20230921

1.DOM对象

内容：创建DOM对象

1.节点操作

createElement（）创建节点

createTextNode（）创建文本节点

appendChild（）添加子节点

父节点.insertBefore（需要插入的节点，父节点中的子节点）

父节点.removeChild（子节点）

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <div id="id_div_01">


    </div>

    <div id="id_div_02">

        <p id="id_p_01">this is a p
        </p>

    </div>

    <div id="id_div_03">

        <p id="id_p_02">11111</p>

    </div>

    <button onclick="func01()">touch1</button>

    <button onclick="func02()">touch2</button>

    <button onclick="func03()">touch3</button>

    <script>

        function func01() {

            var div_1 = document.getElementById("id_div_01");
```

```
            //创建p标签节点
            var p_1 = document.createElement("p");
            //动态创建文本节点
            var txt = document.createTextNode('hello world')
            //向p_1元素节点添加子节点
            p_1.appendChild(txt);
            //向div_1元素节点添加子节点
            div_1.appendChild(p_1);
        }

        function func02(){
            var div_2 = document.getElementById("id_div_02");
            var p_1 = document.getElementById("id_p_01");

            var h_1 = document.createElement("h1");
            h_1.innerText = 'Hello';
            h_1.align = 'center';
            //在div_2里面把h1标签放到p之前
            div_2.insertBefore(h_1,p_1);
        }
        function func03(){
            var div_3 = document.getElementById("id_div_03");
            var p_2 = document.getElementById("id_p_02")

            div_3.removeChild(p_2)


        }

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

案例动态添加文本框

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <div id="id_div_01">

        <input type="text"  id="id_ipt_01" placeholder="请输入电话号">
        <br>

        <button onclick="func01()" id="id_but_01">添加电话号</button>

    </div>

    <script>

        function func01() {

            var div_1 = document.getElementById("id_div_01");
```

```
            var ipt1 = document.getElementById("id_ipt_01");
            var ipt2 = document.createElement('input');
            var br = document.createElement('br');
            var button = document.getElementById("id_but_01");

            ipt2.type = 'text';
            ipt2.placeholder = "请输入电话号";

            div_1.insertBefore(ipt2,button);
            div_1.insertBefore(br,button);

        }

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

2.查看节点信息

parentNode 父节点（一个）（只能获取其父节点的信息）

childNodes 所有子节点（数组）（一个父类中可以有很多子类）

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <div id="id_div_01">

        <p id="id_p_01">p1</p>

        <p id="id_p_02">p2</p>

        <input type="text" value="1">

    </div>

    <button onclick="func01()">查看父节点</button>

    <button onclick="func02()">查看子节点</button>

    <button onclick="func03()">查看节点信息</button>
    <script>

        function func01() {

            var p1 = document.getElementById("id_p_01");

            // 获取p1节点的父节点
            var parent_obj = p1.parentNode;
            console.log(parent_obj);
```

```
        }

        function func02(){

            var div_1 = document.getElementById("id_div_01")

            //查看子节点 返回的数组
            //注意:html代码格式会被认为是子节点

            var div_child = div_1.childNodes;

            console.log(div_child);

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

                console.log(div_child[i]);

            }

        }

        function func03(){

            var div_1 = document.getElementById("id_div_01")

            console.log(div_1.nodeName);
            console.log(div_1.nodeType);
            console.log(div_1.nodeValue);
            for(var i = 0; i < div_1.childNodes.length ;i++){

                if(div_1.childNodes[i].nodeName == 'INPUT'){

                    console.log(div_1.childNodes[i].value);

                }

           }

        }

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

3.动态生成表格

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
```

```
<body>
    <div id="id_div_01">



    </div>


    <button onclick="createTable()">touch</button>


    <script>

        function createTable(){

            var div_1 = document.getElementById('id_div_01');

            var table_new = document.createElement('table');

            table_new.setAttribute('width', 500);
            table_new.setAttribute('height',500);
            table_new.setAttribute('border', '1px');
            table_new.setAttribute('align', 'center');

            //添加行

            for (var i = 1; i <= 5 ; i++) {
                var tr_new = table_new.insertRow()
                for (var j = 1; j<=3;j++){

                    var td_new = tr_new.insertCell()
                    td_new.innerText='内容'+i+'-'+j
                    // tr_new.appendChild(td_new)

                }

            }
            // table_new.appendChild(tr_new)

            div_1.appendChild(table_new)

        }

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

2.BOM对象

定义：浏览器对象模型，用于操作浏览器窗口的一组接口

BOM/DOM区别

DOM接口错做浏览器内的body元素

BOM操作浏览器本身，比如窗口大小，浏览器执行时间

BOM主要对象：window,navigator,screen,history,location（后面的对象都是window的属性对象，获取方式为 window.XXX)

1.window

循环执行和延迟执行 定时器

setInterval

clearInterval

setTimeout

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

    <button onclick="stopTimer()">stopInterview</button>
    <button onclick="stopdlay()">stopTimeout</button>
    <script>

        // 设置定时器

        // window.setInterval(func01,0)
        // //函数表达式：函数名 或者匿名函数

        // function func01() {

        //     console.log(Date.now().toLocaleString())

        // }

        // 停止定时器

        // 如果需要停止setInterval 必须要有定时器对象

        var timer = setInterval(function(){
            console.log('ok')
        },1000);

        function stopTimer(){

            window.clearInterval(timer)

        }

        //格式同setInerval
        var delay = setTimeout(function(){

            alert('hello')
```

```
        },2000)

        function stopdlay(){

            clearTimeout(delay)

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

定时器案例

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <div id="id_div_01">
        当前事件
    </div>
    <button onclick="startTime()">开始</button><br>
    <button onclick="stopTime()">暂停</button><br>


    <script>

        var div_01 = document.createElement("id_div_01");

        function showData() {

            var date_now = new Date();
            date_now = date_now.toLocaleString();
            var div01 = document.getElementById('id_div_01');
            div01.innerText=date_now;

        }


        function startTime() {

            timer = setInterval(showData,0)


        }

        function stopTime() {

            clearInterval(timer);

        }
```

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

open()

close()

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

    <title>Document</title>
</head>
<body>

    <!-- // 格式：open('页面路径''页面打开方式') -->
    <button onclick="window.open('demo_03.html')">打开新窗口</button>
    <button onclick="window.open('https://www.baidu.com')">新窗口打开百度</button>
    <button onclick="window.open('https://www.bvaidu.com','_top')">原窗口打开百度
</button>

    <button onclick="window.close('')">关闭</button>
    <script>



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

2.navigator

封装在window对象中，作用是 可以返回浏览器的各种信息

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>

        console.log(window.navigator.appCodeName)
        console.log(navigator.appName)
        console.log(navigator.userAgent)
        console.log(navigator.language)
        console.log(navigator.plugins)
```

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

3.screen对象

表示窗口对应的屏幕信息

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>

        console.log(window.screen.width)
        console.log(screen.height)
        console.log(screen.availWidth)
        console.log(screen.availHeight)
        console.log(screen)
        console.log(window.devicePixelRatio)

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

4.history

窗口的历史浏览记录

length历史列表中的url数量

back（）后退 forward（）前进 go（step）跳转至当前页面的第step个url路径（step参数可以为负值，往回退）

5.lacation对象

表示窗口地址对象

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

    <button onclick="location.href='https://www.baidu.com'">页面跳转</button>
```

```
    <button onclick="location.reload()">页面刷新</button>
    <script>
        //当前窗口的地址域名以及端口
        console.log(window.location.host)
        //当前窗口域名
        console.log(location.hostname)
        //当前地址的请求方式
        console.log(location.protocol)
        //端口号
        console.log(location.port)

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

3.json数据格式

定义：json是javaScript Object Notation的缩写，是一种轻量级的数据交换形式，基于ECMAscript的标准。

特点：采用js原生格式，处理json不需要任何特殊API和工具包

是一种xml的替代方式，比xml更小，更容易解析

json的书写格式采用key-value（键值对形式）

json对象可以无限嵌套

1.json案例

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <!-- //json对象 -->
    <script>

        var json_obj = {

            'id':100,
            'username':'tom',
            'age':18,
            'content':{
                'qq':2479362948,
                'tele':18640928449

            }
        }

        console.log(json_obj);
        console.log(json_obj.content.qq)
```

```
        //json字符串
        var json_str = '{  "id":100,"username":"tom","age":18,"content":
{"qq":2479362948,"tele":18640928449}}'

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

2.json转换

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>


    <script>

        var str = '{"id":101,"name":"xmz","age":27}'
        //json字符转json对象
        var json_obj = JSON.parse(str);

        console.log(json_obj.name);

        json_obj.name = 'lisan'
        var json_str = JSON.stringify(json_obj);
        console.log(json_str);

    </script>

</body>
</html>
```

作业

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>生成表格</title>
    </head>
    <body>
        <div>
            <button onclick="generateTable()">生成表格</button>
            <div id="tableContainer"></div>
        </div>

        <script>
```

```
            var info = '[{"id":101,"name":"jack","age":27},{"id":102,"name":"张三","age":20},{"id":103,"name":"李四","age":22},{"id":104,"name":"赵四","age":45}]';

            function generateTable() {
                var tableData = JSON.parse(info);
                var tableContainer = document.getElementById("tableContainer");

                // 创建表格元素
                var table = document.createElement("table");
                table.setAttribute("border", "1");

                // 创建表头
                var thead = document.createElement("thead");
                var headerRow = thead.insertRow();
                var headers = ["ID", "Name", "Age"];

                for (var i = 0; i < headers.length; i++) {
                    var th = document.createElement("th");
                    th.textContent = headers[i];
                    headerRow.appendChild(th);
                }

                thead.appendChild(headerRow);
                table.appendChild(thead);

                // 创建表格内容
                var tbody = document.createElement("tbody");

                for (var i = 0; i < tableData.length; i++) {
                    var row = tbody.insertRow();
                    var cell1 = row.insertCell(0);
                    var cell2 = row.insertCell(1);
                    var cell3 = row.insertCell(2);

                    cell1.textContent = tableData[i].id;
                    cell2.textContent = tableData[i].name;
                    cell3.textContent = tableData[i].age;
                }

                table.appendChild(tbody);

                // 将表格添加到页面中
                tableContainer.innerHTML = ""; // 清空容器
                tableContainer.appendChild(table);
            }
        </script>
    </body>
</html>
```