Javascript

- Aravind

Day 2: Java script

Topics for Discussion

- Introduction to Javascript
- Data Types and operators
- Conditional statements
- Loops
- Variable declarations

Introduction to Javascript

- JavaScript is an object-based scripting language which is lightweight and cross-platform.
- JavaScript Translator (embedded in the browser) is responsible for translating the JavaScript code for the web browser.
- It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document
- With JavaScript, users can build modern web applications to **interact directly without reloading the page every time**. The traditional website uses js to provide several forms of interactivity and simplicity.

Features Of Javascript

- All popular web browsers support JavaScript as they provide built-in execution environments.
- JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.
- JavaScript is a weakly typed language, where certain types are implicitly cast (depending on the operation).
- JavaScript is an object-oriented programming language that uses **prototypes** rather than using classes for inheritance.
- It is a light-weighted and interpreted language.
- It is a **case-sensitive** language.
- JavaScript is supportable in several operating systems including, Windows, macOS, etc.
- It provides good control to the users over the web browsers.

Usage

- Within Script tag
- With Head tag
- Use of alert (pop up)
- Using External File (name.js)

Javascript Comment

- To make code easy to understand
- Avoid unnecessary code

Single line represented by → //

Multi line represented by → /**/

Datatypes and Operators

Primitive Data types

- String → "",'
- Number → 1,2,3
- Boolean → true/false
- Undefined → hoisting
- Null → no value

Non-Primitive Data types

- Object
- Array

contd

Operators:

- Addition - +
- Subtraction - -
- Multiplication - *
- Division -/
- Modulus (Remainder) - %
- Increment - ++
- Decrement - --

contd

Operator	Description	Example
==	Is equal to	10==20 = false
===	Identical (equal and of same type)	10==20 = false
!=	Not equal to	10!=20 = true
!==	Not Identical	20!==20 = false
>	Greater than	20>10 = true
>=	Greater than or equal to	20>=10 = true
<	Less than	20<10 = false
<=	Less than or equal to	20<=10 = false

Variable Declaration

- Variable declaration happens by using 3 keywords
- Name must start with a letter (a to z or A to Z), underscore(_), or dollar($) sign
- After first letter we can use digits (0 to 9), for example value1
- JavaScript variables are case sensitive, for example x and X are different variables
- Below are the **keywords**
  - Let (old way of usage)
  - Var - can be changed
  - Const - cannot be changed once declared

Loops and Conditional Statement

- If
- If else
- If else if
- Switch
- For
- For in
- While
- Do-while

Day 3 Javascript

Topics for discussion

- Arrays and Array Methods
- Js Function
- Working with objects
  - String and Numbers

Javascript Array

- Javascript array is a representation of similar elements
- How to represent
    - By array literal
    - By creating instance of Array directly (using new keyword)
    - By using an Array constructor (using new keyword)

JS Function

- Used to perform operations
- Used to reuse a block of code

Syntax:

function name(){

// block of code

}

Function - usage

- **apply()** → method is used to call a function contains this value and an argument contains elements of an array.
- **bind()** → method is used to create a new function
- **call()** → method is used to call a function contains this value and an argument provided individually.
- **toString()** → method returns a string.

Working with Objects

- object is an entity having state and behavior (properties and method)
- Syntax as below,

  object={property1:value1,property2:value2.....propertyN:valueN}

Representation of Functions

Syntax

```
() => expression

param => expression

(param) => expression

(param1, paramN) => expression

() => {
  statements
}

param => {
  statements
}

(param1, paramN) => {
  statements
}
```

Day 4

Topics to Discuss

- OOPS concept in JavaScript
- Inheritance- Types with Def and Examples,Super keyword
- Prototype in JavaScript
- Polymorphism in JavaScript

OOPS

- OOP(Object Oriented Program) is to **separate concerns and responsibilities** into **entities**
- **OOP is very useful on large scale projects, as it facilitates code modularity and organization.**
- Entities are coded as **objects**, and each entity will group a given set of information (**properties**) and actions (**methods**) that can be performed by the entity.

What is Class

- Classes are the blueprints or molds that we're going to use to create the actual objects.
- Class names are declared with a **capital first letter** and **camelCase by convention**. The class keyword creates a constant, so it cannot be redefined afterwards.
  - Example: ClassA
  - classA
- **Classes must always have a constructor** method that will later on be used to **instantiate** that class.
- **A constructor in JavaScript is just a plain old function that returns an object.**
- The **"this" keyword** points to the class itself and is used to define the class properties within the constructor method.
- Methods can be added by simply defining the function name and its execution code.

OOPS - Inheritance

- Inheritance is the ability to **create classes based on other classes**
- we can define a parent class (with certain properties and methods), and then children classes that will inherit from the parent class all the properties and methods that it has.
- we use the **extends** keyword to declare the parent class we want to inherit from.
  - Class nameA **extends** nameB
- Then on the constructor method, we have to declare the "**power**" parameter and use the "**super" function to indicate that property is declared on the parent class**.
- A class can only have **one parent class to inherit** from.
- If a child class inherits any properties from a parent class, it must first assign the parent properties calling the super() function before assigning its own properties.
- Children classes can override the parent's properties and methods.

OOPS- Encapsulation

- Encapsulation decides which information to show and which information to hide
- Encapsulation is implemented through public and private properties and methods.
- Javascript by default is always public
- Encapsulation is useful in cases where we need certain properties or methods for the inner working of the object, but we don't want to expose that to the exterior.

Javascript Prototype

- JavaScript is a prototype-based language that facilitates the objects to acquire properties and features from one another.
- Syntax is as below ,
  - ClassName.prototype.methodName
- Whenever an object is created in JavaScript, its corresponding functions are loaded into memory. So, a new copy of the function is created on each object creation

OOPS- Abstraction

- Abstraction is the process of hiding the implementation details and to show only the functionality
- Please note that an instance of Abstract Class cannot be created.
- *JavaScript Abstraction reduces the duplication of the code.*
- Use of "**instanceof**" - used to check the type of object at runtime
  - Returns true or false
  - If the returned value is true, then it indicates that the object is an instance of a particular class and if the returned value is false then it is not.
  - **Syntax :** var myVar = objectName instanceof objectType
- The `Object.create()` static method creates a new object, using an existing object as the prototype of the newly created object.
- We cannot have implementation of the function inside the abstract class

OOPS - Polymorphism

- Polymorphism is a core concept of an object-oriented paradigm that provides a way to perform a single action in different forms.
    - Overloading - compile time polymorphism
    - Overriding - runtime polymorphism
- It provides an ability to call the same method on different JavaScript objects.

Exception Handling

- Exception handling is a process or method used for handling the abnormal statements in the code and executing them
- Throw statement is used
- Try…catch block is used
- Exception Handling Statement,
    - throw statements
    - try…catch statements
        - Can have one try block and multiple catch blocks
    - try…catch…finally statements
        - Finally → whatever we write in finally that will get displayed or performed . Irrespective of the function/method output

Read and Write using JS

- **'fs'** in javascript is nothing but the module required by javascript to perform the operation
  - Fs → File System module
- fs.readFile() and fs.writeFile() methods are used to read and write the files
- fs.readFile()
  - **fs.readFile(file_name, encoding, callback_function)**
  - **filename:** It contains the filename to be read, or the whole path if the file is saved elsewhere.
  - **encoding:** It stores the file's encoding. **'utf8'** is the default setting.
  - **callback function:** This is a function that is invoked after the file has been read. It requires two inputs:
    - **err:** If there was an error.
    - **data:** The file's content.
- fs.writeFile()
  - **fs.writeFile(file_name, data, options, callback)**
  - **file_name**: It's a string, a buffer, a URL, or a file description integer that specifies the location of the file to be written. When you use a file descriptor, it will function similarly to the fs. write() method.
  - **data**: The data that will be sent to the file is a string, Buffer, TypedArray, or DataView.
  - **options:** It's a string or object that may be used to indicate optional output options. It includes three more parameters that may be selected

JSON using JS

- Read and writing json is same as text file but there few keywords
  - JSON.stringify (data) → return json format which can be written to file
  - JSON.parse(data) →takes JSON data as input and returns a new JavaScript object.

JS Map and JS Set

- JS Map - it is used to map key to values
  - Always exists as key-value pair
  - Syntax → new Map([iterable])
  - Iterable → is usually a array
- A map object cannot contain the duplicate keys.
- A map object can contain the duplicate values.
- The key and value can be of any type (allows both object and primitive values).
- A map object iterates its elements in insertion order.

Js Set

- JS Set - used to store the elements with unique values
- Syntax:
  - new Set([iterable])
  - **iterable** - It represents an iterable object whose elements will be added to the new Set.
- A set object uses the concept of keys internally.
- A set object cannot contain the duplicate values.
- A set object iterates its elements in insertion order.

JS Promise

- It provides value after a asynchronous call
- Asynchronous is nothing but the tasks can be started parallely without waiting for any
- Asynchronous helps to complete more tasks in shorter period of time
- How to handle Asynchronous
  - Use of call back
  - Using promise

Intro to Promise

- Promise is a special JavaScript object. It produces a value after an asynchronous operation completes successfully, or an error happens
  - Success → resolve
  - Failure → reject
  - Syntax
    - **let promise = new Promise(function(resolve, reject) { }**
- new Promise() will return promise as object
- Internal properties/states of promise → **state,pending,fulfilled,rejected**
- Promise used executor function
  - Uses then(), try/catch() and finally()
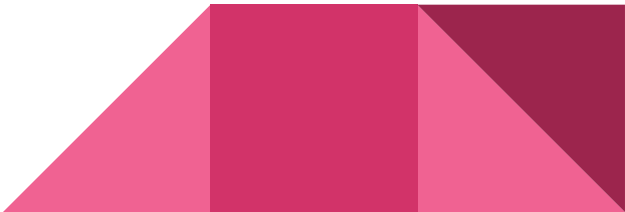
Call back

- callback is a function which is to be executed after another function has finished execution.
- Any function that is passed as an argument to another function so that it can be executed in that other function is called as a callback function.
- callback's primary purpose is to execute code in response to an event.

JS - async/await