Next week:
Wed - Browser perf doubt handling session. Class will start at 7:05AM
Fri - Grid
Sat - Contest.

JS refresher & Execution Context

Agenda

* JS refresher
  - About JS
  - Where to use JS?
  - What JS can't do in browsers?
  - ECMA Script 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 (QA) 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 ? (IQ)

It has only a single stack.

* Ecmascript(ES) vs Javascript. (IQ)

- 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.
```

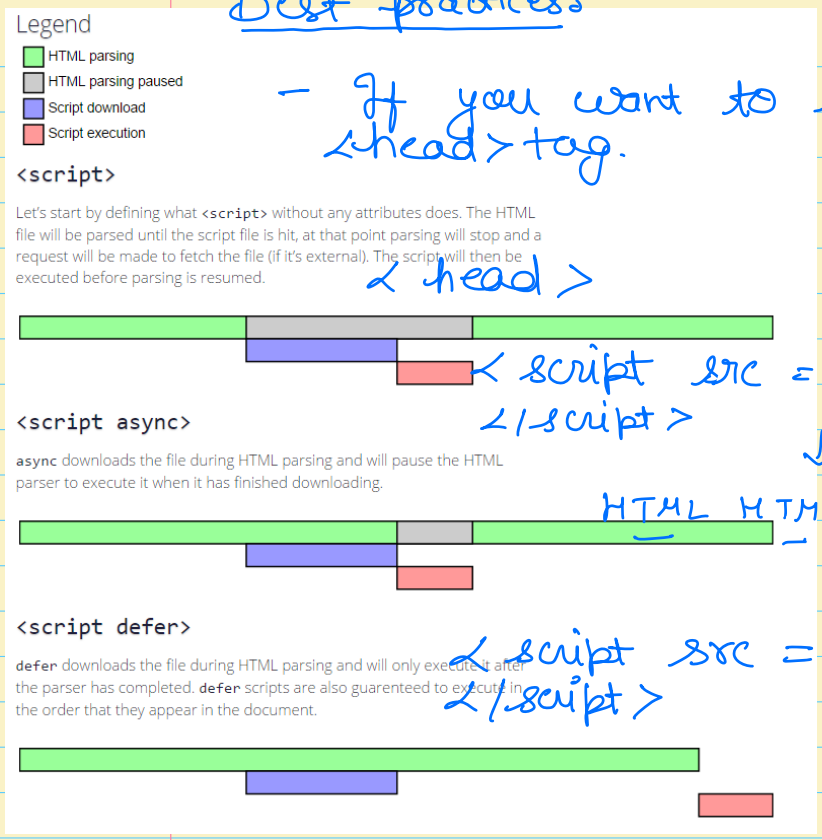
Java is ~~statically~~ typed language.

```
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 extention)
  - 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.

< head >

< script src = " demo.js " async >

< /script >

download JS in parallel

HTML HTML [JS] HTML HTML [JS can execute] HTML HTML

< script src = " demo.js " defer >

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

HTML HTML [JS encountered] HTML HTML HTML [JS get executed]

→ 3rd party library

< script src = " GTM.js " async ></script>

↳ init()

demo.js → In head tag.

↳ gtm.init();

< script src = " demo.js defer >

</script>

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

</ 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
        console.log(d); // scope.
                        // 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 or functions at the top.

```
var a;  // undefined
a = 10;

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

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

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

block scope

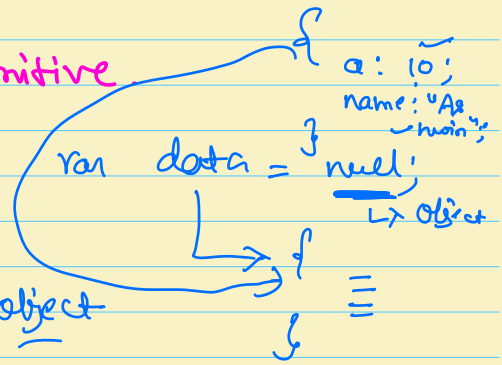
- Data types in JS

primitive → non-primitive

ES5
① number
② string
③ boolean
④ null
⑤ undefined

ES6
⑥ Bigint
⑦ Symbol

non-primitive
→ object
→ array
→ function.



- typeof operator.

typeof null ? // object.

typeof undefined ? // undefined.

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

var a;
// undefined.

- Execution Context: (EC) It's environment where JS code gets executed.
  1. GEC (Global Execution context)
  2. FEC / LEC (Function Execution context)

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.