

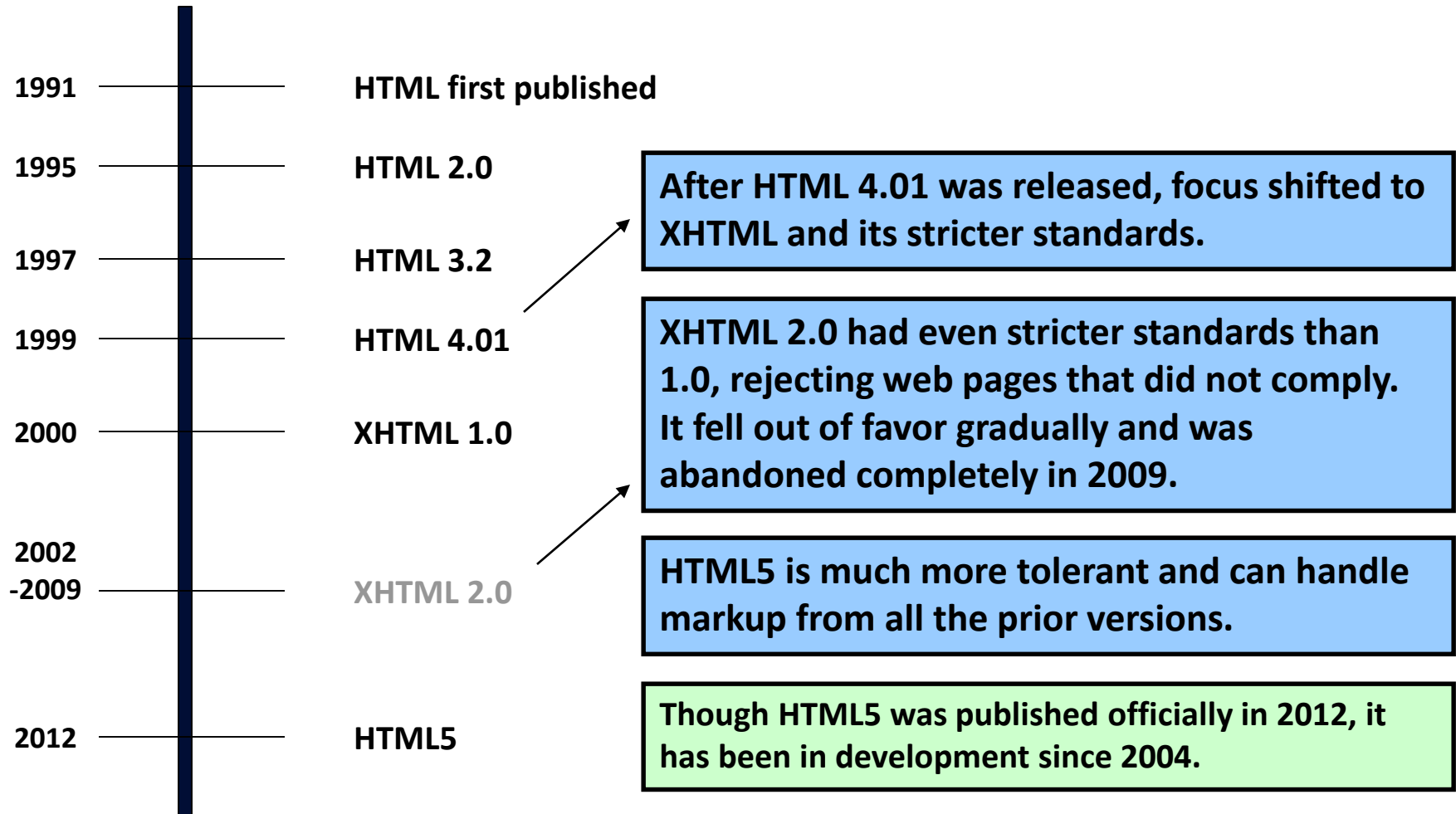
# Client Side Web Development

**Week 3**

**Intro to HTML5**



# History of HTML



# What is HTML5?

- **HTML5 is the newest version of HTML, only recently gaining partial support but most modern browsers have some HTML5 support web browsers.**
- **It incorporates all features from earlier versions of HTML, including the stricter XHTML.**
- **It adds a diverse set of new tools for the web developer to use.**
- **It is still a work in progress.**
- **No browsers have full HTML5 support.**
- **It will be many years – perhaps not until 2018 or later - before being fully defined and supported.**

# Why bother with HTML?

- **Plenty of good frameworks seem to be turning up on a regular basis**
  - Angular, React, Vue, Ionic and more ...
  - If I learn these will it be enough or is HTML still important?
    - » Especially when you consider that HTML isn't supported the same by browsers and the frameworks look after the 'missing' features using polyfills and wrapper libraries?
  - HTML is still evolving and may well do away with external 'helpers' as time goes forward
    - » e.g. Video and audio support as standard now built in

# HTML5: Origins

- **HTML5 is a cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG).**
- **WHATWG was working with web forms and applications, and W3C was working with XHTML 2.0. In 2006, they decided to cooperate and create a new version of HTML.**

# Goals of HTML5

- **Support all existing web pages. With HTML5, there is no requirement to go back and revise older websites.**
- **Reduce the need for external plugins and scripts to show website content.**
- **Improve the semantic definition (i.e. meaning and purpose) of page elements.**
- **Make the rendering of web content universal and independent of the device being used.**
- **Handle web documents errors in a better and more consistent fashion.**

# Some new Elements in HTML5

`<article>`

`<aside>`

`<audio>`

`<canvas>`

`<datalist>`

`<figure>`

`<figcaption>`

`<footer>`

`<header>`

`<hgroup>`

`<mark>`

`<nav>`

`<progress>`

`<section>`

`<source>`

`<svg>`

`<time>`

`<video>`

These are just some of the new elements introduced in HTML5. You will be exploring and hopefully using many of these during this module.

New elements list: <http://www.w3schools.com/html5/>

# HTML5: New Features

- **Canvas element for drawing**
- **Video/audio elements for media playback**
- **Better support for local offline storage**
- **New content specific elements, like article, footer, header, nav, section**
- **New form controls, like calendar, date, time, email, url, search**
- **Others include: Service web workers, Web components, Web Cryptography, WebGL 2**



# First Look at HTML5

Remember the DOCTYPE declaration from XHTML?

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

In HTML5, there is just one possible DOCTYPE declaration and it is simpler:

```
<!DOCTYPE html>
```

Just 15 characters!

The DOCTYPE tells the browser which type and version of document to expect. This should be the last time the DOCTYPE is ever changed. From now on, all future versions of HTML will use this same simplified declaration.

# The <html> Element

This is what the <html> element looked like in XHTML:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
      lang="en">
```

Again, HTML5 simplifies this line:

```
<html lang="en">
```

The lang attribute in the <html> element declares which language the page content is in. Though not strictly required, it should always be specified, as it can assist search engines and screen readers.

Each of the world's major languages has a two-character code, e.g. Spanish = "es", French = "fr", German = "de", Chinese = "zh", Arabic = "ar".

# The <head> Section

Here is a typical XHTML <head> section:

```
<head>
  <meta http-equiv="Content-type" content="text/html; charset=UTF-8" />
  <title>My First XHTML Page</title>
  <link rel="stylesheet" type="text/css" href="style.css" />
</head>
```

And the HTML5 version:

```
<head>
  <meta charset="utf-8">
  <title>My First HTML5 Page</title>
  <link rel="stylesheet" href="style.css">
</head>
```

Notice the simplified character set declaration, the shorter CSS stylesheet link text, and the removal of the trailing slashes for these two lines.

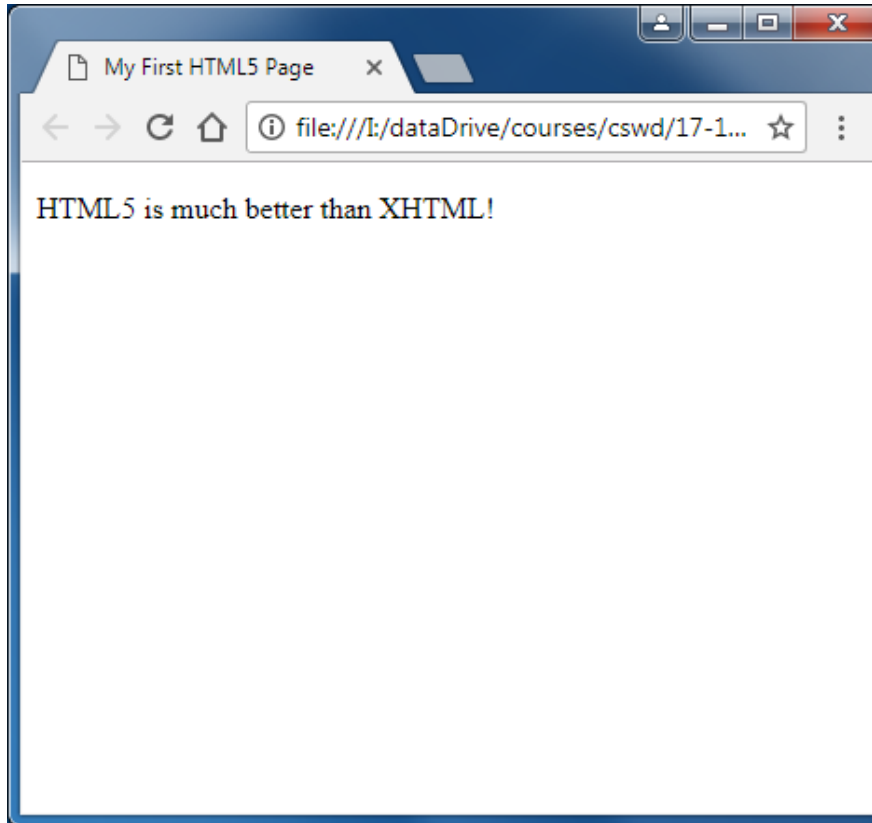
# Basic HTML5 Web Page

Putting the prior sections together, and now adding the `<body>` section and closing tags, we have our first complete web page in HTML5:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>My First HTML5 Page</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <p>HTML5 is much better than XHTML!</p>
</body>
</html>
```

**basicHTML5.html** : Let's open this page in a web browser to see how it looks...

# Viewing the HTML5 Web Page




Even though we used HTML5, the page looks exactly the same in a web browser as it would in XHTML. Without looking at the source code, web visitors will not know which version of HTML the page was created with.

# CSS3 Overview

- **CSS3 is not part of the HTML5 specification, but it is an integral part of writing HTML5 applications.**
- **CSS3 is being developed in tandem with HTML5 and provides many new styles to make web pages look and function better than ever.**
- **Things that were once the realm of Photoshop, such as gradients and shadows, are now easily added via styling:**
  - **Rounded corners**
  - **Gradients**
  - **Box and text shadows**
  - **Fonts**
  - **Box sizing**
  - **Stroke and outlines**
  - **Improved selectors**
  - **Box sizing**
  - **Transparencies**
  - **Improved selectors**
  - **Multiple background images and border images**
  - **Multiple columns and grid layout**
  - **Animation, movement and rotation**

# CSS Effect Example

```
.overkill {  
    border: 3px solid blue;  
    color: red;  
    font-size: 40px;  
    background-color: gold;  
    background: linear-gradient(45deg, #FF0000, #0000FF, #00FF00) ;  
    -webkit-border-radius: 40px  
    -moz-border-radius: 40px;  
    border-radius: 40px;  
    -webkit-box-shadow: 18px 18px 16px #ff3388;  
    -moz-box-shadow: 18px 18px 16px #ff3388;  
    box-shadow: 18px 18px 16px #ff3388;  
    text-shadow: 18px 18px 2px grey;  
    filter: dropshadow(color=#ff3388, offx=8, offy=8);  
}
```



# HTML5: Support

- You may well ask: "How can I start using HTML5 if older browsers don't support it?" But the question itself is misleading.
- HTML5 is not one big thing; it is a collection of individual features.
- You can only *detect* support for individual features, like canvas, video, or geolocation.



# HTML5: Support

- Love it or hate it, you can't deny that HTML 4 is the most successful markup format ever.
- HTML5 builds on that success.
- You don't need to throw away your existing markup.
- You don't need to relearn things you already know.
- If your web application worked yesterday in HTML 4, it will still work today in HTML5.

# HTML5: Support

- **Whether you want to draw on a canvas, play video, design better forms, or build web applications that work offline, you'll find that HTML5 is already well-supported.**
  - Firefox, Safari, Chrome, Opera, and mobile browsers already support canvas, video, geolocation, local storage, and more.
  - Google already supports microdata annotations. Even Microsoft — rarely known for blazing the trail of standards support — will be supporting most HTML5 features in the Internet Explorer 9 upwards.

# HTML5: Media types

- Every time your web browser requests a page from a web server it includes "header info"
  - the **User-Agent** identifies itself and the platform it is running on
- The server response includes a header that has a similar indication of its identity eg
  - Server: Apache/2.4 (xxx) PHP/6.1
- Headers are important, because they tell your browser how to interpret the page markup that follows.
- Another header, **Content-Type**, is a two-part specification, called **Internet Media Types**, of the data-type of the response's body
  - older name is **MIME** = **Multipurpose Internet Mail Extensions**
  - types and sub-types registered with Internet Assigned Number Authority (IANA)

# HTML5: MIME types

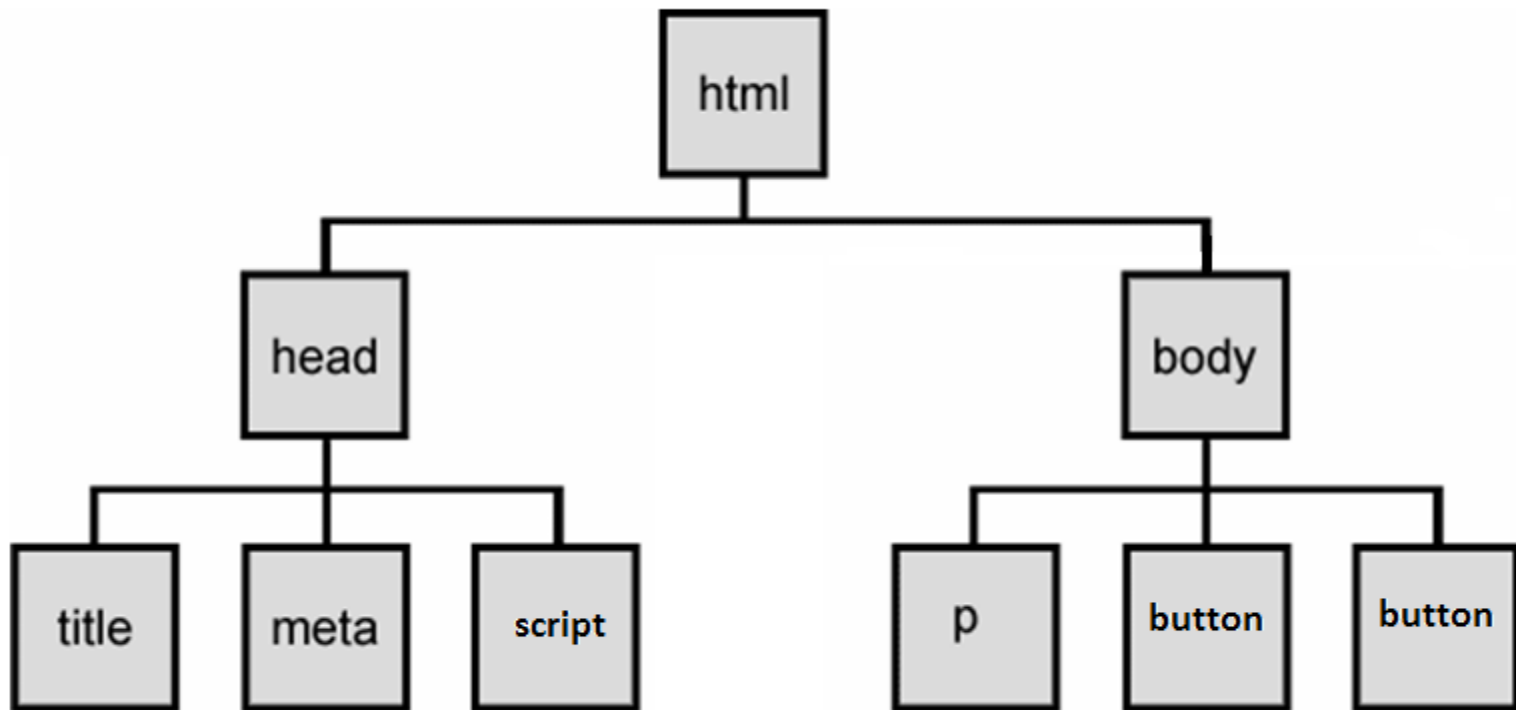
- The most important header is called Content-Type, and it looks like this:
  - Content-Type: text/html
- "text/html" is called the "content type" or "MIME type" of the page.
- Others are available for *application*, *audio*, *image*, *message*, *model*, *multipart*, *text* and *video*
  - image/jpeg for JPEG images, image/png for PNG images, text/css for CSS stylesheet etc
- This header is the only thing that determines what a particular resource truly is, and therefore how it should be rendered.
- Everything has its own MIME type, The web runs on MIME types.

# HTML5: Detection

- When your browser renders a web page, it constructs a **Document Object Model**, a collection of objects that represent the HTML elements on the page.
- The DOM provides a standard way in which every element on a document is represented in the DOM by a different object.
- In browsers that support HTML5 features, certain objects will have unique properties.
- A quick peek at the DOM will tell you which features are supported.

We'll definitely be coming back to the DOM in a later lecture!

# Sample DOM tree



**Q<sup>2</sup>. Draw up a skeleton html file that the above DOM could represent.**

# HTML5: Detection

- [Modernizr] is an open source, MIT-licensed JavaScript library that detects support for many HTML5 & CSS3 features.
- To use it, include the following `<script>` element at the top of your page...
- `<script src="modernizr.js"></script>`



Can include a path eg `src/js/modernizr.js`

# HTML5: Detection

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

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

```
  <title>HTML5 sure is fun</title>
```

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

```
</head>
```

```
<body>
```

```
  ...
```

```
</body>
```

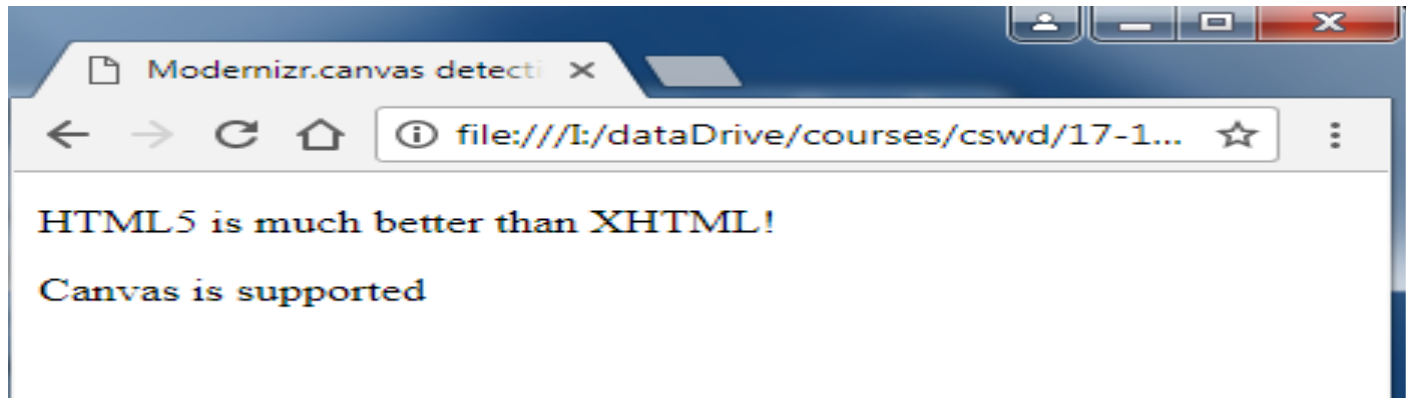
```
</html>
```

*When Modernizr runs, it creates a global object called Modernizr, that contains a set of Boolean properties for each feature it can detect.*



# HTML5: Detection

```
if (Modernizr.canvas) {  
    document.write("Canvas is supported");  
} else {  
    document.write("Canvas is not supported");  
}
```



More on HTML5 detection (and Modernizr) here:  
<http://diveintohtml5.org/detect.html>

# HTML5: Page structure elements

Some of the new markup elements are handy for creating better page structure.



# Semantic page elements

- new semantic elements in HTML5 are essentially the same as `<div>` elements.
- designed to give HTML markup more meaning than just wrapping everything in a `<div>` tag.
- new semantic elements include:
  - `<article>` : Defines an article in the document
  - `<aside>` : Defines content aside from the other page content
  - `<footer>` : Defines the footer for a section in the document
  - `<header>` : Defines the header for a section in the document
  - `<nav>` : Contains page navigation links
  - `<section>` : Defines a section in a document
- Use these tags to indicate the structure of your document so that the browser can render its content appropriately.

If interested in the [SEMANTIC WEB](#) then W3C has produced specifications of a language called [RDF \(Resource Description Framework\)](#) that is intended to allow metadata to be specified in a standard way.

# HTML5: Video

- **Until now, there hasn't been a standard for showing video on a web page.**
- **Today, most videos are shown through a plugin (like Flash).**
  - **However, not all browsers have the same plugins.**
- **HTML5 specifies a standard way to include video with the video element.**

# HTML5: Video

- Currently, there are 3 supported video formats for the video element:

Format	IE	Firefox	Opera	Chrome	Safari
Ogg	No	3.5+	10.5+	5.0+	No
MPEG 4	No	No	No	5.0+	3.0+
WebM	No	No	10.6+	6.0+	No

# HTML5: Video

```
<!DOCTYPE HTML>
```

```
<html>
```

```
<body>
```

```
<video src="test.mp4"  
width="320" height="240"  
controls="controls">
```

Your browser does not support the video tag.

```
</video>
```

```
</body>
```

```
</html>
```



# HTML5: Video

- The last example uses an mp4 file, and will work in Firefox, Opera and Chrome.
- To make the video work in Safari and future versions of Chrome, we must add an MPEG4 and WebM file.
- The video element allows multiple source elements.
- Source elements can link to different video files.
- The browser will use the first recognized format:

# HTML5: Video

```
<video width="320" height="240"  
controls="controls">
```

```
  <source src="movie.ogg" type="video/ogg" />
```

```
  <source src="movie.mp4" type="video/mp4" />
```

```
  <source src="movie.webm" type="video/webm" />
```

Your browser does not support the video tag.

```
</video>
```



# HTML5: Video

Attribute	Value	Description
audio	muted	Defining the default state of the the audio. Currently, only "muted" is allowed
autoplay	autoplay	If present, then the video will start playing as soon as it is ready
controls	controls	If present, controls will be displayed, such as a play button
height	<i>pixels</i>	Sets the height of the video player
loop	loop	If present, the video will start over again, every time it is finished
poster	<i>url</i>	Specifies the URL of an image representing the video
preload	preload	If present, the video will be loaded at page load, and ready to run. Ignored if "autoplay" is present
src	<i>url</i>	The URL of the video to play
width	<i>pixels</i>	Sets the width of the video player

# HTML5: Audio

- **Until now, there has never been a standard for playing audio on a web page.**
- **Today, most audio is played through a plugin (like Flash). However, not all browsers have the same plugins.**
- **HTML5 specifies a standard way to include audio, with the audio element.**
- **The audio element can play sound files, or an audio stream.**

# HTML5: Audio

- Currently, there are 3 supported formats for the audio element:

Format	IE 8	Firefox 3.5	Opera 10.5	Chrome 3.0	Safari 3.0
Ogg Vorbis	No	Yes	Yes	Yes	No
MP3	No	No	No	Yes	Yes
Wav	No	Yes	Yes	No	Yes

# HTML5: Audio

```
<!DOCTYPE HTML>
```

```
<html>
```

```
<body>
```

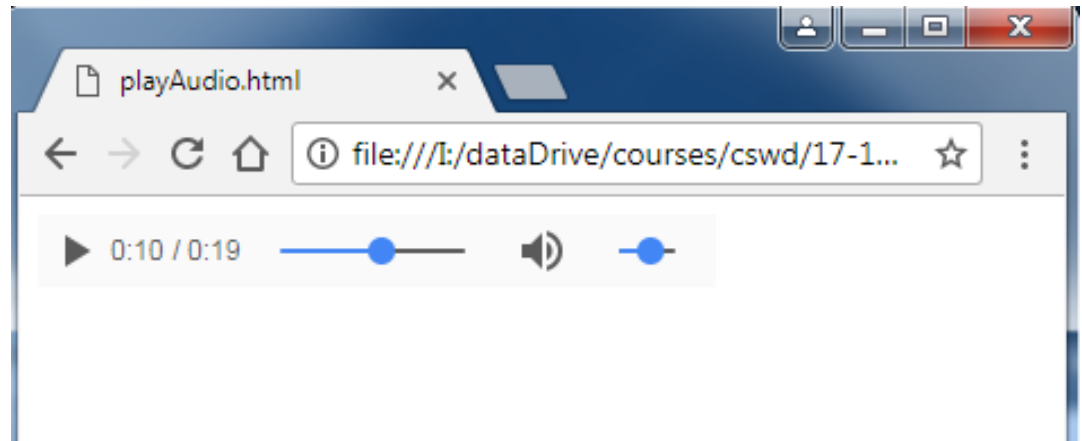
```
  <audio src="test.mp3" controls="controls">
```

Your browser does not support the audio element.

```
  </audio>
```

```
</body>
```

```
</html>
```



# HTML5: Audio

- The last example uses an mp3 file, and will work in Firefox, Opera and Chrome.
- To make the audio work in Safari, the audio file must be of type MP3 or Wav.
- The audio element allows multiple source elements.
- Source elements can link to different audio files. The browser will use the first recognized format.

# HTML5: Audio

```
<audio controls="controls">
```

```
  <source src="song.ogg" type="audio/ogg" />
```

```
  <source src="song.mp3" type="audio/mpeg" />
```

Your browser does not support the audio element.

```
</audio>
```

# HTML5: Audio

Attribute	Value	Description
autoplay	autoplay	Specifies that the audio will start playing as soon as it is ready.
controls	controls	Specifies that controls will be displayed, such as a play button.
loop	loop	Specifies that the audio will start playing again (looping) when it reaches the end
preload	preload	Specifies that the audio will be loaded at page load, and ready to run. Ignored if autoplay is present.
src	<i>url</i>	Specifies the URL of the audio to play

# HTML5: Canvas

- The HTML5 canvas element uses JavaScript to draw graphics on a web page.
- A canvas is a rectangular area, and you control every pixel of it.
- The canvas element has several methods for drawing paths, boxes, circles, characters, and adding images.
- Adding a canvas element to the HTML5 page.
- Specify the id, width & height of the element:

```
<canvas id="myCanvas" width="200" height="100">  
</canvas>
```



# HTML5: Canvas

- The canvas element has no drawing capabilities of its own.
- All drawing must be done inside a JavaScript tag:

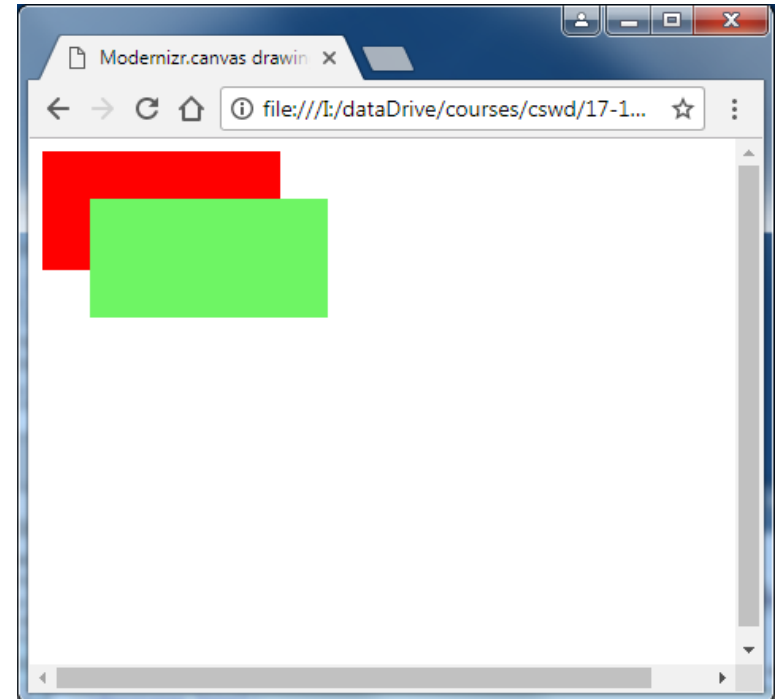
```
<script type="text/javascript">
    var
    c=document.getElementById("myCanvas") ;
    var ctx = c.getContext("2d") ;
    ctx.fillStyle="#FF0000" ;
    ctx.fillRect(0,0,150,75) ;
    ctx.fillStyle = "rgb(110,245,100)" ;
    ctx.fillRect (30,30,315,50) ;
</script>
```

```

<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Modernizr.canvas drawing - 08/09/17</title>
  <script src="./src/js/modernizr.js"></script>
  <script>
    function draw() {
      var canvas = document.getElementById("myCanvas");
      if (canvas.getContext) {
        var ctx = canvas.getContext("2d");
        ctx.fillStyle="#FF0000";
        ctx.fillRect(0,0,150,75);
        ctx.fillStyle = "rgb(110,245,100)";
        ctx.fillRect (30,30,150,75);
      }
    }
  </script>
</head>
<body>
<canvas id="myCanvas" width="480" height="320">
</canvas>

<script>
if (Modernizr.canvas) {
  // let's draw some shapes!
  draw();
} else {
  // no native canvas support available :(
  document.write("Canvas is not supported");
}
</script>
</body>
</html>

```



**drawCanvas.html**

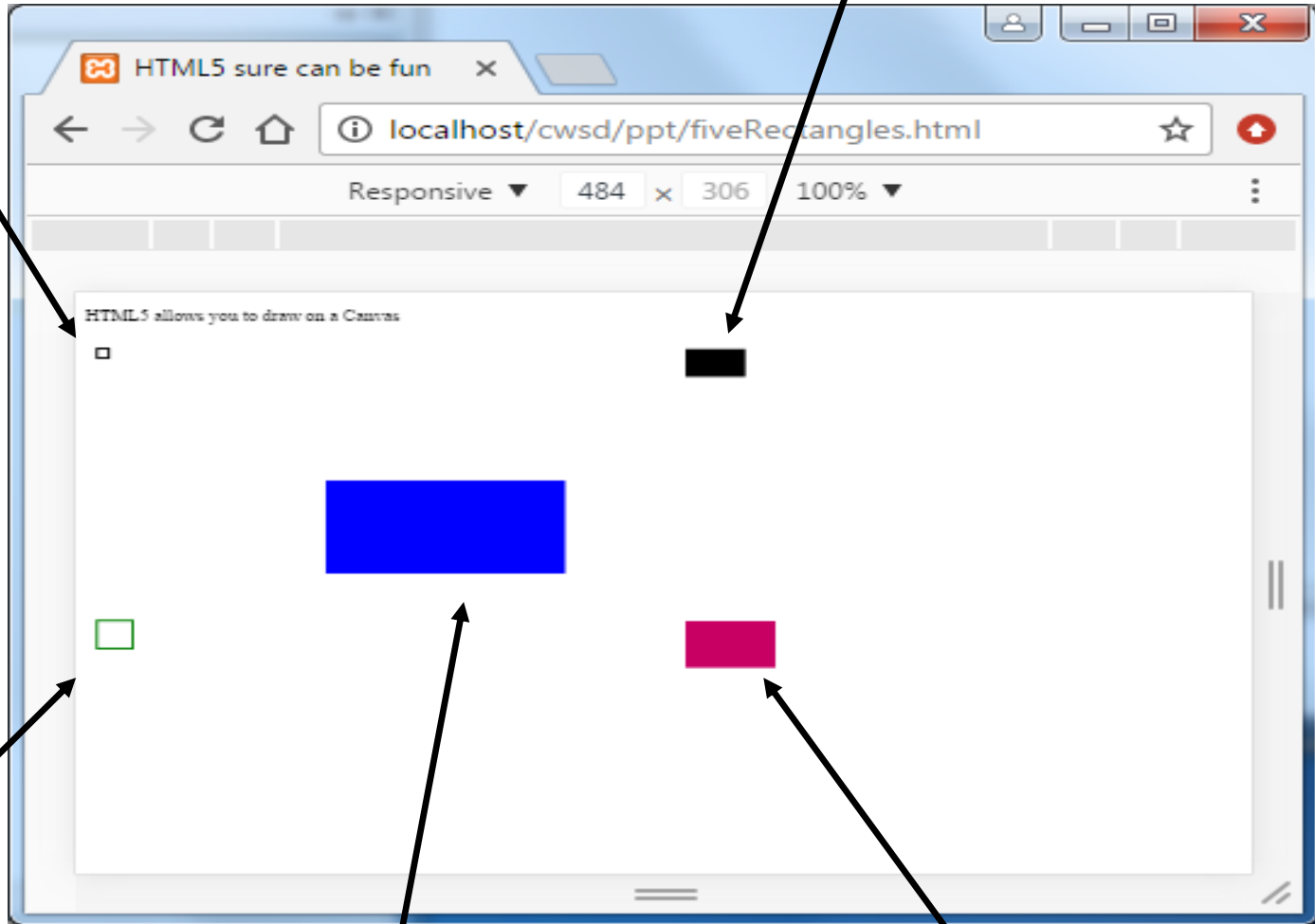
# 2<sup>nd</sup> set of projects: Drawing

- canvas element
- Use code to define a so-called context. Methods of this object do the work!
- Screen geometry: upper left corner is origin.
- colours defined by red-green-blue values or a small set of named colours,
  - [https://www.w3schools.com/colors/colors\\_hex.asp](https://www.w3schools.com/colors/colors_hex.asp)
  - stroke versus fill
- draw Rectangles

**CSWD**

10, 10, default colour,  
10 by 10, stroke

500,10,default colour,  
50 X 30, fill



10,300,green,30 by 30,  
stroke

fillStyle #0000FF (blue)  
200,150, 200 X 100, fill

rgb(200,0,100)  
500,300, 75 X 50, fill

# Draw Rectangles

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Four rectangles</title>
  <meta charset="UTF-8">
  <script src="/src/js/modernizr.js"></script>
  <script type="text/javascript">
    function init() {
      if (!Modernizr.canvas) {
        document.write("Canvas is not supported");
      } else {
        var ctx;
        ctx = document.getElementById('myCanvas').getContext('2d');
        ctx.lineWidth = 2;
        ctx.strokeRect(1, 1, 598, 378); // draw border
        ctx.strokeRect(10, 10, 10, 10);
        ctx.fillRect(500, 10, 50, 30);
        ctx.strokeStyle = "green";
        ctx.fillStyle = "rgb(200,0,100)";
        ctx.strokeRect(10, 300, 30, 30);
        ctx.fillRect(500, 300, 75, 50);
        ctx.fillStyle = "#0000FF"; // blue
        ctx.fillRect(200, 150, 200, 100);
      }
    }
  </script>
</head>
<body onload="init();">
<canvas id="myCanvas" width="640" height="380"></canvas>
</body>
</html>
```

# Web Storage (localStorage)

- **Storage capacity is 5MB per domain**
- **There's no expiration**
  - so must manually remove data when finished with
- **Any user with local privileges can bypass any authentication procedures implied by your code!**
  - Not recommended for storing sensitive info
  - XSS can be used to steal data or load malicious data in storage items
- **Watch out for HTML5 usage:**
  - `localStorage.setItem( "itemName", itemValue )`
  - `localStorage.getItem("itemName" )`

# Web Storage (localStorage demo)

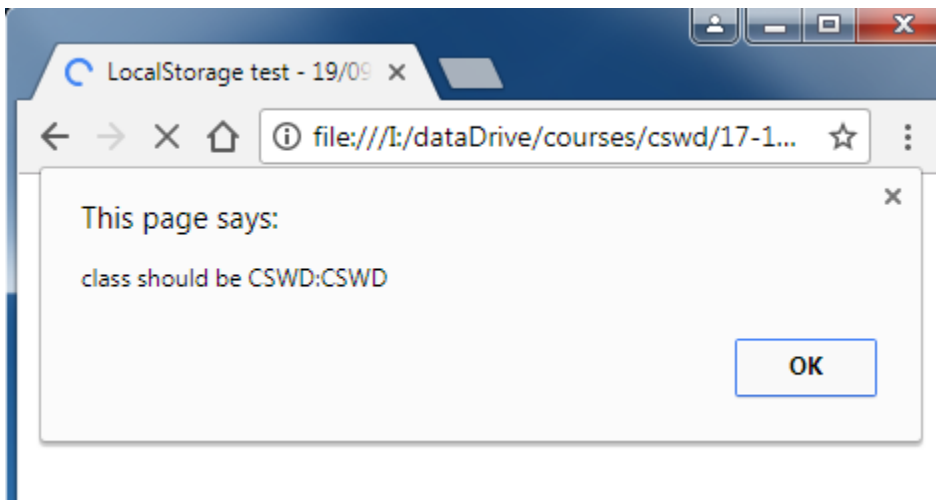
```
<script type="text/javascript">
var myDataLocalStorage = (function () {
    var getItem = function ( name ) { return
        localStorage.getItem(name); };
    var setItem = function (name, value) {
        localStorage.setItem(name, value); };
    var resetItem = function (name) {
        localStorage.setItem(name, ""); };
    return {
        get: getItem,
        set: setItem,
        reset: resetItem
    };
})();
</script>
```

**We'll come back to this pattern later!**

# Demo

- Possible usage:

```
myDataLocalStorage.set("class", "CSWD" );  
alert( myDataLocalStorage.get("class") );  
myDataLocalStorage.reset("class");  
alert( myDataLocalStorage.get("class") );
```



We'll come back to this when we do data storage stuff!



# HTML5 Manifest

- If using browser cache using the cache header, the browser does not respond properly when offline, as it does when online ☹
- HTML5 introduces a new way of programming which ensures that the browser cache can be used up to cache individual pages of the website which work in the same way even when offline.
- Application cache works with a **manifest** file e.g.  
`<html manifest="cache.appcache">`
- Here in this code, the HTML points to the cache file
  - We generally give the extension as appcache to the external header.
  - The content of this file is a plain text and will hold all the files that need to be used as application cache.

We'll come back to this when we do data storage stuff!

# HTML5: New Input types

- HTML5 has several new input types for forms.
- **email** - `<input type="email" value="foo@bar.com"/>`
- **url** - `<input type="url" value="http://www.gcu.ac.uk"/>`
- **number** - `<input type="number" value="23" min="0" max="100" step="1"/>`
- **range** - `<input type="range" value="20" min="0" max="100" step="10"/>`
- **date pickers (date, month, week, time, datetime )**  
`<input type="date" value="2017-10-02" min="2017-10-02"/>`
- **color** - `<input type="color" value="#FF0000"/>`
- **file (File API)** - `<input type="file" />`
  - The File API allows you to get information about, and load the contents of, files that the user selects.

# HTML5: Input types

Input type	IE	Firefox	Opera	Chrome	Safari
email	No	No	9.0	No	No
url	No	No	9.0	No	No
number	No	No	9.0	7.0	No
range	No	No	9.0	4.0	4.0
Date pickers	No	No	9.0	No	No
search	No	No	11.0	No	No
color	No	No	11.0	No	No

- **Note: Opera has the best support for the new input types.**
- **However, you can already start using them in all major browsers.**
- **If they are not supported, they will behave as regular text fields.**

# HTML5: Input - e-mail

- The email type is used for input fields that should contain an e-mail address.
- The value of the email field is automatically validated when the form is submitted.

- E-mail:

```
<input type="email" name="userEmail" />
```

- Tip: Safari on the iPhone recognizes the email input type, and changes the on-screen keyboard to match it (adds @ and .com options).

# HTML5: Input - url

- The url type is used for input fields that should contain a URL address.
- The value of the url field is automatically validated when the form is submitted.
- Homepage:  

```
<input type="url" name="userURL" />
```
- Tip: Safari on the iPhone recognizes the url input type, and changes the on-screen keyboard to match it (adds .com option).

# HTML5: Input - number

- The number type is used for input fields that should contain a numeric value.
- Set restrictions on what numbers are accepted:
- Points:

```
<input type="number" name="points" min="1" max="10" />
```

Attribute	Value	Description
max	<i>number</i>	Specifies the maximum value allowed
min	<i>number</i>	Specifies the minimum value allowed
step	<i>number</i>	Specifies legal number intervals (if step="3", legal numbers could be -3,0,3,6, etc)
value	<i>number</i>	Specifies the default value

# HTML5: Input - range

- The range type is used for input fields that should contain a value from a range of numbers.
- The range type is displayed as a slider bar.
- You can also set restrictions on what numbers are accepted:

```
<input type="range" name="points" min="1" max="10" />
```

# HTML5: Input – date pickers

- **HTML5 has several new input types for selecting date and time:**
  - **date** - Selects date, month and year
  - **month** - Selects month and year
  - **week** - Selects week and year
  - **time** - Selects time (hour and minute)
  - **datetime** - Selects time, date, month and year
  - **datetime-local** - Selects time, date, month and year (local time)



# HTML5: Input - search

- The search type is used for search fields like a site search or Google search.
- The search field behaves like a regular text field.

# HTML5: Input – colour picker

- The colour type is used for input fields that should contain a colour.
- This input type will allow you to select a colour from a colour picker:
- colour:

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

# HTML5: Input - form

- HTML5 has several new elements and attributes for forms.
- datalist
- keygen
- output

Attribute	IE	Firefox	Opera	Chrome	Safari
datalist	No	No	9.5	No	No
keygen	No	No	10.5	3.0	No
output	No	No	9.5	No	No

# HTML5: Form elements

- The **datalist** element specifies a list of options for an input field.
  - The list is created with **option** elements inside the **datalist**
  - You associate a **<datalist>** element with a text field by setting an ID on the **<datalist>** element, and referencing it with the **list** attribute of an **<input>** element.

```
<input type="text" list="colorList"/>
<datalist id="color-list">
  <option value="Red"/>
  <option value="Green"/>
  <option value="Blue"/>
</datalist>
```

- The purpose of the **keygen** element is to provide a secure way to authenticate users.
- The **output** element is used for different types of output, like calculations or script output:

# HTML5: Form attributes

Attribute	IE	Firefox	Opera	Chrome	Safari
autocomplete	8.0	3.5	9.5	3.0	4.0
autofocus	No	No	10.0	3.0	4.0
form	No	No	9.5	No	No
form overrides	No	No	10.5	No	No
height and width	8.0	3.5	9.5	3.0	4.0
list	No	No	9.5	No	No
min, max and step	No	No	9.5	3.0	No
multiple	No	3.5	11.0	3.0	4.0
novalidate	No	No	11.0	No	No
pattern	No	No	9.5	3.0	No
placeholder	No	No	11.0	3.0	3.0
required	No	No	9.5	3.0	No

# HTML5 Geolocation API

- Geolocation is widely supported by most modern browsers.
- The accuracy of the location depends on the capabilities of the user's device.
  - Devices that have GPS will give you a very accurate location, while those that don't will try to determine the user's location as close as they can by some other means, such as by IP address.
- The Geolocation API is accessed by using the `navigator.geolocation` object.
- To get the user's location you call the `getCurrentPosition()` method.
  - It takes two parameters- a callback function if it succeeds and a callback function if it fails:

```
navigator.geolocation.getCurrentPosition(  
    function(position) { alert("call was successful"); },  
    function(error) { alert("call failed" + error.message); }  
);
```

# Web Workers

- **Web workers provide a way to run JavaScript code in the background on a separate thread from the main thread of a web application.**
- **Often it seems like JavaScript is multithreaded because of its asynchronous nature, the truth is that there is only one thread.**
- **If you tie that thread up with a long running process, the web page will become unresponsive until it finishes.**
- **Problem is that it is easy to run into concurrency issues when working with threads**
  - **as one thread could be working on the same data as another thread, which could cause corrupted data, or even worse, deadlocks (race conditions)**
- **Fortunately web workers don't give us much of a chance to run into concurrency issues.**
  - **As web workers are not allowed to access non-thread safe components such as the DOM, the window or document**

# Summary

- **Plenty of new HTML5 features to get practice with**
  - Geolocation, web workers, web storage, html forms, canvas etc
- **Next Week:**
  - More JavaScript or DOM / AJAX/ jQuery