JavaScript Cheat Sheet

ZolaXTech · quick reference for everyday JS

Variables

```
let x = 5;
const y = 10;
var z = 15; // avoid

// const = can't reassign
// let = block scoped
```

Data Types

```
typeof 42        // 'number'
typeof 'hi'      // 'string'
typeof true      // 'boolean'
typeof undefined // 'undefined'
typeof null      // 'object'
typeof {}        // 'object'
typeof []        // 'object'
```

Template Literals

```
const name = 'Zola';
`Hello, ${name}!`
`Line1\nLine2` // multiline
```

Arrow Functions

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

const greet = name => {
  return `Hi ${name}`;
};
```

Arrays

```
const arr = [1, 2, 3];
arr.push(4);        // add end
arr.pop();          // remove end
arr.shift();        // remove start
arr.unshift(0);     // add start
arr.slice(1, 3);    // copy range
arr.splice(1, 1);   // remove at i
```

Array Methods

```
arr.map(x => x * 2)
arr.filter(x => x > 1)
arr.reduce((a, b) => a + b, 0)
arr.find(x => x === 2)
arr.includes(3)
arr.forEach(x => console.log(x))
arr.sort((a, b) => a - b)
```

Objects

```
const obj = { name: 'Zola', age: 5 };
obj.name         // dot access
obj['name']      // bracket access
const { name } = obj; // destructure
Object.keys(obj)
Object.values(obj)
Object.entries(obj)
```

Spread / Rest

```
const a = [1, 2];
const b = [...a, 3];  // spread

function sum(...nums) { // rest
  return nums.reduce((a, b) => a + b);
}
```

Conditionals

```
if (x > 0) { ... }
else if (x === 0) { ... }
else { ... }

x > 0 ? 'pos' : 'neg'; // ternary
x ?? 'default';        // nullish
```

Loops

```
for (let i = 0; i < 5; i++) {}
for (const item of arr) {}
for (const key in obj) {}
arr.forEach(item => {});
while (x < 5) { x++; }
```

Promises

```
fetch(url)
  .then(res => res.json())
  .then(data => console.log(data))
  .catch(err => console.error(err));
```

Async / Await

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

Classes

```
class Animal {
  constructor(name) {
    this.name = name;
  }
  speak() {
    return `${this.name} makes a sound`;
  }
}
```

Modules

```
// export
export const x = 5;
export default function() {}

// import
import x, { y } from './file.js';
```

Common Gotchas

```
0 == '0'   // true  (loose)
0 === '0'  // false (strict)
NaN === NaN // false, use isNaN()
[] == false // true (avoid ==)
typeof NaN  // 'number'
```

Prefer === over == to avoid type coercion surprises.