Introduction to HTML, CSS, and JS

UWB Hacks the Internet 2019

Before we begin

- This is a high-level introduction to HTML, CSS, and JS.
- This covers very basic topics.
- No previous experience needed!
- If you have questions, raise your hand to let us know.

All of the files and these slides are available at:

https://github.com/UWB-ACM/Intro-HTML-CSS-JS

Quick Glossary

HTML - Hyper Text Markup Language. Provides a structure for the content that makes up all web pages.

CSS - Cascading Style Sheets. Provides a way to format HTML using sets of rules.

JS - JavaScript. A scripting language that can run in your browser (and sometimes in the server too). **Java != JavaScript**

HTML 101

HTML 101

HTML makes up the structure of nearly everything you see on the web. It assigns tags to blocks of text, which helps your web browser understand it.

```
<html>
 <body>
   <h1>Hello World!</h1>
   <p>This is some
       <code>HTML</code>
   </p>
 </body>
</html>
```

That HTML, rendered in Firefox.

All example code can be found under:
github.com/UWB-ACM/Intro-HTML-CSS-JS

HTML 101 - Syntax

HTML - Hyper Text Markup Language

Markup Language - Is a syntax used for formatting text, most commonly used by HTML and XML.

Markup Language is made up of **tags**:

HTML 101 - Syntax

Rules about Tags:

- Tags need to be closed with a matching pair.
  - Some tags like **br** can't contain text, so they use a different syntax.
- Tags can contain other tags.
- Comments start with **<!--** and end with **-->**.
- Some characters, like **<** are reserved, but you can escape them.
  - Just search for "HTML Escape Characters"
- HTML doesn't care about whitespace.
  - Typically, anything more than a single space is ignored.

```
<b>Good!</b>
<br />

<b>Bad :(</i>
<p><b>Hi!</b></p>

<!-- 3 < 4 -->
<p>3 &lt; 4</p>
```

HTML 101 - Tags

HTML has nearly 100 tags, each with their own practices of when and how they are used. Luckily for us, you only really need to know about a dozen.

https://www.w3schools.com/tags/

`p`	Paragraph
`i`	Italic
`b`	Bold
`br`	Line break
`code`	Code block
`h1 h2 h3`	Headers

```
<p>
   This is a paragraph of
   <i>text</i>.
   It contains <b>words</b>.
   <br />
   This is on a new line!
</p>
```

HTML 101 - Tags

```
<p>
   This is a paragraph of
   <i>text</i>.
   It contains <b>words</b>.
   <br />
   This is on a new line!
</p>
```

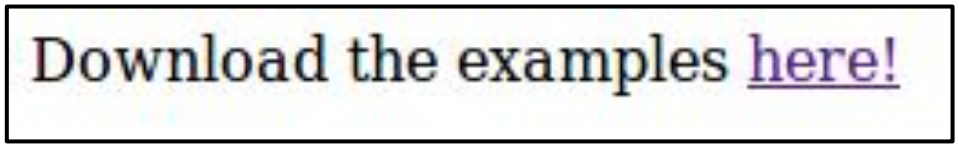
This is a paragraph of *text*. It contains **words**.
This is on a new line!

HTML 101 - Tags

The **a** tag lets you make links.

```
Download the examples
<a href="https://github.com/UWB-ACM/Intro-HTML-CSS-JS">here!</a>
```

Opening tags can define attributes. In this case, we are setting the **href** attribute to the URL for the link. This tells the browser where the tag links to.



The **href** attribute can provide a relative path to content or an absolute path. The example above has an absolute path.

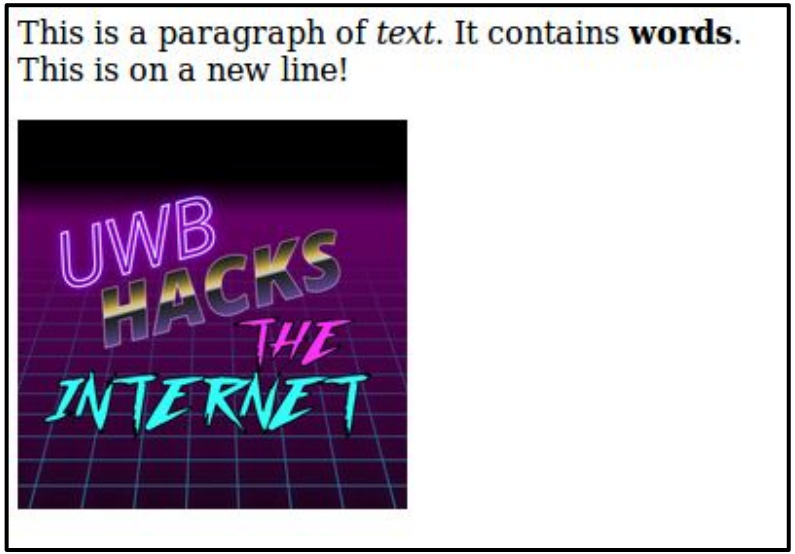
HTML 101 - Tags

The **img** tag lets you embed images.

```
<img src="images/logo.png" alt="UWB Hacks Logo" />
```

We use the **src** attribute to link to the image. Because these tags don't contain text, we don't need a matching tag to close it.

The **src** attribute can provide a relative path to content or an absolute path. The example above has a relative path.



HTML 101 - Tags

How do I make:

- Buttons? **`button`**
- Tables? **`table tr td`**
- Login forms? **`form label input`**
- Lists? **`ul ol li`**

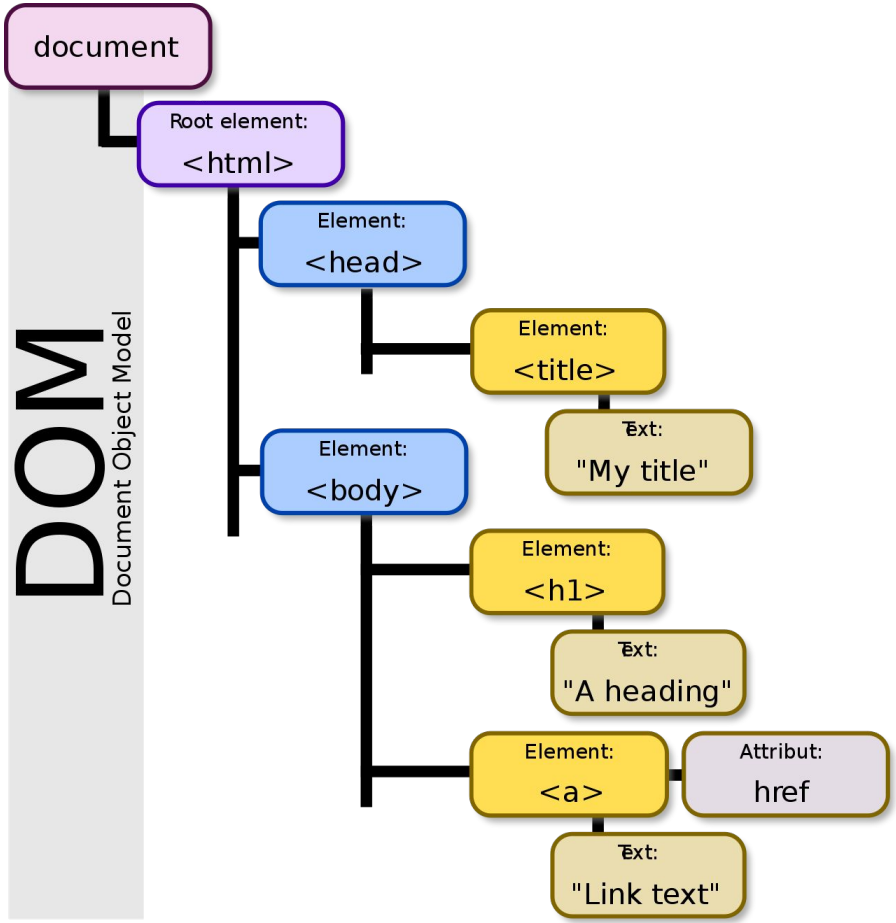
HTML 101 - DOM

Remember how nodes can contain other nodes?

HTML documents are a large tree, where each node is a tag that contains information or other nodes.

Nodes can be nested or sequential.

- Two **<p>** elements are sequential
- A **<b>** tag can be nested inside a **<p>** element to bold text.



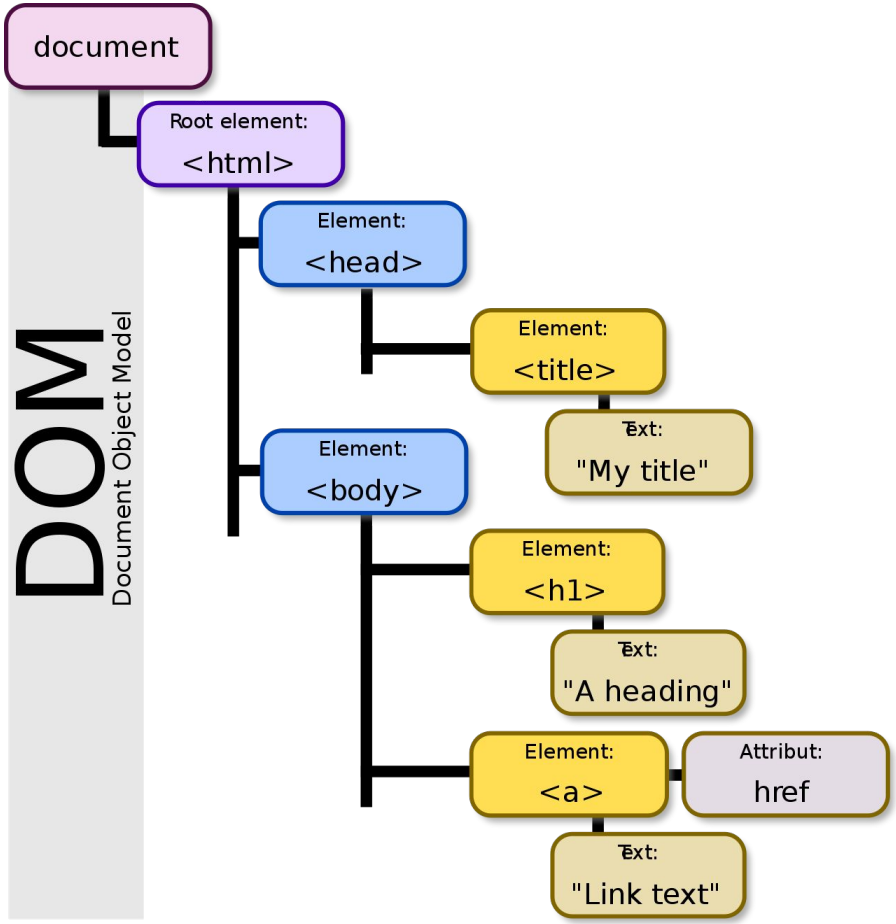
https://en.wikipedia.org/wiki/Document_Object_Model

HTML 101 - DOM

We can assign the **`class`** and **`id`** attribute to DOM nodes so that we can uniquely identify one or many of the same type later on.

The **`class`** attribute is used for defining multiple elements that are in the same group.

The **`id`** attribute is used for uniquely identifying a single element.



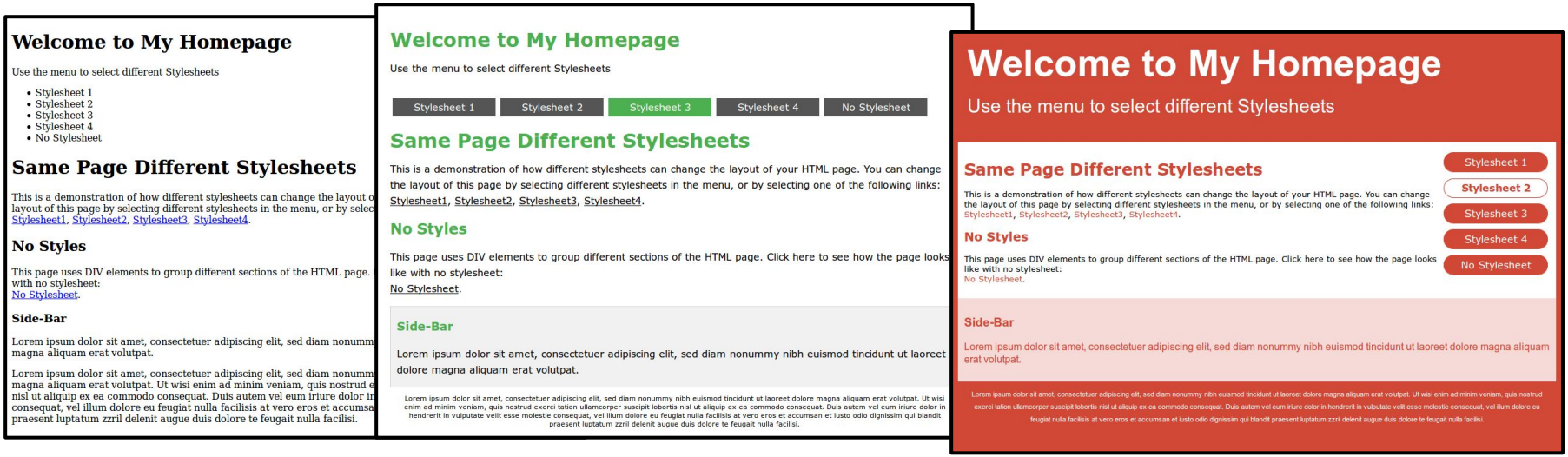
https://en.wikipedia.org/wiki/Document_Object_Model

CSS 101

CSS 101 - Cascading Style Sheets

Since **HTML** is how we structure content in a page, **CSS** is how we make it look nice.

Each of these are made with the same HTML, but different CSS!



https://www.w3schools.com/css/css_intro.asp

CSS 101 - How does it work?

CSS uses **rules** that are applied to elements in the HTML. These rules are made up of the **selector**, the pattern used to find elements to style, and the **declaration block**, the set of values to apply to properties.

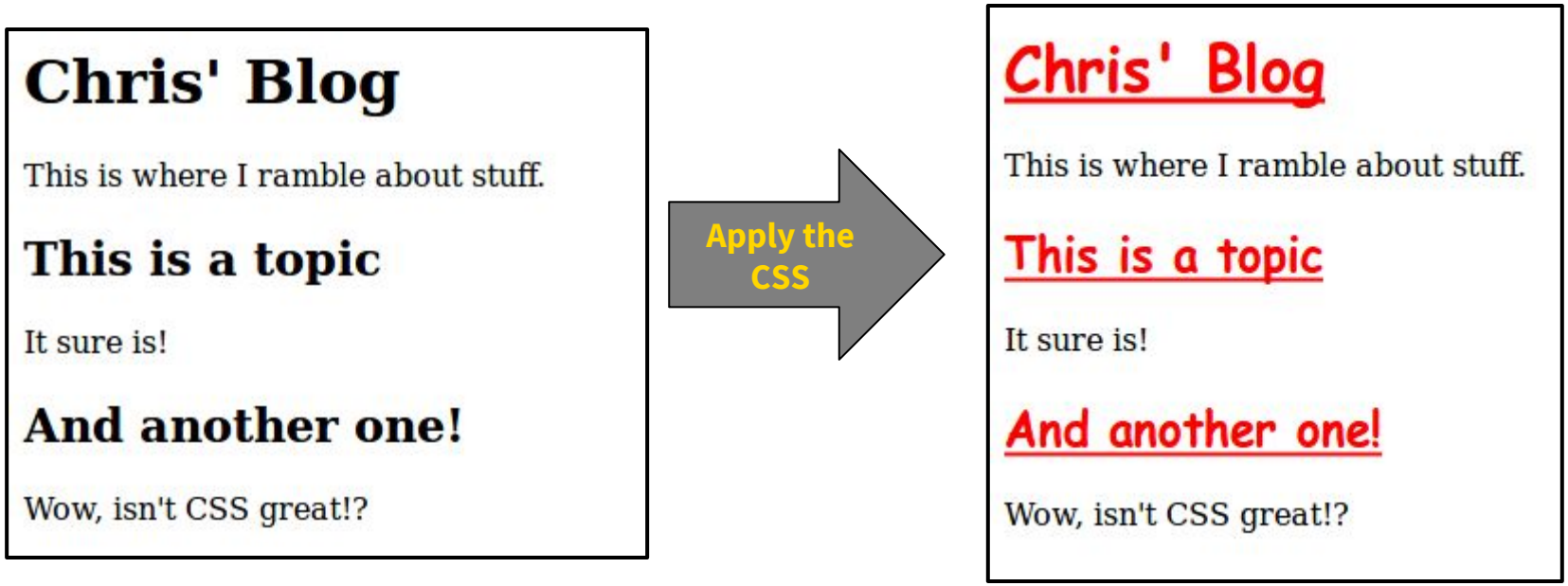
```
h1, h2
{
    color: red;
    font-family: "Comic Sans MS";
    text-decoration: underline;
}
```

selector (apply this style to all h1 and h2 elements)

Set the **color** property to **red**

Set the **font-family** property to "**Comic Sans MS**"

CSS 101 - How does it work?



Actually, I think I've made it worse. At least it's *unique.*

CSS 101 - Adding CSS - Best Practices

There are 3 common ways to add CSS rules to your pages.

However, there is only one (1) best practice.

We will discuss each of these:

1. Inline stylings for individual HTML elements
2. `<style>` delcaration in the `<head>` section of an HTML page
3. Linked stylesheets

CSS 101 - Adding CSS - inline styling

You can also assign style inside HTML tags using the **`style`** attribute.

- Very straightforward
- Re-using style requires copying and pasting :/

```
<p>
  <h1 style="color: red;">
  Style, now with attributes!
  </h1>
This doesn't scale well, though.
</p>
```

CSS 101 - Adding CSS - <style>

You can add CSS rules for all matching HTML tags in a file by using the <style> tag inside the <head> tag..

- Easy and quick to use.
- Not great if your site has multiple HTML pages.

```
<head>
    <style>
        h1 { color: red; }
    </style>
</head>
<body>
    <p>
    <h1> This is my page! </h1>
    There's nothing here yet... how did
you find this?
    </p>
</body>
```

This is my page!

There's nothing here yet... how did you find this?

CSS 101 - Adding CSS - Linked `.css` file

This method is most commonly used, since it's the easiest way to reuse a style across multiple HTML documents for consistency.

```
<head>
   <link rel="stylesheet" type="text/css"
         href="example.css" />
</head>
<body>
   <p>
   <h1> Linked Style </h1>
   The style is defined in example.css
   </p>
</body>
```

```
/* example.css */
h1 { color: red;}
```

CSS 101 - Common Style Attributes

CSS has **tons** of properties that you can set. Here are a few that you may find useful:

`color`	Sets the text color.	`text-style`	Sets if text is underlined.
`background-color`	Sets the color behind an element.	`text-align`	Centers text
`font-weight`	**Sets** the font **weight** (how **bold** it is)	`text-transform`	ALL CAPS all lowercase All Camel Case
`font-family`	Sets the font.	`background-image`	

https://www.w3schools.com/cssref/

CSS 101 - Selectors

Selectors are the patterns used to find elements to apply your style to. There are many types of selectors, but here are just a few.

- `.myclass` Selects all elements with the class attribute set to `"myclass"`
- `#id` Selects the element with the id attribute set to `"id"`
- `*` Selects all elements
- `h1` Selects all `h1` elements
- `h1, h2` selects all `h1` and `h2` elements.
- `p b` selects all `b` elements inside of `p` elements.

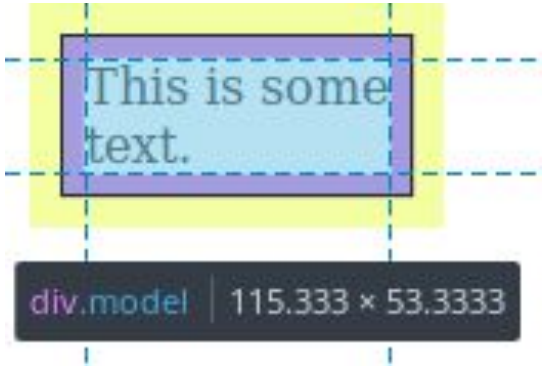
https://www.w3schools.com/cssref/css_selectors.asp

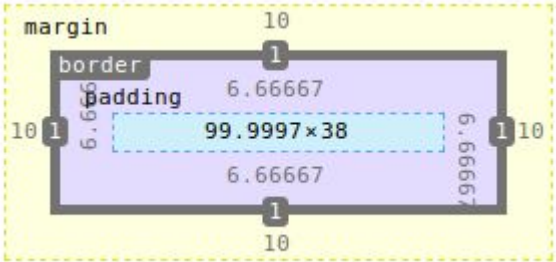
CSS 101 - Box Model

display: block This determines how the element is positioned in the page, by treating everything in that element as a rectangle.

By combining the size of the element's **margins**, **border**, **padding**, and **contents**, the overall size of the element can be determined.

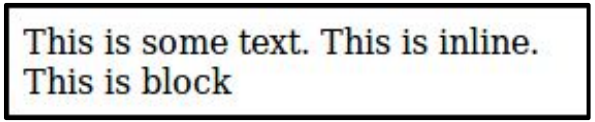
Margins != Padding Margins are the outer space of the element, padding is the inner space.





CSS 101 - Display Modes and Positioning

The **`display`** property changes the way that elements are positioned relative to each other.



- **`display: block`** shows the element on a new line
- **`display: inline`** shows the element on the same line

The position property changes the way elements are positioned in the page.

- **`position: static`** shows elements as they appear in the order of the DOM Tree
- **`position: fixed`** shows elements relative to the browser window

Useful Tip: In-Browser Debugging F12

All modern browsers include debugging tools with a similar set of functionality.

In Firefox and Chrome, these can be accessed with F12, or by right-clicking and selecting “Inspect”.

- The inspector allows you to interactively see your HTML source, and adjust styles in real time.
- The console lets you run JavaScript (more on that later).
- The network tab lets you debug every network request.

Try this on your favorite websites!

JavaScript 101

JS 101

JavaScript is a **scripting** language that runs **in your browser**.

Don’t let the name fool you: **JavaScript is not Java**, and has a very different API

JavaScript runs in a sandbox in your browser.

JavaScript is an interpreted or just-in-time (JIT) compiled language.

HTML and CSS can’t make things interactive, so that’s where JavaScript comes in.

https://www.w3schools.com/js/

JavaScript != Java

JS 101 - Syntax

Some notable quirks of JavaScript syntax:

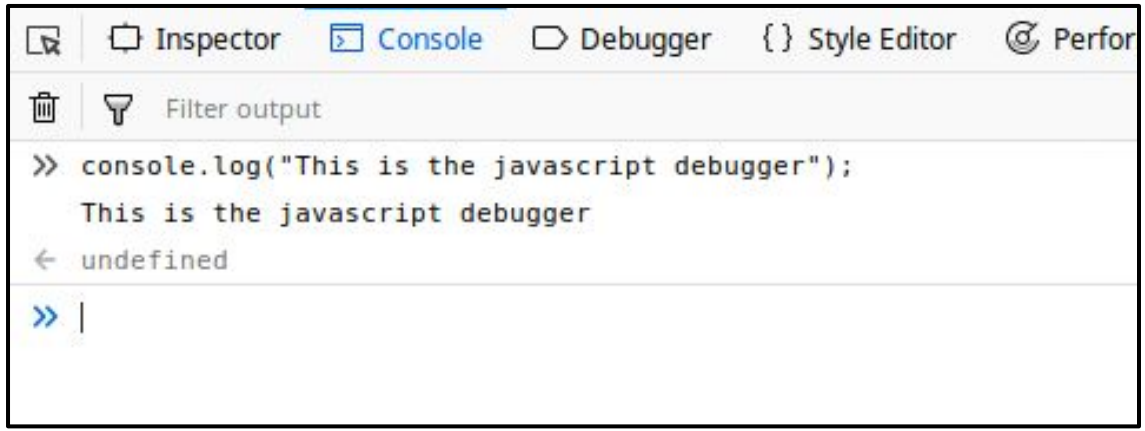
- JavaScript is weakly typed
  - `==` operator compares value (`"1" == 1` is `true`)
  - `===` operator compares value **and type** (`"1" === 1` is `false`)
  - typeof() operator to checks the type of the variable
- Strings can use “double quotes”, or ‘single quotes’
- Comments use // or /* */
- Types are dynamic, one variable can store different types.
  - Not many types: Strings, Numbers (always floating), Booleans, Arrays, Objects, Null, Undefined
  - Use the var, const, or let keywords to define variables.
- Functions are objects

https://www.w3schools.com/js/

JS 101 - Browser Debugger

All modern browsers provide a JS console that you can use for debugging. It's interactive and easy to use. Great for learning things for the first time.

Just open the developer tools with F12, and navigate to the Console.

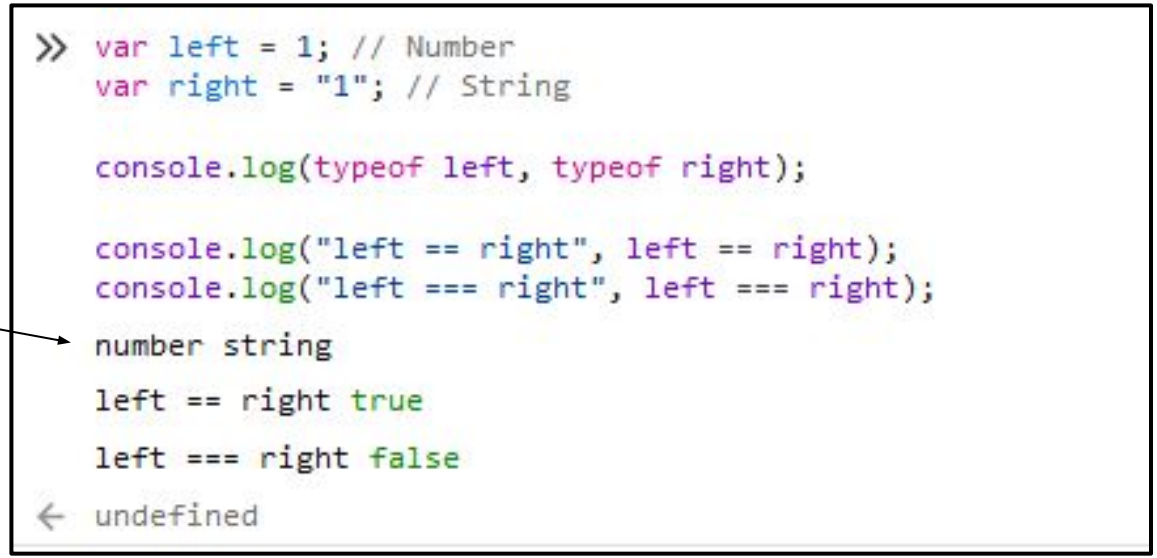


JS 101 - Browser Debugger Example

Code that we want to run

Console output

Return value



This is example also highlights the difference between the == and === operators.

JS 101 - Common Stuff

You can do a lot with JS, but we can't cover everything today.

https://www.w3schools.com/js is an excellent resource for going further with this topic.

`console.log("message")`	Logs "message" to the console. You can also pass objects, not just strings into this method.	`Element.innerHTML`	Gets or sets the string containing the content of the element.
`document.GetElementById("my-element")`	Gets an element from the DOM tree with the matching ID.	`document.createElement`	Creates a new HTML element that can be appended to the DOM

JS 101 - Objects

- Objects can be thought of as wrappers around variables within your JS code.
- Represented as key/value pairs
- Variables assigned within objects can be anything (even functions!)
- Object properties can be re-assigned, deleted, or created.

```
var obj = {
  name: "Carrot",
  "for": "Max",
  details: {
    color: "orange",
    size: 12
  }
}

> obj.details.color   > obj["details"]["size"]
orange                12
```


47

JS 101 - JSON = JavaScript Object Notation

- A lightweight data-interchange format.
  - **Language independent** way of transmitting text. JSON libraries exist for nearly everything.
  - Consists of key-value pairs (a Dictionary), or lists.
- Very commonly used with REST APIs.
  - XML is the next alternative.

JS syntax is used to define an object.

This is stored as a String, using the same syntax.

The Object's attributes be read from this string, then accessed like any other Object.

```
var myObj = { name: "John", age: 31, city: "New York" };
var myJSON = JSON.stringify(myObj);
console.log(myJSON);
// "{\"name\":\"John\",\"age\":31,\"city\":\"New York\"}"
// Quotation mark escaping works the same as other
languages

var obj = JSON.parse(myJSON);
console.log(obj.name);
// "John"
```

https://www.w3schools.com/js/js_json_intro.asp

JS - Security Considerations

At 2:45 PM today, we'll be hosting a presentation about security considerations that are relevant to web apps.

Where to get help if/when you get stuck

https://www.w3schools.com - w3schools is a fantastic resource for learning html/css/js (among others)

https://developer.mozilla.org/en-US/ Mozilla Developer Network Documentation

https://developer.mozilla.org/en-US/docs/Web/HTML

Additional Topics & Technologies

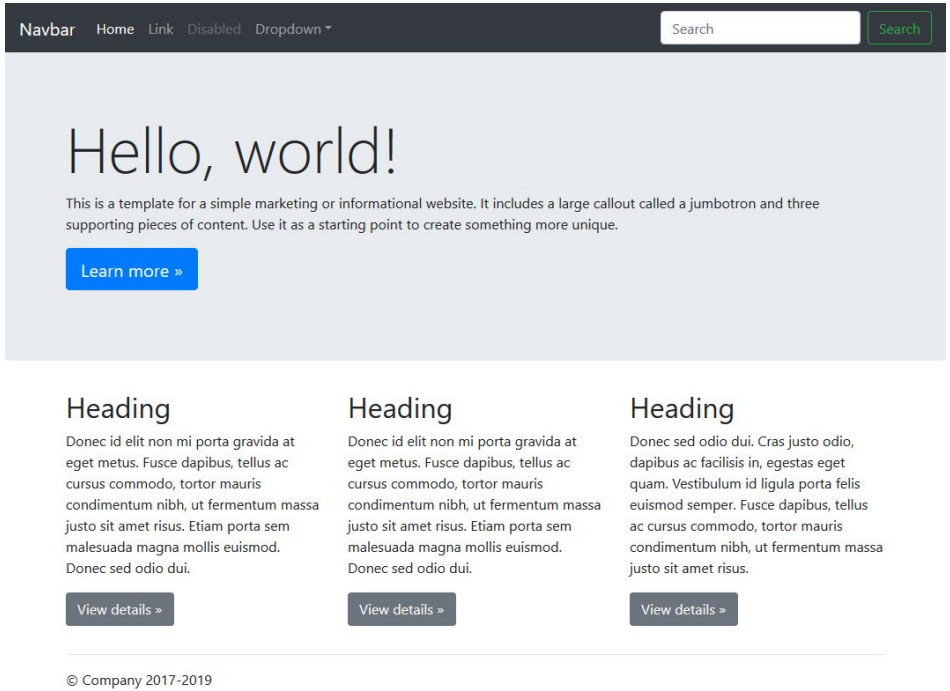
Further Topics - Bootstrap

https://getbootstrap.com/

Bootstrap is a fantastic CSS framework that does the hard stuff for you, including responsiveness and mobile support.

It's very commonly used for a reason, even the UW website uses it!

Their documentation is also really good.



Further Topics - jQuery

https://jquery.com/

jQuery is a JavaScript library which makes things quicker and easier.

Turn `document.getElementById("button").innerHTML = "Next"` into `$("button").html("Next")`

It's also used by Bootstrap.

Further Topics - TypeScript

https://www.typescriptlang.org/

TypeScript is an extension of JavaScript which adds better support for types, among other language features. It compiles into JavaScript.

This is a slightly more advanced topic, as you'll also want to look into Node.JS, NPM, and Gulp.

…And So Much More!

Intro to Web Servers with Python Flask Workshop begins at 12:45 PM

Questions?