Session-01

Javascript

**Thanos is on a mission to make his website standout from his rest of universe**

# Web Fundamentals


**Session-01**

**JS Fundamentals**

# Agenda : Javascript Basics

HTML

HTML + CSS

HTML + CSS
+ JAVASCRIPT

01 JS Intro

02 Var, Let , Const

03 Hoisting

04 Data Types, Operators, Control Statements

05 Functions, Scope, Objects, Arrays

06 Hands-On

# JS- Intro

- JavaScript was invented by Brendan Eich in 1995, and became an ECMA standard in 1997.
- **ECMAScript is the official name** of the language.
- JS Can create, edit, delete HTML and CSS elements
- It is the foundation of frontend web development
- Key ingredient **in frameworks like ReactJS**, Angular, and VueJS
- Solid **backends with platforms like Nodejs, Deno** runs desktop applications like Slack, Atom built on top of JS
- Growing community in AI/ML with JS libraries

**In short, it is everywhere—and for good reasons.**

# JS- Intro



## How to declare variables?

# Syntax & Variables

## 3 ways to declare variables

- var
- let
- const

```
// Declare a variable using var
var myVariable =
  "Hello, world! I can have default undefined value and I can change";
console.log(myVariable);

// Declare a variable using let
let LetVariable = "Hello, world! I can also change";
console.log(myOtherVariable);

// Declare a constant
const ConstantVariable = "I never change!";
console.log(myConstant);
```

<|>esto

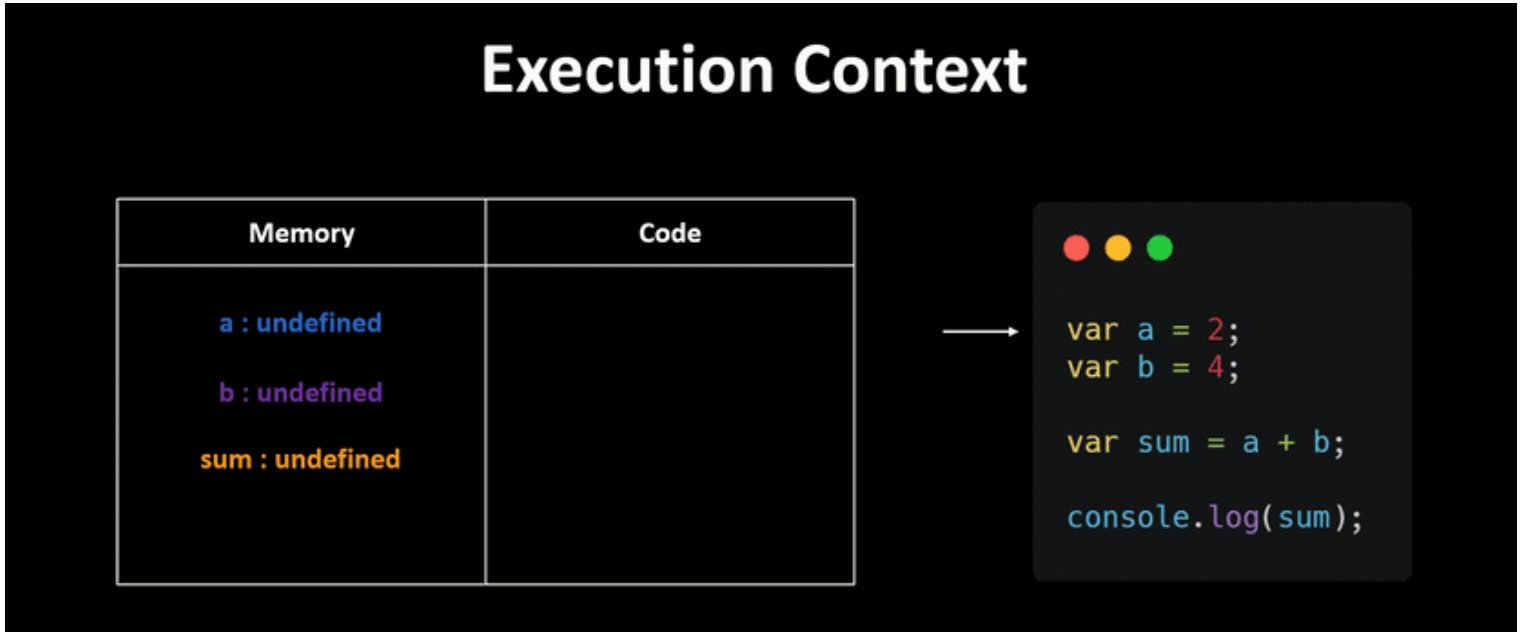
# Syntax & Variables

| | var | let | const |
|---|---|---|---|
| **origins** | pre ES2015 | ES2015(ES6) | ES2015(ES6) |
| **scope** | globally scoped OR **function** scoped. attached to window object | globally scoped OR **block** scoped | globally scoped OR **block** scoped |
| **global scope** | is attached to Window object. | not attached to Window object. | attached to Window object. |
| **hoisting** | **var** is hoisted to top of its execution (either global or function) and initialized as ***undefined*** | **let** is hoisted to top of its execution (either global or block) and left **uninitialized** | **const** is hoisted to top of its execution (either global or block) and left **uninitialized** |
| **redeclaration within scope** | **yes** | **no** | **no** |
| **reassigned within scope** | **yes** | **yes** | **no** |

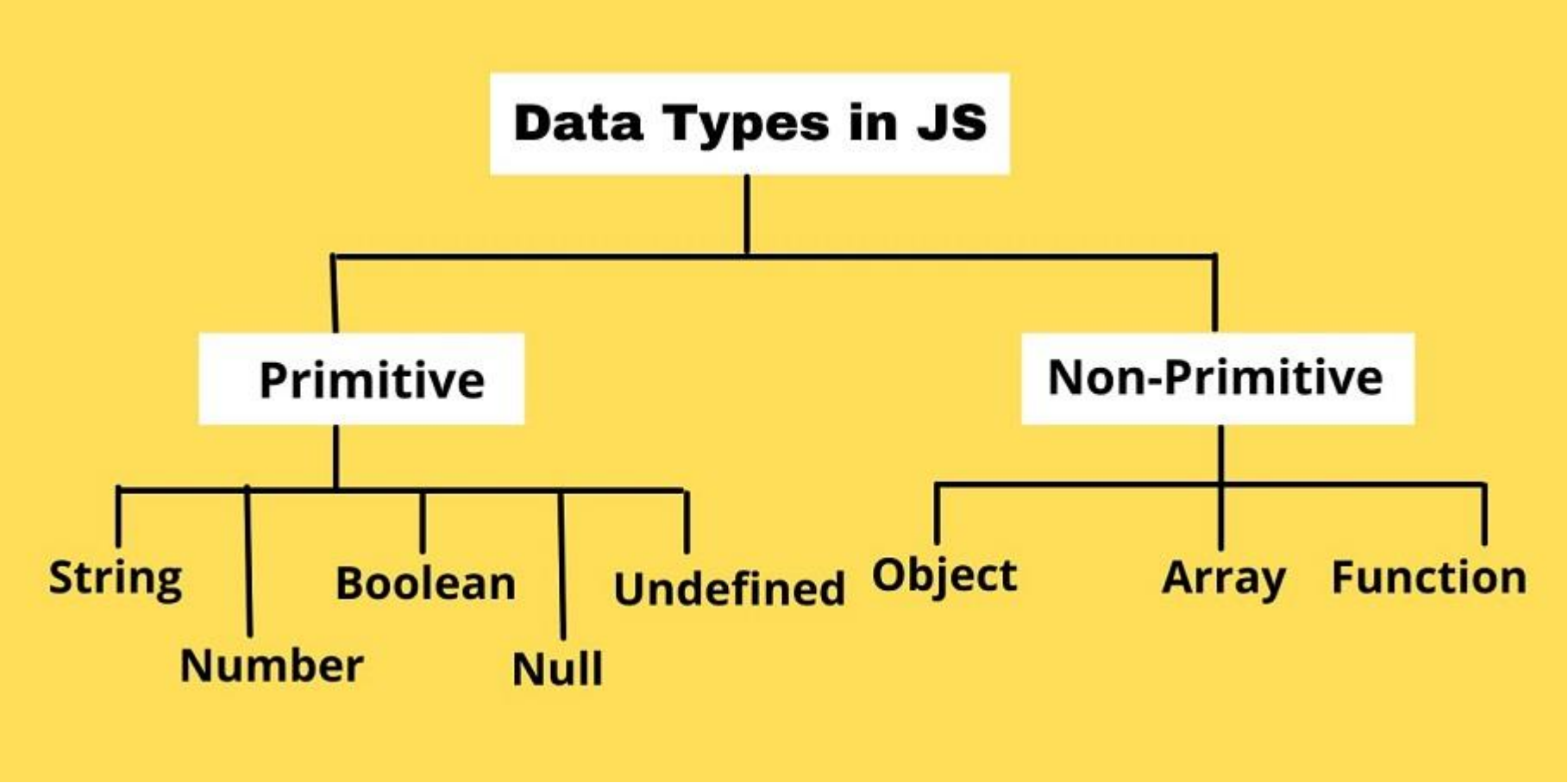
# Hoisting

Some of you might be confused with this concept?

Variable and function declarations are **moved to the top of their respective scopes** during code execution **with default value**



# Data Types


Data Types in JS
Primitive
Non-Primitive
String
Number
Boolean
Null
Undefined
Object
Array
Function

# Input & Output

```
let name = prompt("What is your name?");
if (name !== null) {
  console.log(`Hello, ${name}!`);
} else {
  console.log("You didn't enter a name.");
}
```

Input Prompt

Output Console

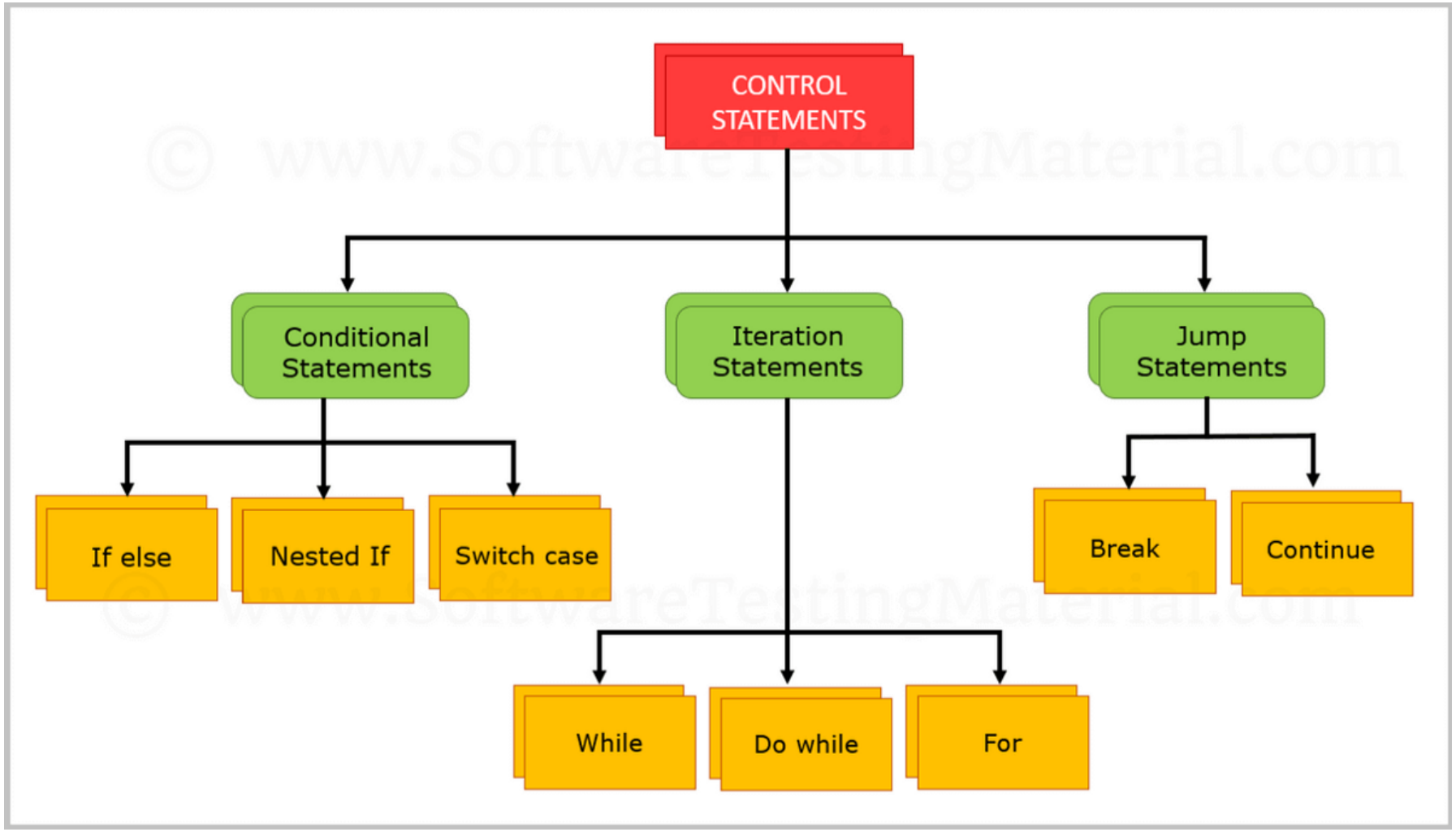
# Arithmetic Operators

1. Addition (+)
2. Subtraction (-)
3. Division (/)
4. Remainder (%)
5. Multiplication (*)
6. Increment (++)
7. Decrement (--)

# Arithmetic Operators- Strings

```
let str1 = "hello";
let str2 = "world";
let char1 = "a";
let char2 = "b";

console.log(str1 + str2);   // Output: "helloworld"
console.log(str1 - str2);   // Output: NaN (Not a Number)
console.log(str1 * str2);   // Output: NaN
console.log(str1 / str2);   // Output: NaN

console.log(char1 + char2); // Output: "ab"
console.log(char1 - char2); // Output: NaN
console.log(char1 * char2); // Output: NaN
console.log(char1 / char2); // Output: NaN
```

# Control Statements



# Comparison Operators

| Operator | Description | Comparison | Result |
|---|---|---|---|
| ** Let X = 5** | | | |
| == | Equals to | x=8 | False |
| === | Equal value & data type | x===5 | True |
| != | Not equal | x!=8 | True |
| !== | Not equal value or data type | x!==5 | False |
| > | Greater than | x>8 | False |
| < | Less than | x<8 | True |
| >= | Greater than or equals to | x>=8 | False |
| <= | Less than or equals to | x<=8 | True |

# Ternary Operator


```
let isRemote = true;
let WhereDidYouPrepare = isRemote ? 'Pesto graduate' : 'other bootcamp';
```

# Truthy Vs Falsy

**What is the output of this JavaScript code?**

JavaScript

```javascript
console.log(null==undefined);
console.log(null==null);
console.log(undefined==undefined);
```

# Truthy & Falsy Values

## Javascript: Truthy & Falsey

| Always Falsy | Always Truthy |
|---|---|
| false | '0' (a string containing a single zero) |
| 0 (zero) | 'false' (a string containing the text “false”) |
| -0 (minus zero) | [] (an empty array) |
| 0n (BigInt zero) | {} (an empty object) |
| '', "", `` (empty string) | function(){} (an “empty” function) |
| null | |
| undefined | |
| NaN | |

# Functions

function is a **block of code** that performs a **specific task and can be reused**

```
function functionName(parameter1, parameter2, ...) {
  // function code here
  return result;
}
```

# Function Expressions / Anonymous functions

A function expression is when we define a function as part of a larger expression, such as assigning it to a variable.

```
const add = function(a, b) {
  return a + b;
};
```

# Arrow Functions

An arrow function is a **shorthand syntax for defining a function expression**.

```
const multiply = (a, b) => a * b;
```

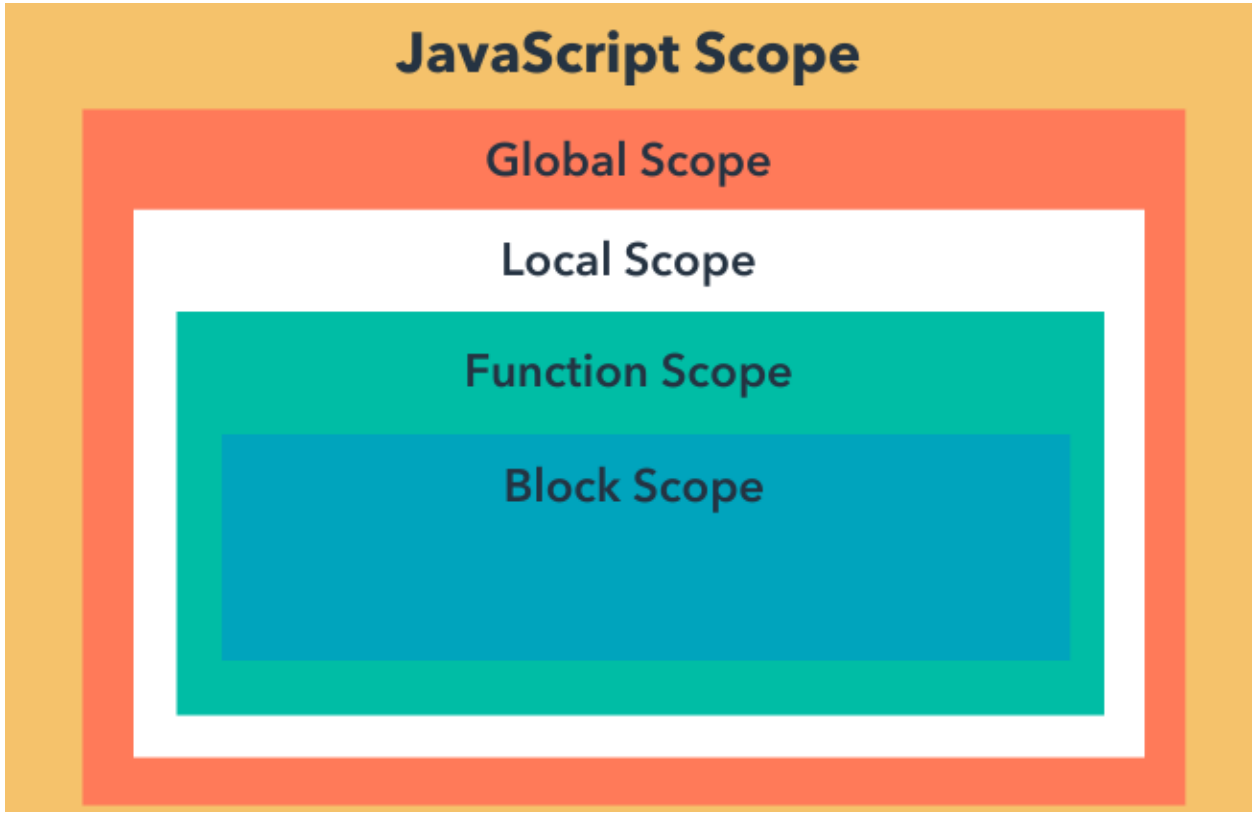
# Let's talk about "Scope"

scope can be a confusing concept for beginners

**Let me make it easy for you!**

# Scope

Scope refers to the **visibility and accessibility of variables and functions** in different parts of your code.



# Global Object

When running JavaScript code in a web browser, global variables are usually assigned as properties to the global window object.

- X has been declared with var, in which case a property named "X" is **added to the global object.**
- X has been declared with let, in which case a property named "X" is not added to the global object, but **X is still accessible from anywhere in the code.**
- X has not been declared with var or let, in which case the variable becomes a property of the global object, even if assigned to inside a function.

# Knowledge check: What is the output?

```
console.log(x);
var x = 5;
```

Undefined
***//Hoisting***

```
function foo() {
  var x = 10;
  if (true) {
    var x = 20;
    console.log(x);
  }
  console.log(x);
}

foo();
```

20
20
***//Scope***

# JS Array

Array is an ordered collection of elements, which can be of any data type

```
// create an array using array literal syntax
let fruits = ['apple', 'banana', 'orange', 'grape'];

// access array elements
console.log(fruits[0]); // Output: "apple"
console.log(fruits[2]); // Output: "orange"
```

## JavaScript Array Methods

JS

- .concat()
- .filter()
- .pop()
- .slice()
- .unshift()
- .shift()
- .sort()
- .find()
- .push()
- .reverse()
- .map()
- .splice()
- .join()
- .toString()

# JS Objects

- Object is a **collection of properties** that have a name and a value
- Objects can contain **other objects, functions, and even arrays**, making them a powerful way to store and organize data

```javascript
// create an object using object literal syntax
let person = {
  firstName: 'John',
  lastName: 'Doe',
  age: 30,
  address: {
    street: '123 Main St',
    city: 'Anytown',
    state: 'CA',
    zip: '12345'
  },
  hobbies: ['reading', 'traveling', 'music'],
  sayHello: function() {
    console.log('Hello, my name is ' + this.firstName + ' ' + this.lastName)
  }
};

// access object properties
console.log(person.firstName); // Output: "John"
console.log(person.address.city); // Output: "Anytown"
console.log(person.hobbies[1]); // Output: "traveling"
```

# String Object & Methods

## JavaScript String Methods

1. charAt()
2. charCodeAt()
3. concat(str1, str2, ...)
4. includes()
5. endsWith()
6. indexOf()
7. lastIndexOf()
8. match()
9. matchAll()
10. repeat()
11. replace()
12. replaceAll()
13. search()
14. slice()
15. split()
16. startsWith()
17. substr()
18. substring()
19. toLowerCase()
20. toUpperCase()
21. toString()
22. trim()
23. valueOf()

# Math Object

Math object in JavaScript provides a set of built-in mathematical functions and constants

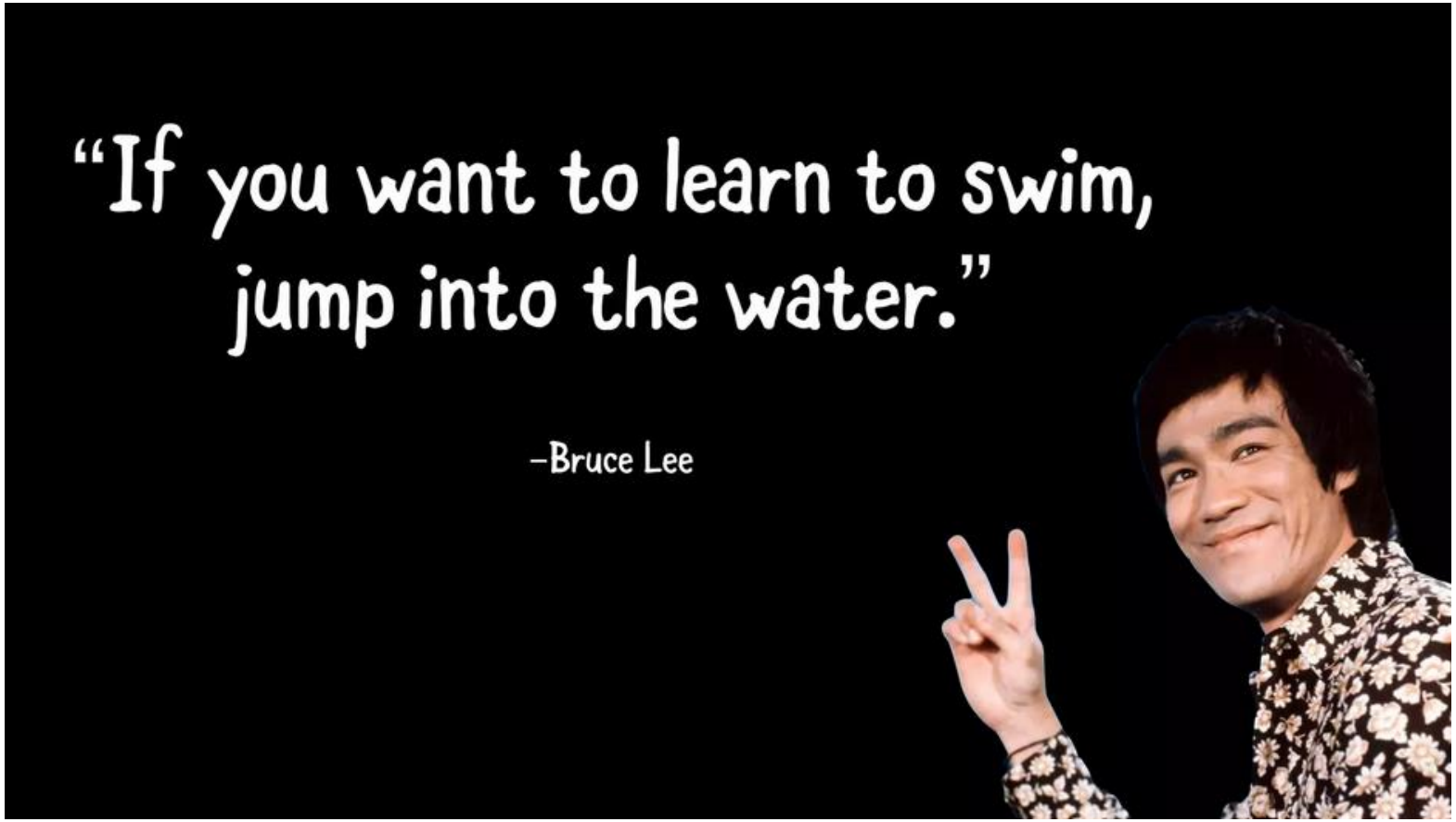
```
// find the absolute value of a number
let num1 = -5;
let absNum1 = Math.abs(num1);
console.log(absNum1); // Output: 5

// round a number to the nearest integer
let num2 = 3.7;
let roundNum2 = Math.round(num2);
console.log(roundNum2); // Output: 4
```

# Exception Handling

```
try {
  // code that might throw an error
} catch (error) {
  // code to handle the error
}
```

Example

```
try {
  let num = 10 / 0; // dividing by 0 will throw an error
  console.log(num);
} catch (error) {
  console.log('An error occurred: ' + error.message);
}
```


"If you want to learn to swim,
jump into the water."
–Bruce Lee

# Q & A