Module 8) javasceipt

Javascript introduction

theory assignment

* question 1 : what is javascript ? explain the role of javascript in web development .
* Javascript is a high-level programming language used to create interactive and dynamic web pages. It enhances websites by enabling animations, form validation, dynamic content updates, and user interaction
* interactive: adds interactivity to web pages (like clicks and form actions)
* dynamic content:updates content dynamically without reloading the page.
* Animations & Multimedia:creates animations and handles multimedia (videos, audio)
* from validation:validates user input on the client side to improve experience
* Frontend & Backend:Used in front-end to build interfaces and in back-end with node.js
* Frameworks:Supports frameworks like react and angular for faster development
* question 2:how is javascript difference different from other programming languages like python of java
* javascript differs from other programming languages like python or java in several key aspects
* Execution environment:javascript mainly runs in web browsers for interactive websites but can run on servers with node.js. python is a general-purpose interpreted language used for web, data science, and automation. java is a compiled language running on the java virtual machine (jvm), often used in large applications and android development.
* typing::javascript and python are dynamically typed (types checked at runtime), offering flexibility. java is statically typed, requiring explicit type declarations, which helps catch errors early.
* object model:javascript uses prototype-based inheritance, where objects inherit directly from other objects. python and java use class-based inheritance, where objects are instances of classes.
* Concurrency:
* javascript uses a single-threaded, event-driven model suited for non-blocking operations. python supports multi-threading but has limitations due to the Global Interpreter Lock (gil). java supports true multi-threading with robust concurrency features.
* question 3: discuss the use of <script>tag in html.how can you link an external javascript file to an html document?
* to include an external javascript file we can use the script tag with the attribute src .you’ve already used the src attribute when using images.the value for the src attribute should be the path to your javascript file.this script tag should be included between the <head> tags in your html document.

variables and data types

* theory assignment
* question 1:what are variables in javascript?how do you declare a variable using valet, and const?
* in javascript, variables are named containers used to store data values.they allow you to hold and manipulate information within your program.
* you can declare variables using three keyboards : valet and const.
* declaring variables:

Declaring variables:

* var : declares a variables with function of global scope.it allows re-declaration and updates within the same scope.
* var var=“hello”;
* let : declares a block-scoped variables .it allows updates but not re-declaration within the same block.
* let let=10;
* const:declares a block-scoped constant, meaning its value cannot be reassigned after its initial assignment.
* const const=true;
* question 2: explain the different data types in javascript.provide example for each.
* javascript has several data types categorised primarily into primitive and non-primitive types.
* primitive data types
* these data types represent single simple value and are immutable
* cannot be changed after creation
* number : represent both integer and floating-point numbers.
* let age=30;
* let price=10;
* string:represent textual data
* let name=“vivek”;
* boolean:represents a logical entity and can only have two values true of false
* let isactive=true;
* let haspermission=false;
* undefined:represents a variable that has been declared but has not yet been assigned a value.
* let city;
* null:represents the intentional absence of any object value.
* let car=null;
* symbol:used to create unique identifiers.
* let id=symbol(“uniqueid”);
* bigint:represent whole numbers larger than the number type can handle.
* let largenumber=98765432112345n;
* non-primitive data type:these data types are used to store collection of data and more complex entities.
* object: a collection of key-value pairs.array and function are special types of objects.
* let person={
* firstname:”Vivek”,
* lastname:”panchal”,
* age:20
* }
* let colors=[“pink”,”white”,”yellow”];
* function greet(){
* console.log(“hello!”);
* }
* what is the difference between undefined and null in javascript?
* undefined indicates a variable hasn’t been initialised , while null is intentionally assigned to indicate no value.understanding the distinction helps write cleaner ,more predictable code in javascript especially when handling default values or checking for missing data.

javascript operators

theory assignment

* question 1: what are the different type of operators in javascript?explain with examples.
* javascript utilises various types of operators to perform actions on values and variables.
* arithmetic operators : perform mathematical calculations.
* let a = 10;
* let b = 5;
* console.log(a+b);
* console.log(a-b);
* console.log(a\*b);
* console.log(a/b);
* console.log(a%b);
* 2.assignment operators : assign value to variables.
* let x = 10;
* x += 5;
* x -= 3;
* 3.comparison operators:compare two values and return a boolean
* let p = 10;
* let q = 5;
* console.log(p == q);
* console.log(p != q);
* console.log(p > q);
* console.log(p === “10”);
* 4.logical operators:combine boolean expressions.
* let isadult = true;
* let haslicence = false;
* console.log(is adult && haslicense);
* console.log(isadult || has license);
* console.log(!isadult);
* question 2:what is the difference between == and === in javascript?
* the == operator compares the value of two variables after performing type conversion if necessary .on the other hand the === operator compares the values of two variables without performing type conversion.

control flow (if-else,swith)

theory assignment

* question 1:what is control flow in javascript?explain how if-else statements work with an example.
* control flow in javascript refers to the order in which statements are executed within a program.by default code runs sequentially from top to bottom but control flow statements such as conditions and loops alter this default order based on specific conditions or for repetitive task
* how if-else works:
* if statements:
* evaluates a condition enclosed in parentheses().if the condition evaluates to true the code block immediately following the if statement ,enclosed by curly braces{} is executed.
* else statement optional :
* if the condition in the if statement evaluates to false ,the code block immediately following the else statement also enclosed in curly braces {} is executed
* let age = 20;
* if (age>=18){
* console.log (“you are an adult.”);
* }
* else{
* console.log(“you are a minor.”);
* }
* let temperature=15;
* if(temperature>25){
* console.log(“it’s hot outside.”);
* }
* else{
* console.log(“it’s hot out outside.”);
* }
* question 2:describe how switch statements work in javascript.when should you use a switch sataement instead of if-else?
* in javascript a switch statement evaluates an expression and then compre its value against a series of case clauses using strict equality (===).if a match is found the code block associated with that case is executed .
* the break keyword is typically used at the end of each case to exit the switch block and prevent “fall-through”to subsequent case clauses.if no case matches the optional default clauses code block is executed.
* when to use switch instead of if-else:
* use switch when you are comparing a single expression against multiple fixed ,discrete value, improving readability and organisation especially with many condition .if else is more suitable for complex conditional logic involving multiple expressions, range boolean evaluations.

Loops (for, whiled-while)

theory assignment

* question 1:explain the difference types of loops in javascript (for, while,do-while).provide a basic example of each.
* javascript offers several types of loops for repeating code execution.the primary types are for, whiled-while.

for loop:

the for loop is used when the number of iterations is known or can be easily determined. It consists of three optional expressions: initialization, condition, and increment/decrement.

for (let i = 0; i < 3; i++){

console.log(i); // Outputs: 0, 1, 2

}

while loop:

the while loop executes a block of code repeatedly as long as a specified condition remains true. the condition is evaluated before each iteration

let count = 0;

while (count < 3) {

console.log(count);

count++;

}

do-while loop:

the do-while loop is similar to the while loop, but it guarantees that the code block will execute at least once, because the condition is evaluated after the first iteration

let num = 0;

do {

console.log(num);

num++;

} while (num < 0);

question 2:What is the difference between a while loop and a do-while loop?

the fundamental difference between a while loop and a do-while loop lies in when the loop's condition is evaluated:

* -while loop
* -while loop is entry control loop
* The condition is checked before the loop body is executed
* If the condition is initially false, the loop body will never execute
* Do while loop
* Do while loop is exit control loop
* The loop body is executed at least once before the condition is checked.
* After the first execution, the condition is evaluated. If it's true, the loop continues; otherwise, it terminates.
* This guarantees that the code within the do block will run at least one time, even if the condition is initially false

function

theory assignment

* question 1:what are functions in javascript?explain the syntax for declaring and calling a function
* in JavaScript, a function is a reusable block of code designed to perform a specific task. functions allow for code organization, modularity, and efficiency by preventing code repetition
* Syntax for declaring function
* A function declaration begins with the function keyword, follow by the function's name, a set of parentheses () which may contain parameters, and finally, curly braces {} enclosing the function's body (the code to be executed)
* function functionName(parameter1, parameter2) {
* return result;
* }
* function: Keyword indicating a function declaration.
* functionName: A unique name for your function.
* (parameter1, parameter2): Optional parameters (input values) the function can accept, separated by commas.
* {}: Curly braces enclosing the function's code block.
* return result;: Optional statement to send a value back from the function
* Calling a function
* functionName(argument1, argument2);
* functionName: The name of the function to be called
* (argument1, argument2): Values passed to the function's parameters, matching their order
* function greet(name) {
* console.log("Hello, " + name + "!");
* }
* greet("Alice");
* greet("Bob");
* question 2:what is the difference between a function declaration and a function expression?
* the difference between a function declaration and a function expression primarily lies in their syntax, how they are parsed by the JavaScript engine, and their behavior regarding hoisting
* function Declaration:
* A function declaration is a standalone statement defined using the function keyword followed by the function name, parameters, and the function body
* function greet(name) {
* return "Hello, " + name + "!";
* }
* hoisting
* function declarations are hoisted, meaning they are moved to the top of their containing scope during the compilation phase. this allows you to call a function declared this way even before its definition appears in the code
* function expression:
* a function expression defines a function as part of a larger expression, often by assigning it to a variable. It can be named or anonymous
* const sayHello = function(name) {
* return "Hi, " + name + "!";
* };
* const calculateSum = function sum(a, b) {
* return a + b;
* };
* function expressions are not hoisted in the same way function declarations are. While the variable itself might be hoisted (e.g., sayHello is hoisted), its assignment to the function value is not. therefore, you cannot call a function expression before the line where it is defined in the code. attempting to do so will result in a referenceerror
* Question 3:discuss the concept of parameters and return values in functions
* parameters are input values passed to a function when it's called, while a return value is the output that a function sends back to where it was called.Think of parameters as ingredients you give to a recipe (function) and the return value as the finished dish
* Parameters:
  + - Variables declared in a function's definition that receive values when the function is called.
* Allow functions to work with different data without needing separate functions for each input.
  + - In a function add(x, y), x and y are parameters
* return Values:
  + the value a function sends back to the calling code after completing its task.
  + enables functions to produce results that can be used in other parts of the program.
  + in the add(x, y) function, the sum of x and y could be the return value.
  + the return statement is used to specify the value to be returned

array

theory assignment

* question 1:what is an array in javascript?how do you declare and initilize an array
* in JavaScript, an array is an ordered collection of elements, which can be of various data types (numbers, strings, booleans, objects, etc.). arrays are used to store multiple values in a single variable, making it easier to manage and manipulate related data.
* there are two primary ways to declare and initialize an array in javascript:
* array literal notation
* this is the most common and generally preferred method due to its conciseness and readability
* let myArray = [];
* let fruits = ["Apple", "Banana", "Orange"];
* let numbers = [1, 2, 3, 4, 5];
* let mixedData = ["Text", 123, true, { key: "value" }];
* array constructor
* this method uses the Array() constructor**Array Constructor:** This method uses the Array() constructor
* let myArray = new Array();
* let colors = new Array("Red", "Green", "Blue");
* let emptyArrayWithLength = new Array(5); // Creates an array with 5 empty slots
* question 2:explain the methods push(),pop(),shift(),and unshift()used in array
* the push(), pop(), shift(), and unshift() methods are fundamental array manipulation methods in JavaScript, each performing a specific operation on the array's elements
* push(): adds one or more elements to the end of an array and returns the new length of the array
* let arr = [1, 2];
* arr.push(3);
* pop(): removes the last element from an array and returns that removed element. if the array is empty, it returns undefined
* let arr = [1, 2, 3];
* let removed = arr.pop();
* shift(): removes the first element from an array and returns that removed element. if the array is empty, it returns undefined
* let arr = [1, 2, 3];
* let removed = arr.shift();
* unshift(): adds one or more elements to the beginning of an array and returns the new length of the array
* let arr = [2, 3];
* arr.unshift(1);

object

theory assignment

* question 1:what is an object in javascript?how are object different from arrays
* in javascript, an object is a collection of key-value pairs, where keys are typically strings (or Symbols) and values can be any data type, including other objects, functions, or primitive values. objects are used to represent entities with properties or attributes, similar to how real-world objects have characteristics
* arrays, on the other hand, are a special type of object used to store ordered collections of data. they use numerical indices (starting from 0)to access their elements
* question 2:explain how to access and update object properties using dot notation and bracket notation
* you simply do object. key to read/modify an existing property or to add a new property. bracket notation is useful when you want to dynamically access a property. the key of this property could come from expressions like getKey() , "my" + "key" , or keyvariable

javascript events

theory assignment

* question 1:what are javascript events?explain the role of event listeners
* javascript events are actions or occurrences that happen in a web browser, typically initiated by user interactions or by the browser itself. these events serve as triggers for executing specific javascript code, enabling dynamic and interactive web applications. examples include a user clicking a button, pressing a key, submitting a form, or a page finishing its loading process.
* the role of event listeners:
* event listeners are crucial for handling these events. an event listener is javascript function that "listens" for a specific event to occur on a particular html element and, upon detection, executes a predefined function (known as an event handler or callback function) in response
* the addeventlistener method is used to attach listeners. It takes:
* The event type(e.g,”click”)
* A callback function to run when the event happens
* Optional settings (like capture)
* question 2:how does the addeventlistener ()method work in javascript?provide an example
* the addEventListener() method in JavaScript allows you to attach an event handler to a specified html element. It "listens" for a particular event to occur on that element and, when the event happens, executes a provided function. this method is preferred over older event handling techniques (like onclick attributes) because it allows multiple event handlers for the same event on a single element without overwriting previous ones.
* how it works:
* select the element:
* you first need to get a reference to the html element you want to attach the listener to, typically using document.getElementById(), document.querySelector(), or similar methods
* specify the event type:.
* you provide a string representing the type of event to listen for (e.g., "click", "mouseover", "keydown").
* define the handler function:
  + you provide a function that will be executed when the specified event occurs. this function is often referred to as the "callback function" or "event handler."
* Optional useCapture parameter:  
  A third, optional boolean parameter (useCapture) determines whether the event is handled during the capturing phase (true) or the bubbling phase (false, default)

Dom manipulation

theory assignment

* question 1:what is the dom(document object model)in javascript?how does javascript interact with the dom
* the document object model (dom) is a programming interface for web documents. it represents the page so that programs can change the document structure, style, and content. the dom represents the document as nodes and objects; that way, programming languages can interact with the page
* question 2:explain the methods getelementbyid(),getelementsbyclassname(),and queryselectors() used to select elements from the dom
* getElementById() selects a single element by its unique id.getElementsByClassName() selects all elements with a specific class name, returning a collection. querySelector() selects the first element matching a css selector, offering more flexibility.
* Detailed explanation:
* getelementbyid():
* this method is used to find a single html element based on its unique id attribute.it returns the element object directly. if no element matches the id, it returns null.because ids should be unique, this is a very efficient way to access a specific element.
* getelementsbyclassname():
* this method selects all html elements that have a specific class name. it returns an htmlcollection, which is a live collection of elements. you can iterate through the collection to access each element. if no elements match the class name, it returns an empty htmlcollection.
* queryselector():
* this method is more versatile. it selects the first element that matches a css selector.this allows you to select elements based on various criteria, including id, class, tag name, attributes, and more. It returns the element object or null if no match is found

javascript timing events(settimeout,setinterval)

theory assignment

* question 1:explain the settimeout() and setinterval () function in javascript.how are they used for timing events
* setTimeout() and setInterval() in javascript are functions used for scheduling code execution with delays.setTimeout() executes a function once after a specified delay, while setInterval() repeatedly executes a function at a fixed interval. both functions accept a callback function and a delay (in milliseconds) as arguments.
* setTimeout()
* purpose:
  + Executes a function after a specified delay.
  + syntax:
* setTimeout(function, delayInMilliseconds);
* setInterval()
* purpose :
  + repeatedly executes a function at a fixed interval.
  + syntax:
* setInterval(function, delayInMilliseconds);
* question 2:provide an example of how to use settimeout()to delay an action by 2 seconds
* the setTimeout() function in javascript can be used to delay the execution of a function or a block of code by a specified number of milliseconds
* function delayedAction() {
* console.log("This message appears after a 2-second delay.");
* }
* setTimeout(delayedAction, 2000);
* console.log("This message appears immediately.");

javascript error handling

theory assignment

* + question :1 what is error handling in javascript?explain the try,catch and finally blocks with an example
* error handling in javascript is the process of managing unexpected events or errors that occur during the execution of a script, preventing the program from crashing and allowing for graceful recovery or informative feedback.
* The try ,catch,finally blocks are used for this purpose:
* try block:
* contains the code that might potentially throw an error. If an error occurs within this block, the execution immediately jumps to the catch block.
* catch block:
* executes if an error is thrown in the try block. It receives an error object as an argument, providing details about the error. this is where you implement your error-handling logic (e.g., logging the error, displaying a message to the user).
* finally block:
* executes after the try and catch blocks, regardless of whether an error or was caught. it is typically used for cleanup operations, such as closing files or releasing resources, ensuring these actions happen no matter what
* try {
* let result = 10 / 0; // This will cause a DivisionByZero error in some contexts, or Infinity in JS
* console.log(result);
* } catch (error) {
* console.error("An error occurred:", error.message);
* } finally {
* console.log("Execution complete, cleanup performed if necessary.");
* }
* question 2:why is error handling important in javascript applications
* error handling is essential in javascript because web applications often deal with unpredictable situations — like user input errors, network failures, or bugs in logic. without proper handling, a single error can break the entire app or confuse the user. , it's not just about catching bugs — it's about building trust. A well-handled error shows that the developer anticipated problems and cared enough to manage them gracefully. it makes the app feel robust**,** professional, and user-friendly, which is exactly what good software should be.