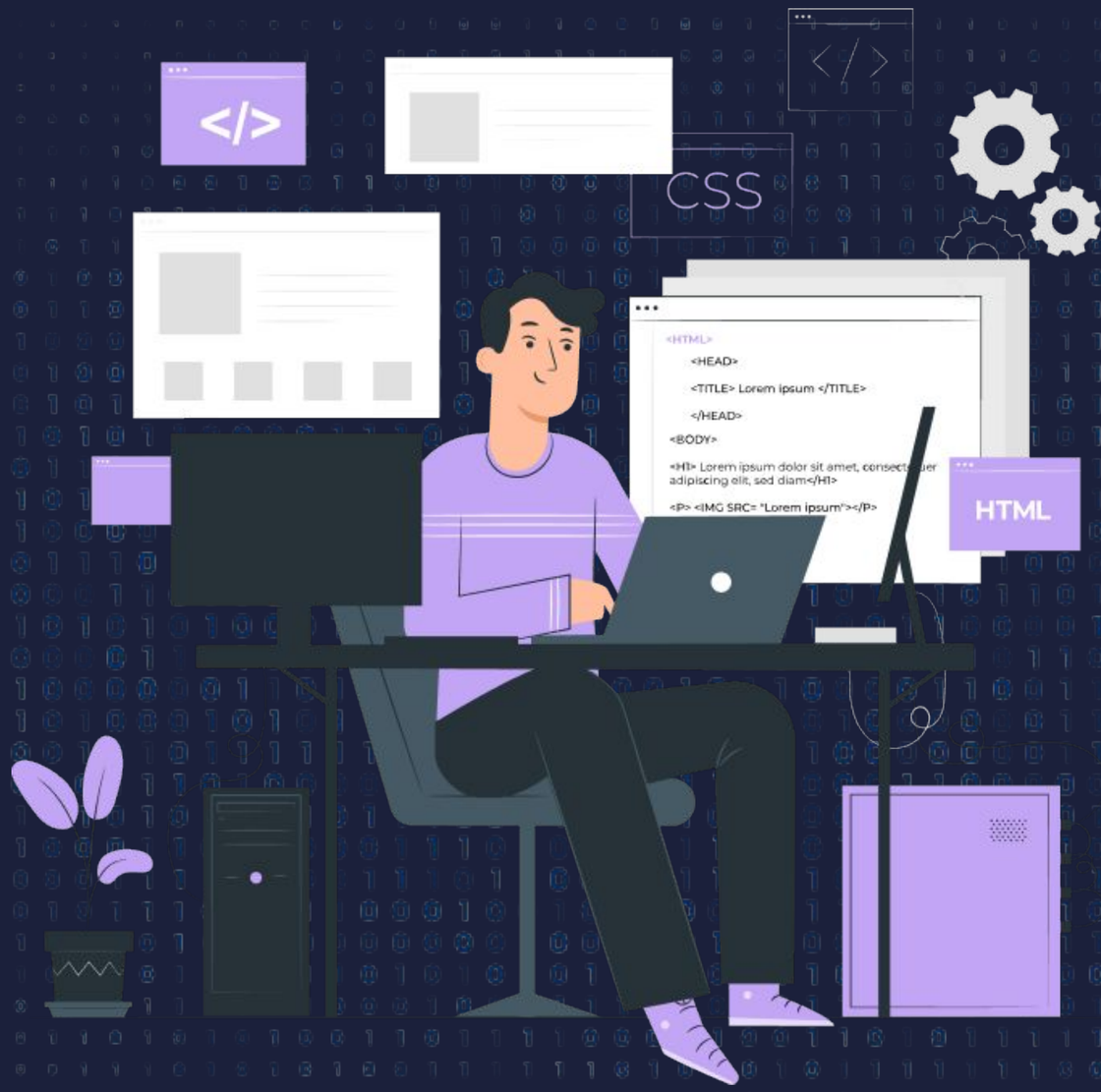
PW SKILLS

Non-primitive data type, Value vs Reference Types



Topics Covered

- What are Non-primitive data types, explain with examples
- Value vs Reference Types

What are Non-primitive data types, explain with examples

- In JavaScript, Non-primitive data types are data types that are not directly defined by the programming language itself.
- They are commonly referred to as reference types because they are assigned and passed by reference rather than by value.
- The non-primitive data types include likes -
    - Array
    - Object
    - Set
    - Map

Array

- An array in JavaScript is a data structure that allows you to store multiple values in a single variable.
- It is an ordered collection of elements, where each element can be of any data type, including numbers, strings, objects, or even other arrays
- Arrays are commonly used to store and manipulate lists of related data

```
// Example An Array of fruits
const fruits = ["Mango", "Orange", "Graphes"]
console.log(fruits)
// output - ["Mango", "Orange", "Graphes"]
```

Object

- An object is a data structure in JavaScript that allows you to store and organize key-value pairs
- It is a collection of properties, where each property consists of a key and its associated value.
- Objects are widely used to represent complex data structures and can contain properties of different data types.

```
const data = {
    name: "John Doe",
    userName: "john",
    email: "alex@gmail.com
}
```

Set

A set is a built-in JavaScript object that represents a collection of unique values.
It allows you to store distinct values of any type, eliminating duplicates.
Sets provide methods for adding, removing, and checking the presence of values, as well as performing set operations like union, intersection, and difference.

```
const dummy = new Set();
dummy.add("one");
dummy.add("two");
dummy.add("three");
console.log(dummy);
//output - Set(3) { 'one', 'two',
'three' }
```

Map

- A map is another built-in JavaScript object that represents a collection of key-value pairs, similar to objects.
- However, maps can have keys of any data type, including objects or functions.
- Maps maintain the insertion order of elements and provide methods to add, remove, and retrieve values based on keys

```
const map = new Map();

map.set("White", "#ffffff");
map.set("black", "#000000");

console.log(map);
// output - Map(2) { 'White' => '#ffffff', 'black' => '#000000' }
```

Value vs Reference Types

Value / Primitive types	Reference Types
Value types are also known as primitive data type	Reference types are Objects
It includes numbers, strings, boolean, null and undefined	It includes mainly objects like array, custom objects, set, map
A copy of the actual value is created and stored independently	Instead of the actual value, a reference (memory address) to the object is stored in the variable.

The main difference between value and reference types is how they store the data or value.

Example of Value types -

```
let name = "John Doe";
let temp = name; // b gets the value 5
temp = "Tom cruise";
console.log(name); // Output: 5 (original value unchanged)
console.log(temp); // Output: 10 (modified value)
```

Example Reference types -

```
let obj1 = { name: "John" };
let obj2 = obj1; // obj2 references the same object as obj1
obj2.name = "Tom ";
console.log(obj1.name); // Output: 'Alice' (modified value)
console.log(obj2.name); // Output: 'Tom' (same object)
```


SKILLS
THANK YOU