### **Seminar 1: while vs do...while**

**Suggested structure:**

1. **Introduction (1–2 mins)**
   * What are while and do...while loops?
   * When are they typically used?
2. **Syntax & Flow (3–4 mins)**
   * Show side-by-side syntax.
   * Use a flowchart or diagram to highlight the key difference:  
      → while checks condition first  
      → do...while runs the block at least once
3. **Examples (3–4 mins)**
   * Simple number loop
   * A scenario where do...while is more appropriate (e.g., input validation)
4. **Wrap-up (2 mins)**
   * Key differences
   * When to use which

### **Seminar 2: for...in vs for...of**

**Suggested structure:**

1. **Introduction (1–2 mins)**
   * What are for...in and for...of?
   * Which data types they work with
2. **Syntax & Use Cases (3–4 mins)**
   * for...in → used for objects (enumerable properties)
   * for...of → used for arrays, strings, sets (iterables)
3. **Examples (3–4 mins)**
   * Looping over an object with for...in
   * Looping over an array with for...of
   * What happens if you use for...in on an array?
4. **Wrap-up (2 mins)**
   * Summary of differences
   * Common mistakes to avoid

### **Seminar 3: Return vs console.log in Functions**

**Introduction (1–2 mins)**

* What is return and what is console.log
* Why understanding the difference matters

**Core Concept Explanation (3–4 mins)**

* return: sends a value out of the function to where it was called
* console.log: prints output to the console but does NOT send data out
* How each affects function behavior and flow

**Examples / Demos (3–4 mins)**

* Simple function with return
* Same function with console.log instead of return
* What happens when you try to assign the function result to a variable

**Wrap-up / Summary (2 mins)**

* Key differences recap
* When to use return vs when to use console.log

### **Seminar 4: Anonymous Functions**

**Introduction (1–2 mins)**

* What is an anonymous function?
* Where do you commonly see them used?

**Core Concept Explanation (3–4 mins)**

* Functions without a name assigned to a variable or passed as arguments
* Contrast with named functions
* Common use cases (event handlers, callbacks)

**Examples / Demos (3–4 mins)**

* Assigning anonymous function to a variable
* Passing anonymous function to setTimeout or array methods

**Wrap-up / Summary (2 mins)**

* Why anonymous functions are useful
* When to use anonymous vs named function

### **Seminar 5: Parameters vs Arguments**

**Introduction (1–2 mins)**

* Basic function parts: parameters vs arguments
* Common confusion between the two

**Core Concept Explanation (3–4 mins)**

* Parameters: variables listed in function declaration
* Arguments: actual values passed when calling function
* How default parameters work

**Examples / Demos (3–4 mins)**

* Function declaration with parameters
* Calling function with different arguments
* Using default parameter example

**Wrap-up / Summary (2 mins)**

* Clear definitions recap
* Why understanding difference helps debugging

### **Seminar 6: Function Declarations vs Function Expressions**

**Introduction (1–2 mins)**

* Two common ways to create functions in JS

**Core Concept Explanation (3–4 mins)**

* Syntax differences
* Hoisting behavior of declarations vs expressions

**Examples / Demos (3–4 mins)**

* Function declaration example
* Function expression example
* What happens if you call before declaration/expression

**Wrap-up / Summary (2 mins)**

* When to use each style
* Key differences recap

### **Seminar 7: map() vs forEach() in Arrays**

**Introduction (1–2 mins)**

* Two popular array methods to process elements

**Core Concept Explanation (3–4 mins)**

* .map() creates a new array and returns it
* .forEach() executes a function on each element, but does not return a new array

**Examples / Demos (3–4 mins)**

* Using .map() to double numbers
* Using .forEach() to log each element
* Show difference in return values

**Wrap-up / Summary (2 mins)**

* Summary of key differences
* When to prefer one over the other

### **Seminar 8: Mutating vs Non-Mutating Array Methods**

**Introduction (1–2 mins)**

* Arrays can be changed or left unchanged by methods

**Core Concept Explanation (3–4 mins)**

* Mutating methods: .push(), .pop(), .splice(), .sort()
* Non-mutating methods: .slice(), .map(), .filter()

**Examples / Demos (3–4 mins)**

* Show .push() modifies original array
* Show .slice() returns new array
* Before/after console logs for both

**Wrap-up / Summary (2 mins)**

* Why it matters for program bugs
* Best practices to avoid unintended mutation

### **Seminar 9: Searching Arrays: .indexOf(), .includes(), .find()**

**Introduction (1–2 mins)**

* Common need to search arrays for values

**Core Concept Explanation (3–4 mins)**

* .indexOf() returns index or -1
* .includes() returns true/false
* .find() returns first matching element or undefined

**Examples / Demos (3–4 mins)**

* Searching for number in array using each method
* Using .find() on array of objects

**Wrap-up / Summary (2 mins)**

* Quick recap of differences
* When to use each method

### **Seminar 10: Value vs Reference in Arrays**

**Introduction (1–2 mins)**

* Arrays are objects — assigned by reference

**Core Concept Explanation (3–4 mins)**

* Copying array variables copies reference, not values
* Mutations affect all references
* How to make a shallow copy (slice(), spread)

**Examples / Demos (3–4 mins)**

* Assign array to new variable and modify
* Show how both variables change
* Make a shallow copy and show separation

**Wrap-up / Summary (2 mins)**

* Why this concept matters for bug prevention
* When to use copies

### **Seminar 11: Object Methods**

**Introduction (1–2 mins)**

* Objects can hold functions as properties called methods

**Core Concept Explanation (3–4 mins)**

* Syntax for methods inside objects
* Why methods are useful for encapsulation

**Examples / Demos (3–4 mins)**

* Simple object with greeting method
* Calling the method and seeing output

**Wrap-up / Summary (2 mins)**

* Recap benefits of methods
* Preview of this (optional teaser)

### **Seminar 12: Object Destructuring**

**Introduction (1–2 mins)**

* Modern syntax to extract properties from objects easily

**Core Concept Explanation (3–4 mins)**

* Basic destructuring syntax
* Renaming variables during destructuring (optional)

**Examples / Demos (3–4 mins)**

* Destructure a simple object
* Use destructured variables in code

**Wrap-up / Summary (2 mins)**

* When destructuring helps simplify code
* Encourage practice with examples

### 

### **Seminar 13: Difference Between Primitive vs Reference Types (with Focus on Objects)**

**Introduction (1–2 mins)**

* JavaScript has primitive and reference data types

**Core Concept Explanation (3–4 mins)**

* Primitive types: copied by value (string, number, boolean, null, undefined, symbol)
* Reference types: copied by reference (objects, arrays, functions)
* Impact on assignment and mutation

**Examples / Demos (3–4 mins)**

* Assign and mutate primitive variable vs object
* Show how changes affect copies differently

**Wrap-up / Summary (2 mins)**

* Why understanding this difference prevents bugs
* Practical takeaway for working with objects and arrays

### **Seminar 14: Truthy and falsy values in JS**

### **Introduction (1–2 mins)**

* What are truthy and falsy values?
* Why does JavaScript treat some values as true or false in conditions?

### **Explanation (2–3 mins)**

* Falsy values: false, 0, -0, 0n, "", null, undefined, NaN
* Truthy values: everything else

### **Syntax & Flow (3–4 mins)**

* Used in conditions: if, while, ||, &&
* Flowchart showing evaluation:  
   → Value gets converted to Boolean  
   → Falsy: block skipped  
   → Truthy: block runs

### **Examples (3–4 mins)**

* Empty input handling
* Assigning default value
* Checking if data exists

### **Wrap-up (2 mins)**

* Only 7 falsy values — rest are truthy
* Helps in writing clean conditions
* Try testing values to see how they behave

### **Seminar 15: Function Declarations vs Function Expressions**

What’s the difference?

Why does hoisting behave differently for them?