JavaScript Cheat Sheet

Updated on 22 January 2024 by Husaif Sayyed

JavaScript is a versatile and widely-used programming language that plays a crucial role in modern web development. Whether you're a seasoned developer or just starting, having a handy JavaScript cheatsheet can be a valuable resource to quickly reference key concepts and syntax. In this blog post, we'll provide you with a comprehensive JS cheat sheet to help you in your coding journey.

Links

JavaScript Doc

Download JavaScript Cheat Sheet PDF

★ Want More Cheat Sheet

Hello World! JavaScript Program

```
console.log('Hello World');
```

This will print Hello World in Console

Variables and Data Types

`var`, `let`, and `const` declarations is used to declaring variables.

Primitive data types: string, number, boolean

Complex data types: object, array

Variable Declarations

```
var variable1;
let variable2 = "Hello";
const constantVariable = 42;
```

Data Types

```
let str = "Hello, World!";
let num = 42;
let bool = true;
let arr = [1, 2, 3];
let obj = { key: "value" };
```

Operators

```
let x = 5;
let y = 10;
```

Arithmetic Operators

```
let sum = x + y;
let difference = x - y;
let product = x * y;
let quotient = x / y;
let remainder = x % y;
```

Comparison Operators

```
console.log(x === y); // false
console.log(x < y); // true
```

Logical Operators

```
let isTrue = true;
let isFalse = false;
console.log(isTrue && isFalse); // false
console.log(isTrue || isFalse); // true
console.log(!isTrue); // false
```

Control Flow

If-Else Statement

```
let day = "Monday";

if (day === "Monday") {
  console.log("It's the start of the week!");
} else if (day === "Friday") {
  console.log("Weekend is almost here!");
} else {
  console.log("It's a regular day.");
}
```

Switch Statement

```
switch (day) {
  case "Monday":
    console.log("It's the start of the week!");
    break;
  case "Friday":
    console.log("Weekend is almost here!");
    break;
  default:
    console.log("It's a regular day.");
}
```

Loops

For Loop

```
for (let i = 0; i < 5; i++) {
  console.log(i);
}
```

While Loop

```
let count = 0;
while (count < 5) {
  console.log(count);
  count++;
}
```

Do-While Loop

```
let num = 0;
do {
  console.log(num);
  num++;
} while (num < 5);
```

For...In Loop (Objects)

```
const person = { name: "John", age: 30 };
for (let key in person) {
  console.log(key, person[key]);
}
```

Functions

Function Declaration

```
function greet(name) {
  console.log(`Hello, ${name}!`);
}
```

Arrow Function

```
const add = (a, b) => a + b;
```

Function with Parameters and Return Value

```
function multiply(x, y) {
  return x * y;
}
```

Arrays and Objects

Arrays

```
const colors = ["red", "green", "blue"];
colors.push("yellow");
colors.pop();
colors.shift();
colors.unshift("purple");
const slicedColors = colors.slice(1, 3);
colors.splice(1, 2);
```

Objects

```
const person = {
  name: "Alice",
  age: 25,
  sayHello: function () {
    console.log(`Hello, my name is ${this.name}`);
  },
};

console.log(person.name); // "Alice"
person.sayHello(); // "Hello, my name is Alice"
```

Error Handling

```
try {
  // Code that might throw an error
  throw new Error("An error occurred");
} catch (error) {
  // Handle the error
  console.error(error.message);
} finally {
  // Code that will always run, whether there's an error or not
  console.log("Finally block executed");
}
```

Asynchronous JavaScript

setTimeout

```
setTimeout(() => {
  console.log("Delayed message");
}, 1000);
```

setInterval

```
setInterval(() => {
  console.log("Repeated message");
}, 2000);
```

Promises

```
const fetchData = () => {
  return new Promise((resolve, reject) => {
    // Simulate asynchronous task
    setTimeout(() => {
      const data = "Data fetched successfully";
      resolve(data);
      // Uncomment the next line to simulate an error
      // reject(new Error("Failed to fetch data"));
    }, 1000);
  });
};

fetchData()
  .then((data) => console.log(data))
  .catch((error) => console.error(error))
  .finally(() => console.log("Promise settled"));
```

Async/Await

```
const fetchDataAsync = async () => {
  try {
    const data = await fetchData();
    console.log(data);
  } catch (error) {
    console.error(error);
  } finally {
    console.log("Async/Await function completed");
  }
};

fetchDataAsync();
```

Template Literals

```
const name = "John";
const age = 25;
console.log(`My name is ${name} and I am ${age} years old.`);
```

Destructuring

Destructuring Arrays

```
const numbers = [1, 2, 3];
const [first, second, third] = numbers;
console.log(first, second, third);
```

Destructuring Objects

```
const person = { firstName: "Alice", lastName: "Doe" };
const { firstName, lastName } = person;
```

Spread and Rest Operators

Spread Operator (Arrays)

```
const arr1 = [1, 2, 3];
const arr2 = [...arr1, 4, 5];
console.log(arr2);
```

Spread Operator (Objects)

```
const obj1 = { a: 1, b: 2 };
const obj2 = { ...obj1, c: 3, d: 4 };
console.log(obj2);
```

Rest Parameter

```
const sum = (...numbers) => numbers.reduce((acc, num) =>
console.log(sum(1, 2, 3, 4, 5));
```

ES6 Classes

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

  makeSound() {
    console.log(`${this.name} says ${this.sound}`);
  }
}

const cat = new Animal("Cat", "Meow");
cat.makeSound();
```

Promises and Fetch API

Fetch API

```
fetch("https://jsonplaceholder.typicode.com/posts/1")
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error(error));
```

Promise.all

```
const promise1 = Promise.resolve("Hello");
const promise2 = new Promise((resolve) => setTimeout({
Promise.all([promise1, promise2])
  .then(values => console.log(values));
```

Local Storage

Local Storage

```
localStorage.setItem("username", "John");
const storedUsername = localStorage.getItem("username
console.log(storedUsername);
```

Session Storage

```
sessionStorage.setItem("token", "abc123");
const storedToken = sessionStorage.getItem("token");
console.log(storedToken);
```

Regular Expressions

```
// Regular Expressions
const regex = /\b\d{3}-\d{2}-\d{4}\b/;
const ssn = "123-45-6789";
console.log(regex.test(ssn)); // true
```

Congratulations on reaching the end of this JavaScript Programming Language Cheat sheet! This resource is designed to make your coding experience easy and efficient. Feel free to bookmark this page or download the PDF for future reference. Happy JavaScript Programming!