

# JS refresher & Execution Context

## Agenda

### \* JS refresher

- About JS
- Where to use JS?
- What JS can't do in browsers?
- ECMAScript vs Javascript.
- How do you add JS in your web page?
- variable & their scopes
- var, let, const
- Data types in JS.
  - primitive
  - non primitive
- typeof operator
- Objects
- JS Code Execution → Hoisting & EC

\* Do you agree JS is powerful programming language?

yes

\* Who invented JS?

- Brendan Eich in 1995, he was the co-founder of Mozilla Corporation.

\* Where to use JS?

\* web apps (frontend + backend)

\* Mobile apps

\* desktop apps

\* Gaming apps

\* IOT (Internet of things)

\* Testing automation framework (TA) can write their test script.

\* What JS can't in the browser?

① Can't read/write files.

② Can't access camera/microphone automatically.

③ Can't access other's domain data directly.

JS doesn't support multi-threading.

↓  
JS runs on single thread.

↓ why? (IA)  
It has only a single stack.

\* ECMAScript(ES) vs Javascript. (IA)

- a scripting language specification  
standardized by ECMA International  
in ECMA-262 (document)

- JS is an implementation of ECMA  
Standard.

→ ES6 - 2015

↓  
ES14 - 2023

\* Java vs Javascript

↓  
① Java is a completely object-oriented  
programming language. whereas JS is an  
object-based PL.

② JS is dynamically typed language.

var a = 10; // number

var a = 'hello'; // string

var a = false; // boolean.

Java is statically typed language.

int a = 10;

char a = 'A';

float a = 20.5;

\* How to add JS in a web page?

- <script> tag.

\* How many ways to add JS in a web page?

There are 2 ways:

- inline/internal JS. (<script> JS code here  
</script>)

- External JS. (.js extension)

- you can use <script> tag inside the  
<head> tag or <body> tag (generally  
at the bottom of closed <body> tag).

## Best practices:

- If you want to include JS in `<head>` tag.

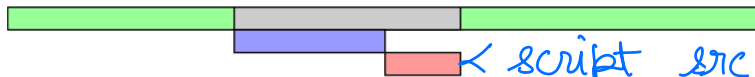
### Legend

- HTML parsing
- HTML parsing paused
- Script download
- Script execution

### `<script>`

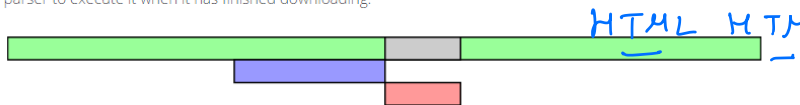
Let's start by defining what `<script>` without any attributes does. The HTML file will be parsed until the script file is hit, at that point parsing will stop and a request will be made to fetch the file (if it's external). The script will then be executed before parsing is resumed.

`<head>`



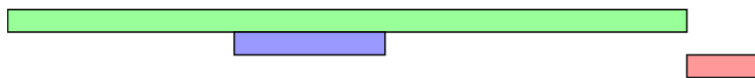
### `<script async>`

`async` downloads the file during HTML parsing and will pause the HTML parser to execute it when it has finished downloading.



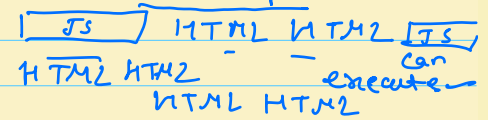
### `<script defer>`

`defer` downloads the file during HTML parsing and will only execute it after the parser has completed. `defer` scripts are also guaranteed to execute in the order that they appear in the document.



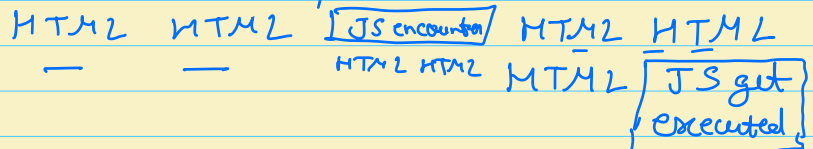
`<script src = "demo.js" async >`  
`</script>`

download JS in parallel



`<script src = "demo.js" defer >`  
`</script>`

download JS & execute at the end of completely parsed of your HTML.



→ 3rd party library

`<script src = "gtm.js" async >`  
`</script>`

→ init()

demo.js  
→ gtm.init();

`<script src = "demo.js" defer >`  
`</script>`

`<script src = "demo.js" defer >`  
`</body>`

## variable & their scope

- variable : just container to hold any value.

- ES5 :

var name = "Ashwani";

} Global scope  
function scope (local scope)

- ES6 :

let name = "Sandeep";

const companyName = "IBM";

3 scopes :

① Global scope

② Local scope (function scope)

③ Block scope.

Global scope : - When a variable is declared outside the function or declared with window object.

- Accessible from any function.

## local scope (function scope):

- When a variable is declared inside the function.
- Accessible within the function only.

## Block scope:

- When a variable is declared inside a block (within the curly braces {} or within a loop)
- Accessible only within that block

Eg: `scoping.js`

```
var a = 10; ✓ // global scope

function print() { // local scope.
  var b = 20; ✓
  console.log(a); // 10 ✓
  console.log(a+b); // 30 ✓
}

var c = 10; // let if c = 0; // false.
if (c) { // true.
  let d = 20; // block scope.
  console.log(d); // 20; ✓
}
```

```

    }
    console.log(d); // reference error: d is not defined
    print();

```

keyword	scope	hoisting	can be reassigned	can be re-declared.
let	block ✓	no ✓	yes ✓	no ✓
const	block ✓	no ✓	no ✓	no ✓
var.	local (function scope) ✓	yes ✓	yes ✓	yes ✓

- what is hoisting?

JS engine moves all the declaration of variables at the top.  
or  
functions

```

var a; // undefined
a = 10;

```

```

function print() {
  var b; // undefined
  var c; // undefined
  b = 20;
  console.log(a); // 10
  console.log(a+b); // 30
  c = 10;
}

```



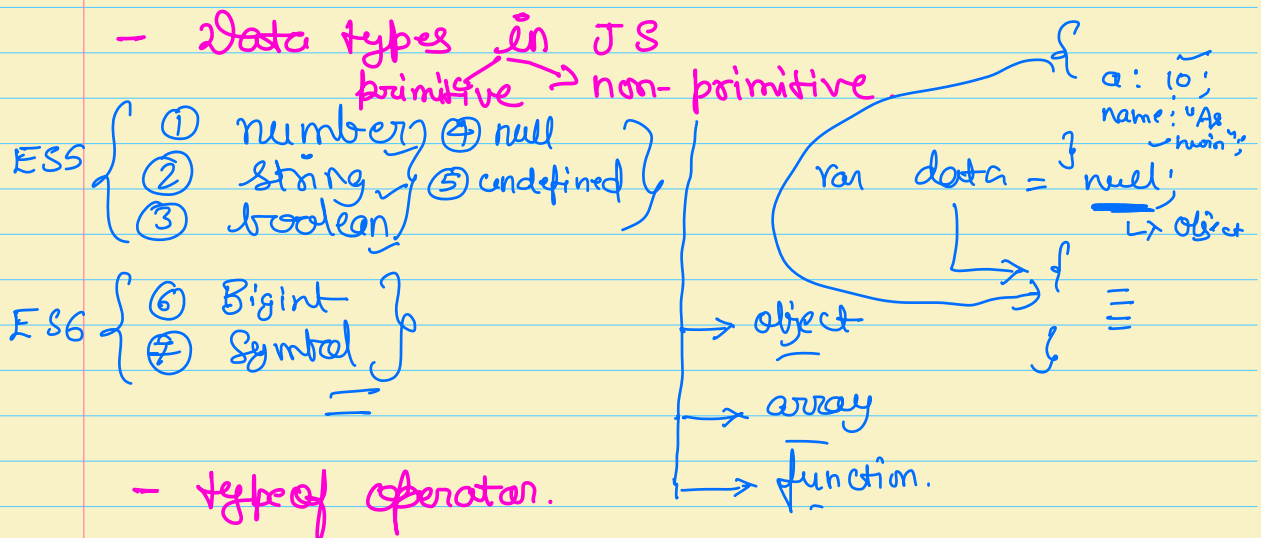
```

if (c) { // true
  let d = 20;
  console.log(d); // 20
}
console.log(d); // error.

```

block scope  
reference  
d is not defined.

## - data types in JS



## - typeof operator.

typeof null ? // object. var data = null;

typeof undefined ? // undefined. // object.

```

var a;
// undefined.

```

- ✓
- ✓
- ✓
- Execution Context: (EC) where JS code gets executed. ✓
- ① GEC ( Global Execution context )
  - ② FEC ( Function Execution context )  
↳ EC

### Doubt

type of null returns "object".

Note: null is not an object actually in JS.

It is a primitive value that represents the intentional absence of any object value.

It is used to indicate that a variable or object property that does not currently have a value assigned to it.