**JavaScript**

Java Script is a synchronous, single threaded language. It is primarily used for building interactive and dynamic feature in web application.

It was created by Brendan eich in 1995, while he was working at Netscape Communication Corporation. In 1997 it was standardized as ECMAScript.

**Advantages**

**Interactivity**

**Speed or efficiency**

**Simplicity**

**Asynchronous programming**

**Versatility**

**Client-side validation**

**Cross-browser compatibility**

**Open source**

**Fast execution**

**Disadvantage**

**Cross-browser inconsistencies**

**No multithreading**

**Security concerns**

**Callback hell**

**Verbose Dom manipulation**

**Not ideal for cpu-intensive tasks**

**Lack of static typing**

**Javascript execution context**

Everything in javascript happens inside a execution context

Execution context

Memory or variable environment code or thread of execution

Key: value code execute

a: 10

fn:{}

When we run a javascript program a execution context created.

Execution context

Creation phase or memory allocation

Memory or variable environment code or thread of execution

n: undefined

square: {function}

square2: undefined

square4: undefined

Code execution pahse

Assign value to n and got to next and a function invocation found when a function invoked in javascript a new execution context is created.

Memory or variable environment code or thread of execution

n: 2 created a execution context for function for **square2**

square: {function}

square2: undefined

square4: undefined

memory code

num: undefined

ans: undefined

**Callback**

Memory or variable environment code or thread of execution

n: 2 created a execution context for function for **square2**

square: {function}

square2: 4

square4: undefined

memory code

n: 2 2\*2

ans: 4

same flow for the square4 as square2 and after the complete execution the execution context destroys.

**Call stack** maintain the order of execution of execution context.it is also known as **execution context stack, program stack, control stack, runtime stack and machine stack**.

1. 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.

We can use a variable or function in our code before its declared, as long as its within the same scope.

Function expression and variable declared with let and const are hoisted to the top of their containing block or function, just like variables declared with var, but they are not initialized with a value of **undefined** and remain in the **temporal dead zone** until they are actually declared in the code.

console.log(x); *// ReferenceError: Cannot access 'x' before initialization*

let x = 5

1. Temporal deadzone

the temporal dead zone is a concept in javascript that refers to the period between the creation of a variable declared with let or const and the actual declaration or initialization of that variable. During this period we cant access the variable.

console.log(x); *// ReferenceError: Cannot access 'x' before initialization*

let x = 5

1. Closure
2. Currying
3. Throttling and debouncing