JavaScript 快速参考

ES6+、DOM、事件、Fetch API、async/await

基础

变量

```
let name = "Alice";    // reassignable
const PI = 3.14;       // constant
var old = "avoid";     // function-scoped (legacy)
```

数据类型

string	文本：**"hello"** 或 **'hello'**
number	整数或浮点数：**42**、**3.14**
boolean	**true** / **false**
null	主动设置的空值
undefined	已声明但未赋值
object	键值对：**{ a: 1 }**
array	有序列表：**[1, 2, 3]**
symbol	唯一标识符

类型检测与转换

```
typeof "hello"     // "string"
typeof 42          // "number"
Number("42")       // 42
String(100)        // "100"
parseInt("3.9")    // 3
parseFloat("3.14") // 3.14
```

字符串

模板字面量

```
const name = "Alice";
`Hello, ${name}!`       // Hello, Alice!
`Total: ${2 + 3}`       // Total: 5
`Multi
 line string`
```

字符串方法

s.length	字符数量
s.toUpperCase()	大写副本
s.toLowerCase()	小写副本
s.trim()	去除首尾空白
s.split(",")	分割为数组
s.includes("x")	包含检测 → bool
s.indexOf("x")	第一个索引（-1 表示无）
s.slice(1, 4)	按索引截取子串
s.replace(a, b)	替换第一个匹配
s.replaceAll(a, b)	替换所有匹配
s.startsWith(x)	检测前缀 → bool
s.endsWith(x)	检测后缀 → bool
s.padStart(n, c)	在开头填充至长度 n

数组

创建与访问

```
const fruits = ["apple", "banana", "cherry"];
fruits[0]          // "apple"
fruits.length      // 3
fruits.at(-1)      // "cherry"
```

可变方法

arr.push(x)	在末尾添加
arr.pop()	移除并返回最后一项
arr.unshift(x)	在开头添加
arr.shift()	移除并返回第一项
arr.splice(i, n)	从索引 i 移除 n 项
arr.sort()	原地排序（字典序）
arr.reverse()	原地反转

不可变方法

arr.map(fn)	转换每个元素
arr.filter(fn)	保留 fn 返回 true 的元素
arr.reduce(fn, init)	累积为单个值
arr.find(fn)	第一个匹配项或 undefined
arr.findIndex(fn)	第一个匹配的索引（-1）
arr.includes(x)	包含检测 → bool
arr.slice(a, b)	部分浅拷贝
arr.join(",")	连接为字符串
arr.forEach(fn)	遍历（无返回值）
[...a, ...b]	合并数组（展开）

对象

创建与访问

```
const user = { name: "Alice", age: 20 };
user.name          // "Alice"
user["age"]        // 20
user.gpa = 3.85;   // add/update
```

解构与展开

```
const { name, age } = user;
const copy = { ...user, age: 21 };
```

对象方法

Object.keys(o)	键数组
Object.values(o)	值数组
Object.entries(o)	[key, value] 对数组
Object.assign(t, s)	将 s 的属性复制到 t
"k" in obj	键是否存在 → bool
delete obj.k	移除属性
Object.freeze(o)	设为不可变（浅层）

控制流

if / else if / else

```
if (score >= 90) {
  grade = "A";
} else if (score >= 80) {
  grade = "B";
} else {
  grade = "C";
}
```

三元与空值合并

```
const status = score >= 60 ? "pass" : "fail";
const name = user.name ?? "Anonymous";
```

switch

```
switch (color) {
  case "red":   stop();   break;
  case "green": go();     break;
  default:      wait();
}
```

循环

for / for...of / for...in

```
for (let i = 0; i < 5; i++) { }

for (const item of ["a", "b"]) { }  // arrays

for (const key in obj) { }  // object keys
```

while / do...while

```
while (count < 10) { count++; }

do { count++; } while (count < 10);
```

break & continue

```
for (let i = 0; i < 10; i++) {
  if (i === 5) break;       // stop loop
  if (i % 2 === 0) continue; // skip
}
```

函数

函数声明与箭头函数

```
function greet(name) {
  return `Hello, ${name}!`;
}
const greet = (name) => `Hello, ${name}!`;
const square = x => x * x; // single param
```

默认参数与 Rest

```
function greet(name = "World") { }

function sum(...nums) {
  return nums.reduce((a, b) => a + b, 0);
}
```

回调

```
[1, 2, 3].map(x => x * 2);    // [2, 4, 6]
[1, 2, 3].filter(x => x > 1); // [2, 3]
setTimeout(() => console.log("done"), 1000);
```

类

```
class Dog {
  constructor(name, breed) {
    this.name = name;
    this.breed = breed;
  }
  bark() { return `${this.name} says Woof!`; }
}

class Puppy extends Dog {
  constructor(name, breed, toy) {
    super(name, breed);
    this.toy = toy;
  }
}
```

错误处理

```
try {
  JSON.parse("bad json");
} catch (err) {
  console.error(err.message);
} finally {
  console.log("always runs");
}
```

JavaScript 快速参考

抛出错误

```
throw new Error("Something went wrong");
```

DOM

选取元素

```
document.querySelector(".cls")    // first match
document.querySelectorAll("li")   // all matches
document.getElementById("main")
```

修改元素

```
el.textContent = "new text";
el.innerHTML = "<b>bold</b>";
el.style.color = "red";
el.classList.add("active");
el.classList.toggle("hidden");
el.setAttribute("data-id", "42");
```

事件

```
btn.addEventListener("click", (e) => {
  console.log(e.target);
});
```

创建元素

```
const li = document.createElement("li");
li.textContent = "New item";
ul.appendChild(li);
el.remove(); // remove element
```

Fetch API

GET 请求

```
fetch("https://api.example.com/data")
  .then(res => res.json())
  .then(data => console.log(data))
  .catch(err => console.error(err));
```

POST 请求

```
fetch(url, {
  method: "POST",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify({ key: "value" }),
});
```

Async / Await

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

并行请求

```
const [users, posts] = await Promise.all([
  fetch("/users").then(r => r.json()),
  fetch("/posts").then(r => r.json()),
]);
```

模块

命名导出

```
// math.js
export const PI = 3.14;
export function add(a, b) { return a + b; }

// main.js
import { PI, add } from "./math.js";
```

默认导出

```
// logger.js
export default function log(msg) { }

// main.js
import log from "./logger.js";
```