Detailed JavaScript Functions & Methods Explanation 📚

I'll explain this code comprehensively with all scenarios and references.

1️⃣ Basic Functions - Introduction

Problem Without Functions

```
var num1 = 6
var num2 = 5
console.log(num1+num2)  // 11

var num3 = 3
var num4 = 2
console.log(num1+num2)  // Still 11 (forgot to change variables!)
```

Problem: Code repetition, hard to maintain

Solution: Using Functions

```
function sum() {
    var num1 = 6
    var num2 = 5
    console.log(num1+num2)
}

sum()  // Calling the function - Output: 11
```

Benefits:

- ✅ Reusable code
- ✅ Easy to maintain
- ✅ Organized structure

Reference: W3Schools - JS Functions

2️⃣ Function Parameters & Arguments

Basic Parameters

```
function sum(num1, num2) {
    console.log(num1 + num2)
}

sum(2, 3)     // 5
sum(10, 20)   // 30
```

Terminology:

- **Parameters**: `num1, num2` (variables in function definition)
- **Arguments**: `2, 3` (actual values passed when calling)

Default Values with OR Operator

```
function sum(num1, num2) {
    num1 = num1 || 0
    num2 = num2 || 0
    console.log(num1 + num2)
}
```

Scenarios:

```
sum()           // 0 + 0 = 0
sum(5)          // 5 + 0 = 5
sum(2, 3)       // 2 + 3 = 5
sum(true, true) // 1 + 1 = 2 (true converts to 1)
sum(2, 3, 4)    // 2 + 3 = 5 (4 is ignored)
```

How `||` Works:

```
num1 = num1 || 0
// If num1 is undefined/null/false/0/"" → use 0
// Otherwise use num1
```

Visual Flow:

```
sum() called
│
├─ num1 = undefined → num1 || 0 → num1 = 0
├─ num2 = undefined → num2 || 0 → num2 = 0
└─ console.log(0 + 0) → 0
```

3️⃣ Types of Functions

1. Built-in Functions

Functions provided by JavaScript:

```
// Number conversions
var num1 = Number("10.6")    // 10.6 (keeps decimal)
var num2 = parseInt("10.6")  // 10 (removes decimal)

console.log("Number:", num1)    // Number: 10.6
console.log("ParseInt:", num2)  // ParseInt: 10
```

Comparison Table:

Function	Input	Output	Use Case
`Number("10.6")`	"10.6"	10.6	Keep decimals
`parseInt("10.6")`	"10.6"	10	Get integer only
`Number("abc")`	"abc"	NaN	Invalid conversion
`parseInt("10abc")`	"10abc"	10	Stops at first non-digit

Reference: W3Schools - Number Methods

---

2. User-Defined Functions

A) Function Statement (Declaration)

```
function sayHello(name) {
    console.log("Welcome " + name)
```

```
}

sayHello("Ahmed")  // Welcome Ahmed
```

Characteristics:

- ✅ **Hoisted** (can be called before declaration)
- ✅ Has a name
- ✅ More readable

Hoisting Example:

```
sayHello("Sara")  // ✅ Works! Output: Welcome Sara

function sayHello(name) {
    console.log("Welcome " + name)
}
```

Reference: W3Schools - Function Definitions

B) Function Expression

```
var result = function(name) {
    console.log("Welcome " + name)
}

result("Ahmed")  // Welcome Ahmed
```

Characteristics:

- ❌ **NOT Hoisted** (must be defined before use)
- ❌ Anonymous (no name)
- ✅ Can be assigned to variables

Hoisting Example:

```
result("Sara")  // ❌ Error! result is not a function

var result = function(name) {
```

```
    console.log("Welcome " + name)
}
```

Why the Error?

```
// JavaScript sees this:
var result = undefined  // Hoisted declaration
result("Sara")          // Trying to call undefined()
result = function(name) { ... }  // Assignment happens later
```

Function Declaration vs Expression - Summary

Feature	Function Declaration	Function Expression
Syntax	`function name() {}`	`var name = function() {}`
Hoisting	✅ Yes	❌ No
Call Before Define	✅ Yes	❌ No
Anonymous	❌ Must have name	✅ Can be anonymous
Use Case	Regular functions	Callbacks, conditionals

4️⃣ Return Statement

Function Without Return

```
function sum(num1, num2) {
    console.log(num1 + num2)
}

var result = sum(2, 3)
console.log(result)  // undefined
```

Why undefined?

- Function prints but doesn't **return** a value
- Default return is `undefined`

Function With Return

```
function sum(num1, num2) {
    return num1 + num2
}

var result = sum(2, 3)
console.log(result)  // 5 ✅
```

Return Scenarios:

```
function sum(num1, num2) {
    return num1 + num2
    console.log("This never runs")  // Code after return is unreachable
}

function checkAge(age) {
    if(age >= 18) {
        return "Adult"
    }
    return "Minor"
}

function getData() {
    return true         // Returns boolean
    return [1, 2, 3]    // Can return arrays
    return {name: "Sara"}  // Can return objects
}
```

Reference: W3Schools - Function Return

5️⃣ Hoisting in Detail

Variable Hoisting

```
console.log(fname)  // undefined (not error!)
var fname = "Mohamed"
console.log(fname)  // Mohamed
```

What JavaScript Does:

```
// Step 1: Hoisting phase
var fname = undefined

// Step 2: Execution phase
console.log(fname)   // undefined
fname = "Mohamed"
console.log(fname)   // Mohamed
```

Function Hoisting

```
sayHello("Ahmed")  // ✅ Works! Output: Welcome Ahmed

function sayHello(name) {
    console.log("Welcome " + name)
}
```

But Function Expression Doesn't Hoist:

```
result("Ahmed")  // ❌ Error: result is not a function

var result = function(name) {
    console.log("Welcome " + name)
}
```

Reference: W3Schools - JS Hoisting

6️⃣ Callback Functions

What is a Callback?

A function passed as an argument to another function.

```
function greet(name, callback) {
    console.log("Welcome " + name)

    if(typeof callback === 'function') {
        callback()
    } else {
        console.log("Is not a function")
```

```
    }
}

function jsTrack() {
    console.log("Hello JS track")
}

greet("Ahmed", jsTrack)
// Output:
// Welcome Ahmed
// Hello JS track
```

Anonymous Callback

```
greet("Ahmed", function() {
    console.log("Callback function")
})

// Output:
// Welcome Ahmed
// Callback function
```

Real-World Callback Example

```
// Simulating async operation
function fetchData(callback) {
    console.log("Fetching data...")
    setTimeout(function() {
        console.log("Data received!")
        callback()
    }, 2000)
}

fetchData(function() {
    console.log("Processing data...")
})

// Output (after 2 seconds):
// Fetching data...
```

```
// Data received!
// Processing data...
```

Reference: W3Schools - JS Callbacks

---

7 String Methods

Common String Methods

```
var name1 = "welcome pd Track"

console.log(name1.toUpperCase())     // "WELCOME PD TRACK"
console.log(name1.toLowerCase())     // "welcome pd track"
console.log(name1.length)            // 16
```

Accessing Characters

```
console.log(name1[0])         // "w"
console.log(name1[40])        // undefined (out of range)
console.log(name1.charAt(0)) // "w"
console.log(name1.charAt())  // "w" (default index 0)
```

Difference between `[]` and `charAt()`:

Method	Out of Range	Read-only
`[index]`	`undefined`	Yes
`charAt(index)`	`""` (empty string)	Yes

String Manipulation

```
var name1 = "welcome pd Track"

// Concatenation
console.log(name1.concat(" ahmed", " mohamed"))
```

```
// "welcome pd Track ahmed mohamed"

// Check endings
console.log(name1.endsWith('ck'))    // true
console.log(name1.endsWith('cki'))   // false
console.log(name1.startsWith('w'))   // true

// Find position
console.log(name1.indexOf('c'))      // 3 (first occurrence)
console.log(name1.lastIndexOf('c'))  // 14 (last occurrence)
```

Reference: W3Schools - String Methods

Split and Join

```
var email = "name=sara;email=sara@gmail.com"

// Split by semicolon
console.log(email.split(';'))
// ["name=sara", "email=sara@gmail.com"]

// Split and rejoin
console.log(email.split(';').join(' | '))
// "name=sara | email=sara@gmail.com"

// Practical example
var csv = "apple,banana,orange"
var fruits = csv.split(',')  // ["apple", "banana", "orange"]
```

Slice Method

```
var text = "Hello World"

console.log(text.slice(0))    // "Hello World" (from index 0)
console.log(text.slice(2))    // "llo World" (from index 2)
console.log(text.slice(2, 5)) // "llo" (from 2 to 5, not including 5)
console.log(text.slice(-5))   // "World" (last 5 characters)
```

Visual Representation:

```
Index:  0  1  2  3  4  5  6  7  8  9  10
Text:   H  e  l  l  o     W  o  r  l  d
        ↑              ↑
     slice(0)      slice(2)
```

---

8 Array Methods

Basic Array Operations

```
var arr1 = ["sara", "ahmed", "mohamed", "sara", "belal", "skss"]

console.log(arr1.length)  // 6
```

Adding Elements

```
var arr = ["sara", "ahmed"]

// Add to end
arr.push("mostafa")
console.log(arr)  // ["sara", "ahmed", "mostafa"]

// Add to beginning
arr.unshift("mahmoud")
console.log(arr)  // ["mahmoud", "sara", "ahmed", "mostafa"]
```

Visual:
```
Before push:     ["sara", "ahmed"]
After push:      ["sara", "ahmed", "mostafa"]

Before unshift:  ["sara", "ahmed", "mostafa"]
After unshift:   ["mahmoud", "sara", "ahmed", "mostafa"]
```

Removing Elements

```
var arr = ["mahmoud", "sara", "ahmed", "mostafa"]

// Remove from end
arr.pop()
console.log(arr)  // ["mahmoud", "sara", "ahmed"]

// Remove from beginning
arr.shift()
console.log(arr)  // ["sara", "ahmed"]
```

Reference: W3Schools - Array Methods

Slice Method (Non-Destructive)

```
var arr1 = ["sara", "ahmed", "mohamed", "belal", "skss"]

console.log(arr1.slice(1, 4))  // ["ahmed", "mohamed", "belal"]
console.log(arr1)              // Original unchanged
```

Parameters:

- `slice(start, end)` : Extracts from `start` to `end` (not including `end`)
- Returns **new array**
- Original array **unchanged**

Examples:

```
var arr = [0, 1, 2, 3, 4, 5]

console.log(arr.slice(2))      // [2, 3, 4, 5]
console.log(arr.slice(1, 4))   // [1, 2, 3]
console.log(arr.slice(-2))     // [4, 5] (last 2)
console.log(arr.slice(0))      // [0, 1, 2, 3, 4, 5] (copy array)
```

Sort Method

```
var arr2 = [3, 4, 6, 100, 90, -3]
```

```
// Default sort (converts to strings!)
console.log(arr2.sort())  // [-3, 100, 3, 4, 6, 90] ❌ Wrong!
```

Why Wrong?

- Default sort converts numbers to strings
- Compares as: `"-3"`, `"100"`, `"3"`, etc.
- `"100"` comes before `"3"` alphabetically!

Correct Numerical Sort

```
var arr2 = [3, 4, 6, 100, 90, -3]

// Ascending order
console.log(arr2.sort(function(a, b) {
    return a - b
}))
// [-3, 3, 4, 6, 90, 100] ✅ Correct!

// Descending order
console.log(arr2.sort(function(a, b) {
    return b - a
}))
// [100, 90, 6, 4, 3, -3] ✅
```

How it Works:

```
function(a, b) {
    return a - b
}

// If a - b < 0: a comes first
// If a - b > 0: b comes first
// If a - b = 0: keep original order
```

Examples:

```
Compare 100 and 3:
100 - 3 = 97 (positive) → 3 comes first
```

```
Compare 3 and -3:
3 - (-3) = 6 (positive) → -3 comes first
```

Reference: W3Schools - Array Sort

📊 Summary Table

Function Types

Type	Syntax	Hoisting	Use Case
Declaration	`function name() {}`	✅ Yes	Regular functions
Expression	`var name = function() {}`	❌ No	Callbacks

String Methods

Method	Purpose	Example
`toUpperCase()`	Convert to uppercase	`"hello"` → `"HELLO"`
`slice(start, end)`	Extract substring	`"hello".slice(1,4)` → `"ell"`
`split(separator)`	Split into array	`"a,b,c".split(',')` → `["a","b","c"]`

Array Methods

Method	Purpose	Mutates Array?
`push()`	Add to end	✅ Yes
`pop()`	Remove from end	✅ Yes
`slice()`	Extract portion	❌ No
`sort()`	Sort elements	✅ Yes

BY. Abdullah Ali

Contact : +201012613453