Cheat Sheet for Javascript

JavaScript Cheat Sheet

1. Basic Syntax

1.1 Comments

- **Single-line**: `// This is a comment`
- **Multi-line**: `/* This is a multi-line comment */`

1.2 Semicolons

- Optional but recommended for clarity.

1.3 Case Sensitivity

- JavaScript is case-sensitive: `myVar` is different from `myvar`.

---

2. Variables and Data Types

2.1 Variables

- **Declaration**: `var`, `let`, `const`

- `var`: Function-scoped
- `let`: Block-scoped
- `const`: Block-scoped, immutable after assignment

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() {};`

2.3 Type Conversion

- **String to Number**: `Number("42")`
- **Number to String**: `String(42)`
- **Boolean to String**: `String(true)`

---

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...of**:

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

- **for...in**:

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

---

5. Functions

5.1 Function Declaration

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

5.2 Function Expression

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

5.3 Arrow Functions

```
const 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**:

```
obj.key1; // dot notation
obj["key1"]; // bracket notation
```

- **Adding/Updating Properties**:

```
obj.newKey = "newValue";
```

- **Deleting Properties**:

```
delete obj.key1;
```

6.2 Arrays

- **Creation**:

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

- **Accessing Elements**:

```
arr[0]; // first element
```

- **Common Methods**:

- `push()`: Adds to the end
- `pop()`: Removes from the end
- `shift()`: Removes from the start
- `unshift()`: Adds to the start
- `slice()`: Returns a portion of the array
- `splice()`: Adds/removes elements

- `map()`, `filter()`, `reduce()`: Functional methods

---

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 Text**: `element.textContent = "new text";`
- **Changing HTML**: `element.innerHTML = "<p>new HTML</p>";`
- **Changing Attributes**: `element.setAttribute("attribute", "value");`
- **Changing Styles**: `element.style.property = "value";`

7.3 Creating and Appending Elements

- **Creating**: `let newElement = document.createElement("div");`
- **Appending**: `parentElement.appendChild(newElement);`

---

8. Events

8.1 Event Listeners

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

8.2 Common Events

- `click`, `dblclick`, `mouseover`, `mouseout`, `keydown`, `keyup`, `submit`, `load`, `resize`, `scroll`

8.3 Event Object

- `event.target`: The element that triggered the event

- `event.preventDefault()`: Prevents default action

---

9. Asynchronous JavaScript

9.1 Callbacks

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

9.2 Promises

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

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

9.3 Async/Await

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

---

10. Error Handling

10.1 try...catch

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

10.2 Throwing Errors

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

---

11. Modules

11.1 Exporting

```
// module.js
export const myFunction = () => {
  // code
};

export default myFunction;
```

11.2 Importing

```
// main.js
import { myFunction } from './module.js';
import myDefaultFunction from './module.js';
```

---

12. ES6+ Features

12.1 let and const

- `let`: Block-scoped variable
- `const`: Block-scoped, immutable variable

12.2 Template Literals

```
let name = "John";
console.log(`Hello, ${name}!`);
```

12.3 Destructuring

- **Arrays**:

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

- **Objects**:

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

12.4 Spread/Rest Operator

- **Spread**:

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

- **Rest**:

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

12.5 Classes

```
class MyClass {
  constructor(prop) {
    this.prop = prop;
  }
  method() {
    // code
  }
}
```

12.6 Promises and Async/Await

- See section 9 for details.

---

13. Tips and Tricks

13.1 Debugging

- **console.log()**: Basic logging
- **debugger;**: Pauses execution for debugging
- **console.table()**: Displays data in a table format

13.2 Short-circuit Evaluation

- `let result = a || b;` // If `a` is truthy, `result` is `a`, else `b`
- `let result = a && b;` // If `a` is truthy, `result` is `b`, else `a`

13.3 Nullish Coalescing

- `let result = a ?? b;` // If `a` is `null` or `undefined`, `result` is `b`

13.4 Optional Chaining

- `let value = obj?.prop?.subProp;` // Safely access nested properties

13.5 Performance Tips

- Avoid global variables
- Use `let` and `const` instead of `var`
- Minimize DOM manipulation
- Use `requestAnimationFrame` for animations

This cheat sheet provides a comprehensive overview of essential JavaScript concepts and features. Use it as a quick reference to enhance your JavaScript programming skills.