JavaScript Basics Cheat Sheet

What is JavaScript?
JavaScript is a scripting language used for adding interactivity and functionality to websites. It is often called the "language of the web" because it's a fundamental part of building interactive and dynamic web pages.

How does JavaScript work?
JavaScript is a client-side language, which means it runs in your web browser (e.g., Google Chrome), allowing you to interact with web pages without needing to constantly communicate with the web server. This means it can change what you see on a web page without needing to reload the whole page. It can also modify the content, structure, and behavior of a web page after it has loaded in your browser.

What can you do with JavaScript?

- **Add Interactivity:** buttons, menus, and forms that respond to user clicks and inputs
- **Manipulate Web Page Content:** update your web page, including text, images, and other elements, without needing to reload the entire page
- **Form Validation:** ensures that data is correctly filled out before it is submitted
- **Handle Events:** responds to events, such as mouse clicks, keyboard presses, and page scrolling
- **Request Data:** fetch data from web servers and APIs, enabling you to display real-time information like the news, the weather, or other web content
- **Animate Elements:** examples are image sliders and fading effects
- **Manage Cookies:** store user information or preferences
- **Control Browser Behavior:** control how web pages behave (e.g., going to new pages, or refreshing the page itself)
- **Create Games:** online video games
- **Validate and Manipulate Dates and Times:** examples are calendars or countdowns
- **Handle Errors and Debug Code:** identify and fix issues in your code

Variables

var: obsolete and has a broader scope, **let:** variables that change, **const:** unchanging values

```
let variableName = value; // Declare a variable
const constantName = value; // Declare a constant
var oldVariable = value; // Declare a variable (limit using 'var' since it can be obsolete)
```

Data Types

```
let number = 23;
let string = 'Hello World!';
let boolean = true;
let array = [1, 2, 3, 4, 5];
let object = { key: 'value' }; // objects are a collection of key-value pairs in JavaScript
```

Operators

```
let sum = 2 + 3; // 5
let difference = 5 - 2; // 3
let product = 2 * 4; // 8
let quotient = 6 / 2; // 3
let remainder = 5 % 2; // 1
```

Comparison

= assigns values **==** checks loose equality/inequality **===** checks strict equality/inequality

```
let isEqual = 5 === '5'; // False. Strict equality
let isNotEqual = 5 !== '5'; // True. Strict inequality
let looseEqual = 5 == '5'; // True. Loose equality
let looseNotEqual = 5 != '5'; // False. Loose equality
let isGreaterThan = 5 > 3;
let isLessThan = 3 < 5;
let logicalAnd = true && false;
let logicalOr = true || false;
```

Control Structures

```
if (condition) {
  // Code to run if the condition is true
} else {
  // Code to run if the condition is false
}

for (let i = 0; i < 5; i++) {
  // Loop from 0 to 4
}

while (condition) {
  // Loop while the condition is true
}
```

JavaScript Functions

```javascript
// Add Two Numbers
function add(a, b) {
  return a + b;
}

const sum = add(5, 3);
console.log(sum); // Outputs: 8
```

```javascript
// Find the Square of a Number
function square(number) {
  return number * number;
}

const result = square(4);
console.log(result); // Outputs: 16
```

```javascript
// Check if a Number is Even
function isEven(number) {
  return number % 2 === 0;
}

const even = isEven(6);
console.log(even); // Outputs: true
```

```javascript
// Concatenate Two Strings
function concatenateStrings(str1, str2) {
  return str1 + ' ' + str2;
}

const message = concatenateStrings('Hello', 'World');
console.log(message); // Outputs: 'Hello World'
```

```javascript
function getRandomNumber(min, max) {
  return Math.floor(Math.random() * (max - min + 1)) + min;
}

const randomNumber = getRandomNumber(1, 10);
console.log(randomNumber); // Outputs a random number between 1 and 10
```

Arrow Functions

```
// Basic Arrow Function
const add = (a, b) => a + b;
const result = add(5, 3);
console.log(result); // Outputs: 8
```

```
// Function with a Single Parameter
const square = (number) => number * number;
const result = square(4);
console.log(result); // Outputs: 16
```

```
// Function with No Parameters
const greet = () => "Hello, World!";
const message = greet();
console.log(message); // Outputs: 'Hello, World!'
```

```
// Function with Multiple Statements
const multiply = (a, b) => {
  const result = a * b;
  return result;
};
const product = multiply(5, 3);
console.log(product); // Outputs: 15
```

```
// Map Function with Arrow Function
const numbers = [1, 2, 3, 4, 5];
const squaredNumbers = numbers.map((number) => number * number);
console.log(squaredNumbers); // Outputs: [1, 4, 9, 16, 25]
```

```
// Filter Function with Arrow Function
const words = ['apple', 'banana', 'cherry', 'date'];
const filteredWords = words.filter((word) => word.length > 5);
console.log(filteredWords); // Outputs: ['banana', 'cherry']
```

Arrays

```
const fruits = ['apple', 'banana', 'cherry'];

// Access the first element (at index 0)
const firstFruit = fruits[0];
console.log(firstFruit); // Outputs: 'apple'

// Access the second element (at index 1)
const secondFruit = fruits[1];
console.log(secondFruit); // Outputs: 'banana'

// Access the third element (at index 2)
const thirdFruit = fruits[2];
console.log(thirdFruit); // Outputs: 'cherry'
```

```javascript
// Modify an Array Element
const fruits = ['apple', 'banana', 'cherry'];
fruits[1] = 'kiwi';
console.log(fruits); // Outputs: ['apple', 'kiwi', 'cherry']
```

```javascript
// Add an Element to the End of the Array
const fruits = ['apple', 'banana', 'cherry'];
fruits.push('date');
console.log(fruits); // Outputs: ['apple', 'banana', 'cherry', 'date']
```

```javascript
// Remove the Last Element of the Array
const fruits = ['apple', 'banana', 'cherry'];
fruits.pop();
console.log(fruits); // Outputs: ['apple', 'banana']
```

```javascript
// Add an Element to the Beginning of the Array
const fruits = ['apple', 'banana', 'cherry'];
fruits.unshift('date');
console.log(fruits); // Outputs: ['date', 'apple', 'banana', 'cherry']
```

```javascript
// Remove the First Element of the Array
const fruits = ['apple', 'banana', 'cherry'];
fruits.shift();
console.log(fruits); // Outputs: ['banana', 'cherry']
```

```javascript
// Find the Index of an Element in the Array
const fruits = ['apple', 'banana', 'cherry'];
const index = fruits.indexOf('banana');
console.log(index); // Outputs: 1
```

```javascript
// Slice and Copy a Portion of an Array
const fruits = ['apple', 'banana', 'cherry', 'date', 'elderberry'];
const slicedFruits = fruits.slice(1, 4);
console.log(slicedFruits); // Outputs: ['banana', 'cherry', 'date']
```

Objects

```javascript
const person = {
  name: 'Max',
  age: 27,
};
const personName = person.name;
console.log('Person\'s Name:', personName); // Outputs: "Person's Name: Max"
```

Loops

```
for (let i = 0; i < array.length; i++) {
  // Loop through an array (e.g., Print values from 0 to the length of array minus 1)
}
```

```
const person = {
  name: 'Max',
  age: 27,
  job: 'Web Developer'
};

for (const key in person) {
  const value = person[key];
  console.log(`Key: ${key}, Value: ${value}`);
}

/* Outputs:
Key: name, Value: Max
Key: age, Value: 27
Key: job, Value: Web Developer */
```

Backticks (``)

- Introduced in ECMAScript 2015 (ES6)
- Here are some ways backticks can be used in JavaScript:
    - **Template Strings:** Allow for more readable and versatile string formatting

```
const name = 'Alice';
const greeting = `Hello, ${name}!`;
console.log(greeting); // Outputs: "Hello, Alice!"
```

- **Multi-line Strings:** Create multi-line strings without explicit line breaks

```
const message = `This is a multi-line string using template strings.`;
console.log(message); // Outputs: This is a multi-line string using template strings.
```

- **HTML Templates:** Can generate dynamic HTML content

```
const title = 'My Web Page';
const content = `
<html>
  <head>
    <title>${title}</title>
  </head>
  <body>
    <h1>Welcome to ${title}</h1>
  </body>
</html>
`;
```

- **SQL Queries:** Can generate dynamic SQL queries

```
const username = 'user123';
const query = `SELECT * FROM users WHERE username = '${username}'`;
```

Promises

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

promise.then(onFulfilled).catch(onRejected);
```

Classes

- **Creating an Object from a Class:**

```
class Person {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }

  greet() {
    console.log(`Hello, my name is ${this.name} and I am ${this.age} years old.`);
  }
}

const person1 = new Person('Max', 27);
person1.greet(); // Outputs: "Hello, my name is Max and I am 27 years old."
```

- **Inheritance with Classes:**

```
class Animal {
  constructor(name) {
    this.name = name;
  }

  speak() {
    console.log(`${this.name} makes a sound.`);
  }
}

class Dog extends Animal {
  speak() {
    console.log(`${this.name} barks.`);
  }
}

const dog = new Dog('Buddy');
dog.speak(); // Outputs: "Buddy barks."
```

- **Using Multiple Instances:**

```
const person1 = new Person('Alice', 30);
const person2 = new Person('Bob', 25);

person1.greet(); // Outputs: "Hello, my name is Alice and I am 30 years old."
person2.greet(); // Outputs: "Hello, my name is Bob and I am 25 years old."
```

Modules

- Exporting Variables from “myModule.js” class

```
// myModule.js
export const name = 'Alice';
export const age = 30;
```

- Importing Variables from “myModule.js” class to “anotherModule.js” class

```
// anotherModule.js
import { name, age } from './myModule.js';

console.log(`Name: ${name}, Age: ${age}`);
```

Array Destructuring

```
const colors = ['red', 'green', 'blue'];
const [firstColor, secondColor] = colors;
console.log(firstColor); // Outputs: 'red'
console.log(secondColor); // Outputs: 'green'
```

Destructuring Function Parameters

```
function printUser({ name, age }) {
  console.log(`Name: ${name}, Age: ${age}`);
}

const user = { name: 'Max', age: 27 };
printUser(user); // Outputs: 'Name: Max, Age: 27'
```

Default Values

```
const person = { name: 'Bob' };
const { name, age = 25 } = person;
console.log(name); // Outputs: 'Bob'
console.log(age); // Outputs: 25 (default value)
```

Nested Destructuring

```
const data = { user: { name: 'Charlie', age: 28 } };
const { user: { name, age } } = data;
console.log(name); // Outputs: 'Charlie'
console.log(age); // Outputs: 28
```

Rest Parameter

```
const numbers = [1, 2, 3, 4, 5];
const [first, second, ...rest] = numbers;
console.log(first); // Outputs: 1
console.log(second); // Outputs: 2
console.log(rest); // Outputs: [3, 4, 5]
```

Async/Await

```javascript
async function fetchData() {
  try {
    const response = await fetch('url');
    const data = await response.json();
  } catch (error) {
    console.error(error);
  }
}
```

Map and Set

```javascript
const map = new Map();
map.set('key', 'value');
map.get('key');
map.has('key');
map.delete('key');

const set = new Set();
set.add('value');
set.has('value');
set.delete('value');
```

Local Storage

```javascript
localStorage.setItem('key', 'value');
localStorage.getItem('key');
localStorage.removeItem('key');
```