JavaScript第一章

本章主要内容

1. JavaScript简介
2. HTML中使用JavaScript的方式
3. 变量

一、JavaScript简介

JS的发展历史

```
1995年，就职于Netscape(网景)公司的 布兰登·艾奇 开发了JavaScript的第一个版本
```

JS的适用业务场景

```
JavaScript作为客户端脚本语言，适用业务场景广泛
    1. 验证用户输入数据
    2. 用户与网页发生交互性行为，触发页面特效
    3. 异步提交数据给服务器(Ajax)，提高用户体验，减轻服务器压力
    4. 数据延时加载(懒加载、瀑布流)
```

JavaScript的组成部分

```
在浏览器中，JavaScript由三部分组成。
    1. ECMAScript JS的基本语法结构(ECMA 欧洲计算机制造商协会)
    2. 文档对象模型(DOM)
    3. 浏览器对象模型(BOM)
```

二、HTML中使用JavaScript的方式

1. 直接在页面的script标签中书写JS

   ```
   <script>
       // JS 代码
   </script>
   ```

2. 使用script标签引入外部的JS文件

```
<script src="JS文件地址">
    // 该位置不要书写JS代码(因为写了也没有用！)
</script>
```

3. JS中的注释格式

```
单行注释
    //
多行注释
    /**/
    不允许相互嵌套

文档注释
    /**
    */
    不允许相互嵌套，多用于说明函数或对象相关信息

注释作用:
    解释说明作用
    便于调试
    便于团队协作开发
```

注意:

```
script标签可以在任意位置书写，但推荐写在body标签的最后一个或者是body标签的外部
```

三、变量

概念

```
1. 变量是一个储存了值的容器
    存储了饭的桶  ---> 饭桶(变量名) 饭(变量值)
2. JS是一个弱类型语言，变量的类型会随着值的变化而发生类型变化
```

声明方式与变量命名规则

1. 使用var关键字声明变量
2. 变量名(标识符)命名规则
    以字母、下划线、$开头，后跟任意长度的数字、字母、下划线、$
    区分大小写
    不能使用系统的关键字、保留字
        关键字：系统已经使用的，表示某个特殊作用
        保留字：系统表示以后可能会用到，先占着

    命名尽量有意义
    推荐使用驼峰命名法
        如果变量名由多个单词组成，第一个单词的首字母小写，后面的单词首字母大写！

示例：

```
var name = '王大锤'
```

附JS中的系统关键字

ECMA-262 描述了一组具有特定用途的关键字。这些关键字可用于表示控制语句的开始或结束，或者用于执行特定操作等。按照规则，关键字也是语言保留的，不能用作标识符。以下就是ECMAScript的全部关键字（带*号上标的是第5 版新增的关键字）：

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		

附JS中的系统保留字

ECMA-262 还描述了另外一组不能用作标识符的保留字。尽管保留字在这门语言中还没有任何特定的用途。但它们有可能在将来被用作关键字。以下是ECMA-262 第3 版定义的全部保留字：

abstract	enum	int	short
boolean	export	interface	static
byte	extends	long	super
char	final	native	synchronized
class	float	package	throws
const	goto	private	transient
debugger	implements	protected	volatile
double	import	public	

指令分隔符

```
1. 指令分隔符，用于表示一个语句的结束
2. 在JS中换行符和分好都是指令分隔符
3. 推荐使用分号(;)作为指令分割符
```

变量类型

```
在JS中显式变量类型有六种(使用typeof进行检测)：
    Undefinded      未定义
    Boolean         布尔值
    String          字符串
    Number          数值
    Function        函数
    Object          对象
```

Undefined 未定义

```
1. 变量只声明，未赋值
2. 变量显式赋值为undefined
```

Boolean 布尔值

用于表示真假的值，多用于条件判断。一共有两种值：

```
true    真
false   假
```

Number 数值

包含了整数、浮点数(小数)、NaN在内的数据类型

各数据类型定义：

```
整数
    十进制
        使用0 1 2 3 4 5 6 7 8 9来表示
    二进制
        以0b开头，由数字0 1表示
    八进制
        以0开头，由数字0 1 2 3 4 5 6 7表示
    十六进制
        以0x开头，由字符0 1 2 3 4 5 6 7 8 9 a b c d e f表示

浮点数
    该数值中必须包含一个小数点，并且小数点后面必须有一位数字

    科学计数法
        对于极大或极小的值，使用科学计数法方便表示
        值相当于 e前面的数值乘以10的指数次幂
        示例：
            3.215e6 ===> 3215000

    注：永远不要测试某个特定的浮点数的值
        var a = 0.1;
        var b = 0.2;
        a+b == 0.3; (这个表达式是不成立的！)

NaN
    1. 不是一个数字的数值类型 Not a Number
    2. 本来应该返回一个数值的却未返回数值的情况下，返回NaN

    特点：
        NaN是一个不是数字的数值类型
        NaN与任何一个数字进行算数运算，都得到NaN
        NaN与任何值都不相等，包括自身

    isNaN() 判断是否是NaN
        如果是，返回布尔值 true
        如果不是，返回布尔值 false
```

数值取值范围：

由于内存限制，JS并不能保存世界上的所有值。

最大值
    Number.MAX_VALUE
最小值
    Number.MIN_VALUE

超出计算值的范围：
    正无穷 Infinity
    负无穷 -Infinity

isFinite() 检测数值是否是无穷大
    如果number是有限数值，则返回true
    如果number是NaN，或者是正负无穷，则返回false

String 字符串

定义字符串的方式：

1. 单引号 ''
2. 双引号 ""
3. 反单引号 `` （ES6语法）
    便于定义多行字符串

注：单双引号定义的字符串无任何区别，但注意符号的单双引号的嵌套

转义符号

转义符号是一些特殊的字符字面量，用于表示非打印字符，或者具有其他特殊用途的字符

\n	换行
\t	制表
\b	退格
\r	回车
\f	进纸
\\	反斜线
\'	转义'
\"	转义"
\xnn	以十六进制代码nn表示一个字符(其中n为0-F)
\unnnn	以十六进制代码nnnn表示的一个Unicode字符(其中n为0-F)

Function 函数

```
函数是一段具有指定功能的已命名的代码块
function hello(){
    console.log('Hello JS!');
}
```

Object 对象

```
对象是一个具有某些特性(属性)和功能(方法)的集合体

// 定义对象
var superMan = new Object();

// name 属性
superMan.name = '钢铁侠';

// intro 方法
superMan.intro = function(){
    console.log('My name is gangtiexia!');
}
```

变量类型转换

JS是一个弱类型语言，变量类型可以进行相互转换

```
Boolean()           转为布尔型
Number()            转为数值型
String()            转为字符串
Function()          转为函数
Object()            转为对象
```

其他类型的值转为布尔值时为假的情况：

```
1. 布尔值的 false
2. 数值类型的 0/NaN/0.0
3. 字符串类型的 空字符串
4. 对象类型中的 null
5. undefined
```

其他类型的值转为数值类型：

Number()

Number()可以用于任何数据类型
1. 布尔值
    true   1
    false   0
2. 数值类型
    简单的传入和传出
3. 对象中的null
    转为0
4. undefined
    转为NaN
5. 字符串
    只包含数字，转为对应十进制数字
    包含有效浮点格式，转为对应的浮点数
    包含有效十六进制格式，转为对应大小的十进制数字
    字符串为空，转为0
    如果字符串包含除以上格式外的其他字符，转为NaN

parseInt()

parseInt()专门用于将字符串转为数值中的整数。

特点:
    忽略字符串前面的空格，查找到第一个非空格字符
    如果第一个字符不是数字字符或者是正负号，转为NaN
    如果第一个字符是数字(正负号)，一直向下查找，找到结尾或第一个非数字字符，转为对应的数值
    能够识别进制

示例:
```
    var num1 = parsetInt('15a');         // 15
    var num2 = parseInt('0xa');      // 10
    var num3 = parseInt('21abc',16);     // 第二个参数描述当前字符是使用某个进制
```

parseFloat()

parseFloat()专门用于将字符串转为数值中的浮点数(小数)。

特点:
    规律大致和parseInt()相同，但是识别第一个小数点