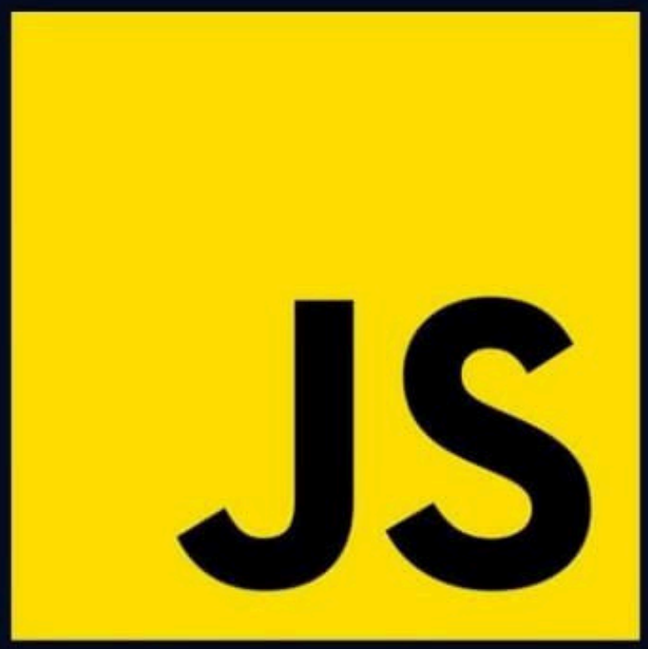
JS

Advanced JS Cheat sheet



JS

Arrays in JavaScript

```
const fruits = ['🍎', '🍌', '🍓', '🍑', '🍎', '🍎'];

// converts the array to a string
fruits.toString(); // 🍎,🍌,🍓,🍑,🍎,🍎

// adds element at the end of the array
fruits.push('🫒'); // ['🍎', '🍌', '🍓', '🍑', '🍎', '🍎', '🫒']

// removes the last element of the array
fruits.pop(); // '🫒'

// checks if the array contains an element
fruits.includes('🍌'); // true

// returns the index of the element
fruits.indexOf('🍑'); // 3

// join the elements of the array with the given separator
fruits.join('+'); // 🍎+🍌+🍓+🍑+🍎+🍎

// return a portion of the array
fruits.slice(1, 3); // ['🍌', '🍓']

// add elements to the array
fruits.splice(1, 0, '🥑'); // ['🍎', '🥑', '🍌', '🍓', '🍑', '🍎', '🍎']
```

JS

Objects in JavaScript

index.js

```
const person = {
    name: 'John',
    age: 30,
    gender: 'male',
};

const jobObject = {
    job: 'developer',
    salary: 1000,
};

// get all object keys
Object.keys(person); // ['name', 'age', 'gender']
// get all object values
Object.values(person); // ['John', 30, 'male']
// get all object entries
Object.entries(person); // [ [ 'name', 'John' ], [ 'age', 30 ], [ 'gender', 'male' ]]

// assign object to another object
Object.assign(person, jobObject);
// { name: 'John', age: 30, gender: 'male', job: 'developer', salary: 1000 }
```

JS

Scope in JavaScript

```js
/* global scope */
const PIE = 3.14;

function foo() {
    console.log(PIE); // 3.14

    /* function scope */
    const age = 32;
    console.log(age); // 32
}

/* block scope */
if (true) {
    const fullName = 'John Doe';
    console.log(fullName); // John Doe
}

console.log(PIE); // 3.14
console.log(age); // ReferenceError: age is not defined
console.log(fullName); // ReferenceError: fullName is not defined
```

JS

Date in JavaScript

index.js

```
const date = new Date(); // 2023-01-22T09:44:48.175Z
date.getDate(); // month's date: 22
date.getMonth(); // Month with 0 index: 0
date.getFullYear(); // Year: 2023
date.getHours(); // Hours: 9
date.getMinutes(); // Minutes: 44
date.getSeconds(); // Seconds: 48
date.getMilliseconds(); // Millisecond: 175
date.getTime(); // Time: 1648101488175
date.setDate(23); // Set date: 23
date.setMonth(3); // Set month: 3
date.setFullYear(2024); // Set year: 2024
date.setHours(10); // Set hours: 10
date.setMinutes(45); // Set minutes: 45
date.setSeconds(49); // Set seconds: 49
date.setMilliseconds(176); // Set Milliseconds: 176
date.setTime(16481014881 76); // Set time: 1648101488176
```

JS

Events in JavaScript

index.html

```
<!-- when use clicks -->
<input type="text" onclick="" />
<!-- when user double clicks -->
<input type="text" ondblclick="">
<!-- when user moves the mouse over an element, it's called mouse down-->
<input type="text" onmousedown="">
<!-- when an element looses focus -->
<input type="text" onblur="">
<!-- when an element gets focus -->
<input type="text" onfocus="">
<!-- when a user moves the mouse over an element -->
<input type="text" onmouseover="">
<!-- when a user moves the mouse out of an element -->
<input type="text" onmouseout="">
<!-- when there is a change -->
<input type="text" onchange="">
<!-- when a user presses a key -->
<input type="text" onkeydown="">
<!-- when a user releases a key -->
<input type="text" onkeyup="">
<!-- when a user presses a key -->
<input type="text" onkeypress="">
<!-- when a user submits a form -->
<form onsubmit=""></form>
<!-- when a user resets a form -->
<form onreset=""></form>
<!-- when a user selects a text -->
<input type="text" onselect="">
```

JS

Error handling in JS

index.js

```
/* error handling in JavaScript */
function foo() {
    // try catch block
    try {
        // ...
    } catch (e) {
        // catch error
        // ...
    }
}

// executes when when a JavaScript Promise that
// has no rejection handler is rejected
window.addEventListener('unhandledrejection', function () {});
```

JS

Async/Await JavaScript

```
// Used async to make the function act asynchronous
async function getWeatherData() {
  try {

    // Used await to make the code wait until promise returns a result
    const res = await fetch('https://jsonplaceholder.typicode.com/posts')
    const data = await res.json()

    return data
  } catch (err) {
    console.log(err)
  }
}
```