JavaScript Notes

Table of Contents

Introduction to JavaScript

- **JavaScript** is a lightweight, interpreted programming language used to make web pages interactive.
- It is an essential part of web development, alongside HTML and CSS.

Variables

var, let, const

- **var**: Function-scoped variable.
- **let**: Block-scoped variable.
- **const**: Block-scoped constant; cannot be reassigned.

Example:

```
var name = "John";
let age = 30;
const isStudent = true;
```

Data Types

- **Primitive Types**: String, Number, Boolean, Null, Undefined, Symbol (ES6), BigInt (ES11).
- **Objects**: Collections of key-value pairs, including arrays and functions.

Example:

```
let aString = "Hello, World!";
let aNumber = 42;
let aBoolean = true;
let aNull = null;
let anUndefined;
let aSymbol = Symbol('symbol');
let aBigInt = BigInt(12345678901234567890);
```

Operators

- **Arithmetic Operators**: `+`, `-`, `*`, `/`, `%`, `++`, `--`
- **Assignment Operators**: `=`, `+=`, `-=`, `*=`, `/=`, `%=`
- **Comparison Operators**: `==`, `===`, `!=`, `!==`, `>`, `<`, `>=`, `<=`
- **Logical Operators**: `&&`, `||`, `!`
- **Ternary Operator**: `condition ? expr1 : expr2`

Example:

```
let sum = 10 + 5;
let isEqual = (sum === 15);
let result = isEqual ? "Correct" : "Wrong";
```

Functions

- Functions are blocks of code designed to perform a particular task.
- **Function Declaration**: Regular named function.
- **Function Expression**: Function stored in a variable.
- **Arrow Function (ES6)**: Shorter syntax for function expressions.

Example:

```
function greet(name) {
    return "Hello, " + name + "!";
}

let greetExpression = function(name) {
    return "Hi, " + name + "!";
};

let greetArrow = (name) => "Hey, " + name + "!";
```

Control Flow

If-Else

- Conditional statements for decision-making.

Example:

```
let time = 20;

if (time < 12) {
    console.log("Good morning");
} else if (time < 18) {
    console.log("Good afternoon");
} else {
    console.log("Good evening");
}
```

Switch

- Multi-way branch statement.

Example:

```
let day = 3;

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");
}
```

Loops

- **For Loop**: Repeats a block of code a specified number of times.
- **While Loop**: Repeats a block of code while a condition is true.
- **Do-While Loop**: Similar to while loop, but executes at least once.

Example:

```
for (let i = 0; i < 5; i++) {
    console.log(i);
}

let j = 0;
while (j < 5) {
    console.log(j);
    j++;
}

let k = 0;
do {
    console.log(k);
    k++;
} while (k < 5);
```

Arrays

- **Array**: A collection of elements.

Example:

```
let fruits = ["Apple", "Banana", "Cherry"];
console.log(fruits[0]); // "Apple"

fruits.push("Orange"); // Add to end
let fruit = fruits.pop(); // Remove from end

fruits.unshift("Strawberry"); // Add to start
let firstFruit = fruits.shift(); // Remove from start

fruits.forEach(function(item, index) {
    console.log(index, item);
});
```

Objects

- **Object**: A collection of key-value pairs.

Example:

```
let person = {
    firstName: "John",
    lastName: "Doe",
    age: 25,
    greet: function() {
        console.log("Hello, " + this.firstName);
```

```
    }
};

console.log(person.firstName); // "John"
person.greet(); // "Hello, John"
```

DOM Manipulation

- **Document Object Model (DOM)**: A programming interface for web documents.
- Use JavaScript to manipulate HTML and CSS.

Example:

```
document.getElementById("myElement").innerHTML = "New Content";
document.querySelector(".myClass").style.backgroundColor = "yellow";

let newElement = document.createElement("div");
newElement.textContent = "Hello, World!";
document.body.appendChild(newElement);
```

Events

- **Events**: Actions that happen in the browser (e.g., clicks, keypresses).
- Use `addEventListener` to handle events.

Example:

```
document.getElementById("myButton").addEventListener("click", function() {
    alert("Button was clicked!");
});
```

ES6 Features

Let and Const

- Block-scoped variables and constants.

Template Literals

- Multi-line strings and string interpolation using backticks.

Example:

```
let name = "John";
let message = `Hello, ${name}!`;
```

```
console.log(message);
```

Arrow Functions

- Shorter function syntax.

Default Parameters

- Set default values for function parameters.

Example:

```
function greet(name = "Guest") {
    console.log(`Hello, ${name}`);
}
```

Destructuring

- Extract values from arrays or objects.

Example:

```
let [a, b] = [10, 20];
let { firstName, lastName } = person;
```

Spread and Rest Operators

- Spread: Expand arrays or objects.
- Rest: Combine arguments into an array.

Example:

```
let arr = [1, 2, 3];
let newArr = [...arr, 4, 5]; // [1, 2, 3, 4, 5]

function sum(...numbers) {
    return numbers.reduce((acc, num) => acc + num, 0);
}
```

Asynchronous JavaScript

Promises

- A way to handle asynchronous operations.

- **Promise States**: Pending, Fulfilled, Rejected.

Example:

```
let promise = new Promise(function(resolve, reject) {
    let success = true;
    if (success) {
        resolve("Success!");
    } else {
        reject("Error!");
    }
});

promise.then(function(result) {
    console.log(result);
}).catch(function(error) {
    console.log(error);
});
```

Async/Await

- Syntactic sugar for working with promises.

Example:

```
async function fetchData() {
    try {
        let response = await fetch("https://api.example.com/data");
        let data = await response.json();
        console.log(data);
    } catch (error) {
        console.log(error);
    }
}

fetchData();
```

These notes provide a solid introduction to JavaScript basics. You can expand on each topic by adding more examples and explanations as you continue to learn!