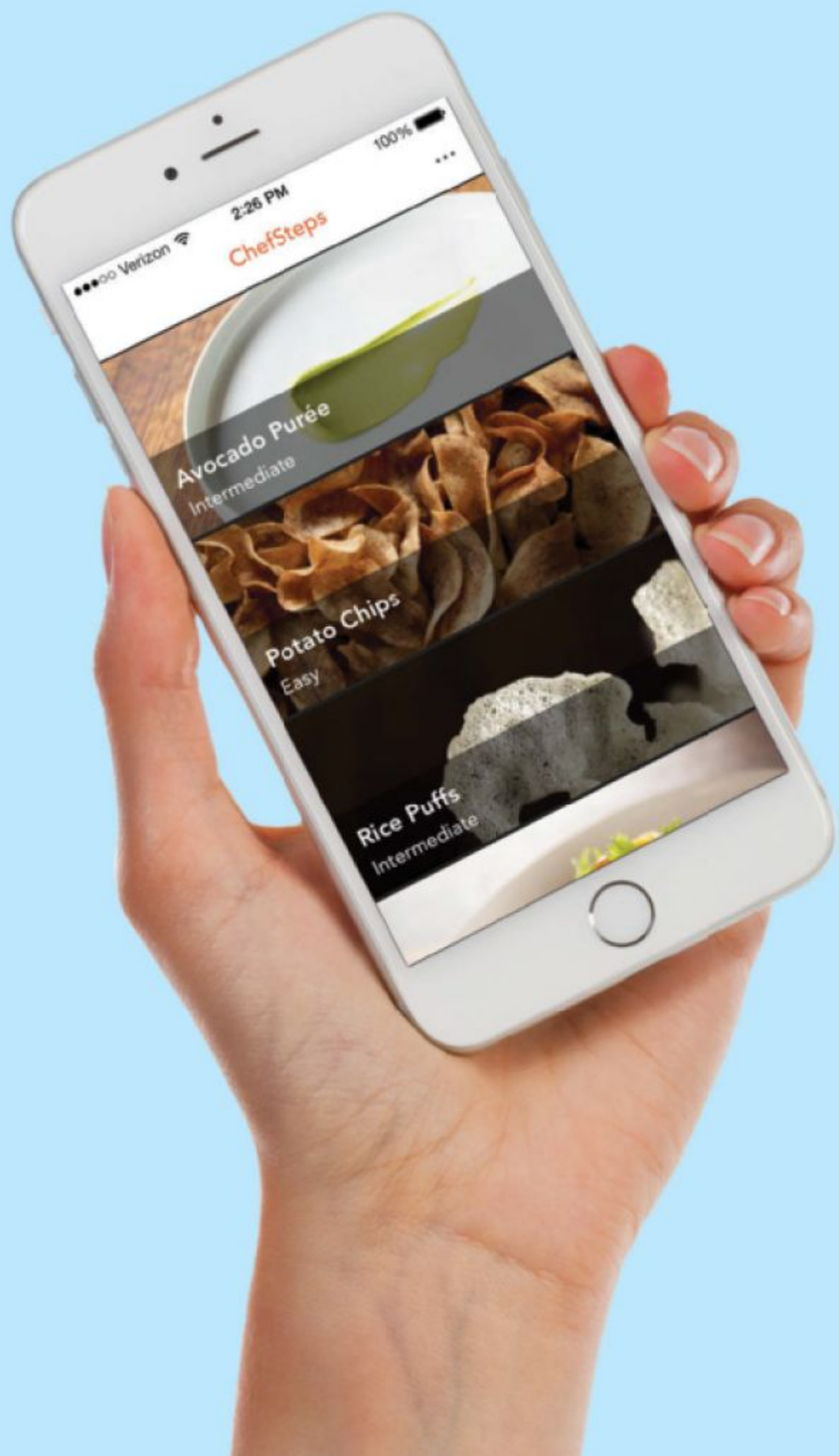
Chrome debugging

You can make use of Chrome's Timeline and Profiles panel in the Developer Tools to check your CPU, memory usage and execution time for making sure you have optimal code.

Try to break it

A final piece of advice would be for you to test it to destruction. From the very beginning on paper, test your idea out, give it to people who might use it, ask them what they think or what they would do with it.

Once you start writing code, test it on as many different devices as you can, especially old phones. Don't just test with the excellent PhoneGap mobile app as that isn't a finished build of your app, it's almost an emulator for PhoneGap – instead make regular builds and test these thoroughly. If at any point you are not using something that requires touch input, test in a web browser until you get it working then move it to the phone. Remember PhoneGap has done a lot of the technically hard work, the creativity is up to you.



Best UI frameworks

Working with PhoneGap you will need a UI framework to style up your app. check out these essential resources

Ionic Framework
ionicframework.com

Ionic offers mobile-optimised components, gestures and tools for building graphically rich apps and comes with its own prototyping tool and desktop app for building interfaces.

Sencha Touch
sencha.com/products/touch/#overview

Sencha Touch is an MVC-based framework built on Sencha's web app technology ExtJS. It lets you create native-looking HTML5 apps, and has UI widgets, layouts, animations, back-end data packages and charting packages, but it has a steep learning curve.

Foundation for apps
foundation.zurb.com/apps.html

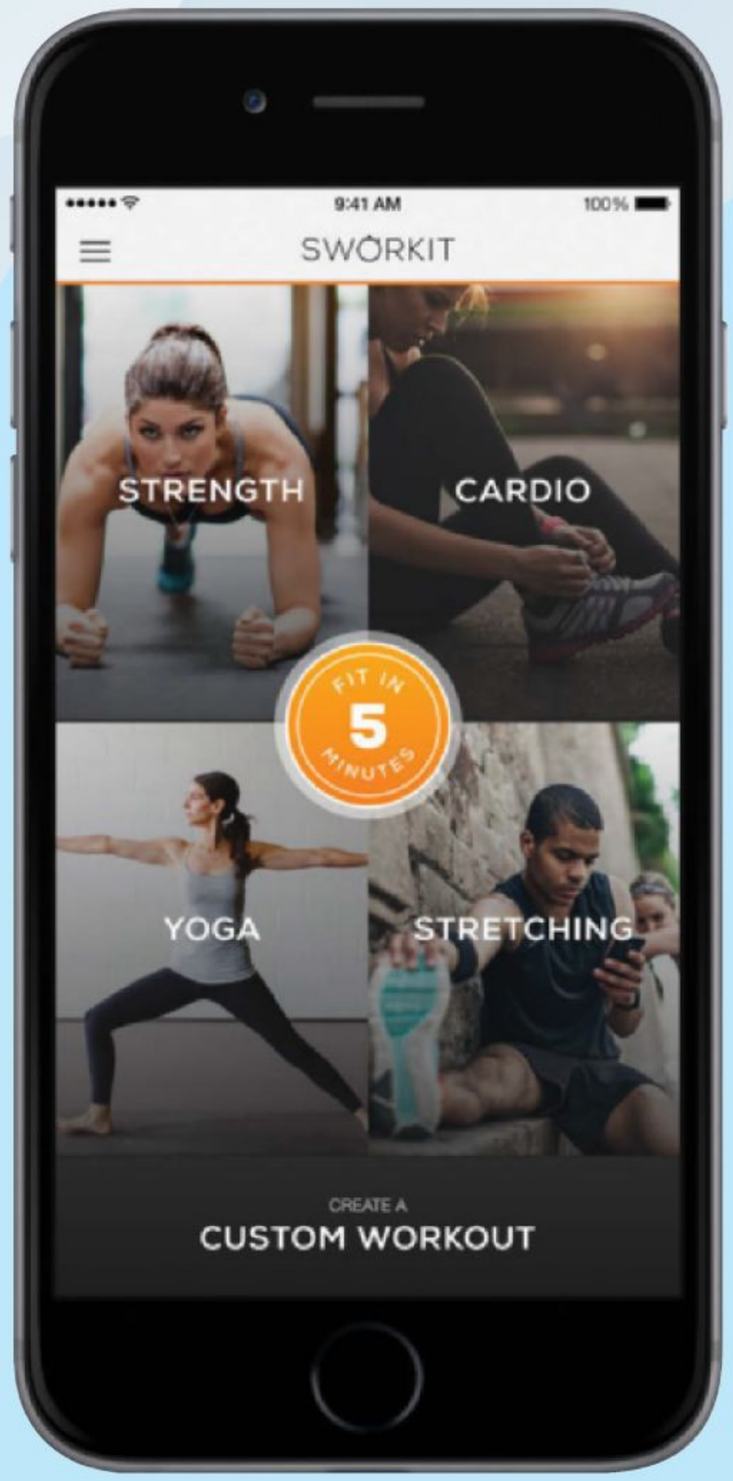
Foundation is probably best known for its responsive web framework. Now they've created a version for apps built on top of AngularJS with UI components and motion. A designer can build the UI for Angular apps, without knowing any Angular beforehand.

jQuery Mobile
jquerymobile.com

Probably the best known due to the ubiquity of jQuery. The mobile version offers a touch-optimised framework that allows for UI elements, navigation and motion. It can easily be branded using ThemeRoller for jQuery Mobile.

Ten million downloads... and Sworkit is not even a native app

PhoneGap apps do not have to be considered poor relations to native apps, as the Sworkit team will testify



PhoneGap gives Sworkit the vital competitive advantage to build and deploy quickly with just a small team of developers. Thanks to cross-platform development, they've been able to create nine apps that provide specific workouts to a wide audience. Sworkit has had over ten million downloads and still continues to grow.

The Sworkit apps are now a complete fitness platform that was built by just three full-time engineers. As mobile devices and mobile operating systems become even more performant, and with frameworks like Ionic providing the UI, the team no longer see hybrid development as a lesser alternative to native development.

Instead, they look to make use of great advantages such as their single codebase, incredible web stacks and APIs, to immediately deploy new updates to app stores without new submissions. What's more, they have the entire web development community as a hiring pool.

Must-know resources

Learn from others to make your app idea a reality

Devgirl's Weblog
devgirl.org
Holly Schinsky is a developer evangelist for Adobe who is currently focused on mobile development using PhoneGap, HTML/CSS and JavaScript. She has over 16 years experience in software development and her blog is filled with tutorials and announcements.

Brian LeRoux on Twitter
twitter.com/brianleroux
Brian LeRoux is a member of the PhoneGap team and he actually wrote the very first DOM UI library for mobile JavaScript, called XUI. He doesn't have time to work on that now as he is too busy working on all things PhoneGap!

Shazron's Cordova Blog
shazronatadobe.wordpress.com
Shazron is a developer working on Apache Cordova and has an incredible amount of resources for the more development-minded reader. There are a number of posts on why certain things don't work, and it's interesting to read how these solutions are added to PhoneGap.

NC Tech-lutions
bit.ly/1Q3D7rG
This YouTube channel has around 20 tutorials here that will help you go from being a complete newbie with PhoneGap to developing a complete app. The videos are fairly small so you get something to take away without having to watch hours of footage.

Submit an app

Using the cloud-based service PhoneGap Build can easily build your very own app in a variety of different formats to submit to either Apple or Andriod app stores. However, you will need to first become an officially registered developer. For iOS, for example, you must create a provisioning profile for the app. This is then uploaded to PhoneGap Build so that it can successfully build your app. Then you can submit your app for approval, and once approved the app will be able to go live. Remember, it's a good idea to read the developer's guides for Android (**bit.ly/1MA7bZ7**) and iOS (**apple.co/21S0hoD**) before starting any design ideas so that you stay within their guidelines, and don't waste any time and effort.

5 common mistakes to avoid

Whether it's your first time using PhoneGap or you're an experienced web designer, you need to avoid these pitfalls

Antivirus, antiapp

This is a problem mainly for Windows users, but if you are running antivrus software it could be blocking PhoneGap on your computer from talking to the Phone app. This is a relatively easy fix to make, but without thinking of this, you could waste hours searching for a solution in forums. Go into your antivirus software and make sure you grant access for PhoneGap to communicate. Once this is done it should just require a refresh on the phone to get connected.

Blocked ports

This isn't something that affects the average home user, but can definitely be a problem if you are using PhoneGap when you have an IT department, which is usually in a professional office setup. The IT specialists job is to close security loop holes, so they may well have blocked certain ports from having access. This causes a problem for PhoneGap as it defaults to using Port 3000. Talk through with your IT department to find a suitable port that is unblocked, or maybe they could unblock 3000 for you. It's easy to change ports in PhoneGap – click on the settings icon, change the port number and hit save.

You are not building a website

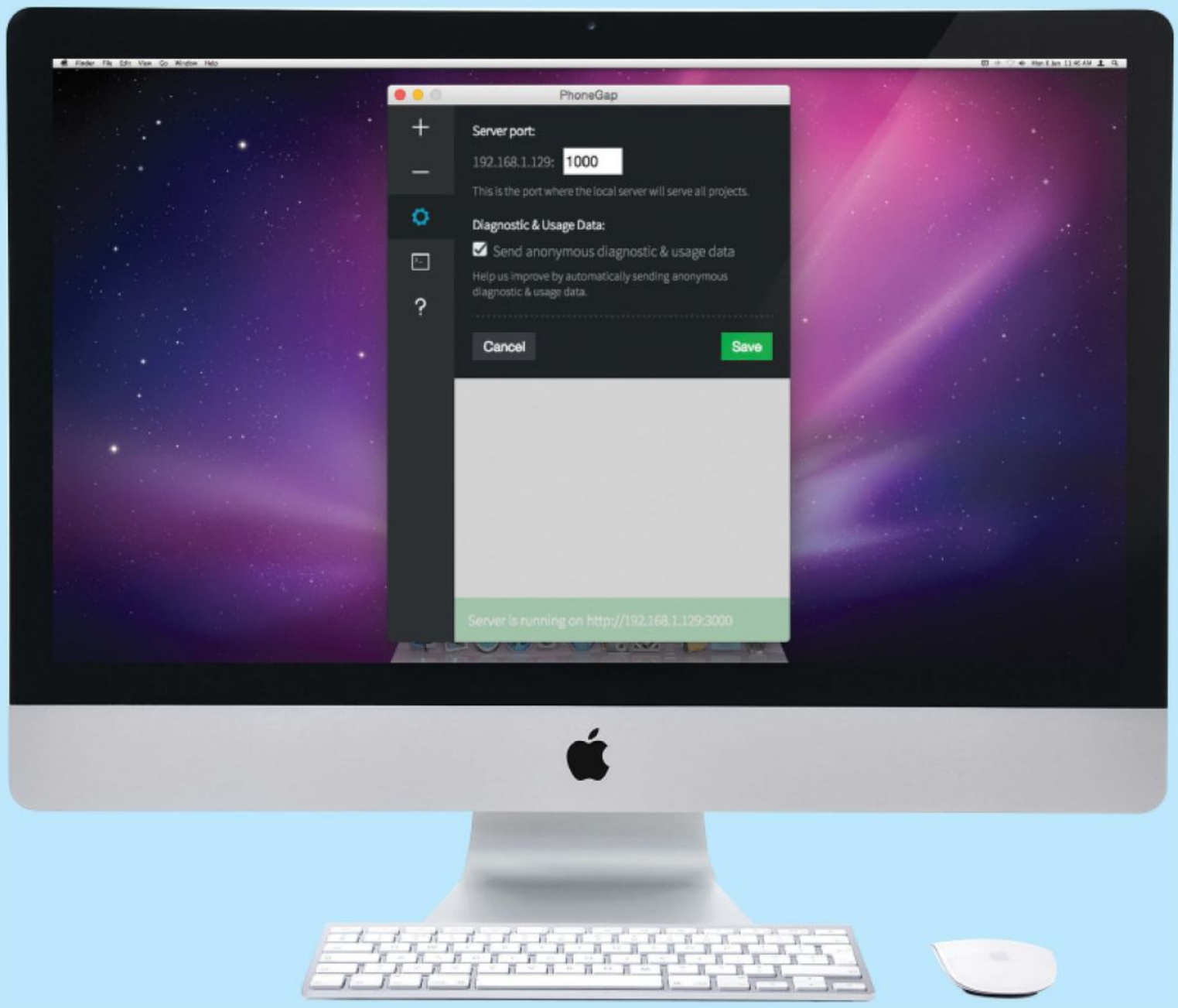
Just because you are using web technologies, it doesn't mean you are building a website. It's amazing how many first-time users make this mistake. You are building an app, give it app-like functionality and a direct purpose for using it. Tap into the hardware of the device, like using the camera or accelerometer, to make your app have specific functionality that makes it useful for the phone. According to IBM, 80 per cent of apps are uninstalled after the first use, make sure yours is not one of them.

Graphical goodness

There is no WebGL support with PhoneGap. PhoneGap uses the native browsers capabilities, which is a cut-down browser as part of the OS that becomes available to apps. It doesn't have the speed for WebGL, so don't use it. There are a number of plugins that force the browser to use the better web engine, but obviously you need to test these thoroughly on a variety of devices before embarking on integrating them in your app. HTML5 Canvas is fine but it tends to run slower than running the stand alone Chrome app.

Build specifically for each device

Just because you can build an app for six different platforms from one codebase with PhoneGap Build doesn't mean you should. Build each UI specifically for the mobile platform, using specific navigation features of that platform as the mental model for your users. Some iOS users will expect tabs in a different place on the screen to Android users. Better still, build your own unique looking UI that can be easily used and is device agnostic. This way there will be less tweaking to do as you move from one platform to another.



DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/bks-894

Desktop apps with NW.js and JavaScript

Use NW.js to create JavaScript-powered desktop apps that can be run on OS X, Linux and Windows

ver the last 25 years, the browser has opened new windows (see what we did there) on content and code distribution. Whole ecosystems have sprung up to support the creation and deployment of apps that can be accessed with a simple URL. Despite how great the browser has become at integrating itself into operating systems, we still sometimes find that we want an app that we can install on our system and run at any time with just that extra little bit of power.

What we need is something with the low-level access of Node with the ease of development of browsers and a way to package it all up as an app that can be run on our traditional OS like any regular, native app. That's what NW.js (formally known as Node-WebKit) is. NW.js bundles io.js and the WebKit browser engine together in a way that lets us access the full power of Node through the DOM. This then enables us to write web apps and also to run them in self-contained windows that look almost exactly like OS native apps.

In this tutorial, we're going to set up NW.js and create a simple to-do app to show how simple it is to write native apps with JavaScript and NW.js. We will be doing this for Mac OS X.

1. Get NW.js

Installing NW.js is pretty simple, all we need to do is head on over to GitHub (**github.com/nwjs/nw.js**) and grab the build for your OS. If we were so inclined we could build NW.js from scratch, and that may be advisable if you want to be certain of stability, but for learning, the prebuilt binaries will do. For this tutorial the 32-bit release of NW.js 0.12.1 was used and tested on OS X 10.10.3.

2. Relocating the NW.js executable

Unzip the package we just downloaded. It contains the prebuilt NW.js executable that we can use to run our apps. When we get to testing and deploying our apps, we'll use the nwjs app bundle included in this folder, so move the entire folder to a safe, memorable place.

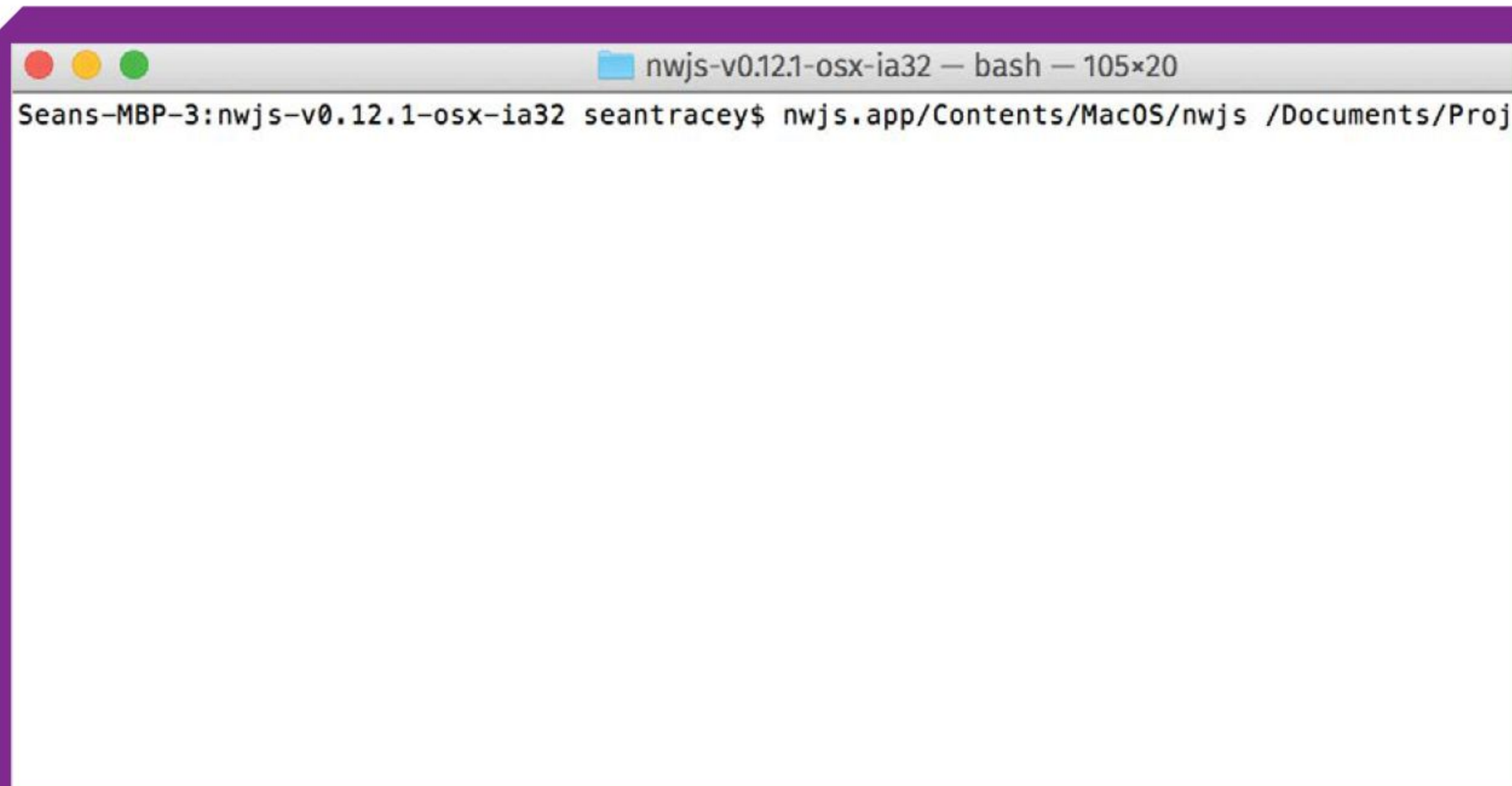
3. Create a NW.js project

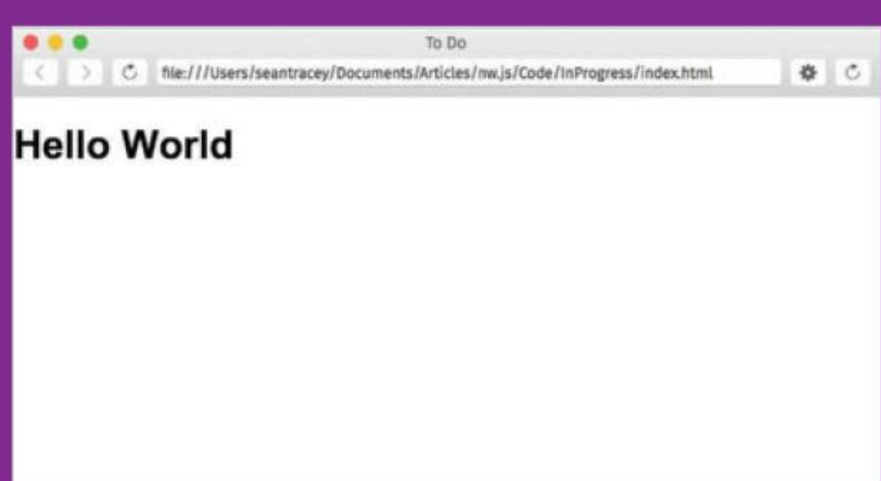
NW.js apps are supersimple with few prerequisite dependencies, so there's no need for a project creation tool like those found in, say, Cordova. Create a new folder somewhere on your system and create a HTML file called 'index.html' and insert the code below. This will make up the UI of our app and load the NW.js code to handle the creating of our window.

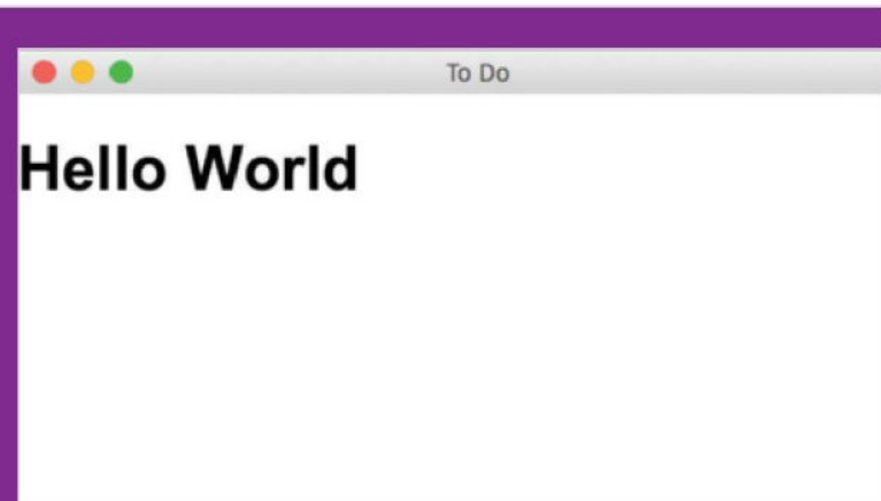
```
<!DOCTYPE html>
<html>
<head>
<meta content="text/html; charset=utf-8"
http-equiv="Content-Type">
<title>To Do</title>
<link rel="shortcut icon" href="favicon.
ico">
<link rel="stylesheet" href="styles.css"
type="text/css" />
<meta name="viewport" content="initial-
scale=1.0, user-scalable=no" />
</head>
<body>
```

Compiling code

There is no compiling per se in NW.js, which means that you may want to keep your source code from prying eyes. There are ways to make it quite hard to get at your source, documented on NW.js' GitHub wiki (**github.com/nwjs/nw.js/wiki**).







<Top left>
• We don't want to have to bundle things every time we want to test our app. Instead, we can point the NW.js binary to our project and run it in terminal

<Bottom left>
• When we first run our app, it looks a lot like a browser. We will need to tweak the manifest file

<Bottom right>
• Once we've adjusted our manifest properly, our app starts to look a lot more like it belongs on our system

```
<h1>Hello World</h1>
</body>
</html>
```

4. Create the manifest

Just like an io.js app, NW.js uses a JSON manifest to get the wheels rolling with our app. Create a file called 'package.json' in the same folder as your index.html. This is the root of your NW.js, all URIs are relative to the package.json file.

5. A simple manifest

Open up your new manifest file. The simplest manifest requires two properties, a main and a title. The main is the file that our app will launch when it's first run, so in our case it's index.html. There are other properties that can affect how our app runs, but for now add this::

```
{
"title" : "To Do Today",
"main" : "index.html"
}
```

6. Point the app

When we're testing out our app, we shouldn't have to package and repackage it every time we correct code. We can use the NW.js executable to run our app by pointing it to the root of our project. Open up a terminal and navigate to the folder downloaded from GitHub.

7. Launch our app

The NW.js executable is buried in the NW.js app in our folder. The executable is the program where the io.js and NW.js code lives. With NW.js, we don't change the app, we create projects that run with it. To run the app, enter:

```
$ nwjs.app/Contents/MacOS/nwjs /path/to/
your/project/root.
```

8. First impressions

A new window will have opened and a NW.js icon will have appeared in your dock. We're running our first NW.js app, and we can see our HTML file – but it still looks like a browser window. To make our app look a little more like it belongs, we can remove certain elements with our package.json manifest.

9. Enhance our manifest

We've got some system chrome we need to get rid of now! Just after the 'main' property of your manifest, add a 'window' object and add the following properties to it:

```
{
"title" : "To Do Today",
```

> To make our app look a little more like it belong, we can remove certain elements with out pack.json manifest

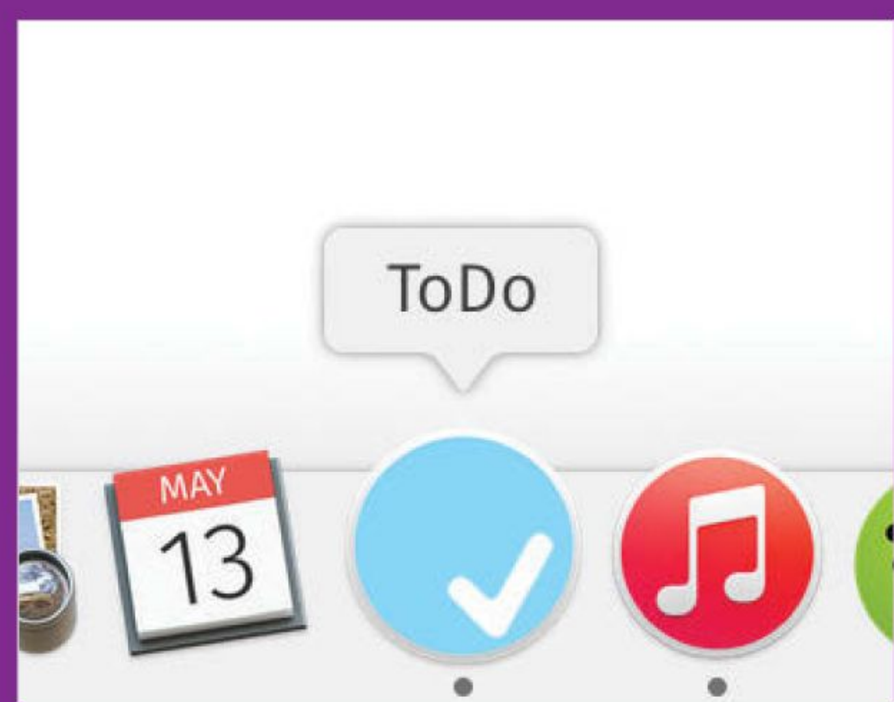
Node-webkit

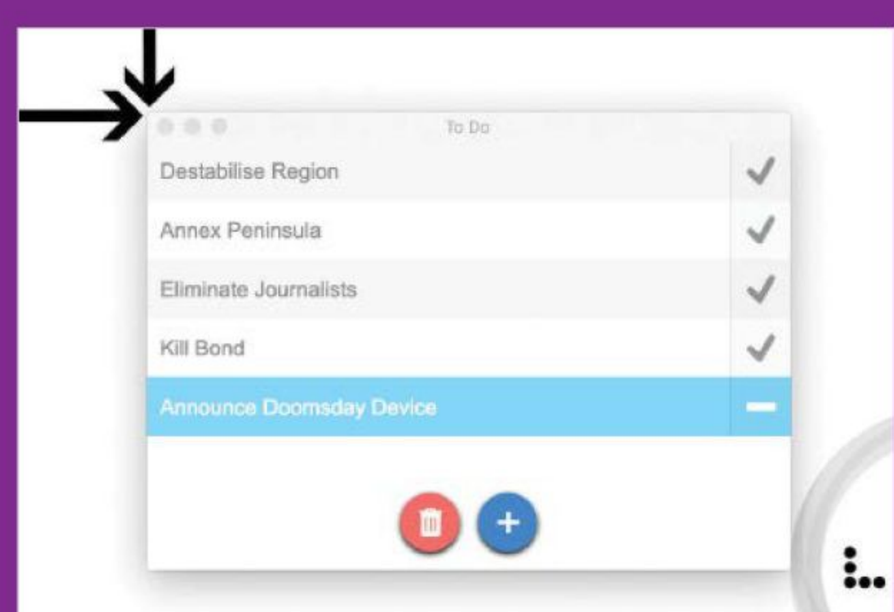
NW.js is, in essence, Node WebKit. The name change accompanied the migration of the project from Node.js to io.js. Rumour is, all the cool kids call it n-dubz for short – enn-double-you-dot-jay-ess is such a mouthful.

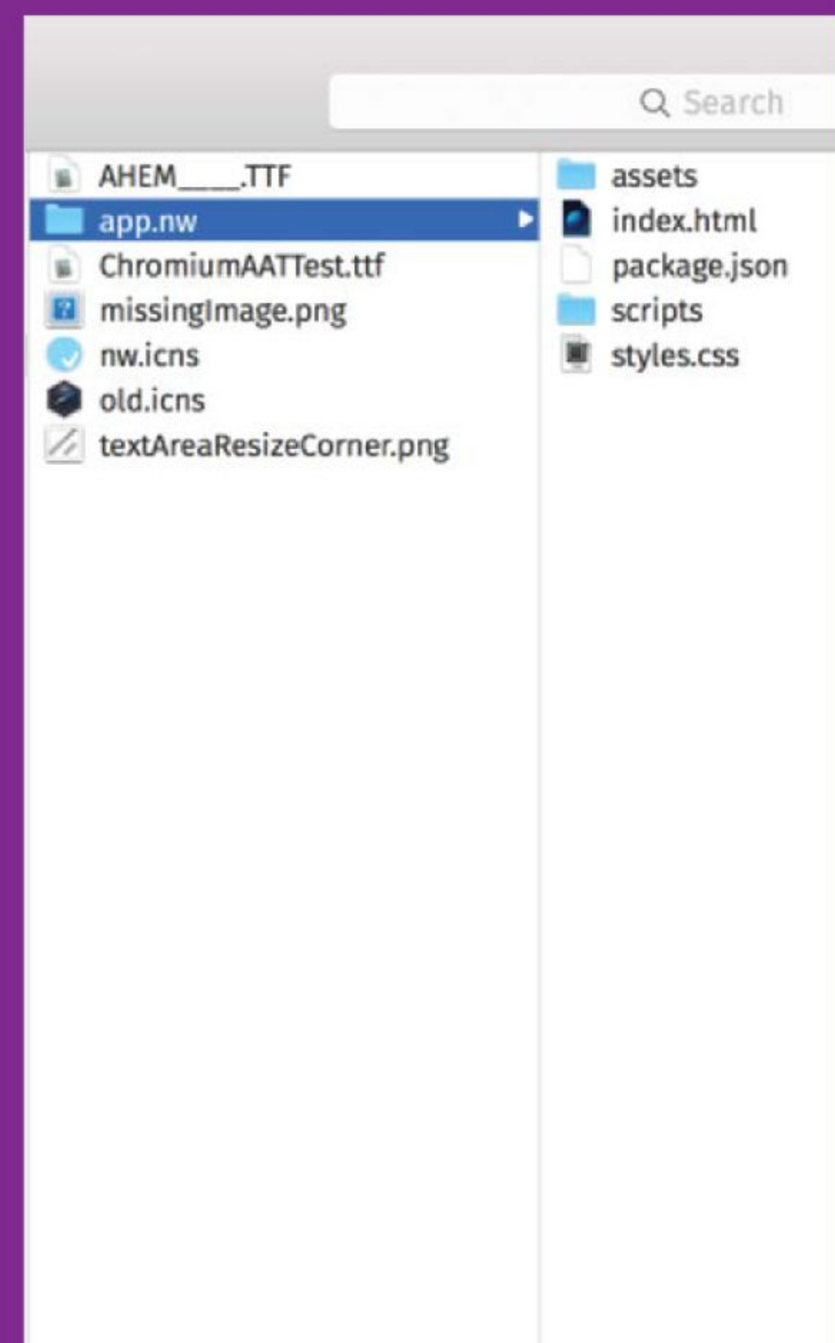
<Top left>
• Making ICNS files is a bit of a pain, fortunately Image2icon has us covered, but only if you distribute your app on OS X

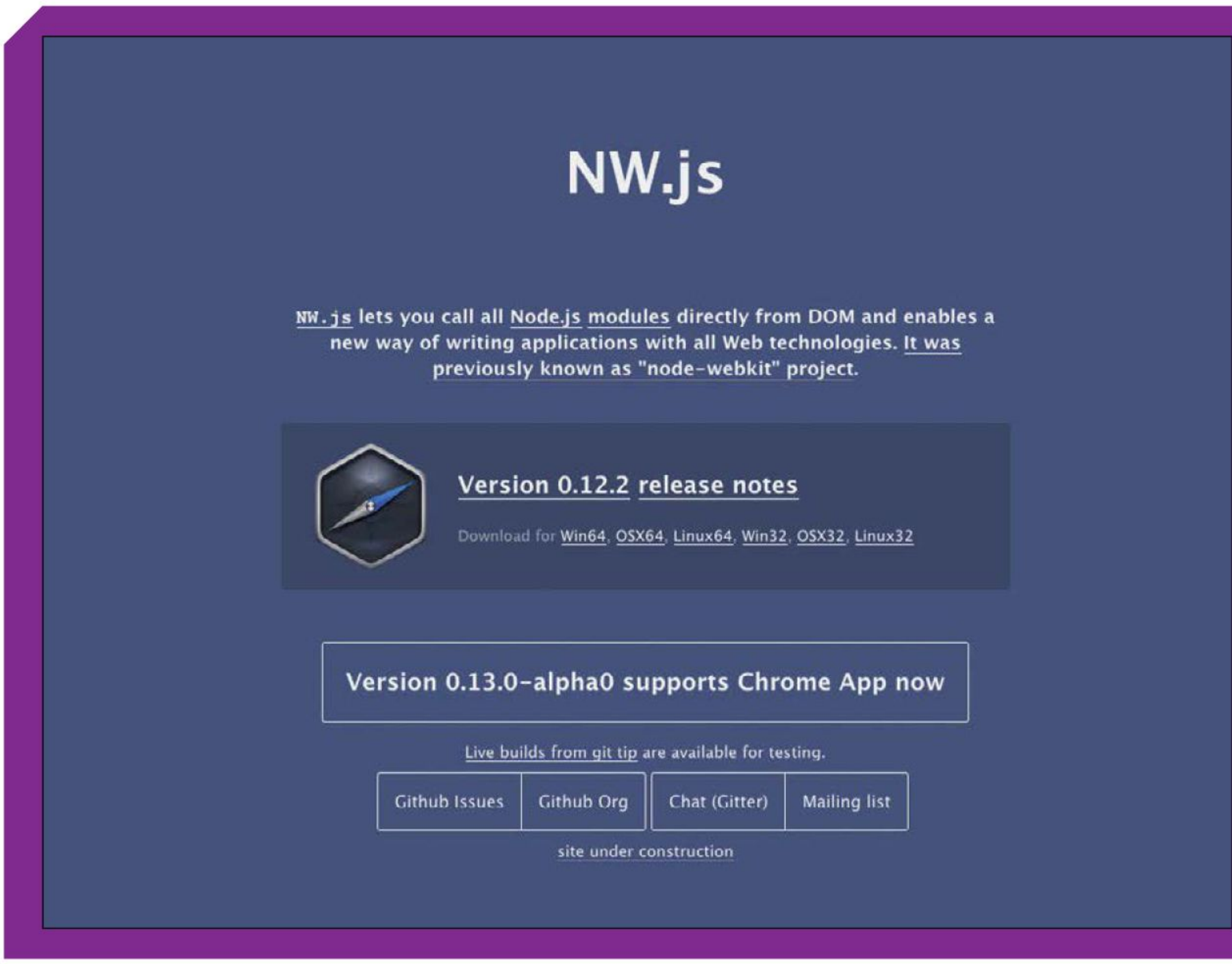
<Bottom left>
• When we position our window with code, the top-left corner is positioned relative to the top and left

<Right>
• We need to navigate the hierarchy to package our NW.js app properly









Node modules

NW.js is, in part, io.js. That means that you can use modules! Well, at least in principle you can anyway. NW.js can run on any of the major platforms, but just like a native app, your packaging process will of course vary depending on the operating system that you're actually targeting. You can't run an EXE on a Mac, just like you can't mount a DMG on Windows for example. If you want to make use of io.js modules that use a language other than common JavaScript, then it's very likely that you will have to compile your package on the system that you're targeting. That being said though, if you're only using Node modules written in common JavaScript (like the fantastic moment.js module, for example), they should work straight out of the box.

```
"main" :
"index.html"
"window" : {
"toolbar" : false,
"frame" : true,
"width" : 480,
"height" : 320,
"show" : true,
"resizable" :
true
}
}
```

The window object affects how our window behaves. We can remove the frame, define height, make it persistent in all workspaces, but for now these properties set us up.

10. Create our GUI

If you repeat Step 7 as our app launches, we no longer have a URL bar and our window is the width and height we defined. All that's left of the system chrome are the close, minimise, maximise and title. We can remove these, but they're handy for UX purposes, so leave them.

11. Add scripts

It's time to start putting the component parts of our GUI together. Open up index.html and amend it like this:

```
<!DOCTYPE html>
<html>
<head>
<meta content="text/html; charset=utf-8"
http-equiv="Content-Type">
<title>To Do</title>
<link rel="shortcut icon" href="favicon.
ico">
<link rel="stylesheet" href="styles.css"
type="text/css" />
<meta name="viewport" content="initial-
scale=1.0, user-scalable=no" />
</head>
<body>
<ol id="list">
</ol>
<div id="UI">
<div id="delete" class="button">
</div>
<div id="add" class="button">
</div>
</div>
<script src="scripts/core.js"></script>
<script src="scripts/window.js"></script>
</body>
</html>
```

This is the markup for our list. The app logic isn't what we're focusing on, so we won't write core.js or styles.css from scratch, you can download them from FileSilo.

12. Access NW.js globals

We'll now write window.js to ease into NW.js coding. Create a folder called 'scripts' in your root folder and a file called 'window.js' in it. Here, we're going to affect the app window with events and listeners for that native feel.

13. NW.js GUI and windows

Just like a Node app, we can use require() to access node

> The app logic isn't what we're focusing on, so don't core.js or styles.css from scratch - download them from FileSilo

modules and NW.js globals. In order to manipulate the properties of the window and listen to events, we need to require the GUI module from NW.js and abstract the window away from it. Insert the following into window.js:

```
var gui = require('nw.gui'),
win = gui.Window.get();
```

14. The menu bar

NW.js is platform agnostic, but there are small OS tweaks we can make for each platform that make the app look more at home. OS X for example has the menu bar with the About and Edit dialogs that are currently missing. Adding the following to window.js will adjust our to-do apps menu bar with more relevant information.

```
var nativeMenuBar = new gui.Menu({ type:
"menubar" });
nativeMenuBar.createMacBuiltin("ToDo", {
// hideEdit: true,
hideWindow: true
});
```

15. Hide our window

When our app loads, we see a quick flash of white before the content of our page is loaded and initialised. In package.json change the value of 'show' to false instead of true. In window.js add win.show(); to the end of the file. Now when our app loads, the scripts will be initialised and will only show once scripts have finished loading.

16. Position the window

When the app is ready we can position the window. When we open a native app on OS X it's generally at the same x and y coordinates as when you closed it. We can reproduce this, just insert the following before win.show():

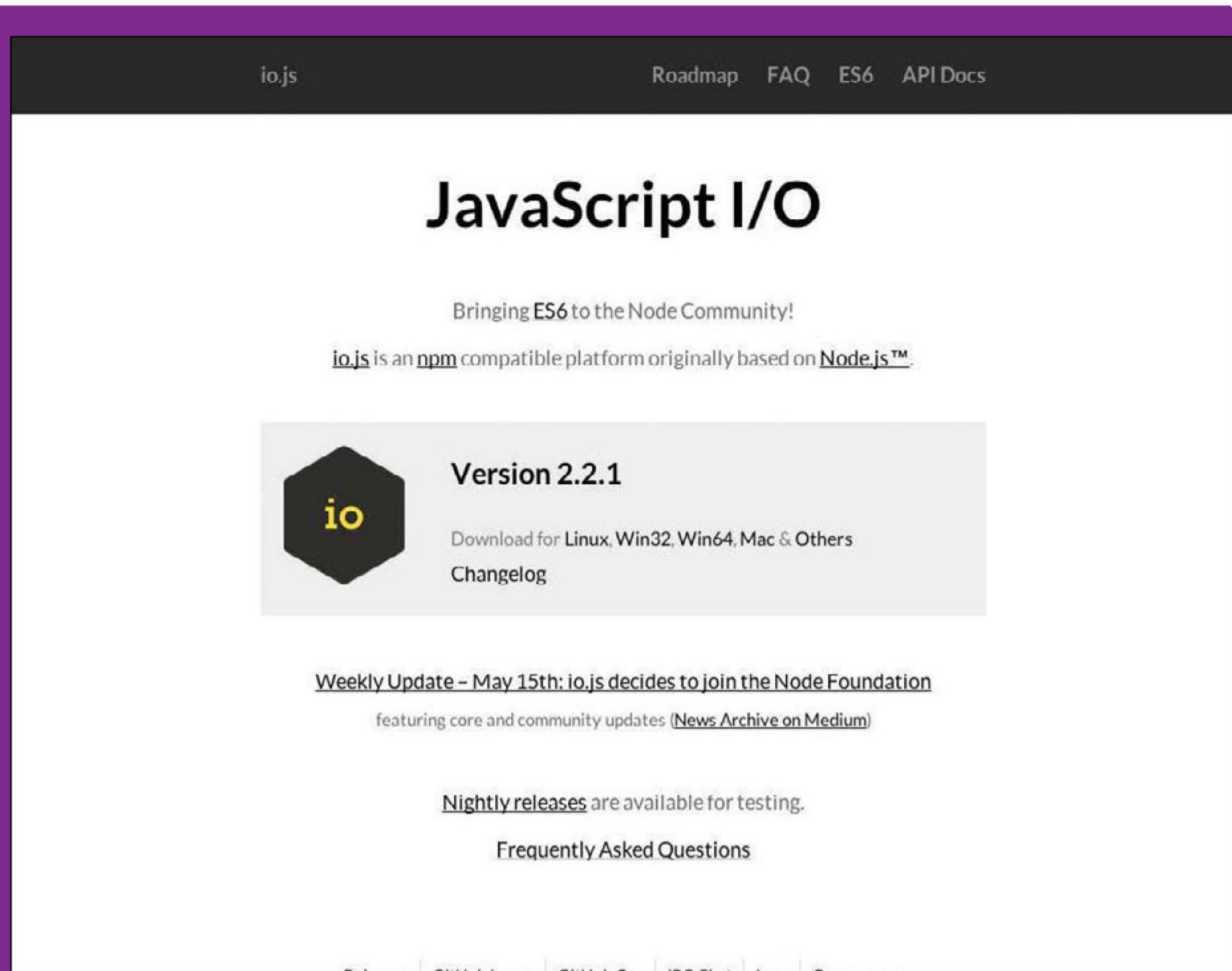
```
var initialPosition = {
top : 50,
left : 50
};
var savedPosition = JSON.
parse(localStorage.
getItem('windowposition'));
win.on('move', function(){
localStorage.setItem('windowposition',
JSON.stringify({left : win.x, top:
win.y}));
});
if(savedPosition !== null){
win.moveTo(savedPosition.left,
savedPosition.top);
} else {
win.moveTo(initialPosition.left,
initialPosition.top);
}
```

On our win object, which is the reference to the NW.js window, we have successfully added an event listener. Every time our user moves their window, the 'move' event is dispatched to win. When this happens, our callback is called and we can save the coordinates that the window has been moved to in localStorage. Now, whenever we move our window, its coordinates will be

> The app logic isn't what we're focusing on, so we won't write core.js or styles.css – you can download them from FileSilo

Node.js and io.js, What's going on?

Node is backed by a company called Joyent – this has been great for many reasons, but a lot of people have felt that the speed at which new JavaScript features and optimisations are being integrated into Node is too slow. These feelings led to io.js – a fork of the Node source code, but maintained entirely by the community. Io.js started to implement new features (like ES6 syntax) straight away, and with the benefits of a rapid-release cycle being well known to the developer community, many projects, like NW.js, migrated to io.js. Now, Joyent and the guys in charge of io.js have agreed to merge all of the changes in io.js back into Node.js and release and develop Node under the newly created Node Foundation.



saved in localStorage and the window will be restored to its previous position.

17. Negative numbers

If our user in question happens to have multiple screens and our app is positioned on one of them, when the screen is unplugged our app could end up being stuck where we can't see it, which is not what you want! If our stored coordinates are less than 0, we're going to override the coordinates and place our window back on the main screen. Just after if(savedPostion !== null){, add the following:

```
if(savedPosition !== null){
if(savedPosition.left < 0){
savedPosition.left = 50;
}
if(savedPosition.top < 0){
savedPosition.top = 50;
}
```

18. Resize the window

Just as our window is always in the exact same place that we left it, it should also be the same size right? Well, that is what we will be moving on to now. We are going to repeat what we've done previously with the move event and localStorage, but this time we're going to listen for the resize event instead. When our window is resized, we'll save the dimensions to localStorage. When our restart our app, we'll set the dimensions of our app to the dimensions stored. If successful, our window will be the same size as we left it every time.

```
var savedPosition = JSON.
parse(localStorage.
getItem('windowposition')),
savedSize = JSON.parse(localStorage.
getItem('windowsize'));
win.on('move', function(){
localStorage.setItem('windowposition',
JSON.stringify( {left : win.x, top: win.y
} ) );
});
win.on('resize', function(width, height){
localStorage.setItem('windowsize', JSON.
stringify( {width : width, height :
height} ) );
});
if(savedPosition !== null){
if(savedPosition.left < 0){
savedPosition.left = 50;
}
if(savedPosition.top < 0){
savedPosition.top = 50;
}
win.moveTo(savedPosition.left,
savedPosition.top);
} else {        win.moveTo(initialPosition.
left, initialPosition.top);
}
if(savedSize !== null){
win.resizeTo(savedSize.width, savedSize.
height);
}
```

19. Click to lock

At the start of this tutorial, you should have downloaded core.js from **filesilo.co.uk/bks-894** and then included it in our scripts folder. Now, we are going to continue by adding a little bit of code to it. In our app, you should see an unlocked padlock in the bottom-right corner of our window, it doesn't do anything at all right now, but when we're finished, it will be a toggle for locking our ToDo app on top of all other windows so that it is visible across all of our workspaces.

20. Create the ICNS file

Open up core.js for editing and scroll down until you find the addEvents function and just before the end of that function, add the following:

```
document.getElementById('lock').
addEventListener('click', function(){
var isLocked = this.getAttribute('data-
window-is-locked');
if(isLocked === "false"){
win.setAlwaysOnTop(true);
win.setVisibleOnAllWorkspaces(true);
      this.setAttribute('data-window-is-
locked', "true");        } else {
win.setAlwaysOnTop(false);
win.setVisibleOnAllWorkspaces(false);
this.setAttribute('data-window-is-locked',
"false");
}
}, false);
```

Now, if we restart our app and click the padlock icon, it will lock and if you click another window or move to another workspace, ToDo will still be on top for us to see. If we click the padlock again, our window will stay on the workspace it's currently on and will disappear behind any other window that has focus.

21. What did we just do?

Our package.json file sets up how our window looks and behaves when we first load it, but as we've seen with the moving and resizing code we wrote a little earlier, we can override these properities with JavaScript if we want to, even after the window has been shown. Our win variable is in the global scope, so it is still possible for us to access it from our app and use DOM events to change its properties – even though it's technically Node.js property. Then when we click on our padlock, we will work on checking the window-is-locked data attribute on the DOM element to get the window's current state. If our window isn't locked, then we can now use 'win.setAlwaysOnTop(true)' and 'win.setVisibleOnAllWorkspaces(true)' to lock it and vice versa.

22. Package our app

If you run the app now, you'll have a fully functioning, persistent to-do list. You can minimise it, close it and resize it. Now we need to package it. Go to the folder you downloaded earlier that was relocated to GitHub. Copy the NWJS.app file into the root of your project folder and right-click it. Select 'Show Package Contents' and navigate to Contents>Resources.

23. Copy files

Open another Finder window and copy all of the files we've created (index.html, styles.css, scripts folder, package.json and all the others) into the newly created app.nw folder. Now if you got back to your project root and double-click NWJS, your app will be run by default.

24. Edit PLIST

Our app still looks like a NW.js app. To affect some changes like the name and the icon, we need to change the app's PLIST file. Right-click NWJS again and select 'Show Package Contents' and then navigate back to Contents. Open the PLIST file. If you have Xcode, it will open its editor, otherwise you can modify it with any text editor. Change the 'Bundle name' and 'bundle display name' to 'ToDo' and save.

25. Create ICNS

The last touch to make any app look at home is the icon it uses. You can either create your own using something like Image2icon or you can use the ICNS file included in our files – these are available for download from FileSilo. Just simply copy this into Content>Resources.

26. Affect the changes

The icon and name changes may not take effect straight away. You can relaunch Finder or change the name of the .app in your project root to anything and then back to 'ToDo'. Then the icon and bundle name should update.

27. Distribute our app

We now have a fully functioning, JavaScript-powered OS X app. You can now move the ToDo.app file anywhere on yours or anyone else's system and fire it up, just like a regular app. When running your app on other people's system, they may need to tweak their security settings to enable your app to run. To do so, simply open System Preferences>Security and Privacy and enable apps to be run from the Mac App Store with the toggle. Now, move ToDo into your applications folder and give it a whirl.

> When running your app on other people's system, they may need to tweak their security settings

DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/bks-894

Create a unique mobile 3D interface

Part 1

Build an augmented menu that hooks into a mobile's device orientation controls to create a bespoke browsing experience with three.js

Creating compelling web experiences for mobile devices has come a long way, there are a whole raft of phone functions that designers can plug in to and access without the need for packaging up the mobile site as an app. In this tutorial what is being achieved with the native mobile web would have been pretty hard to conceive as an app going back just a few years, but now this is running purely as a mobile optimised website.

The site itself will be in 3D and rendered by three.js but instead of this being rendered with the WebGL renderer, the scene will be rendered using the CSS3 renderer, giving us maximum compatibility with devices. Having 3D as part of the design is going to be an important part of the interface. In order to browse around the interface the user will need to rotate their phone and the display will update, placing them in the centre of an augmented interface. Moving the phone around becomes like moving a camera in the interface to explore the links that are on offer to the user. This is made possible through the 'Device Orientation Controls' that come bundled with three.js and can turn our phone into something similar to an Oculus Rift.

This is part one of a two-part tutorial, creating the interface and get the device orientation working, we will focus on the interaction with the menu on page 134.

1. Link the libraries

Open the 'start' folder in a code editor such as Brackets. In the body of the document, add the following code to link up the libraries. This uses the three.js library available from **threejs.org**. Following this is an opening script tag with some global variables that are needed in the project.

```
<script src="js/three.min.js"></script>
<script src="js/DeviceOrientationControls.
js"></script>
<script src="js/CSS3DRenderer.js"></script>
<script>
var camera, scene, renderer;
var circle1, circle2;
```

2. Set the scene

The code now adds the init function, used to initialise the scene. A camera is created, the control system is initiated to run on the device orientation controls. The scene is created now that the camera and control system can work with. The final code starts an array and this contains an object.

```
function init() {
camera = new THREE.PerspectiveCamera( 75,
window.innerWidth / window.innerHeight, 1,
1000 );
controls = new THREE.
DeviceOrientationControls( camera );
scene = new THREE.Scene();
var sides = [{
url: 'img/scene-right.jpg',
position: [ -512, 0, 0 ],
rotation: [ 0, Math.PI / 2, 0 ]
},{
```

3. Create the background

To create the background for the 3D scene a skybox will be created, which is a large cube placed on the outside of all other objects. The code here is holding a reference to each of the sides, their individual position and rotation within the 3D scene in order to view the box correctly.

```
url: 'img/scene-left.jpg',
position: [ 512, 0, 0 ],
rotation: [ 0, -Math.PI / 2, 0 ]
},{
url: 'img/scene-top.jpg',
position: [ 0, 512, 0 ],
rotation: [ Math.PI / 2, 0, Math.PI ]
},{
```

4. Add each side

Each side has to be added so in total there will be six sides for this. If you wanted to create your own background, an image with a 90-degree field of view horizontally and vertically has to be taken that points at all six sides. It's easier to create this with a 3D application as the rotation and field of view can be exact.

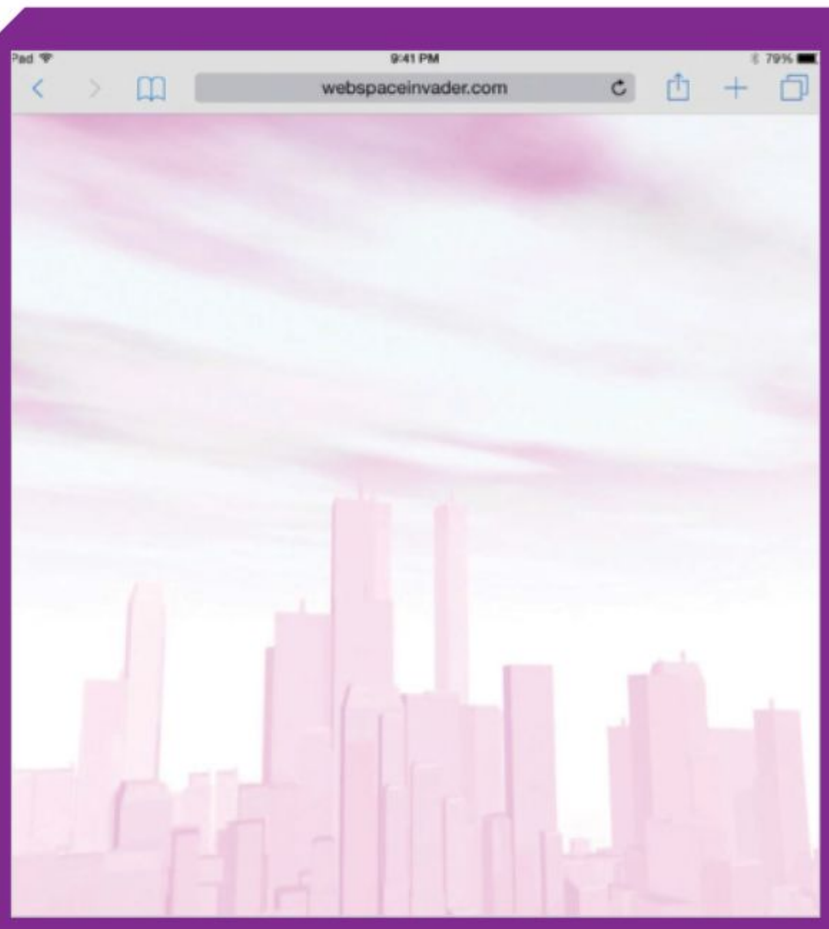
```
url: 'img/scene-base.jpg',
position: [ 0, -512, 0 ],
rotation: [   Math.PI / 2, 0, Math.PI ]
},{
url: 'img/scene-front.jpg',
position: [ 0, 0, 512 ],
rotation: [ 0, Math.PI, 0 ]
},{
```

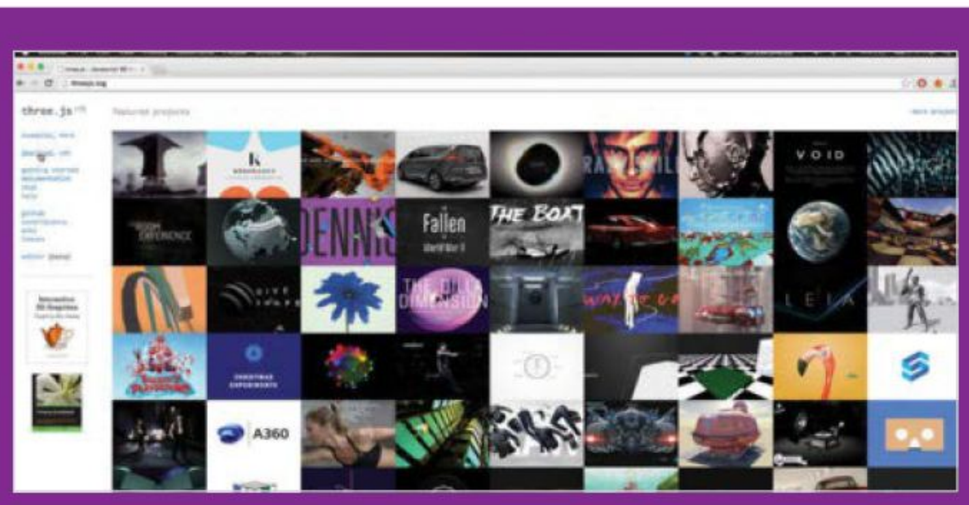
5. Set up the cube

Now as all six sides have been referenced in the array, a cube is created by making a new 3D object and this is added into the scene. A for loop is created to iterate through each position in the array and this will be used to display the images in the scene as a box.

Overlapping sides

The skybox cube is really 1024 pixels by 1024 pixels but it has been set to display at 1026 by 1026 so that there is a slight overlap in all of the sides, as CSS 3D transformations are not as accurate as WebGL.





<Top left>
• The tutorial uses the CSS3D renderer as part of three.js. This renders the display using standard CSS3D transforms, not WebGL, and ensures maximum device compatibility

<Bottom left>
• Looking towards the floor shows the area that will contain an interactive Google Map later in the tutorial

<Left>
• Adding the background images to the cube makes a 3D environment as the user moves their device around them

```
url: 'img/scene-back.jpg',
position: [ 0, 0, -512 ],
rotation: [ 0, 0, 0 ]
}];
var cube = new THREE.Object3D();
scene.add( cube );
for ( var i = 0; i < sides.length; i ++ ) {
```

6. Set each side

Information from each array position is stored in the variable side. A new image element is created and given a specific width. A CSS3DObject is created and added.

```
var side = sides[ i ];
var element = document.createElement( 'img'
);
element.width = 1026;
element.src = side.url;
var object = new THREE.CSS3DObject( element
);
object.position.fromArray( side.position );
object.rotation.fromArray
( side.rotation );
cube.add( object ); }
```

7. Icon images

Now that the background is in place, our attention will turn to creating icons that will sit in front of that. Another array is created, but this time round there is much less information than before and that's because the image needs to be stored for each of the icons that will be placed on the screen.

8. Iterate the icons

Again a for loop is created to iterate through the images. A div element is created to hold each image within. The image is added using the JavaScript innerHTML command to create content inside of it. The image is given a unique id and the class of 'pic' that is used to style it later in the CSS. The source image is also added for the icon.

9. Handle the interaction

A click handler is added for the picture and at the moment this simply returns the value of the id to the console log. In the second part of the tutorial, we will learn how this calls the correct content to load for each of the icons. The image is then added to a CSS3DObject for placing in the scene. Then the position of the icon is calculated around the camera.

```
pic.addEventListener( 'click', function (
event ) {
console.log(event.target.id);
}, false );
var object = new THREE.CSS3DObject( pic );
var phi = (Math.PI*2) / menu.length;
phi = phi * i;
```

10. Position the icon

Using the code from the previous section each icon is positioned at equal positions around the camera. The 350 relates to the radius of the circle or how far from the camera the icons will appear. Once in position each icon is rotated to face the camera in the centre of the scene.

11. Add to the scene

The final step is to tell the object to look at the vector position setup, then the icon is added into the scene. The closing bracket will now finish off the for loop. This will mean that all of the code that is inside of the for loop is applied to each image icon so that they are all positioned to orbit the camera.

```
object.lookAt( vector );
scene.add( object );
}
```

12. Interactive map

This next block of code is to add an interactive Google Map to the page. Visit Google Maps and find a location you'd like to show, then click the menu button and from the slide-out panel, click the 'share' button. In the popup grab the embed code and paste the iframe into the third line of code shown here.

```
element = document.createElement( 'div' );
element.className = 'mapped';
element.innerHTML = 'embed code here';
var elementWidth = 600;
var elementHeight = 600;
```

13. Position the map

There is a large red square on the floor of the 3D scene, which is part of the background images added earlier on. We are positioning the map to be 600 by 600 pixels so, in the embed code added from Google, change the width and height to 600 for both. The map is positioned further down away from the camera and then rotated toward the camera.

```
element.style.width = elementWidth + "px";
element.style.height = elementHeight + "px";
var cssObject = new THREE.CSS3DObject(
element );
```

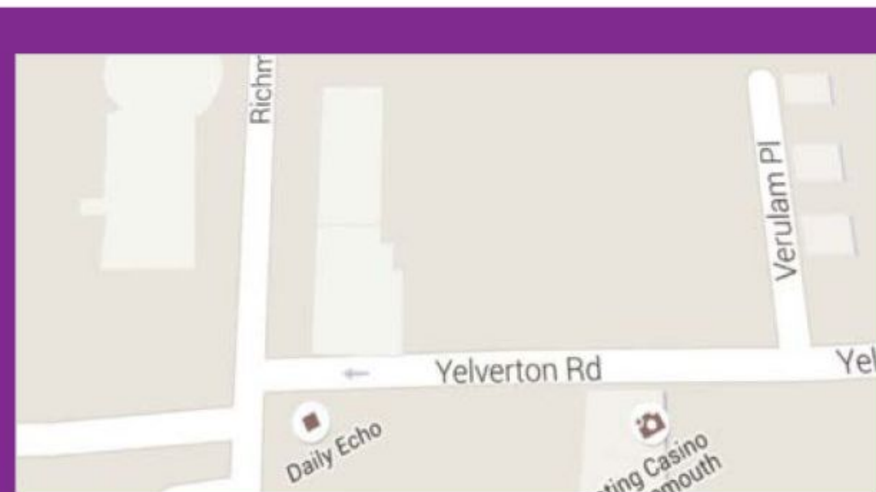
3D CSS transforms

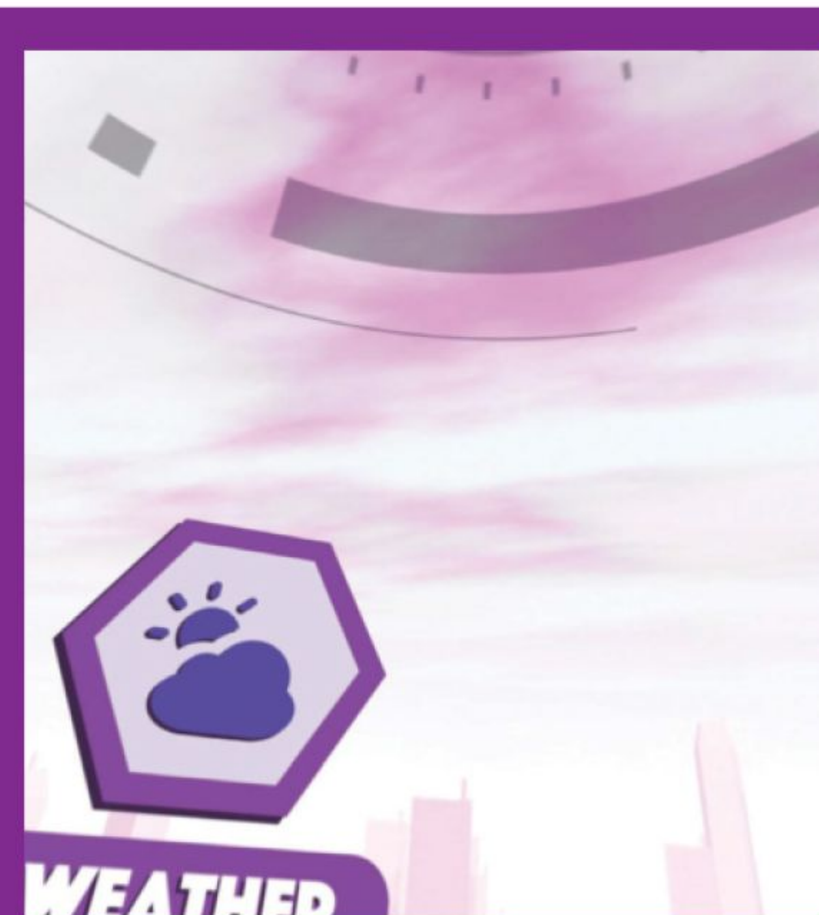
Anything can be displayed in the 3D scene that can be displayed on a regular webpage, such as video, maps, text and images, because the scene is all controlled with CSS 3D transforms.

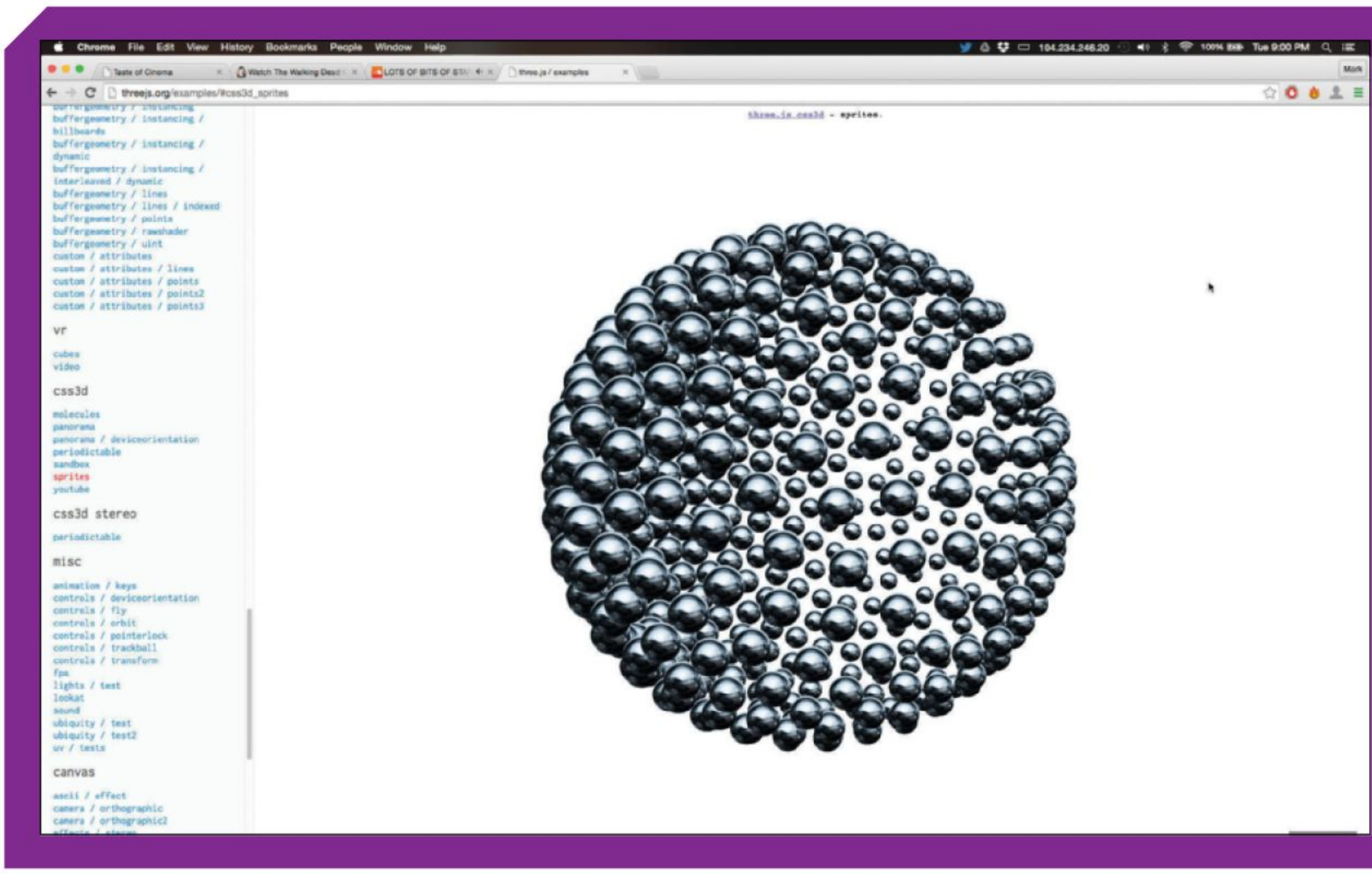
<Top left>
• Point the device towards the floor to reveal an interactive Google Map that shows the location of the event

<Bottom left>
• A circle is added to the scene and placed above the camera and can be seen by tilting the device upwards

<Right>
• Another circle is added and animation sets these circles to rotate around. If the aspect changes to landscape the scene automatically resets to fit the device window







Know your three.js renderer

Three.js is a 3D library that is more used for creating interactive WebGL experiences but it also comes with one or two extra renderers that offer 3D interactive experiences for browsers that can't display WebGL scenes. This list is small but some Android phone vendors such as Samsung have spotty support for WebGL across their product line, so it's safer to use three.js' renderer. This limits the development to creating content that is essentially just mapped onto flat planes, but with a little creativity this can still be very effective. Other renderers available include SVG and the Canvas renderer. SVG doesn't really offer any improvement over CSS but Canvas allows for 3D model formats to be loaded and displayed.

```
cssObject.position.y = -400;
cssObject.rotation.x = -Math.PI/2;
scene.add(cssObject);
```

14. Add animated elements

The next element that is being added is a transparent PNG image with circular lines on it. This is added into the 3D scene and positioned above the camera while being rotated towards the camera. This will be animated to spin later on to provide some animation within the scene.

15. A second circle

Another circle is added and you will notice the pattern that a div element is created, inside here an image is placed. The div needs to be added as a CSS3DObject, this can be positioned and rotated to display correctly. Finally add the object to the scene to be displayed.

16. Display the 3D scene

The way the scene is to be updated is now set. Normally with three.js scenes this is the WebGL renderer but in this tutorial the CSS3D renderer is being used, for maximum compatibility. An event handler is registered to call the onWindowResize function if the screen changes and the animate function is used to update the screen.

```
renderer = new THREE.CSS3DRenderer();
renderer.setSize( window.innerWidth, window.
innerHeight );
document.body.appendChild( renderer.
domElement );
window.addEventListener( 'resize',
onWindowResize, false );
```

```
animate();
}
```

17. Update the screen display

If the phone or tablet device is rotated from portrait to landscape or vice versa, the browser changes dimensions. In order to capture this, the onWindowResize function updates the aspect ratio of both the camera and the renderer so this will work on any size device with any size display.

18. Animate the scene

The animate function updates the screen's display using the browser's requestAnimationFrame command to update the screen at 60 frames per second, or as fast as the device can render it. The circles are rotated on the screen, the controls update and the scene is rendered or updated to display the view.

```
function animate() {
requestAnimationFrame( animate );
circle1.rotation.z += 0.001;
circle2.rotation.z -= 0.002;
controls.update();
renderer.render( scene, camera );
}
```

19. Set it all up

As we come to the end of the JavaScript code, the init() function is called which sets the scene up, placing everything in there and calling the scene to start rendering. After this the script tag is closed. Save this page now as all of theJavaScript code is added in here.

```
init();
</script>
```

20. CSS Body

Open the project.css file from the css folder on FileSilo. This is empty of all code, so add the code shown here for the body tag styling. This sets the background colour to white and sets the margins to zero, so that the scene fills the full browser display. The overflow is hidden and the font set to Helvetica or Arial.

```
body {
background-color: #ffffff;
margin: 0;
cursor: move;
overflow: hidden;
font-family: helvetica, arial, sans-serif;
}
```

21. The menu items

The next CSS rule sets the image icons that are displaying as the menu. This just sets the cursor to a pointer if the user has a cursor on rollover. This won't be necessary for the majority of browsers as mobile and tablets don't usually have mouse pointers!

22. Finish and save

The size of the pictures is set here using the width and height properties. Without this the image icons are just too big on the screen, but it's useful to have the images a little bigger as most mobiles and tablets have high-pixel resolution screens. Save the file and place on a server then connect with your mobile device.

DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/bks-894

Create a unique mobile 3D interface

Part 2

Part two of our tutorial takes an augmented menu that uses the mobile's device orientation controls and adds interaction to create a fully working site experience

In part one of this tutorial in issue 242, a unique mobile augmented website menu was created. The site was created with the help of the three.js library and was rendered using the CSS3 renderer rather than the WebGL renderer to give maximum compatibility with devices. The 3D was an integrated part of the design as the user had to rotate their phone around them to display the interface, augmenting the site around them. This placed the user in the centre of the interface.

The movement of the phone was created using the 'Device Orientation Controls' that come bundled with three.js and this sort of turned our phone into something similar to an Oculus Rift.

Now, in this final part of the tutorial, the interface will be given some content so that it actually becomes useful. This content will be hidden from the user until they click on the appropriate icon in 3D space, at which point a 2D panel will slide in over the top of the existing 3D menu system. The animation in the panels,will be powered by the Animate.CSS library, which will slide on the new content. This gives us access to a quick way to create animated elements.

Each panel will be colour-coordinated to match the icon that has been clicked and then the panel can be removed again by clicking a burger menu icon in the top left to take the user back to the experimental menu.

1. Pick up from part one

This tutorial picks up directly where it was left off at the end of part one. First link up the animate.css library, available from **daneden.github.io/animate.css**, which we've already downloaded and put in the CSS folder. This will power the sliding on of popover content later on.

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

2. Content panels

There are going to be four content panels in the project – these will appear when the user clicks on one of the icons in the menu inside the 3D scene. Add the code as we've below to the body of the page just above the script tags. This creates a top bar section and an area for text.

```
<div id="info1" class="panel hidden
animated">
<div class="head">
<a href="#" onclick="fade1();">&#9776;</a>
<h2>Chat</h2>
</div>
<p>Text here </p>
<p>Text here </p>
</div>
```

3. Panel details

You can fill the text with your own content or even images. You may notice that there is a strange code in the link tag, this is actually the code to create the three lines of the burger icon for the menu button. This will take the users back to the 3D menu screen.

```
<div id="info2" class="panel hidden
animated">
<div class="head">
<a href="#" onclick="fade2();">&#9776;</a>
<h2>Arriving</h2>
</div>
<p>Text here </p>
<p>Text here</p>
</div>
```

4. Panel power

The panels are constructed from regular HTML elements and will be hidden from the site using CSS. Then linking up the CSS Animate code when the user selects the appropriate icon the panel will slide in from the left. Clicking the burger icon will make it slide back to the left.

```
<div id="info3" class="panel hidden animated
weather">
<div class="head">
<a href="#" onclick="fade3();">&#9776;</a>
<h2>Weather</h2>
</div>
<p>Text here</p>
<p>Text here</p>
</div>
```

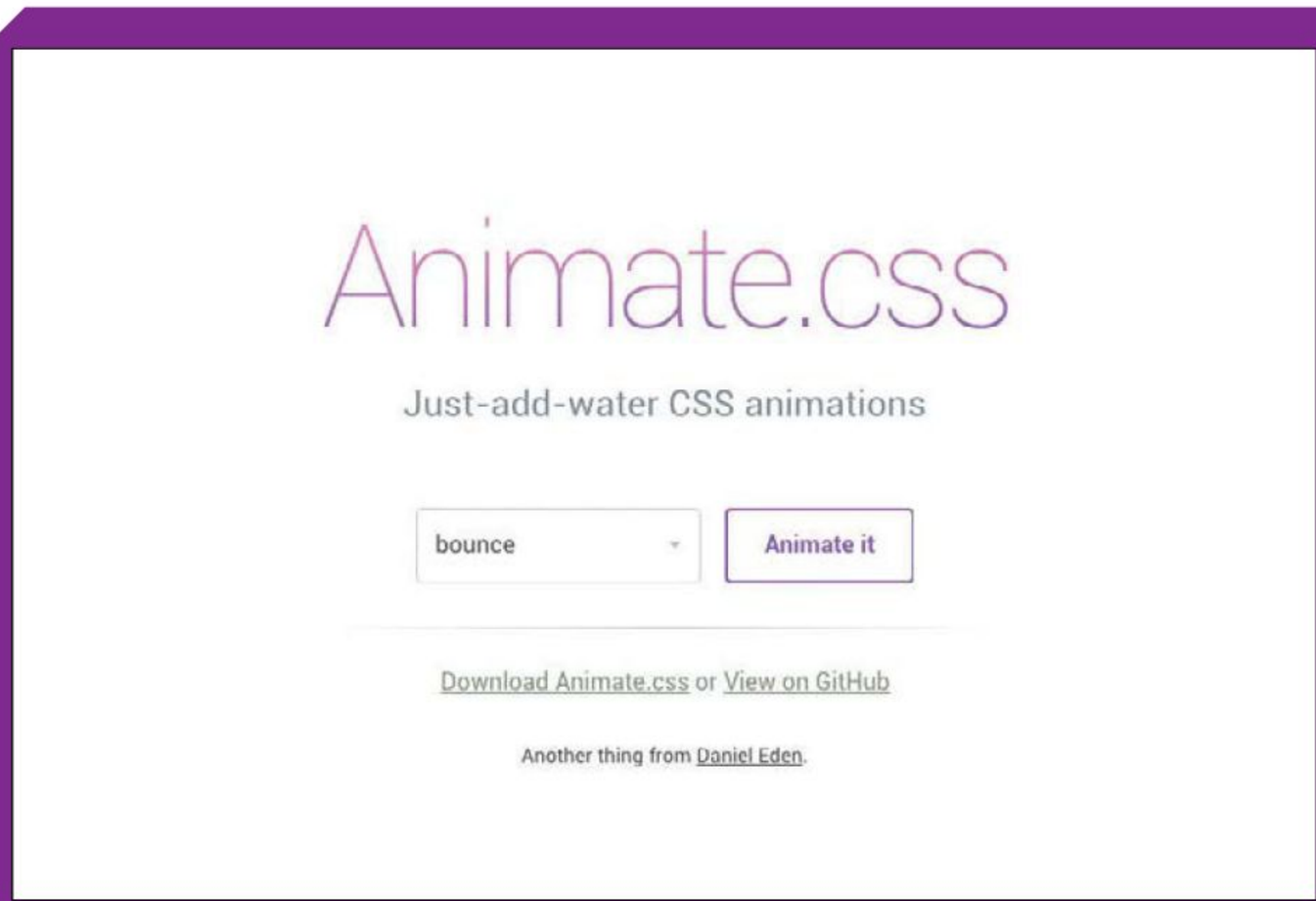
5. Five into four

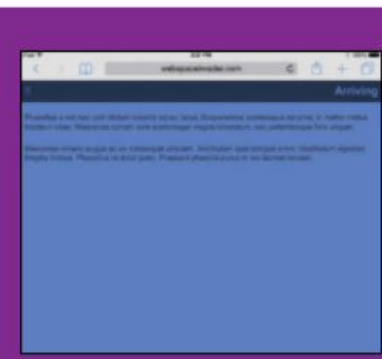
Each panel will be styled up to match the colour of the icon using CSS code – the colour will be used to reinforce the selection made by the user. There are five icons and but only four panels. One icon is the logo of the website so that will not link anywhere, but it will orientate the user.

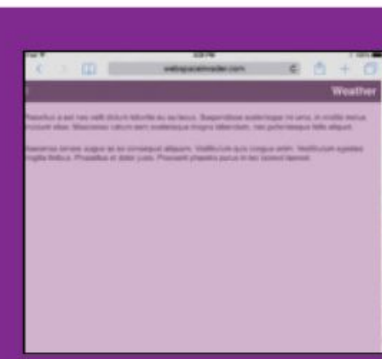
```
<div id="info4" class="panel hidden
animated">
<div class="head">
<a href="#" onclick="fade4();">&#9776;</a>
<h2>Events</h2>
</div>
<p>Text here</p>
<p>Text here </p>
</div>
```

Positioning above

The new panels added in this tutorial are positioned on a z plane above the background CSS panels so that they appear in front of the 3D scene as 2D panels.







Left
The Animate.CSS library is used to create transitions in the app and slide the appropriate content over the top of the existing menu

Top left
The HTML for the panel is created that forms the structure of the overlaid content that appears above the interface when the appropriate icon is clicked

Top right
In the CSS the panels are styled to have individual looks so that each icon slides content over the interface in a similar colour scheme. This content hasn't quite fully fallen into place in the image

6. Responsive srcset behaviour

Save the HTML document for now and open the 'project.css' file from the CSS folder. Add on the code in the next few steps to style the content in the panels appropriately. The first content merely creates the generic style for all of the panels that place this above the other content on the page.

7. Hide and seek

When the App starts up the menu should be visible and nothing else. The hidden class is used to hide all the panels when the App first launches and then only as the user taps the device screen should it be visible to the user. The visible class was actually used during testing but is not used in the final site.

```
.hidden{
display: none;
}
.visible{
display: block;
}
```

8. Break down the parts

After the panel is styled up, each panel will have a head section that has a burger menu button to take users back to the menu and a heading so that they know they are seeing the right content on the screen. This CSS just allows for the head section to have the right height for both of those elements.

```
.head{
height: 45px;
}
```

9. Position the menu

The link in the head section of the panels is the burger menu icon and here the code is floating that icon to the left of the screen. It's given a slight padding around the top and left so it isn't too close to the sides. The text decoration is set to none and ensures that the underline doesn't appear below this.

```
.head a {
display: inline-block;
float: left;
width: 10%;
padding: 10px 0 0 10px;
text-decoration: none;
}
```

10. Lay out the heading

Now, turn your attention to the h2 heading tag, which is made to float to the right of the menu icon. The padding on this reflects the slightly different shape of the text and the icon, with there being less top padding on the text. The text is also aligned to the right so that the menu and heading are on opposite sides of the screen.

```
.head h2{
display: inline-block;
width: 70%;
margin: 0;
padding: 7px 10px 0 0 ;
float: right;
text-align: right;
}
```

11. Paragraph settings

Below the heading there is just a couple of paragraphs, but if this was your own design this could contain a variety of content, from images, video's or anything else you wish to display on a HTML page. As there is only text in the example there is just a 5 pixel padding for the top and bottom and 10 pixels for left and right.

```
p{
padding: 5px 10px;
}
```

12. Individual style

Each panel shares the same generic styling that has been applied in the previous six steps. Now you need to concentrate on giving each panel a colour scheme that fits with the initial icon that it represents. Here the panel is given a light green background with dark green text and header.

13. Blue section

As in the previous step, the styling is now looking at the individual panels. The second panel will be styled up with a light blue background, giving a darker blue colour for the text. The header panel of this is the reverse of that styling with a dark blue background and lighter text.

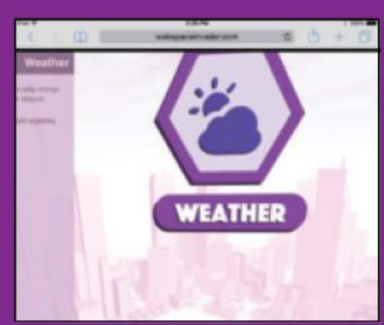
14. Third panel

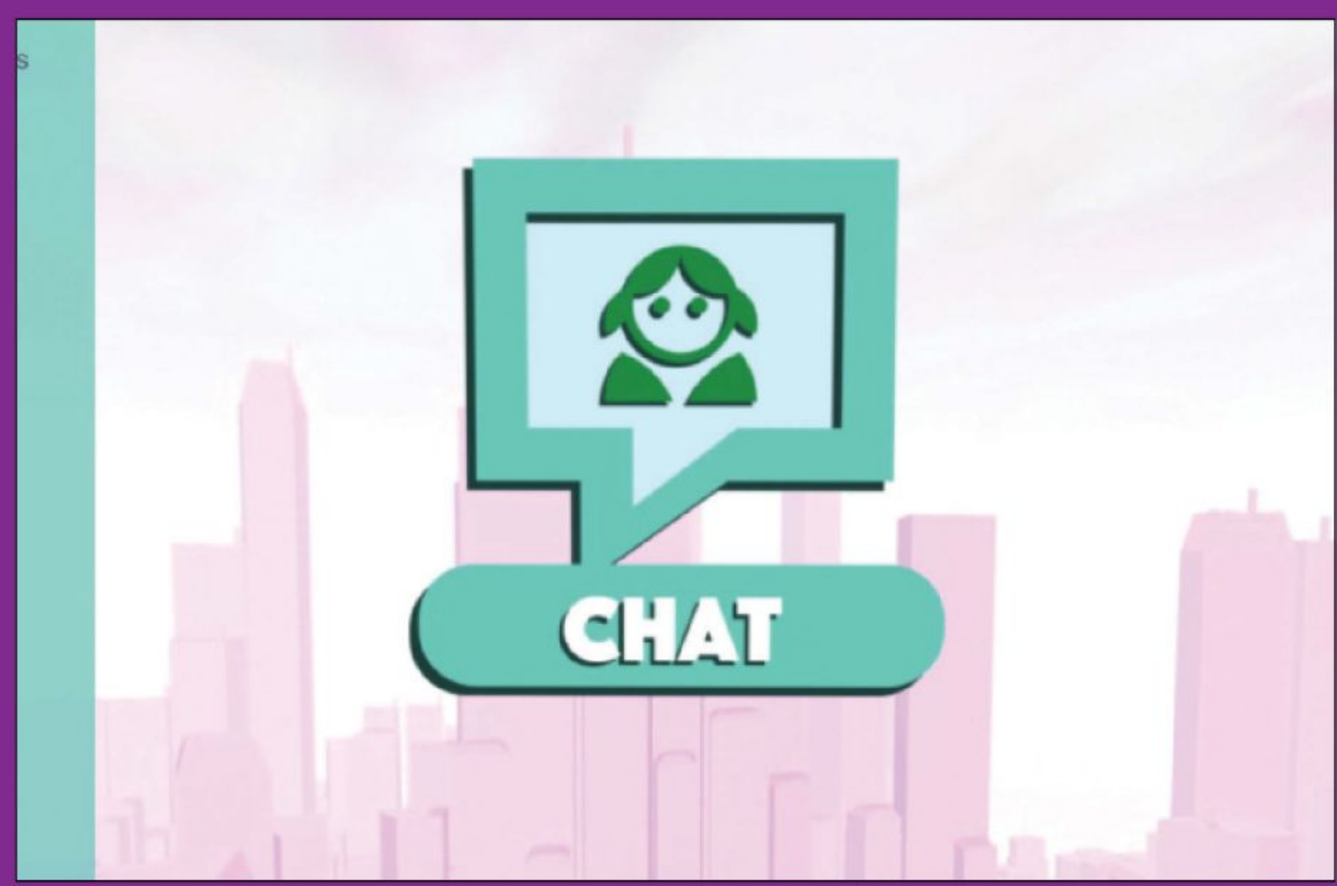
The third panel takes on purple characteristics for the colour scheme with lighter styling for the background and a darker header section on this panel. The colours help to reinforce to the user the icon that they've just clicked on to give them a way to orientate themselves in the mobile web site.

```
#info3.panel{ background-color: #d1b3ce ; }
#info3 .head{
color: #eed8ec;
```

Know your code

HTML is the content of your website while CSS provides the way those elements are rendered by the browser giving a unique design. The JavaScript part of the site provides functionality.

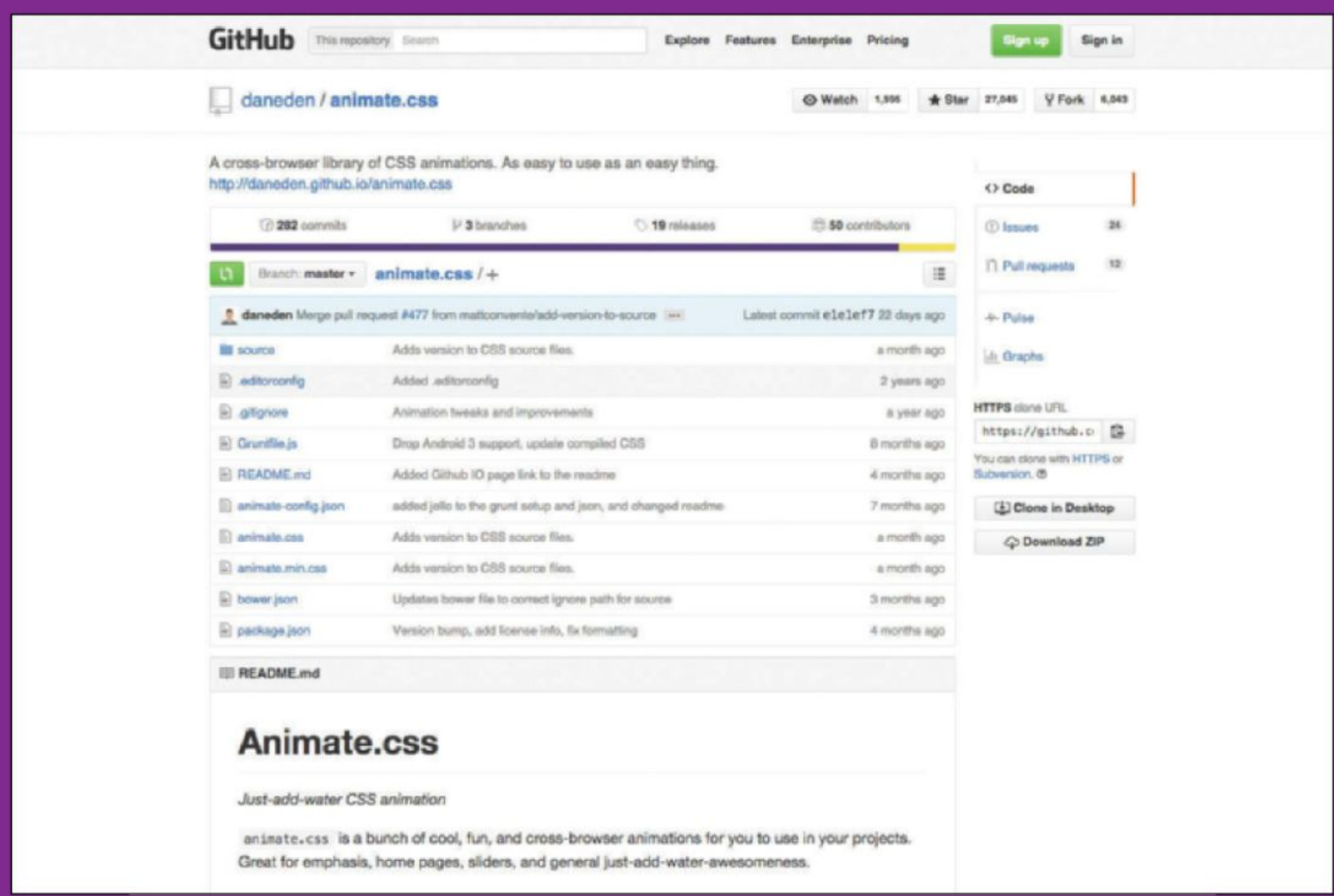




Top left
The Animate.CSS library is used here, and this slides the content onto the screen with a bounce. Here the content is just about to bounce into final position

Top right
As the weather icon is clicked you can see the panel just beginning to be animated into position and its transparency is just fading in here as well

Right
When the user clicks on the burger menu in the top left of the panel the content slides back off the screen to the left as shown in this image. This allows the menu to be accessed again



Using Animate CSS

The Animate CSS library is a relatively lightweight library with the smaller version weighing in at only 52KB. This consists of one CSS file that has keyframes built for various CSS movement and animation. This saves us the time in the tutorial of having to put these together and it is always worth your while as a web designer to build on the shoulder of other libraries. When you can, don't reinvent the wheel!

To use the library, all that needs to happen is that the library is linked to your page, which we did in step one. Then, any content that needs to be animated must have the class 'animated' applied to it. Finally, an animation style should be applied as a class, such as 'bounceInLeft'. This will immediately call the animation into effect.

```
background: #714c6e;
}
#info3 .head a{ color: #eed8ec; }
#info3 p{ color: #584356; }
```

15. Final panel colours

The final panel that is being styled up again reflects the colours of the icon used to access it from the original 3D menu created in part one of this project. This time the colour scheme is given an orange theme with light tints and dark shades used to create the panel. Save the CSS file and close it.

```
#info4.panel{ background-color: #f5c5a7 ; }
#info4 .head{
color: #f5c5a7;
background: #b65212;
}
#info4 .head a{ color: #f5c5a7; }
#info4 p{ color: #b65212; }
```

16. Start the JavaScript

All of the requisite elements are now in place for the design and content of the panels. It's just a case of adding the desired functionality to make these panels work. In the JavaScript code locate the global variables on lines 56 and 57, then add this function below it. This is called into effect when the menu burger icon is pressed on the first panel.

```
function fade1(){
var change = document.getElementById(
'info1' );
change.classList.remove('bounceInLeft');
change.classList.add('bounceOutLeft');
}
```

17. Bounce around

The next code is for the second panel and these series of functions take the ID of the relevant info panel then remove the class that has made them bounce onto the screen from the left. Once this is removed the code to make them leave is added and the elements bounce back out to the left.

```
function fade2(){
var change = document.getElementById(
'info2' );
change.classList.remove('bounceInLeft');
change.classList.add('bounceOutLeft');
}
```

18. Use Animate CSS

The CSS styles that are being added and removed for these sections have not been created here – instead they are from the Animate.CSS library that was hooked up to our page in the first step of this tutorial. You can visit **daneden.github.io/animate.css** to see more styles.

```
function fade3(){
var change = document.getElementById(
'info3' );
change.classList.remove('bounceInLeft');
change.classList.add('bounceOutLeft');}
```

19. Customise the animation

If you visit Dan Eden's Animate.CSS site you can try out different animation styles before applying them to your content. Find styles that work together and make sense with your functionality rather than choosing the most attention grabbing animation that you see.

```
function fade4(){
var change = document.getElementById(
'info4' );
change.classList.remove('bounceInLeft');
change.classList.add('bounceOutLeft');
}
```

20. Grab the link

Following this you'll need to scroll down through the code until you find the code that has the click function event listener on the pic variable. Inside this function, add the two lines of code in bold shown below. These lines check that the icon that has been clicked on is not the first icon, and if so it stores the current icon's ID in the change variable.

```
pic.addEventListener( 'click', function (
event ) {
console.log(event.target.id);
if(event.target.id != "0"){
var change = document.getElementById(
'info'+event.target.id );
```

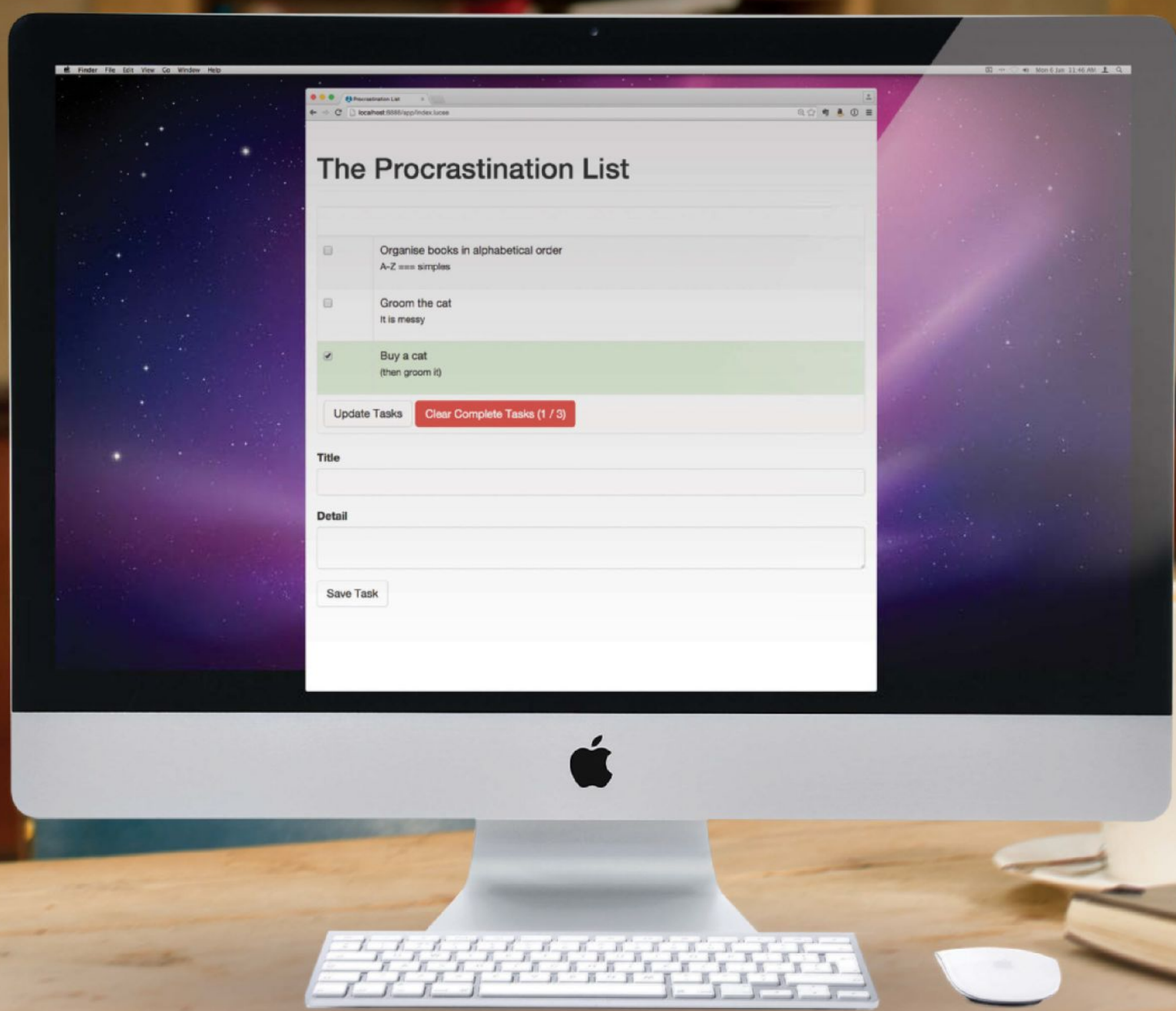
21. Finish the code

Now continue with the following code that turns off the hidden property, and if it has been clicked on previously it removes the bounce out CSS. The bounce in animation is added to bring the panel onto the screen. Save this, upload it to a server and connect using your phone's browser to see it all working.

DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/bks-894

Develop a web app quickly with Lucee

Say hello to the Lucee language, the latest CFML engine, and quickly build a powerful dynamic application



To some, CFML (ColdFusion Markup Language) may not mean anything. To others it may be perceived as an out-of-date language no longer used. In fact CFML is 20-years-old this year and it was born the same year as PHP and JavaScript. If you haven't yet experienced building a CMFL application yet, then now is your chance to do so.

Lucee, a completely open source CFML engine, was released in January 2015. As a fork from an existing CFML engine Lucee did not have to start from scratch and had a fantastic core engine to build upon.

In this tutorial we will download and install an Express edition (self-contained) version of Lucee 5, which includes all of the latest additions to the code syntax and language. We will develop a simple example application using the new Lucee dialect and file extensions, and we will then show an example of how to interact with the application scope for persistence. We will use Lucee's member functions to check for variables within certain scopes and lengths of arrays.

We will also look into creating RESTful API resources using Lucee and mapping components to match REST paths. We will see how to manage sending arguments in the URL as well as dealing with both GET and POST requests to the API.

1. Download Lucee

The Lucee language is available for us to download from a number of platforms in stable and beta versions. Just head on over to **lucee.org/downloads.html** to do just that. We will be selecting Express version 5 to use in this tutorial, which is a version that will work across all operating systems. Now you can just unpack the archive file and use your terminal or command prompt to start up the server.

```
./startup.sh
or for Windows:
./startup.bat
```

2. Default page and admin

After starting up the Lucee engine you will be able to access the server in your browser via **http://localhost:8888**. The default welcome page that appears here will have some useful links to guide you through the language, and it will also have two important links that will take you right through to the server and web administrators too.

3. Application component

Create a new file in the root of the new directory called 'Application.lucee'. This component will hold the application-specific variables and configuration for this subapp. Create the name using a hashed version of both the current path and substring value to help ensure that it is unique.

```
component {
this.name = hash(GetDirectoryFromPath('./'))
& '_procrastination_app';
}
```

4. Run onApplicationStart

The application framework will support multiple event-driven functions which we are then able to interact with. OnApplicationStart will run the very first time that the application instantiates. Here we will then define two variables that are assigned to the application scope, and these are made available for us to access throughout the entire application.

```
boolean function onApplicationStart() {
application.tasks = [];
application.completeTasks = "";
return true;
}
```

5. Directory index

Create a new file called 'index.lucee' in the directory. This will be the default index page for our application. To start we'll simply request a data dump of everything available within the application scope (created from the application configuration file). You can use the tag format or script syntax to do this.

```
<:script>
dump(application);
</:script>
```

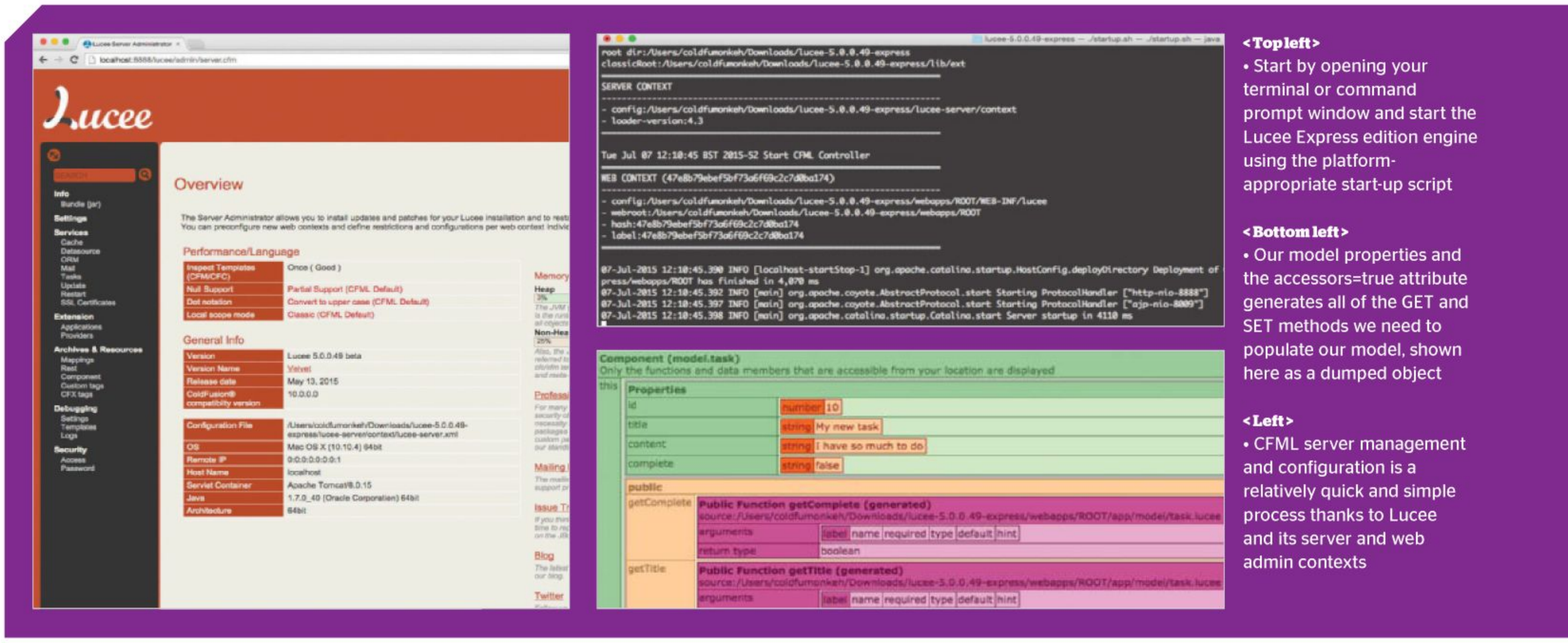
6. Task model definition

Create a new directory of 'model', and inside of this create a new file, 'task.lucee'. This new component contains properties for our model defined, setting the type, name and default value attributes for each. The getters and setters will be created for us thanks to the accessors=true attribute.

```
component accessors="true" {
property type="numeric" name="id";
property type="string" name="title"
default="";
property type="string" name="content"
```

Lucee 5 beta

In this tutorial we have been using the latest version of Lucee 5, still in beta at time of writing but almost production ready. The new Lucee dialect code format and file extensions are only available in version 5.



<Top left>
• Start by opening your terminal or command prompt window and start the Lucee Express edition engine using the platform-appropriate start-up script

<Bottom left>
• Our model properties and the accessors=true attribute generates all of the GET and SET methods we need to populate our model, shown here as a dumped object

<Left>
• CFML server management and configuration is a relatively quick and simple process thanks to Lucee and its server and web admin contexts

```
default="";
property type="boolean"
name="complete"
default="false";
}
```

CFML from the CLI

CommandBox is a stand-alone native command line interface tool written in CFML with package management and much more. It also generates embedded Lucee servers for your local development work. You can find out more by going to **bit.ly/1CqTokB**.

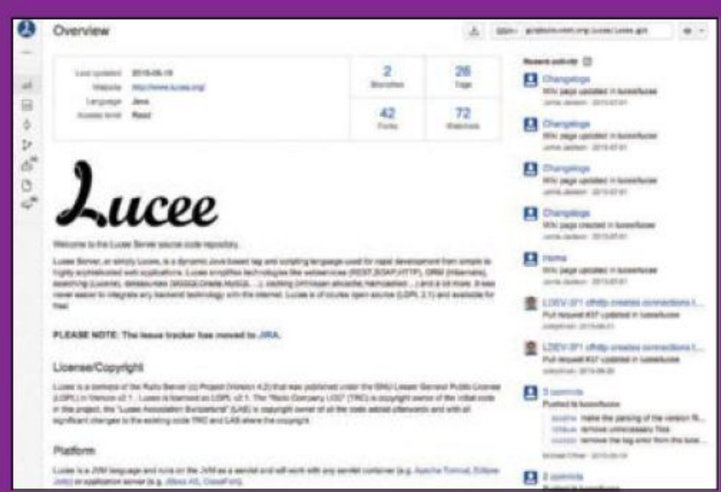
7. HTML layout

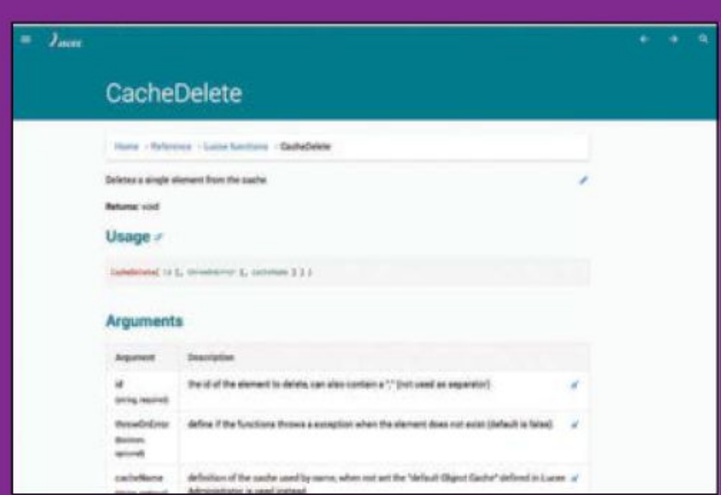
Open index.lucee and start by defining the base HTML layout for the application. We will be using Bootstrap for structure here. You should take note of the output tags wrapping the entire template. It is these output tags that will deal with any string wrapped by # characters as a variable for translation.

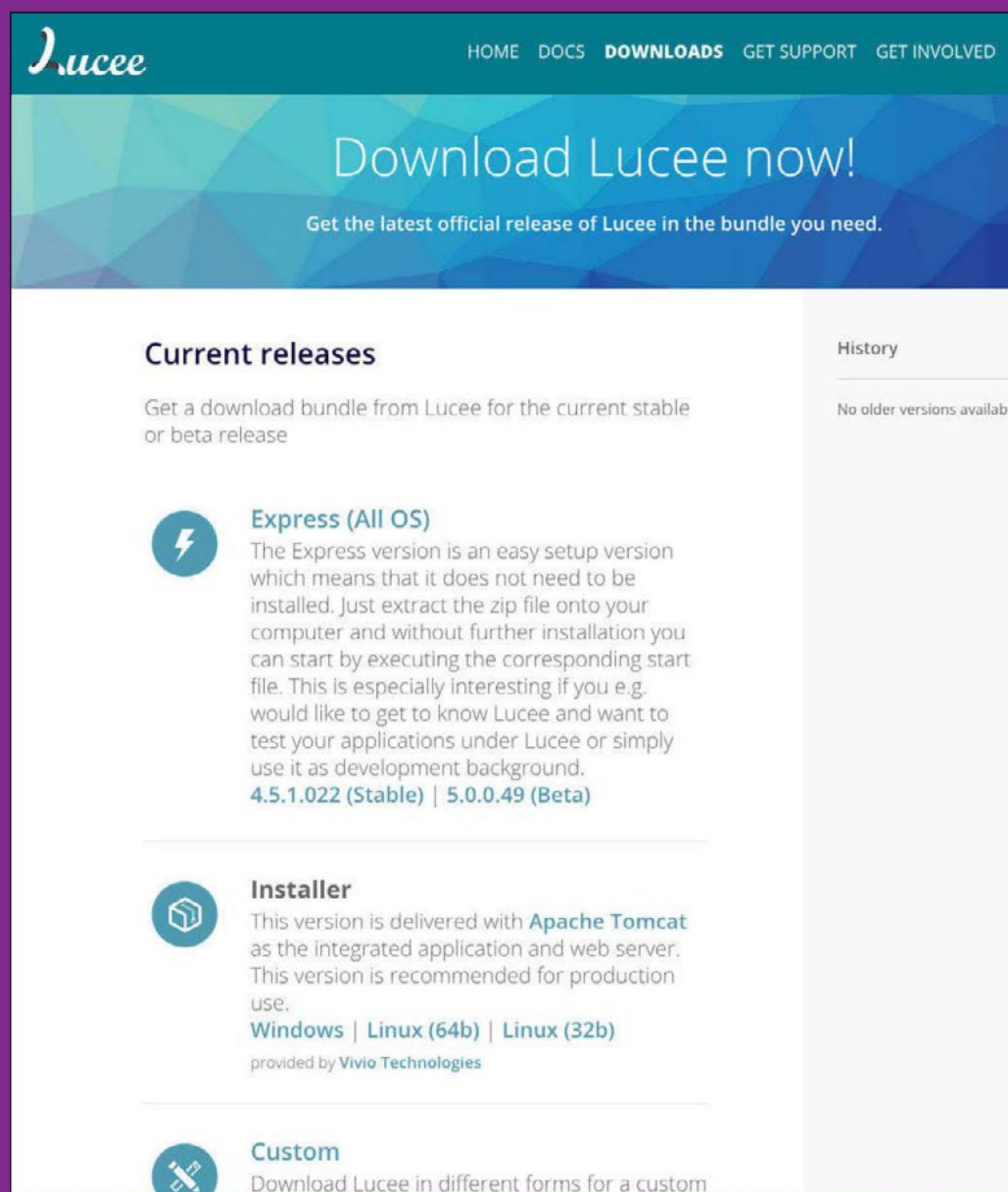
```
<:output>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Procrastination List</title>
<link rel="stylesheet" href="css/bootstrap.
min.css">
</head>
<body>
</body>
<script src="https://ajax.googleapis.com/
ajax/libs/jquery/1.11.3/jquery.min.js"></
script>
<script src="js/bootstrap.min.js"></script>
</html>
</:output>
```

8. Form definition

Create a new form now that will post back to the index.lucee page. This will have a text input field with the name attribute set to title and a textarea field with the name set to content. The submit button will have the name new_task which we will use to identify a submission. Then finally, the variable #strFlash# will be



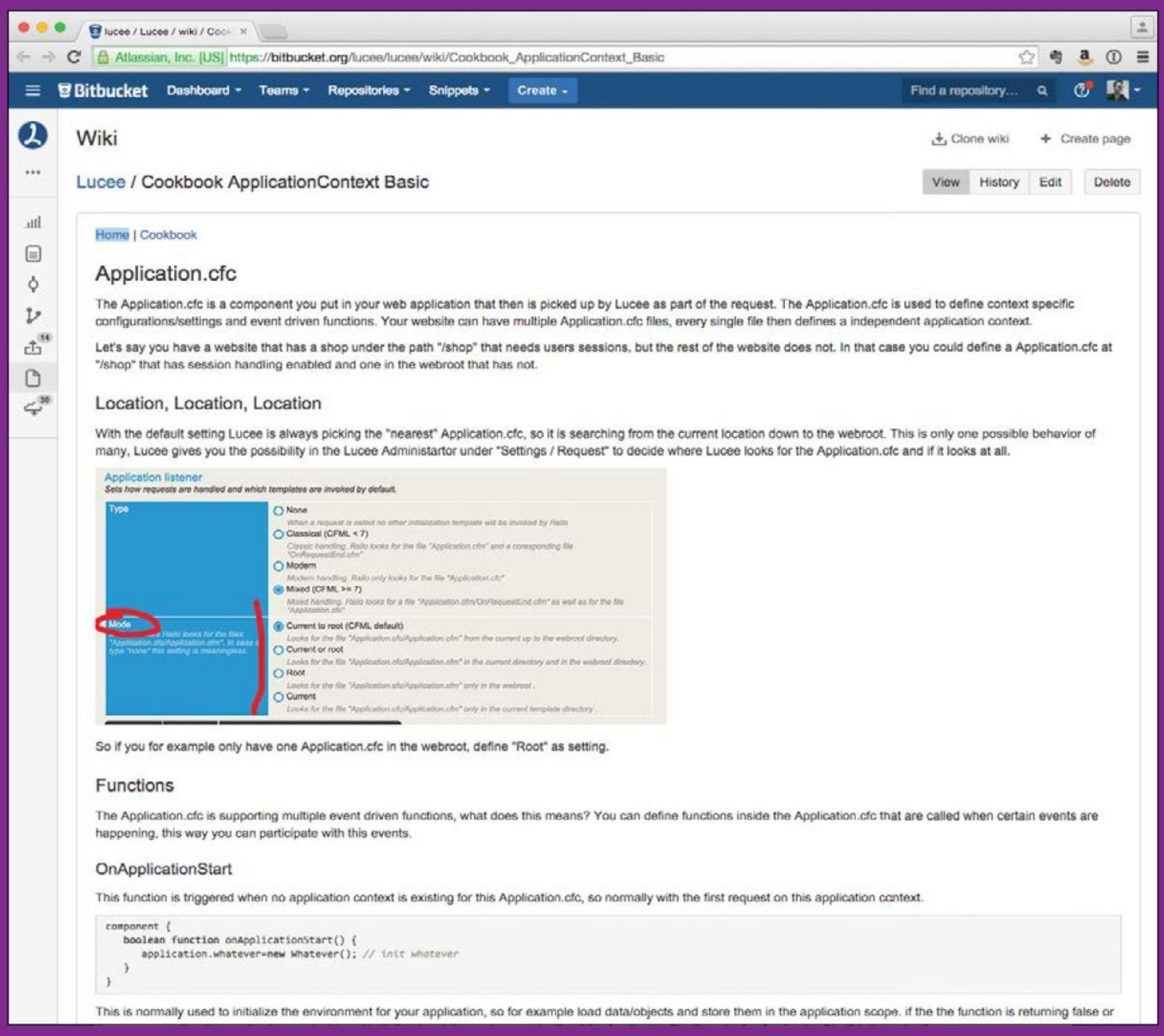




<Top>
• Lucee has been open and available to the community since its initial release and welcomes pull requests and community interaction

<Mddle>
• IFull documentation for the language tags and functions are available from the Lucee site, complete with all attribute variations and options

<Right>
• Lucee is available to download for Windows and Linux-based operating systems, in full installer versions or Express editions



The application framework

Throughout this tutorial we have made extensive use of the application scope to persist variables throughout the application. It's not advised to use that scope as a data store (except for this example application as it saved on setting up a database connection) but it is incredibly useful for persistence and sharing objects, variables and certain bits of data. We created an application.lucee file, which could also have been an application.cfc (ColdFusion Component). This is a special file for CFML projects and enables a lot of event-driven method processing. Make sure that you check out this great introduction to the application framework in Lucee for more information **bit.ly/1gF2Mau**.

> As a fork from an existing CFML, Lucee didn't have to start from scratch and had a fantastic core engine to build upon

used to display any errors that appear.

```
<div class="container">
<h1 class="page-header">The Procrastination
List</h1>
#strFlash#
<form method="POST" action="index.lucee"
class="form-horizontal">
<div class="form-group">
<div class="col-sm-offset-1 col-sm-11">
<button type="submit" name="new_task"
class="btn btn-default">Save
Task</button>
</div>
</div>
</form>
</div>
```

9. Script block

Create a new script block at the top of the index.lucee file. Create a default strFlash variable with an empty value. An if statement will check for the existence of the new_task button in the form scope (available after a form submission). If it exists we can then process the submitted values accordingly.

```
<:script>
strFlash = "";
if ( structKeyExists(form, 'new_task') ) {
}
</:script>
```

10. New model instantiation

If both the title and content values are present in the form submission then we will instantiate a new task model, and we will then populate the properties with the values and insert that component into the task array using the built-in arrayAppend function. If either of the title and content are missing then we'll create the warning message for output.

```
if ( len(form.title) && len(form.content)
) {
task = new model.task(
title=form.title,
content=form.content,
id=arrayLen(application.tasks)+1
);
arrayAppend(application.tasks, task);
} else {
strFlash = '<div class="alert alert-
warning">Please provide a title and task
```

```
detail.</div>';
}
```

11. Display the list

Below the header, we will now add some code to detect if we have any records saved in the array. ArrayLen is a built-in CFML function to check for the length of the provided array. If true, the following HTML panel and table will be displayed to the user.

```
<:if arrayLen(application.tasks)>
<div class="panel panel-default">
<table class="table table-bordered table-
hover">
<thead>
<tr>
<th colspan="2"> 
</th>
</tr>
</thead>
<tbody>
</tbody>
</table>
</div>
</:if>
```

12. Loop over collection

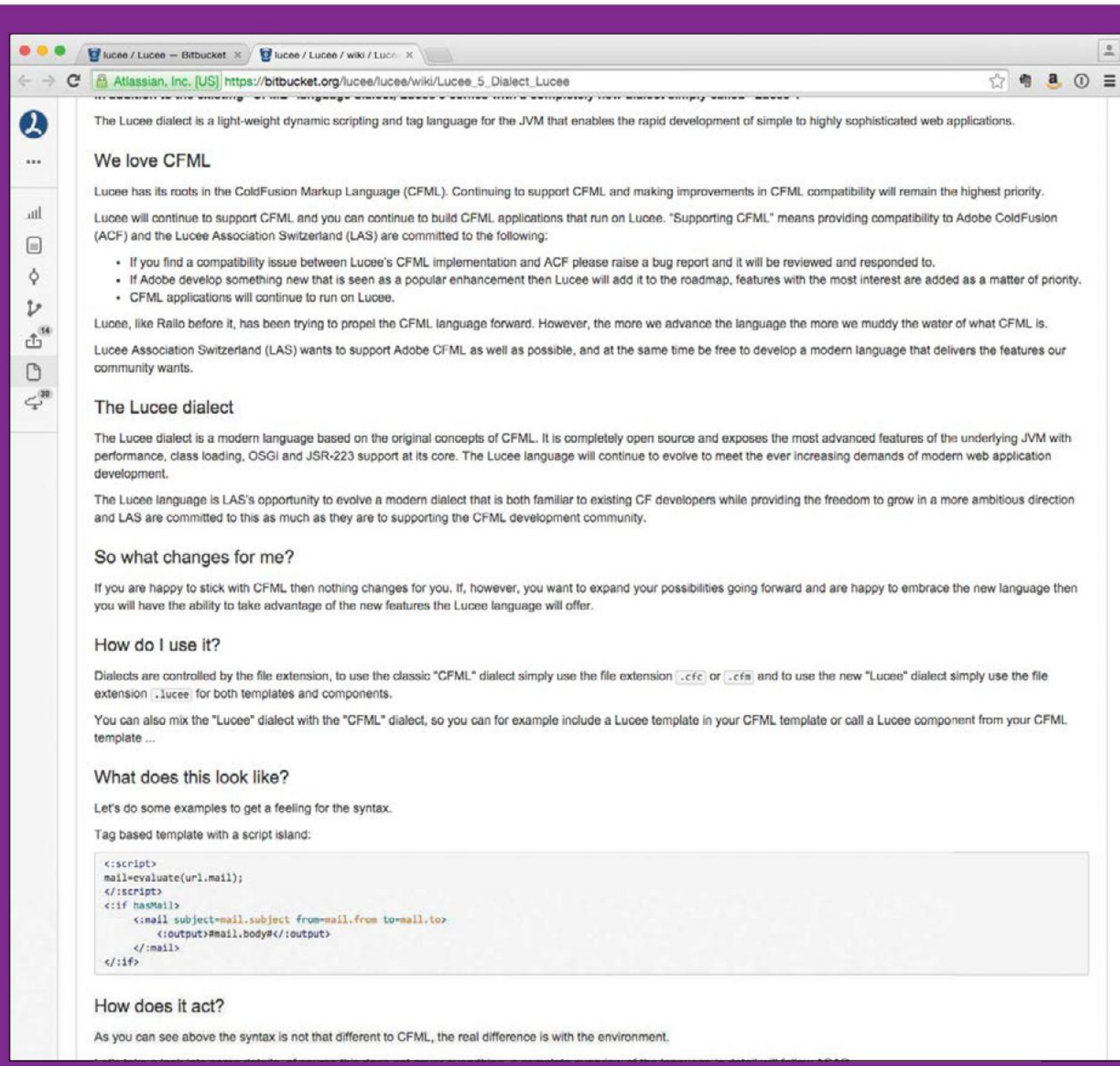
Place a new form between the table body tags to post back to index.lucee. Within this form loop over the tasks array from the application scope. The checkbox id and value hold the task id and the conditional statement will populate the checkbox if already marked as complete.

```
<:loop collection=application.tasks
item="task">
<tr class="<:if (application.tasks[task].
getComplete())>success</:if>">
<td>
<input type="checkbox" name="task"
```

> As a relatively new engine it continues to support CFML and the ecosystem around it which includes compatibility support for existing versions of CFML engines

Introducing the Lucee dialect

Lucee 5 contains an updated enhanced language framework and syntax that is known as the Lucee dialect (or occasionally referenced as LuceeLang). The dialect is based on the original concepts of CFML. As a relatively new engine it continues to support CFML and the ecosystem around it, which includes compatibility support for existing versions of CFML engines including Adobe's ColdFusion engine. As well as helping to maintain backwards-compatible syntax for existing applications it also wants to help drive the CFML environment and update the syntax and available methods and functions. This is where the Lucee dialect comes into play. Find out more about this dialect by going here: **bit.ly/1KaNiF2**.



lucee / Lucee — Bitbucket | lucee / Lucee / wiki / Lucee

Atlassian, Inc. [US] https://bitbucket.org/lucee/lucee/wiki/Lucee_5_Dialect_Lucee

The Lucee dialect is a light-weight dynamic scripting and tag language for the JVM that enables the rapid development of simple to highly sophisticated web applications.

We love CFML

Lucee has its roots in the ColdFusion Markup Language (CFML). Continuing to support CFML and making improvements in CFML compatibility will remain the highest priority.

Lucee will continue to support CFML and you can continue to build CFML applications that run on Lucee. "Supporting CFML" means providing compatibility to Adobe ColdFusion (ACF) and the Lucee Association Switzerland (LAS) are committed to the following:

- If you find a compatibility issue between Lucee's CFML implementation and ACF please raise a bug report and it will be reviewed and responded to.
- If Adobe develop something new that is seen as a popular enhancement then Lucee will add it to the roadmap, features with the most interest are added as a matter of priority.
- CFML applications will continue to run on Lucee.

Lucee, like Railo before it, has been trying to propel the CFML language forward. However, the more we advance the language the more we muddy the water of what CFML is.

Lucee Association Switzerland (LAS) wants to support Adobe CFML as well as possible, and at the same time be free to develop a modern language that delivers the features our community wants.

The Lucee dialect

The Lucee dialect is a modern language based on the original concepts of CFML. It is completely open source and exposes the most advanced features of the underlying JVM with performance, class loading, OSGi and JSR-223 support at its core. The Lucee language will continue to evolve to meet the ever increasing demands of modern web application development.

The Lucee language is LAS's opportunity to evolve a modern dialect that is both familiar to existing CF developers while providing the freedom to grow in a more ambitious direction and LAS are committed to this as much as they are to supporting the CFML development community.

So what changes for me?

If you are happy to stick with CFML then nothing changes for you. If, however, you want to expand your possibilities going forward and are happy to embrace the new language then you will have the ability to take advantage of the new features the Lucee language will offer.

How do I use it?

Dialects are controlled by the file extension, to use the classic "CFML" dialect simply use the file extension `.cfc` or `.cfm` and to use the new "Lucee" dialect simply use the file extension `.lucee` for both templates and components.

You can also mix the "Lucee" dialect with the "CFML" dialect, so you can for example include a Lucee template in your CFML template or call a Lucee component from your CFML template ...

What does this look like?

Let's do some examples to get a feeling for the syntax.

Tag based template with a script island:

```
<:script>
mail=evaluate(url.mail);
</:script>
<:if hasMail>
    <:mail subject=mail.subject from=mail.from to=mail.to>
        <:output>#mail.body#</:output>
    </:mail>
</:if>
```

How does it act?

As you can see above the syntax is not that different to CFML, the real difference is with the environment.

```
id="check[#application.tasks[task]
.getId()#]"
<:if (application.tasks[task].
getComplete())>checked</:if>
value="#application.tasks[task]
.getId()#" />
</td>
<td>
<p>#application.tasks[task].getTitle()#<br
/>
<span class="small">#application.
tasks[task].getContent()#</span>
</p>
</td>
</tr>
</:loop>
```

13. Update the tasks

You need to include the following buttons before the closing form tag in the table. Once an update has been processed and any tasks are complete, check the length of the application scope completeTasks list variable. If it holds data, display the button to clear those items from the list.

```
<tr>
<td colspan="2">
<input type="submit" name="update_tasks"
class="btn btn-default" value="Update Tasks"
/>
<:if listLen(application.completeTasks)>
<a href="/app?clearcomplete=true"
class="btn btn-danger">Clear
Complete Tasks
(#listLen(application.completeTasks)# /
#application.tasks.len()#)</a>
</:if>
</td>
</tr>
```

14. Handle updates

Now go back to the top of index.lucee, as we will now continue by adding another conditional statement within the script block. This conditional statement will be responsible for checking for a form update submission and it loops over the tasks array, setting each to incomplete. After that's done, it will then loop over the submitted tasks (using the checkbox ids) and set those tasks to complete.

```
if ( form.keyExists('update_tasks') ) {
strTasks = "";
for( task in application.tasks ) {
task.setComplete(false);
if ( structKeyExists(form, 'task') ) {
if (listContains(form.task, task.getId())) {
task.setComplete(true);
}
strTasks = form.task;
}
}
application.completeTasks
= strTasks;
}
```

15. Clear completed tasks

If the user submits the button now to clear the completed tasks, the conditional statement will check for the existence of the clearcomplete value that was sent in the URL. This will then loop over all tasks and, once it has found the matching task using the ID, it will go on to delete it from the array before sending the user back to the index page.

```
if ( structKeyExists(url, 'clearcomplete') )
{
for( task_id in listToArray

(application.completeTasks) ) {
for( task in application.tasks ) {
if ( task.getId() EQ task_id ) {
arrayDelete(application.tasks, task);
}
}
}
application.completeTasks = "";
location('/app', false);
}
```

16. Creating REST resources

The great thing about Lucee is that it offers an incredibly quick and easy way to create and generate RESTful resources, written using the familiar component structure. You will need to create a new directory called 'rest' within your project, inside of which we will place all of our components. Create a file called 'random.lucee' with the following code. The rest path attribute dictates the URL that this component will respond to when we call the API.

```
component restpath="/random" rest="true" {
}
```

17. Initial GET Request

Now we will continue by defining the routes and responses for each API resource within the open component block. First let's create a new remotely accessible function that responds to a GET request. This simple example returns a random number nested inside of a JSON response. The function name has no direct impact on the resource, but ideally it should relate to the purpose of the function.

```
remote any function getRandomNumber()
httpmethod="GET" {
return {'random_number':
randRange
(1,1000000) };
}
```

18. Manage URL parameters

Create a new GET resource function in the component to accept a single argument or parameter. The restargsource attribute dictates that the argument will be part of the path and we define the argument value as sent_value. This response uses CFML's built-in dateTimeFormat() method to handle formatting the current time returned from the now() function.

```
remote any function returnResponse(string
sent_value restargsource="Path")
httpmethod="GET"
restpath="{a}"
{
return {
'request_made':
dateTimeFormat(now(),
'dd/mm/yyyy HH:NN'),
'value': arguments.a
};
}
```

19. Handle POST requests

Create another remote method to handle our POST requests. The restargsource attribute will be set to Form, expecting the parameter posted_value in that submission. return the submitted value, the current datetime value and the entire arguments scope, which could be used for API response debugging.

```
remote any function postResponse(string
posted_value restargsource="Form")
httpmethod=
"POST" {
return {
'request_made': dateTimeFormat(now(), 'dd/
mm/yyyy HH:NN'),
'you_submitted':
arguments.posted_value,
'form_data':
arguments
}
}
```

20. Admin configuration

The REST resources need to be defined and mapped in the Lucee administrator. Log in to the web content admin and select Archives & Resources>Rest. Create a new mapping virtual name (use api for brevity) and point the physical to the directory location of the components.

21. Test POST submission

Create a test lucee file in the project root to test the API. Create a new script block and make an HTTP POST request to the local API address. At least one of the http param lines, set as a form field, should be the value expected by the method. Finally dump out the cfhttp result for visual debugging.

```
<:script>
http url="http://localhost:8888/rest/api/
random/" method="POST"
{
http param name="posted_value" value="Lucee
is powerful" type="formField";
}
dump(cfhttp);
</:script>
```

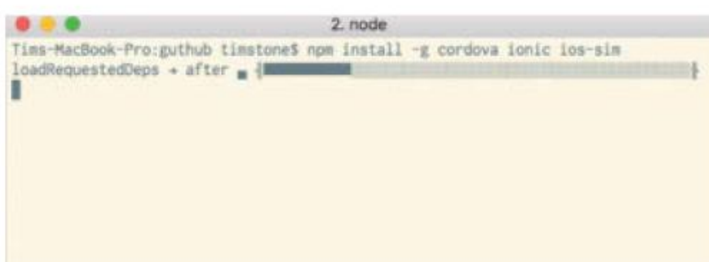
Use Ionic to create a native app

Using Angular and web technologies you'll create a hybrid app for iOS and the web.

Ever wanted to develop a native application but have been put off by learning a new language and ecosystem? Enter the Ionic framework which allows you to apply your JavaScript knowledge to build hybrid apps. Ionic offer other services to build APIs, a WYSIWYG editor, push notification handling, analytics, and much more. We'll be concentrating on just building with the core framework. Ionic is fast but still not as performant as building natively; this makes it very well suited for prototyping or delivering an MVP.

Ionic was released in 2013 by a company called Drifty. Since then it has seen increasing adoption due to its ease-of-use and modern toolchain. It also looks gorgeous out-the-box and isn't opinionated about the editor you use like some similar mobile frameworks. Ionic is powered by Angular so I'll assume some basic working knowledge of Angular so we can focus on what Ionic brings to the table.

To test Ionic out you're going to build a simple app which will list recipes from the Food2Fork API and let you select your favourites which will be displayed in a separate view. You'll learn the basics to get Ionic running and we'll touch upon some of the more advanced features it exposes to take these skills further.



1. Install Ionic's dependencies

Ionic uses a modern workflow so you'll need Node.js 4 installed. If you're on Windows the Visual Studio Community IDE includes everything you need to get started. On a Mac you'll need to install Xcode from the App Store. For Android development follow **http://bit.ly/1Tt3tDF**.

```
$ npm install -g cordova ionic ios-sim
```

2. Scaffold the project

Through the command-line we use Ionic to create a new project called 'guthub' which adds all the necessary files and folders for an app. We then add iOS as a build target then run it. This kicks off building the application and running it in the iOS Simulator on OS X.

```
$ ionic start guthub
$ cd guthub
$ ionic platform add ios
$ ionic run
```

3. Script dependencies

In index.html add the script references for our JavaScript files. Only app.js exists currently so you'll get some 404 errors but it saves having to keep adding files later on. Just like a normal site, index.html is the starting point and includes all the setup needed. Change ng-app on body to guthub.

```
<script src="js/app.js"></script>
<script src="js/controllers/list.js"></
script>
<script src="js/controllers/favourites.
js"></script>
<script src="js/controllers/recipe-detail.
js"></script>
<script src="js/services/recipes.js"></
script>
<script src="js/services/favourites.js"></
script>
<script src="js/services/localstorage.js"></
script>
```

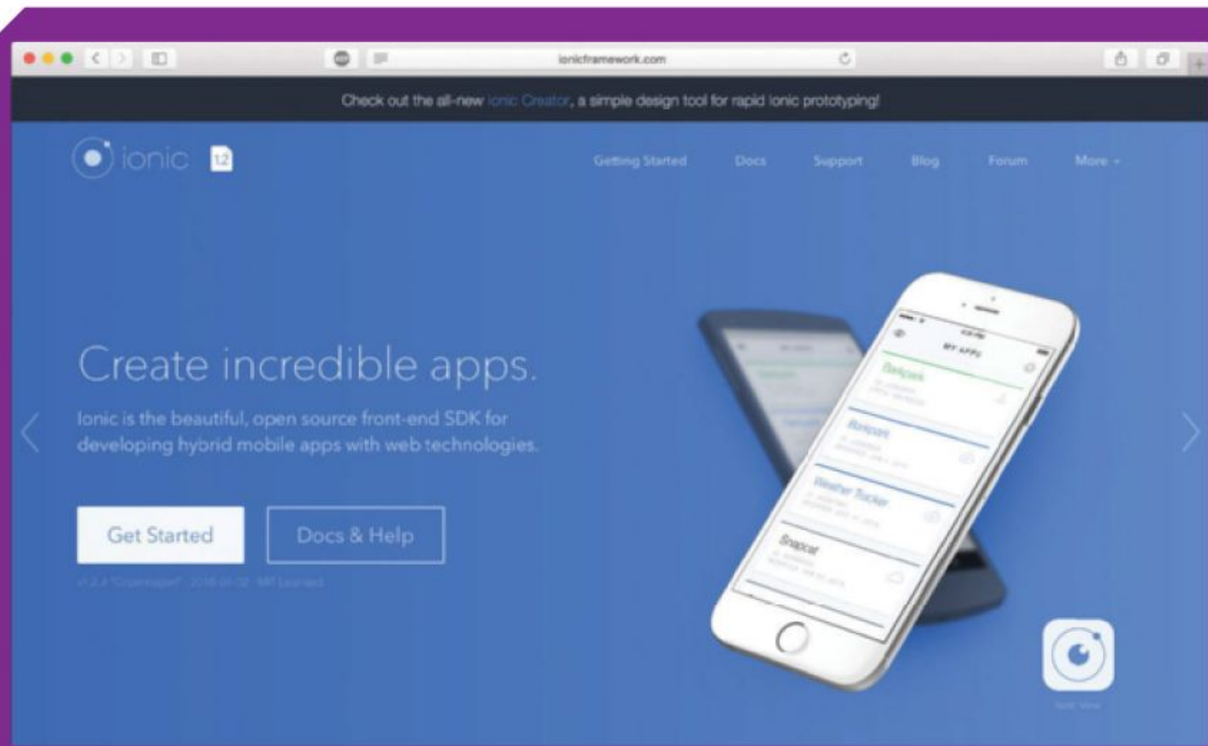
4. Create the recipe service

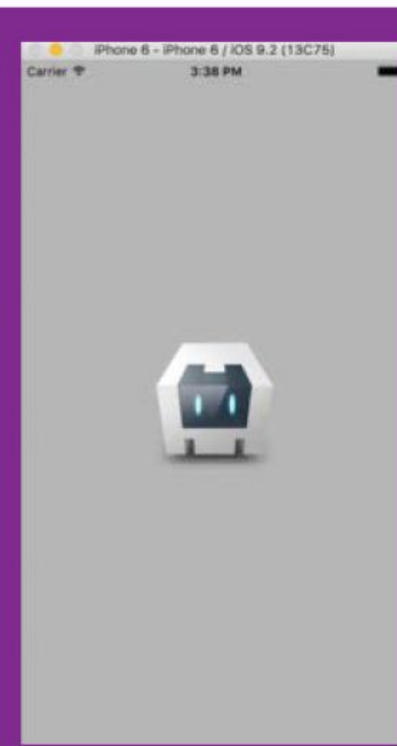
First off we will want some recipes to display in our app, so start by creating a services folder and a file called recipes.js. You can grab the recipe array from **http://bit.ly/1KMI4Pz** and paste it in. We'll mimic an API by exposing a simple get function. You'll also need to add ngResource as a script dependency.

```
<script src="lib/ionic/js/angular/angular-
resource.min.js"></script>
angular.module('guthub.services',
['ngResource', 'ionic.utils'])
.factory('Recipes', function ($resource) {
  return {
    get: function (id) {
      // next step
    }
  };
```

Learn more about the Ionic framework

Ionic has many aspects to it: a CSS framework, icons, directives and more. **http://learn.ionicframework.com** includes videos and guides for different learning styles.





<Left>
• The Ionic website provides you with all the resources you need to get started with Ionic including other services which Drifty provide.

<Centre>
• Building the app invokes xcodebuild without having to separately open Xcode straight from the command-line. It includes live reload within the simulator.

<Right>
• The resources are all customisable and can be built for each platform with $ ionic resource. The splash screen defaults to the Cordova logo.

```
});
```

5. Get recipe function

If the get function is passed an ID then it will loop through the recipes until it finds a match and return it. If nothing is found then null is returned. If no ID is passed as an argument then it will return all of the recipes.

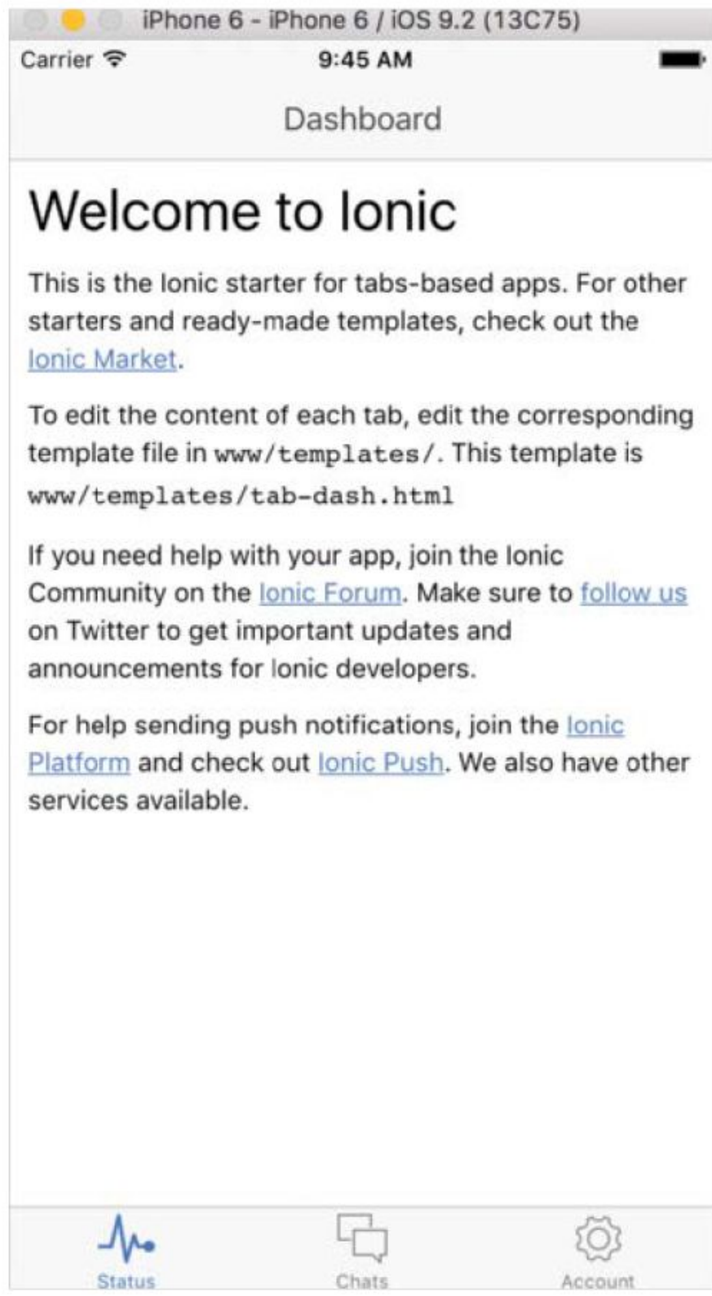
```
if (id) {
  for (var i = 0; i < recipes.length; i++)
{
    if (recipes[i].recipe_id == id) {
      return recipes[i];
    }
  }
  return null;
} else {
  return recipes;
}
```

6. Adding tabs to the view

In the templates folder open tabs.html. We are going to be tweaking it slightly to show two tabs with different icons. icon-off is the icon to show when the tab isn't selected and on is for when it is. Note that we're sending it to a route defined in app.js. The nav-view name links to the template.

Sharing and viewing

Ionic includes an iOS app called Ionic View which will allow you to upload your app to Ionic's cloud service and then view it on any device of your choosing. You can even share it via the CLI to someone else's email address.




```
<ion-tabs class="tabs-icon-top tabs-color-
active-positive">
  <ion-tab title="Recipes" icon-off="ion-
ios-list-outline" icon-on="ion-ios-list"
href="#/tab/recipes">
    <ion-nav-view name="tab-recipes"></
ion-nav-view>
  </ion-tab>
  <ion-tab title="Favourites" icon-off="ion-
ios-star-outline" icon-on="ion-ios-star"
href="#/tab/favourites">
    <ion-nav-view name="tab-favourites"></
ion-nav-view>
  </ion-tab>
</ion-tabs>
```

7. Create recipe list view

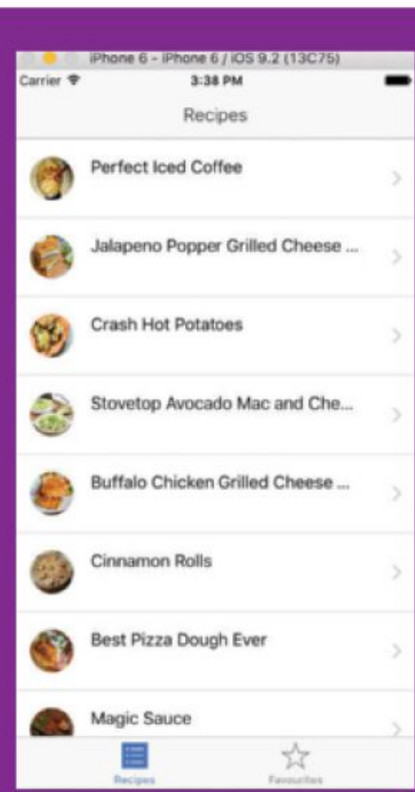
Create a new template file called tab-recipes.html. Each of these custom tags are Angular directives documented at **http://ionicframework.com/docs/api/directive**. A view creates a new header/footer, content creates a content area with customisable scrolling and the list, well, creates a list.

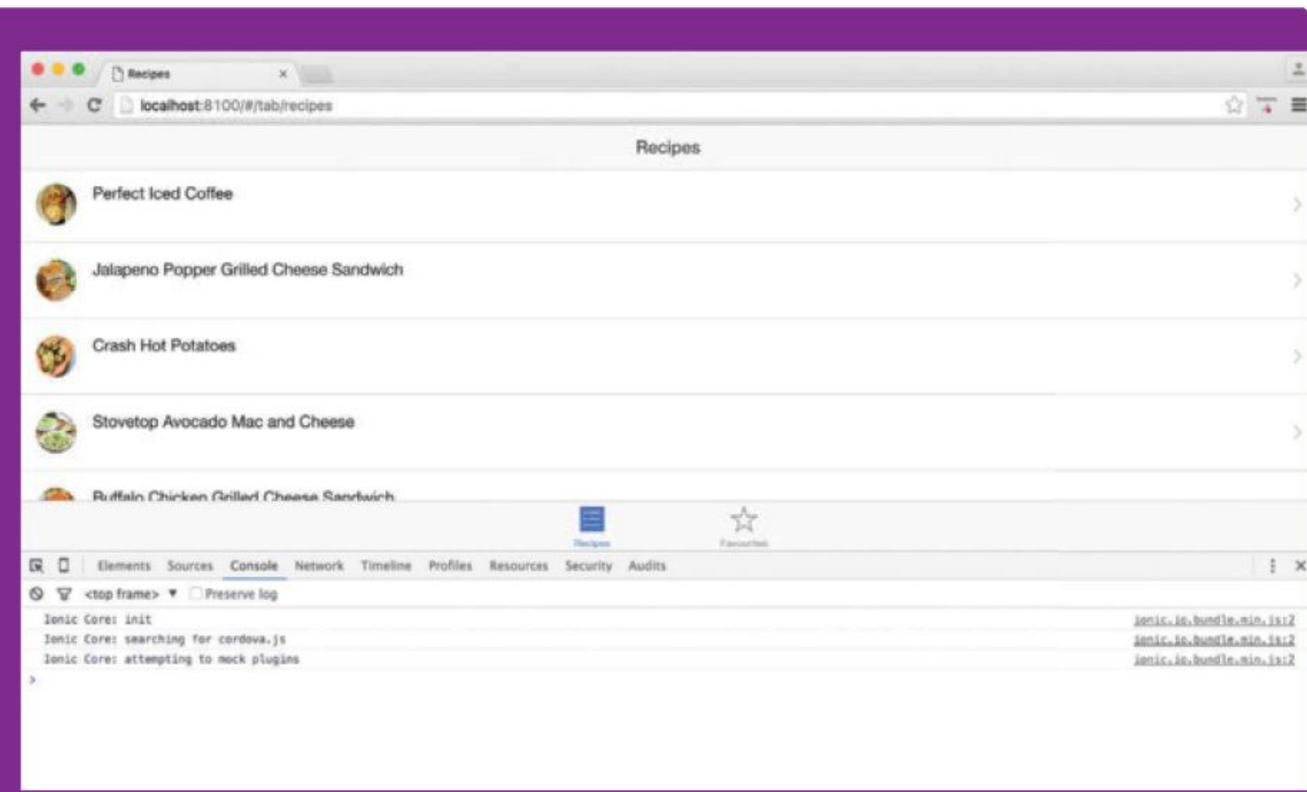
```
<ion-view view-title="Recipes">
  <ion-content>
    <ion-list>
        <!  next step  >
    </ion-list>
  </ion-content>
</ion-view>
```

8. Adding items to the list

To add items to this list use the ion-item directive. Some built-in Ionic classes are used to effectively style the list. Using ng-repeat we'll easily repeat this markup for each recipe in the array.

```
<ion-item class="item-remove-animate
item-avatar item-icon-right" ng-
repeat="recipe in recipes" type="item-text-
wrap" href="#/tab/recipes/{{recipe.recipe_
```

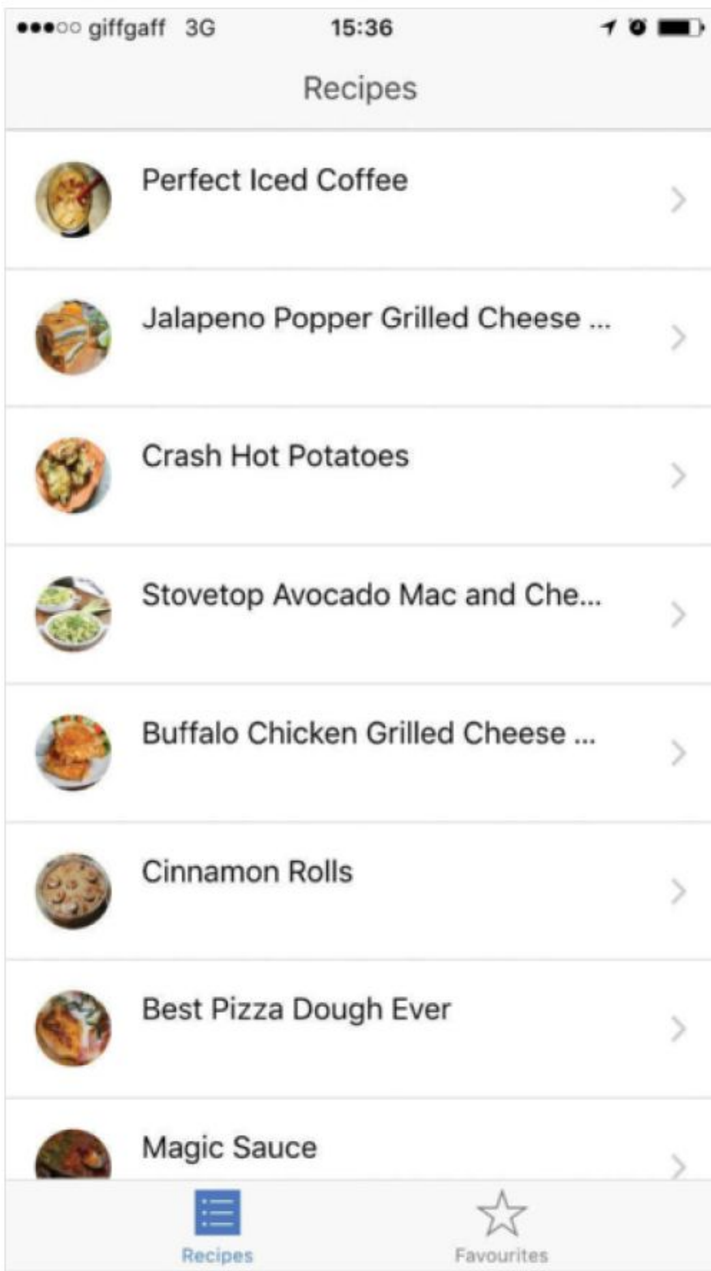


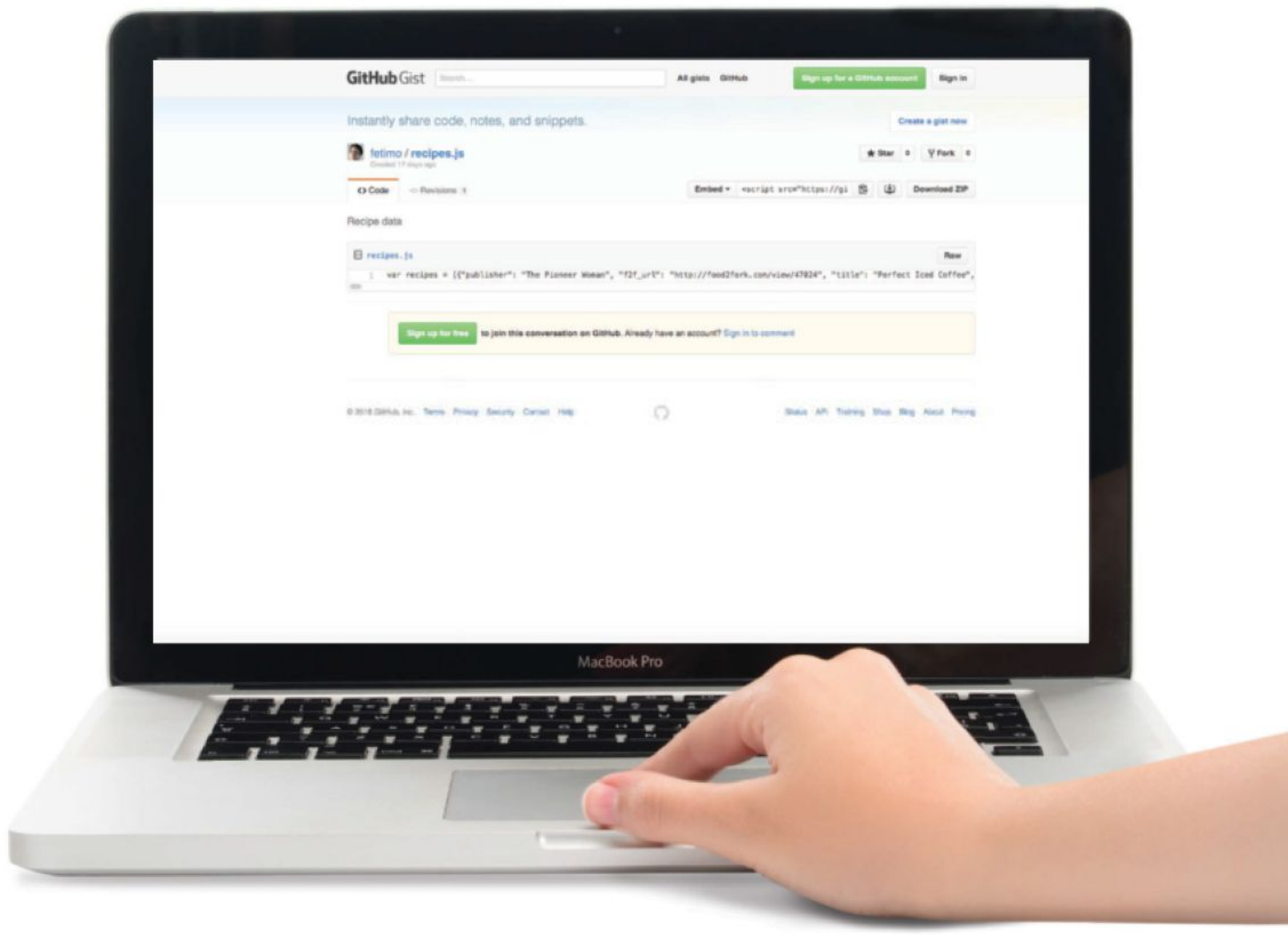


<Left>
•The iOS Simulator will allow you to mimic various conditions like a very low memory or, for instance, shaking the device.

<Centre>
•As good as the simulator is there's nothing like holding your creation in the palm of your hand which you can do with Ionic View.

<Right>
•We're just using web technologies so exactly the same code can be run within the browser for easy debugging. You can deploy code specific to each platform.





> Create a services folder and a file called recipes.js. You can grab the recipe array from http://bit.ly/1KMl4Pz and paste it in

```
id}}">
  <img ng-src="{{recipe.image_url}}">
  <h2>{{recipe.title}}</h2>
  <i class="icon ion-chevron-right icon-
accessory"></i>
</ion-item>
```

9. Create the Recipes controller

In app.js, update the route name and controller to match. Then create a folder called controllers and within it a file called recipes-list.js. We'll inject the Recipes service from earlier and use the get method to fetch all the recipes. It's assigned to the scope so they'll be exposed to the view.

```
angular.module('guthub.controllers',
['guthub.services'])
  .controller('RecipesCtrl', function
($scope, Recipes) {
    $scope.recipes = Recipes.get();
  });
```

10. Recipe detail view

Now we'll create the view that actually lists the data about the recipe and the ability to add it to our favourites. Note that we can dynamically set the view's title using an Angular expression. Ionic includes some utility classes for common tasks like adding padding.

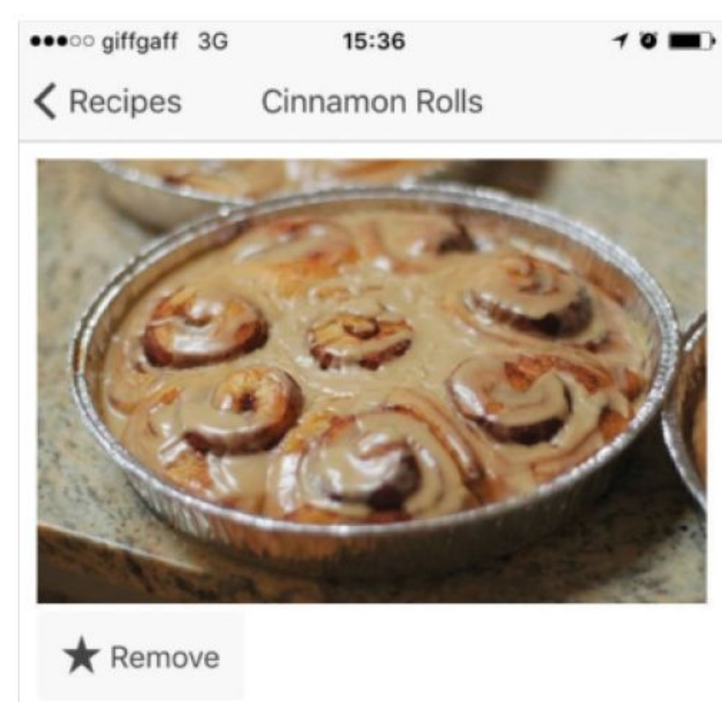
```
<ion-view view-title="{{recipe.title}}">
  <ion-content class="padding">
    <img class="recipe-image" ng-
src="{{recipe.image_url}}">
```

```
    <!  next step  >
  </ion-content>
</ion-view>
```

11. Favourite buttons

You still can't favourite the recipe so let's add a couple of buttons. Again, Ionic includes the button classes which perfectly style it on every platform. On click we'll either add or remove the recipe and toggle which button is shown with ng-show.

```
<button class="button icon-left ion-star
button-positive" ng-
click="addFavourite(recipe)" ng-
```



```
show="!recipe.addedToFavourites">Add</
button>
<button class="button icon-left ion-star
button-negative" ng-
click="removeFavourite(recipe)" ng-
show="recipe.addedToFavourites">Remove</
button>
```

12. The recipe detail controller

When you navigate to a URL we can add state parameters to pass specific data through. We'll use this to get the ID of the recipe we're looking for and add that to the scope. If it's already in the favourites then we'll add the flag to show this.

```
angular.module('guthub.controllers')
  .controller('RecipeDetailCtrl', function
($scope, $stateParams, Recipes, Favourites)
{
    $scope.recipe = Recipes.
get($stateParams.recipeId);
    if (Favourites.
get($scope.recipe.recipe_id)) {
$scope.recipe.addedToFavourites = true;
}
});
```

13. Add/remove favourites

To add the favourite we call Favourites.add. This won't work yet as we need to write the Favourites service. To remove the recipe from your favourites we do the opposite. We could just expose the service to the view but this gives us more granular control.

```
$scope.addFavourite = function () {
Favourites.add($scope.recipe);
};
$scope.removeFavourite = function () {
Favourites.remove($scope.recipe);
};
```

14. Ionic formulas

Ionic has a number of formulas, we'll use the localStorage formula (see **http://learn.ionicframework.com/formulas/localstorage/**). This could even be easily extended to use iCloud Backup to persist data across devices.

```
angular.module('ionic.utils', [])
.factory('$localstorage',
['$window', function($window) {
return {
// next step
};
}]);
```

15. Set and get storage

The local storage service is really just an abstraction for setting and getting named local storage strings.

```
set: function (key,
value) {
  $window.
localStorage[key] =
value;
},
get: function (key,
defaultValue) {
  return $window.
localStorage[key] ||
defaultValue;
}
```

16. Create the favourites service

The favourites service will handle adding, setting, removing, and finding favourites. It starts by seeing if there are previously stored favourites in local storage. If no previous favourites are found then favourites is just an empty array.

```
angular.module('guthub.services')
.factory('Favourites', function
($localstorage) {
  var stored = $localstorage.
get('favourites');
  var favourites = stored ? JSON.
parse(stored) : [];
});
```

17. Add/remove favourites

The add function takes a recipe, sets the addedToFavourites boolean to true and pushes that to the favourites array. We then use the local storage service to set the favourites string by stringifying the array; this preserves the content of objects.

```
return {
  add: function (recipe) {
    recipe.addedToFavourites = true;
    favourites.push(recipe);
    $localstorage.set('favourites', JSON.
stringify(favourites));
    return recipe;
  },
};
```

18. Remove from favourites

The remove method is similar to the add one except it splices the recipe from the array. It uses indexOf to find where in the recipe the matching recipe is. Splice is a 'destructive' method in that it changes the original array.

```
remove: function (recipe) {
  favourites.splice(favourites.
indexOf(recipe), 1);
  recipe.addedToFavourites = false;
  $localstorage.set('favourites', JSON.strin
gify(favourites));
  return recipe;
}
```

19. The Favourites controller

The controller for favourites will simply get the Favourites (using a very similar method to fetching recipes). From this view we'll use a special button to remove from the favourites array directly from the list. Remember to update app.js with the favourite controller!

> The add function takes a recipe, sets the addedToFavourites boolean to true and pushes that to the favourites array.

Using the device's hardware

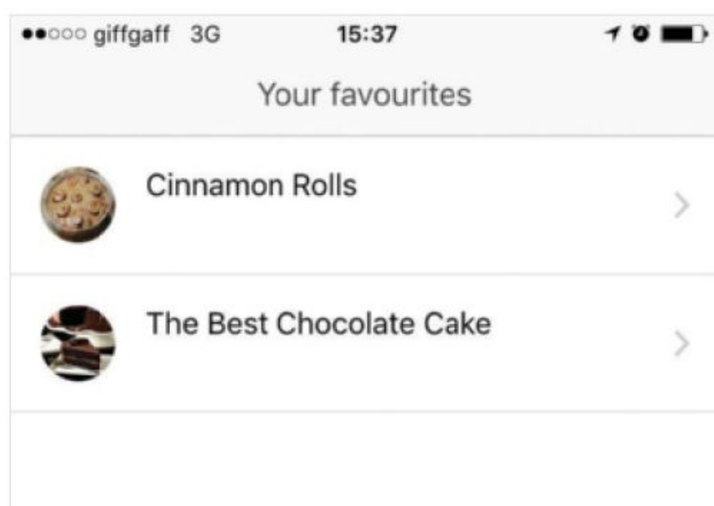
Ionic uses Cordova to hook into the operating system's native functionality. Ionic created ngCordova, a collection of over 70 directives which can be included to make this easier to work with Angular. It includes everything from Touch ID integration to accessing the camera. **http://ngcordova.com/docs/plugins/** has a list of plugins and how to use them. For example, if we wanted to include vibration feedback to an action we would add the module via the command-line and simply inject the service into our controller. Some of these APIs are platform-specific but there is general support for iOS, Android, and Windows Phone.

```
angular.module('guthub.controllers')
.controller('FavouritesCtrl', function
($scope, Favourites) {
  $scope.favourites = Favourites.get();
  $scope.removeFavourite = function (recipe)
{
    Favourites.remove(recipe);
  };
});
```

20. Listing favourites

To list favourites we'll use a similar view to listing recipes for consistency. Note the use of ng-src instead of simply setting the src attribute. This prevents requests to {{recipe.image_url}} which would 404 and be a wasted network request.

```
<ion-view view-title="Your favourites">
  <ion-content>
    <ion-list>
      <ion-item class="item-remove-animate
item-avatar item-icon-right" ng-
```

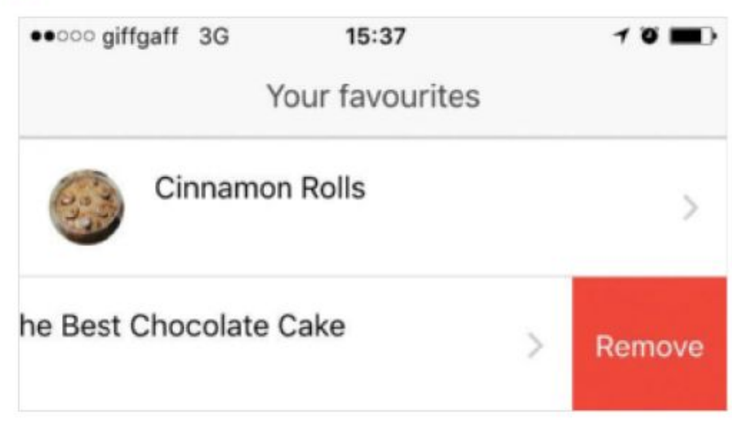



```
repeat="recipe in favourites" type="item-
text-wrap" href="#/tab/recipes/{{recipe.
recipe_id}}">
        <img ng-src="{{recipe.image_url}}">
        <h2>{{recipe.title}}</h2>
      </ion-item>
    </ion-list>
  </ion-content>
</ion-view>
```

21. Removing from the list

Ionic allows for some advanced gestures really easily. The ion-option-button directive will reveal a button when you slide to the left on an ion-item. We'll use this to create a shortcut to the remove favourite action.

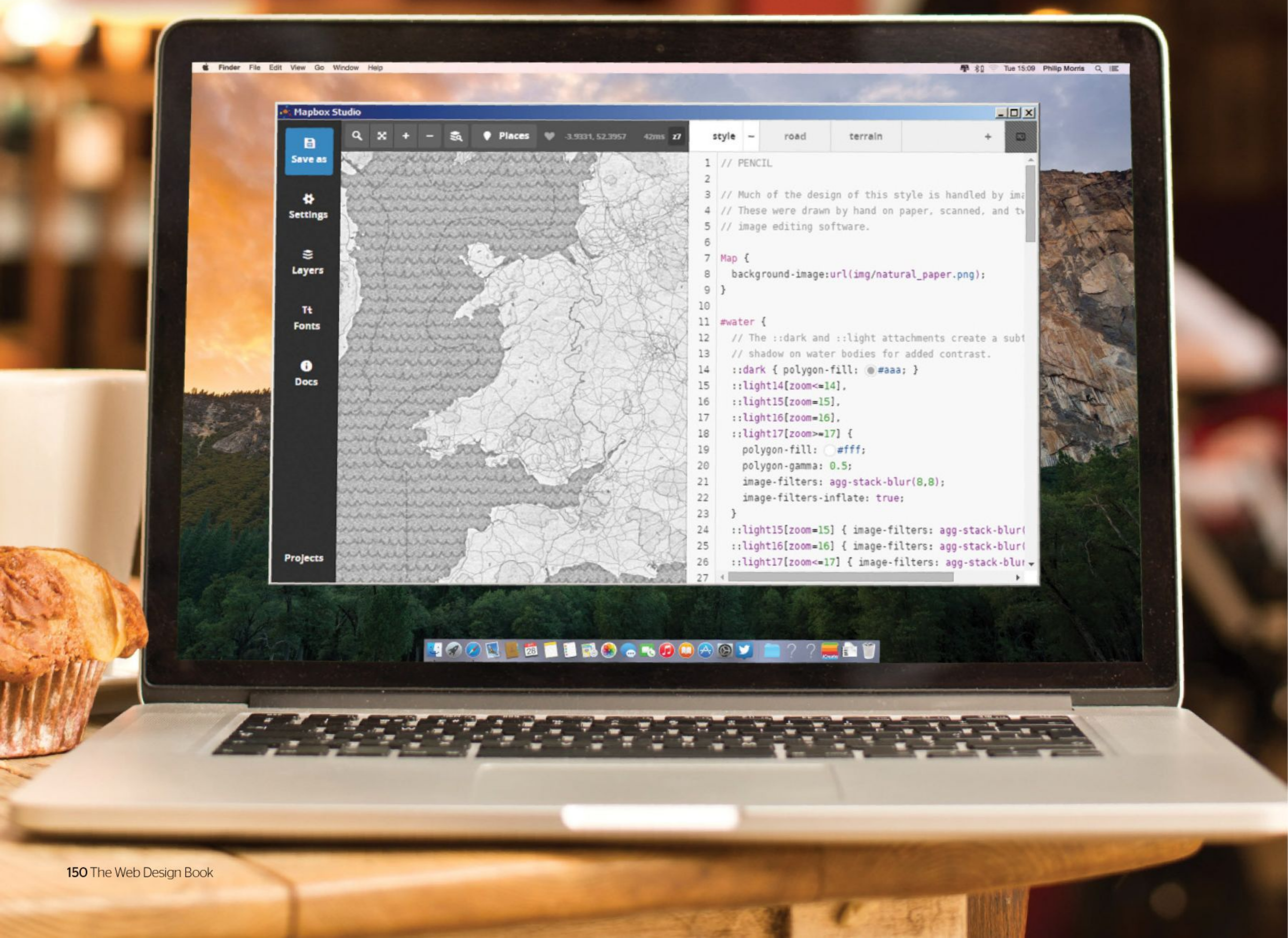
```
<i class="icon ion-chevron-right icon-
accessory"></i>
<ion-option-button class="button-assertive"
ng-click="removeFavourite(recipe)">
Remove
</ion-option-button>
```



DOWNLOAD TUTORIAL FILES
www.filesilo.co.uk/bks-894

Create desktop applications with Electron

Use pseudo elements HTML, CSS and JavaScript to create cross-platform desktop applications



GitHub have created an open source project called Electron (formerly Atom-Shell) that combines Chromium, on which Google's Chrome is built, and Node.js, a JavaScript environment designed for building applications. The result enables the building of cross-platform desktop applications using HTML, CSS and JavaScript. Companies such as Microsoft, Facebook, Slack and more have started to make use of Electron.

Electron applications normally start out with a single JavaScript file which is executed initially. The code within it has control over the main process, can create and control new application windows and listen for important events. Each window that is created can be set to render your UI through the use of HTML, CSS and JavaScript. The JavaScript executed within a window can also access special Electron APIs and the Node.js API.

The Chromium part of Electron is also kept up to date with the latest releases, enabling use of the latest HTML/CSS features and emerging JavaScript standards. The Node.js environment is also well maintained, to the point where it's currently using a fork of Node.js called io.js, which is capable of supporting ES6 JavaScript syntaxes and can make use of npm registry modules (**npmjs.com**).

1. Prebuilt Electron binaries

Compiling Electron from scratch isn't required for building applications. The project makes available precompiled versions of Electron for multiple operating systems, which can be easily installed using Node.js' npm package manager (available via **nodejs.org**).

```
$ npm install -g electron-prebuilt
```

2. Test Electron

Electron should now be available on your system. This can be confirmed by running the 'electron' command, which will display a standard message for the application – just ignore this message for now and close the window.

```
$ electron
```

3. Use package.json

Package.json is required for Electron to know which file to execute for the application. It will also store the name of your application and your version, and 'npm' will add the details of any modules that you install and use. In this tutorial, the application will be used for editing Markdown-based files.

```
{
"name": "wdMarkdown",
"main": "index.js",
"version": "0.1.0"
}
```

4. The application and index.js

The index.js file is the core of the application and will create our first application window to display the application. Depending on the type of application being developed, this file will usually contain the core of your application and do all the heavy lifting required. By using the 'app' object, we can find out when Electron is ready, and a browser window (appearing as a normal application window) can then be created.

```
var app = require('app');
var BrowserWindow = require('browser-
window');
var window = null;
app.on('ready', function() {
window = new BrowserWindow({width: 800,
height: 600});
window.loadUrl('file://' + __dirname + '/
root/index.html');
window.on('closed', function() {
window = null;
});
});
```

5. Test the app

To enable us to test the application so far, a HTML file can be created containing some simple HTML for now. After adding the code below, execute the command 'electron ./' within the same directory as the package.json file created earlier.

```
<html><body>
<h1>Hello World, from Electron</h1>
</body></html>
```

6. Install Bower

Our application has a few dependencies for the UI that will be constructed. By installing Bower, these dependencies can be easily retrieved along with anything else they may require and Bower could be used to keep them up to date.

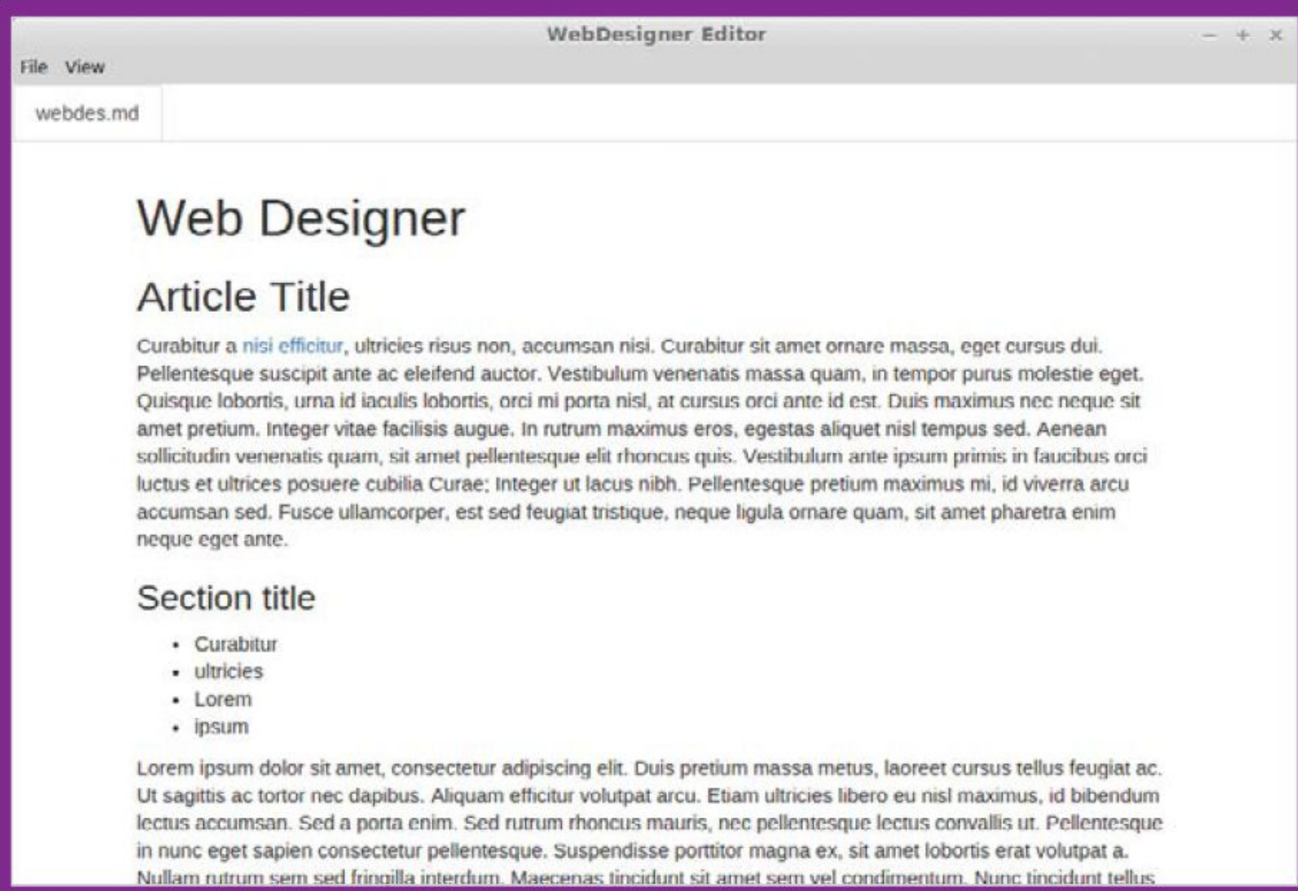
```
$ npm install -g bower
```

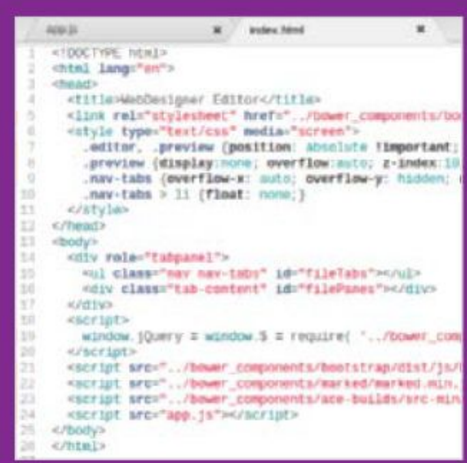
7. Bootstrap framework

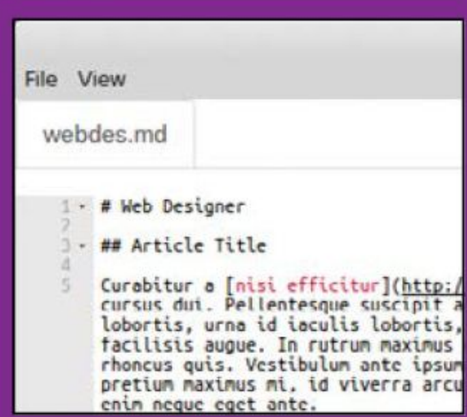
In order to have something to display, a UI needs to be created using HTML and CSS. As a base theme,

Desktop applications

For fuller and more complex applications, consider using npm modules (more information is available at **npmjs.com**) for databases, network communications and more, plus a framework for your UI from the likes of React, AngularJS, Backbone.







<Top left>
• Here is the final UI HTML page. Its primary purpose is to include all the JavaScript files needed to construct UI, Bootstrap for the appearance and our main JavaScript file

<Bottom left>
• Here is what the application will look like when editing a Markdown document. The tab shows the title of the file, and the editor contains the raw Markdown content

<Left>
• By using the view menu we can have the editor switch between viewing the original Markdown content and a rendered HTML version of the content as a preview

Bootstrap can be useful as it provides many useful components and styles. More advanced applications may take advantage of AngularJS or React as well. By using Bower to retrieve a copy of Bootstrap we'll also have a copy of jQuery included as it's required.

```
$ bower install bootstrap
```

8. Markdown rendering

The app will be used for editing Markdown files. Markdown is a widely used markup language used for writing formatted text in a plain text file. It can then be processed by a library such as 'Marked' into HTML.

```
$ bower install marked
```

9. Edit Markdown with Ace

The Ace editor project is a feature-packed code editor that's written in JavaScript and can be used within web pages. For this tutorial, we will be utilising Ace editor for editing the Markdown files as it can provide us with things like line numbering, syntax highlighting and other very useful features.

```
$ bower install ace-builds
```

10. Include Bootstrap and JS

Next, make use of Bootstrap, Ace editor and Marked by including them in our index.html file. Make note of the alternative way jQuery is included, which is due to an issue with jQuery being unsure what kind of environment it is operating within when inside Electron.

11. Styling

The Markdown editor being constructed will provide an edit area for altering the content and a preview of the HTML generated. As the app will be kept simple, the editor and preview areas will fill the application, minus an area for tabs to switch between files.

```
.editor, .preview {position: absolute
!important; top: 58px; right: 0; bottom: 0;
left: 0; z-index:5;}
.preview {display:none; overflow:auto;
z-index:10; background-color: white;}
```

12. Make the tabs

Bootstrap provides the styling and JavaScript required to provide the tabbed interface used for the application. Each tab will represent a file and clicking between them will display a related editor or preview. For now, we only need very basic HTML as we'll build the tabs and related panes using JavaScript.

```
<div role="tabpanel">
<ul class="nav nav-tabs" id="fileTabs"></ul>
<div class="tab-content" id="filePanes"></
div>
</div>
```

13. root/app.js

This file will hold our main UI code. Using the ipc module, it can subscribe and publish events for communication with the main JavaScript process, that Electron executes (./index.js). We'll need to identify files edited and store additional information, plus references to the tab and pane areas of the UI. Also, we will need a small function to grab the ID of the currently active tab.

```
var ipc = require('ipc');
var files = {}, rollingId = 0;
var tabs = $('#fileTabs'), panes =
$('#filePanes');
var ActiveTab = function() {
return id = tabs.find('.active a').
attr('href').slice(1);
};
```

14. FileNew function

To create a file or handle opening an existing one, our UI needs to create a new tab and pane, as well as set a relevant title and content. We will use FileNew for this.

15. Use Ace

With the tab, pane and content now ready, it's time to fire Ace editor into action. Then we can store some information for easier referencing and switch to the newly created tab.

16. Menu events

Now we execute the FileNew function once, present a new file tab on opening, and then listen for menu events sent from the main application process. The menu events, which we'll create soon, are for creating a new file and opening an existing one.

```
FileNew();
ipc.on('file-new', FileNew); ipc.on('file-
open', FileNew);
```

17. Menus

At this point, our Electron app should start up and present a fresh new tab to edit content in. However, we can't open any more or save anything we create. So in this case, a menu is required and it'll be built within the main ./index.js file. The 'fs' and 'path' modules from Node.

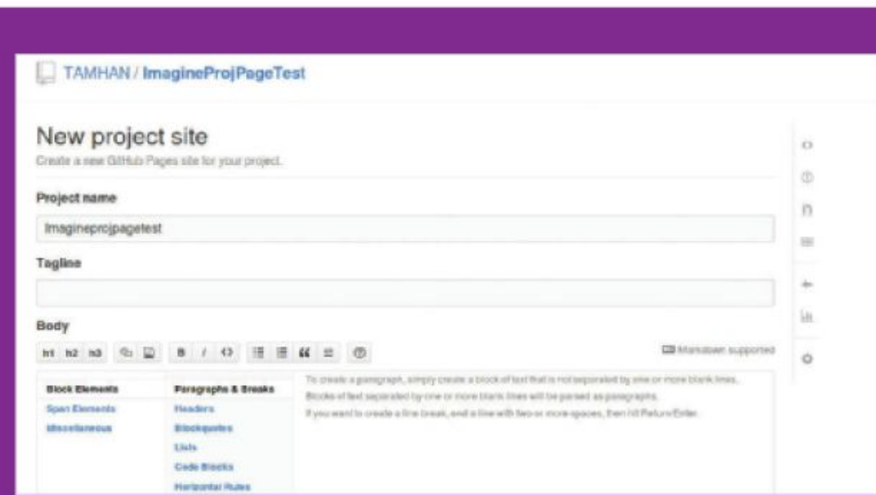
Debugging Electron

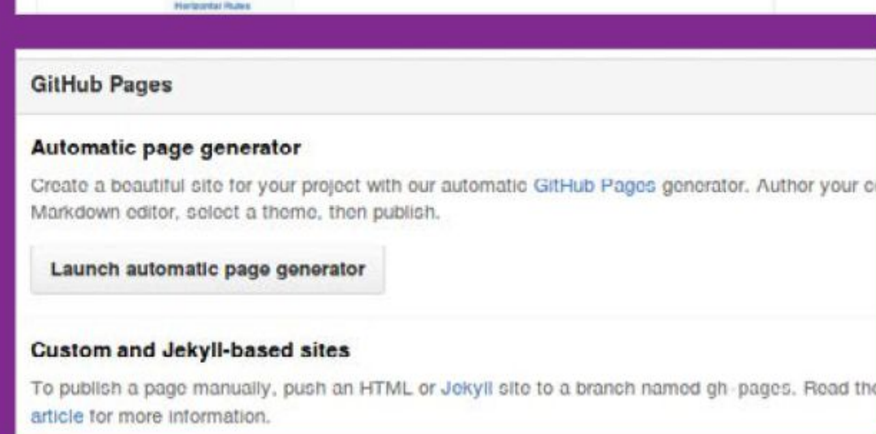
To debug any window you have created using Electron, call the .openDevTools() method on the instance of that window. This will open a developer tools window.

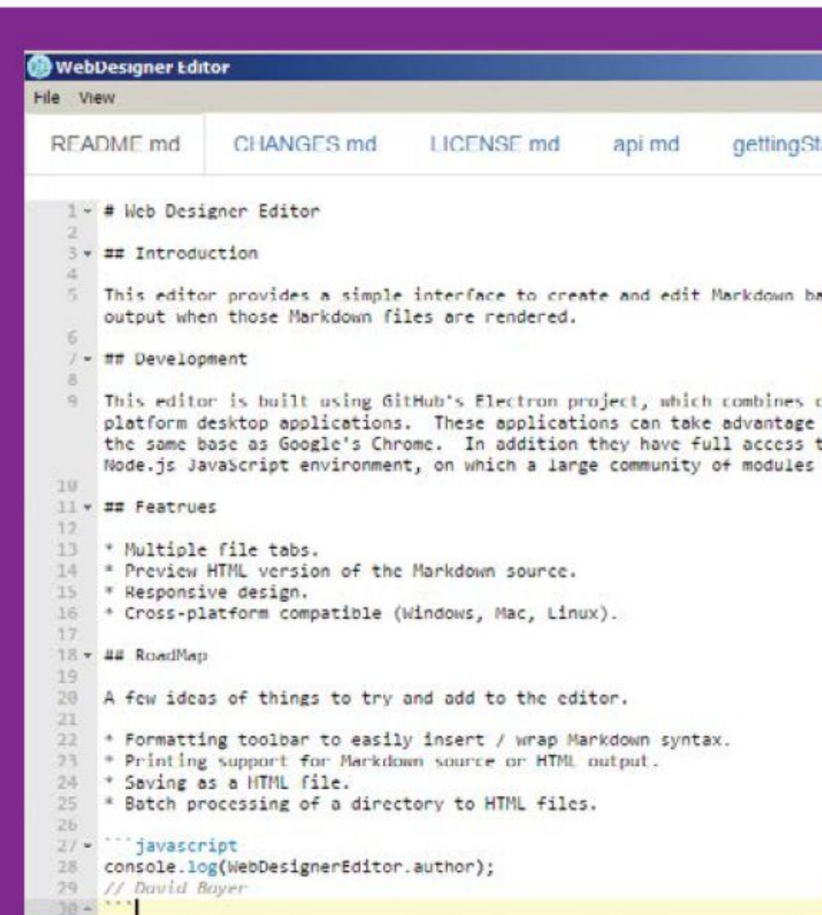
<Top left>
• Microsoft have recently released their own editor, built on top of Electron, called Visual Studio Code

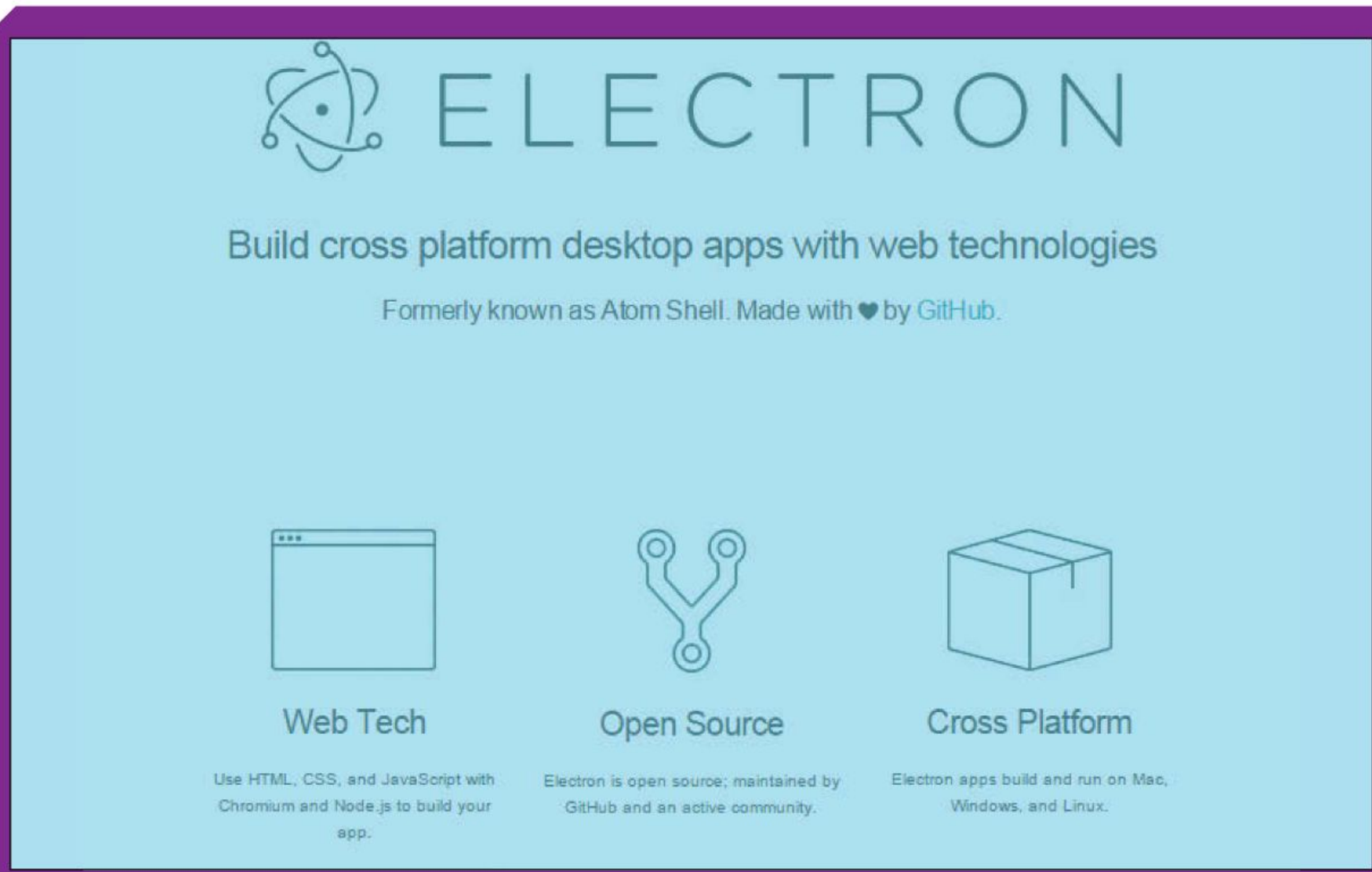
<Bottom left>
• Atom is GitHub's programming editor and is where Electron started out as part of the project

<Right>
• The final finished version of the editor, showing the multiple tabs and some Markdown formatted content









Packaging applications

Through Electron's 'crash-reporter' module, you can configure your application to let users submit crashes to a website under your control. Your application code can also be archived into a more protective ASAR file, before being distributed. This requires use of the 'asar' command installed through npm; 'npm install -g asar'. Electron currently only supports autoupdating on OS X through its 'auto-updater' module. But there's no reason why you couldn't implement a check against a URL for newer versions of your app and then notify the user. GitHub provides prebuilt versions of Electron at **github.com/atom/electron/releases**. Your app code can combine with these releases and redistributed.

js will be used for handling files. 'Dialog' and 'menu' provide opening/saving dialog boxes and an application menu bar.

```
/* Below 'window = null' */
var fs = require('fs'), path =
require('path');
var ipc = require('ipc');
var dialog = require('dialog'), Menu =
require('menu');
/* Within "app.on('ready')" */
var SendEvent = function(name) {
return function() {window.webContents.
send(name);};
};
var template = [
{label: 'File', submenu: [
{label: 'New', click: SendEvent('file-
new')},
{label: 'Open', click: OpenFile},
{label: 'Save', click: SendEvent('file-
save')},
{label: 'Save As', click: SendEvent('file-
save-as')},
{label: 'Close', click: SendEvent('file-
close')},
{type: 'separator'},
{label: 'Quit', click: function() {app.
quit();}}
]}, {label: 'View', submenu: [
{label: 'HTML/Markdown', click:
SendEvent('view-toggle')}
]}
];
Menu.setApplicationMenu(Menu.
buildFromTemplate(template));
```

18. Open a file

Within the ./index.js file, we can now add a function (already referred to in our menu) to show a dialog (provided by Electron), open the file and pass the data back to our UI for display. At this point, executing our app will provide us with a menu where we can open a new file and have it displayed in our editor.

19. Pass file data

Saving a file requires a few more steps. When the save/save as menu item is clicked, we send an event to the UI. The UI holds the data that we're going to save, so it must listen for those events and send the data back to the main process for us to display a dialog and create/update the related file.

20. Write a file

Our UI is sending a file-save event back to our main process (./index.js), providing the reference ID, content and path (new files don't have one). We'll need to listen for this event, saving the file directly if a path is provided or prompting for a location (save-as) first before creating or overwriting a file with the content.

```
ipc.on('file-save', function(evet, data, id,
type, filepath) {
if (filepath) {
return fs.writeFile(filepath, data,
function(err) {
if (err) return console.error(err);
});
}
dialog.showSaveDialog(window, {
filters: [{
name: 'Markdown', extensions: ['md',
'markdown']
}]
}, function(filepath) {
if (filepath && filepath.length > 0) {
fs.writeFile(filepath, data, function(err) {
if (err) return console.error(err);
window.webContents.send('file-saved', id,
path.basename(filepath), filepath); });
} });
});
```

21. Rendering

Markdown is great for quickly producing text and indicating how it's formatted, but it's more useful when you can see how it looks when it's rendered. Our menu contains an option to switch to a rendered view so our UI needs to listen for it, grab the editor content, pass it through 'marked' for conversion and output it for display.

22. Closure

Now we need to close a tab. Listen for the menu event within the UI, and after that it's just a matter of having the editor clean itself up, and remove the tab and pane elements. Finally, switch to another tab if one exists. For the full code in this tutorial, make sure you go to **filesilo.co.uk/bks-894**.

Studio

156 Ice & Sky

Using design for the greater good, Ice & Sky are aim to raise awareness for climate change

162 DPDK

This studio based in Rotterdam thinks diversity is the key to success – and they're right

168 Cartelle

The Dutch design talent continues with this trendy studio with an all-star client list

174 Phoenix the Creative Studio

The Canadian studio making waves in the industry tell us their secrets

“Amaze was born at the very dawning of the World Wide Web. It recognised the power of the new multimedia”

180 All of Us

This London-based agency tells us what they have learned in over ten years in the business

186 Amaze

Digital marketing is just one of the things Amaze excels at

MAKING AN IMPACT

When it comes to change education is the key. Discover how a creative collaboration are telling the story of climate change

ISTORY
TE CHANGE

Claude Lorius was the first scientist to on our climate. His extraordinary life ures. Wild-Touch and Luc Jacquet invite ic to better understand the incredible hat affect us all.

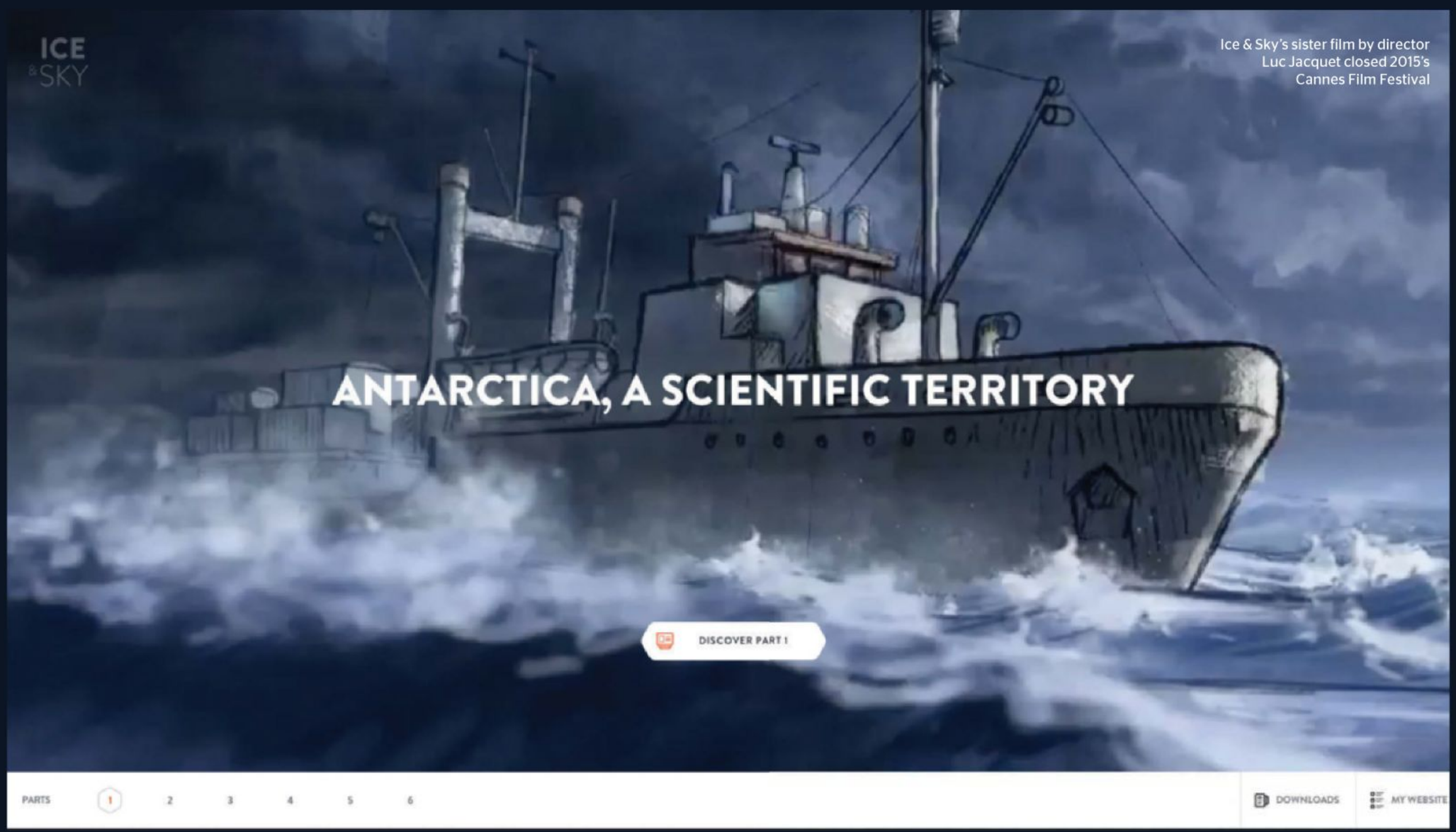
The world is constantly changing, progress is pushing forward, but with progress comes caveats. The serious issue of climate change is one that has been ignored by some, with only the dedicated few bringing the truth to the fore. But, these dedicated few have kept bringing the facts to the table until the outcome could and cannot be ignored.

Education is often the key to making people aware of what's happening outside of their everyday world, and this is very much where the Ice & Sky project makes its mark. It wants to make people aware and therefore it's no surprise that it has taken more than just a single agency to bring the project to life.

The project is led by the Wild-Touch Foundation, created by the film director Luc Jacquet who received an Academy Award for *March of the Penguins*. His ambition was for the Wild-Touch Foundation to be built on two pillars: making scientific knowledge accessible to all through emotion and exceptional imagery.

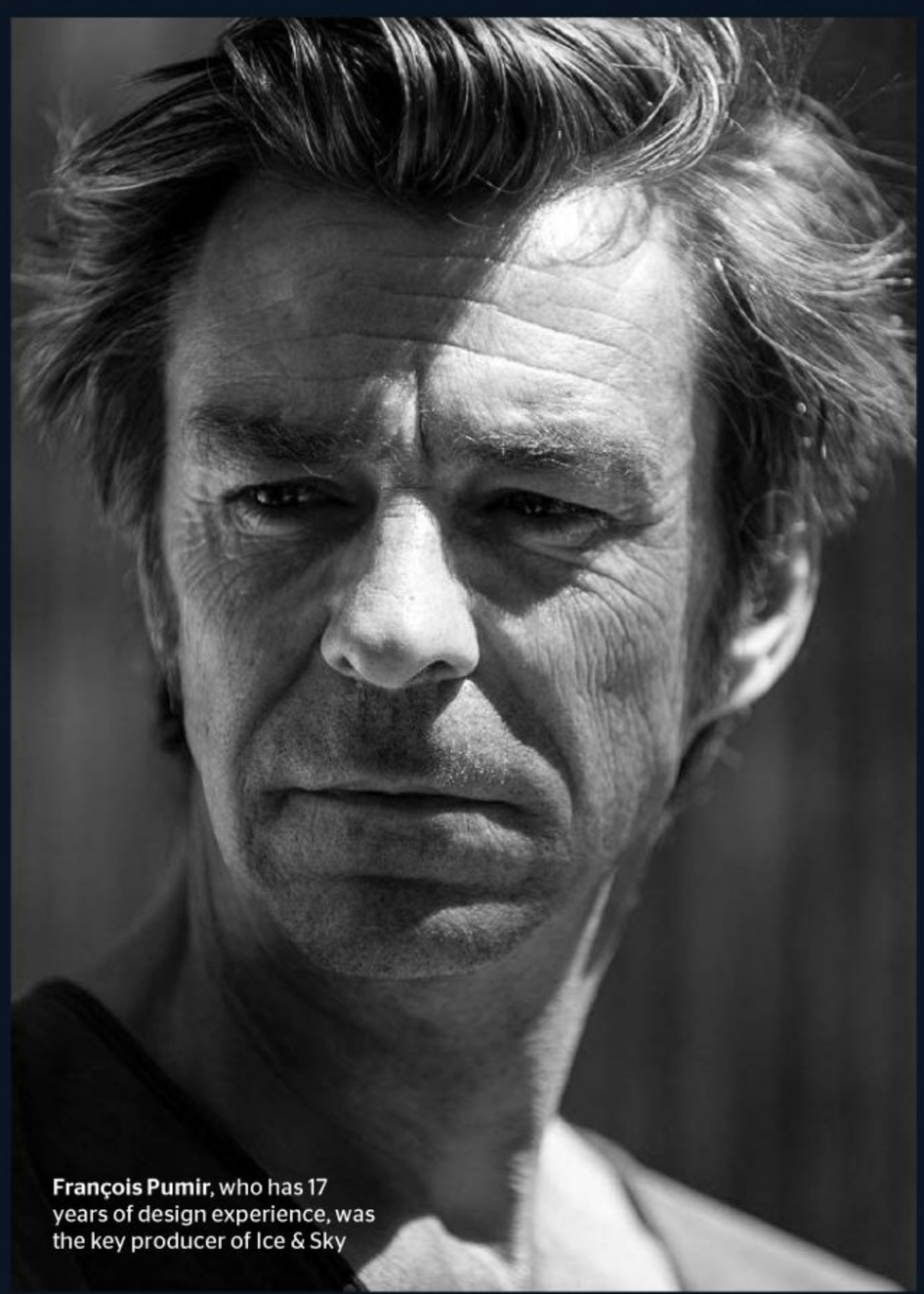
How would they achieve this goal? Dedication to working with renowned international scientific committees. But, how would they bring their work to the

Project	**Ice & Sky**
Web	**education.iceandsky.com**
Agency	**Source, Wild-Touch, François Pumir, Pierre Rossignol**
Web	**bit.ly/1IhtNtJ, bit.ly/1UkQmTK, bit.ly/1IhtPBS, bit.ly/1KNZZsA**
People	**Team of 7**
Time	**9 months in total**



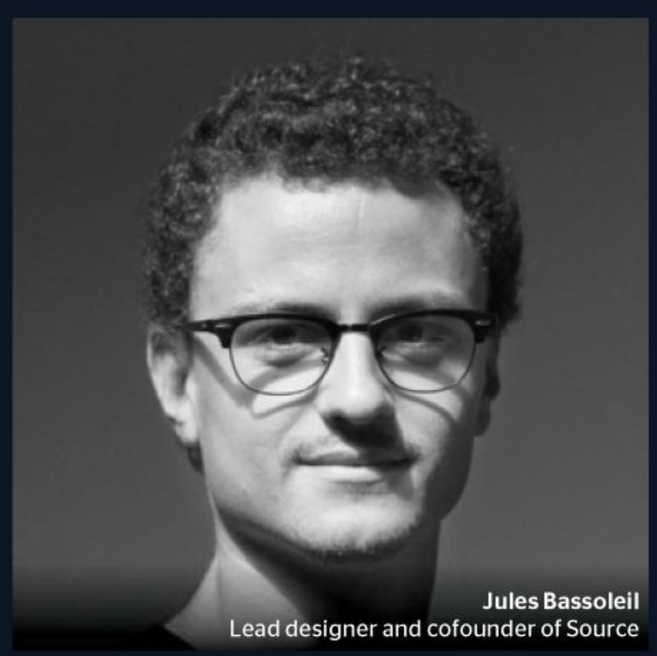
Ice & Sky's sister film by director Luc Jacquet closed 2015's Cannes Film Festival

THE PEOPLE WHO MADE THE PROJECT HAPPEN

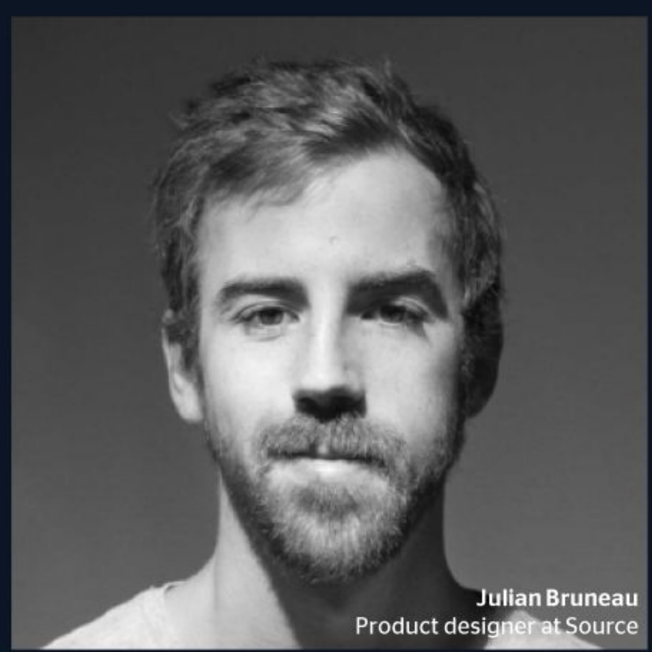


François Pumir, who has 17 years of design experience, was the key producer of Ice & Sky

Pierre Rossignol
Freelance creative developer



Jules Bassoleil
Lead designer and cofounder of Source



Julian Bruneau
Product designer at Source

Martin Charpentier
CEO at Source

masses? Producer and designer on the project François Pumir gives an incisive insight, "To reach the public, they had to create a sensitive and universal language. To tell beautiful stories, they collaborated with various artists on various media."

These fine sentiments are no better demonstrated than via Ice & Sky. The project addresses the question of climate change and uses Claude Lorius' research as a starting point, before moving on to the greater issue of the Anthropocene, an era in which humans are the predominant driver of change.

The project is more than just an online presence – it brings with it its very own feature film by director Luc Jacquet, a collection of TV films and an educational programme. As already mentioned, education is key and the goal of its educational programme is to raise public awareness about climate change and to give educators a tool to work with.

How were the films created? Wild-Touch worked with a collection of international scientific committees, including CNRS and NASA. To show how serious the project is, it is now supported in the US by the Smithsonian Institution and in France by the CNRS.

How do you educate the masses? Tell them a story that needs to be told through beautiful imagery and quality cinematographic resources. The story of climate change has written itself, but how the story was to be presented and created was all thanks to the work of François Pumir, producer and designer for Wild-Touch: "At the beginning, we had to figure out all the content. Each story in this project led to another one and so on. The amount of content was so huge and the climate change history is quite old! Thus, we (Source and I) organised the content so as to get a big picture of all the stories, then we made different scenarios to find the best way to tell a beautiful story to a large audience.

"After getting the frame of content, the second step was dedicated to work on the UX to tell this story. We aimed to keep it as simple as it could be as we knew we were working for a large audience. Moreover, the most important part was to value the cinematographic material and to make the most accurate mediation." Pumir and team opted for a simple framework, where stories would be illustrated fullscreen and content for each would be displayed from top to bottom.

THE 13-PART PUZZLE

The smart thinking that takes a project from concept to the design phase can come in many forms. It may involve hours and hours of head-bashing across creative and marketing to bring the concept to life. Or, as in the case of Ice & Sky, there is a collection of assets that need to be formed into a unique, neat, crowd-pleasing product. Pumir reveals what he and the team had to work with and how they got ready for the design phase. "Wild-Touch's team had already made 13 educational films when we started to work on this project. Narrative was therefore written in those 13 parts in a cinematographic and linear way, not for a digital usage. Thus, the first idea was to create a flexible and simple framework to browse and consult all the content in an efficient way for a large audience.

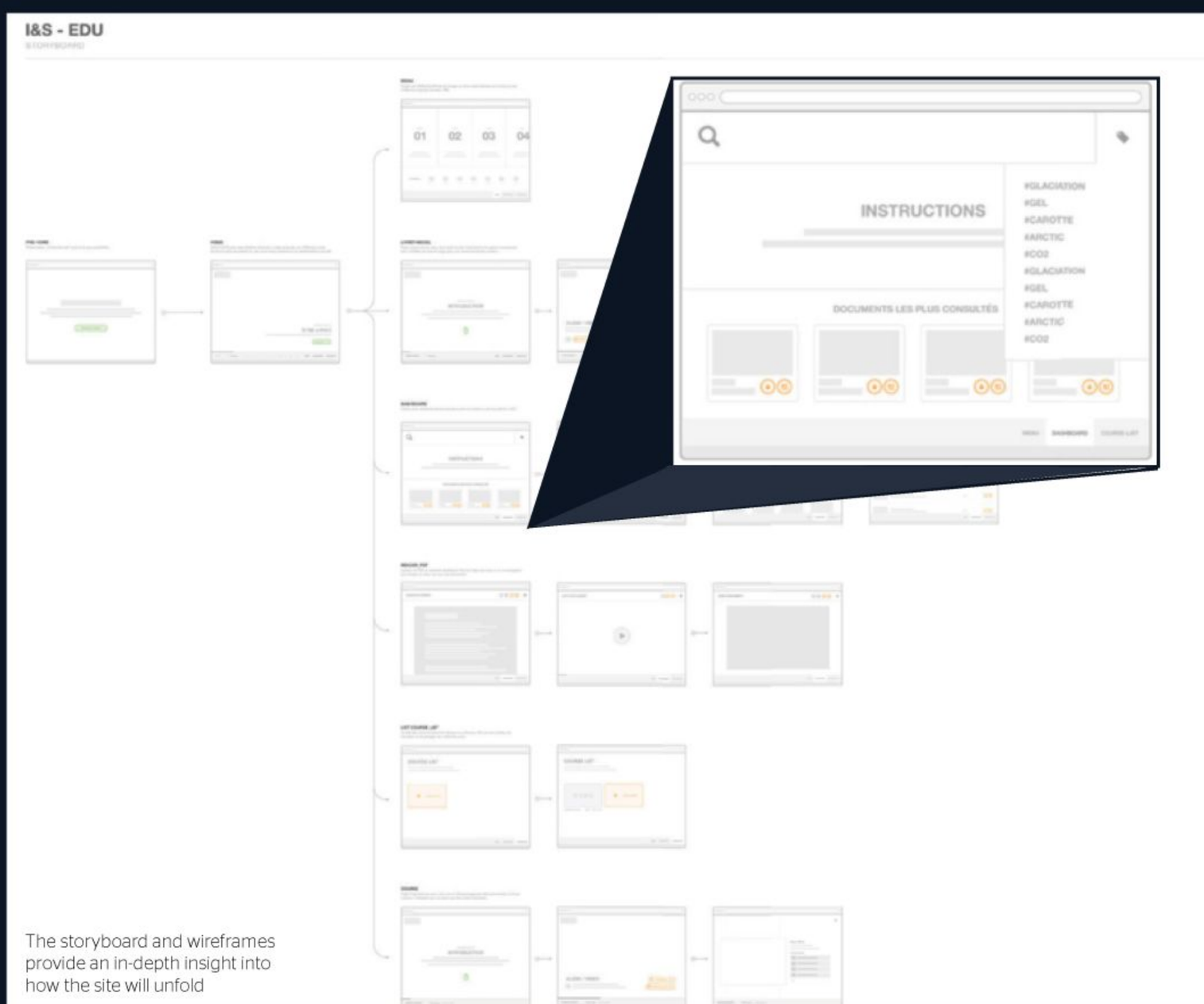
"This framework was based on a few main rules, which drove the design work from the start: all parts of the site have to be seen just like in a film – continuously – in order to figure out the story. The cinematographic material has to be displayed fullscreen to value its beauty thanks to Loic Fontimpe (**storyboard-loic.com**).

"Narrative has to mix different patterns (such as videos, audio and photos) to avoid being boring. Reading on each screen have to be turned to its minimum to avoid being boring again!

"Thus, each screen has to embed additional material such as PDF, videos and sounds to make a deeper level of reading.

"Each piece of content can be saved by users (such as educators) to create their own story in a very simple way (without any account creation which could have inhibited educators from using it).

"Once those rules were drawn, we found a bunch of inspiration to drive our work without making too many iterations. Once those inspirations were validated by Wild-Touch, Source's team could start to work on UX and UI."



The storyboard and wireframes provide an in-depth insight into how the site will unfold

"Then a big part of this project was to make a lot of iterations on the way we were going to tell the story," Pumir continues. "Each word is important and each video was cut precisely to be inserted in the right place of the narrative. I made at least ten versions for each of part of the website to find the right way to tell this story."

A multimedia project the size of Ice & Sky was never going to be the work of a lone wolf. It was always going to be a collaborative effort, and working alongside Wild-Touch's François Pumir (amongst others) were the good people at Source. Their CEO, Martin Charpentier reveals how Source integrated themselves into the project, "Here at Source we don't see it as two separate entities which interact. We don't work for our clients but with our clients. François Pumir has taken part in the conception of Ice & Sky since the beginning and until the end. To [help it succeed], we used several deliverables adapted to each keystone of the project.

"We can split the project in three phases. The first one consisted of thinking of the website as a map, it allowed us to represent the architecture of the website by listing its pages. Also, this document helped to specify what will be within the pages in terms of interaction, content and functionality. At this early stage of the project the site map should not be considered as specification material, it's not a contractual piece but a concrete way to exchange and iterate with our client.

"After that, we build a storyboard of the website, in this kind of deliverable we dig a bit deeper in the content of each page, we use lines, blocks and spaces creating the shape of the future content.

"The storyboard has two functions, one is to clearly see the paths users can go through and it helps us (Source and the client) to choose the right content we'll need to put into each page. In many cases the content materials and the digital experience were produced together. So without digging too much into detail, the storyboard helps to define what content will be needed for each page.

"Once we built the plan of the digital experience and before giving birth to it, we needed to make static

mockups and prototypes. In this phase we first focused on the UX and ergonomics aspect. Here we try and learn the different design patterns. By doing this we also strive to apply the best look and feel. Sometimes, we split it into separate tasks by building at first a selection of high-end wireframes. Our wireframes are meant to solve all interactive and ergonomics issues, allowing the UI designer to focus on his craft."

With a host of assets and initial ideas to hand, it was then time to bring in the creative thinkers to make this thought-provoking project stand out from the crowd with a host of stunning visuals and clever additions. This job was assigned to Source creative director Jules Bassoleil who quickly found inspiration for the story-telling project. "The main challenge here was to imagine a digital product able to handle different type of media such as movies, animations, audio and text. Naturally we took a look around other web documentary styles. We were very interested in Clouds over Cuba (**cloudsovercuba.com**) and The Vanishing Game (**thevanishinggame.com**) which has a very clean UI and a simple navigation system. The problem here is that this kind of experience is too linear and uses only one kind of content.

"One other very good reference is Warsaw Rising (**warsawrising.eu**), a masterpiece in terms of animation and narrative experience, it also gathers a lot of different content including some neat data visualisation.

"We needed to think about an experience to enable us to gather lots of different content while allowing the user to jump from chapter to chapter and picking up content they liked on the way. So we designed a three-layer experience, a navigation allowing users to get access to any chapter, the main container within each part and the last layer which gathers the linked documents for each chapter. By doing this, the user can easily discern the different kinds of content.

"To be compliant with our large audience, we needed to create contrast to make navigation and interactive elements attractive. For this the animations and feedback played an important role. For this reason we spent a lot of time with our mate Pierre Rossignol to make it perfectly fluid and reactive.

"As designers we don't consider that our work stops after producing a static mockup. We've conducted and advised [Pierre Rossignol] on how the animation should be until the release."

The next stage for a completed site is to get it launched to the web masses. How did such a large, all-encompassing project such as Ice & Sky reach its audience? It didn't go down a typical path as Rossignol explains, "The project has received a huge amount of positive feedback and support since the launch, which we're very happy about. We mostly used institutional networks to promote the website.

"Actually it was quite easy to get people visiting the website as the subject speaks to everyone. People are impressed by the quality and the exhaustivity of the content they can find on the platform, and are more willing to share to their friends and networks.

"As an innovative educational platform it has been presented to the National Education Ministry here in France, which helped us reach professors.

"The professors are very excited about the My Website and Downloads features, which allow them to pick content from the platform and create their own little websites with the content they choose for their classes and to make more advanced research in the related documents.

"The project has also been presented in Washington DC and in New York by Luc Jacquet along with actress Marion Cotillard, who supports the project and lent her voice on the videos.

"Main scientific institutions involved in the project who support us include CNRS (National Center of Scientific Research), NOAA, NSF and The Smithsonian Institution who'll support the project for U.S.

"Having the support of those scientific institutions, along with Luc Jacquet and Wild-Touch promoting the whole project abroad has helped the platform reach its target audience for sure. Moreover, a huge campaign will be done with educators to hold the programme from September to the COP21 [Sustainable Innovation Forum] in Paris in November, which will be a sounding board for the project."

Collaboration was key throughout the creation of the entire Ice & Sky programme and once the project was

BUILDING AN INTUITIVE BACKEND FOR ICE & SKY

The unique visual nature of the Ice & Sky project meant that it needed a dynamic nature to complement the content. As always a project is a combination of frontend and backend and getting the balance right is critical to how the finished project engages its audience.

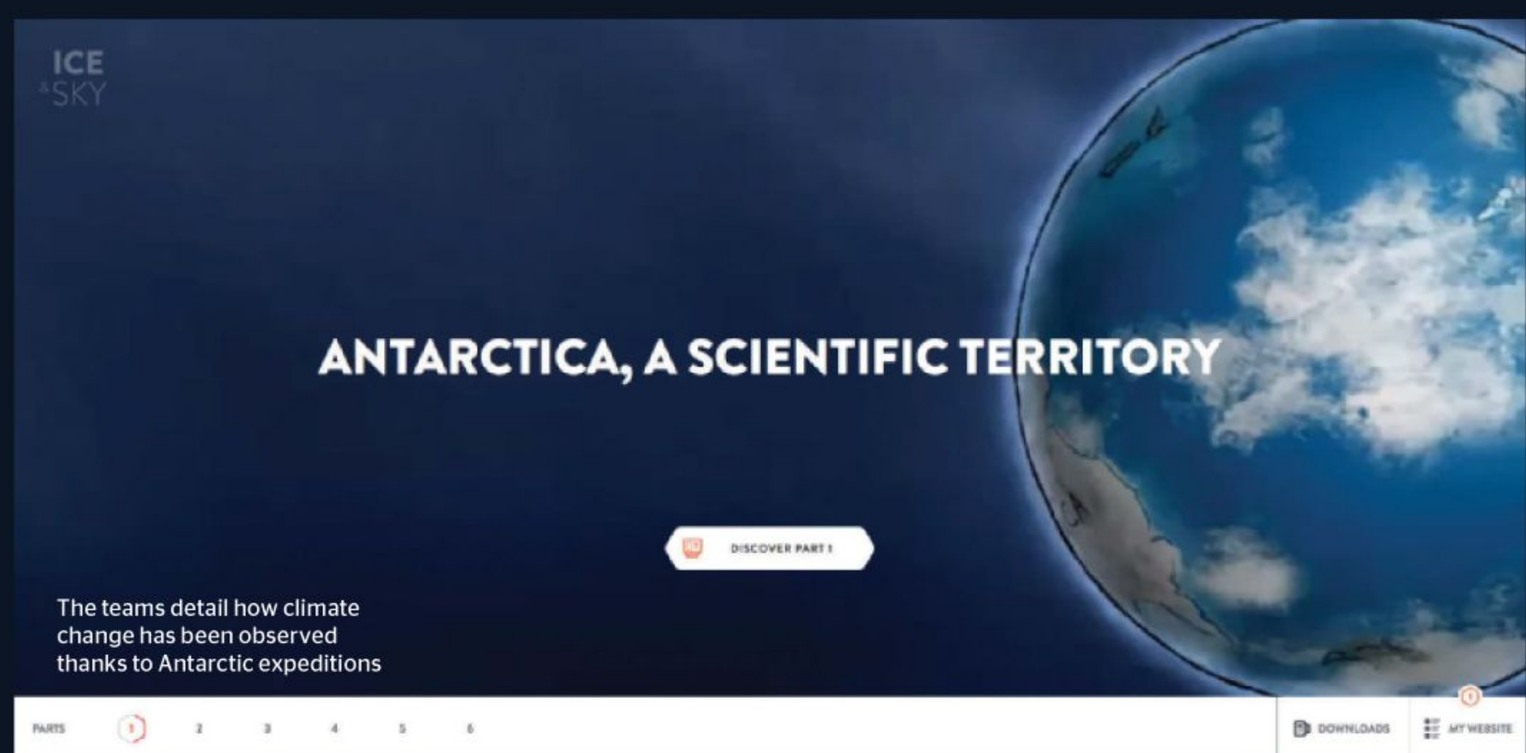
The man assigned with the task of bringing everything together was creative developer Pierre Rossignol. "All the content had to be dynamic and managed by an intuitive back office in order to create and update the content throughout the project. We had a complex structure of content with six main stories and multiple types of content. Every story contains a different number of screens, which contain text, video or audio.

"We built a back office based on WordPress to allow François Pumir and Wild-Touch to create the content and upload all assets easily. The back office generates a JSON file with the content for each language. Our front-end app just gets this JSON file for displaying the right content. This process had many benefits. It allowed the team to test the content and actually see it in the real website during the development phase. Using a JSON file generated only when needed as our only source of content allowed us to have a stand-alone app and not to rely on a back office, which would fetch the content in a database each time a user comes.

"For optimisation matters we hosted all our assets on Amazon Web Services. Using a CDN was crucial regarding the number of videos we had and to provide our users a smooth user experience by reducing the loading time.

"The biggest challenges were in the frontend. We built a complex MVC JavaScript structure based on our custom framework. It was very important to have clear code conventions and keep it clean. We chose to build a one-page web app, so the navigation was a big challenge. This meant loading the content on demand. We needed to smooth every page transition so the user would still be immersed in the content all the way. We also spent a lot of time on the animations. It had to be very smooth and needed to be meaningful, not trivial, in order to give our users the best user experience and enhance the content which was our main goal."

"IT WAS QUITE EASY TO GET PEOPLE VISITING THE WEBSITE AS THE SUBJECT SPEAKS TO EVERYONE. PEOPLE ARE IMPRESSED BY THE QUALITY AND THE EXHAUSTIVITY OF THE CONTENT"

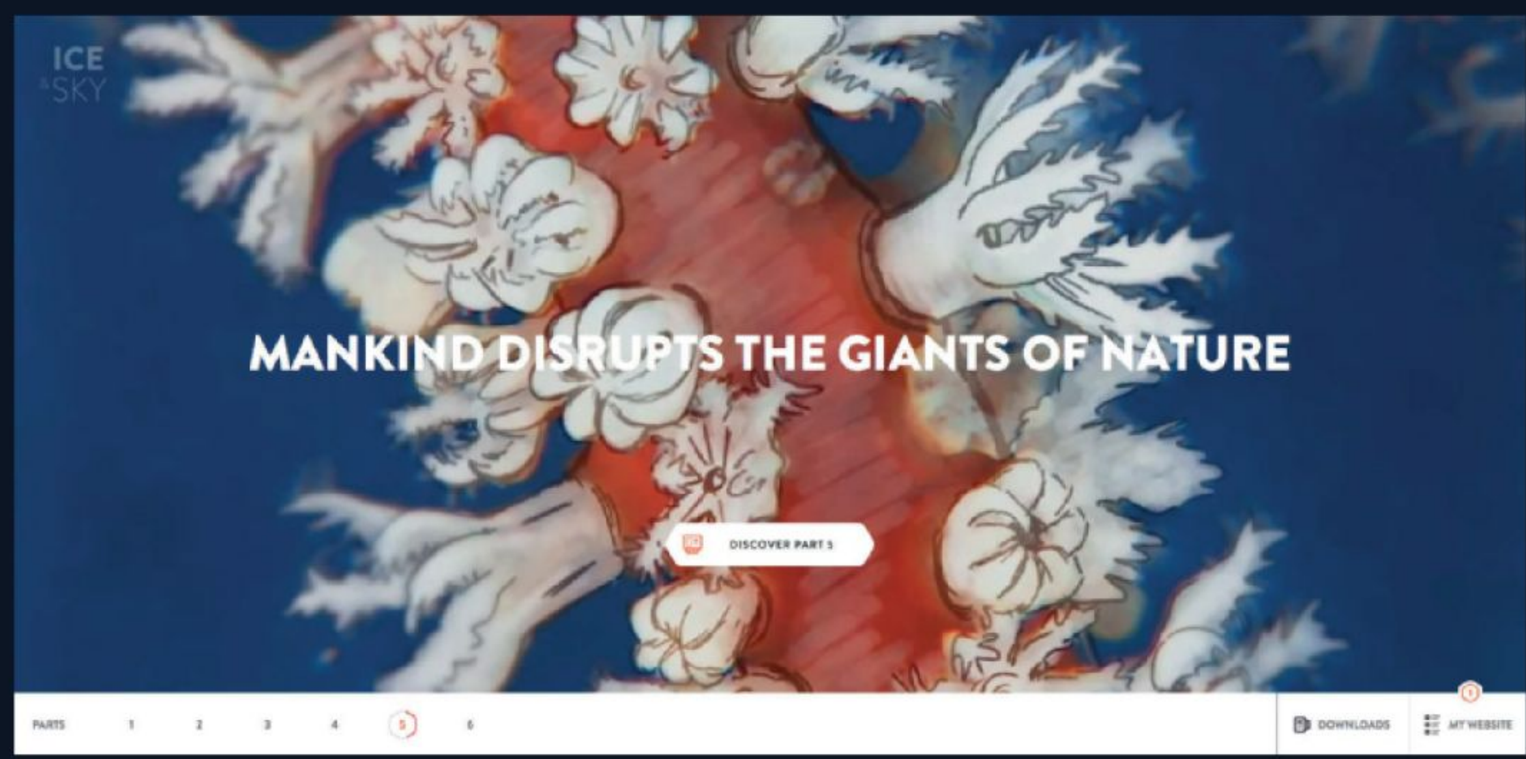


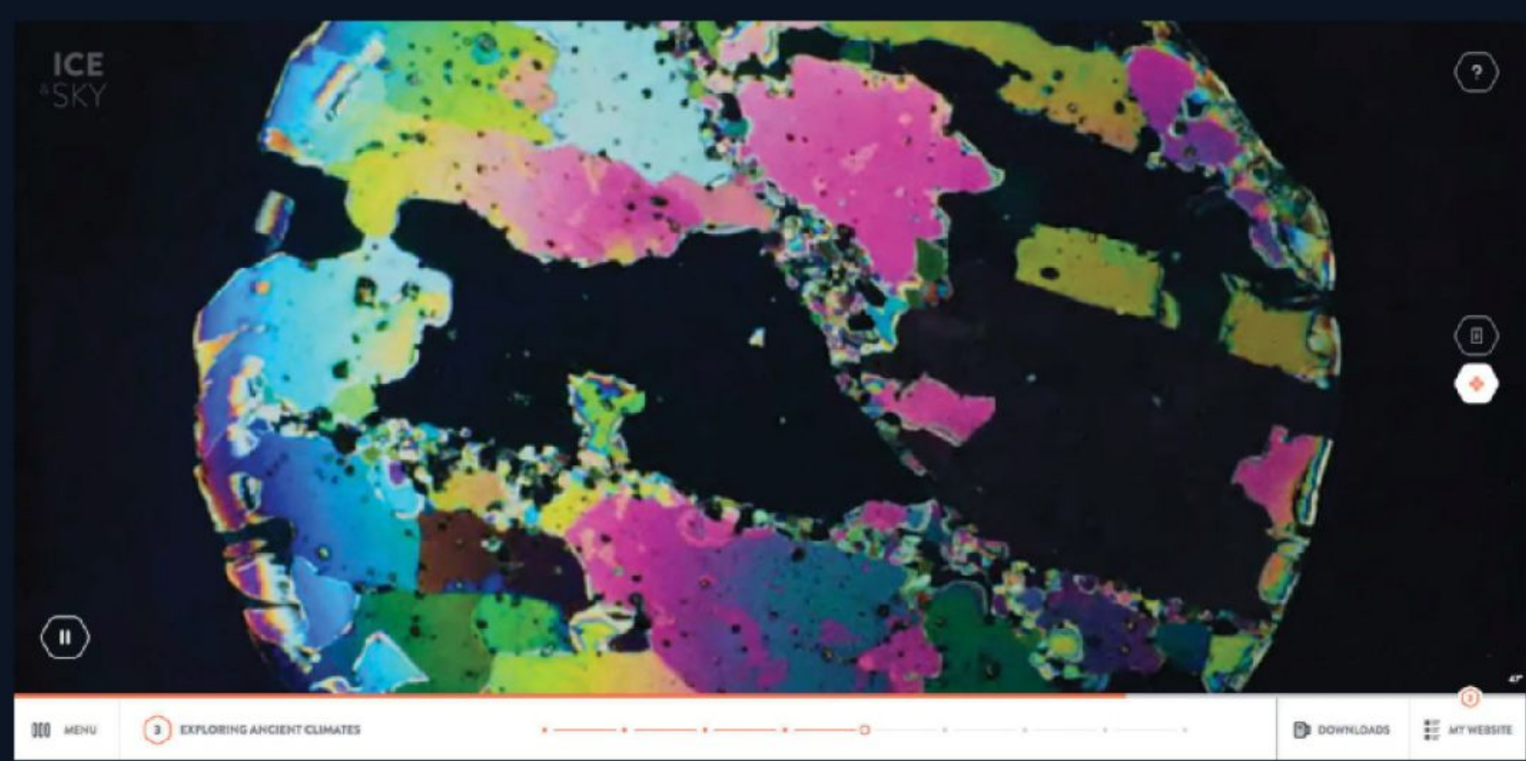
The teams detail how climate change has been observed thanks to Antarctic expeditions

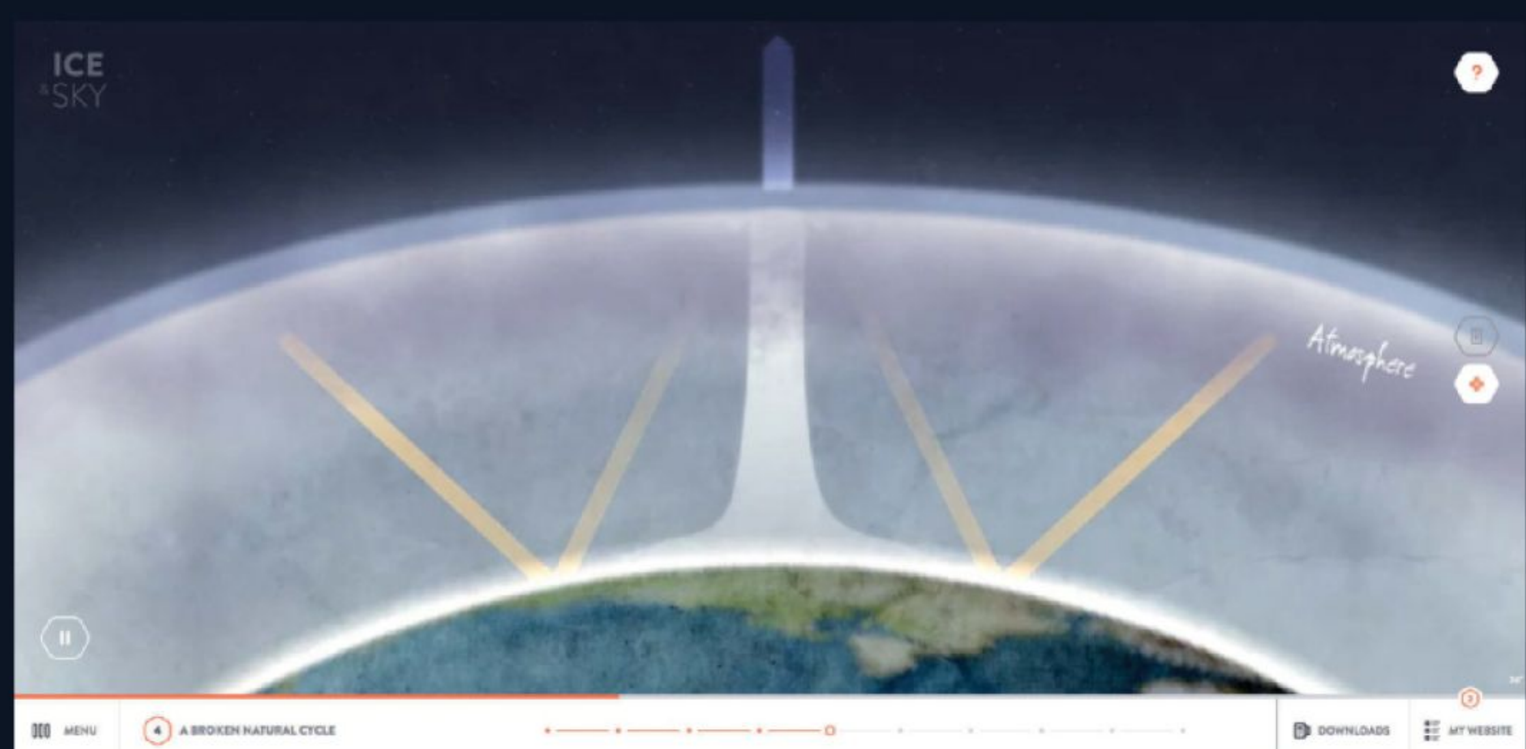
complete things didn't change drastically, meaning the agencies didn't have to go their separate ways as Source CEO Martin Charpentier reveals. "This is not our first [time working] with Wild-Touch. François, Jules and Pierre have been working together for a while, their last project was the Once Upon a Forest (**iletaituneforet-expedition.org**) project which has won many prizes including TheFWA x Adobe Cutting Edge award. So there is no particular process with Wild-Touch, it's more like a relationship based on trust and understanding. We're working like this with most of our clients, each part needs to preserve the other in terms of works, stress and financial aspect. We hope we'll keep this relation with our future clients.

"Of course Wild-Touch got the full rights to our work and the website has been developed, tested and run on Wild-Touch's own servers. For every project a little budget is provided for aftercare, but it's always difficult to specify. We're mostly working on products and not simple websites with clear and defined scopes. Products are meant to evolve and it's often what our clients ask us: to work more to refine their product.

"Before the international release we ran a few user tests, which provided us feedback on the site's weak points. So we worked to improve the site until global release. We don't really see this as aftercare but simply as a part of our duty."







THE BIG QUESTION...

What is the one key change needed to halt climate change altogether?

"Since the beginning of the industrial revolution, humans have been aware of their impact on nature, but industries and short term profits – supported by politicians – enforce these laws and therefore cultures. It's now time for people's consciousness to take the lead on any other considerations if we really want to make a change to these issues."

FRANÇOIS PUMIR
PRODUCER AND CONCEPTOR

"People need to be aware of their impact on the planet and climate. Future generations will suffer the aftermath if their behaviour follows ours. So we need to educate and sensitise the generations to climate change so that they will avoid making the same mistakes as their elders."

JULES BASSOLEIL
LEAD DESIGNER COFOUNDER AT SOURCE

"To me, the key is to keep talking about it in order to involve everybody, make them understand that solutions already exist and we have to act now! To have a real impact, we need to be together."

JULIAN BRUNEAU
PRODUCT DESIGNER AT SOURCE

"Climate change is not a fatality, the climate has always changed and species and human have adapted. It may not be as bad as we think but by understanding it, we'll be able to minimise and overcome its impact on our lives. If we need to change our bad habits, then we should embrace it."

MARTIN CHARPENTIER
CEO AT SOURCE

INVENTING

Standing at the epicentre of design and tech, the team at DPDK are creative thinkers that refuse to be pigeonholed. Diverse skills have lead to a business that shapes digital products that enhance and disrupt our lives

TOMORROW

who DPDK
what Creative strategy and digital production for a wide range of engaging projects and products
where Calandstraat 5a, 3016 CA, Rotterdam, The Netherlands
web DPDK.com

Key clients

Peugeot

Nationale Nederlanden

DS Automobiles

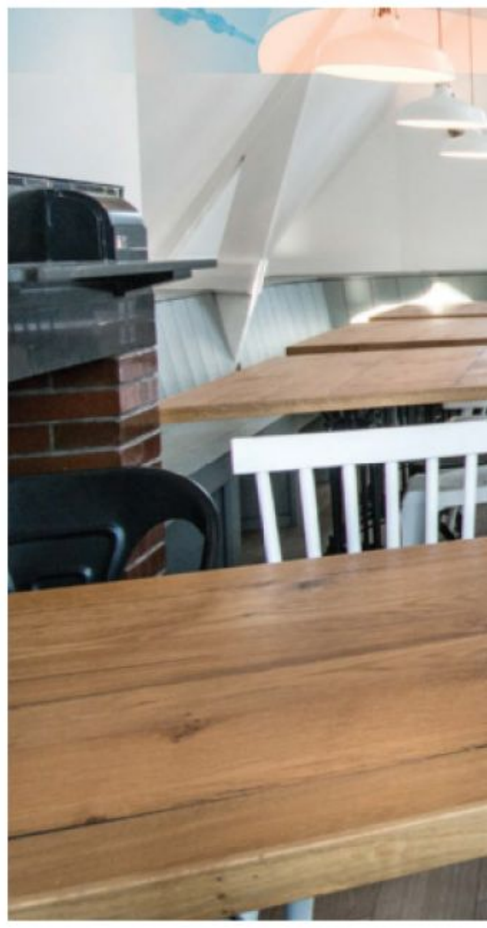
Vitens

Sense / SOA AIDS NEDERLAND

Defining DPDK can be difficult, as the agency offers a diverse range of skills to its clients. Whether they need a production company, design boutique, game studio or technology services, DPDK brings fresh and dynamic solutions to every account. Few agencies have the breadth of skills that DPDK possesses, placing it at the forefront of the digital design that shapes our world.

Pim van Helten, CEO at DPDK, explains how the agency came into being: "Officially, the agency started in 2001 with Rolf Vreijdenberger and myself, but it took shape when I joined the agency after graduating in 2004.

"We offered technical production at leading advertising agencies. Back then we mainly developed with Flash and PHP, which was often the basis for innovative projects of the (old) web. I functioned mainly as a producer, while Rolf functioned as the creative developer. Between 2004 and 2010 we grew the agency fast, mainly because we then focused on adopting creative developers. Because we worked for many different agencies we were able to build up an outstanding portfolio, which encompassed a number of skills that were in demand."



As a calling card, an agency's website is often the first point of contact with a new client. Michael Vromans, creative director, outlines the approach that DPDK took with its design. "Our website should convey what we stand for and who [we] are going to cooperate with. It is therefore extremely important for us that we present ourselves clearly through our website."

Van Helten continues: "In addition, our website is a platform to show how proud we are of our work. Often our projects are complex and developed by a large team. We therefore like to highlight how we came to our final product. Through a case study on our website we outline the background story. That takes a relatively long time to design, but we like it simply because we love our jobs."

How agencies actually earn their work can be manifold. As one of the longest established businesses in the digital space, DPDK has been able to work with some of the world's best known brands. "We are an agency that often pitches for challenging projects," Vromans tells us. "Fortunately, we have a broad customer base and do not have to pitch [by] default as many customers know exactly what they can expect from us. But pitching keeps the agency sharp and brings new insights and ideas to the team. It is also great to outperform the competition!"

"We have a broad customer base and do not have to pitch [by] default, as many customers know what they can expect"

With a world-class reputation that proceeds it, how does DPDK choose which clients to work with? Are there any business criteria that a new client must meet for them to be suitable for DPDK? Vromans continues: "We do not service a limited domain when it comes to the markets our customers do business in.

"We do, however, have a strong preference for a client who understands the 'what' of technology. Understanding the 'what' of technology means that you put customer experience at the heart of your transformation – from [the] outside in. They keep on innovating – it's never over. Every improvement

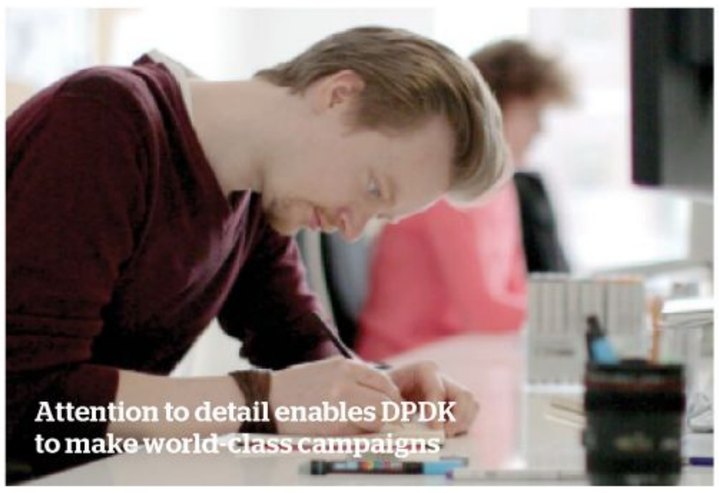
[to] the digital customer experience will open up new possibilities to improve the way they do business. Clients who understand the 'what' of technology often have a digital culture with a strong overarching digital vision.

"Our service is very dependent on cooperation, so we prefer clients who bring out the best in us. We have learned that digital projects can lead to internal process changes for our clients. We assess whether or not our future clients can manage this change prior to a project.

"In addition, over the years we have made our trade-offs within the industry. For example, we do not develop comprehensive e-commerce systems. Today there are specific systems, but in the past we have learned that this expertise does not fit with our core competences. We also do not offer outplacement of our team members. We like to work closely together with our clients, but do so from our own organisation."

Many agencies gain a reputation within a specific sector, or their work has a clear thread running through it that marks it out as *their* work. Does DPDK have any work that defines it? Van Helten explains: "Translating offline to online user experience is one of our specialties. The 'Catch The Dragon' campaign we created for Peugeot for instance is an interactive prolongation of a television commercial. For the film, we combined Peugeot's existing commercial with a 360º virtual reality test drive we had already filmed in the sunny French mountains.

"Peugeot's TV commercial was filmed in Los Angeles by night and features a dragon tattoo coming alive, flying out of the new 208. It was quite a challenge to combine these completely different worlds. We blended the virtual test drive and the commercial into an interactive virtual reality film, in


Attention to detail enables DPDK to make world-class campaigns

INDUSTRY INSIGHT

Pim van Helten, CEO

"We love to translate a story into an idea, and then translate the idea to an engaging user experience through our production. With all the necessary specialists that work together at DPDK we can achieve a cutting edge production within a relatively short time"

DS | SIGNATURE ART

ds-signatureart.com

Your signature is personal and unique. And that's what car brand DS is all about: expressing yourself. To celebrate their partnership with the Dutch Filmfestival, we turn your signature into a piece of art.

Just like in Hollywood, the festival boasts a boulevard of fame with autographs and hand imprints of prize-winning actors and actresses. Inspired by these signatures, we developed the idea of leaving your own signature in the avant garde style of DS.

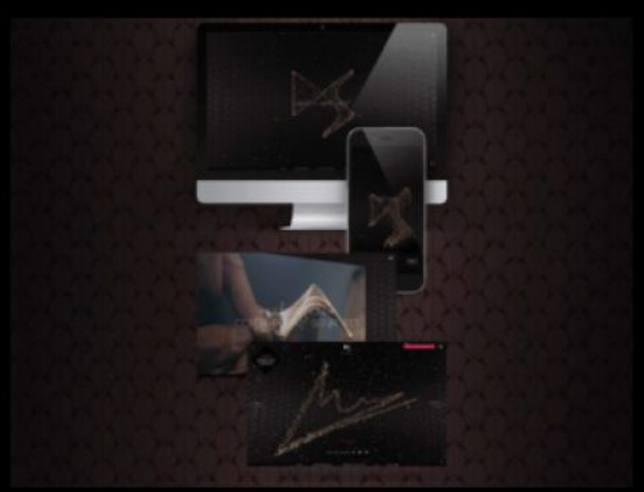
The intro is a seamless mix of code, design and video all melted together. Because of this state of the art approach, the intro even plays on mobile devices.

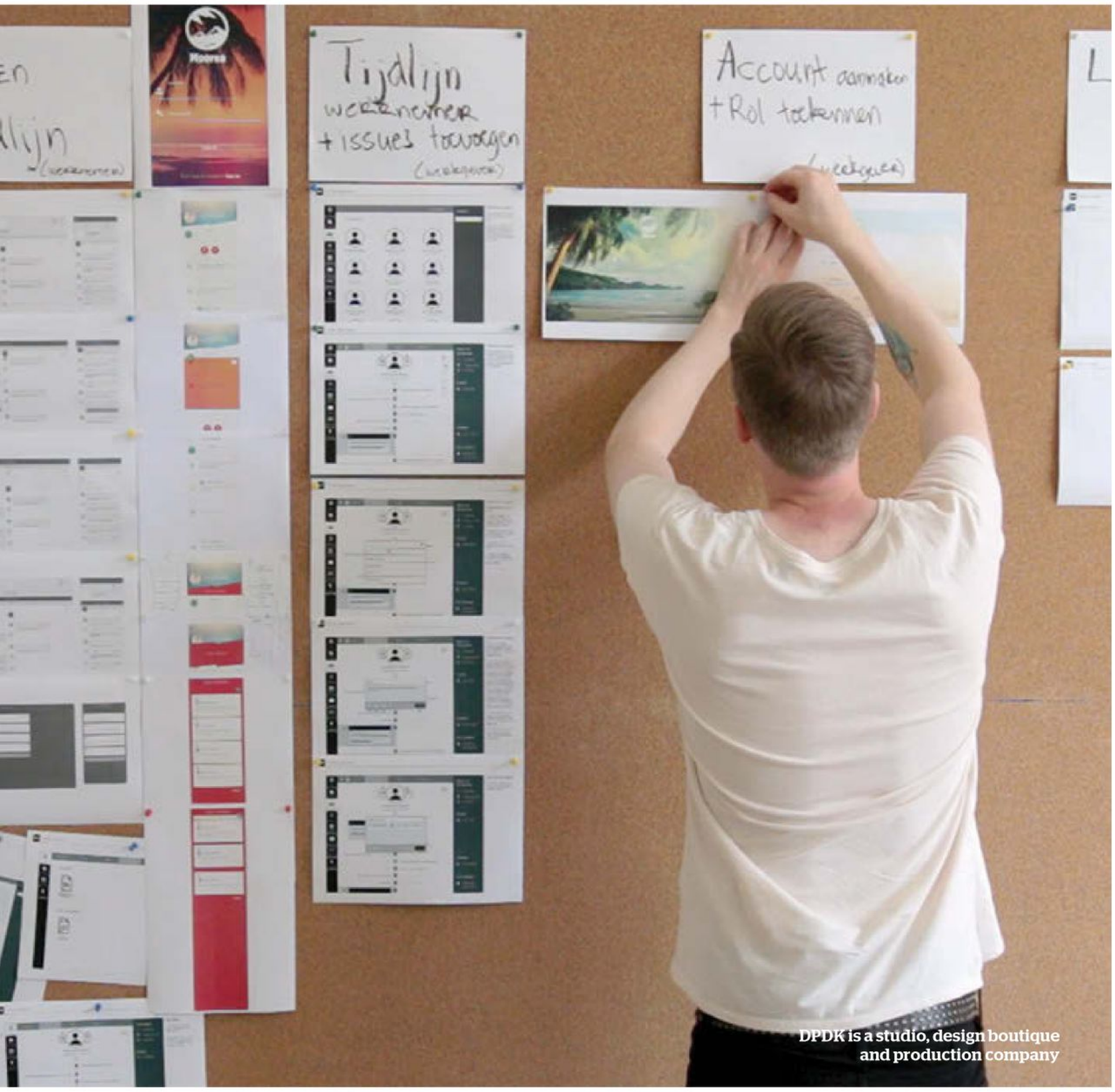
Writing your signature works best on mobile. Just draw with your finger and voila: there it is. Visiting on desktop? No problem. Just connect with your smartphone and draw on there. Your signature will appear on your desktop screen like magic.

To create the right look in 3D signature and statue, we started out with 3D experiments where shapes were simple. In a line of prototypes, we added detail, texture and lighting to end up with an atmosphere that fits the avant garde spirit of DS.

DS signature art is all about you and your personal expression, and as such the interface has been kept to a minimum but always in line with DS style. A soundtrack that really complemented the innovative nature of the project and the brand was designed. It's progressive, modern and a tad experimental.

Signature Art is also available as an installation at the Filmfestival. We built a motion sensor controlled version of the website where visitors can write their signature by writing in thin air.





DPDK is a studio, design boutique and production company

Cooperation and collaboration are at the heart of everything

which you've set out to catch the dragon by moving your smartphone around. The site works flawlessly on mobile devices, which is a unique feature for such a complex product.

"We love to translate a story into an idea, and then translate the idea to an engaging user experience through our production. With all the necessary specialists that work together at DPDK we can achieve a cutting edge production within a relatively short time."

Some of the projects that DPDK has completed are large and complex, so how does DPDK approach each and every project that comes through their doors? Van Helten explains the approach. "Large-scale projects often have a timeline of six to 12 months. Half of our projects we produce within six months. The core of a team consists of two backend developers, two frontend developers, an art director and a producer. Often, a strategist and the creative director also join the team, depending the nature of the project. Larger projects at DPDK usually have the focal point towards technical development, because backend integrations with systems used by our clients are time consuming to develop."

The toolset that is now available to digital developers is massive and continually evolving, with some agencies even developing their own frameworks to be used in their projects. So how does DPDK approach using its toolset? Van Helten outlines its current practices: "Whenever we develop a backend, we work with Drupal. It's the most flexible and reliable Open Source system available and has a large support community.

"For design and frontend there are the usual suspect such as Photoshop, but all in all we do not cling to a certain package simply because it worked last time. When you're focusing on creating cutting edge, you have to be willing to try new methods every single time in order to become cutting edge. The one tool where both the whole team and the client connect is the project management tools Basecamp for project/team communication, Jira for bug tracking, Github for software repositories and Google Apps for sharing documents and presentations."

"We are very excited about how WebGL is influencing the Web. Since WebGL is embraced widely by the mobile industry, we see that this technique is maturing very fast indeed. With this technique it's possible to create rich experiences for advertising campaigns and interactive websites. Also this year we have already experimented with WebGL and WebVR within a number of our productions. It is fantastic to see what's possible and how the Web develops through these tools. We are very excited about the future!"

The vast proportion of prospective clients will often have a very clear idea of how they would like their campaigns to reach their customers. Mobile devices are now at the centre of these relationships, which often influences how agencies like DPDK will approach a piece of work. Vromans explains that: "Across all of our websites we work with responsive designs from a mobile-first approach. We've been seeing mobile traffic grow for years now. Therefore, all products that we develop need that mobile-first approach. We believe that if you have the perfect mobile layout, you can translate the design easily to a much better design for desktop and tablet.

"We also develop many native apps. If you want to use specific features of the mobile phone such as fingerprint, camera, or the NFC-chip then you have to develop native. At the moment, for this type of project, there are no alternatives for developing native, but it could be that in the future fewer and fewer native apps will be developed.

"The ways in which film productions are evolving are fascinating. There doesn't seem to be a week in which there is not a really great idea on Kickstarter that will make shooting killer shots at low cost possible. Think of a drone that automatically follows a dynamic object or an add-on for your camera that records all your movement in order to completely eliminate those movements afterwards, giving you an always stable shot."

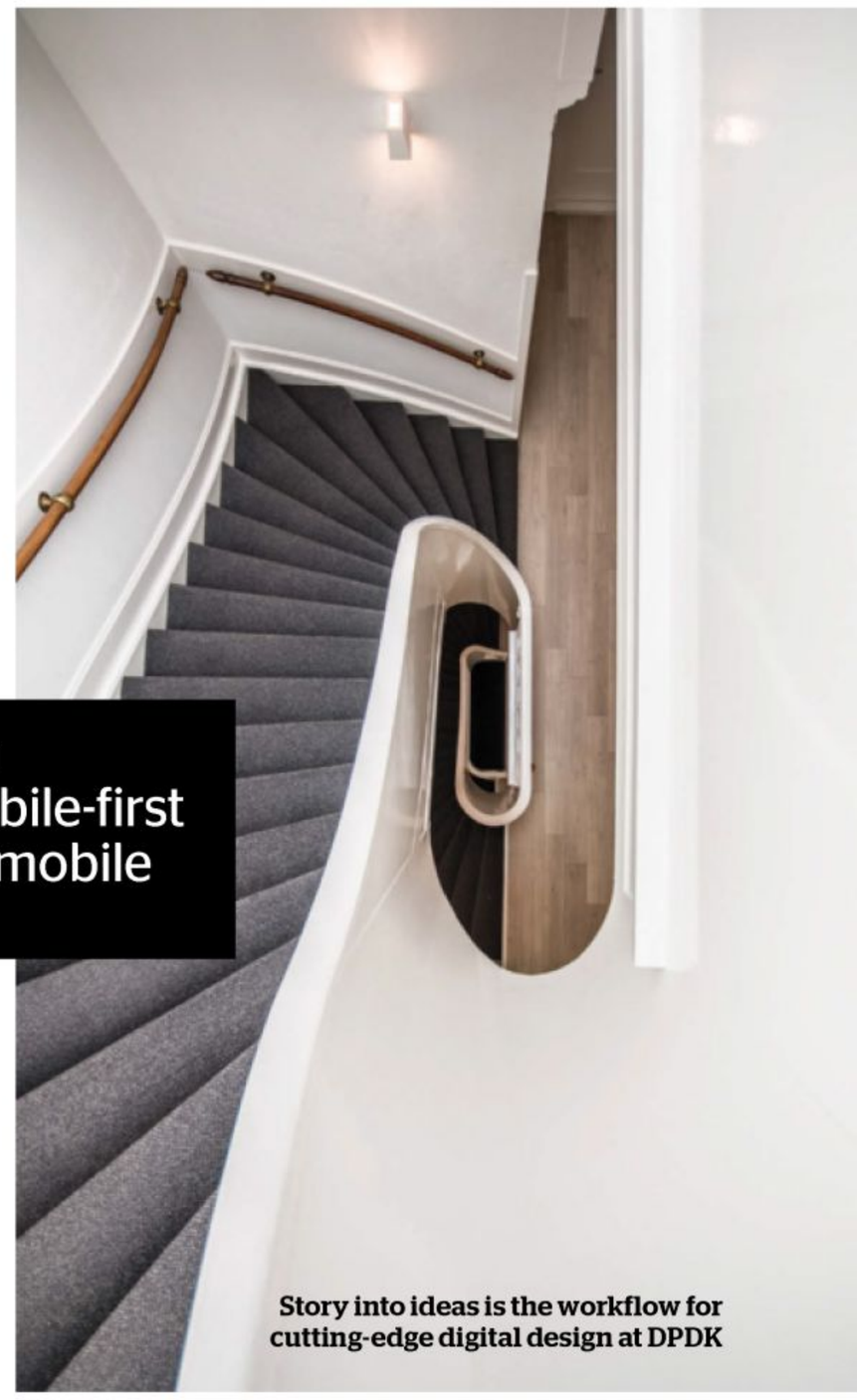
At its core, a creative agency is effectively only as good as the people it employs. But what are the kinds of qualities that DPDK will look for in a prospective employee and what advice would the team give to anyone looking to take a step into the industry and furthering their career?

"In our profession, everyone is now a digital native, so make sure you excel at something," says van Helten. "At DPDK we hire for passion, not for experience. We always look for the team player that is broadly oriented within the industry and wants to be the best in [their] field. For us as an agency, it is important that you connect to the core competences of the agency; management of change, conceptual thinking, team collaboration and production management."

As an agency in constant demand, what does the future hold for DPDK? Vromans concludes: "We expect to both expand and diversify. The digital revolution has only just begun and we believe that our best days are yet to come!"

> "Across our sites we work with responsive designs from a mobile-first approach. We've been seeing mobile traffic grow"

It is difficult to define an agency like DPDK, as it strives to be different with its approach to each brief, but also how the philosophy within the company has developed. Just as the digital space is rapidly evolving, so DPDK remains on the bleeding edge of design.



Story into ideas is the workflow for cutting-edge digital design at DPDK

DPDK has evolved into a fine digital agency

Web - DPDK.com
Founders -
Ferry Perret Gentil
Rolf Vreijdenberger
Pim van Helten
Year founded - 2001
Current employees - 40
Location(s) - Rotterdam

[1] Creative strategy
[2] Digital production
[3] Branding and campaign
[4] Web and mobile
[5] Motion and animation

TIMELINE

Year 2001 - 2005	Year 2005 - 2010	Year 2010 - 2015	Year 2015 - 2020
no of employees 5	no of employees 10 - 25	no of employees 25 – 40	no of employees 40 +
Long freaking nights with Flash as the basis for its creativity.	Servicing production for leading ad agencies gave the team direction.	Collaboration and production strategies formed the base of the DPDK operation.	Expanding core competences, innovation and entrepreneurship.

INTERACTIVE ARTISANS

With a keen sense of design coupled with technical prowess, craftsmanship is at the heart of Cartelle alongside a continual strive to create digital spaces that transforms brands and the lives of their customers

Founded in 2009 by Johnny Slack and Stevijn van Olst, Cartelle began a partnership that would see the two founders as well as their colleagues develop new interactive experiences for brands from all around the world. Johnny explained how Cartelle became a reality: "We started out when I was working at AKQA Amsterdam tasked to help build the creative development team. Stevijn was a freelance art director and creative helping to get new Nike work.

"We were paired up and I would create rapid flash prototypes to coincide with the decks Stevijn was creating to be pitched to Nike. A lot of really good work got thrown in the trash. We eventually grew tired of the agency process and pace. We figured we could just ditch the whole agency thing and start our own interactive studio. We rented out a small spot that used to be an old toyshop just outside the centre of Amsterdam. After some fresh paint and lettering our street window, we started work as Cartelle."

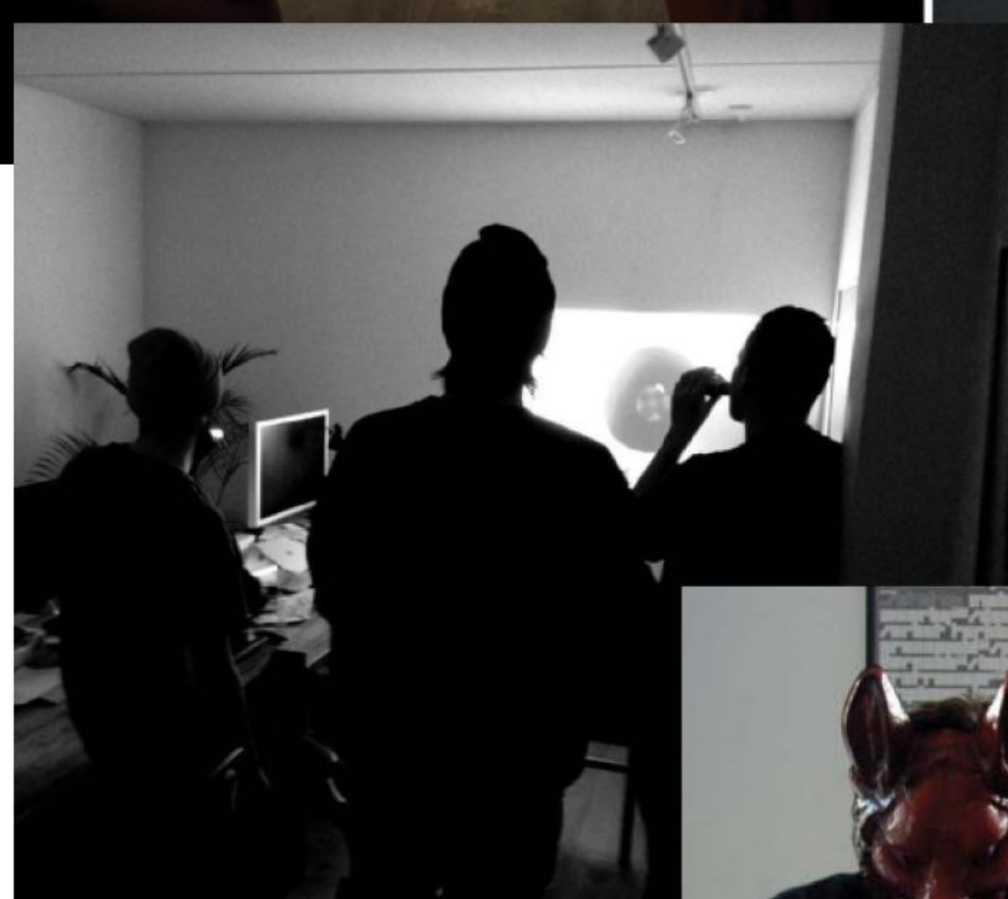
who Cartelle
what An Amsterdam-based interactive studio focused on digital design, motion, development and strategy
where Elandsstraat 22, 1016 SG, Amsterdam
web cartelle.nl

Key clients

Adult Swim
G-Star
adidas
Scotch & Soda
Red Light Radio

As digital artisans, Cartelle are always crafting new digital experiences for their clients

Brands come to Cartelle for cutting-edge design and to encapsulate their brand values



Strong relationships within the core Cartelle team results in original award-winning work

> “We’ll do a self-initiated project to attract the work we want to do”

A sense of humour is vital to working with Cartelle

As with all new businesses much debate is often expended on the naming of a new agency. Stevijn outlined how Cartelle's name came to be: "We always had the name Cartelle from when we decided to start our own thing – but not the spelling. When choosing the name we didn't want to be soft, tech-y, silly or corporate. We liked the sound of the word and that it would coincide with some of the work we do for fun, which is often darker, tongue in cheek and may have references to skulls, sex and weaponry.

"Although it's pronounced the same, the spelling change from Cartel to Cartelle was just so it sounded a bit more premium, like a nicely scented perfume. And also so it could stand on its own as a brand name. The domain name was no problem – luckily it was available. We did receive threats from the Cartelle municipality in Spain to change our name, but we just ignored those and things are just fine now!"

Developing a website that will ultimately be your calling card is vitally important to get right. "We put a tremendous amount of effort into our website," Stevijn continued. "Our latest site went up a few months ago and we found it extremely important to not just represent our work with stills. It seems obvious for an interactive shop, but we wanted [our] site to be chock-full of captures so the substance of the work comes across. It's our gallery and our one opportunity to try and wow visitors. We're only two people with no PR arm or anything like that – if our site and work doesn't speak for itself then we're out of business."

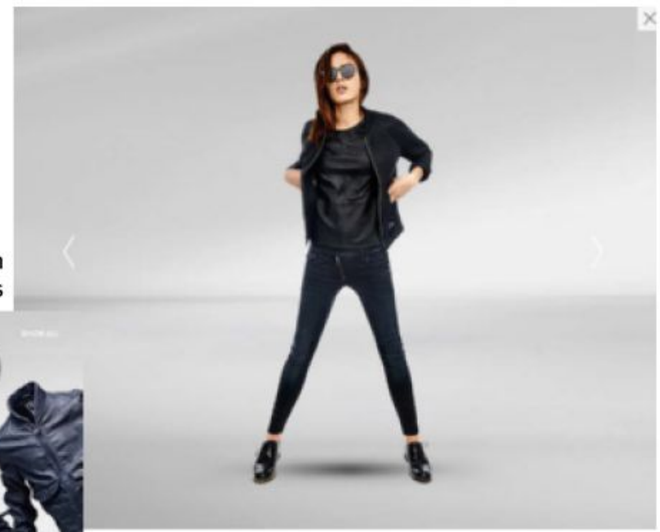
How agencies attract new clients can be manifold. For a small agency like Cartelle, being creative and using their industry contacts has bought them work that has enabled them to showcase their skills. Johnny explains their approach to clientele: "We've been lucky enough to not have to pitch often. That was part of the reason for staying small: so that we don't have to get desperate about the projects we take on. We'll pitch hard if we really want the work – especially if it's a game or entertainment-based. We can often be selective and not give in if the demands are intensive to win a project.

"Having said that, we have had to deal with being part of a pool of other interactive studios (often our friends) bidding and putting in weeks of free work for ad agencies to try and win the same projects. There are always more losers than winners in these scenarios, especially when you're small and are being asked to pitch for free, so we try to avoid this at all costs. It's difficult to give free work to ad agencies that have these insane loft spaces with cabinets made of tortoise shells, ya know?"

All agencies have an understanding of their values and design aspirations. Choosing which clients to work with to achieve those goals can often mean personal work that showcases a skill that then attracts a client, which is how Cartelle often approaches the business aspect of their agency. "I think as far as who we work with, we have to do this by creating the work first and then hope they will come," says Stevijn. "We have a 'dream client' list. If we want to steer the type of clients we want to work with, we make something – we'll do a self-initiated project to attract the work we want to do. This is why we did the Dilla Dimension and Love Letters From Craig for example. They're passion projects that will hopefully attract more work we're passionate about. It's very competitive, so it's difficult to just reach out sometimes with a portfolio link, even if it has big names in it."

Being a small agency it's important to stay agile and ensure deadlines are met. Johnny outlined Cartelle's working practices: "Projects typically take usually nine to twelve weeks or so. We concieve everything together. Stevijn is creative director, art director, a Cinema 4D enthusiast and designer. I take care of front-end development and creative programming – JavaScript and WebGL mainly. We both do the interactive concepts together, as well as things like sound design. Tasks like server-side programming, QA and so on. We bring on a close circle of friends to help out. We use Cinema 4D quite a lot – it's fantastic. We use Logic for sound editing. For anyone interested, Sublime is the editor of choice – alongside Tower and Beanstalk. We use Google apps for pretty much everything and preferably AWS for servers."

Evolving their working practices and the tools they use are reactions to how digital content is now consumed. Johnny explained: "[With] HTML5 and CSS3, because of the growing importance of mobile, the evolution of these technologies will probably be heavily motivated by the need for



A 3D space is used to allow viewers to explore products within the Denim Cloud

Cartelle understands all of the fundamentals involved in communicating brand experiences

G-Star New Denim Arrivals

archive.cartelle.nl/denimcloud

We developed an immersive 3D 'Denim Cloud' showcasing a selection of trademark G-Star art pieces and denim products that highlight the Raw New Denim Arrivals. Users can discover the full product range by fluidly rotating the cloud in 3D space. Raw New Arrivals is our first exploration into the commercial application of WebGL technology.

We found this project especially important because it launched right at the time where displaying products flat in even grids was (and still is) all the rage. We wanted to inspire a new way of thinking about eCommerce and how online experiences can eventually translate to physical spaces.

The platform bridges the gap between an immersive experience, a clean interface and a solid eCommerce integration. The quick-buy feature allows the user to purchase items straight from the experience without interrupting the user journey.

We prototyped the experience in After Effects and Cinema 4D in combination with rapid prototypes. This is usually how we start any project like this. We kept the products as 2D sprites, but they always face the camera.

Once we have actually put all the products in different depths, it gives the feeling of a fully 3D experience. We had a long 2D strip of the products as a reference point to ensure that the composition felt balanced. The behaviour and getting the exact position of the products to match the reference in 3D space was just a couple of the challenges that we faced with this G-Star denim project.

> “Responsive is a nice solution...but the more we do mobile sites, the more different we want to do them from the desktop site”

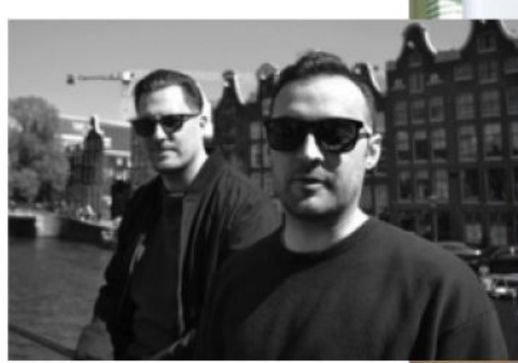
easier mobile development. Development for mobile is a mess and it may be the responsibility of the W3C to ease the headaches of cross-device mobile development. The evolution of jQuery may also be motivated in the same way. I think for all these technologies it will be about bettering smartphone experiences and development.”

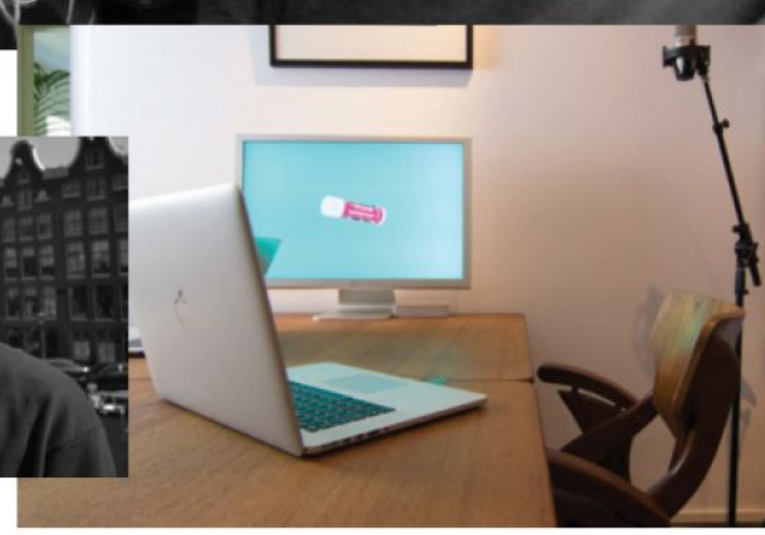
With Johnny also commenting: “Whenever we do a very immersive site they almost always need a separate mobile site because the feeling of the desktop site just doesn’t translate to such a small screen. For grid-based sites that have clear rules then responsive is a nice solution. But the more we do mobile sites, the more different we want to do

As a small agency, individual creativity is always nurtured in the agency

World-class digital experiences are created lovingly at the Cartelle studio

Cool dudes Johnny and Stevijn in their native Amsterdam





industry insight

Johnny Slack, Partner and Creative Technology

> “WebGL is what is most exciting to us. We started as a Flash shop and when Stevie decided to crush Flash – that period will go down as the most boring time in web development. When three.js hit, it shook everything back up again”

them from the desktop site. I think responsive encourages you to try and squeeze a desktop design into a mobile screen, when often it needs to feel completely different – more like an app.

"WebGL is what is most exciting to us. We started as a Flash shop and when Stevie decided to crush Flash – that period will go down as the most boring time in web development. When three.js hit, it shook everything back up again. There were far less rules and we could create more freely. We first applied it commercially on our G-Star New Denim Arrivals site – a WebGL-powered commerce experience. From then on, we have been doing a lot of WebGL work."

As Cartelle work with a very close group of friends and colleagues, what advice would they give to anyone who may be looking to work with them? Johnny continued: "We always like when people have work that is self-initiated. Like they just made it up and executed it because they really wanted to – passion projects tell a lot about how involved that person is in their craft. Our advice would be to try and not get marginalised in your company, try to keep evolving and always have your own personal projects."

The future also looks bright for Cartelle, as they continue to carve their own special niche in the digital space that they have become masters of. Stevijn concluded: "We are focused on creating interactive entertainment content rather than just websites. We have two games coming out this summer for one of our dream clients, so be on the lookout for those. We also have more collaborations in the works with some of our favourite musicians which we're really psyched about and hoping they take shape."

Cartelle prove that smaller agencies can punch well above their weight when the work that is created offers a new perspective to brands and their customers. With a deep understanding of how the digital landscape is evolving, Cartelle are shaping new exciting experiences.

The site also celebrates the life of hip hop artist J Dilla

The Dilla Dimension was launched on Dilla Day 2015

The Dilla Dimension

dilladimension.com

The Dilla Dimension is a 3D interactive, short film that tells the story of two sugar-coated souls and their psychedelic journey through outer space.

Each step of their voyage is crafted to a selection of tracks from J Dilla's classic album, *Donuts*. An alternate universe, blanketed with hypercolour sugar and fluorescent glaze, gives birth to an unlikely cluster of doughnut holes.

Suddenly, they are split apart and scattered across a sea of hypnotic, interstellar chaos. But the impossible takes shape when a love is sparked, leading to a desperate search across space for one another.

The project was self-initiated and we wanted to show what can be done if you apply the latest web technologies to not only hip hop, but also the music industry in general.

It can be seen as a challenge to contemporary music videos and showing musicians and artists what is possible if our industry collides with theirs.

The Dilla Dimension was launched on Dilla Day 2015 to celebrate the ninth year of the album's release and the life of one of hip hop's most innovative and influential figures. Dilla Dimension makes great use of a league of new, awe-inspiring technologies to create a completely digital film that's been made with real-time, generative effects.

It was recognised as Adobe / TheFWA's Cutting Edge site of the week.

CARTELLE

WEB ..... cartelle.nl
FOUNDERS ..... Stevijn van Olst, Johnny Slack
YEAR FOUNDED 2009
LOCATION ..... Amsterdam

SERVICES

> Games and digital experiences
> Interactive music videos
> Campaigns and product launches
> Mobile sites and applications
> Interactive concepting

TIMELINE

2009
Cartelle opens up in a former toyshop in Amsterdam and launches cartelle.nl.
Number of employees **00** (just the 2 founders)

2010
Launched a new award-winning platform for Scotch & Soda, our first client.
Number of employees **00** (just the 2 founders)

2011
Launched several adidas campaigns eg adidas Then Now and Forever.
Number of employees **01**

2012
The relaunch of Cartelle.nl features our first explorations into WebGL.
Number of employees **01**

2013
Launched Converse Street Ready which gets acclaim from Google and FWA.
Number of employees **03**

2014
G-Star New Denim Arrivals launched and we exceed 20 FWA awards.
Number of employees **03**

2015
Cartelle advances into net art with projects like Dilla Dimension.
Number of employees **03**

Changing Percep

tions

Branding, interactive solutions and visual communication in all its forms converge at Phoenix The Creative Studio. Passion and technical skill complement each other, and this results in startling campaigns that speak to their audiences

The team go through a client's moodboard on the bar, whilst enjoying an espresso

Originally founded in 2001, Phoenix The Creative Studio strives to make every brand they work with and every campaign they create not only communicates with its intended audience, but also touches their lives. With open and fresh approaches to what branding means in today's digital spaces, Phoenix has its finger on the pulse of the zeitgeist.

Fouad Mallouk, president and CEO of Phoenix The Creative Studio explains his journey to starting Phoenix: "Victor Mallouk, my grandfather, [was] a successful businessman between 1935-75 in England and Egypt, but left Egypt for Canada after the last war. He then started a small design and print brokerage firm, VN Graphic and Printing, which was enough to sustain my grandparents, but he didn't want to expand his business.

"As a younger man (16-17) I was bored with school so my father suggested I help my grandfather until I figure out what I wanted to do with my life. Watching my grandfather work, I soon discovered that my true passion was the combination between business, design and marketing.

who Phoenix The Creative Studio
what Phoenix The Creative Studio is a marketing agency that specialises in branding, interactive solutions and visual communication in all its forms
where 384, Saint-Jacques West Street, Montreal (Qc), H2Y 1S1
web phoenix.cool

Key clients

DentsuBos

Tonik Agency

The/Studio

Fight 4 Pride

SIR Solutions

"I also studied graphic design and soon realised that my artistic skills where more defined as a project manager, critic and as a director of other artists rather than designing myself. I also found [that I had] a profound love for business development.

"In 2001, at the age of 21, I took everything I had learnt from my grandfather and started Phoenix Media. The first major decision I took was to hire a graphic designer, and a while later several more. As the years went by, I realised that the printing industry had not only reached its peak, but was in decline. So one day, I sold a website. When I walked in the office and told my graphic designer, Louis Paquet who is now our creative director, what I had just done and he looked at me saying 'Are you crazy?' as I've never done that before. So I replied 'Well, you need to figure something out, because we have to deliver it by this date.' A new chapter was born for us.

"Following a severe accident I had a chance to re-evaluate my business and also what I wanted to do. I realised that my business needed the X factor. Following that epiphany, I realised two major things. Firstly the client is not always right, so we needed to build a business/brand that is for the employees rather than the client. And secondly we needed to pick our clients, suppliers and staff based on their values rather than on their expertise alone.

"After that, everything changed. Phoenix The Creative Studio, an agency that specialises in branding, interactive solutions and visual communication in all its forms, was reborn from the ashes of its predecessor. We rebranded, cut most of our clients and challenged everything.

"In 2014, after a few conversations with a successful friend of mine, Étienne Borgeat joined Phoenix and became a partner in the agency. And Louis Paquet – who is one of design's rising stars in North America – having devoted more than ten years of loyalty and demonstrated a huge talent, became creative director and partner later that year."

Since the 'rebirth' of the agency has this changed how Phoenix attracts its clients? "I would say that we are slowly building a strong reputation," Fouad explained. "Today, more than 75 per cent of our workload comes to us. Ever since we established our new values, we have chosen our clients based on if they have at least one of these criteria: firstly, if we will get a certain creative freedom, or 'carte blanche' and we can showcase the project in our portfolio. Secondly if the budget set aside for the project is fair or substantial. And thirdly, if the project is very fun and the team has a lot of pleasure working on it."

"Phoenix The Creative Studio, an agency that specialises in branding, interactive solutions and visual communication in all its forms, was reborn from the ashes of its predecessor. We rebranded, cut most of our clients and challenged everything"

Amongst the many projects completed by the agency is there one that typically defines the Phoenix ethos and why? Fouad explained that this was The Agency Survival Kit (**agencysurvivalkit.com**) "Because it is a custom project for agencies like Phoenix. The goal was to create a product that speaks directly to agencies like us. Whether it is a small design

agency in Australia or a huge New York advertising studio. This is a project we did mainly to have fun and for the thrill, without constraint and maybe in hope of getting some attention. Everyone had fun doing it and we are proud of the results. The project represents us well."

Phoenix as an agency has evolved and changed. Louis outlined how the tools they use have also developed to meet the demands of their clients. "HTML5 is not just a new version of HTML but also a significant improvement to the language's UX capabilities," he began.

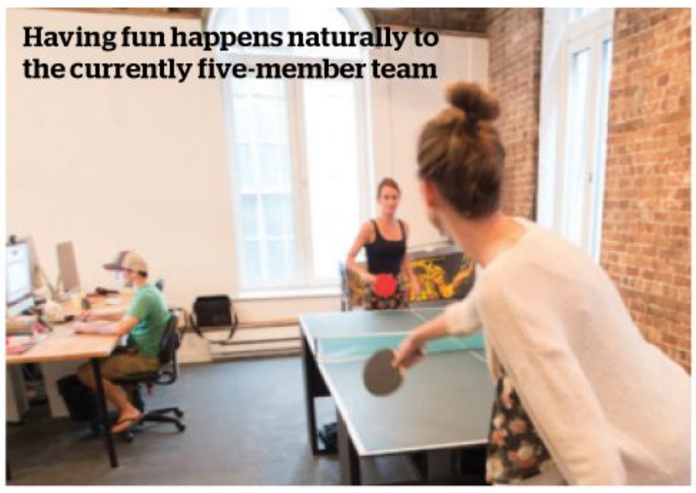
"It's the most significant update of HTML in the last eight years. It promised improved user experience, increased browser features, cross compatibility and more – and delivered. In the near future, capabilities of HTML 5 will offer functionality for the construction of complex web applications including audio, video, geolocation, scalable vector, 3D and more. They will become more of web applications rather than simple websites without platform dependencies. Performance will also no longer be a problem.

He continued: "Web application and applications then can speak to environmental

Industry Insight

Fouad Mallouk
President and CEO, Phoenix The Creative Studio

"Skills can be learned. Values however are different. You either share the same values with someone or you don't. There is more to a good employee than his or her productivity. He or she has to believe in what they do and be proud to work there."



Having fun happens naturally to the currently five-member team

The Phoenix staff are big fans of sports team Montreal Canadiens

Phoenix are masters at providing modern and refined branding

Namale

creationsnamale.com

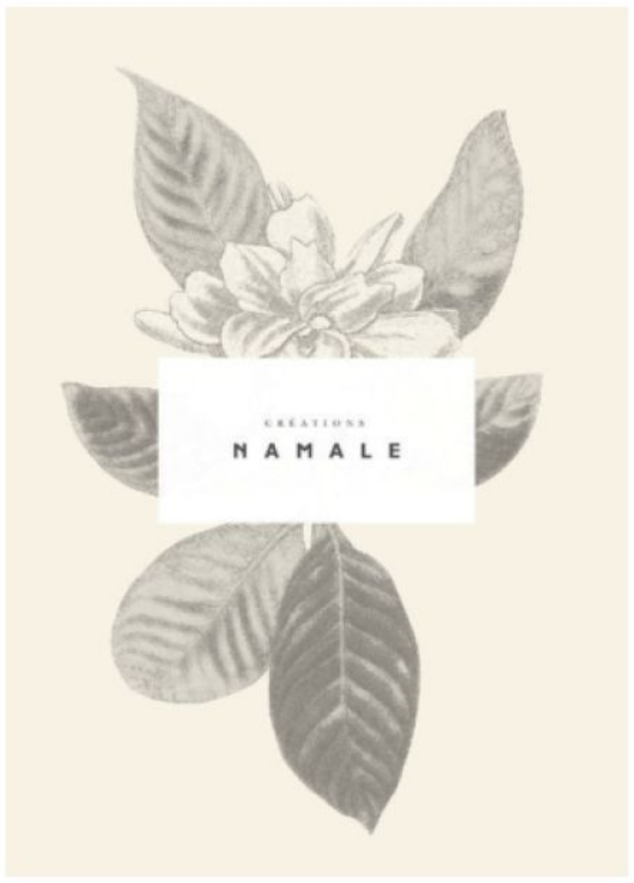
Créations Namale is an online jewellery company selling handmade unique jewels, designed and produced entirely in Quebec with high-grade materials and semiprecious stones.

Phoenix's mandate was the creation of the image, the development of marketing strategy, design and web development, photography, the stationery and all of the collateral for the jewellery company.

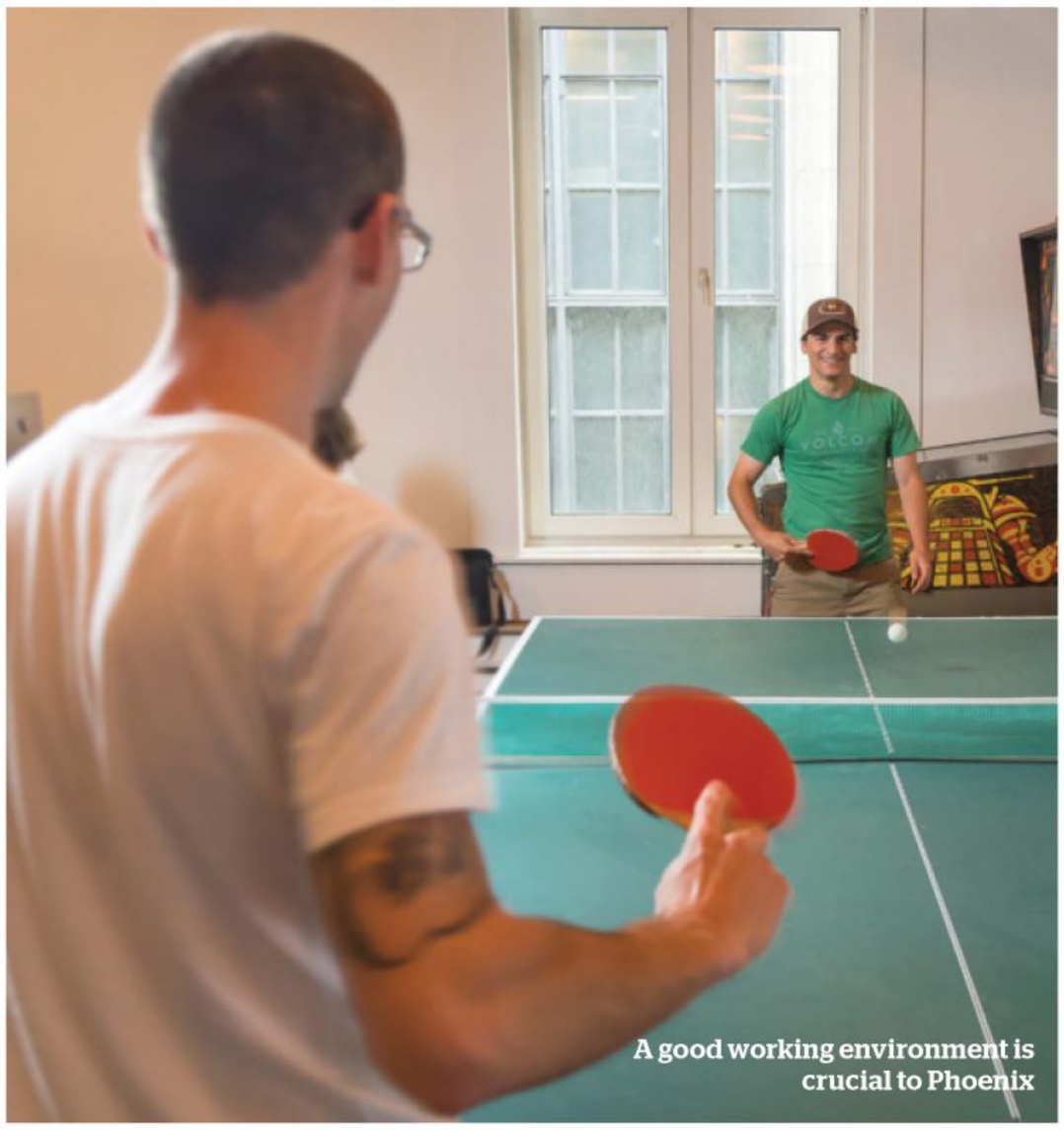
The primary goal was to put the jewellery in the spotlight in an original way, in order to increase sales of Namale. "We want people to discover the jewels of Créations Namale and go to the store and purchase online.

"To get there, we wanted to showcase the jewellery, but not in a traditional way. We wanted to show them in an original 'lookbook', emanating the tone of the company. So we created a refined and clean branding for Créations Namale that also carries over to the website too."

The site needed to present the products of the artist. In addition to seeing the photos of the products in the lookbook, the user can also learn about the jeweller, by means of an innovative and creative navigation, with a sense of serenity that respects the tone of the products. Since the new website went live it has already brought several new clients on the shop. Sales has increased and the jewellery artist has enjoyed the high visibility her new website brought her.



Projects need to have creative freedom, a fair budget and be fun to work on



A good working environment is crucial to Phoenix

A Phoenix employee's happiness goes beyond productivity

elements to act as real augmented reality. For instance, asking for direction to the bathroom during a concert and getting arrows projected on the floor. We have seen the usual light, temperature, and other home automations (or domotics), but imagine pushing the boundaries and see where this can take us."

As an agency that strives to be different, how does Phoenix find the right people to create its ground-breaking campaigns? Fouad explained his approach: "Skills can be learned. Values however are different. You either share the same values with someone or you don't. There is more to a good employee than his or her productivity.

"He or she has to believe in what they do and be proud to work there. Also, you can have a few employees that are the best at what they do, that does not mean that they work well together or can communicate efficiently.

"Setting a good work environment is fundamental. If you want to step into this industry, you have to be innovative, creative, open-minded and have a 'joie de vivre'. Skills will only take you so far, but attitude will take you to the top."

Phoenix is an agency that was literally reborn from the ashes of a previous business. With a core drive to change communication, Phoenix continues to push what is possible with today's tools. Always innovative, this agency defines the digital spaces its works within.

PHOENIX

Web – phoenix.cool
Founder – Fouad Mallouk
Year founded – 2014
Current employees – 5
Location(s) – Montreal, Quebec

[1] Web design and development
[2] Branding and identity
[3] Graphic and advertising design. Photos and video production
[4] Marketing strategies and campaigns
[5] Social media content creation, strategy and management

TIMELINE

Year	Year	Year	Year	Year	Year
Early 2014	Mid-2014	January 2015	February 2015	June 2015	July 2015
No. of employees	No. of employees	No. of employees	No. of employees	No. of employees	No. of employees
2	3	3	4	5	5
A new cycle begins. From the ashes Phoenix The Creative Studio is born.	Étienne Borgeat joins Phoenix The Creative Studio and becomes a partner.	Canadian firm Paprika mandates Phoenix to create their website.	Given Awwwards nominations for Site of the Year, User's Choice and Agency of the Year.	Phoenix designs The/Studio's branding, identity, website design and much more.	In barely 12 months, Phoenix has been given 6 Awwwards, 3 FWAs and 6 CSSDAs.

ALL OF US

Placing design at their epicentre, AllofUs has developed cutting-edge design solutions that continue to evolve and shape the physical and digital spaces their client's customers inhabit

who AllofUs
what AllofUs is an award-winning digital agency specialising in the planning and design of customer experiences for the interconnected age
where 7- 11 Herbrand Street, London, WC1N 1EX
web allofus.com

Key clients

Google
British Museum
Bastei Lübbe
V&A
Ikea

With diverse yet complementary careers that led to the founding of AllofUs, the agency has moved in and out of private ownership to arrive at its current structure that will see new products and services that could disrupt several current industries before too soon.

"The original founders all met at a well-respected interactive design agency," explained MD, Phil Gerrard. "We were, in effect, the management team at Digit. My background is in advertising where I also got involved with digital early on building big websites for clients. When I met the guys at Digit I thought they were doing something really interesting."

Phil then explained the initial ideas at AllofUs: "Using terms like 'creative technology' and 'user experience' has become commonplace, but back

AllofUs continually strive to meld the digital and physical worlds together into ground-breaking design

then – this would be around 2003 – the team was using sensors and other physically interactive ideas to develop new digital campaigns. Installations for exhibitions, for instance, really showed what they could do with the technology. This opened my eyes to what could be done.

"We could see that there was an opportunity to start a new type of agency, which became AllofUs. What we wanted to do was to take design as a principle foundation but also drive what could be achieved with a real experimentation of the technologies we had available. Those two principles drove what the original founders of AllofUs wanted our new agency to be about. So we set up AllofUs and then later sold it to Interpublic. However, we bought back the company so we are independent again."

Being independent again of a parent company is a strategic move that can enable AllofUs to forge ahead with their own unique vision for what an interactive, immersive digital experience looks like today. The three core founders, Phil Gerrard, Orlando Mathias and Nick Cristea, of the new AllofUs offer a set of complementary skills that, when combined together, places the agency at the forefront of design.

Phil has a background in strategic planning and advertising; Orlando has a background in fine art and is the director of R&D, placing innovation at the heart of each AllofUS project; and Nick is a strategic director from a product design background, so he has been looking at how the agency can bring product and industrial design to the digital space.

"We became one of the early developers of product design in the digital environment," said Phil. "I think these skills have enabled us to continue to develop meaningful platforms that have stood as innovations in their sector. A good example is the Sky Player that is now Sky Go. We were one of the first agencies to offer product services design."

As the original founders were close working colleagues before AllofUs, this has fostered a level of trust that has meant that the skillset that the agency contains has been applied to each project with innovation. It has marked the agency as one that not only works in the digital space, but also one that pays close attention to the wider environment their clients inhabit.

Phil continued to divulge: "The agency is strong, I think [that this is] because we have a diverse set of skills, but we all get on as mates and have a similar drive to develop and innovate with technologies. When we set up our store, we didn't want to focus too much on marketing and communications. We never saw the creation of digital content and its link to advertising as the core of our business. Design is what we all wanted to do, but [we wanted] design that creates meaningful experiences using technology and the ethics of good design to move the conversations on.

INDUSTRY INSIGHT

Phil Gerrard, MD and cofounder

> "We never saw the creation of digital content and its link to advertising as the core of our business. Design is what we all wanted to do, but [we wanted] design that creates meaningful experiences using technology and the ethics of good design to move the conversations on"

"We all had this shared collective interest in the cultural sector. So we started to pick up a lot of work from museums and galleries, simply because they were one of the sectors to really embrace new types of technologies. This gave us a head start with how you go about designing a space using sensors and how technologies could be integrated into physical objects for exhibitions.

"The museum and gallery work sat alongside our website work. Over the last five years we have seen how mobile has become the dominant platform. We have also witnessed the development of new sensor technologies. Together these two elements are driving how digital design is evolving. We have always taken an holistic approach and looked at how products, services, the internet and mobile can all be brought together to create new user experiences."

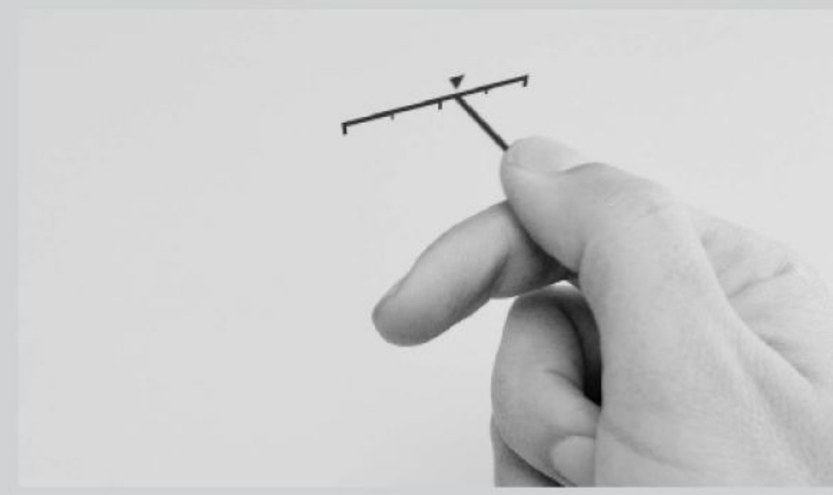
Design then is the core driver behind every piece of work at AllofUs. It's the design sensibilities coupled with technical prowess that brands are looking for when they want to develop multilayered campaigns that speak to their customers. "We try and sit above the technology, the platforms, and

GOOGLE SOLI

google.com/atap/project-soli

In early 2014 we became a partner with a self-proclaimed 'band of pirates' called ATAP (Advanced Technology and Projects). ATAP is Google's maverick team of engineers and they pursue projects in a fast-moving system designed to accomplish in months what usually takes years. They were on course in developing a new gesture-based device utilising radar tech, announced at Google I/O 2015 and named Project Soli.

Soli utilises the unique capabilities of radar technology, essentially tracking submillimetre



gestures at high speeds and accuracy with the added feature of gathering this data whilst hidden under a range of different materials.

Google understood the potential of this technology and approached us to explore real-life uses for a large array of different applications.

Our explorations took us into many industries from automotive to healthcare and the world of IoT. We worked closely with the ATAP teams.

Now that the ATAP team has created the hardware we will be utilising the SDK to prototype and bring to life some of the concepts developed over this period.

Placing design at the centre, brands come to AllofUs to develop new interactive environment for their customers

add our design expertise that utilises these technologies to deliver these new experiences," said Phil. "We actively don't specialise in one particular technology or design types. Taking a wider view allows us to stay relevant and to use the latest technologies for our clients."

Naming your new business is vital to get right, but can often become a long drawn-out affair. AllofUs struggled to name the agency they wanted to start, but a chance conversation resulted in a business name that encompasses what AllofUs stands for, as Phil explained: "The name we came up with was really by chance. We used to have meetings at my flat and sit in the garden and discuss the work we were doing. By the end of the meeting we had a few drinks and our attention would turn to what to call our new agency. We were having real problems agreeing what our name should be when someone said that the agency is about all of us.

"One of our original founders was a graphic designer. He saw something in those words and went away and created a logo design. It took us a while to get used to, as we did initially think it was a bit altruistic, but we could see that the name did say something about our business, which is at its heart all about collaboration."

Being established for over a decade, AllofUs has a client roster that includes some of the world's leading brands. What is most striking, though, is that these brands are diverse and cover a wide range of market sectors. This has been deliberate, as AllofUs actively ensure they are not seen as an agency that specialises in one particular type of work. Placing design at the centre of all they do means that these principles can be applied to any brand from any industry.

Phil explained: "As we have been going now for over a decade, we are well established so a lot of the clients we work with have approached us. Our contacts also are a great help when it comes to locating work. And of course our website is critical in attracting clients, as they can see the kind of work we have been doing. We don't do a great deal with aggressive lead generation. Most of our work now comes through recommendation, which we think is the best way of expanding our client base.

"The variety of clients we have is diverse. We like to work with startups, as this is very much a meeting of minds. We also have corporate clients.

> "We actively investigate new technologies... which can advance the user experience"

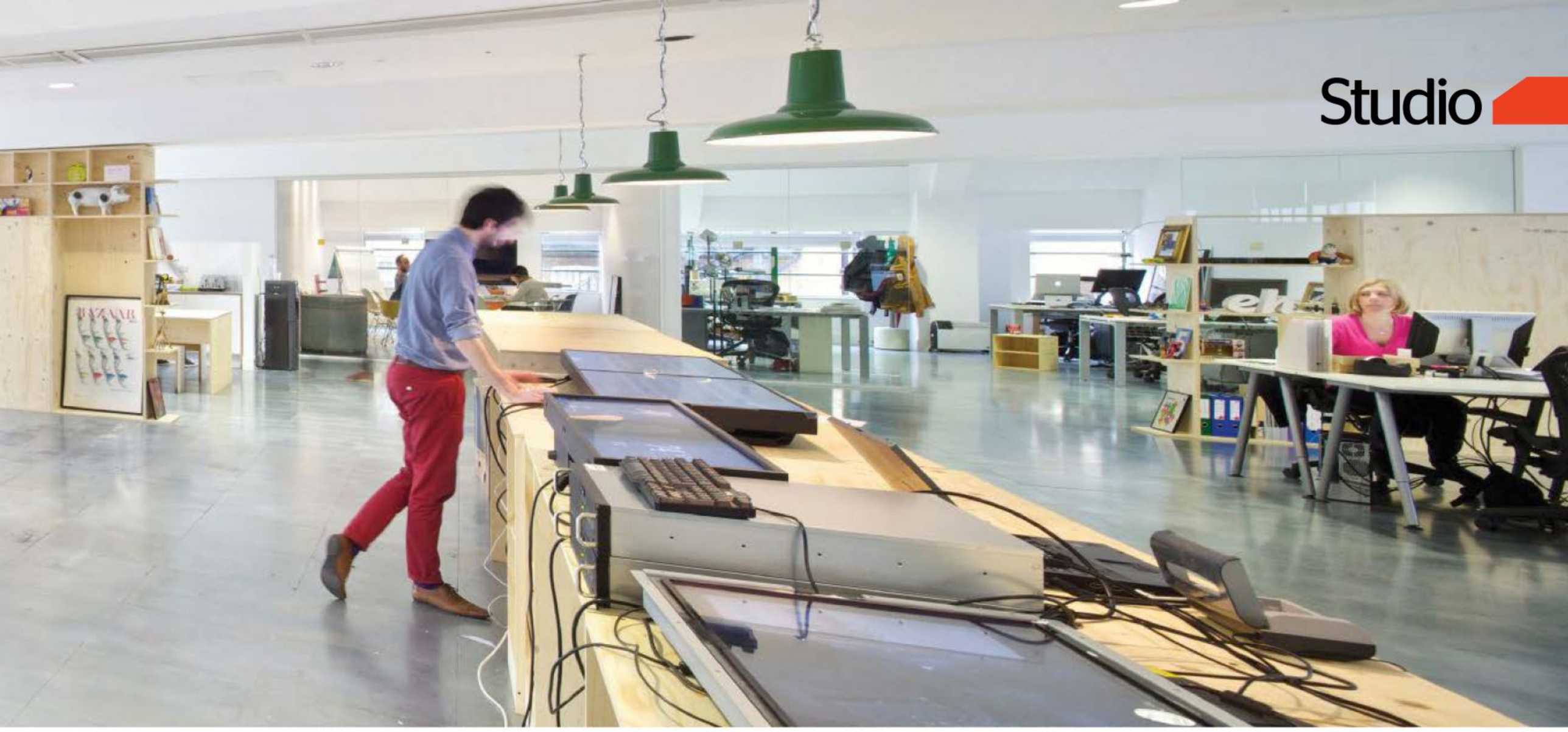
This range means we can work on diverse projects and also helps us attract the right talent to our business. And because of the nature of the work we do, we tend to also get procured by a business. We will work with them to achieve their goals. Of course we look at new clients from a business perspective, but we also ask whether we can achieve the Holy Grail of delivering a project that excites us and also meets the client's brief.

Michael Chase is the creative technology director at AllofUs. He explains their approach to development: "We tend to approach all of our projects with the essential focus being on both the user's needs and the client's requirements for the desired web and/or mobile web platform. This focuses our approach and allows the technologies to fluidly surface directly from the balance of these two audiences. As a result, we are technology agnostic, as it is just a means to an end for AllofUs.

"Some of the most recent solutions we have engineered utilise Sass, which follow SMACSS, BEM and OOCSS methodologies. We also follow a strong component-based development approach in which we use task runners and module bundlers such as Grunt, Gulp and WebPack. All of our solutions are designed and developed with responsive technologies such as CSS3 media queries, flexible images and fluid, proportion-based grids.

"As new technologies, platforms, devices, frameworks and methodologies are released we can shift and adapt our approach and methods to include them in our active development process. We actively investigate new technologies and research emerging digital tools, which can advance the user experience and help us create a dynamic, powerful and performant web applications. AllofUs develop for multiple platforms which include (but are in no way limited to) iOS, Android, openFrameworks, Unity, WebGL and so on, so we have the distinct advantage of being able to take advancements from other mediums and platforms and apply them to our work within the web space. This allows us to stay nimble and agile and not tied to one single technology or vendor yet constantly progress forward while maintaining innovation."

Agencies can become known for a particular type of work that informs the ethos of the agency. For AllofUs, their ethos for work comes directly from their founders. "We do have a set of values that mirrors the values of our founders," Phil explained to us. "We have always wanted to be known as an agency that understands great design, progresses product development and places the user at the heart of the experience. We wanted to move on the conversations.



"These are the values we want to be known for. We have always been chameleon-like, refused to be pigeonholed and [have been] able to evolve as the market changes. What is certain is that technology will constantly reinvent itself. As an agency we want to be at the forefront of that and position ourselves to take advantage of those changes as they take place. I think we have done that quite successfully."

How agencies work is always fascinating to decipher. Phil explained how AllofUs is organised: "We are about 20 people full time. We organise the company with a management team who are also heads of department. We have the user experience team. They are a mix of strategic user experience designers who can understand user research and develop our solutions with the insights that this material gives us.

"The design team come from visual and graphic design backgrounds. Other members of the team have expertise with mechanics and methods. They take the work that the user experience team have done and turn that into a design solution. And the development team concentrate on the front-end design, whether this is using web technologies or mobile across the leading platforms."

As a close-knit team, AllofUs takes their time to find the right people to add to their roster of employees. In an environment of design and digital tools that moves and evolves so quickly, when someone stands out from the crowd AllofUs will immediately take notice.

Phil explained their approach when they are hiring: "When we are looking for a new team member we tend to instantly know whether someone will be a good fit for us. We look for someone that has a multidisciplinary approach to the work they have done. We like people with vision. We also look for detail in the execution of the work they present to us. Collaboration is obviously very important for us, as we work in teams that need to work closely together.

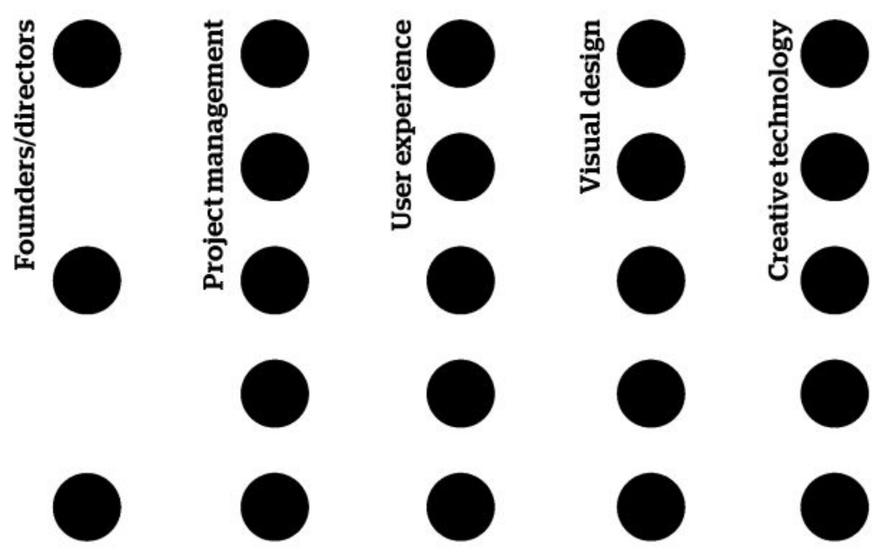
"We tend to get seduced by people that are already doing their own thing. They have understood the market or industry and have produced work that is impressive in that context. You know if someone has gotten themselves a placement working somewhere, you'll look at their CV more favourably, as they have clearly demonstrated they can already work in a team and produce great work.

"Also we take on a lot of graduates and we offer placements. We go into universities and give talks about our industry and what we are looking for when it comes to new employees or when we need to bring in outside help for a project. My way into the industry was actually via a placement, so this can be a great way to get your start."

AllofUs is entering their rebirth as an independent business, which the company relishes as this gives them the freedom to pursue the work that they really want to do for their clients. "As we have just bought the company back, it's almost a whole new beginning for us," Phil concluded. "We want to continue to move the conversation on and do work that is different. We want to develop an agency with its own IP that uses different remuneration models. There are a couple of new projects we are working on at the moment. One is to develop a new reading platform for fiction that we think will be quite disruptive and is being backed by a major publisher."

What is clear is that AllofUs is now in a position to change the conversation about digital design and how our environment interfaces with campaigns and content. Looking past the screen, AllofUs is designing the future, today.

AGENCY BREAKDOWN



AllofUs™

WEB ..... allofus.com
FOUNDERS ..... Phil Gerrard, Orlando Mathias, Nick Cristea
YEAR FOUNDED 2003
LOCATION ..... London, UK

SERVICES

- > Responsive web
- > Multidevice apps
- > Physical environments and objects
- > Brand identity
- > User experience planning

BREAKING BOUNDARIES

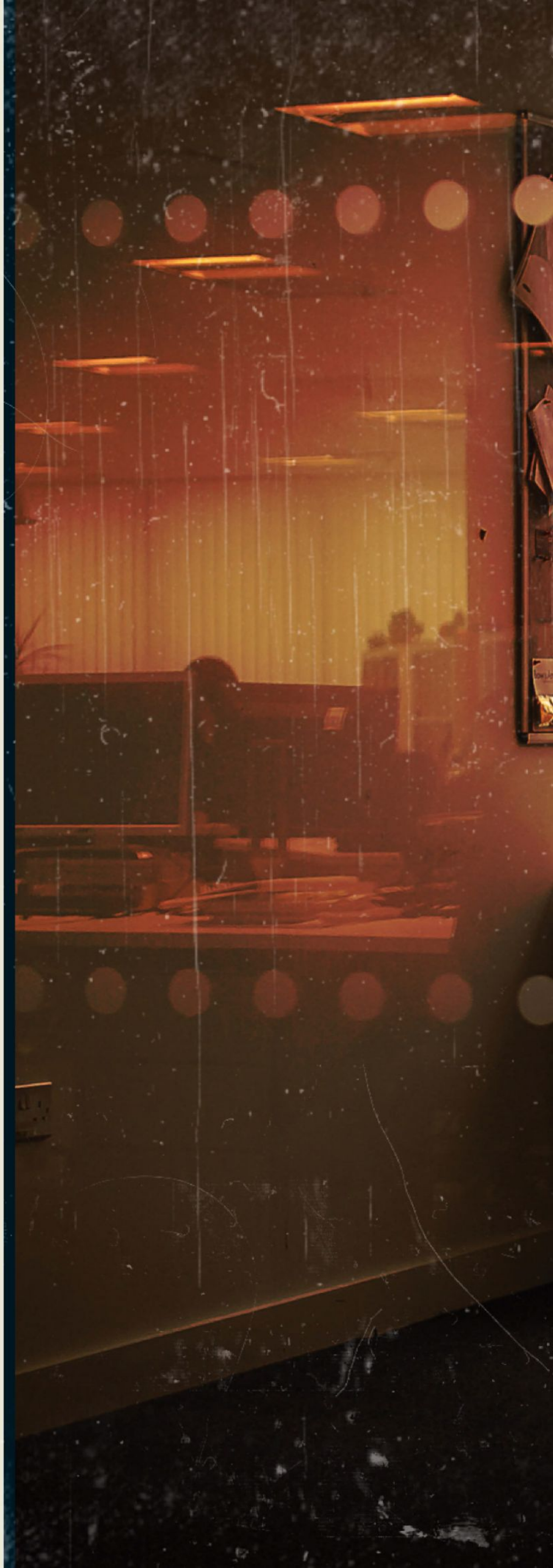
Born out of academia with the aim to explore the impact of digital on learning and communication, Amaze has been at the forefront of digital design for the last two decades and leads with a skill spectrum that makes it a true pioneer

Amaze was established in the year 1995, out of the Learning Methods Unit of John Moores University, Liverpool – with a global scope, Amaze has developed to become not only one of the world's leading digital design agencies, but also a consultancy that looks closely at how human behaviour, branding and the digital landscape integrate together.

CEO Natalie Gross explains: "Founded by the late digital media luminary Roy Stringer, and led by him and Peter Fowler, the unit was originally formed to explore the effects of digital and emerging new media in the way people learn and communicate.

"Amaze was born at the very dawning of the World Wide Web. It recognised the power of the new multimedia, the convergence of computing and design; it understood how images on a screen could sit side by side with narratives. It recognised, immediately, that a new form of communication was being born and that this would impact on information, on learning, on understanding and on business too."

With a basis of learning and understanding, naming the new company was vital to get right. "The name Amaze does have a deeper meaning,"



who Amaze
what A leading, full-service digital marketing, technology and commerce consultancy. Born 20 years ago inside the Learning Methods Unit of John Moores University, Liverpool
where Royalty House, 72-74 Dean Street, London, W1D 3SG
web amaze.com

Key clients

The Co-operative Group
Virgin Trains East Coast
Toyota
Lexus
ASICS

Amaze believes that it's an exciting time to join the digital industry and change the rules regardless of whether it's in relation to learning, shopping, socialising, travel or health care

said Natalie. "In the early years Roy Stringer had taken the idea of Theseus, in which a maze was the central feature (usually referred to as a labyrinth in the Greek story). [He then] used it, as an extension of the noun, an imperative (Amaze Me!) that seemed to chime in exactly with Stringer's presentation of the work to outsiders. Amaze was established in 1995 in the very early days of the internet, and this meant domains weren't quite as difficult to get hold of as they are today."

Getting a presence on the web right was a priority when Amaze was founded and continues to evolve as the agency expands its client base. As Paul Musgrave, creative director at Amaze comments: "As most digital agencies will agree, getting the chance to concentrate on their own site is very challenging, and because of that, it is often the last one to change. Although our own site is important to showcase our expertise, our core focus is always first and foremost on client work. We are currently in the process of creating a new company website – a testament to our original mantra: 'To inspire original thinking through a deeper understanding of technology and human behaviour. We set ourselves a challenge to express some of that mantra through our new brand mark, which has been born from technology and is literally fed by human behaviour. It follows in our tradition of making creative use of web technologies, in this case WebGL, which is how we render our living logo to the screen."

Paul continued: "We are using motion detectors to 'see' the ebb and flow of people through our offices. We've set up network sensors to measure our consumption of digital content. Other sensors watch the trends in consumer interaction with our technology, and active avatars measure activity in our social spaces. All of this in turn will harvests data, which then manifests itself in three dimensions and forms part of our living brand mark. It's an exciting and interactive part of the new website."

Over the last two decades, Amaze has developed its skills and consultancy services. Has the company developed an ethos that can be defined? Natalie continued: "We pride ourselves on developing long-term relationships with our clients, accompanying them on their digital journey and supporting their ongoing business needs and

> "Although our own site is important to showcase our expertise, our core focus is always first and foremost our client work"

challenges, rather than just working with clients on a project-by-project basis. Our partnership with Lexus, for example, dates back to 2004. Since then, Lexus Europe has grown to span 32 countries operating in 26 languages, all serviced by a single European hub website with language sites across mobile, tablet and desktop.

"The Lexus account works across a wide range of our differing agency services, from digital strategy through to UX, creative, design and build, as well as implementing and managing its digital brand and marketing developments, product campaigns and CMS integrations.

"Once Amaze launched the site for Lexus, we then provided them with a dedicated team who are responsible for the rollout of a central solution to sites across Europe. The rollout team supplies project implementation and management, content administration, training and support to each individual country marketing department and their third-party agencies.

industry insight

Natalie Gross, CEO

"Social channels are at their most effective when they are used to interact and talk with people. Relationships are two-way and revolve around shared experiences and conversation. The most successful brands are those that realise this and shape perception, rather than leave it to chance. When used smartly, social platforms help us understand what motivates and demotivates people"

ASICS

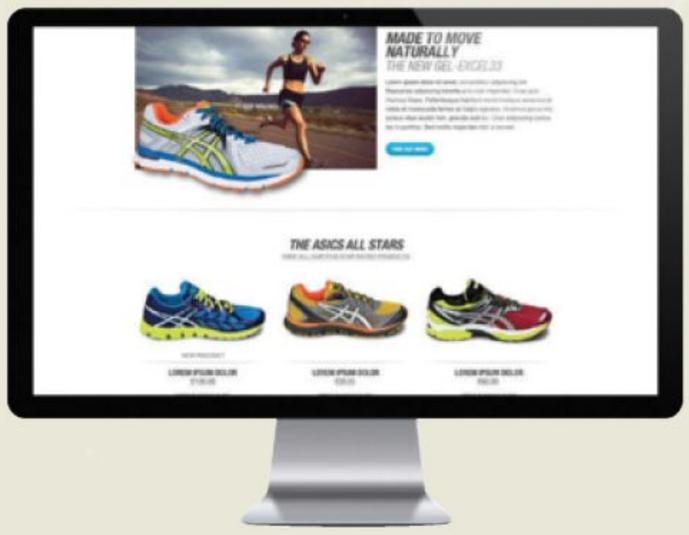
asicsamerica.com

Amaze was appointed by true sports performance brand ASICS to deliver a global eCommerce solution. Our main objective was to assist the corporation by launching its first ever direct-to-consumer channel. We were selected on account of our ability to combine market-leading technology with strategic, technical, creative and brand insights.

We worked with ASICS as strategic digital advisors across technical, brand, creative and global implementation for the whole project. As part of this remit we worked in partnership with ASICS's global HQ and their key regions – Japan, USA, Europe and Australia – to deliver a global eCommerce solution that is scalable and is already successfully delivering to multiple brands that are owned by the ASICS Corporation.

After our initial consultation to select and then architect a technical framework, Amaze began, at an immense pace, to develop the ASICS hybris eCommerce platform. It has been configured using hybris eCommerce accelerators with local integration hubs, templates and merchandising services.

Beyond rolling out the global network of sites, our remit is to manage a continuous enhancement programme for the organisation. As a brand, ASICS is fragmented across multiple markets, categories and target audiences meaning the programme needs to



be carefully considered for each market. Whilst the US operation favours a focus on three to four key sports, Japan operates across an overwhelming range of categories operated by multiple business units.

With long-standing relationships with clients, it's no surprise that Amaze assigns several of their key teams to work on the same projects over the years

TIMELINE

1995 — Number of employees **10**

Amaze is established via the Learning Methods Unit of John Moores University, Liverpool.

2000 — Number of employees **70**

Amaze's first online community for VW is the second largest youth site in Germany.

2002 — Number of employees **100**

Amaze's first pan-continental projects requires diverse solutions across 46 markets.

2010

Unilever is Amaze's first global account.

2012 — Number of employees **200**

Amaze is responsible for transitioning ASICS's business to a global direct-to-consumer sales model.

2015 — Number of employees **250**

Amaze celebrates its 20th birthday.

"Where possible, we also look to push the boundaries and encourage our clients to use the latest or emerging technologies too. Our work with Lexus and Oculus Rift (OR) for instance, was started when OR was in its very infancy and when Lexus was set to launch its new NX: an SUV crossover. The NX looked to target a new, highly sophisticated, tech and design-savvy audience and we were tasked with capturing their attention."

As Matt Clarke, chief technology officer at Amaze, explains: "Our solution was to blend computer-generated imagery (CGI) and OR technology to develop two highly immersive experiences that would allow customers to virtually test drive and configure the new NX, before the car was physically available in showrooms. This, along with the other complementary experiences we developed, was rolled out across the UK and Europe and used the very latest DK2 OR Headsets. [That was] the first time CGI, rather than film-based imagery had been used with the DK2 headsets. The launch was also the first time the industry had seen a mass rollout of new HD devices.

"More recently, we have developed a new virtual-reality driving and racing experience as part of the launch of the new RC F model. The experiences include a pioneering 360-degree video, accessible with an OR headset, as well as via an immersive mobile app, which can be further enhanced into 3D with Google Cardboard."

As some of the projects that Amaze has developed have been over long periods of time, how does the agency approach their working practices? "No one project is the same – so it is very dependent on the complexity of the client requirements," says Matt. "Typically we would assign teams covering strategy, creative, technical, project and account management plus our digital comms (SEO, CRM and social) and data teams. For many of our clients we work with our digital operations services team ensuring ongoing digital governance and support across different markets around the globe."

Amaze, simply because of its long history, has been able to witness first-hand how digital tools have developed over the last two decades. "Whilst HTML5 and CSS3 are our base, front-end technologies, we work on a number of frameworks including jQuery, AngularJS, various Google code libraries and open source code libraries and frameworks," Matt continued. "We concentrate on utilising the best emerging frameworks and developing the best front-end architecture. We are also seeing the evolution of a back-end framework, which is crucial for all back-end technologies. We see that responsive design is the way forward for mobile web and for app development. For native app development we tend to use generic frameworks such as PhoneGap."

Clearly one of the most impactful developments over recent years has been the rise of social media. For Amaze, social networks have become a fertile resource that much of their design services are based upon, as Natalie outlines: "Social channels are at their most effective when they are used to interact and talk with people. Relationships are two-way and revolve around shared experiences and conversation. The most successful brands are those that realise this and shape perception, rather than leave it to chance. When used smartly, social platforms help us understand what motivates and demotivates people: how they feel, how they act and how they behave.

> "By influencing behaviour (beyond simply asking someone to click on a link) people will investigate things further and... complete their own journey themselves"

"Of course, it's not all just about websites. It's also about a channel or application allowing people to complete their objectives. That's why business objectives, aligned with customer objectives, are vital. By influencing behaviour (beyond simply asking someone to click on a link) people will investigate things further and eventually complete their own journey for themselves."

Few agencies have the breadth of experience that Amaze can call upon. How has Amaze approached its recruitment process, and has this changed as Amaze evolved? "Our expectations for a new employee in any discipline are centred around three key areas: hard skills, soft skills and business acumen," said Natalie. "Basic technical knowledge, which we consider as a 'hard skill' is absolutely vital to get the job done, while personal qualities or 'soft skills' are essential for anyone to be successful in the industry: for instance, do they take initiative? Are they inquisitive?

"For anyone looking to enter the digital industry today, they have the chance to change the rules and that's a tremendously exciting opportunity to have. We are in an era of true digital transformation, and the people joining our industry have the opportunity to be part of this.

"In terms of advice, it would have to be: stick to what you are passionate about, keep asking questions about what's coming next and look to discover how these relate to true business goals and challenges."

And what does the future hold for Amaze? Natalie concludes: "This year Amaze turns 20, and we're proud to celebrate two decades of digital innovation. It's been another strong year and we continue to grow globally, as well as develop our commerce practice and service offering, with data intelligence and digital innovation [in particular].

"The launch of Loop Integration – a joint venture between Amaze and systems integrator Contiigo in Chicago – has also seen us expand into new markets something which is only set to increase. Loop is a digital commerce consultancy which specialises in the hybris product suite. The development of our new proposition, Amaze One, a joint venture between St Ives companies Amaze and Occam DM, launched in January 2015. Amaze One is a new breed of customer engagement agency [that works alongside] a seamless mix of data and digital marketing expertise to deliver innovative CRM campaigns for brands. The business looks to handle CRM differently, turning each aspect of every customer interaction into information that creates value for consumers and clients' businesses.

These are just two examples of how Amaze is diversifying, with many more to follow. Here's to 20 more years!

amaze

WEB ..... amaze.com
FOUNDERS ..... Roy Stringer and Peter Fowler
YEAR FOUNDED 1995
LOCATION ..... London, Manchester, Liverpool, Chicago

SERVICES

> Strategy and innovation
> Global design and build
> Commerce
> Digital operations services
> Data intelligence

Special trial offer

Enjoyed this book?

Exclusive offer for new

Try 3 issues for just £5*

* This offer entitles new UK direct debit subscribers to receive their first three issues for £5. After these issues, subscribers will then pay £25.15 every six issues. Subscribers can cancel this subscription at any time. New subscriptions will start from the next available issue. Offer code ZGGZINE must be quoted to receive this special subscriptions price. Direct debit guarantee available on request. This offer will expire 31 March 2017.

** This is an US subscription offer. The USA issue rate is based on an annual subscription price of £65 for 13 issues which is equivalent to $102 at the time of writing compared with the newsstand price of $14.99 for 13 issues being $194.87. Your subscription will start from the next available issue. This offer expires 31 March 2017.

About the mag

Uncover the secrets of web design

Practical projects

Every issue is packed with step-by-step tutorials for HTML5, CSS3, Photoshop and more

In-depth features

Discover the latest hot topics in the industry

Join the community

Get involved. Visit the website, submit a portfolio and follow Web Designer on Twitter

subscribers to...

web designer

Try 3 issues for £5 in the UK*
or just $7.85 per issue in the USA**
(saving 48% off the newsstand price)

For amazing offers please visit
www.imaginesubs.co.uk/wed

Quote code ZGGZINE

Or telephone UK 0844 848 8413+ overseas 01795 592 878

+Calls will cost 7p per minute plus your telephone company's access charge

HOW TO USE FileSilo

EVERYTHING YOU NEED TO KNOW ABOUT ACCESSING YOUR NEW DIGITAL REPOSITORY

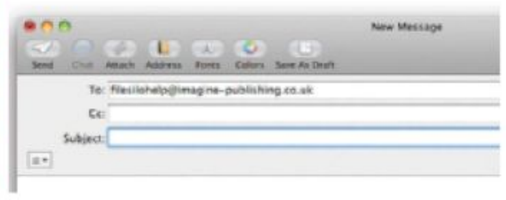
To access FileSilo, please visit filesilo.co.uk/bks-894

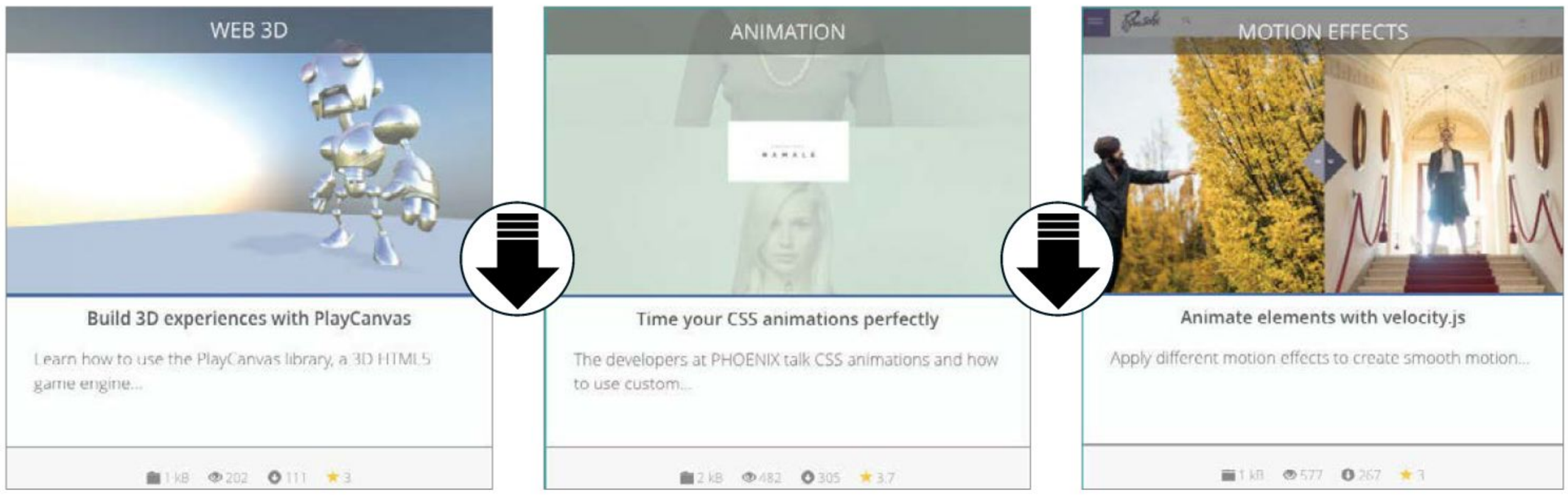
01 Follow the on-screen instructions to create an account with our secure FileSilo system, log in and unlock the bookazine by answering a simple question about it. You can now access the content for free at any time.

02 Once you have logged in, you are free to explore the wealth of content available on FileSilo, from great video tutorials and online guides to superb downloadable resources. And the more bookazines you purchase, the more your instantly accessible collection of digital content will grow.

03 You can access FileSilo on any desktop, tablet or smartphone device using any popular browser (such as Safari, Firefox or Google Chrome). However, we recommend that you use a desktop to download content, as you may not be able to download files to your phone or tablet.

04 If you have any problems with accessing content on FileSilo, or with the registration process, take a look at the FAQs online or email filesilohelp@imagine-publishing.co.uk.





NEED HELP WITH THE TUTORIALS?

Having trouble with any of the techniques in this bookazine's tutorials? Don't know how to make the best use of your free resources? Want to have your work critiqued by those in the know? Then why not visit the Web Designer and Imagine Bookazines Facebook pages for all your questions, concerns and qualms. There is a friendly community of fellow web design enthusiasts waiting to help you out, as well as regular posts and updates from the team behind Web Designer magazine. Like us today and start chatting!

facebook.com/ImagineBookazines
facebook.com/WebDesignerUK

The Web Design Book

The complete guide to desktop and mobile web design

Every aspect of web design explored, from fantastic fonts to perfect plugins you won't want to live without

HTML & CSS

Discover what's new in HTML and CSS and put their power to the test

jQuery & JavaScript

Make your websites more vibrant than ever with the best animations and transitions

Web development

Uncover Google's greatest secrets, unlock the key to free hosting and so much more

Ingenious apps

From desktop to mobile applications, we delve deep into the best app developing

Studio icons

Get a look behind the curtain as we unveil the greatest designers' secrets

Download vital assets for the perfect website

FileSilo

- All the tutorial files
- Interactive video tutorials
- Creative fonts and icons
- CSS + HTML templates
- Free WordPress themes