

## H2 Day04\_JavaScript和JQuery入门

### H3 今日目标

#### 1. BOM(了解)

    window的定时器(掌握)

    window对象及子对象(了解)

#### 2. JQuery简介

#### 3. jquery的使用

#### 4. js对象与jQuery对象转换

#### 5. jquery的页面加载事件

#### 6. jquery的选择器

### H3 1.BOM

BOM:Browser Object Model 浏览器对象模型

Window对象:浏览器窗口对象,该对象下又包含多个子对象

Navigator,Screen,History,Location

Window 对象表示浏览器中打开的窗口

### H4 1.window的常用方法

当调用window方法的时候,该window可以省略

alert(); 警告弹出框(框体上有确认按钮)

confirm();确认弹出框(框体上有两个按钮,确认按钮,取消按钮),当点击确认按钮时候,返回为true,否则为false

prompt();输入确认框(框体上有输入信息提示语,还有一个输入框,确认按钮,当输入内容点击确认后,返回值就是输入框中的内容)



## H4 2.定时器(掌握)

### 1. 周期性定时器(循环定时器)

`setInterval(fn,time);`

`fn`:循环执行的js代码或者函数名称

`time`:周期时间, 单位毫秒

#第一种写法:

```
setInterval(function(){  
    循环执行的js代码    },3000);
```

#第二种写法:

```
function demo(){  
    js代码  
}  
setInterval(demo,3000);
```

#第三种写法:

```
function demo(){  
    js代码  
}  
setInterval("demo()",3000);
```

### 2. 延时定时器(一次性定时器)

`setTimeout(fn,time);`

**fn**:一次性执行的js代码或者函数名称

**time**:延时时间, 单位毫秒

**3.clearInterval(定时器)**;清除周期性定时器

**4.clearTimeout(定时器)**; 清除延时定时器

轮播图示例:

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <meta charset="utf-8">
```

```
    <title></title>
```

```
    <script type="text/javascript">
```

```
      var i = 2;
```

```
      setInterval(function(){
```

```
        var imgObj =
```

```
document.getElementsByTagName("img")[0];
```

```
        imgObj.src="img/"+i+".jpg";
```

```
        i++;
```

```
        if(i==6){
```

```
          i=1;
```

```
        }
```

```
      },3000)
```

```
    </script>
```

```
  </head>
```

```
  <body>
```

```
    <div>
```

```
      
```

```
    </div>
```

```
  </body>
```

```
</html>
```

时间的实时变化

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title></title>
    <script type="text/javascript">
      window.onload=function(){
        setInterval(function(){
          //获取当前系统时间
          var date = new Date();
          var d =
date.toLocaleTimeString();

document.getElementsByTagName("div")
[0].innerHTML="当前时间为:"+d;
          },1000)
        }
      </script>
    </head>
    <body>
      <div></div>
    </body>
  </html>
```

显示隐藏图片

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title></title>
    <style type="text/css">
      img{
        display: none;
      }
    </style>
  </head>
  <body>
    <img alt="隐藏的图片" data-bbox="134 710 291 731" />
  </body>
</html>
```

```

    }
</style>
<script type="text/javascript">
    window.onload=function(){
        //5秒之后， 图片显示
        setTimeout(function(){
            var imgObj =
document.getElementsByTagName("img")[0];
            imgObj.style.display="block";
        },5000);
    }
</script>
</head>
<body>
    
</body>
</html>

```

## 清除定时器

```

<script type="text/javascript">
    var task; 设置成员变量
    window.onload=function(){
        task=setInterval(function(){
            //获取当前系统时间
            var date = new Date();
            var d = date.toLocaleTimeString();
            document.getElementsByTagName("div")[0].innerHTML="当前时间为:"+d;
        },1000);
    }
    function stopTime(){
        clearInterval(task);
    }
</script>
</head>
<body>
    <div></div>
    <input type="button" value="暂停" id="btn" onclick="stopTime()" />
</body>
</html>

```

## H4 3.location(了解)

获取或者改变当前页面的url的路径

href属性

location.href:获取当前页面的url路径

location.href="xxx";修改当前页面的url路径-->跳转到这个路径的对应的页面

## H4 4.history(了解)

在浏览器窗口中访问过的 URL。

forward();前进

back();后退

## H4 5.screen

Screen 对象包含有关客户端显示屏幕的信息。

### Screen 对象属性

属性	描述
<a href="#">availHeight</a>	返回显示屏幕的高度（除 Windows 任务栏之外）。
<a href="#">availWidth</a>	返回显示屏幕的宽度（除 Windows 任务栏之外）。
<a href="#">bufferDepth</a>	设置或返回调色板的比特深度。
<a href="#">colorDepth</a>	返回目标设备或缓冲器上的调色板的比特深度。
<a href="#">deviceXDPI</a>	返回显示屏幕的每英寸水平点数。
<a href="#">deviceYDPI</a>	返回显示屏幕的每英寸垂直点数。
<a href="#">fontSmoothingEnabled</a>	返回用户是否在显示控制面板中启用了字体平滑。
<a href="#">height</a>	返回显示屏幕的高度。
<a href="#">logicalXDPI</a>	返回显示屏幕每英寸的水平方向的常规点数。
<a href="#">logicalYDPI</a>	返回显示屏幕每英寸的垂直方向的常规点数。
<a href="#">pixelDepth</a>	返回显示屏幕的颜色分辨率（比特每像素）。
<a href="#">updateInterval</a>	设置或返回屏幕的刷新率。
<a href="#">width</a>	返回显示器屏幕的宽度。

## H4 6.navigator

Navigator 对象包含有关浏览器的信息

## Navigator 对象属性

属性	描述
<a href="#">appName</a>	返回浏览器的代码名。
<a href="#">appMinorVersion</a>	返回浏览器的次级版本。
<a href="#">appName</a>	返回浏览器的名称。
<a href="#">appVersion</a>	返回浏览器的平台和版本信息。
<a href="#">browserLanguage</a>	返回当前浏览器的语言。
<a href="#">cookieEnabled</a>	返回指明浏览器中是否启用 <code>cookie</code> 的布尔值。
<a href="#">cpuClass</a>	返回浏览器系统的 <code>CPU</code> 等级。
<a href="#">onLine</a>	返回指明系统是否处于脱机模式的布尔值。
<a href="#">platform</a>	返回运行浏览器的操作系统平台。
<a href="#">systemLanguage</a>	返回 <code>OS</code> 使用的默认语言。
<a href="#">userAgent</a>	返回由客户机发送服务器的 <code>user-agent</code> 头部的值。
<a href="#">userLanguage</a>	返回 <code>OS</code> 的自然语言设置。

### H3 2.JQuery

#### H4 1.JQuery简介

是一个轻量级的js框架,这个框架是有JavaScript封装而来,jquery使用起来比较灵活,特点: write less,do more,写很少代码,做很多事情。

jQuery中的我们要学习的主要是两部分 选择器+方法  
jQuery中的选择器就是参考css中选择器,并进行扩展

#### H4 2.JQuery的使用

## 1. 引入jQuery已经封装js文件

```
<script type="text/javascript" src="js文件的引入路径"></script>
```

## 2. 可以使用js文件的方法或者选择器

##官网

<https://jquery.com>

##下载地址

<https://jquery.com/download/>

jquery-3.5.1.min.js 压缩版 ,体积小,加载快,但是代码没有格式化

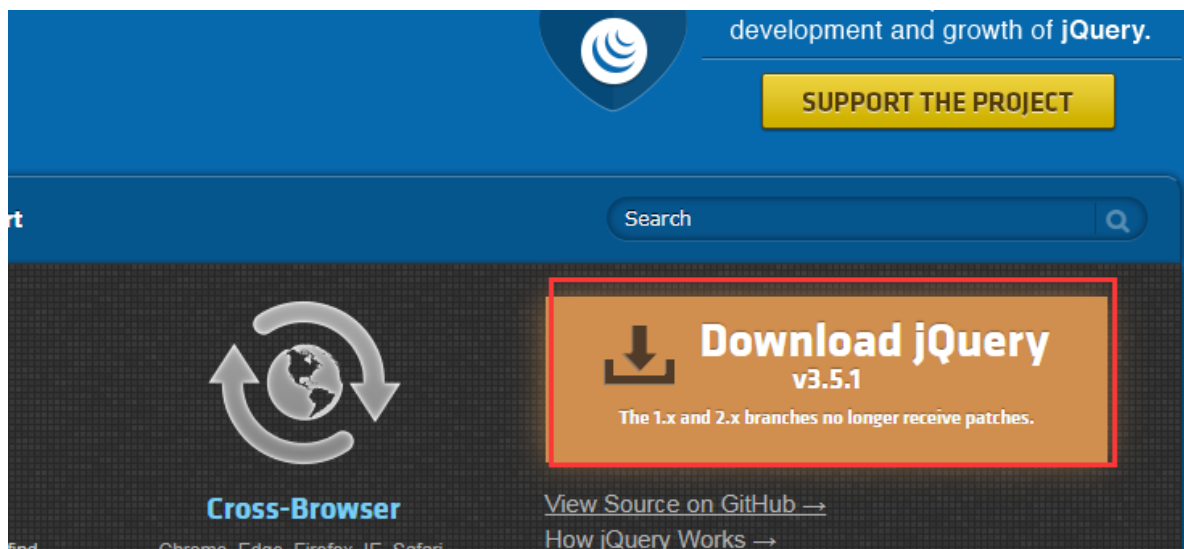
jquery-3.5.1.js 标准版 体积大,加载慢,但是代码格式化

<!--使用方式如下:-->

```
<script src="js/jquery-3.4.1.min.js"
```

```
type="text/javascript"></script>##注意,该标签只能用来引入js文件,需要写js代码另外新建一个script标签
```

```
<script type="text/javascript"></script>
```



To locally download these files, right-click the link and select "Save as..." from the menu.

### jQuery

For help when upgrading jQuery, please see the [upgrade guide](#) most relevant to your version. We also recommend using the [jQuery Migrate plugin](#).

[Download the compressed, production jQuery 3.5.1](#)

压缩版

[Download the uncompressed, development jQuery 3.5.1](#)

标准版

两者任选一个

## H4 3.Js和JQuery对象

##js对象:通过未封装的方式获取的标签对象称为js对象



比如：

```
document.getElementById();
```

##jquery对象:通过jQuery代码或者已经封装的代码获取的标签对象称为jQuery对象(大部分都是通过jQuery选择器获取的标签对象)

##为什么会区分这两对象？

因为js中的方法和jQuery的方法不一样。

也就是说js对象只能使用js中的方法和属性;jQuery对象只能使用jQuery中的属性和方法

##两个对象之间可以相互转化？ 答案是肯定的

js对象--->jQuery对象

```
var obj = document.getElementById();
```

```
$(obj)
```

-----

Jquery对象--->Js对象

```
$obj jquery对象
```

```
$obj.get(index)或者$obj[index]
```

##jQuery对象使用jQuery方法返回的还是一个jQuery对象

## H4 4.Jquery页面加载

ready()事件和onload事件的区别？

1.read()事件在页面的结构加载完成后执行，onload事件在页面的结构及内容都加载完成后执行,因此执行时机read()更快一点

2.read()事件可以执行多次,而onload只能执行一次

3.使用方式也不一样

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

```
}
```

-----

```
$(document).ready(function(){
```

```
});
```

==>简化

```
$(function(){
```

```
});#####在写jQuery代码之前一定要加这个事件
```

## H4 5.选择器

作用：查找对应的标签对象

选择器的基本格式写法：

```
$("选择器")
```

3个JQuery的属性

html()====>js中innerHTML

text()====>js中的innerText

val()====>js中的value

## H5 1.基本选择器

1.id选择器

```
<div id="d1"></div>
```

```
$("#d1")
```

2.class类选择器

```
$(".c1")
```

```
<div class="c1"></div>
```

```
<p class="c1"></p>
```

3.标签选择器

```
$("div")
```

```
<div></div>
```

```
<div></div>
```

4.混合选择器

```
$(".c1,p")
```

```
<div class="c1"></div>
```

```
<p></p>
```

```
<div></div>
```

## 5. 通配选择器

`$("*")`

```
<script type="text/javascript">
  $(function(){
    //点击按钮获取第一个div的标签内容
    $("#btn").click(function(){
      console.log($("#d1").html());
    });
  });
</script>
</head>
<body>
  <div id="d1">
    <a href="">超链接</a>
  </div>
  <p>段落标签</p>
  <div>div标签</div>
  <input type="button" id="btn" value="基本选择器"/>
</body>
```

```
    //第二个div和p的标签内容
    $arr = $(".c1"); //获取对应的标签对象
    for( i in $arr){
      console.log($arr[i].innerHTML);
    }
  });
});
</script>
</head>
<body>
  <div id="d1">
    <a href="">超链接</a>
  </div>
  <p class="c1">段落标签</p>
  <div class="c1">div标签</div>
  <input type="button" id="btn" value="基本选择器"/>
```

```

//获取div的纯文本内容
$divObj = $("div"); 标签选择器
for( i in $divObj){
    console.log($divObj[i].innerText);
}

});
});
</script>
</head>
<body>
    <div id="d1">
        <a href="">超链接</a>
    </div>
    <p class="c1">段落标签</p> 运行的结果
    <div class="c1">div标签</div>
    <input type="button" id="btn" value="基本选择器"/>
</body>

```

## H5 2.层级选择器

**ancestor descendant** 在给定的祖先元素下匹配所有的后代元素

**parent > child** 匹配父标签下指定的子标签

**prev + next** 匹配所有紧接在 **prev** 元素后的 **next** 元素

**prev ~ siblings** 匹配 **prev** 元素之后的所有 **siblings** 元素

```


<script type="text/javascript">
    $(function(){
        //获取$("form input")的值
        $arr = $("form input");
        for(var i = 0;i<$arr.length;i++){
            console.log($($arr[i]).val());
        }
    })
</script>
</head>
<body>
    <form>
        <label>Name:</label>
        <input name="name" value="name1"/>
        <fieldset>
            <label>Newsletter:</label>
            <input name="newsletter" value="newsletter" />
        </fieldset>
    </form>
    <input name="none" value="none" />

```

```

        console.log($(".form > input").val());
    })
</script>
</head>
<body>
    <form>
        <label>Name:</label>
        <input name="name" value="name1"/>
        <fieldset>
            <label>Newsletter:</label>
            <input name="newsletter" value="newsletter" />
        </fieldset>
    </form>
    <input name="none" value="none" />

```



```

$arr =$("label + input");
for(var i = 0;i<$arr.length;i++){
    console.log($($arr[i]).val());
}
})
</script>
</head>
<body>
    <form>
        <label>Name:</label>
        <input name="name" value="name1"/>
        <fieldset>
            <label>Newsletter:</label>
            <input name="newsletter" value="newsletter" />
        </fieldset>
    </form>
    <input name="none" value="none" />

```

```

    console.log($("form ~ input").val());
})
</script>
</head>
<body>
    <form>
        <label>Name:</label>
        <input name="name" value="name1"/>
        <fieldset>
            <label>Newsletter:</label>
            <input name="newsletter" value="newsletter" />
        </fieldset>
    </form>
    <input name="none" value="none" />
</body>

```