JavaScript Cheatsheet

Basic Syntax and Data Types

JavaScript is a dynamic, high-level, interpreted programming language for web development and beyond.

- Variable Declaration:
  - `let`: Block-scoped variable (can be reassigned).
  - `const`: Block-scoped constant (cannot be reassigned).
  - `var`: Function-scoped variable (legacy, avoid).
- Basic Types: `number`, `string`, `boolean`, `null`, `undefined`, `symbol`, `bigint`.
- Object Type: `object`, `array`, `function`, `date`, `regexp`.

Operators

- Arithmetic: `+`, `-`, `*`, `/`, `%`, `**` (exponentiation).
- Comparison:
  - `==`, `!=`: Abstract equality (coercion occurs).
  - `===`, `!==`: Strict equality (recommended).
- Logical: `&&` (AND), `||` (OR), `!` (NOT), `??` (nullish coalescing).

Control Flow

- `if-else` / `switch`:

```
if (condition) { ... } else { ... }
```

- Loops:
  - `for (let i = 0; i < 5; i++) { ... }`
  - `for (const item of array) { ... }` (values)
  - `for (const key in object) { ... }` (keys)
- Error Handling:

```
try { ... } catch (error) { ... }
finally { ... }
```

Functions

- Declaration: `function add(a, b) { return a + b; }`
- Expression: `const add = function(a, b) { ... };`
- Arrow Function: `const add = (a, b) => a + b;`
- Rest parameters: `function sum(...nums) { ... }`
- Default parameters: `function greet(name = "Guest") { ... }`

Objects and Arrays

- Object:

```
const user = { name: "Alice", age: 25 };
const { name, age } = user; // Destructuring
```

- Array:

```
const fruits = ["apple", "banana"];
fruits.push("cherry");
const filtered = fruits.filter(f => f.startsWith("a"));
const mapped = fruits.map(f => f.toUpperCase());
```

- Spread Operator: `[...arr1, ...arr2]`, `{ ...obj1, ...obj2 }`

Asynchronous Programming

- Callback: Traditional way, can lead to callback hell.
- Promise: `.then()`, `.catch()`, `.finally()`.
- Async / Await: Modern, readable way to handle promises.

```
async function fetchData() {
    try {
        const res = await fetch(url);
        const data = await res.json();
        return data;
    } catch (err) { ... }
}
```

DOM Manipulation (Browser)

- Selection: `document.getElementById()`, `document.querySelector()`.
- Modification: `element.textContent`, `element.style`, `element.classList`.
- Events: `element.addEventListener('click', callback)`.

Modules (ESM)

- Export: `export const value = 10;`, `export default function() { ... }`
- Import: `import { value } from './module.js'`, `import MyFunc from './module.js'`

Google Style Guide

Basic Rules

- Use `let` and `const`. Avoid `var`.
- Use 2-space indentation (no tabs).
- Use single quotes `'` for strings.
- End statements with semicolons `;` (required by style guide).
- Avoid non-standard or experimental features.

Naming Conventions

- Variables and Functions: `lowerCamelCase`
- Classes: `PascalCase`
- Constants: `UPPER_SNAKE_CASE`
- Files: `lower_with_underscores.js`

Formatting

- Braces: Place same-line as the statement.
- Arrays and Objects: Use trailing commas in multi-line configurations.
- Arrow Functions: Use parentheses for single parameters for consistency.

Programming Practices

- Use strict equality `===`.
- Use template literals `Hello ${name}` for string interpolation.
- Prefer `for-of` loops over traditional `for` loops where possible.
- Avoid modifying objects you don't own (built-in prototypes).

Comments and Documentation (JSDoc)

- Use JSDoc `/** ... */` for all top-level classes and functions.
- Specify types for parameters and return values using `@param {type}` and `@return {type}`.