11.1 今日内容介绍

11.2 [重点]获取标签元素

- 获取：

  > 内置对象：document
  >
  > 方法: getElementById("标签的id名")

- 使用方法:

  方法1:

  ```
  <body>
  <div id="div1">这是一个div标签</div>

  <script>
     var oDiv = document.getElementById("div1");
     alert(oDiv);
  </script>
  </body>
  ```

  方法2:

  ```
  <head>
  <script>

     function fnFoo(){
         var oDiv = document.getElementById("div1");
         alert(oDiv);
     }

     window.onload = fnFoo;
  </script>
  </head>

  <body>
     <div id="div1">这是一个div标签</div>
  </body>
  ```

  简写:

```
<head>
<script>
     window.onload = function(){
         var oDiv = document.getElementById("div1");
         alert(oDiv);
     }
 </script>
</head>

<body>
     <div id="div1">这是一个div标签</div>
</body>
```

11.3 [重点]操作标签元素属性

- 属性的操作:
  - 属性的读取: 标签对象.属性名
  - 属性的设置: 标签对象.属性名=值
    - 标签属性class 对应 对象.className 属性
    - 标签属性style中属性font-size 对应 对象.style.fontSize属性

```
<script>

    // js获取标签的入口代码(必须写在匿名函数中):
    window.onload = function(){
        var oInput = document.getElementById("input1");
        var oA = document.getElementById("link01");

        // 1.获取标签属性: 对象名.属性名
        alert(oInput.name);
        alert(oA.className);

        // 2. 设置标签属性: 对象名.属性名

        // a的属性class, 通过对象访问时, 使用名字className;
        oA.className = "sty02";
        // a的样式修改 font-size, 对象.style.fontSize;
        oA.style.fontSize = "50px";
    }

</script>
```

- 读取或者设置标签包裹的内容:
  - innerHTML
  - ```
    <script>

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

```
        var oDiv = document.getElementById("div1");

        // 1.获取标签承载的内容
        // innerHTML: 表示标签中承载的内容;
        alert(oDiv.innerHTML);

        // 2.设置标签承载的内容
        // oDiv.innerHTML = "<a href='https://www.baidu.com'>百度网</a>";
        oDiv.onclick = function(){
            oDiv.innerHTML  = "<a href='https://www.baidu.com'>百度</a>";
        }
    }
</script>
```

11.4 [重点]数组及操作方法

- 介绍: 一组数据的集合(相当于python的列表)
- 定义:

```
// 定义数组:
// 1. 字面量定义 (类似python列表)
var aList = [1,2,3,'abc'];
alert(aList);

// 2. 实例化对象定义
var aList2 = new Array(1,2,3, 'abc');
alert(aList2);
```

- 多维数组:

```
// 多维数组定义
var aList2 = [[1,2,3],['a','b','c']]
alert(aList2);
alert(aList2[1][2]);
```

- 数组的操作:
  - length
  - [0]
  - push
  - pop
  - splice

```
// 操作数组:

// length: 数组长度
// alert(aList.length);

// [0] : 访问索引0的元素
// alert(aList[0]);
```

```
// push : 插入数据到数组尾部
// aList.push(5);
// alert(aList);

// pop : 从尾部删除数据
// var tmp = aList.pop()
// alert(aList);
// alert(tmp);

// splice(需要删除的起始索引, 删除的个数(可以写0), [插入的新的元素])
// 1.删除
aList.splice(2, 2);
alert(aList);

// 2.插入
aList.splice(1, 0, 100, 200);
alert(aList);

// 3.替换
aList.splice(1,2,1000,2000);
alert(aList);
```

11.5 [重点]循环语句

- 介绍：重复执行一部分代码
- 常用的循环语句：
  - for

```
// 区别: while和for 都是先判断条件再执行.
//       do{}while 先执行一次再判断条件;
// 循环语句
var aList = [1,2,3,4];


// 2.for(var i=0; i < 10; i++){...}
for(var i = 0; i < aList.length; i++){
    // alert(aList[i]);
    console.log(aList[i]);
}
```

  - while

```
// 1.while(){...}
var i = 0;
while(i < aList.length){
    console.log(aList[i]);
    i++;
}
```

  - do-while

```
// 3.do{}while();
var i = 0;
do{
    console.log(aList[i]);
    i++;
}while(i < aList.length);
```

11.6 [重点]字符串及操作方法

- 字符串拼接: `+`

```
// 字符串拼接: +
// 任何类型和字符串 + 操作，其他类型都会转换成字符串类型再拼接.

var iNum = 100;
var iNum2 = 200;

alert(iNum + iNum2);

var sNum = "1000";
alert(iNum + sNum);
```

- 注意:

> 数字和字符串拼接会自动进行类型转换(隐式类型转换)，把数字类型转成字符串类型进行拼接

11.7 [重点]定时器

- 介绍:

> 在一段特定的时间后执行某段程序代码。

- 定时器的使用:
  - setTimeout

```
// 1.只执行一次的定时器
// setTimeout(超时处理函数，延迟执行的时间-单位是ms，处理函数的参数...)
function fnFoo(name){
    alert(name + "您的余额不足...")
}

// var i = setTimeout(fnFoo, 1000, "小明");
// alert(i);
```

  - setInterval

```
// 2.反复执行的定时器
// setInterval(超时处理函数，延迟执行的时间-单位是ms，处理函数的参数...)
var i = setInterval(fnFoo, 2000, "小明");
alert(i);
```

- 清除定时器:
  - clearTimeout

```
function fnFoo(){
    alert("小广告来了....");
}

var i = setTimeout(fnFoo, 2000);
clearTimeout(i);
```

  - clearInterval

```
var i = setInterval(fnFoo, 2000);

function stop(){
    clearInterval(i);
}

<!-- onclick: 按钮点击事件, 这里后面需要执行的js代码; -->
<input type="button" value="充值按钮" onclick="stop();">
```

11.8 [重点]了解jQuery

- 定义：是javascript的函数库.
- 作用：实现用户和页面交互效果
- 优点:
  - 简化javascript代码开发
  - 兼容主流浏览器

11.9 [重点]jQuery用法和入口函数

- 引入:

```
<!-- 使用外链式 导入js -->
<script src="js/jquery-1.12.4.min.js"></script>
```

- 入口函数:
  - 写法1:

```
$(document).ready(function(){
    // 获取标签对象
    var $div = $("#div1");
    // [object Object]
    alert("jquery获取标签对象:" + $div);
})
```

  - 写法2:

```
$(function(){
    // 获取标签对象
    var $div = $("#div1");
    // [object Object]
    alert("jquery获取标签对象:" + $div);
})
```

11.10 [重点]jQuery选择器

- 介绍:

> 选中标签对象, 设置样式和属性.

- 选择器的种类:
  - 标签选择器: `$('div')`
  - 类选择器: `$('.div1')`
  - id选择器: `$('#id1')`
  - 层级选择器: `$('div p')`
  - 属性选择器: `$("input[name='username']")`

```
// jquery选择器 选中标签 设置样式属性...; 选择规则和css的选择器规则一样;
$(function(){
    // 标签选择器
    var $div = $("div");
    alert($div.length);

    // 类选择器
    var $div2 = $(".div1");
    alert($div2.length);


    // id选择器
    var $div3 = $("#id1");
    alert($div3.length);


    // 层级选择器
    var $div4 = $("div p");
    alert($div4.length);


    // 属性选择器 : 选择属性name='username'的input标签;
    var $input = $("input[name='username']");
    alert($input.length);

})
```

11.11 [重点]选择集过滤

- 介绍:

> 选择标签的集合里面过滤自己需要的标签

- 操作:
  - has方法：选取子标签中包含指定选择器的父标签

```
<script src="js/jquery-1.12.4.min.js"></script>

<script>

    $(function(){
        var $divs = $("div");
        alert($divs.length);

        // 选择集过滤:
        // 1.has : 选择 子标签中 包含指定 选择器的 父标签
        var $div1 = $divs.has("#mytext");
        // jquery标签对象.css : 设置标签样式;
        //  css({"样式属性名": "值"})
        $div1.css({"background": "red"});

})

</script>
```

  - eq方法：选取指定索引的标签

```
// 2.eq : 使用索引选择标签
var $div2 = $divs.eq(1);
$div2.css({"color": "blue"});
```

  - html代码:

```
<body>
    <div>
        这是第一个div
        <input type="text" id="mytext">
    </div>

    <div>
        这是第二个div
        <input type="text">
        <input type="button">
    </div>
</body>
```

11.12 [难点]选择集转移

- 介绍:

> 以选择的标签为参照，然后获取转移后的标签

- 操作:

- $('#box').prev();
- $('#box').prevAll();
- $('#box').next();
- $('#box').nextAll();
- $('#box').parent();
- $('#box').children();
- $('#box').siblings();
- $('#box').find('.myClass');

```
<script src="js/jquery-1.12.4.min.js"></script>

<script>

    $(function(){

        // 选择集转移：以当前选中标签为参照,找其他标签;
        var $div = $("#div01");
        // alert($div.length);

        // 1.上一个兄弟标签
        // $div.prev().css({"color":"red"});
        // 2.下一个兄弟标签
        // $div.next().css({"color":"red"});
        // 3.上面所有兄弟标签
        // $div.prevAll().css({"background": "blue"});
        // 4.下面所有兄弟标签
        // $div.nextAll().css({"background": "green"});
        // 5.上下所有兄弟标签
        $div.siblings().css({"color":"red"});

        // 6.找父级标签
        // $div.parent().css({"background":"blue"});
        // 7.找所有子标签
        $div.children().css({"color":"blue"});

        // 8.找满足选择器的子标签
        $div.find(".sp02").css({"font-size": "30px"});

    })
</script>
```

```
<div>
    <h2>这是第一个h2标签</h2>
    <p>这是第一个段落</p>
    <div id="div01">这是一个<span>div</span>第二个<span class="sp02">span</span></div>
    <h2>这是第二个h2标签</h2>
    <p>这是第二个段落</p>
</div>
```

11.13 [重点]获取和设置元素内容

- html方法:
  - 获取元素内容: `$div.html()`
  - 设置元素内容: `$div.html("...")`
    - 追加元素内容: `$div.append("....")`

```
<script src="js/jquery-1.12.4.min.js"></script>

<script>

    // 获取标签对象的入口函数
    $(function(){
        // html方法:可以获取标签内容 和 设置标签内容
        // append方法:追加标签内容
        var $div = $("#div1")

        // 1. 获取标签内容
        alert($div.html())

        // 2. 设置标签内容 : 删除原来内容
        $div.html("<a href='https://www.baidu.com'>百度网</a>")

        // 3. 追加标签内容 : 不会删除原来内容
        $div.append("<br><a href='https://www.itheima.com'>黑马程序员网</a>")

    })

</script>
```

11.14 [重点]获取和设置元素属性

- prop方法:
  - 获取元素属性: `$div.prop("属性")`
  - 设置元素属性
    - `$div.prop({"属性名":"值", ...})`
- val方法:
  - 获取标签value属性: `$input.val()`
  - 设置标签value属性: `$input.val("222222")`
- css方法
  - 获取标签css属性: `$("div").css("color");`
  - 设置标签css属性: `$("div").css({"color":"red"})`

```
<head>

    <script src="js/jquery-1.12.4.min.js"></script>

    <script>
```

```
        // 获取标签对象的入口:
        $(function(){
            var $a = $("#link01");
            var $input = $("#input01");

            // prop方法: 获取和设置标签属性

            // 1. 获取标签属性
            alert($a.prop("id"));

            // 1.2 获取value属性 val()
            // alert($input.prop("value"));
            alert($input.val());


            // 2. 设置标签属性
            $a.prop({"href": "https://www.baidu.com"});

            // 2.2 设置value属性 val("222222")
            // $input.prop({"value": "222222"});
            $input.val("222222");


            // 3. 设置css属性
            $a.css({"color":"red"});
            // 3.2. 获取css属性
            alert($a.css("color"));

        })
    </script>

</head>
<body>

    <a id="link01">这是一个链接</a>
    <input type="text" id="input01" value="111111">

</body>
```

11.15 [重点]jQuery事件

- 常见事件:
  - click() : 点击事件
  - blur() : 失去焦点事件
  - focus() : 获取焦点事件
  - mouseover() : 鼠标进入事件
  - mouseout() : 鼠标离开事件
  - ready() : 页面加载完成事件
- 注意:

> this指的是当前发生事件的对象，但是它是一个原生js对象
>
> $(this) 指的是当前发生事件的jquery对象

```
<script>
    $(function(){
        var $li = $('.list li');
        var $button = $('#button1')
        var $text = $("#text1");
        var $div = $("#div1")

        //  鼠标点击
        $li.click(function(){
            // this指的是当前发生事件的对象，但是它是一个原生js对象
            // this.style.background = 'red';

            // $(this) 指的是当前发生事件的jquery对象
            $(this).css({'background':'gold'});
            // 获取jquery对象的索引值,通过index() 方法
            alert($(this).index());
        });

        //  一般和按钮配合使用
        $button.click(function(){
            alert($text.val());
        });

        //  获取焦点
        $text.focus(function(){
            $(this).css({'background':'red'});

        });

        //  失去焦点
        $text.blur(function(){
            $(this).css({'background':'white'});

        });

        //  鼠标进入
        $div.mouseover(function(){
            $(this).css({'background':'gold'});

        });

        //  鼠标离开
        $div.mouseout(function() {
            $(this).css({'background':'white'});
        });
    });
</script>

<div id="div1">
    <ul class="list">
        <li>列表文字</li>
        <li>列表文字</li>
        <li>列表文字</li>
```

```
    </ul>

    <input type="text" id="text1">
    <input type="button" id="button1" value="点击">
</div>
```

11.16 [重点]事件代理

- 事件冒泡:

  > 事件会向它的父级一级一级传递

- 事件代理:

  > 事件加到父级上，通过判断事件来源，执行相应的子元素的操作。

- 作用:

  > 事件代理首先可以极大减少事件绑定次数，提高性能；其次可以让新加入的子元素也可以拥有相同的操作。

- 使用:

  - `delegate`

```
<head>

    <script src="js/jquery-1.12.4.min.js"></script>

    <script>

        // 获取标签对象的入口函数
        $(function(){

            // 1.一般方式绑定事件
            // 循环绑定 : 1)效率低下; 2)新增的子标签没有绑定事件;
            /*
            var $ul = $("#list");
            var $lis = $("#list li");
            $lis.click(function(){
                $(this).css({"color": "red"});
            })

            $ul.append("<li>6</li>");
            */

            // 2. 使用父标签代理 子标签完成 事件处理
            // 绑定父标签 : 1) 效率高; 2)新增的子标签触发事件也能执行;

            // delegate("子标签选择器", "事件名", 事件触发处理函数);
            var $ul = $("#list");

            $ul.delegate("li", "click", function(){
```

```
                $(this).css({"color": "red"});
            });

            $ul.append("<li>6</li>");

        })
    </script>

</head>
<body>

    <ul id="list">
        <li>1</li>
        <li>2</li>
        <li>3</li>
        <li>4</li>
        <li>5</li>
    </ul>

</body>
```

11.17 [重点]JavaScript对象

- 介绍:

  > JavaScript 中的所有事物都是对象, JavaScript 允许自定义对象，对象可以拥有属性和方法.

- JavaScript创建对象操作:
  - 顶级Object类型:

```
// 2.Object方法
var oPerson = new Object();

oPerson.name = "xiaofang";
oPerson.age = 22;

oPerson.sayInfo = function(){
    alert("你好");
}

console.log(oPerson.name, oPerson.age);
oPerson.sayInfo();
```

  - 对象字面量:

```
// 定义js的对象:
// 1.字面量
var oPerson = {
    name: "xiaoming",
    age: 20,
    sayHello: function(){
        alert("hello");
    }
}

console.log(oPerson.name, oPerson.age);
oPerson.sayHello();
```

11.18 [重点]json

- 介绍:

  > 类似于javascript对象的字符串，它同时是一种**数据格式**

- 格式:
  - 对象格式 : `{"name": "小明"}`
  - 数组格式 : `'[{"name": "小芳"}, {"name": "小王"}]'`
- json转换javascript对象
  - ```
    // json字符串 转换 成为 javascript的对象
    // JSON.parse(json字符串) 返回 javascript对象.
    var oPerson = JSON.parse(json_str);

    console.log(oPerson.name, oPerson.age);

    var aList = JSON.parse(json_strs);

    for(var i = 0; i < aList.length; i++){
        console.log(aList[i].name, aList[i].age);
    }
    ```

11.19 [重点]ajax

- 介绍:

  > 当前端页面想和后台服务器进行数据交互就可以使用ajax了.

- 使用:
  - $.ajax(js对象)

    ```
    $.ajax({
        // 1. url地址
        url: "http://127.0.0.1:9000/data.json",

        // 2. type 请求方式，默认是'GET', 常用的还有'POST'
        type: "GET",
    ```
    ```

```
    // 3. dataType 设置返回的数据格式，常用的是'json'格式
    dataType: "JSON",

    // 4.data 设置发送给服务器的数据，没有参数不需要设置

    // 5. success 设置请求成功后的回调函数
    success: function (response) {
    console.log(response)

    // 进行局部刷新
    // $("#div").append(typeof(response) + response.cityInfo.city)
    $("#div").append(response.data.forecast[11].ymd)
    } ,

    // 6. error 设置请求失败后的回调函数
    error: function () {
    alert("请求失败，请检查参数")
    },

    // 7. async 设置是否异步，默认值是'true'，表示异步，一般不用写
    async: 'true'
})
```

- 对象属性:
  - url : 请求的web服务器地址
  - type： 请求方式
  - dataType : 请求的数据类型
  - data： 请求携带的数据（没有可以不写）
  - success： 设置请求成功的回调函数
  - error : 设置请求失败的回调函数
  - async : 是否异步请求
- 简写:

```
$.get(url,data,success(data, status, xhr),dataType).error(func)
$.post(url,data,success(data, status, xhr),dataType).error(func)
```

```
// ajax的简写方式
// $.get()   $.post()
$.get("data.json", function (response) {
    console.log(response)
    $("#div").append(response.message)
}, "JSON").error(function () {
    alert("请求失败，请检查参数")
},)
```

11.20 知识点总结