Learning HTML and CSS

DebugAcademy.com

@DebugAcademy

Enjoy today's training!

✓ Training material prepared by Debug Academy.

- We train companies *and* individuals looking to change careers
  - Experience training companies such as Northrop Grumman
  - Also provided career changing education for countless individuals
- Get in touch to learn more!

Follow us! @DebugAcademy & DebugAcademy.com

Twitter: @ashabed

E-mail: ashraf.abed@debugacademy.com

https://debugacademy.com

https://drupal.tv

Ashraf Abed

Technical Architect
Acquia Inc. (2014 -2017)
Senior Software Engineer
Carr Systems (2012 – 2014)
Cyber Software Engineer
Northrop Grumman (2011)
Web Developer
Depthskins Design Studio (2009)
Freelance Web Developer
Self-employed (2003 – 2009)
B.S. Electrical Engineering (UMD)

ashraf.abed@debugacademy.com

Founder of Debug Academy
Debugacademy.com

Certified Drupal 8 Grand Master
Certified Drupal 7 Grand Master
Certified Front-end Dev
Certified Back-end Dev
Certified Developer

Co-instructor: Lisa McCray

Drupal Developer
Bixal

Principal & Senior Software Engineer -
SRA, Inc. (now General Dynamics)

Research & Teaching Assistant in
Computer-Supported Cooperative Work

Systems Analyst
Procter & Gamble

lisa.mccray@debugacademy.com

MS, Industrial Engineering/Human Factors
University of Illinois at Urbana-Champaign

BS, Electrical Engineering
Ohio State University

Co-instructor: Jack Garratt

Web Developer
- Debug Academy (2018)

History Teacher
- The Potomac School (2017)

Instructor, Adjunct
- The George Washington University (2015)

Graduate Teaching Assistant
- The George Washington University (2009-2015)

jack.garratt@debugacademy.com

-Developer on drupal.tv and arcsyria.org
-History, PhD 2016 (Germany and Africa)
-Lived in Berlin, Germany (2013-2014)

This presentation comes from our

Career-changing Drupal 8 PT Course

3 month part-time Drupal 8 Web Developer course

- Next course begins **January 27, 2019**
- Applications close as soon as class is full.
- **We build, launch, & contribute Drupal projects in class!**

...Onto the good stuff

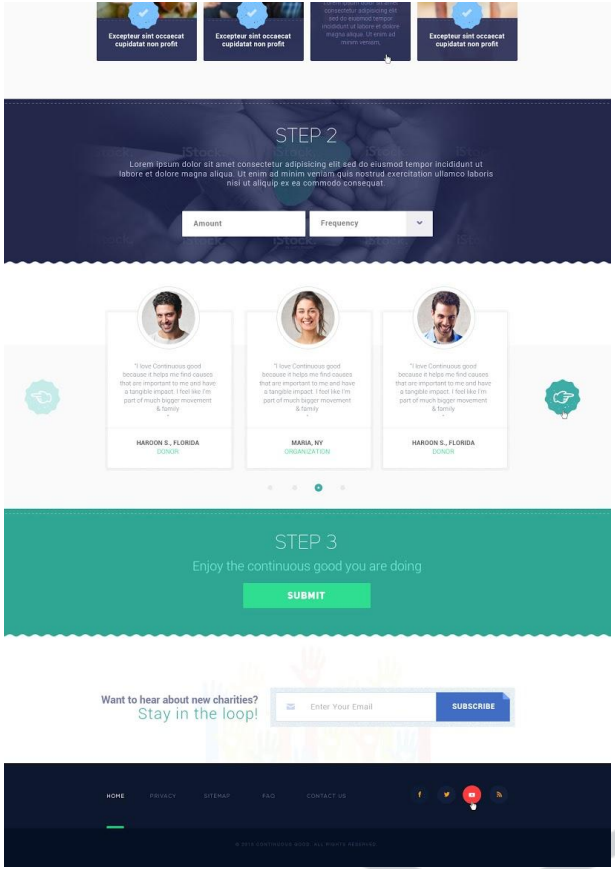
Whom is this class for?

- Those with limited or no background in web development
- Those who need a refresher session on the basics
- Those who want a hands-on approach to building website with live guidance

What tools will we need?

- A Code Editor -- Sublime Text
  - However, if you have one that you particularly enjoy, feel free to use the one that you are most comfortable with.
- An Internet Browser -- Google Chrome
  - Internet browsers can behave differently from each other when reading HTML and CSS. We recommend Google Chrome or Mozilla Firefox. Regardless, ensure that your internet browser is up-to-date
- A File Folder Downloaded from Github -- On the Desktop for Availability
  - This folder will hold all your files that we will write over today. Cherish it, refer back to it, and it will act as a reminder of where you started.

The page we'll begin building



The project files

Download our project:

https://github.com/debugacademy/debugacademy-htmlcss

Press:

- **Clone or Download**

Then:

- **Download as Zip**

And unzip the file on your Desktop

What does it mean to ‘code’ something?

- Programming languages are just like natural languages, except:
  - They possess (relatively) ***very limited*** vocabularies
  - They are explicit
  - They ‘state’ - they never ‘imply’
- Programmers are translators
- To ‘code’ or ‘program’ is to translate
  - From a human-friendly language, such as English
  - To a “programming language”

We do what Google Translate does?

- Well, *not really*. Computers, as fascinating and powerful as they are, are "dumb".
  - People can infer & read between the lines
    - This makes Google Translate 'good enough' for humans
  - Computers read only the lines
    - ***Nothing's implied***, there's *no body language*, there's *no tone*
- ***And*** programming languages have a **very limited vocabulary**.

Programmers comprehensively translate tasks using a limited vocabulary in a language that is unnatural to humans.

And that's why programming takes getting used to.

Different programming languages

Languages have limited vocabularies and capabilities.
Pick what language to use based on what goal you are attempting to accomplish.
You will often use more than one language to accomplish a single goal.
Let's look at a few programming languages:

✓ HTML

```
<input type="checkbox">
```

- Vocabulary:
  - Elements on a page
  - Containers for things on a page
- Understood by: Web Browsers
- Understands: N/A

✓ CSS

```
Input { color: red; }
```

- Vocabulary: Styling, positioning
- Understood by: Web Browsers
- Understands: Web Browser Users (Very limited)

More programming languages (JS)

- ✓ JavaScript
  - Vocabulary:
    - Data: Read, Store, Create, compare, modify, return.
    - Talk to user's web browser.
    - Does not wait for previous line to finish before starting next line.
    - Group code in various ways.
    - Many built in functions.
  - Understood by: Web Browsers
  - Understands: Web Browser Interactions

```
document.getElementById("myBtn")
.addEventListener("click", function(){ alert("Hello World!"); });
```

Don't like what you see? You can even make your own programming language!

Practice: “IRL” programming language

- Vocabulary:
  - LoadPerson(Person’s Name)
  - RotateJoint (Which Joint?, How many degrees?, How fast?)
  - SayWord(Which word?)
  - Locator(Person or Place)
  - “my_value = ” (this allows you to store values)
- Understood by: any human (it uses mind-control)
- Understands / Listens for:
  - Twitter (@DebugAcademy) Follows
  - Newsletter (DebugAcademy.com) Subscriptions

“IRL” Task: Make a user Sit Down

```
Code:
 person = LoadPerson(Twitter Follower)
 person.RotateJoint('left knee', 90, slow)
 person.RotateJoint('right knee', 90, slow)
 person.RotateJoint('hip', -90, slow)
Progress, but that didn't quite work

Refactor:
 person = LoadPerson(Twitter Follower)
 person.RotateJoint('left knee', 90, slow)
       .RotateJoint('right knee', 90, slow)
       .RotateJoint('hip', -90, slow)
It works on my machine!
```

“IRL” Task: Everyone in the room, stand up

How could we accomplish this within the confines of the language?

The language doesn’t know who “everyone” is. It does not speak to the group.

It only “understands”:

- @DebugAcademy twitter follows
- DebugAcademy.com newsletter follows

But nothing is implied in programming.

So, how do we do this within the confines of the language?

Let’s look back at the language “IRL”...

- Vocabulary:
  - **LoadPerson(Person’s Name)**
  - RotateJoint (Which Joint?, How many degrees?, How fast?)
  - **SayWord(Which word?)**
  - Locator(Person or Place)
  - “my_value = ” (this allows you to store values)
- **Understood by: any human (it uses mind-control)**
- **Understands:**
  - **Twitter (@DebugAcademy) Follows**
  - Newsletter (DebugAcademy.com) Subscriptions

Within the confines of the language:

(Manual step) Make the teacher follow DebugAcademy on twitter
Then we can write:

```
person = LoadPerson(Twitter Follower Presenter)
person.SayWord('Follow')
person.SayWord('@DebugAcademy')
person.SayWord('On')
person.SayWord('Twitter')
```

Now the class can follow @DebugAcademy on twitter, setting up the next code:

```
attendee = LoadPerson(Next Twitter Follower)
attendee.RotateJoint('left knee', -90, slow)
        .RotateJoint('right knee', -90, slow)
        .RotateJoint('hip', 90, slow)
```

But we have multiple people in the class...

Learning to code, right? .. And again.

```
attendee = LoadPerson(Next Twitter Follower
attendee.RotateJoint('left knee', -90, slow
        .RotateJoint('right knee', -90, slo
        .RotateJoint('hip', 90, slow)

attendee = LoadPerson(Next Twitter Follower
attendee.RotateJoint('left knee', -90, slow
        .RotateJoint('right knee', -90, slo
        .RotateJoint('hip', 90, slow)

attendee = LoadPerson(Next Twitter Follower
attendee.RotateJoint('left knee', -90, slow
        .RotateJoint('right knee', -90, slo
        .RotateJoint('hip', 90, slow)
```

What if the task were more complex?

We're not finished, and *this is getting complicated.*

Writing ‘better’ code

As is the case with any language, you can write, or you can write well. Here are some rules for writing code “well”:

“KISS”

- Keep it simple, stupid.

“DRY”

- Don’t repeat yourself.
- Writing the same code repeatedly? Create a function for that code.

Single Responsibility

- Ensure every function does only one thing, and does it well.

Separation of Concerns

- Write code so that it is separated into distinct components that function independently whenever possible.

Writing ‘better’ code

As is the case with any language, you can write, or you can write well. Here are some rules for writing code “well”:

“YAGNI”

- You aren’t gonna need it
- Don’t assume functions you aren’t using today will be useful in the future. Don’t write code until you need it.

Make it work, then make it good

- AKA “Avoid premature optimization”.
- Make it work, test it, then improve what needs to be improved.

Easy to understand Naming

- Choose self-explanatory names for your functions and variables.

What is HTML?

- Instructions for a web browser telling it what it needs to display
- The structure and content of most websites is output in HTML
- Nearly all web pages are displayed using HTML
- HTML is a ***relatively*** simple programming language
- HTML is not meant for styling

HTML is a Markup Language

Content is “marked up” with additional information that tells a browser what to do with it.

Marked Up Content:

```
This is the main content heading-> A Tale of Two Cities
This is a section heading-> Book the First - Recalled to Life
This is the start of a chapter-> I. The Period
This is the main body of the text-> It was the best of times, it was the worst of times...
```

Browser’s Output:

A Tale of Two Cities

Book the First - Recalled to Life

I. The Period

It was the best of times, it was the worst of times...

What is HTML?

- The structure and content of most websites is output in HTML
- HTML is a ***relatively*** simple programming language
- Nearly all web pages are displayed using HTML
- HTML is not meant for styling

HTML5 Elements

Have a start tag, an end tag, and the content in-between:

```
<tagname>content</tagname>
```

'HTML element' is everything from the start tag to the end tag:

```
<p>This is an HTML paragraph.</p>
```

HTML5 Attributes

HTML elements can have attributes

- Attributes provide additional information about an element
- Attributes are always specified in the start tag
- Attributes usually come in name/value pairs like: name="value"
  - This is a link
  - <img src="w3schools.jpg" width="104" height="142">

What Do You Need for a Web Page?

HTML for a web page contains

both the marked-up content as

well as some extra information

for the browser that never shows

up on the screen.

Minimal HTML5 Web Page:

More typical HTML5 Web Page:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>title</title>
  </head>
  <body>
    <!-- page content -->
  </body>
</html>
```

Building a page with HTML

<head>

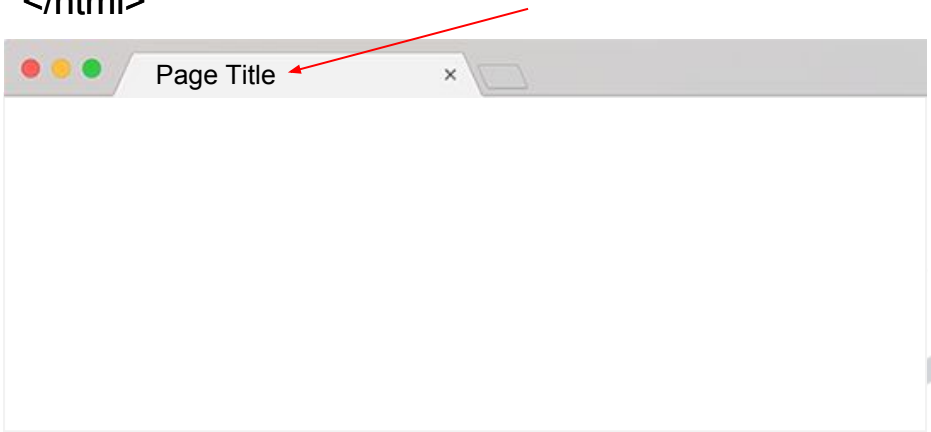
- Contains data which is not directly displayed on the page
- Includes information about the page's contents
- Placed once: between the <html> tag and the <body> tag

<title>

- Defines the title of the browser window
- Placed once: within the <head> tags

```
<html>
  <head>
    <title>Page Title</title>
  </head>

</html>
```



Set our page's <title>

- Open continuousgood/homepage.html in your browser
  - You should see a blank page
- Open SublimeText
  - In SublimeText, Open/edit:
    - **continuousgood/homepage.html**
- In-between the opening & closing **<head>** tags, set the page's title:
  - **<title>Continuous Good</title>**
- Save the file
- Refresh the webpage in your browser to see the changes

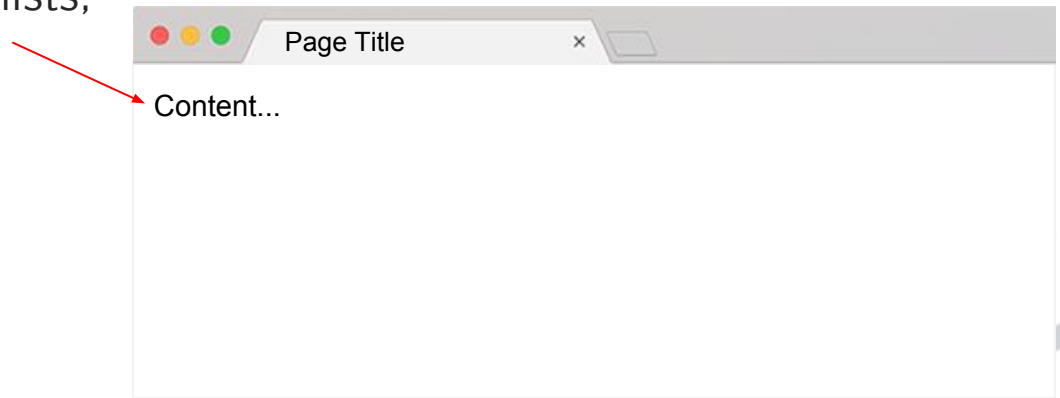
Building a page with HTML

<body>

- Defines the document's body
- Visible content is placed within the body tag
- Contains all content of an HTML document
    - Text, links, images, tables, lists, etc.
- Placed once: After </head>

```
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    Content...


  </body>
</html>
```



Building a page with HTML

<h1>, <h2>, <h3>, ...

- h1 is the primary heading on a page, h2 is the second most prominent heading, then h3, and so on..
- Used by styling and search engines to determine the most prominent text on the page.
- <h1>Site name</h1>
- <h2>Section heading</h2>
- <h3>Sub-section heading.</h3>
- <h2>Other section heading</h2>
- Do not use for highlighting text, or making it bold

```
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <h1>Primary (h1)</h1>
    <h2>Secondary (h2)</h2> Content...
    <h2>Another Secondary (h2)</h2>More...
  </body>
</html>
```

Set our page's headings

- Open SublimeText
  - In SublimeText, Open/**edit**:
    - **continuousgood/homepage.html**
- Where it says: "<!-- The header section -->"
  - Add a line: **<h1>Continuous Good</h1>**
- Where it says: "<!-- The "Welcome To Continuous Good" section -->"
  - Add a line: **<h2>Welcome to Continuous Good</h2>**
- Save the file
- Refresh the webpage in your browser to see the changes

Building a page with HTML

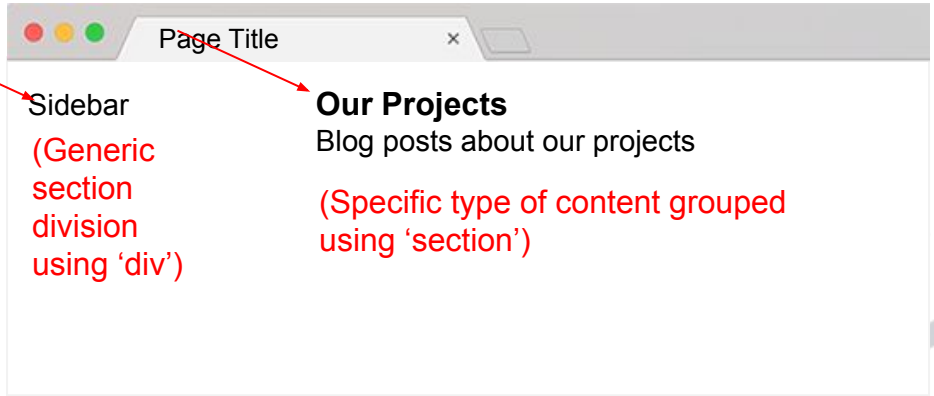
<section>

- A section in the page
- Used to group content (i.e. content relates to a single theme)

** **

- Generic container element
- Used to group block-elements
- Often used for structuring page layouts
- Elements are then arranged and formatted with CSS

```
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <div id="sidebar">Sidebar</div>
    <section id="projects"><h2>Our Projects</h2>
    Blog posts about our projects</section>
  </body>
</html>
```



Building a page with HTML

****

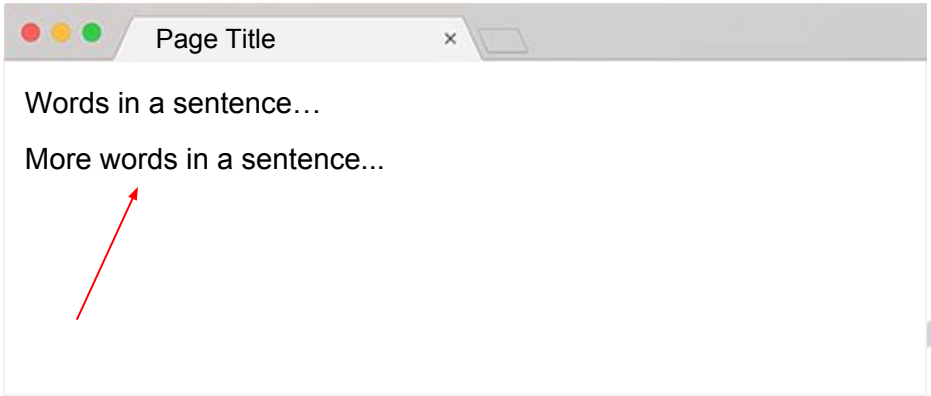
- Used for targeting and grouping inline-elements in a document.
- Provides no visual change by itself.
- Provides a way to style or act on a part of text.

<p>

- Specifies the start and end points for a paragraph

```
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <p>Words in a sentence…</p>
    <p>More <span>words</span>…</p>

  </body>
</html>
```



Set and tags

- Open SublimeText
  - In SublimeText, Open/**edit**:
    - **continuousgood/homepage.html**
- Where it says: “<h2>Welcome to Continuous Good</h2>”
  - Add a span around the words “Welcome to”
    - “<h2>****Welcome to**** Continuous Good</h2>”
- Directly below that <h2>, use paragraph tags to begin a paragraph:
  - My long, heartfelt welcoming message will be written here.

Building a page with HTML

<ul>

- Creates a new unordered list

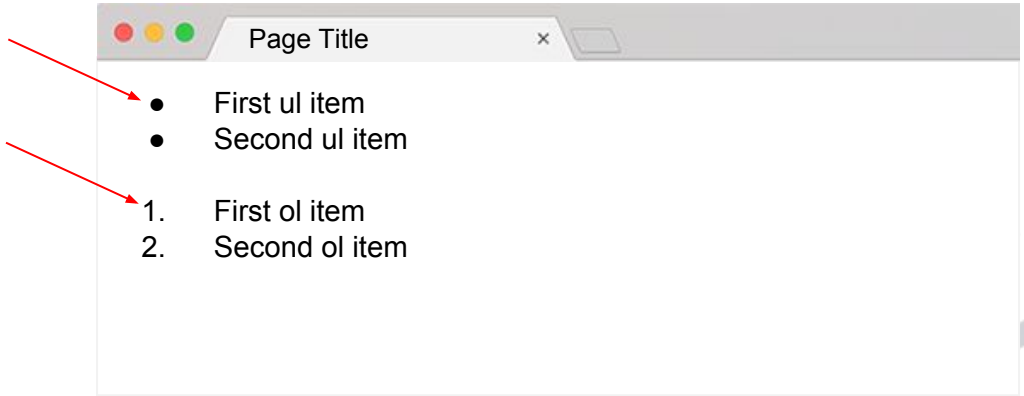
<ol>

- ‘Ordered’ or ‘Numbered’ list

<li>

- List item

```
..<body>
   <ul>
     <li>First ul item</li>
     <li>Second ul item</li>
   </ul>
   <ol>
     <li>First ol item</li>
     <li>Second ol item</li>
   </ol>
 </body>
```



Create the main menu

- In SublimeText, Open/**edit**:
  - **continuousgood/homepage.html**
- Directly below your </h1> tag, create a list of menu links:
  -

    Give Forward

    Explore

    Charities

    About Us

- Save then view your webpage in your browser

Building a page with HTML

- head
- title
- body
- h1, h2, h3, h4…
- section
- div
- table, tr, td, thead, tbody
- span, p
- ul, ol, li

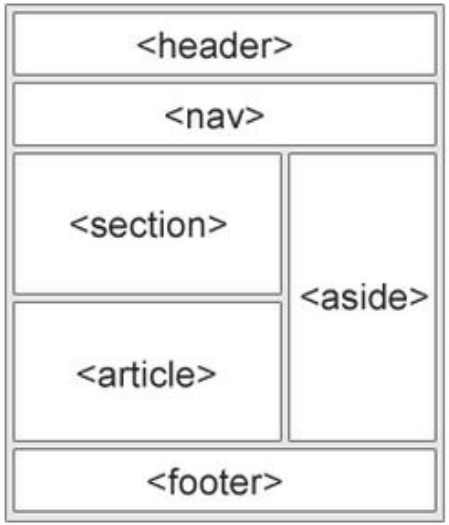
```
<html>
  <head><title>Page title</title></head>
  <body>
    <div id="wrapper">
      <div id="header">Header</div>
      <div id="nav">
        <ul>
          <li>Home</li> <li>About</li> <li>Contact</li>
        </ul>
      </div>
      <div id="content">Content</div>
      <div id="footer">Footer</div>
    </div>
  </body>
</html>
```

HTML5 Semantic Elements

Instead of , , etc, you should use the HTML5 alternatives:

Website Layout Using HTML5

HTML5 offers new semantic elements that define different parts of a web page:



header	Defines a header for a document or a section
nav	Defines a container for navigation links
section	Defines a section in a document
article	Defines an independent self-contained article
aside	Defines content aside from the content (like a sidebar)
footer	Defines a footer for a document or a section
details	Defines additional details
summary	Defines a heading for the details element

Use semantic HTML for our menu

- In SublimeText, Open/**edit**:
    - **continuousgood/homepage.html**
- Wrap your opening and closing <ul> tags in <nav> tags:
    - **<nav>**

      Give Forward (…) About Us

      </nav>
- Save then view your webpage in your browser

Same content, different browsers

And we're done with the HTML!

...What's missing?

DebugAcademy.com

CSS

What is CSS?

- CSS stands for ‘Cascading Style Sheet’
- Applies color, alignment, spacing, and sizing to web pages
- Analogy:
    - HTML: structure and furniture for a house
    - CSS: paint, finishing, and alignment

CSS Syntax

- Selectors: What is being styled
- Properties: What style should be applied
- ‘Style’ includes:
  - Color
  - Spacing
  - Position
  - Size
  - Background

What CSS looks like

```
[Selectors] {
  [style properties]
}
```

CSS Selectors

How to specify "***What*** is being styled"

HTML element

```
div {
}
```

HTML element **ID attribute**

```
#name-of-id {
}
```

HTML element **class attribute**

```
.name-of-class {
}
```

a combination of selectors

```
div.name-of-class {
}

div#name-of-id {
}
```

Writing CSS

- In SublimeText, Open/**edit**:
    - **continuousgood/style.css**
- Target the selector,
    - div {

      background: #dc4523;

      font-size: 20px;

      letter-spacing: 5px;

      }
- Save then view your webpage in your browser
- Color Picker -- htmlcolorcodes.com/color-picker

Reading combined selectors

- Selector “sentences” are read right to left.
- Spaces are read as: "Which is a descendent of"
  - Ex: *header div*: “Div which is a descendent of header”
- Word-combinations are read left to right
  - . = “which has a class of”
  - # = “which has an ID of”
  - Ex: *nav.main*: “nav which has a class of main”
  - Ex: *nav#main*: “nav which has an ID of main”

How would we read: “ *header nav#main li* ” ?

Selector Priority

- **The most specific selector wins**
- Use ‘ICE’ formula to determine most specific selector:
  - I - Number of IDs
  - C - Number of Classes
  - E - Number of Elements
- #unique-id = 1 ID, 0 Classes, 0 Elements = 100
- div.fontsize div.generic-class .another-class = ?

What makes a good CSS class?

- Rule 1: Designed for reusability
    - ‘first-div-on-homepage’ won’t make sense in other contexts
- Rule 2: Not overly restricted by its name
    - Names like ‘twelve-pixel-font’ prevent updates
- Rule 3: Selector does not exceed its intended scope
    - Be sure you’re not targeting extra elements

Most common CSS properties

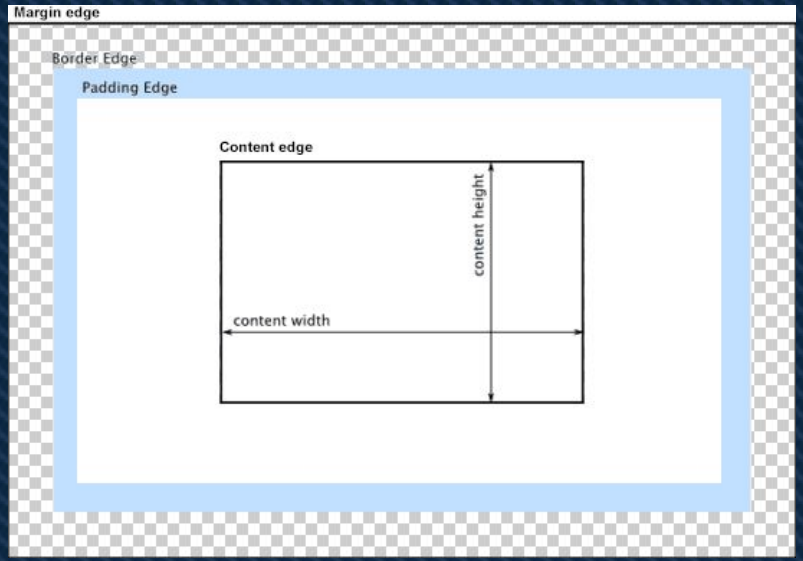
CSS properties you'll be expected to know and understand

- display: [block | inline-block | inline | table-cell | none | initial];
- position: [absolute | fixed | relative | initial | inherit];
- float: [none | left | right | initial | inherit];
- padding: [length in px, pt, em];
- margin: [auto | length in px, pt, em];
- list-style: [disc | circle | none] [inside | outside] [none | url|initial | inherit];
- width: [auto | px | em | %];
- height: [auto | px | em | %];
- border: [px] [solid | dotted] [color];

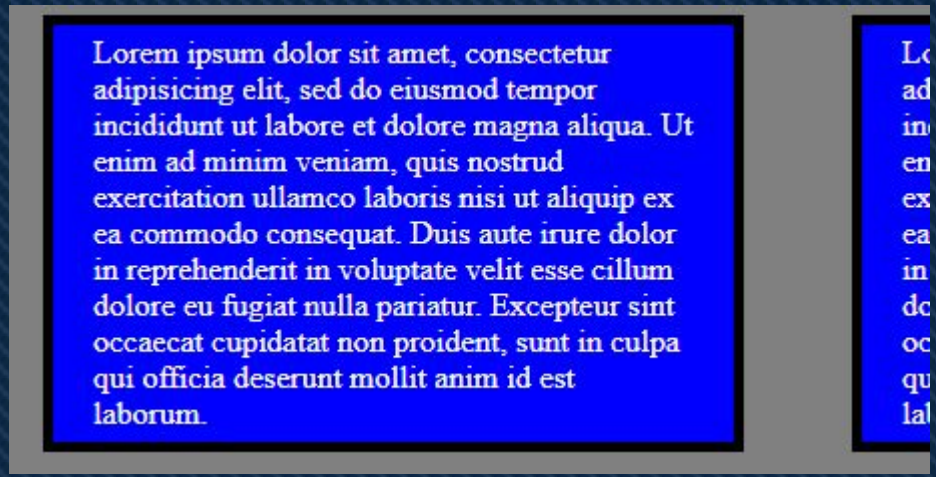
CSS Box Model - Styling HTML Objects

Understanding Content, Padding, Border, and Margin

The “Box”’ Model



HTML “<p>” Example



Writing CSS - The Box Model

- In SublimeText, Open/**edit**:
  - **continuousgood/style.css**
- Target the selector,
  - div {

    padding-top: 15px;

    border: 5px black solid;

    margin-bottom: 5px;

    }
- Save then view your webpage in your browser
- Target only the Welcome by adding a selector

Display

- Every element can be thought of as a rectangular box
    - Display determines how that box behaves
- **inline** (e.g. span, em, b)
    - Element sits ‘inline’, does not break flow, no height or width
- **block** (e.g. div, section, p)
    - Does NOT sit ‘inline’, breaks flow, takes up as much horizontal space as it can (by default)

Display Ctd.

- **inline-block**
    - Similar to inline, but can take a height and width
- **none**
    - Removes the element from the page
- **table, table-cell, table-row** etc.
    - Use these to mimic behavior of table elements (less common)

Position

- Used to adjust an element's location
- **static**
  - default value, as if it is *not positioned*
- **relative**
  - Setting of {top: 5px} on a relatively positioned element will move it down from the top by 5px

Position Ctd.

- **absolute**
  - Element removed from flow. To other elements, as if it isn't there. Relative to nearest positioned ancestor.
- **fixed**
  - Similar to absolute, but relative to document, not ancestors. Also unaffected by scrolling.

Allow menu links to have height set

- In SublimeText, Open/**edit**:
  - **continuousgood/style.css**
- Target the links (a) which are descendants of the nav element
- Set their **display** as inline-block
  - nav li a {

    padding: 12px 10px;

    background-color: #3eada7;

    display: inline-block;

    }
- Save then view your webpage in your browser

CSS Workflow: Eliminate differences

1. Compare the design image to your webpage
2. Spot a single difference (no matter how small!)
3. Research how to address it
4. Write CSS to eliminate that difference

Repeat steps 1-4 until your webpage satisfactorily matches the design

Responsive Design

- Adapts to changes in the size of the device it is being viewed on.

- The devices your site will be viewed on are always changing:
  - iPhone 6, iPhone 6+, Note4, Note5, iPad?, Desktops, etc...
  - Sizes of devices vary, many have landscape and portrait displays
  - New devices every year with different sized screens

- Alternatives to responsive: separate mobile site
  - 2 or more separate sites to maintain.
  - What about tablets? or the size of the mobile devices changing?

- More maintainable to create a single, responsive, website design.

“Isn’t contradictory CSS is required?”

- The styling (css) for a mobile website and a desktop website is often very different

- And sites should ‘gracefully degrade’ if javaScript is disabled

@media queries to the rescue

Media Query

Limits when a section of your CSS applies, based on a user-defined expression, or ‘query’.

Query parameter options:

- Minimum display width
- Maximum display width
- Display type (e.g. all, screen, print)
- The orientation (landscape or portrait)

```
#wrapper {
  width: 850px;
}

@media (max-width: 850px) {
  #wrapper {
    width: 100%;
  }
}
```

Media Query

Example:

```
#wrapper {
  width: 850px;
  background-color: blue;
}
@media (max-width: 850px) {
  #wrapper {
    width: 100%;
  }
}
@media print and (orientation: landscape) and (min-width: 1000px) {
  #wrapper {
    background-color: white;
  }
}
```

Query parameter options:

- Minimum display width
- Maximum display width
- Display type (e.g. all, screen, print)
- The orientation (landscape or portrait)

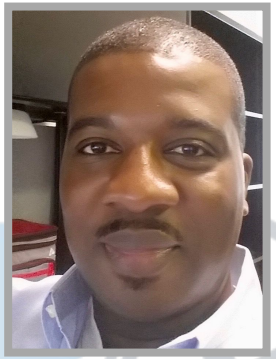
DebugAcademy.com career changers

"I'm a Senior Consultant / Drupal Software Developer at Booz Allen Hamilton, building Drupal sites for various government agencies, such as the Department of Labor, FEMA, and the Department of Homeland Security."

After taking the course, I was able to combine my Drupal skills and Conflict Analysis and Resolution major into a career I love. I now work with nonprofit organizations as a Software Engineer for Beaconfire Red!

Debug Academy not only taught me how to code, but gave me the confidence to keep learning. It was the support of the instructor and my classmates that kept me going. I completed the course and not only did I learn Drupal, HTML, CSS and PHP but I finally found a career I could get excited about!

This has been the best career decision for me. I've learned so much with regards to how to be a valuable member of a Drupal development team. With hands-on training with Git, Composer, Drush, SASS, Drupal, PHP and other technologies. With just 8 weeks of training from the program, I was able to obtain employment as a PHP/Drupal Developer. I HIGHLY recommend this program.



Resources

- **References / Tutorials:**
  - HTML
    - MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/HTML
    - W3Schools: https://www.w3schools.com/html/
  - CSS
    - MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/CSS
    - W3Schools: https://www.w3schools.com/css

This presentation comes from our

Career-changing Drupal 8 PT Course

- Next *part-time* 3 month Drupal 8 Web Developer course:
  - Summer semester begins **June 2019**
  - Applications close as soon as class is full.
  - **We build, launch, *& contribute* Drupal projects in class!**
- Also: **Tailored on-site team training available**

See DebugAcademy.com / follow us at @DebugAcademy

Or contact me directly: ashraf.abed@debugacademy.com / @ashabed