20190726课堂笔记

日期对象的基本操作

```
let time = new Date();
/*
 * 获取当前客户端（本机电脑）本地的时间
 *     这个时间用户是可以自己修改的，所以不能作为重要的参考依据
 *
 * Fri Jul 26 2019 10:02:17 GMT+0800 (中国标准时间)
 *     获取的结果不是字符串是对象数据类型的，属于日期对象(或者说是Date这个类的实例对象)
 */
typeof time;  //=>"object"
```

标准日期对象中提供了一些属性和方法，供我们操作日期信息

- getFullYear() 获取年
- getMonth() 获取月 结果是0~11代表第一月到第十二月
- getDate() 获取日
- getDay() 获取星期 结果是0~6代表周日到周六
- getHours() 获取时
- getMinutes() 获取分
- getSeconds() 获取秒

- getMilliseconds() 获取毫秒
- getTime() 获取当前日期距离1970/1/1 00:00:00 这个日期之间的毫秒差
- toLocaleDateString() 获取年月日（字符串）
- toLocaleString() 获取完整的日期字符串

小时钟案例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>小时钟</title>
    <!-- IMPORT CSS -->
    <style>
        * {
            margin: 0;
            padding: 0;
        }
        #clockBox {
            position: absolute;
            right: 0;
            top: 0;
            padding: 0 15px;
            line-height: 70px;
            font-size: 24px;
            color: darkred;
            /* 设置背景渐变色 */
            background: lightblue;
```

```
            background: -webkit-linear-gradient(top left, lightblue, lightcoral, lightcyan);
        }
    </style>
</head>
<body>
    <div id="clockBox">
        2019年07月26日 星期五 10:25:03
    </div>
    <!-- IMPORT JS -->
    <script>
        let clockBox = document.getElementById('clockBox');
        /*
         * addZero:不足十补充零
         *   @params
         *      val需要处理的值
         *   @return
                处理后的结果（不足十位的补充零）
         * by Team on 2019/07/26
         */
        function addZero(val) {
            val = Number(val);
            return val < 10 ? '0' + val : val;
        }

        /*
```

```
47          * queryDate:获取当前的日期，把其转换为想要的格式
48          *   @params
49          *   @return
50          * by Team on 2019/07/26
51          */
52         function queryDate() {
53             // 1.获取当前日期及详细信息
54             let time = new Date(),
55                 year = time.getFullYear(),
56                 month = time.getMonth() + 1,
57                 day = time.getDate(),
58                 week = time.getDay(),
59                 hours = time.getHours(),
60                 minutes = time.getMinutes(),
61                 seconds = time.getSeconds();
62             let weekAry = ['日', '一', '二', '三', '四', '五', '六'];
63             // 2.拼凑成我们想要的字符串
64             let result = year + "年" + addZero(month) + "月" + addZero(day) + "日";
65             result += " 星期" + weekAry[week] + " ";
```

```
66                 result += addZero(hours) +
   ":" + addZero(minutes) + ":" +
   addZero(seconds);
67                 // 3.把处理好的结果放到盒子中
68                 clockBox.innerHTML = result;
69             }
70             // 加载页面执行方法
71             queryDate();
72             // 定时器控制运动：设置一个
   setInterval定时器（到达指定时间干什么事情的东
   西就是定时器），每隔1000MS执行queryDate方法
73             setInterval(queryDate, 1000);
74         </script>
75 </body>
76 </html>
```

> new Date() 除了获取本机时间，还可以把一个时间格式字符串转换为标准的时间格式

```
1 new Date("2019/7/26");
2 //=>Fri Jul 26 2019 00:00:00 GMT+0800 (中
   国标准时间)
3
4 /*
5  * 支持的格式
6  *     yyyy/mm/dd
7  *     yyyy/mm/dd hh:mm:ss
8  *     yyyy-mm-dd 这种格式在IE下不支持
9  */
```

时间字符串格式化案例

> 字符串处理解决办法

```
// =>不足十位补充零
let addZero = val => {
    val = Number(val);
    return val < 10 ? '0' + val : val;
};

/*
* 字符串处理解决办法
*/
function formatTime(time) {
    // 1.先获取年月日等信息
    let ary = time.split(' '),
        aryLeft = ary[0].split('-'),
        aryRight = ary[1].split(':');
    ary = aryLeft.concat(aryRight);
    // 2.拼接成为我们想用的格式
    let result = ary[0] + "年" + addZero(ary[1]) + "月" + addZero(ary[2]) + "日";
    result += " " + addZero(ary[3]) + ":" + addZero(ary[4]) + ":" + addZero(ary[5]);
    return result;
}
let time = '2019-5-30 12:0:0';
time = formatTime(time);
console.log(time);
// ="2019年05月30日 12:00:00"
```

> 基于日期对象处理

```
/*
 * 基于日期对象处理
 */
function formatTime(time) {
    // 1.把时间字符串变为标准日期对象
    time = time.replace(/-/g, '/');
    time = new Date(time);
    // 2.基于方法获取年月日等信息
    let year = time.getFullYear(),
        month = addZero(time.getMonth() + 1),
        day = addZero(time.getDate()),
        hours = addZero(time.getHours()),
        minutes = addZero(time.getMinutes()),
        seconds = addZero(time.getSeconds());
    // 3.返回想要的结果
    return year + "年" + month + "月" + day + "日 " + hours + ":" + minutes + ":" + seconds;
}
let time = '2019-5-30 12:0:0';
time = formatTime(time);
console.log(time);
// =>"2019年05月30日 12:00:00"
```

> 封装一套公共的时间字符串格式化处理的方式

```
/*
 * 封装一套公共的时间字符串格式化处理的方式
 */
String.prototype.formatTime = function formatTime(template) {
    // 初始化模板
    typeof template === 'undefined' ? template = "{0}年{1}月{2}日 {3}:{4}:{5}" : null;
    // this:我们要处理的字符串
    // 获取日期字符串中的数字信息
    let matchAry = this.match(/\d+/g);
    // 模板和数据的渲染（引擎机制）
    template = template.replace(/\{(\d+)\}/g, (x, y) => {
        let val = matchAry[y] || '00';
        val.length < 2 ? val = '0' + val : null;
        return val;
    });
    return template;
};
let time = '2019-5-30 12:0:0';
console.log(time.formatTime("{1}-{2} {3}:{4}"));
// =>"2019年05月30日 12:00:00"
```

DOM及其基础操作

> DOM：document object model 文档对象模型，提供一些属性和方法供我们操作页面中的元素

获取DOM元素的方法

- document.getElementById() 指定在文档中，基于元素的ID或者这个元素对象
- [context].getElementsByTagName() 在指定上下文(容器)中，通过标签名获取一组元素集合
- [context].getElementsByClassName() 在指定上下文中，通过样式类名获取一组元素集合（不兼容IE6~8）
- document.getElementsByName() 在整个文档中，通过标签的NAME属性值获取一组节点集合（在IE中只有表单元素的NAME才能识别，所以我们一般只应用于表单元素的处理）
- document.head / document.body / document.documentElement 获取页面中的HEAD/BODY/HTML 三个元素
- [context].querySelector([selector]) 在指定上下文中，通过选择器获取到指定的元素对象
- [context].querySelectorAll([selector]) 在指定上下文中，通过选择器获取到指定的元素集合

```
//=> querySelector / querySelectorAll 不兼容IE6~8
let box = document.querySelector('#box');
let links = box.querySelectorAll('a');
// links=document.querySelectorAll('#box a');
let aas=document.querySelectorAll('.aa');
```

JS中的节点和描述节点之间关系的属性

> 节点 ： Node （ 页面中所有的东西都是节点 ）
>
> 节点集合：NodeList （ getElementsByName / querySelectorAll 获取的都是节点集合 ）

- 元素节点（ 元素标签 ）
  - nodeType : 1
  - nodeName : 大写的标签名
  - nodeValue ：null
- 文本节点
  - nodeType : 3
  - nodeName : '#text'
  - nodeValue ：文本内容
- 注释节点
  - nodeType : 8
  - nodeName : '#commen'
  - nodeValue ：注释内容
- 文档节点 document

    - nodeType：9
    - nodeName : '#document'
    - nodeValue ：null
- ……

描述这些节点之家关系的属性

- childNodes ：获取所有的子节点
- children：获取所有的元素子节点（子元素标签集合）
- parent：获取父亲节点
- firstChild：获取第一个子节点
- lastChild：获取最后一个子节点
- firstElementChild / lastElementChild ：获取第一个和最后一个元素子节点（不兼容IE6~8）
- previousSibling：获取上一个哥哥节点
- nextSibling：获取下一个弟弟节点
- previousElementSibling / nextElementSibling ：获取哥哥和弟弟元素节点（不兼容IE6~8）
- ……