**1. Introduction to JavaScript**

* **Overview:**  
  JavaScript is a high-level, interpreted language primarily used to add interactivity and behavior to web pages. It runs in the browser (client-side) and on the server (Node.js).
* **Example:**
* <!DOCTYPE html>
* <html>
* <head>
* <title>Introduction to JavaScript</title>
* </head>
* <body>
* <script>
* console.log("Hello, world!"); // Output to the console
* </script>
* </body>
* </html>

**2. JavaScript Alerts – Adding Behaviour to Websites**

* **Overview:**  
  The alert() function creates a pop-up dialog box with a specified message. It’s useful for simple debugging or informing users.
* **Example:**
* // This will display a popup alert with the message "Welcome to JavaScript!"
* alert("Welcome to JavaScript!");

**3. Data Types in JavaScript**

* **Overview:**  
  JavaScript is dynamically typed and includes several data types:
  + **Number:** For numeric values.
  + **String:** For text.
  + **Boolean:** true or false.
  + **null:** A deliberate non-value.
  + **undefined:** A variable that has not been assigned a value.
  + **Object:** For complex data structures (including arrays and functions).
  + **Symbol (ES6):** For unique identifiers.
* **Example:**
* let age = 30; // Number
* let name = "Alice"; // String
* let isStudent = false; // Boolean
* let job = null; // null
* let salary; // undefined (not yet assigned)
* let person = { // Object
* firstName: "Alice",
* lastName: "Smith"
* };

**4. JavaScript Variables**

* **Overview:**  
  Variables are declared using var, let, or const.
  + var is function-scoped and has been traditionally used.
  + let and const (ES6) are block-scoped. Use const for values that should not change.
* **Example:**
* // Using let for a variable that might change
* let counter = 10;
* counter = 15;
* // Using const for a constant value
* const PI = 3.14159;

**5. Naming and Naming Conventions for JavaScript Variables**

* **Overview:**
  + Variable names can include letters, digits, underscores (\_), and dollar signs ($).
  + They cannot start with a digit.
  + JavaScript is case-sensitive.
  + Common practice is to use camelCase (e.g., myVariable).
* **Example:**
* let firstName = "Alice";
* let \_privateVar = "hidden";
* let $element = document.getElementById("myId");

**6. String Concatenation**

* **Overview:**  
  Strings can be combined using the + operator or template literals (backticks).
* **Example (using + operator):**
* let greeting = "Hello, ";
* let name = "Alice";
* let message = greeting + name + "!";
* console.log(message); // Outputs: Hello, Alice!
* **Example (using template literals):**
* let message2 = `${greeting}${name}!`;
* console.log(message2); // Outputs: Hello, Alice!

**7. String Lengths and Retrieving the Number of Characters**

* **Overview:**  
  Use the .length property to get the number of characters in a string.
* **Example:**
* let myString = "JavaScript";
* console.log(myString.length); // Outputs: 10

**8. Slicing and Extracting Parts of a String**

* **Overview:**  
  Methods such as slice(), substring(), and substr() can extract parts of a string.
  + slice(start, end) extracts characters from start index up to, but not including, end.
* **Example:**
* let text = "Hello, World!";
* let slicedText = text.slice(7, 12);
* console.log(slicedText); // Outputs: World

**9. Challenge: Changing Casing in Text**

* **Overview:**  
  JavaScript provides methods like toUpperCase() and toLowerCase() to change string casing.
* **Example:**
* let lowerCaseText = "hello";
* let upperCaseText = lowerCaseText.toUpperCase();
* console.log(upperCaseText); // Outputs: HELLO
* // Also, you can change to lower case:
* let mixedText = "JaVaScRiPt";
* console.log(mixedText.toLowerCase()); // Outputs: javascript

**12. Functions Part 1: Creating and Calling Functions**

* **Overview:**  
  Functions are defined using the function keyword. They encapsulate code for reuse.
* **Example:**
* // Function definition
* function greet() {
* console.log("Hello, world!");
* }
* // Calling the function
* greet();

**13. Functions Part 2: Parameters and Arguments**

* **Overview:**  
  Functions can accept parameters (inputs) and use arguments when called. This makes functions flexible and reusable.
* **Example:**
* // Function with parameters
* function add(a, b) {
* return a + b;
* }
* let sum = add(5, 3);
* console.log(sum); // Outputs: 8

**14. Functions Part 3: Outputs & Return Values**

* **Overview:**  
  Use the return statement to send a value back from a function.
* **Example:**
* function square(x) {
* return x \* x;
* }
* let result = square(4);
* console.log(result); // Outputs: 16

**15. Challenge: Create a BMI Calculator**

* **Overview:**  
  A BMI (Body Mass Index) calculator takes weight and height as inputs and calculates BMI using the formula:  
  **BMI = weight / (height × height)**  
  (Typically, weight in kilograms and height in meters.)
* **Example:**
* // Function to calculate BMI
* function calculateBMI(weight, height) {
* // Ensure height is not zero to avoid division by zero.
* if (height === 0) {
* return "Height cannot be zero.";
* }
* let bmi = weight / (height \* height);
* return bmi.toFixed(2); // Returns the BMI rounded to 2 decimal places
* }
* // Example usage:
* let weight = 70; // in kilograms
* let height = 1.75; // in meters
* console.log("Your BMI is " + calculateBMI(weight, height));

**16. Additional Challenges & Coding Exercises**

**Life in Weeks Coding Exercise**

* **Overview:**  
  This exercise typically involves calculating the number of weeks you have left (based on an average lifespan) given your current age.
* **Example:**
* function lifeInWeeks(age) {
* const maxAge = 90;
* const weeksPerYear = 52;
* let remainingYears = maxAge - age;
* let weeksLeft = remainingYears \* weeksPerYear;
* return weeksLeft;
* }
* let currentAge = 25;
* console.log("You have " + lifeInWeeks(currentAge) + " weeks left to live (approximately).");

**BMI Calculator Challenge (Coding Exercise 3)**

* **Overview:**  
  Similar to the BMI calculator example above, you may be challenged to improve or build a user-interactive version.
* **Example:**
* // A simple BMI calculator with prompts (for browser use)
* function promptBMICalculator() {
* let weight = parseFloat(prompt("Enter your weight in kilograms:"));
* let height = parseFloat(prompt("Enter your height in meters:"));
* if (isNaN(weight) || isNaN(height) || height === 0) {
* alert("Please enter valid numbers.");
* return;
* }
* let bmi = weight / (height \* height);
* alert("Your BMI is " + bmi.toFixed(2));
* }
* // Uncomment the following line to run the BMI calculator when the script loads.
* // promptBMICalculator();

**Quizzes on JavaScript (Variables, Numbers, Functions)**

* **Overview:**  
  Quizzes help solidify your understanding. Try writing small snippets of code or predicting output to test your knowledge. For example:
  + **Variable Naming Quiz:**  
    Which of these are valid variable names?
  + let 1stName; // Invalid (starts with a number)
  + let \_firstName; // Valid
  + let $firstName; // Valid
  + let first-name; // Invalid (hyphen is not allowed)
  + **Numbers Quiz:**  
    What is the output of:
  + let a = 5;
  + let b = 2;
  + console.log(a % b); // Expected output: 1
  + **Functions Quiz:**  
    What will be logged?
  + function test(x) {
  + return x \* 2;
  + }
  + console.log(test(3)); // Expected output: 6

Below are notes covering each of the topics you listed with explanations and example code. These notes will help you become familiar with the JavaScript syntax and concepts, especially coming from a C++ background. You can run these examples in your browser’s console or include them in an HTML file within a <script> tag.

**99. Random Number Generation in JavaScript: Building a Love Calculator**

* **Overview:**  
  JavaScript’s Math.random() function generates a floating-point number between 0 (inclusive) and 1 (exclusive). You can scale it to generate numbers in any desired range. In a love calculator example, you might generate a random percentage that represents “love compatibility.”
* **Example:**
* // Generate a random love percentage between 0 and 100
* let loveScore = Math.floor(Math.random() \* 101); // 0 to 100 inclusive
* console.log("Your love score is: " + loveScore + "%");

**100. Control Statements: Using If-Else Conditionals & Logic**

* **Overview:**  
  The if-else structure lets you execute code based on conditions. You can chain multiple conditions with else if.
* **Example:**
* let score = Math.floor(Math.random() \* 101);
* if (score > 80) {
* console.log("Great match!");
* } else if (score >= 50) {
* console.log("There's potential.");
* } else {
* console.log("Maybe just friends.");
* }

**101. Comparators and Equality**

* **Overview:**  
  JavaScript has various comparators:
  + == compares values after type coercion.
  + === compares both value and type (strict equality).
  + Other operators include !=, !==, <, >, <=, and >=.
* **Example:**
* let a = 5;
* let b = "5";
* console.log(a == b); // true because of type coercion
* console.log(a === b); // false because the types differ

**102. Combining Comparators**

* **Overview:**  
  Use logical operators (&& for AND, || for OR) to combine comparisons.
* **Example:**
* let age = 25;
* let hasLicense = true;
* // Check if a person is of legal age and has a license
* if (age >= 18 && hasLicense) {
* console.log("You can drive.");
* } else {
* console.log("You cannot drive.");
* }
* // OR operator example:
* if (age < 18 || !hasLicense) {
* console.log("You cannot drive.");
* }

**Coding Exercise 4: BMI Calculator Advanced (IF/ELSE)**

* **Overview:**  
  Enhance your BMI calculator by using if/else statements to categorize the result (e.g., underweight, normal, overweight).
* **Example:**
* function calculateBMI(weight, height) {
* if (height === 0) {
* return "Height cannot be zero.";
* }
* let bmi = weight / (height \* height);
* return bmi;
* }
* function bmiCategory(bmi) {
* if (bmi < 18.5) {
* return "Underweight";
* } else if (bmi >= 18.5 && bmi < 25) {
* return "Normal weight";
* } else if (bmi >= 25 && bmi < 30) {
* return "Overweight";
* } else {
* return "Obese";
* }
* }
* let weight = 70; // in kilograms
* let height = 1.75; // in meters
* let bmiValue = calculateBMI(weight, height);
* console.log("Your BMI is " + bmiValue.toFixed(2) + " and you are " + bmiCategory(bmiValue) + ".");

**105. Collections: Working with JavaScript Arrays**

* **Overview:**  
  Arrays are used to store multiple values in a single variable. JavaScript arrays are dynamic and can hold elements of any type.
* **Example:**
* let fruits = ["apple", "banana", "cherry"];
* console.log("First fruit:", fruits[0]); // Access first element
* console.log("Number of fruits:", fruits.length);

**106. Adding Elements and Intermediate Array Techniques**

* **Overview:**  
  Use methods such as push() to add elements and explore intermediate array techniques like slicing, splicing, mapping, filtering, etc.
* **Example:**
* let numbers = [1, 2, 3];
* // Add an element to the end
* numbers.push(4);
* console.log("After push:", numbers);
* // Remove an element from the beginning
* let first = numbers.shift();
* console.log("Removed element:", first);
* console.log("After shift:", numbers);
* // Using map to create a new array of squared values
* let squares = numbers.map(num => num \* num);
* console.log("Squares:", squares);

**Coding Exercise 6: Who's Buying Lunch? Code Challenge**

* **Overview:**  
  Given a list of names, randomly select one person to pay for lunch. This is a fun exercise in using arrays and random number generation.
* **Example Concept:**  
  Use Math.random() to pick an index from the array.

**107. Who's Buying Lunch Solution**

* **Example Code:**
* let names = ["Alice", "Bob", "Charlie", "Dana", "Evan"];
* // Randomly select an index from the names array
* let randomIndex = Math.floor(Math.random() \* names.length);
* let buyer = names[randomIndex];
* console.log(buyer + " is buying lunch today!");

**108. Control Statements: While Loops**

* **Overview:**  
  A while loop executes a block of code as long as the specified condition is true. It’s useful when you do not know the number of iterations beforehand.
* **Example:**
* let count = 0;
* while (count < 5) {
* console.log("Count is: " + count);
* count++;
* }

**109. Solution to the 99 Bottles Challenge**

* **Overview:**  
  The “99 Bottles of Beer” challenge involves printing out the verses of the song using a loop. A while loop can be used to count down from 99.
* **Example Code:**
* let bottles = 99;
* while (bottles > 0) {
* console.log(bottles + " bottles of beer on the wall, " + bottles + " bottles of beer.");
* bottles--;
* console.log("Take one down, pass it around, " + bottles + " bottles of beer on the wall.\n");
* }

**110. Control Statements: For Loops**

* **Overview:**  
  The for loop is often used when the number of iterations is known. It contains an initializer, a condition, and an increment/decrement statement.
* **Example:**
* for (let i = 0; i < 5; i++) {
* console.log("Iteration " + i);
* }

**111. Introducing the Fibonacci Code Challenge**

* **Overview:**  
  The Fibonacci challenge involves generating the Fibonacci sequence—a series where each number is the sum of the two preceding ones, typically starting with 0 and 1.
* **Example Concept:**  
  Create a loop that computes Fibonacci numbers until a certain limit or for a specific number of iterations.

**Coding Exercise 7: The Fibonacci Exercise**

* **Overview:**  
  Write a function to generate a Fibonacci sequence up to n terms.
* **Example Code:**
* function generateFibonacci(n) {
* if (n <= 0) {
* return [];
* } else if (n === 1) {
* return [0];
* }
* let fib = [0, 1];
* for (let i = 2; i < n; i++) {
* fib.push(fib[i - 1] + fib[i - 2]);
* }
* return fib;
* }
* let fibSequence = generateFibonacci(10);
* console.log("Fibonacci sequence:", fibSequence);

**112. Fibonacci Solution**

* **Overview:**  
  The solution above is one way to generate and output a Fibonacci sequence. It initializes an array with the first two numbers and uses a for loop to calculate the remaining numbers.
* **Review Example:**  
  (See the code in the previous section under Coding Exercise 7 for a working Fibonacci solution.)

These notes and examples cover the topics from random number generation to loops and array manipulation. Experiment with the code by modifying values, adding more conditions, and testing different scenarios to get even more comfortable with JavaScript’s syntax and logic. Happy coding!