
## WEEK 2 / LESSON 4:

# LAYOUT

Leonie Dunnett

# LEARNING OBJECTIVES

- Differentiate between classes and IDs and apply best practices when implementing
- Apply header, footer, sidebar, and multi-column layouts to develop a web page
- Experiment with CSS positioning and floats

# TODAY'S SCHEDULE

- Review
- Introducing div, class and id `Code Along`
- Building page structure
- CSS Position Property
- Positioning with Floats
- Floating Sections `Code Along`
- Layout challenge `Lab`

# REVIEW

What would you like to review?

# INTRODUCING
# DIV, CLASS AND ID

# DIV

`<div>...</div>`

- Stands for 'division'
- Holds no semantic value
- Containing blocks to position, style and group elements
- Some divs can now be replaced with more semantic markup like `<section>`, `<nav>`, `<header>`, etc.

# CLASS AND ID

Classes and id's allow CSS and javascript to:

- target specific elements on a page
- manipulate them uniquely

```
<p class='excerpt'> ... </p>
```

```
<div id='content-wrap'> ... </div>
```

# HOW TO SELECT IN CSS

Classes are denoted by a '.':

```
.excerpt { ... }
```

Id's are denoted by a '#':

```
#content-wrapper { ... }
```

## ID'S ARE UNIQUE

Only element with given `id` on page.

## CLASSES ARE NOT

Many elements with the same class.

# CLASS & ID

When should you use them?


# CLASS & ID

# BUILDING PAGE STRUCTURE

# HTML5 STRUCTURAL ELEMENTS

Structure of web pages used to be built with divisions, but they hold no semantic value.

HTML5 introduced new structurally based elements, helping us give pages more meaning and organisation, including:

`header`, `nav`, `section`, `article`, `aside`, and `footer`

They can be used multiple times per page, so long as each use reflects the proper semantic meaning.

# HEADER

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

- Used as a container for introductory content
- Identifies the top of a page, section, article, etc.
- Often includes a heading, intro text, and navigation

# NAVIGATION

`<nav>...</nav>`

- Defines a set of navigation links
- Intended for large blocks of navigation links
- Not all links should be inside a `<nav>` element!

# FOOTER

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

- Identifies the end of an element or page
- Content should be relevant to the containing element
- Typically contains the document author, copyright info, links to terms of use, contact info, etc.

# ASIDE

`<aside>...</aside>`

- Essentially the 'sidebar' element
- Content should be related to the surrounding content (eg. Could be used within an `<article>` element to identify content related to the author of the article)
- A block level element that will appear on a new line - must be positioned

# SECTION

`<section>...</section>`

- A thematic grouping of content
- Typically with a heading (not always)
- Useful to identify the content as related

# ARTICLE

`<article>...</article>`

- Specifies independent, self-contained content
- Should make sense on its own
- Could be read independently from the rest of the web site eg. Forum/Blog post, Newspaper article

# CSS POSITION PROPERTY

# ALTERING AN ELEMENTS POSITION

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

`static` - uses normal document flow (default)

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

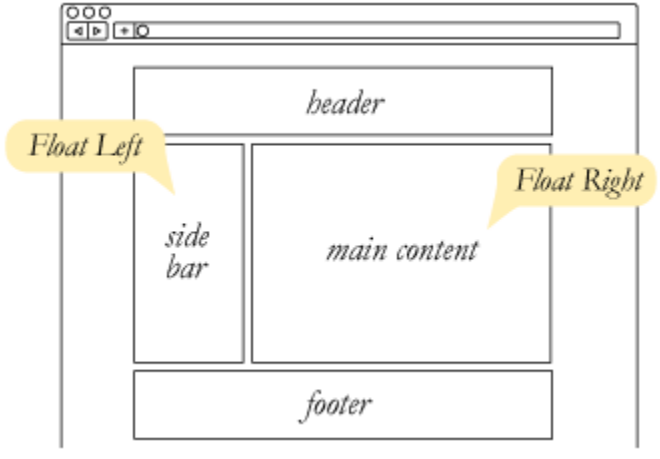
`absolute` - removed from document flow and repositioned using offsets (placed relative to nearest positioned ancestor)

`fixed` - positioned relative to the viewport and repositioned using offsets (stays in the same place when page scrolled)

`inherit` - uses the value of its parent

# POSITIONING WITH FLOATS

Float is a CSS positioning property, that can used to layout a web page.



# THE FLOAT PROPERTY

- Essentially removes element from normal document flow and positions it to the `left` or `right` of its parent
- All other page elements will flow around the floated element
- Originally used to float `<img>` elements to the side of text, allowing the text to wrap around the image
- Enables us to create a (multi-column) layout when used on multiple elements at the same time

# A FLOATED ELEMENT

Will fill container width, so a width percentage will need to be set on elements floating side by side

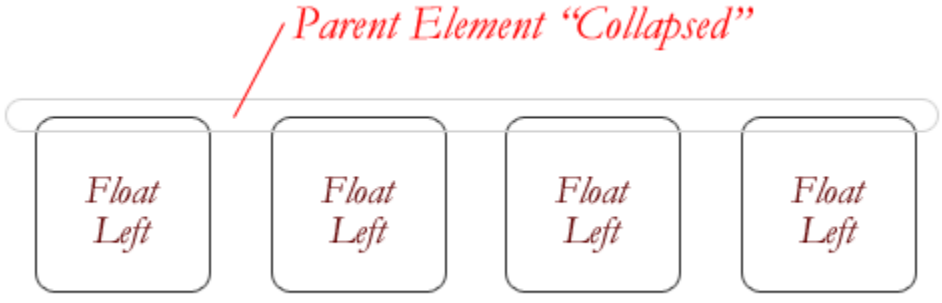
Essentially has its `display` value set to `block`, (if not already) and will be able to except height and width values

# FLOAT PITFALLS

- Not intended to be used for layout, so has a few issues (designed to wrap content around images)
- Taking element out of normal document flow essentially impacts surrounding elements

# COMMON ISSUES

## Parent element collapses



# COMMON ISSUES

Page elements wrap around floated elements, consuming any available space around them - often undesired

# CLEARING & CONTAINING FLOATS

To prevent content from wrapping around floated elements, we need to clear, or contain, those floats and return the page to its normal flow.

# CLEARING FLOATS

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

# CONTAINING FLOATS

- Another option
- Helps to ensure all styles are rendered properly
- Floated elements must reside within a containing parent
- Document flow is completely normal outside of parent

# CONTAINING FLOAT EXAMPLE

## The CSS for that parent element 'group':

```
.group:before,
.group:after {
  content: "";
  display: table;
}
.group:after {
  clear: both;
}
.group {
  clear: both;
  *zoom: 1;
}
```

`:before` and `:after` are dynamically generated elements above and below the element

The use of `table` rather than `block` is only necessary if using `:before` to contain the top-margins of child elements.

# CONTAINING FLOAT EXAMPLE

If you only need IE8 and up, this is fine:

```
.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


# FLOATING SECTIONS


# LAYOUT CHALLENGE

# REVIEW: LEARNING OBJECTIVES

- Differentiate between classes and IDs and apply best practices when implementing
- Apply header, footer, sidebar, and multi-column layouts to develop a web page
- Experiment with CSS positioning and floats

# HOMEWORK

**Review:** Slides and resources

**Code:** Finish Relaxr Landing Page

# RESOURCES

Difference Between ID and Class via CSS Tricks

HTML 5 Semantic Elements via w3schools

CSS positioning via w3schools

CSS Floats via CSS Tricks

The Clearfix Hack via CSS Tricks

# EXIT TICKETS