JavaScript For Web

Week 1, Lecture 2 - Web Basics Review and JavaScript Fundamentals

Instructor: Jason Xu

Today's Overview

- HTML/CSS Review
- JavaScript Fundamentals: Variables
- Tutorial: Assignment 1, Exercise 1

HTML Review

HTML(Hypertext Markup Language) is a language that determines how documents and web pages are displayed in a web browser, the language for the building blocks of any website.

- Elements and Tags
- HTML basic structure
- Text, links, images

HTML Review: Elements and Tags

A full paragraph element looks like this:

```html
<p>some text content</p>
```

- `<p>` is the opening tag.
- `some text content` is the content wrapped within the opening and closing tags.
- `</p>` is the closing tag.
- Some HTML elements does not have a closing tag, e.g. `<br />` and `<img/>` because these self-closing tags don't wrap any content.
- Check list of predefined tags in HTML.

HTML Review: Basic Structure

```html
<!DOCTYPE html>  <!-- HTML5 doctype declaration -->
<html lang="en"> <!-- The root element -->
  <head>  <!-- meta-info about the webpage -->
    <meta charset="UTF-8"> <!-- charset encoding standard -->
    <title>My First Webpage</title> <!-- displayed on the browser tab -->
  </head>

  <body> <!-- all displayed contents -->
    <p>some text content</p>
  </body>
</html>
```

- VSCode shortcut: create a empty `***.html` file, enter `!` on the first line and press `Enter` key to choose the first one.

HTML Review: Text elements.

- `<p>` creates a paragraph.
- `<h1>` to `<h6>` creates 6 different levels of headings(larger and bolder than other text).
- `<em>` *makes text italic.*
- `<b>` **makes text bold.**
- `<!--` and `-->` enclose the comments.
- `<div>` creates an empty container.

Shortcut for HTML comments:

- Mac Users: `Cmd` + `/`
- Windows and Linux Users: `Ctrl` + `/`

HTML Review: Links

```
<a href="https://ccccollege.com/course" target="_blank">click me</a>

<a href="./pages/about.html">About Us</a>
```

- `herf` specifies the destination link.
- `target` specifies where the linked resource will be opened. `_blank` means open in a new tab.
- **Absolute link**: on other websites in the Internet. e.g. `protocol://domain/path`
- **Relative link**: starts from the project folder. e.g. `./pages/about.html`

HTML Review: Images

```html
<img src="./assets/img/logo-310x310.png" alt="The Logo">
```

- `alt` attributes hold a textual replacement for the image. If the image cannot be displayed, Alt text will be displayed.

CSS Review

CSS(Cascading Style Sheets) a style sheet language that determines how a document created in HTML is styled (colors, font styles, layout and responsive features).

- Add styles to HTML
- `class` and `id` attributes
- The cascade

CSS Review: Add style to HTML

```html
<!-- index.html -->

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

<body>
  <div>Hello, World!</div>
  <p>Hi...</p>
</body>
```

```css
/* styles.css */

* {
  color: purple;
}

p {
  background-color: black;
}
```

- **Universal selector**: select elements of any type, every element would have the style applied to it
- **Type selector**: select all elements of given element type(tags).

CSS Review: Class and ID selectors

```html
<!-- index.html -->

<div class="homepage alert-text">Please agree to our terms of service.</div>

<div id="title">My Awesome 90's Page</div>
```

```css
/* styles.css */

.alert-text {
  color: red;
}

#title {
  background-color: red;
}
```

- The major **difference** between classes and IDs is that an element can only have **one** ID.
- ID cannot be repeated on a single page and should not contain any whitespace.

CSS Review: The Cascade

The cascade is what determines which rules actually get applied to our HTML when we have rules that conflict with one another.

- A CSS declaration that is more specific will take precedence over less specific ones.
    - ID(most specific selector) > Class > Type selectors

```html
<!-- index.html -->

<div class="main">
  <div class="list" id="subsection">Blue text</div>
</div>
```

```css
/* rule 1 */
#subsection {
  color: blue;
}

/* rule 2 */
.main .list {
  color: red;
}
```

CSS Review: The Cascade continued

- When no declaration has a selector with a higher specificity, a larger amount of a single selector will beat a smaller amount of that same selector.

```html
<!-- index.html -->

<div class="main">
  <div class="list" id="subsection">Blue text</div>
</div>
```

```css
/* rule 1 */
#subsection {
  color: blue;
}

/* rule 2 */
.main .list {
  color: red;
}
```

- The cascade can be very difficult to identify when we increase the complexity of specificity, chaining and Inheritance. How can we understand that?

The Inspector

In Chrome browser, press F12 OR right-click and click "inspect". We can see the HTML and CSS of current webpage under the Element and Styles panel.

The most important thing for today: when we develop a website, make sure this browser inspector open all the time.

Responsive Design

- “Responsive Design” is the term used to describe creating websites that respond to changes in browser size in order to create something that works on any device.
- **The browser inspector** can help us test how websites response in different screen size.
- **Again**, when we develop a website, make sure this browser inspector open all the time.

JavaScript Fundamentals

JavaScript Fundamentals Overview

- "Hello World" in JavaScript.
- How to declare a variables?
- Rules for naming variables.

Hello World in JavaScript

```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Page Title</title>
</head>
<body>
  <script>
    // Your JavaScript goes here!
    console.log("Hello, World!")
  </script>
</body>
</html>
```

`console.log()` is the command to print something to the developer console in your browser.

- The most important command in JavaScript.

Hello World in JavaScript continued

JavaScript file have the extension `.js` similar to `.css` for stylesheets. External files are used for more complex scripts:

```html
<!-- index.html -->

<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Page Title</title>
</head>
<body>
  <script src="hello.js"></script>
</body>
</html>
```

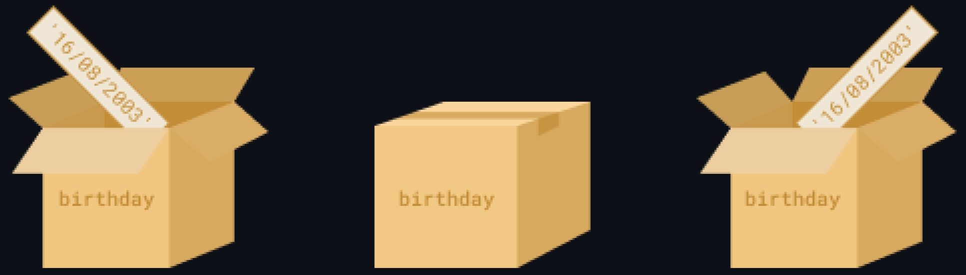
```js
//hello.js

console.log("Hello World")
```

My suggestion: Keep the code for each purpose separate all the time:

- **HTML**: Content & Layout
- **CSS**: Styling
- **JavaScript**: Interactive Elements

JavaScript Variables

- A variable is a "named storage" for data. You can think of variables as simply "storage containers" for data in your code:



- To create a variable in JavaScript, use the `let` keyword:

```
let birthday;
```

- we can put data into it by using the assignment operator `=`:

```
let birthday;
birthday = '16/08/2003'; // store the string '16/08/2003' in the variable named birthday
```

JavaScript Variables

- To be concise, we can combine the variable declaration and assignment into a single line:

```
let birthday = '16/08/2003';
```

- We can also declare multiple variables in one line:

```
let username = 'John', age = 25, message = 'Hello';
```

- It might seem shorter, but for the better readability, we recommend to use a single line per variable:

```
let username = 'John';
let age = 25;
let message = 'Hello';
```

- Comment about `;`: Semicolon in every programming language is very important for the compiler to understand the code. It denotes the end of the line in these languages but in the case of JavaScript, **it is not necessary to add a semicolon to each line**:

```
let message = 'Hello'
```

JavaScript Variables: `var` and `let`

```
let message = 'Hello'
```

- In order version of JavaScripts, you may also find another keyword `var`:

```
var message = 'Hello'
```

The `var` keyword is almost the same as `let`. It also declares a variable, but in a slightly different, “old-school” way. There are subtle(significant) differences between `let` and `var`, we will discuss when we talk aboout ES6 later in this course.

- For now, let's remember **never use `var` in your code**.

JavaScript Variables: Change the value

- We can change the variable value as many time as we want:

```
let message
console.log(message)

message = 'Hello!'
console.log(message) // value changed

message = 'World!' // value changed
console.log(message)
```

- Here is the console output:

```
undefined
Hello!
World!
```

JavaScript Variables: Change the value

- We can also declare two variables and copy data from one into the other.

```
let hello = 'Hello world!'

let message

// copy 'Hello world' from hello into message
message = hello

// now two variables hold the same data
console.log(hello)
console.log(message)
```

- Here is the console output:

```
Hello world!
Hello world!
```

JavaScript Variables: Declaring twice triggers an error

- A variable should be declared only once.
- A repeated declaration of the same variable is an error:

```
let message = "This"

// repeated 'let' leads to an error
let message = "That" // SyntaxError: 'message' has already been declared
```

Here is the console output:

```
Uncaught SyntaxError: Identifier 'message' has already been declared.
```

How to fix that:

```
let message = "This"
message = "That"
```

OR

```
let message1 = "This", message2 = "That"
```

JavaScript Variable: `const` and `let`

- To declare a variable using `let`, we can change the variable value as many time as we want:

```
let message = 'Hello!'
message = 'World!' // value changed
```

- To declare a constant(unchanging) variable, use `const`:

```
const birthday = '16/08/2003'
```

- Variables declared using `const` are called “constants”. They cannot be reassigned. An attempt to do so would cause an error:

```
const birthday = '16/08/2003'

birthday = '25/09/2004' // error, can't reassign the constant!
```

- Here is the output error:

```
Uncaught TypeError: Assignment to constant variable.
```

Rules for naming variables.

There are two limitations on variable names in JavaScript:

- The name must contain only letters, digits, or the symbols `$` and `_`.
- The first character must not be a digit.

Example of valid names:

```
let userName
let test123

let $ = 1 // declared a variable with the name "$"
let _ = 2 // and now a variable with the name "_"
```

Example of incorrect variable names:

```
let 1a // cannot start with a digit
let my-name // hyphens '-' aren't allowed in the name
```

When the name contains multiple words, camelCase is commonly used. That is: words go one after another, each word except first starting with a capital letter: `myVeryLongName`.

Rules for naming variables continued.

Case matters(sensitive).

- Variables named `apple` and `APPLE` are two different variables.

Non-latin letters are allowed, but not recommanded:

```
let 我 = '...'
```

- My suggestion: avoid non-latin letters all the time.

Reserved names

- There is a list of reserved words, which cannot be used as variable names because they are used by the language itself.
- For example: `let`, `class`, `return` and `function` are reserved.
- The code below gives a syntax error:

```
let let = 5; // can't name a variable "let", error!
let return = 5; // also can't name it "return", error!
```

Summary

- HTML/CSS Review
- VS code shortcut for commenting out and creating HTML basic structure.
- Absolute path vs. relative path.
- **Use browser inspector**(Test HTML/CSS, Responsive Design, console, network, etc.)
- Declare JavaScript variables using `let` and `const`, never use `var`
- Rules for naming variables.

The most important for today: Keep your browser inspector open all the time.

Thank you.