Web APIs简介

Web APIs 和 js 基础关联性

js的组成

ECMAscript（JavaScript基础）、DOM（页面文档对象模型）+BOM（浏览器对象模型）→WEB APIs

Web APIs是W3C组织的标准，是js所独有的部分

API和Web API

API

Application Programming Interface，应用程序编程接口，是一些预先定义的函数，目的是提供应用程序与开发人员基于某软件或者硬件得以访问的一组例程的能力，而又无需访问源码，或理解内部工作机制的细节。

简而言之，API就是程序员提供的一种工具，以便能够更加轻松的实现想要的功能

Web API

是浏览器提供的一套操作浏览器功能和页面元素的API（BOM和DOM），主要针对浏览器做交互效果

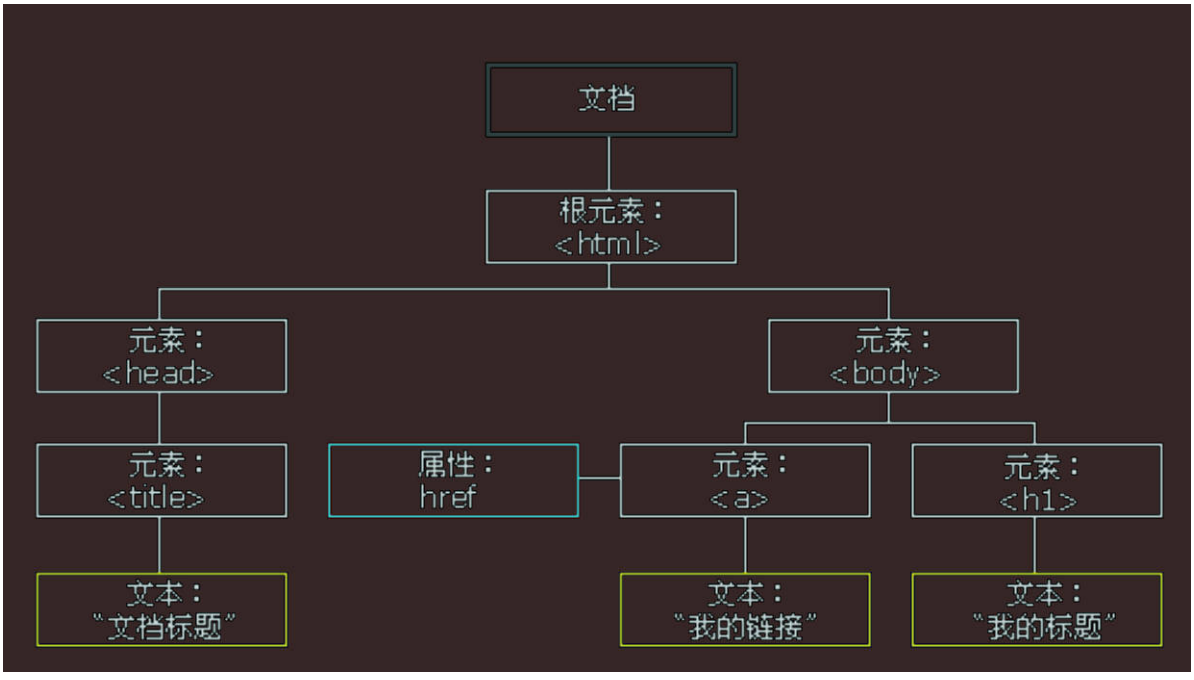
MDN 详细 API : https://developer.mozilla.org/zh-CN/docs/Web/API

DOM

简介

文档对象模型（Document Object Model，DOM）是W3C组织推荐的处理可扩展标记语言html或者xml的标准编程接口。

W3C 已经定义了一系列的 DOM 接口，通过这些 DOM 接口可以改变网页的内容、结构和样式。



文档：一个页面就是一个文档，DOM中使用document表示

元素：页面中的所有标签都是元素，DOM中使用element表示

节点：网页中的所有内容都是节点（标签、属性、文本、注释等）DOM使用node表示

DOM把以上内容看作对象

获取对象

根据ID获取

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

使用console.dir()可以打印我们获取的元素对象，更好地查看对象里面的属性和方法。

根据标签名获取

```
document.getElementsByTagName('标签名');
```

ps

因为得到的是一个对象的集合，所以我们想要操作里面的元素就需要遍历。

得到元素对象是动态的

. 如果获取不到元素,则返回为空的伪数组(因为获取不到对象)

还可以获取某个元素（父元素）内部所有指定标签名的子元素

```
element.getElementsByTagName('标签名');
```

ps

父元素必须是单个对象（必须指明是哪一个元素对象），获取的时候不包括父元素自己。

通过html5新增的方法获取

```
document.getElementsByClassName('类名'); // 根据类名返回元素对象集合
document.querySelector('选择器'); // 根据指定选择器返回第一个元素对象
document.querySelectorAll('选择器'); // 根据指定选择器返回
```

ps

querySelector 和 querySelectorAll里面的选择器需要加符号,比如:document.querySelector('#nav');

获取特殊元素（body，html）

```
// 获取body
document.body
// 获取html元素
document.documentElement
```

事件基础

事件概述

JavaScript 使我们有能力创建动态页面，而事件是可以被 JavaScript 侦测到的行为。

简单理解： 触发--- 响应机制。

网页中的每个元素都可以产生某些可以触发 JavaScript 的事件，例如，我们可以在用户点击某按钮时产生一个 事件，然后去执行某些操作。

事件三要素

1. 事件源 （谁）
2. 事件类型 （什么事件）
3. 事件处理程序 （做啥）

鼠标事件	触发条件
onclick	鼠标点击左键触发
onmouseover	鼠标经过触发
onmouseout	鼠标离开触发
onfocus	获得鼠标焦点触发
onblur	失去鼠标焦点触发
onmousemove	鼠标移动触发
onmouseup	鼠标弹起触发
onmousedown	鼠标按下触发

操作元素

JavaScript 的 DOM 操作可以改变网页内容、结构和样式，我们可以利用 DOM 操作元素来改变元素里面的内 容 、属性等。注意以下都是属性

改变元素内容

```
// 从起始位置到终止位置的内容，但它去除 html 标签， 同时空格和换行也会去掉
element.innerText
// 起始位置到终止位置的全部内容，包括 html 标签，同时保留空格和换行
element.innerHTML
```

显示时间案例

```
function getDate() {
    var date = new Date();
    var year = date.getFullYear();
    var month = date.getMonth() + 1;
    var dates = data.getDate();
    var arr = ['星期日', '星期一', '星期二', '星期三', '星期四', '星期五', '星期六'];
    var day = date.getDay();
    return '今天是：' + year + '年' + month + '月' + dates + '日 ' + arr[day];
}
var div = document.querySelector('div');
div.innerText = getDate();
div.innerHTML = getDate();
```

常用元素的属性操作

1. innerText、innerHTML 改变元素内容
2. src、href
3. id、alt、title

分时显示不同图片和问候语案例

```
var date = new Date();
var h = date.getHours(); // 24小时制
```

表单元素的属性操作

type、value、checked、selected、disabled

京东密码案例

点击眼睛按钮，把密码框类型从password改为text就可以看见里面的代码了

设置一个flag变量，点击一次就切换flag

样式属性操作

我们可以通过 JS 修改元素的大小、颜色、位置等样式。

1. element.style 行内样式操作
2. element.className 类名样式操作

ps

1. JS 里面的样式采取驼峰命名法 比如 fontSize、 backgroundColor
2. JS 修改 style 样式操作，产生的是行内样式，CSS 权重比较高

```
// box是类名
var btn = document.querySeletor('.box');
btn.onclick = function () {
    // 点击之后消失
    btn.style.display = 'none';
}
```

精灵图案例

利用for循环，修改精灵图片的背景位置 background-position

```
let lis = document.querySelectorAll('li');
    for(var i = 0;i<lis.length;i++) {
      var index = i*44;
      // backgroundPosition的取值有两个，一个是x 一个是y坐标
      lis[i].style.backgroundPosition = '0 -' + index +'px';
    }
```

div属于块元素，需要设置宽高才会显示

- js中let与var的区别

  作用域

  var的作用域被规定为一个函数作用域，而let则被规定为块作用域，但是如果两者既没有在函数中，也没有在块作用域中定义，那么两者都属于全局作用域

  - 全局作用域

    var 和 let在全局作用域中被定义时，两者非常相似

    但是被let声明的变量不会作为全局对象window的属性，而被var声明的变量可以

    ```
    let bar = 'hehe';
    var baz = 'lala';

    // undefined
    console.log(window.bar);
    // 'able'
    console.log(window.baz);
    ```

  - 函数作用域

    var和let在函数作用域中声明一个变量，两个变量的意义是相同的

  - 块作用域

    let只在fot循环中可用，而var是对于包围for循环的整个函数可用

    ```
    function  aFun1(){
        // i 对于for循环外的范围是不可见的(i is not defined)
        for(let i = 1; i<5; i++){
            //  i只有在这里是可见的
        }
        // i 对于for循环外的范围是不可见的(i is not defined)
    }
    function aFun2(){
        // i 对于for循环外的范围是可见的
        for(var i = 1;i<5; i++){
            // i 在for 在整个函数体内都是可见的
        }
        // i 对于for循环外的范围是可见的
    }
    ```

let和var重新声明

var允许在同一作用域中声明同名的变量，而let不可以

```
let me  = 'foo';
let me  = 'bar'; //SyntaxError: Identifier 'me' has already been declared

var me = 'foo';
var me = 'bar'; //这里me被替代了，是可以重复声明的
```

let在块作用域中有效，有的时候，为了降低变量污染的风险，在块作用域中使用let来代替var，这样不会污染块作用域的外部作用域，降低bug，使代码更加安全。

显示隐藏文本框案例

当鼠标点击文本框时，里面的默认文字隐藏，当鼠标离开文本框时，里面的文字显示。

```
<input type="text" value="手机">
  <script>
    let txt = document.querySelector('input');
    txt.onfocus = function() {
      console.log('得到了焦点');
      if(txt.value == '手机')
        txt.value = '';
      this.style.color = '#333';
    }
    txt.onblur = function() {
      console.log('失去了焦点');
      if(txt.value === '') {
        txt.value = '手机';
      }
      this.style.color = '#999';
    }
```

颜色等属性需要加上style，且值需要在''内部 但是type和value等值可以不用style，因为它们就是属性，而不是样式

获得焦点onfocus

失去焦点onblur

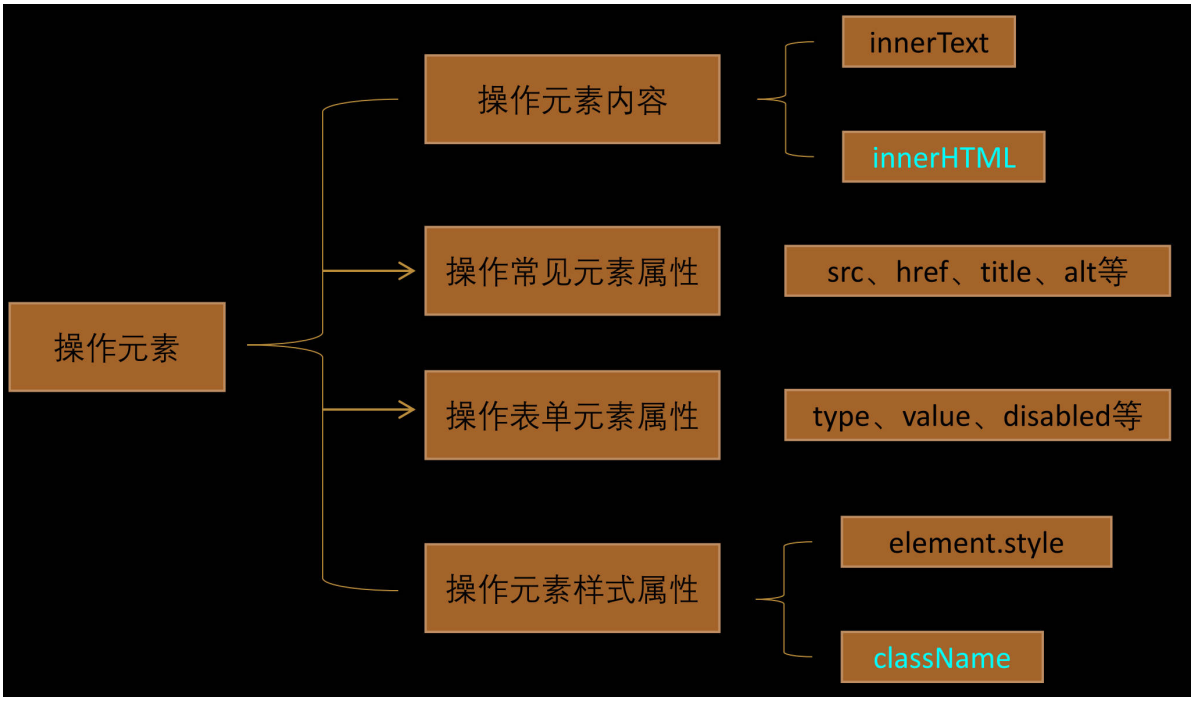
样式属性操作

1. element.style 行内样式操作
2. element.className 类名样式操作

上面讲的是style行内样式操作，如果样式修改较多，可以采取操作类名方式更改元素样式

class因为是个保留字，因此使用className来操作元素类名属性

className 会直接更改元素的类名，会覆盖原先的类名。



操作元素是DOM核心内容

排他思想

如果有同一组元素，我们想要某一个元素实现某种样式， 需要用到循环的排他思想算法：

1. 所有元素全部清除样式（干掉其他人）
2. 给当前元素设置样式 （留下我自己）
3. 注意顺序不能颠倒，首先干掉其他人，再设置自己

tab栏切换

本次案例难点：就是将选项卡上的内容与下面的内容一一进行对应

解决思路：给选项上的tab_list中的li添加自定义属性，属性值从0开始编号

当我们点击tab_list中的某个li时，让对应序号内容进行显示，其余隐藏（排他思想）

H5自定义属性

目的：是为了保存并使用数据，有些数据可以保存到页面中而不用保存到数据库中

通过getAttribute('属性')获取

为了更好地区分到底是内置属性还是自定义属性，H5新增了自定义属性：

1. 设置H5自定义属性

   自定义属性使用data-开头作为属性名并且赋值

   ```
   <div data-index= '1'></div>
   ```

   ```
   // 或者使用js设置
   element.setAttribute('data-index',2)
   ```

2. 获取H5自定义属性

```
element.getAttribute('data-index');
// 或者
element.dataset.index
// 或者(ie 11才开始支持)
element.dataset['index']
```

节点操作

获取元素的两种方式

1、利用DOM提供的方法

document.getElementById()

document.getElementByTagName()

document.querySelector等

逻辑性不强、繁琐

2、利用节点层级关系获取

利用父子兄节点关系获取

逻辑性强，但是兼容性稍差

节点概述

网页中的所有内容都是节点（标签、属性、文本、注释等），在DOM中，使用node来表示

基本属性

nodeType（节点类型）

nodeName（节点名称）

nodeValue（节点值）

● 元素节点 nodeType 为 1

● 属性节点 nodeType 为 2

● 文本节点 nodeType 为 3 （文本节点包含文字、空格、换行等）

实际开发中，节点操作主要操作的是元素节点

节点层级

利用ＤOM树可以把节点划分为不同的层级，常见的是父子兄关系

父级节点

ｎｏｄｅ．ｐａｒｅｎｔＮｏｄｅ

ｐａｒｅｎｔＮｏｄｅ属性可以返回某节点的父节点，注意是最近的一个父节点

如果指定的节点没有父节点则返回ｎｕｌｌ

子节点

1、parentNode.childNodes（标准）

parentNode.childNodes返回包含指定节点的子节点的集合，该集合为即时更新的集合

注意：返回值里面包含了所有的子节点，包括元素节点，文本节点等。

如果只想要获得里面的元素节点，则需要专门处理。 所以我们一般不提倡使用childNodes

```
var ul = document. querySelector('ul');
for(var i = 0; i < ul.childNodes.length;i++) {
    if (ul.childNodes[i].nodeType == 1) {
        // ul.childNodes[i] 是元素节点
        console.log(ul.childNodes[i]);
    }
}
```

2、parentNode.children（非标准）

parentNode.children是一个只读属性，返回所有的子元素节点，它只返回子元素节点，其余节点不返回

3、parentNode.firstChild

firstChild返回第一个子节点，找不到返回null，同样，包含了所有的节点

4、parentNode.lastChild

lastChild返回最后一个子节点，找不到返回null，同样，包含了所有节点

5、parentNode.firstElementChild

返回第一个子元素节点，找不到则返回null。这个方法有兼容性问题，IE9 以上才支持。

6、parentNode.lastElementChild

返回最后一个子元素节点，找不到则返回null。这个方法有兼容性问题，IE9 以上才支持。

注意

实际开发中，firstChild 和 lastChild 包含其他节点，操作不方便，而 firstElementChild 和 lastElementChild 又有兼容性问题，那么我们如何获取第一个子元素节点或最后一个子元素节点呢?

解决

1. 如果想要第一个子元素节点，可以使用 parentNode.chilren[0]
2. 如果想要最后一个子元素节点，可以使用 parentNode.chilren[parentNode.chilren.length - 1]

案例：下拉菜单

鼠标经过：onmouseover onmouseout

兄弟节点

1、node.nextSibling

返回当前元素的下一个兄弟元素节点，找不到则返回null。同样，也是包含所有的节点。

2、node.previousSibling

返回当前元素上一个兄弟元素节点，找不到则返回null。同样，也是包含所有的节点。

3、node.nextElementSibling

返回当前元素下一个兄弟元素节点，找不到则返回null。这个方法有兼容性问题， IE9 以上才支持。

4、node.previousElementSibling

返回当前元素上一个兄弟节点，找不到则返回null。这个方法有兼容性问题， IE9 以上才支持。

注意

如何解决兼容性问题

解决

自己封装一个兼容性的函数

```
function getNextElementSibling(element) {
    var el = element;
    while(el = el.nextSibling) {
        if(el.nodeType === 1) {
            return el;
        }
    }
    return null;
}
```

创建节点

document.createElement('tagName')

创建由 tagName 指定的 HTML 元素。因为这些元素原先不存在， 是根据我们的需求动态生成的，所以我们也称为动态创建元素节点。

添加节点

1、node.appendChild(child)

将一个节点添加到指定父节点的子节点列表末尾，类似于css中的after伪属性

2、node.insertBefore(child,指定元素)

将一个节点添加到父节点的指定子节点前面，类似于css里的before伪元素

案例：留言发布案例

难点：点击按钮之后需要动态建立一个li，添加到ul里面

创建li的同时，把文本域里面的值通过li.innerHTML赋值给li

添加到留言板上，appendChild或者insertBefore

删除节点

node.removeChild(child)

从DOM中删除一个子节点，返回删除的节点

案例：删除留言

添加li时，需要多添加一个删除的连接

阻止连接跳转时需要添加javascript:void(0);或者javascript:;

复制节点（克隆节点）

node.cloneNode()

返回调用该方法的节点的一个副本

注意： 1. 如果括号参数为空或者为 false ，则是浅拷贝，即只克隆复制节点本身，不克隆里面的子节点。 2. 如果括号参数为 true ，则是深度拷贝，会复制节点本身以及里面所有的子节点。

案例：动态生成表格

数据的话采用对象形式存储，所有的数据都是放到tbody里面的行

循环创建行，在行里面需要循环创建单元格，并将数据存入里面

最后一列单元格是删除，需要单独创建单元格

最后添加删除操作，单击删除，可以删除当前行（删除的是a元素节点父节点的父节点）

节点操作

document.write()

直接将内容写入页面的内容流，但是文档流执行完毕，则它会导致页面全部重绘

element.innerHTML

是将内容写入某个 DOM 节点，不会导致页面全部重绘，创建多个元素效率更高（不要拼接字符串，采取数组形式拼接），结构稍微复杂

document.createElement()

创建多个元素效率稍低一点点，但是结构更清晰

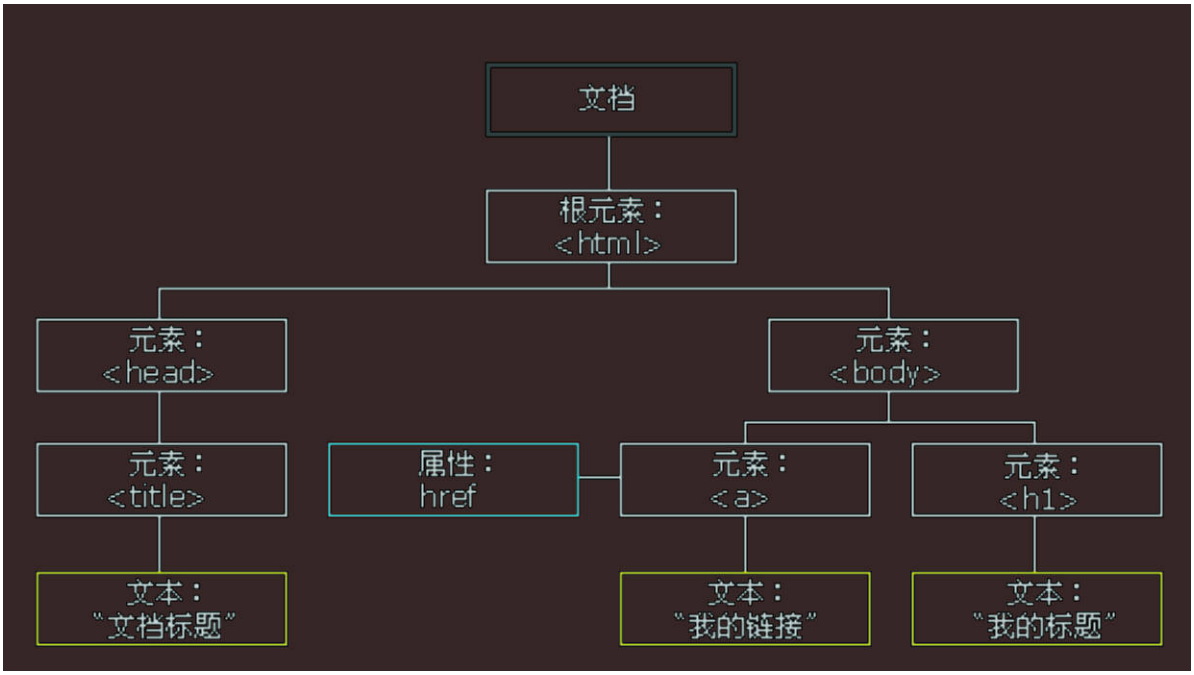
总结：不同浏览器下，innerHTML 效率要比 creatElement 高

DOM重点核心

文档对象模型（DOM，Document Object Model）是W3C组织推荐的处理可扩展标记语言（HTML或者XML）的标准编程接口

W3C 已经定义了一系列的 DOM 接口，通过这些 DOM 接口可以改变网页的内容、结构和样式。

1. 对于JavaScript，为了能够使JavaScript操作HTML，JavaScript就有了一套自己的dom编程接口。
2. 对于HTML，dom使得html形成一棵dom树. 包含 文档、元素、节点



我们获取过来的DOM元素是一个对象（object），所以称为文档对象模型

关于dom操作，主要有创建、增删改查、属性操作、事件操作

创建

document.write

innerHTML

document.createElement

增

appendChild

insertBefore

删

removeChild

改

主要修改dom的元素属性，dom元素的内容、属性，表单的值等

1. 修改元素属性：src、href、title等
2. 修改普通元素内容：innerText、innerHTML
3. 修改表单元素：value、type、disabled等
4. 修改元素样式：style、className

查

1. DOM提供的API方法：getElementById、getElementsByTagName 古老用法，不推荐
2. H5提供：querySelector、querySelectorAll 推荐
3. 利用节点操作获取元素：parentNode、children、previousElementSibling、nextElementSibling 推荐

属性操作

针对自定义属性

setAttrbute：设置dom的属性值

getAttribute：得到dom的属性值

removeAttribute：移除属性

事件操作

给元素注册事件，采取 事件源.事件类型 = 事件处理程序

鼠标事件	触发条件
onclick	鼠标点击左键触发
onmouseover	鼠标经过触发
onmouseout	鼠标离开触发
onfocus	获得鼠标焦点触发
onblur	失去鼠标焦点触发
onmousemove	鼠标移动触发
onmouseup	鼠标弹起触发
onmousedown	鼠标按下触发