Comp 324/424 - Client-side Web Design

Spring Semester 2020 - Week 3

Dr Nick Hayward

HTML Basics - `<body>` - part 6

lists

- unordered list `<ul>`, ordered list `<ol>`, definition list `<dl>`
- `<ul>` and `<ol>` contains list items `<li>`

```
<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 & JS Example - add basic toggle to lists - HTML

```html
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>DOM Manipulation - Node Toggle</title>
  </head>
  <body>
    <header>
      <h3>DOM manipulation - Node Toggle</h3>
    </header>
    <section id="quote">
      <p>
        <blockquote id="berryhead" data-visible="true">
          Shine through the gloom, and point me to the skies...
        </blockquote>
      </p>
    </section>
    <section id="content">
      <h4>Planets...</h4>
      <div id="list-output">
        <span id="toggle">toggle...</span>
        <ul id="planets" data-visible="true">
          <li>Mercury</li>
          <li>Venus</li>
          <li>Earth</li>
          <li>Mars</li>
          <li>Jupiter</li>
          <li>Saturn</li>
          <li>Uranus</li>
          <li>Neptune</li>
          <li>Pluto</li>
        </ul>
      </div>
    </section>
    <!-- load JS files - pre module design example -->
    <script type="module" src="./toggle.js"></script>
  </body>
</html>
```

HTML & JS Example - add basic toggle to lists - JS Option 1

- various options for toggling visibility of DOM nodes
- option 1 relies on inefficient iterator of child nodes
- nodes may include elements, text, attributes…

```javascript
// toggle switch
let toggle = document.getElementById('toggle');

// get node in DOM
let domNode = document.getElementById('planets');

toggle.addEventListener('click', () => {
  // get child nodes
  let nodeChildren = domNode.children;
  console.log(nodeChildren);
  if (domNode.getAttribute('data-visible') === 'true') {
      domNode.setAttribute('data-visible', 'false');
      // modify display property for each child
      for (let child of nodeChildren) {
        child.style.color = '#779eab';
        child.style.display = 'none';
       }
    } else {
      domNode.setAttribute('data-visible', 'true');
      // modify display property for each child
      for (let child of nodeChildren) {
        child.style.color = '#000';
        child.style.display = 'list-item';
       }
    }
});
```

- Demo - List Toggle Children

HTML & JS Example - add basic toggle to lists - JS Option 2

- option 2 uses more efficient DOM properties to access required nodes

- access element nodes & ignores text &c. nodes in DOM...

```js
// toggle switch
let toggle = document.getElementById('toggle');

toggle.addEventListener('click', () => {
  // get sibling element to toggle...
  let siblingNode = toggle.nextElementSibling;

  // check child node element visibiity
  if (siblingNode.getAttribute('data-visible') === 'true') {
    // update visibility
    siblingNode.setAttribute('data-visible', 'false');
    // hide sibling node
    siblingNode.style.display = 'none';
  } else {
    // update visibility
    siblingNode.setAttribute('data-visible', 'true');
    // show sibling node
    siblingNode.style.display = 'block';
  }
});
```

- Demo - List Toggle Sibling
- JS Info - DOM Nodes

HTML & JS Example - add basic toggle to lists - JS Option 3

- add some initial animation

```
...
function slideUp() {
  if (slideHeight < 1) {
    console.log('slide up - height less than 1...');
    return;
  }
  slideHeight -= 2;
  siblingNode.style.height =  slideHeight + 'px';
  requestAnimationFrame(slideUp);
}

requestAnimationFrame(slideUp);
...
```

- Demo - Toggle vertical with animation
- Demo - Toggle horizontal with animation
- MDN - requestAnimationFrame

HTML Basics - `<body>` - part 7

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
  - *e.g. JavaScript…*
- MDN - HTML Forms

HTML - better markup

- web standards are crucial for understanding markup
  - *markup that goes beyond mere presentation*

- improved usage and structure, accessibility, integration…
- with standards, maintenance and extensibility becomes easier
- improved page structure and styling
  - *helps web designers and developers update and augment our code*

- poor markup usage
  - *to achieve a consideration and rendering of pure design*
  - *e.g. nesting tables many levels deep*
  - *adding images and padding blocks for positioning…*

- support for web standards continues to grow in popular browsers
- gives developers option to combine markup and styling
  - *HTML with CSS to achieve greater standards-compliant design*

HTML - markup and standards

- many benefits of understanding and using web standards, e.g.

- *reduced markup*
  - *less code, faster page loading*
  - *less code, greater server capacity, less bandwidth requirements…*

- *separation of concerns*
  - *content, structure, and presentation separated as needed*
  - *CSS used to manage site's design and rendering*
  - *quick and easy to update efficiently*

- *accessibility improvements*
  - *web standards increase no. of supported browsers & technologies…*

- *ongoing compatibility*
  - *web standards help improve chances of compatibility in the future…*

HTML - better structure

- consider *semantic* or *structured* markup
  - *within the context of app usage and domain requirements*

- trying to impart a sense of underlying meaning with markup
  - *correct elements for document markup*

- for a list
  - *use correct list group with list items - e.g. `ul`, `li`...*

- for a table
  - *consider table for data purposes*
  - *structure table & then consider presentation...*

- *semantic* markup helps create *separation of concerns*
  - *separate content and presentation*
  - *improves comprehension and usage*

Semantic HTML - intro

- importance of web standards
  - *and their application to HTML markup and documents*

- standards help drive a consideration of markup, e.g. HTML
  - *usage for what they mean*
  - *not simply how they will look...*

- semantic instead of purely presentational perspective
  - *introduction of meaning and value to the document*

- when pages are processed
  - *impart structure and meaning beyond mere presentation*

- a core consideration for usage of markup languages

- issues persist with HTML element usage
  - *e.g. inline elements such as `<b>` and `<i>`*

Semantic HTML - a reason to care

- Semantic HTML - opportunity to convey meaning with your markup
  - *meaning may be explicit due to the containing element*
  - *implicit due to a structured grouping of elements*

- markup makes it explicit to the browser
  - *underlying meaning of a page and its content*

- notion of meaning and clarity also conveyed to search engines
  - *fidelity with query and result...*

- semantic elements provide information beyond page rendering and design

- use semantic markup correctly
  - *create more specific references for styling*
  - *greater chance of rendering information correctly*

Semantic HTML - example usage

```html
<!-- incorrect element chosen -->
<div id="code">
document.addEventListener('click', function () {
  console.log('Click received...');
});
</div>
```

```html
<!-- correct element chosen -->
<code>
document.addEventListener('click', function () {
  console.log('Click received...');
});
</code>
```

- Demo - semantic example usage

Semantic HTML - correct usage

- need to ensure elements convey their correct meaning
  - *i.e. the meaning expected for the contained content*

- e.g. often see the following elements mis-used and applied incorrectly for markup,
  - *`<p>` - paragraphs*
  - *`<ul>` - unordered list*
  - *`<h1>` to `<h6>` - headings*
  - *`<blockquote>` - blockquote*

- using `<blockquote>` to simply help indent text
  - *instead of CSS margins...*

- or the perennial mis-use of a `<p>`
  - *simply add extra space between elements*

```
<p> <p>
```

Using lists correctly…

```
<li>nice</li>
<li>cannes</li>
<li>menton</li>
```

- list markup looks OK
  - *still fails validation for an obvious reason*
  - *missing structural grouping for list items*
  - *not valid markup…*
- semantics of the overall list are missing
- Demo - basic list items

HTML - a semantic point of view

```html
<ul>
  <li>nice</li>
  <li>cannes</li>
  <li>menton</li>
</ul>
```

- from the perspective of semantics
  - *meant to act as a group of items that belong together*

- denote such groupings with correct semantic markup

- structuring items to clearly denote their meaning and purpose

- consider global attributes
  - *https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes*

- Demo - basic group

HTML - benefits of structure & validation

- define and create a meaningful structure for required markup
  - *improves usage and flexibility as project develops*
  - *provides extensible structure for project*

- for example, benefits include
  - *helps increase ease of CSS styling*
  - *creates properly structured documents*
  - *improves general management of updates to markup*
  - *…*

- easier to understand and easier to maintain and update

- structured, valid markup aids in repurposing data
  - *into various representations of information*

e.g. a standard list

```
<ul>
  <li>nice</li>
  <li>cannes</li>
  <li>menton</li>
  <li>antibes</li>
  <li>grasse</li>
</ul>
```

- Demo - basic group style

e.g. lists for navigation, menus, tabs...

```html
<ul id="menutabs">
  <li><a href="nice">nice</a></li>
  <li><a href="cannes">cannes</a></li>
  <li><a href="menton">menton</a></li>
  <li><a href="antibes">antibes</a></li>
  <li><a href="grasse">grasse</a></li>
</ul>
```

- Demo - basic menu tabs

HTML - markup for headings - part 1

- HTML is flexible in markup usage
  - *due to presentational versus structural considerations*

- headings might be perceived as purely presentational, e.g.

```
<span class="heading">Chapter 1</span>
```

- issues with presentational markup, e.g.
  - *visual browsers with CSS will render as expected*
  - *no CSS, and browsers will render as normal text*
  - *non-visual browsers = normal text and no heading*
  - *accessibility issues...*

- search engines, ranking, spiders...
  - *will not process this markup as a heading*
  - *no semantic meaning...*
  - *recorded as normal text*

- CSS styles can be unique
  - *but restricted to `class` usage with `heading`*

HTML - markup for headings - part 2

- many different ways to markup content with HTML, e.g.

```
<p><b>Chapter 1</b></p>
```

- issues still exist with variant markup options, e.g.
  - *visual browsers will render text in bold & same size as default*
  - *unique styling is problematic…*
  - *search engines, ranking, spiders…*
    - will not process this markup as a heading
    - no semantic meaning…
    - recorded as normal text

HTML - markup for headings - part 3

- use markup correctly with structure and meaning, e.g.

```html
<h3>Chapter 1</h3>
```

- benefits of this markup, e.g.
  - *conveys meaning to contained text*
  - *visual and non-visual browsers treat heading correctly*
    - regardless of any associated styles…
  - *easy to add unique styles with CSS*
  - *search engines &c. will interpret this markup correctly*
    - extract keywords, semantics, structure…

HTML - markup for tables

- great example of poor usage of HTML markup is `<table>` element

- main issue is use of nested tables and spacer elements, images…

- if used correctly in structured markup
  - *tables can be very useful structure*
  - *impart a sense of semantic organisation to data*
  - *creating various interpretive information*

- what is a table for?
  - *structuring data*
  - *data to impart curated information…*

HTML - markup for tables - example 1

- simple table example - columns and rows for *presentation* purposes

```
<p>Travel Destinations</p>
<!-- basic table structure - minimal - rows and columns -->
<table>
  <tr>
    <td><b>Place</b></td>
    <td><b>Country</b></td>
    <td><b>Sights</b></td>
  </tr>
  <tr>
    <td>Nice</td>
    <td>France</td>
    <td>Cours Saleya</td>
  </tr>
  <tr>
    <td>Cannes</td>
    <td>France</td>
    <td>La Croisette</td>
  </tr>
  <tr>
    <td>Antibes</td>
    <td>France</td>
    <td>Picasso museum</td>
  </tr>
</table>
```

example

- Demo - basic table for presentation

HTML - markup for tables - example 2

- add semantic structure & elements to table caption - replace `<p>` with correct `<caption>` usage for a table...

```
<!-- basic table structure - minimal - add a caption -->
<table>
  <caption>Travel Destinations</caption>
  ...
```

- modern browsers style `<caption>` by default
  - *centred above the table*

- modify styling as required

example

- Demo - basic table caption

HTML - markup for tables - example 3

- add a summary attribute to the table

```
<!-- basic table structure - minimal - add summary attribute -->
<table summary="structured table data for travel destinations...">
  <caption>Travel Destinations</caption>
  ...
```

- add further meaning and structure to the table
  - *use of a summary attribute on the table element*
- processed by the browsers for semantics
- particularly useful for non-visual browsers

example

- Demo - basic table with summary

HTML - markup for tables - example 4

- add correct headers `<th>` to the table

```html
<!-- basic table structure - minimal - add table headers -->
<table summary="structured table data for travel destinations...">
  <caption>Travel Destinations</caption>
  <tr>
    <th>Place</th>
    <th>Country</th>
    <th>Sights</th>
  </tr>
  ...
```

Benefits include:

- remove need for presentational markup, bold elements
- visual browsers process structural and presentation qualities of headings
- such heading elements can also be useful for non-visual browsers

example

- Demo - basic table with headers

HTML - markup for tables - example 5

- table markup and accessibility markup…

```html
<!-- basic table structure - accessibility - add ids and headers -->
<table summary="structured table data for travel destinations...">
  <caption>Travel Destinations</caption>
  <tr>
    <th id="place">Place</th>
    <th id="country">Country</th>
    <th id="sights">Sights</th>
  </tr>
  <tr>
    <td headers="place">Nice</td>
    <td headers="country">France</td>
    <td headers="sights">Cours Saleya</td>
  </tr>
  ...
```

- creating a known relationship between the table's header, and its data
- a screen reader, for example, may read this table as follows,
  - *Place: Nice, Country: France, Sights: Cours Saleya*
- established a pattern to the output information for non-visual devices…

example

- Demo - basic table with accessibility

HTML - markup for tables - example 6

- add extra semantic markup for thead, tfoot, tbody…

```html
<!-- basic table structure - add head, foot, body -->
<table summary="structured table data for travel destinations...">
  <caption>Travel Destinations</caption>
  <thead>
    <tr>
      ...
    </tr>
  </thead>
  <tfoot>
    <tr>
      ...
    </tr>
  </tfoot>
  <tbody>
    <tr>
      ...
    </tr>
  </tbody>
</table>
```

- head and foot elements customarily go above the table body
  - *allows modern browsers, readers, &c. to load that data first*
  - *then render the main table content*

Benefits include:

- better underlying structure to data

- greater ease for styling a table due to clear divisions in data and information

- structural and presentational markup now working together correctly…

example

- Demo - basic table with head, foot, body

HTML - presentational vs structural

- consider *presentational* vs *structural*
  - *e.g.usage of quotations in markup*
  - *similar consideration to headings...*

- need to convey meaning and structure

- rather than a mere presentational instruction

- consider HTML's phrase elements
  - *e.g. `<cite>, <code>, <abbr>`*

- each phrase element imparts a sense of underlying meaning
  - *structure & then presentation...*

HTML - minimising markup

- noticeable benefit to creating sites with valid markup
  - *separation of structural from presentational*
  - *general reduction in required markup*

- simply conforming to the W3C's specifications
  - *does not inherently guarantee less code for your project*
  - *possible to include many unnecessary elements & retain valid markup*
  - *markup may still be valid*

- project issues may include:
  - *lack of efficiency*
  - *extraneous markup and code*

- to help minimise markup
  - *consider classes added to markup*
    - are there too many? are they all necessary? &c.
    - avoid class usage for unique reference
  - *avoid `<div>` usage for explicit block-level elements*

HTML5 - intro

- finally became a standard in October 2014

- introduces many new features to HTML standard

- additional features include, e.g.
  - *new canvas element for drawing*
  - *video and audio support*
  - *support for local offline storage*
  - *content specific elements*
    - including article, footer, header, nav, section

  - *form controls such as*
    - calendar, date, time, email, url, search

- new input type attribute values
  - *assigned to provide better input control*

- Check browser compatibility using HTML5 Test

HTML5 - basic template

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>

  </body>
</html>
```

HTML5 - Elements - part 1

- often known simply as **tags**

- elements allow us to add a form of metadata to our HTML page

- for example, we might add

```
<!-- a paragraph element -->
<p>add some paragraph content...</p>
<!-- a first heading element -->
<h1>our first heading</h1>
```

- this metadata used to apply structure to a page's content

HTML5 - Elements - part 2

- we can now add additional structure to our basic template

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <!-- title for the web page appears in the window, tab heading... -->
    <title>Demo 1</title>
  </head>
  <body>
    <h1>Our first web page</h1>
    <p>
      As we build our web apps, more elements and content will be added...
    </p>
  </body>
</html>
```

- Demo - Our first web page

HTML5 - Comments

- comments are simple and easy to add to HTML
- add to HTML code as follows,

```
<!-- a comment in html -->
```

- comment not explicitly visible to the user in the rendered page
- comment appears in the code for reference...

Image - HTML5 sample rendering 1

Our first web page

As we build our web apps, more elements and content will be added to this template.

HTML - sample rendering of demo 1

Source - Demo 1

HTML5 - semantic elements - part 1

- new semantic elements added for HTML5

- known as **block-level** elements
  - *includes the following elements,*

```
<article>
<aside>
<details>
<figure>
<figcaption>
<footer>
<header>
<main>
<nav>
<section>
```

- better structure underlying documents
  - *add clear semantic divisions*

HTML5 - semantic elements - part 2

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <!-- our second demo with lots of new elements -->
    <title>Demo 2</title>
  </head>
  <body>
    <header>
      <h1>Our first web page</h1>
    </header>
    <!-- primary navigation elements, links... -->
    <nav>Option 1</nav>
    <!-- main content -->
    <main>
      <section>
        <p>
          As we build our web apps, more elements and content will be added...
        </p>
        <figure>
          <img src="media/images/philae-demo2.jpg" alt="temple of philae" width="333px"
          height="200px">
        </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 - New elements added

Image - html5 sample rendering 2

Our first web page

Option 1

As we build our web apps, more elements and content will be added to this template.

Temple at Philae in Egypt is Ptolemaic era of Egyptian history. Similar temples include Edfu...
foot of the page...

HTML - sample rendering of demo 2

Source - Demo - New elements added

HTML5 - semantic elements - part 3

- element tag `article` not used in previous demo
- `article` and `section` tag can both cause some confusion
- not as widely used as expected
- `div` element still widely seen in development
- HTML5 is supposed to have relegated `div`
  - *sectioning element of last resort...*
- `article` and `section`
  - *good analogy with a standard newspaper*
  - *different sections such as headlines, politics, health...*
  - *each section will also contain articles*
- HTML specification also states that an `article` element

represents a self-contained composition in a document, page, application, or site and that is, in principle, independently distributable or reusable, e.g. in syndication.

HTML5 - semantic elements and structure - intro

- perceived issue or concern with HTML5 semantic elements
  - *how and when to add them to our document*
  - *where and when do we add them to our page?*

- non-semantic elements often considered simpler to apply
  - *generalised application and context for usage*

HTML5 - semantic elements and structure

header and nav

- `<header>`
  - *used to collect and contain introductory content*
  - *semantically appropriate for the head or top of a page*
  - *technically feasible and acceptable to include multiple `<header>` elements*
    - e.g. `<header>` within main content, sidebar content, an article, a section…

- `<nav>`
  - *short for navigation*
  - *stores and defines a set of links for internal or external navigation*
  - *not meant to define all page navigation links*
  - *often considered suitable for primary site links*
  - *additional links can be placed in*
    - sidebar, footer, main content…
    - no need to consider a `<nav>` element for these links…

HTML5 - Semantic elements and structure

main

- this element tag defines our **main** content
- traditionally the central content area of our page or document
- HTML4 often used a `<div>` element
  - *plus a `class` or `id` to define central content*
  - *e.g.*

```html
<!-- e.g. HTML4 main content -->
<div id="main">

  ...

</div>
```

- HTML5 semantically defines and marks content as `<main>`
- `<main>` should not include any page features such as
  - *nav links, headers etc, that are repeated across multiple pages*
- cannot add multiple `<main>` elements to a single page
- must not be structured as a child element to
  - *`<article>`, `<aside>`, `<footer>`, `<header>`, or `<nav>`*

HTML5 - Semantic elements and structure

section, article, aside - part 1

- `<section>`
  - *defines a section of a page or document*
  - *W3C Documentation defines as follows,*

> *a section is a thematic grouping of content. The theme of each section should be identified, typically by including a heading as a child of the section element.*

- a site can be sub-divided into multiple `<section>` groupings
  - *e.g. as we might consider a chapter or section break in a book…*

- `<article>`
  - *suitable for organising and containing independent content*
  - *include multiple `<article>` elements within a page*
  - *use to establish logical, individual groups of content*
    - again, newspaper analogy is useful to remember
    - e.g. a blog post, story, news report…might be a useful article
  - *key to using this element is often whether content can be used in isolation*

- `<aside>`
  - *used to define some content aside from containing parent content*
  - *normally used to help define or relate material to surrounding content*
  - *effectively acts as supporting, contextual material*

HTML5 - Semantic elements and structure

section, article, aside - part 2

- MDN Documentation suggests,

> *if it makes sense to separately syndicate the content of a* `<section>` *element, use an* `<article>` *element instead*

and

> *do not use the* `<section>` *element as a generic container; this is what* `<div>` *is for, especially when the sectioning is only for styling purposes. A rule of thumb is that a section should logically appear in the outline of a document.*

HTML5 - Semantic elements and structure

figure, figcaption

- **`<figure>` & `<figcaption>`**
  - *as with print media, we can logically group image and caption*
  - *`<figure>` acts as parent for image grouping*
  - *child elements include*
    - `<img>` and `<figcaption>`

```
<figure>
<img src="media/images/philae-demo2.jpg" alt="temple of philae"
width="333px" height="200px">
<figcaption>Ptolemaic temple at Philae, Egypt</figcaption>
</figure>
```

- **updated demo with figure grouping**
  - *Demo - Semantic structuring*

HTML5 - Semantic elements and structure

footer

- `<footer>`
  - *usually contains information about its containing element*

- example 1 - in a footer for an article
  - *might use this element to define and record*
    - author of the article
    - publication date
    - suitable tags or metadata
    - associated documents…

- example 2 - a footer simply placed at the **foot** of a page
  - *record copyright information*
  - *contextual links*
  - *contact information*
  - *small logos…*

- example 2 considered standard usage for `<footer>`
  - *continues from HTML4 and earlier generic usage…*

Image - HTML5 page structure - part 1

semantic elements

```
|-------------------------------------------------------------------|
|                           <header>                                |
|-------------------------------------------------------------------|
|                            <nav>                                  |
|-------------------------------------------------------------------|
|                                         |                         |
|       <section>                         |                         |
|                                         |                         |
|-----------------------------------------|      <aside>            |
|                                         |                         |
|       <article>                         |                         |
|                                         |                         |
|-----------------------------------------------------------------|
|                           <footer>                                |
|-------------------------------------------------------------------|
```

HTML5 - Structure

Image - HTML5 page structure - part 2

semantic elements

```
|----------------------------------------------------------------------|
|                              <header>                                |
|----------------------------------------------------------------------|
|                               <nav>                                  |
|----------------------------------------------------------------------|
|                              <main>                                  |
| |-----------------------------------------------------------------| |
| |                      <section>                 |                 | |
| |                                                |                 | |
| |                                                |                 | |
| |-------------------------------------------     |    <aside>      | |
| |                                                |                 | |
| |                      <article>                 |                 | |
| |                                                |                 | |
| |-----------------------------------------------------------------| |
|----------------------------------------------------------------------|
|                              <footer>                                |
|----------------------------------------------------------------------|
```
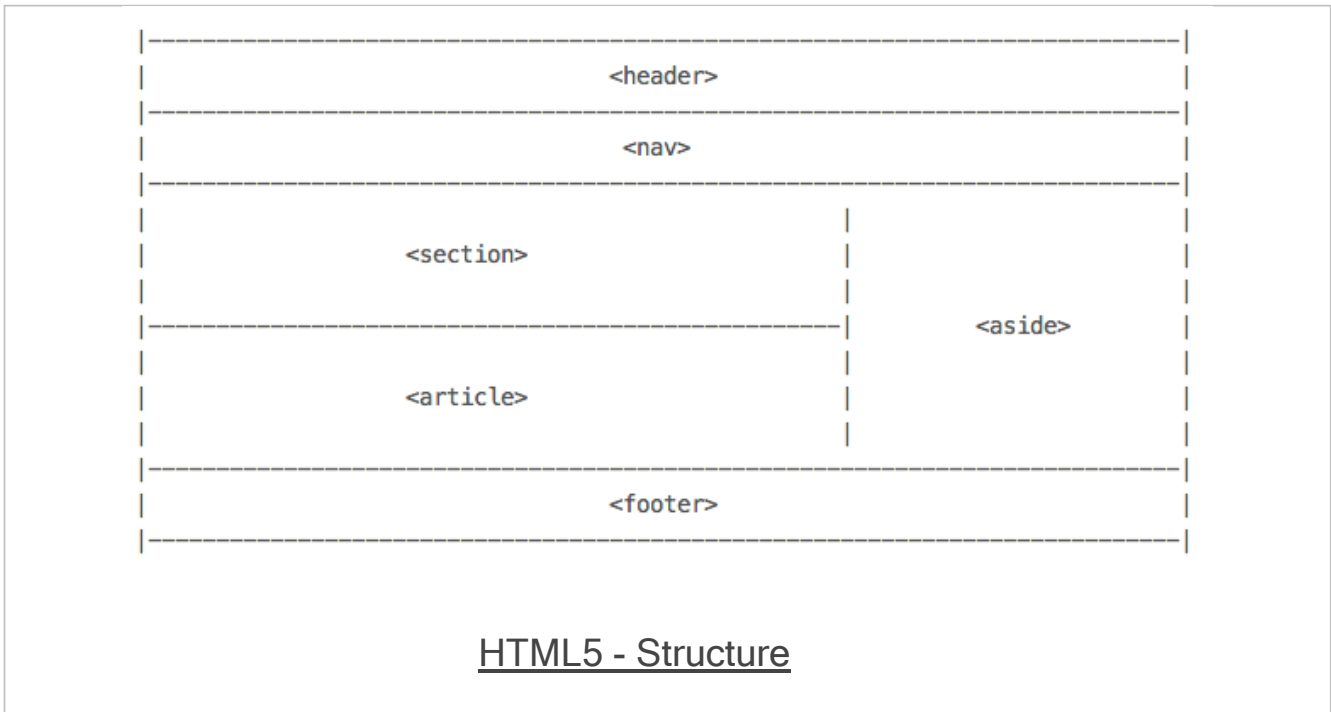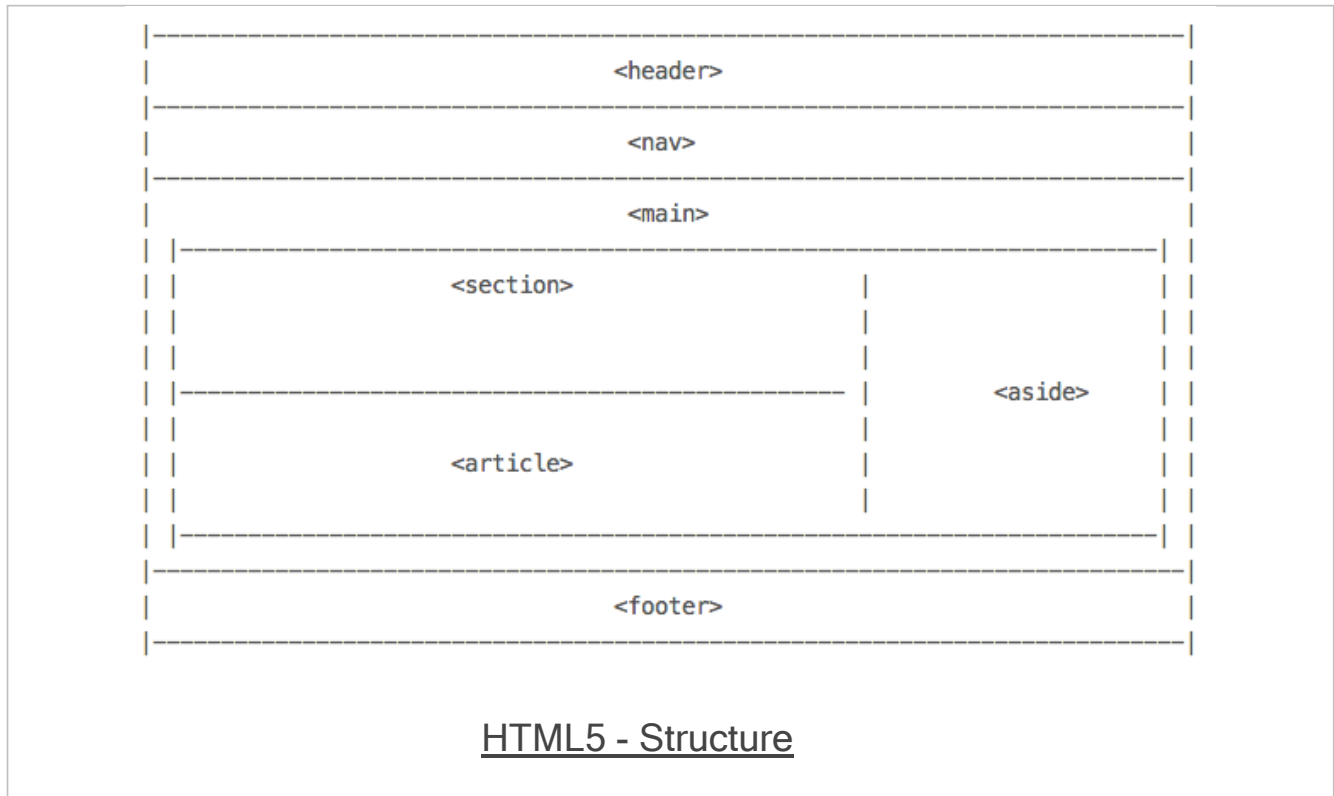
HTML5 - Structure

HTML5 page structure - part 3

- not included `<html>` and `<body>` tags in diagrams
  - *required for all HTML documents*

- divided the page into four logical, semantic divisions
  - *header*
  - *nav*
  - *main*
  - *footer*

- we could also add a sidebar etc for further division of content

HTML5 - extra elements

intro

- many other interesting and useful new HTML5 elements
  - *in addition to semantic elements*

- some struggle for browser compatibility

- useful new elements such as
  - *graphics and media*

- HTML5 APIs introduced as well, including
  - *App Cache*
  - *Drag/Drop*
  - *Geolocation*
  - *Local Storage*
  - *...*

- again, check browser support and compatibility

Browser check

- Can I Use_____?
  - *e.g. Can I Use Drag and Drop?*

video

`<video>` element

- until HTML5, video playback reliant on plugins
  - *e.g. Adobe Flash*
- embed video using element tag `<video>`
- add attributes for
  - *height, width, controls...*
- not all web browsers support all video codecs
- option to specify multiple video sources
- best supported codecs include
  - *MP4 (or H.264), WebM, OGG...*
- good general support for `<video>` element
- check browser support for `<video>` element
  - *Can I use_____video?*

HTML5 - Extra elements - media - part 2

video example

`<video>` - a quick example might be as follows,

```
<video width="300" height="240" controls>
  <source src="media/video/movie.mp4" type="video/mp4">
  <source src="media/video/movie.webm" type="video/webm">
  Your browser does not support the video tag.
</video>
```

- Demo - HTML5 Video playback

audio

`<audio>` element

- HTML5 also supports standardised element for embedded audio
- supported codecs for `<audio>` playback include
  - *MP3 and mp4*
  - *WAV*
  - *OGG Vorbis*
  - *3GP*
  - *m4a*

- again, check browser support and compatibility
  - *Can I use_____audio?*

- fun test of codecs
  - *HTML5 Audio*

audio example

\<audio\> - a quick example might be as follows,

```
<audio controls>
  <source src="media/audio/audio.mp3" type="audio/mpeg">
  Your browser does not support the audio tag.
</audio>
```

- Demo - HTML5 Audio playback

HTML5 - Extra elements - graphics - part 1

canvas

- graphics elements are particularly fun to use
- use them to create interesting, useful graphics renderings
- in effect, we can draw on the page
- `<canvas>` element acts as a placeholder for graphics
  - *allows us to draw with JavaScript*
- draw lines, circles, text, add gradients...
  - *e.g. draw a rectangle on the canvas*

HTML5 - Extra elements - graphics - part 2

canvas example

`<canvas>` will be created as follows,

```html
<canvas id="canvas1" width="200" height="100">
  Your browser does not support the canvas element.
</canvas>
```

then use JavaScript to add a drawing to the canvas

```html
<script type="text/javascript">
var can1 = document.getElementById("canvas1");
var context1 = can1.getContext("2d");
context1.fillStyle="#000000";
context1.fillRect(0,0,150,75);
</script>
```

Result is a rendered black rectangle on our web page.

- Demo - HTML5 Canvas - Rectangle

HTML5 - Extra elements - graphics - part 3

canvas example

A square can be created as follows,

```html
<script type="text/javascript">
function draw() {
/*black square*/
var can1 = document.getElementById("canvas1");
var context1 = can1.getContext("2d");
context1.fillStyle="#000000";
context1.fillRect(0,0,50,50);
}
</script>
```

Again, we end up with the following rendered shape on our canvas.

- Demo - HTML5 Canvas - Square

HTML5 - Extra elements - graphics - part 4

canvas examples

- modify drawing for many different shapes and patterns
  - *simple lines, circles, gradients, images...*

    1. shows different rendered shapes on a canvas.

- Demo - HTML5 Canvas - Assorted Shapes

    2. little retro games

- Demo - HTML5 Canvas - Retro Breakout Game

HTML5 - Extra elements - graphics - part 5

canvas examples - basics

- basic drawing - rectangle & staircase
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic/*

- Example - basic drawing - stepped pyramid
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic2/*

- Example - various colours
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic3/*

- Example - basic drawing - rectangle outlines
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic4/*

- Example - draw lines - line & pyramid
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic5/*

- Example - draw a stickman
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic6/*

- Example - fill paths
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic7/*

HTML5 - Extra elements - graphics - part 6

canvas examples - curves & circles

- Example - arcs and circles
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic8/*

- Example - Bézier curves - quadratic
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic9-quadratic/*

- Example - Bézier curves - cubic
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic9-cubic/*

- Example - arcs and circles - combine shapes to create an *ankh*
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic10-ankh/*

- Example - circle function
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic11-function-circles/*

HTML5 - Extra elements - graphics - part 7

canvas examples - animation & fun

- Example - horizontal animation
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-animation/animation1/*

- Example - animate size
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-animation/animation2/*

- Example - variant mouse colours
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-animation/animation3.1/*

- Example - variant mouse colours
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-animation/animation3.2/*

- Example - random movement and animation
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-animation/animation3.3/*

HTML5 - Extra elements - graphics - part 8

canvas examples - images & files

- Example - draw image to canvas from local file
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-image/basic1/*

- Example - draw image to canvas from local file - dw & dh
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-image/basic2/*

- Example - draw image to canvas from local file - dw & dh plus source crop
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-image/basic3/*

HTML5 - Extra elements - graphics - part 9

canvas examples - move & control

- Example - move ball with keyboard control
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-game/basic-ball-move1/*

- Example - update `move()` to check canvas boundaries
  - *ttp://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-game/basic-ball-move2/*

- Example - move ball on 4-point axis
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-game/basic-ball-move3/*

- Example - move sprite image
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-game/basic-sprite-move1/*

- Example - move sprite image
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-game/basic-ball-move4/*

- Example - check basic collision against blocks
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-game/basic-ball-move5/*

- Example - check basic collision against blocks - horizontal
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-game/basic-ball-move6/*

more fun canvas examples

Some fun examples of animations with HTML5 Canvas API.

- Destroy things in a video - http://www.craftymind.com/factory/html5video/CanvasVideo.html
- Particles - https://codepen.io/eltonkamami/pen/ECrKd
- Curtain - https://codepen.io/dissimulate/pen/KrAwx
- Jelly - https://codepen.io/dissimulate/pen/dJgMaO
- Canvas cycle - http://www.effectgames.com/demos/canvascycle/

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

CSS Basics - stylesheet

- add a link to our CSS stylesheet in the `<head>` element

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

- change will replicate throughout our site wherever the stylesheet is referenced

CSS Basics - `<style>` element

- embed the CSS directly within the `<head>` section of our HTML page

- embed using the `<style>` element

- then simply add standard CSS within this element

- limitations include lack of abstraction for site usage and maintenance
  - *styles limited to a single page...*

```
<style type="text/css">
body {
  color: #000;
}
</style>
```

CSS Basics - inline

- embed styles per element using **inline** styles
  - *limitations and detractors for this style of CSS*
  - *helped by the growth and popularity of React...*

e.g.

```
<!-- with styles -->
<p style="color:#cd0603">a trip to Luxor</p>
<!-- without styles -->
<p>a trip to Karnak</p>
```

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...*
- various CSS frameworks available
- 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, e.g.

- selector
  - *e.g. body or p*

- declaration
  - *property and value pairing*

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

- body is the selector, `color` is the property, and `black` is the value.

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,

```css
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,

```css
/* 'color' can be set to a named value, HEX value (e.g. #444) &c. */
p {
  color: blue;
  font-size: 14px;
}
```

- comments can be added before the selector or within the braces
- Demo - CSS Basics

Image - CSS Syntax

```
    Selector              Declaration

  |———————|        |—————————————————————|
  |   p   |        | { font-size: 14px; } |
  |———————|        |—————————————————————|
                        ^              ^
                        |              |
                    property        value
```

CSS Syntax

CSS Basics - display

- display HTML elements in one of two ways
  - *inline - e.g. <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
- Demo - CSS Basics - Add a Class

CSS Basics - inline elements

Current inline elements include, for example:

- b | big | i | small
- abbr | acronym | cite | dfn | em | strong | var
- a | br | img | map | script | span | sub | sup
- button | input | label | select | textarea
- ...

Source - MDN - Inline Elements

n.b. not all inline elements supported in HTML5

CSS Basics - block-level elements

Current block-level elements include:

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

Source - MDN - Block-level Elements

n.b. *block-level* is not technically defined for new elements in HTML5

CSS Basics - HTML5 content categories - part 1

- **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>, <img>, <label>, <script>, <video>…*
  - *other elements can belong to this category if certain conditions are met. e.g. <a>*

- **Embedded content** - imports or inserts resource or content from another mark-up language or namespace
  - *<audio>, <canvas>, <embed>, <iframe>, <img>, <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>, <img>, <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

HTML5 Content Categories
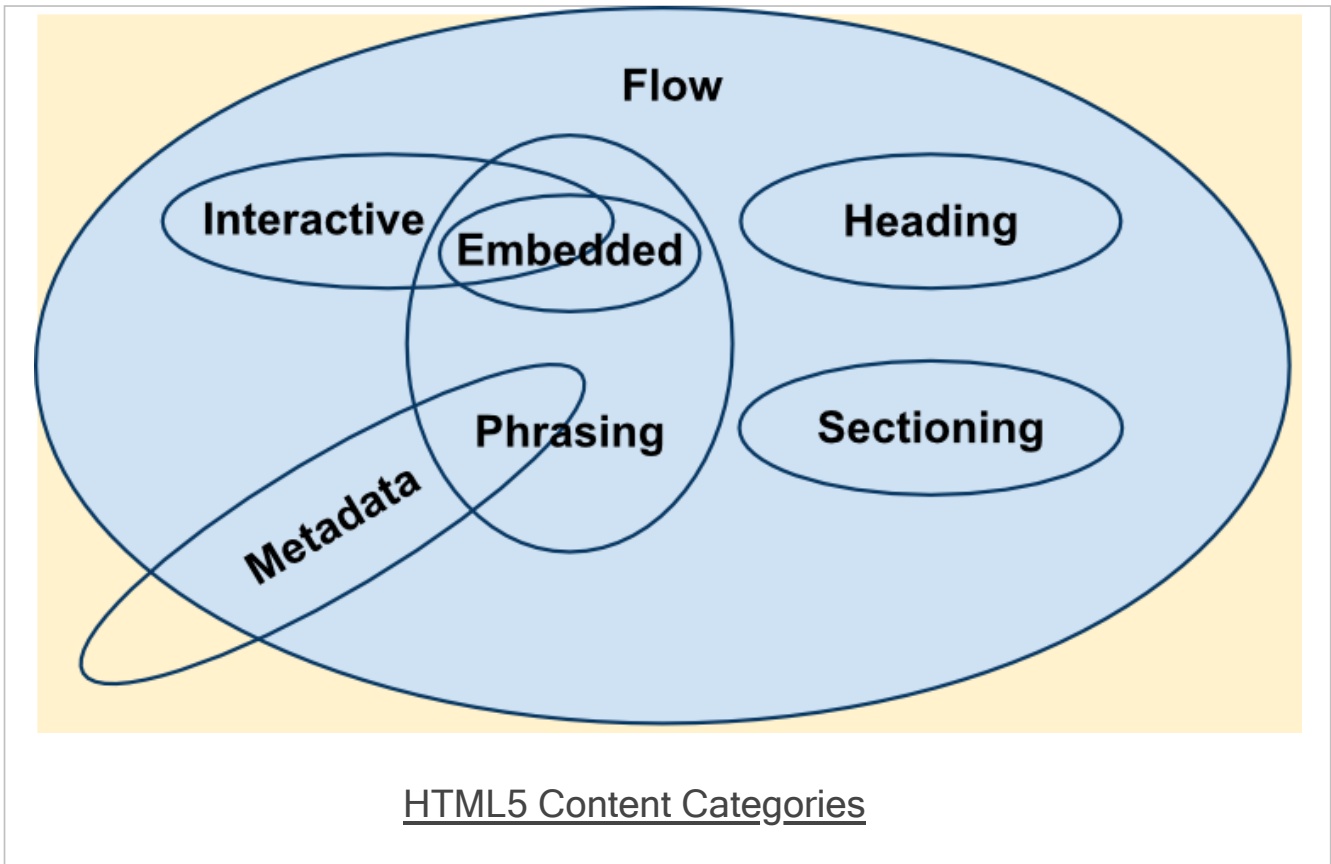
Source - MDN - Content Categories

CSS Basics - box model - part 1

- 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 a series of 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 - CSS 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 - CSS Box Model

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 - CSS Box Model

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*
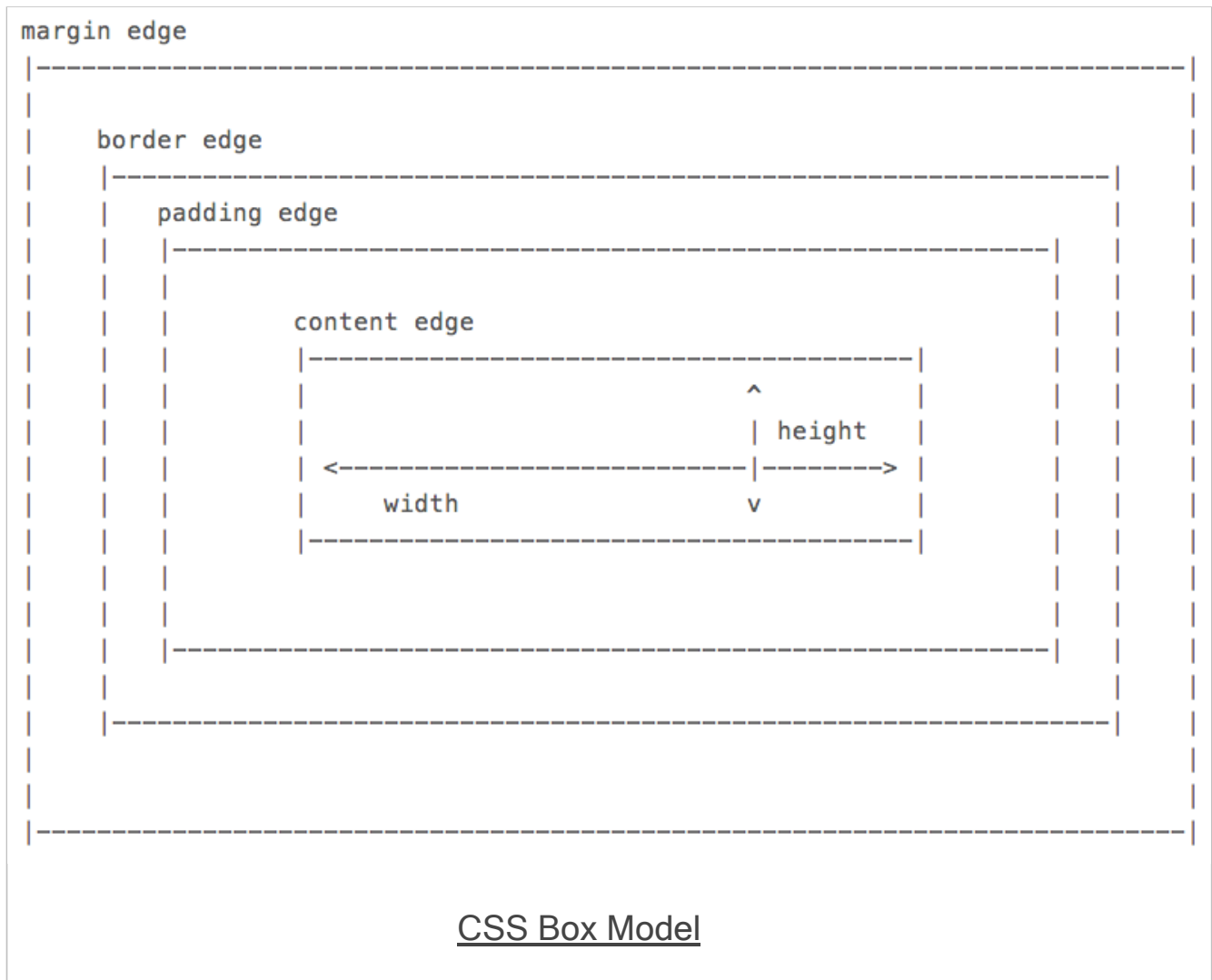 - *margin (sizes calculated clock-wise)*

Demo - CSS Box Model - Margin

- JSFiddle - CSS Box Model

Demo - CSS Box Model

- Demo - CSS Box Model

Image - CSS Box Model

```
margin edge
 |----------------------------------------------------------------|
 |                                                                |
 |                                                                |
 |     border edge                                                |
 |     |----------------------------------------------------|     |
 |     |                                                     |     |
 |     |     padding edge                                    |     |
 |     |     |--------------------------------------|        |     |
 |     |     |                                      |        |     |
 |     |     |       content edge                   |        |     |
 |     |     |     |--------------------------|     |        |     |
 |     |     |     |                    ^     |     |        |     |
 |     |     |     |                    | height   |        |     |
 |     |     |     |<-------------------|------->  |        |     |
 |     |     |     |       width        v     |     |        |     |
 |     |     |     |--------------------------|     |        |     |
 |     |     |                                      |        |     |
 |     |     |                                      |        |     |
 |     |     |--------------------------------------|        |     |
 |     |                                                     |     |
 |     |----------------------------------------------------|     |
 |                                                                |
 |                                                                |
 |                                                                |
 |----------------------------------------------------------------|

                          CSS Box Model
```

Source - MDN - CSS Box Model

Demo - CSS Box Model - Interactive

- interactive Box Model

Demos

- HTML
  - *Basic Attribute*
  - *Basic Attribute 2*
  - *Basic group*
  - *Basic group style*
  - *Basic list items*
  - *Basic menu tabs*
  - *Basic table caption*
  - *Basic table for presentation*
  - *Basic table with accessibility*
  - *Basic table with head, foot, body*
  - *Basic table with headers*
  - *Basic table with summary*
  - *DOM Basics - Sample*
  - *List Toggle Children*
  - *List Toggle Sibling*
  - *Semantic example usage*
  - *Toggle horizontal with animation*
  - *Toggle vertical with animation*

- HTML5
  - *Basic structure*
  - *New elements added*
  - *Semantic structuring*
  - *HTML5 Video playback*
  - *HTML5 Audio playback*

- HTML5 Canvas demos
  - *HTML5 Canvas - Rectangle*
  - *HTML5 Canvas - Square*

- *HTML5 Canvas - Assorted Shapes*
- *HTML5 Canvas - Retro Breakout Game*
- *please see links in slides for further examples...*

CSS
- *CSS Basics*
- *CSS - Box Model*
- *CSS - Interactive Box Model*

CSS - JSFiddle tests
- *CSS Box Model*
- *CSS Box Model Padding*

Resources

part 1

- Jaffe, Jim., *Application Foundations For The Open Web Platform*. W3C. 10.14.2014. http://www.w3.org/blog/2014/10/application-foundations-for-the-open-web-platform/

- MDN - Using Dynamic Styling Information

- The Unicode Consortium

- Unicode Information
  - *Unicode examples*

- W3 Docs for further details

part 2

- Can I Use_____?
  - *e.g. Can I Use Drag and Drop?*

- Check browser compatibility using HTML5 Test

- HTML Colour Picker

- HTML5 Audio & Codecs

- JS Info - DOM Nodes

- MDN Documentation
  - *Block-level Elements*
  - *Content Categories*
  - *CSS Box Model*
  - *CSS Selectors*
  - *Global Attributes*
  - *HTML developer guide*
  - *requestAnimationFrame*

- *Section element*

  - W3C Documentation
    - *Section element*

  - HTML5 Canvas - fun examples
    - *Destroy things in a video*
    - *Particles*
    - *Curtain*
    - *Jelly*
    - *Canvas cycle*