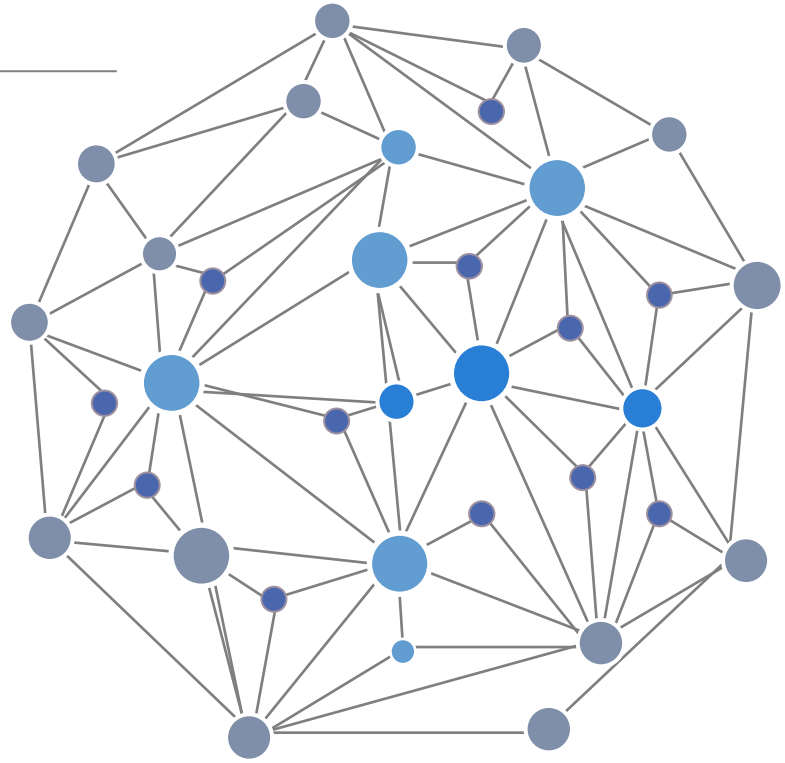
Web 程序设计



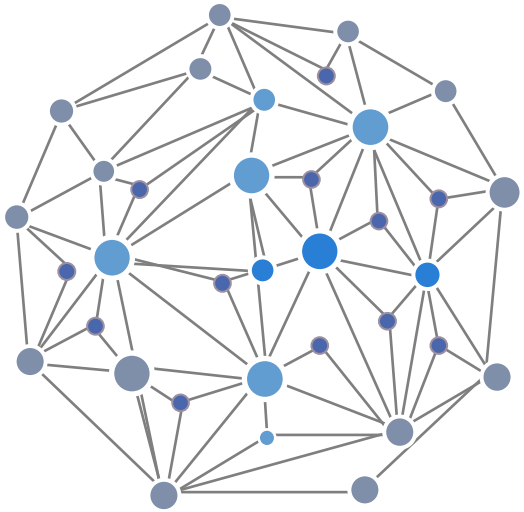
第七讲

Document Object Model

福州大学 计算机与大数据学院
软件工程系 陈昱

DOM 编程基础

- DOM 文档对象模型
- 事件与事件处理
- 行为与结构的分离
- 行为与表现的分离
- 杂项



Document Object Model

文档对象模型

为什么需要 DOM

- 之前我们介绍过：
  通过JavaScript可以调整HTML文档的结构
  - 添加、移除、改变或重排页面上的元素

- 要改变页面上的任何东西，JavaScript 需要可以访问 HTML 文档里的元素

什么是 DOM

- 文档对象模型（DOM）的工作就是提供编程接口允许这种访问

- DOM = Document Object Model

  DOM 定义了访问和处理 HTML/XML 文档的标准方法

什么是 DOM

- DOM 是一个与平台和语言无关的编程接口
- 它可以被任何的程序语言所使用
  - Java, .NET, PHP, Perl, VBScript or JavaScript
  - 不过最常使用 DOM 的语言就是 JavaScript

DOM 能够做什么

- DOM 提供一系列 JavaScript 对象，用来描述页面上每个元素
- JS代码通过DOM操作着HTML页面上的元素
  - 我们可以遍历元素的状态
    - 例如检查一个选项框是否被选中
  - 我们可以改变元素的内容
    - 例如往一个 div 里面插入一些文本
  - 我们可以改变元素的样式
    - 例如使一段文字变成红色

DOM 能够做什么

- 可以在 JS 中通过 DOM 接口实现：
  - 读取并修改 HTML 标记中的文本和属性值
  - 删除文档中的 HTML 元素
  - 创建新的元素并插入到文档中
  - 读取并修改 HTML 元素的样式信息

- 也就是：可以完全在客户端修改一个页面

DOM 的发展史

- DOM level 0
  - 由 Netscape 开发并被 Microsoft 采用
  - 出于向后兼容性，仍然被支持
- 后来就爆发了浏览器大战，Netscape和微软的 DOM 接口开始不兼容
- W3C 出面制定 DOM 标准:DOM level 1/2/3
  - http://www.w3school.com.cn/w3c/w3c_dom.asp

DOM 标准浏览器支持情况

- DOM 0（不是正式标准）
  - Navigator 3.0/IE 3.0, 目前所有浏览器都支持
- DOM 1（W3C, 1998 年）
  - Mozilla, IE 5/6, Konqueror, Opera 5+
- DOM 2（W3C, 2000 年）
  - Mozilla, Safari, Opera (IE6/7仍不支持DOM2)
- DOM 3 （W3C, 2004 年)
- 版本之间的差别在于提供了不同的接口方法，版本越高，方法越多，功能越强大

DOM 举例

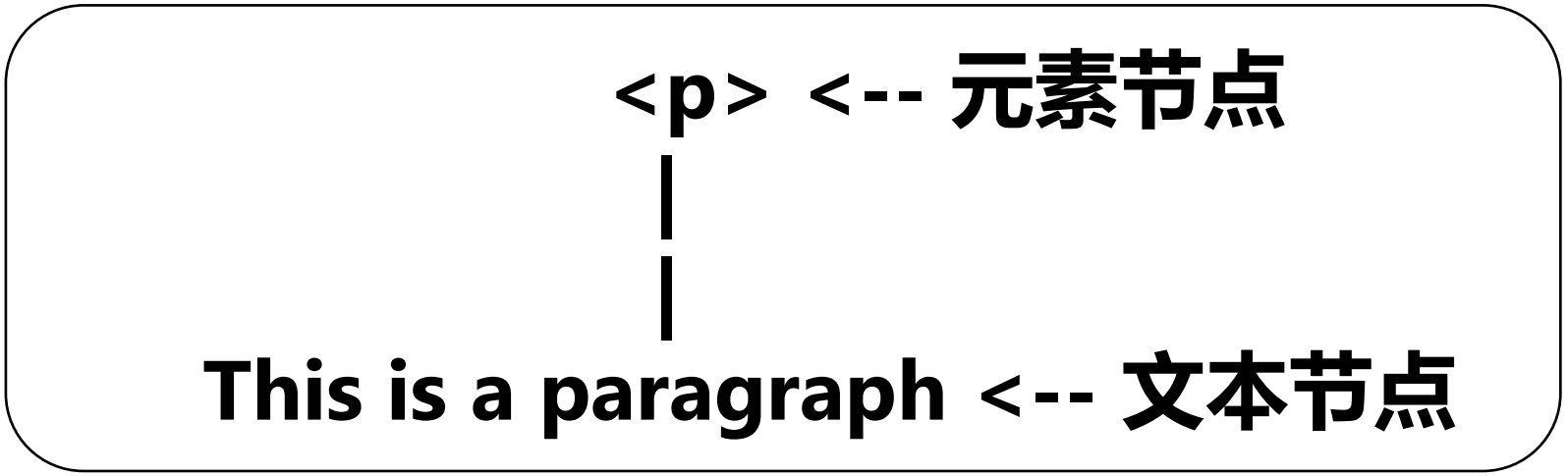
- DOM 是一个表示文档中不同的对象相互之间如何联系的模型 (model)

- 在 DOM 中，文档中的一切都是 DOM 树上的节点 (node)
  - 包括 元素节点，文本节点，属性节点 等
  - 节点之间是父子关系，构成一个树

DOM 举例

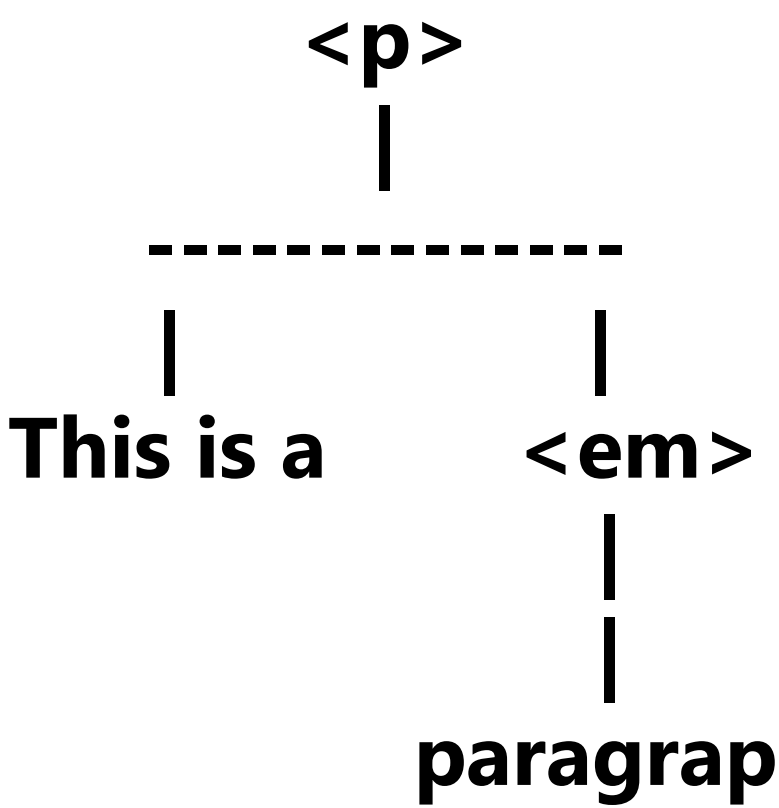
- 对下面的 HTML 代码，树中存在两个节点

```
<p>This is a paragraph</p>
```



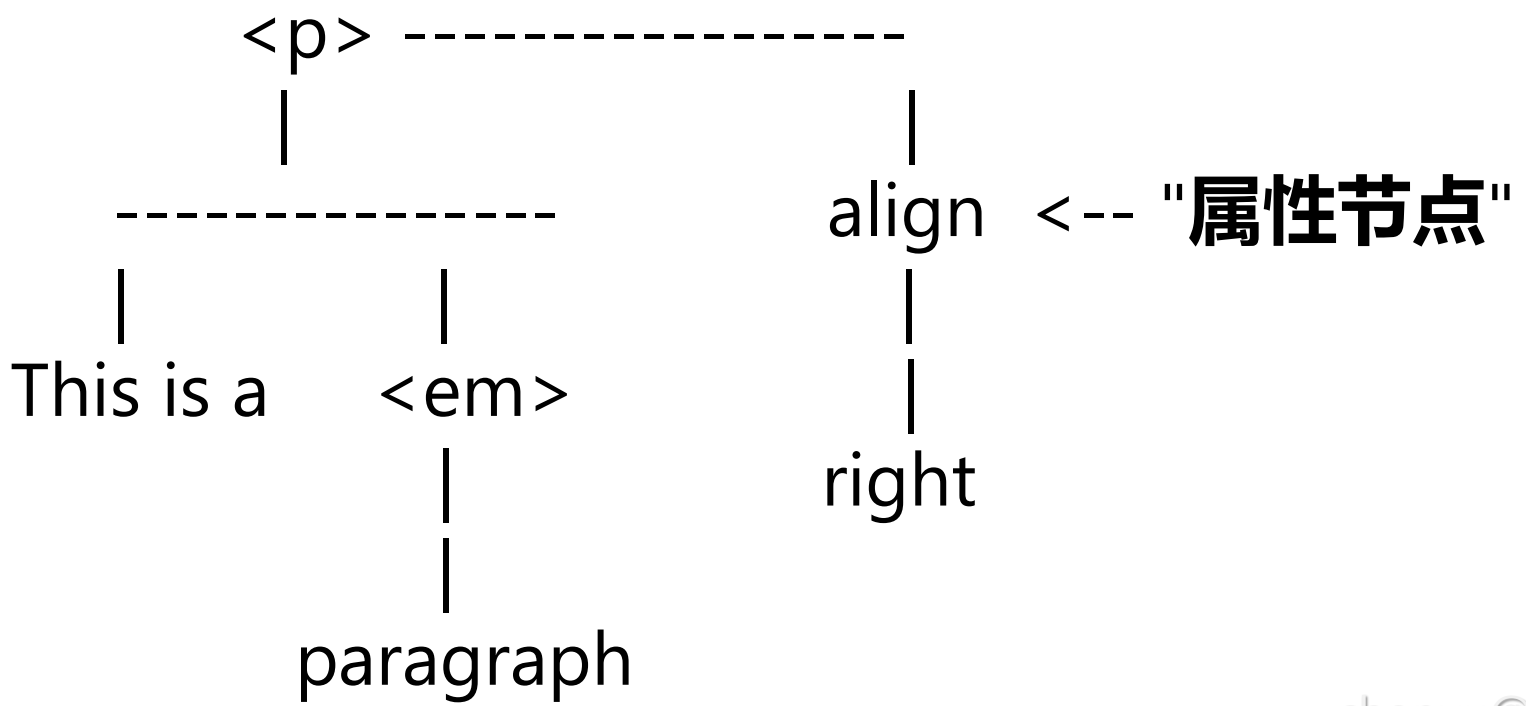
DOM 举例

```
<p>This is a <em>paragraph</em></p>
```



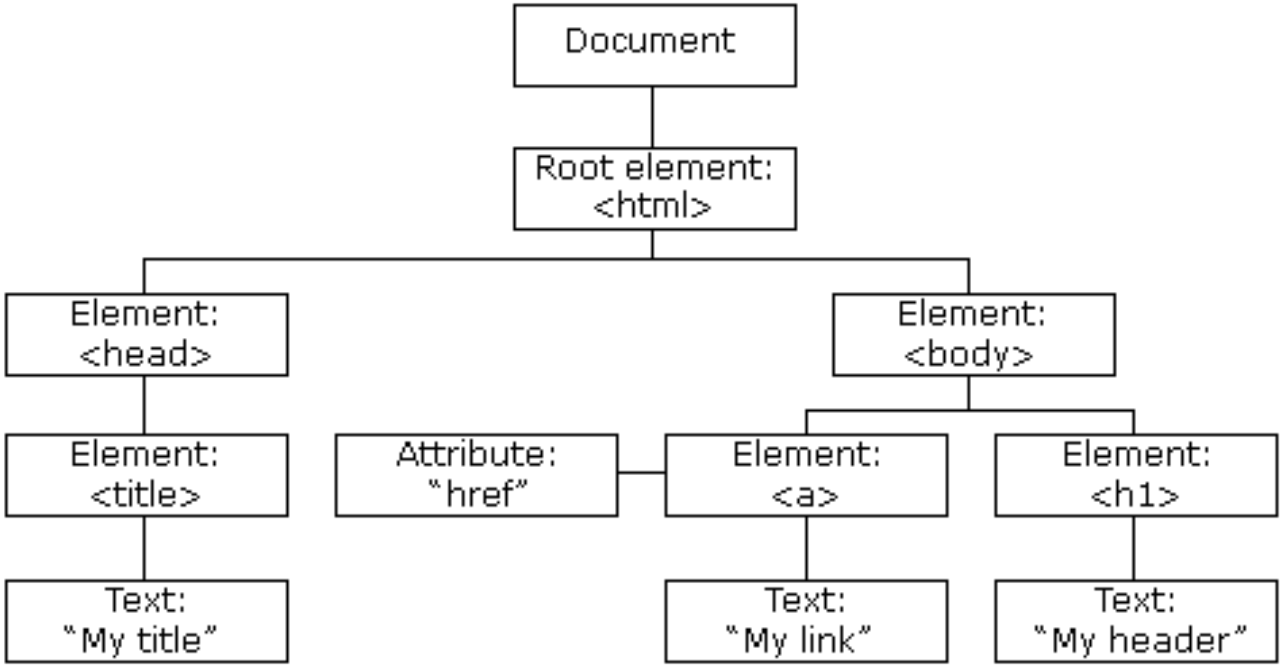
DOM 举例

```
<p align="right">This is a <em>paragraph</em></p>
```



DOM 举例

```
<html>
<head><title>My title</title></head>
<body>
<h1>My header</h1>
<a href="http://mydomain.com">My link</a>
</body></html>
```



DOM 节点(node)

- 根据 DOM，HTML/XML 文档中的一切事物都是节点(node)
- 根据 DOM 标准
  - 整个文档是一个 **document 节点**
  - 每个 HTML 元素是一个元素节点
  - 元素中包含的文本数据是文本节点
  - 每一个 HTML 属性是一个属性节点
    - 文本节点和属性节点都是元素节点的子节点
  - 注释是注释节点

DOM 树

- 节点之间互相关联，形成一棵树
  - 除了document节点，其他节点都有一个父节点(parent-node)
  - 除了叶子节点（一般是文本节点和属性节点），节点都有子节点（children-node)
  - 属于同一个父节点的节点称为兄弟节点(sibling-node)

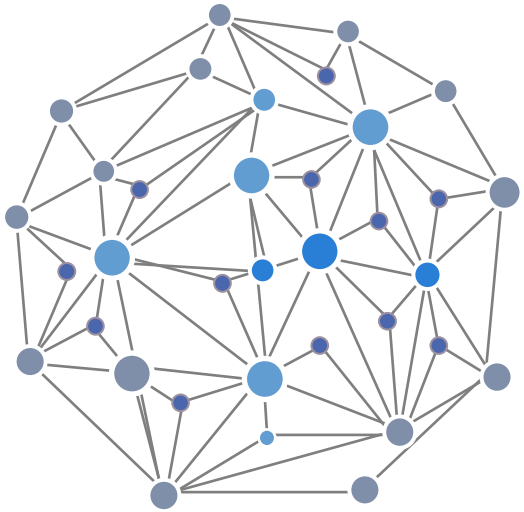
DOM 例子

- 树中的所有节点都是 JavaScript 中的对象
- 元素的属性变成节点对象的命名属性

- 例如：
  - <input type="text" name="address">
  - 表示这个节点的 JS 对象将有两个属性：
  - type 属性将会拥有值： "text"
  - name 属性将会拥有值： "address"

节点是对象

- 节点在 JS 作为对象访问，每个节点拥有许多的属性和方法
- 节点的一些属性
  - nodeName　报告节点的名称
  - nodeValue　提供节点的 “值”
  - nodeType　报告节点的类型
  - attributes　返回元素节点的属性列表
  - parentNode, childNodes, firstChild, lastChild,
  - previousSibling, nextSibling



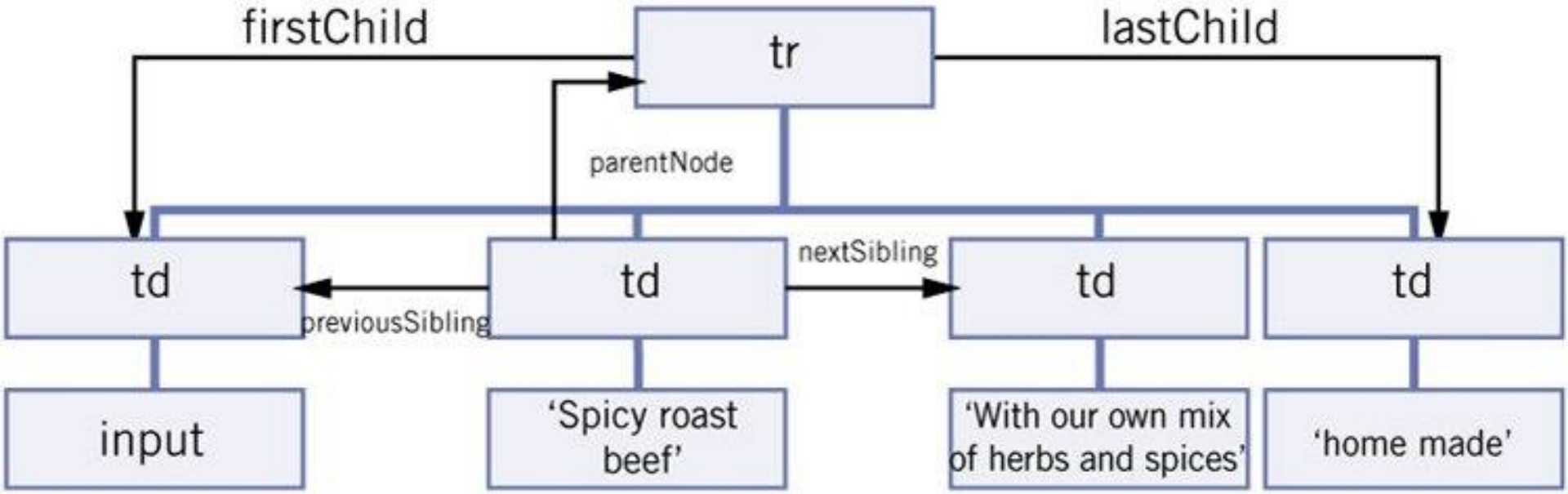
节点的查找与访问

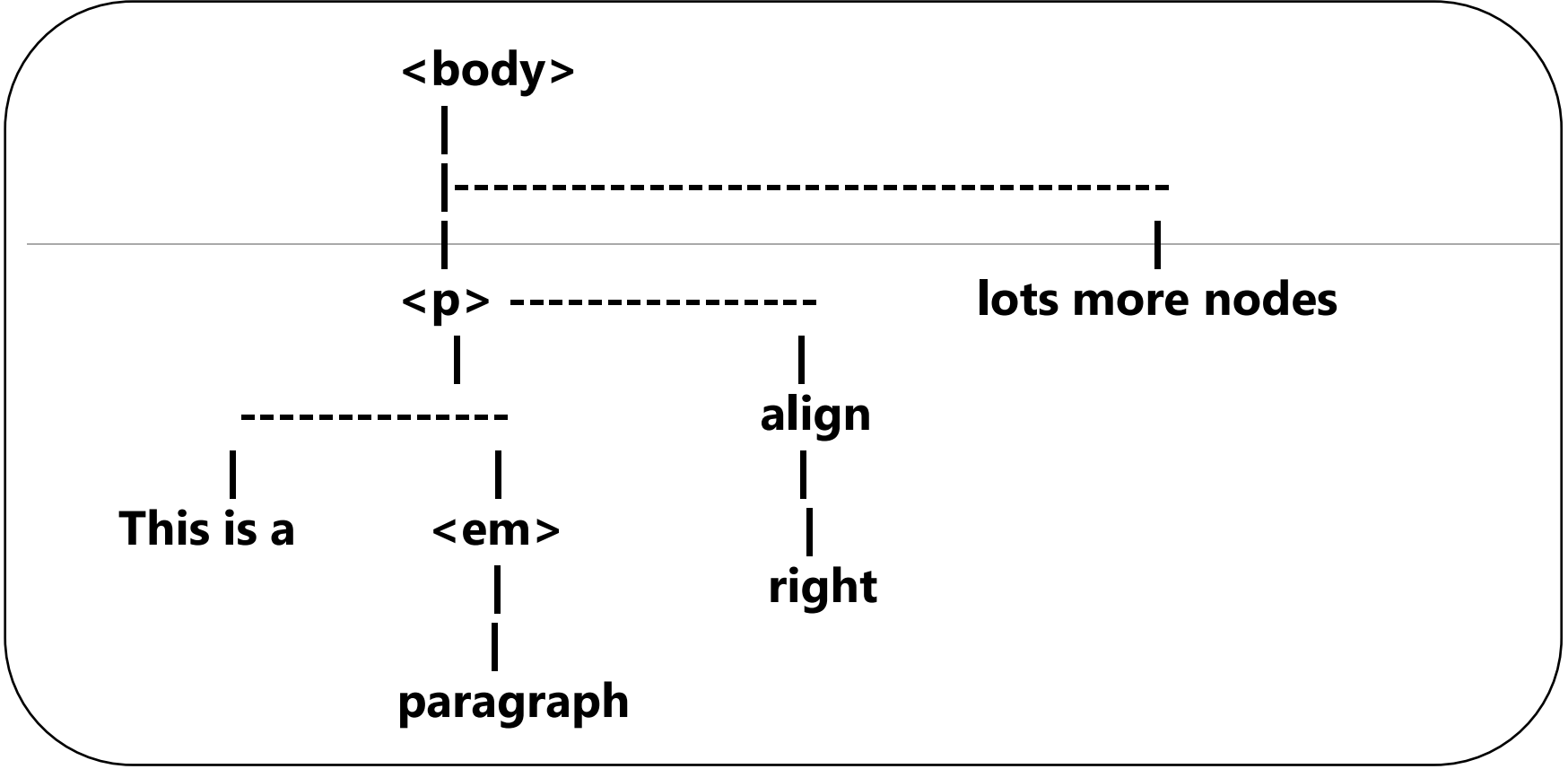
节点的查找与访问

- 在修改一个节点前，首先要找到它
- 对象的查找访问可以通过几种方式：
  - 使用元素节点的 parentNode, firstChild 和 lastChild 等表示关系的属性
  - 通过 document 对象的 getElementById(), getElementsByTag(), getElementsByClassName() 等方法

1、利用 DOM 树结构访问元素

- 使用元素节点的 **parentNode, firstChild** 和 **lastChild** 等属性





- 假设 节点 p 保存在 x 中，则可
- 通过 x.parentNode 访问 body
- 通过 x.childNodes[1] 访问?
- x.parentNode.firstChild.childNodes[1].lastChild; ?

2、使用 id 属性访问

- 通过为元素设置 id 属性来访问
- id 属性的值在文档中是唯一的

```
<form action = "">
    <input type="button"  id=“turnItOn">
</form>
```

- 然后使用 getElementById 方法得到对象
- document.getElementById(“turnItOn")

使用标记名访问

- 通过标记名和 getElementsByTagName () 方法获得元素
- 注意：同样标记名的元素可能有多个，因此方法返回的是一个节点列表

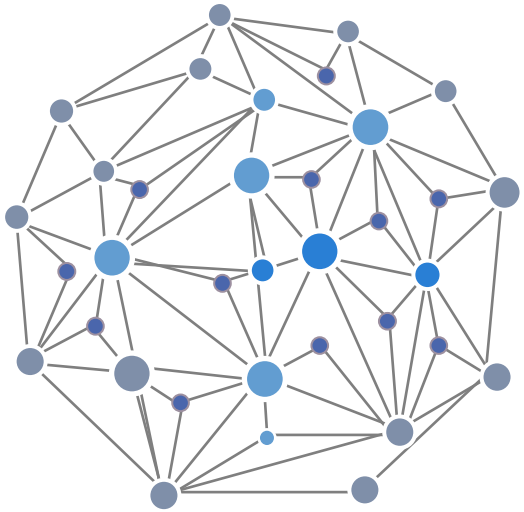
```
var pars = document.getElementsByTagName('p');

for (var i=0;i<pars.length;i++) {
// do something with pars[i]; ie. with every paragraph
}
```

通过类名访问

- 通过类名和 getElementsByClassName () 方法获得元素
- 注意：同样类名的元素可能有多个，因此方法返回的是一个节点列表

```
//查找 class="intro" 的元素
var x=document.getElementsByClassName("intro");
```



通过 DOM 改变 HTML

改变 HTML 内容

- 使用 innerHTML 属性改变 HTML 元素的内容

- document.getElementById(id).innerHTML=new HTML code

文本节点 Text Node 的修改

- 元素中包含的文本数据放在文本节点中
- 通过 **firstChild.nodeValue** 修改

```
<p id="test">I am a JavaScript hacker.</p>

var x = document.getElementById('test');
alert(x.firstChild.nodeValue);
x.firstChild.nodeValue = 'I never hack text nodes.';
```

属性的修改

- getAttribute()
- setAttribute()
- removeAttribute()

```
<img src="pix/logo.gif" id="test" />

var imgEl = document.getElementById('test');
alert(imgEl.getAttribute('src'));
imgEl.setAttribute('src', 'pix/logo2.gif');
```

元素样式的读取/修改

- 通过元素的 style 属性（对象），可以读取/修改元素的内联 CSS 样式
- 注意：只有内联样式才能够被读取和修改

```
<p style="margin: 10%" id="test">Text</p>
```

```
var p = document.getElementById('test');
alert(p.style.margin); // 读取
p.style.margin = '10px'; // 修改
p.style.fontSize = '120%'; // 内联样式优先级高
```

HTML DOM Style 对象

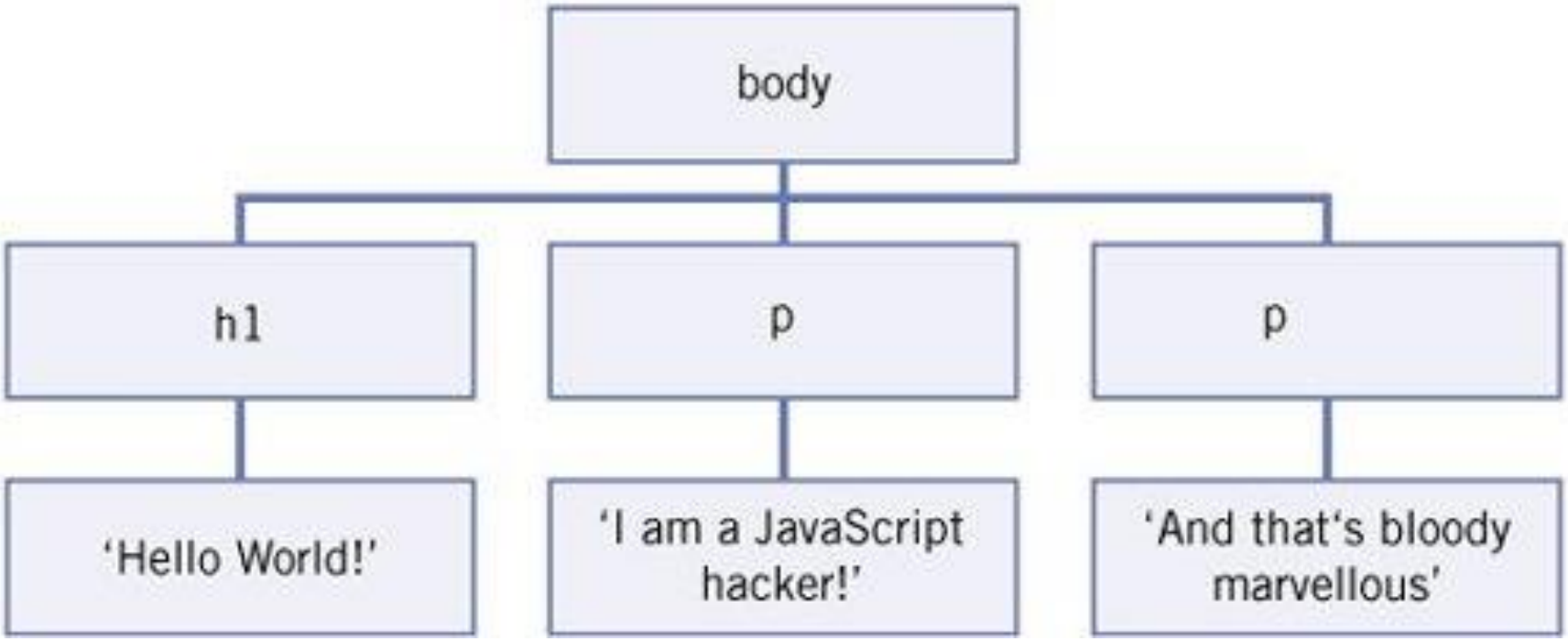
- 一个 HTMLElement 的 style 属性是一个可读可写的 CSS2Properties 对象
- 它为 CSS 规范定义的每一个 CSS 属性都定义一个 JavaScript 属性
- 参考
  - HTML DOM Style 对象 (w3school.com.cn)
- 访问元素通过计算得到的样式结果，使用 **Window.getComputedStyle() 方法**
  - Window.getComputedStyle() - Web API 接口参考 | MDN (mozilla.org)

DOM 树的修改

// 修改 DOM 树的方法

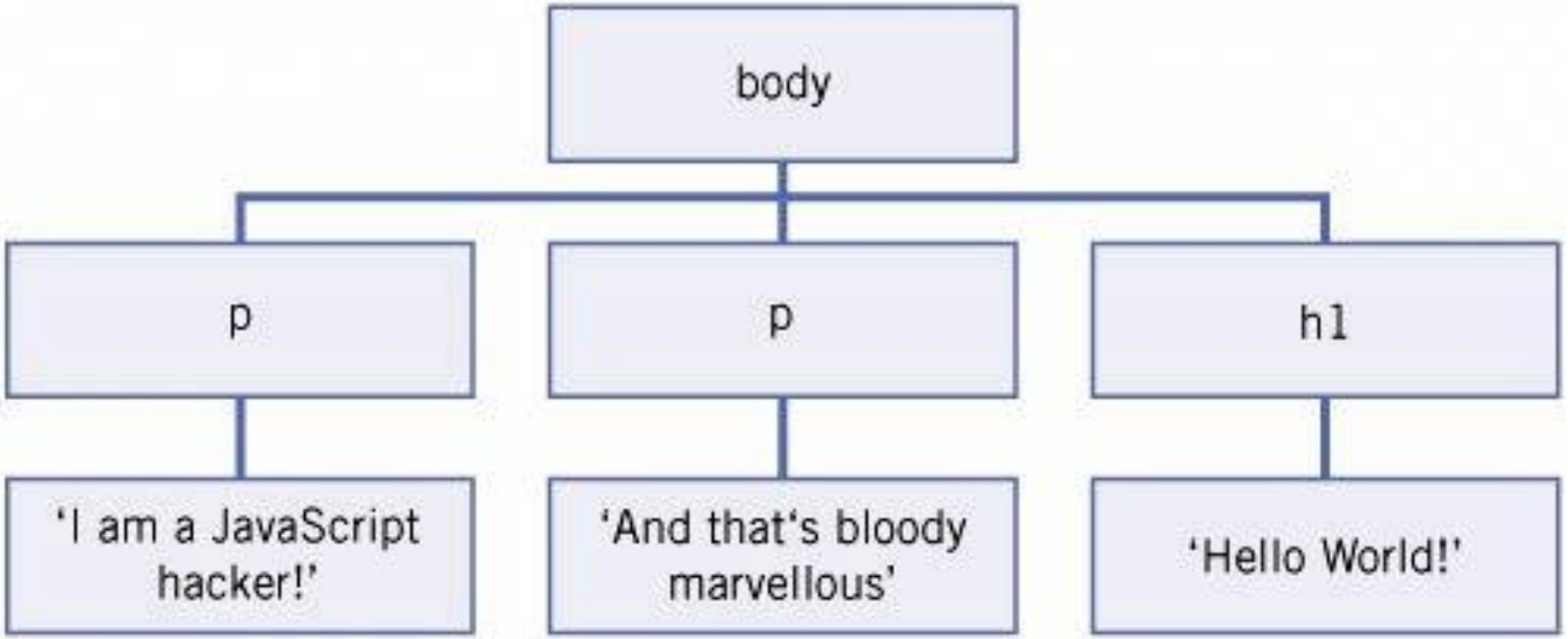
- createElement
  - var x = document.createElement('hr');
- createTextNode
- insertBefore(newChild, referenceNode)
- appendChild(newChild)
- replaceChild(newChild, oldChild)
- removeChild(oldChild)

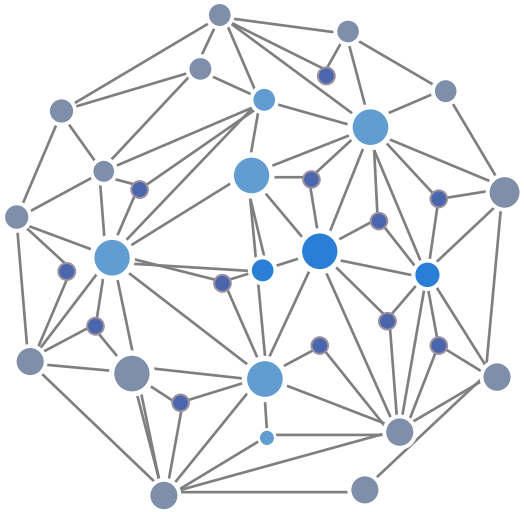
利用 DOM 修改文档




```
var x = document.getElementsByTagName('h1')[0];
x.parentNode.appendChild(x);
```

修改后的结果





事件与事件处理

关于事件，涉及到 DOM 兼容性
比较复杂

事件与事件处理

- 事件：
  - 浏览器中的活动，特别是用户的交互行为
  - 例如移动鼠标，敲击键盘

- 事件驱动编程
  ① 为事件指定事件处理函数
  ② 当事件发生时，执行指定的事件处理函数

事件响应

- 网页中的每个元素都可以产生某些可以触发 JavaScript 函数的事件

```
<input type="text" size="30" id="email"
                onchange="checkEmail()">

<a href="#top"
        onclick="alert ('wow. Javascript.')">
        Click me</a>
```

事件分类列表

- onclick
  - 鼠标点击一个元素时执行
- ondblclick
  - 鼠标双击一个元素时执行
- onmousedown
  - 按下鼠标按键时执行
- onmousemove
  - 鼠标光标在元素上移动时执行
- onmouseout
  - 鼠标光标移开元素时执行
- onmouseover
  - 鼠标光标移到元素上时执行
- onmouseup
  - 当释放鼠标按键时执行

- onkeydown
  - 按下某个按键时执行
- onkeypress
  - 按下和释放某个按键时执行
- onkeyup
  - 释放某个按键时执行

事件分类列表

- onchange
  - 用在表单元素中，当某些东西改变时执行
- onfocus
  - 用在表单元素中，当元素获得焦点时执行
- onblur
  - 用在表单元素中，当元素失去焦点的时候执行
- onreset
  - 用在表单元素中，当表单重置时执行
- onselect
  - 用在表单元素中，当元素被选择时执行
- onsubmit
  - 用在表单元素中，当表单提交时执行

- onload
  - 在body标签中使用，载入页面的时候执行
- onunload
  - 用在body标签中，当关闭页面时执行

注册事件响应函数的方法

- 将事件处理程序连接到事件的过程称为注册（register）：
  - 通过 HTML 元素属性注册
  - 通过 DOM 0 对象属性注册
  - 通过 DOM 2 事件机制注册

1、通过 HTML 元素属性注册

- 通过 HTML 标记的属性定义事件响应函数

```
<input type="text" size="30" id="email"
                onchange="checkEmail()" />

<a href="#top"
      onclick="alert ('wow. Javascript.')">
      Click me</a>
```

事件与标记属性对应表

Event	*Tag Attribute*
blur	onblur
change	onchange
click	onclick
focus	onfocus
load	onload
mousedown	onmousedown
mousemove	onmousemove
mouseout	onmouseout
mouseover	onmouseover
mouseup	onmouseup
select	onselect
submit	onsubmit
unload	onunload

2、通过 DOM 0 对象属性注册

- 通常的做法，先通过 getElementById 获得元素对象，再直接将函数名赋给对象属性

```
var x = document.getElementById('somewhere');
x.onclick = function_name;
```

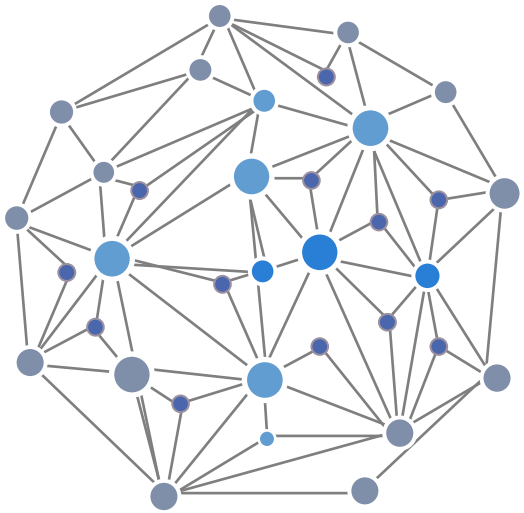
注意：

x.onclick = function_name(); 是错误的

3、通过 DOM 2 事件机制注册

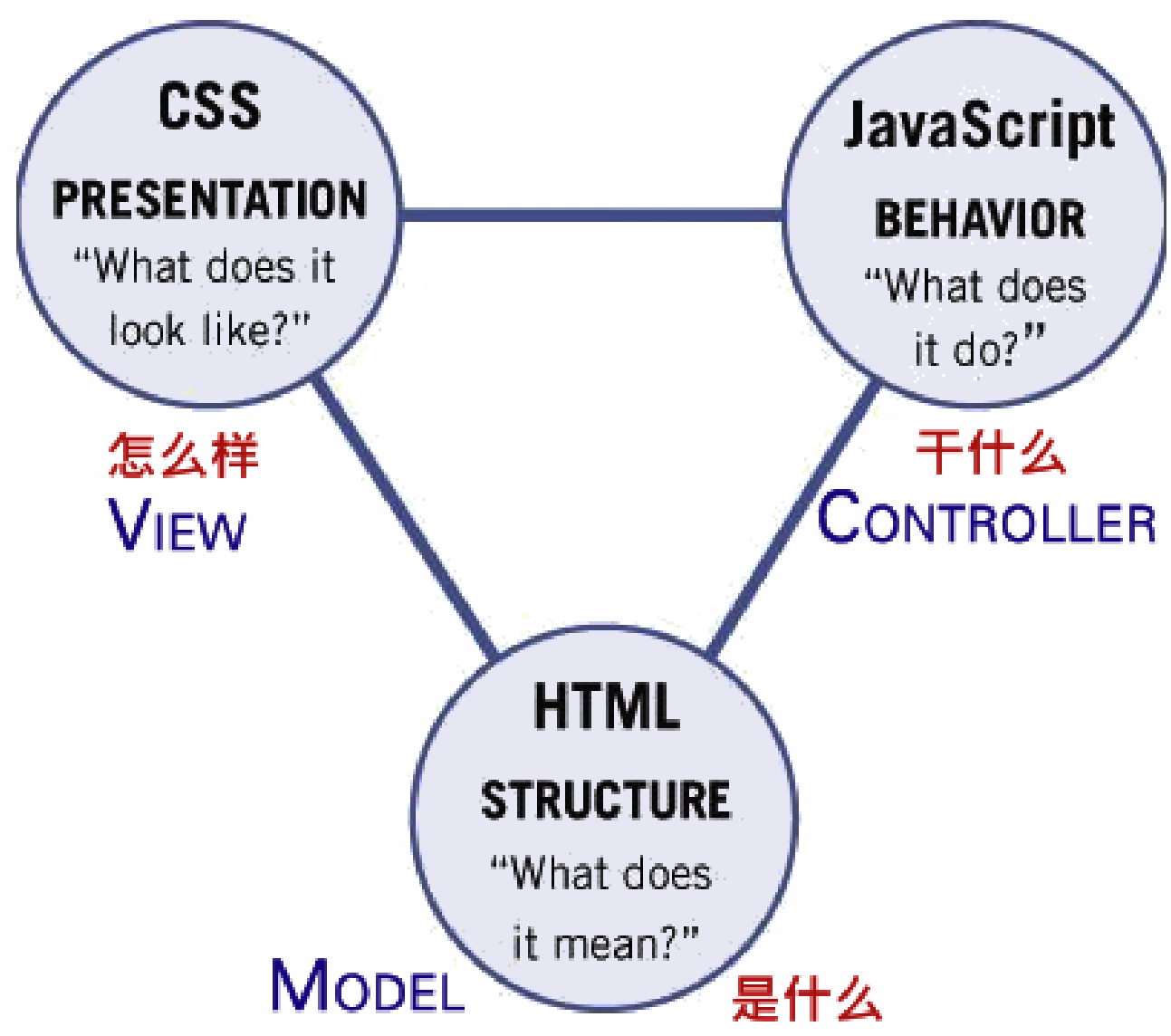
```
var x = document.getElementById('somewhere');
// W3C
x.addEventListener('click',handleEvent,false);
// Microsoft
x.attachEvent('onclick',handleEvent);

function handleEvent(e) {
    var evt = e || window.event;
    // do something with evt, which now
    // refers to the event object
}
```



行为与结构的分离

W3C 三层结构:客户端的"MVC"模式



网页的三层结构

- (X)HTML 结构层标签给予内容含义
- CSS 表现层定义文档该如何显示
- JavaScript 行为层为页面增加交互

- 一个网页必有结构层
- 没有结构层，没有网页
- CSS 和 JavaScript 都是可选项

分离行为和结构

- 分离行为与结构的原则很简单：**不要把任何 JavaScript 代码写入 XHTML 页面中。**

- 采取下面两步骤：
  - 把所有的 JavaScript 函数定义在一个分离的.js 文件中，让所需的 XHTML 页面连接到它。
  - 删除所有的事件处理句柄（即行内的那些诸如 onMouseOver）并归入同一 .js 文件中去

1、分离文件中的函数

- JavaScript 代码属于 .js 文件，而非 HTML 文件，所以这是不好的：

```
<script type="text/javascript">
    function doSomething()
    {
        // JavaScript code
    }
</script>
</head>
<body>
<h1>My HTML page</h1>
        ……
```

1、分离文件中的函数

```
</head>
<body>
<h1>My HTML page</h1>
```

```
// 定义在分离的nifty.js中
function doSomething()
{
    // JavaScript code
}
```

2、删除事件处理句柄

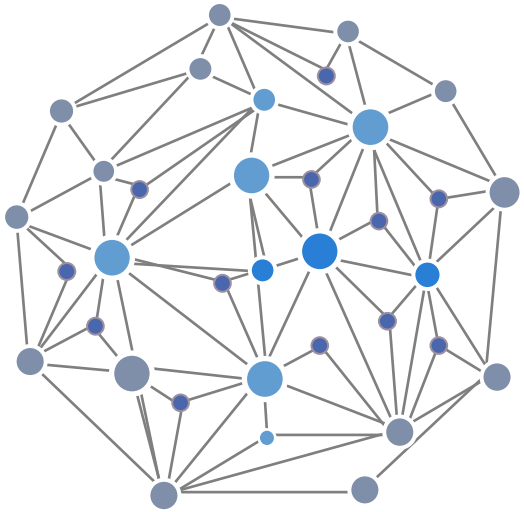
- 第二步是把所有(X)HTML内的JavaScript函数调用移到分离的.js中去。
  - 事实上，99%的(X)HTML内的JavaScript代码是行内事件句柄
- 比如下面的例子，句柄在(X)HTML内，但不应该属于(X)HTML：

```
<a href="home.html"
onMouseOver="mOver('home')"
onMouseOut="mOut('home')">Home</a>
```

2、删除事件处理句柄

```
<a href="home.html">Home</a>
```

```
// 在单独的 .js 文件
var nav = document.getElementById('navigation');
var navLinks = nav.getElementsByTagName('a');
for (var i=0;i<navLinks.length;i++)
{
    navLinks[i].onmouseover = [code];
    navLinks[i].onmouseout = [code];
}
```



行为与表现的分离

Tabs 的两种实现方式

- www.sina.com.cn
- www.microsoft.com

动态换页的两种实现方式

- ***sina.com.cn***
- 通过 js 修改两个div的内嵌样式属性
-

```
document.getElementById("BlkBlackTabcont
ent_10").style.display = "none";
document.getElementById("BlkBlackTabcont
ent_11").style.display = "block";
```

动态换页的两种实现方式

- ***microsoft.com***

XHTML

```
<div id="tbc0_0"
      class ="tabPanel tabActivePanel">
<div id="tbc0_1"
      class ="tabPanel tabHiddenPanel">
```

CSS

```
.tabHiddenPanel {
    display:none;
}
```

动态换页的两种实现方式

- ***microsoft.com***
- 通过 JS 修改元素的 className

JS

```
activeTabPanel.className =
        'tabPanel tabHiddenPanel';

newTabPanel.className =
        'tabPanel tabActivePanel';
```

行为与表现的分离

- 在 JS 中通过 style 属性修改样式理论上是个不好的行为

- 这意味着当有一天我们需要修改整个网站样式的时候，却需要去修改 JS 代码中的 CSS 属性

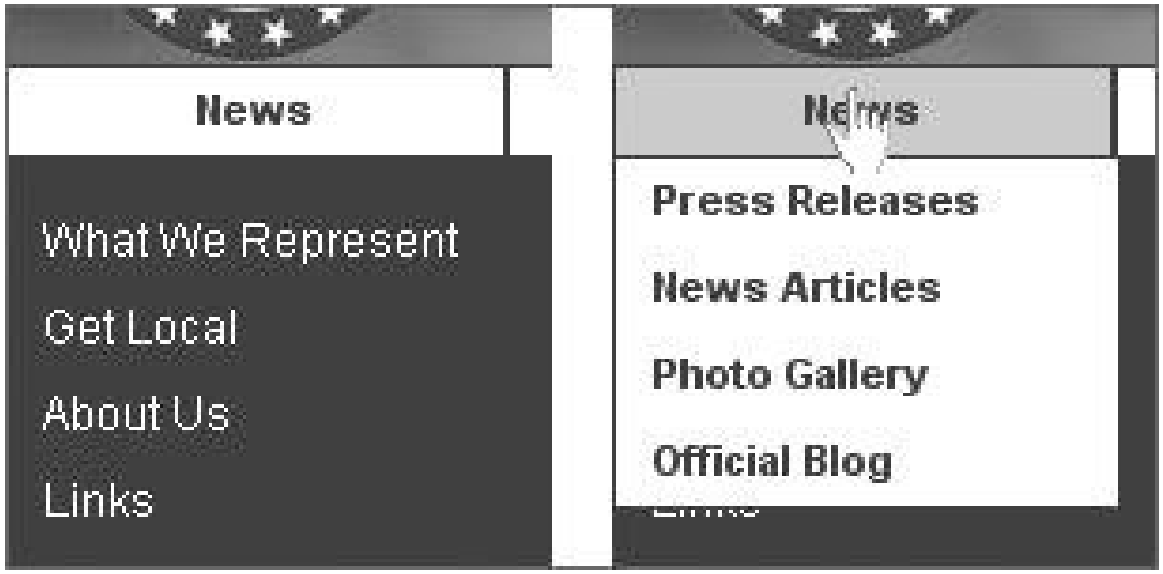
行为与表现的分离

- 解决方法：
  - 第一步，将样式细分成许多类，通过联合类属性指定样式

  - 第二步，用 JS 修改元素的 className

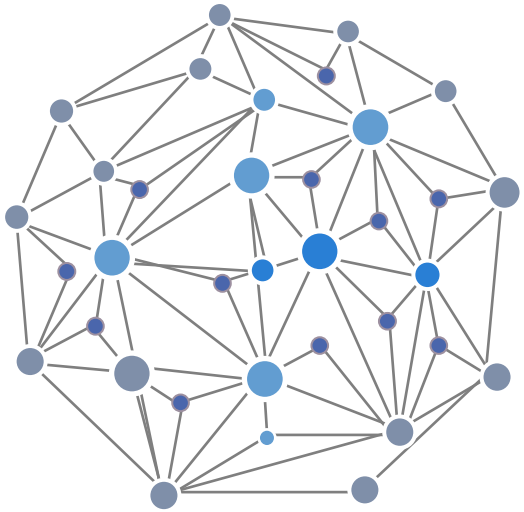
行为与表现的选择

- 行为和表现的选择也存在着许多争议
- 许多动态的效果既可以用 CSS 实现，又可以用JavaScript实现，比如下拉菜单
  - CSS 伪类 vs JavaScript 事件



分离的好处

- 分离有助于网站的方便维护
  - 当使用分离的CSS和JS文件时，就很容易把网站的所有页面链接到这些文件。这带来了维护性上的优势：对CSS文件做的修改，如字体大小，可以立即影响到所有指向这个CSS的网页
  - 可以轻松改变整个 CSS 表现层，给网站一个全新设计，而不需要重写 HTML 结构，也不用重写 JavaScript 行为层



JavaScript Framework

JavaScript 框架

JavaScript Framework

- JS Framework 提供了在进行 JS 和 Ajax 编程时常用的功能，包括许多辅助对象和函数，同时一定程度上封装了浏览器的差异

- 比如：$() 函数，是对 DOM 中使用频繁的 document.getElementById() 方法的一个便利的简写

JavaScript Framework

```
<html>
<head>
<script type="text/javascript"
        src="prototype.js"></script>

<script>
    function test1()
    {
        var d = $('myDiv');
        alert(d.innerHTML);
    }
```

var d = document.getElementById('myDiv');

$() 函数

- 当然自己也可以简单定义：

```
function $(id) {
    return document.getElementByid(id);
}
var dom = $("id");
```

流行的 JavaScirpt Framework

- 轻量级选择
  - JQuery
  - Mootools
  - Prototype
- RIA UI 开发
  - ExtJS
  - Dojo
  - JQuery UI
  - YUI
  - Script.aculo.us

JQuery

- 经过几年发展后已成为最流行的框架
- 可扩展性：有大量用户开发的插件可供使用
  - http://jquery.com/plugins/
- 提供界面库 jQuery UI，不断发展中
  - http://jqueryui.com/
- 常用网站
  - http://jquery.com/，http://jquery.org.cn/
  - 中文入门教程 http://www.k99k.com/jQuery_getting_started.html

用户界面构建

- React
- Vue.js
- AngularJS (Angular 1)
- Angular (原本的 Angular 2)

- 国内
  - Amaze UI

JavaScritp Review

- ECMAScript - JavaScript Core,基础语法
- BOM - 控制浏览器的接口
- DOM - 控制浏览器中的文档的接口
- 结构，表现，行为的分离

第七讲课后练习

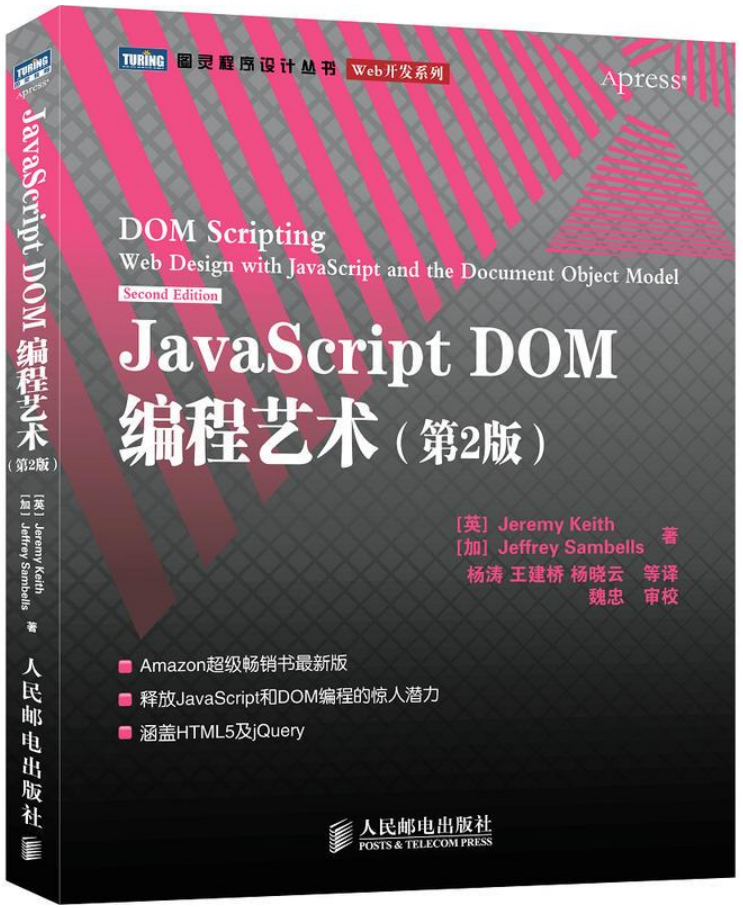
- 学习
  - 课本第18，19，20章
  - https://www.w3school.com.cn/htmldom/
- 课外阅读：
  - JavaScript DOM 编程艺术 (第2版)
  - DOM Scripting（英文版）
- 编程作业 hw5

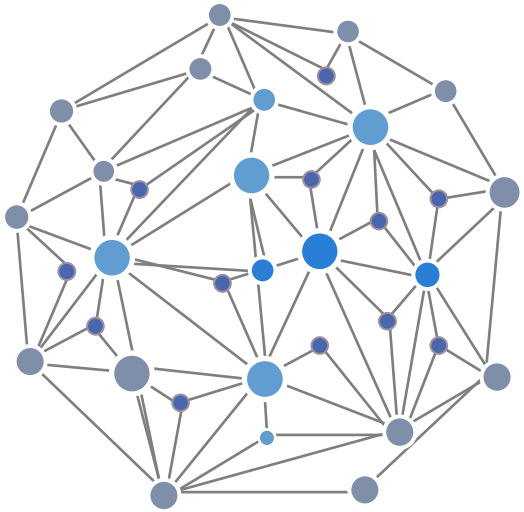
Extended Reading

- DOM Scripting:
  Web Design with JavaScript and the Document Object Model

- 中文版：JavaScript DOM 编程艺术 (第2版)

- 入门推荐





参考资料

W3C DOM 手册

- W3Schools 简明手册
  - https://www.w3school.com.cn/jsref/index.asp
- ppk 整理的 DOM 兼容性手册
  - https://www.quirksmode.org/compatibility.html

THANKS

本章结束

陈昱

福州大学 计算机与大数据学院 软件工程系