Learn HTML & CSS @ Seattle JS

February 26, 2014

Workshop goals

- Learn about how HTML and CSS work
- Learn major HTML elements
- Learn CSS for styling text and layouts
- Make a one-page, styled website with a linked stylesheet
- Discover resources for continued learning

Workshop resources:

dpersing.github.io/seattlejs

Hey, I'm Devon Persing.

I'll be your tour guide this evening.

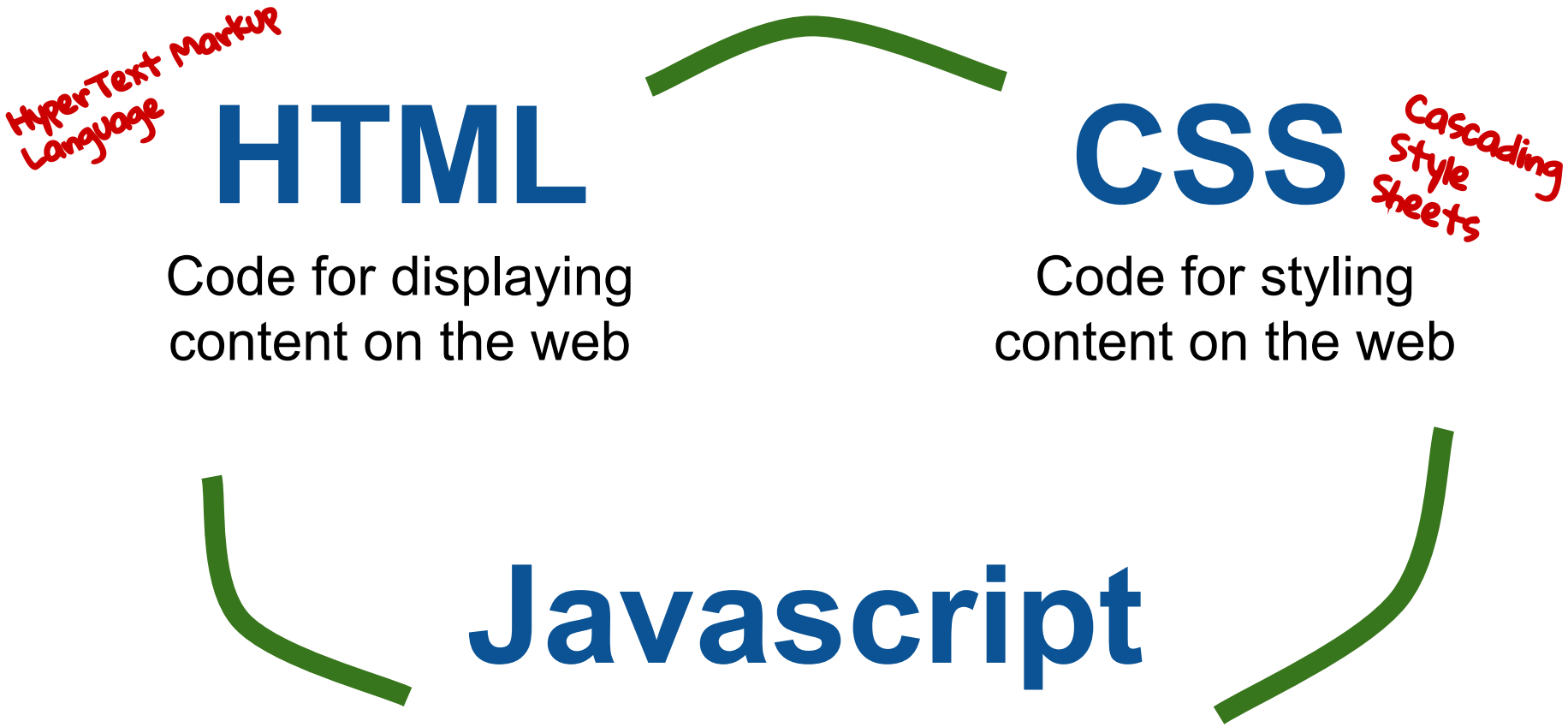
I work at 

I volunteer at 

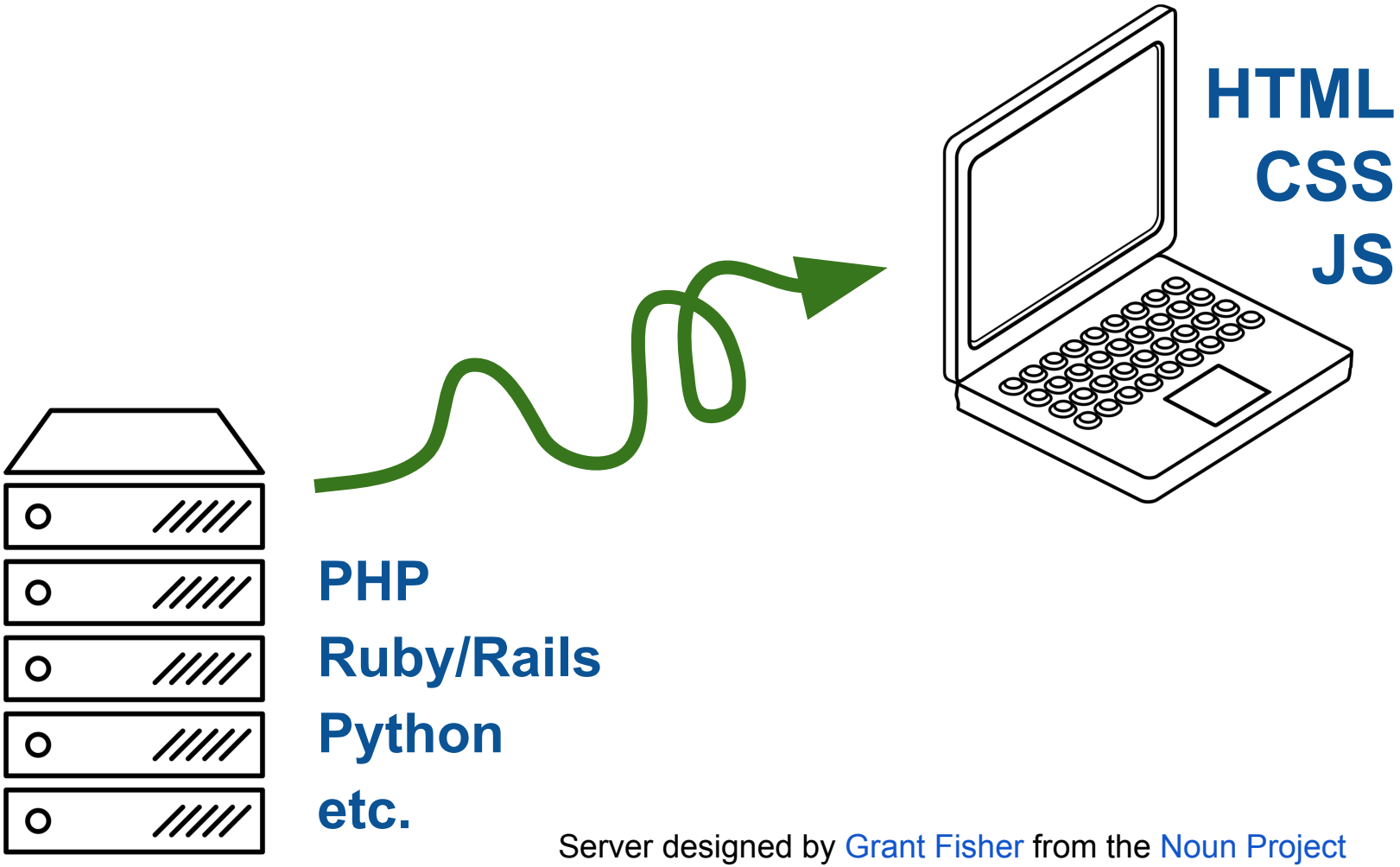
I teach at 

What are HTML and CSS?

Front-end web development



Front-end code works the browser



Server designed by Grant Fisher from the Noun Project
Laptop designed by Simon Child from the Noun Project

End of lecture. Time to code!

- Using a **text editor** and a **browser** for development
- HTML **elements** and **attributes** for text and images
- CSS **selectors** and **declarations**
- CSS styles for **fonts** and **containers**
- Using the **float** technique for positioning content on the page

Setting up HTML

Writing HTML

- HTML is written with **tags** bookended by **angle braces** (< >)
- Chunks of HTML are called **elements**
- Most HTML elements have an **opening and closing tag**, with the element's content inside:

```
<p>Here is some text in an HTML paragraph element.</p>
```

HTML element attributes

- Some HTML elements use **attributes** to perform their function
- Attributes have **names** and **values**
- Attributes are placed in the **opening tag** of an HTML element:

```
<a href="http://www.google.com">Here is a link that goes to Google.</a>
```

The HTML document

- Usually has an **.html** extension
- **`DOCTYPE`**: tells the browser that this it will be displaying an HTML document
- **`head`**: contains **metadata** about the HTML document, like the title and description
- **`body`**: contains the **content** to be displayed in the browser, such as text, images, etc.

Major HTML elements

Headers

- Headers create an outline and are numbered 1-6 from most to least important:

```
<h1>First level header text</h1>
<h2>Second level header text</h2>
<h3>Third level header text</h3>
<!-- etc. -->
```

- A page should *always* have an **<h1>** as its first header

Paragraphs

- Paragraphs encode blocks of text:

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

<h2>About Me</h2>

<p>I like puppies, knitting, and well-
formatted HTML. I live in Seattle.</p>
```

Lists

- Lists encode lists of items
- There are two main kinds of lists:
  - `<ul>` is used for **bullet/unordered** lists
  - `<ol>` is used for **numbered/ordered** lists
- Lists always have one or more list items:
  - `<li>` is used for **each item** in a list

Lists (continued)

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

<h2>About Me</h2>

<p>I like:</p>

<ul>
    <li>puppies</li>
    <li>knitting</li>
    <li>well-formatted HTML</li>
</ul>

<p>I live in Seattle.</p>
```

Links

- Links put the "hypertext" in HTML
- Links use an **href** attribute with a web address (or URL) value to tell the browser what they should link to:

```
<a href="http://www.google.com">Here is a link that goes to Google.</a>
```

Images

- Images have **one HTML tag** and **two attributes**:
  - **`src`** to tell the browser where the image lives
  - **`alt`** to provide a text equivalent of the image for search engines and assistive technologies

```
<img src="image.jpg" alt="a photo of an adorable kitten">
<!-- or -->
<img src="image.jpg" alt="a photo of an adorable kitten" />
```

The CSS document

- Has a **.css** extension
- Is usually linked to in the **`head`** of an HTML document
- Consists of 1+ **selectors** that point to HTML elements and 1+ **declarations** of how to style them in curly braces (**{ }**):

```
selector { declaration; }
```

CSS for text

Major CSS properties for text

- **`font-family`**: the **fonts** or **font stack** used to style content
- **`font-size`**: the size of the font
- **`font-style`**: whether the font is italic
- **`font-weight`**: whether the font is bold
- **`color`**: the color of any content

font-family

- **font-family** can include one or more fonts by name, as well as a font-family, separated by commas

```
body { font-family: Helvetica, Arial, sans-serif; }

p { font-family: Georgia, Times, serif; }
```

`font-size`

- **`font-size`** is the height of the font, usually measured in pixels (**`px`**) or ems (**`em`**)
  - **px** are fixed, whereas **ems** scale based on the defaults of the page

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

p { font-size: 14px; }
```

`font-weight` and `font-style`

- **`font-weight`** is the weight of the font, usually **`bold`** or **`normal`**
  - headers default to **bold**
  - paragraphs and other smaller text default to **normal**

```
a { font-weight: bold; }

h1 { font-weight: normal; }
```

`color`

- **`color`** can be used to style text and is usually given with a hex value, which starts with a **#**
  - hex values can be found online
  - programs like Photoshop and Illustrator provide hex values in the color picker

```
a { color: #FF530D; }
```

Let's make this website look a bit less like it's 1996, shall we?

Another kind of selector

- **Type selectors** target a specific type of element

```
p { color: #E82C0C; }
```

- **Class selectors** let us apply the same styles to *any* HTML element(s) using an HTML attribute

Class selectors

- **Class selectors** are added to CSS with a period in front:

```
.highlight { color: #E82C0C; }
```

- Classes are added to HTML with an attribute:

```
<h2 class="highlight"></h2>
<p class="highlight"></p>
```

Grouping content

- **`div`** elements:
  - create line breaks
  - act like we're putting content **in a box**
- **`span`** elements:
  - *don't* create link breaks
  - act like we're putting content **in shrink wrap**
- There are a bunch of new HTML5 elements that can be used for this, too

Styling blocks of content

- **`background`**: gives an element a background color/and or image
- **`width`** and **`height`**: explicitly set the width and height of an element
- **`margin`** and **`padding`**: explicitly set how much space is in and around an element

CSS backgrounds

- The **`background`** property can add:
  - a background color
  - a background image
- Background images can get pretty complex, so let's stick to colors for now:

```
div { background: #FF530D; }
```

`width` and `height`

- By default:
  - Block-level elements take up the full width of their container
  - Inline elements take up as much space as their content needs
- We can assign specific dimensions to elements, usually using pixels (`px`) for either or percentages (`%`) for width

`width` and `height` (continued)

```
ul {
   width: 200px;
}

body {
   width: 90%;
   max-width: 960px;
}
```

`margin` and `padding`

- Margin and padding are part of the CSS "box-model" that dictates how elements take up space
- Usually measured in pixels (**`px`**) or ems (**`em`**)
- Can be uniform around an element, or can be set on any side (top, right, bottom, left)
- Margin width can be set to **`auto`** to center an element horizontally

`margin` and `padding` (continued)

```
body {
    width: 960px;
    margin: 0 auto;
}

ul {
    margin-left: 40px;
}

.main {
    width: 960px;
    padding: 1em;
}
```

`border-box` to the rescue

- By default, **`width`** and **`height`** only apply to the content, not the padding and margin
- Adding **`padding`** and **`margin`** to things that already take up their whole container can cause the layout to go wonky
- We'll reset the browser's expectations with **`border-box`**

`border-box` (continued)

- *[some handwaving]*

```
*, *:before, *:after {
   -moz-box-sizing: border-box;
   -webkit-box-sizing: border-box;
   box-sizing: border-box;
}
```

Floating content

- `float` lets us take content out of the natural order of the page and create little islands of content, or even columns
- `clear` lets us stop the behavior of floats (if we need to)

Floating content (continued)

```
.main, .resources { float: left; }

.main { width: 60%; }

.resources { width: 40%; }

.footer { clear: both; }
```

"What should I do next?"

- Check out the resources on the workshop page
- Try to replicate the HTML and CSS for a webpage you like to see how it works
- Try an online course at Codeacademy, Treehouse, or many others

Please thank the organizers and volunteers!