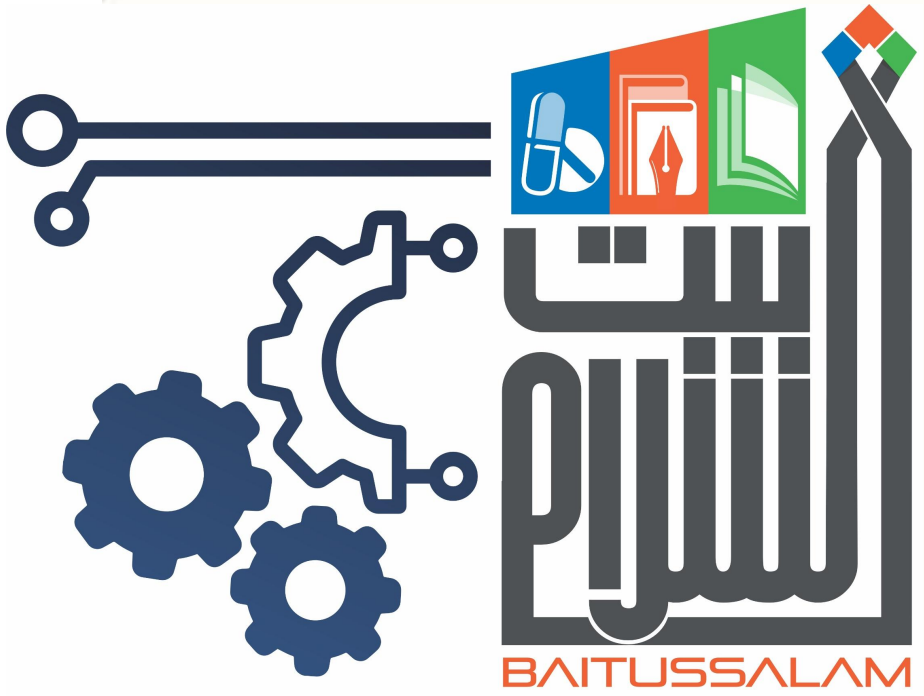

BAITUSSALAM
—TECH PARK—

PSDC-201

BAITUSSALAM
—TECH PARK—

JavaScript Arrays and Objects

JavaScript Data Types

JavaScript data types are divided into primitive and non-primitive types.

Primitive Data Types

They can hold a single simple value. String, Number, BigInt, Boolean, undefined, null, and Symbol are primitive data types.

Non-Primitive Data Types

They can hold multiple values. Objects are non-primitive data types.

Arrays

1. **Collection**: An array is like a box that can hold many items.
2. **Order**: Arrays allow us to organize related data by grouping them within a single variable.
3. **Index**: Each item in the array can be accessed using a number, called an index, that shows its position. The first item is at position 0, the second at position 1, and so on.

How to Create An Array

To create an array in JavaScript, we use square brackets [] and put the items inside, separated by commas.

Here’s an example:

```
let fruits = ['Apple', 'Banana', 'Cherry'];
```

- **fruits is the name of the array.**
- **'Apple', 'Banana', and 'Cherry' are the items in the array.**

Access Array Elements

Javascript arrays are zero-indexed

You can get an item from the array by referring to its index (position).
Remember, the first item is at index 0, the second item is at index 1, and so on.

```
3  console.log(fruits[0]); // Outputs: Apple
4  console.log(fruits[1]); // Outputs: Banana
5  console.log(fruits[2]); // Outputs: Cherry
```

Add and Remove Array Element

```
// Add an item to the end using push():
fruits.push("Date");
console.log(fruits); // Outputs: ['Apple', 'Banana', 'Cherry', 'Date']

// Add an element at the beginning of array using unshift():
fruits.unshift("Strawberry");
console.log(fruits); // Outputs: ['Strawberry', 'Apple', 'Banana', 'Cherry', 'Date']

// ************************************************************

// Remove the last item using pop():
let lastFruit = fruits.pop();
console.log(lastFruit); // Outputs: Date
console.log(fruits); // Outputs: ['Strawberry', 'Apple', 'Banana', 'Cherry']

// Remove the first item using shift():
console.log(fruits); // Outputs: ['Strawberry', 'Apple', 'Banana', 'Cherry']
```

Change / Modify Array Element

We can add or change elements by accessing the index value.

For example,

```
25
26  let fruits = ['Apple', 'Banana', 'Cherry', 'Date'];
27
28  // Modify elements
29  fruits[0] = 'Apricot';
30  fruits[2] = 'Cantaloupe';
31
32  console.log(fruits); // Outputs: ['Apricot', 'Banana', 'Cantaloupe', 'Date']
```

Common Array Methods

1. **array.concat:** Combines two or more arrays into one.
2. **array.includes:** Checks if an array contains a specific item.
3. **Array.isArray:** Checks if something is an array.

app.js > ...

```
// array.concat() // merge two arrays
let set1 = ['red', 'green']
let set2 = ['blue', 'pink']

let combinedSet = set1.concat(set2)
console.log(combinedSet) // Outputs: ['red', 'green', 'blue', 'pink']

// Check if an Array Contains an Item: array.includes
let hasRedColor = toys.includes('red')

console.log(hasRedColor) // Outputs: true

// Array.isArray: Checks if something is an array
let toys = ['Toy Car', 'Toy Train']
let notToys = 'Toy Car'

let isArray1 = Array.isArray(toys)
let isArray2 = Array.isArray(notToys)

console.log(isArray1) // Outputs: true
console.log(isArray2) // Outputs: false

```

Slice vs Splice

slice() is a method that creates a new array containing elements from the original array. It doesn't modify the original array. We specify the starting and ending indexes, and it returns the elements within that range as a new array.

Use Cases:

- When you need to extract a portion of an array without modifying the original array.
- Useful for operations where you need to work with a subset of the original array.

```
1   const fruits = ['apple', 'banana', 'cherry', 'date', 'fig']
2
3   // Using slice to create a new array containing elements
4   // from index 1 to index 3 (exclusive)
5   const slicedFruits = fruits.slice(1, 4)
6
7   console.log(slicedFruits)
8   // Output: ['banana', 'cherry', 'date']
9
10  console.log(fruits)
11  // Output: ['apple', 'banana', 'cherry', 'date', 'fig']
```

Slice vs Splice

1. splice() is a method that changes the contents of an array by removing or replacing existing elements and/or adding new elements.
2. It directly modifies the original array.
3. We specify the starting index, number of elements to remove, and optional new elements to add.
4. It returns an array containing the removed elements.

```
13  const months = ['Jan', 'Feb', 'March', 'April', 'May']
14
15  // Using splice to remove elements starting
16  // from index 2 and add 'June' and 'July' in their place
17  const removedMonths = months.splice(2, 2, 'June', 'July')
18
19  console.log(removedMonths)
20  // Output: ['March', 'April']
21
22  console.log(months)
23  // Output: ['Jan', 'Feb', 'June', 'July', 'May']
```

Use Cases:

- When we need to remove elements from an array or replace them with new elements.
- Useful for directly modifying the original array, such as when performing operations like adding, removing, or replacing elements.

Loop Over the Array

app.js

```
let colors = ["red", "green", "blue", "orange"];

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

```

1. Set up a for loop with a counter starting at 0.
2. Loop condition: Run the loop as long as the counter is less than the array length.
3. Access each array item inside the loop using the counter.
4. Increment the counter after each loop iteration.

JavaScript Array Exercises

1. Sum of all numbers in array

2. Find the largest number in an array

3. Remove all duplicate values from array and print new array with all the unique value

Most important Array Concept

app.js > ...

```
1  let colorSetOne = ['red', 'blue']
2
3  let colorSetTwo = colorSetOne
4
5  colorSetOne.push('green')
6
7  console.log('colorSetOne', colorSetOne)
8  console.log('colorSetTwo', colorSetTwo)
```

A **shallow copy** means that it creates a new array, but the elements within that array are still references to the same objects as the original array.

In JavaScript, when we assign an array (or any object) to another variable, we're not creating a new copy of that array; instead, we're creating a **reference to the same array** in memory.

So, when we modify the array through one reference, the change is reflected in all other references pointing to that array.

Its Time to Learn Objects

Wait! We should these tricks first

Extras #1: How to swap two variable values in JS

Extras #2: Simple trick to convert string to number

Extras #3: String Concatenation

JavaScript Objects

- Objects are collections of properties
- Properties are key-value pairs
- To access data we use keys

Create New Object

Object Constructor

const obj = new Object({name: ‘Ali’})

Object initializer / Object literal syntax

const obj = {name: ‘Ali’}

Remember

- *All keys in object are converted to string except symbol*
- *What if there are 2 keys with same name?*

Object Exercise #1

Define a new variable called `product`. It should be an object literal with the following properties:

- `name` - set to the string `"HP Elitebook Sleeve"`
- `inStock` - set to the boolean `true`
- `price` - set to the number `1000`
- `totalUnits` - set to the number `7`
- `colors` - set to an array of at least three strings like `["black", "white", "gray"]`

Way to access object properties

1. *Using . Dot notation*
2. *Using [] square bracket*

What's the difference between?

- *How to access nested object?*
- *How to access array inside object?*
- *How to* Accessing Object Properties Dynamically?

app.js > ...

```
1  // Accessing Object Properties using Dot Notation
2
3  let person = {
4    name: 'John',
5    age: 30,
6    city: 'New York'
7  };
8
9  console.log(person.name); // Outputs: John
10 console.log(person.age); // Outputs: 30
11 console.log(person.city); // Outputs: New York
12
13 // Example 2: Accessing Object Properties using Bracket Notation
14
15 console.log(person['name']); // Outputs: John
16 console.log(person['age']); // Outputs: 30
17 console.log(person['city']); // Outputs: New York
```

Object Exercise #2

JavaScript Copy Caption

```
const restaurant = {
    name: 'Ichiran Ramen',
    address: `${Math.floor(Math.random() * 100) + 1} Johnson Ave`,
    city: 'Brooklyn',
    state: 'NY',
    zipcode: '11206',
}
```

- **Your task to is to create a variable named** `fullAddress` **that points to a string** using the information from `restaurant`.
- `fullAddress` should point to a string that includes the address, city, state, and zipcode from the restaurant object. Make sure to add any necessary commas or spaces between the values as seen in the exact expected output format shown below.
- Expected output: `"64 Johnson Ave, Brooklyn, NY 11206"`

Modify, Add and Delete JS object properties

app.js > ...

```js
let person = {
  name: 'John',
  age: 30,
  city: 'New York'
};

// Modifying Object

// Modify the value of the 'age' property
person.age = 35;

// Adding new property to Object
// Add a new 'city' property
person.city = 'New York';

// Add a new 'job' property
person.job = 'Software Engineer';

// Delete the object property
// Delete the 'city' property
delete person.city;

console.log(person);
// Outputs: { name: 'John', age: 30 }
```

Object Exercise #3

1. **Task**: Create an object named `student` with the following properties:
   - `name` (a string)
   - `age` (a number)
   - `subjects` (an array of strings)
   - `isEnrolled` (a boolean)
2. **Instructions**:
   - Log each property of the `student` object to the console.
   - Add a new property `grade` with a value of "A" to the `student` object.
   - Change the `isEnrolled` property to `false`.

Array of Objects

1. **Task**: Create an array named `library` that contains three objects. Each object should represent a book and have the following properties:
   - `title` (a string)
   - `author` (a string)
   - `yearPublished` (a number)
2. **Instructions**:
   - Add a new book named "System Design" object to the `library` array.
   - Loop over the library and search for the new book you just added and console all its properties
   - Remove a book by title
   - update the book years publish date

Let’s build a ToDo app

Rules

"quit" — quit application

"list" — show all todos

"new" — will ask for a new todo

"delete" — will ask for deleting a specific todo

Assignment

Handle string/unknown input while deleting todos.

If the user types any invalid number, it will console "not a valid number."

The End

BAITUSSALAM
—TECH PARK—