Web开发（二）

--- 第二章 JavaScript 基础语法

河北师范大学软件学院
Software College of Hebei Normal University

JavaScript 语法概述

- 语法特点:
  - 弱类型：变量的数据类型可以任意转换
  - 动态类型：变量声明创建时不用指定数据类型

内容提纲

- **JavaScript 基础语法**
- **JavaScript 变量及内置数据类型**
- **JavaScript 流程控制结构**

JavaScript 基础语法

- JavaScript 语句
  - 语句就是命令，它告诉浏览器要做什么
  - 语句以分号结束

```
<script type="text/javascript">
    console.log("Hello World")
</script>
```

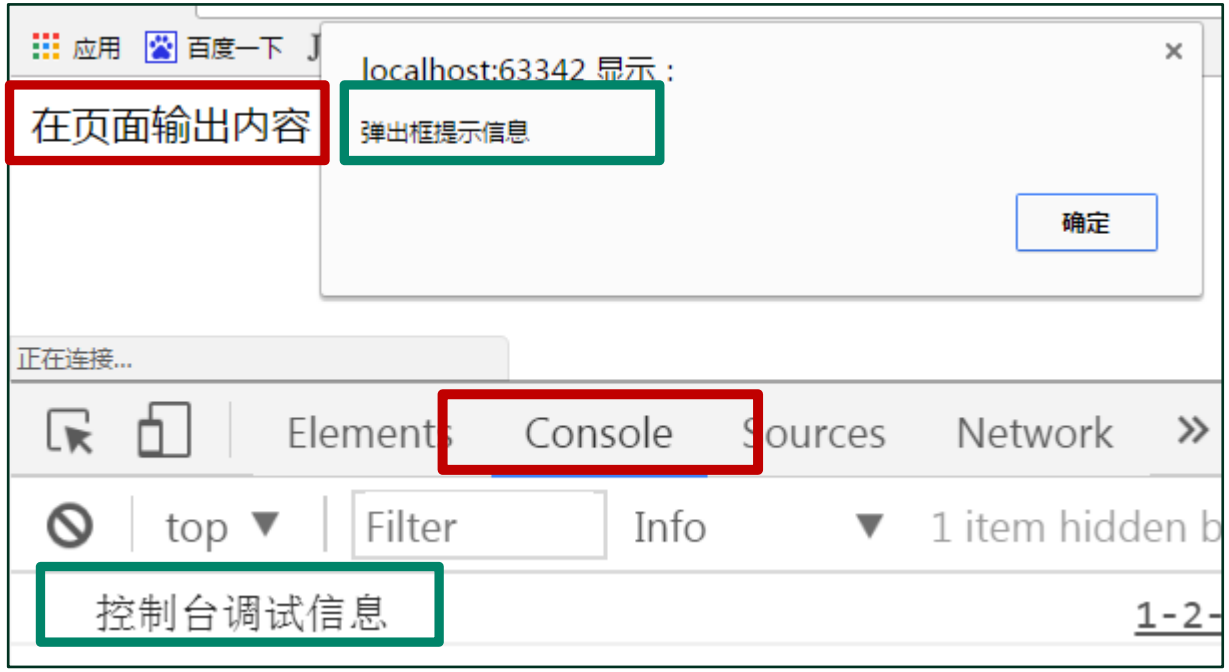
- JavaScript 语句块
  - 多个语句可放在 “{” 和 “}” 内，形成一个语句块

```
if(true){
    document.write("第一条语句执行");
    document.write("<br/>");
    document.write("第二条语句执行");
}
```


demo 2-1.html

JavaScript 基础语法

- JavaScript 输出内容的 3 种方式:
  - document.write( )　页面输出内容
  - console.log( )　控制台输出
  - alert( )　弹出框输出



demo 2-2.html

JavaScript 基础语法

- JavaScript 注释:
  - 单行注释：//
  - 多行注释：/* */
  - 提高代码的可读性
  - JavaScript 不会执行注释

内容提纲

- JavaScript 基础语法
- JavaScript 变量及内置数据类型
- JavaScript 流程控制结构

字面量

- 1、2、1.0、3.1415926...
- 'hello'、"world"、 "34" ...
- true、false

认识变量

- 什么是变量?
  - 变量是存储信息的容器

    例如： x = 1;

    y = 3;

    sum = x + y;
  - 在 JavaScript 中，这些字母被称为变量

变量的声明和赋值

- 变量声明
  - 使用关键字 var 创建变量
  - JavaScript 为动态类型语言，声明变量时，不需指明数据类型
- 变量赋值
  - 使用 “=” 为变量赋值
  - 值为字符串时需用 ‘ ’ 或 “ ” 引起来

变量命名

- 变量命名
  - 变量名区分大小写
  - 变量名以字母或 '_' 或 '$' 开头
  - 变量名不能是关键字，保留字

```
var  sum;   ✓
var  SUM;   ✓
var  2add;  ✗
var  if=3;  ✗
```

JavaScript关键字

break	do	instanceof	typeof
case	else	new	var
catch	finally	return	void
continue	for	switch	while
debugger*	function	this	with
default	if	throw	delete
in	try		

JavaScript 原始数据类型

- JavaScript 原始数据类型
  - 5 种原始数据类型：Number、String、Boolean、Undefined、Null
  - 判断变量在某一时刻的数据类型，使用 typeof 运算符
- JavaScript 是弱类型语言
  - 弱类型是指不同类型的变量之间可以相互赋值，但在某一时刻，一个变量只存在某一种数据类型

原始数据类型

- Number 类型:
  - 1、3.1415926、1e6、NaN、....
- String 类型:
  - 用 ‘’ 或 “” 引起一组字符
  - 如: 'hello'、"world"、 "34"
- Boolean 类型:
  - true 或 false

原始数据类型

- Undefined 类型:
  - 只有一个值 undefined
  - 指声明了但未赋值的变量，如 var a;
- Null 类型:
  - 只有一个值 null，表示值为空
  - null 不等同于空的字符串（“”）或 0

原始数据类型

```
<script>
    var a = 200;//整型 Number
    var b = 92.5;//浮点型 Number
    var c = "I'm String";//字符串型 String
    var d = true;//布尔类型
    var e;   //undefine
    var f = null;//nu
    alert(typeof  c);
</script>
```

demo 2-3.html

原始数据类型

- 认识 NaN　(Not a Number)
  - 表示一个没有意义、不正确的数值
  - console.log(typeof NaN);　——Number
  - 与自身不相等　—— NaN !=NaN
- 认识 isNaN( ) 函数
  - 用来检测参数是否为 NaN 值
  - 参数是 "NaN" 时返回 true，否则返回 false
  - isNaN("123abc") —— true

demo 2-4.html

运算符

- 算术: +、-、*、/、%、++、--
- 赋值: =、+=、-=、*=、/=、%=
- 字符串拼接: +
- 比较: ==、===、!=、>、<、<=、>=
- 逻辑: 与(&&)、或(||)、非(!)
- 条件: 变量名 = (条件)？表达式1 : 表达式2

使用“+”拼接字符串

```
<script>
    var x = 3;
    var y = "3";
    var z = 5;
    z += y;
    var a = y + z;
    document.write(x+y+'<br/>');
    document.write(z+'<br/>');
    document.write(a+'<br/>');
    document.write(x+y+z+'<br/>');
</script>
```

33

53

353

3353

demo 2-5.html

比较运算符

- 比较运算符
  - ==：值相等则为 true
  - ===：类型和值都须相同则为 true

```
var x = 3;  // Number
var y = 3; // Number
var z = "3"; // String
```

```
alert(x == y);   // true
alert(x === y);  // true
alert(x === z);   // false
```


demo 2-6.html

条件(三目)运算符

```
<script>
    var a = 39;
    var b = 30;
    document.write( a>=b?  "a大于等于b" : "a小于b");
    var age = 12；
    var msg = age>18 ? "成年人" : "未成年";
    console.log(msg)；
</script>
```

true

?

false

?

demo 2-7.html

运算符优先级

运算符	说明
( )	表达式分组
++ -- !	自加、自减、非
* / %	相乘、相除、求余数
+ - +	相加、相减、字符串串联
< <= > >=	小于、小于或等于、大于、大于或等于
== != === !==	相等、不相等、全等，不全等
&&	逻辑“与”
||	逻辑“或”
? :	条件运算
=	赋值运算

通过 () 改变优先级

运算符

- 23 + " 2 " = ?
- 15/2 = ?
- 23 – true = ?
- " 95 " == 95
- " 95 " === 95
- typeof 75

—— 232

—— 7.5

—— 22

—— true

—— false

—— number

数据类型转换

- 隐式转换
  - 转换成 String 类型：用 + 连接

    如：var sum = "img" + 3 + ".jpg";

    img3.jpg
  - 转换成 Boolean 类型：变量前面加 !!
- 显示(强制)转换
  - 全局函数

    如：parseInt()、String()

转换为布尔类型规则 1

- 数值转换为布尔类型:
  - 0, 0.0, -0 → **false**
  - NaN → **false**
  - 其他数值，比如1, 2, 3, -5 → **true**
- undefined 转换为布尔类型:
  - undefined → **false**

demo 2-8.html

转换为布尔类型规则 2

- null 转换为布尔类型:
  - null → **false**
- 字符串转换为布尔类型:
  - 空字符串 “” → **false**
  - 非空字符串 ‘hello world ’ → **true**
- 总结：非0数字和非空字符串转为true，其余均为false
- 强制转换为 Boolean 类型: Boolean( )

转换为数值类型规则 1

- 布尔转换为数值类型:
  - false → 0
  - true → 1
- undefined 转换为数值类型:
  - undefined → NaN
- null转换为数值类型:
  - null → 0

demo 2-9.html

转换为数值类型规则 2

- 字符串转换为数值类型:
  - 字符串内容为纯数字 → 数字本身

    如:“123” → 123
  - 字符串为非纯数字 → NaN

    如:“1a2b3c” → NaN
- 强制转换为 Number 类型:
  - parseInt()、parseFloat()、Number()

转换为字符串类型规则 1

- 数值转换为字符串类型:
  - 转换为数值本身

    如: 12345 → "12345"

    NaN → "NaN"
- 布尔值转换为字符串类型:
  - true → "true"
  - false → "false"

demo 2-10.html

转换为字符串类型规则 2

- undefined转换为字符串类型:
  - undefined → "undefined"
- null 转换为字符串类型:
  - null → "null"
- 强制转换为 String 类型:
  - String( )

数据类型转换

```
<script type="text/javascript">
// 转换成 Number 类型
    var a = '123.456img';
    var a1 = parseInt(a);
    var a2 = parseFloat(a);
    document.write("a1=" + a1 +'<br/>'+ "a2=" + a2+'<br/>');
    document.write(typeof a2+'<br/>');

// 转换成 String 类型
    var b = 3.1415926;
    var b1 = b + "";
    document.write(typeof(b1)+'<br/>');

// 转换成 Boolean 类型
    var c = 'img' + 3 + '.jpg';
    var c1 = !!c;
    document.write('c1='+c1)
    alert(typeof c1);
</script>
```

a1=123

a2=123.456

number

string

c1 = true

boolean

demo 2-11.html

运算符左右数据类型转换规则

- + 左右出现字符串时，作为字符串连接运算符使用
- -、*、/、% 左右出现字符串（布尔）时，将字符串（布尔）转换为数值类型
- 比较运算符左右出现字符串（布尔），会转换为数值类型
- 逻辑运算符会将数据类型转换为布尔类型之后再做运算

demo 2-12.html

内容提纲

- JavaScript 基础语法
- JavaScript 变量及内置数据类型
- JavaScript 流程控制结构

程序的流程控制

- 程序 = 数据 + 算法

- 任何复杂的程序算法都可以通过“顺序”，“分支”，“循环”三种基本的程序逻辑组合实现

程序的流程控制

- 分支语句
  - if…else if…else 语句
  - switch - case 语句
- 循环语句
  - for 语句
  - while 语句

分支语句

转换为Boolean值

```
<script>
    var box = 100;
    if (box >= 100) {
        alert("甲");
    } else if (box >= 90) {
        alert("乙");
    } else if (box >= 80) {
        alert("丙");
    } else if (box >= 70) {
        alert("丁");
    } else if (box >= 60) {
        alert("及格");
    } else {
        alert("不及格");
    }
</script>
```


demo 2-13.html

分支语句

```
<script>
    var score = "优";
    switch (score) {
        case "优":
            document.write("该同学成绩优异");
            break;
        case "良":
            document.write("该同学成绩良好");
            break;
        case "及格":
            document.write("该同学成绩及格");
            break;
        case "不及格":
            document.write("该同学成绩不及格");
            break;
        default:
            document.write("该同学成绩未知");
    }
</script>
```


demo 2-14.html

循环语句

```
for (var i = 1; i < 4; i++) {
    console.log(i);
}
```

```
var i = 1; //i=3;
do {
    document.write(i + "<br/>");
    i++;
} while (i < 3)
```

```
var i = 1; //i=3;
while (i < 3) {
    document.write(i + "<br/>");
    i++;
}
```


demo 2-15.html

循环语句

- 如何终止循环?
  - 终止循环: break ;
  - 跳过本次循环: continue ;

demo 2-16.html

循环语句

动手做：demo 2-17.html

- 使用 for 循环，向文档中动态写入一个 4 行 4列的表格，表格单元格内容为
- 使用 while 循环，向文档中动态写入一个4 行 4列的表格，表格单元格内容为

小结

- JavaScript 中的语句和语句块
- 变量和原始数据类型
- 运算符及其优先级
- 数据类型转换

代码规范的重要性

- 方便代码的交流和维护。
- 不影响编码的效率，不与大众习惯冲突。
- 使代码更美观、阅读更方便。
- 使代码的逻辑更清晰、更易于理解。

JavaScript 代码规范

Thank You!

河北师范大学软件学院
Software College of Hebei Normal University