NEW

HTML5 & CSS3

The Complete Manual

The essential handbook for web designers

Welcome to

HTML5 & CSS3

The Complete Manual

HTML5 & CSS3 were first implemented in 2012. Since then, web design has excelled, becoming an art form that almost seems like technical wizardry. In fact, all of these sites has been created using the written craft of HTML and CSS, with the added help of Javascript and jQuery. In this book we aim to take you from the early stages of web design, all the way to adding responsive elements to your website with the use of brilliant tools such as Bootstrap and Foundation. In no time, you'll be creating web wizardry yourself, so let's get started.

HTML5 & CSS3

The Complete Manual

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
Jen Neal

Senior Art Editor
Greg Whitaker

Designer
Perry Wardell-Wicks

Photographer
James Sheppard

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

ISBN 9781910439418

Part of the web designer bookazine series

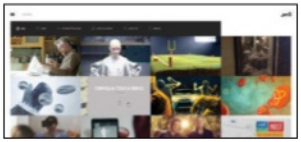
IMAGINE PUBLISHING

Contents

What you can find inside the bookazine

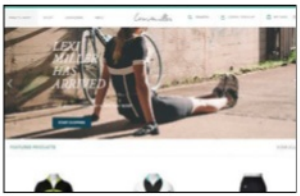
Introducing HTML

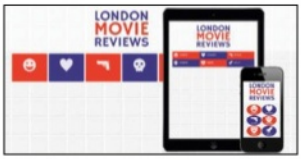
Introducing CSS



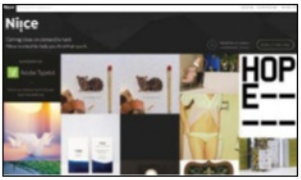
Create your own website

Responsive design





Reference




HTML
5

An introduction to HTML

HyperText Markup Language, more commonly know as HTML, is the basic building block of the web. It provides the structure, content and connection between pages, allowing web designers to create interactive experiences. Easy to learn, it offers limitless possibilities…

The majority of web pages are made up of one or more files that are downloaded to a computer, which in turn are interpreted by a web browser and finally rendered out to screen. At its simplest, a web page is a plain text file that contains special instructions about what kind of content is contained within. These instructions are written in HTML – the language used on the world wide web.

HTML, or HyperText Markup Language to give it the full name, is the core building block of a web page. HTML is a markup-based, human-readable language that's designed to be simple to write, and easy to understand. A markup language is one where bits of data are literally marked as being of a particular type. So, for example, a piece of data that you want to render as a paragraph would be 'marked up' with a paragraph marker.

Marking up

Markers are referred to as tags, and they have less-than and greater-than symbols around them. This paragraph would be marked up with a tag at the start to denote a paragraph, and a matching tag at the end to signify the end of the paragraph. HTML offers many tags for marking up content, and browsers are programmed to interpret these tags, formatting the contents of each tag according to a set of preset rules. This allows the tag to render text in bold, and the (for emphasis) tag to render in italics. One of the major benefits of this system of marking up content is that it's easy to read – you don't need any special software to either write or read HTML documents, as they're just plain text. This makes it easy to create a functioning web page using nothing more complex than a text editor.

An introduction to HTML

What exactly is HyperText?

Why is the language called HyperText Markup? Because the key thing that makes the World Wide Web a web, rather than a series of disconnected pockets of content, is the ability to link one document to another – these links are HyperText because clicking on them takes you somewhere else in the vast web of pages! As well as a basic HTML (text) document, web pages can call in external files (or 'assets') that include stylesheets, images, audio files, video, flash and scripts. The web browser looks at the HTML document when it downloads it, and looks for any external files that are called into the

<right> All websites are built using HTML, CSS and JavaScript

<right> The simple, user-friendly Calendar app interface actually works well for complex agenda management

What is HTML5?

The first requirement of any WordPress installation is the need for web space. There are thousands of web hosting companies who will happily supply space for a small fee. However, to determine what sort of web hosting package is needed the user needs to decide how much web space is needed and how much traffic is expected.

Users can get 200MB of web space and gigabytes of traffic for a very small fee. But if more space is likely to be needed, ie a photo blog, go for more. Don't worry too much about traffic to start with. New sites are not likely to get huge amounts of traffic immediately and this can be changed at a later date very quickly and easily.

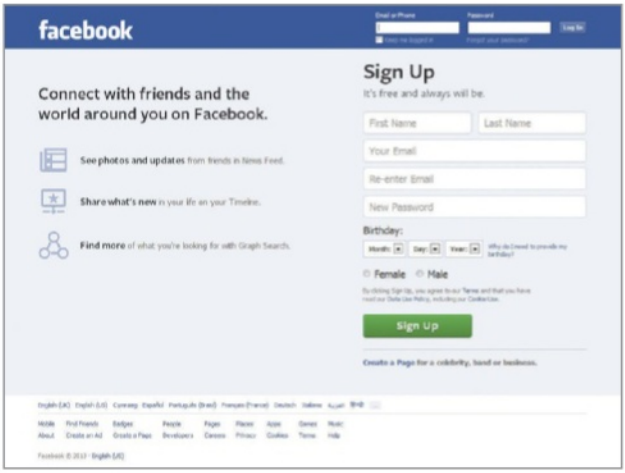
A small UK company that provides cheap and efficient hosting is Z-Host (**www.z-host.co.uk**). It provides packages from as little as £15 a year (100MB of web space and 10GB of monthly traffic), perfect for first time bloggers. Alternatively, choose 1GB of web space and 40GB of monthly traffic for £60 a year. At the other end of the scale there is a popular choice with web designer Media Temple (**www.mediatemple.net**). This offers packages from $20 a month (approx £15) but offers gigabytes of storage and 1TB network transfer rates. Other reputable web hosts to consider are Fasthosts (**www.fasthosts.co.uk**), 1&1 (**www.1and1.co.uk**) and Heart Internet (**www.heartinternet.co.uk**). Once a package has been bought and paid for the web host will send all the details (username, passwords etc) needed to take control of the web space.

To host a WordPress blog at a desired URL, ie **mywebsite.co.uk**, a domain name needs to be purchased. Try **www.123-reg.co.uk**, this offers .co.uk domain names from £2.99 a year and .com domain names from £9.99 a year. Another well-respected domain name supplier is Easily (**www.easily.co.uk**).

If the prospect of finding web space and getting a domain name seems too much like hard work, there is the hosted option. Go to **www.wordpress.com**, click Sign Up Now and all that's needed is an email address. This gives a new user a unique WordPress URL, ie myname.wordpress.com and hosts the account.

An introduction to HTML

<right> Once you know HTML you can create both websites and web apps for the likes of iPhone, Android and even Facebook!



"Every web browser only has to support one kind of page structural language, which is easier for web designers"

<right> Apple's website typifies what can be achieved with just HTML and some scripts and styles

page. If it finds any files called in, the browser will send a message back to the web server asking for those files. The final result you see in your web browser window may be the result of more than 20 individual files, brought together by the web browser and rendered as a single web page!

Recognising an HTML document

You'll notice as you browse around the web that different pages have different names, and more importantly they have different extensions at the end of the filename. Some end with .htm or .html, which are obviously HTML documents, but others might end with .cfm, .php, .asp or .aspx. What are these file types?

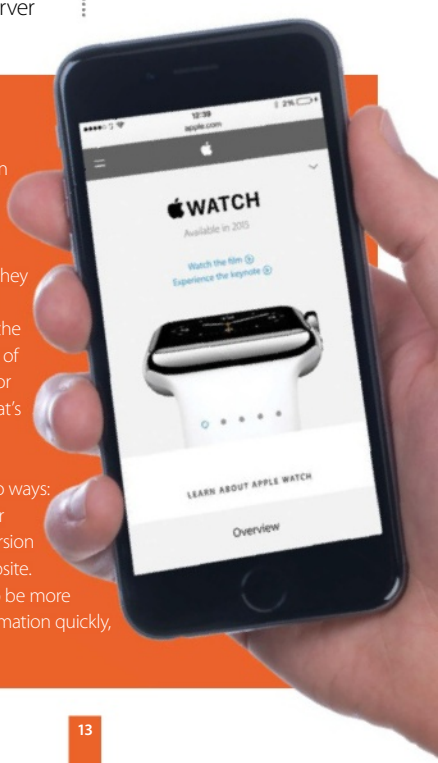
As far as the web browser is concerned, they're just normal HTML documents like every other on the web. What these different extensions signify is that some sort of processing has happened on the server before the page was sent to your browser, allowing the website to generate some unique-to-you content.

All the examples we've noted above have similar characteristics. They're scripts, or small computer programs, that run on the server

HTML & mobile browsing

When you access the web from your mobile phone or tablet, you'll often notice special pages that have been formatted especially for these devices, rather than the full desktop version of a website. You might have wondered whether these pages use a special language especially for mobile devices? The good news is that, just like the rest of the web, they use standard HTML. The only real difference between a page optimised for a mobile device and one orientated towards a desktop computer is the CSS styles that have been applied. The content, and often the structure, of the page remains plain HTML and there's no need to learn special skills or additional languages to be able to create sites that on these devices. That's not to say that you shouldn't consider how your users are accessing the web page and what information they might need!

Typically, a mobile-optimised website will be presented in one of two ways: either it will be the full website presented to render nicely on the smaller screen size that you find on phones and tablets, or it will be a special version of the website that doesn't contain the same information as the full website. This latter approach supposes that a mobile visitor to your site is likely to be more goal-orientated than a desktop visitor, and aims to present the key information quickly, such as 'how to find us', or 'our contact details'.

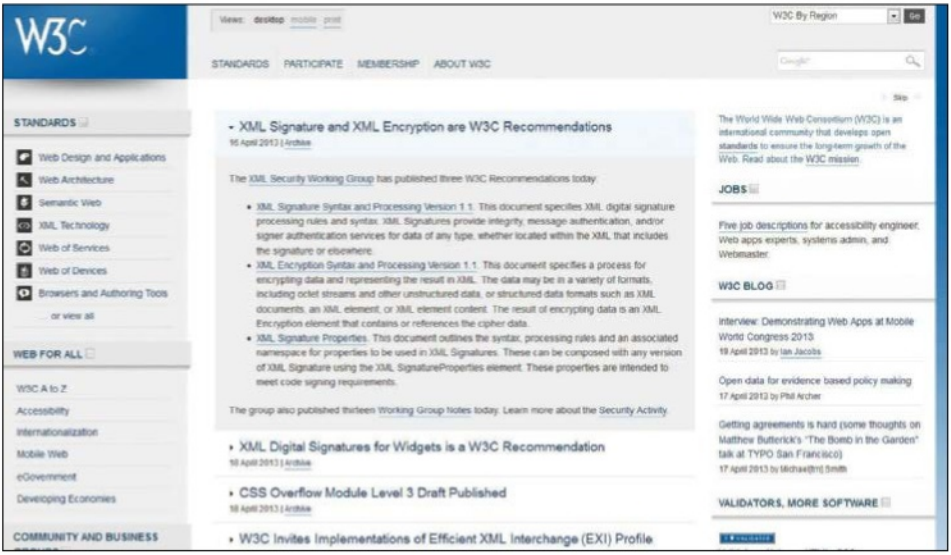


and make some decisions about what content to show. This might be as simple as including your name at the top of the page when you're logged in, or as complex as a fully loaded e-commerce web store, but all these 'server-side languages' output the same final result – plain HTML. It's important that the web works this way, as every web browser only has to support one kind of page structural language, which makes it easier for both web designers and browser vendors, not to mention for users of the web who know that all they need to access any website is a web browser and an internet connection.

What are these semantics?

The HTML format is very accommodating; beyond the required elements of <html>, <head> and <body> you can arrange your content any way you like. This is both a benefit and a potential issue as the flexibility to present (and mark up) content in any way also means that you can quickly end up with a confusing mess! A movement to present information in a semantic fashion existed at the birth of the language, and has seen a resurgence in recent years.

The general idea is that content should be marked up according to its type and importance. <h1> heading tags, for example, should



<below> The new HTML5 specification has been drawn up by the W3C, an organisation set up to act as the gatekeeper for web standards

"Every web browser only has to support one kind of page structural language, which is easier for web designers"

only be used to mark up the most important piece of info on a page, <h2> for the second most important and so on. If you think about the analogy of a book, the <h1> tag might be the book or chapter name, <h2> would be a section header and so on. By using this approach, the HTML describes the importance of each piece of info which helps search engines identify what your page is about, and more importantly special software, such as screen readers that translate web pages into audio for people with sight problems, can make more sense of the document.

In the past, content has been marked up using heading and paragraph tags, but content has been grouped using the same single nondescript tag, . This tag is, by itself, nothing more than a method for delimiting different pieces of information so that it might be used around a navigation bar, or just as likely around an article on a web page. As HTML5 has been introduced, new semantically orientated tags have been introduced that allow designers to mark up navigation with a <nav> tag, headers with a <header> area and articles with <article>. There are more of these semantic tags available, but some browsers are still implementing HTML5 so we're in a period of transition where you'll still see a huge number of plain old tags used.

What about Flash?

Up until the launch of the iPhone, the de facto method for presenting interactive rich content was Adobe Flash. This is a plug-in that sits inside an HTML document, but runs as a separate program within the web browser. When Apple released the iPhone it decided not to support Flash, largely because of the problems Flash had with crashing browsers, and the lack of support for touch screens. This decision helped kick-start a movement away from the plug-in towards the combination of HTML, CSS and JavaScript.

Whereas five or six years ago most interactive image galleries were created using Flash, today the majority are rendered using

"There's been an explosion in smartphone use over the past few years which has led increasingly to the web being accessed on the move"

standard HTML and either CSS, JavaScript or most commonly both. Flash is still useful for certain types of content – it's a very popular method for showing video on the web, and games are currently easier to code in Flash.

Flash isn't nearly as accessible as HTML, either from a developer point of view or from a user perspective, especially with regard to disability access. Adobe has worked to improve this over the years, but the technology isn't ever going to be as accessible as a plain-text-based system such as HTML.

HTML: function, not form

In the early days of the internet, HTML included a large variety of methods for changing the way content displayed. For example, an early option was the <font> tag which allowed you to specify the typeface that you'd like the content and information to be rendered in. This allowed designers to very precisely choose how their content was displayed, but when a website consisted of 50 pages and the designer wanted to change the font, it was a mammoth task to work through every page and replace every single reference to the choice of font.

As a solution, CSS was introduced. CSS, as you'll see elsewhere in this book, is a system for applying styles – defining the form and aesthetic of a page. This means that now we can treat HTML as a structural language only – we don't define the way a web page looks using HTML, just how it's structured and content arranged in a semantic hierarchy. The net result is that by separating form and function we can now change a font that's used throughout a big website with just a single change to the CSS stylesheet, saving time and ensuring design consistency throughout.

What's next for HTML?

HTML has come a long way since its invention in a computer lab in Switzerland, and it's continuing to develop. The HTML5 specification

is already partially implemented in browsers, but other elements of the language are still being discussed, and there's a good probability that we'll see further enhancements in the coming years such as the ability to specify different resolution images to be downloaded to desktop computers from those sent to mobile devices.

There's been an explosion in smartphone use over the past few years which has led increasingly to the web being accessed on the move. These devices have a different screen orientation to the traditional computer, often have less bandwidth and computing power, but expect to have a fantastic browsing experience. The HTML language, through HTML5, is already adapting to meet this challenge, but as the specification matures you can expect to see further enhancements with this usage scenario in mind. It's an exciting time to be a web designer!

<below> The new features in HTML5 allow you to embed video and audio directly in your pages, and when combined with CSS can provide amazing special effects

HTML glossary

We demystify the definition of the most popular HTML jargon

a

The a tag defines a hyperlink, which is used to link to another web page. The <a> tag is typically coupled with the href attribute which will include the link's destination. For example, link text

aside

This is a tag that was introduced in HTML5 and refers to content that is an aside to the main content. The content in the aside tag should be relevant to its surrounding content.

audio

The <audio> tag allows for the inclusion of audio files which can be read directly by the browser without the need for a plug-in. Between the audio tags will be the source od the audio and any related attributes

body

The body tag is where all the content seen on screen is stored. It sits after the head tag and between the html tag.

br

Br is short for break, or more specifically line breaks. It is an empty tag which means it isn't a set of tags just a single tag. It is used to split long headlines or long sentences.

canvas

The <canvas> tag is a container where the necessary code is placed to show the graphics created by the code. The canvas tag can be styled eg width, height, border, to make it fit better with the graphics that are placed upon your web page.

class

The class attribute can be applied to the div tag eg 'div class='. This will have a unique name that is not in the HTML specification. Unlike the id option the class identifier can be applied to more than one element.

div

The div tag is used to create blocks of content in a web page. These are known as block elements. The tag will be coupled with id eg 'div id=' to create a unique name which typically reflects what the content block is eg leftcol for left column.

footer

The footer is the part of a page that typically sits at the bottom of a page. This will include elements that need to be seen on every page and will typically have navigation and link to an about page and contact page

form

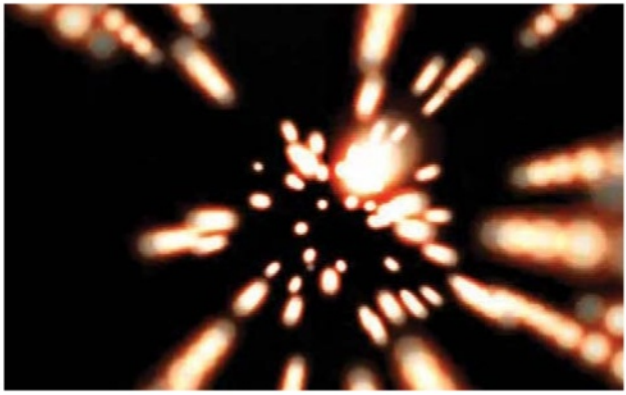
The <form> tag is a container that will contain all the necessary elements to create a form. This could be text fields, checkboxes, labels and submit buttons, your usual form garb.

head

This tag contains a host of information that is typically hidden from the user, but is important for the browser to read. It will include the page title, keywords, scripts and links to stylesheets. The head tag comes immediately after the html tag and before the body tag.

header

A web page is typically split into different parts including the header.



<above> The canvas tag is a container for dynamic graphics

The header is typically at the top of the page and includes elements that will typically be seen on every page eg the site title, logo and navigation. For a tutorial on how to create your header, go to page 60.

html

The html tag is the tag that defines a HTML document. It is the tag that all other tags live within. You will find a HTML tag at the beginning and end of a document

img

Img is short for image and defines where an image will go in a web page. The img tag is typically paired with src, which shows the source of the image and alt. Alt text is what will show in the browser if the image is not available.

li

The li tag is sort for list. These are found inside the ul (unordered) or ol

<above> The video tag is a placeholder for adding video into a web apge

<above> The elements of a HTML form are stored inside the <form> tags

(ordered) tag and a set of li tags are required for each item. If three items are needed in a list then three sets of li tags are needed. The text inside the each list item will be displayed on a web page. It is important to close each of the li tags.

nav

A tag introduced in HTML5, nav is a semantic tag that is short for navigation. These are used to store any navigation or menus that are typically found at the top of a web page, usually within the header.

p

This is the HTML tag for paragraph. This works exactly as a paragraph would in a book. The paragraph of text would be surrounded by an opening and closing set of tags eg .

script

Web pages often include code beyond HTML and CSS that add dynamic elements. A simple example is Google Analytics code and other code written in JavaScript. The code is stored inside a set of <script> tags to help identify the code.

section

The section tag was introduced in HTML5 and is a generic section of a web page that will contain general content such as text and images. It can have its own id (name) to help identify the section, and is able to have a CSS class added. It is often has a title as well.

table

Tables are made up of rows and columns and were used to create page layouts before CSS became popular. They are now used to show off tables of data.

```
<html lang="en">
<head>
<meta charset=utf-8>
<meta name="viewport"
<title>HTML5 Demo: Vid
<link rel="stylesheet"
<script src="js/h5util
<body>
<section id="wrapper">
<div id="carbonads-con
class="carbonad"><div
type="text/javascript"
document.createElement
```

<above> The script tag is where code, or link to the code source is placed

tag

The term tag refers to all the elements of the HTML specification. The HTML language is made up of tags that are contained in a set of brackets eg <body>.

title

The title tag is stored inside the head tag and determines the title of the page. This title will appear in the title bar or tab in a web browser. It is important that the title is concise and informative as it will be read by the user and search engines. Do not include notes within the tag as this will also show in your title.

ul

Lists are an integral part of web pages and there are several options for creating lists. ul is short for unordered list which means the list is bullet pointed by default. The alternative option is ol, ordered list, which adds numbers or letters eg a,b,c

video

The <video> tag is a set of tags, which will contain any related video information. For example, a basic version would contain width and height attributes along with the source of the video.

Create a basic layout

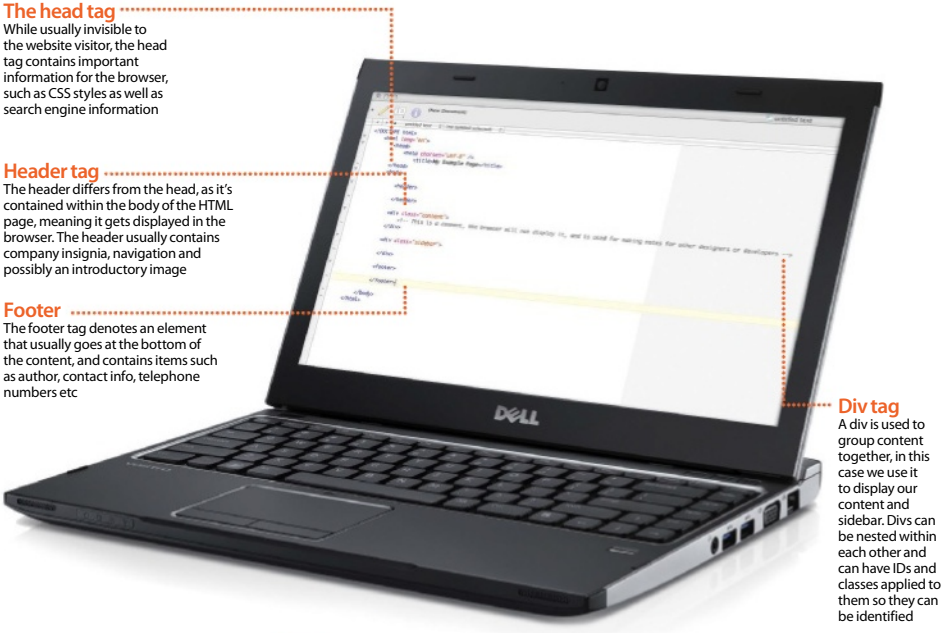
Code up your first basic skeleton for a webpage using HTML5 – it's really quite simple to do

In this tutorial we'll make an HTML5 layout for a simple webpage. While initially HTML may look confusing, there's a relatively small amount of code that you'll need to remember. There are a few basic building blocks such as which you will find yourself using regularly, then others such as <caption> which you will use much less often.

If you have made pages before in the past using HTML4 then you should feel at home, just be aware of the new tags. If you've not written any HTML before then good news, as it's now easier to understand! HTML5 has added in a lot of new elements which help the browser interpret the layout of the page easier. <header> and <footer> are two examples, and describe common top and bottom elements of a page respectively.

When starting out in web design it's important to remember to 'open' and then 'close' tags correctly. Every time you open a new command, get into the practise of having your end tag ready, so you don't end up with any loose bits messing with your code.

Open your text editor of choice and let's get started!



Set up your first HTML page

The HTML base

01 First up, the doctype tells the browser what kind of content to expect. In HTML5 it's simple:

```
<!DOCTYPE html>
<html lang="en">
</html>
```

Create the head

02 Next up is the <head> element, where we place all the files we wish to include within our page, eg CSS style sheets, which we'll learn about later.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<title>My Example Page<title>
</head>
</html>
```

Add the body

03 The <body> is where all the viewable page content goes. In it we'll add the header, the main content area, sidebar and a footer perhaps for a sitemap.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<title>My Example Page</title>
</head>
<body>
</body>
</body>
```

Add the header

04 The <header> is typically used to hold the main site image, with a title and possibly site navigation. You can have multiple <header> tags on a page.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<title>My Example Page</title>
</head>
<body>
<header>
</header>
</body>
</html>
```

Sidebar and content

05 The main content can be placed within a 'div', then a class applied to it, place it after the closing header tag, but before the closing body tag:

```
<div class='content'>
</div>
```

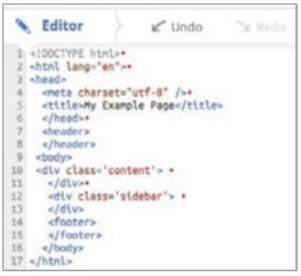
The sidebar can be placed after it, also with a class:

```
<div class='sidebar'>
</div>
```

Add the footer

06 The <footer> element usually contains quick links to common areas of the site such as 'Contact us', and sometimes has copyrights or addresses.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<title>My Example Page</title>
</head>
<body>
<header>
</header>
<div class='content'>
</div>
<div class='sidebar'>
</div>
<footer>
</footer>
</body>
</html>
```



"When starting out in web design it's important to remember to 'open' and then 'close' tags"

Code a link

Adding in links to other pages is a key part of web design

While links may seem like a simple element to add to a webpage, they are the key building blocks of the entire internet. Similar to pages within a book, separate webpages within a website help to break up content by topic or purpose. Then, by adding in a navigation bar, it means that visitors will be able to quickly between content.

Links are also one of the most important elements that Google and other search engines use to find and index your site. The kind of content that you link out to, and the kind of content contained within sites that link to your pages, are vital in informing the search engine on how to categorise your site for its search results.

This tutorial will take you through adding in links leading to other pages in your website, and how to link out to other people's sites. Once you have added some links, your site can start to function how it's intended to, with visitors being able to view all the content contained with ease.

The basic 'a' tag

01 Adding a hyperlink to your webpage is really simple, but there are a couple of customisations to take into consideration. A link is denoted using the 'a' tag, short for anchor. To start off your link, enter in:

```
001 <a> </a>
```

However, unlike some other HTML tags, an anchor needs a little more information for it to work properly.

Href

02 The link needs a destination so that when it's clicked, the browser knows where to go. This is added in using 'href'. For example:

```
001 <a href="http://www.webdesigner.com"></a>
```

would send the browser to that external website.

Internal links

03 To link to another page in your own site, you simply add the name of the page:

```
001 <a href="contact.html" ></a>
```

This assumes that the contact.html page is within the same folder as the page we are working on. See the 'Relative paths' boxout for more information.

Add link text

04 To make the link visible to the user, add in some text between your opening and closing <a> tags. For example:

```
001 <a href="gallery.html"> See My Gallery</a>
```

Open the page in your browser, and assuming the gallery.html page exits in the same directory, clicking the text will take you there.

Open a new page

05 If you want your link to open in a new tab or window, you can see a target to the link:

```
001 <a href="gallery.html" target="_blank"> See My Gallery</a>
```

"Links are the key building blocks of the entire internet"

Create lists in your website

Learn how to quickly create various lists within your webpages using HTML

Lists are an effective way of presenting important information in a way that's quick to read and digest. They are also a good method for giving a brief description or introduction into content to follow, to help your users find what it is they are looking for.

Coding up a list in HTML is quick and very straightforward, and once you've got the hang of the basics you can start to apply your own styles using CSS to make them really stand out from the crowd, which we'll come to later

There are two main types of lists in HTML: ordered and unordered. An ordered list stars at 1 and then increases. This is useful when you need to give preference to the higher items, or you want to display a step-by-step guide.

An unordered list is simply bullet-pointed and is useful when you just need to display a list where the order is irrelevant. By default, lists will be indented from the surrounding content and have a circular bullet point. This tutorial will show you how to add lists to your pages, both ordered and unordered, and also how to create lists within lists.

Unordered list

01 An unordered list by default is shown slightly indented with a bullet point denoting each item. To add in an unordered list you use the <ul> tag.

Add list items

02 Within your list, items are added using the list item tag like so:

```
<ul>
<li> </li>
```

Text can go between the open and close li tags; for example:

```
</ul>
<li> Tea </li>
<li>Coffee</li>
</ul>
```

Nested lists

03 Lists can be embedded within other lists just by opening and closing a new <ul> within another <ul>, and then list items can be added to that too. They will then be indented again within that <ul>.

```
<ul>
<li> Tea </li>
  <ul>
  <li>Breakfast</li>
<li>Earl Grey</li>
<li>Coffee</li>
<ul>
<li>Latte</li>
</ul>
</ul>
```

Ordered lists

04 If you wish to number your list items then you can use the <ol> tag in place of the <ul>. This will then replace the bullet points with ascending numeric values for you list of items.

Definition lists

05 There is one other type of list in HTML – the definition list. It's quite rarely used but allows an indented description to be added to each list item. An example definition list would be:

```
<dl>
<dt>Snowboard</dt>
  <dd>- Great for freestyle tricks
</dd>
  <dt>Cross Country Skis</dt>
  <dd>- Ideal for exploring on
</dd>
</dl>
```

All about div tags

Use a div tag to create consistency on your website

An HTML page is made up of a series of tags which tell the web browser what to display and where. The main building block of a page is a div tag, short for division. If you look at a newspaper page, you will see that text is grouped together in columns. Images with captions are also together with a margin around them. A div in HTML is similar to this, and it's usual to group together content in a similar way. While HTML5 has added a few new tags with more semantic names such as 'header' and 'footer', the main body of most pages is still constructed using divs.

There is no limit on how many divs you have within a page and most webpages you visit online will contain many, all nested within each other. Divs can contain text, images, video and audio, as well as other HTML elements such as articles and sections.

This tutorial will show you how to make your first divs and then how to apply an ID or class to them so when you're ready you can apply CSS styles, or use JavaScript on them. We'll be using plenty of them throughout the book, and they are really simple. Don't forget to close each and every div tag that you use else you will find your website takes a wonky look to it.

Properties of the div

The basic div

01 A div is started by using this simple piece of code: . Following on from that, you can then insert all the content you want to contain within the div.

```
001 <div>
002   This is where you will add your
div content.
```

Closing the div

02 Once you have inserted all the content, you need to make sure you close off the div by using . Note the forward slash denoting the ending of the current div:

```
001 <div>
002   This is where you will add your
div content.
003 </div>
```

Adding an ID

03 A div can have a unique identifier so it can be recognised in a style sheet or by any JavaScript you may add. To apply an ID to a div, use:

```
001 <div id"mydivID">
002   This is where you will add your
div content.
003 </div>
```

"Divs can contain text, images, video and audio, as well as other HTML elements"

Adding a class

04 Classes are similar to IDs, but many items on a page can have the same class. To apply a class to a div, use:

```
001 <div id"mydivID">
002   This is where you will add your
div content.
003 </div>
```

Using classes makes styling multiple elements much easier. It allows you to create CSS coding in one place that will then apply to all of the specified classes rather than having to input the styling information into each individual div tag.

Nesting divs

05 Many webpages are made up of a few main elements such as a header and footer, and then a main content div. Within a div it's possible to have other divs. For example:

```
001 <div id="content">
002       <div class="leftColumn"
003       </div>
004       <div class="rightColumn"
005       </div>
006 </div>
```

Doing this allows you to have a universal style to the 'content' div, and then apply different styles to the divs contained within.

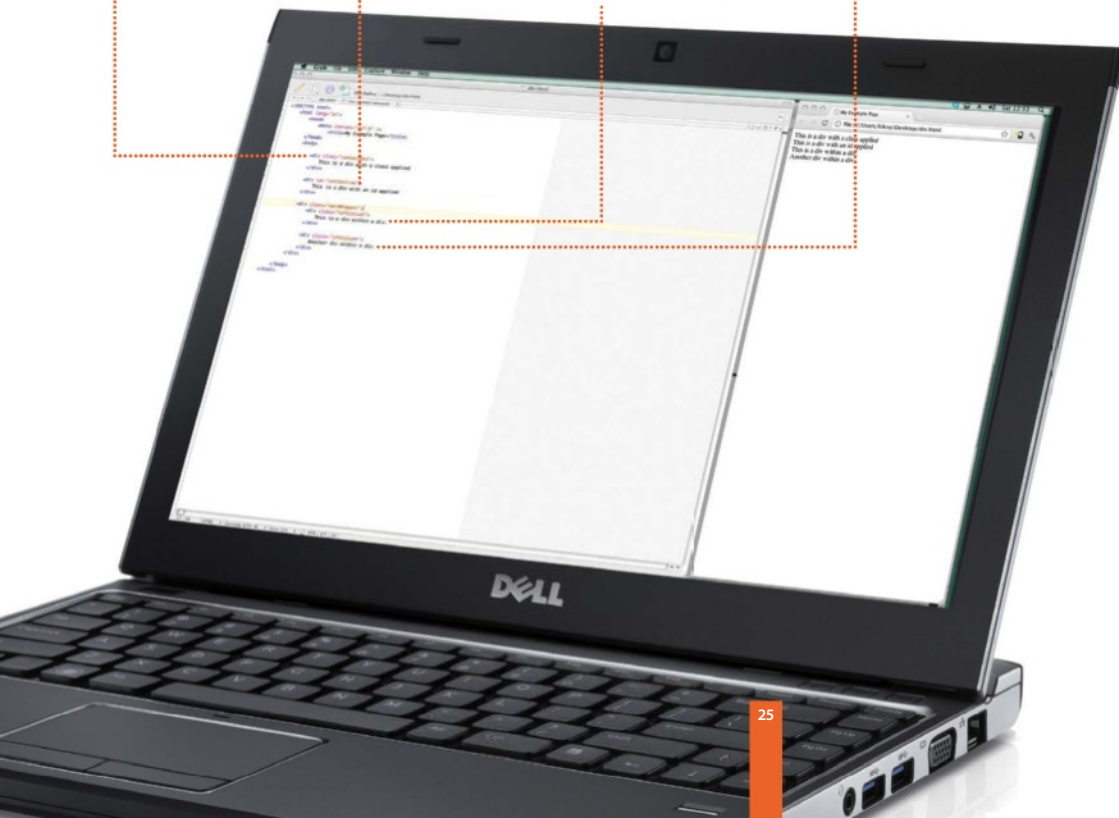
"There is no limit on how many divs you have within a page and most webpages you visit online will contain many"

Div classes
Classes are widely used in web development to allow CSS styles to be associated with elements within your HTML page

IDs
IDs are used in a similar way to classes, but there should only be one element on a page with a particular ID. IDs allow you to target specific elements using JavaScript for manipulation

Indenting divs
Once your pages get more complicated they can feature many divs all nested within each other. This can make it hard to track where one opens and another closes. It's standard practice to indent the content of a div using the tab button

Code highlighting
Most HTML editors offer code colour highlighting of some sort. This can help you when checking which divs have classes applied etc

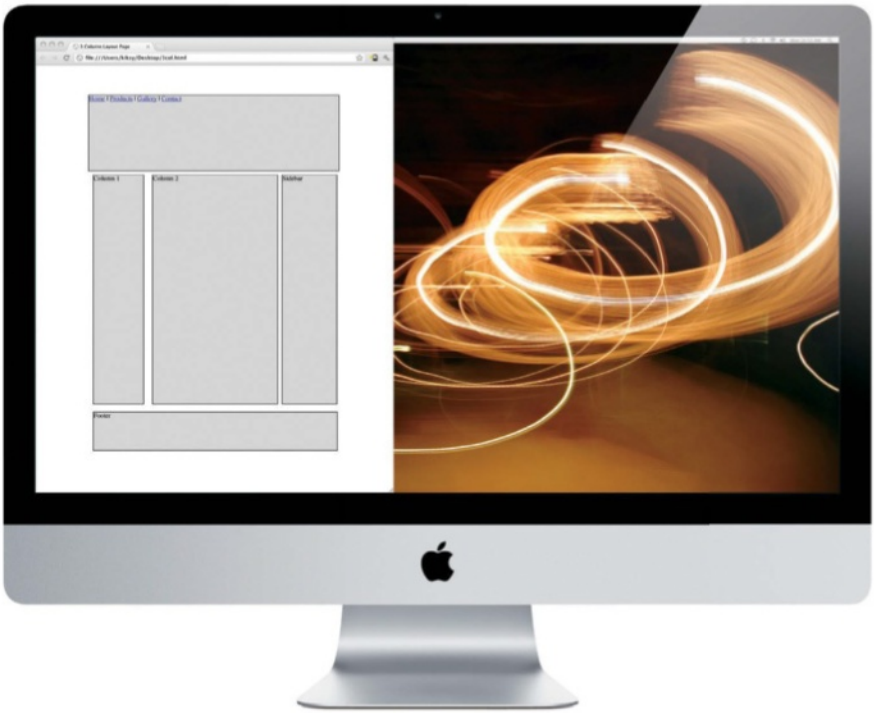


Create a three-column layout

Make a three-column webpage structure using HTML5

HTML and CSS essentially allow for a huge range of freedom when designing your pages, but there are a few layout principles that have developed over the years which it's a good idea to follow.

If you have a quick look at some of your favourite sites on the internet it's quite likely that they will use a column layout, akin to a newspaper. A common layout for a site is to have the site or company logo at the top of the page, then have a column down the left for navigation or links, a main content section in the centre, and then a sidebar on the right with supplementary information or Facebook and Twitter feeds. This is then commonly rounded off with a site-wide footer containing copyright information, the name of the site designer, site links and sometimes a contact address. This tutorial will show you how to code up the HTML scaffolding for a three-column layout using modern HTML5 elements. Once you have mastered this, it can then be adapted into a two- or more-than-three-column layout if you so wish.



Set up your HTML page

01 To start off, open up your favourite editor and create the initial HTML elements as usual. The first thing we need to add is the HTML5 doctype declaration:

```
001 <!DOCTYPE html>
```

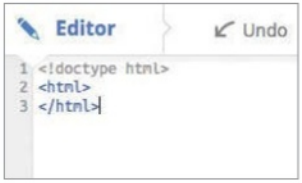
This tells our browser that we are using the HTML specification within the page, and it should be placed right at the top of your HTML file.

Open HTML tag

02 Next up we need to start off our HTML by entering the HTML tag – <html> – and then making sure we close this with </html>.

This tag tells the browser that contained within this section is HTML code. All the page content that we will be creating will need to go within these tags.

```
001 <!DOCTYPE html>
002 <html>
003 </html>
```



Insert the head

03 The head tag is an important one in HTML, and shouldn't be confused with the 'header' tag. Content within the head is not displayed to the user, but is used to include your CSS stylesheets, JavaScript and metadata. Within your <html> tags, place <head>, then </head> to close it off.

```
001 <!DOCTYPE html>
002 <html>
003 <head>
004 <!-- CSS Styles and javascript  to
be used
005 </head>
006 </html>
```

Add a title

04 The title tag is used to display text in the browser's title bar at the top (if it has one), and also used as a guide for search engines to identify what content is contained within the page. Add a <title> tag within your head. Next, add a page title to describe the content, and close the title tag with </title>.

```
001 <head>
002 <title>
003 3 Column Layout
004 </title>
```

Define character set

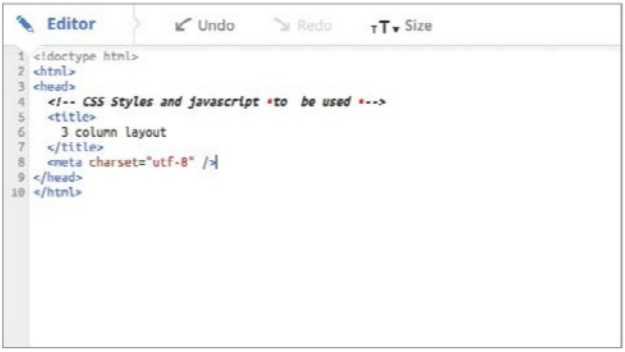
05 There are many different ways of displaying characters in computing, and a few different standards depending on language; eg Greek or Arabic text looks very different from Chinese. Luckily for us, almost all the web uses the UTF-8 standard, which we define within our head with:

```
001 <meta charset="utf-8" />
```

UTF-8 does a good job of displaying most characters correctly on your website.

The body tag

06 All the content you wish to be displayed to the user is put within a 'body' tag. This goes after your 'head' tag has been closed, but within your <html> tag. Start it off with <body>, then close it off in the normal way: </body>. We will now place all code within the body tag.



> "There are many different ways of displaying characters, and a few different standards depending on language"

Create a three-column layout

Adding comments

07 Sometimes you may wish to add a note to a location within your page, perhaps as a reminder or to make it easier to see where certain elements start or finish. Comments are started using '<!--', followed by the comment text, and then ended with '-->'. For example:

```
<!-- Start header -->
```

The container

08 It's quite common to wrap all the main content within the body in a 'container' div. This can make styling and centring easier when you start to write up your CSS. Add a container div within the body with and , then give it an ID so we can style it later:

```
<body>
<div id="container">
</div>
```

```
    <meta name="descripti
    <meta name="robots" c
</head>
<body>
    <div id="container">
    <header>
    <nav>
```

The header

09 The top element to our page is usually referred to as a 'header' and contains the site title, logo, navigation, and sometimes adverts. To create a header we use the header tag, which is new in HTML5. To open it enter <header> and then to close, </header>. A page can have multiple headers, although they cannot be contained within each other.

```
Editor   Undo   Redo   Size
<!doctype html>
<html>
<head>
  <!-- CSS Styles and javascript to  be used -->
  <title>
    3 column layout
  </title>
    <meta charset="utf-8" />
    <meta name="description" content="Fishing in Dorset">
    <meta name="robots" content="noindex">
</head>
<body>
    <div id="container">
    <header>
    <nav>
        <a href="home.html">Home</a> | <a  hef="products.html">Products</a>
href="gallery.tml">Gallery</a>
    </nav>
    </header>
```

Nav tag

10 Nearly all pages contain a navigation or menu bar of some sort to allow visitors to get around the site. HTML5 now has a tag specifically for this. The 'nav' tag is where to place links to other items. Let's add one to our page – <nav> – and then close off again with </nav>.

```
<header>
<nav>
</nav>
</header>
```

Navigation items

11 To add links to the navigation we use the <a> or anchor tag. Eg Products . If you don't yet know your exact site layout, it's common to use a '#' in place of the link. A hash symbol can also be used to target a specific div within a page.

```
<header>
<nav>
<a href="home.html">Home</a> |
<a hef="products.html">Products</a> |
<a
 href="gallery.tml">Gallery</a>
</nav>
</header>
```

```
/head>
body>
    <div id="container">
    <header>
    <nav>
        <a href="home.html">Home</
ref="gallery.tml">Gallery</a>
    </nav>
    </header>
    <div id="col1">
        Column 1
    </div>
```

First column

12 Now we can add our first left column. This goes after the header tag, but still within our container div. Add this in with and then don't forget to close: . Again, you can give it an ID or class, such as:

```
<div id="col1">
</div>
```

This column might contain a menu or adverts, for example.

"The top element to our page is usually referred to as a 'header' and contains the site title, logo, navigation, and adverts"

Second column

13 The second column is usually the widest in the centre, and normally houses the main page content. Add it to the page the same way as before, but give it a different ID this time.

```
<div id="col2">
</div>
```

Don't be concerned if you can't see any content within your browser – as without styling, divs are essentially invisible and only the content contained is displayed.

```
</nav>
</header>
<div id="col1">
    Column 1
</div>
<div id="col2">
    Column 2
</div>
<div id="sidebar">
    Sidebar
</div>
```

Third column

14 The third and final column is then added in exactly the same way, with a unique ID:

```
<div id="sidebar"></div>
Column 1
</div>
<div id="col2">
Column 2
</div>
<div id="sidebar">
Sidebar
</div>
```

This column might contain a menu or adverts, for example.

The footer element

15 The footer is used to mark the bottom of the page, and often contains a list of common links, along with copyright and/or contact information. To add a footer we simply use the HTML5 footer tag – <footer> – and as always, close it off using </footer>. Place this after your sidebar column in the code.

```
<footer>
Footer
</footer>
```

Metadata (optional)

16 If you know what content is going into the pages and wish to improve search engine optimisation, you can add meta information to the head. Meta information helps Google and other search engines to index and organise sites by content. The meta description tag is used to give a short account of the page content:

```
<meta name="description"
content="Fishing in
Dorset">
```

Stopping Google indexing (optional)

17 To stop Google and other search engines indexing your page, you can add the following code to your head:

```
<meta name="robots"
content="noindex">
```

This means the page will not be shown in Google's search results. This can be useful on client login pages, or out-of-date pages that you wish to archive.

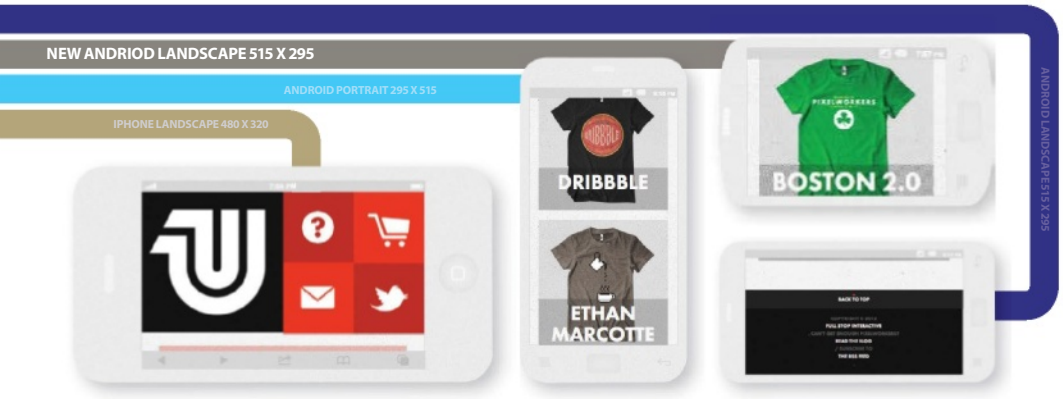
> "The second column is usually the widest in the centre, and normally houses the main page content"

The end result

18 Save the file now, making sure to have the .html extension at the end. Open the file within a browser to see the result. As there is no styling applied yet, it won't look particularly attractive by any means, but you've created a basic web page layout that's now ready for CSS and content to be added.

```
<!DOCTYPE html>
<html>
<head>
<title>
3 Column Layout
</title>
<meta charset="utf-8" />
<meta name="description"
content="Fishing in Dorset">
<meta name="robots"
content="noindex">
</head>
<body>
<div id="container">
<header>
<nav>
<a href="home.html">Home</a> | <a
hef="products.html">Products</a> | <a
href="gallery.tml">Gallery</a>
</nav>
</header>
<div id="col1">
Column 1
</div>
<div id="col2">
Column 2
</div>
<div id="sidebar">
Sidebar
</div>
<footer>
Footer
</footer>
</div>
</body>
</html>
```


Style. CSS
ANDROID LANDSCAPE 320 X 240
IPAD LANDSCAPE 1024 X 768
KINDLE LANDSCAPE 1024 X 600
ANDROID PORTRAIT 240 X 320
BOSTON
STRIKE OUT
7:57 PM
DESIGN IS A JOB
DESIGN IS A JOB
DRIBBBLE



An introduction to CSS

If HTML provides the structure for a website, CSS provides the form. This is the language that will make your site stand out

Cascading Style Sheets, commonly abbreviated to CSS, is a language used for describing how HTML should be presented. CSS documents provide all the format, colour, positioning and other characteristics of a design that you will see on virtually any webpage.

Originally introduced alongside HTML4, the language was designed to separate the structure of a document from the presentation, improving the control of a design for web designers. CSS also helped to improve accessibility of web pages, reduce HTML code complexity and crucially the amount of repetition of markup that had to be created to describe how a page should look.

As well as these benefits, the separation of form and function allows different designs to be displayed to users according to their device, simply by providing a different stylesheet. This means that, for example, a mobile phone can load the same web page as a desktop browser, but present the content within the page in a different manner. Similarly, CSS can be used to present page content one way on screen and a second when printed, or to provide direction to screen reader software over areas for emphasis.

Additionally, CSS is user-configurable, so while the designer might specify that text should be shown in 18pt red Arial, the visitor to their site can load their own stylesheet to show the text in 12pt blue Times instead, should they wish to. As you can see, CSS offers a great deal of flexibility by design. To fully understand what it's capable of, we need to look a bit more at what CSS is, why

"CSS offers absolute positioning of an element according to set co-ordinates"

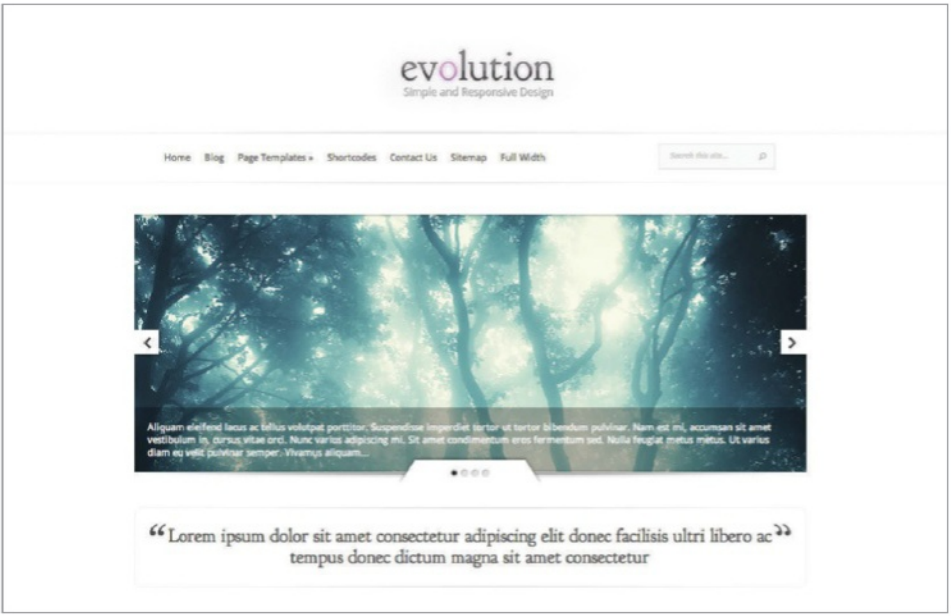
and how it came about, and examine how the implementation of the language works across different web browsers.

The cascading bit

CSS is designed to use a priority scheme that determines how styles should be applied to HTML elements. The cascading bit of the language name refers to how this scheme works. Broadly, styles are applied in chronological order, but also according to specificity. If a style property is declared more than once for a particular element, the last chronological declaration will be used, unless the earlier declaration was more specific about the elements it should apply to than the latter. This sounds really confusing, but in practice it allows you to apply a set of broad styles that provide basic properties for the font that text should be rendered in, for example, and use more specific selectors to subsequently set the size, colour or font weight. To help make sense of it all, let's take a look at what CSS looks like in code.

CSS syntax

Just like HTML, CSS is designed to be human-readable in its raw form. English keywords are used with a simple syntax to



Recent versions of web browsers have vastly improved their implementation of CSS standards

CSS3 opens the web up to more users by allowing individual stylesheets on pages

The advantages of CSS

CSS offers many benefits over the previous idea of pairing visual characteristics into HTML. The most obvious one is that by separating content from presentation, the aesthetic of a web page or entire website can be tailored to individual user profiles or devices with ease. A single HTML page can be rendered in many different ways, as amply demonstrated by the CSS Zen Garden experiment (**csszengarden.com**), and this characteristic can be used to ensure that mobile devices see a layout appropriate to their form while a desktop computer sees a different rendering more aligned to the larger screen estate. CSS supports many different usage scenarios including print and audio.

As well as the ability to tailor the presentation according to device, user characteristic or location, CSS also helps to ensure consistency across an entire website. Where multiple pages reference the same CSS sheet, a change in the CSS is instantly reflected across all those pages, saving the designer time and reducing the risk of inconsistencies. Along the same vein, CSS allows multiple elements to be selected and styled with a single selector. This reduces the amount of code required to achieve a design.

CSS makes the web more accessible; a user can choose to load their own stylesheet onto any document. Useful for visually-impaired users this characteristic also allows designers to separate styles into multiple stylesheets with each applying an additional layer of styling for easy organisation and adaptation of a design.

describe a list of rules, style properties and style values. Each rule in a stylesheet is defined by means of a selector, which identifies the HTML elements the rule should apply to. Within the rule, a series of properties and values are defined so that the visual characteristics for an element are built up property by property. Let's take a quick look at an example:

```
h1
      font-family:  arial, helvetica,
sans-serif;
     font-weight:   bold;-
font-size: 18pt;
     color:      red;
     margin:  10px;
}
```

In the code above we've defined the rule using the h1 { } selector. This specifies that the style properties within should apply to all <h1> tags found on the page. Within the curly brackets we define a series of properties, such as font-family, font-size etc, and the values we'd like to set for each. As a result, every heading 1 on our page would be rendered in Arial (or Helvetica if Arial wasn't available), bold 18pt red text. The entire <h1> would have a ten-pixel margin around it.

More complicated selectors

In the previous example we used a simple selector to establish which elements the rule should apply to. Selectors can be nested so that only specific elements have the rule applied. An example might look like the code that follows:

```
#content article h1 {
     color: blue;
     }
```

Here, the rule would only apply to HTML <h1> elements that are nested inside an <article> element, itself within any other element that has an ID attribute of "content". Our page would now show all <h1> tags as red, bold, 18pt Arial, with the exception of those headings inside an <article>, itself inside an element with an ID of #content where the colour would be blue instead. This works because the CSS sheet cascades the styles and also has the priority scheme we discussed earlier. As our second rule is more specific in its selector, it takes priority over the more generic rule we created first. We could have also positioned our second rule chronologically after our first to ensure it would take priority, but sometimes the specificity rules that are a part of CSS would prioritise over chronology. By combining specific selectors with

"Selectors can be nested so that only specific elements have the rule applied"

different rules, and using specificity and priority it's possible to style a document with a few SS rules.

Different ways to load a style

There are three different ways a style can be applied to an HTML element. The first is through an external file that is loaded into the HTML document in the <head> section. The second is to include a <style> declaration in the <head> section of the document, writing your rules and properties directly inside the <style> area, and the third is directly onto an element using the style attribute. The latter is referred to as an 'inline' style.

When calculating the style properties of an element, CSS is loaded first from any external files and applied, then from the <head> section of the page, and finally from any inline styles. Where the same property is set by more than one of these three methods, they apply in the order as described above.

Positioning elements

One of the most common areas of CSS that new designers struggle to grasp is how elements can be positioned on the page. CSS offers a number of different positioning systems, and these can be mixed and matched within a single page design. The first, and default, positioning scheme is 'relative' positioning. This operates on the basis that an HTML document is chronological. Much like a word processor, content that appears beneath other content has its position on the page dictated by the content above which pushes it down into position.

"CSS offers absolute positioning of an element according to set co-ordinates"

In addition to relative positioning, CSS also offers absolute positioning where an element is positioned according to a set of co-ordinates relative to the container element. This allows elements to be very precisely positioned on the page, relative to the whole page.

Similar to absolute positioning is fixed positioning. This works, as with absolute, using a series of co-ordinates but rather than being positioned relative to the page, 'position: fixed' elements are always relative to the viewport (the visible portion of the browser screen). As you scroll, position fixed elements remain in the same place in the browser window.

The final, and often the most confusing, system is the

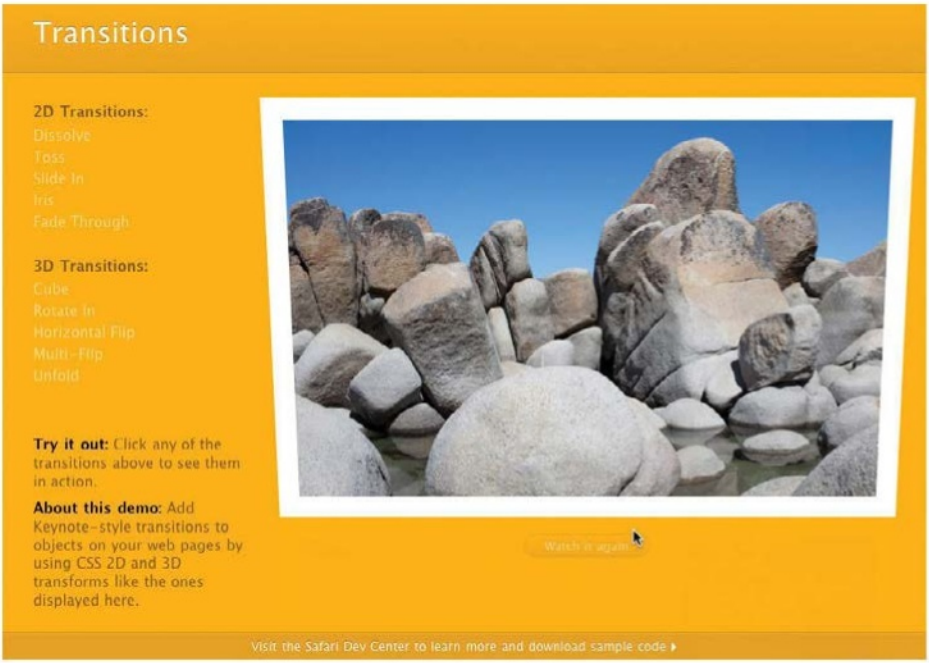
floating element. This system allows elements to push to one side or the other of the containing element, with content and other elements automatically flowing around it. This takes some understanding and experimentation, but is fundamental to achieving some common layouts so it's worth spending a little extra time getting to know how positioning, and especially float, works.

History and potential hiccups

By now you're starting to see the value and potential of CSS, but it's not all as perfect as it might at first seem. There are a number of problems that web designers have to overcome when using CSS to style their web pages, and to fully understand those issues it's worth taking a quick trip down memory lane.

When HTML3.2 was released, a series of new tags were added that allowed web designers to specify different visual characteristics for their content. These included the likes of <font>, <b> (for bold text), <color>, and many more. These tags provided the control over presentation that web designers were craving, but a problem quickly emerged: by mixing style with content the pages designers worked on

"CSS offers absolute positioning of an element according to set co-ordinates"



quickly became unmanageable, especially when a web site had lots of pages. Something as simple as changing the font a website used might have taken days to change, working through each HTML page in the website and changing references from one font to another.

As a consequence, the W3C quickly realised that content should be separate from presentation, and developed CSS. CSS1 debuted alongside HTML4 in 1996, but it wasn't until some years later that web browser vendors had managed to fully implement the language. In the meantime, a newer version of CSS had been created in 1998, and Netscape and Internet Explorer version 6 had both partially implemented CSS2.

As well as the slow implementation, from the start there were issues with the way the CSS specification had been written, leaving room for individual interpretation over what each part of the specification meant. This resulted in Internet Explorer rendering styles in one way, while the other popular browser of the day, Netscape, rendered differently. As well as

differences in opinion, both these popular browsers had their own bugs and quirks that resulted in strange behaviours when interpreting and rendering CSS. To further compound the issue, as the web was taking off and becoming mainstream, once the damage had been done there was no easy way to backtrack. The net consequence of these unfortunate events was that web designers had to start using workarounds to code different style rules for different browsers, taking into account the bugs, quirks and differences in rendering.

The arrival of HTML5

As the web has matured, so have the technologies used to create it and CSS is no different. HTML is now just about at version 5, and CSS3 first came into existence at the beginning of the century. It's only in the last couple of years that either technology has started to be properly implemented by browser vendors however, and the combination of HTML version 5, CSS version 3 and JavaScript is often referred to collectively as HTML5 (erroneously it must be said!).

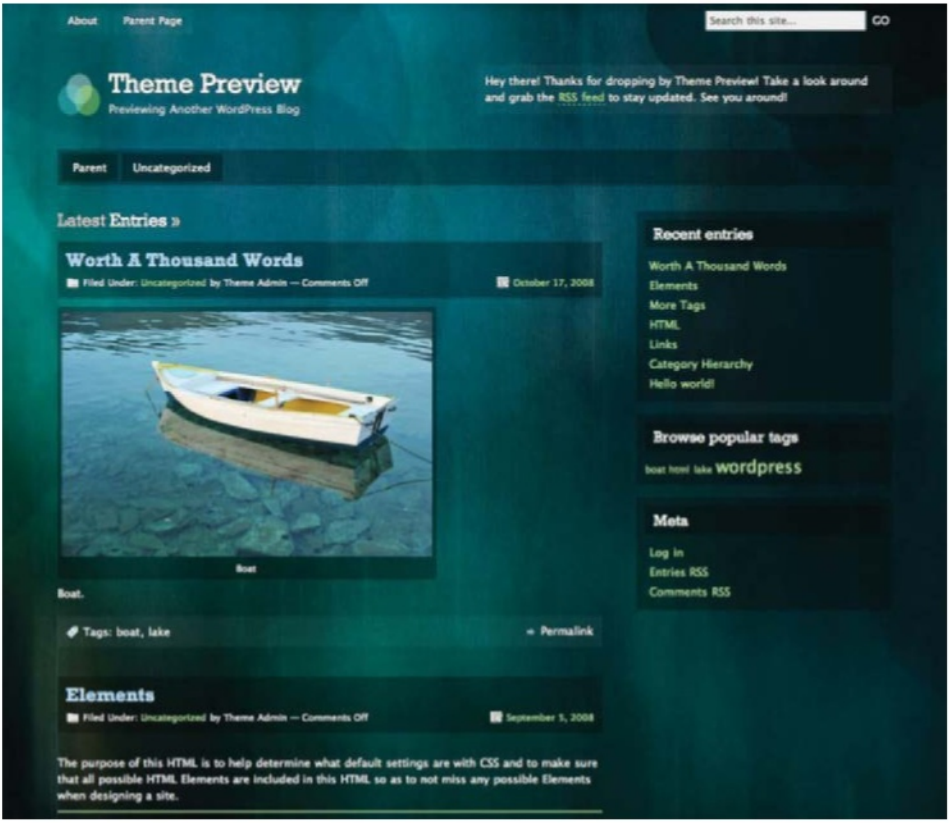
As mobile devices have become increasingly popular,

"It's only in the last few years that technology has been implemented properly by browser vendors"

driving the desire for access to the web on the move, and across different screen sizes, so the browser vendors have been coaxed into accelerating their implementations. This has provided quite a relief to web designers who only a year ago were still implementing workarounds for Internet Explorer version 6 – a decade-old piece of software!

Differences of opinion

Despite the recent strides towards adoption of CSS3, and the retirement of Internet Explorer 6 by Microsoft, there continues to be quite a lot of differences between the way different browsers render CSS. The most common issue today is where browsers implement CSS properties that haven't yet been fully ratified by the W3C. To ensure compliance, vendors prefix these properties with a name that represents the browser's maker. So, for example, to create a CSS drop shadow we currently have to have a rule that specifies properties for box-shadow, -webkit-box-shadow, -moz-drop-shadow, ms-drop-shadow and o-drop-shadow. The first property is the actual CSS3



<above> CSS allows websites to be tailored according the device upon which the website is accessed

specification property, while the others are the early implementations by Safari and Chrome, Firefox, Internet Explorer, and Opera respectively.

Vendor prefixes are the subject of much debate currently as they've become more prevalent in recent years as vendors rush to outdo each other in their CSS implementations. For web designers the new opportunities CSS3 presents are slightly offset by the continuing need to provide browser-specific properties for most of the modern properties.

What does the future hold?

As you've read, CSS3 is still being implemented and defined by both the W3C and browser vendors. And as for the future? Don't expect to see it any time soon, but CSS4 has been in development since 2009. It's still in early draft of course, but promises all manner of exciting new styling options and properties. If we're all still designing websites in another ten years, no doubt we'll be discussing matching selectors and reference combinators!

Styling with CSS

Understand the essential properties and values needed to style pages

background

Background is a property that will determine how the background of an element is styled. It has many associated properties eg background-color, background-size.

border

The border property, as its name suggest adds a border to the associated element. Borders can be added to all sides of an element or a specific element such as the top, right, left or bottom. There are several styles eg solid, dotted etc and the option to chose a width.

box-shadow

The box-shadow property allows for the implementation of multiple drop shadows on box elements, ie div elements. It can specify values for color, size, blur and offset. Text-shadow uses the same principles but applies the drop shadows to text.

font-family

Fonts can be styled individually or as part of a family. The font-family property is used should the first choice font not be available then a replacement option can be called upon. These will be set out in the value. Eg verdana has a font family that includes a bold and light option.

float

The float property is used to position an element specifically within another element. Float has two main values left and right. Selecting left will position an element to the left. Conversely, if assigned the right value, it will be positioned to the right.

The background property allows for background images which can enhance any design

height

This is as simple as the name suggests. The height of an element is determined by the height property. This can be a fixed height using pixels or a flexible height using percentages.

letter-spacing

To improve legibility and add flexibility to web typography the letter-spacing property can be used. The value eg 2px will add the stated space between each letter of the styled text.

line-height

Line height is the space between the lines in text. The line-height property can use a fixed value with pixels or alternatively use a number value eg 2. The default option is normal and is often the preferred choice.

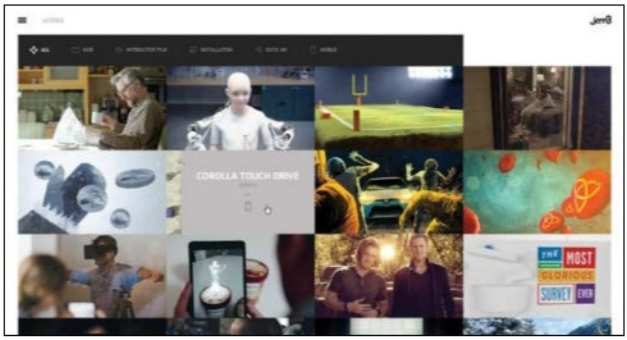
margin

Margins help create space between content. When adding a 5px margin to an element it will create a 5px space all around the element. Margins can be separated so it is only applied to one part of an element eg margin-top.

max-height

The max-height property ensures that an element does not exceed a specific height. It is often used in conjunction with max-width, which ensures that an element doesn't exceed a specific width. Both properties are typically

"Margins help create space between content"



<above> The width and height property determine how every element on a page is sized

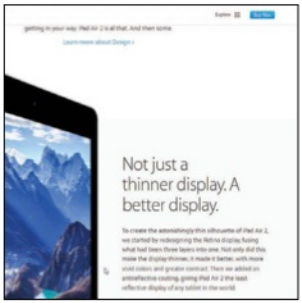
used in responsive designs to ensure that a design display as intended.

overflow

If an element has content that extends beyond its containing element, for example, an image, the overflow property can be called into action. If the hidden value is used the image outside of the element will be hidden. Using scroll will add scroll bars so all the image can be seen.

padding

Padding is very similar to margin, except the padding is applied to the inside of an element. If text is to close to the top of an element adding padding will give it breathing space.



<above> The spacing between lines of text can be adjusted with line-height making the text more legible

position

There are four values for the position property, the default is static. The others are absolute, relative and fixed.

property

There are hundreds of properties in CSS, these are accompanied by a selector and value. They tell the browser how a certain part of the related HTML element will be styled.

selector

A CSS selector is the identifier that is link to HTML. The selector will refer to a HTML element eg h1, h2. The browser will then read the style information (property and value) in the selector and apply it to the HTML element.

syntax

To understand CSS you need to understand the syntax that is used. CSS is made up of three parts the selector, property and value. Typically written as follows:

```
001 selector {property: value}
```

text-align

Text can be aligned to the left, right or centre of the element the text is contained within. A fourth value, justify, will spread text across the full width of its containing element.

text-decoration

The text-decoration property specifies what can be added to text. You can add under- and overlines and a strike through to your text.

value

The value element is associated with a property and selector. It will specifically determine how a property is styled. For example, if background-colour is the property the value will be the appropriate code colour eg #CCCCCC.

width

The width property is typically used in combination with the height property. A fixed width can be determined by using pixels. To makes sure an element is full width of an element or page its can be set to 100%.

z-index

The z-index property determines the stack order of an element. An element with greater stack order, for example 100, is always on top of an element with a lower stack order. Note that the z-index property only works with positioned elements such as position:fixed;.

<above> Adding shadows to text is simple using the text-shadow property

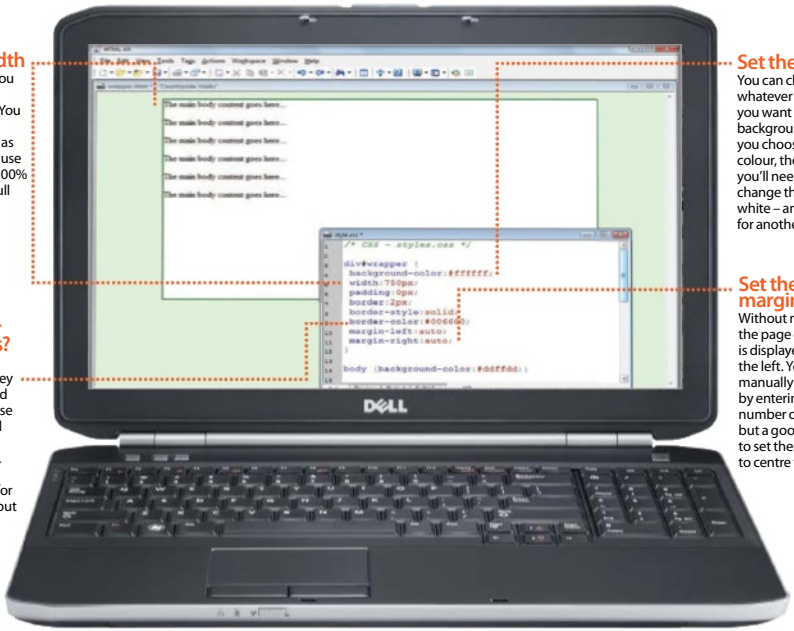
Centre your page

Wrap your page's content in div tags to give control over the way it's displayed

When you're designing a webpage, you have to remember that a browser window can be almost any shape and size. Desktop computers have a wide range of screen sizes, and mobile phone and tablets are used to browse the web too. Your nicely designed page could be ruined if you simply allow it to flow into the browser window regardless of the latter's size or shape. By creating a wrapper you can specify the width of the webpage content and its position in the browser window. It is an essential part of the design process.

'Wrapper' is a commonly used name for a piece of CSS code that is used with a div tag to size and position the content of a webpage. A div tag is placed at the start of the page content just after the opening body tag and again just before the closing body tag, thereby enclosing or wrapping the content.

In CSS we can define how everything between the div tags is displayed. It can be centred on the page, positioned a certain number of pixels from the left, a border can be added, its width can be fixed or variable and so on. It is an essential component of any webpage and it's quite straightforward.



The CSS link

01 CSS definitions can be stored in a webpage or a separate file. If it is in a separate file then any webpage can access it and it saves having to add it to each new page. Add a link in the HTML <head> tag to a file called style.css.

```
001 <link rel="stylesheet" type="text/
css" href="style.css" />
```

Add div tags

02 We want to specify the size and position of the webpage content, so we must place div tags at the start and end of the page, thereby enclosing all the content. The CSS definition will be called 'wrapper' and we refer to it with id="wrapper":

```
001 <div id="wrapper">
```

Create the CSS

03 We don't yet have any CSS, so create a new file called style.css and add a div ID called #wrapper. Set the width to 750 pixels. This value determines the width of the webpage content because everything is contained within the two div tags.

```
001 div#wrapper {
002         width: 750px;
003 }
```

Flexible page width

04 Set the page width using absolute dimensions and it may not fit on small screens like mobile phones or tablets, or it may look tiny on computers with large monitors. An alternative is to set it to a percentage of the browser width instead.

```
001 div#wrapper {
002         width: 80%;
003 }
```

Set the margins

05 We have specified the width for the page content, but not its position. With margin-left you can specify the number of pixels from the left the content is to be displayed. An alternative is to set both margins to auto to centre it in the browser.

```
001 div#wrapper {
002         width: 80%;
003         margin-left: auto;
004         margin-right: auto;
005 }
```

Add the borders

06 Your page design may look good if there is a border, and this specifies one that is two pixels wide. The border colour is set to dark green (006600) and the border style is set to solid. Alternatives include dotted, dashed, double and ridge.

```
001 div#wrapper {
002         width: 80%;
003         margin-left: auto;
004         margin-right: auto;
005         border: 2px;
006         border-style: solid;
007         border-color: #006600;
008 }
```

Browser compatibility

07 There are several web browsers, but unfortunately they all display webpages in slightly different ways. Compatibility between browsers is helped by adding this as the first line of the HTML file:

```
001 <!DOCTYPE html PUBLIC "-//W3C//DTD
XHTML 1.0 Transitional//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
```

Find a copy-and-pastable version at **http://bit.ly/lxdBQ**.

"You can specify the width of the webpage content and its position"

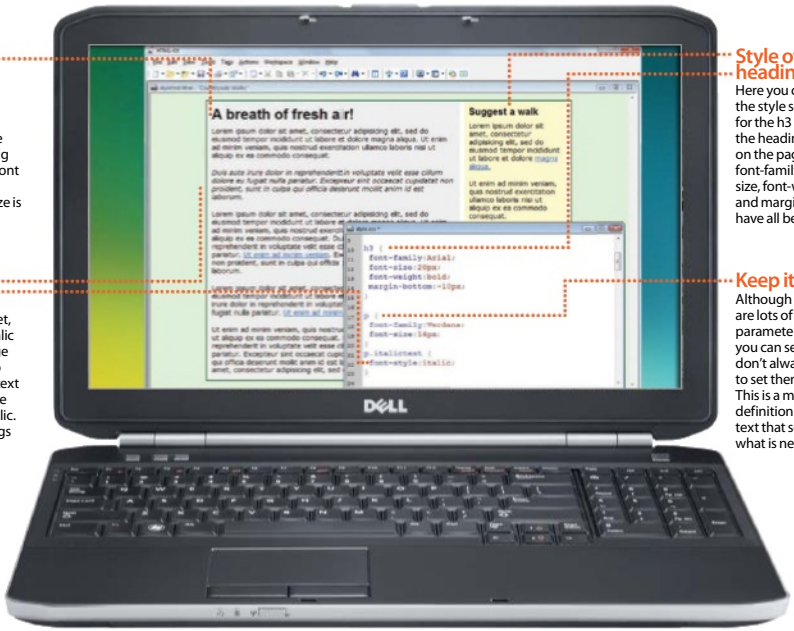
Define body and heading styles

Create CSS rules for displaying the text on the page

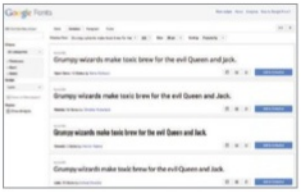
Every webpage contains text and it is frequently the main content. Therefore you should ensure that it looks good by choosing the fonts and styles to fit your design and the image you want to portray. There might only be a small amount of text if the main content focuses on images, but even so, there will be titles, links and captions. There's no getting away from text on webpages.

You can add CSS code to a webpage to define how the text is displayed, but placing it in a separate .css file and linking to it in the page's head section means that the same text styles are applied everywhere in your site. HTML font tags have been replaced by CSS font-family definitions, old-fashioned <i> and <b> tags are no longer needed, and font-style and font-weight enable you to set the styles to normal, italic, bold and so on. It might seem like extra work for one webpage, but when you are creating a site with 100 pages it is a considerable time-saver.

Writing the CSS rules for the text is simple and you can create a class that defines a text style and then use it over and over again in the page. We only look at headings and body text here, but expanding the CSS yourself should not present any problems.



Customise your text



Pick a font

01 Using CSS you can set the font for all the text enclosed by tags on all webpages with this in your style.css file. Link to it in the page header using <link rel="stylesheet" type="text/css" href="mystyle.css" />. This bit of code sets the font family to Verdana:

```
001 p {
002 font-family: Verdana;
003 }
```

Select a family

02 When a font name contains spaces, you must put quotes around it or the CSS won't work. It is also a good idea to offer an alternative font in case the visitor to your site doesn't have the one you specified. Choose a family such as serif (fancy) or sans-serif (plain).

```
001 p {
002 font-family: Verdana, sans-serif;
003 }
```

"You can create a class that defines a text style and then use it over and over again"

Size and style

03 The text may be fine at its default size, but if it isn't, you can set the size in pixels. The size you need depends on the font, so experiment with 10px to 16px. The font-style can be normal or italic and the font-weight can be normal, bold, bolder or lighter.

```
001 p {
002 font-family: Verdana, sans-serif;
003 font-size: 14px;
004 font style: normal;
005 font-weight: normal;
006 }
```

Define a class

04 There may be paragraphs that you want to make bold or italic, so here are three classes that define three styles of text. It varies the font-style and font-weight, but you could also create small text, large text, styles for panels, captions for images and so on.

```
001 p.normal {
002 font-family: Verdana, sans-serif;
003 font-size: 14px;
004 font-style: normal;
005 font-weight: normal;
006 }
007 p.italictext {
008 font-family: Verdana, sans-serif;
009 font-size: 14px;
010 font-style: italic;
011 font-weight: normal;
012  }
013 p.boldtext {
014 font-family: Verdana, sans-serif;
015 font-size: 14px;
016 font-style:normal;
017 font-weight: bold;
018 }
```

Use different classes

05 Step 4 created three CSS classes for the tag. When you want text to appear in one of these styles, you add a class parameter to the tag. The style is used for all the following text until you either change it with another tag or end it with .

```
001 <p class="normal">This is normal
text</p>
002 <p class="boldtext">This is bold
text</p>
003 <p class="italictext">This is
italic text</a>
```

Define the headings

06 In addition to body text, there are also headings using h1 to h6 tags. You don't need to define all of them, just the ones you use. Do it exactly like the p tag and set the font, size, style and so on. We've also reduced the bottom margin below the header.

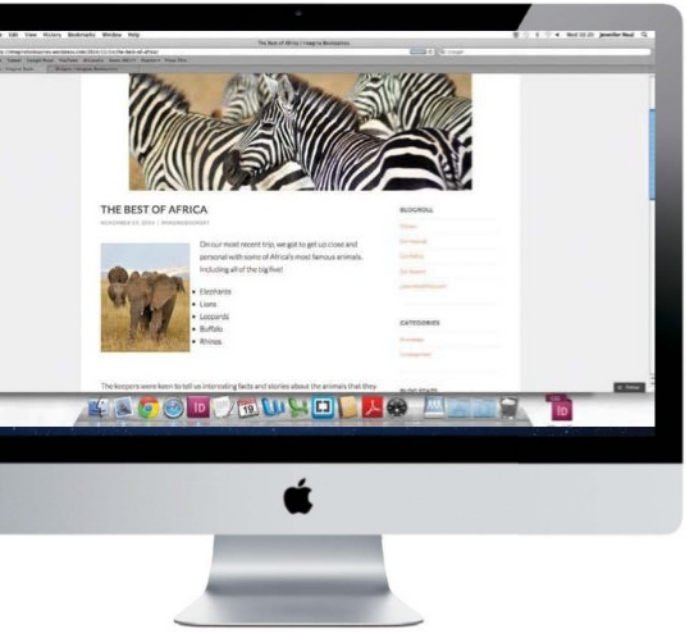
```
001 h1 {
002   font-family: Arial;
003   font-size: 30px;
004   font-weight: bold;
005   margin-bottom:-10px;
006 }
007 h2 {
008   font-family: Arial;
009   font-size: 20px;
010   font-weight: bold;
011   margin-bottom:-10px;
012 }
```

Style your lists to stand out

Standard lists tend to be a bit boring, so spice them up with some CSS styling

Although some web pages may require a lot of text, it is generally best to keep it brief and to the point. People scan a page for interesting content and then move on if they don't spot anything straight away. Lists are therefore very useful for drawing the visitor's attention to some content and for presenting it in a way that can be digested quickly.

The HTML <ol> and <ul> tags are used to create ordered (numbered) and unordered (bullet) lists, which we explored on p22. They are useful if unexciting, but with CSS list-style-type you can change the bullet to different shapes like a circle or square. It is even possible to replace it with an image specially created to fit in with your site's design and colour scheme using list-style-image. You can change the font used to display lists so they match the body text. If you have used HTML lists you will have noticed that they are indented, but with CSS you have fine control over the positioning using padding and margins. You can tighten up the layout or space it out – it's your choice.



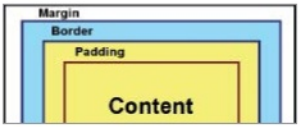
Customise your lists

Create a class

01 You should have a separate file called style.css for CSS code, so load it and create a new class called liststyle like this:

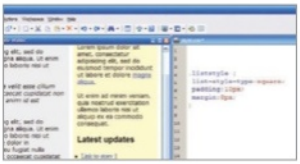
```
001 .liststyle {
002       list-style-type: square;
003 }
```

It contains just one item called list-style-type, but there are over 20 different possible values. These include circle, square, lower-roman, disc, decimal, upper-latin, and others. Try each one to see what they are like.



Boxed lists

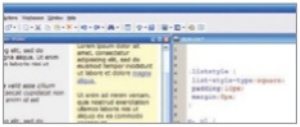
02 CSS displays each element of web page content in a series of boxes. This means that a list defined by a ul or ol tag is separated from other elements on the page by a margin. Inside the margin there may be a border, but it is optional and not always used. Between the border and the content is padding or space.



Position the list

03 With those boxes in mind, we can position the list within the space allocated to it on the page. Padding and margin have been added to the liststyle class and set to 10 and 0 pixels respectively. Experiment with different values to see how they affect the position of the list.

```
001      padding: 10px;
002      margin: 0px;
003 }
```



Style the text

04 Notice in the last step that the list text has a different font to the body text. That's because no style is set for lists. We defined the font and size for the body text in a previous tutorial and we can simply add ul to the p code so it uses the same font and size as the body text.

```
001  p. ul{
002      font-family: Verdana;
003      padding: 10px;
004      font-size: 14px;
005 }
```

Choose a background

05 Many HTML elements can have a background colour and it is applied to both the content and padding areas, but not the border or margin. A background-color of #ffd0d0 has been set, but if you are not familiar with this notation, you can use common names like red, yellow, blue and so on. This is less flexible, though, hex codes can be easily found on Photoshop or online.

```
001      background-color: #ffd0d0; }
```

Inside vs outside

06 You'll notice that the bullet blobs are outside of the background colour. This is their default position, but it's possible to change this and make them appear in the background area. In this step, list-style-position is set to inside and this makes the list look more attractive.

```
001      list-style-position: inside;
002 }
```

"With CSS you have fine control over the list's positioning using padding and margins"

Add a border

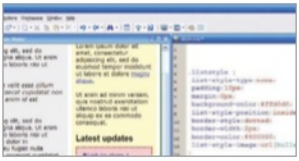
07 As we've discussed, each HTML element can have a border. We haven't specified one so far, so one hasn't been drawn. However, we can easily define one by setting border-width to the number of pixels and border style to dotted, solid, dashed, double, groove, ridge, inset or outset.

```
001      border-style: dotted;
002      border-width: 2px;
003 }
```

Colour the border

08 Most parts of the list can be coloured and this includes the border. The border-color has been set to #800000, which is a medium-dark red. It's a good idea to specify the colour even if it's black, because default settings for one web browser might be different to the defaults in another. Better to be safe.

```
001      border-width: 2px;
002      border-color: #880000;
003 }
```



DIY bullets

09 Blue squares for bullets? Yes, we removed the standard bullets by setting list-style-type to none at the start of the CSS code. At the end we have list-style-image:url(bullet.png), which tells the browser to use an image called bullet.png that we created earlier in a paint program and saved with the web page.

```
001 border-color: #880000;
002 list-style-image:url(bullet.png);
003 }
```

Turn lists into navigation bars

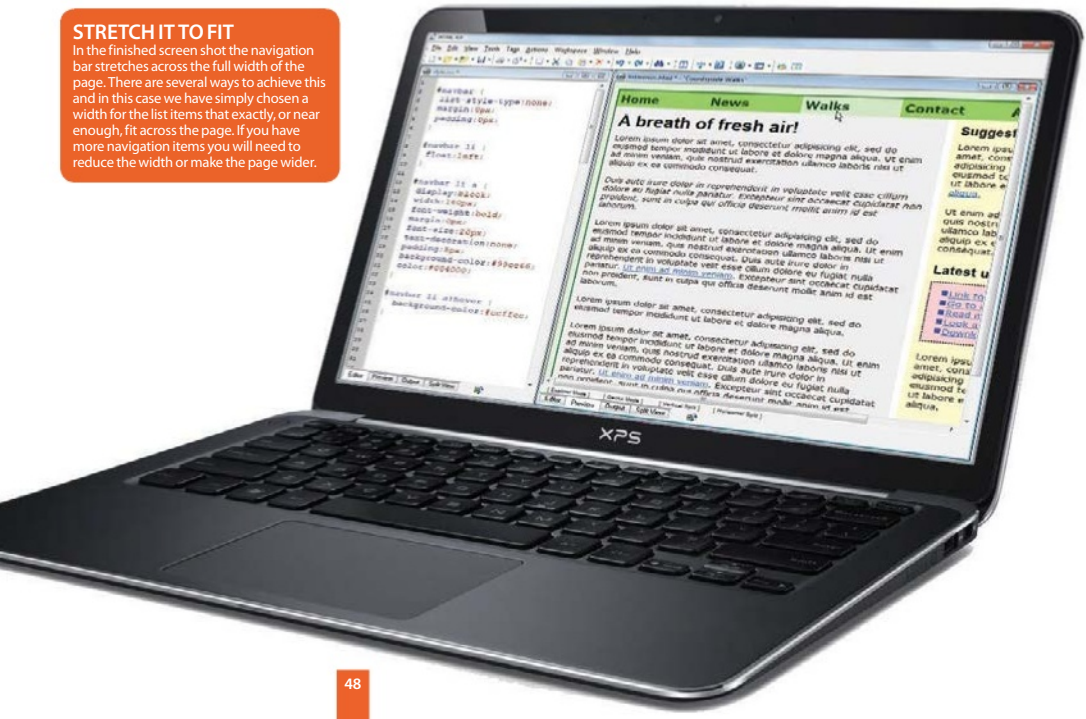
Every website needs some sort of navigation, so why not use CSS for it?

Website navigation is an important topic and it is something that you need to put some thought into as you are designing your website. At one time we would have used button images created in a paint or photo program and then used JavaScript to create a rollover effect when the mouse hovered over one. But that's yesterday's technology and modern sites use CSS instead.

A CSS navigation bar looks like a regular unordered list in the HTML of the page, but using CSS we can manipulate the list elements and turn them into the website's navigation. Instead of the list running down the page, the list elements can be displayed across it instead. We can style the links just like any other text and create a custom look for the navigation bar. The result looks nothing like its original plain HTML layout and it really shows off the power of CSS.

You can copy the code into your own web pages and it will work fine. All you need to do is to change the link text to correspond to the pages or sections on your own site, and style the text to fit in with the fonts and colours that you use. It is really quite straightforward once you see how it works for yourself.

STRETCH IT TO FIT
In the finished screen shot the navigation bar stretches across the full width of the page. There are several ways to achieve this and in this case we have simply chosen a width for the list items that exactly, or near enough, fit across the page. If you have more navigation items you will need to reduce the width or make the page wider.



Create a list

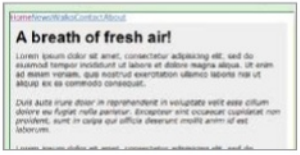
01 An unordered list (ul) is used to provide links to other sections of the website. It's all standard HTML and it's exceedingly dull compared to CSS. Try it and see. In the ul tag though, we have added id="navbar" and we'll use it to change the way the list is displayed.

```
<ul id="navbar"
<li><a href="index.html">Home</a></li>
<li><a href="news.html">News</a></li>
<li><a href="walks.html">Walks</a></li>
<li><a href="contact.html">Contact</a></li>
<li><a href="about.html">About</a></li>
</ul>
```

Make it plain

02 In the last tutorial we saw how to style lists and the bullets can be removed by setting list-style-type to none. The padding and margin can also be set to zero. This produces a very plain list of links at the top of the page that's not very exciting.

```
#navbar {
list-style-type: none;
margin: 0px;
padding: 0px;
}
```



Floating elements

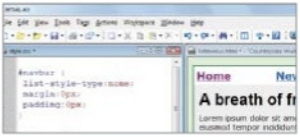
03 Normally each list item is displayed on a separate line, but it is possible to remove the line breaks by using float. Setting the list elements (#navbar li) to float left causes them to be displayed one after the other on the same line, like a navigation bar.

```
#navbar li {
  float: left;
}
```

Spread them out

04 All the links are bunched up together and step 3 doesn't look much like a navigation bar. What we need to do is to make each link a fixed width. We define the a link within li of navbar (#navbar li a) as a block and set its width to 140 pixels.

```
#navbar li  a {
  display: block;
  width: 140px;
}
```



Style the text

05 Remember how we styled some text back on page 44? We can do the same here. Add the formatting information to #navbar li a to change the way that <a> links look in <li> items. Set the font size and weight, margin and padding, and so on.

```
#navbar li  a {
  display: block;
  width: 140px;
  font-weight: bold;
  margin: 0px;
  font-size: 20px;
  padding: 5px;
}
```

Colour and decoration

06 Three items have been added and immediately noticeable is the colour. The text colour has been set to dark green and the background has been set to light green with color and background-color. Finally, text-decoration:none removes the underline from the links.

```
#navbar li  a {
  display: block;
  width: 140px;
  font-weight: bold;
  margin: 0px;
  font-size: 20px;
  padding: 5px;
  text-decoration: none;
  background-color: #99cc66;
  color: #004000;
}
```

"The result looks nothing like its plain HTML layout and it shows off the power of CSS"

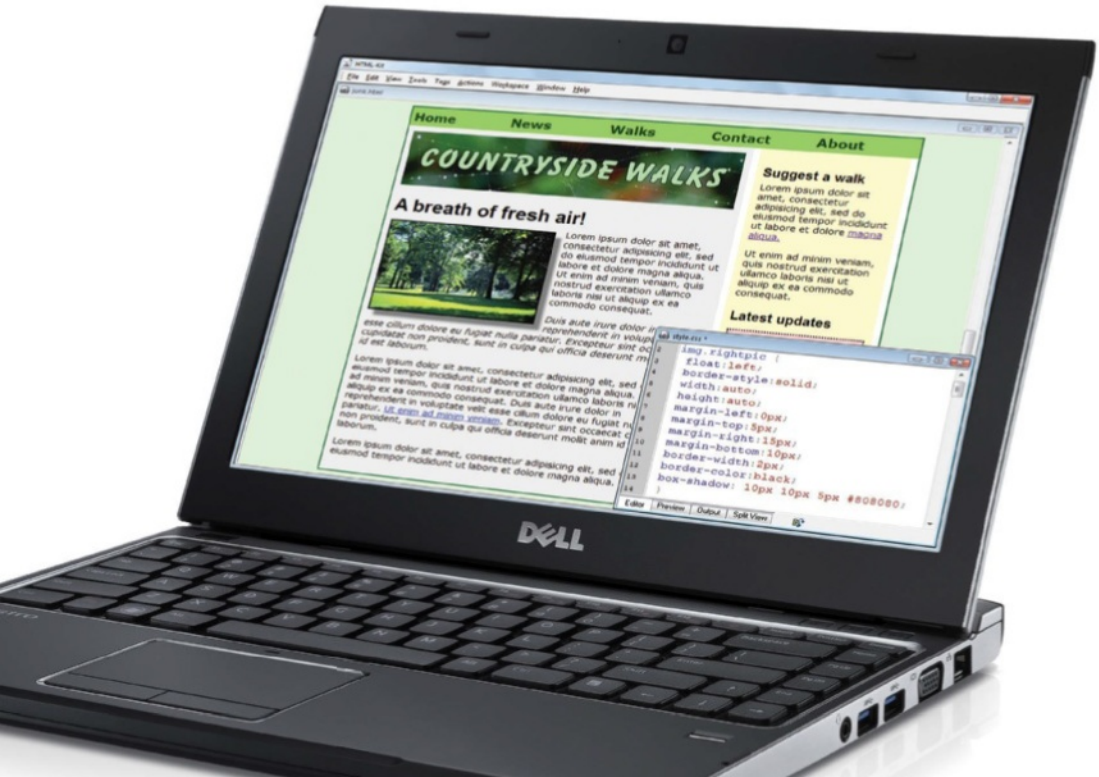
Format images using CSS

Use CSS to gain precise control over the display of images on your website

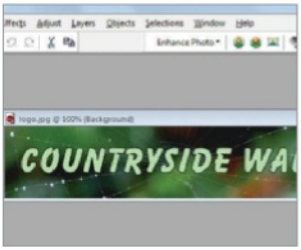
A webpage would be pretty dull without any images on it! Some images may appear on every page, such as a graphical website logo or heading, while others may be specific to each page. You will need to create the images for your site before you start. Remember to use them sparingly and keep the file size as small as possible to ensure that webpages load quickly for visitors.

Images are inserted into webpages using the HTML <img> tag and there are attributes to position it left and right, set the size and so on, but those are dated. If you use CSS to format images instead, there are many more options available to you. For example, you can add a border, set its thickness and its colour. The spacing around each side of the image can be individually adjusted, text can be wrapped around it on the left or the right and so on. The latest web browsers that are up to speed with CSS version 3 can display advanced image attributes like drop shadows that give the page a 3D look.

Forget the old HTML way of doing things and take advantage of CSS and its powerful features. Our tutorial will guide you through the process.



A guide to CSS images



Create the images

01 Load your paint program or photo editor and create the images for your webpage. There may be a graphical logo, as shown here, and one or more other images to be displayed in the page. It is usually best to save photos as JPEGs and illustrations as 256-colour PNGs.

Insert the images

02 Images are inserted into the HTML of the page using the img tag, with src pointing to the filename. Add an alt attribute as an image title: it's useful for search engines. You want to make sure that your image sits in a complimentary place on the layout, this includes images on the page. Add class="picright" to the img tag:

```
001 <p><img class="picright"
src="photo.jpg" alt="Grass and tress">
```

"If you use CSS to format images, there are many more options available to you"

Build a class

03 Adding img.picright to the CSS file adds the class we need to format the image. The float:right; causes the image to be displayed on the right – float:left; would display it on the left. In both cases the following text will wrap around the image on the left or right:

```
001 img.picright {
```

Add some borders

04 As with many other HTML elements, in CSS images can have borders. The border-style can be solid, dotted, dashed and so on, the border-width can be specified in pixels and the colour selected with border-color. Use colour names or rgb(x,y,z) values where x,y,z are numbers 0-255.

```
001 img.picright {
```

Define the margins

05 You may find that the text runs right up to the image, which might not look very attractive. It can be prevented by setting margins. We've exaggerated the effect with a 30-pixel margin; 10 is usually fine. Margin-top, left, bottom and right can be set separately if necessary.

```
001 margin: 30px;
```

Size the images

06 If you don't specify a size for the images, they will be displayed at whatever size they are. In some situations, though, you might want to display an image at a different size, such as when showing thumbnails. Set the width and height to the number of pixels to make the image that size.

```
001 width: 350px;
002 height: 200px;
```

Add a background image

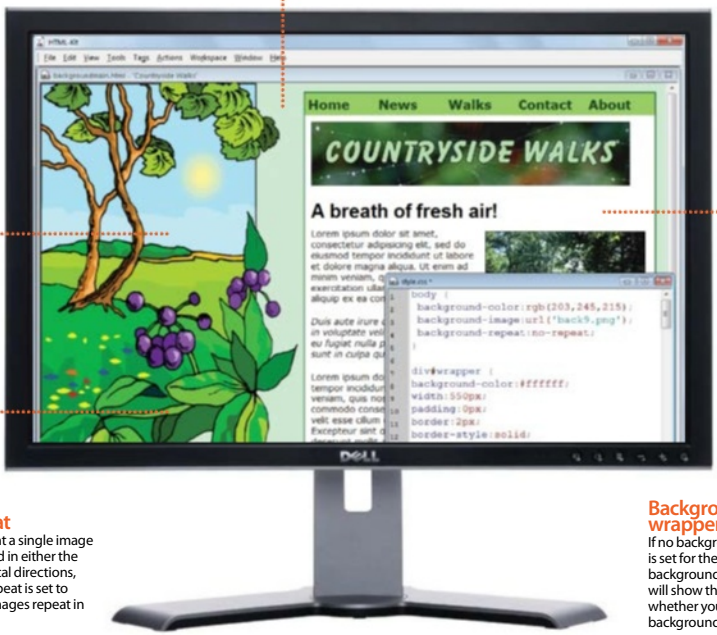
A great image can really enhance your page design

The fewer images your website has, the faster it will load, but plain colours for the background and lots of text might be a turn off for your visitors. A few key images can greatly enhance the look of a website and a background image has the advantage of being used on every web page. This means that the browser only has to download it once and then it caches it ready to be used on the next page. The end result is a great page design with no time wasted downloading large images.

In this tutorial we will start with a plain coloured background and then show how to add an image. It can be a large one that fills the browser or a small one that can be stretched to fill or tiled across or down the screen. Tiled images are great for reducing the size of a page because they are so small (for more information on how to create your own, check out page 160). If you use a large image you must bear in mind the size of the file and try to minimise it without affecting the quality.

Create an image
Load your photo editor or paint program. Create an image or find some clipart that you can use as is or easily modify. Save it as a compressed JPEG or 256-colour PNG

The background colour
The background colour of the page is set using CSS definitions in the body tag. It is set to the same colour as the background for the image so the two blend in like one



Set the repeat
In this case we want a single image that is not repeated in either the horizontal or vertical directions, so background-repeat is set to no-repeat. Tiled images repeat in both directions

Background wrapper colour
If no background colour is set for the wrapper, the background colour of the page will show through. It's up to you whether you set a #wrapper background colour

Impressive backgrounds

Pick a colour

01 The simplest type of background is a plain colour. This is achieved by placing a background-color style in the body tag stored in the CSS file, style.css. Use words like red, yellow, blue, green or the #RRGGBB hex value.

Use a photo

02 It is easy to use a photo for the background and we just add a line to the body CSS that links to it. Add background-image:url('back1.jpg'), replacing the filename with whatever your file is called.

```
001 body {
002 background-color: rgb (128,128, 128);
003 background-image:url('back1.jpg');
004 }
005
```

Modify the wrapper

03 As well as inserting an acceptable number of hyperlinks, you can also flag up certain content. Place words and rogue websites, IP addresses and emails in this box and any posts containing such content will immediately be placed in the moderation queue where you can assess whether or not to allow it to be published on your site. If your banned words form part of larger words, they too will be picked up.

```
001 div#wrapper {
002   width: 750px;
003   padding: 0px;
004   margin-left: auto;
005   margin-right: auto;
006 }
```

Change the text

04 We can change the text colour. We created some CSS code to set the font and size in an earlier tutorial. Now we can add color:white to the p tag definition. This image has both dark and light areas, so it's not perfect.

```
001 p, ul {
002   font-family: Verdana;
003   font-size: 14px;
004   color: white;
005 }
```

A background wrapper

05 Here, the background is solid, but the text is on an image. Anything can have a background, so we can move background-image:url('back2.jpg') from the body tag to the #wrapper definition. We chose a paler picture!

```
001 div#wrapper {
002   background-image:url('back2.jpg');
003   width: 750px;
004   padding: 0px;
005   border: 2px;
006   border-style: solid;
007   border-color: #006600;
008   margin-left: auto;
009   margin-right: auto;
010   }
```

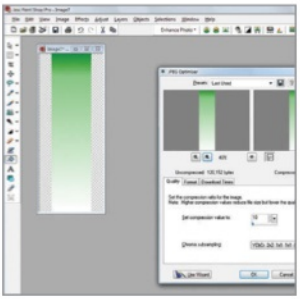
Resize the background

06 What size should the background image be? The web page should load quickly so visitors aren't waiting around, so it is best to create either a small image or use lots of JPEG compression.

"Tiled images are great for reducing the size of a page"

Use smaller images

07 If you look down the left side it is clear that the web browser has repeated the image down the page because it didn't fill the full height. It has repeated it across the page too. It doesn't look too bad on the left, but the right edge shows all the joins. Sometimes you can get away with repeating images and sometimes not.



Create a gradient

08 We have created a small image that has a simple gradient that runs from dark green at the top to white at the bottom. This will be used as the background for the page and even when exported at a reasonable quality it doesn't create a very big file. Tiled backgrounds are great for creating fast-loading web pages.

See through wrapper

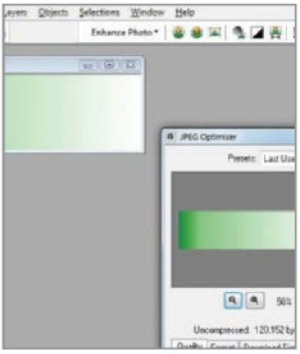
09 By removing the background colour in the #wrapper, the background shows through. You can see that the background image has tiled nicely across the screen and each image fits seamlessly with the next. However, it is ugly where it has been tiled vertically.

Tile it across

10 Now this is far better. What we've done here is to add:

```
001 background-repeat:repeat-x;
```

to the CSS in the body definition. This instructs the browser to repeat the image across the page only (the x direction). The background colour is set to white, which is the final gradient colour so it blends in.



Horizontal gradients

11 The gradient background image was loaded back into the photo editor and rotated through 90 degrees to make a horizontal gradient. Notice how small it is, both in physical dimensions and in file size. The image does not need to fill the whole screen because it is going to be tiled. This reduces the file size.

Tile it down

12 The image has been tiled down the screen, but not across it. This is because of the background-repeat:repeat-y; instruction in the CSS code for the body tag. It is possible to repeat either in the x (horizontal) or y (vertical) directions. If you don't specify then it is tiled in both directions.

Fixed vs scrolling

13 The background image is tiled and the #wrapper background is white. The main difference, though, is the background-attachment in the body CSS definition. If background-attachment is set to fixed, the background stays still when the page scrolls. Set it to scroll and it scrolls with the page. You have to try it to see the difference.

Grab a background

14 If you are not very good at creating your own backgrounds in a drawing or paint program you will find lots of sources of background images on the web. Background Labs (**www.backgroundlabs.com**) and GRSites (**www.grsites.com/archive/textures/**) have lots of images and a large number of them are textures and images for tiling on the page.

Resize vs stretch

15 This image is from Background Labs. It is a small image will not fill the browser window and it is not suitable for tiling across or down the page. You have two choices and you can either resize it in the photo editor or resize it within the page using CSS.

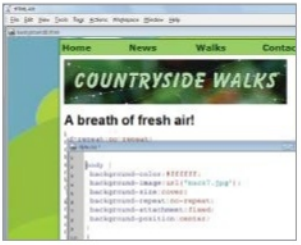
Resize the background

16 The CSS code for the body tag has been modified and background-size:1280px 1024px; has been added. This is a CSS 3 feature that is only supported by the latest versions of web browsers. This one has ignored it! In IE9, Chrome and Firefox it will stretch the image to the size specified.

```
001 body {
002   background-color: #ffffff;
003   background-image:url('back7.jpg');
004   background-size: 1280px 1024px;
005 }
```

Cover the background

17 Specify a size for the background and someone with a laptop may find part of the image is hidden. Someone with a large monitor may have a browser width of 1500 pixels and then the background will be tiled. Use background-size:cover; and the image is stretched to fit.



Background positioning

18 One final bit of background-related CSS is the positioning. In the body code is background-position:center; and this centres the background image in the browser window. Other values include left top, left center, right top, right center, center bottom and so on, or just use pixels like 50px.

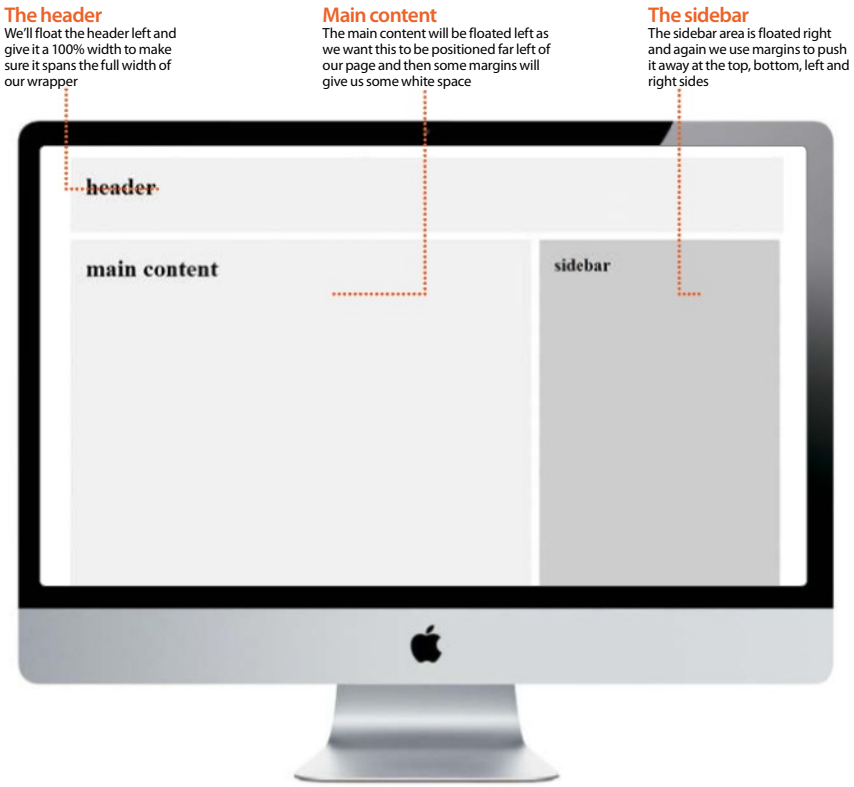
```
001 body {
002   background-color: #ffffff;
003   background-image:url('back.jpg');
004   background-size: cover;
005   background-repeat: no-repeat;
006   background-attachment: fixed;
007   background-position: center;
008 }
```

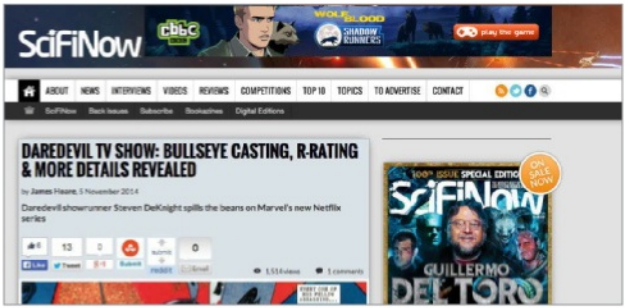
Style a two-column layout

Use the HTML and CSS skills you've learnt so far to create and style a two-column page layout

Using HTML and CSS to create and style a two-column website layout – one of the most common layouts – is surprisingly straightforward and very flexible. Almost all websites today use some kind of column layout with at least a sidebar and main content areas. So it's one of the basic skills to have when designing websites, and once you understand these fundamentals you can move on to bigger and more elaborate designs.

In this tutorial, then, we are going to open up our favourite text editor (we will be using Adobe Dreamweaver, but the humble NotePad or TextEdit is fine) and learn how we can shorten, lengthen and swap any column we need just by using CSS. Then we will discuss CSS floats, why we use percentages and a few other tips along the way.





Getting started

01 First thing we need to do is create new 'index.html' and 'styles.css' files and place them in the same directory. It is considered best practice to also place all your CSS files within their own folder called 'css' as some use separate CSS files for certain things such as Internet Explorer-specific styles. However, we will keep it simple here and leave them sitting in the same location.

Linking the CSS

02 Now that we have our new index.html and styles.css files, we will need to link them up together within the <head></head> tags of our HTML. So open up the index.html file and just underneath the closing </title> tag, type in your link to your CSS.

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
    <title>3 column layout</title>
<link rel="stylesheet" hreg="style.css">
</head>
<body>
</body>
</html>
```

The wrapper

03 With the 'index.html' file still open, lets add in a 'wrapper' div that will be used to centre all our content on the screen later on using CSS. What we have done here is use an ID (<div id='') instead of using a class because this is going to be a main part of our page's structure and will only be used the once throughout our code.

```
<body>
        <div id="wrapper">
        </div>
 </body>
```

Header

04 Firstly, we'll add the header. We are going to give it an ID ('<div id=' called 'header') and place this just underneath our wrapper div. Then we have added in a title of our section within an '<h1>' header tag so we can know what's what.

```
<body>
    <div id="wrapper">
        <div id="header">
        <h1>header</h1>
        </div><!-- END header -->
    </div><!-- END wrapper -->
    </div>
</body>
```

Main content

05 So next up we need to add a main content section that will be positioned on the left side of our layout (see page 56). Like we did in the last step we have created an ID tag and called 'main_content', to clarify it from the header class, and added it in just underneath our 'header'. And again so we can see what section is what, we have added in a title using the '<h1>' tag.

```
<div id="main_content">
        <h1>main content</h1>
</div><!-- END content -->

```

Sidebar section

06 The sidebar is next and this is the area that will be positioned over to the right and would normally be used for extra navigation links or a search field – things like that. So again we will use an ID called 'sidebar' and add it just under the 'main_content' closing div ' '.

```
<div id="sidebar">
        <h2>sidebar</h2>
</div><!-- END sidebar -->
```

The footer

07 Now the footer speaks for itself and will be our last HTML section we will be adding. So again let's give it an ID name of 'footer' and add it in underneath the closing 'main_content' div. Then again place in a header tag using a '<h3>'. You will also notice we have added in an HTML comment to every end div tag ( ) to give us a clear indication of where each section ends. This will help you if your HTML mark-up gets very busy.

```
    <div id="footer">
        <h3>footer</h3>
    </div><!--END footer-->
</div><!-- END wrapper-->

```

CSS reset

08 Now open up your 'styles.css' file and what we are going to do first is add what is called a 'reset'. The reset uses the universal selector and this just tells all browsers (including IE) to clear all default styles, such as padding, margins, line-height etc. Doing this will give you a clean slate to work from.

```
/* reset */

* {
  padding: 0;
  margin: 0;

}
```

Wrapper styles

09 Now the reason for our wrapper is to 'wrap' everything within a containing section (element) so we can centre it within the viewpoint of our screen. So here we have used margin to zero out the top and bottom margins and allow CSS to automatically calculate the left and right space based on our width. Our width will be 960px because that aligns up nicely with a screen resolution of 1080 x 760px (lowest common denominator).

```
#wrapper {
   width: 960px;
   margin: 0 auto;
}
```

The header

10 Now, for the header section we are going to float it left and give it a width of 100%, which will guarantee our header spans the whole width of our '#wrapper'. Then we push the header down 10px using margin. Here we have used what is called shorthand CSS, which enables us to only pick the top margin. Now give the background a colour and then 100px height.

```
#header {
         float: left;
         background: #f1f1f1;
         margin: 10px 0 0 0;
         width: 100%;
         height: 100px;
}
```

Main content styles

11 Now to add some style. First thing is to float it left. Then give it an off-white background colour and then by using margins we can give it some breathing space. We then specify its width and height taking into consideration the overall width of our wrapper so we have enough room for our sidebar.

```
#main_content {
         float: left;
         background: #f0f0f0;
         margin: 10px 5px 10px 0;
         width: 620px;
         height: 630px; }
```

The sidebar styles

12 The sidebar will be positioned to the right-hand side and this is at most expected by the user/visitor. So it makes sense to float to right. And then use margins again to give us some white space. Our sidebar background will be a darker grey than the rest and it will be a fixed width of 324px, which should give us plenty of room for any inner content. Then we give it the same height as the main content section.

```
#sidebar {
         float: right;
         margin: 10px 5px 5px 6px;
         background: #ccc;
         width: 324px;
         height: 630px;
}
```

The footer styles

13 The footer styles should be fairly straightforward now. However, we have a new property to look at, which is the 'clear:' property. Setting this to 'both' will make all floated elements pump up over the footer allowing our footer to sit where it should be. Then we float the footer to the left and give it some dimensions.

```
#footer {
        clear: both;
        float: left;
        background: #ddd;
        width: 100%;
        height: 100px;
}
```

Moving the sidebar

15 To get your sidebar to the right, change the float value of the '#main_content' rule from 'left' to 'right'. But what you want to be conscious of is naming your divs. Name your sidebar something like ' ', something that is clear to you.

```
#main_content {
        float: right;
        background: #f0f0f0;
        margin: 10px 5px 10px 0;
        width: 620px;
        height: 630px;
}
```

Bigger two columns

16 As you can imagine, it's very easy to make two larger content areas and not have a sidebar at all – even though it still can be called 'sidebar'. So in your CSS, if we think about how much space we have to play with, which is 960px, and divide that into two. However, then we have to compensate for our margins, so it may take you some messing about to get the right spacing without pushing anything out.

Be inspired

17 Now you've finished your basic two-column layout, take at look at some of your favourite websites and see which layout they employ. As shown above, **www.webdesignermag.co.uk** uses a two-column layout too.

Research alternatives

18 The two-column layout is in no way the be all and end all. Within this book you will find the tools to create a three-columned layout, and be able to learn the skills your acquire to customise any layout to whatever you have chosen.

Create a header

Learn how to create a clean, simple and effective header for your website

Having clean and clear header for your website can be the difference between a successful and user-friendly experience and a cluttered and disorganised website that no one will want to visit again! You don't want a first-time user coming to your site and not know how your navigation works, or your company branding (logo) hidden under a load of over-useless graphics or text – it doesn't even really matter how effective the rest of your layout is.

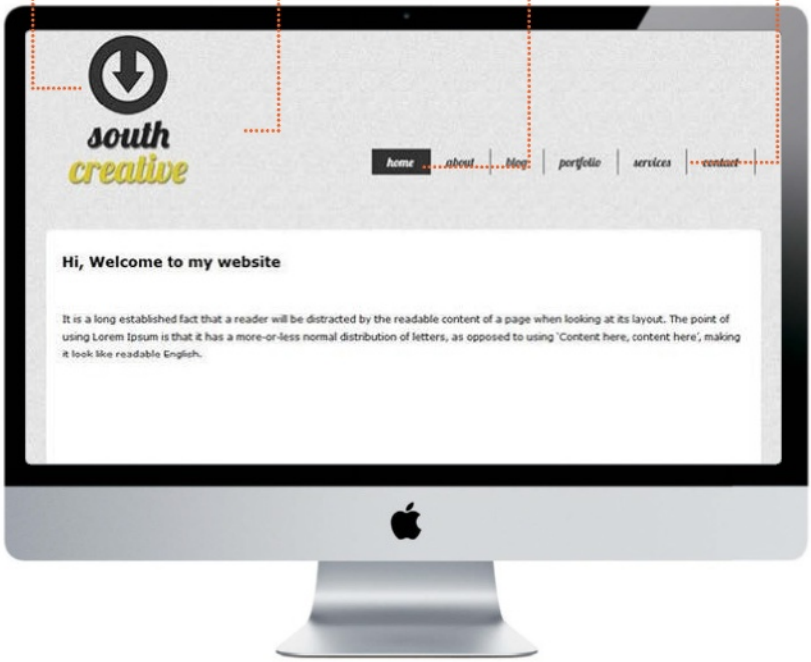
Yes, this sort of thing has been known to happen (more often that web designers would like to admit!) and this is why so many designers are turning to a relatively new role called user experience or 'UX'. So in this tutorial we will take a look at how we can create a simple, clean and useful header for your site.

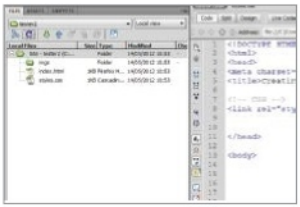
The logo
The logo can be either a graphic or just text. You will also notice we have used the same font for both the logo and navigation

The background
Creating a textured background will always help stand your web page out from the crowd – gone are the days of white backgrounds with nothing but text and images

The hover state
The hover state is nicely styled and can be easily changed whenever needed just by changing both the background colour or font colour

Separators
Adding a separator to our navigation links doesn't need to be pure CSS as we did, but it is handy using the 'border' property





Rinse and repeat

01 To get started with this tutorial, you need to have set up your layout; follow the tutorial on page 56 to get started!

Adding the logo

02 Now let's place our logo within the header section; you can use your own or grab ours off the disc. We are going to use an ID again and call it 'logo'. Then we will wrap it within an anchor tag and link it to our page by adding in the index page within '<a href="index.html">'

```
<body>
    <div id="wrapper">
        <div id="header">
        <a href="index.html"><div id="logo"></div></a>
        </div><!-- END header -->
</body>
```

Main content

04 We will now finish off our HTML markup with some content underneath our header. Just underneath our closing header div ' ', let's add in a div ID of 'main_content' and add in a welcome title using header tags '<h2>' and then some dummy text using '<p>' tags.

```
    <div id="main_content">
        <h2>Hi, welcome to my website</h2>
        <p>It is a long established fact that a reader will judge a website by it's layout.
    </div><!-- END main content -->
```

```
<title>Creating a sidebar</title>

<!-- CSS -->
<link rel="stylesheet" href="styles.css">

</head>

<body>

<div id="wrapper">

    <div id="header">

    <a href="index.html"><div id="logo">

      <h1>Your logo goes here!</h1>
```

Navigation list

03 In this step we are going to add in the navigation. What we will use here is a standard unordered list and each list item (<li>) will have a link to each page using the '<a href="' attribute. Then we will give our unordered list a class name of 'navigation'.

```
<body>
        <div id="wrapper">
                <div id="header">
                <a href="index.html"><div id="logo"></div></a>
                        <ul class="navigation"></li>
                        <li><a href="">home</a></li>
                        <li><a href="">about</a></li>
                        <li><a href="">blog</a></li>
                        <li><a href="">portfolio</a></li>
                        <li><a href="">services</a></li>
                        <li><a href="">contact</a></li>
                        </ul>
                </div><!-- END header --> </div>
 </body>
```

The CSS

05 Now open up your 'styles.css' file and at the top put in our simple reset. The reset does exactly that – resets every element to zero, which will clear all the default styles that most browsers put on. Then, using the body tag we can give our page a textured background using an image and leaving it to repeat across the page. And then we set the 'font-family' and 'size', here we have added multiple fonts just incase our original chosen font isn't supported.

```
* {
  padding: 0;
  margin: 0;
}

body {
  background: url('imgs/bg.jpg');
  font-family: Verdana, Geneva, sans-serif;
  font-size: 12px;
}
```

The wrapper

06 Now we are going to centre our header and content using the 'wrapper' div. What we are doing here is making sure we have no margin at the top or bottom and automatically find the difference on the left and right that is in relation to our fixed width – which we set to 960px. 960 pixels is mostly used because it fits nicely within the lowest common denominator of screen resolutions (1080 x 760).

```
#wrapper {
        margin: 0 auto;
        width: 960px;
}
```

The footer

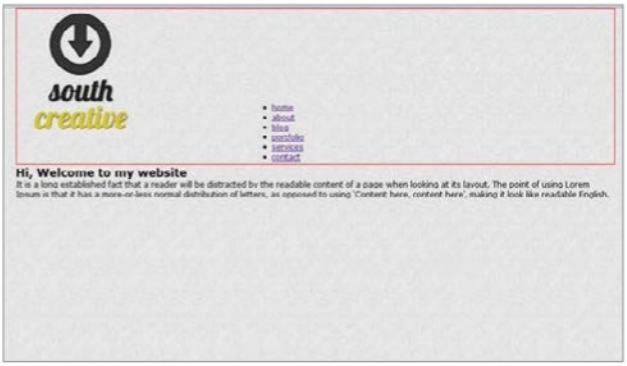
07 Now for our header we are going to float it left and give it a 100% width so it spans the full width of our wrapper div, then give it a height of 250px. Let's also add in a red border so we can see the header more clearly and get a better visual idea of where we need our logo and navigation, which we will add in over the next few steps – then once happy we will remove it.

```
#header {
        float: left;
        width: 100%;
        height: 250px;
          border: 1px solid #f00;
}
```

The logo

08 Adding the logo is going to be very simple. All we are going to do is locate our logo that we have inside our 'imgs' folder and then make sure it doesn't repeat. Then position it on the left by floating it left and then specify its height and width. A thing to remember here is that if you don't specify its dimensions you will not see it on the webpage.

```
#logo {
   background: url('imgs/sc_logo.
png') no-repeat;
   float: left;
   width: 200px;
   height: 200px;
}
```



The navigation

09 Now let's style our navigation. We do this by floating the whole element right then positioning it 150px down and then pulling it over to the right using a negative margin of '-50px'. Never be worried about using negative margins as they work very well. Then we give the navigation a width of 600px.

```
.navigation {
        float: right;
        margin: 150px -50px 0 0;
        width: 600px;
}
```

Navigation buttons

10 At the moment our navigation looks nothing like a horizontal navigation bar, more like a basic list of links. So let's make it go horizontal by floating each '<li>' item left, which will push them all horizontal. Then we can take away the default item bullets by specifying the 'list-style' being none.

```
.navigation li {
        float: left;
        list-style: none;
}
```

The content

12 Now let's style our page content. While we're here let's get sneaky and add some CSS3 properties to make the background have rounded corners by using 'border-radius: 6px'. We then style the paragraphs give it some padding to both the paragraph text and header tags.

```
#main_content {
        float: left;
        width: 900px;
        height: 400px;
        background: #fff;
        border-radius: 6px;
}
        #main_content p, h2 {
                line-height: 22px;
                margin-top: 10px;
                padding: 20px;
        }
```

Apply the Google font

14 The next step is to click the 'use' grey button located at the very bottom right and scroll down the page slightly. What we have first is the 'link' tag line that will link straight to the Google font servers and will always be available. That way you can guarantee everyone will be able to see your chosen font. Copy the link to your clipboard. Then paste it into your 'index.html' file just underneath our other CSS link within the '<head>' tag.

```
<!-- CSS -->
<link rel="stylesheet" href="styles.css">
<link href='http://fonts.googleapis.com/css?family=Lobster' rel='stylesheet'
type='text/css'>
</head>
```

Navigation anchors

11 To give each anchor tag some padding, specify the text-decoration as 'none' and then create a separator using 'border-right'. We can remove the red border and add in a hover state that will change the background and text colour.

```
.navigation li a {
        float: left;
        color: #333;
        padding: 6px 20px;
        text-decoration: none;
        border-right: 1px solid #333;
        font-size: 16px; }

.navigation li a:hover {
        background-color: #333;
        color: #fff; }
```

Find Google Fonts

13 Looking at our navigation, we can clearly see that it doesn't look very attractive. So let's spruce it up somewhat by heading over to Google Fonts **www.google.com/fonts** and type in 'lobster' within the search field and you should see at least two choices of fonts. The one we want is the top style. Then click the blue 'add to collection' button to the right and now it ready to use.

Google font-family

15 Now let's copy the 'font-family' property and then open up the 'styles.css' file and locate the '.navigation li a' rule (a CSS rule is everything within the curly brackets) and paste the new 'font-family' underneath the 'font-size' property.

```
.navigation li a {
    float: left;
    color: #333;
    padding: 6px 20px;
    text-decoration: none;
    border-right: 1px solid #333;
    font-size: 16px;
    font-family: 'Lobster', cursive;
}
```

Create a sidebar

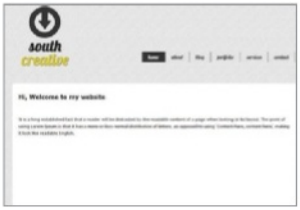
Learn how to create a clean navigation system for your website

A sidebar is without doubt the area that is most often created by web designers, because it's the most needed. It can hold all sorts of content: things such as extra navigation links, a search field and perhaps some small thumbnail images that relate to your website – it can be anything.

So in this tutorial we're going to create a simple page layout that has a functional navigation and a nice and simple sidebar that includes some content within. We're going to include within the sidebar a list of links for that extra navigation we mentioned and a search bar above and also some thumbnail images at the bottom to act as though we have a Flickr section – very useful for any photography-based sites. So open up your favourite text editor and let's get started.

Getting started

01 Follow the first steps from the Create a header tutorial on page 60. This will help you set up the basic layout that we will be adding to in this tutorial, to create the following:



Sidebar navigation

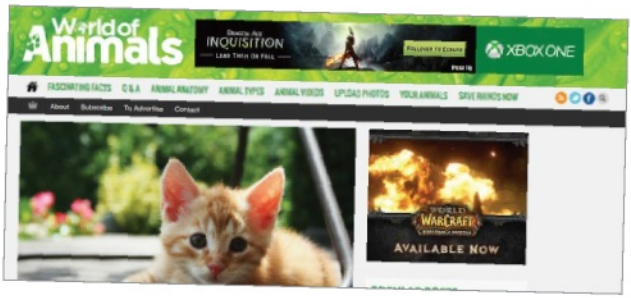
03 Now what we need to do is now is add in what will be our sidebar navigation. This is the exactly the same code as our top main navigation so it's just a case of copying and pasting that into the sidebar. But we will need to give it a different class name, which can be 'sidebar_list'. After the </form> tag add:

```
<ul class="sidebar_list">
    <li><a href="">home</a>
    <li><a href="">about</a>
    <li><a href="">blog</a>
    <li><a href="">portfolio</a>
    <li><a href="">services</a>
    <li><a href="">contact</a>
</ul>
```

Sidebar and search

02 Let's now add in our sidebar section. All we do here is create a div ID with the name of 'sidebar' just underneath the closing header div ' '. Then, using the code above, within our sidebar section let's add in a search field that will just sit there and look pretty. After the <!-- END header --> tag add:

```
<div id="sidebar">
    <form id="search form"       action="" method="get"
    <input id="search_term" type="text" value="" placeholder="search..."/>
    <input class="submit_button" type="submit" value="search">
    </form>
</div><!-- END sidebar -->
```



Sidebar images

04 We'll now add some images to sit below our navigation list. We have the same image here for both, set to the size of 100x100px. We've given them a class name of 'thumb' so we can use CSS to position them better without relying on the 'img' tag. Also we've given this section a title with the '<h3>' tag. After your navigation list add:

```
<h3>My flickr images</h3>
<img src="imgs/thumbnail.jpg" height="100" width="100" class="thumb">
<img src="imgs/thumbnail.jpg" height="100" width="100" class="thumb">
```

The CSS

05 Open up your 'styles.css' file and at the top add in the universal selector that will allow you to reset every element(<p>, <h1>, div, etc) to zero margin and padding. Then using the 'body' tag we can set the background colour to an off white '#f1f1f1' and set our default font style and size.

```
* {
 padding: 0;
 margin: 0;
}

body {
   background: #f1f1f1;
   font-family: Verdana, Geneva, sans-serif;
   font-size: 12px;
}
```

The wrapper

06 So with our wrapper acting as a container for all our page content, we can center everything using a fixed width and margin. We're specifying 0 pixels on top and bottom, with auto margins on the left and right. This is an easy and most often the best way of centring your page on your screen.

```
#wrapper {
        margin: 0 auto;
        width: 960px;
}
```

Stick to the right

If you spend some time surfing the internet in some detail, the chances are that you won't see too many websites with the sidebar fixed over to the left side – it is far more likely to be over to the right. Even though it is relatively easy to swap it over, you wouldn't be doing anyone any favours if you thought you would be clever and original by plonking it on the left – internet users are now accustomed to seeing this section on the right-hand side of their screens. If you're struggling to add content in the sidebar, then you can add a short summary about you or your business or perhaps even a small YouTube video about your site.

The header & logo

07 We're styling our header by giving it a 100% width with a height of 250px. Adding the logo is also going to be very simple as we'll only be using text for now. We've given ours a drop shadow using the 'text-shadow' property.

```
001 #header {
002         float: left;
003         width: 100%;
004         height: 250px;
005 }
006 #logo h1{
007         font-size: 35px;
008         float: left;
009         margin-top: 130px;
010         color: #dac91a;
011         text-shadow: 1px 1px 3px #333;
012 }
```

The navigation

08 Now let's style our navigation. We can do this by floating the whole element right, positioning it 150px down and then pulling it over to the right using a negative margin of '-50px'. Then we give the navigation a width of 600px. To bump everything to a horizontal position we float all the '<li>' left, then create our button separator by adding a 'border-right' to the 'navigation li a' selector.

```
001 .navigation {
002         float: right;
003         margin: 150px -50px 0 0;
004         width: 600px;
005 }
006 .navigation li {
007         float: left;
008         list-style: none;
009 }
010 .navigation li a {
011                         float: left;
012         color: #333;
013         padding: 6px 20px;
014         text-decoration: none;
015         border-right: 1px solid #333;
016         font-size: 16px;
017 }
```

Navigation hover state

09 At the moment our navigation looks nothing like a functional navigation menu and it still needs a couple of things added. The hover state will be the first thing we are going to add here, and then in the next step we will shoot over to Google fonts to download the 'lobster' font and use that.

```
001 .navigation li a:hover {
002
003         background-color: #333;
004         color: #fff;
005 }
```

Google fonts

10 Now let's go to google.com/webfonts and search for 'lobster'. Add it to your collection and click 'use'. Then add the '<link=' code to the index.html file just underneath your main CSS link. Then locate the '.navigation li a' rule and add in the 'font-family' property. Do the same for your '#logo' rule.

```
<!-- CSS -->
<link rel="stylesheet" href="styles.css">
<link href='http://fonts.googleapis.com/css?family=Lobster' rel='stylesheet'
type='text/css'>

.navigation li a {
        float: left;
        color: #333;
        padding: 6px 20px;
        text-decoration: none;
        border-right: 1px solid #333;
        font-size: 16px;
        font-family: 'Lobster', cursive;
}
```

"A sidebar can hold all sorts of content: extra navigation links, a search field, even some small images"

The search field

13 In this step let's style our search field. First of all we push it down slightly and then make sure anything underneath it is 50px away. Then we slightly push it away from the left edge using 20px. And lastly we give the input field some padding to make it a bit more attractive.

```
#search_form {

        margin: 10px 0 50px 20px;
}

        #search_form input{

                padding: 5px;
}
```

The sidebar

11 We want float the sidebar over to the right and set a fixed height and width. Then we can style the '<h3>' header tag and just push that off the left side using padding. Then give the text a dark grey colour of '#333'.

```
#sidebar {
        float: right;
        width: 300px;
        height: 100%;
        background: #fff; }
#sidebar h3 {
        padding-left: 20px;
        color: #333; }
```

Sidebar list

12 Now we are going to style the sidebar navigation list. We want to push it away from the edges by using a 20px margin all round. Then we add in a nice subtle dotted line using the 'border-bottom' property.

```
.sidebar_list {
    margin: 20px;  }
.sidebar_list li {
    list-style: none;
    margin: 10px;
    padding-bottom: 10px;
    border-bottom: 1px dotted #ddd;
        }
.sidebar_list li a{
    text-decoration: none;
    color: #333;
    font-size: 13px;
 }
```

Finishing off

14 No finish, we need to style our thumbnail images. So we float them left so they bump up to each other and then space them out by using margin.

```
.thumb {
    float: left;
    margin: 22px;
    border: 1px solid #fff;
    box-shadow: 0px 4px 6px #999;
}
```

Add content to your website

Learn how you can add content to the main area of your website

Adding content to your website is either done dynamically using a content management system (CMS) or by hand. Either way, you are going to have to learn how it's styled and what options you have. For instance, will you have just text or text with images? How about a video that's either embedded using YouTube's, or your own flash or HTML5 player?

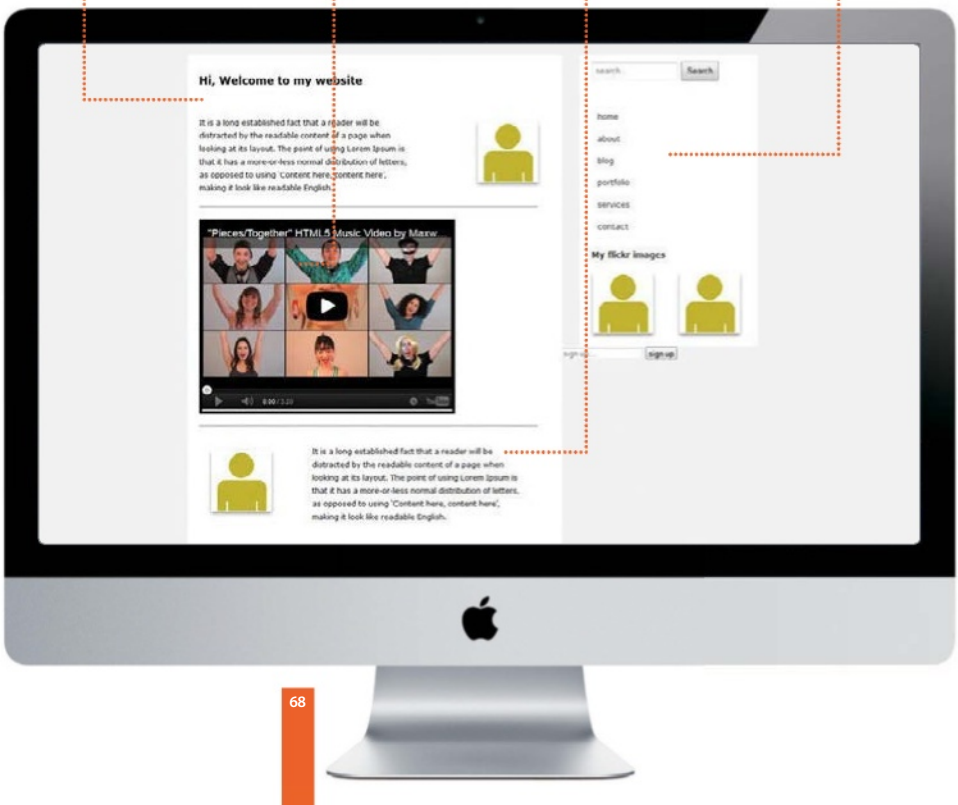
Of course your choices should be dependent on what niche your website caters for and thus knowing who will be visiting your website. So in this tutorial, we will add a few elements of content where one will include a video (which you can grab whatever video you want, you don't have to use our one) and also some text and an image. So open up the index.html and styles.css files within your text editor and let's get cracking!

The top section
It's a good idea to add some text about your website so users can see what you are all about before going any further

The middle section
We have added a video from YouTube here, but of course this could just as easily be a image of your company or yourself!

The bottom section
The bottom section is again used as a text-based section that describes the middle section video

The width
You will need to think about the width of your main content area(s) and plan ahead accordingly



Getting started

01 Now you have most of your template built from the previous tutorials in this book, you will be able to add this into the main_content id tag.

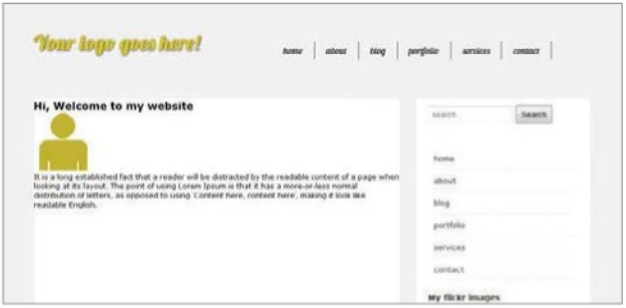
Welcome text

03 Now we need to add some 'welcome' text. This is normally a good way of letting the user know what your website is all about from the off. Here we will just wrap about three or four sentences within some '<p>' tags.

Main content header

02 Now in this step we are going to give our main content area a title that will welcome everyone to the website. We do this very easily by adding in a '<h2>' tag and a nice message within. What we will do is use CSS later to see what we can do to make this look a little nicer.

```
<h2>Hi, Welcome to my website</h2>
```



Adding an image

04 Now we can add in an image within our welcome text. The idea here is to have a small thumbnail image of something that relates to the website. So place your image tag '<img src=' above the welcome text just sitting underneath the header. Then we will use CSS later to float it to the right.

```
<img src="imgs/thumbnail.jpg" height="100" width="100" class="main_thumb">
```

Styling our content

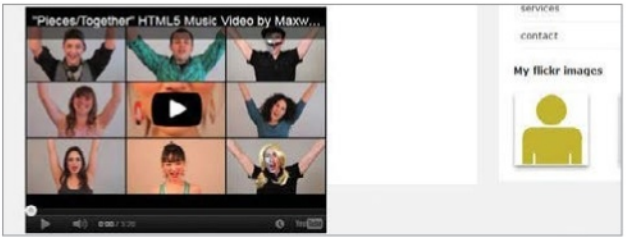
05 Let's open up our 'styles.css' file. We can point to the '#main_content' div and then hock onto the 'p' tag. We can then use multiple selectors here and use the same style properties and values for the<h2> tag.

```
#main_content p, h2 {
        line-height: 22px;
        margin-top: 10px;
        padding: 20px;
        width: 350px;
    }
```

Content image

06 Now let's float the image over to the right and add some styles to it. We are going to use the same CSS as we did on the sidebar thumbnails. Then we're going to make sure we have enough white space around the image.

```
.main_thumb {
    float: right;
    margin: 40px;
    border: 1px solid #fff;
    box-shadow: 0px 4px 6px #999;
}
```



Video

07 Next we want to add the video to our main content area. Now we can get really technical and use an HTML5 or flash player here, but to keep things really simple, we can shoot over to youtube.com and copy and paste the embed code within our 'index.html' page. Then we want to be able to shift this video about using CSS, so it would make sense to wrap a div around it with an ID of 'video'.

```
<div id="video">
<iframe width="420" height="315" src="http://www.youtube.com/embed/
kNn44gHqanM" frameborder="0" allowfullscreen></iframe>
</div>
```

Positioning the video

09 All we are going to do here is position the video using CSS. Now it makes sense to float the video left. Then we are going to use margin to push it away from the edges and anything that may be surrounding the video will now have a generous amount of white space. Then let's add a black 5px border to finish off our video.

```
#video {
        float: left;
        margin: 20px;
        border: 5px solid #000;
}
```

Styling the horizontal rule

11 Styling the horizontal rule '<hr>' is going to be a very simple task, but requires some thought. If you want to increase its height, you can't just call 'height: 10px' – we have to instead call upon its 'border' property 'border: 5px solid #ddd'. We are going to decrease its width, make it grey and give it space to the left:

```
hr {
color: #ddd;
width: 570px;
margin-left: 20px;
}
```

The main content height

08 Now first thing you will notice is that the video is now hanging out at the bottom of our content area. This is because on our '#main_content' rule we specified a fixed height of '500px;' So all we need to do now is locate that rule within our 'styles.css' file and change it from 'height: 500px;' to 'height: 100%';

```
#main_content{
        float: left;
        height: 100%;
        width: 630px;
        background: #fff;
        border-radius: 6px;
        margin-bottom: 40px;
}
```

Horizontal rule

10 What we are going to do now is add a separator just above the video. This will be easily styled using CSS, so we won't need to depend on some graphic design software such as Photoshop or Fireworks. It's simply an HTML tag called a horizontal rule '<hr>' and we can add that just above the video div within our 'index.html' file.

```
<hr>
<div id="video">
```

Adding more content

12 So now you should get the idea of how and what we can add for our main content. To get a real feel of it, let's add something underneath our video. What we can do then is add in another thumbnail image. So the best way would be to use the same image but give it a different class name of 'main_thumb2' and place this directly under our closing div of our 'video'.

```
<img src="imgs/thumbnail.jpg" height="100" width="100" class="main_thumb2">
```

More text

13 So the next step would be to add in some more text that could be used for a summary/excerpt of our video. Underneath our new thumbnail image, place some more dummy text that is wrapped within paragraph tags '<p>' with a class name of 'vid_excerpt' so we can style it using CSS.

```
<img src="imgs/thumbnail.jpg" height="100" width="100" class="main_thumb2">
<p class="vid_excerpt">It is a long established fact that a reader will be
distracted by  the readable content of a page when looking at its layout.
The point of using Lorem Ipsum is that it has a more-or-less normal
distribution of letters, as opposed to using 'Content here, content here',
making it look like readable English.
      </p>
```

The excerpt

15 Now all we need to do is float our video excerpt to the right and give it some margins with 50px to the right, and 40px to the bottom.

```
.vid_excerpt {
    float: right;
    margin: 0 50px 40px 0;
    }
```

Styling the content

14 What we need to do now is style our new content using some simple CSS. So, open up your 'styles.css' file and position our thumbnail image first. We will float it left and then give it a 40px margin all around. Then to finish it off we can give it the same subtle drop shadow as the other thumbnail we used in the sidebar.

```
.main_thumb2 {
    float: left;
    margin: 40px;
    border: 1px solid #fff;
    box-shadow: 0px 4px 6px #999;
}
```

Bottom content

16 When looking at our HTML, it would make more sense to wrap our bottom thumbnail and excerpt content within its own div. That way we can have more control of this section if at all we need to position it further. So, create a div with an ID of 'bottom_content' and make sure you use an HTML comment to mark the ending div ' <!-- END bottom content -->'

```
<div id="bottom_content">
</div><!-- END bottom content -->
```

Adding another divider

17 Now what we want to do is add in another horizontal rule '<hr>' above the excerpt text so we can then section out the video properly. So just above the thumbnail image link, place in your '<hr>' tag. But before we view it in the browser, we need to go back into our CSS and float the 'hr' tags left. This will push the bottom one underneath the video.

```
hr {
  color: #ddd;
  width: 570px;
  margin-left: 20px;
  float: left;
}
```

Going further

18 As we said in the intro text to this tutorial, it's down to you as to what your content should include. Later in this book we explore adding maps, newsletters, image sliders and more!

Add content to your footer

Finish off your first site with a footer to be proud of

In the past, the humble footer was not much more than a slim strip of colour with nothing more interesting than the copyright text written within. But in many modern websites, we have seen a huge improvement in footer designs and the content held within them.

These days, they are a lot higher and contain all sorts of content which can include contact forms, newsletter sign-up fields, social media integration and much more. Ours for instance, in addition to a newsletter sign-up field and generic About Us text, features another set of links for navigation, so our readers don't have to scroll back to the top or to the sidebar if they want visit another page.

As crazy as it sounds (and this would have sounded incredibly crazy no more than a few years ago), the footer has really taken the modern web by storm and has now developed into an area that needs good planning – like all parts of your website, it is important to offer the right amount of content without confusing the user. So in this tutorial we will build a simple footer and include some useful content that you would most likely see in a modern layout.

Getting started

01 Open up your 'index.html' file from the last project and scroll down to the very bottom of the document. The first we are going to do here is just underneath our closing 'main_content' div, add in a div with an ID of 'footer'.

```
001 <div id="footer">
002 </div><!-- END footer -->
```

About us

02 In this step we are going to add some 'about us' text that will be positioned to the left of our footer. So first thing to do would be to create a div with a class name of 'about_txt' within our footer ID making sure we add on a HTML comment to the closing div. For content, we will start with a '<h4>' header tag that includes the words 'about us'. Then underneath that we will add in some dummy text that is wrapped within paragraph tags '<p>'

```
001 <div class="about_txt">
</div><!-- END about txt -->
```

Footer links

03 It is always a good idea to place your site's navigation within the footer so the user doesn't need to scroll all the way up to the top to navigate your site. So let's use an un-ordered list and give it a class name of 'footer_links' and place it just underneath the about text.

```
001 <ul class="footer_links">
002   <li><a href="">home</a></li>
003   <li><a href="">about</a></li>
004   <li><a href="">blog</a></li>
005   <li><a href="">portfolio</a></
li>
006   <li><a href="">services</a></li>
007    <li><a href="">contact</a></li>
008</ul>
```

Newsletter sign up

04 Many websites would have either a contact form or newsletter sign up field within their footer. What we will do is keep it simple and add in a newsletter sign up field. We will give it an ID name of 'newsletter' and we will place this straight under the footer links. We can then position this later over to the right using CSS.

```
001 <form id="newsletter" action=""
method="get">
002<input id="search_term" type="text"
placeholder="newsletter sign up...." />
003<input class="submit_button"
004type="submit" value="sign up">
005        </form>
```

Footer background

05 Now we have finished our HTML mark-up, but if we viewed it in the browser now it would look a bit of a mess! So let's open up the 'styles.css' file, scroll down to the very bottom and start adding some CSS for our footer. First thing we can do is clear the floats so it pops underneath everything above and give it some dimensions and a background colour of white.

```
#footer {
  clear: both;
  height: 200px;
  width: 100%;
  background: #fff;
  margin-bottom: 20px;
  }
```

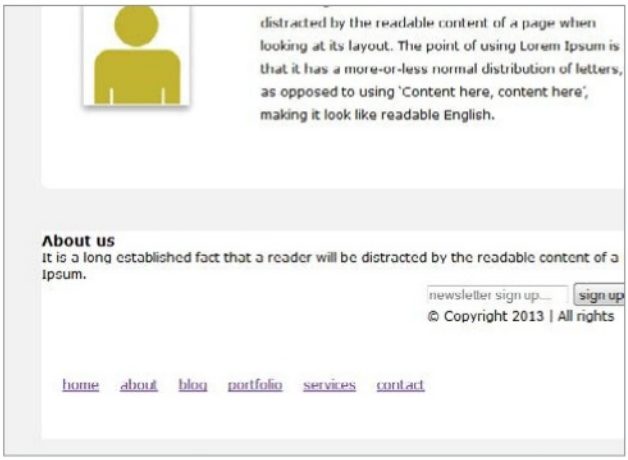
The links

06 Now in this step let's style our footer links so they look more like a horizontal navigation list. We are going to float it left and push it down using margin and give it 20px space at the bottom. Then let's give it a width of 370px.

```
.footer_links {
   float: left;
   margin: 90px 0 20px 0px;
   width: 370px;
  }
```

List items

07 Now style the list items'<li>'. Here we are removing the default bullet points and then floating all the items left, which will push each one up and next to each other giving us a horizontal alignment. Then using 'margin-left' we can create some space between each other.




```
.footer_links li {
      list-style: none;
      float: left;
      margin-left: 20px;
}
```

Finishing the navigation

08 Now we're going to finish off our footer navigation. What we need to do is remove the default underline and then give the text a light grey colour. Then for our hover state, let's just change the colour of the text to black '#000', giving us a subtle rollover effect.

```
.footer_links li a {
  text-decoration: none;
  color: #333;
  }
  .footer_links li a:hover  {
  color: #000;
  }
```

About text

09 Styling the about text is going to be very simple. We first float it left and then give it a width of 250px. We then give it some padding to allow some white space all around. One thing to remember by using padding is it will make the box element ('div') 20 pixels bigger as the padding effects the inner element, not the outer element, and pushes it out.

```
.about_txt {
  float: left;
  width: 250px;
  padding: 20px;
  }
```

"The footer has taken the web by storm and has now developed into an area that needs good planning"

Introduction to Responsive Web Design

Ultimate guide to discovering responsive web design and how to start using it today

Responsive web design

With the growing plethora of internet enabled devices it is important to ensure your website is responsive. Understanding responsive web design practices will help you build websites suitable for every screen size whether large, medium or small.

<above> More websites, including the popular Mashable news website, are becoming responsive

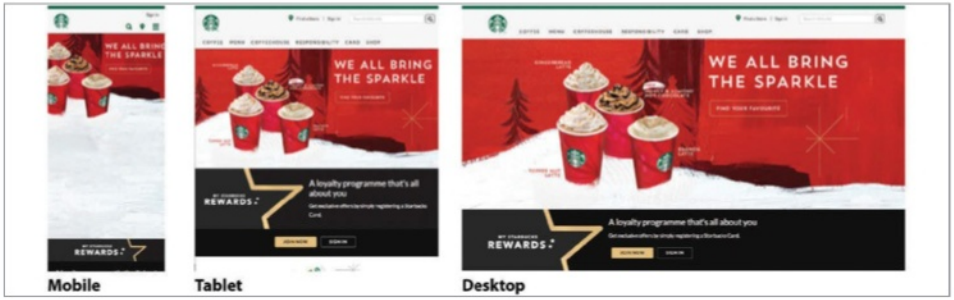
What is Responsive Web Design?

With the advent and increasing number of users browsing the web on mobile devices and with no evidence of slowing down the purpose of one design that fits all is more vital than ever before. Responsive web design is the answer to this problem. Responsive web design is the approach that allows websites to adapt to users' behaviour and environment based on screen size, platform and orientation.

Ethan Marcotte the man who coined Responsive Web Design (checkout his original seminal article on responsive web design at **alistapart.com/article/responsive-web-design**) discusses what responsive web design is. Its not a single piece of technology but rather a collection of ideas and principles and combination of techniques allowing your website to respond to the multitude of different screen sizes whether this be phones, tablet devices, desktop, game consoles, TV or even wearable.

<above> Ethan Marcotte's original article who coined responsive web design back in 2010

With new devices and screen sizes being developed everyday its important to ensure your website adapts to these changing screen sizes today and for the future.



Examples of Responsive Websites

Many modern websites you see today are actually in-fact responsive. Take the Microsoft website as an example (**www.microsoft.com**). When you resize this website in your browser window you will notice the website responds and adjust to accommodate the website content to fit in your browser window.

Another great example is the Starbucks website (**www.starbucks.com**). It magically resizes to fit comfortably in the new width of the web browser. Notice when viewing this on a mobile device the navigation collapses into drop down navigation activated by the burger menu icon. Orientation of the screen changes when you view this site portrait and via landscape.

Fluid Layouts

In the past we had liquid layouts, where only the layout columns structural elements and text were truly fluid which expands and contracts as the browser window size changes. But this was not truly flexible, images were not fluid which could easily break layouts. This included structural layouts which directly broke the layout if pushed enough.

This is where fluid grids excels where traditional liquid layouts failed. Fluid grids are a fundamental key principle of responsive web design. Rather than designing for rigid pixels or arbitrary percentage values, we can use grids to provide structure and use relative units to provide fluidity defined by using maximum width to carefully design in proportions.

When designing in traditional pixel based unit, there is a formula we can use to identify the percentage values, by using:

```
001  Target / Context = Result
```

To see this formula in action we have a HTML page with:

```
001 <div class="wrapper">
002   <section><!-- content --></section>
003   <aside><!-- content --></aside>
004 </div>
```

Quick Tip

Common responsive breakpoints you may want to use http://responsivedesign.is/develop/browser-feature-support/media-queries-for-common-device-breakpoints.

The CSS:

```
001 .wrapper {width: 960px;}
002 section,
003 aside {margin: 20px;}
004 section {float: left; width: 640px;}
005 aside {float: right; width: 300px;}
```

To convert the fixed units and turn them relative we can use the flexible grid formula:

```
001 section,
002 aside {margin: 2.083333333%; /* 20 / 960 = 0.02083333333  */ }
003 section {float: left; width: 66.666666666667%; /* 640 / 960 = 0.66666666666667
*/}
004 aside {float: right; width: 31.25%; /* 300 / 960 = 0.3125 */ }
```

We set our context as our wrapper which is 960 and the target value is either our section, aside and margin. Using this formula we can apply this throughout our grid to build a dynamic website to scale every viewport size.

Designing Responsive Websites

When designing any website, planning and setting out user goals is essential to crafting the best possible experience; users need to be able to interact with your website with minimal thinking.

Its beneficial to organise the site structure and mapping out the architecture and possible framework that dictates the different break points the site will respond to. Its good to wireframe and map out your website using traditional tools simply with paper and pen. Or use helpful digital software such as the online wireframing tool Balsmiq (**balsamiq.com**).

After mocking up your website, its not necessary to design pixel perfect desktop designs. Because it will only be part of one static snapshot of a layout which is in constant fluid motion. Nor does it make sense to design every single screen imaginable.

When designing a responsive website it is important to use a grid system. Typically grids are measured in pixels but now each grid will become a percentage of a page width. Be aware when the browser window size changes the column width will too, this will vary between resolutions.

Quick Tip

Use Responsify to generate your responsive template including grids columns, gutter width and breakpoints **http://app.responsify.it/**

Setup grids and breakpoints

1 Calculate grids and breakpoints
Head over to **http://gridpak.com/** to setup the number of columns and breakpoints.

2 Download
Click download your Gridpack to download the PNG for design purposes and CSS. Give credit for your resources on your site.

Modular design

To make your designs more fluid and ensuring it adapts to whatever environment it is viewed on even on a narrow screens such as mobile devices you will need to think of each element of your webpage to be modular.

This includes breaking down the page into components for example the header, navigation, main content, sidebar and footer. Imagine this as a puzzle that can be reassembled or re-organised to fit on smaller screen size.

Its important to keep the information architecture coherent for example the links need to be grouped together in a navigation and that content should still be linear and easy to navigate.

Navigation
Navigation changes to a mobile friendly version when the width reduces from 539px using media queries

Search
The search is hidden on mobile devices and replaced with a glyphicon to activate it

Responsive Images
All images are set to max-width: 100% to ensure they scale when the screen size changes

Flexible grids
Components are neatly structured using flexible grids so they resize in proportion on smaller screen

Setting up breakpoints with media queries

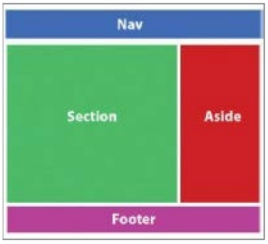
1 Setting up mobile breakpoints This will be set for screen width of 320px and max width of 480px.

2 Setting up tablet breakpoints This will be set for screen width of 768px up to 1024px.

3 Setting up desktop breakpoints This will be set for a minimum screen size of 1224px .

4 Adding your custom styles You can now apply specific styles for specific devices or screen widths.

Media queries and flexible layouts



1 Style mobile screen To resize website to fit on less space on a mobile screen.

2 Add in media query Add a media query for screens of a maximum of 320 pixels.

```
@media screen and (max-width: 320px) { }
```

3 Remove floats Within media query set width to auto on the section and aside.

```
  section, aside {
    float: none;
    width: auto;
  }
```

4 Widths auto span The widths on the elements now auto expand across the viewport on mobile devices.

Media Queries

Once the design of your website becomes more complex for example spanning across three or more columns. This becomes a problem as the screen width becomes too narrow for mobile users. This is where CSS3 media queries comes into play. Media queries allows you to specify CSS styles to be applied once conditions are met such as when a browser reaches a specific width.
There are a few methods to incorporate media queries on your website. First method is by linking this within the HTML <head>:

```
<link href="responsive.css" rel="stylesheet" media="all and (max-width: 960px)">
```

Or using the @import rule:

```
@url(responsive.css) all and (max-width: 960px) {  }
```

Most common practice and to avoid further HTTP requests is to just use @media within your existing style sheet:

```
@media screen and (max-width: 960px) {  }
```

We can set the media type to either all, screen, print, tv and even braille. Each media query includes a media type followed by one or more expression. Logical operators in media queries help build powerful expressions. We can use the 'and' operator to add extra conditions:

```
@media all and (min-width: 600px) and (max-width: 940px) {...}
```

The above selects all media types between 600px and up to 940px.

Besides setting the min- and max-width we can also use the orientation media feature to determine if a device in landscape or portrait mode.

Media support for older browsers

1 Media queries support for older browsers
Download Respond.js over at **github.com/scottjehl/Respond**

2 Link to JavaScript
Link to respond.js in your <head> right after where you have put your style sheets.

```
@media screen and (orientation: landscape) {...}
```

Unfortunately older browsers such as Internet Explorer do not quite understand media queries. To combat this we can simply add the 'only' keyword.

```
@media only screen and (min-width: 600px)
```

User agents using HTML4 algorithm will now simply ignore media queries that you have inputted.

Viewport

If you built a responsive website and try to view it on a mobile device you will notice the website has scaled to the desktop size. By adding a viewport meta tag we can ensure browsers scale the webpage to fit the screen size its on, with:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Fluid images

Most websites contain some sort of media typically imagery. With a responsive website content is expected to fluidly change to the screen resolution, this implies images will need to as well. Luckily there is a easy solution to scale images when the viewport changes. In the CSS simply add:

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

Clearfix
Using the clearfix hack to automatically clear after itself so it won't be necessary to add additional markup

Media query
Percentage grids are placed in a media query based on screens with a minimum width of 450px

Zoom: 1 fix
Applying the zoom: 1 to fix old browsers such as Internet Explorer to fix the hasLayout bug

Calculated grids
This grid structure has 12 columns which are broken down into calculated percentage widths

Retina Images

1 **Serve high-res images to retina displays** Download Retina.js at **http://imulus.github.io/retinajs/**.

2 **Copy retina.js to your scripts folder** Copy the plugin to your JavaScript folder.

3 **Add script to <head>** In the <head> add `<script type="text/javascript" src="js/retina.js"></script>`

4 **Reference retina images** For retina images save them as image@2x.png

Quick Tip

Its a good idea to add height: auto; when using responsive images in case the height does not resize in proportion with the width.

Compressing CSS

1 Compress CSS
To reduce file sizes and the amount of HTTP requests head over to **http://www.shrinker.ch/**

2 Upload files
Upload all your CSS files and click shrink

3 Copy shrinked file
Copy the shrinked code you have created into a new CSS file called compressed.css

4 Update CSS link
In the <head> make sure you update the CSS link to the new compressed.css

Quick Tip

Download GuideGuide Photoshop plugin (**http://guideguide.me/**) to set up your own guides in Photoshop, with many variants and ease.

Applying this fix will allow images to scale down dependent on the containers width.

Mobile first

When designing for responsive websites its a good rule to follow the mobile first principle.

Mobile first is an approach where you design for the mobile experience first and then work your way upwards towards tablets and desktops. This forces us to focus on the advocate constrains of available space on mobile screens that do not have enough real estate for ads, social media buttons etc. It would be wise to use media queries to limit unused styles and scripts should not be loaded for mobile devices including media assets.

Browser testing

Its fundamentally important to always test your website in as many web browsers, operating systems and devices as possible. With the growing number of devices and screen resolution its very hard to have an extension assets of hardware available for thorough testing.

There are some free online tools available to help. There is the Responsive Web Design Testing Tool by Matt Kersley (**mattkersley.com/responsive/)** and Responsive Test (**responsivetest.net)** both allow you to enter in a URL to preview your website will look at different resolution and devices.

There are some more powerful paid for tools to help give a better preview and interaction on real world web browsers and devices. Spoon.net (**https://spoon.net/browsers**) has a collection of web browsers you can run locally.

Cross Browser Testing (**crossbrowsertesting.com/**) and Browser Stack (**www.browserstack.com/**) are very similar whereby they allow you to run live emulators from legacy web browsers to actual working iOS, Android and Windows mobile devices including tablet devices for you testing pleasure.

Download Foundation Framework

1 Head to Download
Go to **foundation.zurb.com/develop/download.html** You may customise the components and grid system

2 Click Custom Build
Unzip the downloaded pack and get started building your responsive website, it's fairly straightforward from there.

Responsive Frameworks

Building a new responsive website can be daunting, with the technically issues surrounding responsive web design, calculating number of columns, grids, percentage widths, breakpoints and further technical hurdles that may appear through development. A solution is to adopt an existing responsive framework.

There is a huge benefit when relying upon a responsive frame to begin it won't be necessary for you to calculate your own fluid grids instead most modern frameworks will allow you to get going and starting building your website. Essentially picking up a framework will help save tremendous amount of time and generally they come packed with predefined styles, components, font sizes, form elements, CSS resets and more.

Popular frameworks will have an active community and even extend the framework further with new and robust plugins. There will traditionally come with clear documentation for you get started when using the framework and understanding all the code and components available.

Two very popular and well maintained responsive grid frameworks available are Bootstrap and Foundation.

Bootstrap (getbootstrap.com) originally created by the team from Twitter, this framework is one of the most popular open source responsive framework out there. This library consist of a huge amount of components and re-usable functionality such as drop downs, glyphicons, styled form elements, progress bars, JavaScript powered plugins including carousels, alters, tooltips, transitions and many more.

Zurb Foundation (foundation.zurb.com/) is one of the most advanced responsive front-end framework. It was built with speed in mind so all the components and JavaScript plugins have been performance tuned. Foundation as the name suggests is used as a starter for any responsive website, it contains a responsive grid system, mobile friendly navigation, customisable components, some handy media plugins including flexible videos, lightbox and carousel slider.

Which ever framework you choose always make sure you carefully read the documentation and make sure it suits the needs of your website.

Futher reading

There are a lot more reading to further your discovery on responsive web design and to continue on improving your skills.

Websites that contain great resources on responsive web design include This is Responsive **http://bradfrost.github.io/this-is-responsive/** a collection of news and resources. A beginners guide to responsive web design resource list **www.targetlocal.co.uk/responsive-web-design-resources/**.

Books that help further your skills include HTML5 for Web Designers h**ttp://www.abookapart.com/products/html5-for-web-designers** and CSS3 for Web Designers **www.abookapart.com/products/css3-for-web-designers** and Responsive Web Design **www.abookapart.com/products/responsive-web-design** all are by A Book Apart.

Checkout some great examples of the best responsive websites out there on Awwwards **www.awwwards.com/websites/responsive-design/** and more to see over on Get with the Future **socialdriver.com/2014/03/05/65-best-responsive-website-design-examples-2014/.**

Bootstrap

Responsive design made easy

Bootstrap is a powerful yet approachable front-end framework designed to make it quicker and easier to develop user-friendly websites

It was in August 2011 that Mark Otto, a developer at Twitter, announced a new framework that the social media platform had developed internally, with the aim of providing a consistent API while developing the popular micro-blogging tool.

Bootstrap was designed to help Twitter's developers avoid the mish-mash of competing and incompatible frameworks and libraries. It provides a unified toolset for all front-end development requirements including advanced typographical control, a built-in grid system, and the option to use components to extend functionality to suit any particular needs that go beyond the basic framework common to every website. Twitter have since produced one of their most compatible and functional websites to date, with the sleek new design not only being pleasing on the eye, but providing its users with a basis of comparison to the other social networks.

Fundamentally Bootstrap is pure CSS, but as it's built with LESS (a pre-processor that offers flexibility beyond what normal CSS can achieve) and has Sass support, there is a range of additional benefits over and above plain styles.

The core CSS is broken into components, allowing it to be easily managed and adapted. Each file in the set of Less source files deals with one particular characteristic, so you have a type source file, a forms source file and a tables source file.

Bootstrap is designed to be mobile-first, meaning its fully responsive which includes a powerful four tiered grid system so your layouts can adapt to phones, tablet, desktops and large desktops screens. Given that much of our internet usage is moving onto smaller devices, this seems to be a sensible move.

On top of the CSS framework, a series of jQuery plug- ins provide functionality such as support for drop-down menus, modal boxes and much more.

Taken together, these different elements provide the basis of a plug-and-play, standards-compliant and user- friendly HTML5 website. The idea is to facilitate rapid application development without having to build your own solution each and every time you start a project.

We'll take you through the Bootstrap framework, explaining how to use it and why it will revolutionise the way you create websites. We'll show you some excellent examples of Bootstrap-powered websites, and point you in the direction of useful additional resources that will help you take full advantage of all the possibilities on offer. It's particularly useful to understand why the framework is structured the way it is, so let's start there.

Bootstrap: What and why

To get the most from Bootstrap, you need to understand how it's structured, and why

There may be an element of confusion as to why there are so many individual files that make up the library. The basic principle behind segmentation of the different component parts of the library is that each can be updated without adversely affecting the others, making the system more manageable. The benefit of using the LESS system is that you can combine all your CSS into a single file when it comes to pushing your code live.

Looking at the Less source files (a CSS preprocessor Bootstrap was built with), lets focus on seven key files that made up part of the uncompiled CSS. Since there are a dozens more. The original core functionality remains the basis of Bootstrap, so let's have a quick look at what it contains:

Reset Based on Normalize.css reset rules, this removes all the browser formatting defaults, giving you a completely clean slate to work with.

Variables A set of globally shared and commonly used values including colour values, typography, scaffolding, iconography, components, spacing, tables, buttons, media queries breakpoints and more. .

Mixins Mixins are useful way to to mix-in a bunch of properties from one rule set into another. Bootstrap has a collection of imported mixins to be used, which includes a vendor-prefix which provides all browser vendor prefixes for you in your final compiled CSS.

Scaffolding The other reset styles that normalize.css does not include such as box-sizing, body reset and form elements. As well as basic page templates and structure for your site.

Type Styles for all the typographical aspects of your design.

Forms Styles that deal with formatting all things related to your forms.

Tables A set of styles that provide presentation for tabular data. There are a number of jQuery plug-ins you can use to add specific functionality to your design. Note that although the framework is structured in this way, if you simply download the latest release you'll get pre-compiled and minimised code, so you don't need to worry about using the Less.js compiler.

Get started

Bootstrap works by providing a clean, uniform approach to creating the most common user interfaces on the web. Whether you're creating a simple brochure website, or a complex form, the same core components make up the site design in terms of styles and content structure. By using Bootstrap you can save time, pulling in ready-made controls, layout elements and styles. This leaves more time to work on aesthetics and testing.

The first thing you'll need to get started with Bootstrap is the framework itself. Visit getbootstrap.com to download the latest version of Bootstrap. All the media queries, grid sizing are all ready and setup included in this download.

Once you've downloaded the framework, you simply need to plug in the CSS and JavaScript files to create your first Bootstrap page.

The grid

A fundamental aspect of Bootstrap is the 100% fluid grid system built into the framework. There are four grid options available, but the principle approach to working with a grid remains the same. The responsive grid scales up to a maximum of 12 columns as the device or viewport increases. You can define how many columns each box should span, allowing you to simply and effectively align, without having to calculate widths manually.

Benefits

We take you through just a few of the benefits of Bootstrap

Modular
The Bootstrap framework is modular, so you only need to call in the elements you need for your specific project, making it relatively lightweight to deploy.

Responsive
The grid system is fully responsive, allowing you to design once and deploy your website across multiple device profiles, all without having to do any complex mathematics or calculations.

Quick to develop
Bootstrap is built on standard CSS and JavaScript making it very quick to develop the accompanying HTML, and easy to get clean cross-browser results without having to start from the ground up each time.

Extensible
There are over a dozen jQuery plug-ins designed specifically to work with Bootstrap, providing out-the-box functionality for the likes of carousels, tooltips, dropdowns and transitions. It is also easy to develop your own plugins.

Pre-built components
Bootstrap includes a series of user-interface components that you can instantly deploy, including drop-down navigation bars, auto-pagination, media objects, progress bars and many more.

Why work with templates

A brief outline as to how templates can help you with developing your design

Although Bootstrap is a pure CSS and JavaScript library built with Less.js, there's a flourishing community of templates. These HTML packs often include images and specific CSS additions to give you a head start with regards to developing your design.

Bootstrap itself provides a number of templates in the Get Started section of the **getbootstrap.com** website, under Examples, demonstrating a number of different components and plug-ins in action. These are barebones pages, only designed to provide the structural elements of a typical layout, and covering common design patterns such as a blog, dashboard, login form or full-width image carousel. It's well worth having a look at these when you're first starting out with Bootstrap, if for no other reason than to understand how useful the framework is in rapidly developing common functional and structural requirements.

Once you move outside the official Bootstrap website, however, there is a wide range of free and commercial templates available that go far beyond merely providing the structural framework, and move towards a fully-realised design that you can deploy as-is if desired. Some of these templates handle a specific area of functionality, such as an admin system, basic eCommerce website, or photographer's portfolio, while others take a more generic approach. Common to almost all templates is that you'll need to wire in your own functionality to handle any server-side processing required, but given a fully-realised front-end design, it becomes a much quicker process to develop prototype web apps if you use a template to provide you with a flying start.

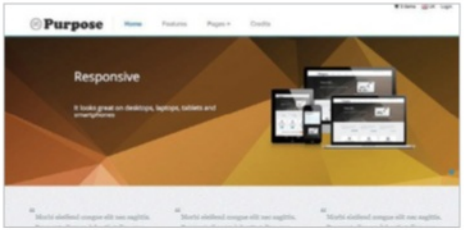
When you're looking for templates or themes, be sure to check that they're using the standard Bootstrap source code as a basis, and that ideally their additional CSS design elements are incorporated in a separate file, or as a fork from the main Bootstrap branch. This will allow you to (relatively) safely apply any subsequent Bootstrap updates without having to spend a lot of time reverse engineering the changes.

Custom download

Get a custom download of Bootstrap by visiting getbootstrap.com and clicking on Customise. You can then specify which components, variables, plug-ins and base-styles are included. The download includes minimised files for low-bandwidth deployment to production sites, and uncompressed versions for testing.

Six of the best

We give you a quick rundown of our six favourite templates and some of the features they offer designers

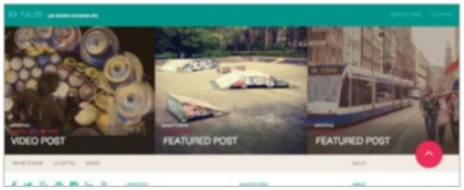


1. mPurpose

This free theme from **http://bootstrapzero.com/bootstrap-template/mpurpose** is a fully responsive multipurpose template. It includes over 30 sample pages from portfolio layouts, business/agency layouts to carousel slider, eCommerce shopping layouts and more. This is ideal for a web design agency site.

4. JA Appolio

JA Appolio is a Joomla theme built on Bootstrap, available free from **http://www.joomlart.com/joomla/templates/ja-appolio**. A desirable portfolio template, it bundles with amazing features including parallax design, blog ready, a selection of 5 colour schemes and 9 bonus pages.

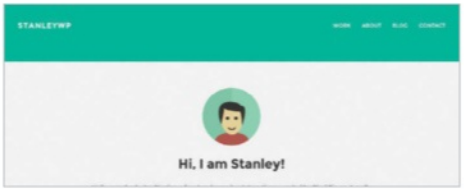


2. Fullby

Free from **http://www.marchettidesign.net/fullby/**, Fullby is a Wordpress theme using Bootstrap. It features a responsive video slider, animated shuffling of blog posts, animated widgets, and wide selection of colours to choose from.

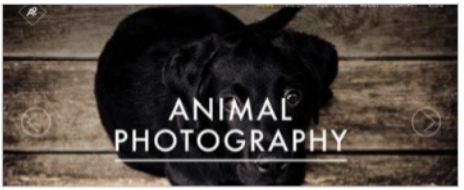
5. DashGum

Also free **http://bootstrapzero.com/bootstrap-template/dashgum** a 15 page dashboard panel or admin theme, this boost some unique interactive components including charts, tables, calendar notifications, to-do lists and more.



3. StanleyWP

This theme, by **http://gentsthemes.com/themes/stanleywp-twitter-bootstrap-wordpress-theme/** is also free. This Bootstrap template demonstrates a truly flat style looking personal portfolio can be so simplistic, ideal for anyone running their own portfolio and/or blog.

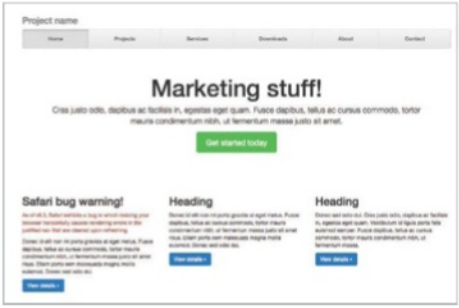


6. Awesome Photography

This premium theme **https://wrapbootstrap.com/theme/awesome-photography-template-WB04B42CG** is perfect for any photographers. This template deals with heavy imagery with some interactive animated elements to allow you to present your art beautifully and clean.

Using components for your user interface

Bootstrap goes beyond merely providing a responsive grid system: it's a fully featured user-facing toolkit that includes a whole set of user interface elements, known as components, that offer simple, speedy integration



Among the key features of the Bootstrap framework is its concept of components. These are re-usable user interface controls that cover the most common interactions, and range from drop-down menu systems all the way to modal pop-ups.

Components are implemented using a combination of styles and scripts, and are typically initiated simply by applying a specific set of classes to your markup. The documentation (available from **getbootstrap.com/components**) provides some great examples for each of the components that ship as part of the core Bootstrap build, but each of these is just the starting point. The idea is that you will dress the component to suit your own website's layout and aesthetic treatment, altering or extending functionality as you require, to provide a perfect fit for all of your needs.

Better still, because the component system is entirely modular, you're not limited in any way to the twenty or so that come in the box; there are a raft of additional components available on the web that extend the basic functionality still further. Some of these are not much more than a jQuery plug-in, but tend to be designed around the Bootstrap ecosystem, offering similar syntax and integration to the standard set.

Although it's often tempting to build your own solution to a particular user interaction requirement, by using off-the-shelf components you can focus your time on enhancing the all-important user experience, and rapidly prototype your ideas without impacting negatively on time or cost.

It's also extremely convenient to use some of the more basic components, such as list groups, panels and wells, as these inherit the global style definitions automatically. This means you can easily control the look and feel of common content containers without having to explicitly define each one individually.

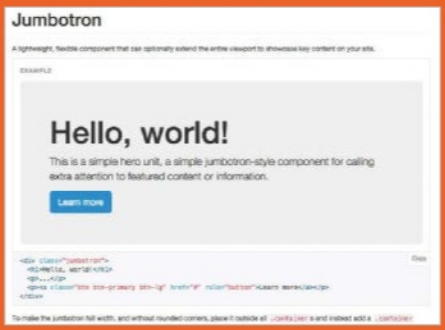
Naturally, you can create your own additional components that you'll be able to subsequently re-use in future projects. The Bootstrap documentation walks you through the process of integration, or you can use a third-party service such as Jetstrap (**jetstrap.com/**) to build your own custom user experience elements and designs.

Customisation

Components are little more than HTML markup classified using a combination of structure and classes. As a result, they're incredibly flexible and infinitely customisable. If you want to change the look or the way to interact, you simply need to edit the associated CSS and/or JavaScript.

5 Popular UI Components

A collection of elements that can be instantly added to a site.



1. Jumbotron

This component encapsulates a popular layout technique: an oversized typographical box. Typically such an area includes a call to action, so the Bootstrap Jumbotron has a button built in to the design – which is, of course, easily customised.

2. Tabs and Pills

A popular navigational device, the tab allows you to show lots of content in a small space. The pill can be used to work similarly or as an intra-page navigation system. The default styling is configurable, so you'll see a lot of these dressed to the suit the site in question.

3. Alerts

Useful when you're trying to draw your user's attention to important information – either in response to their input, or as a consequence of something new or time-limited – alerts are an extremely common call-out approach to highlighting feedback. Bootstrap provides four styles, including a dismissible alert.

4. Input groups

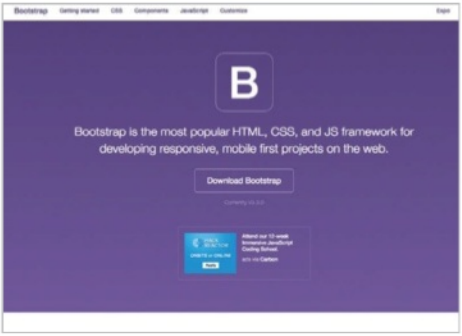
Input groups make it easy to develop clean, validated user forms that respond consistently. The core styling is simple and follows the current trend for minimalist aesthetics, you can also append or prepend text or buttons either side of text based inputs.

5. Badges

A great way to convey to users that there are new items to read, actions to take or feedback to collect. The Bootstrap implementation is clean, allowing you to avoid extraneous markup, while rendering in a familiar, user-friendly bubble.

Build your first Bootstrap webpage

Bootstrap is designed to make developing a fully functioning user interface simple – we'll show you just how quick and easy it is to get started

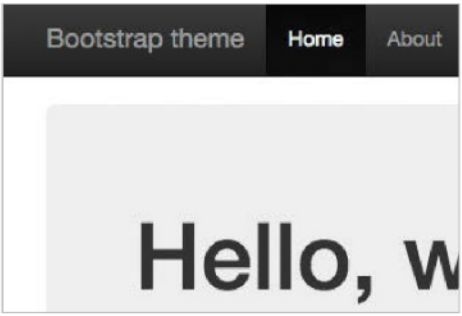


Download the framework

01 First of all, be sure to visit **getbootstrap.com** and click on the Download Bootstrap button. Grab the standard download, and unzip the contents to your website folder ready for use.

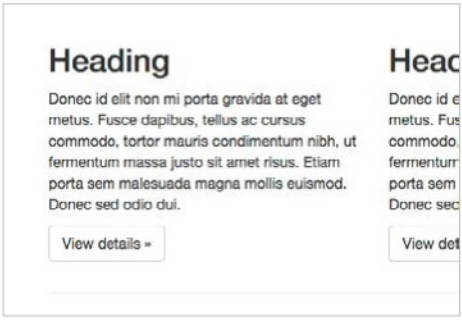
Basic HTML

02 Bootstrap plugs straight in to your regular HTML page, so start by creating a simple blank page that includes the jQuery library. Add a link to the minimised version of the Bootstrap CSS file, and add the minimised script. Best practice says we should put our scripts at the end of our HTML to speed up loading for users.



Add a nav bar

03 You can easily add a navigation bar to your page by inserting an unordered list, each containing a list item. Give the <ul> a class of 'navbar'. This hierarchy of markup defines the way the navigation will be presented, including all hover states and effects.

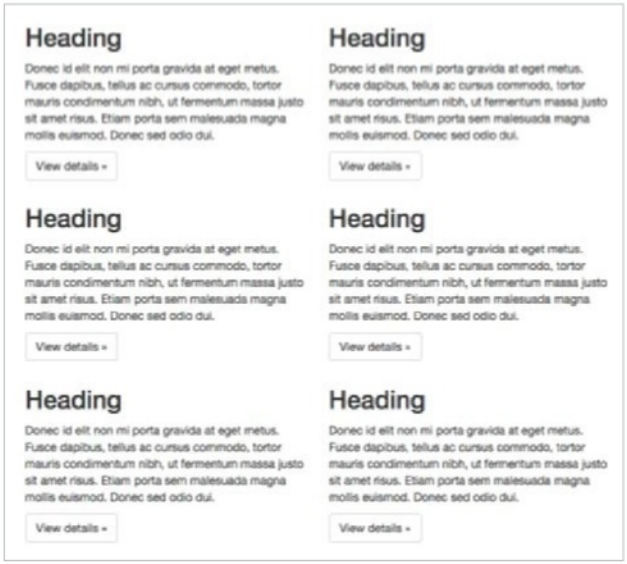


Add grid-aware containers

04 Bootstrap uses classes to define how different elements should be styled, including the way they fit into the grid system. Add three <divs> to your page and give each a class of 'col-md-4'. This aligns them to the four-column grid.

Add a Jumbotron

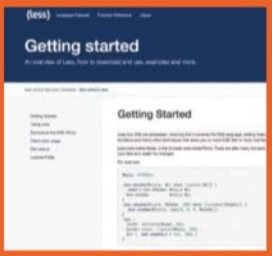
05 A popular way to draw attention to your key message is by using a Jumbotron – a large heading and first paragraph that clearly spells out what your website is all about. Add the code from Bootstrap component section **http://getbootstrap.com/components/#jumbotron** to add a Jumbotron above your three four-column boxes.



Make it responsive

06 Bootstrap is fully responsive from the beginning, your webpage will be instantly responsive. Try expanding or collapsing your web browser to see the responsiveness nature in action. This webpage is now mobile, tablet and desktop friendly.

A little bit LESS.js



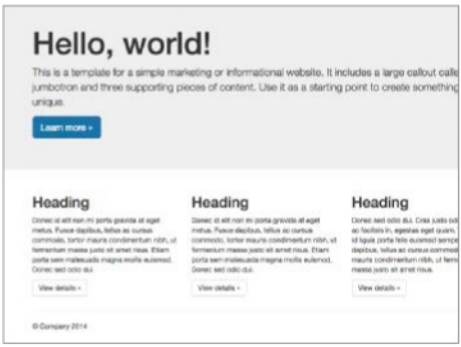
CSS is a fantastically easy language to learn, and it increasingly offers a wide range of functionality as browser vendors have embraced CSS3 in earnest. The biggest problem with CSS is that it's static in nature – inevitably you end up writing the same styles over and over again, with no way to write your colour choice once as a variable, and re-use the variable across the stylesheet. There have been some efforts to include CSS variables natively in the browser, but they're not very well supported, and are quite limited in their capabilities.

LESS offers an easy-to-understand solution. The system extends CSS with variables, operators and functions. What this means in practice is that you can define your core properties as variables and re-use the variables – this is as opposed to using the explicit values they contain, making it quicker and easier to change styles globally.

LESS offers a lot more than this, but does require compilation. You can get hold of all the full details at the **lesscss.org** website.

Create a carousel with Bootstrap

We get in depth with one of the most visual components within Bootstrap: the carousel. Read on to find out how to customise the carousel to fit your own design

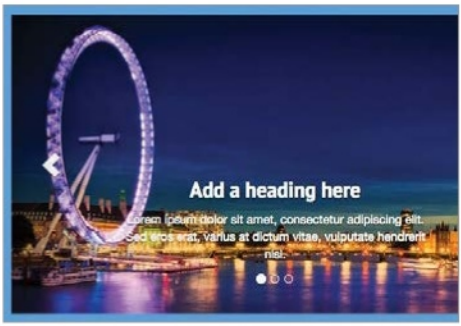


Grab a template

01 The easiest way to start this project is to grab one of the ready-made example templates on the **getbootstrap.com** website. We've opted for the Jumbotron template because we can replace the Jumbotron with our carousel. Copy the HTML source code, and paste it into a new page to get started.

Replace the hero

02 We're going to replace the Jumbotron hero panel with a carousel. Visit the documentation for the Carousel plug-in, found within the JavaScript section of the Bootstrap website, and copy the code example. We'll paste this in place of the Jumbotron in our template.



Build the HTML

03 Paste in the code on your clipboard and have a quick look at how it's structured. You'll notice that there are two principal sections: the carousel-inner, which contains each slide in the carousel, and the carousel-indicators that provide the dots demonstrating the currently visible slide.

Add user controls

04 Also within the sample code is a pair of navigational elements, both rendered as anchor tags within the HTML. These are used to page through the individual carousel slides. The classes and data-side properties are extremely important here, as they provide the hook to the carousel itself.

Responsive carousel

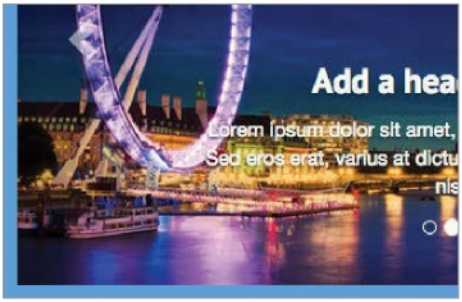
The carousel slideshow component comes fully responsive be sure that your images are wide enough to encompass all common screen widths. This can create a big impact, especially on portfolio websites where images are used to show off your work.

Tweak the behaviour

05 You'll notice that by default the carousel will play, changing the slide every 5000 milliseconds (which is the same as 5 seconds). You can update this value by passing an argument to the .carousel() call. Pass in a value assigned to the interval property, and you can reduce or increase the time to suit your needs.

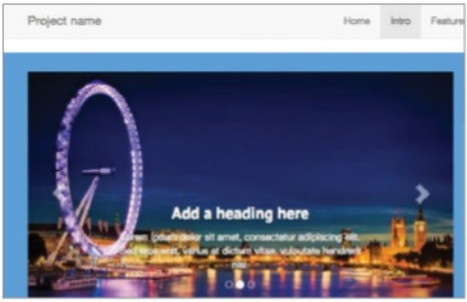
Change the settings

06 Refer to the documentation to view the additional methods and properties you can set to alter the settings for the carousel. A good example of a change you can instigate is to add a button that calls the pause method, preventing the carousel from automatically moving to the next slide.



Customise the look

07 As with every other plug-in and component within the Bootstrap framework, you can fully adjust the look and feel to suit your own website. Simply edit the CSS styles for the different elements with a class that begins .carousel – so to change the background colour of the caption, edit .carousel-caption, for example.



Plug in your content

08 Finally, all you need to do is plug in your own images and captions, including any links you'd like to include, and test your page. Keep in mind that all the different elements of the stock carousel can be customised, so you're not stuck with the default user controls, caption position or carousel size. Have fun!

Resources

Bootstrap has a lot of resources available on the web, check out five of our favourites below

Bootstrap
getbootstrap.com
Straight from the source, the Bootstrap website provides an off-the-shelf download of the framework, a customisable download builder, and a full set of documentation for the framework covering all the standard plug-ins, components as well as the grid system. This is also the place to get the latest official news

Bootstrap Google+
https://plus.google.com/communities/110370020620457615447
This is the Google+ community group for Bootstrap, visit here if you'd like to seek help from other users, or to learn about the latest news. This group currently boast over 4000 active members. There are regular updates when new templates, components and plugins are launched.

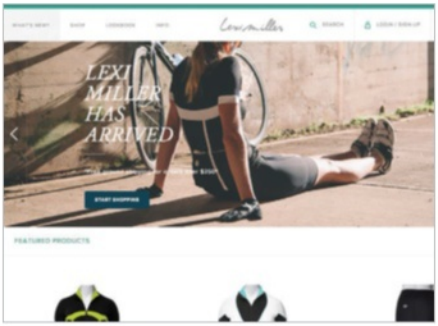
Bootsnipp
bootsnipp.com
A handy resource site that gathers together HTML snippets, themes, components and plug-ins in a gallery that's easy to browse. Users can contribute snippets, sharing their own work with the community for mutual benefit, and the site also features a handy page of links to other Bootstrap resources for quick reference

Start Bootstrap
http://startbootstrap.com/bootstrap-resources/
Kickstrap calls itself a 'complete kit for making websites', building on top of Bootstrap with additional support for libraries such as Raphael without the pain of manually bringing together disparate resources. A useful resource for sites that need to go further than basic Bootstrap

Bootswatch
bootswatch.com
Bootswatch is a simple theme site for Bootstrap, but rather than focus on the complete end product, it provides a core set of styles that integrate beautifully with the basic Bootstrap components to create markedly different aesthetic treatments. It's limited in the range of themes it offers, but many are worthy of forming the basis for your own customisations.

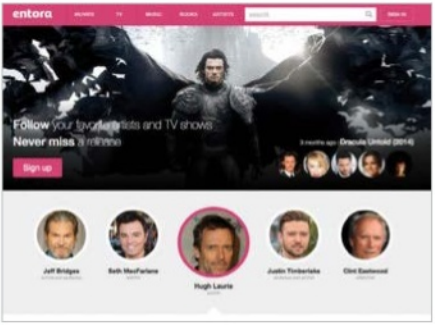
Be inspired

Bootstrap is the underlying framework for some amazing websites – take a look at two of the best to get some inspiration for your next project



Lexi Miller
http://leximiller.com/

This clean, simplistic and gorgeous looking feminine website sells fashionable athletic clothing. It takes advantage of the responsive native in the Bootstrap framework and utilises it as its own.

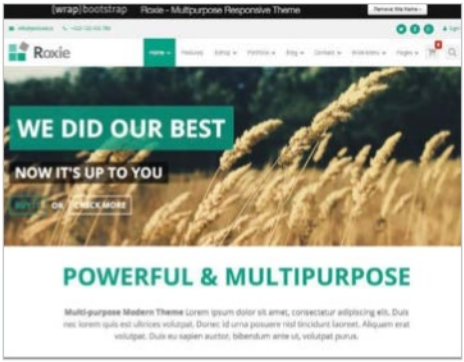


Entora
http://entora.com/

A beautiful entertainment source to follow your favour artists. This website takes advantage of the mobile first fluid grid system using the extra small grid option to force grids to be horizontal at all times.

What's the next step?

So you're all set up with the basics, but what happens now? We give some guidance as to where to go from here with Bootstrap, including how to approach customisation



Now that you've got an overview of how Bootstrap works, what makes up the library, and how to leverage the framework to build a website, you're ready to start building your sites using Bootstrap.

Your first step should always be to create a customised download, rather than relying on the kitchen-sink approach of downloading everything. This will make your site leaner and more bandwidth-friendly, which your users will thank you for. Once you've got the framework, build your own theme – working on top of an existing template or theme if you'd like a quick start – and develop your content in situ.

Bootstrap is particularly powerful for rapidly prototyping user interfaces, making it an ideal solution for generating style tiles or non-functioning mockups when presenting work to clients. Keep in mind that your prototypes can go on to form the basis of the final product too, so you're saving time upfront as well as in the website or application build phase.

Once you're confident, you may want to develop custom functionality or components that simply aren't covered by the default install of Bootstrap, or that work differently to the standard behaviour. You don't need a huge amount of technical knowledge or skill to be able to generate your own assets that you can go on to reuse in future projects. Similarly, if you're a front-end developer you may find that you want to develop custom widgets and plug-ins that leverage jQuery. This process is very straightforward, and there's a full set of documentation on the **getbootstrap.com** website.

Bootstrap 3

Since the release of version 3, lets look back comparing to the predecessor and on advancement it has made.

The difference between Bootstrap 2 and Bootstrap 3 is that version 3 takes a mobile-first approach to ensure your designs is fully responsive on the get go.

Under the hood the developers have focused on trimming and rationalising the framework so that, for example, rather than internally referring to an alert box as 'red', a more semantic naming scheme is used. The CSS has been tidied up too.

There are some structural changes to the way you build your web apps, so the shift to version 3 from version 2 will require some HTML alterations, but the fundamental approach remains the same.

Build a custom Foundation template

Get started using Zurb's Foundation framework to build a responsive template that can be reused for any project

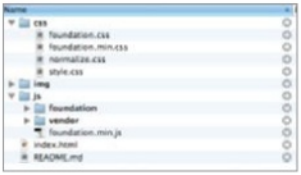
When starting a new responsive web design project, trying to custom develop your own framework that includes a flexible grid system, media queries, typography, JavaScript media including video, carousels, lightboxes and more can be exhausting. Not to mention the changing web arena, you'll have to rigorously maintain and update your framework to keep up to date with all the new features.

Using an open source framework like Foundation by Zurb helps to relieve the stress on developers. This robust and flexible framework is well documented and has a host of ready components to reuse; the best part is you can easily customise it, adding and removing functionality to suit your needs. Foundation is built to be mobile first, so as soon as you start using this framework for your web project it's instantly responsive.

Through this tutorial we'll touch upon some of Foundation's core components and how to assemble these to create your own custom template, which can be reused for any project.

Download Foundation

01 To begin, navigate over to the Zurb Foundation framework website at **foundation.zurb.com**. Click on the 'Download Foundation 5' button and you will be presented with multiple choices to download Foundation. You can select from a complete package, lightweight, custom or Sass. In this tutorial we'll be using pure CSS to allow beginners to follow along, so go ahead and download the complete version.



Unarchive the zip

02 Unarchive the Foundation framework you just downloaded; inside you will find a CSS directory that includes the complete Foundation CSS file and Normalize a CSS reset. Within the JS directory is the complete JavaScript library and scripts. The index.html has all the standard HTML5 boilerplate ready. Start by removing everything within the body tag.

HTML head

03 Within the HTML head we'll add a custom Google Font called Cabin, which can be found and downloaded at **www.google.com/fonts#UsePlace:use/Collection:Cabin**. It's good practice not to directly edit the foundation.css, which contains all the framework styles. We should house all our custom styles in a separate CSS file called style.css.

```
<link href='http://fonts.googleapis.com/css?family=Cabin:400,500' rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="css/foundation.css" />
<link rel="stylesheet" href="css/style.css" />
```

Basic styles

04 Now that we have our head of our HTML set up we can start applying some default styling. Create a new style.css in the CSS directory. Let's change the default font colour to a dark grey and make sure all of our headings are now using the new Google font Cabin.

```
body {
    color: #666;
}
h1, h2, h3, h4, h5, h6 {
    font-family: 'Cabin', 'sans-serif';
    font-weight: 500;
    color: #666;
}
```

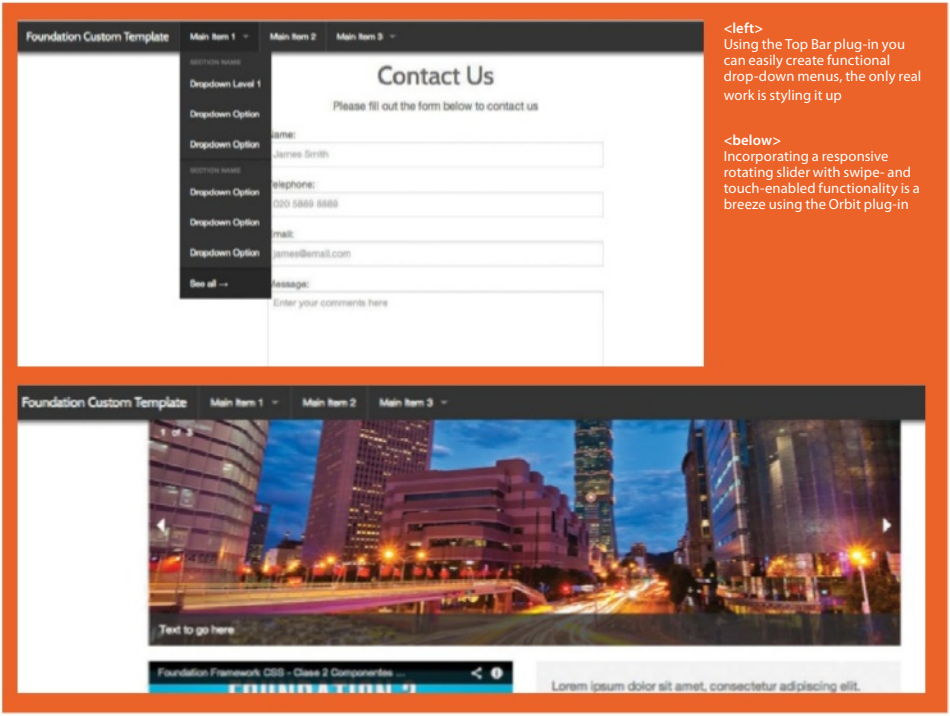
Build the navigation

05 Create a <nav> wrapper and within this have a <ul> with a <li> item for the main title, with a second <li> for the mobile menu

<above>
The Top Bar plug-in is fully responsive navigation, even the drop-down menus are device friendly

"Throughout this tutorial we have focused on pure CSS but you could use Sass, which allows greater customisation of the framework using nested styles, variables, function and reusable code."

Build a custom Foundation template



<left>
Using the Top Bar plug-in you can easily create functional drop-down menus, the only real work is styling it up

<below>
Incorporating a responsive rotating slider with swipe- and touch-enabled functionality is a breeze using the Orbit plug-in

(this will only appear on a small mobile screen). The top-bar class is used for presentational purposes, the data-option stick_on will force the navigation to stick to the top of the browser window.

```
<nav class="top-bar" data-topbar data-
options="   sticky_on: large">
      <ul class="title-area">
         <li class="name">
            <h1> <a
            href="#">Foundation
            Custom Template</
               a></h1>
         </li>
         <li class="toggle-
            topbar menu-
         icon"><a
            href="#"><span>Menu
         </span></a></li>
      </ul>
</nav>
```

Sub menu

06 Creating drop-down menus can sometimes be overly complex and tricky but Foundation makes the whole process very simple. Within the <nav> create a <section> with the class top-bar-section. Within this we create a <ul> and our <li> items, making sure to add the class has-dropdown to act as our secondary menu.

```
<section class="top-bar-section">
      <ul class="left">
         <li class="has-
            dropdown">
            <a class="active"
               href="#">
            Main Item 1</a>
            <ul class="dropdown">
               <li><a
               href="#">Dropdown
               Option</a></li>
            </ul>
         </li>
      </ul>
</section>
```

Style the menu

07 The beauty of an advanced framework such as Foundation is that it provides a lot of default styling for us, which means we can concentrate on structuring our page and customising our styling a lot less. You will notice the menu will have its own custom styling already. We'll just add a drop shadow with some browser vendor prefixes.

```
.top-bar {
   -webkit-box-shadow: 0px 3px
      7px 0px rgba
   (50, 50, 50, 0.75);
```

```
    -moz-box-shadow:    0px 3px
       7px 0px rgba
    (50, 50, 50, 0.75);
    box-shadow:         0px 3px
    7px 0px rgba
    (50, 50, 50, 0.75);
}
```

Orbit slider plug-in

08 Creating a carousel slider is extremely simple when we use the Orbit slider plug-in. First we'll wrap the code within a row class, followed by a large-12 columns grid to lay out our carousel. Apply a orbit-container class and a data- orbit data-attribute using a <li> item to separate each slide.

```
<div class="row">
   <div class="large-12 columns">
      <ul class="orbit-container"
      data-orbit>
         <li>
            <img src="img/
            slides/slide-1.jpg"
            alt="slider 1" />
            <div class="orbit-
               caption">
            Text to go here
            </div>
         </li>
      </ul>
      </div>
   </div>
```

Orbit JavaScript

09 Before the ending </body> tag there is an initial Foundation JavaScript, simply used to call all plug-ins on the page. Using this we can customise the plug-in initialisation. We can configure the options for the Orbit plug-in, so let's change the default slide effect to a fade animation and change the speed at which it rotates.

```
<script>
      $(document).foundation({
         orbit: {
            animation: 'fade',
            animation_speed:500,
            time_speed: 1000,
            pause_on_hover:true,
            bullets: false,
         }
      });
   </script>
```

Basic content

10 The grid system is one of Foundation's most powerful features, making it easy to create multi-device layouts. The row class is used to hold all the columns, defining a grid column of 12 indicates the number of columns we need in a row. Below is a basic structure of content using the grid system.

```
<div class="row main-content">
      <div class="large-12
         columns">
         <h1>Our latest work</h1>
         <p>Lorem ipsum….</p>
         <img src="img/photo.png"
          alt="" />
      </div>
   </div>
```

Make a child theme

11 Adding a video is extremely easy. Foundation will automatically help you to scale your video from an intrinsic ratio, so you can guarantee it can be properly scaled and viewed on any device. Simply add the class flex-video as a containing around your video.

```
<div class="row">
      <div class="large-6
         columns">
         <div class="flex-
            video">
         <iframe
      width="560" height="315"
   src="//www.youtube.com/embed/_
   ejeqxRdGwk"frameborder="0"
      allowfullscreen></iframe>
      </div>
   </div>
```

Panel component

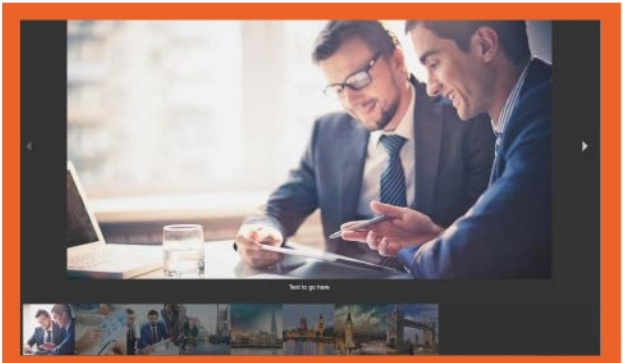
12 Let's have some accompanying content next to our video. We'll use another six-column grid and to add components that help outline sections on a page we can use the panel component. All you have to do is create a wrapping called panel encasing your content and Foundation will automatically apply a border and background colour to this panel.

```
<div class="large-6 columns">
<div class="panel">
      <p>Lorem ipsum ….</p>
   </div>
</div>
```

Lightbox gallery

13 To build a lightbox gallery of photos to slide through with a variable height, we can use the Clearing plug-in. Again, wrap the code in a row class with the grid columns, open a <ul> and apply a class of clearing-thumbs and a data-attribute

"Foundation contains a whole host of helpful components and the best place to read and understand all this is by looking at their detailed documentation at foundation.zurb.com/docs/index.html."



<above>
The Orbit plug-in cannot handle variable height images but the Clearing Lightbox plug-in can and makes scrolling through images friendly

of data-clearing. Each <li> item will act as a separate photo slide.

```
<div class="row gallery">
      <div class="large-12
         columns">
         <h2>Gallery</h2>
         <p>View our latest
         photos</p>
         <hr />
         <ul class="clearing-
         thumbs"data-clearing>
            <li>
               <a href="img/
                  photo.jpg">
                  <img src="img/
                   photo.jpg"
   alt="" data-caption="Text to go
      here" />
               </a>
            </li>
         </ul>
      </div>
</div>
</div>
```

Gallery styles

14 Let's apply some styling to the gallery. We'll give this section a light grey background and some top and bottom padding. Each <li> item will have a bottom and right margin so they're equally separated. Using the CSS pseudo class first-child we can give the first <li> item a margin-left.

```
.gallery {
   padding: 20px 0;
   background-color: #f2f2f2;
}
.clearing-thumbs li {
   margin: 0 20px 20px 0;
}
.clearing-thumbs li:first-child
{
   margin-left: 10px;
}
```

Common styles

15 To create a uniformed look and give all our elements equal spacing, we'll style the main content, gallery and the subsequent contact form (in the following step) with some margin and make the content aligned centre. If you ever use Sass with Foundation, you can always use the @ extend for this case.

```
.main-content, .gallery,
.contact-us {
   margin-bottom: 30px;
   text-align: center;
}
```

Contact form

16 Creating user-friendly and HTML5 validation-ready forms is wonderfully simple in Foundation.

Let's create a contact form: first, open a <form> tag and within this give it a data-abide data-attribute to enable validation. Make sure the input field has a required attribute with a pattern to define restraints on what users should input.

```
   <form data-abide>
      <div class="name-field">
         <label>Name:
            <input type="text"
            placeholder="
            James Smith"
            required pattern="
            [a-zA-Z]+" />
            </label>

      </div>
   </form>
```

Error message

17 If you do not fill out the input field or input the incorrect data this will return an error. However this does not display an error message to the user, which is not helpful. To apply an error message, add a <small> tag with the class error and place your error message within this.

```
   <small class="error">Please
enter your name</small>
```

Email input

18 It's the same markup again to set the email input field, use the input type attribute and place the pattern name within this to help validate the input field. To set an error message within the same , add the <small> tag with the class error.

```
<div class="email-field">
   <label>Email:
      <input type="email"
      placeholder=
      "james@email.com" required
      />
   </label>
   <small class="error">
Please enter your email</small>
</div>
```

Message field

19 A contact form would not be complete without a message field. Similarly to our input fields we created previously, and using the same structure, we will place a <textarea> in place of the input field. Again, apply an error message directly below this. Following this we also add a Submit button to finish it off.

```
<div class="message-field">
   <label>Message:
      <textarea
      placeholder="Enter your
      comments here" required></
      textarea>
   </label>
   <small class="error">Please
   enter your comments</small>
</div>
<button type="submit">Submit</
button>
```

Form styles

20 Now with our HTML structure of our contact form completed, we can look at applying some styling. Because Foundation has a lot of in-built default styles, we don't need to heavily customise the form styling – and it's already responsive. We will just make sure the labels are aligned to the left and provide an explicit height for the <textarea> field.

```
label {
   text-align: left;
   color: #666;
}
textarea {
   height: 200px;
}
```

Build the footer

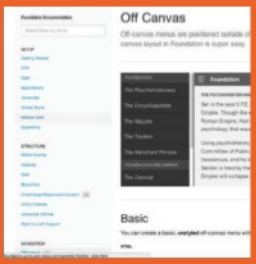
21 No website template would be complete without a footer. Let's add a <footer> tag that wraps the footer content inside. Again we'll need to dictate a row class and wrap a large-12 columns grid within our content. We will then separate the footer to two equal halves using two children large-6 column grids.

```
<footer class="row">
      <div class="large-12
      columns">
         <hr />
         <div class="large-6
            columns">
         <p>&copy;Copyright
         notice to go here</
p>
         </div>
         <div class="large-6
            columns">
         <ul class="inline-
            list right">
            <li>
               <a
                href="#">Link
                4</a>
            </li>
         </ul>
      </div>
   </footer>
```

Style the footer

22 Finally we'll style our footer. Let's give this a gradient background colour of a light black fading into a dark black. Included are some browser vendor prefixes. And that's all it takes! We've quickly constructed our own custom template using the Foundation framework and haven't had to write our own components, with minimal styling involved.

```
footer {
   color: #fff;
   background: rgb(43, 43, 43);
   background: -moz-linear-
   gradient(90deg,
rgb(43, 43, 43) 30%, rgb(29, 29,
   29) 70%);
   background: -webkit-linear-
   gradient
   (90deg, rgb(43, 43, 43) 30%,
   rgb
   (29, 29, 29) 70%);
   background: linear-
   gradient(180deg, rgb
   (43, 43, 43) 30%, rgb(29, 29,
   29) 70%);
}
```

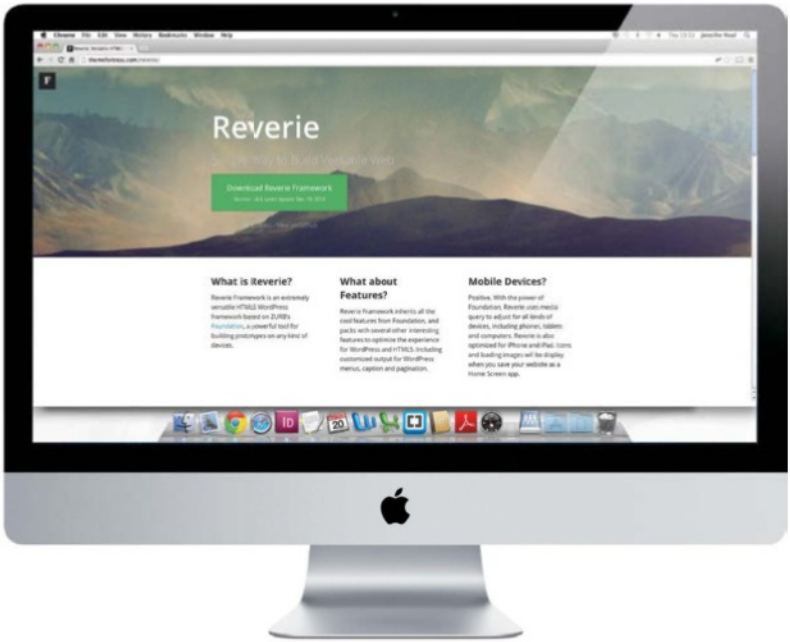


Foundation's wide range of navigation options

Foundation offers a wide range of navigation options and within this tutorial we've only covered Top Bar, which functions as a main navigation includes incorporating a complex responsive drop-down menu. You also have additional options to easily make this navigation stick to the top of the web browser.

Other popular navigation choices include Off Canvas, which many may have seen on mobile devices. This type of navigation is positioned outside of the viewport and slides in when activated by the user. Again, as it is highly customisable, you could even have menus appearing on both sides of the canvas.

Other more simplified navigation options that help keep track of the page while scrolling is Magellan. It acts like a Sub Nav but remains in a fixed position at the top of your screen.



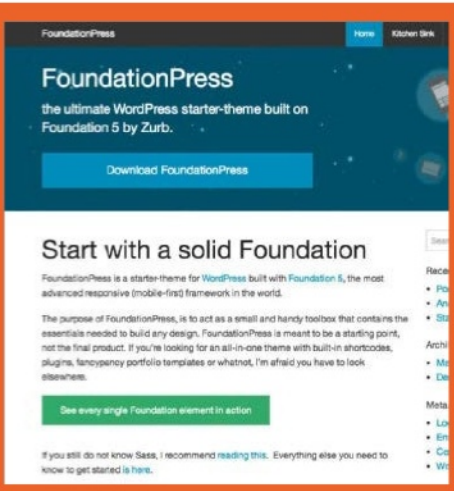
Build a responsive WordPress theme

As the most advanced responsive framework available, Zurb's Foundation is ideal for your latest WordPress project

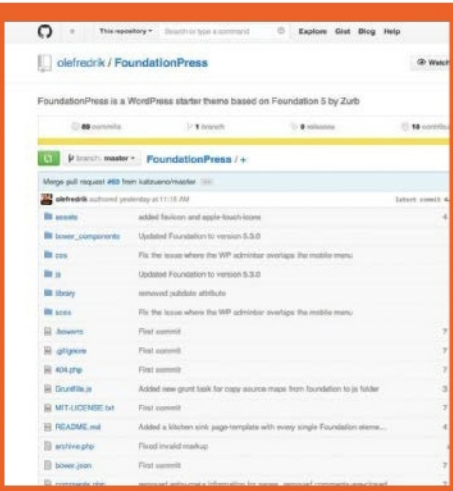
Previously to incorporate Zurb's Foundation framework into WordPress you had to manually link the necessary JavaScript and CSS using the WordPress functions 'wp_enqueue_script' and w'p_enqueue_style'. But now, thanks to Zurb and the popularity of WordPress, Zurb has provided a starter-theme called FoundationPress, which you can download directly from its GitHub page to start building your own WordPress theme with the Foundation framework.

The purpose of this ultimate starter theme is to act as a springboard; it comes packed with useful re-useable components, a 12-column responsive grid, JavaScript functions and much more. This starter theme contains all the necessary design elements, including the JavaScript and CSS libraries.

Still, FoundationPress is not an all-in-one WordPress theme with plugins, shortcodes, custom options or custom templates – it is only to be used as a starting point. Luckily it has done most of the hard work for us including setting up widgets, navigations, displaying blog posts and general clean-up of WordPress.



Start building responsive WordPress themes using Zurb's own advance Foundation framework starter theme, conveniently called FoundationPress



You can download Zurb's Foundation WordPress starter theme, FoundationPress, with ease over on GitHub

Download FoundationPress

01 To begin, make sure you have the latest version of WordPress installed and set up. Head over to the FoundationPress starter theme GitHub page: github.com/olefredrik/foundationpress and download this theme. Once you've downloaded it, extract it and upload it to your theme's directory in WordPress and activate it.

Custom stylesheet

02 Start by opening up the 'header.php' file in your text editor. You'll want to add in your own custom stylesheet so that when it comes to upgrading FoundationPress it won't override any custom work. You may use Sass with Foundation, but to allow beginners to follow along this tutorial, we'll be using pure CSS.

```
001   <link rel="stylesheet"
      href="<?php echo get_template_
      directory_uri(); ?>/style.css"
      />
```

Customise the menu

03 WordPress already features a built-in Appearance Menus Screen, enabling users to create custom navigation. FoundationPress takes this a step further by allowing us to create multiple navigations on either side on the nav bar. When you create a new menu in WordPress, you have the option to enable your navigations for mobile devices.

Customise the header

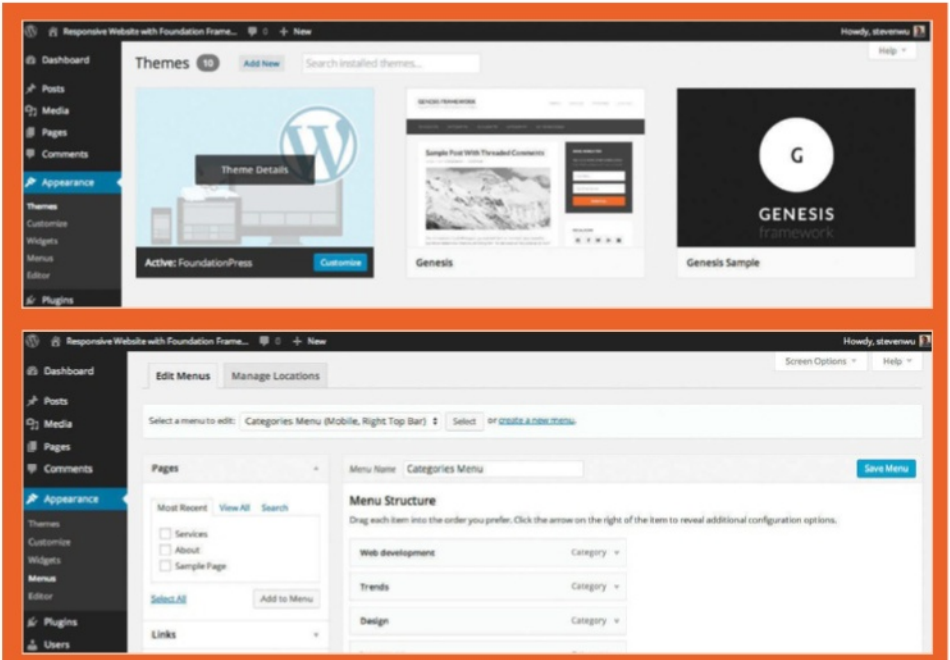
04 FoundationPress has already written a lot of the WordPress basic theming. We're going to dive in and modify it. To give this theme a visual punch we're going to add in a header with the website name, including a gradient background. Open up the 'header.php' and paste in the following HTML code.

```
001   <header class="main-head">
002     <div class="row">
003       <div class="large-12
            columns">
004         <h1 class="title"><?php
             bloginfo
            ( 'name' ); ?></h1>
005       </div>
006     </div>
007   </header>
```

Style the header

05 With our header implemented we'll now need to style it. Open up 'style.css' in the FoundationPress starter theme, start by giving the main header a gradient background, using CSS3 gradient. Position the main h1 above the header. Let's make all characters upper-case and give it a text shadow to stand out from the light backdrop.

```
001   .main-head {
002     margin-bottom: 2em;
```



<top>
In WordPress, activate FoundationPress starter theme. This theme includes all the Foundation framework library

<bottom>
FoundationPress allows you to order menus to be on the left or right side of the menu and enable for mobiles

```
003       padding: 1.2em;
004       background: #00b3d3;
005       background: -moz-radial-
gradient(center,ellipse cover,#00b3d3
          0%, #007295 100%);
006       background: -webkit-
          gradient(radial, center
          center, 0px, center center,
          100%, color-stop(0%,#00b3d3),
          color-stop (100%,#007295));
007       background: -webkit-radial-
          gradient (center, ellipse
      cover, #00b3d3 0%,#007295 100%);
008     }
009     .main-head h1 {
010       text-align: center;
011       font-weight: 900;
012       text-transform: uppercase;
013     letter-spacing: 10px;
014       text-shadow: 1px 1px 2px rgba
          (50, 50, 50, 0.59);
015       font-size: 2.5rem;
016       color: #fff;}
```

Add a search

06 Currently in our widgets list, the search form is located in the sidebar. In WordPress Widgets we can disable the search widget. Now inside our 'head.php' at the very bottom, paste in the following function, which will print out the search form. We're going to place our search form just below the header of our website.

```
001   <?php get_search_form(); ?>
```

Customise search

07 With the get search form function added in our 'header.php' we can now structure this search form by editing the 'searchform.php'. Directly below the form tag we have applied a row and changed the default grid layout to use the large grid system as well as setting some custom classes to be referenced in our CSS.

```
001   <div class="row">
002       <?php do_action
          ('foundationPress_
          searchform_top'); ?>
003       <div class="large-8 columns
          searchbox">
004           <input type="text" value=""
              name="s"id="s"placeholder
          ="<?php esc_attr_e('Search',
          'FoundationPress');?>">
005       </div>
006       <?php do_
          action('foundationPress_
          searchform_before_search_
          button'); ?>
```

```
007      <div class="large-4 columns
         searchbutton">
008         <input type="submit"
            id="searchsubmit"value
            ="<?php esc_attr_e
         ('Search','FoundationPress');
         ?>" class="prefix button">
009      </div>
```

Modify index.php

08 The 'index.php' is the main template in WordPress theming hierarchy. We're going to modify it and remove some of the grid components and place them in 'content.php' instead. Remove the row and grid classes just below the 'get_header' function and replace it with the code below. Finally, migrate the <?php get_sidebar(); ?> just above the 'get_footer' function.

```
001   <?php get_header(); ?>
002   <div class="row" data-equalizer>
```

Blog post

09 With the homepage structure completed, we'll want to structure each blog post in a grid column of four rows; this will lay out three blog posts organised next to each other on a large monitor. Using the grid system will force our layout to be responsive. Remove the original code all the way down to the <footer> tag in 'content.php'.

```
001   <div class="large-4 columns"
      role="main">
002      <article id="post-<?php the_
         ID(); ?>"
         <?php post_class('panel'); ?>
         data-equalizer-watch>
003         <header>
004            <h2><a href="<?php
               the_permalink();
               ?>"><?php the_title();
               ?></a></h2>
005         </header>
006         <div class="entry-content">
007            <figure><a href="<?php
               the_permalink
               (); ?>"><?php if (
               has_post_
               thumbnail() ) {the_
               post_thumbnail
               ('large'); } ?></a></
               figure>
008            <?php the_excerpt(); ?>
009            <p>Posted on <?php
               the_time('F jS, Y
               '); ?> in <?php the_
               category(', ');
               ?></p>
010            <p class="byline
               author">Written by
               <?php the_author_posts_
               link(); ?>
               </p>
011            <a href="<?php the_
               permalink(); ?>"
               class="button">Read
               more</a>
012         </div>
```

Style the blog

10 With our blog post structure set up in a grid column in a set of threes, we'll now implement some simple styling to this homepage. Back in our 'style.css', we'll add some box shadow so the panels don't look so flat. We don't need to style the panels themselves as these styles are already set up inside Foundation.

```
001   <div class="row main-content">
002         <div class="large-12
               columns">
003            <h1>Our latest work</h1>
004            <p>Lorem ipsum….</p>
005            <img src="img/photo.png"
               alt="" />
006         </div>
007      </div>
```

```
008   }
009
```

Style search form

11 With our blog posts neatly organised in a row of threes, our search field and search button are not quite aligned with the rest of the design. Let's fix this by adding some padding. With a mobile-first approach, Foundation takes care of the responsive nature – it's unnecessary to add styles for mobile or tablet devices.

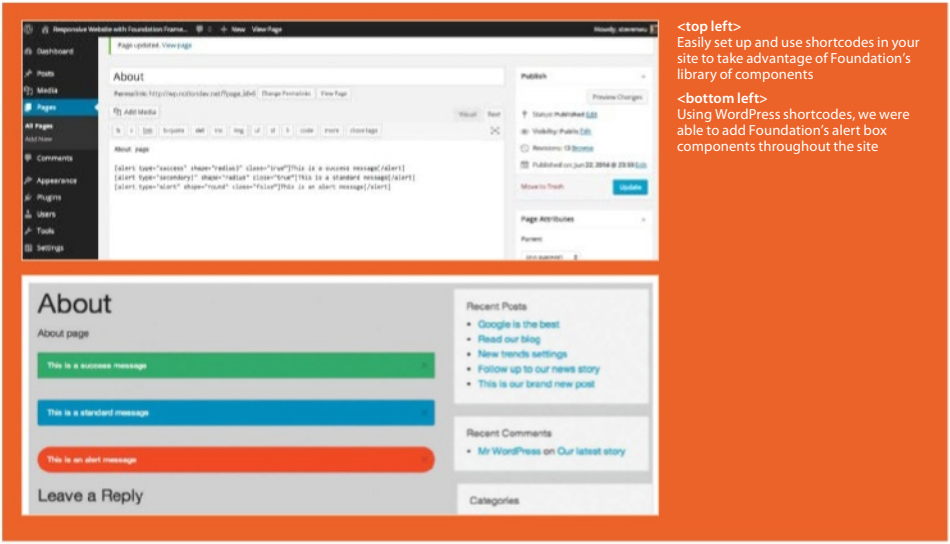
```
001   .searchbox, .searchbutton  {
002      padding: 0 15px;
003   }
004
```

Move the sidebar

12 Rather than having our sidebar situated on the right-hand column, we will position it directly below all the blog posts and just above the footer. Inside the 'sidebar.php' we're going to use Foundation's Equalizer component, which will set equal height to all the widget panels. We simply place the data-equalizer attribute to the parent container here.

```
001   <div class="bottom-sidebar">
002      <aside id="sidebar"
         class="row" data-equalizer>
003         <?php do_action
            ('foundationPress_
            before_sidebar'); ?>
004         <?php dynamic_
         sidebar("sidebar-widgets");
```

"Working with Sass provides greater flexibility and control over this theme. All Sass variables are located in scss/config/_variables.scss and your site structure within scss/site/_structure."



<top left>
Easily set up and use shortcodes in your site to take advantage of Foundation's library of components

<bottom left>
Using WordPress shortcodes, we were able to add Foundation's alert box components throughout the site

```
        ?>
005     <?php
        do_action('foundationPress_
        after_sidebar'); ?>
006     </aside>
007   </div>
```

Widgets

13 Now to actually lay out our widgets, we will need to open up the 'widget-areas.php' located in the library directory. Inside the first array on line 8, remove what is currently there and replace it with the below, which uses the large-4 grid and a 'data-equalizer-watch' attribute that the Equalizer component requires to set equal heights to each panel.

```
001 'before_widget' => '<article
id="%1$s"class="widget %2$s
large-4 columns"><div
class="panel" data-equalizer-watch>',
```

Style the sidebar

14 Within WordPress in the Widgets panel, we only require three widgets so remove any of the other existing ones. When you check the homepage you will notice our widget panels all have equal heights to the tallest panel. To finish up with our sidebar beneath our blog posts, we will give this section its own background colour and some padding.

```
001   .bottom-sidebar {
002     padding-top: 10px;
003     background-color: #cecece;
004   }
005
```

Set up shortcodes

15 There are many components available in Foundation, we convert some of them into shortcodes so that we can reuse these components whenever we're in the editor. Creating shortcodes requires two steps: create a primary handler method and connect the handler to WordPress. Inside 'functions.php' set up our primary function. Listed here.

```
001   function foundation_add_alerts
      ( $atts, $content = null ) {
002     extract( shortcode_atts(
        array(
003       'type'  => '',
004       'shape' => '',
005       'close' => 'true',
006       'class' => ''
007      ), $atts ) );
008
```

Shortcode array

16 In the previous code snippet we noted that our function receives the parameters of type, shape and close. We're going to use this to display alert boxes using one of Foundation's components. The type attribute will display success, warnings or informative information using this shortcode. Let's set up an array that will capture these attributes.

```
001   $class_array[] = ( $shape ) ?
      $shape : '';
002   $class_array[] = ( $type ) ?
      $type : '';
003   $class_array[] = ( $class ) ?
      $class : '';
```

```
004   $class_array = array_filter(
      $class_array );
005   $classes = implode( ' ', $class_
      array );
```

Shortcode markup

17 When this shortcode is executed we want to make sure that it's using the correct formatted markup and CSS classes. We need to use a wrapping class alert-box. This is very important because we want to make sure that Foundation calls the Alert JavaScript plugin for us. This particular plugin is in fact handled in the 'foundation.alert.js'.

```
001   $output  = '<div class="alert-
      box ' . $classes . '">';
002      $output .= do_shortcode(
         $content );
003      $output .= ( 'false'
         != $close ) ? '<a
      class="close" href="">&times;</
      a>' : '';
004      $output .= '</div>';
005      return $output;
006   }
007
```

Hook into WordPress

18 Before finishing off our shortcode we're going to ensure that it has all been registered correctly, otherwise WordPress won't know what to do with it. To do this we use the 'register_shortcodes' function and the 'add_shortcode' method. The first parameter defines the shortcode in the editor while the second points to the function that we created previously.

```
001   function register_shortcodes() {
002      add_shortcode('alert',
         'foundation_add_alerts');
003   }
004   add_action('init',
         'register_      shortcodes');
```

Shortcodes to use

19 We can now use the following shortcodes in the editor to display alert boxes, which is a native component from the Foundation framework. All we have to do is pass in which type of alert box type we want displayed by selecting the attributes: success, secondary or alert. We can even choose what type of shape to use and whether we want a close off function.

```
001   [alert type="success"
      shape="radius"
      close="true"]This is a success
      message[/alert]
002   [alert type="secondary)"
      shape="radius"
      close="true"]This is a standard
      message[/alert]
003   [alert type="alert" shape="round"
      close="false"]This is an alert
      message[/alert]
```

Alert boxes

20 Without any CSS styling or JavaScript and just using the shortcodes we have set up, we can implement alert boxes wherever we like in our WordPress site. By simply using shortcodes we can easily convert many of the rich components from Foundation to be used easily throughout our site.

Footer

21 To finish up we're just going to add in a footer navigation by using the WordPress WP list pages. Open up the 'footer.php' and just below the closing </section> tag, remove everything including the <footer> tag. Replace it with the following code, which will display a list of all pages on our site.

```
001   <footer class="main-footer">
002      <div class="row"
003         <ul>
004            <?php wp_list_pages();
               ?>
005         </ul>
006         <?php do_
            action('foundationPress_
            before_footer'); ?>
007         <?php dynamic_
            sidebar("footer-
            widgets"); ?>
008         <?php do_
            action('foundationPress_
            after_footer'); ?>
009      </div>
010   </footer>
```

Style the footer

22 Finally, we're just adding the final touches by styling up our footer. Through this tutorial we've taken the starter theme FoundationPress and quickly customised it as our own theme using a variety of Foundation's components and functionality. Using FoundationPress allowed us to focus on the front-end development and less on the actual backend efforts.

```
001   .main-footer {
002      padding: 10px 0;
003      background-color: #9c9c9c;
004   }
005   .pagenav ul {
006      margin: 10px 0 0 0;
007   }
008   .main-footer li {
009      float:left;
010      margin-right: 10px;
011      list-style: none;
012   }
013
```

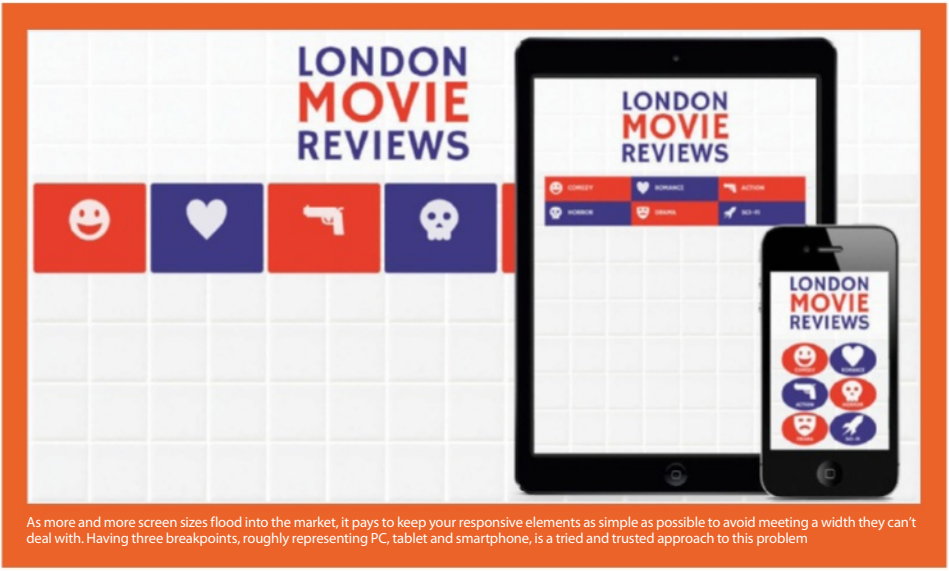
Here we have our fully responsive WordPress site, resizing to perfectly display on all screen sizes

Make a responsive menu for a retina screen

Make sure your menu is both responsive and ready for retina screens by using custom fonts instead of images

Designing the navigation element for your website in this age of mobile devices and differing screen widths can be a real pain in the mouse. Balancing usability with aesthetic credibility is the challenge that responsive design now presents to all designers. There are a wide variety of solutions and techniques available to solve the issue, from fixed drop-down menus to side-sliding variations. The problem is that, while these solutions all solve the problem of maintaining functionality at different screen widths, they can sometimes lacking visual impact.

These days it's nice to see a menu that remembers to look good, too. Longer, scrolling pages are reducing the number of menu elements some sites need to present, and those sites, in particular, can afford to be a little more adventurous with how they deliver the navigation. So it's time to give the responsive menu a much needed paint job and raise it above the purely functional. This tutorial will demonstrate how to create a responsive menu that is both eminently usable and visually striking. And to keep it retina friendly, we'll be using a custom-made icon font rather than images.



As more and more screen sizes flood into the market, it pays to keep your responsive elements as simple as possible to avoid meeting a width they can't deal with. Having three breakpoints, roughly representing PC, tablet and smartphone, is a tried and trusted approach to this problem

Choose your icons

01 In order to create the custom icon font, we'll be using the excellent IcoMoon tools at **icomoon.io/app/#/select**. You have a choice here of creating your own icons as vector images to import into the font creator or choose from the extensive selection already available. We'll choose ours from the library as they have exactly what we need.

Create a custom font

02 Having selected your icons, click the bottom Font button. On the next page, you can amend the icon's tags and corresponding letters used to call each icon. Click on Download and you will get access to the folder with its font files and stylesheet. Place the font folder into your root.

Add the custom font

03 Copy the custom CSS that came with your download and paste it into the top of your site stylesheet. Ensure that the path for the font files is still correct. You may also want to make sure that each icon is assigned to the font family individually by name rather than with the default blanket icon class.

```
@font-face {
  font-family: 'icomoon';
  src: url('../fonts/icomoon.eot');
  src: url('../fonts/icomoon.eot?#iefix')format('embedded-opentype'),
       url('../fonts/icomoon.woff')  format('woff'),
    url('../fonts/icomoon.ttf')  format('truetype')
url('../fonts/icomoon.svg#icomoon')  format('svg');
  font-weight: normal;
  font-style: normal;
}
.icon-happy,.icon-loved,.icon-uniF4F1,.icon-skull,.icon-uniF539,.icon-rocket,.icon-menu {
    font-family: 'icomoon';
    speak: none;
      font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
  }
    .icon-happy:before {content: "\e603";}022  .icon-loved:before {content: "\e61c";}
  .icon-uniF4F1:before {content: "\e613";}
  .icon-skull:before {content: "\e607";}
  .icon-uniF539:before {content: "\e612";}026  .icon-rocket:before {content: "\e600";}

```

Base CSS

04 Before we start putting all the menu HTML into the index.html, we just need to prepare some base CSS for the project. Start by creating a container, main and a header into which the site title will be placed, making sure to call the stylesheet in the head. Use the following CSS to style the wrapping elements.

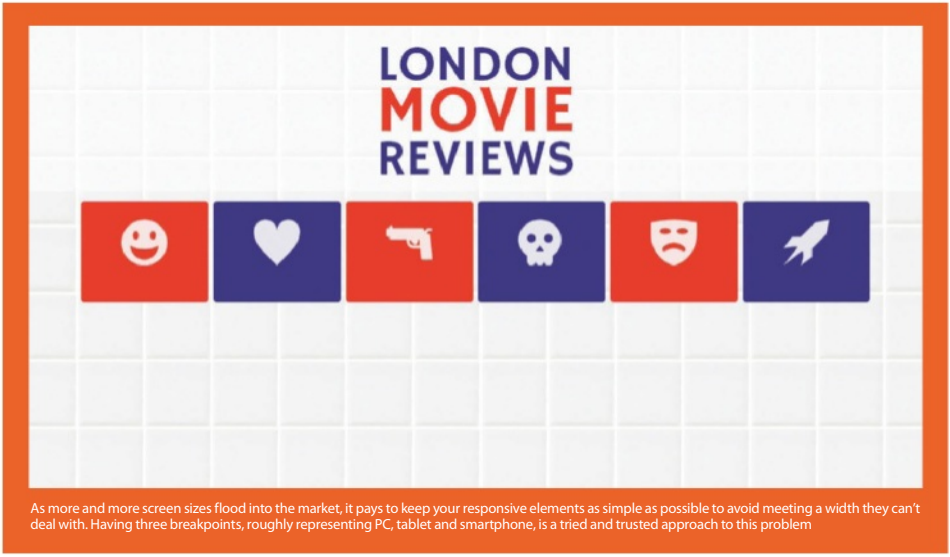
```
.main,
.container > header {
    width: 100%;
    margin: 0 auto;
      padding: 16px 32px;
}
.main {
    max-width: 1300px;
    min-height: 640px;
}
header {
    text-align: center;
    font-size: 16px;
      background: rgba(255,255,255,0.6);  }
header h1 {
    font-family: 'Hammersmith One', sans- serif;
    font-size: 75px;
      line-height: 0.8;
    margin: 0;
    font-weight: 300;
    color:#3f3782;
    }
  .movie {
    font-size:102px;
    color:#e63c2b;
  }
```

Menu HTML

05 Create your menu <ul> using span classes for both the icon and the text label. The span class 'fon' will create the styling for the custom icons, while the class 'item' is used for the labels. The icon name, as declared in the IcoMoon CSS calls each custom icon to its element.

```
<div class="container">
<header>
<h1>LONDON<br><span class="movie">MOVIE</span><br>REVIEWS</h1>
</header>
<div class="main">
<nav id="menu" class="nav">
<ul>
<li><a href="#"><span class="fon ico-  happy"></span><span class="item">COMEDY</span></a></li>
<li><a href="#"><span class="fon ico-  loved"></span><span class="item">ROMANCE</  span></a></li>
```

"Avatars are the images that appear alongside comments. It's worth allowing them because they brighten them up"



As more and more screen sizes flood into the market, it pays to keep your responsive elements as simple as possible to avoid meeting a width they can't deal with. Having three breakpoints, roughly representing PC, tablet and smartphone, is a tried and trusted approach to this problem

```
010 <li><a href="#"><span class="fon
ico-  uniF4F1"></span><span
class="item">ACTION</span></a></li>
011 <li><a href="#"><span class="fon
ico-  skull"></span><span
class="item">HORROR</span></a></li>
012 <li><a href="#"><span class="fon
ico-  uniF539"></span><span
class="item">DRAMA</ span></a></li>
013 <li><a href="#"><span class="fon
ico-  rocket"></span><span
class="item">SCI-FI</span></a></li>
014 </ul>
015 </nav>
016 </div>
017 </div>
```

Non-media query CSS

06 There are some styles that apply to all three of the upcoming menu sizes, setting some navigation styles and using a WebKit trick to eliminate blue bordering on click. Here we also colour our navigation element backgrounds, using the nth property to alternate the colour between the site theme red and blue.

Large screen query

07 Our first media query targets screens of 800px or more. The layout is a horizontal strip of rounded squares, with each element at a percentage for maximum flexibility. Icon and label are stacked one above the other. We set the position and size of our '.fon' icon, while the '.item' label begins opaque, to be animated later

First transition step

08 Now we can arrange hover effects. The first step will see the .item label fade in on hover, with a .5s transition, while the '.fon' icons change their colour to match the appearing label. Both elements are now black from white.

Second transition step

09 The black characters don't look bad against the coloured elements, but what would really make the highlighted icon stand out is having no background at all. So, we'll add a transition that fades out the background on hover, leaving just the icon and visible label. We can also close that first media query here.

```
001 .nav li:hover {
002   background:none;
003   -moz-transition: background
```

"So, we'll add a transition that fades out the background on hover, leaving just the icon and visible label"

```
.5s;
004   -o-transition: background
.5s;
005   -ms-transition: background .5s;
006      transition: background .5s;
007   }
008 }
```

Add the background

10 We'll finish off the site properly with an appropriate background image. In keeping with the London Underground theme, we're using for our London Movie site, we're going with a white tile motif, and making sure the Google font is used throughout.

```
001 body {
002      font-family: 'Hammersmith
One', sans- serif;
003   background: url(../img/bg.jpg);
004   }
005   margin: 20px;
006       border: 1px solid red; /*
take off asap  */
007 }
008
```

Second media query

11 Our second media query targets screen widths between 520px and 799px, which covers tablet screens and some larger smartphones at landscape. Our layout will differ considerably at this size. Two rows of three blocks, no margins and no rounded corners. Our fonts are also considerably smaller and sit side by side.

Tablet animation

12 In a reversal of our hover effects for large screens, this time we're going to change the background colour of the li element to black and give the white font and icon an off-white shade and a very slight fade. With that, all of the tablet styles are completed, so make sure you remember to close the media query here.

```
001 .nav li:hover {
002      background:rgba(0,0,0,0.9);
003      -moz-transition: background
.5s;
004      -o-transition: background
.5s;
005      -ms-transition: background
.5s;
006      transition: background .5s;
007   }
008   .nav a:hover .fon,
009   .nav a:active .fon,
010   .nav a:active .item,
011 .nav a:hover .item{
012      color:rgba(259,259,259,0.9);
013      -webkit-transition: color
.5s;
014      -moz-transition: color .5s;
015      -o-transition: color .5s;
016      -ms-transition: color .5s;
017   transition: color .5s;
018   }
019   }
020
```

Rather than having to adapt the layout per screen width, we'll use each breakpoint as an opportunity for redesigning

Make a responsive menu for a retina screen

Target smaller screens

13 The final media query sets the layout for screens smaller than 520px, targeting smartphones. Since we have so few menu elements, we can be bold and eschew the trend for JavaScript drop-down menus. We'll also lose the corners and have six circular elements, with icon and label fitting together snugly.

Header and fonts

14 The font sizes for the site title need to be reduced for this screen size, using iPhone as a guide. We can also allow for a nice, large font size for our icon and we need to make sure that fumbling fingers are able to click on the elements. Notice that the icons remain crisp at all sizes – that's the benefit of our custom font.

Background changes

15 Since the order of the elements has now changed, we'll need to do some tweaking to the background colours. At the moment we have one column of red and one column of blue, which looks a little bit odd. In order to keep the alternating pattern, we need to arrange a one, two, two, one repeat of the colours.

```
.nav li:nth-child(6n+1) {
    background: rgb(230, 60, 43);
  }
  .nav li:nth-child(6n+2) {
    background: rgb(63, 55, 130);
  }
  .nav li:nth-child(6n+3) {
  background: rgb(63, 55, 130);
  }
.nav li:nth-child(6n+4) {
    background: rgb(230, 60, 43);
  }
  .nav li:nth-child(6n+5) {
    background: rgb(230, 60, 43);
  }
  .nav li:nth-child(6n+6) {
    background: rgb(63, 55, 130);}
```

Small screen transitions

16 This time, we'll still fade out the background of the clicked element, but change the icon and label to the same colour as the background it sits on. The CSS used here fades the background of all li elements on hover, but the colour changes will be done separately. Here we deal with the red elements.

Last transition

17 Next, we're coding the blue elements. Close the media query and we're done. We now have three very distinct layouts between the three breakpoints. Your menu at small-screen width doesn't have to be a small version of the large screen layout. You have the chance to give each menu its own design.

Having fewer menu elements means that we can avoid having to hide them. Instead we can arrange them within our screen space

The HTML in full

In this tutorial we've covered all of the CSS in detail, so now we take the opportunity to get a closer look at the inner workings of the HTML

IE will use its most up-to-date rendering engine. Chrome=1 activates the Chrome Frame plug-in for early IE versions

Here we tell the browser that the site is mobile ready and make sure the user can't scale the content, which they don't need to do

Make sure you remember to include the Google Font declaration in the head of your index.html

Though it may seem span-heavy, the span is one of the most under-appreciated HTML elements. Use them freely!

```
001     <!DOCTYPE html>
002     <html lang="en">
003     <head>
004     <meta charset="UTF-8" />
005     <meta http-equiv="X-UA-Compatible"
content="IE=edge,chrome=1">
006     <meta name="viewport" content="width=device-width,    initial-
scale=1.0, user-scalable=no">
007        <title>London Movie Reviews</title>
008        <link rel="stylesheet" type="text/css" href="css/style.        css"
/>
009        <link href='http://fonts.googleapis.com/
css?family=Hammersmith+One' rel='stylesheet' type='text/css'>
010     </head>
011     <body>
012        <div class="container">
013        <header>
014     <h1>LONDON<br><span class="movie">MOVIE</
span><br>REVIEWS</
015     h1>
016        </header>
017        <div class="main">
018        <nav class="nav">
019        <ul>
020     <li><a href="#"><span class="fon icon-happy"></span><span
class="item">COMEDY</span></a></li>
021        <li><a href="#"><span class="fon icon-loved"></
span><span class="item">ROMANCE</span></a></li>
022    <li><a href="#"><span class="fon icon-uniF4F1"></span><span
class="item">ACTION</span></a></li>
023     <li><a href="#"><span class="fon icon-skull"></span><span
class="item">HORROR</span></a></li>
024    <li><a href="#"><span class="fon icon-uniF539"></span><span
class="item">DRAMA</span></a></li>
025     <li><a href="#"><span class="fon icon-rocket"></span><span
class="item">SCI-FI</span></a></li>
026     $(window).scroll(function(){
027     </ul>
028     </nav>
029     </div>
030    </div>
031 </body>
032   </html>
```

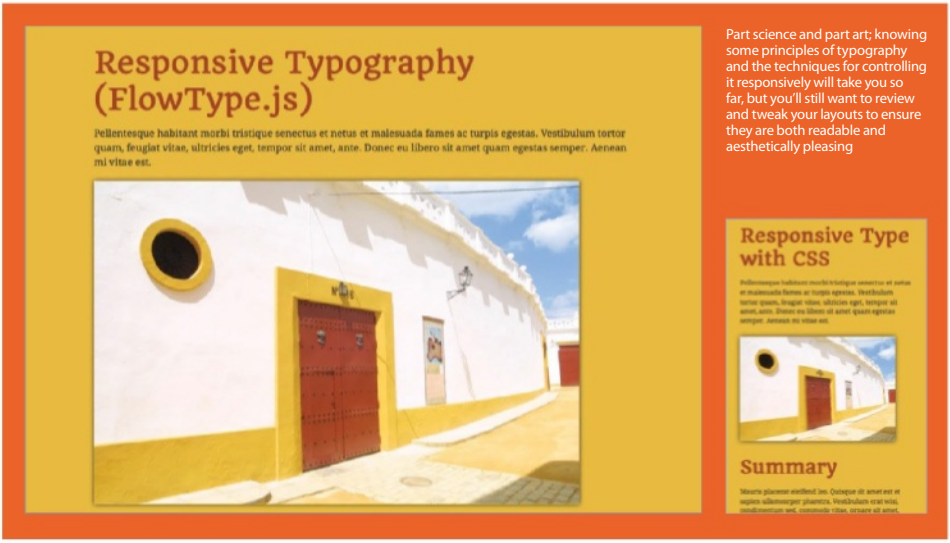
"Your menu doesn't have to be a small version of the large screen layout. You can give each menu its own design"

Techniques for creating responsive typography

Make your typography responsive to deliver a first class user experience on every device

If you look across a range of interests you'll often find some mixed-up priorities. In photography, many beginners focus on the best camera body they can afford whereas experienced photographers know it's the quality of the lens that's all-important. Some golfers will spend hours at the driving range but completely neglect their short game.

With responsive web design, the temptation can be to concentrate on the overall layout, and which divs do what as the viewpoint changes, but what about your text? People have arrived at your site to read what you have to say and if they don't find it easy or enjoyable they simply won't stay. If you want your website to be responsive and effective then you really need to know a little about responsive typography. As with many other web design disciplines you'll have to deal with browser compatibility issues, and as you venture further, you'll appreciate the diversity of the issues you need to consider and the refinements that are possible. You can achieve a great deal with CSS only but there are also a few handy jQuery plugins too. It's time to focus on the typography.



The measure of it

01 For optimum readability your line length (or measure) should be 45-75 characters. Within sensible limits, select your font and the width it is to occupy. You can then set the font-size of your body text. In this case the size is set to 18px in the root element 'html'.

```
@import url(http://fonts.googleapis.com/
    css?family=Droid+Serif|Quando);
* {
    margin: 0;
    padding: 0;
}
html { font-size: 18px;
    font-family: 'Droid Serif', serif;
    background-color: #e7bb4a;
    color: #333;
}
```

Keep it relative

02 Using rem units (as in h2 here) enables the relative size of the element to be fixed according to the root element. You might have several dozen rem-sized elements in your document and simply changing the font-size in your html element will automatically change their size.

```
h2 { font-size: 2.8rem;
    margin-bottom: 15px;
    font-family: 'Quando', serif;
    color: #a2422a;
}
p {
    line-height: 1.45;
    margin-bottom: 20px;
}
```

Manage your widths

03 Remember that maintaining suitable column widths is essential in achieving good readability. On a desktop screen, if you have only one column of text you'll need to allow plenty of negative space around the text block to ensure a good user experience. The img element shows an example of using rem units to set a relative size to the bottom margin.

```
#page-wrap {
    width: 60%;
    margin: 100px auto;
}
img {
        max-width: 90%;
        margin-bottom: 2rem;
        box-shadow: 0 0 8px #333333;
}
```

Breakpoints

04 Decide on the breakpoints to use for your site. Using ems means your breakpoints will be respected even if the user changes the text zoom in their browser. At each breakpoint you have the opportunity to tweak the factors that will affect the readability of the text at that screen size.

```
/* ===== == = 20em (320px) = == ===== */
@media only screen and (min-width : 20em) {
    html {font-size: 12px;}
    #page-wrap { width: 85%; margin: 1rem
        auto; }
}
/* ===== == = 30em (480px) = == ===== */
@media only screen and (min-width : 30em) {
        html {font-size: 14px;}
        #page-wrap { width: 80%; margin:
            1.5rem auto; }
}
```

Subtle and small

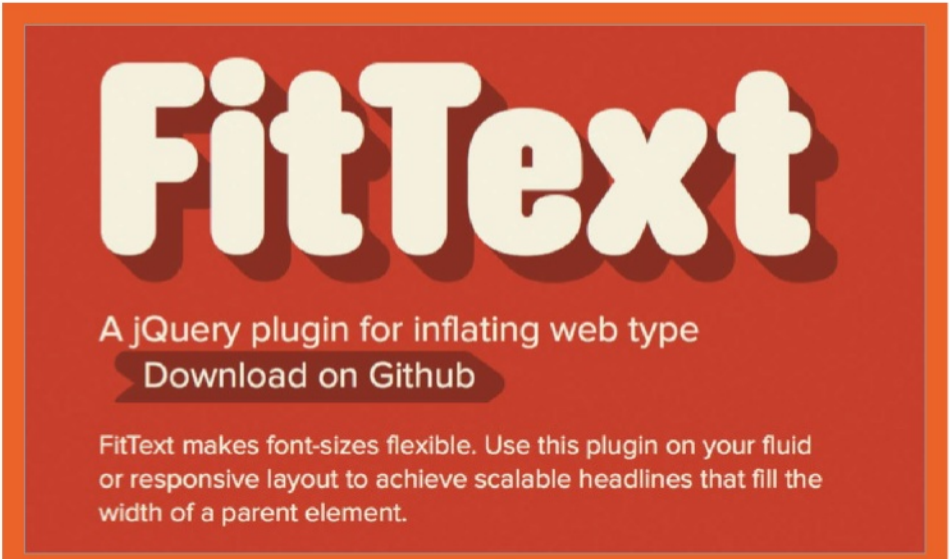
05 On smaller screens the difference between your largest heading and smallest text should be reduced, so you will typically use smaller rem values. You will also tend to use much narrower margins to enable you to make the most of the available screen space.

```
/* ===== == = 37.5em (600px) = == ===== */
@media only screen and
    (min-width: 37.5em) {
        html {font-size: 16px;}
        #page-wrap { width: 75%; margin: 2rem
            auto; }
    }
/* ===== == = 48em (768px) = == ===== */
@media only screen and (min-width : 48em) {
        html {font-size: 18px;}
        #page-wrap { width: 65%; margin: 2.5rem
            auto; }
    }
```

Large and bold

06 At larger sizes you'll really want to make use of impressively sized headings and large margins to give each element its own space.

```
/* ===== == = 56.25em (900px) = == ===== */
@media only screen and (min-width :
    56.25em) {
        html {font-size: 20px;}
        #page-wrap { width: 60%; margin: 3rem
            auto; }
    }
/* ===== == = 68.75em (1100px) = == ===== */
@media only screen and (min-width :
    68.75em) {
        html {font-size: 22px;}
        #page-wrap { width: 55%; margin: 3.5rem
            auto; }
    }
/* ===== == = 81.25em (1300px) = == ===== */
@media only screen and (min-width :
    81.25em) {
        html {font-size: 24px;}
        #page-wrap { width: 50%;
```



Headline text is the most noticable text on a page and any good headline should scale down so it is easy to read on all screens. FitText.js is a jQuery plugin that does exactly what it says, make text fit, whatever the screen size

```
margin: 4rem
        auto; }
015   }
016
```

Width=device-width

07 Switching to the HTML, you'll need to set how 'width' is handled. This enables you to determine how the page is scaled rather than leaving it to the device to decide. Without this tag, smartphones will tend to 'zoom out' in an attempt to show the whole page.

```
001   <!DOCTYPE html>
002   <html>
003   <head>
004      <title>Responsive Typography</
title>
005      <meta charset="UTF-8" />
006      <meta name="viewport"
content="width=
         device-width, initial-
scale=1">

007       <link rel="stylesheet"
href="css/style.
         css">
008   </head>
009
```

Using FlowType.js

08 User experience brand firm Simple Focus developed this plugin to simplify the process of maintaining optimum line length using jQuery and without the need to use media queries. First you'll need to link to jQuery and the plugin in the head of your HTML document.

```
001      <script src="js/jquery-
1.11.0.min.js">
         </script>
002      <script src="js/flowtype.
js"></script>
```

Max mins

09 After the HTML content you call the FlowType script and set several variables. The minimum and maximum values are the threshold at which the plugin stops resizing text, and the minFont and maxFont the smaller and largest size in pixels that the plugin will use.

```
001   <script>
002      $('body').flowtype({
003      minimum   : 500,
004      maximum   : 1200,
005      minFont   : 12,
006      maxFont   : 40,
```

```
007       fontRatio : 45
008   });
009   </script>
010
```

Font ratio

10 Each font is different and the fontRatio variable enables you to tweak the font size to achieve your desired result. Increase the value to make the font smaller, and vice versa. It should go without saying that you need to check how your page looks at different widths.

Media queries again

11 FlowType does a great job of managing line lengths to optimise readability but it won't make the other tweaks that you will want to make to reflect the needs of smaller and larger screens. You'll still want to use smaller margins on tiny screens and larger margins on big ones so a simplified set of media queries can be used.

```
001   @media only screen and
      (min-width : 20em) {
002   #page-wrap { width: 85%; margin:
1em
      auto; }}
003   @media only screen and
      (min-width : 30em) {
004   #page-wrap { width: 80%; margin:
1.5em
      auto; }}
005   @media only screen and
      (min-width: 37.5em) {
006   #page-wrap { width: 75%; margin:

      2em auto; }}
007   @media only screen and
      (min-width : 48em) {
008   #page-wrap { width: 65%; margin:

      2.5em auto; }}
009   @media only screen and
      (min-width : 56.25em) {
010   #page-wrap { width: 60%; margin:

      3em auto; }}
```

R for root

12 By default, Flowtype is applied to the body of the page. However, as the heading is set using rem units, the heading will not be re-sized. There are two ways round this. First you can set Flowtype to apply to the root element by changing 'body' to 'html' towards the bottom of your HTML, or you can use em instead.

```
001   $('html').flowtype
```

Compatibility and fallback

13 For older browsers that do not support rem you will want to include fallback. On the face of it the fallback is simple – just define the size in pixels first. However, that could lead to a lot of manual changes so you might want to explore using LESS or SASS mixins to minimise duplicated code.

```
001   h1   { font-size: XXpx; font-
size:
      YYrem; }
```

Viewport sized typography

14 Another method, growing in popularity, for creating responsive text makes use of viewport units. Each unit is equivalent to 1% of the viewport width or height. Browser support is currently lower than for rem units but in exchange you get fully fluid text and could probably make do with only two media queries to avoid inappropriately small or large text.

```
001   h1 {
002   font-size: 5.9vw;
003   }
004   h2 {
005   font-size: 3.0vh;
006   }
007   p {
008   font-size: 2vmin;
009   }
010
```

FitText.js

15 Sometimes you might just want help with making your headings work nicely at all sizes. In that case FitText.js might be the best solution for you. Once set up, you simply apply an id for each heading to be sized and when the function is called, pass some optional configuration settings.

```
001   <h1 id="fittext1">Squeeze with
FitText
      </h1>
002   <h1 id="fittext2">Squeeze with
FitText
      </h1>
003   <h1 id="fittext3">Squeeze with
FitText
      </h1>
004   $("#fittext1").fitText();
005   $("#fittext2").fitText(1.2);
006   $("#fittext3").fitText(1.1, {
minFontSize:
       '50px', maxFontSize: '75px' });

007
```

A more modern scale

16 Jason Pamental (**bit.ly/LdQ9Ut**) has considered the challenges of responsive typography in great detail. To quote Jason, "The problem

"For older browsers that do not support rem you will want to include fallback"

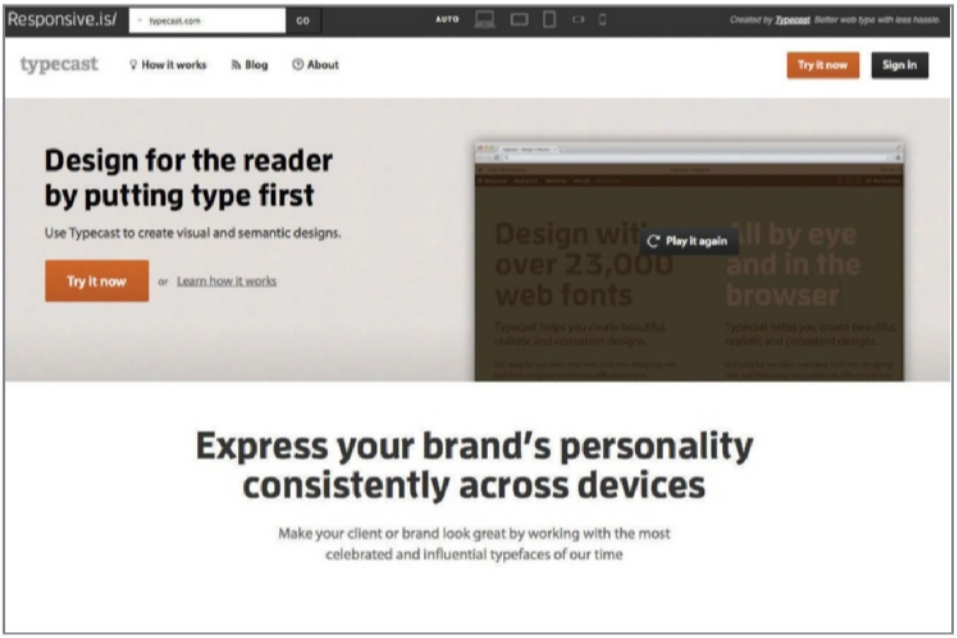
is, as the screen size shrinks and fewer elements are visible, the relative scale between elements becomes exaggerated. What's needed is greater subtlety and flexibility to maintain a more balanced proportion and better readability across all experiences."

Do it online

17 If you'd like more layout support with a focus on typography, **typecast.com** provides a suite of online tools to rapidly prototype and test your designs. Not for you? Typecast's responsive.is adds a utility bar to the top of your page where you can check sites based on five different breakpoints all without having to re-size your browser manually.

BODY	FONT SIZE	LINE HEIGHT	CHARACTERS
Print	12pt	1.25em	60-75
Desktop (lg)	1em (16px)	1.375em	60-75
Desktop	1em (16px)	1.375em	60-75
Tablet (lg)	1em (16px)	1.375em	60-75
Tablet (sm)	1em (16px)	1.25em	60-75
Phone	1em (16px)	1.25em	35-40

"You can check sites based on five different breakpoints all without having to re-size your browser manually"



FlowType.js

Simple Focus is the name of the agency that developed this plugin which is apt here as the code is both simple and focused.

```
(function($) {
   $.fn.flowtype = function(options) {

// Establish default settings/variables
// ====================================
      var settings = $.extend({
         maximum   : 9999,
         minimum   : 1,
         maxFont   : 9999,
         minFont   : 1,
         fontRatio : 35
      }, options),

// Do the magic math
// =================
      changes = function(el) {
         var $el = $(el),
            elw = $el.width(),
            width = elw > settings.maximum ? settings.maximum
            : elw < settings.minimum ? settings.minimum : elw,
            fontBase = width / settings.fontRatio,
            fontSize = fontBase > settings.maxFont ? settings.
            maxFont : fontBase < settings.minFont ? settings.
            minFont : fontBase;
         $el.css('font-size', fontSize + 'px');
      };

// Make the magic visible
// ======================
      return this.each(function() {
      // Context for resize callback
         var that = this;
      // Make changes upon resize
         $(window).resize(function(){changes(that);});
      // Set changes on load
         changes(this);
      });
   };
}(jQuery));
```

Variables are set here, but can be overridden when the plugin is called in the HTML

The width of the element is checked to see if it is outside the maximum and minimum parameters, in which case the maximum or minimum values are used

Similarly, when a new font size is calculated, a check is made to see if this is outside the extremes set by the user and, if so, the largest or smallest font size is set as appropriate

The code applies the newly calculated numbers as inline CSS to the element that is selected. Because this new CSS is inline, it overwrites whatever you have set in your linked stylesheets, creating a natural fallback in case a user has JavaScript disabled

"What's needed is greater subtlety and flexibility to maintain a more balanced proportion and better readability"

Create a HTML5 responsive video player

Discover how to create a fully responsive HTML5 video player that features custom control icons to fit your designs

Everyone knows that video is the best way to keep your audience engaged, however, you now want to add some of that video goodness onto your fancy responsive website. You scurry off to YouTube and grab that all-important embed code, lovingly paste it in, only to find that it looks terrible on tablets and phones. Not only that but the controls are all the same boring icons. Time for a change, in the form of an HTML5 video player that adapts to your responsive site and fits in with your existing UI with some custom controls.

We'll be basing this tutorial around Bootstrap, for a couple of reasons. The first being that it's the most popular responsive framework out there, so many people will already be using it; the second reason is that it comes bundled with Glyphicons, which we'll use to create player controls. You can, of course, create your icons using standard PNG images if you so wish.

Using this technique we will end up with a video that looks consistent, as the standard HTML5 video player is rendered completely different across different browsers using the browsers built-in controls, here we make it look identical across all platforms, and respond to smaller devices.

Get set up

01 Create a new HTML file and add in the following code into the <head> section. This will add the Twitter Bootstrap framework that is required to make your page responsive. If you are already using Bootstrap and are simply integrating into an existing site, you can skip this part of the tutorial.

```
001   <link rel="stylesheet" href="//
netdna.
      bootstrapcdn.com/bootstrap/3.1.1/
css/
      bootstrap.min.css">
002   <script src="https://ajax.
googleapis.
      com/ajax/libs/jquery/1.11.0/
jquery.min.js">
      </script>
003      <script src="//netdna.
bootstrapcdn.com/
      bootstrap/3.1.1/js/bootstrap.
min.js">
      </script>
```

The video container

02 The below code adds a very basic responsive video container based on the Twitter Bootstrap Jumbotron element. Place this code within the body section of your page; it will render a full-width container at the top of the site. If you are implementing the video player in an existing site you can skip this step.

```
001   <div class="jumbotron">
002      <!--  Begin video content -->
003   </div>
```

Demonstrate responsiveness

03 Just to add some context to our site, we're going to place two responsive columns underneath our jumbotron. These columns have no bearing on the video element, but it will make the final product look more like an actual website. Add this

"Time for a change, in the form of an HTML5 video player that adapts to your responsive site"

code underneath the previous step above the closing </body> tag.

```
<div class="container">
   <div class="row">
      <div class="col-md-6">
         <h2>Left Col</h2>
         <p>Donec id [...] dui.</p>
      </div>
      <div class="col-md-6">
         <h2>Right Col</h2>
         <p>Donec id [...] dui.</p>
      </div>
   </div>
</div>
```

Obtain footage

04 Now we have our code set up, we need an actual video to show. For this we'll be using a completely open source short animation created by Blender Foundation, 'Big Buck Bunny'. Download it from the link below in any format you wish – they even have a 4K and 3D version as well as the entire source used to create it.

```
http://bbb3d.renderfarming.net/download.html
```

Embed video

05 Now we have our footage we're going to embed it into the jumbotron using the standard HTML5 markup. Add the following code within the jumbotron , replacing the <!-- Begin Video content --> comment. Be sure to look at the output in different browsers to see how the video controls are rendered differently.

```
<video width="320" height="240"
      controls>
   <source src="big_buck_bunny_1080p_
      h264.mp4" type="video/mp4">
   Your browser does not support the
      video tag.
</video>
```

Video formats

06 Depending on which browser and operating system you are using, the previous step may not have shown any video at all. If this is the case you may need to add the OGG version of the video into your code as well. Grab the OGG file from the video download site and add the following line of code.

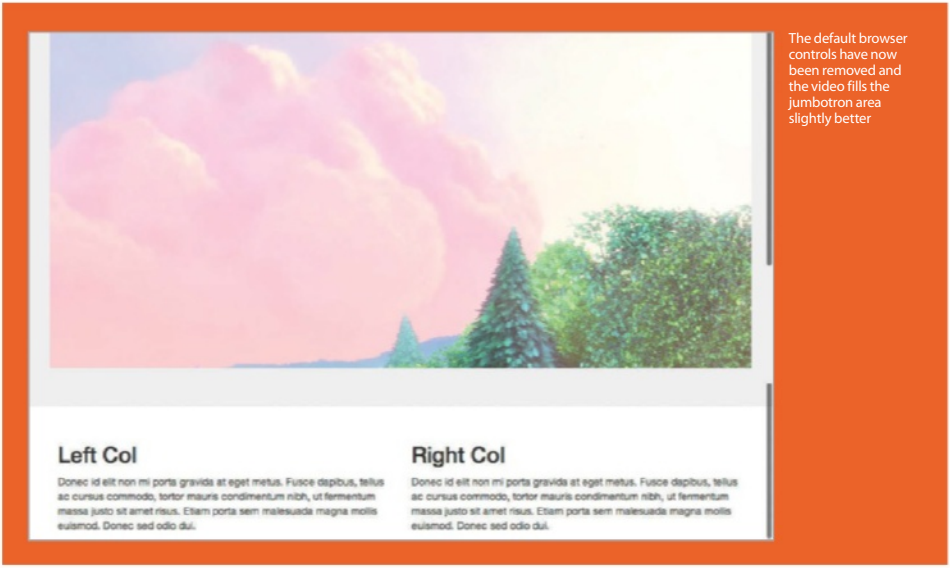
```
<source src="big_buck_bunny_1080p.ogg"
   type="video/ogg">
```

Getting responsive

07 Now it's time to make use of that large jumbotron area and fill it up with our video. Adjust your video tag height and width to the

By adding in a poster image, we can display a title screen that will disappear when the user presses Play

Create a HTML5 responsive video player



The default browser controls have now been removed and the video fills the jumbotron area slightly better

following, to stretch the video across the jumbotron container. The video will adjust its height accordingly, so the aspect ratio is maintained correctly.

```
<video width="100%" controls>
```

Remove spacing

08 Our jumbotron features grey padding around the video – let's get rid of that to make our video fill the entire space. Add a CSS block above your video tag with the following code within it. This will override the default Bootstrap CSS, but there is no need for !important tags.

```
<style type="text/css">
    .jumbotron {
        padding: 0;
        margin-bottom: 0;
    }
</style>
```

The poster element

09 You'll notice when we first load the page you're presented with a black video screen. Until you hover over it, it's not very obvious that it's a video at all. There are two ways around this, the first is to set the video to autoplay, or we can add the following code to add an intro slide.

```
    <video width="100%" controls
poster="poster.jpg">
```

Removing controls

10 Now we need to remove the standard HTML5 video player controls. This is easily done by updating the video tag, removing the controls element, as per the below. Keep in mind that while we're coding the new controls we will be unable to actually play the video.

```
<video width="100%"
poster="poster.jpg">
```

Get video elements

11 We will be using jQuery in order to add controls back into our player, simply because Bootstrap needs it as well. Add the following code block to the bottom of your page, just above the closing </body> tag. This will cast the video element onto a variable called 'videoPlayer' for use later on.

```
<script>
    $( document ).ready(function()
{
```

"This will cast the video element onto a variable called 'videoPlayer' for use later on"

```
003         var videoPlayer =
$('#videoPlayer');
004      });
005   </script>
```

Add an ID

12 You may have noticed in the previous step, we added a jQuery var that targets the ID of videoPlayer. The next step is for us to add this ID to our video tag to pair things up. Targeting based on ID is good if you have multiple video elements on a page that you want to all behave differently to one another.

```
001   <video width="100%"
poster="poster.jpg" id
      ="videoPlayer">
```

The controls

13 Now it's time to add the controls back in, or at least the HTML framework that will power them. This block of HTML adds all the controls in and over the next few steps we will hook them up via jQuery. The controls use the Bootstrap Glyphicons we mentioned previously. Add this under your video tag, within the jumbotron.

```
001         <div class="videoControls">
002         <a href="#"
class="btnPlay"><span
            class="glyphicon glyphicon-
play">
            </span></a>
003         <div class="videoTime">
004            <span class="current"></
span>/<span
               class="duration"></
span>
005         </div>
006         <a href="#"
class="btnMute"><span
            class="glyphicon glyphicon-
volume-up">
            </span></a>
007         <a href="#"
class="btnFullscreen">
            <span class="glyphicon
glyphicon-
            fullscreen"></span></a>
008      </div>
```

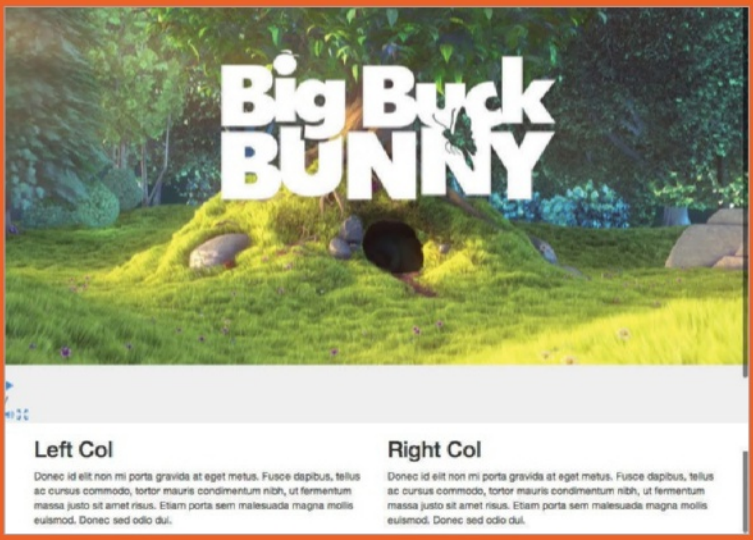
Hook up play

14 The first control we're going to hook up is the play button. This will be a multi-function button that will change from a Play button into a Pause button when activated. Add this code within your script tag, directly underneath your videoPlayer var declaration. Run it in your browser to test it out.

```
001   $('.btnPlay').click(function() {
002         videoPlayer[0].play();
003      })
```

Switch the button

15 Now we can play the video, we can switch the Play control to a Pause when it is active. We'll do this via an 'if' statement that updates the button class, replacing the Play glyphicon with the Pause icon. The 'if' statement will also resume playback and switch the icon back. Replace the



The video control HTML has now been added, although they are not currently functioning at all

Create a HTML5 responsive video player

previous step with the below.

```
001   $(‘.btnPlay’).click(function() {
002         if(videoPlayer[0].paused)
{
003            videoPlayer[0].play();
004            $(‘.glyphicon-play’).
attr(‘class’,
               ‘glyphicon glyphicon-
pause’);
005         }
006         else {
007            videoPlayer[0].pause();
008         $(‘.glyphicon-pause’).
attr(‘class’,
            ‘glyphicon glyphicon-
play’);
009         }
010         return false;
011      })
```

Display current time

16 We’ll now hook up the current time span via jQuery. You may have noticed the forward slash that is currently being outputted, this will serve as the break between the current time and the total time. Add the following code block underneath the previous step within your document ready script block.

```
001   videoPlayer.on(‘timeupdate’,
function() {
002         $(‘.current’).
text(videoPlayer[0].
            currentTime);
003      });
```

Less accuracy

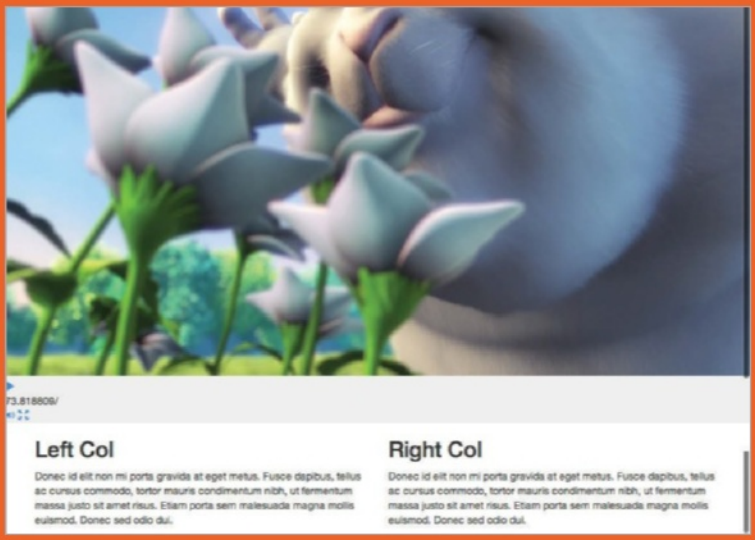
17 You may have noticed that our time includes milliseconds as well as seconds; this is probably a little bit too accurate for most situations. Altering the previous step to the following code will round that number up to the nearest second. You can extend this functionality to display hours, minutes and seconds as well.

```
001   $(‘.current’).text(Math.round
      (videoPlayer[0].currentTime));
```

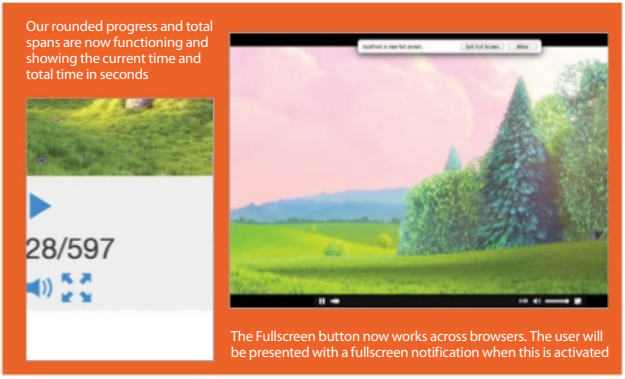
Total time

18 The next thing to do is to get the total duration of the video and display this next to our current time. Add this code underneath the previous step to hook up the span in our control . Again; you will need to round the number to get rid of the milliseconds.

> “Going fullscreen works slightly differently for different browsers, so there will be two jQuery commands”



The current time is now being displayed as a float, a bit of jQuery rounding will remove the decimal places




```
001      videoPlayer.
on('loadedmetadata',
         function() {
002         $('.duration').text(Math.
round
            (videoPlayer[0].duration));
003      });
004
```

Mute the audio

19 Combining everything that we have learned form the Play/Pause button, we can hook up the Mute/Unmute button with the following 'if' statement. Place this underneath the previous step, within the document ready bracket. This will switch the button class to the Mute icon, and back again when a user interacts with it.

```
001   $('.btnMute').click(function() {
002         if (videoPlayer[0].muted
== false) {
003            videoPlayer[0].muted =
true;
004            $('.glyphicon-volume-
up').attr('
               class', 'glyphicon
glyphicon-volume-
               off');
005         } else {
006            videoPlayer[0].muted =
false;
007            $('.glyphicon-volume-
off').attr('
               class', 'glyphicon
glyphicon-volume-
               up');
008         }
009      });
```

Go fullscreen

20 The final button we need to hook up is the Fullscreen button. Going fullscreen works slightly differently for different browsers, so there will be two jQuery commands that will do the exact same thing. Add this underneath the previous step and run your browser to see the results. Press ESC to exit fullscreen.

```
001   $('.btnFullscreen').on('click',
      function() {
002         videoPlayer[0].
webkitEnterFullscreen();
003         videoPlayer[0].
mozRequestFullScreen();
004         return false;
005      });
006
```

Add some style

21 Now we'll make our controls' appearance look a bit better by adding the following code within our CSS block from Step 8. When dealing with floating elements, don't forget to add a clear underneath to prevent other elements from being affected in weird ways. This code will space our elements out nicely.

```
001   .videoControls {
002         width: 100%;
003         font-size: 30px;
004      }
005      .videoControls span,
.videoControls div  {
006         display: inline-block;
007      }
008      .btnMute, .btnFullscreen {
009         float: right;
010         padding-left: 20px;
011         margin-right: 20px;;
012      }
013      .btnPlay {
014         padding-right: 20px;
015         margin-left: 20px;
016      }
```

Change the colours

22 The final stage is to amend the colour scheme and change it from the default Bootstrap blue. Feel free to adjust the colours to match your site's aesthetic. Add this code underneath the previous step within your CSS block. For a production environment it is recommended that you put all CSS and JS in separate files. There you have it! Your own custom responsive HTML5 video player is now ready to be shown off.

```
001   .videoControls span,
.videoControls div {
002         display: inline-block;
003         color: #999;
004      }
005      .btnMute, .btnFullscreen {
006         float: right;
007         padding-left: 20px;
008         margin-right: 20px;
009         color: #999;
010      }
011      .btnPlay {
012         padding-right: 20px;
013         margin-left: 20px;
014         color: #999;
015      }
016      .glyphicon:hover {
017         color: #555;
018      }
019
```

Essential web design resources

Get your hands on the best resources to help create a better web experience

365 PSD

365psd.com/

365 PSD is a brilliant resource. Get your hands on a free PSD file everyday. These include from UI kits, icon templates and much more. Simply select a PSD and download ready to use.

Apple Watch concept PSD

watch.janlosert.com/

The Apple Watch is a product that many will love. This set of PSDs allows for the creation of concept versions of your favourite websites and what they could look like on the much smaller Apple Watch screen. Simply download and start getting creative.

Awwwards

www.awwwards.com

A popular destination for those looking for some design inspiration. The site presents a Site of the Day, along with a Site of the Month, offering recognition to well-designed sites and designs efforts and talent. Plus, it offers expert opinion and insight on site featured.

Behance

www.behance.net/

Behance is a showcase for creative. If you have work you want the world to see than this is a great place to start. Plus, employers will be scouring the site looking for creative talent.

Color

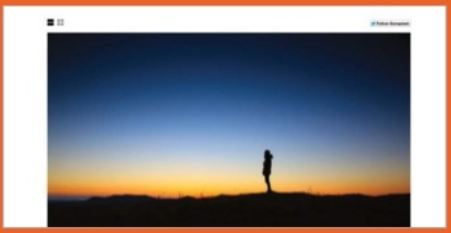
color.hailpixel.com

Choosing a colour is never easy but this simple to use site not only makes it easy it also makes it fun. Move the move cursor around the screen to pick a colour. Click again to pick another and so on to create a palette of choice. It allows you to clearly see if your chosen colours will clash.

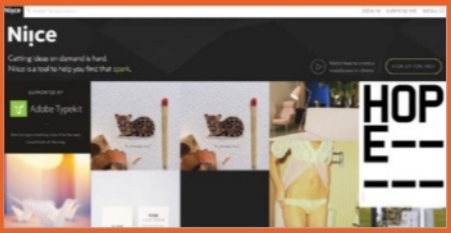
COLOURlovers

www.colourlovers.com

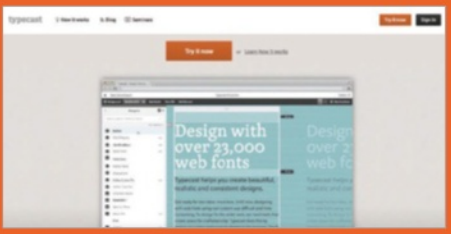
The color palette of a website can make a big difference to how it is viewed. To get an insight to the latest colour trends and a myriad of well defined palettes this site is a must visit.



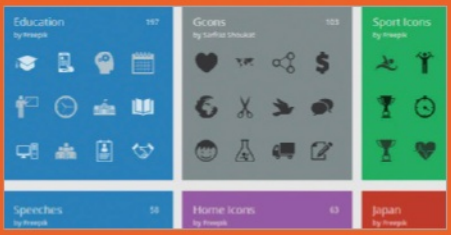
Get free hi-res images for projects



Search and find inspiration with Niice



Test out web fonts and typography



Get thousands of free vector icons

Dribbble

dribbble.com/

If in need of design inspiration then Dribbble should one of your first ports of call. Check out what other designers are doing and get 'inspired' .

Flaticon

www.flaticon.com/

Icons are great for quick, simple and neat visual representations. Getting the right set is important and Flaticon provides over 500 free sets to choose from. The icons are available in all the standard formats and can be used as web font icons.

Flat UI Pro

designmodo.com/flat/

Flat UI Pro is a flat UI kit that offers hundreds of basic components including icons and glyphs. These are perfect for a quick start on new designs and new projects.

Font Squirrel

www.fontsquirrel.com/

Font Squirrel brings together an impressive collection of fonts that are free for commercial use. Browse through the neatly defined collections and generate the code needed to add to a web design.

Google Developers Academy

developers.google.com/academy/

This site provides a set of online classes spanning many different Google developer tools and platforms including Maps and YouTube. The course materials are provided for developers of all skill levels with curriculum-based learning that goes beyond the usual technical documentation.

Google Fonts

www.google.com/fonts/

A great collection of free fonts optimised for the web. Download to test and try in designs. Alternatively, select a font and weights and use the generated code to use the font in live web design projects. You'll have thousands of fonts to hand instantly and license free.

HTML5 UP

html5up.net/

This is the place to be if looking to get a headstart on your latest design project. It offers a selection of contemporary page templates that are all screen friendly. These free designs can be downloaded and modified to match a specific design.

Niice

niice.co

Inspiration is an integral part of the design process and this niche search engine is here to help. Simply type in a keyword and Niice scours a host of popular inspirational/creative sites before returning plenty of visual inspiration to get your creative juices flowing freely.

Pixeden

www.pixeden.com

Thousands of quality web resources are available here. The site offers plenty of free graphics, icons, vectors, but its offers even more if you sign up to one of their premium plans.

Reddit web design

www.reddit.com/r/web_design/

The latest web news is interspersed with loads of titbits of web-related information. There are links to new resources, newly designed websites and useful tutorials.

Smashing Magazine

www.smashingmagazine.com/

Smashing Magazine is an online magazine for professional Web designers and developers. It focuses a host of different elements including techniques, best practices and valuable resources.

Typecast

typecast.com/

The Typecast platform allows users to create visual and semantic typographic designs. Then they can be checked for readability and rendering to make sure that they look like the designer intended. You can then share a working prototype.

Underscores

underscores.me/

The backbone of WordPress is the theme. There are thousands of free and premium themes available but if a custom theme is required, a starter theme such as Underscores is a great starting point.

Unsplash

unsplash.com/

High-quality photography can transform a web site and engage visitors. Unsplash provides a collection 'do whatever you want' images that have no restrictions.

W3Schools

www.w3schools.com/

This site covers all web design and development bases. If you want to learn HTML, CSS, JavaScript or PHP start here. There are hundreds of examples and explanations. Plus, there is an online editor that allows users to experiment with code and see the results.

Essential web design tools

Discover the best tools to help build a better web site more efficiently

3D CSS Text

http://www.3dcsstext.com/

Create 3D text with code via the extremely easy to use online tool. Simply pick a font, colours, angle, size, height and shadow and the code is automatically generated. Simply copy and paste to get your new design element onto your site.

Adobe Color CC

https://color.adobe.com/create/color-wheel/

Create your own colour palette with this simple but comprehensive colour creation tool. Choose a base colour, select a colour rule and fine tune to create the perfect palette for your project. It will allow you to see what works well and what doesn't.

Animate.css

http://daneden.github.io/animate.css/

Take the hard work out of creating CSS animations with this extensive library of 'Just-add-water' CSS animations. Try out animations before downloading the complete collection of code. Once happy, you can copy and past the code into your own project.

BigVideo

http://dfcb.github.io/BigVideo.js/

This is jQuery plugin that makes it easy to add fit-to-fill background video to websites. You can use it as a player to show video playlist or alternatively, you can use it for big background images. It creates brilliant results every time.

Bitly

www.bitly.com

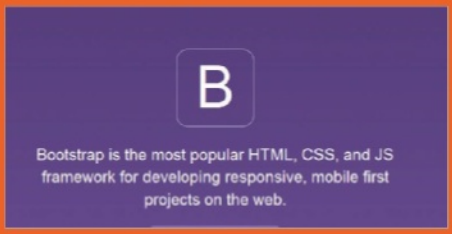
Bitly is a popular link shortener. It is ideal for shortening long links for anywhere where space needs to be saved. It's incredibly easy to use. Add link, shorten and copy and paste. You can sign up to the website in order to store all of your previous URL shortenings in one convenient place.

Bootstrap

http://getbootstrap.com/

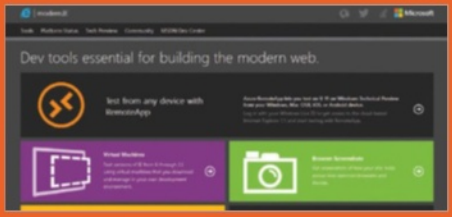
This is a popular and widely used HTML, CSS and Javascript framework for building responsive sites. It contains everything a user needs to create a site for desktop, tablet and smartphones. Get out our tutorial on page 82.

Create 3D with only CSS



A responsive framework to help build sites quickly

Quickly create the perfect colour palette



Get tools to test with Internet Explorer

Buffer

www.bufferapp.com

Buffer helps manage multiple social media accounts at once. Users can quickly schedule content from anywhere on the web, collaborate with team members, and analyze statistics on how posts perform.

Can I use

http://caniuse.com/

Need to know what HTML tags work in what browser? You do and this well-conceived and informative online tool will help tell you exactly what works where.

Codecademy

http://www.codecademy.com/

Need to brush up on your HTML and CSS? Want to know how to use jQuery? Sign up and learn how to code interactively with a host of free to access online courses.

CleanCSS

www.cleancss.com

Make your CSS better. CleanCSS is a powerful CSS optimizer and formatter. Basically, it takes your CSS code and makes it cleaner and more concise.

Ink Interface Kit

http://ink.sapo.pt/

Ink is an interface kit that provides a host of files to help create responsive web interfaces quickly and efficiently. It is based on a simple grid system and fully customisable.

Ionic

http://ionicframework.com/

This is a great tool for creating HTML5-friendly web apps. Free and open source, Ionic offers a library of mobile-optimized HTML, CSS and JS components, gestures, and tools for building highly interactive apps

IOS GUI

www.teehanlax.com/tools

Creative agency Teehan+Lax has put together its own collection of iOS UI templates. Get components for the iPhone 6, iPad, iOS7, iOS8 all in PSD format make them easy to customise in Photoshop.

JS Fiddle

www.jsfiddle.net

This online tool provides a test bed for any code. Split into separate windows for HTML, CSS and Javascript. Simply add code and run to see the result. When working perfectly drop the code into a web page and see the results.

Lipsum

http://lipsum.com/

If you need some placeholder text to fill out your pages then this is a great resource. Choose how many words, paragraphs or even bytes and lists. Then simply copy and paste into a page. It will allow you to figure out how much text you can input into your layout without it throwing off the design, or becoming too much like a wall of text for visitors.

Litmus

www.litmus.com

The complete email testing and analysing tool. See how your campaign emails will look, check so your emails don't become spam and analyse data to see if your emails are being read. A brilliant tool for companies that rely on newsletters, promotional emails and keeping up with their readership.

Macaw

http://macaw.co/

Macaw provides the same flexibility as an image editor, eg Photoshop, but also writes HTML and CSS.

Modern.IE

www.modern.ie

A host of development tools for testing the Internet Explorer browser. Make sure your sites will work with the browser on.

Namechk

http://namechk.com/

Want to know if the Twitter name you want is still available? This is the easy way to get an instant hit of availability for dozens of popular sites.

Normalize.css

http://necolas.github.io/normalize.css/

To get better all-browser consistency get this tool. Normalize.css makes browsers render all elements more consistently and in line with modern standards. It precisely targets only the styles that need normalizing.

Sublime Text

http://www.sublimetext.com/

Every web designer needs a text editor and Sublime Text is one of the best. Used by the pros, but still a great tool worth a try.

Webflow

www.webflow.com

For those who don't like, or perhaps know how, to code. Webflow is a drag-and-drop website builder for designing custom, professional websites without code. Its familiar GUI will make it easy for newcomers and designers.

Not just for dummies

for Beginners

A clear, comprehensive series for people who want to start learning about iPhone, iPad, Mac, Android and Photoshop

BUY YOUR COPY TODAY

Print edition available at www.imagineshop.co.uk

Digital edition available at www.greatdigitalmags.com

Available on the following platforms

Print Tablet Phone Desktop

facebook.com/ImagineBookazines twitter.com/Books_Imagine

HTML5 & CSS3

The Complete Manual

The essential handbook for web designers

- ✔ **Introducing HTML**
  Discover the history behind HTML5 and how it has developed into a brilliant language
- ✔ **Create your first page**
  Grasp the basics of HTML to create your very first web page within the <html> tags
- ✔ **All about CSS**
  Learn the basics of CSS coding and customise your HTML with different classes and styles
- ✔ **Terminology guides**
  With all the key terms in one place, understand the jargon of web design
- ✔ **Customise your website**
  Use CSS and HTML together to create a responsive and engaging website
- ✔ **Guide to responsive design**
  Understand all there is to know about how to adapt your website across devices
- ✔ **In-depth tutorials**
  Learn how to add menus, typography and video players to your site
- ✔ **Cross-browser use**
  Ensure that your website is compatible across multiple browsers including Chrome and Safari

ip IMAGINE PUBLISHING
DigitalEdition
GreatDigitalMags.com
VOLUME 01