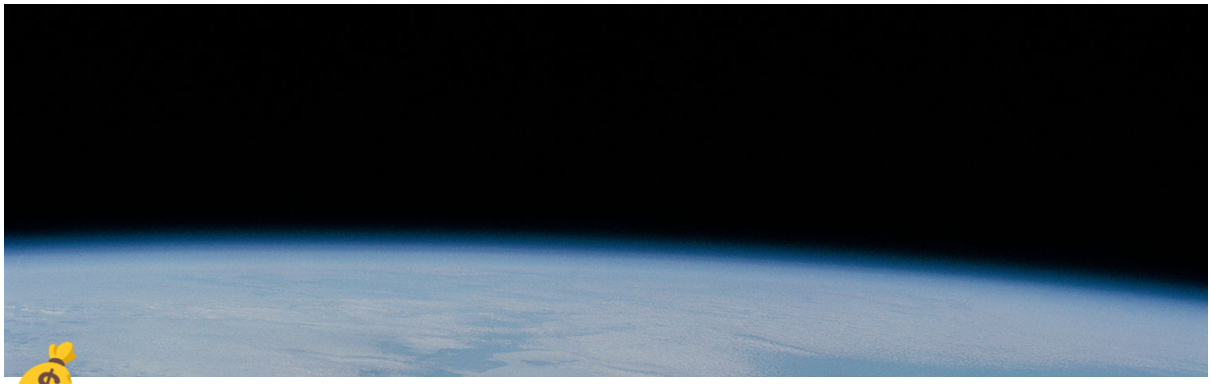

KOSURI’S DEVELOPMENT

INDEX

JAVA SCRIPT

JavaScript is a high-level, interpreted programming language that is widely used on the web. It is often used to add interactivity and dynamic functionality to web pages, such as form validation, animation effects, and more. JavaScript can be used on both the client and server sides of a web application, making it a versatile language for web development.

- Java script is a scripting language.
- Client side language.
- JS is single threaded type language.
- Object based language.

Difference b/w Java and Java Script ?

Java is a programming language used primarily for building standalone applications, while JavaScript is a scripting language used primarily for enhancing web pages. While they share some similarities in their syntax, they are fundamentally different languages with different use cases and capabilities.

Versions of Java Script & ITS Features;

Mostly used version is **ES6** : - **European Computer Manufacturers Association Script**

- Arrow functions and template literals are introduced in ES6.
- Introduced LET & CONST Keywords in ES6
- De-structuring , promises etc..
- ES7 introduced async and await keywords for asynchronous programming.
- ES8 introduced async iterators and rest/spread properties.
- Latest version is ES12.

Template Literals in ES6:

```
let x = “RAVI”;
console.log(`${x} is super`); // RAVI is super
```

Arrow Function in ES6:

```
let function = (a,b) ⇒ a*b;
```

Why Java Script is single threaded ?

JavaScript is single-threaded because it has a single call stack and a single memory heap. This means that only one thing can happen at a time, and any other tasks must wait their turn in line. However, JavaScript is also asynchronous, meaning it can execute non-blocking code and handle multiple tasks simultaneously through callbacks, promises, and async/await functions.

Is Java Script synchronous?

By default, JavaScript is a **synchronous**, single threaded programming language. However, JavaScript is also asynchronous, meaning it can execute non-blocking code and handle multiple tasks simultaneously through callbacks, promises, and async/await functions.

So Java Script acts as both synchronous & asynchronous.

Write a code of Java Script for Displaying some text .

Sure! Here's a simple code in JavaScript to display some text:

```
console.log("Hello, world!");
```

Basic Key words used in JS ?

Some of the basic keywords used in JavaScript are:

- **var** - used to declare a variable
- **let** - used to declare a block-scoped variable
- **const** - used to declare a constant value
- **function** - used to define a function
- **if/else** - used to define conditional statements
- **for/while** - used to loop through code
- **return** - used to return a value from a function
- **null** - represents a null value or no value at all
- **undefined** - used to represent an undefined value or variable

Difference between LET , VAR & CONST ?

When declaring variables in JavaScript, you have three options: `var`, `let` and `const`. Each has its own characteristics and is used for different purposes.

Keyword	Scope	Can Be Updated	Can Be Redefined	Example
`var`	Global-scoped	Yes	Yes	`var x = 5;`
`let`	Block-scoped	Yes	No	`let y = 10;`
`const`	Block-scoped	No	No	`const z = 15;`

```
let  fruits = ["apple", "banana", "orange", "mango"];
fruits.push("goa , i know its not a fruit");
console.log(fruits); //["apple", "banana", "orange", "mango", "goa , i know its not a fruit"]
```

- **var** declares a variable with global scope, can be updated, and can be redefined.
- **let** declares a variable with block scope, can be updated, but cannot be redefined.
- **const** declares a variable with block scope, cannot be updated, and cannot be redefined.

Write a simple code for all arithmetic equations in java script in single file and comment the output?

Here's a simple code in JavaScript to perform all arithmetic equations and log the output to the console:

```
let num1 = 10;
let num2 = 5;

console.log(num1 + num2); // Output: 15 (Addition)
console.log(num1 - num2); // Output: 5 (Subtraction)
console.log(num1 * num2); // Output: 50 (Multiplication)
console.log(num1 / num2); // Output: 2 (Division)
console.log(num1 % num2); // Output: 0 (Modulus)
console.log(num1 ** num2); // Output: 100000 (Exponentiation)
```

This code declares two variables, `num1` and `num2`, and performs all arithmetic equations using these variables. The output of each equation is logged to the console using `console.log()`.

Write the difference between == & ===

- **==** is used for loose equality comparison, which means that it only checks if the values are equal, not necessarily of the same type. For example, `5 == "5"` will return `true`.
- **===** is used for strict equality comparison, which means that it checks if the values are equal and of the same type. For example, `5 === "5"` will return `false`.

```
let x = 5;
let y = "5"
console.log(x==y); // true
console.log(x===y); // false
```

What is a Data Type and give examples ?

In JavaScript, data types can be broadly classified into two categories: primitive types and object types. Here are some examples of data types in JavaScript:

1. **Primitive Types:**

- Number: Represents numeric values. Examples: 3, 3.14, -10.
- String: Represents a sequence of characters. Examples: "Hello", 'JavaScript'.
- Boolean: Represents either true or false.

- Null: Represents the intentional absence of any object value.
- Undefined: Represents a variable that has been declared but has not been assigned a value.
- Symbol: Represents a unique identifier. Introduced in ECMAScript 6.

2. **Object Types / Non primitive**

- Object: Represents a collection of key-value pairs. Examples: {name: "John", age: 25}.
- Array: Represents an ordered collection of elements. Examples: [1, 2, 3], ['apple', 'banana', 'orange'].
- Function: Represents a reusable block of code that performs a specific task.
- Date: Represents a specific moment in time.

Write a code using TYPEOF in JS simply

```
var variable1 = 10;
var variable2 = "Hello, world!";
var variable3 = true;

console.log(typeof variable1); // Output: "number"
console.log(typeof variable2); // Output: "string"
console.log(typeof variable3); // Output: "boolean"
```

FUNCTIONS

What are functions in JS ? Give a simple code for that?

In JavaScript, functions are blocks of reusable code that perform a specific task. They allow you to encapsulate logic and execute it whenever needed. Functions can take input parameters and return a value.

```
// Function definition
function greet(name) {
return "Hello, " + name + "!";
}
// Function call
var greeting = greet("John");
console.log(greeting); // Output: Hello, John!
```

Functions are 2 types : -

1. Regular functions &
2. Function Expressions - These are functions which are assigned to a variable.

JAVASCRIPT HOISTING

Hoisting is a behavior in JavaScript where variable and function declarations are moved to the top of their containing scope during the compilation phase, before the code is executed. This means that you can use variables and call functions before they are actually declared in the code.

```
greet(); // Output: Hello!

function greet() {
console.log("Hello!");
}
```

NOTE: Hoisting is only valid in case of normal function but not for function expression that is a function assigned to a variable.

What is lexical scoping ?

Lexical scope is the ability for a function scope to access variables from the parent scope.

```
function outerFunction() {
var outerVariable = 'Outer';

function innerFunction() {
var innerVariable = 'Inner';
console.log(innerVariable); // Output: Inner
console.log(outerVariable); // Output: Outer
}

innerFunction();
}

outerFunction();
```

STRINGS

Text written b/w inverted comas “ RAVIKANTH “

var text = "RAVIKANTH";

```
// slice(start, end): Extracts a section of the string
var slicedText = text.slice(2, 6);
console.log(slicedText); // Output: VIK

// splice(start, deleteCount, items): Removes or replaces existing elements and/or adds new elements
var splicedText = text.split('');
splicedText.splice(2, 4, 'X', 'Y', 'Z');
console.log(splicedText.join('')); // Output: RAXYZANTH

// replace(searchValue, replaceValue): Replaces a specified value with another value in a string
var replacedText = text.replace('K', 'M');
console.log(replacedText); // Output: RAMIVANTH

// replaceAll(searchValue, replaceValue): Replaces all occurrences of a specified value with another value in a string (requires ES202
var replacedAllText = text.replaceAll('A', 'E');
console.log(replacedAllText); // Output: REVIKENTH

// concat(string): Combines two or more strings
var concatenatedText = text.concat(" SINGH");
console.log(concatenatedText); // Output: RAVIKANTH SINGH

// trim(): Removes whitespace from both ends of a string
var trimmedText = "  " + text + "  ";
console.log(trimmedText.trim()); // Output: RAVIKANTH

// charAt(index): Returns the character at a specified index in a string
var charAtIndex = text.charAt(4);
console.log(charAtIndex); // Output: K

// charCodeAt(index): Returns the Unicode value of the character at a specified index in a string
var charCodeAtIndex = text.charCodeAt(3);
console.log(charCodeAtIndex); // Output: 73

// startsWith(searchValue): Checks if a string starts with a specified value
var startsWithRAV = text.startsWith("RAV");
console.log(startsWithRAV); // Output: true

// endsWith(searchValue): Checks if a string ends with a specified value
var endsWithANTH = text.endsWith("ANTH");
console.log(endsWithANTH); // Output: true

// toUpperCase(): Converts a string to uppercase
var uppercaseText = text.toUpperCase();
console.log(uppercaseText); // Output: RAVIKANTH

// toLowerCase(): Converts a string to lowercase
var lowercaseText = text.toLowerCase();
console.log(lowercaseText); // Output: ravikanth
```

CONCATENATE

```
let t1 = “ravi”
let t2 = "kanth"
let t3 = t1.concat(t2)
console.log(t3); // "ravikanth"
```

ARRAYS

Arrays in JavaScript are data structures that store multiple values in a single variable. They can hold elements of any data type, including numbers, strings, objects, and even other arrays. Arrays are commonly used to organize and manipulate collections of related data.

Here's an example of an array in JavaScript:

```
var fruits = ["apple", "banana", "orange", "mango"];
```

Some of the basic key words used to do specific Tasks are:

pop();

pop() will remove last element from array .

```
let  fruits = ["apple", "banana", "orange", "mango"];
fruits.pop();
console.log(fruits); // ["apple", "banana", "orange"]
```

push();

Adding an element at the end of an array.

```
let  fruits = ["apple", "banana", "orange", "mango"];
fruits.push("goa , i know its not a fruit");
console.log(fruits); //["apple", "banana", "orange", "mango", "goa , i know its not a fruit"]
```

shift();

Removes first element of the array.

```
let  fruits = ["apple", "banana", "orange", "mango"];
let x = fruits.shift();
console.log(fruits); //["banana", "orange", "mango"]
console.log(x); // apple
```

unshift();

Add new element at the beginning of the array.

```
var fruits = ["banana", "orange", "mango"];
fruits.unshift("apple");
console.log(fruits); // ["apple", "banana", "orange", "mango"]
```

Concat();

```
let t1 = ["1"  , "2" ]
let t2 = ["3"  , "4" ]
let t3 = t1.concat(t2)
console.log(t3); // ["1"  , "2" , "3"  , "4" ]
```

Splice();

Removes some portion of array.

```
var arr = [1, 2, 3, 4, 5, 6, 7, 8];
arr.splice(2, 3);
console.log(arr); // [1, 2, 6, 7, 8]
```

Slice();

It cuts some portion and returns the removed portion in an array subset.

```
var arr = [1, 2, 3, 4, 5, 6, 7, 8];
var newArr = arr.slice(2, 5);
console.log(newArr); // [3, 4, 5]
```

OBJECTS

In JavaScript, objects are one of the core data types and are used to represent a collection of related data and functionality.

In real life Car is object - A car has properties weight , color , width etc..

```
// Object definition
var person = {
name: "John",
age: 30,
occupation: "Engineer",
sayHello: function() {
console.log("Hello, my name is " + this.name + ".");
}
};
// Accessing object properties
```

```
console.log([person.name](http://person.name/)); // Output: John
console.log(person.age); // Output: 30

// Calling object method
person.sayHello(); // Output: Hello, my name is John.
```