CSS & HTML Level 1: Week 2

Devon Persing*, Instructor
Tiffany Kosa, TA

*This week you'll be taught by Ava Collins!

Browser designed by Fernando Vasconcelos from The Noun Project

Review!

Clock designed by Brandon Hopkins from The Noun Project

Content. Design. Code.

- **Content** is the reason we make web pages
- **Design** is how we create user experiences
- **Code** is how we deliver content and experience

Delivering content

- **HTML** structures content
- **CSS** creates style and layout
- **Javascript** adds extra interactivity

We started coding!

```
<!DOCTYPE html>

<html lang=”en”>

     <head>
        <meta charset=”UTF-8”>
        <!-- Author: Me | Date: Today | HTML and CSS Level 1: Week One -->
        <title>My First Page</title>
     </head>

     <body>
        <h1>My First Page</h1>

        <p>Yes, we’re going to write some HTML today.</p>
     </body>

</html>
```

HTML elements

- most elements have **opening and closing tags**:

  ```
  <p>stuff</p>
  ```

- some elements have **attributes** that give them more meaning:

  ```
  <a href="index.html">a link</a>
  ```

HTML documents

- **`<!DOCTYPE html>`** tells the browser it's an HTML file
- **`<html>`** wraps all of the **metadata** and **content**
- **`<head>`** wraps all of the **metadata**
- **`<body>`** wraps all of the **content**

`<head>` elements

- **`<title>`** appears in the browser bar
- **`<meta>`** elements have **attributes** that give information about the page:
    - **`charset`** tells the browser what symbols to expect
    - **`description`** tells search engines what the page is about
    - **`author`** tells who wrote the page

`<body>` elements

- all elements you want to **appear on the page** to visitors
- **semantic elements**, like:
  - **headers: `<h1>` to `<h6>`**
  - **text: `<p>`, `<ul>` and `<ol>`**
  - **images: `<img>`**
  - **links: `<a>`**
- **container elements...**

File structure

- **Make subdirectories** for CSS, JS, and media files
- Start with HTML files in the **main directory**
- Make your homepage **index.html**

Rules of file naming

- **No spaces** in file names
- **Capitalization** matters
- Use only **letters**, **numbers**, **hyphens** (-), and **underscores** (_)
- Filenames must **start with a letter**

Types of file paths

Absolute paths

- Full URL of the page or file

`http://google.com`

`http://dpersing.github.io/svc/img/svc-logo.png`

Relative paths

- URL in relation to the file you're in

`svc/img/svc-logo.png`

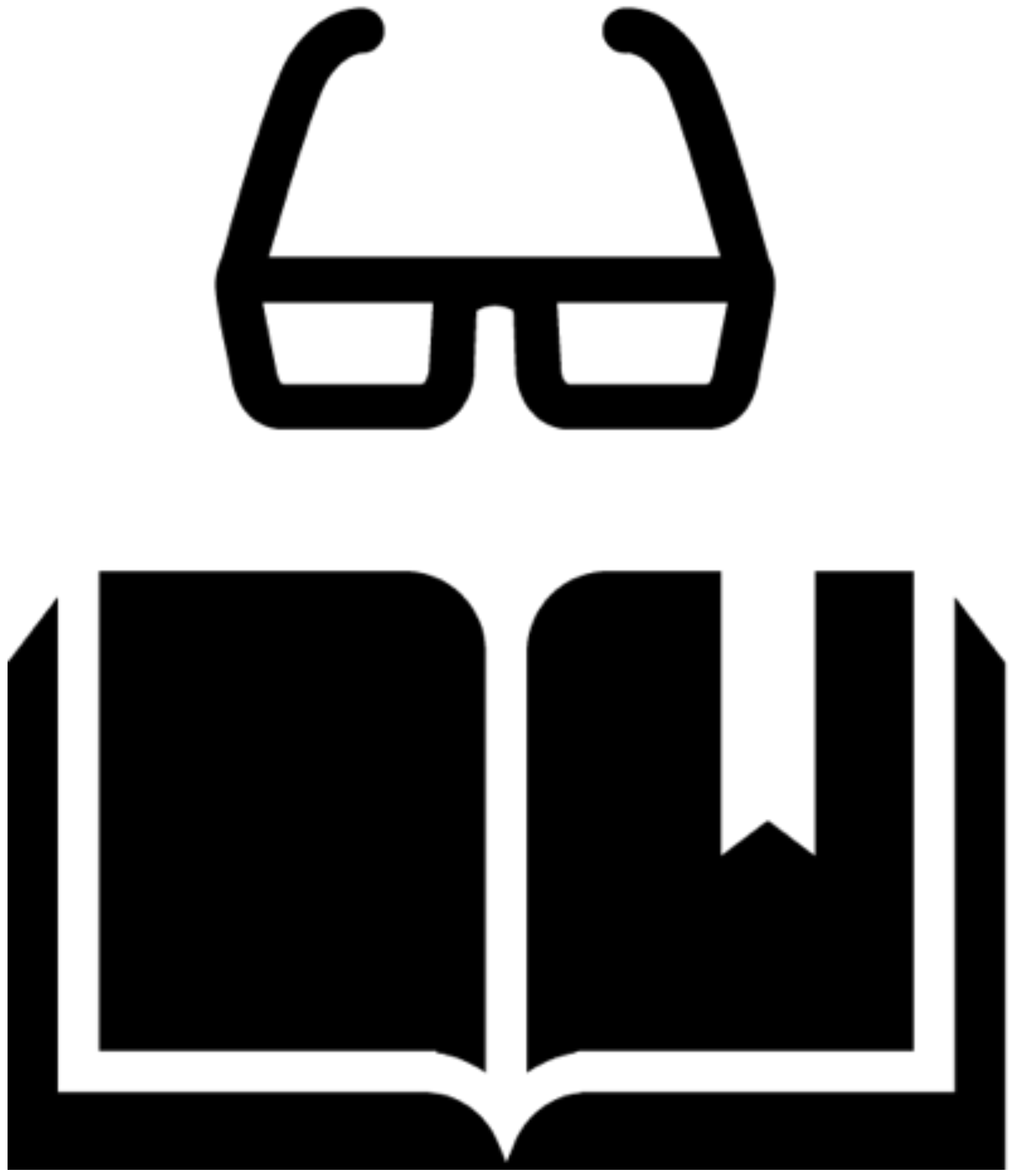
`../svc/img/svc-logo.png`

Good practices

- Leave `<!-- comments -->` for yourself and others
- Standardize your **file structure**
- Standardize your **filenaming**
- **Indent your code** so it's readable

Questions? Show and tell?

Newish stuff



Reading designed by Adrijan Karavdić from The Noun Project

Block and inline elements

Block elements

start a new line by default

So far we know:

- `<h1>...<h6>`
- `<p>`
- `<ul>, <ol>, <li>`

Inline elements

don’t start a new line by default

So far we know:

- `<a>`

More inline elements

- `<em>` tags imply emphasis, and are displayed with italics by default
- `<strong>` tags imply text that should stand out, and are displayed with bold text by default
- HTML used to use `<i>` (italic) and `<b>` (bold) to performed these functions...what do you think changed?

Inline-block elements

Inline-block elements line up with other inline or inline-block elements, but maintain their height and width

So far we know:

- `<img>`

I love my polaroid camera!!!!!!! by Chase Zalewski on Flickr

Generic block and inline elements

`<div>` elements are block elements without semantic meaning

```
<div>

  <h1>My Page</h1>

  <p>Here is my first
  HTML page.</p>

</div>
```

`<span>` elements are inline elements without semantic meaning

```
<h1>My Page</h1>

<p>Here is <span>my
first</span> HTML
page.</p>
```

What do you think `<div>` and `<span>` elements are used for?

Completely new stuff!

Fireworks designed by Juan Pablo Bravo from The Noun Project

Getting graphics web-ready

Web image types

- **JPG** is traditional for **photos**
- **GIF** is traditional for **animation, illustrations** and **transparency**
- **PNG*** was designed for the web for **photos, illustrations,** and **transparent images**

***When in doubt, make a PNG.**

Video time

- Saving for web from Photoshop
- Saving for web from Illustrator

Add an image to your site

1. Open a file in **Photoshop** or **Illustrator**
2. **Save** it for the web
3. Add it to your **site files in an `img` folder**
4. Include it in one of your HTML pages using the **`<img>`** tag*

***Don't forget to give it an `alt` attribute!**

Introduction to CSS

{ }

Cascading Stylesheets

- **CSS** brings style and format to **HTML**’s content
- Provides a **consistent and scalable** means for designing single pages and entire sites

4 kinds of styles

- **Browser default** styles are built into every browser
- **External** styles are linked to in the `<head>` of an HTML document
- **Internal** styles are written in the `<head>` of an HTML document
- **Inline** styles appear in the opening tag of an HTML element

Let’s start with internal styles

```
<style>
  /* styles all go here, indented for neatness! */
</style>
```

- Added in the `<head>` of an HTML document
- Only apply to the HTML document they are written in
- Let’s try some!

Anatomy of a CSS rule

selector

```
h1 { font-size: 2em; }
```

declaration

- **Selector** is the HTML element you want to style
- **Declaration** is how you want to style it
- A single rule can have **multiple declarations**
- Each declaration **ends with a semicolon (;)**

Anatomy of a CSS declaration

```
h1 { font-size: 2em; }
```

$\underset{\text{property}}{\texttt{font-size}}$: $\underset{\text{value}}{\texttt{2em}}$

- Each **declaration** has a **property** and a **value**
- The **property** is the aspect you want to change
- The **value** is exactly **how** you want the aspect to change

Major kinds of selectors

- **Type selectors** match element names

```
h1 { color: #ff0000; )

h1, h2, h3 { color: #ff0000; }
```

- **Descendent selectors** point to an element that is the child of another element

```
p a { text-decoration: none; }
```

Font properties

- `font-size: 2em;` (or `px` or `%`)
- `font-family: /* font stack */;`
  - e.g., `'Helvetica Neue', Helvetica, Arial, sans-serif;`
  - e.g., `Georgia, serif;`
- `font-style: italic;`
- `font-style: bold;` (or `normal` or `600`)
- `font-variant: small-caps;`
- `line-height: 1.5em;` (or `px` or `%`)

Text properties

- `text-decoration: underline;` (or `none`)
- `text-transform: uppercase;`
- `text-align: center;` (or `left` or `right`)
- `text-indent: 1em;` (or `px` or `%`)

Changing text color

- Color **values** can be expressed several ways
- For text color, the **property** is `color`

```
color: #ff0000;
color: rgb(255,0,0);
color: red;*
```

***Technically correct, but not preferred.**

"How will I remember all this?"

- You probably won't (I don't!)
- Use online references, like:
  - Mozilla's **Getting Started with CSS** guide*
  - Mozilla's **CSS Reference***

***These are both on the class website!**

How about external styles?

- External styles can be used by **multiple HTML pages**
- Create **consistent styles across your whole site**
- **Make a change in one place** instead of on every web page

Move'em on out!

1. Create a new text file and save it as **styles.css in your css folder**
2. Copy/paste your internal styles to the new file
3. Delete your **`<style>...</style>`** wrapper from your HTML document
4. Save both files
5. Refresh your page...

Linking an external stylesheet

```
<link href=″css/styles.css″
rel=″stylesheet″>
```

- External stylesheet links go inside the `<head>` element of your HTML pages
- Add the link to each HTML document to which the styles should apply

The style cascade

The cascading part of CSS

- Inheritance
- Precedence
  - Rule Order
  - Specificity of HTML elements
  - Specificity of stylesheet location

Inheritance

- Most styles are **passed from parents to children**

```
body { color: #0000cc; }
```

all text in the <body> will be this color...

- Inheritance is overridden **when a child is styled with different values for the** same property

```
p { color: #ff0000; }
```

...except <p> elements, which will be this color instead

Rule order

- **If the same property is styled for a single element multiple times, the** last the browser reads one takes precedence

```
p { color: #666666; }
ul { color: #000000; }
p { color: #ff0000; }
```
this one wins because it's last

Specificity of HTML elements

- If one style is more specific than another, it takes precedence

`p { color: #666666; }` this styles <p> elements

`a { color: #cc0000; }` this styles <a> elements

`p a { color: #ff0000; }` this styles <a> elements that are inside <p> elements only

Specificity of stylesheet location

- Styles that are **“closer” to the elements they style** take precedence
  - Browser default styles
  - External stylesheets
  - Internal stylesheets
  - Inline styles
- **Let’s try some examples...**

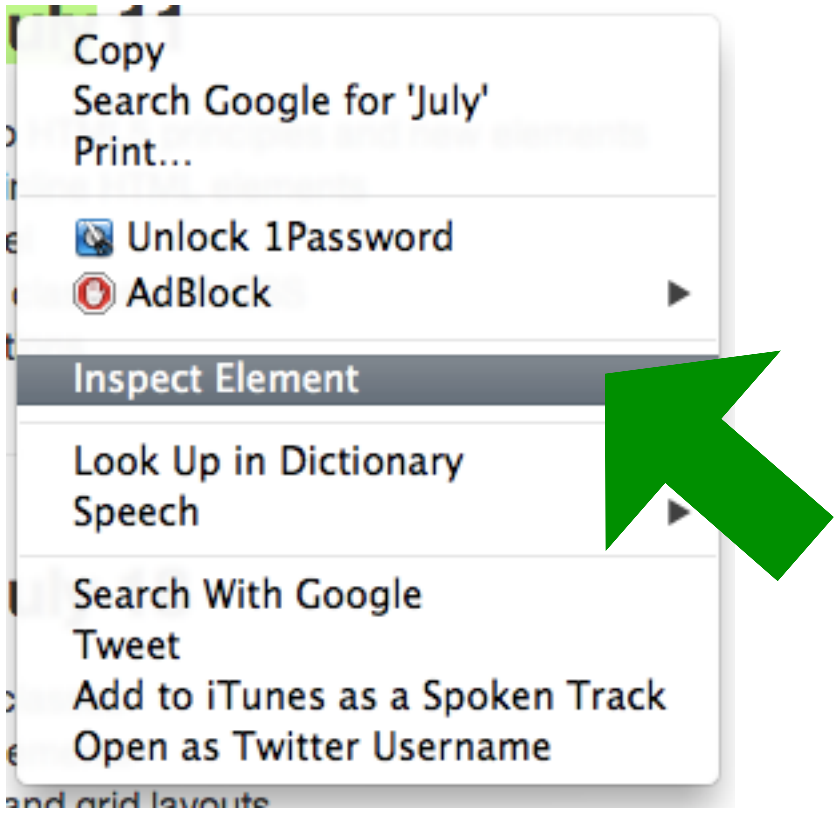
least specific

$\downarrow$

most specific

Use your browser!

- Right click on an element and choose "Inspect Element"
- See what styles are being applied and which are being overridden



Validation

Validating HTML and CSS

- Validating is an easy way to make sure your code is properly formatted and correct
- **HTML: http://html5.validator.nu/**
- **CSS: http://jigsaw.w3.org/css-validator/**
- Let’s test’em out...

Enjoy the holiday next week!

Seattle fireworks over Lake Union by Jeff Croft on Flickr

Thank you, Ava!

Thumbs Up designed by Chris Brunskill from The Noun Project

For next time

- Create a **header image or logo** for your site and add it to all your pages
- Style your site with **an external stylesheet**
- **Validate** your HTML and CSS
- Check out the **online resources for this week**
- ***HTML and CSS***: read ch. 10-12

Next time

- Semantic HTML5 container elements
- The CSS block model
- Using ids and classes with CSS
- CSS abbreviations
- Overriding browser defaults for style

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