JS Assignment Theory

01) <u>JavaScript Introduction</u>

Q1. What is JavaScript? Explain the role of JavaScript in web development.

- JavaScript is programming language.
- Javascript developed and managed by ECM (ecma) script organization.
- Javascript is most popular client side scripting language.

  examples : operator | variables | functions | array | loop | conditions

- Javascript provides modules for code reusability.
- Javascript provides functionality and fast load on web browser.
- JavaScript adds interactivity and dynamic content to web pages.
- javascript file extention is .js
- javascript provides some output methods to print content on web browser.

  examples : a) document.write()

  b) alert()

  c) prompt()

  d) confirm()

  e) window.print()

  f) console.log()

  g) document.getElementById("");

  h) document.getElementById("").value;

  i) document.getElementById("").src;

  j) document.getElementById("").style;

Jjavascript run on browser.

a) used scripting method

b) used console method

c) used browser console method

d) used node js method

e) used external javascript

Q2. How is JavaScript different from other programming languages like Python or Java?

JavaScript is different from other programming languages like Python or Java in several ways:

Scripting vs. programming: JavaScript is a scripting language, which means it's executed line by line, whereas programming languages like Java are compiled beforehand.

Dynamic typing: JavaScript is dynamically typed, meaning you don't need to declare the data type of a variable before using it. In contrast, languages like Java are statically typed.

Prototype-based: JavaScript is a prototype-based language, which means it uses prototypes to create objects. This is different from class-based languages like Java.

Functional style: JavaScript supports functional programming, which allows you to write code that's more concise and easier to read. So it's different.

Q3. Discuss the use of <script> tag in HTML. How can you link an external JavaScript file to an HTML document?

Using the script tag to include an external JavaScript file

To include an external JavaScript file, we can use the script tag with the attribute src .

script used the src attribute when using images.

The value for the src attribute should be the path to your JavaScript file.

02) <u>Variables and Data Types</u>

Q1. What are variables in JavaScript? How do you declare a variable using var, let, and const?

Varialbles used are stored a data value.

- Var : var is access globally

var is redeclare and re-assign the values

examples :   var a="Hello";

                    var a=10;

                    var b=20.56;

                    var c=154545454n;

- let: let is one type of declare any variables inside of javascript

let is used to re-assign but can not redeclare the values

```
    let a=10;
    console.log(a);
```

- const : const can not change the values on run time

const is also can not re-assign and redeclared the values

```
const a=110;
console.log(a);
```

Q2. Explain the different data types in JavaScript. Provide examples for each.

Defination: it's specify the different sizes and values that can be stored in a variable.

Type :

String Boolean Undefined Null Number Symbol Object

Some examples:

let name="raj"; //string let length = 22; //number const age=22; //number let age = false; //boolen

Q3. What is the difference between undefined and null in JavaScript?

undefined: A variable is defined when it has be declared but not assigned a value.

null: null is an assignment value that represents no value or no object.

03) <u>JavaScript Operators</u>

Q1. What are the different types of operators in JavaScript? Explain with examples.

Defination : Operators are symbols that perform specific operations on operands

Types of oprators:

1. Arithmetic Operators :

- Additiom

- Subtraction
- Multiplication
- Division
- Modulus
- Incriment ++
- Decrement --

2. Assignment Operators:

- Assignment =
- Subtract and assign -=
- Divide and assign /=
- Multiply and assign *=
- Modulus and assign %=
- Add and assign +=

3. Comparison Operators:

- Equal to ==
- Not equal to !=
- Greater than >
- Less than <
- Greater than or equal to >=
- Less than or equal to <=

4. Logical Operators:

- Logical and &&
- Logical or ||
- Logical not !

5. Bitwise Operators:

- Bitwise and &
- Bitwise or |
- Bitwise xor ^
- Bitwise not ~
- Left Shift >>
- Right shift <<

6. Ternary / condation operator ?

Examples

- Arithmetic Operators let sum = a + b; let difference = a - b; let product = a * b; let quotient = a / b; let remainder = a % b;
- Assignment Operators a += 5; // a = a + 5 b -= 2; // b = b - 2
- Comparison Operators let isEqual = a == b; let isNotEqual = a != b; let isGreaterThan = a > b;
- Logical Operators let isTrue = true && false; let isFalse = true || false; let isNotTrue = !true;
- Ternary Operator let result = a > b ? "a is greater" : "b is greater";

Q2. What is the difference between == and === in JavaScript?

The main difference between the two operators is how they compare values.

The == operator compares the values of two variables after performing type conversion if necessary.

The === operator compares the values of two variables without performing type conversion.

04) <u>Control Flow (If-Else, Switch)</u>

Q1. What is control flow in JavaScript? Explain how if-else statements work with an example.

DEFINITION : Control flow refers to the order in which statements are executed in a program.

- if-else statement :

An if-else statement is a fundamental control flow statement in programming that allows you to execute different code blocks based on a specific condition.

EXAMPLE :

let age = 18;

if (age >= 18) { console.log("You are an adult."); } else { console.log("You are a minor."); }

Q2. Describe how switch statements work in JavaScript. When should you use a switch statement instead of if-else?

switch statements are used to perform different actions based on different conditions.

Use switch to select one of many blocks of code to be executed.

You should use a switch statement instead of an if-else statement

when you want to make a multi-branch decision based on the value of a single expression

05) <u>Loops (For, While, Do-While)</u>

Q1. Explain the different types of loops in JavaScript (for, while, do-while). Provide a basic example of each.

looping is uesd to print number of again and again or repeted more then once time there we used loop.

- types of loop in javascript

  a) for

  b) nested for

  c) forIn

  d) Foreach

  e) forOf

  f) while

  g) do while

- for: for loop is executed when condition is true.

```
var i;
for(i=0;i<=10;i++){
console.log(i);
}
```

- while:while loop executed when condition is true.

```
var i;
while(i<=10)
{
  console.log(i);
  i++;
}
```

- do-while: do will be executed condition true or false while is executed while condition is true.

```
var i=0;
do{
  console.log(i);
  i++
}
while(i<=10);
```

Q2. What is the difference between a while loop and a do-while loop?

- While Loop:

The condition is checked before the loop executes.

If the condition is false at the beginning, the loop might not execute at all.

```
while (condition) {
  code
}
```

- Do-While Loop:

The condition is checked after the loop executes, ensuring that the loop will run at least once.

Even if the condition is false, the loop body will be executed at least once.

```
do {
  code
} while (condition);
```

06) <u>Functions</u>

Q1. What are functions in JavaScript? Explain the syntax for declaring and calling a function.

A function is a block of code i.e called functions or A function is a block of code i.e used to complete any task i.e called function

A function is re-usables.

Syntax : How to declare a function

```
 function functionname()
 {
   statements;
 }
 functon call
```

Q2. What is the difference between a function declaration and a function expression?

- FUNCTION DECLARATION :

hoisted, meaning they can be used before they are declared in the code.

always have a name.

- FUNCATION EXPRESSION :

Not hoisted, meaning they must be declared before they are used.

without a name or named.

Q3. Discuss the concept of parameters and return values in functions.

Parameters are used to pass data into the function. They are part of the function's signature.

Return values are used to send data out of the function.

The function execution result is typically captured by the caller.

07) <u>Arrays</u>

Q1. What is an array in JavaScript? How do you declare and initialize an array?

array is used to stored multiple data in a single elements or variables there we used array.

examples : key =>[value];

0=>["APPLE"];

var emp=["APPLE","BANANA","GRAPES"];

To declare an array with literal notation you just define a new array using empty brackets. It looks like this:

let myArray = [];

To initialize an array, you can declare the array and assign values to it at the time of creation.

Q2. Explain the methods push(), pop(), shift(), and unshift() used in arrays.

- push() method:

The push() method in JavaScript is used to add one or more elements to the end of an array.

It modifies the original array and returns the new length of the array.

array.push(element1, element2, ..., elementN)

- pop() method:

The pop() method in JavaScript is used to remove the last element from an array and return that element.

It modifies the original array by reducing its length by one.

array.pop();

- shift() method:

The shift() method in JavaScript is used to remove the first element from an array and return that element.

This method changes the original array by reducing its length by one.

array.shift()

- unshift() method:

The unshift method in JavaScript adds one or more elements to the starting of an array and

returns the new length of the array.

array.unshift(element1, element2, ..., elementN)

08) <u>Objects</u>

Q1. What is an object in JavaScript? How are objects different from arrays?

DEFINITION : An object is instances or examples of class

DIFFERENT FROM ARRAYS :

Arrays often store elements of the same data type, whereas objects can store values of different data types associated with their keys.

Arrays are ordered lists of elements accessed by numerical indices, while objects are unordered collections of key-value pairs accessed by unique keys.

Q2. Explain how to access and update object properties using dot notation and bracket notation.

- dot notation

Use the dot (.) operator followed by the property name.

Dot notation is cleaner and more readable but requires the property name to be a valid JavaScript identifier

object.property

- bracket notation

Use square brackets ([]) with the property name as a string.

Bracket notation is more versatile and allows access to properties with names that are not valid identifiers

object['property']

09) <u>JavaScript Events</u>

Q1. What are JavaScript events? Explain the role of event listeners.

A JavaScript event is a specific action that occurs within a web page or application,

JavaScript's event listener function allows you to create custom responses to events like mouse clicks, keyboard clicks,

and window resizing.

An event listener is a function in JavaScript that waits for an event to occur then responds to it.

- javascript event:

a)mouse event

b)keyboard event

c)window event

Q2. How does the addEventListener() method work in JavaScript? Provide anexample.

```
<body>
   <p id="demo"></p>
   <button type="button" id="btn">Print Name ?</button>
   <script>
    var btn=document.querySelector("#btn");
    btn.addEventListener("click",function(){
      var nm=prompt('Enter your Name ?');
      document.getElementById("demo").innerHTML="My name is :"+nm;
    })
   </script>
</body>
```

10) <u>DOM Manipulation</u>

Q1. What is the DOM (Document Object Model) in JavaScript? How does JavaScript interact with the DOM?

DOM is a programming interface for web documents. It represents the structure of a web page as a tree of nodes, where each node represents a part of the document, such as an element, attribute, or text.

JAVASCPRIT INTERFACES WITH DOM :

document.getElementById('id'): Selects an element by its id attribute.

document.getElementsByClassName('class'): Selects elements by their class attribute.

document.querySelector(): Selects the first matching element based on a CSS selector.

document.querySelectorAll(): Selects all matching elements.

Q2. Explain the methods getElementById(), getElementsByClassName(), and querySelector() used to select elements from the DOM

document.getElementById('id'): Selects an element by its id attribute.

document.getElementsByClassName('class'): Selects elements by their class attribute.

document.querySelector(): Selects the first matching element based on a CSS selector.

11) <u>JavaScript Timing Events (setTimeout, setInterval)</u>

Q1. Explain the setTimeout() and setInterval() functions in JavaScript. How are theyused for timing events?

Timing events is you to execute code at specific intervals or after a certain delay. This is achieved using the setTimeout() and setInterval() functions.

- setTimeout()

setTimeout(function, milliseconds) Executes a function, after waiting a specified number of milliseconds.

- setInterval

setInterval(function, milliseconds) Same as setTimeout(), but repeats the execution of the function continuously.

Q2. Provide an example of how to use setTimeout() to delay an action by 2 seconds.

```
setTimeout(function() {
   console.log("This message is displayed after 2 seconds.");}, 2000);  // 2000 milliseconds = 2 seconds
```

12) <u>JavaScript Error Handling</u>

Q1. What is error handling in JavaScript? Explain the try, catch, and finally blockswith an example.

Error handling in JavaScript is a process to detect and handle errors that occurs during the execution of a program

try : encloses code that might throw an error. catch : executes if an error occurs within the try block. finally blocks : executes regardless of whether an error occurred.

try { // Code that might throw an error } catch (error) { // Handle the error here } finally { // Code that always executes, regardless of an error }

Q2. Why is error handling important in JavaScript applications?

exception handling is required for building robust, reliable and user friendly applications in JavaScript.

errors can occur due to programming mistakes, incorrect user input, etc.

Errors can disrupt code execution and lead to bad user experience.