JavaScript Basics - Study Notes

1. What is JavaScript?

JavaScript is a **high-level, interpreted programming language** used to make web pages interactive. It runs in web browsers and allows developers to create dynamic content, handle events, and manipulate web page elements in real-time.

Key Features:

- Dynamically typed language
- Supports object-oriented, functional, and imperative programming
- Runs on both client-side (browser) and server-side (Node.js)

---

2. Frontend vs Backend JavaScript

Frontend JavaScript	Backend JavaScript
Runs in the browser	Runs on the server
Manipulates DOM, handles user interactions	Handles database operations, authentication
Uses browser APIs (document, window)	Uses Node.js and its modules
Example: Form validation, animations	Example: API creation, file handling

---

3. Document and Window Object

Window Object:

- Represents the browser window
- Global object in JavaScript
- Contains methods like `alert()`, `setTimeout()`, `localStorage`
- Example: `window.alert("Hello")`

Document Object:

- Represents the HTML document loaded in the window
- Part of the window object (`window.document`)
- Used to access and manipulate HTML elements
- Example: `document.getElementById("demo")`

---

4. What is DOM?

DOM (Document Object Model) is a programming interface that represents the HTML document as a tree structure. It allows JavaScript to access and manipulate HTML elements, attributes, and styles dynamically.

Example:

```javascript
document.getElementById("myId").innerHTML = "New Content";
```

---

5. What is a Variable?

A **variable** is a container that stores data values. Variables are declared using `var`, `let`, or `const` keywords.

Example:

```javascript
let name = "John";
const age = 25;
```

---

6. Difference Between var, let, and const

Feature	var	let	const
Scope	Function-scoped	Block-scoped	Block-scoped
Reassignment	Yes	Yes	No
Redeclaration	Yes	No	No
Hoisting	Yes (initialized as undefined)	Yes (not initialized)	Yes (not initialized)

Example:

```javascript
var x = 10;  // Can be redeclared and updated
let y = 20;  // Cannot be redeclared, can be updated
const z = 30; // Cannot be redeclared or updated
```

---

7. Data Types

Primitive Data Types (Immutable)

1. **String** - Text values: `"Hello"`
2. **Number** - Numeric values: `42`, `3.14`
3. **Boolean** - True or false: `true`, `false`
4. **Undefined** - Variable declared but not assigned: `let x;`
5. **Null** - Intentional absence of value: `let x = null;`
6. **Symbol** - Unique identifiers (ES6)
7. **BigInt** - Large integers (ES11)

Non-Primitive Data Types (Mutable)

1. **Object** - Collections of key-value pairs: `{name: "John", age: 25}`
2. **Array** - Ordered list of values: `[1, 2, 3]`
3. **Function** - Reusable code blocks

---

8. Difference Between == and ===

== (Loose Equality)	=== (Strict Equality)
Compares values only	Compares both value and type
Performs type coercion	No type coercion

Example:

```javascript
5 == "5"   // true (type coercion)
5 === "5"  // false (different types)
```

---

9. typeof Operator

The `typeof` operator returns the data type of a variable or expression.

Example:

```javascript
```

```javascript
typeof 42        // "number"
typeof "Hello"   // "string"
typeof true      // "boolean"
typeof undefined // "undefined"
typeof null      // "object" (known bug)
typeof [1,2,3]   // "object"
```

---

10. NaN (Not-a-Number)

NaN represents a value that is not a legal number. It results from invalid mathematical operations.

Checking NaN:

```javascript
isNaN("Hello")   // true
isNaN(123)       // false
Number.isNaN(NaN) // true (recommended)
```

---

11. Undefined vs Null

Undefined	Null
Variable declared but not assigned	Intentional absence of value
Default value for uninitialized variables	Must be explicitly assigned
Type: `undefined`	Type: `object`

Example:

```javascript
let a;        // undefined
let b = null; // null
```

---

12. Operators in JavaScript

Arithmetic Operators

`+` (addition), `-` (subtraction), `*` (multiplication), `/` (division), `%` (modulus), `**` (exponentiation)

Comparison Operators

`==`, `===`, `!=`, `!==`, `>`, `<`, `>=`, `<=`

Logical Operators

- `&&` (AND) - Both conditions must be true
- `||` (OR) - At least one condition must be true
- `!` (NOT) - Inverts the boolean value

---

13. Pre-increment (++a) vs Post-increment (a++)

Pre-increment (++a): Increments first, then returns the value **Post-increment (a++):** Returns the value first, then increments

Example:

```javascript
let a = 5;
console.log(++a);  // 6 (incremented, then returned)
console.log(a);    // 6

let b = 5;
console.log(b++);  // 5 (returned, then incremented)
console.log(b);    // 6
```

---

14. + Operator with Strings

When `+` is used with strings, it performs **concatenation** (joining strings together).

Example:

```javascript
"Hello" + " " + "World"  // "Hello World"
"5" + 3                  // "53" (number converted to string)
5 + 3                    // 8 (addition)
```

---

15. Conditional Statements

Simple if

```javascript
if (age >= 18) {
    console.log("Adult");
}
```

if-else

```javascript
if (score >= 50) {
    console.log("Pass");
} else {
    console.log("Fail");
}
```

Multiple if (if-else-if ladder)

```javascript
if (marks >= 90) {
    console.log("Grade A");
} else if (marks >= 75) {
    console.log("Grade B");
} else if (marks >= 50) {
    console.log("Grade C");
} else {
    console.log("Fail");
}
```

Nested if

```javascript
if (age >= 18) {
    if (hasLicense) {
        console.log("Can drive");
    } else {
        console.log("Cannot drive without license");
    }
}
```

---

16. Switch Case

The `switch` statement evaluates an expression and executes code blocks based on matching cases.

Syntax:

```javascript
switch (day) {
  case 1:
    console.log("Monday");
    break;
  case 2:
    console.log("Tuesday");
    break;
  case 3:
    console.log("Wednesday");
    break;
  default:
    console.log("Invalid day");
}
```

Note: The `break` statement prevents fall-through to the next case.

---

17. Ternary Operator

A shorthand for `if-else` statements. Syntax: `condition ? valueIfTrue : valueIfFalse`

Example:

```javascript
let age = 20;
let status = (age >= 18) ? "Adult" : "Minor";
console.log(status);  // "Adult"

// Equivalent if-else:
let status;
if (age >= 18) {
  status = "Adult";
} else {
  status = "Minor";
}
```

---

18. How to Select HTML Elements Using JavaScript?

JavaScript provides multiple methods to select HTML elements:

Common Selection Methods:

1. getElementById() - Selects element by ID

```
javascript

let element = document.getElementById("myId");
```

2. getElementsByClassName() - Selects elements by class name (returns HTMLCollection)

```
javascript

let elements = document.getElementsByClassName("myClass");
```

3. getElementsByTagName() - Selects elements by tag name (returns HTMLCollection)

```
javascript

let paragraphs = document.getElementsByTagName("p");
```

4. querySelector() - Selects first matching element using CSS selector

```
javascript

let element = document.querySelector(".myClass");
let element2 = document.querySelector("#myId");
```

5. querySelectorAll() - Selects all matching elements (returns NodeList)

```
javascript

let elements = document.querySelectorAll(".myClass");
```

---

19. Difference Between getElementById and querySelector

getElementById	**querySelector**
Selects by ID only	Selects by any CSS selector
Faster performance	Slightly slower but more flexible
Returns single element or null	Returns first matching element or null

getElementById	querySelector
Syntax: `getElementById("id")`	Syntax: `querySelector("#id")` or `querySelector(".class")`

Examples:

```javascript
// getElementById - only for IDs
let elem1 = document.getElementById("header");

// querySelector - can use any CSS selector
let elem2 = document.querySelector("#header");
let elem3 = document.querySelector(".btn");
let elem4 = document.querySelector("div > p");
let elem5 = document.querySelector("[type='text']");
```

---

20. How to Change Text or Style Using JavaScript?

Changing Text Content:

1. innerHTML - Sets or gets HTML content

```javascript
document.getElementById("demo").innerHTML = "<strong>Bold Text</strong>";
```

2. innerText - Sets or gets visible text only

```javascript
document.getElementById("demo").innerText = "Plain Text";
```

3. textContent - Sets or gets all text content

```javascript
document.getElementById("demo").textContent = "All Text";
```

Changing Styles:

1. Inline Styles (style property)

```javascript
```

```
let element = document.getElementById("box");
element.style.color = "red";
element.style.backgroundColor = "yellow";
element.style.fontSize = "20px";
element.style.border = "2px solid black";
```

2. Adding/Removing CSS Classes

```
javascript

let element = document.getElementById("box");

// Add class
element.classList.add("active");

// Remove class
element.classList.remove("active");

// Toggle class
element.classList.toggle("highlight");

// Check if class exists
if (element.classList.contains("active")) {
    console.log("Has active class");
}
```

Example:

```
javascript

// Change multiple properties
let box = document.querySelector(".box");
box.innerHTML = "Updated Content";
box.style.color = "white";
box.style.backgroundColor = "blue";
box.style.padding = "10px";
```

---

21. What are Events in JavaScript?

Events are actions or occurrences that happen in the browser, which JavaScript can detect and respond to.

Common Event Types:

Mouse Events:

- `click` - Element is clicked
- `dblclick` - Element is double-clicked
- `mouseover` - Mouse pointer moves over element
- `mouseout` - Mouse pointer moves out of element
- `mousedown` - Mouse button is pressed
- `mouseup` - Mouse button is released

Keyboard Events:

- `keydown` - Key is pressed down
- `keyup` - Key is released
- `keypress` - Key is pressed (deprecated)

Form Events:

- `submit` - Form is submitted
- `change` - Input value changes
- `focus` - Element gets focus
- `blur` - Element loses focus
- `input` - Input value is modified

Window Events:

- `load` - Page finishes loading
- `resize` - Window is resized
- `scroll` - Page is scrolled

Adding Event Listeners:

Method 1: Inline HTML (Not Recommended)

```
html

<button onclick="alert('Clicked!')">Click Me</button>
```

Method 2: DOM Property

```
javascript
```

```javascript
let btn = document.getElementById("myBtn");
btn.onclick = function() {
    alert("Button clicked!");
};
```

Method 3: addEventListener (Recommended)

```javascript
let btn = document.getElementById("myBtn");
btn.addEventListener("click", function() {
    alert("Button clicked!");
});
```

Complete Example:

```javascript
// Select button
let button = document.querySelector("#submitBtn");

// Add click event
button.addEventListener("click", function() {
    // Change text
    document.getElementById("output").innerHTML = "Button was clicked!";

    // Change style
    this.style.backgroundColor = "green";
    this.style.color = "white";
});

// Multiple events on same element
button.addEventListener("mouseover", function() {
    this.style.transform = "scale(1.1)";
});

button.addEventListener("mouseout", function() {
    this.style.transform = "scale(1)";
});
```

Event Object:

When an event occurs, an event object is automatically passed to the handler:

```javascript
```

```
button.addEventListener("click", function(event) {
    console.log(event.type);        // "click"
    console.log(event.target);      // Element that triggered event
    console.log(event.clientX);     // Mouse X coordinate
    console.log(event.clientY);     // Mouse Y coordinate
});
```

Quick Tips for Students:

- Practice writing code regularly
- Use `console.log()` to debug and understand code flow
- Experiment with examples in browser console
- Always use `===` instead of `==` for comparisons
- Prefer `let` and `const` over `var` in modern JavaScript
- Use `addEventListener` for better event handling
- Practice DOM manipulation by creating small interactive projects

---

Created by Gatam Mukherjee