# Scripting Language

JavaScript

# Use Case


PhoneBook
Id
Name
Number
Operations
Add Contact
Name
Phone
Save Contact
View/Update
Name
Phone
Update Contact
© Sonam Soni 2024

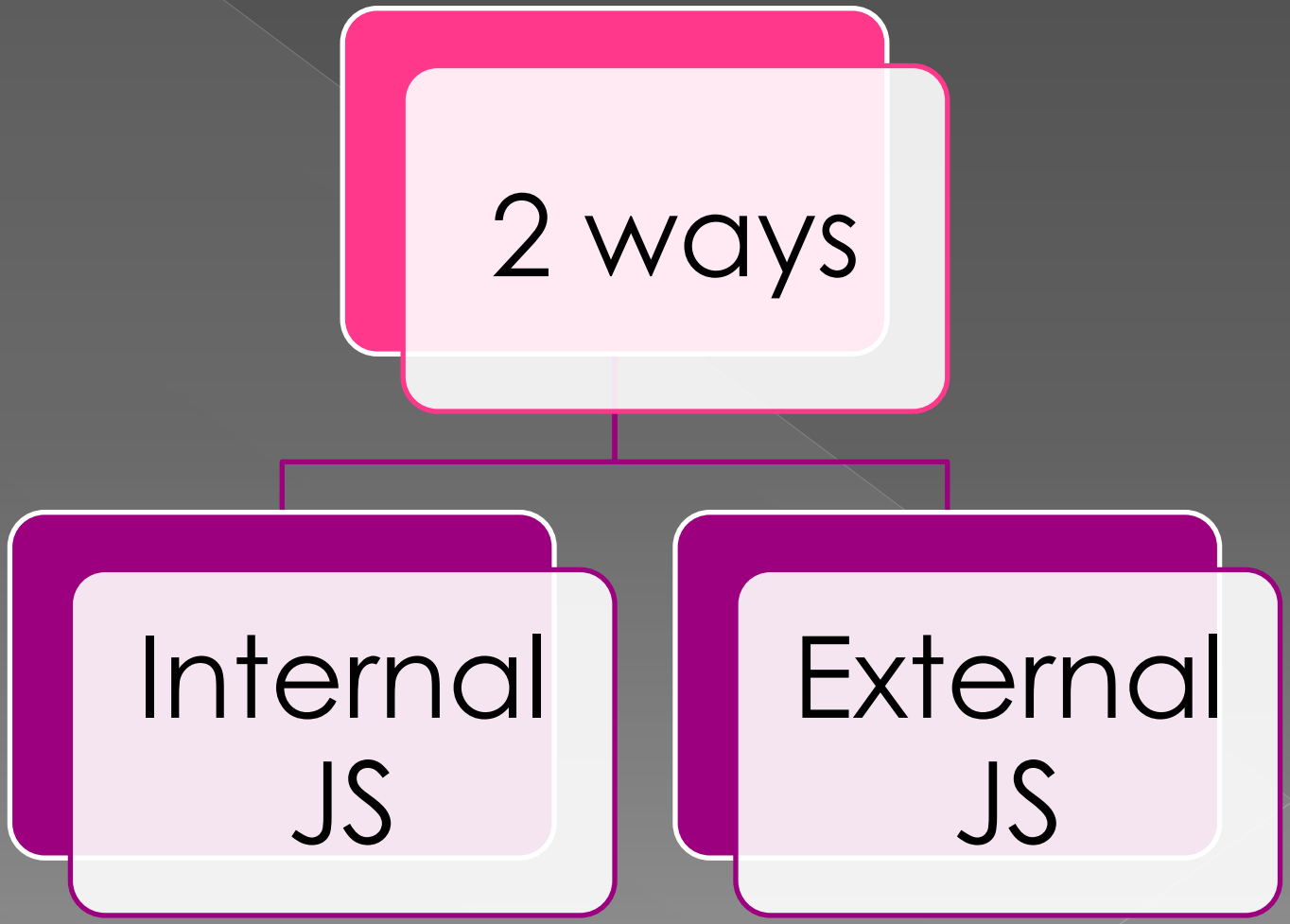
# Use Case Description

- Create one index.html page with bootstrap included and the layout of the application shown in screenshot.
- The User can add phone number, can see all added phone numbers, edit them and delete as well

# What is JavaScript

- JavaScript is a versatile programming language primarily used for adding interactivity and dynamic behavior to web pages.
- It is also called as client side scripting
- We can use JS as server side scripting as well.

# How to include JS in HTML?



# Syntax

- JavaScript syntax consists of statements, variables, operators, expressions, and comments.
- Example:
- var x = 5;
- var y = 6;
- var z = x + y;

# Variables

- Variables are used to store data values.
- var: function-scoped
- let: block-scoped
- const: block-scoped, constant values

# Datatypes

- JavaScript supports different data types:
- String
- Number
- Boolean
- Array
- Object
- Null
- Undefined

# Operators

- JavaScript operators are used to perform operations on variables and values:
- Arithmetic operators: +, -, *, /
- Assignment operators: =, +=, -=
- Comparison operators: ==, ===, !=, !==
- Logical operators: &&, ||, !

# Conditional Statements

- If statement
- If else statement
- Switch case statement
- Break & continue

# Iteration

- Repetitive task we can complete using Loops
- 3 types
  - For loop
  - Do while loop
  - While loop

# Functions

- Functions are reusable blocks of code designed to perform a particular task.
- Example:

```
function myFunction(p1, p2) {
  return p1 * p2;
}
```

# Function Declaration

- A function declaration consists of the function keyword, followed by:
  - The name of the function.
  - A list of parameters to the function, enclosed in parentheses and separated by commas.
  - The JavaScript statements that define the function, enclosed in curly brackets, { }.

```
function square(number) {
    return number * number;
}

square(10);
```

# Function Expression

- Function declares as a variables

```
var factorial = function fac(n) {
return n < 2 ? 1 : n * fac(n - 1);  };

console.log(factorial(3)); //output will be 6
```

# Variable Scoping

- When you declare a variable outside of any function, it is called a *global* variable, because it is available to any other code in the current document.
- When you declare a variable within a function, it is called a *local* variable, because it is available only within that function.

# Hands On

- Write a JavaScript function which calculate simple Interest (I = p * r * n /100)
- Write a JavaScript program which calculate area and circumference of circle.
  - A= pi * r * r
  - C= 2 * pi * r
- Create a function which display the table of 7.
- Create a function which displays the Fibonacci series for 10  numbers: 0,1,1,2,3,5,8,13,21,25

# Arrays

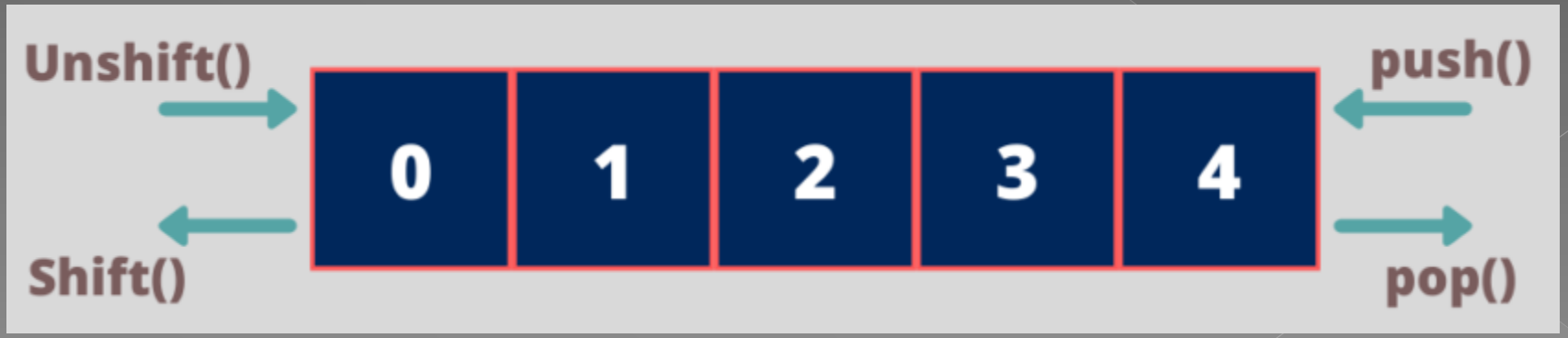
- Arrays in JavaScript are used to store multiple values in a single variable. An array is a special variable that can hold more than one value at a time.
- Array creation
- Literal:
  - let fruits = ["Apple", "Banana", "Cherry"];
- Array Constructor:
  - let fruits = new Array("Apple", "Banana", "Cherry");

# Working with Array

- Accessing Array Elements:
  - let firstFruit = fruits[0];  // Apple
  - let secondFruit = fruits[1]; // Banana
- Modifying Array Elements:
  - fruits[0] = "Mango";
- Array Properties: length
  - let numberOfFruits = fruits.length; // 3

# Array Methods

- **push()**: Adds a new element to the end of an array.
- **pop()**: Removes the last element from an array.
- **shift()**: Removes the first element from an array.
- **unshift()**: Adds a new element to the beginning of an array.



# Array Methods

- **concat()**: Joins two or more arrays.
- **slice()**: Returns selected elements in an array as a new array.
- **splice()**: Adds/removes elements from an array.
- **indexOf()**: Returns the first index at which a given element can be found in the array.
- **includes()**: Checks if an array contains a specified element.

# Activity:

- Create and array of numbers and write a logic to remove duplicate numbers from an array.
- Create js file which take numbers from the user and calculate the average of all numbers.

# Working with Strings

- Strings in JavaScript are sequences of characters used for storing and manipulating text.
- JavaScript provides several methods to work with strings effectively.
- Creating Strings:
  - let singleQuoteString = 'Hello, world!';
  - let doubleQuoteString = "Hello, world!";
  - let templateLiteralString = `Hello, world!`;
- String property:
  - length: Returns the length of a string.

# String methods

- charAt(): Returns the character at a specified index.
- indexOf():Returns the index of the first occurrence of a specified value, or -1 if not found.
- slice():Extracts a part of a string and returns it as a new string.
- substring():Similar to slice() but cannot accept negative indices.
- substr(): Similar to slice(), but the second parameter specifies the length of the extracted part.

- replace(): Replaces a specified value with another value in a string.
- toUpperCase(): Converts a string to uppercase letters.
- toLowerCase(): Converts a string to lowercase letters.
- trim(): Removes whitespace from both ends of a string.
- split(): Splits a string into an array of substrings.
- includes(): Checks if a string contains a specified value.
- startsWith(): Checks if a string starts with a specified value.
- endsWith(): Checks if a string ends with a specified value.

# Hands On

- Create an array of Strings which contains the values like [“Mind”,”SpRInT”,”Pvt”,”Ltd”] then replace the array values with corresponding Uppercase values only.

  [“MIND”,”SPRINT”,”PVT”,”LTD”]
- Write a JS program which takes input from the user in string and print all values and its length as well.
  - Sonam: 5
  - Alex : 4
  - Catherine: 9

# Objects

- Objects in JavaScript are collections of key-value pairs, where the keys (properties) are strings (or symbols) and the values can be any type of data, including other objects and functions.
- Objects are fundamental to JavaScript and are used to represent and manipulate complex data.

# Creating Objects

```
let person = {
    firstName: 'John',
    lastName: 'Doe',
    age: 30,
    isEmployed: true,
    address: {
        street: '123 Main St',
        city: 'New York',
        zip: '10001'
    },
    greet: function() {
        console.log('Hello, ' + this.firstName + ' ' + this.lastName);
    }
};
```

# Access Objects Data

- Using Dot:

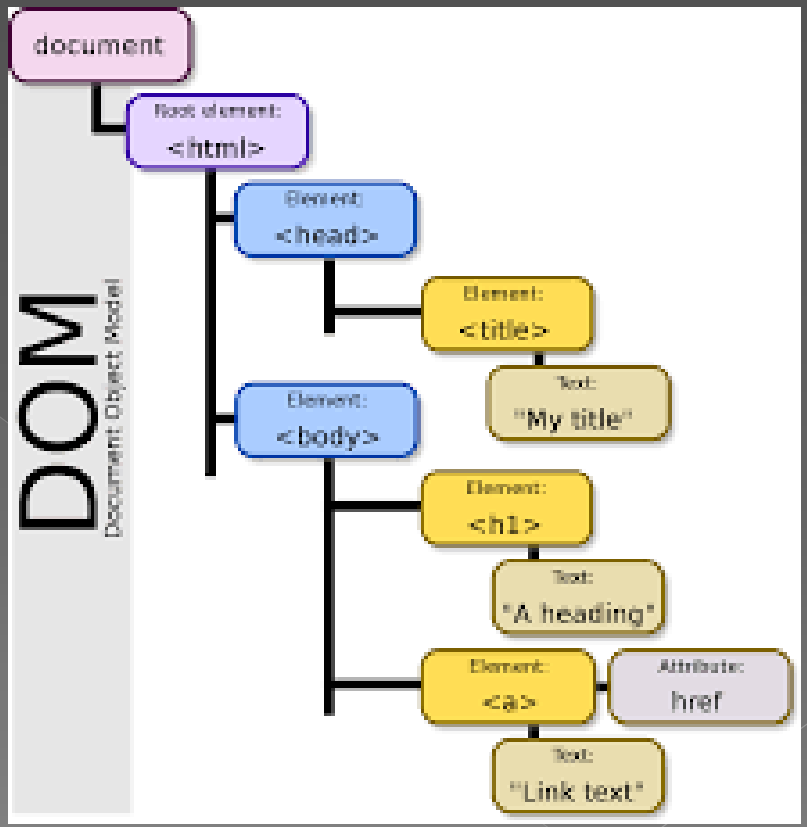
```
console.log(person.firstName); // Output: John
person.age = 31;
console.log(person.age); // Output: 31
```

- Using Brackets:

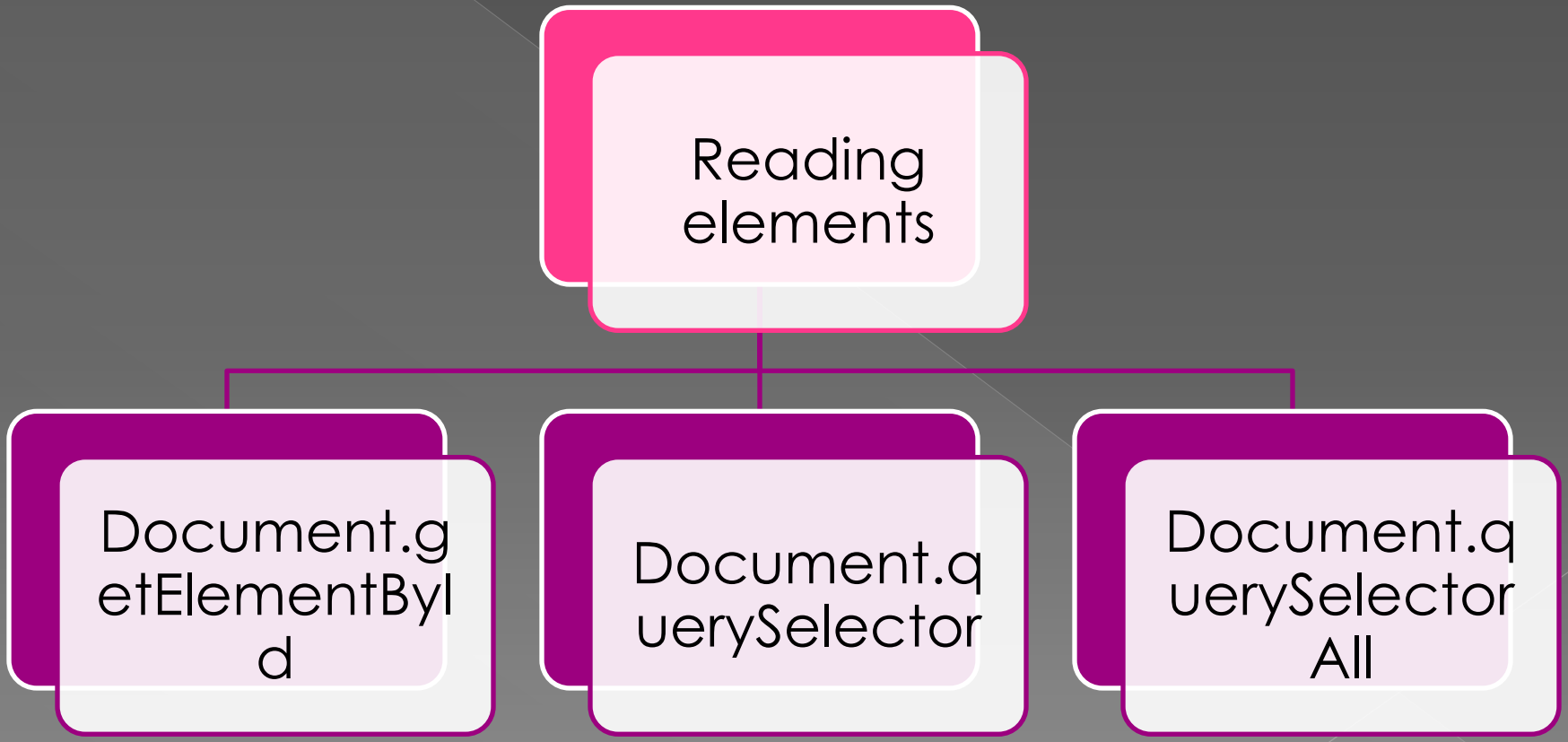
```
console.log(person['lastName']);
person['isEmployed'] = false;
console.log(person['isEmployed']);
let propName = 'age';
console.log(person[propName]);
```

# DOM

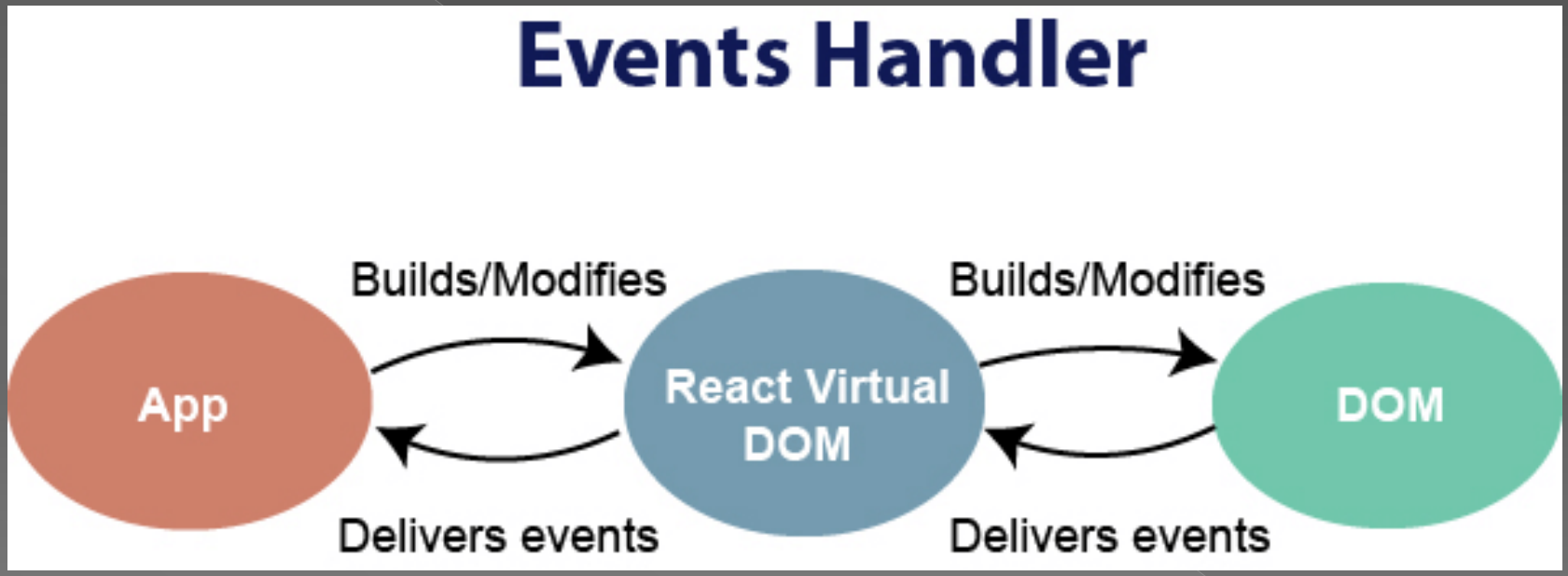
- The Document Object Model (DOM) is a cross-platform and language-independent application programming interface that treats an HTML document as a tree structure wherein each node is an object representing a part of the document.
- The DOM represents a document with a logical tree.



# DOM Manipulation



# Event Handling



# Form Validation

- Data validation using JavaScript conditional logics

# Hands on DOM and event

- Create Simple todo Application
- Add todo
- Get All todos
- Delete todo

# Advanced JS

- Advanced topics introduced in ES6
- Arrow functions
- Class & Object
- OOPs Pillars
- Map
- Set

# Arrow Functions

- How to declare:

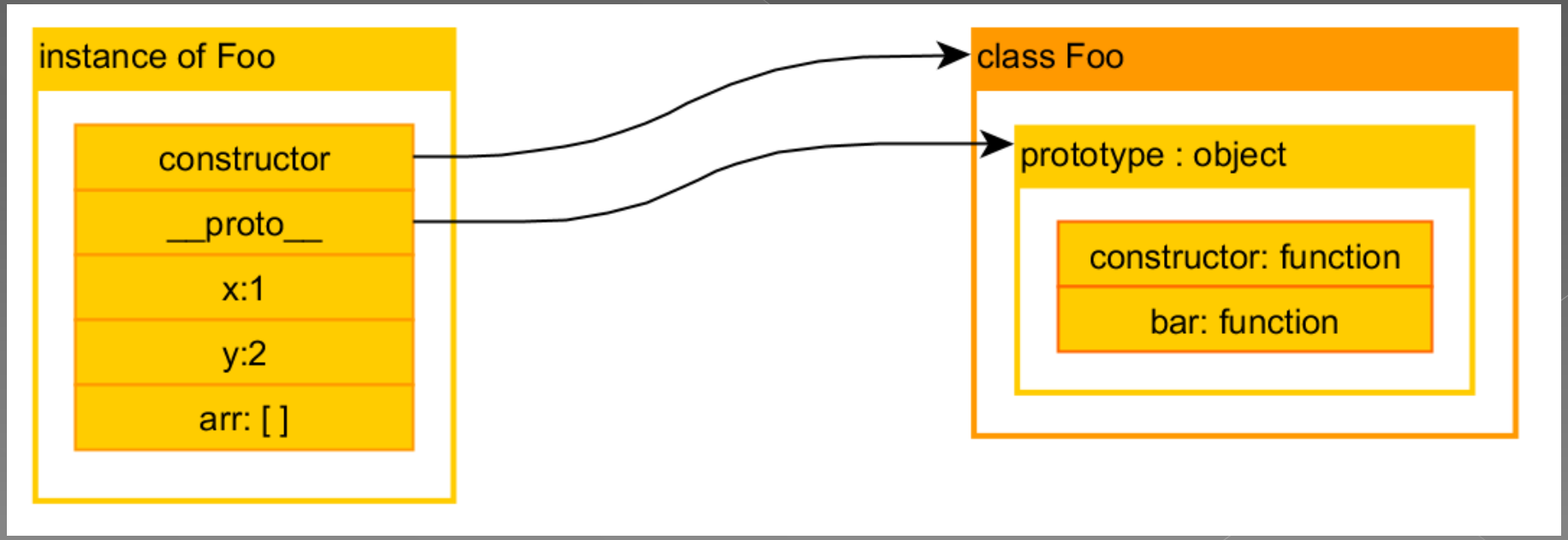
```
Const greeting = ()=>{
    console.log('Good Morning');
}
```

- With return value & parameters

```
    Const greeting = (name)=>{
    return 'welcome '+name
}
```

# Class & objects

- Earlier in Old JS when we have no options to creating class we are declaring the functionaries using functional constructor & prototypes.
- Later on when ES6 introduced we can use it by creating class and making its instances for usage.



# Activity on class & Object

- Create a class of Account which is having properties:
  - Acc_holder_name
  - Acc_no
  - balance
- Methods:
  - Diposit
  - Withdraw
  - Check balance
- Access those methods and check the output

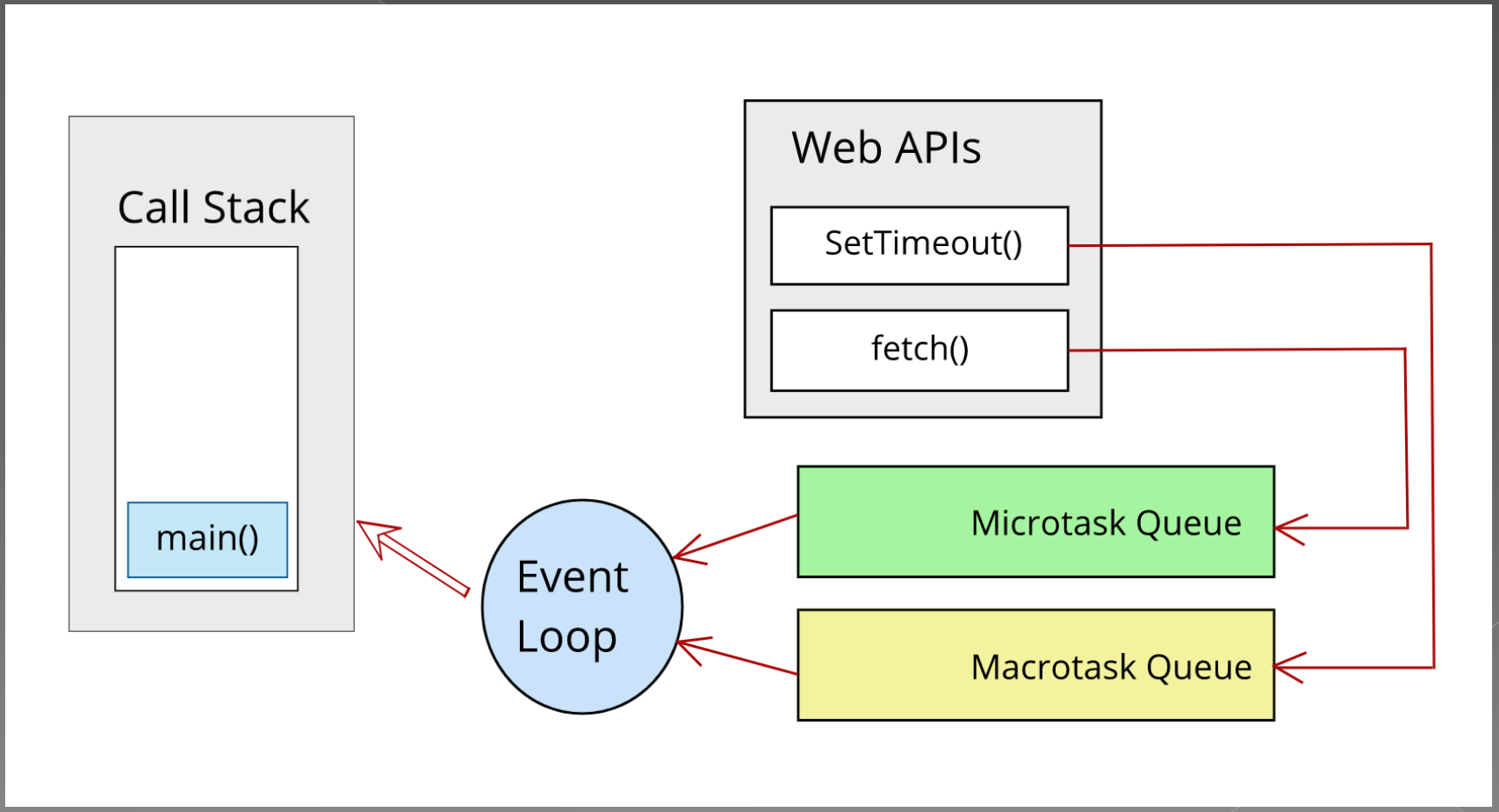
# Set & Map

- Set is an inbuilt class provided by JS which is used to create unique set of data
- No duplicates allowed.
- Map is an inbuilt class provided by JS which is used to store data in the form of key value pairs.
- In map duplicate values allowed but not duplicates keys allowed, it will override

# Asynchronous JS

- Asynchronous functionality:  one task will not wait for the other task
- JavaScript is single threaded which is non blocking
- So you can execute multiple task in parallels.

# Event Loop


Call Stack
main()
Web APIs
SetTimeout()
fetch()
Event Loop
Microtask Queue
Macrotask Queue

# Higher Order Functions

- Map()
- Finter()
- Find()
- findIndex()
- Reduce()

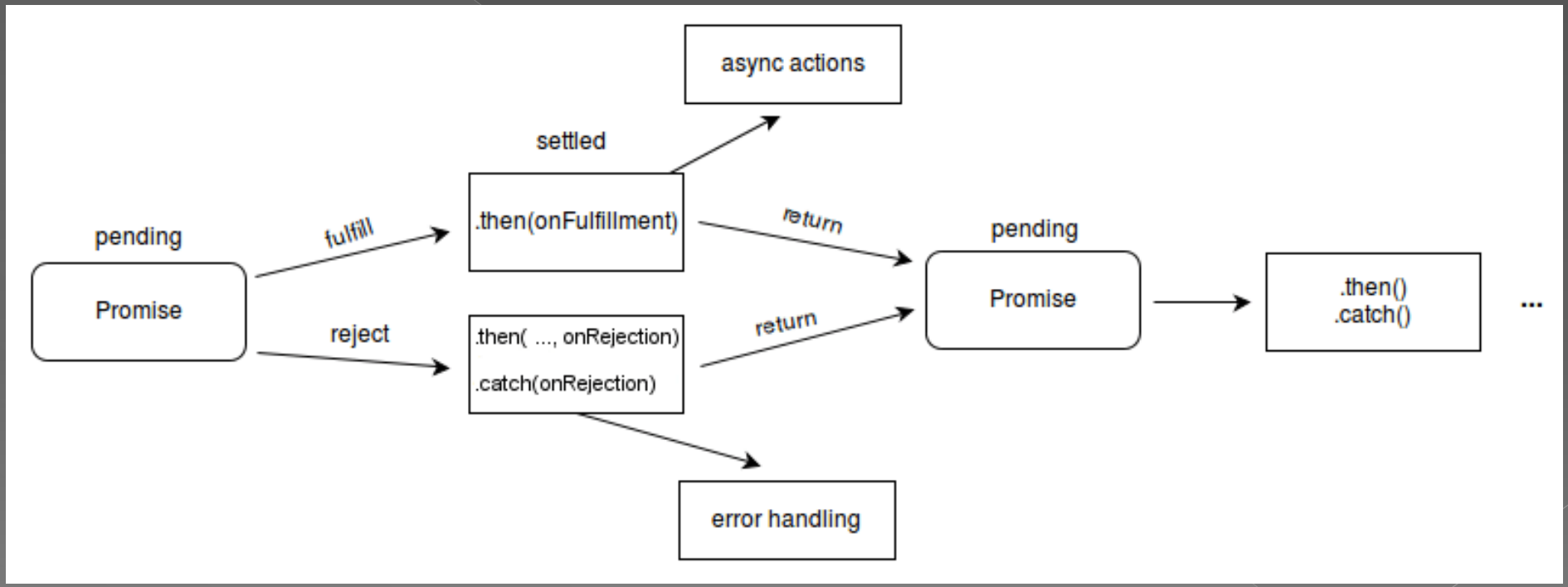
# Callbacks

- Passing functions as parameters
- Why we need callbacks?
- To manage asynchronous functionality of JavaScript.

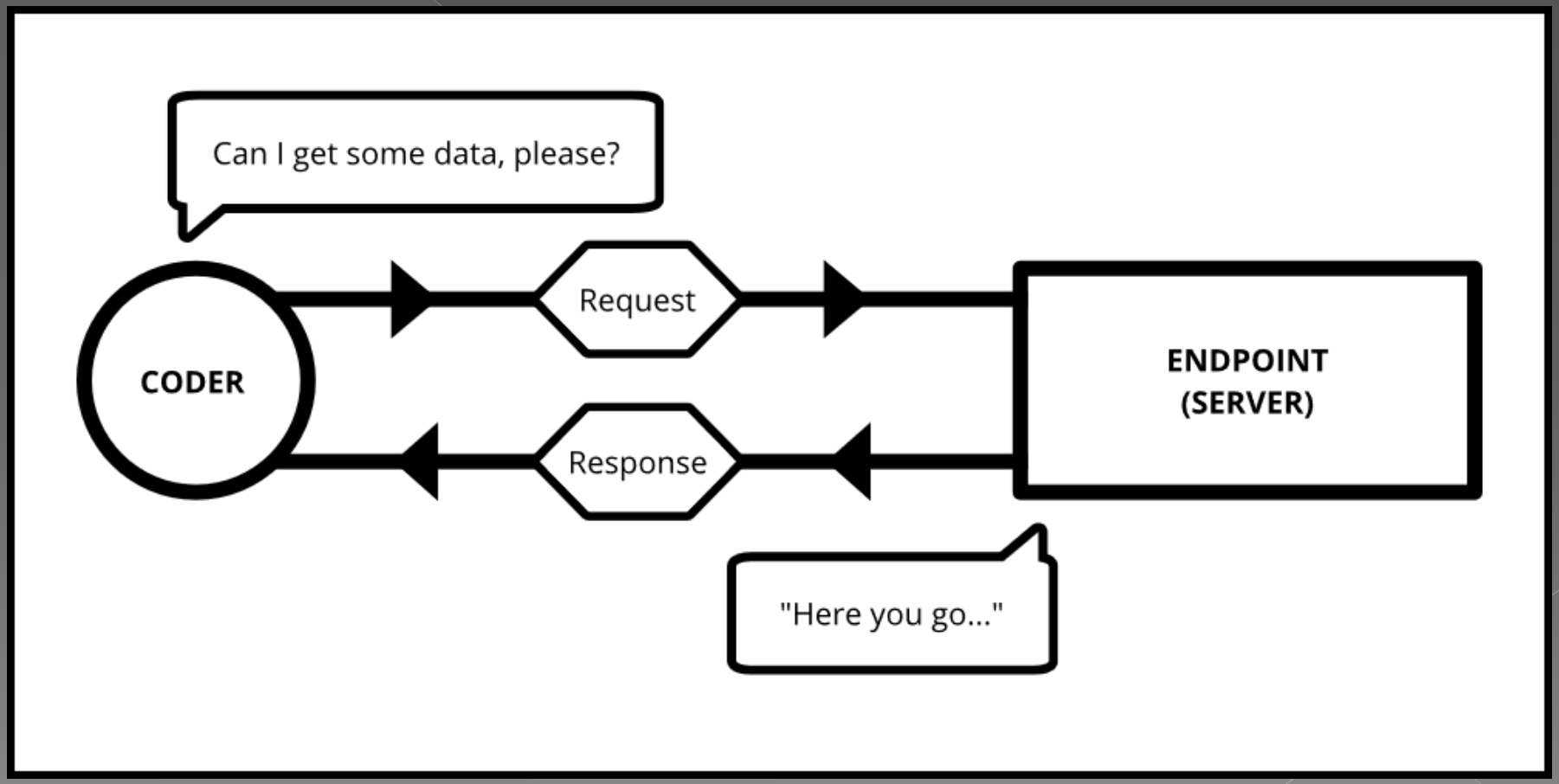
# Callback Hell

```
doSomething(param1, param2, function(err, paramx){
    doMore(paramx, function(err, result){
        insertRow(result, function(err){
            yetAnotherOperation(someparameter, function(s){
                somethingElse(function(x){
                });
            });
        });
    });
});
```

# Promises


async actions
settled
pending
fulfill
.then(onFulfillment)
return
pending
Promise
.then()
.catch()
...
reject
.then( ..., onRejection)
.catch(onRejection)
return
Promise
error handling

# Fetch API


Can I get some data, please?
Request
CODER
ENDPOINT
(SERVER)
Response
"Here you go..."

# Activity

- Fetch Data from API
- Integrate with HTML
- Represent your data using DOM

# Use Case implementation

- Implementing same Usecase using class and Objects
- Phonebook project
- Create class contact and implementing all functionalities with methods
- Utilize that class object for implementation

# Expected Output