字符串，对象，数组的常用方法加解析和案例

JavaScript 字符串、对象和数组常用方法详解

一、字符串(String)常用方法

方法	说明	参数	返回值	案例
length	字符串长度	-	数字	`'hello'.length` → 5
indexOf()	查找子串位置	(子串, 开始位置)	索引/-1	`'JS'.indexOf('S')` → 1
slice()	提取子字符串	(开始索引, 结束索引)	新字符串	`'JavaScript'.slice(0,4)` → "Java"
replace()	替换字符串	(查找值, 替换值)	新字符串	`'abc'.replace('b','x')` → "axc"
toUpperCase()	转大写	-	新字符串	`'hello'.toUpperCase()` → "HELLO"
toLowerCase()	转小写	-	新字符串	`'WORLD'.toLowerCase()` → "world"
split()	分割字符串	(分隔符)	数组	`'a,b,c'.split(',')` → ["a","b","c"]
trim()	去两端空格	-	新字符串	`' text '.trim()` → "text"
includes()	是否包含	(子串)	布尔值	`'JS'.includes('S')` → true
startsWith()	是否以...开始	(子串)	布尔值	`'JS'.startsWith('J')` → true

JavaScript

```
// 字符串综合案例：格式化用户名
function formatUsername(name) {
  return name.trim().toLowerCase().replace(/\b\w/g, char => char.toUpperCase());
}

console.log(formatUsername('  john DOE ')); // "John Doe"
```

二、对象(Object)常用方法

方法	说明	参数	返回值	案例
Object.keys()	获取属性名数组	(对象)	数组	`Object.keys({a:1,b:2})` → ["a","b"]
Object.values()	获取属性值数组	(对象)	数组	`Object.values({a:1,b:2})` → [1,2]
Object.entries()	获取键值对数组	(对象)	数组	`Object.entries({a:1,b:2})` → [["a",1],["b",2]]
Object.assign()	合并对象	(目标, 源)	新对象	`Object.assign({a:1},{b:2})` → {a:1,b:2}
Object.freeze()	冻结对象	(对象)	冻结对象	`Object.freeze(obj)` 禁止修改
hasOwnProperty()	检测属性	(属性名)	布尔值	`obj.hasOwnProperty('name')`
JSON.stringify()	转JSON字符串	(对象)	字符串	`JSON.stringify({x:1})` → '{"x":1}'
JSON.parse()	解析JSON	(JSON字符串)	对象	`JSON.parse('{"x":1}')` → {x:1}

JavaScript

```
// 对象综合案例：合并并过滤对象
function mergeObjects(...objects) {
  const merged = Object.assign({}, ...objects);
  return Object.fromEntries(
    Object.entries(merged).filter(([_, v]) => v !== null)
    );
}

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

三、数组(Array)常用方法

方法	说明	参数	返回值	案例
push()	添加元素到末尾	(元素)	新长度	`[1,2].push(3)` → [1,2,3]
pop()	移除末尾元素	-	移除元素	`[1,2,3].pop()` → 3 (数组变[1,2])
shift()	移除首元素	-	移除元素	`[1,2,3].shift()` → 1 (数组变[2,3])
unshift()	添加元素到开头	(元素)	新长度	`[2,3].unshift(1)` → [1,2,3]
slice()	提取子数组	(开始, 结束)	新数组	`[1,2,3,4].slice(1,3)` → [2,3]
splice()	增删元素	(起始, 删除数, 新增项)	被删数组	`[1,2,3].splice(1,1,'a')` → (数组变[1,'a',3])
concat()	合并数组	(数组)	新数组	`[1,2].concat([3,4])` → [1,2,3,4]
join()	连接成字符串	(分隔符)	字符串	`[1,2,3].join('-')` → "1-2-3"
map()	映射新数组	(回调函数)	新数组	`[1,2,3].map(x => x*2)` → [2,4,6]
filter()	过滤数组	(回调函数)	新数组	`[1,2,3].filter(x => x>1)` → [2,3]
reduce()	累积计算	(回调, 初始值)	计算结果	`[1,2,3].reduce((a,b)=>a+b,0)` → 6

方法	说明	参数	返回值	案例
find()	查找元素	(回调函数)	元素/undefined	`[1,2,3].find(x => x>1)` → 2
some()	是否满足条件	(回调函数)	布尔值	`[1,2,3].some(x => x>2)` → true
every()	是否全部满足	(回调函数)	布尔值	`[1,2,3].every(x => x>0)` → true
sort()	排序	(比较函数)	排序后数组	`[3,1,2].sort((a,b)=>a-b)` → [1,2,3]

JavaScript

```
// 数组综合案例：数据处理
const users = [
  { id: 1, name: 'John', age: 25 },
  { id: 2, name: 'Alice', age: 30 },
  { id: 3, name: 'Bob', age: 20 }
];

// 1. 过滤年龄大于20的用户
// 2. 提取用户名并转为大写
// 3. 按字母顺序排序
const result = users
  .filter(user => user.age > 20)
  .map(user => user.name.toUpperCase())
    .sort();

console.log(result); // ["ALICE", "JOHN"]
```

四、方法使用技巧总结

1. **链式调用**：合理组合方法实现复杂操作

JavaScript

```
  ' Hello World '
        .trim()
        .toLowerCase()
    .split(' ')
    .map(word => word[0].toUpperCase() + word.slice(1))
    .join(' ');
  // "Hello World"
```

2. **深拷贝对象/数组：**

```JavaScript
const deepCopy = JSON.parse(JSON.stringify(original));
```

3. **数组去重：**

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

4. **对象动态属性：**

```JavaScript
const dynamicKey = 'name';
const obj = { [dynamicKey]: 'John', age: 30 };
```

5. **数组空位处理：**

```JavaScript
[1,,3].flatMap(x => x || 0); // [1,0,3]
```

6. **函数参数处理：**

```JavaScript
function sum(...numbers) {
  return numbers.reduce((a,b)=>a+b);
    }
sum(1,2,3); // 6
```

掌握这些核心方法及其组合应用，能够高效处理 JavaScript 中的各种数据操作需求。