JavaScript Full Cheatsheet with Examples

1. Variables

```
let x = 10;
const PI = 3.14;
var y = 'Hello';
```

2. Data Types

```
let str = 'Hello';
let num = 123;
let bool = true;
let undef;
let obj = null;
let sym = Symbol('id');
```

3. Operators

```
let sum = 5 + 3;
let isEqual = (5 === '5');
let notTrue = !true;
```

4. Functions

```
function greet(name) {
  return 'Hello ' + name;
}
const greetArrow = name => 'Hi ' + name;
```

5. Loops

```
for(let i=0; i<5; i++) {}
while(condition) {}
do {} while(condition);
arr.forEach(el => console.log(el));
```

6. Conditionals

```
if (x > 10) {}
else if (x === 10) {}
else {}
switch(value) { case 1: break; default: break; }
```

7. Objects

```
let person = { name: 'John', age: 30 };
console.log(person.name);
let { name } = person;
```

8. Arrays

JavaScript Full Cheatsheet with Examples

```
let arr = [1, 2, 3];
arr.push(4);
let doubled = arr.map(x => x * 2);
```

9. Classes

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

10. Promises

```
new Promise((resolve, reject) => { resolve('done'); })
.then(res => console.log(res));
async function f() { await fetch(url); }
```

11. ES6 Features

```
let [a, b] = [1, 2];
let obj = {a, b};
let arr2 = [...arr];
```

12. DOM Manipulation

```
document.querySelector('#id').addEventListener('click', () => {});
document.createElement('div');
```

13. Error Handling

```
try {
  riskyCode();
} catch (e) {
  console.error(e);
} finally { console.log('Done'); }
```

14. Modules

```
// module.js: export default function greet() {}
// app.js: import greet from './module.js';
```

15. Scope & Closures

```
function outer() {
  let count = 0;
  return function inner() {
    count++;
    return count;
```

```
  };
}
```

16. Event Loop

```
// JS is single-threaded but async via event loop
setTimeout(() => console.log('Hi'), 0);
console.log('Bye');
```

17. Memory Management

```
// JS uses garbage collection for unreferenced memory
```

18. Higher-order Functions

```
function operate(fn, x) {
  return fn(x);
}
operate(x => x*2, 5);
```

19. Regular Expressions

```
let regex = /abc/;
regex.test('abc123');
```

20. Destructuring

```
let {name, age} = person;
let [first, second] = arr;
```