JavaScript Syllabus for revision and interview preparation .

JavaScript:

- **Basics:** Variables, data types, functions, scope, hoisting.
- **Advanced Concepts:** Closures, promises, async/await, event loop.
- **DOM Manipulation:** Selectors, events, manipulating HTML/CSS.
- **ES6+ Features:** Arrow functions, destructuring, spread/rest operators, modules.

JavaScript Syllabus

1. **Basic Syntax and Operators:**
    - Variables (var, let, const)
    - Data types (string, number, boolean, object, array, etc.)
    - Operators (arithmetic, comparison, logical)
2. **Control Structures:**
    - Conditionals (if, else, switch)
    - Loops (for, while, do-while)
3. **Functions:**
    - Function declaration, expression
    - Arrow functions
    - Higher-order functions (map, filter, reduce)
4. **Objects and Arrays:**
    - Object properties, methods
    - Array methods (push, pop, shift, unshift, splice, slice, etc.)
5. **DOM Manipulation:**
    - Selecting elements (getElementById, querySelector, etc.)
    - Modifying elements (textContent, innerHTML, styles)
    - Event handling (addEventListener, event object)
6. **Asynchronous JavaScript:**
    - Callbacks
    - Promises
    - Async/await
7. **ES6+ Features:**
    - Template literals
    - Destructuring
    - Spread/rest operators
    - Modules (import/export)
8. **Error Handling:**
    - Try/catch
    - Throwing errors

JavaScript Interview Preparation Syllabus

---

1. Basic Syntax and Operators

Variables (var, let, const):

- **Q: What are the differences between var, let, and const?**
  - **A:**
    - `var` is function-scoped and can be redeclared and reassigned.
    - `let` is block-scoped and can be reassigned but not redeclared within the same scope.
    - `const` is block-scoped and cannot be reassigned or redeclared.

Data types (string, number, boolean, object, array, etc.):

- **Q: What are the different data types in JavaScript?**
  - **A:**
    - Primitive Types: String, Number, Boolean, Null, Undefined, Symbol, BigInt.
    - Non-primitive Types: Object (including Arrays, Functions).

Operators (arithmetic, comparison, logical):

- **Q: What are the different types of operators in JavaScript?**
  - **A:**
    - **Arithmetic Operators:** `+`, `-`, `*`, `/`, `%`, `++`, `--`
    - **Comparison Operators:** `==`, `===`, `!=`, `!==`, `>`, `<`, `>=`, `<=`
    - **Logical Operators:** `&&`, `||`, `!`

---

2. Control Structures

Conditionals (if, else, switch):

- **Q: How does a switch statement work in JavaScript?**
  - **A:** A `switch` statement evaluates an expression, matching the expression's value to a case clause, and executing the associated statements. It uses `case` and `default` keywords.

Loops (for, while, do-while):

- **Q: What are the different types of loops in JavaScript?**
  - **A:**
    - `for` loop
    - `while` loop
    - `do-while` loop
    - `for...of` loop (for iterable objects)
    - `for...in` loop (for object properties)

---

3. Functions

Function declaration, expression:

- **Q: What is the difference between function declarations and function expressions?**
  - **A:**
    - **Function Declarations:** Named functions defined using the `function` keyword and can be hoisted.
    - **Function Expressions:** Functions defined as expressions can be anonymous and are not hoisted.

Arrow functions:

- **Q: What are arrow functions and how do they differ from regular functions?**
  - **A:** Arrow functions provide a shorter syntax and do not have their own `this`, `arguments`, `super`, or `new.target`. They are best suited for non-method functions.

Higher-order functions (map, filter, reduce):

- **Q: What are higher-order functions in JavaScript?**
  - **A:** Functions that take other functions as arguments or return them as results. Examples include `map`, `filter`, and `reduce`.

---

4. Objects and Arrays

Object properties, methods:

- **Q: How do you create and access properties in a JavaScript object?**
  - **A:**
    - Creating: `const obj = { key: 'value' };`
    - Accessing: `obj.key` or `obj['key']`.

Array methods (push, pop, shift, unshift, splice, slice, etc.):

- **Q: What are some common array methods in JavaScript?**
  - **A:**
    - `push()`: Add elements to the end.
    - `pop()`: Remove the last element.
    - `shift()`: Remove the first element.
    - `unshift()`: Add elements to the beginning.
    - `splice()`: Add/remove elements from a specific index.
    - `slice()`: Return a shallow copy of a portion of an array.

---

5. DOM Manipulation

Selecting elements (getElementById, querySelector, etc.):

- **Q: How do you select an HTML element using JavaScript?**
  - **A:**
    - `getElementById(id)`
    - `querySelector(selector)`
    - `getElementsByClassName(className)`
    - `getElementsByTagName(tagName)`

Modifying elements (textContent, innerHTML, styles):

- **Q: How do you change the text content of an HTML element?**
  - **A:** Use the `textContent` or `innerHTML` properties.
    - Example: `element.textContent = 'New Text';`

Event handling (addEventListener, event object):

- **Q: How do you add an event listener to an HTML element?**

    - **A:**
        - Use the `addEventListener` method.
        - Example: `element.addEventListener('click', function);`

---

6. Asynchronous JavaScript

Callbacks:

- **Q: What is a callback function in JavaScript?**
    - **A:** A function passed as an argument to another function, which can be executed later.

Promises:

- **Q: What is a promise in JavaScript?**
    - **A:** An object representing the eventual completion or failure of an asynchronous operation. It can be in one of three states: pending, fulfilled, or rejected.

Async/await:

- **Q: How do async/await work in JavaScript?**
    - **A:**
        - `async` functions return a promise.
        - `await` pauses the execution of an `async` function until the promise is resolved.

---

7. ES6+ Features

Template literals:

- **Q: What are template literals in JavaScript?**
    - **A:** String literals allowing embedded expressions, denoted by backticks (`) and` ${}` for expressions.
        - Example: `` const greeting = `Hello, ${name}!`; ``

Destructuring:

- **Q: How does destructuring assignment work in JavaScript?**

    - **A:** It allows unpacking values from arrays or properties from objects into distinct variables.
        - Example: `const {name, age} = person;`

Spread/rest operators:

- **Q: What do the spread and rest operators do in JavaScript?**
    - **A:**
        - **Spread Operator:** Expands iterable elements.
        - **Rest Operator:** Collects multiple elements into an array.
        - Example: `const arr1 = [1, 2, 3]; const arr2 = [...arr1, 4, 5];`

Modules (import/export):

- **Q: How do JavaScript modules work?**
    - **A:** Use `export` to make functions, objects, or primitives available to other modules, and `import` to use them in another file.

Example:
javascript
Copy code

```
// module.js

export const name = 'John';



// main.js

import { name } from './module';
```

        -

---

8. Error Handling

Try/catch:

- **Q: How do you handle errors in JavaScript?**
    - **A:** Use `try` to wrap code that may throw an error and `catch` to handle the error.

Example:
javascript

Copy code
```
try {

  // code that may throw an error

} catch (error) {

  console.error(error);

}
```

  - ■

Throwing errors:

- **Q: How do you throw a custom error in JavaScript?**
  - **A:** Use the `throw` statement followed by an error object or a custom message.
    - Example: `throw new Error('This is a custom error');`

Sample Questions and Answers

1. **Q: What is the difference between `==` and `===`?**
   - **A:** `==` checks for value equality with type coercion, while `===` checks for both value and type equality without type coercion.
2. **Q: What is event delegation?**
   - **A:** Event delegation is a technique of using a single event listener to manage events for multiple elements based on event bubbling.
3. **Q: Explain how `reduce` works in JavaScript.**

A: `reduce` executes a reducer function on each element of the array, resulting in a single output value.
javascript
Copy code
```
const sum = [1, 2, 3, 4].reduce((acc, curr) => acc + curr, 0); // sum
is 10
```

   - ○

4. **Q: What is the purpose of `async` and `await` in JavaScript?**
   - **A:** `async` functions return a promise, and `await` pauses execution until the promise is resolved, making asynchronous code look synchronous.
5. **Q: How can you prevent default behavior for an event in JavaScript?**
   - **A:** Use `event.preventDefault()` inside the event handler.

This structured and comprehensive syllabus should help you create a detailed PDF for your JavaScript interview preparation. Study each topic thoroughly, practice coding regularly, and build projects to reinforce your understanding.