

# 02 JavaScript

## • 1 概念

- JavaScript（简称：JS） 是一门跨平台、面向对象的脚本语言，是用来控制网页行为，实现页面的交互效果。JavaScript 和 Java 是完全不同的语言，不论是概念还是设计。但是基础语法类似。
- 组成：
  - ECMAScript：规定了JS基础语法核心知识，包括变量、数据类型、流程控制、函数、对象等。
  - BOM：浏览器对象模型，用于操作浏览器本身，如：页面弹窗、地址栏操作、关闭窗口等。
  - DOM：文档对象模型，用于操作HTML文档，如：改变标签内的内容、改变标签内字体样式等。

## • 2 核心语法

### • 2.1 引入方式

- 内部脚本：将JS代码定义在HTML页面中
  - JavaScript代码必须位于 `<script></script>` 标签之间
  - 在HTML文档中，可以在任意地方，放置任意数量的 `<script>`
  - 一般会把脚本置于 `<body>` 元素的底部，可改善显示速度
- 外部脚本：将 JS代码定义在外部 JS文件中，然后引入到HTML页面中

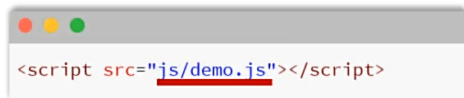


```
<body>
  <!-- 页面内容 -->

  <script>
    alert('Hello World');
  </script>
</body>
```



```
demo.js
alert('Hello World');
```



```
<script src="js/demo.js"></script>
```

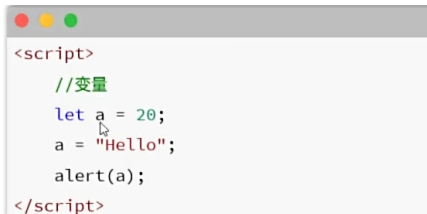
## JS书写规范

- 结束符：每行结尾以分号结尾，结尾分号可有可无

### • 2.2 基础语法

#### • 2.2.1 变量

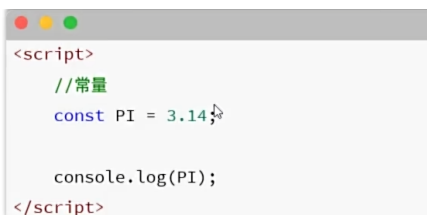
- JS中用 `let` 关键字来声明变量（**弱类型语言**，变量可以存放不同类型的值）。
- 变量名需要遵循如下规则：
  - 只能用 字母、数字、下划线（`_`）、美元符号（`$`）组成，且数字不能开头
  - 变量名严格区分大小写，如 `name` 和 `Name` 是不同的变量
  - 不能使用关键字，如：`let`、`var`、`if`、`for`等



```
<script>
  //变量
  let a = 20;
  a = "Hello";
  alert(a);
</script>
```

#### • 2.2.2 常量

- JS中用 `const` 关键字来声明常量。
- 一旦声明，常量的值就不能改变（不可以重新赋值）。



```
<script>
  //常量
  const PI = 3.14;

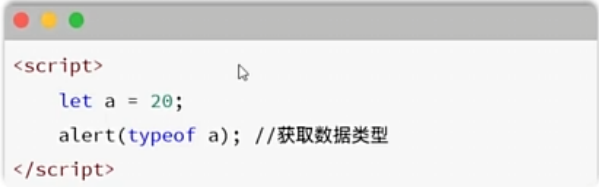
  console.log(PI);
</script>
```

- 2.2.3 输出语句

- `window.alert()` : 弹出警告框 (使用频次较高)
- `console.log()` : 写入浏览器控制台 (使用频次较高)
- `document.write()` : 向HTML的body内输出内容

- 2.2.4 数据类型

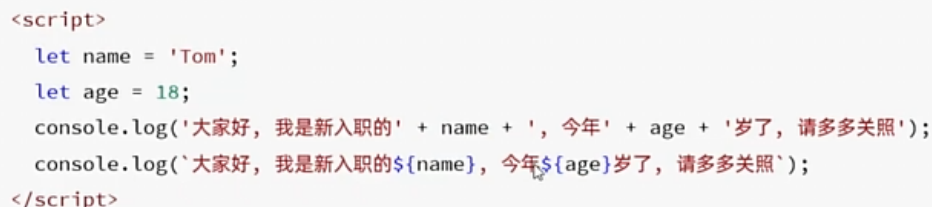
- JavaScript的数据类型分为: 基本数据类型和引用数据类型(对象)。
- 基本数据类型:
  - **number**: 数字(整数、小数、NaN(Not a Number))
  - **boolean**: 布尔。true, false
  - **null**: 对象为空。JavaScript是大小写敏感的, 因此null、Null、NULL是完全不同的
  - **undefined**: 当声明的变量未初始化时, 该变量的默认值是 undefined
  - **string**: 字符串, 单引号、双引号、反引号皆可, 推荐使用单引号
- 使用 **typeof** 运算符可以获取数据类型:



```
<script>
  let a = 20;
  alert(typeof a); //获取数据类型
</script>
```

- 2.2.5 模板字符串

- 模板字符串语法:
  - `` `` (反引号, 英文输入模式下按键盘的tab键上方波浪线 ~ 那个键)
  - 内容拼接变量时, 使用 `${}` 包住变量



```
<script>
  let name = 'Tom';
  let age = 18;
  console.log('大家好, 我是新入职的' + name + ', 今年' + age + '岁了, 请多多关照');
  console.log(`大家好, 我是新入职的${name}, 今年${age}岁了, 请多多关照`);
</script>
```

- 2.2.6 函数

- 介绍：函数（ `function` ）是被设计用来执行特定任务的代码块，方便程序的封装复用。
- 定义：JavaScript中的函数通过`function`关键字进行定义，语法为：

```
function functionName(参数1, 参数2 ...){
    //要执行的代码
}
```

```
function add(a, b){
    return a + b;
}
```

- 调用：函数名称（实际参数列表）

```
let result = add(10,20);
alert(result);
```

**注意：** 由于JS是弱类型语言，形参、返回值都不需要指定类型。在调用函数时，实参个数与形参个数可以不一致，但是建议一致。

## ● 2.2.7 匿名函数

- 匿名函数是指一种没有名称的函数，可以通过两种方式定义：函数表达式 和 箭头函数。

```
let add = function(a, b){
    return a + b;
}
```

函数表达式

```
let add = (a, b) => {
    return a + b;
}
```

箭头函数

- 匿名函数定义后，可以通过变量名直接调用。

```
let result = add(10,20);
alert(result);
```

## ● 2.2.8 自定义对象

- 定义格式

```
let 对象名 = {
    属性名1: 属性值1,
    属性名2: 属性值2,
    属性名3: 属性值3,
    方法名: function (形参列表) {}
}
```

```
let user = {
    name: 'Tom',
    age: 20,
    gender: '男',
    sing: function () {
        alert(this.name+'唱着最炫的民族风')
    }
}
```

```
let user = {
    name: 'Tom',
    age: 20,
    gender: '男',
    sing() {
        alert(this.name+'唱着最炫的民族风')
    }
}
```

- 调用格式

```
对象名.属性名;
对象名.方法名();
```

```
console.log(user.name);
user.sing();
```

```

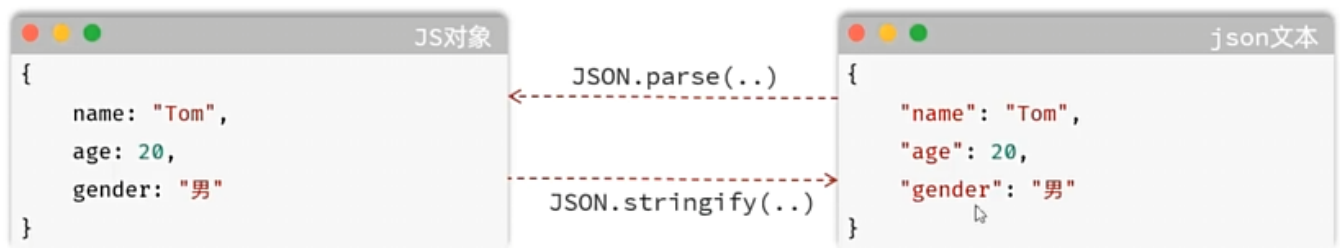
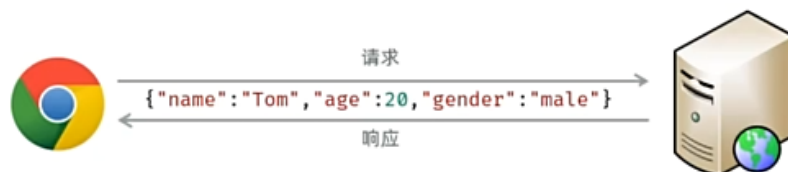
let user = {
  name: 'Tom',
  age: 18,
  gender: '男',
  sing: () => { //注意：在箭头函数中，this并不指向当前对象 - 指向的是当前对象的父级
    alert(this + ':悠悠的唱着最炫的民族风~')
  }
}

```

### • 3 json

## json

- 概念: JavaScript Object Notation, JavaScript对象标记法 (JS对象标记法书写的**文本**)
- 由于其语法简单, 层次结构鲜明, 现多用于作为**数据载体**, 在网络中进行数据传输。



- 即key均用双引号括起来, value如果是字符串形式也用双引号括起来

- JSON.parse : 将json字符串转为js对象
- JSON.stringify: 将js对象转为json字符串

### • 代码实现

```

//3. JSON - JS对象标记法
let person = {
  name: 'itcast',
  age: 18,
  gender: '男'
}

alert(JSON.stringify(person)); //js对象 --> json字符串

let personJson = '{"name": "heima", "age": 18}';
alert(JSON.parse(personJson).name);

```

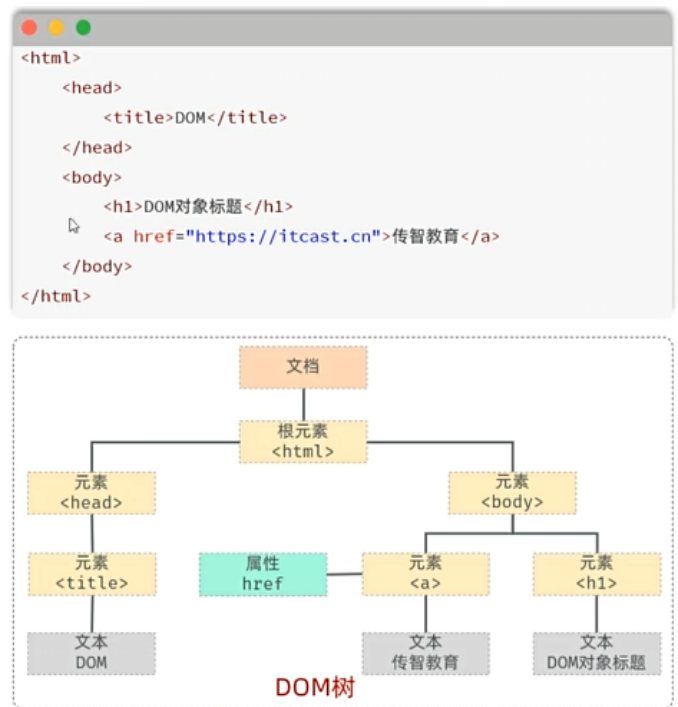
## • 4 DOM

- 概念: Document Object Model, 文档对象模型。
- 将标记语言的各个组成部分封装为对应的对象:

- ◆ Document: 整个文档对象
- ◆ Element: 元素对象
- ◆ Attribute: 属性对象
- ◆ Text: 文本对象
- ◆ Comment: 注释对象

- JavaScript 通过DOM, 就能够对HTML进行操作:

- ◆ 改变 HTML 元素的内容
- ◆ 改变 HTML 元素的样式 (CSS)
- ◆ 对 HTML DOM 事件作出反应
- ◆ 添加和删除 HTML 元素



### • 4.1 DOM 操作

- DOM操作核心思想: 将网页中所有的元素当做对象来处理。(标签的所有属性在该对象上都可以找到)

- 操作步骤:

- 获取要操作的DOM元素对象
- 操作DOM对象的属性或方法 (查文档或AI)

- 获取DOM对象

- 根据CSS选择器来获取DOM元素, 获取匹配到的第一个元素: `document.querySelector('选择器')` `#sid .txt span`
- 根据CSS选择器来获取DOM元素, 获取匹配到的所有元素: `document.querySelectorAll('选择器')`

注意: 得到的是一个NodeList节点集合, 是一个伪数组 (有长度、有索引的数组)

```
<span id="sid">DOM元素1</span>
<span class="txt">DOM元素2</span>
<span class="txt">DOM元素3</span>
```

### • 4.2 代码实现

```
<script>
    //1. 修改第一个h1标签中的文本内容
    //1.1 获取DOM对象
    // let h1 = document.querySelector('#title1');
    //let h1 = document.querySelector('h1'); // 获取第一个h1标签

    let hs = document.querySelectorAll('h1');

    //1.2 调用DOM对象中属性或方法
    hs[0].innerHTML = '修改后的文本内容';
</script>
```

## • 5 事件监听



- 事件：HTML事件是发生在HTML元素上的 "事情"。比如：

- 按钮被点击
- 鼠标移动到元素上
- 按下键盘按键

- 事件监听：JavaScript可以在事件触发时，就立即调用一个函数做出响应，也称为 **事件绑定或注册事件**。

- 语法：事件源.addEventListener('事件类型', 事件触发执行的函数);

- 事件监听三要素

- 事件源：哪个dom元素触发了事件，要获取dom元素
- 事件类型：用什么方式触发，比如：鼠标单击 click
- 事件触发执行的函数：要做什么事

```
<input id="btn" type="button" value="点我一下试试2">
<script>
  document.querySelector('#btn').addEventListener('click',()=>{
    alert('试试就试试');
  })
</script>
```

- 早期版本写法（了解）：事件源.on事件 = function() { ... }

```
<input id="btn" type="button" value="点我一下试试2">
<script>
  document.querySelector('#btn').onclick = function () {
    alert('试试就试试');
  }
</script>
```

- 区别：on方式会被覆盖，addEventListener方式可以绑定多次，拥有更多特性，推荐使用

## ● 5.1 常见事件

### 鼠标事件



click: 鼠标点击

mouseenter: 鼠标移入

mouseleave: 鼠标移出

### 键盘事件



keydown: 键盘按下触发

keyup: 键盘抬起触发

### 焦点事件



focus: 获得焦点触发

blur: 失去焦点触发

### 表单事件



input: 用户输入时触发

submit: 表单提交时触发

## ● 5.2 JS 程序优化

```
xxxx.html
<script src="./js/eventDemo.js" type="module"></script>
```

```
js/eventDemo.js
import { printLog } from './utils.js'

//click: 鼠标点击事件
document.querySelector('#b2').addEventListener('click', () => {
  printLog("我被点击了...");
})

//mouseenter: 鼠标移入
document.querySelector('#last').addEventListener('mouseenter', () => {
  printLog("鼠标移入了...");
})

//mouseleave: 鼠标移出
document.querySelector('#last').addEventListener('mouseleave', () => {
  printLog("鼠标移出了...");
})
```

```
js/utils.js
export function printLog(msg){
  console.log(msg);
}
```

JS模块化 (export, import)