CSS & HTML Level 1: Week 4

Devon Persing, Instructor
Tiffany Kosa, TA

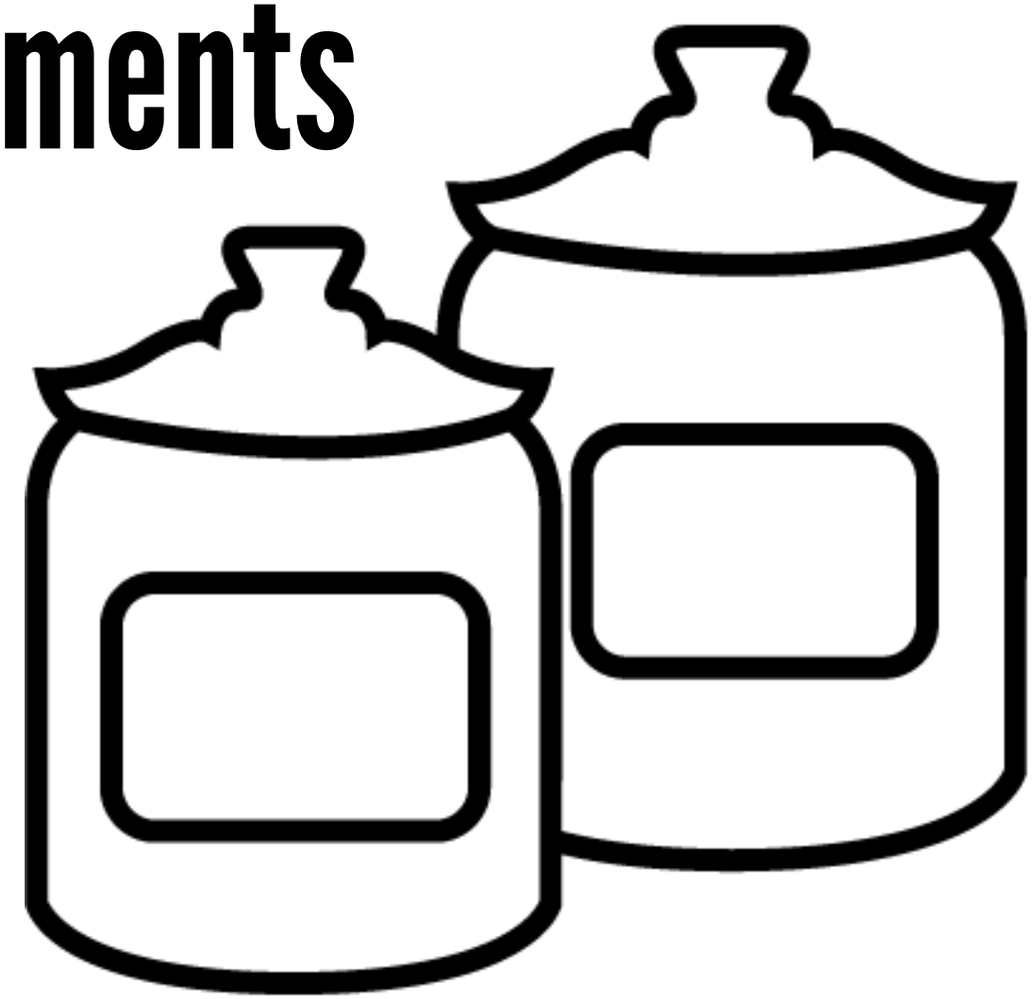
Browser designed by Fernando Vasconcelos from The Noun Project

Questions?

This week

- New HTML5 container elements
- Inline-block elements
- Multi-column and grid layouts
- Floated elements
- CSS pseudo-classes
- HTML data tables

HTML5 container elements



Canister designed by Anna Evans from The Noun Project

Pre-HTML5 structural containers

- XHTML had **no way to give container elements semantic value**
- We'd fake it with **generic containers** and **semantic-sounding `ids`**

```
<div id="header"></div>

<div id="main"></div>
<div id="sidebar"></div>

</div id="footer"></div>
```

Why create semantic containers?

- **Consistency** across layouts and sites
- Continued **separation of style and content** to make restyling content easier
- Better support for **accessibility** by giving assistive tech users more opportunities for navigation content

Support for older IE browsers

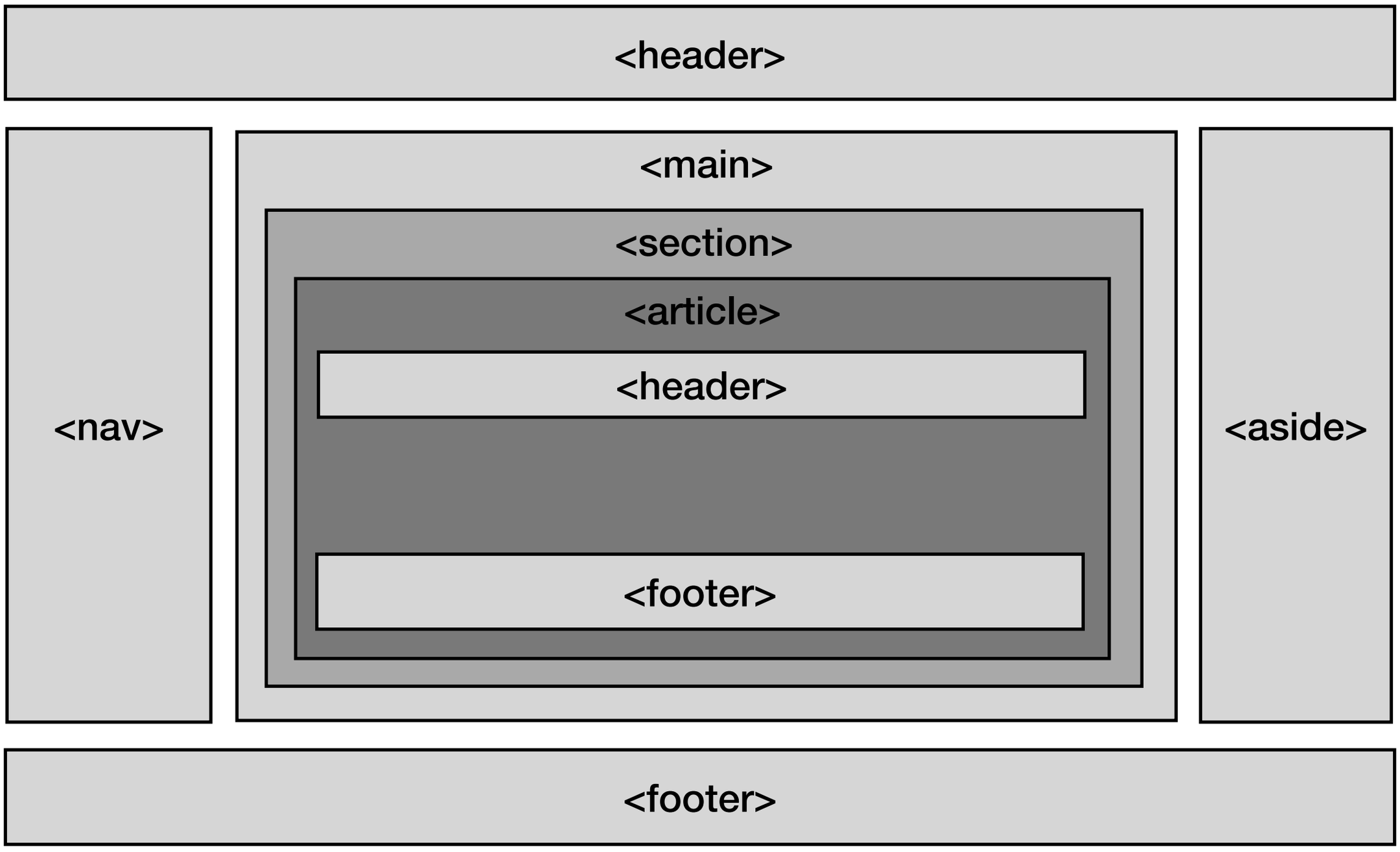
- No native support for HTML5 structural elements in IE8 and older
- These browsers can be "tricked" into displaying and styling HTML5 elements using Javascript
- HTML5 shiv (or shim): https://code.google.com/p/html5shim/

Installing HTML5 shim on your site

1. Go to https://code.google.com/p/html5shim/ and download the zip file
2. Unzip it, and add it to your site files (hint: maybe in a **js** folder?)
3. Paste this into the `<head>` element of all of your pages:

```
<!--[if lt IE 9]>
<script src="your-file-location"></script>
<![endif]-->
```

An example HTML5 layout



HTML container elements

- `<header>`: header of a container
- `<nav>`: navigation grouping
- `<main>`: main content wrapper (new!)
- `<section>`: like content wrapper
- `<article>`: article content
- `<aside>`: supportive content
- `<footer>`: footer of a container

main

- **<main>** wraps the main content of a page
- Main is **used only once per page**

```
<div id="header"></div>

<main></main>
<div id="sidebar"></div>

</div id="footer"></div>
```

Updating our reset styles

- `<main>` is so new it's not in our reset styles, so let's add it before we continue

```
html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, canvas, details, embed,
figure, figcaption, footer, header, hgroup,
menu, nav, output, ruby, section, summary,
time, mark, audio, video, main {
    margin: 0;
    padding: 0;
    border: 0;
    font-size: 100%;
    font: inherit;
    vertical-align: baseline;
}
/* HTML5 display-role reset for older browsers */
article, aside, details, figcaption, figure,
footer, header, hgroup, menu, nav, section, main {
    display: block;
}
```

article

- `<article>` is what is sounds like: a **standalone** piece of content
- There can be multiple articles on a page

- **Blog posts**
- **News articles**
- **Comments**
- **Reviews**

```
<article>
  <h2>Blog Post</h2>
  <p>Either the well was very
  deep, or she fell very slowly,
  for she had  plenty of time as
  she went down to look about
  her and to wonder what was
  going to happen next.</p>
</article>
```

section

- `<section>` wraps thematically related content, often with its own heading
- There can be multiple sections in a page

- **A group of blog posts**
- **An article section on a single topic**
- **A sidebar widget with its own header**

```
<section>
  <h3>About SVC</h3>

  <p>SVC dates back to
  the days before
  Photoshop, the web, and
  iPads; 1971, to be
  exact.</p>
</section>
```

header

- `<header>` wraps any introductory content and/or navigation aids
- There can be multiple headers in a page

- **The header of a site**
- **The header of a section**
- **The header of an article**

```
<header>
  <img src="logo.png"
  alt="Site logo">

  <h1>Site Title</h1>
</header>
```

footer

- `<footer>` wraps any closing information about a container
- There can be multiple footers in a page

- **The footer of a site**
- **The footer of an article (ex.: a list of category links or tags for a blog post)**

```
<footer>
  <small>&copy; 2013 Web
  Footers</small>
</footer>
```

nav

- `<nav>` wraps major navigation elements
- There can be multiple groups in a page

- Typically wraps a list of links

```
<nav>
  <ul>
    <li><a href="index.html">Home</a></li>
    <li><a href="about.html">About</a></li>
    <li><a href="contact.html">Contact</a></li>
  </ul>
</nav>
```

aside

- `<aside>` wraps tangentially related content
- There can be multiple asides in a page

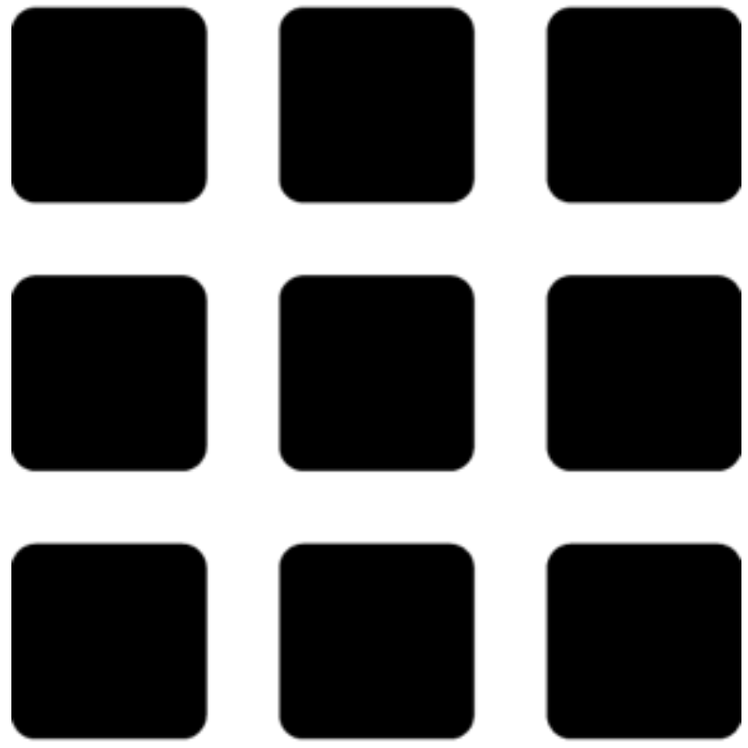
- Typically contains content that doesn't stand on its own without other content
  - **A sidebar**
  - **A pullquote from an article**
  - **A supporting fact for an article**

Notes on using HTML5

- Online resources will be most up to date (ex.: html5doctor.com)
- Elements will come and go HTML5 is an experiment in documentation and process
- HTML5 is an experiment in documentation and process
- If you're not sure, a `<div>` is fine!

Let's put our content in meaningful HTML5 containers.

Multi-column and grid layouts



A brief history of web layouts

- Before CSS, we used **`<table>` elements** to make web layouts :(
- With CSS we can use a variety of declarations to arrange elements on the screen
  - **Pros:** Any content can display anywhere!
  - **Cons:** Any content can display anywhere!

What's inline and block again?

- **Inline elements** line up with their neighbors (ex.: `<span>`, `<a>`)
- **Block elements** create breaks with their neighbors (ex.: `<div>`, `<h1>` etc., `<p>`, `<ul>`, `<li>`)
- We can use CSS to make each behave like the other on the page

Modify display behavior

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

```
span { display: block; }
```

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

```
div { display: inline; }
```

inline + block = inline-block

- Displays elements **side-by-side like inline elements** but also with **the box model of block elements**

- Optimal for lining up content containers into **multi-column layouts within a container**

```
section { display: inline-block; }
```

Box model got you down?

- **`box-sizing: border-box`** works in newer browsers to simplify box model math

```
main {
  width: 200px;
  padding: 20px;
  box-sizing: border-box;
}
```

- a 200px-wide container with **`border-box`** will always be 200px wide, even with padding

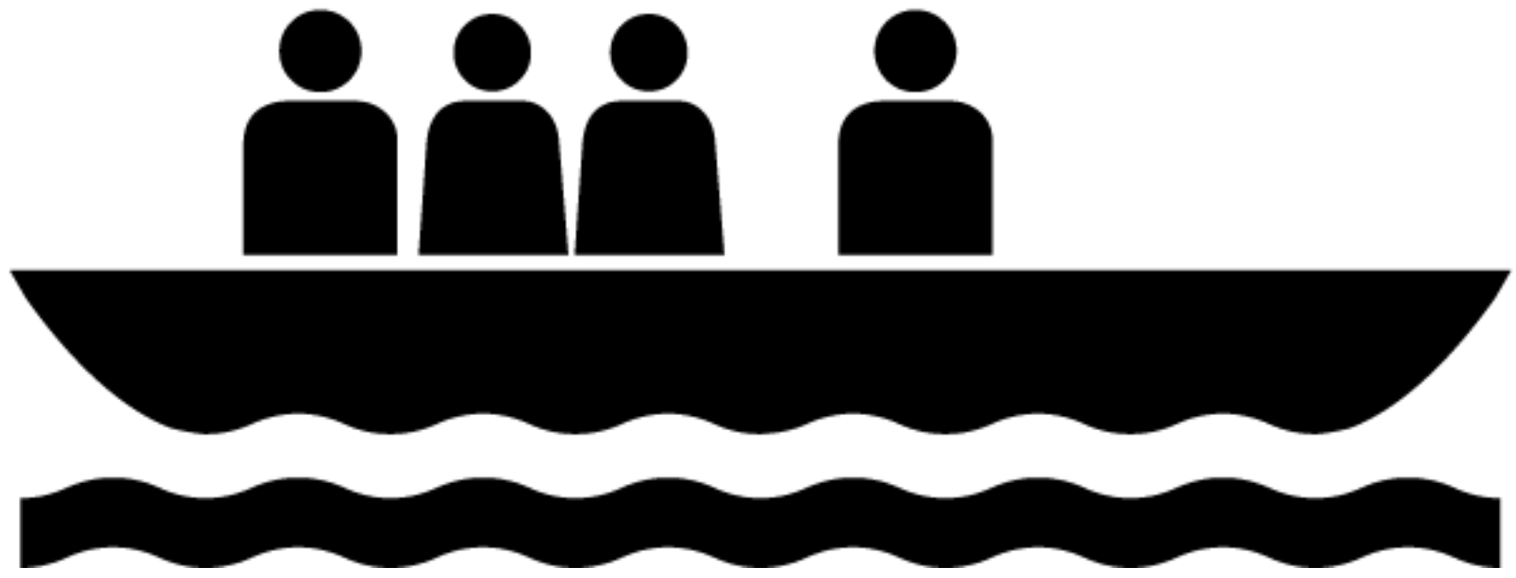
Making columns

```
<body>
  <main><!-- main page content --></main>
  <aside><!-- sidebar content --></aside>
</body>
```

```
body { width: 920px; }
main, aside { display: inline-block; }
main { width: 600px; }
aside {
  width: 300px;
  margin-left: 20px;
}
```

Let's make some columns.

Floating elements



Boat Tour designed by Studio Het Mes from The Noun Project

Floating block-level elements

- Any element can be made to display like a block element
- Block elements can be floated **left** or **right**

```
aside { display: block; float: right; }
```

- Takes the element out of the structure of the page and **flows the rest of the content around it**

Floating content within a container

```
<section>
  <img src="image.png" alt="a floated image">
  <p><!-- paragraph content --></p>
</section>
```

```
img { float: right; }
```

Let's float an image inside a container.

Floating columns

```
<body>
  <main><!-- main page content --></main>
  <aside><!-- sidebar content --></aside>
</body>
```

```
body { width: 920px; }
main {
  float: left;
  width: 600px;
}
aside {
  float: right;
  width: 300px;
  margin-left: 20px;
}
```

Let's float elements in columns (one left, one right) instead of using `inline-block`.

Floating a grid

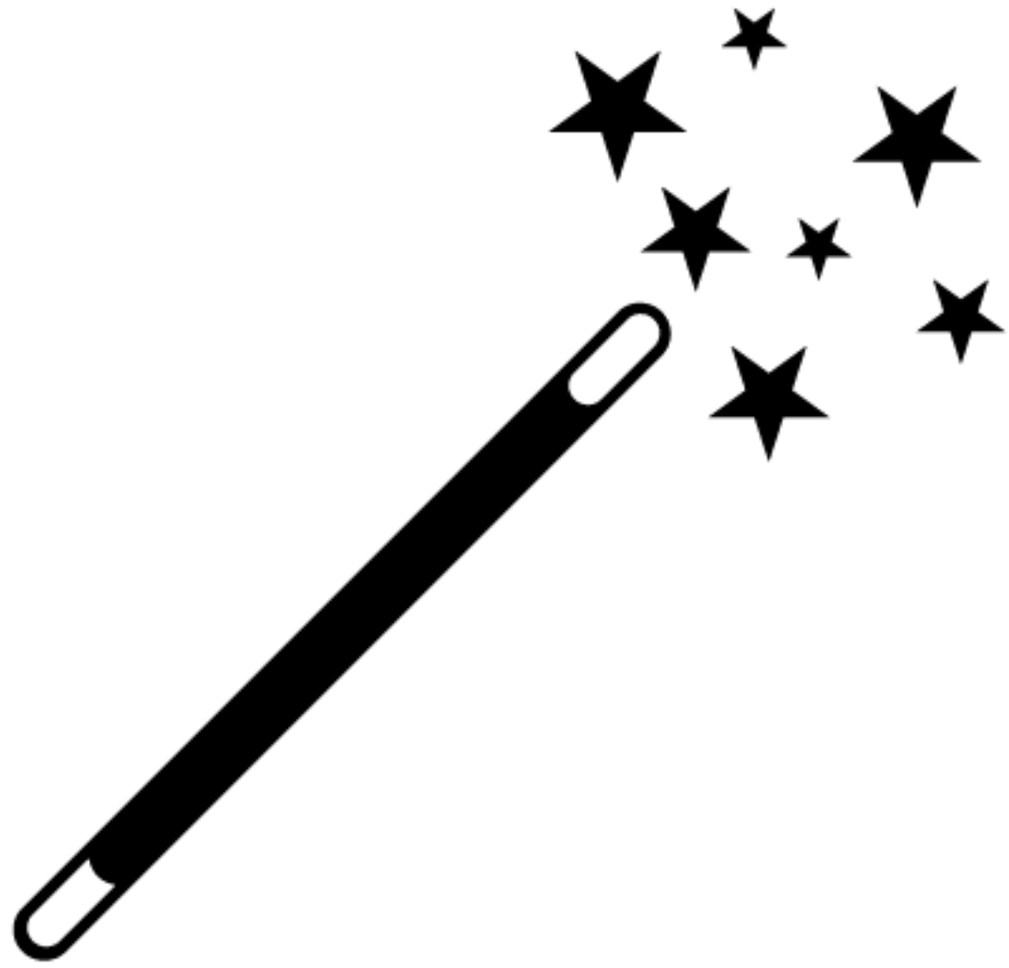
```
<main>
  <section><!-- section content --></section>
  <section><!-- section content --></section>
  <section><!-- section content --></section>
  <section><!-- section content --></section>
</main>

main { width: 600px; }
section {
  float: left;
  width: 300px;
  height: 200px;
}
```

Let's float elements in a grid (instead of using `inline-block`): first left, then right.

inline-block versus floating

- `display: inline-block` maintains the page structure but tries to act like text (see http://css-tricks.com/fighting-the-space-between-inline-block-elements/ for hacks)
- `float` allows content to flow around the floated element and needs to be cleared (see http://www.w3schools.com/css/css_float.asp for hacks)

CSS pseudo-classes



Magic Wand designed by John O'Shea from The Noun Project

Pseudo-classes can style states

- **Pseudo-classes** can be used to style **states of an element**
- Usually used to style **<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 that has a mouse hover
- **:focus** is for a link that has keyboard focus with the Tab key
- Browsers have their own **default :focus** styles for accessibility

```
a:hover, a:focus {
  /* typically they'll be styled together like this */
}
```

`:hover` for other elements

- `:hover` can also be used to style hover states for non-interactive elements

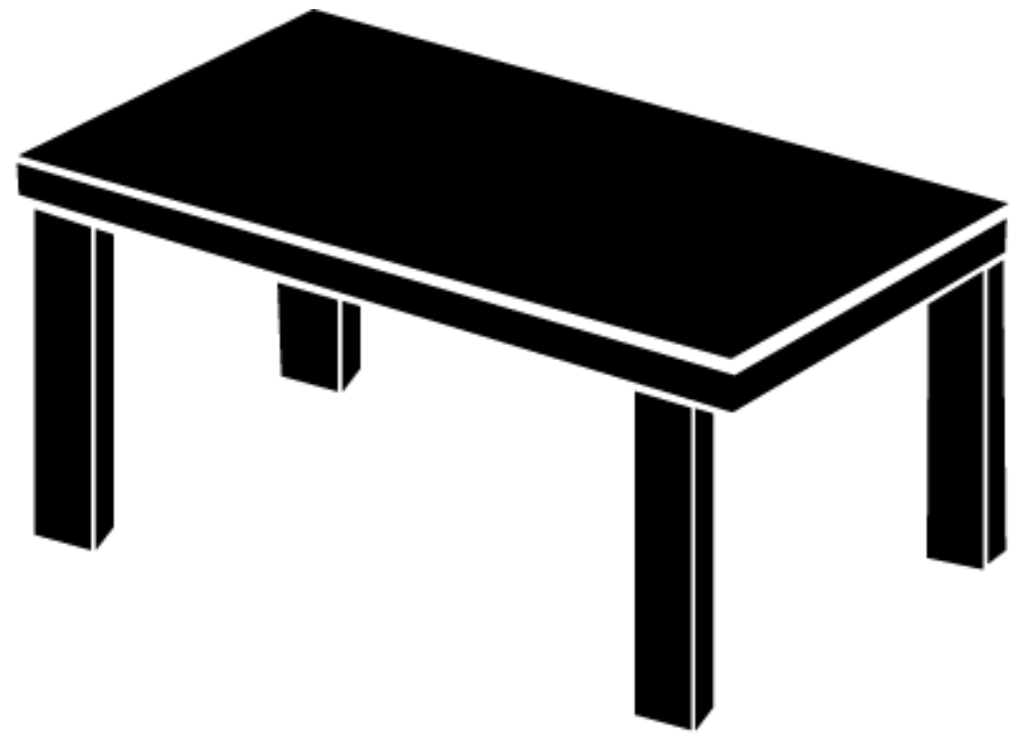
```
tr { /* a table row with one background... */
  background: #99ff66;
}

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

CSS selectors are evolving

- **Pseudo-classes**, **pseudo-elements**, **combinators**, and **attribute selectors** create ways to style targeted content

- To learn more of these techniques, and see which ones work in which browsers: http://www.quirksmode.org/css/selectors/

HTML data tables



What's a `<table>` for?

- Presenting data in a tabular format
- That's it!

- Product feature comparisons
- Financial data

- Listings
- Lists of people with data
- Etc.

Remember content first?

- Tables start with planning...
  - What **content** needs to be displayed?
  - What's the most logical way to **organize** the content?
  - Where do the **headers and data rows** go?
  - How should different parts of the table be **styled**?

Basic table elements

- `<table>` wraps all other table elements
- `<tr>` creates a row of generic table cells
- `<th>` creates a table header cell
- `<td>` creates a single table cell

Diagram of a table

```
<table>
```

`<tr>`	`<th>column head</th>`	`<th>column head</th>`	`<th>column head</th>`	`<th>column head</th>` `</tr>`
`<tr>`	`<th>row head</th>`	`<td>data! </td>`	`<td>data! </td>`	`<td>data! </td>` `</tr>`
`<tr>`	`<th>row head</th>`	`<td>data! </td>`	`<td>data! </td>`	`<td>data! </td>` `</tr>`
`<tr>`	`<th>row head</th>`	`<td>data! </td>`	`<td>data! </td>`	`<td>data! </td>` `</tr>`

```
</table>
```

Table attributes

- **`colspan`** indicates the number of **columns** a cell should span

```
<th colspan="3">This table header spans
three columns</th>
```

- **`rowspan`** indicates the number of **rows** a cell should span

```
<td rowspan="3">This table cell spans three
rows</td>
```

Let's make a table!

Styling table elements

- **`border-spacing`** controls space between adjacent cells

```
border-spacing: 10px 5px;
```

- **`border-collapse`** unifies adjacent borders and removes border-spacing

```
border-collapse: collapse;
```

- **Background, border, margin, and padding styles** can be applied too

Additional table elements

- `<thead>` wraps a row or rows of table header cells
- `<tbody>` wraps rows of regular cells
- `<tfoot>` wraps a row or rows of footer cells (ex.: totals of figures in your table)

For next week

- Style your **links** with **pseudo-classes**
- Add **HTML5 container elements**
- Style a **multi-column layout or grid**
- Make and style a **data table** for your project, if appropriate
- ***HTML5 for Web Designers***: ch. 5
- ***HTML and CSS***: ch. 6, 14, 15, 17

Next week

- Lingering questions
- Project demos (Email me your project homepage URL if you want to share)
- More browser tools
- iFrames
- Web fonts
- Overviews of related tech and issues

Questions?

- Visit http://dpersing.github.io/svc
  - **Class slides**
  - **Code examples from class**
  - **Additional general and class-specific resources**
- Email me at dep@dpersing.com