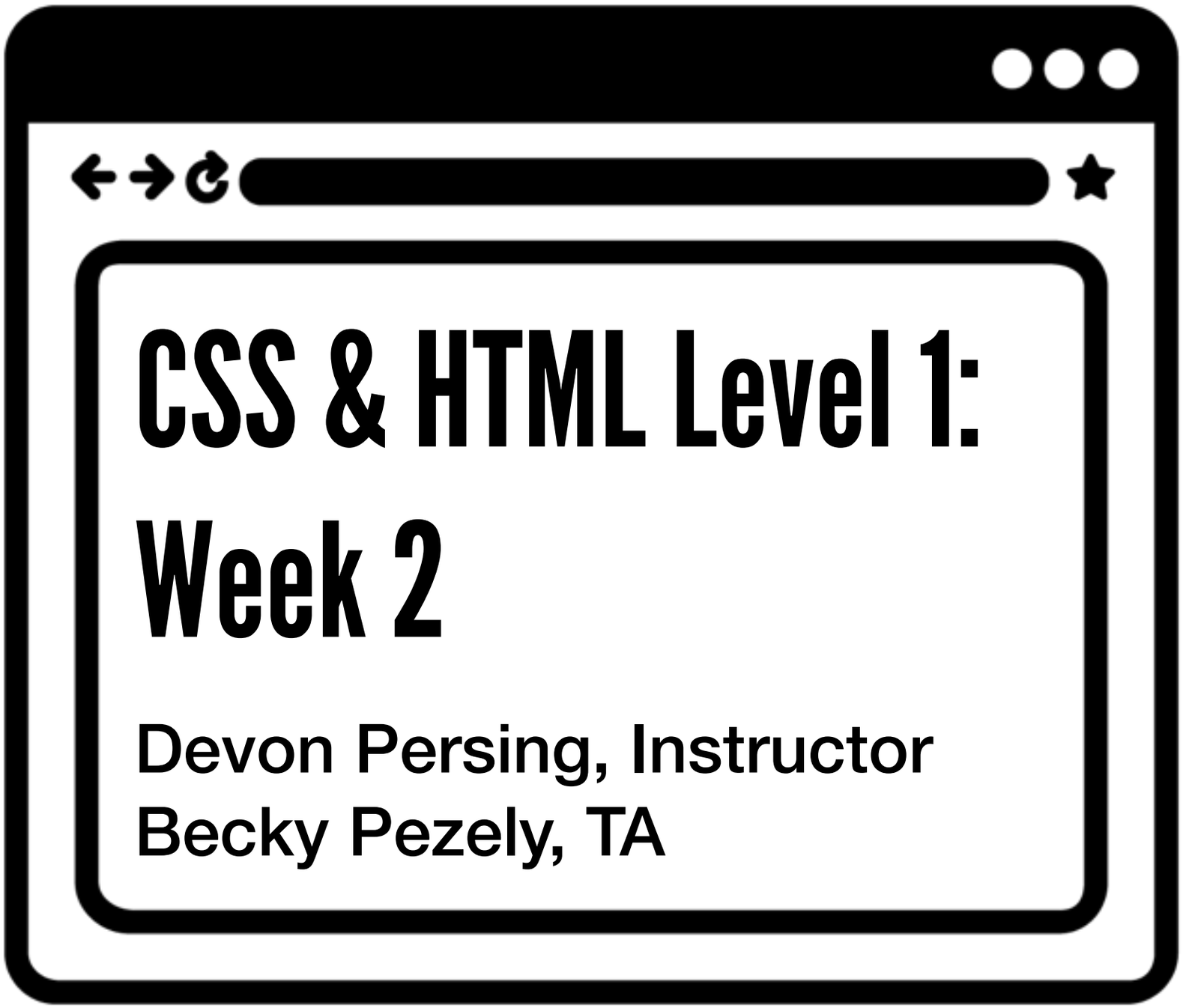


Browser designed by Fernando Vasconcelos from The Noun Project

Today!

- Review from week 1
- Prepping **images** for the web
- Some more **HTML**:
  - New elements
  - **Block versus inline** elements
- Introduction to **CSS**

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

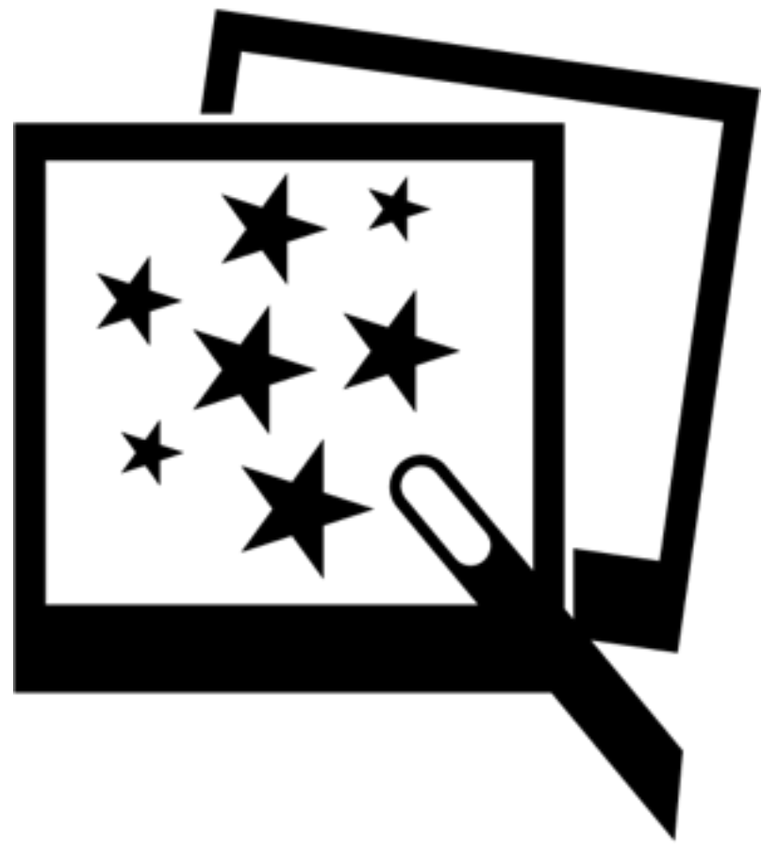
../svc/img/svc-logo.png

Good practices

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

Questions?

Getting graphics web-ready



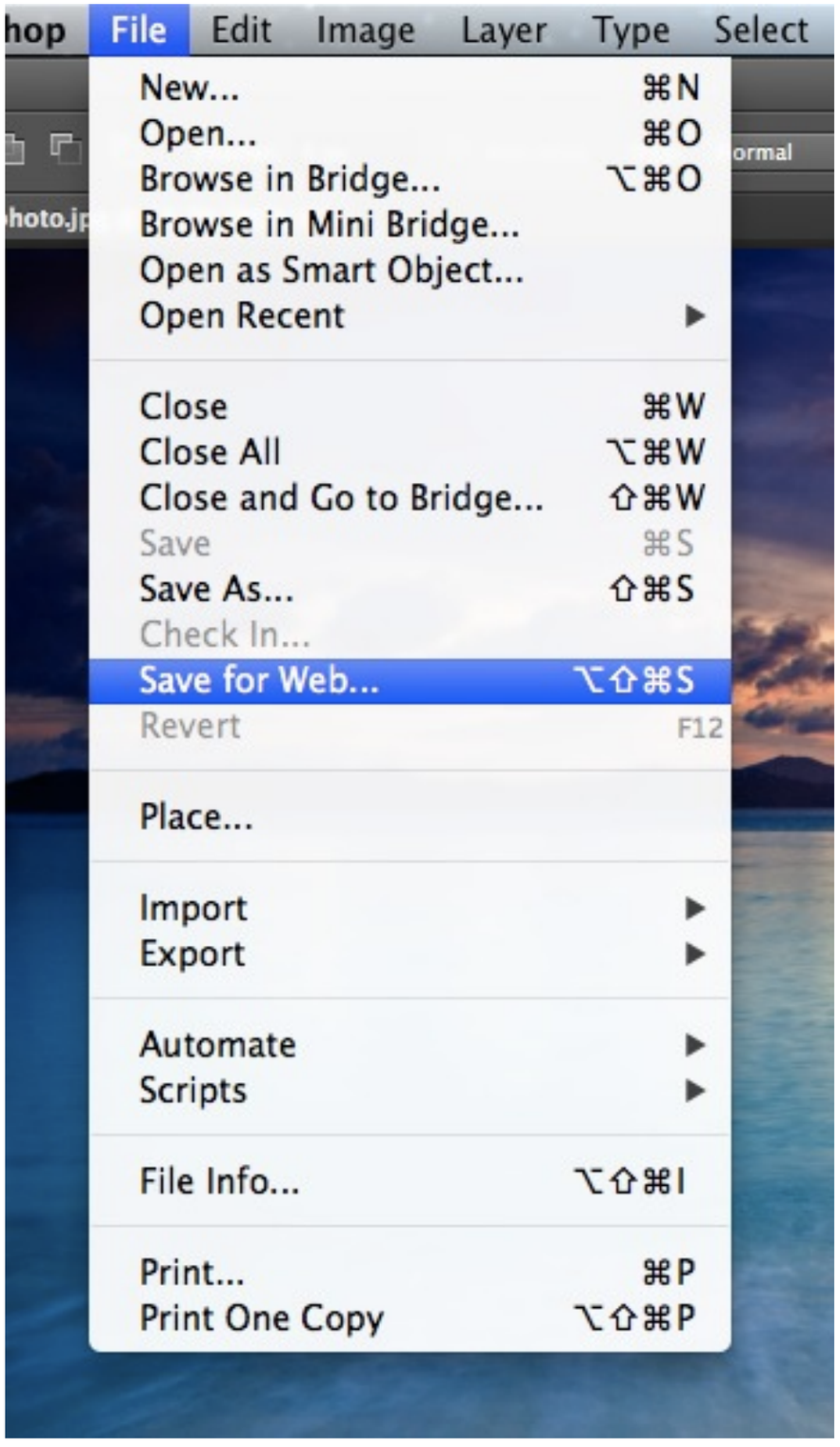
Why do I need to save for web?

- **Strips out layers, metadata, and other bulk from your image**
- **Minimizes file size to help load time in the browser**
- **Optimizes images for RGB display at the correct resolution for the browser**

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**

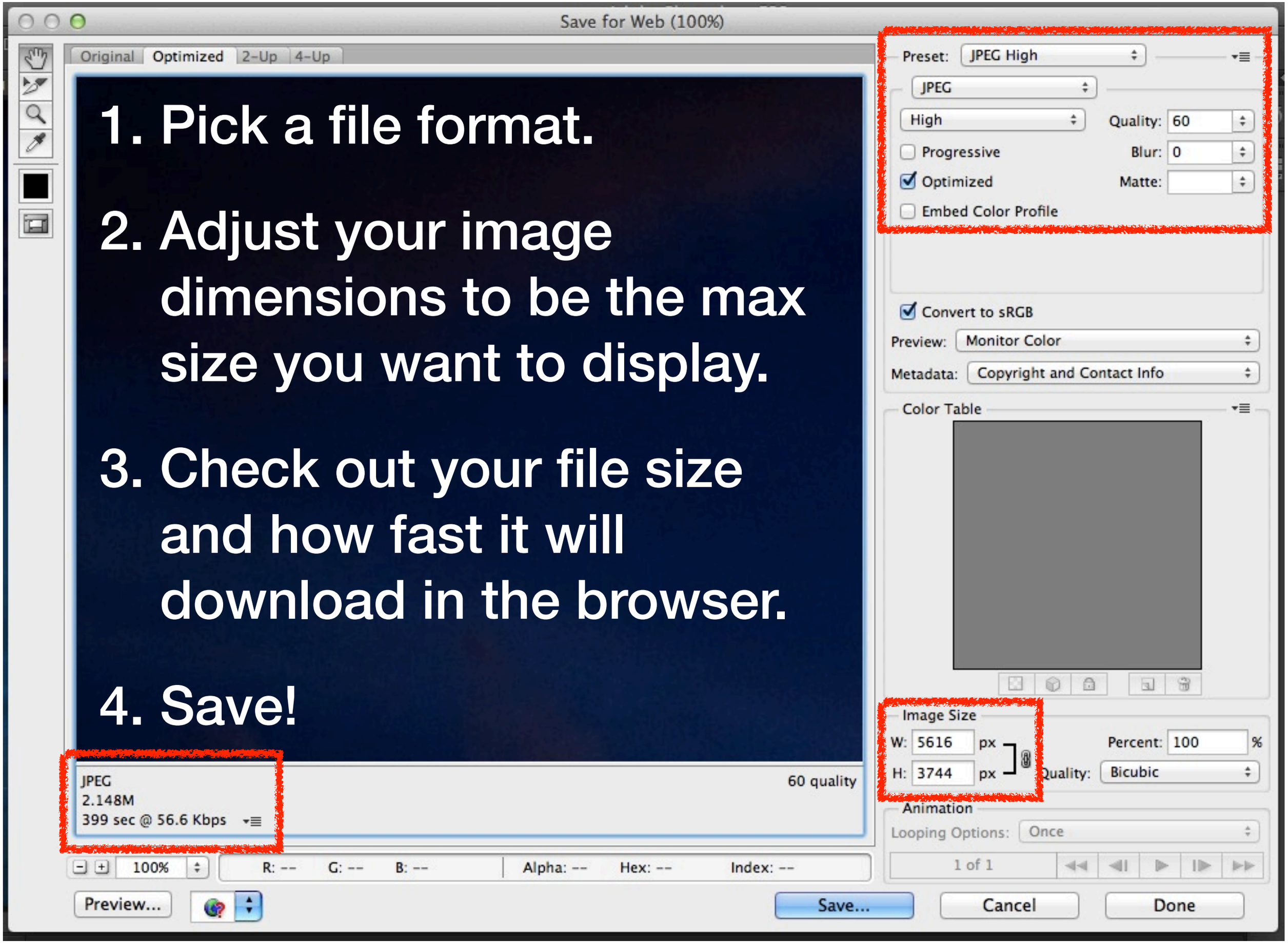
***When in doubt, make a PNG.**



Use “**Save for Web...**”

(or “**Save for Web and Devices...**” in some versions)

instead of “Save” or “Save As...”


Save for Web (100%)
Original
Optimized
2-Up
4-Up
1. Pick a file format.
2. Adjust your image dimensions to be the max size you want to display.
3. Check out your file size and how fast it will download in the browser.
4. Save!
JPEG
2.148M
399 sec @ 56.6 Kbps
60 quality
100%
R: --
G: --
B: --
Alpha: --
Hex: --
Index: --
Preview...
Preset: JPEG High
JPEG
High
Quality: 60
Progressive
Blur: 0
Optimized
Matte:
Embed Color Profile
Convert to sRGB
Preview: Monitor Color
Metadata: Copyright and Contact Info
Color Table
Image Size
W: 5616 px
H: 3744 px
Percent: 100 %
Quality: Bicubic
Animation
Looping Options: Once
1 of 1
Save...
Cancel
Done

Videos demos!

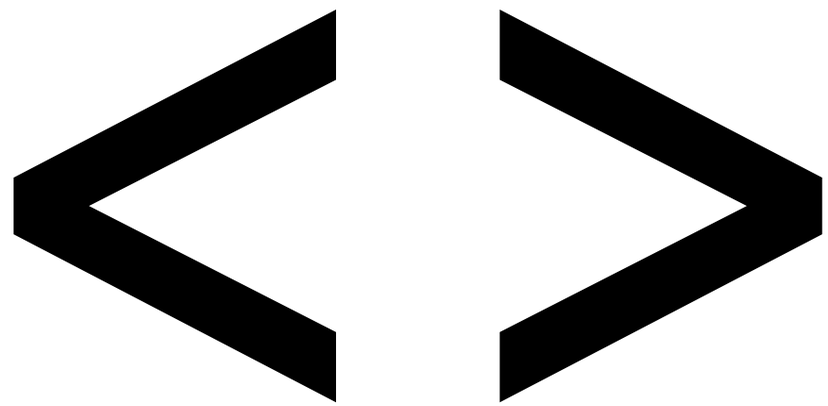
- Saving for web from Photoshop
- Saving for web from Illustrator
- Look for these on the class site!

Let's add an image to our site

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

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

Some more HTML elements



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>`

General 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>
```

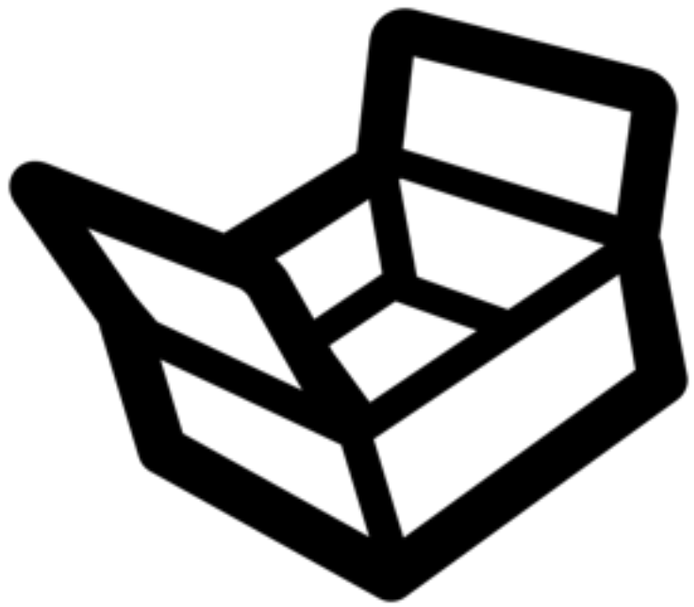
`<span>` elements are inline elements without semantic meaning

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

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

`<div>` elements

- Functions **like a box** to put related or adjacent content in
- Used to create **divisions** in an HTML page for **layout and style**
- Can **nest** inside each other



Open Box by Andrew McKinley from The Noun Project

`<span>` elements

- Functions **like an invisible thread** that wraps content without affecting its position
- Used to **style other inline content**
- Can **nest** inside each other and other elements

Rope designed by Sergi Delgado from the Noun Project

The elusive inline-block element

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

More inline elements

- `<em>` tags imply **spoken emphasis**, and are displayed with **italics** by default
- `<strong>` tags are used for **contextual importance**, and are displayed with **bold** text by default

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

What about <i> and <b>?

In XHTML:

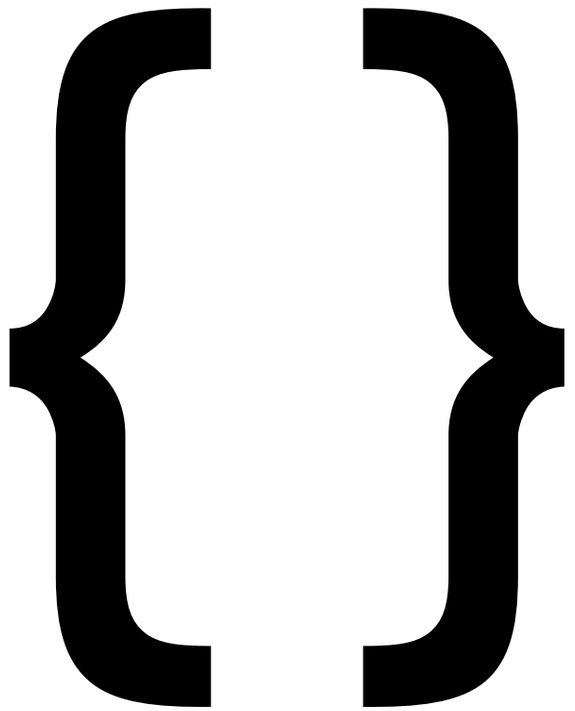
- **<i> means *italic***
- **<b> means bold**

In HTML5:

- **<i> means *an alternate mood or voice***
- **<b> means stylistically offset**

Why do you think they changed?

Introduction to CSS



Cascading Stylesheets

- **CSS** brings style, formatting, and layout to **HTML**
- Provides a **consistent and scalable** way to design single pages and entire sites
- **Separates look & feel from content** so that sites can be restyled over time

Anatomy of a CSS rule

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

selector — 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** (and **multiple selectors!**)
- **Declarations** are grouped within { }

Anatomy of a CSS declaration

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

property value

- Each **declaration** has a **property** and a **value**
- The **property** is the aspect of the HTML element you want to style
- The **value** is exactly **how** you want the aspect to be styled
- Each declaration ends with a **;**

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 (“inside”) of another element

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

Major font properties

- `font-size: 2em;` (or `px` or `%`)
- `font-family: Helvetica, sans-serif;` (or another font stack)
- `font-style: italic;` (or `normal`)
- `font-weight: bold;` (or `normal` or `600`)
- `line-height: 1.5em;` (or `px` or `%`)

Major text properties

- `text-decoration: underline;` (or `none`)
- `text-transform: capitalize;`
- `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!**

Styles can be in 4 locations

- **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! */
  /* these are CSS comments, btw */
</style>
```

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

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!

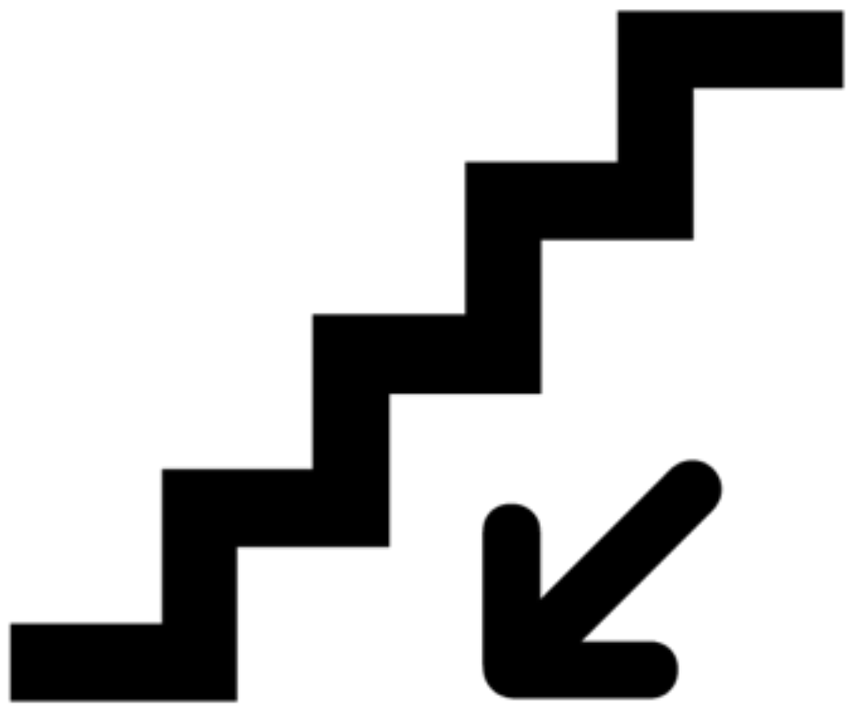
1. Create a new text file and save it as **styles.css in our css folder**
2. Copy/paste oue internal styles to the new file
3. Delete our `<style>...</style>` wrapper from our HTML document
4. Save both files
5. Refresh our page...

Linking an external stylesheet

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

- Tells an HTML page to load a 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**
  - **Which styles are passed down to children?**
- **Precedence**
  - **Which rule is seen first by the browser?**
  - **How specific are the selectors?**
  - **Is the rule inline, internal, external, etc.?**

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 selector multiple times,** the last one the browser reads takes precedence

```
p { color: #666666; }
ul { color: #000000; }
p { color: #ff0000; }
```
this one wins!

Specificity of HTML elements

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

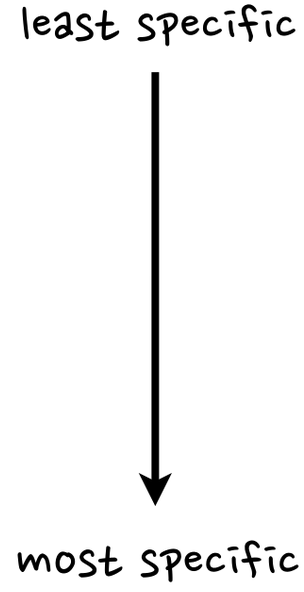
```
p { color: #666666; }
```
this styles <p> elements

```
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

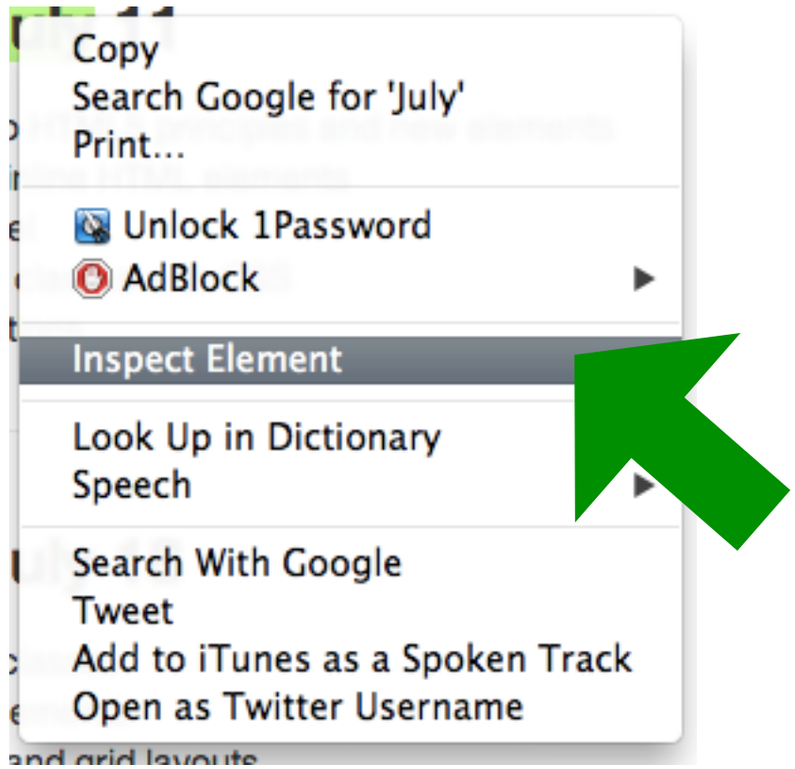
- Styles that are **“closer” to the elements they style** take precedence
  - Browser default styles
  - External stylesheets
  - Internal stylesheets
  - Inline styles



- Let’s try **some examples...**

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

- **Validation** is an easy way to make sure your code is properly formatted and will work as expected
- **HTML: http://html5.validator.nu/**
- **CSS: http://jigsaw.w3.org/css-validator/**

For next time

- Create a **header image or logo** for your site and add it to all your pages
- Add **`<div>`** elements to your pages
- Style your site with **an external stylesheet**
- **Validate** your HTML and CSS
- ***HTML and CSS***: read ch. 10-12

Next time

- New 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