

## 一、JavaScript 概述

1. 什么是JavaScript
  - 1) JS 介绍
  - 2) JS 组成
2. 使用方式

## 二、基础语法

1. 语法规则
2. JS的变量与常量
  - 1) 变量
  - 2) 常量
3. 数据类型
  - 1) 基本数据类型 ( 简单数据类型 )
  - 2) 引用数据类型
  - 3) 检测数据类型
4. 数据类型转换
  - 1) 强制类型转换
  - 2) 隐式类型转换 ( 自动转换 )
5. 运算符
  - 1) 赋值运算符
  - 2) 算数运算符
  - 3) 符合运算符
  - 4) 自增或自减运算符
  - 5) 关系运算符/比较运算符
  - 6) 逻辑运算符
  - 7) 三目运算符

# 一、JavaScript 概述

---

## 1. 什么是JavaScript

---

### 1) JS 介绍

简称JS，是一种浏览器解释型语言,嵌套在HTML文件中交给浏览器解释执行。主要用来实现网页的动态效果，用户交互及前后端的数据传输等。

### 2) JS 组成

1. 核心语法 -ECMAScript 规范了JS的基本语法
2. 浏览器对象模型 -BOM Browser Object Model，提供了一系列操作浏览器的方法
3. 文档对象模型 -DOM Document Object Model，提供了一系列操作的文档的方法

## 2. 使用方式

---

1. 元素绑定事件
  - 事件：指用户的行为（单击，双击等）或元素的状态（输入框的焦点状态等）
  - 事件处理：元素监听某种事件并在事件发生后自动执行事件处理函数。

- 常用事件：onclick (单击事件)
- 语法：将事件名称以标签属性的方式绑定到元素上，自定义事件处理。

```
<!--实现点击按钮在控制台输出-->  
<button onclick="console.log('Hello world');">点击</button>
```

## 2. 文档内嵌。使用undefined标签书写 JS 代码

- 语法：

```
<script type="text/javascript">  
  alert("网页警告框");  
</script>
```

- 注意：undefined标签可以书写在文档的任意位置，书写多次，一旦加载到script标签就会立即执行内部的JS代码，因此不同的位置会影响代码最终的执行效果

## 3. 外部链接

- 创建外部的JS文件 XX.js，在HTML文档中使用undefined引入

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

- 注意：undefined既可以实现内嵌 JS 代码，也可以实现引入外部的 JS 文件，但是只能二选一。

## 4. JS 输入语句

- alert(""); 普通的网页弹框
- prompt(""); 接收用户输入的弹框，返回用户输入的内容
- console.log(); 控制台输出，多用于代码调试
- document.write("

# Hello

");实现在动态在网页中写入内容。

1. 在使用事件方式写入时，会重写网页内容
2. 可以识别HTML标签,脚本代码可以在文档任何地方书写，如果是普通写入（不涉及事件），区分代码的书写位置，在当前位置中插入，如果是在head中嵌入，内容会作为body的首行内容显示

## 二、基础语法

### 1. 语法规范

1. JS是由语句组成,语句由关键字,变量,常量,运算符,方法组成.分号可以作为语句结束的标志,也可以省略
2. JS严格区分大小写
3. 注释语法 单行注释使用 // 多行注释使用 /\* \*/

### 2. JS的变量与常量

## 1) 变量

1. 作用：用于存储程序运行过程中可动态修改的数据
2. 语法：使用关键var声明,自定义变量名

```
var a;           //变量声明
a = 100;         //变量赋值
var b = 200;     //声明并赋值
var m,n,k;       //同时声明多个变量
var j = 10,c = 20; //同时声明并赋值多个变量
```

3. 命名规范：
  - 变量名,常量名,函数名,方法名自定义,可以由数字,字母,下划线,\$组成,禁止以数字开头
  - 禁止与关键字冲突(var const function if else for while do break case switch return class)
  - 变量名严格区分大小写
  - 变量名尽量见名知意,多个单词组成采用小驼峰,例如："userName"
4. 使用注意：
  - 变量如果省略var关键字,并且未赋值,直接访问会报错
  - 变量使用var关键字声明但未赋值,变量初始值为undefined
  - 变量省略var关键字声明,已被赋值,可正常使用.影响变量作用域

## 2) 常量

1. 作用：存储一经定义就无法修改的数据
2. 语法：必须声明的同时赋值

```
const PI = 3.14;
```

3. 注意：
  - 常量一经定义,不能修改,强制修改会报错
  - 命名规范同变量,为了区分变量,常量名采用全大写字母
4. 操作小数位 toFixed(n); 保留小数点后 n 位 使用：

```
var num = 3.1415926;
//保留当前变量小数点后两位
var res = num.toFixed(2);
```

## 3. 数据类型

### 1) 基本数据类型（简单数据类型）

1. number 数值类型
  - 整数
    1. 十进制表示

```
var a = 100;
```

## 2. 八进制表示 以0为前缀

```
var b = 021; //结果为十进制的 17
```

## 3. 十六进制 以0x为前缀

```
var c = 0x35; //结果为十进制的 53
```

使用：整数可以采用不同进制表示,在控制台输出时一律会按照十进制输出

### ○ 小数

#### 1. 小数点表示

```
var m = 1.2345;
```

2. 科学计数法 例: 1.5e3 e表示10为底,e后面的数值表示10的次方数 1.5e3 等价于  $1.5 * 10(3)$

2. string 字符串类型 字符串: 由一个或多个字符组成,使用""或"表示,每一位字符都有对应的Unicode编码

```
var s = "100";  
var s1 = "张三";
```

3. boolean 布尔类型 只有真和假两个值,布尔值与number值可以互相转换。true 为 1, false 为 0

```
var isSave = true;  
var isChecked = false;
```

4. undefined 特殊值,变量声明未赋值时显示undefined

```
var a;  
console.log(a); //undefined
```

5. null 空类型 解除对象引用时使用null,表示对象为空

## 2) 引用数据类型

主要指对象,函数等

## 3) 检测数据类型

typeof 变量或表达式 typeof (变量或表达式)

```
var n = "asda";  
console.log(typeof n); //string  
console.log(typeof(n)); //string
```

## 4. 数据类型转换

---

不同类型的数据参与运算时,需要转换类型

## 1) 强制类型转换

1. 转换字符串类型 方法: toString() 返回转换后的字符串

```
var a = 100;  
a = a.toString(); //"100"  
var b = true;  
b = b.toString(); //"true"
```

2. 转换number类型

- Number(param) 参数为要进行数据类型转换的变量或值, 返回转换后的结果: 如果转换成功,返回number值 如果转换失败,返回NaN,(Not a Number), 只要数据中存在非number字符,一律转换失败, 返回 NaN
- parseInt(param) 参数为要解析的数据 作用: 从数据中解析整数 过程: 1. 如果参数为非字符串类型,会自动转成字符串 2. 从左向右依次对每一位字符转number,转换失败则停止向后解析,返回结果
- parseFloat(param) 作用: 提取number值, 包含整数和小数部分

## 2) 隐式类型转换 (自动转换)

1. 当字符串与其他数据类型进行"+"运算时,表示字符串的拼接,不再是数学运算 转换规则: 将非字符串类型的数据转换成字符串之后进行拼接, 最终结果为字符串
2. 其他情况下,一律将操作数转number进行数学运算

# 5. 运算符

## 1) 赋值运算符

= 将右边的值赋给左边变量

## 2) 算数运算符

+ - \* / % 加 减 乘 除 取余

## 3) 符合运算符

+= -= \*= /= %=

## 4) 自增或自减运算符

++ -- 变量的自增和自减指的是在自身基础上进行 +1或-1 的操作

注意:

- 自增或自减运算符在单独与变量结合时, 放前和放后没有区别
- 如果自增或自减运算符与其他运算符结合使用, 要区分前缀和后缀, 做前缀, 那就先++/--, 再进行赋值或其他运算, 如果做后缀, 就先结合其他运算符, 再进行++ / --

## 5) 关系运算符/比较运算符

```
> <
>= <=
==(相等) !=(相等)
===(全等) !==(不全等)
```

1. 关系运算符用来判断表达式之间的关系,结果永远是布尔值 true/false

2. 使用

- 字符串与字符串之间的比较 依次比较每位字符的Unicode码,只要某位字符比较出结果,就返回最终结果
- 其他情况 一律将操作数转换为number进行数值比较,如果某一操作数无法转换number,则变成NaN参与比较运算,结果永远是false

3. 相等与全等

- 相等: 不考虑数据类型,只做值的比较(包含自动类型转换)
- 全等: 不会进行数据类型转换,要求数据类型一致并且值相等才判断全等

## 6) 逻辑运算符

1. && 逻辑与 表达式同时成立,最终结果才为true;全1则1

2. || 逻辑或 表达式中只要有一个成立,最终结果即为true; 有1则1

3. ! 逻辑非 对已有表达式的结果取反 注意: 除零值以外,所有值都为真

## 7) 三目运算符

语法:

```
表达式1 ? 表达式2 : 表达式3;
```

过程: 判断表达式1是否成立,返回布尔值 如果表达式1成立,执行表达式2; 如果表达式1不成立,执行表达式3;