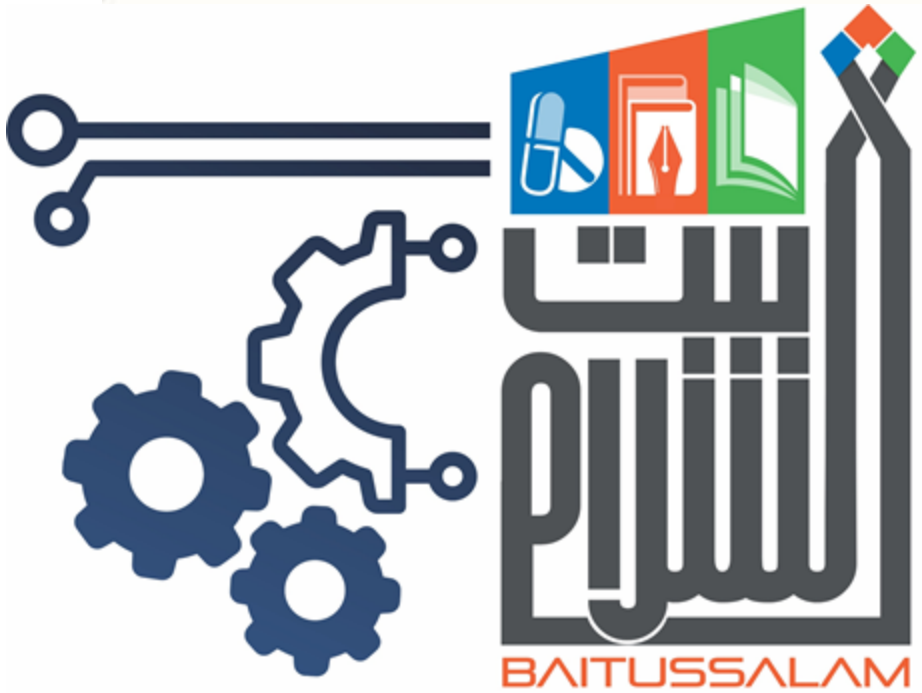

BAITUSSALAM
—TECH PARK—

BAITUSSALAM
—TECH PARK—

PSDC-201

Introduction to programming with javascript and Typescript

JavaScript Fundamentals

BAITUSSALAM
—TECH PARK—

Agenda

- Prerequisite
- Introduction to JavaScript
- Role of JavaScript in Web Development
- JavaScript syntax and data types
- Manipulating String/Numbers
- Arrays and it’s methods.
- Objects

Prerequisite

- HTML
- CSS
- Basic Programming Concepts

What is JavaScript ?

- JavaScript is a high-level, interpreted programming language.
- It's commonly used for creating interactive effects within web browsers.
- It is also used on servers to create scalable applications.

Interpreting JavaScript

JavaScript: A programming language used to add interactivity and dynamic behavior to web pages.

Browser interprets JavaScript code to handle user interactions, animations, and other dynamic features.

Examples: Form validation, image sliders, interactive games.

```
// Sample array
const colors = ["red", "green", "blue", "yellow"];

// String to check for
const searchString = "blue";

// Using includes method
if (colors.includes(searchString)) {
  console.log(`${searchString} is in the array.`);
} else {
  console.log(`${searchString} is not in the array.`);
}
```

Role of JavaScript in Web Development

Enhanced User Experience:

JavaScript allows for dynamic content updates without reloading the entire webpage.

Interactive elements like forms, animations, and pop-ups enhance user engagement.

Client-Side Scripting:

JavaScript runs on the client side, meaning it's executed by the user's browser.

This reduces server load and improves website performance by offloading tasks to the client's machine.

Role of JavaScript in Web Development

DOM Manipulation:

The Document Object Model (DOM) represents the structure of a webpage.

JavaScript enables developers to manipulate DOM elements, altering their content, style, and behavior.

Asynchronous Programming:

JavaScript supports asynchronous operations, allowing tasks to execute independently without blocking other processes.

This is crucial for handling events, fetching data from servers, and ensuring smooth user interactions.

Role of JavaScript in Web Development

Cross-Browser Compatibility:

JavaScript code is supported by all major web browsers, including Chrome, Firefox, Safari, and Edge.

This ensures consistency in the behavior and appearance of web applications across different platforms.

Rich Ecosystem of Libraries and Frameworks:

JavaScript has a vast ecosystem of libraries (e.g., jQuery, React) and frameworks (e.g., Angular, Vue.js) that streamline development.

These tools provide pre-built components, modules, and utilities to expedite the creation of complex web applications.

Javascript Syntax and Data Types

There are two different types of data types in javascript:

- **Primitive data type**

  Primitive data types in JavaScript are like different kinds of puzzle pieces that computers use to understand and handle information, such as numbers, words, yes or no questions, empty spaces, and things that haven't been figured out yet.

- **Non-primitive data type**

  Non-primitive data types in JavaScript are like special helpers that make computers do tricky things, like sorting and following instructions step by step.

Variable And Its Scope

In JavaScript, **var**, **let**, and **const** are used to declare variables, but they have different behaviors and scoping rules.

var v = 10;

let a = 15;

const = 50;

Javascript Syntax and Data Types

- **Primitive Data Type**
  - *String*
  - *Boolean*
  - *Number*
  - *Undefined*
  - *Null*

- **Non-Primitive Data Type**
  - *Object*
  - *Array*
  - *Function*

Primitive Data Types

1. **Numbers**:
Numbers are like the digits you use for counting or doing math. They can be big or small, positive or negative, and even have decimal points. For example, if you want to know how many cookies you have, you'd use a number data type.

2. **Strings**:
Strings are like words or sentences. They're made up of letters, numbers, or symbols. So, if you want to write your name or a message to your friend, you'd use a string data type.

3. **Booleans**:
Booleans are like yes or no questions. They can only have two values: true or false. For instance, if you want to know if it's raining outside, you'd use a boolean data type. If it's raining, it's true; if it's not raining, it's false.

4. **Null**:
Null means nothing or empty. It's like having an empty box. There's nothing inside it. Sometimes, programmers use null when they want to say that something doesn't exist or has no value.

5. **Undefined**:
Undefined means something hasn't been defined or set yet. It's like having an empty space where something should be, but it hasn't been put there yet. It's used when we forget to give something a value or when it's not clear what the value should be.

Non Primitive Data Types

1. **Objects**:
Objects are like treasure chests that can hold many different types of things inside them. They can store numbers, words, and even other objects! For example, an object could represent your toy collection, with each toy having its own name and color.

2. **Arrays**:
Arrays are like special containers that can hold multiple items in a list. They're like shelves with lots of compartments where you can organize things. For instance, an array could store your favorite snacks in order, from first to last.

3. **Functions**:
Functions are like magic spells that perform specific tasks when called upon. They're like recipes that tell the computer what steps to follow. For example, a function could be a recipe for making your favorite sandwich, with instructions on how to put it together.

Javascript Variables

Number

let length = 10;

let width = 20;

String

let firstName = “ahmed”;

Let lastName = “ali”;

Boolean

let x = true;

Let y = false;

Object

const person = { “firsName” : “Ali”, “lastName” : “Hasan”} ;

Array Object

const colors = [“Red”, “Blue”, “Green”, “Yellow”];

Date Object

const date = new Date(“2024-03-21”);

Arithmatic Operators

Assignment Operator

- Is used to assign values like **let x = 10**;

Addition/Subtraction Operator

- Is used to add values like **let x = 10; y = x + 40; z = x – 5;**

Addition/Subtraction Assignment Operator

- Is used to assign with addition into the same variable like **let x = 10; x += 5; x -=3;**

Math Operators

Addition (+): Adds two numbers together.

Example: 3 + 5 equals 8.

Subtraction (-): Subtracts one number from another.

Example: 7 - 4 equals 3.

Multiplication (*): Multiplies two numbers.

Example: 2 * 6 equals 12.

Division (/): Divides one number by another.

Example: 10 / 2 equals 5.

Modulus (%): Returns the remainder of a division operation.

Example: 10 % 3 equals 1, because 10 divided by 3 equals 3 with a remainder of 1.

Exponentiation ()**: Raises one number to the power of another.

Example: 2 ** 3 equals 8, because 2 raised to the power of 3 equals 8.

Escape character in String

Sometimes, we need to include special characters in strings, like quotes or newlines.

The Backslash (\ ) Escape Character

It allows you to include special characters in a string.

Example:

```
let text = "He said, \"Hello World!\"";

let exampleString = "First line\nSecond line\tTabbed!";
```

Escape character in String

1. Print a path, your output should look like this

Dr suggested, “Do not drink cold water”

2. Print a path, your output should look like this

C:\Users\Name\Documents

3. Write a string that spans multiple lines and includes a tab

Should print something like:
This is line one
	This is line two with a tab

Template Literals (back ticks)

We use backticks (`) to create a template literal:

Example

let greeting = `Hello, World!`;

Embedding Expressions

```
12   let student = "Alice";
13   let greeting = `Hello, ${student}!`;
14
```

Template Literals Exercise

Use a template literal to output the following message:

```
Laptop price is 40000
```

```
const product = {
  name: 'Laptop',
  price: 40000
}
```

Product name and price will be dynamic

String Methods

- Strings in JavaScript have built-in methods for performing various operations.
- These methods make it easy to manipulate and analyze string data.

length Property:
The length property returns the number of characters in a string

toUpperCase() Method:
The toUpperCase() method converts a string to uppercase letters

toLowerCase() Method:
The toLowerCase() method converts a string to lowercase letters

indexOf() Method:
The indexOf() method returns the index of the first occurrence of a specified value in a string.
It returns -1 if the value is not found.

String Methods

1. **split method:** splits a string into an array of substrings based on a specified separator
2. **slice method:** extracts a part of a string and returns it as a new string
3. **trim method:** removes whitespace from both ends of a string

app.js > str2

```
1  // split method
2  let text = "Hello, World!";
3  let words = text.split(" ");
4
5  console.log(words) // ['Hello,', 'World!']
6
7  // trim method
8  let message = "  Hello, World!  ";
9  let trimmedText = text.trim();
10
11 console.log(trimmedText) // Hello, World!
12
13 // slice method
14 let greeting = "Hello, World!";
15 let slicedText = text.slice(7, 12);
16
17
18 console.log(slicedText) // World
```

String Methods

Write a program to reverse a string

Input: we are learning Javascript

Output: tpircsavaJ gninrael era ew

String Methods

Write a progrram to mask an email address. The program will print the masked version of the email.

The masked version should hide the characters of the username part, leaving only the first and last characters visible, and replacing the hidden characters with asterisks (*). The domain part should remain unchanged.

Example:

```
"example@example.com"   ->   // Output: "e*****e@example.com"
"john.doe@gmail.com"    ->   // Output: "j******e@gmail.com"
"a@b.com"               ->   // Output: "a@b.com"
```

no masking if username length is 2

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 > ...

```
1  // array.concat() // merge two arrays
2  let set1 = ['red', 'green']
3  let set2 = ['blue', 'pink']
4
5  let combinedSet = set1.concat(set2)
6  console.log(combinedSet) // Outputs: ['red', 'green', 'blue', 'pink']
7
8  // Check if an Array Contains an Item: array.includes
9  let hasRedColor = toys.includes('red')
10
11 console.log(hasRedColor) // Outputs: true
12
13 // Array.isArray: Checks if something is an array
14 let toys = ['Toy Car', 'Toy Train']
15 let notToys = 'Toy Car'
16
17 let isArray1 = Array.isArray(toys)
18 let isArray2 = Array.isArray(notToys)
19
20 console.log(isArray1) // Outputs: true
21 console.log(isArray2) // Outputs: false
22
```

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.

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.

```
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']
```

JavaScript 2D Array

JavaScript 2D refers to working with two-dimensional arrays, which are like grids with rows and columns. Each element in the array has two indices to access it: one for the row and one for the column.

Example: Creating a 2D array

```
let grid = [
  [1, 2, 3],
  [4, 5, 6],
  [7, 8, 9]
];

// Accessing an element
console.log(grid[0][0]); // Output: 1 (first row, first column)
console.log(grid[1][2]); // Output: 6 (second row, third column)
```

In this example, `grid` is a 2D array with three rows and three columns. We can access elements by specifying their row and column indices.

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 > ...

```
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

The End

BAITUSSALAM
—TECH PARK—