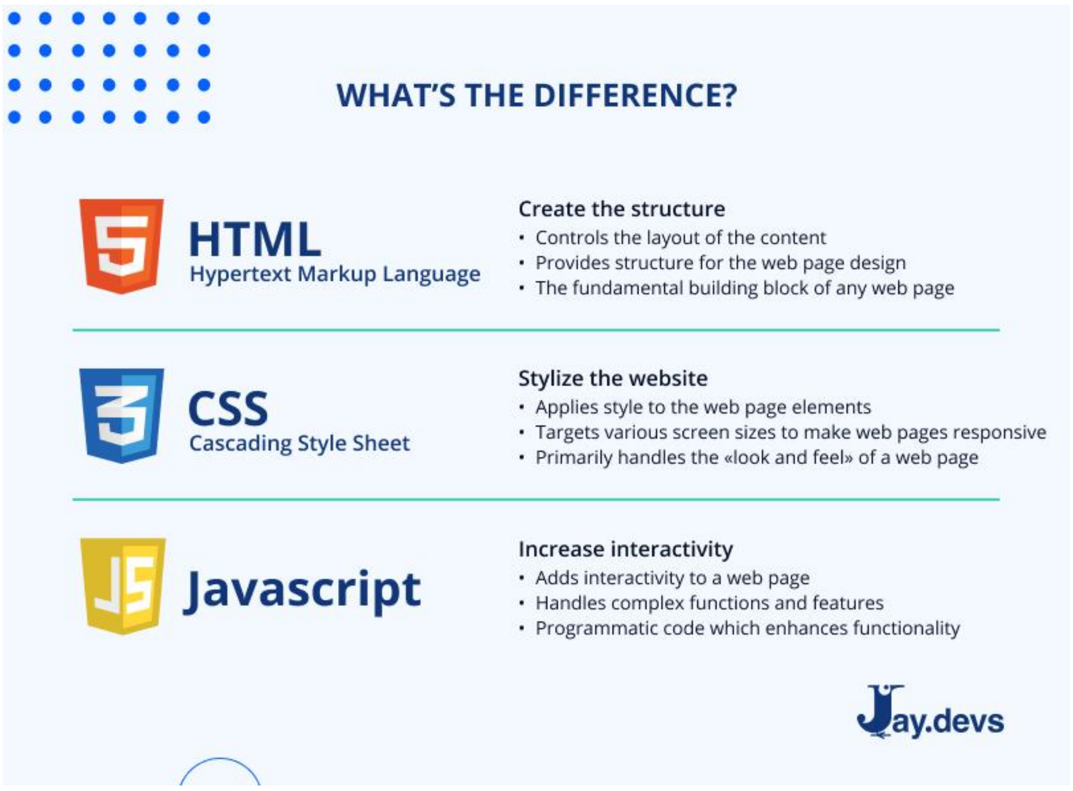
Week 5: Web Development

1. Revision and Quiz



WHAT'S THE DIFFERENCE?

HTML
Hypertext Markup Language

Create the structure
- Controls the layout of the content
- Provides structure for the web page design
- The fundamental building block of any web page

CSS
Cascading Style Sheet

Stylize the website
- Applies style to the web page elements
- Targets various screen sizes to make web pages responsive
- Primarily handles the «look and feel» of a web page

Javascript

Increase interactivity
- Adds interactivity to a web page
- Handles complex functions and features
- Programmatic code which enhances functionality

Jay.devs

2. Arrays

- Arrays in JavaScript store multiple values in a single variable.
- They can hold different data types and change size dynamically.
- Accessed with square brackets, starting from index 0.
- Utilize a `length` property to determine the number of elements.

- Arrays support various operations like adding/removing elements and iteration.

Methods

push(): Adds one or more elements to the end of an array and returns the new length of the array.

pop(): Removes the last element from an array and returns that element.

shift(): Removes the first element from an array and returns that element.

unshift(): Adds one or more elements to the beginning of an array and returns the new length of the array.

concat(): Joins two or more arrays and returns a new array.

slice(): Extracts a section of an array and returns a new array.

splice(): Adds or removes elements from an array.

indexOf(): Returns the first index at which a given element can be found in the array, or -1 if it is not present.

lastIndexOf(): Returns the last index at which a given element can be found in the array, or -1 if it is not present.

includes(): Determines whether an array includes a certain element, returning true or false.

3. Objects

- **Pairs:** Key-value pairs store data in objects.

- **Properties:** Keys hold various data types.

- **Access:** Use dot or bracket notation to access properties.

 Properties can be accessed using dot (object.property) or bracket notation (object['property']).

- **Literal:** Objects created with curly braces and key-value pairs.

- **Methods:** Functions stored as object values.

- **Dynamic:** Objects can be altered after creation.

- **Inheritance:** Objects can inherit properties.

- **JSON:** Data interchange format akin to object literal syntax.

Objects organize data efficiently in JavaScript, offering flexibility and adaptability in programming.

Methods

Object.keys(obj): Returns an array of a given object's own enumerable property names.

Object.values(obj): Returns an array of a given object's own enumerable property values.

Object.entries(obj): Returns an array of a given object's own enumerable string-keyed property [key, value] pairs.

Object.assign(target, ...sources): Copies the values of all enumerable own properties from one or more source objects to a target object.

Object.getOwnPropertyNames(obj): Returns an array of all properties (enumerable or not) found directly upon a given object.

Object.freeze(obj): Freezes an object: other code can't delete or change any properties.

Object.seal(obj): Prevents new properties from being added to an object and marks all existing properties as non-configurable.

4. Functions

In JavaScript, functions are a fundamental concept used for defining reusable blocks of code. Here's a breakdown of different types of functions:

1. **Named Functions:**
   - Defined using the `function` keyword followed by the function name.
   - Can be declared before or after they are called.
   - Example:
   ```javascript
   function greet(name) {
       return 'Hello, ' + name + '!';
   }
   ```

2. **Arrow Functions (ES6):**
   - Introduced in ES6, providing a shorter syntax compared to named functions.
   - Does not have its own `this` or `arguments` binding.
   - Example:
   ```javascript
   const greet = (name) => {
       return 'Hello, ' + name + '!';
   };
   ```

3. **Anonymous Functions:**
   - Functions without a name, often assigned to variables or used as arguments to other functions.
   - Example:
   ```javascript
   const greet = function(name) {
       return 'Hello, ' + name + '!';
   };
   ```

4. **Immediately Invoked Function Expressions (IIFE):**
   - Functions that are executed immediately after they are created.
   - Enclosed within parentheses to avoid polluting the global scope.
   - Example:
   ```javascript
   (function() {
       console.log('I am immediately invoked.');
   })();
   ```

5. **Higher-Order Functions:**
   - Functions that can take other functions as arguments or return functions.
   - Commonly used for functional programming paradigms.
   - Example:
   ```javascript
   const numbers = [1, 2, 3, 4, 5];
   const doubled = numbers.map(function(num) {
       return num * 2;
   });
   ```

5. String Methods

length: Returns the length of a string.

charAt(index): Returns the character at the specified index.

charCodeAt(index): Returns the Unicode value of the character at the specified index.

concat(str1, str2, ...): Combines two or more strings and returns a new string.

indexOf(searchValue, [fromIndex]): Returns the index of the first occurrence of a specified value in a string, or -1 if not found.

lastIndexOf(searchValue, [fromIndex]): Returns the index of the last occurrence of a specified value in a string, or -1 if not found.

slice(startIndex, [endIndex]): Extracts a section of a string and returns a new string.

substring(startIndex, [endIndex]): Similar to slice(), but does not accept negative indices.

substr(startIndex, [length]): Extracts a specified number of characters from a string, starting at the specified index.

toUpperCase(): Converts a string to uppercase.

toLowerCase(): Converts a string to lowercase.

replace(searchValue, newValue): Replaces a specified value with another value in a string.

trim(): Removes whitespace from both ends of a string.

startsWith(searchString, [position]): Checks if a string starts with the specified value.

endsWith(searchString, [position]): Checks if a string ends with the specified value.

includes(searchString, [position]): Checks if a string contains the specified value.

6. Higher-Order Functions (HOFs):

- Functions that can take other functions as arguments or return functions.

- Taking Functions as Arguments:

 - Example: **map, filter, reduce**.

- HOFs can produce functions as return values, enabling composition and currying.

- HOFs promote code reuse and abstraction, leading to cleaner and more maintainable code.

- They allow for the creation of generic, reusable functions that can be applied to various scenarios.

lets practice Map filter,

Lab Tasks

Easy: Array Practice
Write a function called reverseArray that takes an array as input and returns a new array with the elements reversed. For example, reverseArray([1, 2, 3]) should return [3, 2, 1].

Medium: Object Practice
Create an object named car with properties brand, model, and year. Write a function called carInfo that takes the car object as input and returns a string with the car's information. For example, if car has the values { brand: 'Toyota', model: 'Camry', year: 2022 }, the function should return 'The Toyota Camry was manufactured in 2022.'.

Hard: Function Practice
Write a higher-order function called applyFunction that takes two arguments: an array of numbers and a function. The function should apply the given function to each element of the array and return a new array with the results. For example, if the array is [1, 2, 3] and the function is (x) => x * 2, the result should be [2, 4, 6].