WEEK 3 / LESSON 5:

LAYOUT LAB

Leonie Dunnett

TODAY'S SCHEDULE

- Review
  - HTML Review
  - CSS Basics
  - The Box Model
  - Layout
  - Tools and Testing
- Layout Lab

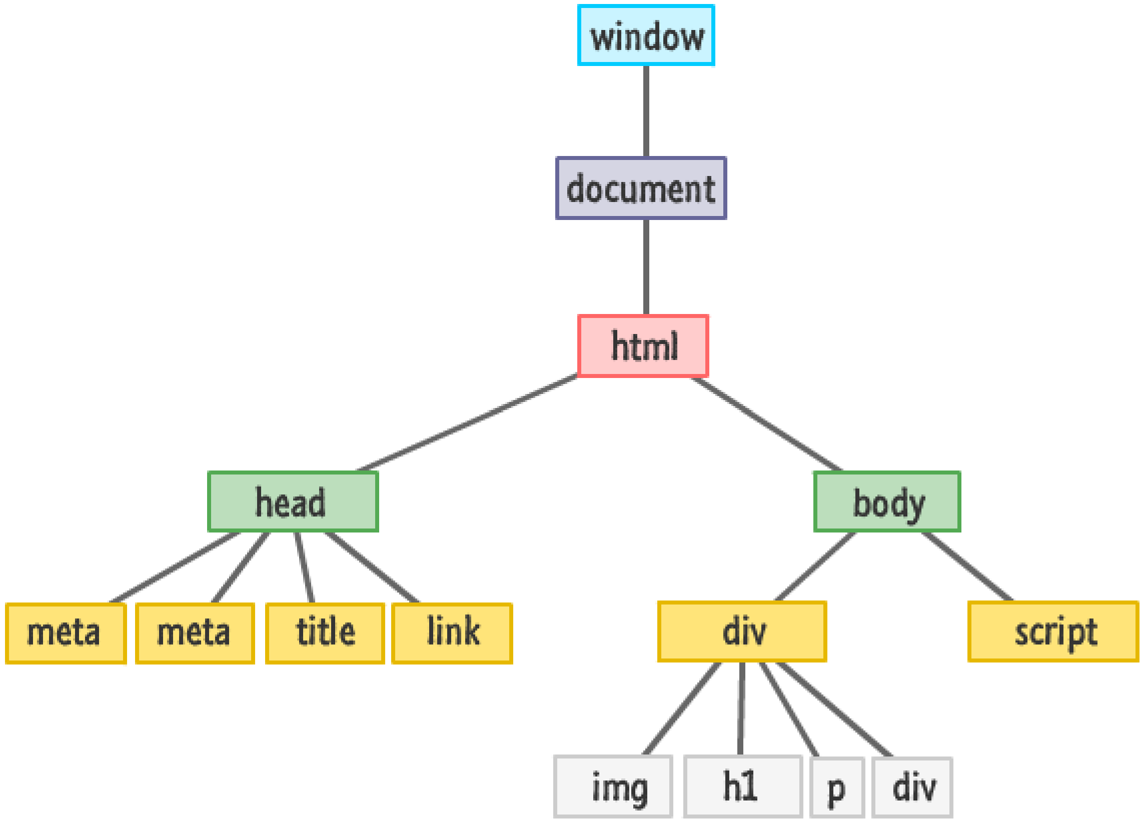
HTML REVIEW

Structure, elements, attributes, style sheet:

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Some web page</title>
        <link rel="stylesheet" href="css/style.css">
    </head>
    <body>
        <h1>Heading text here</h1>
        <p>Paragraph text here with a
            <a href="LinkUrl">Link</a>
        </p>
    </body>
</html>
```

THE DOM (DOCUMENT OBJECT MODEL)

A tree-like structure of your HTML document for accessing and manipulating its content.



HTML5 STRUCTURAL ELEMENTS

`header nav section article aside footer`

- give pages more meaning and organisation
- pages used to be built with `div`'s (no semantic value)
- can be used multiple times per page
- each use must reflect proper semantic meaning

WHEN TO USE <NAV>

The HTML 5 specification defines <nav> as:

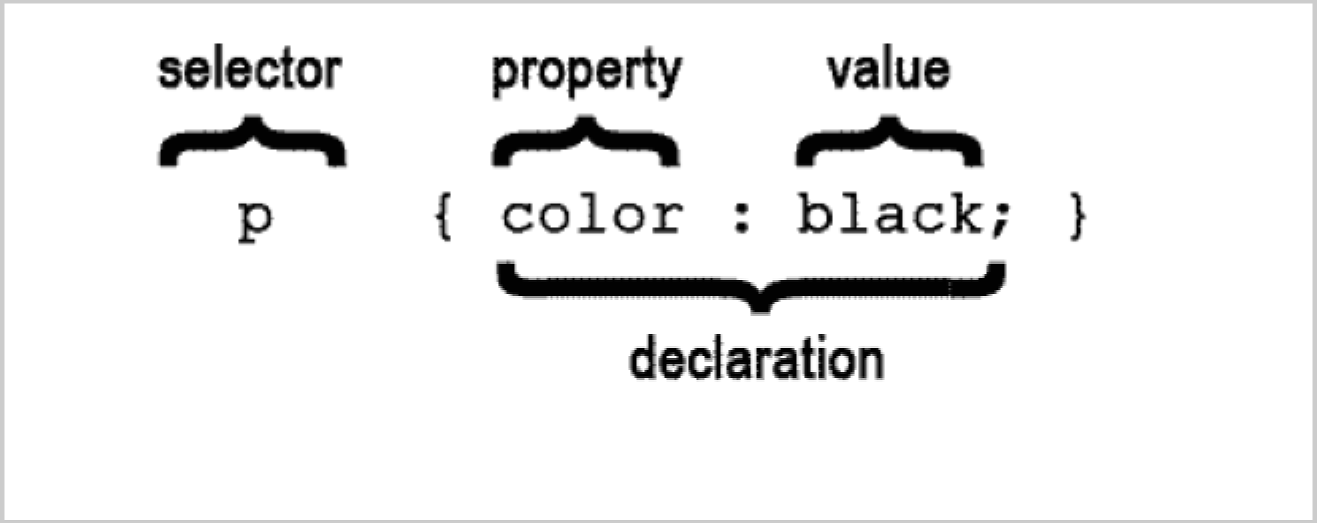
The nav element represents a section of a page that links to other pages or to parts within the page: a section with navigation links.

Not all groups of links on a page need to be in a nav element only sections that consist of major navigation blocks are appropriate for the nav element.

In particular, it is common for footers to have a list of links to various key parts of a site, but the footer element is more appropriate in such cases, and no nav element is necessary for those links.

The key phrase there is "major navigation" (previously primary navigation).

CSS BASICS



```
p   { color : black; }
```

This whole thing is called a **rule**.

CLASS AND ID

Help us target elements with css and javascript.

ID'S ARE UNIQUE

One element with `id` per page

```
#content-wrap { ... }
```

CLASSES ARE NOT

Many elements with same `class`

```
.post { ... }
```

FAVOUR CLASSES IN CSS

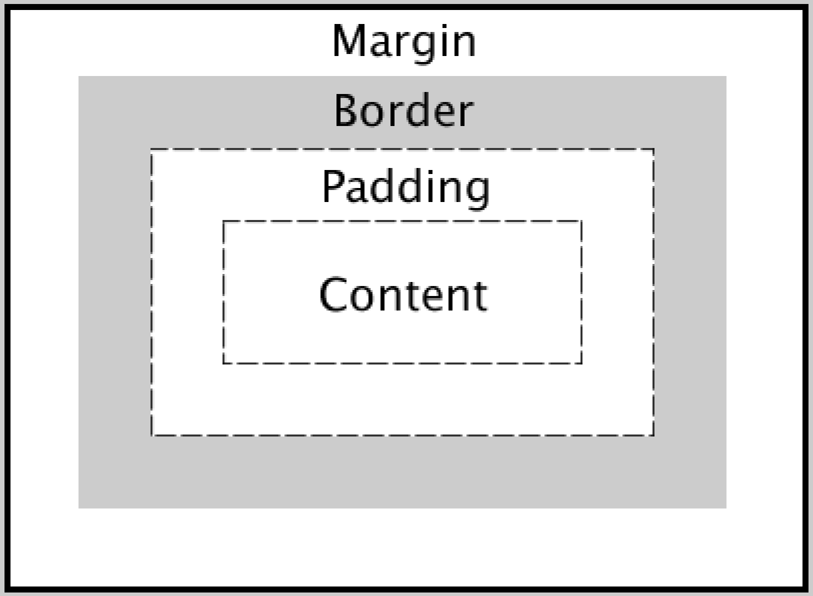
They render quickly and allow styles to be reused.

Overall goal: short selectors, decreasing specificity, creating cleaner, more reusable code.

```
/* Bad */
#content-wrap header nav {...}

/* Good */
.main-nav {...}

/* Bad */
article.post {...}

/* Good */
.post {...}
```

THE BOX MODEL

Describes the physical properties of an elements box.



Everything on the page is a rectangular box.

Total width and height is **additive**.

BOX SIZING RESET

Default box-size is `content-box`. Set it to `border-box`:

```
html {
  box-sizing: border-box;
}
*, *:before, *:after {
  box-sizing: inherit;
}
```

Learn more about this reset: Box Sizing Best Practice

LAYOUT

BLOCK VS. INLINE ELEMENTS

```
display: block | inline | inline-block;
```

BLOCK-LEVEL ELEMENTS

`display: block;`

- **Begin on a new line**
- Stack
- Occupy available width
- Can be nested
- Can wrap inline-level elements

eg. `<p>`, `<div>`

INLINE-LEVEL ELEMENTS

`display: inline;`

- **Don't begin on a new line**
- Fall into normal document flow
- Only maintain the contents width
- Will not accept width and height properties
- Can also be nested
- Can't wrap block-level elements

eg. `<span>`

(`<a>` are an exception to this)

INLINE-BLOCK ELEMENTS

`display: inline-block;`

Element behaves as a block-level element, but will not begin on a new line by default.

There is often space between `inline-block` elements, removed by removing the whitespace between elements, or more preferably, commenting out the whitespace.

POSITIONING WITH INLINE-BLOCK

CSS

```
section {
  display: inline-block;
  width: 60%;
}
aside {
  display: inline-block;
  width: 40%;
}
```

HTML (add comment between inline-block elements)

```
<header>...</header>

<section>...</section><!-- remove whitespace
 --><aside>...</aside>

<footer>...</footer>
```

POSITIONING WITH FLOATS

```
float: left | right | none;
```

A FLOATED ELEMENT

- is removed from normal document flow
- is positioned to the `left` or `right` of its parent
- essentially has its `display` value set to `block`
- needs a width percentage

CLEARING & CONTAINING FLOATS

Returns page to its normal flow.

Clearing: `clear: left | right | both;`

Containing floats (in parent element with `class="group"`):

```
.group:after {
  content: "";
  display: block;
  clear: both;
}
```

THE OVERFLOW METHOD

- Another option
- Setting the CSS overflow property on a parent element

  `overflow: auto | hidden;`
- Parent will expand to contain the floats, clearing it for succeeding elements
- Careful not to hide content or trigger unwanted scrollbars

CSS POSITION PROPERTY

`position: static | relative | absolute | fixed | inherit;`

`static` - normal document flow (default)

`relative` - normal flow with offsets (top left right bottom)

`absolute` - removed from normal flow, repositioned with offsets (relative to nearest positioned ancestor)

`fixed` - positioned relative to viewport, repositioned using offsets (same place when scroll)

TOOLS AND TESTING

- Chrome Debugger
  (or Firebug and Web Dev Tools in Firefox)
- html validator
- css validator
- Browser test on all popular browsers and devices

LAYOUT LAB

Relaxr Blog Page

HOW TO START?

- Draw a DOM (Group exercise)
- Add basic HTML markup
- Add page text
- Begin adding body HTML markup to the page
- Style bit by bit (text, backgrounds, colors, layout, etc)

HOMEWORK

Review: Slides and resources

Code: Relaxr Blog Page (Assignment 3)

RESOURCES

- Google Fonts
- CSS Font Shorthand Cheat Sheet via Impressive Webs
- Via Shay Howe:
  - Working with Typography
  - Setting Backgrounds & Gradients
  - Writing Your Best Code
  - Detailed Positioning