Meta Elevate - Meta Front End Developer

1. DOM
    a. Document Object Model
    b. Tree structure of the objects
    c. The DOM has a series of objects, each representing a single HTML element
    d. At the root of the DOM is the HTML object
    e. HTML object contains the head and body objects
    f. Head object houses the title object which holds the text object
    g. The body object houses the DIV objects which then can contain header, paragraph, and text objects such as ‘<div>’, ‘<h1>’, ‘<p>’, and ‘text’.
    h. The Document Object Model allows you to update all HTML elements on a web page.
    i. Many JavaScript libraries and frameworks rely on the DOM, one of these libraries is the react library.
2. Webpages
    a. Typically have hundreds of elements
3. Common DOM and Javascript uses:
    a. Updating the elements on an HTML page based on interaction from the user
        i. Notification for incorrect password
        ii. Notification for message received
        iii. Interaction with a video player
4. Accessibility
    a. In this course we will learn the proper way to use the DOM and HTML elements so that the content is accessible to everyone
    b. Having text that is not contained within proper tags like paragraphs or heading tags makes it harder for assistive technologies to interact with the content.
5. Extra resources
    a. HTML Elements Reference (Mozilla)
        i. https://developer.mozilla.org/en-US/docs/Web/HTML/Element
    b. The Form Element (Mozilla)
        i. https://developer.mozilla.org/en-US/docs/Web/HTML/Element/form
    c. What is the Document Object Model? (W3C)
        i. https://www.w3.org/TR/WD-DOM/introduction.html
    d. ARIA in HTML (W3C via Github)
        i. https://w3c.github.io/html-aria/
    e. ARIA Authoring Practices (W3C)
        i. https://www.w3.org/TR/wai-aria-practices-1.2/
6. Selecting and Styling
    a. The HTML is the frame and stricture of the building
    b. CSS is the paint, wallpaper, fixtures, artwork, and overall style or look and feel. In other words, CSS tells the web browser how to display HTML elements on the screen.
    c. CSS rule

i. A declaration block starts with a left curly brace and ends with a right matching curly bracket
ii. In between the curly brackets are the style declarations
iii. The first element of the CSS rule is he selector
    1. This indicates the CSS element or elements we want to style.
iv. Example
    1. H1 {
        Color: green;
    }
        a. H1 is the selector
        b. Color is the property
        c. Green is the value
v. ID selector
vi. “What if I only want to style one particular type of element on a web page; ie. not all h1 elements but just one h1 element - id selector
    1. In HTML document, Add an ‘id’ to the html tag - <h1 id=”header1”>
    2. In CSS document, reference id with # like #header1
        a. Example
            i. #header1 {
                Color: green
            }
vii. Class selector
    1. Good for selecting multiple elements of the same class without selecting all elements
    2. Tighter scope of element selection than root element.
    3. Looser scope of element selector than ID.
    4. Hierarchy should be
        a. Root element
        b. Class(es) of similar elements
        c. IDs for very specific elements
    5. IDs and classes can be used interchangeably I believe but should not be
    6. Element with class selector:
        a. HTML - <p class=”introduction”>
        b. CSS:
            i. P.introduction {
                Margin:2px;
            }
    7. Descendant selectors
        a. Descendant selectors are useful if you need to select HTML elements that are contained within another selector.
            i. Example:

                1. HTML:
                   ```
                   <div id="blog">
                   <h1>Latest News</h1>
                   <div>
                   ```
                2. CSS:
                   ```
                   #blog h1 {
                           color: blue;
                   }
                   ```
        b. The CSS rule will select all h1 elements that are contained within the element with the ID blog. The CSS rule will not apply to the h1 element containing the text Archives.
    8. Child Selectors

7. Text and Color
    a. RGB
        i. A color model that creates colors by adding varying degrees of red, green, and blue together
        ii. This is how the human eye sees color
    b. RGBA
        i. An extension of the RGB color model which adds the alpha channel. The alpha channel adds opacity or transparency of the color
    c. HSL
        i. A newer color model that is defined as hue, saturation, and lightness.
8. Box Model
    a. Hiearchy
        i. Margin
        ii. Border
        iii. Padding
        iv. Content
    b. Document Flow
        i. Block vs inline elements
            1. Block take up the full width of the screen and the height of the content
            2. Inline take up as much width and height of the content and can be on the same line, hence the name inline
    c. Additional CSS resources
        i. CSS Reference (Mozilla)
        ii. https://developer.mozilla.org/en-US/docs/Web/CSS/Reference
        iii. HTML and CSS: Design and build websites by Jon Duckett
        iv. https://www.amazon.com/HTML-CSS-Design-Build-Websites/dp/1118008189/
        v. CSS Definitive Guide by Eric Meyer
        vi. https://www.amazon.com/CSS-Definitive-Guide-Visual-Presentation/dp/1449393195/
9. Working with libraries

a. Libraries and frameworks are called dependencies
    i. Components can have multiple dependencies. A dependency can have a dependency of its own. This is called a dependency tree. Large projects can have 100s of dependencies in its tree
    ii. Dependency relying on other frameworks and libraries = dependency
    iii. Package manager takes care of downloading all needed dependencies
    iv. NPM is the most common package manager for Front End Development
    v. Node Package Manager = NPM
    vi. You will use a bundling tool to gather all your dependencies and combine them so that they can be referenced from your HTML file.

b. A web page and its components need the dependencies to be uploaded/present on the web server in order to render on the internet

10. Introduction to responsive design
    a. 3 principles
        i. Flexibile Grids
        ii. Fluid Images
        iii. Media Queries
11. Getting started with bootstrap
    a. <div class="container">
        i. <div class ="row">
            1. <div class ="col">
                a. <h1>Our Menu</h1>
                b. <h2>FalafelM/h2>
                c. <p>Chichpea, herbs and spices.</p>
                d. <img src"image.jpg" class="img-fluid"/>
                    i. The class "img-fluid" ensures that the image scales to its parent column's width
    b. The first element that needs to be added when setting up a bootstrap website is the container element.
12. Using Bootstrap styles
    a. Bootstrap has a large CSS library built by Bootstrap's developers using thousands of use cases
    b. Class Fixes
        i. Response breakpoints
            1. Extra small < 576
            2. Small (sm) >= 576px
            3. Medium (md) >= 768px
            4. Large (lg) >= 992px
            5. Extra large (xl) >= 1200px
            6. Extra extra large (xxl) >= 1400px
        ii. A class abbreviation does not exist for Extra small because this is the default breakpoint in Bootstrap CSS rules
    c. Modifiers

        i. Bootstrap modifiers add a CSS class to change the visual style of components.
        ii. (in context of alerts)
            1. Primary <- uses bootstrap's default color; blue
            2. Secondary
            3. Success
            4. Info
            5. Warning
            6. Danger <- uses color red
            7. Light
            8. Dark
    d. <div class="col-6"> becomes <div class="col-lg-6"> (large)

13. Bootstrap Grid
    a. Building a website using responsive design requires a responsive grid and responsive breakpoints
    b. Bootstrap provides both of these as part of its library
    c. The bootstrap grid system helps us to create web page layouts through a series of rows and columns that house our content. The bootstrap grid system always has a container, rows, and columns
    d. The container is the root element of your grid.
14. Bootstrap components
    a. Bootstrap includes a pre-made set of UI elements and styles to help you build your website.
        i. These are called bootstrap components
    b. Card component
    c. Alert component
    d. Alert-info component (makes alert blue)
15. SPA = Single Page Applications
    a. Examples
        i. Social Media Network
        ii. Maps app
        iii. Messenger App
    b. Traditional multi-page websites became server intensive as it takes more resources to load individual pages.
    c. Two ways of delivering code to browser:
        i. All at once - Bundles
        ii. Dynamically as needed - Lazy Loading
    d. User interacting with the page updates the page contents
        i. PUT request to server
            1. Server returns JSON - which is less resource heavy than returning a full page
16. What is React?
    a. Available since 2013
    b. Open source Javascript library

   c. Core contributors and companies maintain it
   d. Can develop single page applications
   e. Can also develop mobile applications with React Native
   f. The key concept behind React is that it allows you to define components that you can combine to build a web application.

17. React component
    a. Small piece of user interface
       i. Examples
          1. Music player
          2. Photo gallery
    b. This component model allows:
       i. Isolated Development
       ii. Isolated Testing
       iii. Re-using Components
18. How React Works
    a. A React Element has a 1:1 relation to the HTML Element that displays on the page.
    b. React uses the virtual DOM to update the actual/Browser DOM when it needs to.
       i. This ensures your application is fast and responsive to users
       ii. Reach checks for changes to the virtual DOM and pushes updates from the virtual DOM to the Brwoser DOM as needed. This means elements of the DOM that are not updated remain the same - reconciliation
       iii. React updates the virtual DOM and compares it to the previous version of the virtual DOM. If a change has occurred, only that element is updated in the browser DOM. Changes on the browser DOM cause the displayed webpage to change.
       iv. React builds a representation of the browser Document Object Model or DOM in memory called the virtual DOM. As components are updated, React checks to see if the component's HTML code in the virtual DOM matches the browser DOM. If a change is required, the browser DOM is updated. If nothing has changed, then no update is performed.
       v. As you know, this is called the reconciliation process
       vi.
    c. React Fibre Architecture
       i. The principle of the React Fibre Architecture is its like a priority system
          1. What is most important to load and when?
    d. Component Hierarchy
       i. Every React application has at least 1 component
          1. Begins with root
             a. Example
                i. App
                   1. NewItemBar
                   2. ShoppingList
                      a. ShoppingItem

b. ShoppingItem
c. ShoppingItem
d. ShoppingItem

19. Programming With Javascript
    a. Week 1
        i. High-level and low-level languages
            1. The zeros and ones of binary code is a low-level language because it's closer to being understood by a CPU. JavaScript on the other hand is a high-level language. This means that it has to be converted to binary code so that a CPU can work with it.
            2. As a high-level language, JavaScript does need to be converted to binary code so that a CPU can work with it.
        ii. JavaScript invented in 1995?
        iii. Jquery was the first library to solve a lot of browser compatibility issues
        iv. React came along in 2011 and then some others followed
        v. Other Frameworks and Libraries:
            1. Vue.JS
            2. Angular
            3. Knockout
            4. Backbone
            5. Ember
        vi. With millions of websites containing JavaScript code from different versions and libraries, there is a lot of old code. This is known as a legacy code. While you probably won't use jQuery to build a modern website today, you might still come across it in a project that is still actively running. But don't worry, sometimes beginners think they have to learn or even master all the different technologies associated with JavaScript. However, that's not really necessary. To be a well-rounded developer, you need to learn and master the basics of plain JavaScript without the frameworks. Once you have this foundation, the pathway will become easier for you to learn a framework built on top of JavaScript, such as React.
        vii. Variables
            1. Declare by using
                a. Var
                b. Let
                c. Const
        viii. Data Types
            1. 7 primitive Data Types in JS:
                a. String - text in single or double quotes
                b. Number - numerical values
                c. Boolean - true/false
                d. Null - absence of value
                e. Undefined - unassigned value

f. BigInt - like an extra large box that can hold a very large range of numbers

g. Symbol - Unique identifier. Ike having a bunch of boxes that all say “Dishes” but each with different serial #s

ix. Operators

1. Are used to perform operations on variables and values.
2. An operator is used to manipulate individual data items and return a result
3. Assignment Operator - ‘=’
4. Arithmetic Operators
   a. ‘+’ - addition
   b. ‘-’ - subtraction
   c. ‘/’ - division
   d. ‘*’ - multiplication
5. Comparison Operators
   a. ‘>’ - greater than
   b. ‘<’ - less than
   c. ‘==’ - equal to (compares value not type)
   d. ‘===’ - equal to (compares value and type)
   e. ‘!=’ - not equal to
6. Logical operators
   a. ‘&&’ - Checks for both conditions to be true
   b. ‘||’ - checks for at least one condition to be true
   c. ‘!’ - returns false if the result is true
7. Modulus Operator - ‘%’
   a. This checks how many times you can fit one number into the other and then return the remainder. If I type 9 modulus 8, the value of one is returned, or 16 modulus 8 returns a zero. This is because the number eight divides into the number 16 evenly. There is no remainder as represented here by the zero.
8. Additional operators
   a. Logical AND operator: &&
   b. Logical OR operator: ||
   c. Logical NOT operator: !
   d. The modulus operator: %
   e. The equality operator: ==
   f. The strict equality operator: ===
   g. The inequality operator: !=
   h. The strict inequality operator: !==
   i. The addition assignment operator: +=
   j. The concatenation assignment operator: += (it's the same as the previous one - more on that later)

x. Numbers

            1. The number of data type is a foundational part of JavaScript as a programming language because it represents both integer and decimal point numbers.
            2. Without it, you wouldn't be able to code.
            3. Foundational data type that represents integers and decimal points
        xi. Strings
            1. Strings are used to represent and work with a sequence of characters while programming in JavaScript.
            2. You can use single or double quotation marks to make a string literal.
        xii. Booleans
            1. Used to check if a statement is true or false
            2. Booleans only have two values: true and false
            3.
    b. Week 3
        i. Programming Paradigms
            1. Human languages have many variations and forms such as formal/informal and slang. They all perform the exact same function, communication.
            2. Programming languages are similar and have many styles.
                a. Just like human languages no one style is better suited than the other.
            3. Two commonly used paradigms:
                a. Functional Programming (FP)
                b. Object-Oriented Programming (OOP)
        ii. camelCase
        iii. Functional Programming
            1. In functional programming, data and functions that operate on it are clearly separated, not combined inside objects.
            2. JavaScript allowing me to use the return keyword as described above, I can have multiple function calls, returning data and manipulating values, based on whatever coding challenge I have in front of me.
            3. Being able to return custom values is one of the foundations that makes functional programming possible.
            4. In functional programming, we use a lot of functions and variables.
            5. When writing FP code, we keep data and functionality separate and pass data into functions only when we want something computed.
            6. In functional programming, functions return new values and then use those values somewhere else in the code.
        iv. Object-Oriented Programming

            1. In this style, we group data and functionality as properties and methods inside objects.
            2. OOP helps us model real-life objects.
            3. It works best when the grouping of properties and data in an object makes logical sense; meaning the properties and methods “belong together”.
    v. “To summarize this point, we can say that the Functional Programming paradigm works by keeping the data and functionality separate. It's counterpart, OOP, works by keeping the data and functionality grouped in meaningful objects.”
    vi. There are many more concepts and ideas in functional programming
        1. Here are some of the most important ones:
            a. First-class functions
                i. It is often said that functions in JavaScript are “first-class citizens” which means that a function in JavaScript is just another value that we can:
                    1. Pass to other functions
                    2. Save in a variable
                    3. Return from other functions
                ii. A function in JavaScript is just a value; almost no different than a string or a number.
                    1. For example, in JavaScipt it is perfectly normal to pass a function invocation to another function.
            b. High-order function
                i. A high-order function is a function that has either one or both of the following characteristics:
                    1. It accepts other functions as arguments
                    2. It returns functions when invoked
                    3. There is no “special way” of defining high-order functions in JavaScript. It is simply a feature of the language. The language itself allows me to pass a function to another function, or return a function from another function.
            c. Pure functions and side-effects
                i. A pure function returns the exact same result as long as it’s given the same values.
                ii. Another rule for a function to be considered pure is that it should not have side-effects.
                iii. A side-effect is any instance where a function makes a change outside of itself
                    1. This includes:

                a. Changing variable values outside of the function itself, or even relying on outside variables
                b. Calling a Browser API (even the console itself)
                c. Calling Math.random() - since the value cannot be reliably repeated

- vii. Function Calling and Recursion
    1. Functions that repeat tasks are helpful; unless they run endlessly
    2. What recursive functions are
        a. When a function calls itself, this is what is known as recursion
    3. How to avoid getting stuck in an infinite loop
        a. Need a condition to stop. In the provided example, a variable was added above the function called counter and the condition ‘if counter = 0: return;’ within the function. After that middle line the function called itself again
            i. The order is important
- viii. Scope
    1. All about accessibility
        a. It determines which parts of the code are accessible and which parts are inaccessible.
        b. For example what variables can a function access within code?
    2. Scope Chain in Javascript
        a. The code that exists outside of a function is referred to as global scope
        b. All the code inside of a function is known as local scope or function scope.
        c. If a variable is defined within a function, then you can say it is scoped to that function. This is also known as local scope.
        d. This chain of scope references is referred to as the Scope Chain.
        e. Each function keeps a reference to its parent scope
    3. A nice way to think about how scope works in Javascript is a two-way mirror
        a. This is a piece of glass where only one side is transparent
        b. For example, if a restaurant uses two-way glass, people outside the restaurant can’t see what’s happening inside but the people inside can see what is happening outside.

c. Additional resources
- i. Here is a list of resources that may be helpful as you continue your learning journey.

ii. These resources provide some more in-depth information on the topics covered in this module.
iii. Mozilla Developer Network Expressions and Operators
iv. Mozilla Developer Network Operator Precedence and Associativity
v. JavaScript Primitive Values
vi. ECMA262 Specification
vii. jQuery Official Website
viii. React Official Website
ix. StackOverflow Developer Survey 2021 Most Popular Technologies
x. Emojis

20. Writing Statements
    a. Conditional Statements
        i. If statement
            1. The if statement checks a condition and will execute a code block if the condition is met or true.
        ii. If Else statement
            1. Same as above with added functionality. If something is true execute nested code block under if. If not true (else) execute nested code block under else.
            2. Better suited for
        iii. Else-If statement
            1. Same as above with added functionality. If something is true execute nested code block under if. If the first statement is not true and the second or later (else if) condition is true, execute the nested code block under else if. If not true (else) execute nested code block under else.
        iv. Else statement
            1. Execute if all other conditional statements above = false
        v. Switch statement
            1. Uses case and checks for parameter as a condition on each case. Default is the last option and it is similar to else.
            2. if there are a lot of possible outcomes, it is best practice to use a switch statement because it is easier less verbose. Being easier to read, it is easier to follow the logic, and thus reduce cognitive load of reading multiple conditions.

21. Looping Constructs
    a. Loops are used to continually execute repeated blocks of code until a certain condition is met/satisfied
    b. Loops are similar to conditionals; a condition must be satisfied in order for the code to execute.
    c. Different kind of loops
        i. For
            1. Set the value of the counter
            2. Specify the condition - i < 3;

3. Increment the counter - i++
4. For (var i = 1; i <= 3; i++) {

```
For (var i = 1; i <= 3; i++) {
        console.log(i)
}
```

5. Exit condition - works with the incrementor to prevent the loop from running forever by specifying at which value to terminate the loop.

ii. While

1. Similar to the for loop; however the start counter is set outside of the while loop and the incrementing is done inside the loop’s body.
2. Code that repeats as long as a specified condition is true
3. var i = 1;

```
var i = 1;
while (i < 6) {
   console.log(i)
   i = i +1;
}
console.log('Countdown finished!');
```

4.

iii. Nested

1. Putting a loop within another loop.
2. A practical way to think about it is creating a for loop that counts years
    a. Then nesting a for loop that counts months within the above parent loop
    b. Result would be something like:
        i. Year
            1. Month
            2. Month
            3. Month
        ii. Year
            1. Month
            2. Month
            3. Month
3. For (var year = 2023; year < 2025; year++) {

```
For (var year = 2023; year < 2025; year++) {
        console.log(year);
        For (var month = 6; month < 9; month++) {
                console.log(“-----”, month)
        }
}
```

d. Examples of loops being used:

i. If I'm coding an email client, I will get some structured data about the emails to be displayed in the inbox, then I'll use a loop to actually display it in a nicely-formatted way.

ii. If I'm coding an e-commerce site selling cars, I will get a source of nicely-structured data on each of the cars, then loop over that data to display it on the screen.
iii. If I'm coding a calendar online, I'll loop over the data contained in each of the days to display a nicely-formatted calendar.
iv. There are many, many other examples of using loops in code.
v. Using loops with data that is properly formatted for a given task is a crucial component of building software.
vi. In the lessons that follow, we'll learn about different ways of grouping related data and of displaying it on the screen using JavaScript.
vii. When combined with what you've already learned about loops, this gives you the skills to build various kinds of user interfaces where there is repetitive information.
viii. Some more specific examples include:
    1. looping over blog post titles in some structured data, and displaying each blog post title on a blog home page
    2. looping over social media posts in some structured data, and displaying each social media post based on some conditions
    3. looping over some structured data on clothing available for sale in an online clothing store, and displaying relevant data for each item of clothing

e. Additional resources:
    i. Comparison Operators
    ii. Truthy
    iii. Falsy
    iv. Conditional statements
    v. In JavaScript, there is also a shorthand version of writing a conditional statement, known as the conditional (ternary) operator: Conditional (ternary) operator

22. Functions
    a. One of the basic principles of programming can be summed up to DRY; Don't Repeat Yourself. Thanks to functions you can avoid repetition
    b. With functions, you can take several lines of code that performs a set of related actions and then group them together under a single label.
    c. Then when you need to run the code that you've saved, you can just invoke or call the function
        i. You can run the code as many times as you want
    d. These values are known as function parameters and are placed inside the function definition.
    e. The values passed to functions are arguments
    f. Arguments are parameters
    g. Example
        i. Function addTwoNums(a,b) {
                Var c = a+b;

            console.log(c);
        }
    ii. To call the function and pass parameters 2,4:
        1. addTwoNums(2,4);

23. Storing data in arrays
    a. A collection of pieces of data organized into an object
    b. Array literal syntax = []
        i. Var train1 = [“car1”, “car2”, “car3”];
            1. Indices = [0, 1, 2]
                a. 0 = car1
                b. 1 = car2
                c. 2 = ca3
            2. console.log(train1[0]);
                a. Result = car1
24. Objects
    a. A way to assemble a collective set of variables under one label
    b. Adding a property is known as extending the object
    c. Key value pairs of variables
        i. Variable name becomes property key
        ii. Variable value becomes property value
    d. Three common ways to build objects:
        i. One of the most common ways of building an object in JavaScript is using the object literal syntax: {}.
            1. Var user = {}; // create an object
        ii. Sometimes, an entire object can be immediately built, using the object literal syntax, by specifying the object's properties, delimited as key-value pairs
            1. //creating an object with properties and their values

```
        var assistantManager = {
        rangeTilesPerTurn: 3,
        socialSkills: 30,
        streetSmarts: 30,
        health: 40,
        specialAbility: "young and ambitious",
        greeting: "Let's make some money"
    }
```

        iii.
    e. Meant to mimic real-world objects
        i. Example 1 (dot-notation):
            1. dog.legs = 4;
            2. Dog.sound = “Woof”;
        ii. Example 2 (object literal): dog = {
            legs: 4,

Sound: “Woof”,

}

f. Dot notation
    i. Placing a dot operator between the object name and the property
        1. Example - human.hands = 2;
        2. Can access properties of the object like so:
            a. console.log(object.property); // displays property’s value in dev console

g. Object Literals
    i. To access the object, we can console log the entire object:
        1. console.log(object); // displays the object in dev console

h. Bracket Notation
    i. Example:
        1. Var house = {};
           House[“rooms”] = 4;
           House[‘color’] = “pink;
           house[”priceUSD”] = 12345;
        2. console.log(house);
            a. // {rooms: 4, color: 'pink', priceUSD: 12345}

i. Arrays are objects
    i. That means that arrays also have some built-in properties and methods.
    ii. One of the most commonly used built-in methods on arrays are the push() and the pop() methods.

j. Math object
    i. JavaScript has handy built-in objects. One of these popular built-in objects is the Math object.
    ii. Here are some of the built-in number constants that exist on the Math object:
        1. The PI number: Math.PI which is approximately 3.14159
        2. The Euler's constant: Math.E which is approximately 2.718
        3. The natural logarithm of 2: Math.LN2 which is approximately 0.693
    iii. Rounding methods
        1. Math.ceil() - rounds up to the closest integer
        2. Math.floor() - rounds down to the closest integer
        3. Math.round() - rounds up to the closest integer if the decimal is .5 or above; otherwise, rounds down to the closest integer
        4. Math.trunc() - trims the decimal, leaving only the integer
    iv. Arithmetic and calculus methods
        1. Here is a non-conclusive list of some common arithmetic and calculus methods that exist on the Math object:
            a. Math.pow(2,3) - calculates the number 2 to the power of 3, the result is 8
            b. Math.sqrt(16) - calculates the square root of 16, the result is 4

c. Math.cbrt(8) - finds the cube root of 8, the result is 2
d. Math.abs(-10) - returns the absolute value, the result is 10
e. Logarithmic methods: Math.log(), Math.log2(), Math.log10()
f. Return the minimum and maximum values of all the inputs: Math.min(9,8,7) returns 7, Math.max(9,8,7) returns 9.
g. Trigonometric methods: Math.sin(), Math.cos(), Math.tan(), etc.

v. Random Method
   1. A part of the Math object that can generate a number between 0 and 0.99

vi. Ceil Method
   1. A part of the Math object that rounds a decimal up to the nearest integer

vii. Combining above methods
   1. build some code that will combine the two to return a random integer between 0 and 10. The first step is to create a variable and assign it the value of the math dot random method multiplied by 10. I need to pass this variable through the ceil method to ensure that an integer instead of a decimal value is created. So, I declare the variable rounded and assign it the value of the math dot ceil method containing the decimal variable inside the parentheses. Finally, I use the console dot log method to output the variable rounded. If I run the code, notice that a value between 0 and 10 is output to the console.

k. Iterable
   i. An iterable is any datatype that can be iterated over using a “for of” loop.
   ii. Arrays and Strings are Iterables
   iii. In the world of JavaScript, it can often be said that strings behave like arrays. Strings are array-like.
   iv. You can run a for loop over an array of letters or a string and get the same result

l. String cheat sheet
   i. String Functions
      1. Length of a string - “string.length”
      2. Read individual character in a string - “string.charAt(0);”
      3. The concat() method joins two strings - "Wo".concat("rl").concat("d"); // 'World'
      4. The indexOf returns the location of the first position that matches a character:
      5. The lastIndexOf finds the last match, otherwise it works the same as indexOf.
      6. The split method chops up the string into an array of sub-strings - "ho-ho-ho".split("-"); // ['ho', 'ho', 'ho']
      7. Upper and lower case:

            a. greet.toUpperCase(); // "HELLO, "
            b. greet.toLowerCase(); // "hello, "
    m. Object Methods
        i. The same way you can add key/value pairs of data to an object, you can make one of those values a function.
        ii. Remember, value of an key in an object is considered a property
        iii. When a function is a property of an object, it is then referred to as a method
            1. This is a function that can be accessed only through the JavaScript object that it is a member of.
    n. Additional resources
        i. JavaScript Functions
        ii. JavaScript Object Basics
        iii. typeof operator in JavaScript
        iv. Arrays are "list-like objects"
    o.

25. typeOf
    a. JavaScript operator that evaluates a parameter and returns the data type as a string
26. Bugs and Errors
    a. Bug
        i. A bug causes a program to run in an unintended way
    b. Error
        i. An error causes a program to stop running
        ii. Some of the most common error types:
            1. Syntax Error
            2. Type Error
            3. Reference Error
27. Try/Catch Blocks
    a. Try
        i. If a piece of code throws an error, it can get wrapped inside a try block.
    b. Catch
        i. Then you can catch the error with the catch block, and use it to do something
        ii. The catch block accepts something called an error which is an object
    c. Throw
        i. Using the throw keyword, you can force an error to be thrown from the try block to the catch block
        ii. It is important to remember that you can use the throw keyword outside the the try block, but it will not be possible to catch it.
    d. Structure
        i. The try block starts with the try keyword
            1. Inside of the curly braces, you place the code that you think will throw an error

        ii. Next is the catch block which catches the error that the try block produces. It begins with the catch keyword and in parenthesis, you have a built-in error object that you can name whatever you like
            1. Inside the curly braces, you place the code you would like to execute

28. Syntax, logical, and runtime errors
    a. Most common errors in JavaScript:
        i. ReferenceError
            1. A ReferenceError gets thrown when, for example, one tries to use variables that haven't been declared anywhere
        ii. SyyntaxError
            1. Any kind of invalid JavaScript code will cause a SyntaxError
        iii. TypeError
            1. A TypeError is thrown when, for example, trying to run a method on a non-supported data type
        iv. RangeError
            1. A RangeError is thrown when we're giving a value to a function, but that value is out of the allowed range of acceptable input values
    b. Other errors:
        i. AggregateError
        ii. Error
        iii. InternalError
        iv. URIError
29. Types of empty values
    a. Undefined
        i. In JS, there may be times you're building something that hasn't been clearly defined yet and so you can't assign a value to it
        ii. fortunately, there is a way to store it so that you can assign it later using the undefined data type.
        iii. The undefined data type can only hold one value, undefined
    b. Null
        i. Intentional absence of any object value
        ii. It is also the return value of some built-in JavaScript methods
    c. Empty strings
        i. Var name1 = ' ';
        ii. Var name2 = " ";