Web Fundamentals Refresher

Download Demo Code <../web-fundamentals-refresher-demo.zip>

Goals

- Make sure you're ready for this course!
- Review essential topics in HTML and CSS
- Review essential topics in JavaScript

Essential Downloads

You need to have each of the following installed on your machine:

- Text editor - **VSCode highly recomended**
- Web browser - **Chrome highly recomended**

HTML Must Knows

- You should be familiar with the following `html` elements:

Essential html elements

```
<div></div>

<h1></h1> -> <h6></h6>

<p></p>

<span></span>
```

You should be able to explain block element vs inline element

Html list elements

```
<ul></ul>

<ol></ol>

<li></l1>
```

What is the difference a `ul` and a `ol` tag?

- Know these additional html elements:

```
<script></script>

<link rel="" type="" href="" />
```

```
<a href=""></a>

<img src="" alt="" />
```

- What is the `script` tag used for?
- What is the difference between a `link` tag and a `anchor` tag?
- Why is it important to include an `alt` attribute inside of an `img` tag?

- Understand the main elements of an html table:

```
<table>
  <thead>
    <tr>
      <th></th>
      <th></th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td></td>
      <td></td>
    </tr>
    <tr>
      <td></td>
      <td></td>
    </tr>
  </tbody>
</table>
```

HTML forms

```
<form>
  <label for="name">name</label>
  <input type="text" name="name" placeholder="" />
  <input type="submit" value="Submit" />
</form>
```

- Understand how to connect a label to an input tag using the `for` attribute
- Understand the most common input tag attributes
  - type
  - name
  - placeholder
  - value

CSS Must Knows

CSS specificity

- Why is the specificity of our selectors important?
- In the following code, what color will the background of the h1 tag be?

```
<!DOCTYPE html>
<html>
  <link rel="stylesheet" href="main.css"
  <body>
    <div>
      <h1 id="id-style" class="class-sty
    </div>
  </body>
</html>
```

```
h1 {
  background: crimson;
}

.class-style {
  background: violet;
}

#id-style {
  background: dodgerblue;
}
```

CSS selectors

- Know when to use a class selector and when to use id selector
- What is wrong with the following code?

```
<!DOCTYPE html>
<html>
  <link rel="stylesheet" href="main.css"
  <body>
    <div>
      <h1 id="id-style">Hi</h1>
      <h1 id="id-style">There</h1>
    </div>
  </body>
</html>
```

```
h1 {
  background: crimson;
}

.class-style {
  background: violet;
}

#id-style {
  background: dodgerblue;
}
```

CSS properties

- You should be comfortable styling html elements with common css properties

```
div {
  background: crimson;
  color: grey;
  font-family: Helvetica;
  font-size: 18px;
  cursor: pointer;
}
```

- You should be comfortable controlling the space between html elements
- Understand the difference between padding and margin

```
div {
  width: 10%;
  height: 20%;
  margin: 10px;
  padding: 20px;
}
```

- Understand how to control the amount of margin and padding on the:
  - top
  - right
  - bottom
  - left

```
div {
  margin-top: 10px ;
  margin-right: 12px;
  margin-bottom: 10px;
  margin-left: 8px;
}
```

CSS shorthand properties

- Be able to apply shorthand properties to the following property name types:
  - margin
  - padding
  - border

```
div {
  margin: 10px 12px 10px 8px;
}

div {
  margin: 20px;
}

div {
  border: 2px solid grey;
}
```

CSS positioning

- Understand how to position elements using the following property types:
  - static
  - fixed
  - relative
  - absolute

JavaScript Must Knows

Declaring variables

You do not have to know all the details of var, let, and const. However, you should be familiar with these concepts:

Keyword	Can Reassign	Can Redeclare	Scope Rules
var	yes	yes	function
let	yes	no	block
const	no	no	block

We will be using let and const primarily. You'll dive into the differences later in the course.

Conditional logic

- You should understand differences between:
  - if
  - else if
  - else

You should understand 'js expressions' are always converted to a boolean value when passed to a control statement

```
if (<js expression>) {

} else if (<js expression>) {

} else {

}
```

You should be able to explain the difference between the following two snippets of code

```
let n = 10;

if (n > 0) {
  console.log('n is valid')
} else if (n < 100) {
  console.log('n is valid')
} else {
  console.log('n is not valid')
}
```

```
let n = 10;

if (n > 0) {
  console.log('n is valid')
}

if (n < 100) {
  console.log('n is valid')
} else {
  console.log('n is not valid')
}
```

Logical operators

- You should have an understanding of the logical operators:
    - OR - ||
    - AND - &&
    - NOT - !

Primitive data types

- You should be familiar with the following 5 primitive data types:
    - Numbers
    - Strings
    - Booleans
    - Undefined
    - Null

Numbers

- You should be comfortable with:

  -Converting a number to a string

    - Generating a random number
    - Rounding a number the following functions:
        - Math.round()
        - Math.ceil()
        - Math.floor()

Strings

- You should be comfortable with:
  - Creating a string
  - Converting a string to a number
  - Iterating through each element in a string
  - Making a copy of a string

Booleans

- You should know the difference between a 'truthy' and 'falsy' value
- Know the following 6 'falsy' values in Javascript:
  - undefined
  - 0
  - "
  - false
  - null
  - NaN
- Know two approaches for converting an expression to 'truthy' or 'falsy'
  - Boolean(<expression>)
  - !!<expression>

Iteration

- You should be very comfortable with the syntax for iterating through a string or an array using a 'for loop'

```
for (let i = 0; i < array.length; i++) {
  console.log(array[i]);
}
```

- You should understand the syntax for iterating through a string or an array using a 'while loop'

```
let i = 0;

while (i < array.length) {

  console.log(array[i]);

  i++;
}
```

You should understand the difference between a 'for…of' and 'for…in' loop

```
let arr = ['a', 'b', 'c', 'd'];

for (let n of arr) {
  // what will n be?
  console.log(n);
}
```

```
let arr = ['a', 'b', 'c', 'd'];

for (let n in arr) {
  // what will n be?
  console.log(n);
}
```

- You should comfortable writing a nested for loop
- For example, how would you print each element in the following array of sub arrays?

```
let matrix = [
  ['a', 'b', 'c'],
  ['d', 'e', 'f'],
  ['g', 'h', 'i'],
]

for (let i = 0; i < matrix.length; i++) {
  let subArr = matrix[i];

  for (let j = 0; j < subArr.length; j++) {

    console.log(subArr[j]);
  }
}
```

Arrays

- You should be comfortable with:
  - Creating an array
  - Getting and setting elements in an array
  - Iterating through arrays
  - Making copies of arrays

Objects

- You should be comfortable with: - Creating objects
  - Getting and setting key value pairs in an array
  - iterating through objects
  - making copies of objects
- You should know the difference between dot and bracket notation
- You should understand the subtle differences between the following code samples:

```
let arg = 'hi';
let obj = {}

obj.arg = 'there';

  => { arg: 'there' }
```

```
let arg = 'hi';
let obj = {};

obj[arg] = 'there';

  => { hi: 'there' }
```

- Dot notation uses the literal ‘arg’ string as the key
- Bracket notation allows you to pass arguments dynamically

Arrays and objects

- You should be understand what the following code will do and why

```
// will this return true or false?

[] === []
```

- Here we are comparing if the array is the same actual array not the values held in the array
- The same result happens when comparing objects

Functions

- You should know the syntax for creating functions with and without parameters
- Understand how to return values from a function
- How to invoke functions with and without arguments
- function scope vs global scope
- Placing functions on objects (methods)