JavaScript Cheatsheet

1 Variables

Variables store data. Use `let`, `const`, or `var` (avoid `var` for modern code).

```
let name = "Alice"; // Mutable
const PI = 3.14159; // Immutable
var oldWay = "Deprecated"; // Avoid
```

2 Data Types

JavaScript has dynamic typing with several primitive and object types.

```
// Primitives
let string = "Hello"; // String
let number = 42; // Number (includes integers, floats)
let boolean = true; // Boolean
let nullValue = null; // Null
let undefinedValue; // Undefined
let symbol = Symbol("id"); // Unique identifier
let bigInt = 1234567890123456789On; // BigInt

// Objects
let obj = { key: "value" }; // Object
let arr = [1, 2, 3]; // Array
```

3 Operators

Operators perform operations on values (arithmetic, comparison, logical, etc.).

```
// Arithmetic
let sum = 5 + 3; // 8
let mod = 10 % 3; // 1

// Comparison
console.log(5 === 5); // true (strict equality)
console.log(5 !== "5"); // true (strict inequality)

// Logical
let and = true && false; // false
let or = true || false; // true
let not = !true; // false
```

4 Conditionals

Control flow with `if`, `else`, `switch`, or ternary operator.

```
// If-Else
let age = 18;
if (age >= 18) {
  console.log("Adult");
} else {
```

```
  console.log("Minor");
}

// Ternary
let status = age >= 18 ? "Adult" : "Minor"; // "Adult"

// Switch
let day = 1;
switch (day) {
  case 1:
    console.log("Monday");
    break;
  default:
    console.log("Other day");
}
```

5 Loops

Iterate over data with `for`, `while`, `do-while`, or array methods.

```
// For Loop
for (let i = 0; i < 3; i++) {
  console.log(i); // 0, 1, 2
}

// While
let i = 0;
while (i < 3) {
  console.log(i++); // 0, 1, 2
}

// For...of (Arrays)
let arr = [1, 2, 3];
for (let val of arr) {
  console.log(val); // 1, 2, 3
}

// ForEach
arr.forEach(val => console.log(val)); // 1, 2, 3
```

6 Functions

Functions encapsulate reusable code. Use declarations or arrow functions.

```
// Function Declaration
function greet(name) {
  return `Hello, ${name}`;
}

// Arrow Function
const add = (a, b) => a + b;

// Default Parameters
function log(message = "Hi") {
  console.log(message);
}
log(); // "Hi"

// Rest Parameters
const sumAll = (...numbers) => numbers.reduce((sum, n) => sum + n, 0);
console.log(sumAll(1, 2, 3)); // 6
```

7 Objects

Objects store key-value pairs and support methods.

```
let person = {
  name: "Bob",
  age: 30,
  greet() {
    return `Hi, I'm ${this.name}`;
  }
};
console.log(person.greet()); // "Hi, I'm Bob"
```

8 Arrays

Arrays store ordered lists and have powerful built-in methods.

```
let numbers = [1, 2, 3];
numbers.push(4); // [1, 2, 3, 4]
numbers.map(n => n * 2); // [2, 4, 6, 8]
numbers.filter(n => n > 2); // [3, 4]
```

9 ES6+ Features

9.1 Destructuring

Extract values from objects or arrays.

```
// Object Destructuring
let { name, age } = person;
console.log(name, age); // "Bob", 30

// Array Destructuring
let [first, second] = numbers;
console.log(first, second); // 1, 2
```

9.2 Spread/Rest

Spread expands elements; rest collects arguments.

```
// Spread
let arr1 = [1, 2];
let arr2 = [...arr1, 3, 4]; // [1, 2, 3, 4]
let obj1 = { a: 1 };
let obj2 = { ...obj1, b: 2 }; // { a: 1, b: 2 }

// Rest
function collect(...args) {
  return args;
}
console.log(collect(1, 2, 3)); // [1, 2, 3]
```

9.3 Promises

Handle asynchronous operations.

```
let promise = new Promise((resolve, reject) => {
  setTimeout(() => resolve("Done"), 1000);
});
promise.then(result => console.log(result)); // "Done" after 1s
```

9.4 Async/Await

Syntactic sugar for promises.

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

9.5 DOM Manipulation

Interact with HTML elements.

```
// Select elements
let div = document.querySelector("#myDiv");
let allParas = document.querySelectorAll("p");

// Modify content
div.innerHTML = "<p>Hello, DOM!</p>";
div.style.backgroundColor = "blue";

// Create element
let newEl = document.createElement("div");
newEl.textContent = "New Div";
document.body.appendChild(newEl);
```

9.6 Event Handling

Respond to user actions.

```
let button = document.querySelector("#myButton");
button.addEventListener("click", () => {
  console.log("Button clicked!");
});

// Event object
button.addEventListener("click", (event) => {
  console.log(event.target); // Button element
});
```

10 Notes

- Use `const` for variables that wont be reassigned.
- Prefer arrow functions for concise, non-method functions.
- Use `try/catch` for error handling in async code.
- Avoid mutating arrays/objects directly; use immutable methods like `map`, `filter`, or spread.