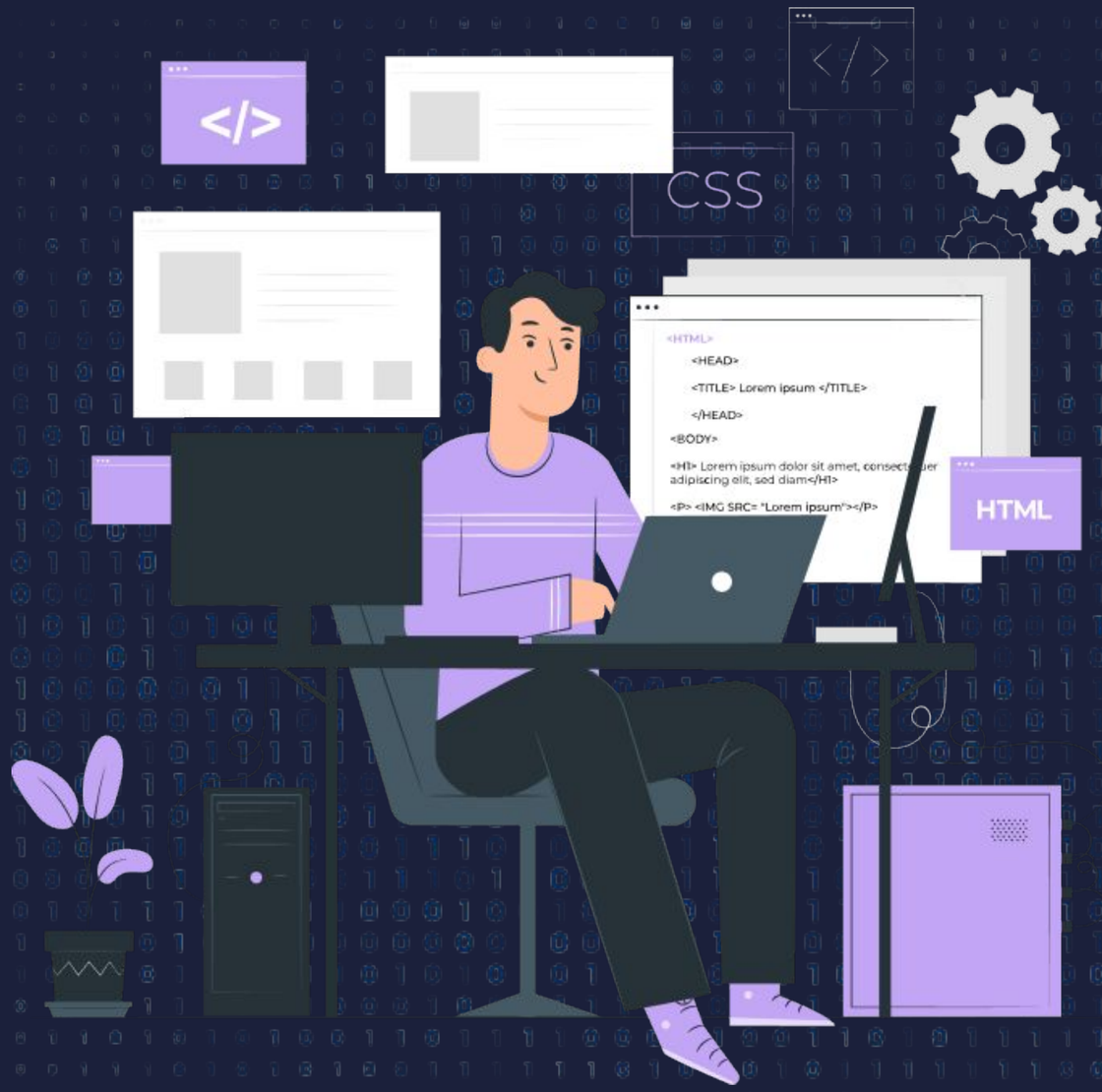
PW SKILLS

What is an Array, Index in Array and Accessing from Array



Lecture CheckList

- What is An Array and Example
- Why do we need Arrays in Javascript?
- Declaration of Array
- Array index and storing
- Accessing elements in an array
- Change values in an array
- Delete keyword in arrays & its problem
- Iterating of Array items

What is An Array and Example ?

- An array in JavaScript is a data structure that stores an ordered list of elements.
- It can hold elements of any data type, including numbers, strings, objects, and even other arrays
- Arrays are a type of object in JavaScript and have a number of built-in methods for adding, removing, and manipulating elements.

Example

```
let player= ["Sunday","Monday", "Tuesday", "Wednesday"];
let numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0];
let array = ["Hello", 20, true]
```

Array property - **array.length**, it is the array property which return the number of item in the array.

Example

```
const language = ["javascript", "python", "java", "golang"]
const len = language.length
console.log("Array length =", len)
// Array length = 4
```

Why do we need Arrays in Javascript?

- Grouping Related Data
- Storing large amounts of data
- Improving performance
- Ease of use
- Better readability

Declaration of Array

- **Using square brackets [ ]**

```
let fruits = ["Apple", "Mango", "Banana", "Kiwi"];
```

- **Using the Array constructor**

```
let numbers = new Array(1, 2, 3, 4, 5);
let emptyArray = new Array(5); // Creates an array with 5 empty elements.
```

It is generally not recommended to create an array using the Array constructor in JavaScript, due to the following reasons -

- Inconsistent behaviour
- Simplicity and readability
- Potential security risks

- **Using an array literal**

```
let colors = Array("Black", "Red", "White", "Blue");
```

Array index and Storing

```
let players= [ ]; // create an empty array
fruits[0] = "Virat"; // store "Virat" at index 0
fruits[1] = "Rohit"; // store "Rohit" at index 1
fruits[2] = "Suryakumar"; // store "Suryakumar" at index 2
```

Accessing elements in an array

```
let players = ["Virat", "Rohit", "Suryakumar"];
// To access the first element "Virat", you would use the following code.

players[0]; // returns "Virat"
players[1]; //returns "Rohit"
players[2]; // returns "Suryakumar"
players[3]; // returns undefined
players[-1]; // returns undefined
players[-3]; // returns undefined
```

Changing values in an array

```
const hobbies = ["Coding", "Learning", "Watching movies"]

hobbies[0] = "building Project" // change to building Project
hobbies[2] = "Football"  // change to Football

// output -
[ 'building Project', 'Learning', 'Football' ]
```

Delete keyword in arrays & its problem

In JavaScript, the delete keyword is used to delete a property from an object, including an array element. However, using delete to remove elements from an array can lead to unexpected behaviour and introduce problems.

Here are some impacts and problems associated with using the delete keyword on arrays:

- Sparse Arrays
- Array length
- Performance Impact
- Array Integrity

Example - delete in an array -

- const array = [1, 2, 3, 4];
- console.log(array.length); // output - 4
- delete array[2];
- console.log(array); // output - [1, 2, <1 empty item>, 4]
- console.log(array[2]); // output - undefined
- console.log(array.length); // output - 4 same length even after delete

Iterating of Array items

In JavaScript, An array items can be iterated or looped in the following ways -

- for loop
- for....of loop
- while loop

for loop

```
const array = [1, 2, 3];

for (let index = 0; index < array.length; index++) {
  const element = array[index];
  console.log(element);
}
// - output -
1
2
3
```

for... of loop

```
const array = [1, 2, 3];
for (const iterator of array) {
  console.log(iterator);
}
// output ---
1
2
3
```

While loop

```
const array = [1, 2, 4, 3];
let i = 0;
while (array.length > i) {
  console.log(array[i]);
  i++;
}
// output -
1
2
3
4
```


SKILLS
THANK YOU