

Comp 388/424 - Client-side Web Design

Spring Semester 2016 - Week 3

Dr Nick Hayward

Contents

- HTML basics
- DOM basics
- HTML <head> element
- CSS

Adding a table

- organise data within a table starting with the `<table>` element
- three primary child elements include
 - *table row, table header, table data*
 - `<tr>`, `<th>`, `<td>`

```
<table>
<tr>
<th>header 1</th>
</tr>
<tr>
<td>row 1, cell 1</td>
</tr>
</table>
```

- also add a `<caption>`
- span multiple columns using the `colspan` attribute
- span multiple rows using the `rowspan` attribute

HTML Basics - <body> - part 6

Organising a list

- unordered list , ordered list , definition list <dl>
- and contains list items

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

```
<ol>  
<li></li>  
</ol>
```

- definition list uses <dt> for the item, and <dd> for the definition

```
<dl>  
<dt>Game 1</dt>  
<dd>our definition</dd>  
</dl>
```

HTML Basics - <body> - part 7

Using forms

- used to capture data input by a user, which can then be processed by the server
- <form> element acts as the parent wrapper for a form
- <input> element for user input includes options using the *type* attribute
 - *text, password, radio, checkbox, submit*

```
<form>  
Text field: <input type="text" name="textfield" />  
</form>
```

- process forms using
 - eg: *JavaScript, PHP...*

DOM Basics - intro

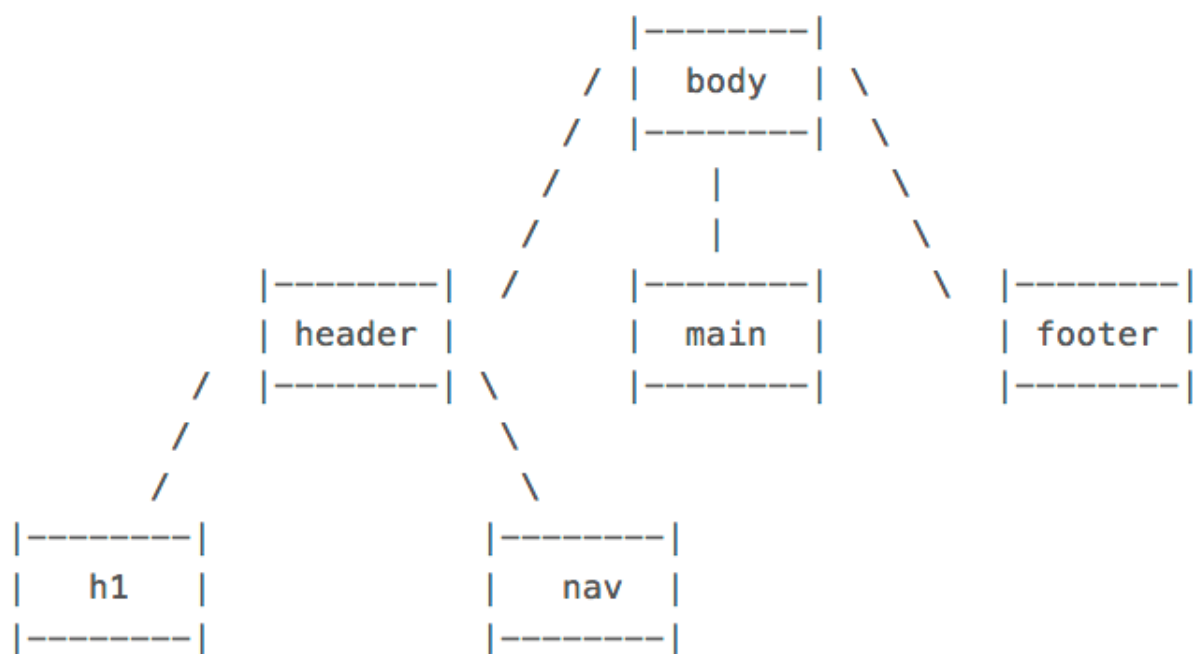
A brief introduction to the document object model (DOM).

This is particularly useful for CSS styling, and manipulating an underlying HTML document with JavaScript.

DOM Basics - what is DOM?

- presents a platform and language independent way to access and manipulate the underlying structure of our HTML document
- structured as a representation of a tree data structure
 - *its manipulation follows this same, standard principle*
- DOM tree is constructed using a set of nodes
 - *tree is designed as a hierarchical representation of the underlying document*
- each node on our tree is an element within our HTML document
- inherent hierarchical order originates with the *root* element
 - *at the top of our tree*
 - *descends down following lineage from node to node*
- each node is a child to its parent
 - *we can find many siblings per node as well*

Image - HTML DOM



HTML DOM

DOM Basics - useful elements

element tag	usage & description
<html>	container element for a HTML document
<head>	contains metadata and document information
<body>	contains main content rendered as the HTML document
<header>	page header...
<nav>	navigation, stores and defines a set of links for internal or external navigation
<main>	defined primary content area of document
<footer>	page footer...
<section>	a section of a page or document
<article>	suitable for organising and containing independent content
<aside>	defines content aside from the content which contains this element
<figure>	logical grouping of image and caption
	image - can be local or remote using url in src attribute
<figcaption>	image caption
<h1>, <h2>...	headings from 1 to 6 (1 = largest)
<a>	anchor - link to another anchor, document, site...
, , <dl>	unordered, ordered, definition lists
	list item, used with , ...
<dt>	definition term, used with <dl>
<dd>	definition description, used with <dl>
<table>	standard table with rows, columns...
<tr> >	table row, used with <table>
<th>	table heading, used with <table> and child to <tr>
<td>	table cell, used with <table> and child to <tr>
<div>	non-semantic container for content, similar concept to <section>
	group inline elements in a HTML document

<code><canvas></code>	HTML5 element for drawing on the HTML page
<code><video></code>	HTML5 element for embedding video playback
<code><audio></code>	HTML5 element for embedding audio playback

NB: *`<div>` and `` can be used as identifiers when there is no other suitable element to define parts of a HTML5 document.*

DOM Basics - sample

```
<!DOCTYPE html>
<html>
<head>
<base href="media/images/">
<meta charset="UTF-8">
<!-- week 3 - demo 1 -->
<title>Week 3 - Demo 1</title>
</head>
<body>
  <header>
    <h1>Ancient Egypt</h1>
  </header>
  <nav></nav>
  <main>
    <section>
      <p>
        Welcome to the Ancient Egypt information site.
      </p>
      <figure>
        
        <figcaption>Ptolemaic temple at Philae, Egypt</figcaption>
      </figure>
    </section>
    <aside>
      Temple at Philae in Egypt is Ptolemaic era of Egyptian history.
    </aside>
  </main>
  <footer>
    foot of the page...
  </footer>
</body>
</html>
```

- Demo 1 - DOM Sample

HTML - <head> element

- part of a HTML document's metadata
- allows us to set metadata for a HTML page
- customised just for that page or replicated as a site-wide implementation
- we can add numerous additional elements to <head>
- add similar links and code for JavaScript
 - use the <script> element & attributes such as *type* and *src*

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

- add a <title> element with text added as the element content
- set a default base address for all relative URLs in links within our HTML

```
<base href="/media/images/" target="_blank">
```

- links now simply use the base URL or override with full URL

```
  
<a href="http://www.flickr.com">Flickr</a>
```

HTML - <head> element example

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

  <title>Sample...</title>
  <meta name="description" content="sample metadata">
  <meta name="author" content="COMP424">

  <link rel="stylesheet" type="text/css" href="style.css">
  <script type="text/javascript" src="script.js">

</head>
```

CSS Basics - intro

- CSS allows us to define stylistic characteristics for our HTML
 - *helps us define how our HTML is displayed and rendered*
 - *colours used, font sizes, borders, padding, margins, links...*
- CSS can be stored
 - *in external files*
 - *added to a `<style>` element in the `<head>`*
 - *or embedded as inline styles per element*
- CSS not intended as a replacement for encoding semantic and stylistic characteristics with elements
- add a link to our CSS stylesheet using the `<style>` element.

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

- change will replicate throughout our site wherever the stylesheet is referenced
- embed styles per element using **inline** styles
 - *limitations and detractors for this style of CSS*
 - *helped by the growth and popularity of React...*

CSS Basics - pros

Pros

- inherent option and ability to abstract styles from content
- isolating design styles and aesthetics from semantic markup and content
- cross-platform support offered for many aspects of CSS
 - *CSS allows us to style once, and apply in different browsers*
 - *a few caveats remain...*
- useful frameworks such as **Bootstrap**
- support many different categories of device
 - *mobile, screen readers, print, TVs...*
- accessibility features

CSS Basics - cons

Cons

- still experience issues as designers with rendering quirks for certain styles
 - *border styles, wrapping, padding, margins...*
- everything is global
 - *CSS matches required selectors against the whole DOM*
 - *naming strategies can be awkward and difficult to maintain*
- CSS can become a mess very quickly
 - *we tend to add to CSS instead of deleting*
 - *can grow very large, very quickly...*

CSS Basics - intro to syntax

- simple, initial concepts for CSS syntax
- follows a defined syntax pattern, eg:
- selector
 - eg: *body* or *p*
- declaration
 - *property and value pairing*

```
body {  
  color: black;  
  font-family: "Times New Roman", Georgia, Serif;  
}
```

CSS Basics - rulesets

- a CSS file is a group of rules for styling our HTML documents
- rules form **rulesets**, which can be applied to elements within the DOM
- rulesets consist of the following,
 - a selector - *p*
 - an opening brace - *{*
 - a set of rules - *color: blue*
 - a closing brace - *}*
- for example,

```
body {  
  width: 900px;  
  color: #444;  
  font-family: "Times New Roman", Georgia, Serif;  
}
```

- [HTML Colour Picker](#)

CSS Basics - comments

- add comments to help describe the selector and its properties,

```
/* color can be set to a named value or HEX value (eg: #444) */  
p {  
  color: blue;  
  font-size: 14px;  
}
```

- comments can be added before the selector or within the braces

Image - CSS Syntax

Selector

```
|-----|  
|  p  |  
|-----|
```

Declaration

```
|-----|  
| { font-size: 14px; } |  
|-----|
```

^

|

property

^

|

value

CSS Syntax

CSS Basics - display

- display HTML elements in one of two ways
 - *inline* - eg: *a* or *span*
 - *displays content on the same line*

```
<div class="content">
  <p>
    <a href="...">Philae</a> is a <span>Ptolemaic</span> era temple in Egypt.
  </p>
</div>
```

- more common to display elements as `block-level` instead of `inline` elements
- element's content rendered on a new line outside flow of content
- a few sample block elements include,
 - *article*, *div*, *figure*, *main*, *nav*, *p*, *section*...
- *block-level* is not technically defined for new elements in HTML5

CSS Basics - inline elements

Current inline elements include:

- `b` | `big` | `i` | `small` | `tt`
- `abbr` | `acronym` | `cite` | `code` | `dfn` | `em` | `kbd` | `strong` | `samp` | `var`
- `a` | `bdo` | `br` | `img` | `map` | `object` | `q` | `script` | `span` | `sub` | `sup`
- `button` | `input` | `label` | `select` | `textarea`

Source - [MDN - Inline Elements](#)

CSS Basics - block-level elements

Current block-level elements include:

- address | article | aside | blockquote | canvas | dd | div | dl
- fieldset | figure | figcaption | footer | form
- h1 | h2 | h3 | h4 | h5 | h6
- header | hgroup | hr | main | nav | noscript
- ol | output | p | pre | section | table | tfoot | ul | video

Source - [MDN - Block-level Elements](#)

CSS Basics - HTML5 content categories - part I

- *block-level* is not technically defined for new elements in HTML5
- now have a slightly more complex model called **content categories**
- includes three primary types of content categories

These include,

- **main content categories** - describe common content rules shared by many elements
- **form-related content categories** - describe content rules common to form-related elements
- **specific content categories** - describe rare categories shared by only a small number of elements, often in a specific context

CSS Basics - HTML5 content categories - part 2

- **Metadata content** - modify presentation or behaviour of document, setup links, convey additional info...
 - `<base>`, `<command>`, `<link>`, `<meta>`, `<noscript>`, `<script>`, `<style>`, `<title>`
- **Flow content** - typically contain text or embedded content
 - `<a>`, `<article>`, `<canvas>`, `<figure>`, `<footer>`, `<header>`, `<main>`...
- **Sectioning content** - create a section in current outline to define scope of `<header>` elements, `<footer>` elements, and *heading* content
 - `<article>`, `<aside>`, `<nav>`, `<section>`
- **Heading content** - defines title of a section, both explicit and implicit sectioning
 - `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>`, `<hgroup>`

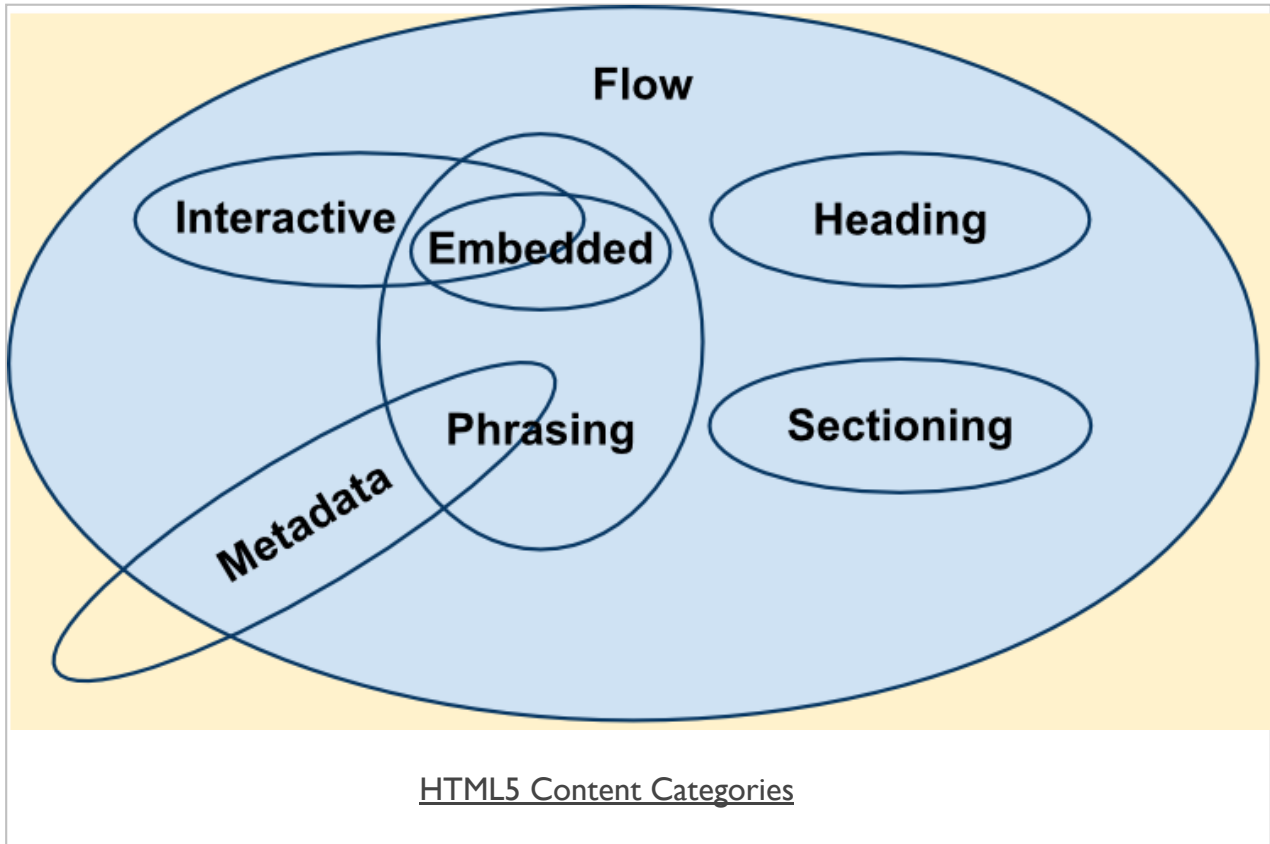
Source - MDN Content Categories

CSS Basics - HTML5 content categories - part 3

- **Phrasing content** - defines the text and the mark-up it contains
 - `<audio>`, `<canvas>`, `<code>`, ``, `<label>`, `<script>`, `<video>`...
 - *other elements can belong to this category if certain conditions are met. eg: `<a>`*
- **Embedded content** - imports or inserts resource or content from another mark-up language or namespace
 - `<audio>`, `<canvas>`, `<embed>`, `<iframe>`, ``, `<math>`, `<object>`, `<svg>`, `<video>`
- **Interactive content** - includes elements that are specifically designed for user interaction
 - `<a>`, `<button>`, `<details>`, `<embed>`, `<iframe>`, `<keygen>`, `<label>`, `<select>`, `<textarea>`
 - *additional elements, available under specific conditions, include*
 - `<audio>`, ``, `<input>`, `<menu>`, `<object>`, `<video>`
- **Form-associated content** - elements contained by a form parent element
 - `<button>`, `<input>`, `<label>`, `<select>`, `<textarea>`...
 - *there are also several sub-categories, including listed, labelable, submittable, resettable*

Source - MDN Content Categories

Image - HTML5 Content Categories



Source - MDN - Content Categories

CSS Basics - box model - part I

- consideration of the CSS box model
- a document's attempt to represent each element as a rectangular box
- boxes and properties determined by browser rendering engine
- browser calculates size, properties, and position of these required boxes
- properties can include, for example,
 - *colour, background features, borders, width, height...*
- box model designed to describe an element's required space and content
- each box has the standard four edges,
 - **margin** edge
 - **border** edge
 - **padding** edge
 - **content** edge

CSS Basics - box model - part 2

Content

- box's **content area** describes element's actual content
- Properties can include `color`, `background`, `img...`
 - *apply inside the **content** edge*
- dimensions include **content width** and **content-height**
- content size properties (assuming that the `box-sizing` property remains default) include,
 - *width, min-width, max-width, height, min-height, max-height*

Demo - CSS Box Model

- Demo 2 - Box Model

CSS Basics - box model - part 3

Padding

- box's **padding area** includes the extent of the padding to the surrounding border
- background, colour etc properties for a content area extend into the padding
 - *we often consider the padding as extending the content*
- padding itself is located in the box's **padding edge**
- dimensions are the width and height of the **padding-box**.
- control space between padding and content edge using the following properties,
 - *padding-top, padding-right, padding-bottom, padding-left*
 - *padding* (sizes calculated clock-wise)

Demo - CSS Box Model Padding

- [JSFiddle - Demo](#)

CSS Basics - box model - part 4

Border

- **border area** extends **padding area** to area containing the borders
- it becomes the area inside the **border edge**
- define its dimensions as the width and height of the **border-box**
- calculated area depends upon the width of the border we set in the CSS
- set size of our border using the following properties in CSS,
 - *border-width*
 - *border*

Demo - CSS Box Model Border

- [JSFiddle - Demo](#)

CSS Basics - box model - part 5

Margin

- **margin area** can extend this border area with an empty area
 - *useful to create a defined separation of one element from its neighbours*
- dimensions of area defined as width and height of the **margin-box**
- control size of our margin area using the following properties,
 - *margin-top, margin-right, margin-bottom, margin-left*
 - *margin* (sizes calculated clock-wise)

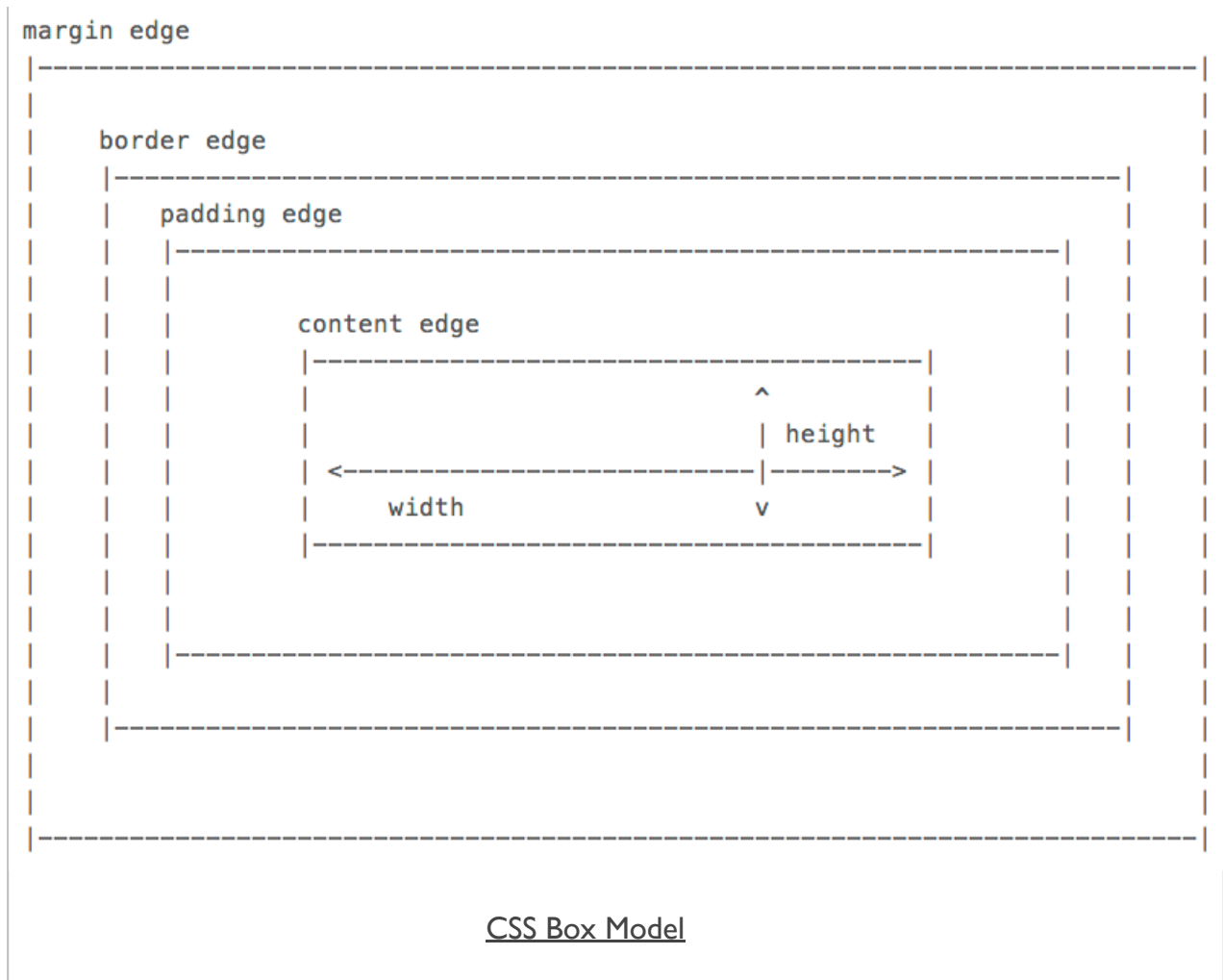
Demo - CSS Box Model Margin

- [JSFiddle - Demo](#)

Demo - CSS Box Model

- Demo 2 - Box Model

Image - CSS Box Model



Source - MDN - CSS Box Model

CSS Basics - selectors

- **selectors** are a crucial part of working with CSS
- basic selectors such as

```
p {  
  color: #444;  
}
```

- above ruleset adds basic styling to our paragraphs
 - sets the text colour to *HEX* value 444
- simple and easy to apply
 - applies the same properties and values to all paragraphs
- specificity requires classes, pseudoclasses...

CSS Basics - classes

- add a **class** attribute to an element, such as a `<p>`
 - *can help us differentiate elements*
- also add a **class** to any DOM element
 - *eg: add different classes to multiple `<p>` elements*

```
<p class="p1">paragraph one...</p>
<p class="p2">paragraph two...</p>
```

- we can now select our paragraphs by class name within the DOM
- then apply a **ruleset** for each class
- style this class for a specific element

```
p.p1 {
  color: #444;
}
```

- style all elements with the class p1, and not just `<p>` elements

```
.p1 {
  color: #444;
}
```


CSS Basics - pseudoclasses

- add a class to links or anchors, styling all links with the same ruleset
- we might also want to add specific styles for different link states
- styling links with a different colour
 - eg: *whether a link has already been used or not*

```
a {  
  color: blue;  
}  
  
a:visited {  
  color: red;  
}
```

- visited is a CSS **pseudoclass** applied to the <a> element
- browser implicitly adds this pseudoclass for us, we add style

```
a:hover {  
  color: black;  
  text-decoration: underline;  
}
```

- pseudoclass for link element, <a>, hover

CSS Basics - complex selector - part I

- our DOM will often become more complicated and detailed
- depth and complexity will require more complicated selectors as well
- lists and their list items are a good example

```
<ul>
  <li>unordered first</li>
  <li>unordered second</li>
  <li>unordered third</li>
</ul>
<ol>
  <li>ordered first</li>
  <li>ordered second</li>
  <li>ordered third</li>
</ol>
```

- two lists, one unordered and the other ordered
- style each list, and the list items using rulesets

```
ul {
  border: 1px solid green;
}
ol {
  border: 1px solid blue;
}
```

Demo - Complex Selectors - Part I

- Demo 3 - Complex Selectors Part I

CSS Basics - complex selector - part 2

- add a ruleset for the list items, ``
- applying the same style properties to both types of lists
- more specific to apply a ruleset to each list item for the different lists

```
ul li {  
  color: blue;  
}  
ol li {  
  color: red;  
}
```

- also be useful to set the background for specific list items in each list

```
li:first-child {  
  background: #bbb;  
}
```

- pseudoclass of `nth-child` to specify a style for the second, fourth etc child in the list

```
li:nth-child(2) {  
  background: #ddd;  
}
```

Demo - Complex Selectors - Part 2

- Demo 4 - Complex Selectors Part 2

CSS Basics - complex selector - part 3

- style odd and even list items to create a useful alternating pattern

```
li:nth-child(odd) {  
  background: #bbb;  
}  
li:nth-child(even) {  
  background: #ddd;  
}
```

- select only certain list items, or rows in a table etc
 - eg: every fourth list item, starting at the first one

```
li:nth-child(4n+1) {  
  background: green;  
}
```

- for **even** and **odd** children we're using the above with convenient shorthand
- other examples include
 - *last-child*
 - *nth-last-child()*
 - *many others...*

Demo - CSS Complex Selectors - Part 3

- Demo 5 - Complex Selectors Part 3

Next week - Quiz

Next week's quiz will include the following,

- everything covered in the first three weeks of the course
 - *includes today's lecture*
 - *further details on the course website as part of this week's assignment*

CSS - test and try out

- JSFiddle - Box Model

Demos

- Demo 1 - DOM Basics
- Demo 2 - Box Model
- Demo 3 - Complex Selectors Part 1
- Demo 4 - Complex Selectors Part 2
- Demo 5 - Complex Selectors Part 3

References

- [CSS Tricks - nth child recipes](#)
- [JSFiddle - CSS Basics](#)
- [MDN - CSS](#)
- [CSS box model](#)
- [MDN - HTML developer guide](#)
- [Block-level elements](#)
- [Content categories](#)
- [Inline elements](#)
- [Perishable Press - Barebones Web Templates](#)
- [W3 CSS](#)
- [W3 Schools - CSS](#)
- [W3 Schools - HTML Block and Inline Elements](#)
- [W3 Schools - HTML Colour Picker](#)
- [W3 Web Style Sheets - Even & Odd](#)