JavaScript 基础整理

# JavaScript 基础整理手册

## 1. 变量与数据类型

| 类别 | 示例/说明 |
| --- | --- |
| 变量声明 | `let age = 25;` `const PI = 3.14;` `var name = 'John';` |
| 基本数据类型 | `String`, `Number`, `Boolean`, `undefined`, `null`, `Symbol`, `BigInt` |
| 类型检测 | `typeof 'hello'` → "string" |
| 类型转换 | `Number('123')`, `String(123)`, `Boolean(1)` |
| 模板字符串 | `` `Hello ${name}!` `` |

## 2. 运算符

| 类型 | 运算符 |
| --- | --- |
| 算术运算符 | `+`, `-`, `*`, `/`, `%`, `**` (幂运算) |
| 比较运算符 | `==`, `===`, `!=`, `!==`, `>`, `<`, `>=`, `<=` |
| 逻辑运算符 | `&&`, ` |
| 赋值运算符 | `=`, `+=`, `-=`, `*=`, `/=`, `%=` |
| 三元运算符 | `condition ? expr1 : expr2` |

## 3. 流程控制

JavaScript

```
// 条件语句
if (age > 18) {
  console.log("Adult");
} else if (age > 13) {
  console.log("Teenager");
} else {
  console.log("Child");
}

// Switch语句
switch(day) {
  case 1:
    console.log("Monday");
    break;
  default:
    console.log("Weekend");
}

// 循环结构
for (let i = 0; i < 5; i++) { /*... */ }

while (condition) { /*... */ }

do { /*... */ } while (condition);
```

### 4. 函数

JavaScript

```
// 函数声明
function sum(a, b) {
  return a + b;
}

// 箭头函数
const multiply = (a, b) => a * b;

// 默认参数
function greet(name = "Guest") {
  return `Hello ${name}`;
}

// 剩余参数
function showItems(...items) {
  console.log(items);
}
```

### 5. 对象与数组

| 操作 | 示例 |
| --- | --- |
| 创建对象 | `const person = { name: 'Alice', age: 30 }` |
| 访问属性 | `person.name` 或 `person['name']` |
| 创建数组 | `const nums = [1, 2, 3]` |
| 数组方法 | `push()`, `pop()`, `shift()`, `unshift()`, `slice()`, `splice()` |
| 高阶函数 | `map()`, `filter()`, `reduce()`, `forEach()`, `find()` |

### 6. 字符串操作

```JavaScript
const str = "JavaScript";

// 常用方法
str.length;        // 10
str.toUpperCase(); // "JAVASCRIPT"
str.substring(0,4); // "Java"
str.includes("Script"); // true
str.split("");     // ["J","a","v","a","S","c","r","i","p","t"]
```

### 7. 日期与时间

```JavaScript
const now = new Date();

now.getFullYear();  // 当前年份
now.getMonth();     // 月份 (0-11)
now.getDate();      // 日期 (1-31)
now.getHours();     // 小时 (0-23)

// 格式化日期
now.toLocaleDateString(); // "2023/10/28"
```

### 8. 错误处理

```JavaScript
```

```
try {
  // 可能出错的代码
    nonExistentFunction();
} catch (error) {
  console.error("Error:", error.message);
} finally {
  console.log("This always runs");
}

// 抛出错误
throw new Error("Custom error message");
```

### 9. 异步编程

JavaScript

```
// Promise
const fetchData = () => new Promise((resolve, reject) => {
  setTimeout(() => resolve("Data received"), 1000);
});

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

// Async/Await
async function getData() {
  try {
    const data = await fetchData();
    console.log(data);
  } catch (err) {
    console.error(err);
    }
}
```

### 10. DOM 操作

JavaScript

```
// 选择元素
const btn = document.getElementById('myBtn');
const items = document.querySelectorAll('.item');

// 事件监听
btn.addEventListener('click', () => {
  console.log('Button clicked');
});

// 修改内容
document.querySelector('#title').textContent = 'New Title';

// 样式操作
element.style.backgroundColor = 'blue';
element.classList.add('active');
```

### 11. 实用代码片段

JavaScript

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

// 对象合并
const merged = {...obj1, ...obj2};

// 深浅拷贝
const shallowCopy = [...array];
const deepCopy = JSON.parse(JSON.stringify(obj));

// 延迟执行
setTimeout(() => { /*...*/ }, 1000);

// 定时循环
setInterval(() => { /*...*/ }, 5000);
```

### 12. ES6+ 新特性

| 特性 | 示例 |
| --- | --- |
| 解构赋值 | `const { name, age } = person;` |
| 模块化 | `import module from './module.js';` |
| 可选链 | `user?.address?.city` |
| 空值合并 | `const name = input ?? 'Anonymous';` |
| Promise.allSettled | `Promise.allSettled([promise1, promise2])` |

| 特性 | 示例 |
| --- | --- |
| 动态导入 | `const module = await import('./module.js')` |

### 13. 最佳实践原则

1. **命名规范**：使用驼峰命名法（`myVariableName`）
2. **避免全局污染**：使用模块作用域
3. **严格模式**：始终在文件顶部添加 `'use strict';`
4. **错误优先回调**：`function(err, data) { ... }`
5. **代码格式化**：使用 ESLint + Prettier
6. **内存管理**：及时清除事件监听器和定时器
7. **性能优化**：避免嵌套循环，使用事件委托

```JavaScript
// 事件委托示例
document.getElementById('list').addEventListener('click', event => {
  if (event.target.matches('li.item')) {
    console.log('Item clicked:', event.target.textContent);
    }
});
```

这份手册覆盖了 JavaScript 开发中最常用的核心概念与技巧，可作为日常开发的快速参考指南。

(注:文档部分内容可能由AI生成)