ES6 常用语法

ES6 常用语法精华整理

1. 变量声明

语法	说明	示例
`let`	块级作用域变量	`let x = 10;`
`const`	块级作用域常量	`const PI = 3.14;`

特点：

- 替代 `var`，解决变量提升问题
- `const` 声明必须初始化且不可重新赋值
- 支持块级作用域（`{}` 内有效）

2. 箭头函数

JavaScript

```
// 传统函数
function sum(a, b) {
 return a + b;
}

// 箭头函数
const sum = (a, b) => a + b;

// 单参数简写
const square = n => n * n;

// 无参数
const sayHello = () => console.log('Hello!');
```

特点：

- 更简洁的语法
- 没有自己的 `this`（继承父级作用域）
- 不能作为构造函数（不能使用 `new`）

- 没有 `arguments` 对象

3. 模板字符串

JavaScript

```
const name = 'John';
const age = 30;

// 多行字符串
const message = `
  Hello ${name}!
  You are ${age} years old.
  Next year you'll be ${age + 1}.
`;

console.log(message);
```

特点：

- 支持多行文本
- 支持变量插值 `${}`
- 支持表达式计算

4. 解构赋值

JavaScript

```
// 数组解构
const [first, second] = [1, 2, 3];
console.log(first); // 1

// 对象解构
const { name, age } = { name: 'Alice', age: 25 };
console.log(name); // 'Alice'

// 函数参数解构
function greet({ name, age }) {
  return `Hello ${name} (${age})`;
}
```

特点：

- 简化数据提取
- 支持默认值：`const { role = 'user' } = person`
- 支持重命名：`const { name: userName } = person`

5. 默认参数

```JavaScript
function createUser(name = 'Guest', role = 'user') {
  return { name, role };
}

console.log(createUser()); // {name: 'Guest', role: 'user'}
console.log(createUser('Alice')); // {name: 'Alice', role: 'user'}
```

特点：

- 参数默认值在未传值或 `undefined` 时生效
- 默认值可以是表达式或函数调用

6. 剩余参数 & 扩展运算符

```JavaScript
// 剩余参数（函数参数）
function sum(...numbers) {
  return numbers.reduce((total, num) => total + num, 0);
}
sum(1, 2, 3); // 6

// 扩展运算符（数组）
const arr1 = [1, 2];
const arr2 = [3, 4];
const combined = [...arr1, ...arr2]; // [1, 2, 3, 4]

// 扩展运算符（对象）
const obj1 = { a: 1, b: 2 };
const obj2 = { ...obj1, c: 3 }; // {a:1, b:2, c:3}
```

7. 对象字面量增强

```JavaScript
```

```
const name = 'John';
const age = 30;

// 属性简写
const person = { name, age };

// 方法简写
const app = {
  start() {
    console.log('App started');
  }
};

// 计算属性名
const key = 'userRole';
const user = {
  [key]: 'admin'
};
```

8. Promise

JavaScript

```
const fetchData = () => {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      Math.random() > 0.5
        ? resolve('Data received')
        : reject('Error occurred');
    }, 1000);
  });
};

fetchData()
  .then(data => console.log(data))
  .catch(error => console.error(error));
```

高级用法：

JavaScript

```
// Promise.all - 所有成功才算成功
Promise.all([promise1, promise2])
  .then(results => console.log(results));

// Promise.race - 第一个完成的结果
Promise.race([promise1, promise2])
  .then(firstResult => console.log(firstResult));
```

9. async/await

```JavaScript
async function getUserData() {
  try {
    const response = await fetch('/api/user');
    const data = await response.json();
    return data;
  } catch (error) {
    console.error('Fetch error:', error);
    throw error;
     }
}

// 使用示例
getUserData()
  .then(user => console.log(user))
  .catch(console.error);
```

特点：

- 使异步代码看起来像同步
- 基于 Promise 实现
- 必须用 `try/catch` 处理错误

10. 模块化

```JavaScript
// math.js
export const PI = 3.14;

export function sum(a, b) {
  return a + b;
}

// 默认导出
export default function multiply(a, b) {
  return a * b;
}

// app.js
import mult, { PI, sum } from './math.js';

console.log(PI); // 3.14
console.log(sum(2, 3)); // 5
console.log(mult(2, 3)); // 6
```

11. Class 类

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

  greet() {
    return `Hello, my name is ${this.name}`;
     }

  // 静态方法
  static createAdult(name) {
    return new Person(name, 18);
     }
}

class Employee extends Person {
  constructor(name, age, position) {
    super(name, age);
    this.position = position;
     }
}

const john = new Person('John', 30);
const mary = Employee.createAdult('Mary');
```

12. 其他重要特性

语法	说明	示例
Map/Set	新数据结构	`const map = new Map();`
Symbol	唯一标识符	`const id = Symbol('id');`
迭代器	可迭代对象	`for (const item of array) {...}`
生成器	按需生成值	`function* gen() { yield 1; }`
尾调用优化	递归优化	严格模式下自动优化
指数操作符	幂运算	`2 ** 3` → 8

ES6+ 实用技巧

1. **可选链操作符**：避免深层属性访问错误

```JavaScript
```

```
const city = user?.address?.city; // 不会抛出错误
```

2. **空值合并运算符**：提供默认值

JavaScript

```
const username = input ?? 'Anonymous';
```

3. **对象属性动态名**：

JavaScript

```
const key = 'name';
const person = { [key]: 'John' };
```

4. **数组快速操作**：

JavaScript

```
// 数组去重
const unique = [...new Set([1,2,2,3])];

// 数组查找
const found = array.find(item => item.id === 10);
```

5. **函数参数解构默认值**：

JavaScript

```
function setup({ width = 100, height = 200 } = {}) {
  console.log(width, height);
    }
```

ES6 是现代 JavaScript 开发的基石，掌握这些核心语法能大幅提升开发效率和代码质量。建议在项目中配置 Babel 转译器以兼容旧浏览器。