JavaScript Cheat Sheet

1. Basic Syntax

1.1 Comments

- **Single-line Comment**: `// This is a comment`

- **Multi-line Comment**:

```
/*
This is a
multi-line comment
*/
```

1.2 Semicolons

- Optional but recommended for clarity: `let x = 5;`

1.3 Case Sensitivity

- JavaScript is case-sensitive: `let Name = "John";` is different from `let name = "John";`

---

2. Variables and Data Types

2.1 Variables

- **Declaration**:

- `let`: Block-scoped variable
- `const`: Block-scoped constant
- `var`: Function-scoped variable (avoid in modern JS)

2.2 Data Types

- **Primitive Types**:

- `Number`: `let num = 42;`
- `String`: `let str = "Hello";`
- `Boolean`: `let bool = true;`
- `Null`: `let n = null;`
- `Undefined`: `let u = undefined;`
- `Symbol`: `let sym = Symbol("foo");`

- **Complex Types**:

- `Object`: `let obj = { key: "value" };`
- `Array`: `let arr = [1, 2, 3];`
- `Function`: `let func = function() { return "Hello"; };`

2.3 Type Conversion

- **String to Number**: `let num = Number("42");`
- **Number to String**: `let str = String(42);`
- **Boolean Conversion**: `let bool = Boolean(0); // false`

---

3. Operators

3.1 Arithmetic Operators

- `+`, `-`, `*`, `/`, `%`, `**` (Exponentiation)

3.2 Assignment Operators

- `=`, `+=`, `-=`, `*=`, `/=`, `%=`

3.3 Comparison Operators

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

3.4 Logical Operators

- `&&` (AND), `||` (OR), `!` (NOT)

3.5 Ternary Operator

- `condition ? expr1 : expr2`

---

4. Control Structures

4.1 Conditional Statements

- **if-else**:

```
if (condition) {
  // code
} else if (anotherCondition) {
  // code
} else {
  // code
}
```

- **switch**:

```
switch (expression) {
  case value1:
    // code
    break;
  case value2:
    // code
    break;
  default:
    // code
}
```

4.2 Loops

- **for**:

```
for (let i = 0; i < 10; i++) {
  // code
}
```

- **while**:

```
while (condition) {
  // code
}
```

- **do-while**:

```
do {
  // code
} while (condition);
```

- **for-in**:

```
for (let key in object) {
  // code
}
```

- **for-of**:

```
for (let value of array) {
  // code
}
```

---

5. Functions

5.1 Function Declaration

```
function functionName(parameters) {
  // code
  return result;
}
```

5.2 Function Expression

```
let functionName = function(parameters) {
  // code
  return result;
};
```

5.3 Arrow Functions

```
let functionName = (parameters) => {
  // code
  return result;
};
```

5.4 Default Parameters

```
function functionName(param1 = defaultValue1, param2 = defaultValue2) {
  // code
}
```

5.5 Rest Parameters

```
function functionName(...args) {
  // code
}
```

5.6 Immediately Invoked Function Expressions (IIFE)

```
(function() {
  // code
})();
```

---

6. Objects and Arrays

6.1 Objects

- **Creation**:

```
let obj = {
  key1: "value1",
  key2: "value2"
};
```

- **Accessing Properties**:

- Dot Notation: `obj.key1`
- Bracket Notation: `obj["key1"]`

6.2 Arrays

- **Creation**:

```
let arr = [1, 2, 3];
```

- **Accessing Elements**:

- `arr[0]`

- **Common Methods**:

- `push()`, `pop()`, `shift()`, `unshift()`, `slice()`, `splice()`, `map()`, `filter()`, `reduce()`

---

7. DOM Manipulation

7.1 Selecting Elements

- **By ID**: `document.getElementById("id")`
- **By Class**: `document.getElementsByClassName("class")`
- **By Tag**: `document.getElementsByTagName("tag")`
- **By Query**: `document.querySelector("selector")`
- **All By Query**: `document.querySelectorAll("selector")`

7.2 Modifying Elements

- **Changing Content**:

- `element.innerHTML = "new content";`
- `element.textContent = "new text";`

- **Changing Attributes**:

- `element.setAttribute("attribute", "value");`
- `element.getAttribute("attribute");`

- **Changing Styles**:

- `element.style.property = "value";`

7.3 Events

- **Adding Event Listeners**:

```
element.addEventListener("click", function() {
  // code
});
```

- **Removing Event Listeners**:

```
element.removeEventListener("click", function);
```

---

8. Asynchronous JavaScript

8.1 Callbacks

```
function fetchData(callback) {
  // code
  callback(data);
}
```

8.2 Promises

```
let promise = new Promise((resolve, reject) => {
  // code
  if (success) {
    resolve(result);
  } else {
    reject(error);
```

```
  }
});

promise.then(result => {
  // code
}).catch(error => {
  // code
});
```

8.3 Async/Await

```
async function fetchData() {
  try {
    let result = await promise;
    // code
  } catch (error) {
    // code
  }
}
```

---

9. Error Handling

9.1 try-catch

```
try {
  // code
} catch (error) {
  // code
} finally {
  // code
}
```

9.2 Throwing Errors

```
throw new Error("This is an error");
```

---

10. Modules

10.1 Exporting

```
export const variable = "value";
export function functionName() {
```

```
  // code
}
```

10.2 Importing

```
import { variable, functionName } from "./module.js";
```

---

11. ES6+ Features

11.1 Destructuring

- **Array Destructuring**:

```
let [a, b] = [1, 2];
```

- **Object Destructuring**:

```
let { key1, key2 } = { key1: "value1", key2: "value2" };
```

11.2 Spread/Rest Operator

- **Spread**:

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

- **Rest**:

```
function functionName(...args) {
  // code
}
```

11.3 Template Literals

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

11.4 Classes

```
class ClassName {
  constructor(parameters) {
    // code
  }
  methodName() {
    // code
  }
}
```

11.5 Enhanced Object Literals

```
let key = "value";
let obj = {
  key,
  method() {
    // code
  }
};
```

---

12. Tips and Tricks

12.1 Debugging

- **console.log()**: `console.log("Debugging message");`
- **debugger**: `debugger;`

12.2 Short-circuit Evaluation

- **Logical OR**: `let result = a || b;`
- **Logical AND**: `let result = a && b;`

12.3 Nullish Coalescing

- **Nullish Coalescing Operator**: `let result = a ?? b;`

12.4 Optional Chaining

- **Optional Chaining**: `let value = obj?.key?.value;`

12.5 Performance Tips

- **Avoid Global Variables**
- **Use `let` and `const` instead of `var`**

- **Minimize DOM Manipulation**

This cheat sheet provides a comprehensive overview of essential JavaScript concepts, syntax, and best practices. Use it as a quick reference to enhance your JavaScript skills.

By Ahmed Baheeg Khorshid

ver 1.0