《JavaWeb课程》教案

第3章 js

[1. 回顾上一章: [10分钟] 3](#_Toc6546)

[2. 课程知识点讲解： 3](#_Toc32133)

[2.1. javascript简介 [80分钟] 3](#_Toc2119)

[2.1.1. javascript定义 3](#_Toc11357)

[2.1.2. javascript特点 3](#_Toc31354)

[2.1.3. javascript组成 3](#_Toc2742)

[2.1.4. 基本结构 3](#_Toc11574)

[2.1.5. javascript执行原理： 4](#_Toc17228)

[2.1.6. 在网页中引入javascript的其他方式： 4](#_Toc25278)

[2.1.7. js代码注释 4](#_Toc4091)

[2.2. 核心语法 [60分钟] 5](#_Toc4988)

[2.2.1. 数据类型 5](#_Toc7553)

[2.2.2. 变量（弱数据类型，typeof） 5](#_Toc24390)

[2.2.3. 内置对象Math,Date,Array,String 7](#_Toc25125)

[2.2.4. 运算符 10](#_Toc2544)

[2.2.5. 逻辑控制语句（简单介绍，java中讲过） 11](#_Toc9865)

[2.2.6. 函数 13](#_Toc25921)

[2.2.7. 事件 16](#_Toc14911)

[2.3. DOM [80分钟] 17](#_Toc23494)

[2.3.1. DOM定义 17](#_Toc23000)

[2.3.2. DOM与网页的关系 17](#_Toc11041)

[2.3.3. 常用获取DOM 的方法 17](#_Toc14973)

[2.3.4. 鼠标移入、移出事件 17](#_Toc4532)

[2.4. BOM [60分钟] 19](#_Toc3933)

[2.4.1. BOM定义 19](#_Toc4838)

[2.4.2. BOM的应用场景： 20](#_Toc16239)

[2.4.3. window对象的方法： 20](#_Toc20052)

[2.4.4. history对象 20](#_Toc30297)

[2.4.5. location对象 20](#_Toc2464)

[2.5. 正则表达式 [80分钟] 22](#_Toc20858)

[2.5.1. onsubmit事件 22](#_Toc18890)

[2.5.2. onfocus事件, onblur事件 22](#_Toc12912)

[2.5.3. 正则表达式： 22](#_Toc47)

[3. 本章总结 [20分钟] 25](#_Toc19081)

[4. 测试题 [分钟] 25](#_Toc12454)

[5. 作业 [30分钟] 25](#_Toc20225)

[6. 扩展部分（选讲内容）： 25](#_Toc24910)

[7. 学员问题汇总（经验积累）： 25](#_Toc1024)

编创教师:

课时：420分钟

本章主要目标

掌握js定义、js特点、js组成

熟练运用js的语法

掌握js的ＤＯＭ操作及常用事件

掌握BOM的常用方法和属性

掌握表单事件及正则表达式的运用

本章重点

熟练掌握变量声明，函数，事件和定时器的使用

运用正则表达式验证表单

本章难点

函数的理解

事件的运用

本章授课注意事项

1、本章采用由浅入深，逐层深入的的授课思路，教学过程中注意控制进度，避免出现前一阶段没有跟上，后一阶段只能放弃学习的情况

2、本章知识点讲解中主要使用先讲解理论再结合代码，并与之前学过的java做对比，帮助学员理解js语法和对js的运用。

# 回顾上一章: [10分钟]

回顾css样式。



鼠标悬停时如何改变样式。

答：添加伪类 ：hover

几种定位方式？

答：4种

分别是什么？

答：static、absolute、relative、fixed

# 课程知识点讲解：

javascript简介 [80分钟]



观察<https://www.foxmovies.com/> 这个页面中的效果能用html和css实现吗？

要实现这样的效果就需要用到javascript。

Javascript的作用主要是实现页面的动态效果和表单验证。

### javascript定义

js定义：**基于对象**和**事件驱动**的**嵌入式**脚本语言。

基于对象： 使用对象的属性和方法。有js内置对象，也可以自定义对象。

事件驱动： 例如：当用户点击的时候，执行函数。点击就是触发了事件

嵌入式： 不能单独执行，需要嵌入到html中使用。

### javascript特点

JS特点（语法和java类似）：JavaScript一种直译式脚本语言，是一种动态类型、弱类型（java强类型）、基于原型（java基于类）的语言。它的解释器被称为JavaScript引擎，为浏览器的一部分，广泛用于客户端的脚本语言。js是一种**解释性**脚本语言（代码不进行预编译）

### javascript组成

**ECMAScript**，描述了该语言的语法和基本对象。

**文档对象模型（DOM）**，描述处理网页内容的方法和接口。

**浏览器对象模型（BOM）**，描述与浏览器进行交互的方法和接口。

### 基本结构

<script type="text/javascript">

<!—

JavaScript 语句;

//—>

</script >

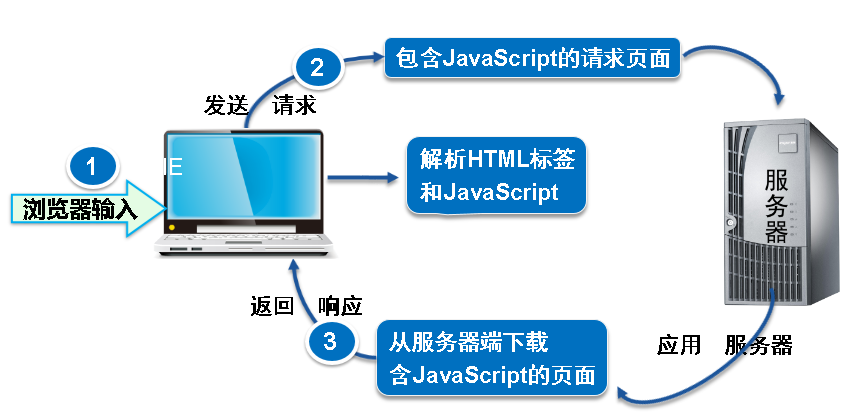


1、script标签放在Html的任何位置都可以

2、script标签的数量可以有多个，没有限制。但最好使用一个，一般放在页面底部。

1. 代码中的注释的作用： 不支持 JavaScript 的浏览器会把脚本作为页面的内容来显示。为了防止这种情况发生，我们可以使用这样的 注释来解决 。但在基本不用，几乎所有浏览器都支持js。
2. script标签中的type属性。type 属性规定脚本的 MIME 类型。对于 JavaScript，其 MIME 类型是 "text/javascript"。默认的是javascript，一般省略不写。

### javascript执行原理：



### 在网页中引入javascript的其他方式：

1、 <button onclick="alert(123);">点击</button>

2、 从外部引入js文件 <script src=’js/abc.js’></script>



**案例代码**：（源文件名称）

<body>

<button onclick="alert(123);">点击</button>

<script>

document.write("<h1>这是一个标题</h1>");

</script>

<!--<script src="js/abc.js"> </script>-->

</body>



1. 外部文件不能包含<script>标签，通常将.js文件放到网站目录中单独存放脚本的子目录中（一般为js）， 这样容易管理和维护
2. js代码注释：当页面中有不需要执行的语句，或需要做标注时，使用注释。

1、单行注释 //

2、多行注释 /\* \*/



练习1：

参考代码：使用document.write在页面中输出内容。并使用3种js的引入方式来实现。

核心语法 [60分钟]

### 数据类型

数据类型： 5 种简单数据类型（也称为基本数据类型）：Undefined、Null、Boolean、Number和 String。

还有 1种复杂数据类型——Object，Object 本质上是由一组无序的名值对组成的。

undefined：未定义

 boolean：布尔值 例：true false

 string：字符串 例：‘abc’或 “abc”

 number：数值 例：100 2.5

 object：对象或 null 例：{“name”：“tom”}

null： 空对象

### 标识符

标识符，就是指变量、函数、属性的名字，或者函数的参数。

标识符可以是按照下列格式规则组合起来的一或多个字符：

 **第一个字符**必须是一个字母、下划线（\_）或一个美元符号（$）；

 其他字符可以是字母、下划线、美元符号或数字。

按照惯例，ECMAScript 标识符采用**驼峰大小写格式**，也就是第一个字母小写，剩下的每个单词的首字母大写，

例如：firstSecond myCar doSomethingImportant

关键字：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。

保留字： 尽管保留字在js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。

ECMAScript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 |  |  |

ECMAScript的保留字:

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |  |

### 变量

变量：使用关键字 var 。可以把变量看做存储数据的容器。

ECMAScript 定义变量函数的语法：

1、变量名 函数名 避免关键字和保留字

2、ECMAScript 中的一切（变量、函数名和操作符）都区分大小写。

变量名 test 和变量名 Test 分别表示两个不同的变量

var 变量名; 例: var num;

var 变量名，变量名，……; 例: var num1, num2, num3;

var 变量名＝值; 例: var a = 1;

var 变量名=值，变量名＝值; 例: var a =1, b=2, c=3;

### typeof 操作符

作用：查看变量或值的数据类型

对一个值使用 typeof 操作符可能返回下列某个字符串：

 "undefined"——如果这个值未定义；

 "boolean"——如果这个值是布尔值；

 "string"——如果这个值是字符串；

 "number"——如果这个值是数值；

 "object"——如果这个值是对象或 null；

"function"——如果这个值是函数。



**案例描述**：

任务描述

**案例代码**：（源文件名称）

var num;

console.log(typeof(num));

num=10;

console.log(typeof(num));

num=3.14;

console.log(typeof(num));

num=”sdfgsdf”;

console.log(typeof(num));

num=true;

console.log(typeof(num));

num=false;

console.log(typeof(num));

num=“10”;

console.log(typeof(num));

num=null;

console.log(typeof(num));

num=function () { }

console.log(typeof(num));



1. javascript是弱数据类型，在声明变量时不指定数据类型。
2. 变量中可存储任何类型的数据
3. 变量中存储何种类型数据，变量就是何种类型
4. 可以不声明变量，直接使用（不推荐）



练习1：练习声明变量，将所有的数据类型的都练习，并在控制台输出。

### 内置对象Array

数 组：数组对象是使用单独的变量名来存储一系列的值。

创建一个数组，有三种方法：常规方式、简洁方式、字面量式

1: 常规方式:

var 变量名= new Array();

2: 简洁方式:

var 变量名 =new Array( 值1, 值2, 值3，……);

3: 字面:

var 变量名=[值1, 值2, 值3，……];

数组的属性和方法：

var x=week.length // week中元素的数量

删除数组的最后一个元素 - pop()

数组的末尾添加新的元素 - push()

删除数组的第一个元素 - shift()

在数组的开头添加新元素 - unshift()

join() 把数组的所有元素放入一个字符串。元素通过指定的分隔符进行分隔。

reverse() 颠倒数组中元素的顺序。



**案例描述**：

任务描述

**案例代码**：（源文件名称）

创建数组：

1: 常规方式:

var week = new Array();

week [0]="Monday";

week [1]="Tuesday";

week [2]="Wednesday";

2: 简洁方式:

var week =new Array("Monday","Tuesday","Wednesday");

3: 字面:

var week =["Monday","Tuesday","Wednesday"];

var x=week.length;

console.log(x); // 数组长度

var num1= [1, 2, 3, 4];

num.pop();

console.log(num1);

var num2= [1, 2, 3, 4];

num.push(5);

console.log(num2);

var num3= [1, 2, 3, 4];

num.shift();

console.log(num3);

var num4= [1, 2, 3, 4];

num.unshift(0);

console.log(num4);

var num5= [1,2,3,4];

num5.reverse();

console.log(num5);

var num6= [1,2,3,4];

console.log(num6.join(‘|’));



练习1：练习/2.2数组方法/彩色条纹，练习/2.2数组方法/切换彩色条纹

### 内置对象String

字符串：一个字符串用于存储一系列字符

字符串方法：

indexOf() 方法

使用 indexOf() 来定位字符串中某一个指定的字符首次出现的位置。没有则返回-1

match() 方法

使用 match() 来查找字符串中特定的字符，并且如果找到的话，则返回这个字符。

split()方法

通过将字符串划分成子串，将一个字符串做成一个字符串数组。



**案例描述**：

任务描述

**案例代码**：（源文件名称）

字符串：

var str = ‘123’；

console.log(typeof （str）); // string

console.log(str.length);

indexOf方法：

var str01="Hello world!"

document.write(str01.indexOf("Hello")) // 0

document.write(str01.indexOf("World")) // -1

document.write(str01.indexOf("world")) //6

match() 方法:

var str02="Hello world!"

document.write(str02.match("world") + "<br />") // world

document.write(str02.match("World") + "<br />") // null

document.write(str02.match("worlld") + "<br />") //null

document.write(str02.match("world!")) //world!

split()方法：

var str03 = "aa.bb.cc"

console.log(str03.split(".")); //使用打印看数组结构



1）一个字符串可以使用单引号或双引号

2）字符串的索引从零开始, 所以字符串第一字符为 [0],第二个字符为 [1]



练习1：练习/2.2字符串方法/反向排列输入的值

### 内置对象Date

**日期**：

通过new关键字来定义日期对象。

Date 对象自动使用当前的日期和时间作为其初始值。

getDate() 从 Date 对象返回一个月中的某一天 (1 ~ 31)。 每月的1-31号

getDay() 从 Date 对象返回一周中的某一天 (0 ~ 6)。 周一到周日 0：周日 1-6：周一至周六

getFullYear() 从 Date 对象以四位数字返回年份。

getHours() 返回 Date 对象的小时 (0 ~ 23)。

getMinutes() 返回 Date 对象的分钟 (0 ~ 59)。

getMonth() 从 Date 对象返回月份 (0 ~ 11)。 0代表1月，以此类推

getSeconds() 返回 Date 对象的秒数 (0 ~ 59)。

getTime() 返回 1970 年 1 月 1 日至今的毫秒数。



**案例描述**：

任务描述

**案例代码**：（源文件名称）

不带参数，当前时间:

var d=new Date();

d.getDate(); //日期

d.getDay(); //一周的某天

d.getFullYear(); // 年

d.getHours(); // 小时

d.getMinutes(); //分钟

d.getMonth(); //月

d.getSeconds() ; //秒

d.getTime(); //返回从 1970 年 1 月 1 日至今的毫秒数

带参数：

var d1 = new Date(2000,1,1); //设置了一个特定的日期 (2000 年 1 月 1 日)

var d2 = new Date("October 13, 1975 11:13:00");

var d3 = new Date(79,5,24,11,33,0);



1. 日期对象的getDay()方法获取的值代表什么？



练习1：练习/2.2日期对象/系统时间

### 运算符

算术运算符用于执行变量与/或值之间的算术运算。

给定 y=5，下面的表格解释了这些算术运算符：



赋值运算符用于给 JavaScript 变量赋值。

给定 x=10 和 y=5，下面的表格解释了赋值运算符：



### 逻辑控制语句

**if ... else**： 用于基于不同的条件来执行不同的动作。

If判断：

if (条件 1)

{

当条件 1 为 true 时执行的代码

}

else if (条件 2)

{

当条件 2 为 true 时执行的代码

}

else

{

当条件 1 和 条件 2 都不为 true 时执行的代码

}

**switch 语句**：选择要执行的多个代码块之一

**while 循环**：while 循环会在指定条件为真时循环执行代码块。

while (条件)

{

需要执行的代码

}

**do/while 循环**会至少执行一次，即使条件是 false。

do/while 循环会执行一次代码块，在检查条件是否为真之前，然后如果条件为真的话，就会重复这个循环。

do

{

需要执行的代码

}

while (条件);



**案例描述**：

任务描述

**案例代码**：（源文件名称）

// Switch语句 ：

var day=new Date().getDay();

switch (day)

{

case 0:

x="Today it's Sunday";

break;

case 1:

x="Today it's Monday";

break;

case 2:

x="Today it's Tuesday";

break;

case 3:

x="Today it's Wednesday";

break;

case 4:

x="Today it's Thursday";

break;

case 5:

x="Today it's Friday";

break;

case 6:

x="Today it's Saturday";

break;

}

// while

while (i<5)

{

x=x + "The number is " + i + "<br>";

i++;

}

// do while

do

{

x=x + "The number is " + i + "<br>";

i++;

}

while (i<5);

// for循环：

var cars=["BMW","Volvo","Saab","Ford"];

for (var i=0; i<cars.length; i++) {

console.log(cars[i]);

}



练习1：数组去重

<script type="text/javascript">

var arr = [1,3,5,6,76,5,2,34,2,2,45,1,2];

function unique(arr){

var res =[];

var json = {};

for(var i=0;i<arr.length;i++){

if(!json[arr[i]]){

　　res.push(arr[i]);

　　json[arr[i]] = 1;

}

}

return res;

}

unique(arr);

console.log(unique(arr));

</script>

### 函数

系统函数：

parseInt(),parseFloat(),isNaN()

**parseInt():**parseInt() 函数可解析一个字符串，并返回一个整数。

**parseFloat()：**parseFloat() 函数可解析一个字符串，并返回一个浮点数。

**isNaN()**： 函数用于检查其参数是否是非数字值。如果参数值为 NaN 或字符串、对象、undefined等非数字值则返回 true, 否则返回 false。



**案例描述**：

学习系统函数的使用

**案例代码**：（源文件名称）

<script>

document.write(parseInt("10") + "<br>");

document.write(parseInt("10.33") + "<br>");

document.write(parseInt("34 45 66") + "<br>");

document.write(parseInt(" 60 ") + "<br>");

document.write(parseInt("40 years") + "<br>");

document.write(parseInt("He was 40") + "<br>");

document.write(parseFloat("10") + "<br>");

document.write(parseFloat("10.33") + "<br>");

document.write(parseFloat("34 45 66") + "<br>");

document.write(parseFloat(" 60 ") + "<br>");

document.write(parseFloat("40 years") + "<br>");

document.write(parseFloat("He was 40") + "<br>");

document.write(isNaN(123)+ "<br>");

document.write(isNaN(-1.23)+ "<br>");

document.write(isNaN(5-2)+ "<br>");

document.write(isNaN(0)+ "<br>");

document.write(isNaN("Hello")+ "<br>");

document.write(isNaN(NaN)+ "<br>");

</script>



1）字符串中只返回第一个数字。开头和结尾的空格是允许的。如果字符串的第一个字符不能被转换为数字，那么 parseFloat() 会返回 NaN非数字。

2）注意区分形参和实参



练习1：使用上述系统函数，自己设置参数进行联系。每个函数尝试10次

2.2.10.2 **自定义函数**

有名的函数、匿名函数。这里重点讲解有名的函数

无参数时：

function 函数名() {

alert('123');

}

函数名(); // 函数调用

有参数时：

function 函数名(形参) {

alert('123');

}

函数名(实参); // 函数调用

有返回值情况：

function 函数名() {

// 一些代码

return xx； // 返回值

}

函数名(); // 函数调用



**案例描述**：

学习自定义函数的使用

**案例代码**：（源文件名称）

**function play (time) {**

**alert(time + '点了,去玩耍了！');**

**}**

**function eat (time) {**

**alert(time + '点了,吃早饭了！');**

**}**

**function study (time) {**

**alert(time + '点了,好好学习！');**

**}**

**eat(8); // 函数调用**

**study (10);**

**play (19);**

**function myFunction(){**

**var x=5;**

**return x;**

**}**

**var myVar=myFunction();**

**console.log(myVar);**



练习1：

结合if判断:

var time = 20;

if (7<time && time<9) {

eat(time);

}else if (9<=time && time<=18) {

study (time);

}else if (18< time < 22) {

play (time);

}

function eat (time) {

console.log(time + '点了,吃早饭了！');

}

function study (time) {

console.log(time + '点了,好好学习！');

}

function play (time) {

console.log(time + '点了,去玩耍了！');

}

### 事件

函数是由事件驱动的或者当它被调用时执行的可重复使用的代码块。

onclick点击事件

**<button onclick="message()">点击这里</button>**

onload事件：onload 事件会在页面或图像加载完成后立即发生。

**<body onload="SomeJavaScriptCode">**

给img加onload事件：

**<img src="logo.png" onload="loadImage()" />**



**案例描述**：

学习事件的使用：点击事件、onload事件

**案例代码**：（源文件名称）

**综合onload事件:**

**<body>**

**<img id="myImg" width="1000" src="http://img2.niutuku.com/desk/1208/1401/ntk-1401-8806.jpg" />**

**</body>**

**<script>**

**window.onload = function(){**

**alert("页面dom加载完成");**

**}**

**document.getElementById("myImg").onload = function(){**

**alert("图片已经加载完成");**

**}**

**</script>**



1）onload 通常用于 <body> 元素，在页面完全载入后(包括图片、css文件等等。)执行脚本代码。



练习1：练习/2.2 TAB切换效果

DOM [80分钟]

当网页被加载时，浏览器会创建页面的文档对象模型（Document Object Model）。

HTML DOM 模型被构造为对象的树：



### DOM定义

定义了访问和操作 HTML 文档的标准方法。

DOM就是是关于如何获取、修改、添加或删除文档 元素的标准。

### DOM与网页的关系

html文档中的元素都是一个个节点。JS可以获取修改、添加或删除DOM 节点，来改变页面结构。

### 常用获取DOM 的方法

|  |  |  |
| --- | --- | --- |
| 方法 | 区别 | 获取到的元素 |
| getElementById(‘’) | 通过id名获取 | 单个元素 |
| document.getElementsByTagName(‘’) | 通过标签名获取 | 一组元素 |
| document.getElementsByClassName(‘’) | 通过class名获取 | 一组元素 |



**案例描述**：

练习获取页面中的元素节点并添加事件

**案例代码**：

<body>

<button id =“myBtn”></button>

<div class =“box”></div>

<p>123</p>

<p>456</p>

</body>

<script>

var myBtn = document.getElementById("myBtn");

var aP = document.getElementByTagName("p");

var box = document.getElementByClassName("box");

console.log(myBtn);

console.log(aP);

console.log(box);

// 获取到DOM元素后添加事件

/\*

document.getElementById("myBtn").onclick=function(){

message();

};

function message(){

document.getElementById("box").innerHTML=‘点击事件’;

}

\*/

</script>



**案例代码**：（源文件名称）案例/2.3DOM/实现全选及全不选效果



**案例代码**：（源文件名称）案例/2.3DOM/点击导航下拉菜单.html



1）获取页面中的DOM的方式？

****

1）js代码要写在html标签之后，否则获取不到DOM。或者使用 window.onload = function () { js代码 }



练习1：实现留言板功能

见 : 练习/2.3 DOM事件/06综合\_留言板.html

练习2：实现树形菜单

见：练习/2.3制作树形菜单

### 鼠标移入、移出事件

onmouseover鼠标移入事件 和onmouseout鼠标移出事件



**案例描述**：

练习添加鼠标事件

**案例代码**：

<style>

div {

background-color:#D94A38;

width:120px;

height:20px;

padding:40px;

}

</style>

<!-- this代表当前对象。这里就是div -->

<div onmouseover="mOver(this)" onmouseout="mOut(this)" >

Mouse Over Me

</div>

<script>

function mOver(obj){

obj.innerHTML="你好！";

}

function mOut(obj){

obj.innerHTML="再见！";

}

</script>



**案例代码**：（源文件名称）案例/2.3DOM/鼠标移入移出导航下拉菜单.html



练习1：将案例 鼠标移入移出导航下拉菜单.html 自己动手实现

BOM [60分钟]

### BOM定义

浏览器对象模型 (BOM) 是 JavaScript 与 浏览器 ‘交流’的桥梁。它提供了很多对象,用于访问浏览器的功能;这些功能与任何网页内容无关。

### BOM的应用场景：

网页的前进后退，查看浏览历史记录，查看浏览器信息等。

### window对象的方法：

alert(),confirm(),prompt();

网页的打开关闭： open(),close();

定时器： setTimeout(),setInterval();



**案例描述**：

alert(),confirm(),prompt();3种方法的使用

**案例代码**：（源文件名称）案例/2.4BOM/alert-confirm-prompt.html



**案例描述**：

网页的打开关闭： open(),close();

**案例代码**：（源文件名称）案例/2.4BOM/open-close.html



**案例描述**：

练习定时器

**案例代码**：

**开启setInterval：**

<p id="demo"></p>

<script>

var myVar=setInterval(function(){myTimer()},1000);

function myTimer(){

var d=new Date();

var t=d.toLocaleString();

document.getElementById("demo").innerHTML=t;

}

</script>

**停止setInterval：**

<p id="demo"></p>

<button onclick="myStopFunction()">停止时钟</button>

<script>

var myVar=setInterval(function(){myTimer()},1000);

function myTimer(){

var d=new Date();

var t=d.toLocaleTimeString();

document.getElementById("demo").innerHTML=t;

}

function myStopFunction(){

clearInterval(myVar);

}

</script>

**开启setTimeout()：**

3秒后弹出‘Hello’

<button onclick="myFunction()">点击</button>

<script>

function myFunction(){

setTimeout(function(){alert("Hello")},3000);

}

</script>

**停止setTimeout()：**

<p>点击第二个按钮来阻止第一个函数运行。（你必须在3秒之前点击它）。</p>

<button onclick="myFunction()">点我</button>

<button onclick="myStopFunction()">停止弹框</button>

<script>

var myVar;

function myFunction(){

myVar=setTimeout(function(){alert("Hello")},3000);

}

function myStopFunction(){

clearTimeout(myVar);

}

</script>



1）window对象的属性有？

2）window对象的方法可以如何简写？

****

1. window对象的几个属性和方法，可以省略 window. ,直接使用。
2. 两种定时器的区别：

setTimeout：在一定时间之后执行函数；

setInterval：每隔一定时间就执行一次函数。



练习1：

见 练习/2.4BOM/定时器-图片切换

### history对象

history对象方法：

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

### location对象

Location：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对象属性：

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）

location.protocol 返回所使用的 web 协议（http: 或 https:）



**案例描述**：

location对象属性的使用

**案例代码**：

<script>

document.write(location.href);

document.write(location.hostname );

document.write(locationpathname);

document.write(location.port);

document.write(location.protocol);

</script>



**案例描述**：

历史记录的前进后退 以及 页面的刷新、跳转功能的实现

**案例代码**：（源文件名称）案例/2.4BOM/ history.html history1.html location.html

正则表达式 [80分钟]

### **正则表达式**：

正则表达式描述了字符的模式对象。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。

方法：test() 方法搜索字符串指定的值，根据结果并返回真或假。

exec() 方法检索字符串中的指定值。返回值是被找到的值。如果没有发现匹配，则返回 null。

RegExp 修饰符：修饰符用于执行不区分大小写和全文的搜索。

|  |  |
| --- | --- |
| i | 执行对大小写不敏感的匹配。 |
| g | 执行全局匹配（查找所有匹配而非在找到第一个匹配后停止）。 |
| m | 执行多行匹配。 |

方括号用于查找某个范围内的字符：

|  |  |
| --- | --- |
| **表达式** | **描述** |
| [[abc]](http://www.w3school.com.cn/jsref/jsref_regexp_charset.asp) | 查找方括号之间的任何字符。 |
| [[^abc]](http://www.w3school.com.cn/jsref/jsref_regexp_charset_not.asp) | 查找任何不在方括号之间的字符。 |
| [0-9] | 查找任何从 0 至 9 的数字。 |
| [a-z] | 查找任何从小写 a 到小写 z 的字符。 |
| [A-Z] | 查找任何从大写 A 到大写 Z 的字符。 |
| [A-z] | 查找任何从大写 A 到小写 z 的字符。 |
| [adgk] | 查找给定集合内的任何字符。 |
| [^adgk] | 查找给定集合外的任何字符。 |
| (red|blue|green) | 查找任何指定的选项。 |

元字符（Metacharacter）是拥有特殊含义的字符：

|  |  |
| --- | --- |
| **元字符** | **描述** |
| [.](http://www.w3school.com.cn/jsref/jsref_regexp_dot.asp) | 查找单个字符，除了换行和行结束符。 |
| [\w](http://www.w3school.com.cn/jsref/jsref_regexp_wordchar.asp) | 查找单词字符。 |
| [\W](http://www.w3school.com.cn/jsref/jsref_regexp_wordchar_non.asp) | 查找非单词字符。 |
| [\d](http://www.w3school.com.cn/jsref/jsref_regexp_digit.asp) | 查找数字。 |
| [\D](http://www.w3school.com.cn/jsref/jsref_regexp_digit_non.asp) | 查找非数字字符。 |
| [\s](http://www.w3school.com.cn/jsref/jsref_regexp_whitespace.asp) | 查找空白字符。 |
| [\S](http://www.w3school.com.cn/jsref/jsref_regexp_whitespace_non.asp) | 查找非空白字符。 |
| [\b](http://www.w3school.com.cn/jsref/jsref_regexp_begin.asp) | 匹配单词边界。 |
| [\B](http://www.w3school.com.cn/jsref/jsref_regexp_begin_not.asp) | 匹配非单词边界。 |
| \0 | 查找 NUL 字符。 |
| [\n](http://www.w3school.com.cn/jsref/jsref_regexp_newline.asp) | 查找换行符。 |

量词：

|  |  |
| --- | --- |
| n+ | 匹配任何包含至少一个 n 的字符串。 |
| n\* | 匹配任何包含零个或多个 n 的字符串。 |
| n? | 匹配任何包含零个或一个 n 的字符串。 |



**案例描述**：

正则表达式的语法及定义

**案例代码**：

以下这两种写法是等价的。

var re = new RegExp("\\w+",'g');

var re = /\w+/g;

<script>

var str = "Is this all there is?";

var re = /is/g;

document.write(str.match(re));

/\*

var patt1=new RegExp("e");

document.write(patt1.test("The best things in life are free"));

var patt2=new RegExp("e");

document.write(patt2.exec("The best things in life are free"));

\*/

</script>



**案例描述**：

正则表达式的常用用法

**案例代码**：（源文件名称） 案例 / 2.5表单、正则 / 03正则表达式.html



1）正则表达式的定义方式？修饰符的作用？

2）正则的两个方法？如何使用

****

1. 通常使用简易写法： /\w+/g，需要将正则存入变量时，使用： new RegExp("\\w+",'g');

### onsubmit事件

表单提交事件

### onfocus事件, onblur事件

表单获取焦点事件、失去焦点事件



**案例描述**：

通常正则表达式与表单结合使用

**案例代码**：（源文件名称） 案例 / 2.5表单、正则 / 01表单事件.html 02onfocus、onblur.html



练习1：

参考代码：(文件路径：练习/2.5正则表达式/邮箱格式)

练习2：

参考代码：(文件路径：练习/2.5正则表达式/02表单验证)

# 本章总结 [20分钟]

见 其他参考资料/导图

# 测试题 [分钟]

Key1 测试题1：

Key1 测试题2：

# 作业 [30分钟]

writer01 习题1：轮播图 （见： 案例/轮播图）

writer01 习题2：省市县三级联动 （见： 案例/省市县联动）

# 扩展部分（选讲内容）：

* 扩展部分1：
* 扩展部分2：

# 学员问题汇总（经验积累）：

* 学员听课中提问的问题：
* 学员练习中遇到的问题：