目录

- BOM：一个浏览器窗口就是一个BOM
  - `open()`： `open("url","窗口名",'width=300,height=300,top=200,left=200');`
  - `history`： `.length` `.back()` `.forward()` `.go()`
  - `location`： `protocol://hostname:port/pathname?search#hash`
    - `.assign(url)` `.replace(url)` `.reload()`
- DOM: `html`开始到`html`结束
- 获取元素节点
  - `document.getElementById('id')`
  - `node.getElementsByClassName('class')`
  - `node.getElementsByTagName('tag')`
  - `document.getElementsByName('name')`
  - `document.querySelector(".li")`
  - `document.querySelectorAll(".li")`
  - 元素节点属性
    - `innerHTML` `innerText` `outerHTML`
- 获取`style`行间属性值
  - `node.属性` (获取行内, `object`)
  - 获取当前有效样式 (只能获取属性值，不能设置属性值);
    - IE: `node.currentStyle["属性"]`
    - 谷歌: `getComputedStyle(node)["属性"]`
      - 三目运算符解决兼容性
  - `node.getattribute()` (获取行内,`style`字符串)
    - `node.getAttribute('')`
    - `node.setAttribute('','v')`
    - `node.removeAttribute('')`
      - 可以设置/访问自定义样式
- 访问子节点
  - 获取元素节点和文本节点(文本节点唯一获取方式)
    - `node.childNodes` `.firstChild` `.lastChild`
  - 获取元素节点
    - `node.children` `.firstElementChild` `.lastElementChild`
- 获取属性节点 `attributes`
  - `node.attributes["id"]`
  - `node.attributes[0]`
    - 唯一获取属性节点的方式
- 节点的属性： `nodeName/nodeType/nodeValue`
- `document.styleSheets[0]` 获取页面的`style`样式

1.认识BOM

- BOM: browser object model (浏览器对象模型)
- **一个浏览器窗口就是一个BOM**(`window`)
  - 无论ECMAscript多么强大，还是要遵循浏览器规则

- `window` 方法 （一般情况下可以省略 `window` ）
  - 【注】所有的属性、方法、变量、函数、对象 前都可以加 `window`
  - `window.alert();` ：弹出警告框

  - `window.confirm("内容");` ：弹出一个带 `确定` 和 `取消` 的对话框
    - 返回值：点击 `确定` 返回 `true` ；点击 `取消` 返回 `false`

  - `window.prompt('内容', '默认值');` ：弹出一个带输入框的提示框
    - 第一个参数: 面板上要*显示的内容*。
    - 第二个参数：输入框里面的默认值 (可以不传入)
    - 返回值: 点击 `确定` ，返回输入框里的内容；点击 `取消` ，返回 `null`
  - `window.open()` 方法 (open打开窗口方法优先级大于 `html` 的 `target`)
    - `window.open(URL,name,specs)`
      - `URL` ：字符串，跳转的 `url`
      - `name` ：字符串， `_blank` (默认,新窗口) / `_self` (本窗口) / `_parent` (在父框架打开) / 自定义 `name`
        - `name` 给打开的窗口起个名字，第二次打开新窗口是在起名字的窗口内打开
      - `specs` ：一串特殊含义的字符串，可以控制打开窗口的属性
        - `width` ： 窗口显示区的宽度,以像素计
        - `height` ： 窗口显示区的高度,以像素计
        - `top` ： 窗口的y坐标
        - `left` ： 窗口的x坐标

```
open("链接地址","窗口名",'width=300,height=300,top=200,left=200');
```

2.history对象 (window下的一个属性)

- `window.history` ：掌管的是,**当前窗口**(注意不是浏览器)**的历史记录**
  - 只要当前窗口加载的 `url` 不一样，就会产生新的历史记录
  - `history.length` 输出当前窗口**历史记录的条数**(包括本窗口)

  - `history.back()` ：返回上一条历史记录
  - `history.forward()` ：返回下一条历史记录
  - `history.go(n)` ：前进/后退 n 条历史记录, 传入 `0` 刷新当前页面

3.location对象的属性 (location 地址栏)

- `url`：统一资源定位符。
- 完整的`url`：
  中文： 协议://IP(域名):端口号/路径?查询字符串#锚点
  英文： `protocol://hostname:port/pathname?search#hash`
- 协议： `location.protocol`
  `file`：本地磁盘文件访问
  `http`
  `https`：(证书认证协议)

- `IP`(域名)-主机名： `location.hostname`
  `localhost`： (本地主机)
  `www.baidu.com`：百度

- 端口号： `location.port` (端口号，默认隐藏)
  - 是当前电脑中使用网络的软件，随机给它分配一个编号 0~65535
  - `hostname.port` 可以直接定位到当前使用网络的程序。
    - 浏览器的默认端口号 `8080`
    - `http`默认端口号 `80`
    - `https`默认端口号 `443`

- 路径： `location.pathname`当前文件的路径

- 查询字符串： `location.search` (前后端交互学习)
  - `?name1=value1&name2=value2`

- 锚点： `location.hash` (前端路由利用了`hash`)

```
alert(location)     //输出整个url  类型为object
alert(location.href)//输出整个url  类型为String
```

4.location对象的方法

```
alert(window.location === window.document.location);    //true
```

- `location.assign('url')`： 在当前窗口进行跳转
- `location.replace('url')`： 在当前窗口替换成新的`URL` 不会产生历史记录
  【注】
- `location.reload()`： 刷新本页面

```
// 传参true，不经过浏览器缓存，强制从服务器重载
location.reload(true);
```

5.认识DOM

- DOM : document object model 文档对象模型
  - `document` : html开始到/html结束

```
<div id="div1" title="div" class="box" style="height: 100px;">
  div文本
</div>
```

- 元素节点 `<div></div>`
- 属性节点 `id="div1"` `title="div"` `class="box"` `style="width: 100px;"`
- 文本节点 `div文本`

- 元素节点的获取
  - `document.getElementById(id)`
  - 功能：通过`id`获取符合条件的元素。（`id`必须唯一）
  - 返回值：符合条件的一个节点。
- 属性节点的获取
  - `node.id` 获取id属性值
  - `node.title` 获取title属性值

  - 获取`class`： `node.className`
    - `class`在JS中是关键字，要通过`className`获取

  - 获取 `style` CSS样式
    - `node.style.属性` ： 获取style的指定属性值
      - `oDIV.style.width`
      - 如果CSS样式中带`-`，访问时要把`-`去掉，从第二个单词开始，首字母大写
        - 获取`background-color`： `oDIV.style.backgroundColor`
    - 这种方法只能访问行间样式，赋值也是赋值在行间。
    - 值的修改必须传入字符串

      ```
      oDIV.style.width = "50px"
      ```

- 扩展

```
width = oDIV.style.width;
// 这种方法获取的是属性内容，修改width，并不会修改oDIV.style.width
  // 原因是 : oDIV.style.width对应的值是基本数据类型String,和 '=' 的特性

oDiv.style["width"] = 10px; 也可以设置属性值
```

6.获取元素节点的方法 (获取到的元素节点都是 字符串类型)

- `document.getElementById('id');`

- `node.getElementsByTagName('标签名');`
  - 功能: 从 `node` 节点开始，通过标签名获取符合条件的元素节点。
  - 返回值：伪数组/类数组

- `node.getElementsByClassName('class名字');` (IE8以下不兼容)
  - 功能: 从 `node` 节点开始，通过 `class` 名字获取符合条件的元素节点。
  - 返回值：伪数组/类数组

- `document.getElementsByName(name属性的值);` 使用在表单元素
  - 功能: 通过 `name` 属性的值获取符合条件的元素节点。
  - 返回值：伪数组/类数组 注：只能全局查找
    - 除表单元素外，设置 `name` 属性是没有用的。一般使用在表单元素

- `document.querySelector();` IE8以下不兼容
  - 返回值：一个元素节点，找到符合条件的第一个元素节点
- `document.querySelectorAll();` IE8以下不兼容
  - 返回值：一个伪数组。
  - 参数：字符串，CSS选择器格式的字符串(具体看html第一部分,笔记2)

- 伪数组
  - 可以通过 `伪数组.length` 输出长度，`伪数组[下标]` 访问对应的元素。

```
var oLitag = document.getElementsByTagName("li");
alert(oLitag.length);
alert(oLitag[0].innerHTML);

console.log(oLitag); // Prototype指向HTMLCollection
console.log([]);     // Prototype指向Array
// 使用起来和数组类似，一般情况，把这种叫做伪数组/类数组(自己起的);
```

- 获取指定区域下的 `li` 节点，如获取 `id="ol1"` 下的 `li`

```
var oOL = document.getElementById("ol1");    //获取ol节点
var oLis = oOL.getElementsByTagName("li");

// 使用 document.querySelector()
var oLill = document.querySelector("#ul1 #li");
```

7.自定义byClassName方法

- 兼容IE8以下，通过 `class` 名字获取节点的方法

```
function elementsClassName(node,classStr){
  // 1.获取node节点下面所有的子节点
  var nodes = node.getElementsByTagName("*");
  var arr = [];   //存储符合条件的子节点

  // 2.遍历，筛选符合条件的子节点
  for(var i = 0; i < nodes.length; i++){
    if(nodes[i].className === classStr){
      arr.push(nodes[i]);
    }
  }
  return arr;
}
```

8.获取当前有效样式 （只能获取值，不能设置值）

- 通过`node.style.xxx`的方式只能访问内联(行间)的css样式

- `node.getcurrentStyle["width"]` ： IE兼容
- `getComputedStyle(node)["width"]` ： 火狐谷歌
  - 跨浏览器兼容问题 (和获取class兼容一样)

```
function getStyle(node,styleName){
  //利用三木运算符，判断getcurrentStyle是否为真
  return node.getcurrentStyle ? node.getcurrentStyle[styleName] :
   getComputedStyle(node)[styleName];
}
```

  - 这种方法是获取 **节点所有拥有的样式，不止局限于行内**

9.练习(答案看9_练习.html)

- 写一个定时器，每一秒修改一次div内文本颜色和文字大小
  - 利用`Math.random()`随机`rgb(a,b,c)`的值
- 最开始 文本是默认大小，开始的时候增大，当增大了6次以后，文本开始缩小，缩小了6次以后，文本开始增大
  - 当变化次数%6 == 0时，让增加的值*-1;

10.attribute获取行间属性值 （获取行间属性值总结）

- `node.getAttribute("class")` ： 获取属性节点的值
- `node.steAttribyte("class","box5")` ： 设置属性节点的值
- `node.removeAttribute("class")`： 删除属性节点
  - 和`node.xxx`一样, 只能获取行内

- 访问属性节点
  - 通过`node.属性`访问

        - `oDiv.id` `oDiv.title`
    - 通过 `nide.getAttribute("属性")`
        - `oDiv.getAttribute("id")` `oDiv.getAttribute("title")`
- 两者区别
    1. `class`的访问
        - `node.className`
        - `node.getAttribute("class");`
    2. `getAttribute()`可以访问行间 自定义属性

```
alert(node.xxx);          //undefined
alert(node.getAttribute("xxx"));     //yyy
```

    3. `setAttribute()`可以新增行间 自定义属性
        - `node.属性` 无法设置自定义属性

```
oDiv.zzz = "123";
alert(oDiv.zzz);     //123
```

            - 原因，`oDiv`是一个对象，`oDiv.zzz`实际上是给对象添加的一个属性。
        - `setAttribute()`自定义行间属性

```
oDiv.setAttribute("zzz","ooo");
alert(oDiv.getAttribute("zzz"));     //ooo
```

            - 访问自定义样式通过`getAttribute()`
    4. 删除属性节点

```
oDiv.title = "";
 // 只是将属性值替换成空字符串，不是真正意义上的删除

oDiv.removeAttribute("title");
 //真正删除
```

    5. **获取的结果不一致**

```
console.log(oD.style)   // object
console.log(oD.getAttribute("style")); // 'width:100px'
```

    - `getAttribute("attr")`和`node.属性`无法获取`<style>`标签中的样式
    - `set/getAttribute()`只能设置/获取整个行内`style`的字符串值

- 获取行间属性值方法总结：
    1. `node.属性`(获取行内，`object`)
        - `node.id` `node.className` `node.style.属性`
    2. 获取当前有效样式 (只能获取属性值，不能设置属性值);
        - IE: `node.currentStyle["属性"]`

    - 谷歌: `getComputedStyle(node)["属性"]`
        - 三目运算符解决兼容性
3. `node.getattribute()` (获取行内, `style` 字符串)
    - `node.getAttribute('')` `node.setAttribute('','v')` `node.removeAttribute('')`
        - 可以设置/访问自定义样式

11.元素节点的属性

- `node.innerHTML`
    - 赋值：会解析标签
    - 取值：获取标签间内容
- `node.innerText`
    - 赋值：不会解析标签
    - 取值：获取标签间纯文本
- `node.outerHTML`
    - 赋值：会解析标签
    - 取值：从外标签开始到外标签结束

```
<div> div文本 <em id="emm">em文本</em> </div>
alert(oEm.outerHTML);   //<em id="emm">em文本</em>
```

- `innerHTML` 和 `outerHTML` 的区别
    - `node.innerHTML` 获取标签间内容： `em文本`
    - `node.outerHTML` 获取的内容包括标签： `<em id="emm">em文本</em>`

12.元素节点的属性-访问子节点

- `childNodes`：访问当前节点下所有子节点
- `firstChild`：访问子节点中的首位
- `lastChild`： 访问子节点中的末位
- `nextSibling`：下一级同级节点
- `previousSibling`：上一级同级节点
    - 上述方法都包含文本节点；文本节点只能通过元素节点的子节点获取
    - 返回值：伪数组

```
alert(oDiv.childNodes);     //[object NodeList]
```

    - 不能访问属性节点

- 节点的属性

	nodeType	nodeName	nodeValue
元素节点	1	标签名	null
属性节点	2	属性名	属性值
文本节点	3	#text	文本内容

- 获取子节点注意点：空格、回车 看不见，是字符。

- 将文本节点剔除 (`childNodes` 可以遍历筛选)
  - 只获取子节点中的元素节点 (IE8以下不兼容)

  - `children`：访问当前节点下所有元素子节点
  - `firstElementChild`：第一个元素节点
  - `lastElementChild`： 最后一个元素节点
  - `nextElementSibling`： 下一同级元素节点
  - `previousElementSibling`：上一同级元素节点
    - 不能访问 属性节点 和 文本节点

- 区分 文本节点 和 元素节点间的内容(`innerHTML/innerText`)
  - 文本节点只能通过(`childNodes`)子节点获取

13.属性节点(attributes) 真正意义上的获取属性节点

- `node.attributes` ： 获取当前元素节点上的所有属性节点
  - 返回值： `NamedNodeMap` 集合 (1.无序 2.不重复)
    - 和 `Set/Map` 有区别的
- `attributes` 属性节点两种方式
  - `node.attributes["id"]`
  - `node.attributes[0]`
- 输出 属性节点的 `nodeName/nodeType/nodeValue` 只能通过 `attributes`

14. document.styleSheets[0] 获取页面的 `style` 样式

- 通过 `document.styleSheets[0]` 可以获取 当前页面的 `style` 样式

```
var style = document.styleSheet[0]
```

  - `style.insertRule( rule, index )`
    - `rule` ： 要添加到样式表的规则。
      1. `'@keyframes box{...}'`

2. `'#box{ ... }'`

- `index`：要把规则插入或附加到 `cssRules` 数组中的位置。
  - 一般使用`0`, 方便删除

- `style.deleteRule(index)`：删除对应下标的 样式表规则
- `style.rules`：所有样式