# 第十天 - JS 入门

## 一 Javascript 简介

JavaScript 是属于网络的脚本语言！
JavaScript 被数百万计的网页用来改进设计、验证表单、检测浏览器、创建 cookies，以及更多的应用。
JavaScript 是因特网上最流行的脚本语言。
JavaScript 很容易使用。
JavaScript 是一种轻量级的编程语言。
Javascript 文件的扩展名是.js。
JavaScript 是可插入 HTML 页面的编程代码。
JavaScript 插入 HTML 页面后，可由浏览器执行。
JavaScript 存在于全世界所有 Web 浏览器中，能够增强用户与 Web 站点和 Web 应用程序之间的交互。
Javascript 用于 HTML 和 web，更可广泛用于服务器、PC、笔记本电脑、平板电脑和智能手机等设备。

## 二 Javascript 推荐书籍

《Javascript 高级程序设计》
如果需要电子版，可以找老师@张凯。

## 三 Javascript 可以做什么？

1 动态输出 HTML

```
document.write("<h1>这是一个标题</h1>");
// 说明：
// 请使用 document.write() 仅仅向文档输出写内容。
// 如果在文档已完成加载后执行 document.write，整个 HTML 页面将被覆盖。
```

2 对事件做出反应

```
<button onclick="alert(我被点击了!')">点击这里</button>
```

3 改变 HTML 内容

```
var x = document.getElementById("demo");  //查找元素
x.innerHTML = "Hello JavaScript";       //改变内容
// 说明：
// 如需从 JavaScript 访问某个 HTML 元素，您可以使用 document.getElementById(id) 方法。
// 请使用 "id" 属性来标识 HTML 元素。
```

4 改变 HTML 样式

```
var x = document.getElementById("demo");   //找到元素
x.style.color = "#ff0000";              //改变样式
```

5 验证用户输入

```
var age = document.getElementById("age");
if (age == ‘’) {
    alert(‘用户年龄不能为空’);
}
```

这仅仅是冰山一角…

## 四　开始使用 Javascript

### 1　HTML 页面使用 Javascript 的几种方式

对比 HTML 页面使用 CSS 的方式，页面中使用 Javascript 有内部 Javascript、外部 Javascript、内联（行内）Javascript 三种方式。

#### 1-1 内部 Javascript

```
// 方式一
<head>
  <script>
    alert(‘Hello’);
  </script>
</head>

// 方式二
<body>
  <script>
    alert(‘Hello’);
  </script>
</body>
```

#### 1-2 外部 Javascript（推荐使用）

```
// 方式一
<head>
  <script src=”path/to/file.js”></script>
  <script src=”path/to/file2.js”></script>
</head>

// 方式二
<body>
  <script src=”path/to/file.js”></script>
  <script src=”path/to/file2.js”></script>
</body>
```

#### 1-3 内联（行内）Javascript

```
// 方式一
<button onclick=”alert(‘我被点击了’)”>按钮</button>
// 方式二
```

```
<button onclick=”hello()”>按钮</button>
<head>
  <script>
  function hello() {
    alert(‘我被点击了’);
  }
  </script>
</head>
// 或者
<body>
  <script>
  function hello() {
    alert(‘我被点击了’);
  }
  </script>
</body>
```

推荐把外部 Javascript 文件的引入位置放在 body 结束标签的前面以得到最好的用户体验。外部脚本内不能包含 <script> 标签。

**学习心得：**

第十一天 - JS 入门

## 五 Javascript 语句

Javascript 语句

Javascript 语句向浏览器发出的命令。语句的作用是告诉浏览器该做什么。

Javascript 语句使用分号作为一条语句的结束符号。

通常我们在每条可执行的语句结尾添加分号。（分号可选，要求必须使用）

使用分号的另一用处是在一行中编写多条语句。

## 六 Javascript 代码

Javascript 代码（或 Javascript）是 Javascript 语句的序列。

浏览器会按照编写顺序来执行每条语句。

## 七 Javascript 代码块

JavaScript 语句通过代码块的形式进行组合。

块由左花括号开始，由右花括号结束。

块的作用是使语句序列一起执行。

JavaScript 函数是将语句组合在块中的典型例子。

```
// 一个简单的函数例子
function hello() {
  alert('Hello World');
}
```

## 八 其它需要注意点

Javascript 对大小写敏感（区分大小写）

Javascript 会忽略多余的空格。您可以向脚本添加空格，来提高其可读性。

您可以在文本字符串中使用反斜杠对代码行进行换行。

```
// 可以
document.write("Hello \
World!");
// 不可以 document.write \
("Hello World!");
```

Javascript 是脚本语言。浏览器会在读取代码时，逐行地执行脚本代码。而对于传统编程来说，会在执行前对所有代码进行编译。

## 九 Javascript 注释

Javascript 不会执行注释，可以用来阻止代码执行。

我们可以添加注释来对 Javascript 进行解释，或者提高代码的可读性。

单行注释以 // 开头；多行注释以 /* 开头，以 */ 结束。

```
// alert('hello');
/*
alert('a');
abler('b');
*/
```

```
alert('your age is 22'); // 提示用户

/*******************************************
* author: situ
* since: 2012-09-21
* 测试例子
*******************************************/
function test () {
}
```

## 十 Javascript 变量

变量是存储信息（数据）的容器。

```
// js 定义变量的语句
var x = 2;
var y = 3;
var z = x + y;
// 代数的表达方式，在 JavaScript 中，这些字母被称为变量。
设 x = 2
设 y = 3
设 z = x + y
求 z 的值?
```

与代数一样，Javascript 变量可用于存放值（比如 x=2）和表达式（比如 z=x+y）。
变量可以使用短名称（比如 x 和 y），也可以使用描述性更好的名称（比如 age, sum, total）。
Javascript 语句和 Javascript 变量都对大小写敏感。

## 十一 Javascript 命名规则

变量必须以字母开头
变量也能以 $ 和 _ 符号开头（不过我们不推荐这么做）
变量名称对大小写敏感（y 和 Y 是不同的变量）

或者这样说：js 变量是以字母、$、_ 开头的，字母、$、_ 和数字的组合。

## 十二 Javascript 数据类型简介

JavaScript 变量还能保存其他数据类型，比如文本值 (name="Bill Gates")。
在 JavaScript 中，类似 "Bill Gates" 这样一条文本被称为字符串。
JavaScript 变量有很多种类型，但是现在，我们只关注数字和字符串。
当您向变量分配文本值时，应该用双引号或单引号包围这个值。
当您向变量赋的值是数值时，不要使用引号。如果您用引号包围数值，该值会被作为文本来处理。

## 十三 声明 Javascript 变量

在 Javascript 中创建变量通常称为“声明”变量。
我们使用 var 关键词来声明变量

```
// 变量声明后，值是空的。
```

```
// 在计算机程序中，经常会声明无值的变量。未使用值来声明的变量，
// 其值实际上是 undefined。
var username;
username = ‘张三’; // 可以用等号为变量赋值
// 或者
var username = ‘张三’;
// 一条语句，声明多个变量
var username, age = 22, email=”zhangsan@126.com”;
// 如果重新声明 JavaScript 变量，该变量的值不会丢失
var username = ‘张三’;
var username;
```

一个编程的好习惯：在代码的开始处，统一声明所有的变量

## 十四　Javascript 关键字和保留字

关键字是语言保留，不能用作标识符，用于表示控制语句的开始或结束，或者用于执行特定操作的特殊字符。

1　ECMAScript 的关键字：

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

还有一些目前在 js 中还没有任何特定的用途，但将来有可能作为关键字的保留字。

2　保留字

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

**学习心得：**

_____

_____

_____

_____

_____

第十二天 - JS 入门

## 十五 Javascript 数据类型

基本类型：字符串、数字、布尔、Null、Undefined
复杂类型：对象（js 对象，数组...）

Javascript 是弱类型（动态类型）语言
也就是说，相同的 js 变量可以用作不同的类型。

```
var x;
x = 8;
x = ‘test’;
```

### 1 Javascript 字符串（String）

字符串是存储字符（比如 "Bill Gates"）的变量。
字符串可以是引号中的任意文本。您可以使用单引号或双引号。
可以在字符串中使用引号，只要不匹配包围字符串的引号即可。
如果要嵌套引号，请确保引号匹配，多于两层的引号嵌套要通过转义字符完成。

```
\n        // 换行
\r        // 回车
\’        // 单引号
\”        // 双引号
\\        // 反斜杠
```

### 2 Javascript 数字（Number）

JavaScript 只有一种数字类型。数字可以带小数点，也可以不带。

```
var x = 8;
var y = 1.2; // 小数或者浮点数
```

### 3 Javascript 布尔（Boolean）

布尔（逻辑）只能有两个值：true 或 false。

```
var a = true;
var b = false;
```

### 4 Javascript Null 和 Undefined

都表示变量不含有值。
Null 的唯一值是 null，表示指向对象的空指针（不指向任何对象）。
Undefined 的唯一值是 undefined，表示变量未赋值。
可以通过将变量的值设置为 null 来清空变量。

## 十六 声明变量类型

当您声明新变量时，可以使用关键词 "new" 来声明其类型：

```
var carName = new String(‘bmw’);
var x = new Number(1);
```

```
var y = new Boolean(true);
var cars = new Array(1, 2, 3);
var person = new Object();
```

JavaScript 变量均为对象。当您声明一个变量时，就创建了一个新的对象。

## 十七 Javascript 运算符

1 算术运算符

算术运算符用于执行变量与/或值之间的算术运算。

```
+       // 求和或字符串的连接
-
*
/
%       //取余
++      //累加
--      //递减
```

例子：

```
var a = 5;
var b = ‘test’;
var c = ‘5’;

// 求值
a + a
a + b
a + c
b + c
```

2 赋值运算符

```
=       //赋值
+=      // x += 3 等价于 x = x + 3
-=
*=
/=
%=
```

3 比较运算符

比较运算符在逻辑语句中使用，以测定变量或值是否相等。

```
==
===
!=
>
<
>=
<=
```

==和===的区别？

4　逻辑运算符

逻辑运算符用于测定变量或值之间的逻辑。

```
&&
||
!
```

**学习心得：**

第十三天 - JS 入门

## 十八 Javascript 条件语句

条件语句用于基于不同的条件来执行不同的动作。
在 JavaScript 中，我们可使用以下条件语句：
if 语句 - 只有当指定条件为 true 时，使用该语句来执行代码
if...else 语句 - 当条件为 true 时执行代码，当条件为 false 时执行其他代码
if...else if....else 语句 - 使用该语句来选择多个代码块之一来执行
switch 语句 - 使用该语句来选择多个代码块之一来执行

### 1 if 语句

只有当指定条件为 true 时，该语句才会执行代码。

```
// 语法
if (条件) {
  // 条件为true 时，执行的代码
}

// 例子
if (age < 18) {
  info = '未成年人';
}
```

### 2 if ... else ... 语句

当指定条件为 true 时，执行 if 后的代码块；条件为 false 时，执行 else 后的代码块。

```
// 语法
if (条件) {
  // 条件为true 时，执行的代码块
} else {
  // 条件为false 时，执行的代码块
}

// 例子
if (age < 18) {
  // info = '未成年人';
} else {
  // info = '成年人';
}
```

### 3 if ... else if ... else 语句

使用 if....else if...else 语句来选择多个代码块之一来执行。

```
// 语法
if (条件 1) {
  // 条件 1 为true 时，执行的代码块
} else if (条件 2) {
```

```
  // 条件 2 为 true 时，执行的代码块
} else {
  // 条件 1 和条件 2 都为 false 时执行的代码块
}
// 例子
if (x < 0 ) {
  // info = '负数';
} else if (x > 0) {
  // info = '正数';
} else {
  // info = '零';
}
```

4 switch 语句

用于基于不同的条件来执行不同的代码块。

使用 default 关键词来规定匹配不存在时做的事情。

```
// 语法
switch(待判断的值) {
  case 值 1:
  // 执行代码块 1;
  break;
  case 值 2:
  // 执行代码块 2;
  break;
  // 更多 case ...
  default:
  // 默认会执行的代码块（上面条件都不满足）
}
// 例子
switch (x) {
  case 0:
  info = '星期一';
  break;
  case 1:
  info = '星期二';
  break;
  // 更多星期 ...
  default:
  info = '请输入一个合法的数字';
}
```

工作原理：首先设置一个待判断的值（通常是一个变量）。随后表达式的值会与结构中的每个 case 的值做比较。如果存在匹配，则与该 case 关联的代码块会被执行。请使用 break 来阻止代码自动地向下一个 case 运行。

**学习心得：**

第十四天 - JS 入门

## 十九　循环语句

循环可以将代码块执行指定的次数。
如果你希望一遍又一遍地运行相同或类似的代码，循环语句最适合。
循环又叫迭代。

Javascript 中，循环语句包括下面几种：
for - 循环代码块一定的次数
for/in - 循环遍历对象的属性（暂略）
while - 当指定的条件为 true 时循环指定的代码块
do/while - 同样当指定的条件为 true 时循环指定的代码块

### 1　for 循环

```
// 语法
for (语句 1; 语句 2; 语句 3) {
  // 被执行的代码块
}
```

说明：
语句 1，在循环开始前执行，只执行一次，一般用来初始化变量
语句 2，评估循环执行的条件，true 则继续循环，false 则停止循环，每次循环前执行
语句 3，在每次循环完成后执行

```
// 例子
for (var i=0; i<10; i++) {
  console.log(i);
}
```

### 2　while 循环

while 循环会在指定条件为真时循环执行代码块。

```
// 语法
while (条件) {
  // 需要执行的代码
}

// 例子
var i = 0;while (i < 10) {
  console.log(i);
  i++;
}
// 如果忘记增加条件中所用变量的值，该循环永远不会结束。能导致浏览器崩溃。
```

### 3　do ... while 循环

do/while 循环是 while 循环的变体。该循环在检查条件是否为真之前总会执行一次，然后

如果条件为真的话，就会重复这个循环。

```
// 语法
do {
  // 需要执行的代码
} while (条件);

// 例子
var i = 0;
do {
  console.log(i);
  i++;
} while (i < 10);
// 别忘记增加条件中所用变量的值，否则循环永远不会结束！
```

## 二十　break 和 continue 语句

break 语句用于跳出循环。

continue 用于跳过循环中的一次循环。

break 语句跳出循环后，会继续执行该循环语句之后的代码（如果有的话）。

continue 语句中断当前正在执行的一次循环，直接继续循环中的下一次。

```
// break 语句
for (var i = 0; i < 10; i++) {
  if (i == 5 ) {
    break;
  }
}
// continue 语句
for (var i = 0; i < 10; i++) {
  if (i == 5 ) {
    continue;
  }
}
```

## 二一　Javascript 错误处理

try 语句测试代码块的错误。

catch 语句处理错误。

throw 语句创建自定义错误。

### 1　错误一定会发生

当 Javascript 引擎执行 Javascript 代码时，会发生各种错误：

可能是语法错误，通常是程序员造成的编码错误或错别字。

可能是拼写错误或语言中缺少的功能（可能由于浏览器差异）。

可能是由于来自服务器或用户的错误输出而导致的错误。

当然，也可能是由于许多其他不可预知的因素。

2 Javascript 抛出错误

当错误发生时，当事情出问题时，Javascript 引擎通常会停止，并生成一个错误消息。
描述这种情况的技术术语是：Javascript 将抛出一个错误。

3 Javascript 测试和捕捉

try 语句允许我们定义在执行时进行错误测试的代码块。
catch 语句允许我们定义当 try 代码块发生错误时，所执行的代码块。
Javascript 语句 try 和 catch 是成对出现的。

```
// 语法
try {
  // 被执行的代码
} catch (error) {
  // 被执行代码有错误在这里处理
}
// 例子
try {
  alert2('Hello');
} catch (error) {
  alert(error.message); // 没做任何处理，只是显示错误信息
}
```

4 throw 语句

throw 语句允许我们创建自定义错误。
正确的技术术语是：创建或抛出异常（exception）。
如果把 throw 与 try 和 catch 一起使用，那么您能够控制程序流，并生成自定义的错误消息。

```
// 语法
throw 异常信息（异常信息可以是 JavaScript 字符串、数字、逻辑值或对象）
```

**学习心得：**

第十五天 - JS 入门

## 二二 Javascript 函数入门

1 什么是函数？

函数是由事件驱动的、被调用时执行的、可重复使用的代码块。

2 函数定义的两种方式？

2-1 函数声明

```
// 语法
function 函数名() {
  // 函数体
}
```

2-2 函数表达式

```
// 语法
var 函数名 = function() {
  // 函数体
};
```

3 函数的调用

```
// 语法
函数名();

// 函数声明和函数表达式调用顺序上的区别
函数声明可以先调用再声明
函数表达式必须先声明再调用
```

4 函数的传参

```
// 语法
函数名(参数 1, 参数 2, 参数 3...);
```

5 函数的返回值

```
// 语法
function test() {
  return 函数的返回值;
}

// 说明
函数可以没有返回值，如果需要返回值，通过 return 语句返回 return 语句后面的语句不会被执行
```

## 二三 Javascript 对象入门

对象是带有属性和方法的特殊数据类型。

Javascript 中的所有事物都是对象：字符串、数值、数组、函数...

Javascript 允许自定义对象。
Javascript 提供多个内建对象，比如 String、Date、Array 等。

1　创建自定义 JavaScript 对象

```
// 创建空对象
var obj = new Object();
var obj = new Object; // 不推荐
var obj = {}; // 对象字面量方式（推荐）

// 给对象增加属性
obj.type = ‘笔记本电脑’;
obj.name = ‘联想’;
obj.price = 6500;
var obj = {
  type: ‘笔记本电脑’,
  name: ‘联想’,
  price: 6500
};
// 访问对象的属性
var type = obj.type;
var name = obj[‘name’]; // 属性名可以为变量
// 给对象增加方法
obj.getName = function() {
  return this.name; // obj.name;
}
var obj = {
  getName: function () {
    return this.name;
  }
};
// 修改对象的属性值
obj.price = 8000;
// 删除对象的属性
delete obj.type;
// 调用对象的方法
obj.getName();
obj[‘getName’](); // 不推荐
// 遍历对象的属性（for / in 语句）
for (var o in obj) {
  console.log(o);
  console.log(obj[o]); // console.log(obj.o); ?
}
```

**学习心得：**

第十六天 - JS 入门

2 Javascript 内置对象 Number

所有 JavaScript 数字均为 64 位。

整数（不使用小数点或指数计数法）最多为 15 位。

小数的最大位数是 17，但是浮点运算并不总是 100% 准确。

```
var x = 0.1 + 0.2; // 值为?
```

如果前缀为 0，则 JavaScript 会把数值常量解释为八进制数，如果前缀为 0 和 "x"，则解释为十六进制数。

```
var y = 0377;
var z = 0xFF;
```

常用方法：

toFixed()

3 Javascript 内置对象 String

String 对象用于处理已有的字符块。

```
// length 属性
var str = 'abcdefg';
console.log(str.length);

// 常用方法
charAt()        // 返回指定位置的字符
indexOf()       // 返回某个指定的字符串值在字符串中首次出现的位置
lastIndexOf()   // 返回一个指定的字符串最后出现的位置，从后向前搜索
split()         // 把一个字符串分割成字符串数组
substring()     // 用于截取字符串中介于两个指定下标之间的字符
substr()        // 在字符串中截取下标从参数 1 开始的指定数目（参数 2）的字符
toLowerCase()   // 把字符串转换为小写
toUpperCase()   // 把字符串转换为大写
```

4 Javascript 内置对象 Date

日期对象用于处理日期和时间。

4-1 定义（声明）Date 对象

```
var date = new Date(); // 通过 new 关键字
```

4-2 常用方法，get 方法都对应一个 set 方法

```
getFullYear()
getMonth()
getDate()
getDay()
getHours()
```

```
getMinutes()
getSeconds()
getTime()
```

```
// 日期的大小比较
var today = new Date();
var date = new Date();
date.setFullYear(2010);
if (today > date) {
  console.log('比今天早');
} else {
  console.log('比今天晚');
}
// 得到7天后的日期
var myDate = new Date();
myDate.setDate(myDate.getDate() + 7);
```

**学习心得：**

第十七天 - JS 入门

5 Javascript 内置对象 Array

数组对象用来在单独的变量名中存储一系列的值。

```
// 声明（定义）数组
var arr = new Array();   // 使用关键词 new 来创建数组对象
var arr = new Array(10); // 初始化数组的长度
var arr = [];            // 数组字面量

// 访问数组
var str = arr[0];

// 修改数组中已有的值
arr[0] = 'test'; // 字符串'test'会覆盖掉原来的字符串'a'

// 清空数组
arr.length = 0;

// 遍历数组
var len = arr.length, i;
for (i = 0; i < len; i++) {
  console.log(arr[i]);
}
var index;
for (index in arr) { // 不推荐
  console.log(arr[index]);
}
```

常用属性和方法

```
length
concat()
join()
pop()
push()
reverse()
shift()
sort()
unshift()
slice()
splice()
```

6 Javascript 内置对象 Boolean

Boolean（逻辑）对象用于将非逻辑值转换为逻辑值（true 或者 false）。

开发过程中一般不用逻辑对象进行逻辑值的转换，常见的方式是利用 Javascript 的隐式逻辑

转换功能达到非逻辑值转换成逻辑值的目的。
隐式转换规则：

| 数据类型 | 转换成 true | 转换成 false | 说明 |
| --- | --- | --- | --- |
| 字符串 | 任何非空字符串 | 空字符串 | 空格不是空字符串 |
| 数字 | 非 0 数字 | 0 | |
| Null | -- | null | |
| Undefined | -- | undefined | |
| 对象 | 任何对象 | null | |

7 Javascript 内置对象 Math

Math（算数）对象的作用是：执行常见的算数任务。
Math 对象提供多种算数值类型和函数。无需在使用这个对象之前对它进行定义。

7-1 常用方法：

```
abs()
ceil()
floor()
max()
min()
pow()
random()
round()
```

7-2 随机数公式：

```
值 = Math.floor(Math.random() * 可能值的总数 + 第一个可能的值)
```

8 Javascript 内置对象 RegExp

RegExp 对象用于规定在文本中检索的内容。
RegExp 是正则表达式的缩写。
当您检索某个文本时，可以使用一种模式来描述要检索的内容。RegExp 就是这种模式。
（详细内容暂略）

**学习心得：**

_____

_____

_____

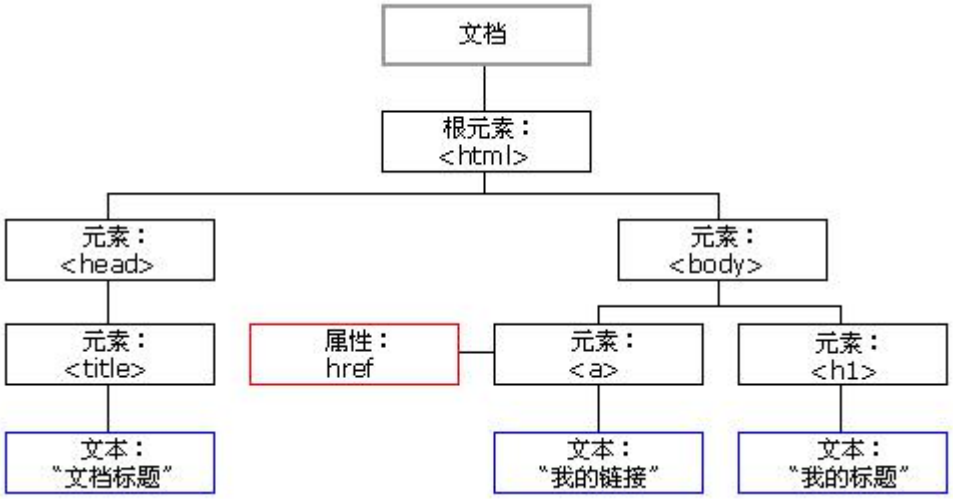
_____

_____

第十八天 - JS 入门

## 二四 HTML DOM

DOM 是 Document Object Model（文档对象模型）的简称。
通过 HTML DOM，Javascript 可访问 HTML 文档中的所有元素。
当网页被加载时，浏览器会根据 html 页面结构创建该页面的文档对象模型。
HTML DOM 模型被构造为对象的树，如下图：




通过可编程的对象模型，Javascript 获得了足够的能力来操作 HTML。
Javascript 能够改变页面中的所有 HTML 元素
Javascript 能够改变页面中的所有 HTML 属性
Javascript 能够改变页面中的所有 CSS 样式
Javascript 能够对页面中的所有事件做出反应

### 1 查找 HTML 元素

通常，你需要通过 Javascript 操作 HTML 页面中的元素。
为了做到这件事情，您必须首先找到该元素。有四种方法来做这件事：

#### 1-1 通过 id 找到 HTML 元素（最常用，效率高）

```
var myDiv = document.getElementById('myDiv');
```

如果找到该元素，则该方法将以对象（在 myDiv 中）的形式返回该元素。
如果未找到该元素，则 myDiv 为 null。

#### 1-2 通过标签名找到 HTML 元素

```
var span = myDiv.getElementsByTagName('span');
```

#### 1-3 通过 name 属性来找到表单元素

```
var gender = document.getElementsByName('gender');
```

#### 1-4 通过类名找到 HTML 元素（ie9+）

```
var redDiv = document.getElementsByClassName('red-div');
```

1-5 通过 CSS 选择器找到一个元素（ie9+）

```
var redDiv = document.querySelector('red-div');
```

1-6 通过 CSS 选择器找到一组元素（ie9+）

```
var redDiv = document.querySelectorAll('red-div');
```

2 操作 HTML 元素

document.write() 可用于直接向 HTML 输出流写内容。（不常用）

innerHTML 属性是修改 HTML 内容的最简单的方法。

```
var myDiv = document.getElementById('myDiv');
myDiv.innerHTML = '<h1>测试内容</h1>';
```

3 改变 HTML 属性。

```
var myDiv = document.getElementById('myDiv');
myDiv.title = '这是提示内容'; // 测试一下其它属性，都可以的哈
```

4 操作 CSS

```
var myDiv = document.getElementById('myDiv');
myDiv.style.color = 'red';
myDiv.style.backgroundColor = 'blue'; // 去掉中横线，大写首字母
var myDiv = document.getElementById('myDiv');
myDiv.style.color = 'red';
myDiv.style.backgroundColor = 'blue'; // 去掉中横线，大写首字母
```

5 对事件做出反应（绑定事件）

Javascript 中需要掌握的事件很多，目前我们只关心鼠标的单击（click）事件。

当鼠标在一个元素上单击时，就触发了 click 事件。对事件做出反映的方式：

```
// 事件属性
<button onclick="alert(''我被点击了)">一个按钮</button>
// 绑定事件
var myDiv = document.getElementById('myDiv');
myDiv.onclick = sayHello;function sayHello() {
  alert('Hello');
}
// 或者
var myDiv = document.getElementById('myDiv');
myDiv.onclick = function () {
  alert('Hello');
}
```

6 操作 HTML 文档

6-1 添加元素（追加）

如需向 HTML DOM 添加新元素，您必须首先创建该元素（元素节点），然后向一个已存在的元素追加该元素。

```
var myDiv = document.getElementById('myDiv');
var span = document.createElement('span');
var text = document.createTextNode('我是新的 span 元素');
span.appendChild(text);
myDiv.appendChild(span);
```

6-2 删除元素

如需删除 HTML 元素，必须先获得该元素的父元素，然后通过父元素删除子元素。

```
var myDiv = document.getElementById('myDiv');
var mySpan = document.getElementById('mySpan');
myDiv.removeChild(mySpan);
// 或者
mySpan.parentNode.removeChild(mySpan);
```

6-3 插入元素

```
var myDiv = document.getElementById('myDiv'); // 父元素
myDiv.insertBefore(newDom, targetDom);
// newDom 待插入元素
// 参考元素，即要插入到哪个元素前面
```

6-4 替换元素

```
var myDiv = document.getElementById('myDiv'); // 父元素
myDiv.replaceChild(newDom, targetDom);
// newDom 新元素
// 将要被替换掉的元素
```

**学习心得：**

第十九天 - JS 入门

## 二五 HTML BOM

BOM 是浏览器对象模型（Browser Object Model）。
浏览器对象模型 (BOM) 使 Javascript 有能力与浏览器“对话”（操作浏览器）。
浏览器对象模型尚无正式标准。
由于现代浏览器已经（几乎）实现了 Javascript 交互性方面的相同方法和属性，因此常被认为是 BOM 的方法和属性。

### 1 window 对象

所有浏览器都支持 window 对象。它表示浏览器窗口。
所有 JavaScript 全局对象、函数以及变量均自动成为 window 对象的成员。
全局变量是 window 对象的属性。
全局函数是 window 对象的方法。
甚至 HTML DOM 的 document 也是 window 对象的属性之一。

### 2 window 尺寸

有三种方法能够确定浏览器窗口的尺寸（浏览器的视口，不包括工具栏和滚动条）。
对于 Internet Explorer、Chrome、Firefox、Opera 以及 Safari：

```
window.innerHeight // 浏览器窗口的内部高度
window.innerWidth // 浏览器窗口的内部宽度
```

对于 Internet Explorer 8、7、6、5：

```
document.documentElement.clientHeight || document.body.clientHeight
document.documentElement.clientWidth || document.body.clientWidth
```

### 3 打开新窗口

```
window.open()
```

### 4 window 屏幕

```
window.screen    // 对象包含有关用户屏幕的信息。
screen.availWidth // 可用的屏幕宽度，以像素计，减去界面特性，比如窗口任务栏
screen.availHeight // 可用的屏幕高度，以像素计，减去界面特性，比如窗口任务栏
```

### 5 window Location

window.location 对象用于获得当前页面的地址 (URL)，并把浏览器重定向到新的页面。

```
location.hostname // 返回 web 主机的域名
location.pathname // 返回当前页面的路径和文件名
location.port     // 返回 web 主机的端口 （80 或 443 等）
location.protocol // 返回所使用的 web 协议（http:// 或 https://）
location.href     // 返回当前页面的 URL
location.reload() // 重新加载页面（刷新页面）
location.replace() // 跳转到新的页面（不能返回）
```

### 6 window 历史

```
history.back()     // 与在浏览器点击后退按钮相同
history.forward() // 与在浏览器中点击按钮向前相同
history.go()      // 可以实现 back(), forward(), reload()方法
```

7 window Navigator

window.navigator 对象包含有关访问者浏览器的信息。可以用于浏览器检测，但由于 navigator 数据可被浏览器使用者更改、浏览器无法报告晚于浏览器发布的新操作系统等原因可能误导开发者。UA 信息可以查看 navigator.userAgent。

8 消息框

```
alert()
confirm()
prompt()
```

9 定时器

在一个设定的时间间隔之后来执行代码，而不是在函数被调用后立即执行。称之为计时事件。实现计时事件的函数称为定时器。

9-1 延迟执行定时器

```
setTimeout()/clearTimeout()
```

9-2 循环执行定时器

```
setInterval()/clearInterval()
```

**学习心得：**

第二十天 - JS 入门

## 二六 Javascript 开发技巧与案例

1 Javascript 有多少中数据类型？分别是？

2 null 和 undefined 的区别？

3 == 和 === 的区别？

4 i++ 和 ++i 的区别？

5 && 和 || 的特殊用法总结。

6 合理使用 if ... else ... 语句和 ? : 表达式（三元表达式）。

7 有规律的字符串和数组间的互相转换方法？

8 判断一个字符串以某个字母开头？

9 制作一个电子时钟，要求误差尽量小。

10 掌握数字向上取整、向下取整、四舍五入、随机数方法的使用。

11 如何控制复选框的全选和取消全选。

12 掌握数组的遍历方法。

13 掌握数组的排序方法。

14 掌握数组的去重方法。

16 闪烁显示页面的标题。

17 用 Javascript 实现兼容所有浏览器的 document.getElementsByClassName()方法，要求在新的浏览器内直接调用浏览器的原生方法。

18 用 Javascript 实现兼容所有浏览器的 placeholder。

19 掌握非逻辑值转换成逻辑值的规则。

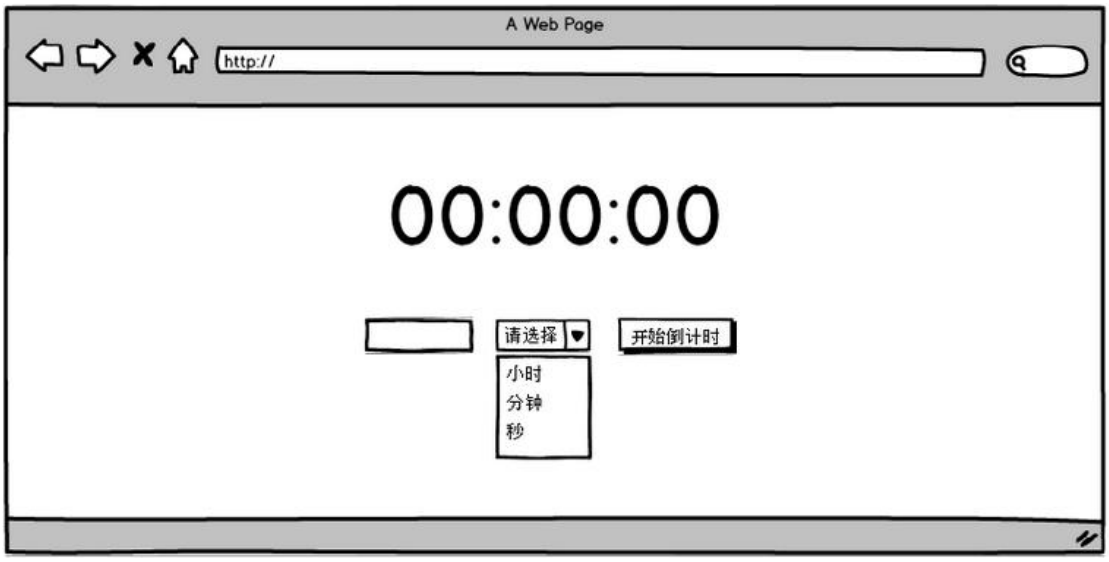
20 写一个函数，返回传入参数的数据类型。

21 带进度条的按钮

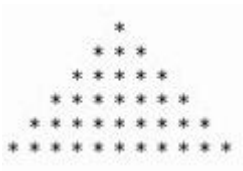
22 什么是 innerText？如何实现一个兼容所有浏览器的 innerText 方法？

23 用 Javascript 实现点击非超链接打开新窗口，考虑当前页面打开和新窗口打开的情况。

24 实现可设定时间的倒计时功能，如图：




25 用 js 打印金字塔，如图：



26 根据一个数组，动态生成下拉菜单。


默认一个空的下拉菜单和一个按钮
点击按钮后，下拉菜单有了按已知数组内容生成的选项
效果如上图
数组项：北京，上海，天津，杭州，深圳

27　用 Javascript 实现 99 乘法表。

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1*1=1 | | | | | | | | |
| 2*1=2 | 2*2=4 | | | | | | | |
| 3*1=3 | 3*2=6 | 3*3=9 | | | | | | |
| 4*1=4 | 4*2=8 | 4*3=12 | 4*4=16 | | | | | |
| 5*1=5 | 5*2=10 | 5*3=15 | 5*4=20 | 5*5=25 | | | | |
| 6*1=6 | 6*2=12 | 6*3=18 | 6*4=24 | 6*5=30 | 6*6=36 | | | |
| 7*1=7 | 7*2=14 | 7*3=21 | 7*4=28 | 7*5=35 | 7*6=42 | 7*7=49 | | |
| 8*1=8 | 8*2=16 | 8*3=24 | 8*4=32 | 8*5=40 | 8*6=48 | 8*7=56 | 8*8=64 | |
| 9*1=9 | 9*2=18 | 9*3=27 | 9*4=36 | 9*5=45 | 9*6=54 | 9*7=63 | 9*8=72 | 9*9=81 |

提示：for 循环

28 用 Javascript 实现随机选择工具。


默认情况下有下面几个点可以选择：故宫、长城、十三陵、十渡、圆明园、颐和园
用户点击开始，随机快速显示上面的某个地点
用户点击停止，显示当前的随机地点
下面有个文本框（如图），在文本框输入地点，点击“新增”，则在上面默认的选项中加入新的选项
可一次增加多个选项，选项间用“|”分割
添加完成后清文本框
如果添加的项已经存在，提示用户，并且不进行添加

29 根据一个由对象组成的数组，渲染一个表格到页面。

30 Javascript 实现表单简单验证。

31 Javascript 函数进阶。

32 Javascript 作用域。

33 Javascript 引用类型。

34 Javascript 实现 tab。