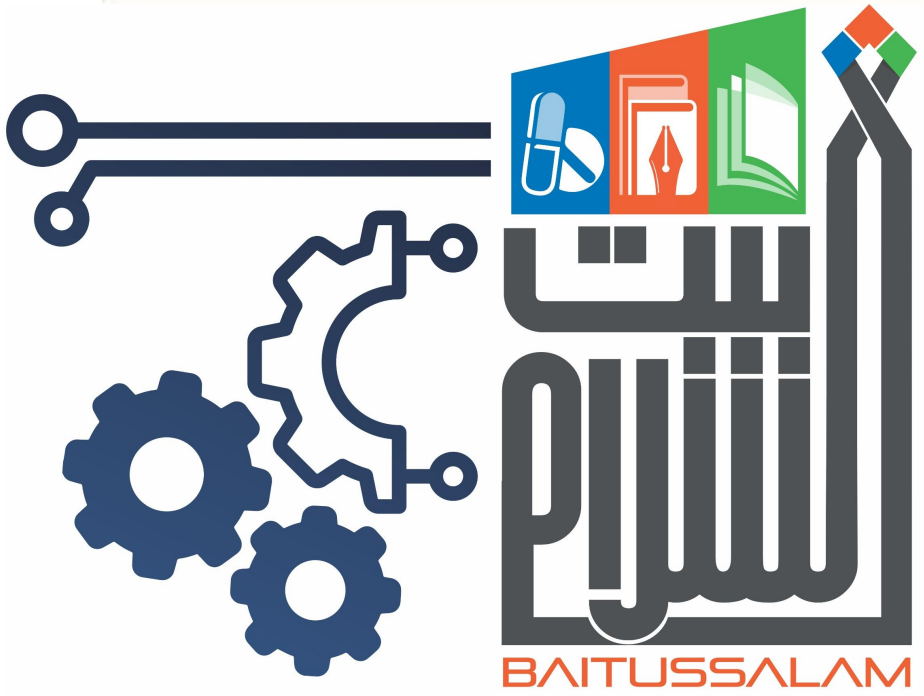


BAITUSSALAM
—TECH PARK—

PSDC-201

BAITUSSALAM
—TECH PARK—

String Methods, 2D Array & Functions

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 Exercises

BAITUSSALAM
—TECH PARK—

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)

BAITUSSALAM
—TECH PARK—

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

BAITUSSALAM
—TECH PARK—

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

JavaScript 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

```
// split method
let text = "Hello, World!";
let words = text.split(" ");

console.log(words) // ['Hello,', 'World!']

// trim method
let message = "  Hello, World!  ";
let trimmedText = text.trim();

console.log(trimmedText) // Hello, World!

// slice method
let greeting = "Hello, World!";
let slicedText = text.slice(7, 12);


console.log(slicedText) // World
```

String Methods Exercise #1

Write a program to reverse a string

Input: we are learning Javascript

Output: tpircsavaJ gninrael era ew

String Methods Exercise #2

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

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.

For of… vs For in… loop

For of … loop [array]

```
17  let arr = [1, 2, 3, 4];
18  for (let value of arr) {
19    console.log(value);
20    // Output: 1, 2, 3, 4 (iterates over array values)
21  }
22
23  let str = "Hello";
24  for (let char of str) {
25    console.log(char);
26    // Output: H, e, l, l, o (iterates over string characters)
27  }
28
```

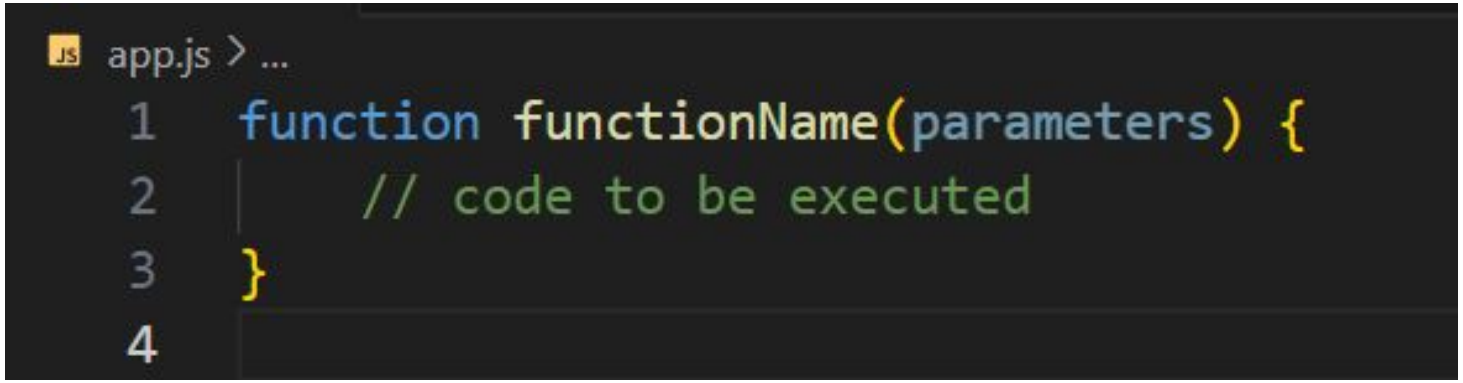
```
31  let obj = { a: 1, b: 2, c: 3 };
32  for (let key in obj) {
33    console.log(key + ": " + obj[key]);
34    // Output: a: 1, b: 2, c: 3 (iterates over object properties)
35  }
36
```

For in… loop [object]

Introduction to JavaScript Functions

- Functions are reusable blocks of code that perform a specific task.
- They help make your code modular and easier to manage.

JavaScript Function Syntax



Parameters, Arguments and return keyword

- Functions can accept parameters (inputs) to perform tasks with different values, parameter is placeholder

- An argument is a value passed to a function when it is called, argument is actual value

- Functions can return a value using the return keyword.

```
39  // Function definition with parameters
40  function add(a, b) {
41      // a and b are parameters
42      return a + b;
43  }
44
45  // Function call with arguments
46  let result = add(3, 5);
47  // 3 and 5 are arguments
48  console.log(result);  // Output: 8
49
```

Console.log() vs return

- `console.log()`:
  - Outputs data to the console.
  - Useful for debugging.
  - Does not affect the function's return value.
- `return`:
  - Ends function execution.
  - Returns a value to the caller.
  - The function result can be used elsewhere in the code.

Function Exercise #1

JavaScript Function Exercise: Truncate a String

Objective:

Design a JavaScript function called `truncateString` that shortens a given string if it exceeds a specified maximum length.

Function Definition:

- Create a function named `truncateString`.
- The function should take two parameters: the input string `str` and the maximum length `maxLength` to truncate the string.

```
// Example calls
console.log(truncateString("Lorem ipsum dolor sit amet", 10));
// Output: "Lorem ipsu..."

console.log(truncateString("Hello, world!", 8)); // Output: "Hello, w..."
console.log(truncateString("JavaScript", 15)); // Output: "JavaScript"
```

Function Exercise #2

Discount Calculator

Scenario: Imagine running an online store where prices and discounts change often. You need to calculate discounts fast and accurately. This is where JavaScript functions come in handy.

The `calculateDiscount` function is designed to take two parameters: `price`, the original price of the product, and `discountPercentage`, the percentage of the discount to apply.

Formula to calculate discount: `price * discountPercentage / 100`

Output will be:

Original Price: 1000, Discounted Price: 700)

Function Exercise #3

Count Vowels in a String

Objective:

Create a JavaScript function named `countVowels` that takes a `string` as input and returns the number of vowels (both lowercase and uppercase) present in the string.

```
console.log(countVowels("hello")); // Output: 2
console.log(countVowels("HELLO")); // Output: 2
console.log(countVowels("Javascript")); // Output: 3
console.log(countVowels("")); // Output: 0
```

The End

BAITUSSALAM
—TECH PARK—