Making Websites for Beginners

Clarissa Littler

August 2, 2016

What we'll cover

- The basic technology that goes into a webpage

What we'll cover

- The basic technology that goes into a webpage
- Simple examples of how to use HTML and CSS

What we'll cover

- The basic technology that goes into a webpage
- Simple examples of how to use HTML and CSS and maybe JavaScript

What we'll cover

- The basic technology that goes into a webpage
- Simple examples of how to use HTML and CSS and maybe JavaScript
- Resources to continue your learning

What we won't cover

- How to build the back-end of a site

What we won't cover

- How to build the back-end of a site
- How to program in JavaScript in general

What we won't cover

- How to build the back-end of a site
- How to program in JavaScript in general
    - Though there are free supplements for that

What we won't cover

- How to build the back-end of a site
- How to program in JavaScript in general
    - Though there are free supplements for that
- A majority of CSS and HTML

Client and server

Two pieces that talk to each other to make a site

Server

- Sends data to the browser
- Saves information for long term use
- Receives requests from the client

Client

- Receives data from the server
- Renders server data into a usable page
- Handles the user interface

The three pieces of a web page

- HTML

The three pieces of a web page

- HTML
- CSS

The three pieces of a web page

- HTML
- CSS
- JavaScript

HTML

What does HTML do?

HTML describes the content of the page, but not how it looks

CSS

What does CSS do?

CSS describes how a page looks, but not its content

JavaScript

What does JavaScript do?

The dynamics and the user interface of the page

What is HTML

HyperText Markup Language

- HyperText

What is HTML

HyperText Markup Language

- HyperText
- Markup

Tags and Elements

```
<body>
  <h1>This is a header</h1>
  <p>This is a paragraph of text,
      where some of the text is <b>bold</b>, and
      after this paragraph, there will be a numbered list
  </p>

  <ol>
    <li>lists are made of "list items"</li>
    <li>like these</li>
  </ol>
</body>
```

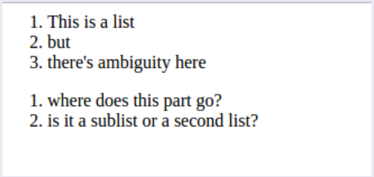
Whence closing tags

Without closing tags

```
<body>
  <ol>
    <li>This is a list
    <li>but
    <li> there's ambiguity here
  <ol>
    <li> where does this part go?
    <li> is it a sublist or a second list?
```

Whence closing tags

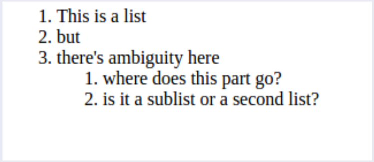
This?



1. This is a list
2. but
3. there's ambiguity here

1. where does this part go?
2. is it a sublist or a second list?

Or this?



1. This is a list
2. but
3. there's ambiguity here
    1. where does this part go?
    2. is it a sublist or a second list?

Whence closing tags

With closing tags

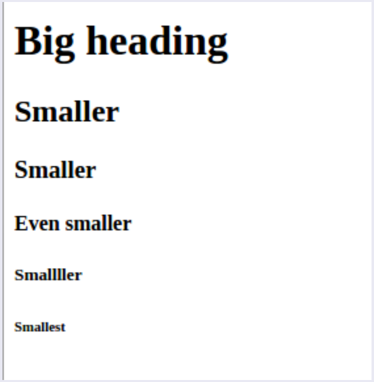
```
<body>
  <ol>
    <li>This is a list</li>
    <li>but</li>
    <li> there's ambiguity here</li>
  </ol>
  <ol>
    <li> where does this part go?</li>
    <li> is it a sublist or a second list?</li>
  </ol>
</body>
```

The basic template

```
<!doctype html>
<html>
  <head>
    ...
  </head>
  <body>
    ...
  </body>
</html>
```

Headers

```
<!doctype html>
<html>
  <body>
    <h1>Big heading</h1>
    <h2>Smaller</h2>
    <h3>Smaller</h3>
    <h4>Even smaller</h4>
    <h5>Smallller</h5>
    <h6>Smallest</h6>
  </body>
</html>
```

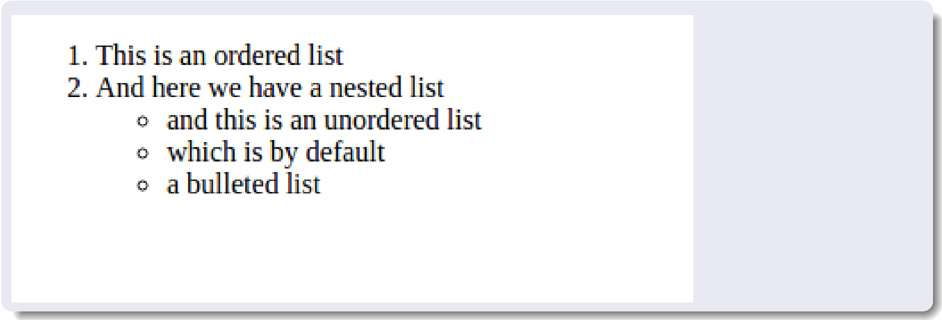


Big heading
Smaller
Smaller
Even smaller
Smallller
Smallest

Lists

```html
<!doctype html>
<html>
  <body>
    <ol>
      <li>This is an ordered list</li>
      <li>And here we have a nested list
        <ul>
          <li>and this is an unordered list</li>
          <li>which is by default</li>
          <li>a bulleted list</li>
        </ul>
      </li>
    </ol>
  </body>
</html>
```

Lists



1. This is an ordered list
2. And here we have a nested list
    - and this is an unordered list
    - which is by default
    - a bulleted list

Nesting

What goes in what

- paragraphs can't nest in paragraphs

Nesting

What goes in what

- paragraphs can't nest in paragraphs
- headers can't nest in headers

Nesting

What goes in what

- paragraphs can't nest in paragraphs
- headers can't nest in headers
- lists *can* nest in lists

Exercise 1

Make your own page following the template below: use at least two of the following tags `<ol>`, `<ul>`, `<h1>`, `<p>`, `<b>`, and `<i>`

```
<!doctype html>
<html>
  <body>
    your code here
  </body>
</html>
```

Anchors and Attributes

```
<a href="https://multcolib.org">This is a link</a>
```

Exercise 2

Create your own page that uses at least two links and test them to ensure they work

Inline and Block Elements

Inline

Elements that don't take up space beyond their text

Block

Elements that take up room beyond their text

Div and span

Semantic markup

Div and span are used to group related elements together

Cascading Style Sheets

What is CSS?

Cascading style sheets control the appearance of elements

CSS Entries

```
selector {
    property: value;
    property: value;
    property: value;
}
```

Adding CSS to a page

Style tags

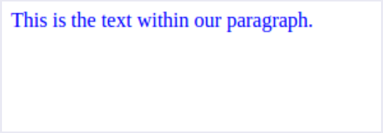
```
<!doctyle html>
<html>
  <head>
    <style>
      ...
    </style>
  </head>
  <body>
    ...
  </body>
</html>
```

Selecting elements by ID

```
<!doctype html>

<html>
  <head>
    <style>
      #para {
        color: blue;
      }
    </style>
  </head>
  <body>
    <p id="para">This is the text within our paragraph.</p>
  </body>
</html>
```

Selecting elements by ID



This is the text within our paragraph.

Selecting elements by class

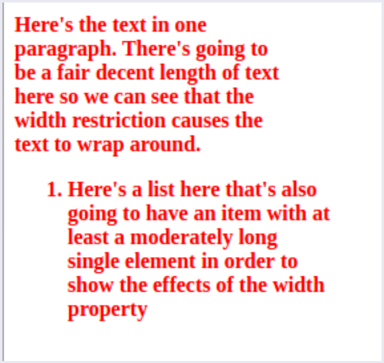
```
.ourClass {
    color: red;
    width: 200px;
    font-weight: bold;
}
```

Selecting elements by class

```
<p class="ourClass">Here's the
text in one paragraph.
There's going to be a fair
decent length of text here so we
can see that the width
restriction causes the text to wrap around.</p>

<ol class="ourClass">
  <li>Here's a list here that's
  also going to have an item
  with at least a moderately long
  single element
  in order to show the
  effects of the width property</li>
</ol>
```

Selecting elements by class



Here's the text in one paragraph. There's going to be a fair decent length of text here so we can see that the width restriction causes the text to wrap around.

1. Here's a list here that's also going to have an item with at least a moderately long single element in order to show the effects of the width property

Exercise 3

Open the file `exer3.html` and then add in CSS declarations to make both paragraphs have `width: 200px` and the first paragraph have a color of `blue`

```
<!doctype html>
<html>
  <head>
  </head>
  <body>
    <p class="theClass" id="firstPara">
    This is a paragraph that has some text in it
    and, y'know, stuff and things</p>
    <p class="theClass" id="sndPara">
      This is the second paragraph by gum</p>
  </body>
</html>
```

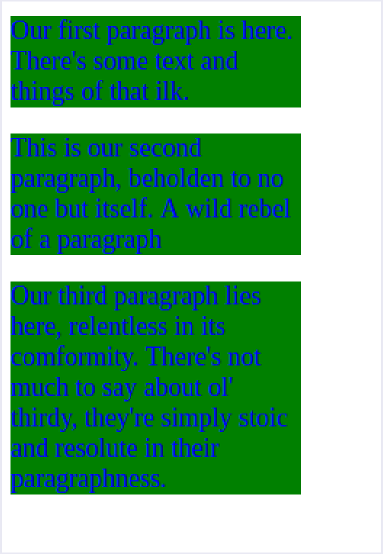
Selecting elements by type

```
p {
    font-size: large;
    background-color: green;
    color: blue;
    width: 200px;
}
```

Selecting elements by type

```
<p>Our first paragraph is here.
  There's some text and things of that ilk.</p>
<p>This is our second paragraph,
  beholden to no one but itself.
  A wild rebel of a paragraph</p>
<p>Our third paragraph lies here,
  relentless in its comformity.
  There's not much to say about ol' thirdy,
  they're simply stoic and
  resolute in their paragraphness.</p>
```

Selecting elements by type


Our first paragraph is here. There's some text and things of that ilk.

This is our second paragraph, beholden to no one but itself. A wild rebel of a paragraph

Our third paragraph lies here, relentless in its comformity. There's not much to say about ol' thirdy, they're simply stoic and resolute in their paragraphness.

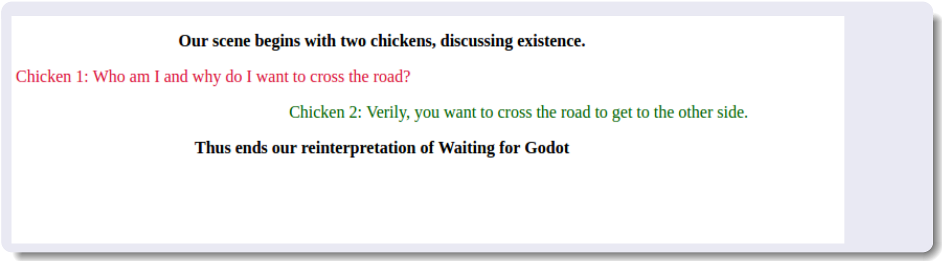
Another example with divs

```css
.character1 {
  color: crimson;
  text-align: left;
}
.character2 {
  color: darkgreen;
  text-align: right;
}
.description {
  font-weight: bold;
  text-align: center;
}
.script {
  width: 700px;
}
```

Another example with divs

```
<div class="script">
  <p class="description">
    Our scene begins with two chickens,
    discussing existence.
  </p>
  <p class="character1">
    Chicken 1: Who am I and why do I want to cross the road?
  </p>
  <p class="character2">
    Chicken 2: Verily, you want to cross the road
    to get to the other side.
  </p>
  <p class="description">
    Thus ends our reinterpretation of Waiting for Godot
  </p>
</div>
```

Another example with divs



Our scene begins with two chickens, discussing existence.

Chicken 1: Who am I and why do I want to cross the road?

Chicken 2: Verily, you want to cross the road to get to the other side.

Thus ends our reinterpretation of Waiting for Godot

Specificity

combining type and class

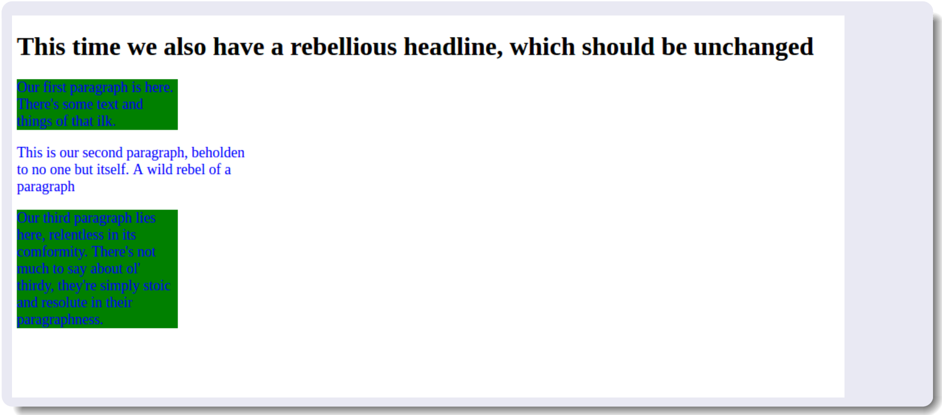
```
p {
    font-size: large;
    background-color: green;
    color: blue;
    width: 200px;
}
p.rebel {
    width: 300px;
    background-color: white;
}
```

Specificity

```
<h1 class="rebel">This time we also have a rebellious header,
which should be unchanged</h1>

<p>Our first paragraph is here.
  There's some text and things of that ilk.</p>
<p class="rebel">This is our second paragraph,
  beholden to no one but itself.
  A wild rebel of a paragraph</p>
<p>Our third paragraph lies here,
  relentless in its comformity.
  There's not much to say about ol' thirdy,
  they're simply stoic and resolute
  in their paragraphness.</p>
</div>
```

Specificity



This time we also have a rebellious headline, which should be unchanged

Our first paragraph is here. There's some text and things of that ilk.

This is our second paragraph, beholden to no one but itself. A wild rebel of a paragraph

Our third paragraph lies here, relentless in its comformity. There's not much to say about ol' thirdy, they're simply stoic and resolute in their paragraphness.

Specificity

choosing children of an element

```
#divvy p{
  width: 200px;
  font-weight: bold;
}
```

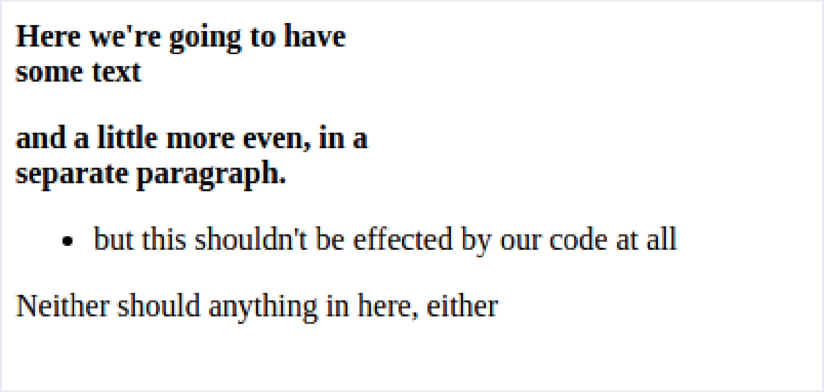
Specificity

choosing children of an element

```
<div id="divvy">
  <p> Here we're going to have some text </p>
  <p> and a little more even, in a separate paragraph. </p>

  <ul>
    <li>but this shouldn't be effected by our code at all</li>
  </ul>
</div>
<p>Neither should anything in here, either</p>
```

Specificity



Here we're going to have some text

and a little more even, in a separate paragraph.

- but this shouldn't be effected by our code at all

Neither should anything in here, either

Exercise 4

Using the following skeleton, found in exer4.html, add CSS declarations so that the first paragraph has *blue* text, the second paragraph has *red* text, and the third paragraph has *green* text.

```
<body>
  <p>our first paragraph</p>
  <div>
    <p>our second paragraph</p>
    <div>
      <p>our third paragraph </p>
    </div>
</body>
```

What is JavaScript?

JavaScript is a programming language that runs in the browser and provides the dynamics, the interaction in any web site

Programming is speaking a language

- All language is communication

Programming is speaking a language

- All language is communication
- Programming languages are special languages

Programming is speaking a language

- All language is communication
- Programming languages are special languages
- Computers need precision

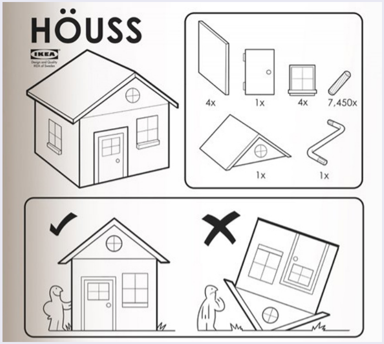
Programming is speaking a language

- All language is communication
- Programming languages are special languages
- Computers need precision they're not as smart as us

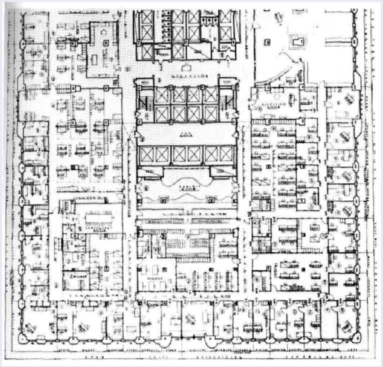
Why programming is hard

The precision of instructions computers need is unnatural for the human mind

Why programming is hard



Why programming is hard



Why programming is learnable

- Precise thinking may be unnatural

Why programming is learnable

- Precise thinking may be unnatural
- But it's not impossible

Why programming is learnable

- Precise thinking may be unnatural
- But it's not impossible
- It takes time

Why programming is learnable

- Precise thinking may be unnatural
- But it's not impossible
- It takes time and practice

Why programming is learnable

- Precise thinking may be unnatural
- But it's not impossible
- It takes time and practice
- Like learning any language

The JavaScript console

- Every browser can run JavaScript

The JavaScript console

- Every browser can run JavaScript
- The `console` allows you to test code

The JavaScript console

Let's try it!

Syntax

- Syntax is the grammar of a language

Syntax

- Syntax is the grammar of a language
- Even stricter rules than human languages

Syntax

- Syntax is the grammar of a language
- Even stricter rules than human languages
- “Dog not can to ridebike nor can to cook”

Syntax

- Syntax is the grammar of a language
- Even stricter rules than human languages
- “Dog not can to ridebike nor can to cook”
- Computers can’t guess

Evaluation of code

- Syntax doesn't do anything

Evaluation of code

- Syntax doesn't do anything
- Saying "I have a trillion dollars" doesn't make it so

Evaluation of code

- Syntax doesn't do anything
- Saying "I have a trillion dollars" doesn't make it so
- An *interpreter* runs (or *evaluates*) code

Arithmetic

Numbers

- 1
- 0.5
- -20
- ...

Operations

- +
- -
- *
- ...

Sequences

- Need to do more than a single step of code at a time

Sequences

- Need to do more than a single step of code at a time
- List the steps line by line

Sequences

- Need to do more than a single step of code at a time
- List the steps line by line separate by semicolons

Variables

I have a friend, let's call her "Cassandra"...

Variables function both as storage containers and pronouns

Creating Variables

```
var nameOfVariable = initialValueInIt;
var numberOfToes = 10;
```

Assigning variables

```
var musicalsThatShouldExist = "The Walking Dead on Ice";
musicalsThatShouldExist = "Werner Herzog Sings The Blues";
```

Mini-exercise

Test yourself

Go to your console and try to

- create a variable
- change a variable

Objects

- Phone books

Objects

- Phone books
- Contact lists

Objects

- Phone books
- Contact lists
- Mall directories

Objects

- Phone books
- Contact lists
- Mall directories
- Dictionaries

Making Objects

```
var obj = {prop1 : 0, prop2 : 1};
var otherObject = {};
```

Objects

Type the following in your console

```
var obj = {prop1 : 0, prop2 : 1, prop3 : "thing"};
obj.prop1;
obj.prop2;
obj.prop3;
```

Objects

Type the following in your console

```
var obj = {};
obj.numberOfChickens = 2;
obj.numberOfChickens;
```

Functions

Functions in math

$$f(x) = x + 10$$

Functions

Functions in JavaScript

```
function f(x) {
    return x + 10;
}
```

Using functions

First example of a function, a function that writes data to the console

```
console.log
```

Example

Navigate to the file `consoleExample.html` and then check the console to see what happened

Example

```
<!doctype html>
<html>
  <head>
    <script>
      console.log("we're printing one message");
      console.log("and another message!");
    </script>
  </head>
  <body>
    Check your console!
  </body>
</html>
```

Multi-argument functions

```
function moreFun (anArgument,anotherArgument) {
    console.log(anArgument + anotherArgument);
}

moreFun(10, 20);
```

Functions with no arguments

```
function noArgs () {
    return 10;
}
```

What is the Document Object Model?

The DOM

The document object model (DOM) is the representation of the web page *as JavaScript objects*

Putting the document in DOM

`document` is the object that holds most of the important methods

When to load code

```
window.onload = function () {
    ...
};
```

Creating elements in code

Creating elements in code

- `document.createElement`

Creating elements in code

- `document.createElement`
- `document.createTextNode`

Creating elements in code

- `document.createElement`
- `document.createTextNode`
- `document.body`

Creating elements in code

- `document.createElement`
- `document.createTextNode`
- `document.body`
- `.appendChild`

Creating elements

```
<!doctype html>
<html>
  <head>
    <script>
      window.onload = function () {
         var newHeader = document.createElement("h1");
         var textNode = document
           .createTextNode("This is a header!");
         newHeader.appendChild(textNode);
         document.body.appendChild(newHeader);
      };
    </script>
  </head>
  <body>
  </body>
</html>
```

Exercise 4

Exercise

use the `document.createElement` function to make a single

```
<!doctype html>
<html>
  <head>
    <script>
    </script>
  </head>
  <body>
  </body>
</html>
```

Finding elements

Finding elements

- `document.getElementById`

Finding elements

- `document.getElementById`
- `.firstChild`

Finding elements

- `document.getElementById`
- `.firstChild`
- `.nodeValue`

getElementById

```html
<body>
  <ol id="list1">
    <li>This is a list</li>
  </ol>
  <ol id="list2">
    <li>This is our second list</li>
  </ol>
</body>
```

getElementById

```
window.onload = function () {
    var newItem =
      document.createElement("li");
    var newText =
        document
        .createTextNode("item in the second list");
    newItem.appendChild(newText);
    var secondList = document.getElementById("list2");
    secondList.appendChild(newItem);
};
```

Changing CSS properties

```
<!doctype html>
<html>
  <head>
    <script>
      window.onload = function () {
        var h = document.getElementById("header");
        h.style.color = "red";
      }
    </script>
  </head>
  <body>
    <h1 id="header">This is a header!</h1>
  </body>
</html>
```

Exercise 5

Exercise

use `document.getElementById` and the `.style` property to change the text color of the paragraph to green

```
<!doctype html>
<html>
  <head>
    <script>
    </script>
  </head>
  <body>
    <p id="para">Here's our text.</p>
  </body>
</html>
```

What we've learned

- What a webpage is

What we've learned

- What a webpage is
    - HTML

What we've learned

- What a webpage is
    - HTML
    - CSS

What we've learned

- What a webpage is
    - HTML
    - CSS
    - JavaScript

What we've learned

- HTML

What we've learned

- HTML
    - Elements

What we've learned

- HTML
    - Elements
    - Tags

What we've learned

- HTML
    - Elements
    - Tags
    - Semantic markup

What we've learned

- HTML
  - Elements
  - Tags
  - Semantic markup
  - Content, not appearance

What we've learned

- CSS

What we've learned

- CSS
    - Style, not substance

What we've learned

- CSS
    - Style, not substance
    - Selectors

What we've learned

- CSS
  - Style, not substance
  - Selectors
  - Classes

What we've learned

- JavaScript

What we've learned

- JavaScript
    - A general purpose programming language

What we've learned

- JavaScript
  - A general purpose programming language
  - Can be run by every browser

What we've learned

- JavaScript
  - A general purpose programming language
  - Can be run by every browser
  - Connects to HTML via Document Object Model

What to learn next

- More HTML tags

What to learn next

- More HTML tags
- So much more CSS

What to learn next

- More HTML tags
- So much more CSS
- Frameworks for styling

What to learn next

- More HTML tags
- So much more CSS
- Frameworks for styling
    - Bootstrap is a very popular one

What to learn next

- More HTML tags
- So much more CSS
- Frameworks for styling
    - Bootstrap is a very popular one
- JavaScript programming

Thanks for attending!

Thanks for being in this class