HTML & CSS: Week 5

Instructor: Devon Persing

This week

- Layout review and wrap-up (including responsive layouts!)
- Pseudo-selectors
- Embedding content with iframes
- Web fonts
- Javascript, CSS3 and CSS4 overviews
- Other odds and ends

Layouts continued

3 main ways to adjust the page flow

Historically, we've used three main ways to create web layouts. I like to call them:

1. **Floating Down the River**
2. **Behavior Modification**
3. **Dictator for Life**

We'll try them out in our **Layout Laboratory**.

Method 1: Floating Down the River

- The **`float`** property lets us arrange elements like islands in the flow of the page:
  - To the left of the flow (**`left`**)
  - To the right of the flow (**`right`**)
- The **`clear`** property resets content that comes after floated content, like a bridge across the river (or a dam?)
  - Use **`clear:`** `both;` to clear **`left`** and **`right`** floats

Method 2: Behavior Modification

- **Inline** elements like **`<span>`** and **`<a>`** line up with their neighbors
- **Block** elements like **`<div>`**, **`<p>`**, **`<ul>`**, etc. create line breaks
- **Inline-block** elements like **`<img>`** line up but also maintain the box like block elements
- We can use the CSS **`display`** property to make elements behave like one or the other

Modify `display` behavior with CSS

- Make any element **behave like a block element**
- Make any element **behave like an inline element**
- Make any element **behave like an inline-block element**

```
a { display: block; }
div { display: inline; }
article { display: inline-block; }
```

The `inline-block` layout trick

- **`inline-block`** was designed to display text elements like links, so it includes a tiny bit if space at the right for readability
- To fix this, give your inline-block elements a **negative right margin**:

```
div {
   display: inline-block;
   margin-right: -4px;
}
```

A few more ways to use `display`

- Make a normally-inline element **stand out**
- Style the same HTML differently in **different contexts**
- Manipulate how/when content is displayed using **Javascript**
- There are **many** more values for **display**

Method 3: Dictator for Life

- The **`position`** property lets us arrange elements:
  - In relation to the flow (**`relative`**)
  - In a very specific place outside of the flow or within another **`relative`** element (**`absolute`**)
  - In relation to the browser window (**`fixed`**)
- How **position** is applied depends on to where the element is in the flow by default

Tweaking the `position`

- We can further dictate where elements go down to the pixel with a few additional elements
- **`left`**, **`right`**, **`top`** and **`bottom`** add or subtract pixels between positioned elements and their containers

```
div { position: absolute; right: -10px; top: 30px;
}
```

Let's make a positioned layout.

Responsive web design

- Allows layouts to **adjust to the size of a device or browser window**
- Uses **% of the parent container** instead of fixed pixel widths
- We can use **CSS media queries** to call different styles based on the size of a user's device or browser window, along **breakpoints**

@media queries

- Designed to use different styles based on the way content is being displayed
- Previously most commonly used to style web pages for print

```
@media all and (max-width: 520px) {
   /* styles for smaller devices */
}
```

Media queries for layout example

```
/* basic widths for larger browser window/screen */
main { width: 80%; }
aside { width: 20%; }

/* styles for smaller browser window/screen override
previous widths */
@media all and (max-width: 520px) {
    main, aside {
        width: 100%;
        /* change other styles at different browser sizes!
        */
        background: #ccc;
        font-size: 1em;
    }
}
```

Let's add some media queries.

CSS pseudo-classes

Pseudo-classes are conditional

- **Pseudo-classes** are added to a selector to add conditional styles to an element
- Most often used to style **states** of **<a>** elements and form elements

```
a:link { /* the default state of a link */ }
a:visited { /* a link that's been clicked */ }
a:hover { /* a link that has a mouse hover */ }
a:focus { /* a link that has keyboard focus */ }
a:active { /* a link that is currently being clicked */ }
```

`:hover` versus `:focus`

- `:hover` is for a link or other interactive element that has a **mouse hover**
- `:focus` is for a link or other interactive element that has **keyboard focus**
- Browsers have their own default `:focus` styles for **accessibility**

```
a:hover, a:focus {
    /* it's good practice to style :hover and :focus
together so they're both accounted for */
}
```

`:hover` for non-interactive elements

- `:hover` can be used to style hover states for some non-interactive elements to create a more dynamic experience

```
tr { /* a table row with one background... */
    background: #99ff66;
}

tr:hover { /* ...could have another on hover */
    background: #ff6600;
}
```

Let's add pseudo-class styles to our links.

Some other nifty pseudo-classes

- `:first-letter` styles the first letter of a block of text
- `:first-child` and `:last-child` style the first and last children of a parent
- `:nth-child()` can be used to style even or odd children, or do some math to style every 5th, etc.
- `:before` and `:after` can be used to add style-only pseudo-content to elements

Using `:after` to `clear`

```
.clearfix:after {
    content: ".";
    visibility: hidden;
    display: block;
    height: 0;
    clear: both;
}
```

- Use the `:after` pseudo-selector to create a clearing container after floated elements

CSS selectors are evolving

- **Pseudo-classes**, **pseudo-elements**, **combinators**, and **attribute selectors** create extremely targeted ways to style content that degrade gracefully in older browsers
- To learn more of these techniques, and see which ones work in which browsers: **http://www.quirksmode.org/css/selectors/**

Demos!

Embeddable content

Embedded content and media

- Embedded content is what it sounds like: content, usually media, that is embedded in our HTML page
- We already know one embeddable element: the **`<img>`** tag
- Probably the next most common type of embedded content is the **`<iframe>`**

`<iframe>` implementation

- Used to load content from **another HTML document** into an HTML page
- iframes have a **`src`** attribute, just like an image
- Commonly used to:
  - Embed **YouTube videos**
  - Add **social widgets** (like the Facebook Like button)
  - Load 3rd party ads on a page

Good practice for `<iframe>` elements

- **Include fallback HTML** in case the iframe fails to load
- **Specify the iframe's dimensions** with CSS or HTML attributes

```
<iframe src="page.html" width="200" height="400">
   If you can see this, your browser doesn't support iframes. <a href="page.html">Here's a direct link to the content.</a>
</iframe>
```

An example YouTube iframe

- There's very little reason to make your own iframes to include in your own pages
- Let's drop a YouTube iframe into a page and see how it works

```
<iframe width="640" height="360" src="//www.
youtube.com/embed/oDdUg6d1K-A?rel=0"
frameborder="0" allowfullscreen></iframe>
```

`<video>` and `<audio>` elements

- HTML5 introduced **`<video>`** and **`<audio>`** embeddable elements (and others)
- Adds **default playback controls** that can be manipulated with Javascript
- Can fall back to Flash media
- Still very experimental!
- **Example HTML5 video**

Web fonts

Freedom from Arial!

- Web fonts let us style sites with **fonts that users may not have** on their own device
- Web font services **licence fonts for online use** specifically
- Files are either **hosted by a service** or **downloaded from a service and served with your pages**
- Fonts are added via CSS

A note about licensing

- **Not all fonts can be used online**, even if you own their rights for print, they're in Adobe products, etc.
- Fonts with online licensing will come with **documentation saying so**
- **Exception:** If you own the rights to use a font with design software, you can use it to make images that are published online

Some web font options

- **Google Fonts** is free and hosted
- **TypeKit** (owned by Adobe) is hosted and subscription based or bundled with Creative Cloud
- **FontSquirrel** is free and not hosted
- **FontDeck** is subscription based and not hosted
- **And many others!**

Let's try out Google Fonts.

Extra goodies

Some HTML and CSS-related stuff

- Javascript
- CSS3 and CSS4
- Mobile-first thinking
- Accessibility
- Version control

Javascript

- The third pillar of the web along with HTML and CSS
- Embedded into an HTML document with the `<script>` tag
- Allows for additional interactivity and data manipulation that isn't possible with HTML and CSS alone

Javascript use examples

- Hiding, showing, moving, etc., content based on user actions
- Drawing content on the screen based on data (ex.: Chart.js)
- Collecting data about the type of browser, device, and internet connection a user has, and serving them content that is appropriate

Javascript libraries

- A set of **pre-made scripts**
- A platform for **common user interface patterns** like slideshows, moving/hiding/showing, widgets, validation
- **Heavily tested** and **prevents having to roll your own** Javascript to complete a common task
- Probably the most common is **jQuery**

Javascript and CSS frameworks

- A set of **pre-made scripts and styles** for quickly prototyping or iterating on projects
- **Heavily tested** and **prevents having to roll your own** Javascript and styles to complete a common task
- Probably the most common is **Twitter Bootstrap**
- ...which I'm using for the class site

CSS3 and CSS4

- **CSS3+4** techniques add extra **refinements**, **depth**, **transitions**, **animations**, **rotations**, and **typography options**
- Frequently **combined with Javascript**
- Range from simple (rounded corners) to **full-blown interactive experiences** previously only possible with Flash or Javascript (ex: Animate.css)

Mobile and tablet-first thinking

- Means thinking about scaling up using **progressive enhancement**
- **Defining the base experience that can work on a smartphone** and add enhancements to tablets, then laptops and desktops
- Only add bells and whistles **when a system can more easily support them**

Why think mobile-first?

- 20% of worldwide web usage is on mobile devices[1]
- Mobile usage for everything besides talking on the phone has tripled since 2011[2]
- 63% of adults in the US use their phones to use the internet[3]

1 Browser stats for Q4 [2013]
2 US Time Spent on Mobile to Overtake Desktop
3 PEW Internet: Mobile

Web accessibility (a11y)

- **Web accessibility** is about providing support for people in four major use cases:
  - Blindness and low vision
  - Deafness
  - Poor motor skills
  - Cognitive/learning disabilities
- HTML, CSS, and Javascript can be written to support each use case

Developing for a11y

- **Logical content order** and **semantic elements**
- Media **alternatives** (ex.: `alt` attributes)
- **Keyboard** focus (`:focus`) and interactions
- Sufficient **color contrast**
- W3C's **WCAG 2.0 guidelines**

Version control for code

- **Version control** is a method of storing versions of files with comments so that changes are recorded
- Helps **prevent** accidental deletions, additions, mistakes, and errors in live code
- Popular version control systems include **Git** and **Subversion** (or **svn**)
- Using have command line and GUI tools

Version control integration

- Version control lets us safely share code between developers and collaborate on projects
- Can be integrated into systems for deploying code onto live sites
- For example, the code for our class site is stored online in GitHub, and the site is served with GitHub Pages

What else?

Before we go...

1. Visit www.svcseattle.com/evaluation/
2. Choose **"HTML and CSS - Level 1 (Persing) (Winter 14)"** from the dropdown
3. Evaluate this course, me, and SVC
4. Please be honest and constructively critical!

Stay in touch!

dep@dpersing.com

linkedin.
com/devonpe
rsing

@devonpersing

Thank you!

It was really fun.