makeuseof

GET STARTED WITH
HTML5!

BY MATTHEW HUGHES

By Mathew Hughes
http://www.matthewhughes.co.uk/

Published July 2013

share:

Think you've got what it takes to write a manual for MakeUseOf.com? We're always willing to hear a pitch!
Send your ideas to justinpot@makeuseof.com.

Table Of Contents

1. Introduction

You've heard of it: HTML5. Everybody is using it. It's being heralded as the savior of the Internet, allowing people to create rich, engaging web pages without resorting to using Flash and Shockwave.

But what actually is it?

Well, that's not an easy question to answer. HTML5 is used to describe a really diverse group of things. It's a standard of writing web pages. It's a collection of APIs. It's a new way of adding interactivity to web pages.

HTML5 is all that and more. So what's this book about?

I'm going to assume that you at some point have touched on HTML and CSS. Perhaps you've created your own Wordpress theme, or edited a MySpace layout back in the day. Perhaps you've read MakeUseOf's very own XHTML guide. The point is, I'm assuming that you know your way around a web page and that what we discuss in this guide won't be too alien to you.

The aim of this guide is not to teach you the entirety of HTML5. That would be entirely out of the scope of this book. The aim is to provide a gentle introduction to these amazing new web technologies, and to show you some cool ways of incorporating them into your websites.

Why would you want to learn HTML5?

It's a fair question. In a world of smartphones and apps, is it really important to learn how to program web pages?

Well, believe it or not, it's really common to write smartphone applications using HTML5 technologies. Until recently, the Facebook app for Android was written using HTML5, CSS and Javascript.

Blackberry is another major company that is immensely keen on HTML5. This is obvious in the latest iteration of their mobile operating system, Blackberry OS 10, where they are actively encouraging developers to develop applications for their phones using web technologies.

The new Firefox OS smartphones run entirely on HTML5 apps, too. A working knowledge of HTML5 is essential in today's smartphone climate.

In addition, learning HTML5 is good for your career. Don't believe me? According to Indeed.com, the average annual salary for a HTML5 developer is an eye-popping $89,000. With more and more companies changing their websites to use HTML5 technologies, developers who know the HTML5 stack are sought after – now more than ever.

1.1 Prerequisites

This guide assumes a couple of things. Firstly, it presupposes that you know how the web works, and that you know how to create a basic web page. You should be able to cobble some HTML elements together and to be able to present some information in a web browser. Seeing <div> and <p> tags isn't too daunting, and you're not afraid of getting your hands dirty in some source code.

Secondly, this guide assumes you know what CSS is and how it works. We don't expect you to be design geniuses, nor are you expected to know the entire CSS specification off the back of your hand. You should, however, be able to apply styling to an element on a web page, be able to link to a CSS file and know the different between an ID and a class and how to apply styling to each of them.

If you're scratching your head at the above, don't worry. One of the best things about HTML and CSS is that it's really, really easy. In fact, MakeUseOf has an incredible XHTML guide that will bring you up to speed really fast.

After reading that guide, you might also want to have a look at the following articles:

- *8 Websites With Quality Coding Examples*

- *6 Blogs To Follow Great Web Designers*

You're also going to need a modern text editor and browser. Any version of Internet Explorer that is older than IE 9 and

some older versions of Safari, Chrome and Firefox will struggle with many features that are part of HTML5 and may prevent you from following this guide.

As a result, you're encouraged to download a modern browser. I recommend Google Chrome, and I will be using it in each example.

Beyond that, all you're going to need is a willingness to learn. Oh, and a text editor.

1.2 Text Editors For Web Development

Your text editor is what you're going to use to write your code. You may be wondering what a text editor is.

Well, firstly it is not a word processor. Programs such as Microsoft Word and Apple's Pages are totally unsuited to web development. That is because they attach additional information to your HTML, CSS and Javascript files that makes it hard for your web browser to read.

A text editor shoots out characters into a text file, and not much else. This allows you to create files that have no extra formatting, and can be saved with any extension of your choosing.

Your computer already comes with one. If you're using a Windows PC, then Notepad is the text editor that you've likely got installed.

On a Mac, the situation is slightly different. OS X happens to come with four different text editors. These are called Vim, Emacs, Pico and Nano. However, unlike Notepad, all of them happen to work in the terminal.

This is a little bit intimidating for people who are new to web development and should not be used by people who are new to software development. We won't be using them in this guide. However, when you get a bit more confident with software and web development, it's definitely worth having a look at Vim and Emacs. They are both powerful text editors, and when mastered can save you an awful lot of time.

On Linux, the default text editor varies between distributions. On Ubuntu, it is likely Gedit, which is a rather pleasant text editor that is not too dissimilar from Notepad.

However, in this course we shall be writing our code using three different tools.

The first is Sublime Text 2. I honestly cannot recommend this highly enough. It comes with all the things that makes life easier for a beginning developer. Firstly, it'll make your code easier to read by coloring certain parts. Secondly, it allows you to switch between files easily and to manage entire projects of files. This is ideal for switching between files, and editing multiple bits of code on the fly.

The third is the Javascript console that is built into Google Chrome. This allows us to write Javascript and see it being run immediately and will be used to explain basic programming concepts.

The second is a website called Codepen.io. This remarkable website will allow you to code HTML, CSS and Javascript in the browser and is free to use. It will also allow you to see your changes instantly.

2. Semantic Markup

In this chapter, you will learn about Semantic Markup, and how to organize your code based upon its content.

Until recently, HTML code was generally organized with <div> tags. These allowed you to create a group of elements and then apply styling to those elements.

This worked, but there was room for improvement. The problem with <div> tags was that it wasn't semantic. Div doesn't actually mean anything, really.

Semantic markup is a new feature in HTML5. It brings in new tags, which work in the same way as a 'div' tag, but are for tagging common parts of a page.

So, how do they work? Consider the following code.

```
1   <!doctype html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <title>My Site</title>
6   </head>
7   <body>
8       <div id="header">
9       <h1>My favorite foods</h1>
10  </div>
11  <div id="navigation"
12      <a href="home.html">Home</a>
13      <a href="contactme.html">Contact</a>
14  </div>
15  <div id="food">
16      <ul>
17          <li>Quiche</li>
18          <li>Cheeseburgers</li>
19          <li>Hot Dogs</li>
20      </ul>
21  </div>
22
23  </body>
24  </html>
25
```

In this piece of code, we have a navigation bar, a title and a list. This isn't too dissimilar to most websites you're likely to ever go on, when you think about it.

Let's have a look an article on MakeUseOf. You'll notice that there is a part of the page that is reserved entirely for navigating to other articles. You'll also notice that there's another part of the page that contains the words that constitutes an article. Towards the top of the page, you'll see a header containing the MakeUseOf logo and some other links.

When you think about it, a lot of websites follow these conventions. Most websites have a part that is reserved for navigation. They usually have a body of content. They more than likely have a header.

Semantic tags are tags that allow you to define parts of a website which are commonly found on most websites. They don't add anything to the page, but allow you to group tags based upon their content and apply stylings to those groups.

So, remember that code we had before? Let's look at it with some Semantic markup added.

```
     index.html                          form.html           x       video.
 1   <!doctype html>
 2   <html lang="en">
 3   <head>
 4       <meta charset="UTF-8">
 5       <title>My Site</title>
 6   </head>
 7   <body>
 8       <header>
 9       <h1>My favorite foods</h1>
10   </header>
11   <nav>
12       <a href="home.html">Home</a>
13       <a href="contactme.html">Contact</a>
14   </nav>
15   <article>
16       <ul>
17           <li>Quiche</li>
18           <li>Cheeseburgers</li>
19           <li>Hot Dogs</li>
20       </ul>
21   </article>
22
23   </body>
24   </html>
25
```

As you can see, the code is much easier to read. You know which parts are which and there is no ambiguity. This is important, because it makes it easier to write good, clean code. Should you ever decide to become a professional web designer, this becomes paramount – you never know who will be reading the work that you produce.

So, let's look at some more semantic markup tags.

2.1 Section

Section is a really useful tag. It's used for grabbing huge swathes of information and content that are marked with a heading or a title. Think of this like a chapter in a book. A chapter has a title, and may also contain pictures, diagrams, graphs and words. A section tag would be used to contain all of that.

2.2 Article

The article tag is used for what it sounds like; Containing content such as a blog post or a news story. This content should be able to be detached from the rest of the blog and still make coherent sense.

2.3 Aside

This tag is reserved for content that is related to, but not an integral part of the web page. This could be a bunch of facts that relate to a news story, or the biography of a user on a blog.

2.4 Header

Lots of web pages have a bar on the top of the page that contains a logo, some information pertaining to the site and perhaps some links. In Semantic markup, you'd use a Header tag to contain all of this.

2.5 Nav

This element is reserved for the navigation part of your website. This would contain links to other websites or to other pages on the website. Within the context of MakeUseOf, this could be the part of the page that is below the header.

2.6 Footer

This tag is reserved for the bottom part of the page. Here, you could put some contact details, copyright information, a

map or some links to your 'about me' page.

2.7 Test Yourself

- *What is Semantic Markup, and what is it used for?*

- *I am making a web page and I want to use a semantic tag to contain a biography about me. Which one do I use?*

3. Forms

If you've ever done a bit of web design, you probably know how to create a simple form in HTML. If you're really clever, you probably know how to take the information you get from your form and how to do something with it, such put it into a database.

Forms are massively important. They are the basis of most of the things we do on the Internet. Every time you create a status update on your favorite social network, buy something from Amazon, or send an email, you've probably used a HTML form.

What you probably didn't know is that the way we create forms has radically changed in HTML5. It's also significantly better. In this chapter, we're going to look at some of the cool things you can now do, just with plain old markup.

So, what's so cool about the new way we can write forms in HTML5? Firstly, you can ensure that some fields must be filled in order to submit, just by changing the markup of the form itself. In addition, you no longer have to write mountains of JavaScript or PHP to do this. It trivially easy.

Secondly, you can ensure that your users can only submit certain types of information to your form. So, let's suppose you've got a website for your mailing list and you only want people to be able to submit actual email addresses? You can do that, just by using HTML5. It really is incredibly powerful.

Thirdly, you can make your forms look better by giving certain fields a placeholder. This will make them significantly more intuitive, as you can show your users an example of what you're expecting for a form.

3.1 Improving a Form

So, let's look at a form and see how we can make it better.

```
index.html          ●        form.html          ×

1   <!doctype html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <title>Document</title>
6   </head>
7   <body>
8       <form action="submit.php" method="post">
9           Text: <input type="text" name="first name">
10          Email: <input type="text" name="email">
11          Favorite Color: <input type="text" name="Color">
12          <input type="submit">
13      </form>
14  </body>
15  </html>
```

This form is pretty basic. It takes in a name, an email and a favorite color, and then allows the user to submit that. It contains no validation of what information gets placed into it, and there's nothing stopping users from submitting this form with some empty fields. Let's change all that.

So, the first thing we're going to want to do is ensure that the email field takes only an email. This used to be a pretty hard task indeed, as you'd have to create all sorts of arcane Regex code. Well, not any more. You just have to just change the type of the input from 'text' to 'email'. When you try to submit that form with gibberish, it'll complain and

insist that you submit an email.

3.2 Input Types and Patterns

There are other input types, which you can require. These include telephone numbers, web addresses, search forms, and even color pickers! As HTML5 is constantly evolving, it stands to reason that soon we'll be able to specify more input types in the near future.

Furthermore, for things such as phone numbers which vary depending on the locality, you can specify patterns for inputs. These are created using something called 'Regular Expressions' and are rather complicated, but immeasurably powerful.

We're also going to want to provide an example of an email in our field, so the user has no ambiguity of what he or she has to submit. That's really easy to do. Just create a new attribute of 'placeholder' with an example email address.

We are going to ensure that our 'Favorite Color' field is required. In the last angle bracket (>) in the Email input tag, just write 'required'. That's it. Now, when you try to submit your form without a value, it'll produce an error message.

The really incredible thing about these error messages is that the user doesn't have to write them or write any code to create them. You just change a field to make it required, and it just works. With that said, it is possible to customize them, should you want to.

That was an incredibly brief introduction to the power of forms in HTML5. If you wish to read more, I recommend that you visit these links.

Further Reading:

- *CSS Tricks – Let's write semantic markup*

- *HTML5 Doctor – Let's Talk About Semantics*

3.3 Test Yourself

It's your birthday next week, and you want to create a registration form so you know how much cake you need to create. Open up your text editor and create a form with the following fields.

- *Name*

- *Email address*

- *Phone number*

- *Allergies*

Ensure that the name, email and phone number fields are mandatory, and that the Email and Phone Number fields are set with the 'email' and 'tel' input types. Create a placeholder for the allergy field with the value 'pollen, eggs, quiche'.

Play around with the form. Try submitting required fields as empty, and try insert non-numerical characters into the phone number field. In the email field, insert something that isn't an email address. What happens?

GET STARTED WITH HTML5!

4. Media

There used to be a time when the only way in which you could insert some video or audio into a web page was by using something like Flash, Shockwave or SilverLight.

This wasn't ideal. Firstly, none of these frameworks worked all that well on mobile devices. They just weren't equipped for the modern world of smartphones and tablets.

In addition, they were proprietary formats. As a result, users of Linux and OS X could get a pretty second-rate experience or were even prevented from consuming media services, as it was not available for their platform.

Finally, they had a propensity for being slow. If you were on a underpowered or older computer, you would not have a good experience viewing video using these frameworks. Flash was particularly notorious for this.

4.1 How HTML5 Makes Video and Audio Awesome

HTML5 changed this by allowing web developers to include video and audio into their web pages with just a few lines of code. It works a treat on mobile devices and works on every modern web browser.

As a result, major companies such as YouTube, Vimeo and Netflix are taking advantage of the HTML5 revolution. Why don't you join them?

4.2 All About Codecs

In this chapter, you're going to learn how to use the power of HTML5 to include audio and video in your web pages.

Firstly, I'm going to have to start of with a caveat. Whilst you can use HTML5 video in every modern web browser, it does not work the same across each web browser. The codecs used by each browser vary. In Internet Explorer, you're limited to using MP4 video. Chrome is a bit more generous and allows you to use WebM, MP4 and Ogg Theora video. Opera is a bit more restrictive and only allows you to use Theora and WebM video.

As a result, you've got to be a little bit clever with how you insert video into your web page. So, let's see how it works.

4.3 Starting With Video

To start with, you're going to need to create some opening and closing <video> tags. It is in here where you will link to your video files. But first, you're going to want to set a poster. What does that mean?

Well, when you're waiting for your video to load, the person visiting your site can see a picture that relates to the video. To do that, just give your video tags an attribute of 'poster' with a value of the image that you wish to link to. It should look like this.

HTTP://MAKEUSEOF.COM
MATTHEW HUGHES, HTTP://WWW.MATTHEWHUGHES.CO.UK
13
share:

The next thing we're going to want to do is create a fallback. What does this mean? So, suppose you're using one of the older, less awesome browsers out there. Lots of these older browsers do not support HTML5 video and therefore cannot play HTML5 video. You're going to want to leave them a message informing them that they're going to want to upgrade their browser and that until they do so, they will not be able to watch your video.

To do that, you just write your message inside of your video tags. Nothing else is required. Once you've done that, you're going to be left with some code that looks like this.

Now, let's add some video. I'm going to test this on Google Chrome, so I'm going to link to an MP4 film. To do that, I create a Source tag and give it an attribute of src which has a value of the video I want to include.

```
1   <!doctype html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <title>Document</title>
6   </head>
7   <body>
8       <video poster="poster.jpeg">
9           <source src="video.mp4" />
10          Your browser is too old to play this video. Upgrade your browser and then try again.
11      </video>
12  </body>
13  </html>
```

My page is now ready to be opened in my web browser. I've linked to a film which is really, really large and as a result, when opened one can only see the poster.

4.4 Adding Audio

Audio can be inserted into your web page in a manner that is very reminiscent of how we inserted video into our page.

Firstly, one creates some audio tags. These audio tags contain an attribute of 'controls'. This gives the user who visits the page the ability to pause, play rewind and fast-forward the audio that is being played.
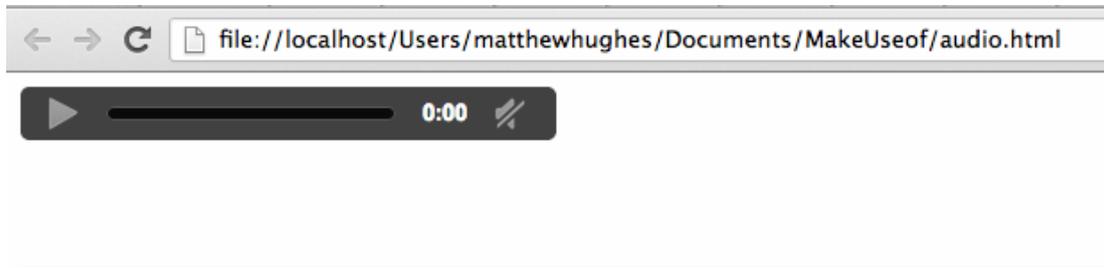
Then, you include a source tag to the MP3 file you wish to link to. You don't really have to worry all that much when it comes to codec compatibility. Most recent web browsers have the ability to play MP3 audio, although it's good practice to also include a '.ogg' and a '.wav' file – just in case.

Finally, you can create a fallback for older browsers. This is done in the same manner in which you created the fall-back for your video.

The end result looks a bit like this.

```
1   <!doctype html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <title>Document</title>
6   </head>
7   <body>
8       <audio controls>
9           <source src="piano.mp3"/>
10      </audio>
11
12  </body>
13  </html>
```

When you open this in your web browser, it should look a bit like this.

file://localhost/Users/matthewhughes/Documents/MakeUseof/audio.html

0:00

4.5 Test Yourself

- *What is the purpose of having a poster in your video tags?*

- *What codecs can you not use in Internet Explorer?*

- *If I wanted the ability to pause some audio, what attribute would you add to your 'audio' tag?*

Further Reading:

HTML5 Rocks Video

5. CSS3 Transformations And Animations

CSS traditionally was used to handle the layout and design of a web page. This is still true, but in its latest iteration it has gained the ability to handle animations and transformations of elements and images.

People have done some amazing things with CSS3, from creating a digital clock to writing a full Pong game. Someone even used it to re-create the introductory credits to Mad Men. It's a truly powerful technology and when it is mastered can be used to add an amazing level of functionality to your web page.

In this chapter, I'm going to give you a brief introduction to CSS3, and show you how to add some amazing effects to your page.

First, navigate to codepen.io and create a new pen. We're going to use this as our workspace for the duration of this chapter.

We're going to start simple and create a simple image transform that rotates an image 3 degrees when hovered. First of all, create a div tag and give it an ID. In the example below, I've given it an ID of 'muo'.

5.1 CSS Hover Effects

In that div, include an image of your choice. I've included a copy of the logo for MakeUseOf.

You'll then need to write some stylesheet rules. In the example below, I've created a top and left margin to give the image some room. I've also included a curious looking stylesheet rule that starts with '#muo:hover'. What is that?

When you attach ':hover' to a stylesheet rule, be it to an element, an ID or a class, you're effectively telling the browser to apply this styling when your mouse governs the element. Pretty cool, right?

Inside the '#muo:hover' rule, we've got a line that says '-webkit-transform: rotate(3deg)'. As I'm sure you've guessed, this is telling the browser to rotate that div element by three degrees.

However, it's worth noting that this tag only works in Chrome and Safari. If you want your code to work in Firefox or

Internet Explorer 9 and above, you're going to want to change your CSS file to include the following lines.

```
CSS                                          ⤢  ⚙

#muo:hover {
    transform: rotate(3deg)
    -ms-transform: rotate(3deg);
    -webkit-transform: rotate(3deg);
}

#muo {
    margin-top: 30px;
    margin-left: 10px;

}
```

Now, when you hover over the image, it looks like this:

5.2 Using CSS3 To Resize Images

So, why stop there? Did you know that you can also use the 'transform' method to enlarge or shrink an image. Let's change our CSS file to include the following lines.

```
#muo:hover {
    transform: scale(1.5, 1.5);
    -ms-transform: scale(1.5, 1.5);
    -webkit-transform: scale(1.5, 1.5);
}
```

As you can see, we've now included a new transform rule, but this time we're telling it to do something called 'scale'. This is a really beautiful way to increase the size of an image. It takes two parameters (those numbers you see in-between those parentheses), and they represent the amount by which you increase the height and the width of the element.

As you can see from the code, I'm going to increase the size of the MakeUseOf div logo by 50%. You can test this works by hovering over it. You'll see that now the 'MakeUseOf' logo is now significantly more stretched.

This was a very gentle introduction to CSS3 transformations. Despite CSS3 being very new indeed, you can now see that you can do lots of very interesting manipulations with it.

5.3 Test Yourself

- *How do we apply a styling to an element when hovering?*

- *How do you rotate an image using CSS3?*

- *How do you scale an image using CSS3?*

- *What happens if you pass your transform method 'translate(50px, 50px)'?*

Further Reading:

HTML5 Rocks - Presentation

6. Just Enough Javascript

If you want to use script in your web browser, you have to use [Javascript](). There's no two ways about it, sadly. It's a [language that has many fans](), and many detractors too. As languages go, it has many warts. There's a reason the most notable book about the language is called 'Javascript: The Good Parts'.

It'll be impossible to teach you how to use Javascript in a single chapter. That's not the aim here. The aim is to teach you enough Javascript so that you can understand the next chapter, which is about using a technology called Canvas to make drawings and animations.

6.1 Accessing The Console

To do this, we're going to use the Javascript console that is built into every copy of Google Chrome. To access this, you can right click on any web page and then press 'Inspect Element'. Then click on 'Console'. You should see this.


It's traditional that the first program any budding developer ever writes is the 'Hello World' program. This is a simple program that prints the phrase 'Hello World', and not much else. In your console, type 'console.log("Hello world!");.
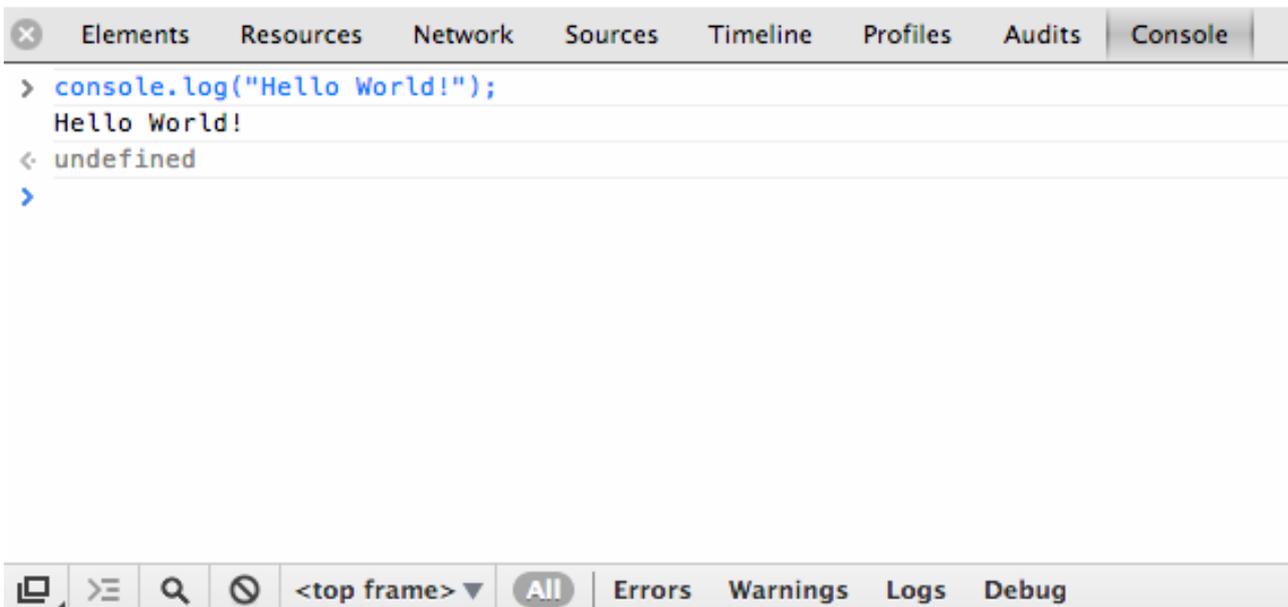

6.2 Your First Program

So, what exactly did we do? First off, we called something called 'console.log'. This is a bit of code that is built into the computer that simply prints out whatever you tell it to. We then attached some parentheses to it, and included the in double quotes 'Hello World'. This is called 'passing arguments', and the type of argument we passed is called a string. Whenever you want to do something involving letters and special characters, you simply must use single quotes. However, if you want to do anything using numbers, you usually don't need to use quotes, as seen below.
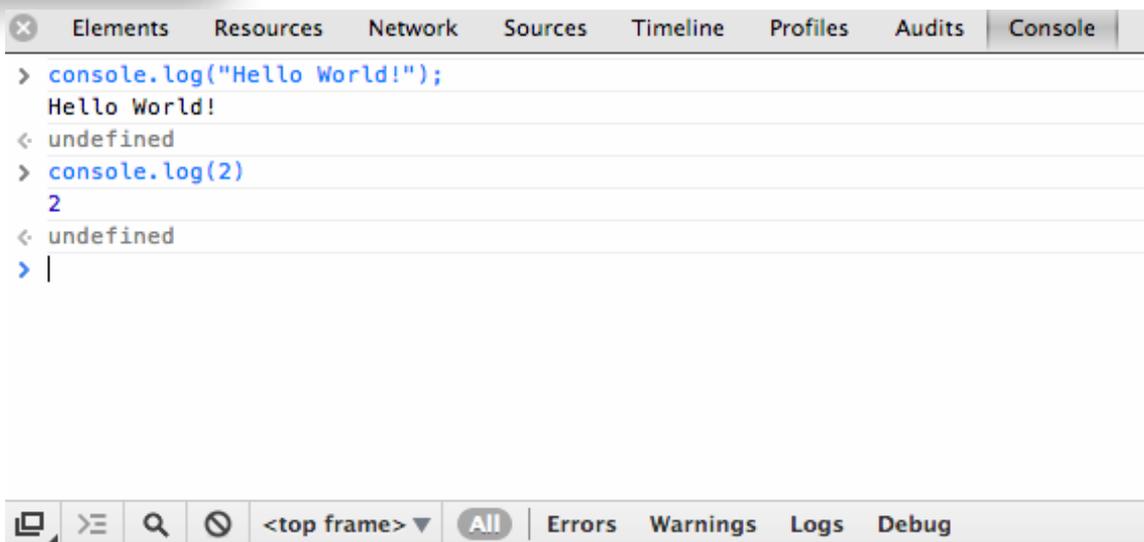
You can also pass variables to 'console.log' too. Variables sound complicated, but all they really are is a space to put chunks of information. These often are numbers or letters. To do that, you declare a variable using the 'var' keyword, give it a name and then with an equals sign, you give it a value. So, I'm going to create a variable called 'hello' and then give it a value of 'Hello World!'. I'm then going to pass that to console.log.
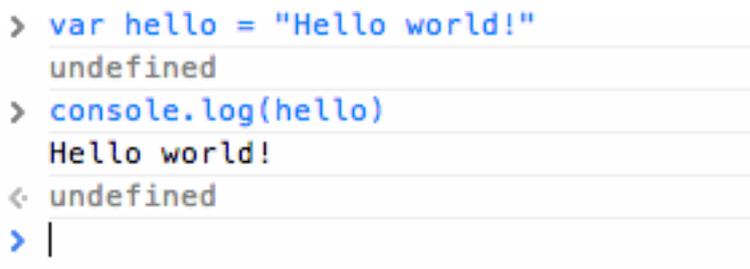
Note how I didn't pass 'hello' to console.log using quotes. That's because I wanted to print to the console the contents of 'hello' and not 'hello' itself.

6.4 What Functions Do

It can be a bit tedious rewriting the same chunk of code over and over again, so it is for this reason we write functions. Functions are easier than you think. All they are is chunks of code that we can reuse without rewriting the same code again. Below, we've created a function called 'sup' and are passing it an argument using parentheses which is then logged to the screen. We call 'sup' by sending to the console 'sup("Hello world!");'.
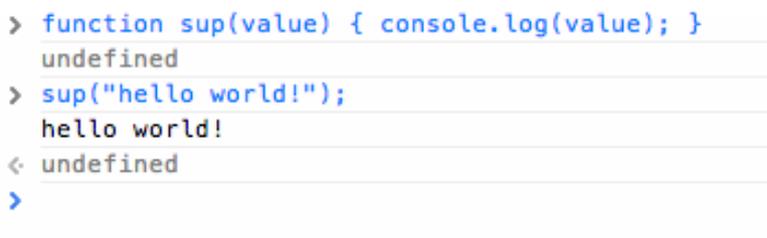
6.5 Repeating An Action With A 'For' Loop

Suppose you wanted to do the same action for a set number of times. It is for that reason why we'd use a 'for' loop. They look scary at first, but are so easy to do once you understand them. You start off by writing 'for()'.

In those parentheses, we're going to want to create a variable that counts how many times we have performed an action. So, we get something that looks like this 'for(var i = 0;)'.

We then want to check that i has not met a condition. So, in this case, we want to see that it's less than 10. So, after the semicolon, we write 'i < 10'. Our loop now looks like this: 'for(var i = 0; i < 10;)'.

If i is less than 10, we want to add it by one and then do something. So, we put 'i = i + 1'. Our loop is almost finished: 'for(var i = 0; i < 10; i = i + 1)'. Note how the last part does not have a semicolon.

After that, we're going to want to do an action. So, after the last parentheses, we write some curly braces and in-between them we we're going to console.log the value of i. This will create a counter that counts up to nine.

```
> for (var i = 0; i < 10; i = i + 1) { console.log(i); }
  0
  1
  2
  3
  4
  5
  6
  7
  8
  9
< undefined
>
```

The last two programming constructs we're going to look at are 'if' statements and 'while' loops.

6.6 If Statements

An 'if' statement performs an action if a certain criteria is met. They are similar to 'for' loops in construction, and work as follows. Suppose you have a variable called 'cheeseburgers' and you want to see if it has a value of 'tasty'. If it does, you want to log 'yum, cheeseburgers' to the screen. To do that you would write something like this.

```
> var cheeseburgers = "tasty"
  undefined
> if(cheeseburgers == "tasty") {console.log("yum, tasty cheeseburgers!")}
  yum, tasty cheeseburgers!
< undefined
> |
```

Note how I wrote 'if(cheeseburgers == "tasty")'. You use double or triple equals to check equality and single equals to assign a value.

6.7 While Loops

Finally, a 'while' loop executes an action whilst a criteria is met. So, imagine you want to log 'yum, cheeseburgers' whilst cheeseburgers equal tasty. To do that, you'd write the following.

```
   Elements   Resources   Network   Sources   Timeline   Profiles   Audits  | Console
> while(cheeseburgers = "tasty") { console.log("yum, cheeseburgers"); }
```

It's worth noting that this would enter an infinite loop, and you should avoid doing an action on a value that is not likely to change. This can cause your browser to lock up, or your code not to function.

As I mentioned before, this was a very brief introduction to programming constructs in Javascript. You are encouraged to read more about this fascinating, albeit huge subject.

6.8 Test yourself

- *I want to count down from 30. Write a 'for' loop that would do that.*

- *I want to create a variable called 'makeuseof' and give it a value of 'awesome'. How is that done?*

- *I want to create a function that prints 'MakeUseOf Is Awesome' when called. Write that function.*

Further Reading:

"Javascript : The Good Parts" by Douglas Crockford

MDN Javascript Guide

7. Creative Canvas

Canvas is a cool technology that allows you to draw images and create animations without having to resort to using Flash or Silverlight. People have used it to create bizarre and wonderful things, including a hairdryer simulator and various video games. It's a wonderful and unfathomably large piece of technology, in this tutorial, I'm going to give you a brief introduction to it.

It's worth noting that Canvas only works on modern web browsers. If you're using an old version of IE, Chrome or Firefox, you may not be able to follow this chapter. If that's the case, you should consider downloading the latest version of Google Chrome, which was the web browser in which I created this tutorial.

7.1 Getting Started With Canvas

First of all, you're going to need to open your web browser and navigate to codepen.io. Create a new pen.

Now, we're going to have to declare a canvas element. Create two opening and closing Canvas tags. In them, you should pass it three attributes. These are the width and height of the Canvas element, along with the ID you are giving it. Like before when you inserted some video, you should include a fallback message.

```js
var demo = document.getElementById('MakeUseOf')
var context = demo.getContext('2d');
context.fillStyle = 'red';
```

Now, we're going to want to write some Javascript code that will draw something to the screen. We're going to start basic and create a simple red square.

```js
var demo = document.getElementById('MakeUseOf');
var context = demo.getContext('2d');
context.fillStyle = 'red';
context.fillRect(30, 30, 50, 50);
```

We're going to create a variable (I called it 'demo'), and then select the canvas element and assign it to that variable. To do that, you use document.getElementByID() and pass in the ID of the element you wish to select.

The second line in our script creates another variable called 'context' and then calls 'demo.getContext('2d')' on it. This told the browser that we'll be working on a 2d image, then passed the necessary functions we'd need to in order to draw to the screen.

The third and fourth lines are the ones that actually do the drawing to the screen. The third line fills a rectangle with the color red, whilst the fourth line calls fillRect, which positions it and defines its length and width.

That's not impressive though. Let's do something a bit more advanced and use the magic of Javascript and Canvas to create MakeUseOf a brand new logo.

7.2 Shapes And Text

Let's delete our fourth line, and replace it with one that places our rectangle in the top left corner and stretches it out for the length of our canvas.

The first two arguments define where we wish to position the x and y axis of the shape. Let's set these two to '0' for now. The third argument refers to the width of the shape. Let's set that to '200', and then leave the fourth argument to

'50'. You now should have something that looks a little bit like this.

This is a great start, but it doesn't mention MakeUseOf at all. So, we're going to add some text. Let's create a variable containing 'makeuseof', and we'll call that variable 'MakeUseOf'.

We're then going to want to create another context variable. Call this one 'context2', and make sure that it's 2d. It is this that we will use to write our text in.

We're going to want our text to be colored blue and to overlay our red square. So, just like before, we're going to want to give it a fillStyle of 'blue'. Now, we're going to select the characteristics of our text. We want it to be 20px large, bold formatted and using an Arial font. We call font on context2 and assign it the value "bold 20px arial".

Because we want this text to overlay our previous red box, we need to call 'textBaseLine' on context2 and give it a value of top. Once that has been completed, we call 'fillText' on context2 and pass it the variable containing our text and the x and y coordinates in which we intend to place our text. The end result of our code is something like this.

The image that is produced by the code looks like this.

7.3 A Word On Canvas

Whilst this was an incredibly basic introduction to Canvas, you should understand that it is also an incredibly large technology, and an incredibly powerful one to boot. This guide simply served as an introduction to making graphics using this new technology.

7.4 Test Yourself

- *Add the following slogan to the image you created: "The best tech site ever!"*

- *Create a 'for' loop that runs for ten iterations. See if you can move your drawing down the canvas, a pixel at a time.*

- *Wrap your drawing in a function. What happens if you don't call it?*

Further Reading:

HTML5 Rocks – Integrating canvas into your web apps.

Treehouse – How To Draw With Canvas

8. Where Next?

Thank you for reading my incredibly brief guide to the new technologies found within HTML5. It's undeniable that HTML5 is the technology of the future. It is being adopted by most technology, as it is easy to write and powerful beyond measure. People are doing incredible things with it all the time, and I have no doubt that in the future, you will be one of those people. I'm honored to have been part of your journey into the wild and wonderful world of HTML5.

I implore you to keep on learning. Keep on coding. Continue to level up and improve, and in no time at all you will be using the technologies that have been introduced in this short guide to create wonderful products.

If you have any questions, please do not hesitate to contact me. My email addresses are me@matthewhughes.co.uk and mhughes@makeuseof.com

Guide Published: July 2013

Did you like this PDF Guide? Then why not visit MakeUseOf.com for daily posts on cool websites, free software and internet tips?

If you want more great guides like this, why not subscribe to MakeUseOf and receive instant access to 50+ PDF Guides like this one covering wide range of topics. More-over, you will be able to download free Cheat Sheets, Free Giveaways and other cool things.

Home:	http://www.makeuseof.com
MakeUseOf Answers:	http://www.makeuseof.com/answers
PDF Guides:	http://www.makeuseof.com/pages/
Tech Deals:	http://www.makeuseof.com/pages/hot-tech-deals

Follow MakeUseOf:

RSS Feed:	http://feedproxy.google.com/Makeuseof
Newsletter:	http://www.makeuseof.com/pages/subscribe-to-makeuseof-newsletter
Facebook:	http://www.facebook.com/makeuseof
Twitter:	http://www.twitter.com/Makeuseof

Think you've got what it takes to write a manual for MakeUseOf.com? We're always willing to hear a pitch! Send your ideas to justinpot@makeuseof.com.

Download Other MakeUseOf PDF Guides!

http://makeuseof.com/pages