# JavaScript Cheat Sheet

## 1. Introduction to JavaScript

JavaScript is a dynamic programming language used to create interactive and dynamic websites. It is essential for web development, alongside HTML and CSS. JavaScript runs on the client-side, meaning it is executed by the user's browser.

---

## 2. Data Types

JavaScript has various data types that are divided into two categories: primitive types and reference types.

**Primitive Data Types:**

- **String**: A sequence of characters.

  ```
  javascript
  Copy code
  let str = "Hello, World!";
  ```

- **Number**: Represents both integer and floating-point numbers.

  ```
  javascript
  Copy code
  let num = 42;
  let price = 19.99;
  ```

- **Boolean**: Represents true or false.

  ```
  javascript
  Copy code
  let isActive = true;
  ```

- **Undefined**: A variable that has been declared but not assigned a value.

  ```
  javascript
  Copy code
  let x;
  console.log(x); // undefined
  ```

- **Null**: Represents no value or non-existence.

  ```
  javascript
  Copy code
  let obj = null;
  ```

- **Symbol**: A unique and immutable data type often used as object property keys.

  ```
  javascript
  Copy code
  let sym = Symbol('description');
  ```

- **BigInt**: For integers larger than the Number type can safely store.

  ```
  javascript
  Copy code
  ```

```
let bigInt = 1234567890123456789012345678901234567890n;
```

**Reference Data Types:**

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

  ```
  javascript
  Copy code
  let person = { name: "John", age: 30 };
  ```

- **Array**: An ordered list of values.

  ```
  javascript
  Copy code
  let fruits = ["apple", "banana", "cherry"];
  ```

- **Function**: A block of code designed to perform a particular task.

  ```
  javascript
  Copy code
  function greet(name) {
    return "Hello, " + name;
  }
  ```

---

## 3. Variables and Constants

- **var**: Declares a variable (function-scoped, can be redeclared).

  ```
  javascript
  Copy code
  var x = 10;
  ```

- **let**: Declares a block-scoped variable (cannot be redeclared in the same scope).

  ```
  javascript
  Copy code
  let y = 20;
  ```

- **const**: Declares a block-scoped constant (value cannot be reassigned).

  ```
  javascript
  Copy code
  const z = 30;
  ```

---

## 4. Operators

- **Arithmetic Operators**:

  ```
  javascript
  Copy code
  let sum = 5 + 2;      // 7
  let diff = 5 - 2;     // 3
  let prod = 5 * 2;     // 10
  let quotient = 5 / 2; // 2.5
  let remainder = 5 % 2; // 1
  let exponent = 5 ** 2; // 25
  ```

- **Assignment Operators**:

  javascript
  Copy code
  ```
  let a = 10;
  a += 5; // a = a + 5
  a -= 5; // a = a - 5
  a *= 5; // a = a * 5
  a /= 5; // a = a / 5
  ```

- **Comparison Operators**:

  javascript
  Copy code
  ```
  5 == 5   // true (loose equality)
  5 === 5  // true (strict equality)
  5 != 6   // true (loose inequality)
  5 !== 6  // true (strict inequality)
  5 > 3    // true
  5 < 3    // false
  ```

- **Logical Operators**:

  javascript
  Copy code
  ```
  true && false  // false
  true || false  // true
  !true          // false
  ```

- **Ternary Operator**:

  javascript
  Copy code
  ```
  let result = (5 > 3) ? "Yes" : "No";
  ```

---

## 5. Control Structures

### if...else:

javascript
Copy code
```
let age = 18;
if (age >= 18) {
  console.log("Adult");
} else {
  console.log("Minor");
}
```

### switch:

javascript
Copy code
```
let fruit = "apple";
switch (fruit) {
  case "apple":
    console.log("Apple selected");
    break;
  case "banana":
    console.log("Banana selected");
    break;
```

```
  default:
    console.log("No match found");
}
```

### for loop:

javascript
Copy code
```
for (let i = 0; i < 5; i++) {
  console.log(i);  // Prints 0, 1, 2, 3, 4
}
```

### while loop:

javascript
Copy code
```
let i = 0;
while (i < 5) {
  console.log(i);  // Prints 0, 1, 2, 3, 4
  i++;
}
```

---

## 6. Functions

### Function Declaration:

javascript
Copy code
```
function add(a, b) {
  return a + b;
}
```

### Function Expression:

javascript
Copy code
```
const multiply = function(a, b) {
  return a * b;
};
```

### Arrow Functions:

javascript
Copy code
```
const subtract = (a, b) => a - b;
```

---

## 7. Arrays

### Creating an Array:

javascript
Copy code
```
let colors = ["red", "green", "blue"];
```

### Accessing Array Elements:

javascript
Copy code
```
let firstColor = colors[0]; // "red"
```

**Array Methods:**

- **push()**: Adds an element at the end of the array.

  javascript
  Copy code
  ```
  colors.push("yellow");
  ```

- **pop()**: Removes the last element from the array.

  javascript
  Copy code
  ```
  colors.pop();
  ```

- **shift()**: Removes the first element from the array.

  javascript
  Copy code
  ```
  colors.shift();
  ```

- **unshift()**: Adds an element at the beginning of the array.

  javascript
  Copy code
  ```
  colors.unshift("purple");
  ```

- **forEach()**: Loops through each element of the array.

  javascript
  Copy code
  ```
  colors.forEach(color => console.log(color));
  ```

---

## 8. Objects

### Creating an Object:

javascript
Copy code
```
let person = {
  name: "John",
  age: 30,
  greet: function() {
    console.log("Hello, " + this.name);
  }
};
```

### Accessing Object Properties:

javascript
Copy code
```
let name = person.name;     // John
let age = person["age"];    // 30
```

### Adding/Modifying Properties:

javascript
Copy code
```
person.city = "New York";
person.age = 31;
```

### Methods in Objects:

javascript
Copy code
```
person.greet();  // Outputs: "Hello, John"
```

---

## 9. DOM Manipulation

JavaScript allows you to manipulate the DOM (Document Object Model) to update HTML content, styles, and attributes.

### Selecting Elements:

javascript
Copy code
```
let element = document.getElementById("myElement");
let elements = document.getElementsByClassName("myClass");
let queryElement = document.querySelector(".myClass");
```

### Changing Content:

javascript
Copy code
```
element.innerHTML = "New Content";
```

### Changing Styles:

javascript
Copy code
```
element.style.color = "red";
```

### Event Listeners:

javascript
Copy code
```
element.addEventListener("click", function() {
  alert("Element clicked!");
});
```

---

## 10. ES6 Features

### Destructuring Assignment:

javascript
Copy code
```
let person = { name: "John", age: 30 };
let { name, age } = person;
```

### Template Literals:

javascript
Copy code
```
let name = "John";
let greeting = `Hello, ${name}!`;  // "Hello, John!"
```

### Spread Operator:

javascript
Copy code

```
let arr1 = [1, 2, 3];
let arr2 = [...arr1, 4, 5];  // [1, 2, 3, 4, 5]
```

### Rest Parameter:

javascript
Copy code
```
function sum(...numbers) {
  return numbers.reduce((acc, curr) => acc + curr, 0);
}
```

---

## 11. Asynchronous JavaScript

### Callbacks:

javascript
Copy code
```
function fetchData(callback) {
  setTimeout(() => {
    callback("Data loaded");
  }, 1000);
}

fetchData(data => {
  console.log(data);  // Outputs: "Data loaded"
});
```

### Promises:

javascript
Copy code
```
let promise = new Promise((resolve, reject) => {
  let success = true;
  if (success) {
    resolve("Success");
  } else {
    reject("Failure");
  }
});

promise
  .then(result => console.log(result))  // Outputs: "Success"
  .catch(error => console.log(error)); // Outputs: "Failure"
```

### Async/Await:

javascript
Copy code
```
async function getData() {
  let result = await promise;
  console.log(result);  // Outputs: "Success"
}
```