

回顾

1.html作用：书写网页的，整个网页的架构

2.html属于不用编译的语言，直接使用浏览器运行即可

3.html都是由预定义的标签：

1) 超链接标签：

2) form表单标签：属性： action:提交数据的服务器地址 method:请求方式。默认是get post

3) 书写在form表单标签中的input标签，表示输入项。

a: 属性： type,具有属性值： text password radio checkbox submit reset button date等

b: 属性： name:后台通过name="username"属性值获取value="哈哈" <input value="哈哈" name="username"/>

4)table标签：表格，主要用来让页面更加规整

tr:行

属性： align:left right center

td:列

属性：

align:left right center

rowspan:跨行

colspan:跨列

th:列，自动加粗 居中

属性：

rowspan:跨行

colspan:跨列

5) 图片标签：img src="图片地址，都是服务器地址" src的地址值不能是本地的地址值

4.css: 美化页面的

5.css中的选择器：

1) 类选择器：.class属性值{}

2) 标签名选择器：标签名{}

3) id选择器：#id属性值{}

6.css样式：

1) display样式：

a:可以将行内标签转换为块级标签：display:block

b:可以将块级标签转换为行内标签：display:inline

c:隐藏某个标签：display:none

2) 盒子模型：

边框 元素 内边距 外边距

7.js核心语法：

1.变量：let 常量 const

2.数据类型：

1) 原始数据类型：number string boolean null undefined

2) 引用数据类型：Date Object

3.运算符：

=== 严格比较 比较数据类型

4.六种假：

false 0 '' null undefined NaN

5.输出语句：

1) 弹出框：alert()

2) 输出页面：document.write()

3) 输出到控制台：console.log()

6.全局函数：

1) parseInt("数据") 字符数据中的数值从前往后解析出整数 parseFloat() 解析小数

2) Number(布尔); true 1 false 0 将布尔类转换为数值

```
3) Boolean(其他类型) 将其他类型转换为布尔 空字符串 ===false 0====false  
null===false...  
idea无法输入中文  
-Drecreate.x11.input.method=true
```

1.JS中的对象(掌握)

1. Array数组对象 (重点)

数组对象是使用单独的变量名来存储一系列的值。

1.1创建一个数组

创建一个数组，有三种方法。

【1】常规方式:

```
let 数组名 = new Array();
```

```
> let arr = new Array();  
< undefined  
✖ Failed to load resource: net::ERR  
> arr.length  
< 0  
> arr[0]=10  
< 10  
> arr[1]=1.2  
< 1.2  
> arr[2]='柳岩'  
< "柳岩"  
> arr  
< ▶ (3) [10, 1.2, "柳岩"]  
>
```

【2】简洁方式: 推荐使用

```
let 数组名 = new Array(数值1,数值2,...);
```

```

> let arr2 = new Array(1.2,10,'杨幂',true);
< undefined
> arr2
< ▶ (4) [1.2, 10, "杨幂", true]
> |

```

【3】 字面:在js中创建数组使用中括号 推荐使用

```
let 数组名 = [数值1,数值2,...];
```

```

> let arr3=[10,1.2,10,false];
< undefined
> arr3
< ▶ (4) [10, 1.2, 10, false]
>

```

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
<script type="text/javascript">
  /*
    创建数组对象方式一:
  */
  let arr = new Array(10, 1.2, true, '锁哥');
  //获取数据1.2
  // console.log(arr[1]); //1 表示数组索引
  //遍历数组
  //length 数组对象中的属性,表示数组长度
  /* for (let i = 0; i < arr.length; i++) {
    //获取数组数据
    let x = arr[i];
    console.log(x);
  }*/

```

//采用方式一即上述方式定义数组 let arr = new Array(10, 1.2, true, '锁哥');注意: 如果只给一个number类型的值, 那么此时数值表示数组长度, 数组中的数据都是empty

```
let arr2 = new Array(5);
```

//报错: Invalid array length 无效的数组长度

//采用方式一即上述方式定义数组 let arr = new Array(10, 1.2, true, '锁哥');注意: 如果只给一个number类型的值,

// 那么此时数值表示数组长度, 要求不能给小数, 数组长度不能是小数

```
// let arr3 = new Array(1.2);
```

```

//采用方式一即上述方式定义数组 let arr = new Array(10, 1.2, true, '锁哥');注意：如果
只给一个非number类型的值，
// 那么此时数值表示数组的元素
let arr4 = new Array('2');

//向arr数组中存false
// arr[4] = false;
// console.log(arr[4]);false

//js中的数组长度是可变的

//向数组中添加数据
// arr[7] = 100;//[10, 1.2, true, '锁哥', empty × 3, 100]

console.log(arr.length);//修改前：长度是4
//修改数组长度
//修改数组长度
arr.length = 2;
console.log(arr.length);//修改前：长度是2,数据是 10 1.2
</script>
</body>
</html>

```

小结：

创建数组有三种方式：

1.常规方式:let arr=new Array();

2.简写方式：let arr=new Array(数据1,数据2。。。);

注意：如果小括号中只有一个**number类型**的值，表示数组长度，不能是小数。

The screenshot shows a browser console with the following interactions:

- Attempt 1:** `let arr4=new Array(1.2);` results in an error: `Uncaught RangeError: Invalid array length at <anonymous>:1:10`. A red arrow points from this error to a text box explaining that a decimal value cannot be used as an array length.
- Attempt 2:** `let arr4=new Array(10);` successfully creates an array of length 10. A red arrow points from the `10` parameter to the same explanatory text box.
- Attempt 3:** `let arr5=new Array('哈哈');` successfully creates an array with one element. A green arrow points from the `'哈哈'` parameter to a text box explaining that a non-number value represents a single data element.
- Attempt 4:** The console shows the structure of `arr5`: `["哈哈"]` with `length: 1` and `__proto__: Array(0)`. A green arrow points from this structure to the same explanatory text box.

Text Box 1 (Red): 如果使用Array方式创建数组，并且创建数组对象时只给一个number类型的数据，此时number类型的数据表示数组长度，不能是小数，小数不能作为数组长度

Text Box 2 (Green): 如果Array的参数是一个不是number类型的数值，此时表示数组中的一个数据，数组长度是1

3.字面：let myCars=[数据1,数据2,数据3,...];

```

> let arr8=[10];
< undefined
> arr8
< ▼ [10] i
    0: 10
    length: 1
    ▶ __proto__: Array(0)
>

```

如果使用[]创建数组。那么这里的10表示数组的数据，数组长度是1

4.js中的数组定义使用中括号，并且可以存储不同类型的数据

1.2数组的特点

1.js中的数组可以存储不同类型的数据

2.js中的数组长度是可变的

```

> let a1=[10,20,1.2,'锁哥']; ➡数组长度是4
< undefined
> a1.length=2 ➡ 修改数组长度变为2，最后数组的数据是：
< 2
> a1
< ▶ (2) [10, 20]
> a1[4]='岩岩' 此时数组索引是0 1，这里是给索引是4的位置添加数据，索引2 3的位置是空
< "岩岩"
> a1
< ▼ (5) [10, 20, empty × 2, "岩岩"] i
    0: 10
    1: 20
    4: "岩岩"
    length: 5
    ▶ __proto__: Array(0)
<

```

3.可以使用数组名[索引]操作数组的数据

4.可以使用循环遍历数组

```

> for(let i=0;i<a1.length;i++){console.log(a1[i]);}
10
20
2 undefined
岩岩

```

1.3数组中的函数

方法名	功能
concat()	用于拼接一个或多个数组
reverse()	用于数组的反转
join(separator)	用于将整个数组使用分隔符拼接成一个字符串。相当于split()反操作
sort()	对数组进行排序，不带参数是按照编码值进行排序。 如果不是按照编码值排序,那么必须指定比较器。 说明：由于字符串都是按照编码值比较大小的，所以也可以将这个方法这样理解使用： 如果要对数字进行排序，必须指定比较器。如果是按字符串进行排序，那么就是无参。
pop()	删除并返回数组的最后一个元素
push()	添加一个或者多个元素到数组的最后面

【1】代码演示1

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
  <script type="text/javascript">
    /*
      数组中的函数：
      1.concat()用于拼接一个或多个数组
      2.reverse()用于数组的反转
      3.join(separator)用于将整个数组使用分隔符拼接成一个字符串。相当于split()反操作
      4.pop()删除并返回数组的最后一个元素
      5.push()添加一个或者多个元素到数组的最后面
    */
    //1.concat()用于拼接一个或多个数组
    //定义数组
    let a1=[10,20,30];
    let a2=[1.1,200,false];
    let a3=['柳岩',3.14,true];
    //拼接 [10, 20, 30, 1.1, 200, false, "柳岩", 3.14, true]
    let a4 = a1.concat(a2,a3);

    //2.reverse()用于数组的反转
    let a5 = a1.reverse();//[30, 20, 10] 此时a1的数据也是[30, 20, 10]，a1和a5共享一个数空间

    //3.join(separator)用于将整个数组使用分隔符拼接成一个字符串。相当于split()反操作
    let str = a2.join("_");
    console.log(str);//1.1_200_false

    //4.pop()删除并返回数组的最后一个元素
```

```

        console.log(a3.pop());//a3数组的数据是: ["柳岩", 3.14]    pop函数返回的是true

//5.push()添加一个或者多个元素到数组的最后面
a2.push(10,'杨幂');//[1.1, 200, false, 10, "杨幂"]

//splice(index,n) 从数组中删除元素    。index表示从哪个索引删除,n表示删除数据个数

let arr = [10,20,30,40];
arr.splice(1, 2);
console.log(arr);

</script>
</body>
</html>

```

【2】代码演示2:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <script type="text/javascript">
        /*
            6.sort()
                对数组进行排序, 不带参数是按照编码值进行排序。
                如果不是按照编码值排序,那么必须指定比较器。
                说明: 由于字符串都是按照编码值比较大小的, 所以也可以将这个方法这样理解使
用:
                如果要对数字进行排序, 必须指定比较器。如果是按字符串进行排序, 那么就是无
参。

                */
            //定义数组
            let a1=['kaja','aha','abclaja','0aa','abcde','ahd','AKAJA'];
            //使用数组对象中的函数对上述数组进行排序
            a1.sort();//["0aa", "AKAJA", "abcde", "abclaja", "aha", "ahd", "kaja"] 默
认是大小升序, 按照编码值升序排序

            let a2=[10,108,2,9,99,34,345,200];
            /*
                [10, 108, 2, 200, 34, 345, 9, 99] 这里是按照编码值升序排序
            */
            // a2.sort();

            /*
                对上述数组a2进行大小降序排序。
                分析: 这里的需求不是按照编码值排序了, 是按照数值大小排序, 那么我们在sort函数的参数
位置必须传递一个比较器, 即匿名函数
                function(o1,o2){
                    代码
                }
                升序: o1 - o2
                降序: o2 - o1
                最后结果:

```

```
        [345, 200, 108, 99, 34, 10, 9, 2] 数值大小降序
    */
    a2.sort(function (o1,o2) {
        return o2 - o1;
    });

</script>
</body>
</html>
```

小结:

1.

对于排序函数sort如果操作的数组存储的是字符串，那么就是无参的。按照编码值升序排序。

对于排序函数sort如果操作的数组存储的是数值，那么就是有参的。按照大小排序。

```
a2.sort(function (o1,o2) {
    //降序排序: o2 - o1
    return o2 - o1;//[456, 123, 100, 30, 9, 8, 2]
});
```

2. RegExp正则对象（重点）

1.创建正则对象

有两种方式:

【1】方式一

```
let 正则对象 = new RegExp("正则表达式");
```

【2】方式二推荐使用

```
let 正则对象 = /正则表达式/;
说明：这种写法在双斜杠中不用书写引号
```

注意:

1.方式二在//中直接书写正则表达式，不用书写单引号 ^0-9\$

2.在js中由于一些浏览器兼容问题，书写正则符号的时候最好加上边界词:

以什么开始: ^

以什么结尾: \$

2.验证函数

使用RegExp中的test函数，使用格式：

```
let result = 正则对象.test(被验证的字符串);
```

说明：

如果被验证的字符串满足正则表达式则test函数返回true，否则返回false

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
  <script type="text/javascript">
    /*
      需求：使用正则表达式验证指定的字符串是否满足长度是6.
    */
    //1.创建正则对象
    /*
      1.在正则表达式中任意字符使用符号： .
      2.在正则中表示出现的次数：
          {n,m} 最少出现n次，最多出现m次
          {n,} 最少出现n次
          {n} 正好出现n次
      3. .{6} 表示正好出现6位任意字符

      4. ^ 表示以什么开始    $ 表示以什么结尾
    */
    // let reg = new RegExp("^.{6}$");
    let reg = /^.{6}$/;
    //2.使用正则对象调用test函数验证字符串
    let result = reg.test("abj1a9");
    console.log(result);
  </script>
</body>
</html>
```

小结：

1.创建正则表达式对象：

```
1)let reg = new RegExp('正则表达式');
2)let reg = /正则表达式/;
```

注意：别忘记在正则表达式中加上边界词：

^ 表示以什么开始
\$ 表示以什么结尾

2.使用正则调用test函数验证字符串是否满足正则：

```
let result = reg.test(字符串);
```

3.String对象

在js中string属于基本类型(原始数据类型), 然后js将其包装成了引用类型(复合数据类型)。

【1】创建String对象方式 1.let txt = new String("string"); 使用构造方法创建对象 了解 2.let 对象名 = "字符串"; 使用双引号 理解 3.let 对象名 = '字符串'; 使用单引号 掌握 4.let 对象名 = `字符串`; 使用反单引号(键盘上的波浪线, 在esc下面) 掌握 从es6开始的目的是为了简化字符串的拼接

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
  <script type="text/javascript">
    /*
      【1】创建String对象方式:
      1.let txt = new String("string"); 使用构造方法创建对象 了解
      2.let 对象名 = "字符串"; 使用双引号 理解
      3.let 对象名 = '字符串'; 使用单引号 掌握
      4.let 对象名 = `字符串`; 使用反单引号(键盘上的波浪线, 在esc下面) 掌握 从
      es6开始的目的是为了简化字符串的拼接
    */
    //1.let txt = new String("string"); 使用构造方法创建对象 了解
    let s = new String("柳岩");//s保存地址值
    console.log(s.toString());//柳岩

    // 2.let 对象名 = "字符串"; 使用双引号 理解
    let s1 = "杨幂";
    console.log(s1);//杨幂

    //3.let 对象名 = '字符串'; 使用单引号 掌握

    let s2 = '赵丽颖';
    console.log(s2);//赵丽颖

    // 4.let 对象名 = `字符串`; 使用反单引号(键盘上的波浪线, 在esc下面) 掌握 从es6开
    始的目的是为了简化字符串的拼接
    let i = 10;
    let s3 = '哈哈' + i + '呵呵';//+是拼接
    console.log(s3); //哈哈10呵呵
    //在反单引号中${获取数据的变量名}
    let s4=`嘿嘿${i}嘻嘻`; //es6的新语法
    console.log(s4);//嘿嘿10嘻嘻
  </script>
</body>
</html>
```

小结:

创建String对象方法:

1. 双引号
2. 构造方法 了解 `let s1=new String("abc");`
3. 单引号 推荐 `let s2='def';`
4. 反单引号: `let s3 = `efg`;`

注意: 使用反单引号主要目的为了方便字符串的拼接, 省去了字符串+拼接的麻烦

```
let s3 = `efg`;
```

``${s3}锁哥`` 结果是 `efg锁哥`

`${变量}`这种写法只能书写在反单引号中, `${变量}`大括号中的变量不一定是反单引号定义的, 可以是单引号定义的

4.自定义对象

- 格式

```
let 对象名称 = {  
    属性名称1:属性值1,  
    属性名称2:属性值2,  
    ...  
    函数名称:function (形参列表){}  
    ...  
};
```

- 代码实现

```
<!-- 文档类型声明标签, 告知浏览器这个页面采取html版本来显示页面 -->  
<!DOCTYPE html>  
<!-- 告诉浏览器这是一个英文网站, 本页面采取英文显示, 单也可以书写中文 -->  
<html lang="en">  
  
<head>  
    <!-- 必须书写, 告知浏览器以UTF-8编码表编解码中文, 如果不书写就会乱码 -->  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <!-- 标题标签 -->  
    <title>Title</title>  
</head>  
<body>  
<script type="text/javascript">  
    /*  
        自定义对象: *****很重要  
        let 对象名称 = {  
            属性名称1:属性值1,  
            属性名称2:属性值2,  
            ...  
            函数名称:function (形参列表){}  
            ...  
        };  
    */  
    //自定义对象 person表示对象名  
    let person = {  
        username:"柳岩",
```

```

    age: 20,
    // 定义函数
    eat: function(a) { // a=100
        // 在自定义对象的函数中不能直接使用自定义对象中的属性，否则报错：
selfObjectDemo06.html:32 Uncaught ReferenceError: username is not defined
        // console.log(username+"干饭人，干饭魂,a="+a);
        // 使用对象person调用属性名可以获取属性值
        // console.log(person.username+"干饭人，干饭魂,a="+a);
        // 后期经常使用的方式：这里的this表示当前自定义对象person
        console.log(this.username+"干饭人，干饭魂,a="+a);
    }
};
// 如果在自定义对象外部获取自定义对象中的属性值，那么格式：对象名.属性名
// console.log(person.username); // "柳岩"
// console.log(person.age); // 20
// 调用自定义对象中的函数：对象名.函数名(实参);
person.eat(100); // 干饭人，干饭魂,a=100
</script>
</body>
</html>

```

2.BOM (Browser Object Model)

操作浏览器的。常用的浏览器对象：

1.window对象：Window 对象表示浏览器中打开的窗口。

2.location对象：Location 对象包含有关当前 URL 的信息。Location 对象是 window 对象的一部分，可通过 window.Location 属性对其进行访问。

3.history对象：History 对象包含用户（在浏览器窗口中）访问过的 URL。

History 对象是 window 对象的一部分，可通过 window.history 属性对其进行访问。

1.window对象

表示浏览器窗口对象。属于最大的窗口对象，在js中可以使用window对象调用全局函数，属性以及 document history location对象。

1.1方法：消息框

【1】弹出框

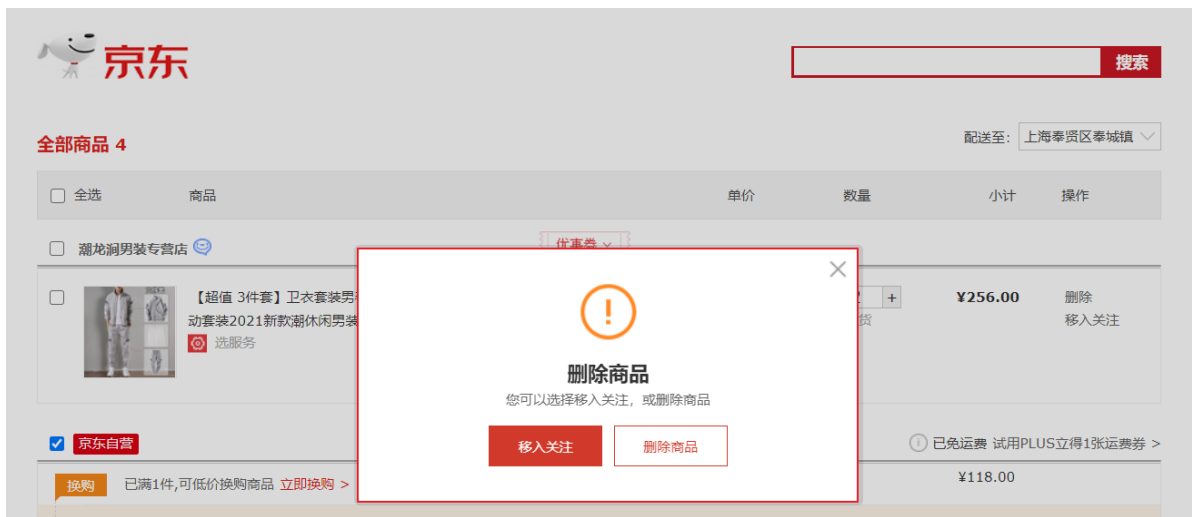
```

window.alert(弹出信息);

```

【2】确认框 掌握

京东购物车删除商品的弹出确认框：



```
let result = window.confirm(确认信息);
```

说明:

- 1) 确认框具有两个按钮: 确定 取消
- 2) 当点击确定按钮, 该函数就会返回true
- 3) 当点击取消按钮, 该函数就会返回false

```
/*
```

【2】确认框

```
let result = window.confirm(确认信息);
```

说明:

- 1) 确认框具有两个按钮: 确定 取消
- 2) 当点击确定按钮, 该函数就会返回true
- 3) 当点击取消按钮, 该函数就会返回false

```
*/
```

```
let result = window.confirm('这么好的商品确认删除吗?');  
console.log(result);
```

localhost:63342 显示

这么好的商品确认删除吗?

确定

取消

【3】信息提示框

```
let result = window.prompt(提示信息, 默认值);
```

说明:

- 1) 信息提示框具有两个按钮: 确定 取消
- 2) 当点击确定按钮, 该函数就会返回信息框的值
- 3) 当点击取消按钮, 该函数就会返回null

/*

【3】信息提示框

```
let result = window.prompt(提示信息,默认值);
```

说明:

- 1) 信息提示框具有两个按钮: 确定 取消
- 2) 当点击确定按钮, 该函数就会返回信息框的值
- 3) 当点击取消按钮, 该函数就会返回null

*/

```
let result = window.prompt('请输入您的姓名', '柳岩');  
console.log(result);
```

localhost:63342 显示

请输入您的姓名

柳岩

确定

取消

1.2定时器

- 定时器setInterval

`let timer = window.setInterval(code, millisec)` 按照指定的周期(间隔)来执行函数或代码片段。

参数1: `code` 必须写。 执行的函数名或执行的代码字符串。

参数2: `millisec` 必须写。时间间隔, 单位: 毫秒。

`window`可以理解为浏览器窗口。后面会讲解。

`timer` 返回值: 一个可以传递给 `window.clearInterval`(定时器的返回值) 从而取消对 `code` 的周期性执行的值。

在关闭定时器时需要使用定时器的返回值作为参数, 否则不知道关闭哪个定时器。

例如:

方式: 函数名 , `setInterval(show, 100);` ✓ `show` 表示函数名, 100表示每隔100毫秒执行这个函数。

案例:

需求: 开启一个定时器, 每隔1秒钟输出一 hello world。

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <meta charset="UTF-8">  
  <title>Title</title>
```

```

</head>
<body>
  <script type="text/javascript">
    /*
      1. 定时器
      let 返回值 = window.setInterval(调用函数,时间间隔属于毫秒);
      2. 取消定时器
      window.clearInterval(定时器返回值);
    */
    //需求: 开启一个定时器, 每隔1秒钟输出一 次 hello world。
    /*
      第一个参数表示匿名函数, 传递给setInterval函数底层, 底层负责调用, 我们只需要书写
      函数体传递即可
      第二个参数: 表示时间间隔单位是毫秒
    */
    /* window.setInterval(function () {
      console.log('hello world');
    }, 1000);*/

    //需求: 定时器执行一次hello world就取消定时器
    let timer = window.setInterval(function () {
      console.log('hello world');
      // 取消定时器 timer 表示上述定时器的返回值, 执行一次匿名函数体代码之后再执行该代
      码进行取消定时器
      window.clearInterval(timer);
    }, 1000);
    //取消定时器 timer 表示上述定时器的返回值
    //window.clearInterval(timer);
  </script>
</body>
</html>

```

- 定时器setTimeout

window.setTimeout(code,millisec)

参数:

code 必需。要调用的函数。

millisec 必需。在执行代码前需等待的毫秒数。

注意:

该定时器只执行一次

需求: 开启一个定时器 1秒之后输出Hello world, 并且只输出一次。

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
  <script type="text/javascript">
    /*
      需求: 开启一个定时器 1秒之后输出Hello world, 并且只输出一次。
    */
    //window.setTimeout(code,millisec)
    window.setTimeout(function () {

```

```

        console.log('hello world');
    }, 1000);
</script>
</body>
</html>

```

小结:

定时器setTimeout(匿名函数,毫秒)页面加载完成过指定的毫秒之后在执行, 并且只执行一次.

setInterval()定时器, 如果不取消就会一直执行

1.3案例_定时切换图片

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>JavaScript演示</title>
</head>
<body>

<input type="button" onclick="on()" value="开灯">

<input type="button" onclick="off()" value="关灯">

<script>

    function on(){
        document.getElementById('myImage').src='../imgs/on.gif';
    }

    function off(){
        document.getElementById('myImage').src='../imgs/off.gif'
    }
    var x = 0;

    // 根据一个变化的数字, 产生固定个数的值:  2   x % 2      3   x % 3
    //定时器
    setInterval(function (){

        if(x % 2 == 0){
            on();
        }else {
            off();
        }

        x ++;

    },1000);

</script>

</body>

```



```
</html>
```

2 Location对象(理解)

- 1.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。
- 2.对于location对象的属性说明：

Location 对象属性	
属性	描述
hash	设置或返回从井号 (#) 开始的 URL (锚)。
host	设置或返回主机名和当前 URL 的端口号。
hostname	设置或返回当前 URL 的主机名。
href	设置或返回完整的 URL。
pathname	设置或返回当前 URL 的路径部分。
port	设置或返回当前 URL 的端口号。
protocol	设置或返回当前 URL 的协议。
search	设置或返回从问号 (?) 开始的 URL (查询部分)。

例如URL: http://127.0.0.1:8020/day03/定时弹广告/05.location.html?__hbt=150384448335#abc

属性	对应的值
hash:	#abc
host:	127.0.0.1:8020
hostname:	127.0.0.1
href:	http://127.0.0.1:8020/day03/定时弹广告/05.location.html?__hbt=1503844483351#abc ,同时也可以跳转到新的url
pathname:	/day03/定时弹广告/05.location.html
port:	8020
protocol:	http:
search:	?__hbt=150384448335

需求：设置location的href值来实现窗口的跳转。

就是在页面加载的时候直接访问<http://www.baidu.com>

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
  <script type="text/javascript">
    /*
      需求：设置location的href值来实现窗口的跳转。
      就是在页面加载的时候直接访问http://www.baidu.com
    */
    window.location.href = 'http://www.baidu.com';

  </script>
</body>
</html>
```

小结：

location对象表示浏览器url地址，有一个比较重要的属性是href，可以实现跳转到其他服务器。

3.History对象（理解）



1.window.history 对象包含浏览器的历史。表示浏览器的历史记录对象。

2.**window.history**对象在编写时可不使用 window 这个前缀。就是我们在使用的时候可以不写window对象

3.History对象常见的函数：

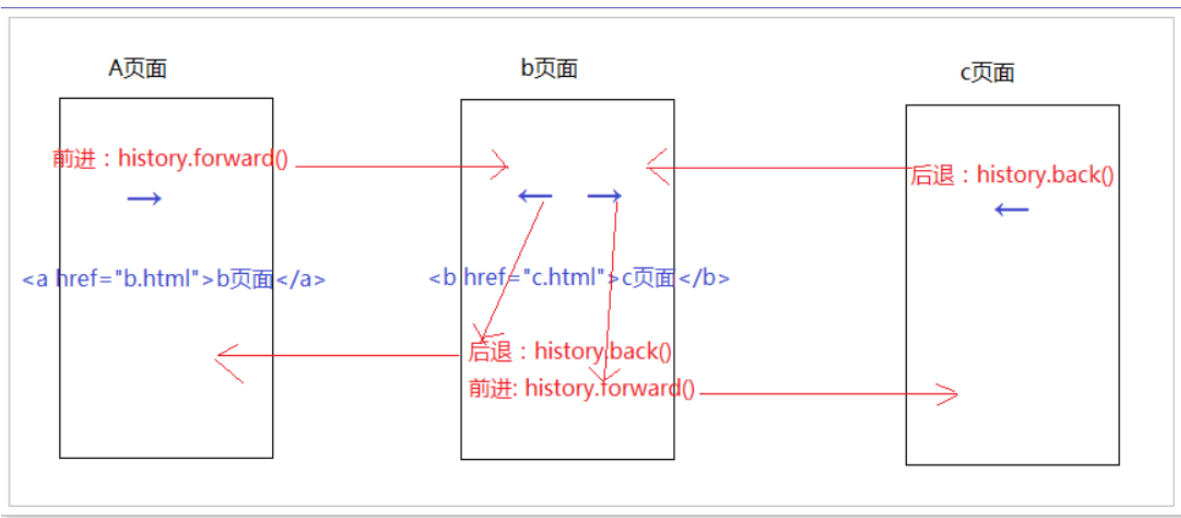
- 1.**history.back()** - 与在浏览器点击后退按钮相同 后退一步
- 2.**history.forward()** - 与在浏览器中点击按钮向前相同 前进一步
- 3.**history.go(整数)** 如果参数是正整数那么就是前进，如果参数是负整数就是后退。

举例：

- 1 前进一步 2 前进2步。。。。
- 1 后退一步 -2 后退2步。。。

4.代码实现：

【1】需求：



【2】技术点

点击事件: onclick,使用格式:

```

标签对象.onclick = function(){
    js代码
}
  
```

【3】步骤

- 1.创建三个html页面
- 2.在a页面中书写a标签跳转到b页面，同时定义一个按钮，给按钮绑定单击的js事件，实现前进
- 3.在b页面中书写a标签跳转到c页面，同时定义两个按钮，给按钮绑定单击的js事件，实现后退和前进

4.在c页面定义一个按钮，给按钮绑定单击的js事件，实现后退

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>a页面</title>
</head>
<body>
  <!--2.在a页面中书写a标签跳转到b页面，同时定义一个按钮，给按钮绑定单击的js事件，实现前进-->
  <a href="b.html">到b页面</a>
  <button id="btn">-></button>
  <script type="text/javascript">
    //1.获取上述标签对象
    let btn = document.getElementById('btn');
    //2.使用标签对象调用单击事件
    btn.onclick = function () {
      //前进到b页面
      // history.forward();
      history.go(2); //前进2步
    };
  </script>
</body>
</html>

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>b页面</title>
</head>
<body>
  <!--3.在b页面中书写a标签跳转到c页面，同时定义两个按钮，给按钮绑定单击的js事件，实现后退和前进-->
  <a href="c.html">到c页面</a>
  <button id="btn1"><-</button>
  <button id="btn2">-></button>
  <script type="text/javascript">
    //1.实现后退
    document.getElementById('btn1').onclick = function () {
      history.back();
    };
    //2.实现前进
    document.getElementById('btn2').onclick = function () {
      history.forward();
    }
  </script>
</body>
</html>

<!DOCTYPE html>
<html lang="en">
```

```

<head>
  <meta charset="UTF-8">
  <title>c页面</title>
</head>
<body>
  <!--4.在c页面定义一个按钮，给按钮绑定单击的js事件，实现后退-->
  <button id="btn"><--</button>
  <script type="text/javascript">
    //实现后退
    document.getElementById('btn').onclick = function () {
      history.back();
    };
  </script>

</body>
</html>

```

小结：

1.history对象表示浏览的历史记录对象

2.history对象中的常见函数：

1. back() 后退一步
2. forward() 前进一步
3. go(参数) 正数 前进 负数 后退

3.给某个标签绑定点击事件：

```

标签对象.onclick=function(){

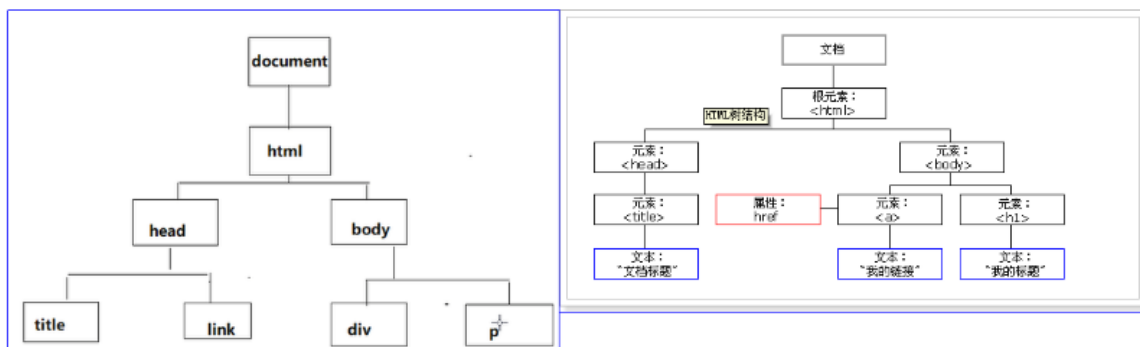
}

```

3.DOM概述 理解

DOM:document object model 文档对象模型。用来操作html页面中所有html标签的。在使用dom操作html标签之前，浏览器会将html页面中的标签加载到内存中形成一个dom树，上面最大的对象是document。我们可以通过document调用属性或者函数获取html标签。

html文件对应的DOM树内存图：



4.DOM对象中的属性、方法介绍

1标签属性和文本的操作：(了解)

属性名	描述
element.getAttribute(属性的名称);	根据标签的属性的名字来获取属性的值
element.setAttribute(属性的名称, 值);	给标签设置一个属性
element.removeAttribute(属性的名称);	根据标签的属性的名字来删除属性
element.children;	获取当前元素（标签）的子元素注意：获取的子级（一层），不管嵌套(不包括孙子，只包括儿子)
element.nodeName/tagName;	获取标签名 注意：获取的是大写的字符串的标签名
element.innerHTML;	获取当前元素（标签）的文本内容 哈哈

【1】练习1

```
<input type="text" name="username" id="txt" value="java" />
```

需求1：获取属性 name的值

需求2：给文本框设置一个属性 hello, 值是world

需求3：删除属性 value

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
  <input type="text" name="username" id="txt" value="java" />
  <script type="text/javascript">
    //获取input标签对象
    let oInput = document.getElementById('txt');
    // 需求1：获取属性 name的值
    //name表示input标签的属性名，根据name属性名获取name的属性值  username
    console.log(oInput.getAttribute('name'));
    // 需求2：给文本框设置一个属性 hello, 值是world
    oInput.setAttribute('hello', 'world');
    // 需求3：删除属性 value element.removeAttribute(属性的名称);

  </script>
</body>
</html>
```

【2】

练习二：

```
<div id="div">我是div</div>
```

需求一：获取标签体的内容

需求二：设置标签体的内容：

```
<b>我是加粗的</b>
```

代码示例(标签体的内容)：

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
  <!--练习二：-->

  <div id="div">我是div</div>
  <script type="text/javascript">
    //1.需求一：获取标签体的内容
    //1.1获取标签对象
    let oDiv = document.getElementById('div');
    //1.2获取文本
    console.log(oDiv.innerHTML);
    console.log(oDiv.innerText);
    //需求二：设置标签体的内容：<b>我是加粗的</b>
    //2.1根据标签对象调用属性修改div的文本内容
    // oDiv.innerHTML = '<b>我是加粗的</b>';
    oDiv.innerText = '<b>我是加粗的</b>';
  </script>

</body>
</html>
```

注意：

1.innerHTML和innerText都是操作标签文本的，但是innerHTML解析html标签的，innerText是不解析html标签的，会原样显示。以后使用innerHTML

【3】练习三：

```

<ul id="ul1">
  <li>aaa
    <ul>
      <li>111</li>
      <li>222</li>
      <li>333</li>
    </ul>
  </li>
  <li>bbb</li>
</ul>

```

需求一：获取ID为ul1的所有的子级

需求二：获取ID为ul1下面第一个li的标签名和内容

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
<ul id="ul1">
  <li>aaa
    <ul>
      <li>111</li>
      <li>222</li>
      <li>333</li>
    </ul>
  </li>
  <li>bbb</li>
</ul>
<script type="text/javascript">
  //需求一：获取ID为ul1的所有的子级
  //1.1获取id为ul1的标签
  let oul = document.getElementById('ul1');
  //1.2使用标签对象oul调用属性获取子标签
  let arrLis = oul.children;
  //1.3遍历
  /* for(let i=0;i<arrLis.length;i++){
    console.log(arrLis[i].innerHTML);
  }*/

  //需求二：获取ID为ul1下面第一个li的标签名和内容
  //获取标签名是大写
  console.log(arrLis[0].tagName);//LI
  //文本内容
  console.log(arrLis[0].innerHTML);

</script>
</body>
</html>

```

小结：

1. 标签对象.children 获取指定标签的所有的子标签(儿子)
2. 标签对象.innerHTML : 获取标签的文本。解析html标签的
3. 标签对象.innerHTML = 数据 修改文本值。解析html标签的
4. 获取标签的属性值: 标签对象.getAttribute(属性名);

2 获取标签(元素) 掌握

方法名	描述
document.getElementById(id属性值);	通过元素(标签)的id属性值获取标签对象, 返回的是 单个的标签对象 注意: 只能从document下面去获取
document.getElementsByName(name属性值);	通过元素(标签)的name属性值获取标签对象, 返回的是标签对象的 数组 。 注意: 只能从document下面去获取
document/parentNode.getElementsByTagName(标签名);	通过元素(标签)的名称获取标签的对象, 返回的是标签对象的 数组 。 注意: 可以从document或者某一个父级标签下面去获取
document/parentNode.getElementsByClassName(类名);	通过元素(标签)的class属性获取标签对象, 返回的是标签对象的 数组 。 注意: 可以从document或者某一个父级标签下面去获取

注意: 只有对象才可以调用属性和函数, 数组不能调用。必须将数组中的每个对象取出才可以调用。

【1】练习一:

```
<div class="red">div1</div>
<div>div2</div>
<div class="red">div3</div>

<p>p1</p>
<p class="red">p2</p>
<p>p3</p>

<form action="#">
  用户名: <input type="text" name="username" value="suoge"/>
  密码: <input type="password" name="password" value="123"/>
  性别: <input type="radio" name="gender" value="male" />男
      <input type="radio" name="gender" value="female" />女
      <input type="submit" value="提交" />
</form>
```

需求1: 让页面上所有div字体颜色变蓝色

需求2: 让页面上所有class为red的元素背景色变红色

需求3: 获取页面上name为 username 和 gender的元素, 并输出其value属性的值

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
<div class="red">div1</div>
<div>div2</div>
<div class="red">div3</div>

<p>p1</p>
<p class="red">p2</p>
<p>p3</p>

<form action="#">
  用户名: <input type="text" name="username" value="suoge"/>
  密码: <input type="password" name="password" value="123"/>
  性别: <input type="radio" name="gender" value="male"/>男
        <input type="radio" name="gender" value="female"/>女
        <input type="submit" value="提交"/>
</form>
<script type="text/javascript">
  // 需求1: 让页面上所有div字体颜色变蓝色
  //1.1获取页面中所有的div
  let arrDivs = document.getElementsByTagName('div');
  //1.2遍历数组
  for (let i = 0; i < arrDivs.length; i++) {
    let oDiv = arrDivs[i];
    //使用对象oDiv操作css样式
    oDiv.style.color = 'blue';
  }
  // 需求2: 让页面上所有class为red的元素背景色变红色
  //2.1根据class属性值获取所有class属性值是red的标签
  let arrReds = document.getElementsByClassName('red');
  //2.2遍历数组取出每个标签
  for (let i = 0; i < arrReds.length; i++) {
    let oRed = arrReds[i];
    //2.3使用对象操作css样式
    oRed.style.backgroundColor = 'red';
  }
  // 需求3: 获取页面上name为 username 和 gender的元素, 并输出其value属性的值
  //document.getElementsByName('username')获取的是一个数组, 数组中只有一个标签用户名:
  <input type="text" name="username" value="suoge"/>
  //document.getElementsByName('username')[0] 取出数组索引是0的标签对象
  let oUsername = document.getElementsByName('username')[0];
  //使用标签对象oUsername调用函数获取value属性值
  console.log(oUsername.getAttribute('value'));

  //获取name是gender的元素, 并输出其value属性的值
  console.log(document.getElementsByName('gender')
[0].getAttribute('value'));//male
```

```
console.log(document.getElementsByName('gender')
[1].getAttribute('value')); //female
</script>
</body>
</html>
```

小结:

1.根据标签id属性值获取标签对象:

```
document.getElementById(id名称);
```

2.根据标签名获取标签的数组:

```
let arrDivs = document.getElementsByTagName('标签名');
```

3.根据name属性值获取标签的数组:

```
let arr = document.getElementsByName('name属性值');
```

4.根据class属性值获取的数组:

```
let arrReds = document.getElementsByClassName('class属性值');
```

3新增元素（增）了解

方法名	描述
document.createElement(元素名称)	在页面上根据标签名来创建元素，返回创建的标签（元素）对象注意：只能从document下面去创建元素 <ul style="list-style-type: none">
parentNode.appendChild(要添加的元素对象);	在某一个父级元素（标签）下面去添加一个元素，每次添加到父级元素的 最后面 ，相当于一个追加的操作
parentNode.insertBefore(要添加的元素对象，在谁的前面添加);	在某一个父级元素（标签）下面去添加一个元素，可以指定在哪一个子元素（标签）的前面添加

【1】练习1:

```
<ul>
  <li>bbb</li>
</ul>
```

需求1: 创建一个li添加到ul的最后面

需求2: 再次添加一个li要求添加到ul的第一个位置

```
<!DOCTYPE html>
```

```

<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
<ul>
  <li>bbb</li>
</ul>
<script type="text/javascript">
  // 需求1: 创建一个li添加到ul的最后面
  //1.1创建li标签
  let oLi = document.createElement('li');//<li></li>
  //1.2给新创建的li标签添加文本 <li>ccc</li>
  oLi.innerHTML = 'ccc';
  //1.3获取ul标签对象 document.body 获取body标签
  let oul = document.body.children[0];
  //1.4使用ul标签对象调用函数将创建的li追加到ul的最后的子标签位置
  oul.appendChild(oLi);

  // 需求2: 再次添加一个li要求添加到ul的第一个位置
  //2.1创建li标签
  let oLi2 = document.createElement('li');//<li></li>
  //2.2给新创建的li标签添加文本 <li>aaa</li>
  oLi2.innerHTML = 'aaa';
  //2.3使用父标签对象oul调用函数: parentNode.insertBefore(要添加的元素对象, 在谁的前面添加);将创建的li标签添加到ul的第一个位置
  // oul.children[0] 获取的是此时ul中的第一个子标签<li>bbb</li>
  oul.insertBefore(oLi2, oul.children[0]);

</script>
</body>
</html>

```

小结:

1.创建标签:

```
document.createElement('标签名')
```

2.添加父标签的最后一个孩子位置:

```
父标签对象.appendChild(子标签对象)
```

3.向父标签的指定位置添加:

```
父标签对象.insertBefore(要添加的子标签对象, 添加到哪个子标签前面的对象)
```

4删除元素 (删)

方法名	描述
<code>element.remove();</code> 掌握	删除当前的元素（标签）掌握

【1】练习1:

```
<ul>
    <li>aaa</li>
    <li>bbb</li>
    <li>ccc</li>
</ul>
```

需求：删除第一个li标签。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<ul>
    <li>aaa</li>
    <li>bbb</li>
    <li>ccc</li>
</ul>
<script type="text/javascript">
    //需求：删除第一个li标签。 element.remove();掌握删除当前的元素（标签）掌握
    document.getElementsByTagName('li')[0].remove();
</script>
</body>
</html>
```

小结:

删除：删除标签对象.remove();

5.事件(掌握)

1.概述

js事件可以是浏览器行为，也可以是用户行为。

如果是浏览器行为例如是页面加载完成事件。onload

如果是用户行为就是用户的一切操作都是发生的事件，鼠标操作事件(单击，双击，悬浮等)，键盘操作事件(键盘录入等)。

2.常见事件

事件名	描述
onload	某个页面或图像被完全加载之后触发
onsubmit	当表单提交时触发该事件---注意事件源是表单form
onclick	鼠标点击某个对象
ondblclick	鼠标双击某个对象
onblur	元素失去焦点(输入框)
onfocus	元素获得焦点(输入框)
onchange	用户改变域的内容。一般使用在select标签中。
onkeydown	某个键盘的键被按下
onkeypress	某个键盘的键被按下或按住，一直按住
onkeyup	某个键盘的键被松开
onmousedown	某个鼠标按键被按下
onmouseup	某个鼠标按键被松开
onmouseover	鼠标被移到某元素之上 除了输入框
onmouseout	鼠标从某元素移开 除了输入框
onmousemove	鼠标被移动

3.html标签绑定事件

标签绑定事件：就是让上述的js事件作用在某个标签上

1.静态绑定

就是将事件名作为标签的属性名，在属性值中调用js函数。

需求：

- 1、给div标签绑定单击事件,输出div的文本
- 2、给div标签绑定鼠标悬浮事件，使其背景颜色变红
- 3、给div标签绑定鼠标移出事件，使其背景颜色变蓝

说明：定义一个div，并给div设置宽、高和背景颜色灰色：#ccc。

设置div的代码：

```

<style type="text/css">
    div {
        width: 200px;
        height: 200px;
        background-color: #ccc; /*灰色*/
    }
</style>

```

就是将事件名作为html标签的属性名，在属性值中调用js函数。

```

<标签名 事件名="调用的js函数" 事件名="调用的js函数" ... ></标签名>

```

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style type="text/css">
        div {
            width: 200px;
            height: 200px;
            background-color: #ccc; /*灰色*/
        }
    </style>
</head>
<body>
    <!--静态绑定事件:就是将事件名作为html标签的属性名，在属性值中调用js函数。-->
    <!--
        这里的this表示当前div标签对象
    -->
    <div onclick="fn1(this);" onmouseover="fn2(this);"
onmouseout="fn3(this);">div1</div>
    <script type="text/javascript">
        //需求:
        // 1、给div标签绑定单击事件,输出div的文本
        function fn1(obj) { //obj=this
            //obj接收的是this表示上述的div标签对象
            console.log(obj.innerHTML);
        }
        // 2、给div标签绑定鼠标悬浮onmouseover事件,使其背景颜色变红
        function fn2(obj) { //obj=this
            obj.style.backgroundColor = 'red';
        }
        // 3、给div标签绑定鼠标移出onmouseout事件,使其背景颜色变蓝
        function fn3(obj) {
            obj.style.backgroundColor = 'blue';
        }

    </script>
</body>

```

```
</html>
```

小结:

html静态绑定js事件:

```
<div onclick="fn1(this);" onmouseover="fn2(this);"
onmouseout="fn3(this);">div1</div>
```

2.动态绑定:

就是根据dom技术获取某个标签对象, 使用标签对象调用事件名, 然后将匿名函数赋值事件名。

```
标签对象.事件名 = function(){
}

```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <style type="text/css">
    div {
      width: 200px;
      height: 200px;
      background-color: #ccc; /*灰色*/
    }
  </style>
</head>
<body>
  <!--动态绑定事件:就是根据dom技术获取某个标签对象, 使用标签对象调用事件名, 然后将匿名函数赋值事件名.-->
  <div>div1</div>
  <script type="text/javascript">
    //需求:
    // 1、给div标签绑定单击事件,输出div的文本
    //1.1获取标签对象
    let oDiv = document.getElementsByTagName('div')[0];
    //1.2给div绑定单击事件
    oDiv.onclick = function () {
      //这里的this是调用当前事件onclick的标签对象oDiv
      console.log(this.innerHTML);
    };
    // 2、给div标签绑定鼠标悬浮onmouseover事件, 使其背景颜色变红
    oDiv.onmouseover = function () {
      this.style.backgroundColor = 'red';
    };
    // 3、给div标签绑定鼠标移出onmouseout事件, 使其背景颜色变蓝
    oDiv.onmouseout = function () {
      this.style.backgroundColor = 'blue';
    };
  </script>

```



```
</body>
</html>
```

小结:

1.html绑定js事件有两种方式:

1) 静态绑定: 将事件名作为标签的属性名, 在属性值中调用js函数

```
<div onclick="fn(this);">柳岩</div>
<script type="text/javascript">

    // 1、给div标签绑定单击事件,输出div的文本
    function fn(obj) { //obj=this
        console.log(obj.innerHTML);
    }

</script>
```

2)动态绑定: 使用标签对象调用js事件, 使用匿名函数给其赋值

```
let oDiv = document.getElementsByTagName('div')[0];
//给上述div动态绑定js单击事件
oDiv.onclick = function () {
    //this表示调用当前事件onclick的标签对象oDiv
    // console.log(oDiv.innerHTML);
    console.log(this.innerHTML);
};
```

2.静态绑定和动态绑定区别?

静态绑定缺点: js和html标签耦合在一起

动态绑定: 解耦合。

6.综合案例_会动的时钟(作业:课下完成)

1.目标

2018/1/27 下午2:46:29

开始

暂停

2.分析

- 1.最开始页面不显示时间，有两个按钮 开始 暂停。开始按钮是可以点击的，暂停按钮不能点击
- 2.当点击开始按钮后，设置开始按钮不可用，暂停按钮可用。然后将当前系统时间放到按钮上面。每隔1秒中更新一下页面显示的时间。
- 3.当点击暂停按钮，设置开始按钮可用，暂停按钮不可用。同时停止时间的走动

3.步骤：

- 1.创建一个html页面
- 2.在页面中书写html标签
- 3.获取开始按钮的标签对象，并给绑定单击事件，并绑定匿名函数 ??
- 4.在开始按钮的单击事件绑定的函数中设置开始按钮不可用，暂停按钮可用
- 5.获取当前系统时间 ???
- 6.获取显示时间的标签
- 7.使用显示时间的标签对象调用属性将当前系统时间显示到标签文本中 ??
- 8.开启定时器，每隔1秒更新文本时间
- 9.点击暂停，给暂停按钮绑定单击事件，绑定匿名函数
- 10.在暂停按钮的函数中，设置开始按钮可用，暂停按钮不可用
- 11.停止时间走动，取消定时器

4.代码实现

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
<h1 style="color: darkgreen" id="clock">现在的时间</h1>
<hr/>
<input type="button" value="开始" id="btnStart">
<input type="button" value="暂停" id="btnPause" disabled="disabled">
<script type="text/javascript">
  // 3. 获取开始按钮的标签对象，并给绑定单击事件，并绑定匿名函数 ??
  let timer;
  document.getElementById('btnStart').onclick = function () {
    // 4. 在开始按钮的单击事件绑定的函数中设置设置开始按钮不可用，暂停按钮可用
```

```

    /*
        让开始按钮不可用就是给开始按钮标签的disabled属性值使用js代码设置为true，表示开始
        按钮有disabled即不可用
    */
    document.getElementById('btnStart').disabled = true;
    /*
        让暂停按钮可用就是给暂停按钮标签的disabled属性值使用js代码设置为false，表示暂停按钮
        没有disabled即可用
    */
    document.getElementById('btnPause').disabled = false;
    // 5.获取当前系统时间
    let date = new Date();
    // console.log(date);
    //5.1将系统时间转换为能够看懂的时间
    //toLocaleString() 据本地时间格式，把 Date 对象转换为字符串。
    // console.log(date.toLocaleString());
    let timeDateStr = date.toLocaleString();
    // 6.获取显示时间的标签 <h1 style="color: darkgreen" id="clock">现在的时间
</h1>
    // 7.使用显示时间的标签对象调用属性将当前系统时间显示到标签文本中 ??
    //在js中操作标签的文本使用:标签对象.innerHTML 获取文本值，修改文本值: 标签对
    象.innerHTML=新的值
    document.getElementById('clock').innerHTML = timeDateStr;
    // 8.开启定时器，每隔1秒更新文本时间
    timer = window.setInterval(function () {
        //每隔1秒就将当期系统时间赋值给h1标签文本一次
        //new Date().toLocaleString() 获取当前系统时间并转换为字符串
        document.getElementById('clock').innerHTML = new
Date().toLocaleString();
    }, 1000);
};
// 9.点击暂停，给暂停按钮绑定单击事件，绑定匿名函数
document.getElementById('btnPause').onclick = function () {
    // 10.在暂停按钮的函数中，
    // 10.1设置开始按钮可用
    document.getElementById('btnStart').disabled = false;
    // 10.2暂停按钮不可用
    document.getElementById('btnPause').disabled = true;
    // 11.停止时间走动，取消定时器
    window.clearInterval(timer);//timer表示定时器返回值
};


</script>
</body>
</html>

```

小结:

1.操作某个标签的文本使用: 标签对象.innerHTML

7.完成注册页面的校验(掌握，课下完成)



[购物车](#) | [我的订单](#) | [我的当当](#) | [礼品卡](#) | [手机当当](#) | [企业销售](#) | [帮助](#)

欢迎光临当当网，请[登录](#) [免费注册](#)

新用户注册

企业用户注册

[当](#) [当](#) [首页](#) [注册帮助](#)

贴心提示：请勿设置与邮箱密码相同的账户登录密码或支付密码，防止不法分子窃取您的当当账户信息，[谨防诈骗](#)！

手机号码

123

×

手机格式不正确，请重新输入

登录密码

....

×

密码长度6-20个字符，请重新输入

确认密码


...

×

两次输入的密码不一致，请重新输入

验证码

请输入验证码

 [换张图](#)

☒ 我已阅读并同意 [《当当交易条款》](#) [《当当社区条款》](#) 和 [《当当隐私政策》](#)

立即注册



[购物车](#) | [我的订单](#) | [我的当当](#) | [礼品卡](#) | [手机](#)

欢迎光临当当

新用户注册

企业用户注册

[当](#) [当](#) [首页](#) [注册帮助](#)

贴心提示：请勿设置与邮箱密码相同的账户登录密码或支付密码，防止不法分子窃取您的当当账户信息，[谨防诈骗](#)！

手机号码

13677778888

✓

密码过于简单

登录密码

.....

✓

确认密码

.....

✓

验证码

请输入验证码

 [换张图](#)

☒ 我已阅读并同意 [《当当交易条款》](#) [《当当社区条款》](#) 和 [《当当隐私政策》](#)

立即注册

1.目标

要求：

- 1、用户名必须是6-10位的字母或者数字
- 2、密码长度必须6位任意字符
- 3、两次输入密码要一致

用户名

密码

确认密码

性别 ☐ 男 ☐ 女

爱好 ☐ 唱歌 ☐ 跳舞 ☐ 编程

用户名不能为空
密码不能为空
确认密码不能为空
请选择性别
请选择爱好

说明：只要有一个输入项不满足要求则阻止表单提交。都满足才可以提交表单。

2.实现

1.知识点

1.1js事件

【1】鼠标离开输入框的事件，离焦事件onblur

【2】点击提交按钮执行的提交事件onsubmit。

补充：

1.对于onsubmit事件表示表单提交就执行的事件，如果阻止表单提交，那么该事件绑定的函数返回false，如果提交表单，那么该事件绑定的函数返回true，默认返回的是true。

2.在js中阻止事件都是返回false，不阻止返回true，默认是true。

```
/*
    如果表单的数据有一项是非法的立刻阻止表单提交，全部合法可以提交表单。
    在js中提交表单会执行onsubmit事件，对于该事件，如果阻止表单提交返回false.可以提交表单
    返回true
    默认返回true
*/
//1. 给form表单绑定onsubmit事件
//1.1获取form表单标签对象
let oForm = document.getElementById('myForm');
//1.2使用表单对象oForm调用onsubmit事件
oForm.onsubmit = function () {
    alert(10);
    //默认返回的是true
    //阻止表单提交返回false
    return false;
};
```

小结:

```
<!--注册页面-->
```

```
<!--
```

💡 注意: 如果使用静态绑定, 这里需要在调用函数前面加 `return`, 否则不会起作用

```
-->
```

```
<!--<form action="#" method="post" id="myForm" onsubmit="return false;"-->
```

```
<!--<form action="#" method="post" id="myForm" onsubmit="false;"-->
```

正确的

错误的

2.代码实现

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>欢迎注册</title>
  <link href="css/register.css" rel="stylesheet">
</head>
<body>

<div class="form-div">
  <div class="reg-content">
    <h1>欢迎注册</h1>
    <span>已有帐号? </span> <a href="#">登录</a>
  </div>
  <form id="reg-form" action="#" method="get">

    <table>

      <tr>
        <td>用户名</td>
        <td class="inputs">
          <input name="username" type="text" id="username">
          <br>
          <span id="username_err" class="err_msg" style="display:
none">用户名不太受欢迎</span>
        </td>
      </tr>

      <tr>
        <td>密码</td>
        <td class="inputs">
          <input name="password" type="password" id="password">
          <br>
          <span id="password_err" class="err_msg" style="display:
none">密码格式有误</span>
        </td>
      </tr>

      <tr>
        <td>手机号</td>
        <td class="inputs"><input name="tel" type="text" id="tel">
          <br>
          <span id="tel_err" class="err_msg" style="display: none">手机
号格式有误</span>
```

```

        </td>
    </tr>

</table>

<div class="buttons">
    <input value="注 册" type="submit" id="reg_btn">
</div>
<br class="clear">
</form>

</div>

<script>

    //1. 验证 用户名是否符合规则：长度 6~12,单词字符组成
    function checkUsername(){
        //1.1获取用户名的input输入框的value值
        //表示的是：<input name="username" type="text" id="username"
onblur="checkUsername()">
        let userNameValue = document.getElementById('username').value;
        //1.2创建正则对象
        //\w 单词字符：[a-zA-Z_0-9]
        //在//中书写正则，关于反斜线我们书写一个即可
        // let reg = /^[a-zA-Z_0-9]{6,12}$/;
        let reg = /\w{6,12}$/;
        //1.3判断输入的值是否复合正则
        if(reg.test(userNameValue)){
            //说明输入的内容满足正则,合法
            //用户名合法了，应该隐藏<span id="username_err" class="err_msg"
style="display: none">用户名不太受欢迎</span>
            document.getElementById('username_err').style.display='none';
        }else{
            //说明输入的内容不满足正则，不合法
            /*
                <span id="username_err" class="err_msg" style="display: none">用
用户名不太受欢迎</span>
            说明：display: none 属于css样式，表示隐藏。我们可以通过js代码来操作
css样式：
                element.style  设置或返回元素的 style 属性。
                这里操作的是css中样式名为display的样式，只要我们给其设置值为block
就可以让标签显示
            */
            //获取用户名的span标签并操作css样式，给样式名display的值设置为block就可以显示
用户名名的span标签
            document.getElementById('username_err').style.display='block';
        }
    }

    //1. 验证密码是否符合规则
    //1.1 获取密码的输入框
    var passwordInput = document.getElementById("password");

    //1.2 绑定onblur事件 失去焦点
    passwordInput.onblur = checkPassword;

```

```

function checkPassword() {
    //1.3 获取用户输入的密码
    var password = passwordInput.value.trim();

    //1.4 判断密码是否符合规则：长度 6~12
    var reg = /^w{6,12}$/;
    var flag = reg.test(password);

    //var flag = password.length >= 6 && password.length <= 12;
    if (flag) {
        //符合规则
        document.getElementById("password_err").style.display = 'none';
    } else {
        //不符合规则
        document.getElementById("password_err").style.display = '';
    }

    return flag;
}

```

```

//1. 验证手机号是否符合规则
//1.1 获取手机号的输入框
var telInput = document.getElementById("tel");

//1.2 绑定onblur事件 失去焦点
telInput.onblur = checkTel;

```

```

function checkTel() {
    //1.3 获取用户输入的手机号
    var tel = telInput.value.trim();

    //1.4 判断手机号是否符合规则：长度 11，数字组成，第一位是1

    //var flag = tel.length == 11;
    var reg = /^[1]\d{10}$/;
    var flag = reg.test(tel);
    if (flag) {
        //符合规则
        document.getElementById("tel_err").style.display = 'none';
    } else {
        //不符合规则
        document.getElementById("tel_err").style.display = '';
    }

    return flag;
}

```

```

//1. 获取表单对象
var regForm = document.getElementById("reg-form");

//2. 绑定onsubmit 事件
regForm.onsubmit = function () {
    //挨个判断每一个表单项是否都符合要求，如果有一个不符合，则返回false

    var flag = checkUsername() && checkPassword() && checkTel();
}

```



```

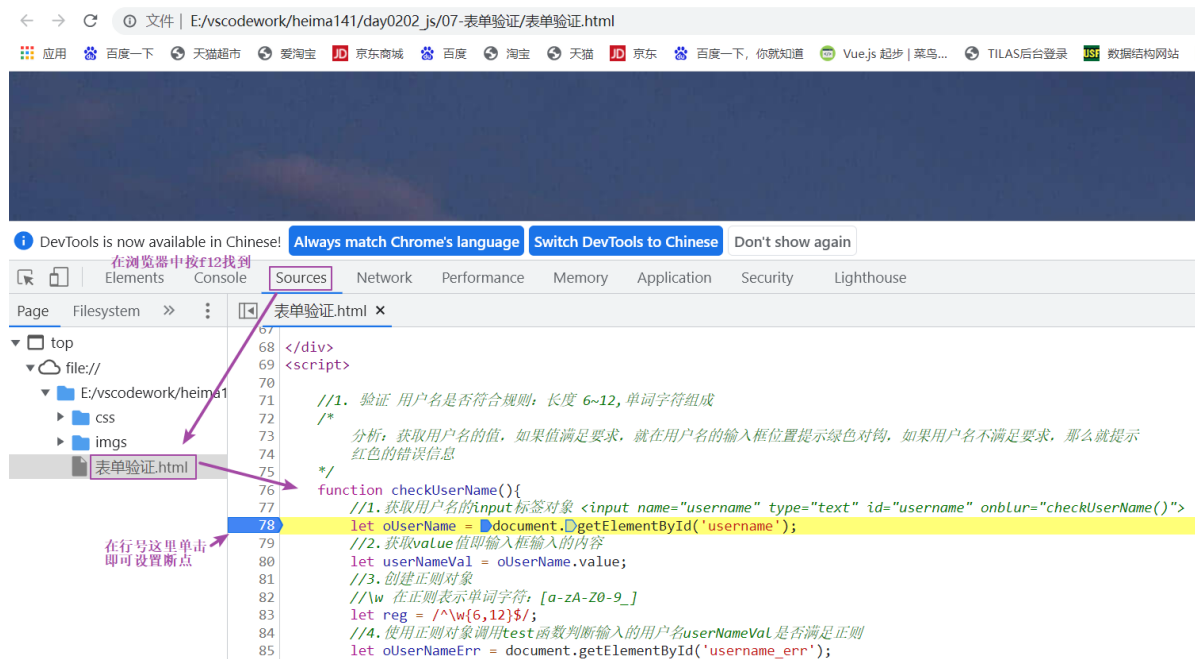
        return flag;
    }

</script>
</body>
</html>

```

8.在浏览器中进行debug讲解(掌握)

1.浏览器加载js代码，在js代码前面点击即可设置断点



2.debug模式按钮讲解

