USN

Internal Assessment Test 1 – March 2025

Sub:	**Full Stack Development**				Sub Code:	BIS601	Branch:	ISE	
Date:	26/03/2025	Duration:	90 min’s	Max Marks:	50	Sem/Sec:	V I/ A, B, C		OBE

	<u>Answer any FIVE FULL Questions</u>	MARKS	CO	RBT
1 a)	Explain difference between primitive and reference data types in JavaScript? Provide detailed examples to explain how primitive types behave differently from reference types when assigned to new variables or passed to functions.	5M	CO1	L2
1 b)	Evaluate the closures in JavaScript, and how do they relate to functions and scopes.	5M	CO1	L2
2 a)	Explain the differences between `==` and `===` operators in JavaScript, and why is one preferred over the other in most cases?	5M	CO1	L2
2 b)	Describe the difference between a function declaration, function expression, and arrow function in JavaScript. Provide examples	5M	CO1	L2
3 a)	Evaluate "this" keyword in JavaScript, and how is its value determined in different contexts? Give examples.	5M	CO1	L2
3 b)	Explain the behavior of loops like `for` and `while` when they interact with asynchronous operations (e.g., promises and `setTimeout`), and how you can avoid common issues like closures in loops or accessing incorrect values.	5M	CO1	L2
4 a)	Evaluate difference between `getElementById()` and `querySelector()` in DOM manipulation? Provide examples	5M	CO2	L2
4 b)	What is event bubbling and event capturing in JavaScript? Explain the concepts of event propagation, including the phases of capturing, target, and bubbling. How does this affect event handling in the DOM.	5M	CO2	L2
5 a)	Explain how you enhance a form using JavaScript for better user experience and validation?	5M	CO2	L2
5 b)	Explain different types of form validations such as required field validation, email format validation, password strength validation, and real-time validation using event listeners. Provide examples of implementing these validations in JavaScript.	5M	CO2	L2
6 a)	Apply Document Object Model (DOM) on event delegation, and explain how it represents an HTML document in JavaScript? Give example.	5M	CO2	L3
6 b)	Create the DOM as a tree structure, with nodes representing HTML elements, attributes, and text. How can JavaScript interact with the DOM to manipulate the webpage?	5M	CO2	L3

Faculty Signature CCI Signature HOD Signature

Internal Assessment Test 1- Oct. 2024

SCHEME & SOLUTION

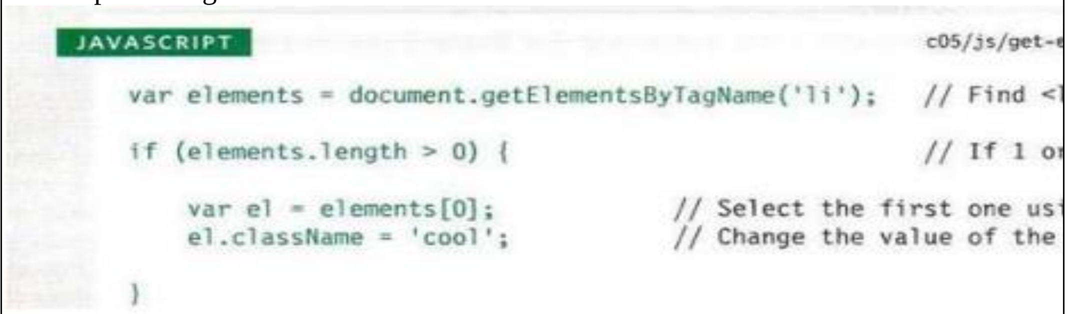
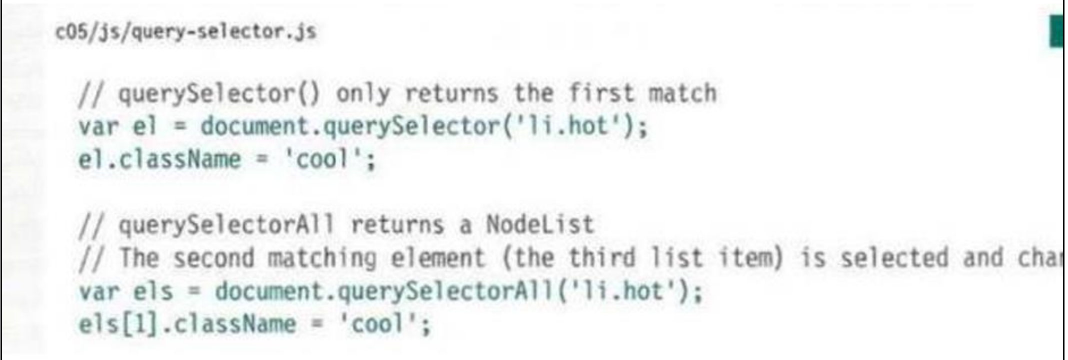
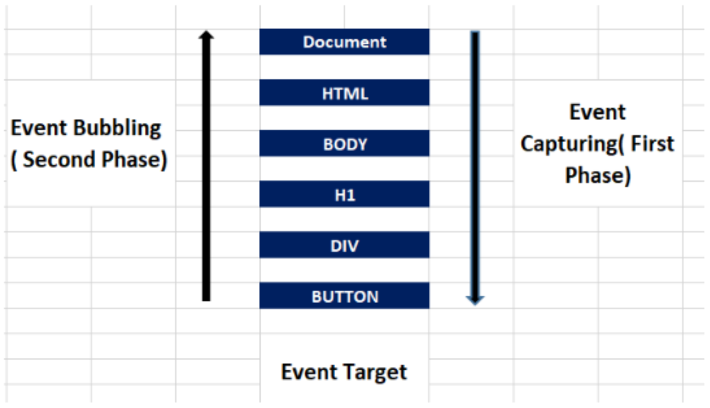
	Cryptography and Network Security	Sub Code: 21IS71	Branch:	ISE
	Answer any FIVE FULL questions	MARKS	CO	RBT
1a	Primitive Data Types Primitive types are the most basic types in JavaScript. They represent single values and are immutable (cannot be changed after creation). These types include: Non-primitive data types, also called reference types, are more complex than primitive types. They store references to the data rather than the actual data. Modifying an object or array will affect all references to it.	3+2=5M	CO1	L2
1b	A **closure** is the combination of a function bundled together (enclosed) with references to its surrounding state (the **lexical environment**). In other words, a closure gives a function access to its outer scope. In JavaScript, closures are created every time a function is created, at function creation time.	3+2=5M	CO1	L2
2a	In JavaScript, == (loose equality) performs type coercion before comparison, while === (strict equality) does not, requiring both value and type to be identical for a true result.	3+2=5M	CO1	L2

2b	**Function Declaration**
	A function declaration defines a function with a specified name. It is hoisted to the top of its scope, meaning it can be called before it is defined in the code.
	
	`function add(a, b) {`
	`  return a + b;`
	`}`
	
	Function Expression
	A function expression defines a function as part of an expression. It can be anonymous (without a name) or named. Function expressions are not hoisted and must be defined before they are called.
	`const multiply = function(a, b) {`
	`  return a * b;`
	`};`
	
	`const subtract = (a, b) => {`
	`    return a - b;`
	`};`
	
	Arrow Function
	An arrow function provides a concise syntax for writing function expressions. It does not have its own this context, arguments object, or super keyword, and cannot be used as a constructor. Arrow functions are often used for short, simple functions.
	`const divide = (a, b) => a / b;`
		3+2=5M	CO1	L2

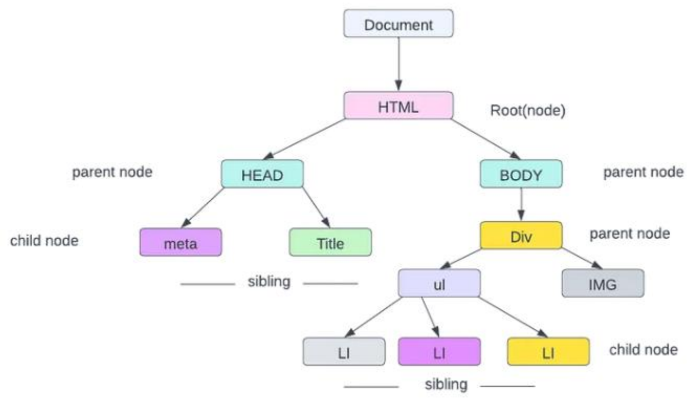
3a	The this keyword in JavaScript is a reference variable that is automatically assigned a value when a function is called. It refers to the object that is executing the current piece of	2+3M	CO1	L2

	JavaScript code. The value of this is not determined by how or where a function is declared, but by how it is called – the call-site.
	In JavaScript, the this keyword refers to the object that is currently executing the code. Its value is determined by how a function is called, not where it is defined. Here's
	how this behaves in different contexts:
	1. Global Context
	When this is used outside of any function, it refers to the global object. In browsers, this is usually the window object.
	JavaScript
	`console.log(this === window); // true (in browsers)`
	2. Function Context
	• **Simple Function Call**: In a regular function call, this refers to the global object (or undefined in strict mode).
	JavaScript
	`function showThis() {`
	`  console.log(this === window);`
	`}`
	`showThis(); // true`
	• **Method Call**: When a function is called as a method of an object, this refers to the object that owns the method.
	JavaScript
	`const obj = {`
	`  name: 'John',`
	`  greet: function() {`
	`    console.log('Hello, ' + this.name);`
	`  }`
	`};`
	`obj.greet(); // Hello, John`
	• **Constructor Call**: When a function is used as a constructor with the new keyword, this refers to the newly created instance.
	JavaScript
	`function Person(name) {`
	`  this.name = name;`
	`}`
	`const person = new Person('Alice');`
	`console.log(person.name); // Alice`			

3b	Loops allow you to execute a block of code repeatedly.
	Types of Loops:
	
	1. for Loop:
	o Used when the number of iterations is known.
	`for (let i = 0; i < 5; i++) {`
	`console.log(i); // Output: 0, 1, 2, 3, 4`
	`}`
	
	2. while Loop:
	o Executes as long as the condition is true.
	`let i = 0;`
	`while (i < 5) {`
	`console.log(i); // Output: 0, 1, 2, 3, 4`
	`i++;`
	`}`
		2+3=5M	CO1	L2

4a	Selecting by Tag Name • The getElementsByTagName() method selects all elements with a specific tag name.  JAVASCRIPT c05/js/get-e `var elements = document.getElementsByTagName('li');` `// Find <l` `if (elements.length > 0) {` `// If 1 or` `var el = elements[0];` `// Select the first one usi` `el.className = 'cool';` `// Change the value of the` `}` Selecting Using Query Selectors • querySelector(): Selects the first element that matches a given CSS selector. • querySelectorAll(): Selects all elements that match a given CSS selector.  c05/js/query-selector.js `// querySelector() only returns the first match` `var el = document.querySelector('li.hot');` `el.className = 'cool';` `// querySelectorAll returns a NodeList` `// The second matching element (the third list item) is selected and cha` `var els = document.querySelectorAll('li.hot');` `els[1].className = 'cool';`	4+1=5M	CO2	L2
4b	**The capturing phase** The first phase is the **capturing phase**, which occurs when an element nested in various elements gets clicked. Right before the click reaches its final destination, the click event of each of its parent elements must be triggered. This phase trickles down from the top of the DOM tree to the target element. **The bubbling phase** The **bubbling phase**, which is the last phase, is the reverse of the capturing phase. In this phase, the event bubbles up the target element through its parent element, the ancestor, to the global window object. By default, all events you add with addEventListener are in the bubble phase.  Event Bubbling (Second Phase) Document HTML BODY H1 DIV BUTTON Event Capturing(First Phase) Event Target	4+1=5M	CO2	L2
5a	For more customized validation logic, JavaScript event listeners such as addEventListener can be used to trigger validation functions on form submission or input changes. This approach allows developers to define complex validation rules and provide instant feedback to users.	4+1=5M	CO2	L2

5b	. Required Field Validation This validation ensures that certain fields must be filled out before the form can be submitted. `if (usernameInput.value.trim() === '') {` `usernameFeedback.textContent = 'Username is required.';` 2. Email Format Validation This validation checks if the entered email address is in a valid format. `const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;` `if (!emailPattern.test(emailInput.value)) {` `emailFeedback.textContent = 'Please enter a valid email address.';` 3. Password Strength Validation This validation checks if the password meets certain criteria, such as length, inclusion of numbers, and special characters. `passwordInput.addEventListener('input', function() {` `const password = passwordInput.value;` `const isValid = password.length >= 8 && /[A-Z]/.test(password) && /[0-]/.test(password) && /[!@#$%^&*]/.test(password);` 4. Real-time Validation Using Event Listeners Real-time validation provides immediate feedback to users as they fill out the form. This can be applied to various fields, such as required fields, email, and password. `const form = document.getElementById('myForm');` `// Real-time validation for username` `document.getElementById('username').addEventListener('input', function() {` `const usernameFeedback = document.getElementById('usernameFeedback');` `usernameFeedback.textContent = this.value.trim() === '' ? 'Username is required.' : '';` `});` `// Real-time validation for email` `document.getElementById('email').addEventListener('input', function() {` `const emailFeedback = document.getElementById('emailFeedback');` `const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;` `emailFeedback.textContent = !emailPattern.test(this.value) ? 'Please enter a valid email address.' : '';` `});` `// Real-time validation for password`	1+1+1+2 =5M	CO2	L2
6a	Event delegation is a technique in JavaScript where a single event listener is attached to a parent element instead of multiple listeners on individual child elements. **Example of Event Delegation** Let's implement event delegation using the above HTML structure. We will add a click event listener to the **<ul>** element that will handle clicks on its **<li>** children.	3+2=5M	CO2	L3
6b	The Document Object Model (DOM) represents an HTML or XML document as a tree-like structure, organizing its elements into a hierarchy.	3+2=5M	CO2	L3

	 Document HTML Root(node) parent node HEAD BODY parent node child node meta Title Div parent node sibling ul IMG LI LI LI child node sibling			