老马经典DOM与特效教程

@马伦-flydragon

课程内容简介

- 第一章：开发环境搭建
- **第二章：HTML5各种标签及语义化**
- 第三章：CSS3及整站开发
- 第四章：JavaScript基础
- 第五章：JavaScript高级
- 第六章：移动端开发[css3 + html5]
- 第七章：前端高级框架
- 第八章：NodeJs
- 第九章：前端工作流及工程化

老马自我介绍

老马联系方式

QQ：515154084

邮箱：malun666@126.com

微博：http://weibo.com/flydragon2010

百度传课：

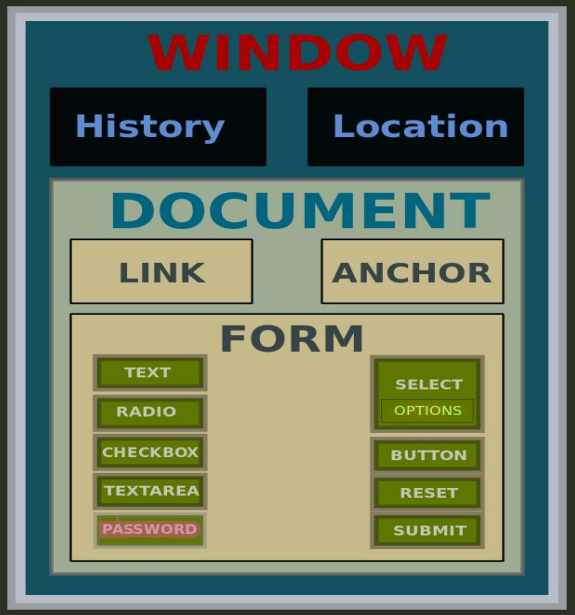
https://chuanke.baidu.com/s5508922.html

课程内容介绍

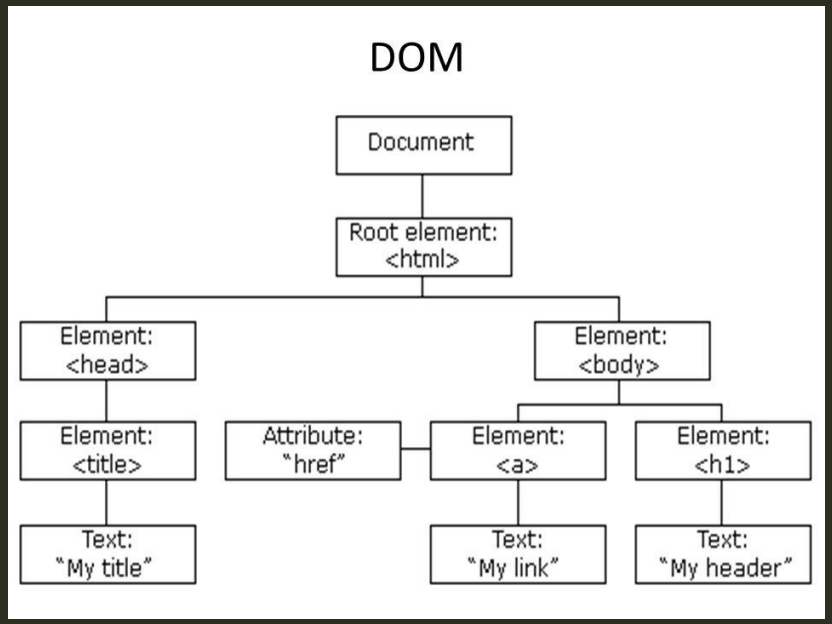
- BOM
- DOM
- 特效案例

第一节：DOM与BOM的概念

- BOM
  - BOM是浏览器对象模型，是Browser Object Model的缩写。
  - 它提供了一整套的从浏览器相关的前端页面相关的api。
- DOM
  - 就是**文档对象模型，是**Document Object Model的缩写。是对HTML文档页面进行编程控制的一系列接口API
  - 它提供了对文档的结构化的表述，并定义了一种方式可以使从程序中对该结构进行访问，从而改变文档的结构，样式和内容。DOM 将文档解析为一个由节点和对象（包含属性和方法的对象）组成的结构集合。简言之，它会将web页面和脚本或程序语言连接起来。

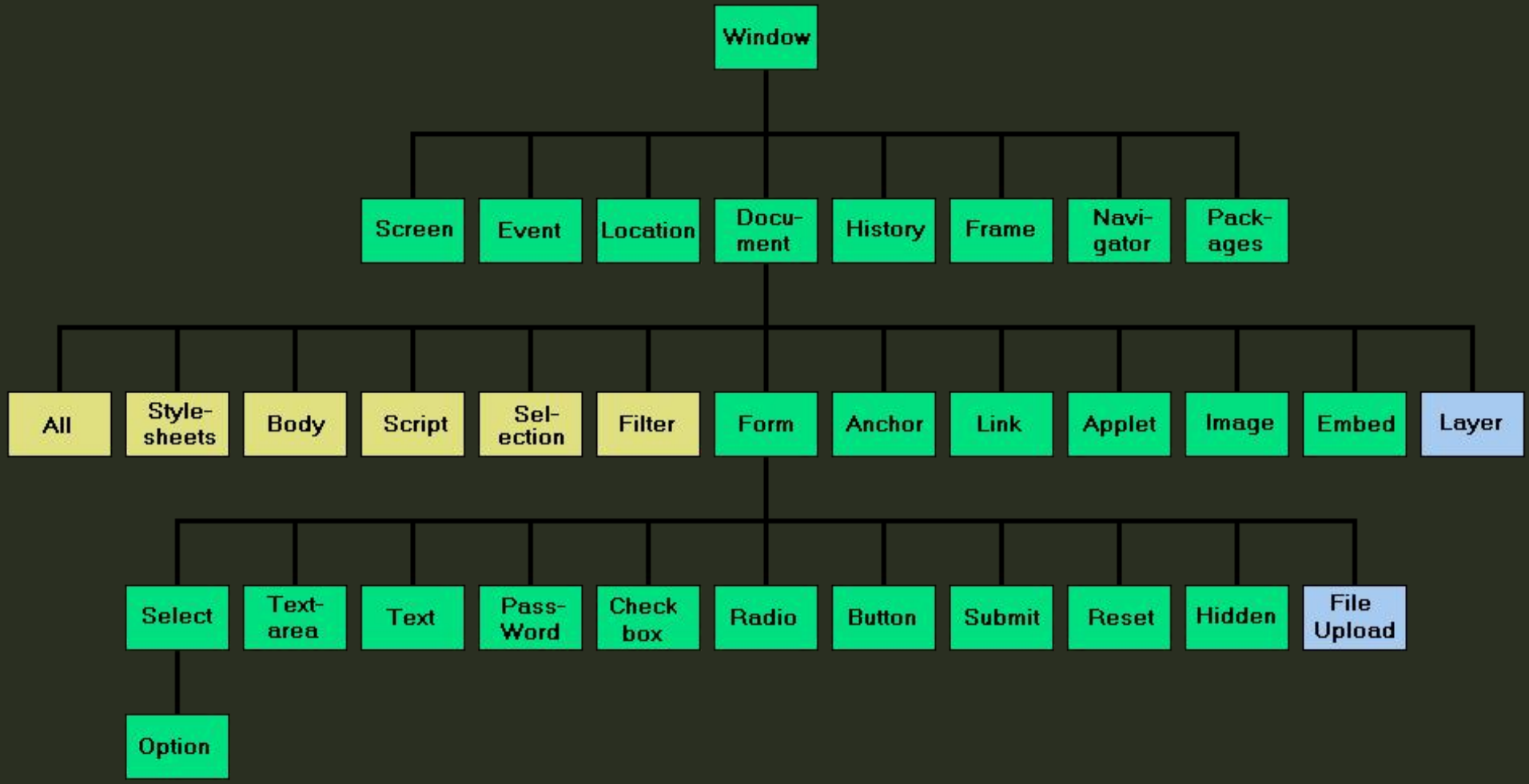


文档结构树




```
<!DOCTYPE html>
<html lang="en">
 <head>
      <title>My title</title>
 </head>
 <body>
       <h1>My header</h1>
       <a href="#">My link</a>
  </body>
</html>
```

BOM+DOM完整结构图



第二节：DOM访问节点

- document对象
- 节点访问方式详解
- 元素的详解
- 元素集合接口详解
- 节点集合详解

document对象

- document对象对应整个文档。整个文档相关的操作和编程的api都是通过document对象进行实现。
- document的常用属性
  - URL： 获取当前文档的URL地址，只读。
  - title： 获取当前文档head中的title的文字内容，可写入更改。
- 时钟补充：
  - setInterval
    - 用法
      - var intervalID = window.setInterval(code, delay);
      - var intervalID = window.setInterval(func, delay[, param1, param2, ...]);// ie 不支持额外参数
    - 方法重复调用一个函数或执行一个代码段，在每次调用之间具有固定的时间延迟。返回一个 intervalID。delay是延迟的毫秒数。
  - 超时执行某个函数setTimeOut
    - 用法
      - var iteervalID = window.setTimeOut(fun, delay);
    - 延迟delay毫秒后执行fun函数体。
  - 清除时钟：clearInterval();
    - window.clearInterval(*intervalID*)
    - intervalID是你想要取消的定时器的ID,这个ID是个整数,是由setInterval()返回的.

案例

- 案例：
  - 标题跑马灯效果
  - 通过数组的join方法实现
  - 通过字符串的slice方法实现
  - 请实现，5秒后停止跑马灯的效果。

访问DOM树上的节点

- **getElementById(id)**
  - **参数：id，是标签的唯一的id属性的值。区分大小写**
  - **返回Element对象**
  - var elem = document.getElementById("para1");
- **getElementsByTagName(tagName)**
  - **根据标签名字获取所有同名的标签对象。**
  - **参数：标签名,如果是*返回所有标签**
  - **返回：**HTMLCollection 此对象是动态的集合，如果标签发生变化自动更新集合。
- **getElementsByName(name)**
  - **根据标签的name属性获取Element的集合。**
  - **参数：标签的name属性值**
  - **返回：** 是一个 NodeList 集合。集合不随着标签的变化而变化。
- **querySelector(selectors)**
  - **根据CSS的选择器进行选取元素（返回满足条件的第一个元素）**
  - **参数：**是一个字符串，包含一个或是多个 CSS 选择器 ，多个则以逗号分隔。
  - **返回：element对象**
- **querySelectorAll(selectors)**
  - **返回满足CSS选择器条件的所有元素**
  - **参数：**是一个字符串，包含一个或是多个 CSS 选择器 ，多个则以逗号分隔。
  - **返回：** NodeList 类型的对象.
- 除了getElementById外，其他方法都可以用于任何一个节点的后代元素。

Element元素对象（接口）详解

- Element对象
  - 元素，是所有标签元素的基础对象。封装了所欲标签元素的公共方法和属性。
  - 常用属性
    - **Element.attributes 获得所有属性key-value集合**
    - **Element.className 获得类名**
    - **Element.id 获取元素的id**
    - **Element.innerHTML 获取元素内部包裹的html标签字符串**
    - **Element.tagName （只读）获取元素的标签名字符串**
  - 常用方法
    - **Element.getAttribute(attName) 返回属性的字符串值**
    - **Element.removeAttribute(**attrName**)**从指定的元素中删除一个属性
    - **Element.setAttribute(name,value) 设置属性**
    - **Element.hasAttribute() 检测属性是否存在**
    - **Element.getElementsByClassName()**
    - **Element.getElementsByTagName()**

HTMLCollection元素对象（接口）详解

- HTMLCollection对象
  - 元素的动态集合，还提供了用来从该集合中选择元素的方法和属性。当其所包含的文档结构发生改变时，它会自动更新。不是真正的数组，是伪数组。
  - 常用属性
    - **HTMLCollection.length** 返回集合当中子元素的数目
  - 常用方法
    - **HTMLCollection.item()** 根据给定的索引（从0开始），返回具体的节点。如果索引超出了范围，则返回 null
      - var c = document.getElementByTagName(" div "); // c是 HTMLCollection 类型实例

        var img0 = c.item(0);

        var img1 = c[1];
    - HTMLCollection.namedItem()根据 Id 返回指定节点，或者作为备用，根据字符串所表示的 name 属性来匹配。根据 name 匹配只能作为最后的依赖，并且只有当被引用的元素支持 name 属性时才能被匹配。如果不存在符合给定 name 的节点，则返回 null。
      - elem1 = document.forms["myForm"];
      - elem2 = document.forms.namedItem("myForm");
    - 将HTMLCollection转成真正的数组
      - Array.prototype.slice.call(c);
  - 注意！！！！：动态变化的，当文档结构发生变化的时候，集合自动变更新。不要用for in循环HTMLCollection伪数组。

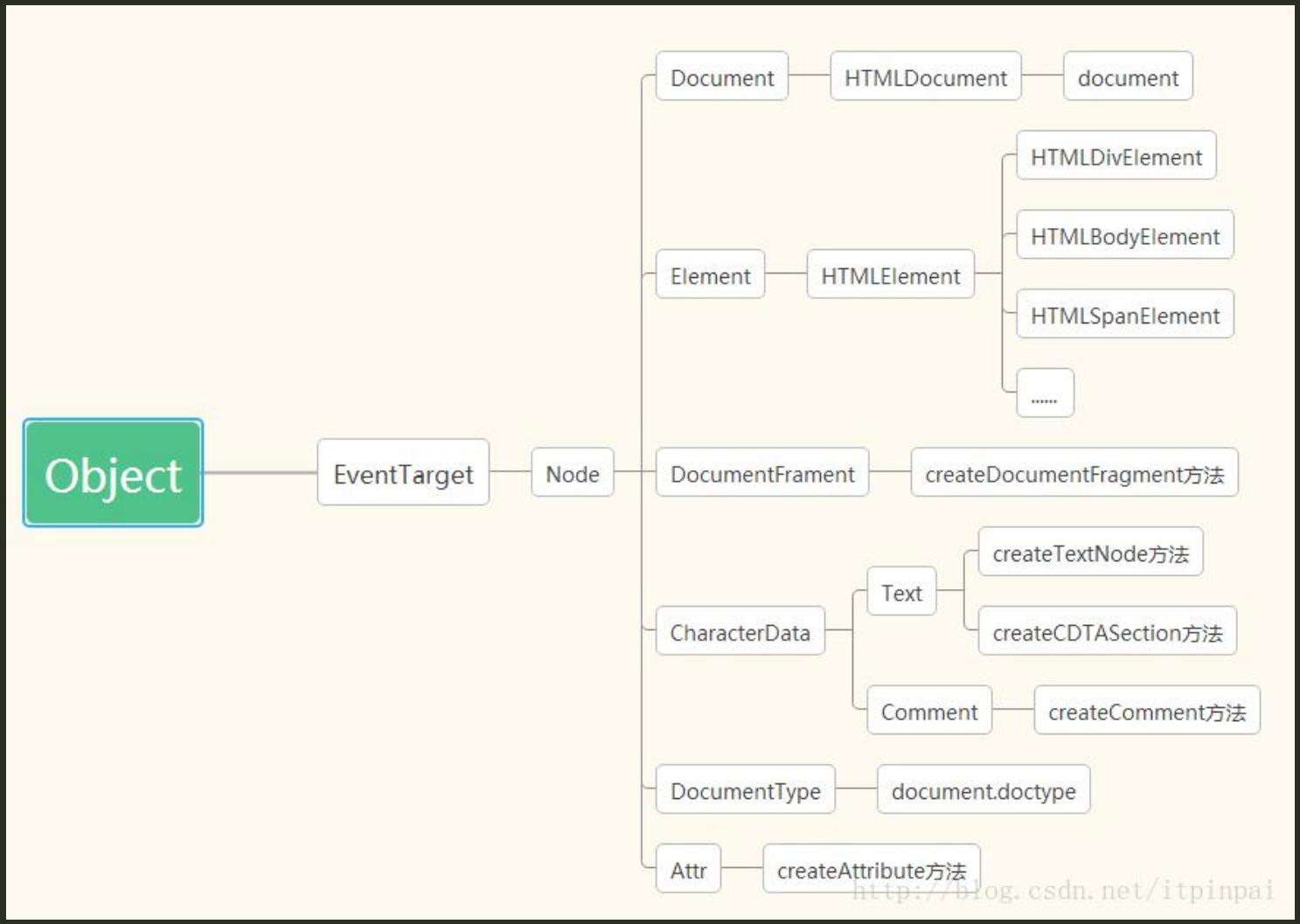
了解Node对象(接口)

- Node是节点的抽象接口。是对节点的API的封装。
- 节点和元素的区别：
  - 元素只能是标签，比如：h1、p、span等标签
  - 节点：既包括标签，也可以是属性节点、文本节点、文档类型节点、注释节点等
  - 元素的类型Element继承自Node接口
- 文档参考地址：https://developer.mozilla.org/zh-CN/docs/Web/API/Node
- 常用属性：
  - 子节点**Node.childNodes ，实时更新**
  - **第一个子节点： Node.firstChild**
  - **Node.lastChild 最后一个节点**
  - **Node.nextSibling** 返回与该节点同级的下一个节点，如果没有返回null
  - **Node.nodeType 节点类型**
  - **Node.parentNode 父节点**
  - **Node.textContent**
  - **Node.previousSibling**
- 常用方法：
  - **Node.appendChild() 添加子节点**
  - **Node.cloneNode() 克隆子节点**
  - **Node.insertBefore()**
  - **Node.removeChild()**
  - **Node.replaceChild()**

NodeList对象节点集合

- NodeList 对象是一个节点的集合。
- 只有一个属性：length表示集合中元素的个数。
- 它不是纯数组，是个伪数组。原型链是： NodeList.prototype --> Object.prototype --> null
- 方法只有一个：NodeList.item()
  - 返回NodeList对象中指定索引的节点,如果索引越界,则返回null.等价的写法是nodeList[idx],不过这种情况下越界访问将返回undefined.
- 转换成真正数组用，数组原型的slice方法借用调用。
  - Array.prototype.slice.call(c);
- 注意
  - 不要用for in进行循环，因为会把length和item也被访问到。所以最好用索引进行循环遍历
  - querySelectorAll返回的NodeList是一个固定的集合，不会随着文档变化而变。
  - Node节点的childNodes属性是实时动态的更新的。【后面会补充Node对象的详解】

DOM继承关系图



第三节：事件

- 事件三要素
- 事件绑定
- 事件信息

事件的概念

- 当某件事发生的时候会触发对应的事情的发生。比如：当猎人在森林里开枪打小动物的时候，其他听到枪声的动物都会感觉受到威胁，然后四散跑开。
- 热水壶当在水烧开的时候自动会关闭烧水的开关。
- 当风扇的开关被按下后，风扇就会转动起来。
- 以上描述都是一个个的事件。一个事件包括：事件源、事件名、事件响应的行为，这三个称为事件的三要素。

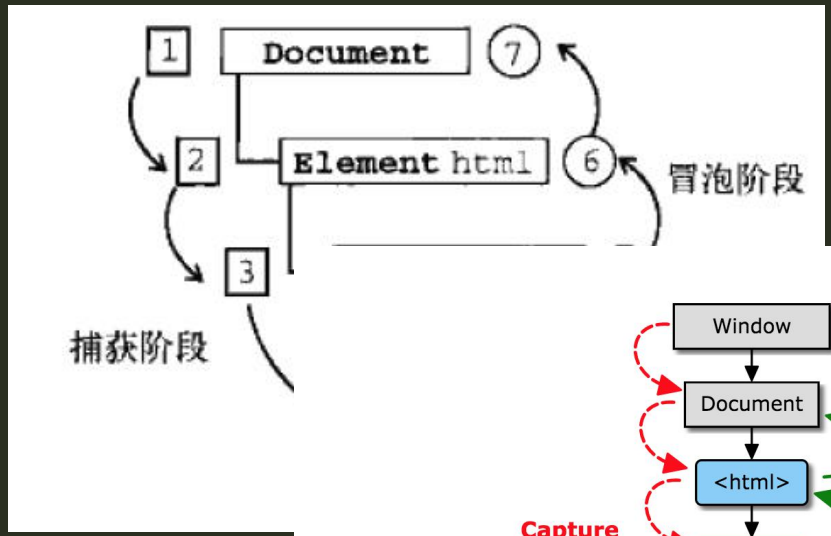
元素绑定事件(DOM0级)

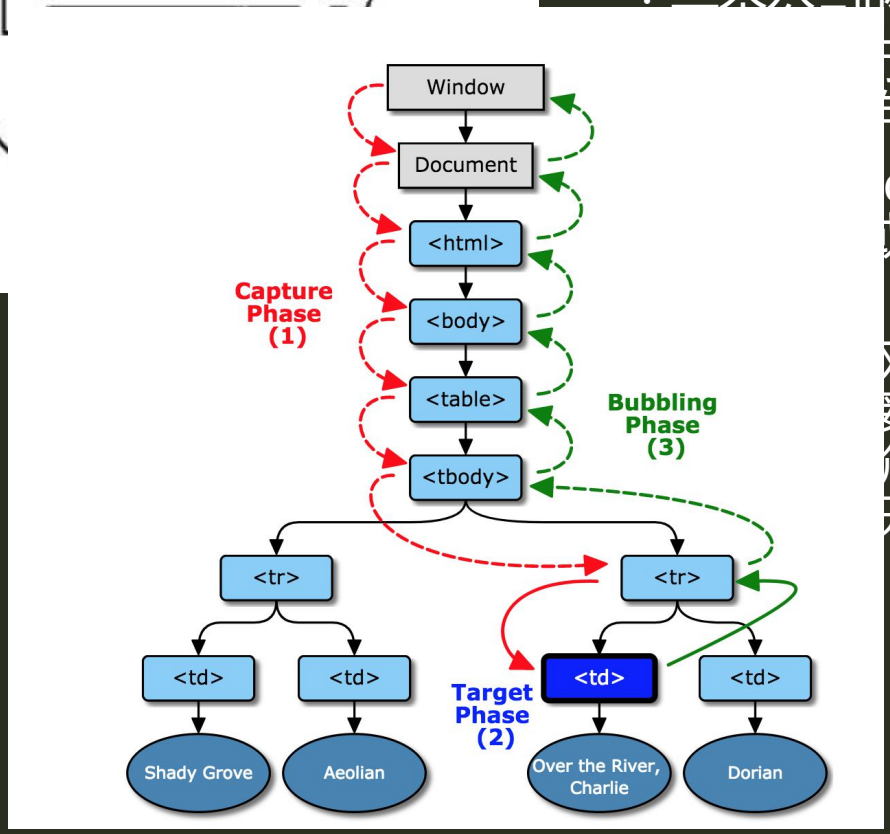
- 第一种：直接在HTML中给标签添加事件属性。
  - 例如： 点击我
  - 不推荐使用，代码写在HTML中，没有智能提示，而且Js代码和HTML结构混杂在一起不利于进行调试和修改。另外让HTML的结构可读性变的很差，尤其是Js代码很多的时候。
- 第二种：通过Js代码给元素添加事件属性

```
<p id="t">我是段落标签</p>
<script>
  var t = document.getElementById('t');
  t.onclick = function(e) {
    alert('点击了段落。');
  };

  // 解绑事件
  // t.onclick = null;
```

  - 此种绑定方式，兼容性最好，推荐大家使用。但是一个标签只能绑定一次事件响应方法。
- 事件三要素：事件源、事件响应方法、事件类型

事件流





- 事件分为捕获阶段和冒泡阶段。捕获阶段就是事件信息从顶层 向下层传递的过程，而冒泡是事件反应处理从底层向上层反馈的过程，例如 ：一个公司的文件和法规从上层一行传达，而反应是从底向上反馈。

ddEventListener来实
或者冒泡阶段的事件响

对象上设置的事件属性
接编写标签属性的方式
介段执行。IE9之后才支
持，之前版本只支持冒

绑定事件(DOM2级)

- **addEventListener注册事件**
  - 语法：target.addEventListener(type, listener[, useCapture]);
  - 参数：
    - type，事件的类型。注意，在DOM0级中所有的事件都包含on，比如：onclick，ondbclick，onmouseleave等。但是在此方法中并没有on前缀。只有事件的类型的字符串，比如：click、dbclick、mousedown、keyup、keydown等。具体参考：https://developer.mozilla.org/zh-CN/docs/Web/Events
    - listener：事件处理方法
    - useCapture：默认值是false，可以省略。意思是否是在捕获阶段触发事件响应方法。true表示是，false是在冒泡阶段触发。
  - 优点：
    - 可以控制是在捕获阶段还是在冒泡阶段触发事件响应方法。
    - 可以注册多个事件响应方法，不会覆盖。
    - 事件响应方法（事件处理程序）执行顺序就是事件注册顺序

案例

- 有五个☆标签（span）请给所有的标签绑定一个事件，当span标签被点击的时候，自己变成实心★，其他的都变成空心☆。
- 实现动态添加li标签的效果，要求li的内部文本从1开始，按钮每点击一次，自动添加一个li。

解除绑定事件(DOM2级)

- **removeEventListener**
  - 参数同addEventListener()。注意一点就是：listener必须是跟添加事件的响应方法相同的函数！！！

```
var div = document.getElementById('div');
var listener = function (event) {
  /* do something here */
};
div.addEventListener('click', listener, false);
div.removeEventListener('click', listener, false);
```

IE8的事件绑定和解绑

- IE8包括之前的浏览器版本都没有实现addEventListener方法。但是实现了自己的一套方法：attachEvent()和detachEvent()方法。
- IE8包括之前的版本只支持冒泡，所以注册的事件都只能在冒泡阶段触发。
- attachEvent和detachEvent的参数都是一样的，跟addEventListener的前两个参数是一样的。
  - 第一个参数eventNameWithOn：on+事件类型的字符串，比如：onclick
  - 第二个参数事件响应方法：类型是函数，可以是匿名函数。

跨浏览器兼容绑定事件

- 由于Ie不支持addEventListener方法注册事件响应方法。
- 所以为了兼容所有的浏览器那么注册事件响应方法的时候需要先判断后处理

```
if (el.addEventListener) {
  el.addEventListener('click', modifyText, false);
} else if (el.attachEvent)  {
  el.attachEvent('onclick', modifyText);
}
```

事件响应方法的执行顺序

- 设置标签的事件属性或者直接给元素设置事件属性的最优先执行
- addEventListener的事件响应方法执行顺序跟注册顺序一致。
- attachEvent注册的事件响应方法的执行顺序不确定，执行过程不能依赖它的注册顺序。

第四节：事件对象

- DOM的事件触发的时候，会产生一个事件对象Event，事件对象中封装事件相关信息：事件源对象、事件类型、事件处理操作方法，在某些鼠标事件中，还会包括鼠标的相关位置信息、鼠标按键等信息。在键盘的事件中，还会包括按键的信息。
- 获得事件对象：
  - 标准浏览器是在事件处理程序的第一个参数就是事件对象。
  - ie是通过window.event来获得当前事件对象。
- 常用属性：
  - Event.target 事件源对象，IE6-IE8 Event.srcElement
  - Event.type 事件类型
  - Event.cancelBubble 是否取消事件冒泡，IE下使用
  - Event.returnValue 布尔类型，false则阻止默认行为
- 常用方法：
  - 阻止默认行为 Event.stopDefault() 比如：连接跳转
    - Ie不能用，在事件响应方法中returnValue = false。
  - 阻止事件冒泡 Event.stopPropagation() 标准浏览器支持
    - Ie不能用，用cancleBubble=true

IE下的事件对象

- IE的事件对象不会传入到事件响应方法的参数中。但是可以使用window.event获取到。
- 事件的兼容处理综合案例

阻止事件冒泡与默认行为

- 阻止事件冒泡
  - window.event.cancelBubble = true；// ie
  - e.stopPropagation(); // 非ie，可以阻止事件冒泡或捕获。
- 阻止默认行为（阻止默认行为不能阻止事件冒泡）
  - window.event.returnValue = false； // ie
  - e.preventDefault(); // 非ie
- 综合写法：

js停止冒泡

```
function myfn(e){
window.event? window.event.cancelBubble = true : e.stopPropagation();
}
```

js阻止默认行为

```
function myfn(e){
window.event? window.event.returnValue = false : e.preventDefault();
}
```

事件处理程序的返回值

- 在使用事件属性注册事件处理程序中的返回值有时候非常有用。
- 常用点：
  - 当键盘的onkeypress事件中，如果事件处理程序返回false，则表示过滤某个键。
  - 在表单的onclick事件中，return false则表示阻止默认的提交操作，当然也可以用event.returnValue = false；
  - window对象的onbeforeunload事件中，如果返回字符串，表示浏览器窗口在跳转前，会弹出确认消息的对话框，返回的字符串会显示在对话框上。

案例

- 实现只能输入数字的文本框
  - html5智能表单标签实现
  - 通过监听文本框的键盘的事件控制过滤部分按键
- 浏览器在离开页面的时候提醒用户是否要离开

- 表单点击提交的时候校验用户名文本框不能为空

第五节：事件类型

- 文档加载和准备就绪事件
- 焦点获取和失去事件
- 鼠标事件
- 鼠标滚轮事件
- 键盘事件
- 拖放事件
- 文本输入事件
- ....

文档加载完成事件

- onload事件：
  - 在window对象或者img对象上进行触发。当网页所有内容都加载完成后（包括图片下载完成）则会触发window的onload事件。图片加载完成后也会触发图片的onload事件。
  - 其实在图片和异步的脚步加载完成之前，只要dom文档加载完成后就可以执行js的相关的dom操作，没必要等图片等加载完成。
  - 优化：
    - DOMContentLoaded事件是DOM文档加载完成后就会触发（ff最先引入，IE9开始支持）目前已经纳入标准HTML5
    - ie早起版本不支持DOMContentLoaded事件，但是可以监听document的状态变化来实现。document的onreadystatechange事件中，document的readyState为interactive 时代表文档加载完成
    - document.readyState 属性描述了文档的加载状态,分为：loading仍然加在中，interactive 文档可以访问，但图片等未加载完成，complete全部加载完成
    - jQuery内部的document.ready就是如此实现。
- onunload事件，跟onload相反
- onbeforeunload事件：当网页刷新或者页面关闭之前。

窗口的事件

- onresize事件
  - 当窗口大小发生改变的时候触发。
  - 由于某些浏览器实现的是改变一个像素就会触发一次。所以一次调整过程中，事件可能重复触发多次，所以不要做大量计算的代码。ff是在改变大小结束后触发一次。
- onscroll事件
  - 当窗口发生滚动的时候，会导致此事件发生。
  - 获取窗口的滚动的X和Y方向的值：pageXOffset和pageYOffset，但是ie浏览器不支持。需要其他方式代替。
  - 兼容处理：

```
var h = document.documentElement.scrollTop || document.body.scrollTop;
```

滚动案例1

- 实现当页面加载到底部的时候，自动加载新内容（用动态创建标签替代。
- 参考：
  - document.body.clientHeight可以获取body的高度（要有HTML的标准头部）
  - document.documentElement.clientHeight; 获取视口的高度（所有浏览器都兼容）
  - 获取滚动的高度：document.documentElement.scrollTop || document.body.scrollTop;
    - chrome：document.body.scrollTop
    - ie/ff: document.documentElement.scrollTop

滚动案例2

- 实现 hao123.com滚动到某位置，搜索框固定定位。

焦点事件

- 获取焦点 focus
  - 当标签元素获得焦点的时候触发此事件。
  - 此事件不会冒泡，浏览器兼容非常好。
- 失去焦点 blur
  - 当标签失去焦点时候触发。此事件不冒泡。
  - 兼容性比较好。
- 获取焦点focusin，同focus，但是会冒泡或者捕获。
- 失去焦点focusout，同blur，但会冒泡或者捕获。
- Element.focus()方法可以让标签元素获得焦点。

焦点事件案例

- 案例：实现文本框输入文本不能为空。
- 实现文本框placeholder功能
- 类百度搜索框输入案例

第六节：鼠标事件

- click事件（单击），可以通过点击鼠标左键（默认情况），也可以通过键盘的回车键触发。
- dbclick双击事件。
- mousedown：按下鼠标键的时候触发，不能通过键盘触发。
- mouseup: 用户释放鼠标键的时候触发。不能通过键盘触发。
- **mouseenter：首次进入元素范围触发，不冒泡。**
- **mouseover**：当鼠标在元素上面或者子元素上面时触发此事件，会冒泡。
- **mouseleave**：当鼠标离开目标元素位置，不冒泡。
- mouseout：当鼠标离开目标元素或者子元素都会触发此事件，会冒泡的。【不常用】
- **mousemove:** 鼠标在元素内部移动时重复触发此事件。【重点】

鼠标事件对象（接口）

- 鼠标事件中对应的事件对象都是MouseEvent对象
- MouseEvent继承自UIEvent。UIEvent继承自Event（事件对象）
- UIEvent就是增加了pageX和pageY属性。而MouseEvent又覆盖了pageX和pageY属性。
- 鼠标事件对象中包括了：鼠标按键信息、键盘配合鼠标的信息、鼠标的位置信息等。

鼠标事件对象的坐标位置

- 客户区坐标（视口内的坐标）
  - clientX：视口内横坐标，从视口最左边开始计算。
  - clientY：视口内的纵坐标。
- 页面坐标
  - 没有滚动的时候，clientX和pageX相同。IE8不支持pageX
  - pageX就是鼠标点击位置相对于整个文档页面的水平坐标。
  - pageY是鼠标点击位置相对于整个文档（包括不可视区域）的纵坐标。
  - pageX = event.clientX + (document.documentElement.scrollLeft|| document.body.scrollLeft)// for<=ie8
  - pageY = event.clientY + (document.documentElement.scrollTop || document.body.scrollTop); // for<=ie8
- 屏幕坐标
  - 鼠标点击的时候，通过screenX和screenY来对应鼠标在屏幕的坐标。

- offsetX 和offsetY是相对于事件源对象左上角的距离。

案例

- 蝴蝶跟着鼠标飞案例
- 拖动div移动案例

鼠标事件对象的鼠标按钮信息

- event.button(mousedown事件中使用)
- 标准的DOM模型中，按下鼠标后，事件对象event中的button属性可能有三个值：0主键（左键），1中键（滚轮），2鼠标次键。
- IE的模型跟DOM的标准不一致。

- 0：表示没有按下按钮。
- 1：表示按下了主鼠标按钮。
- 2：表示按下了次鼠标按钮。
- 3：表示同时按下了主、次鼠标按钮。
- 4：表示按下了中间的鼠标按钮。
- 5：表示同时按下了主鼠标按钮和中间的鼠标按钮。
- 6：表示同时按下了次鼠标按钮和中间的鼠标按钮。
- 7：表示同时按下了三个鼠标按钮。

鼠标键的兼容处理

- 由于网页中一般只对应三个按钮即可，很少有组合按键
- 所以对应按键的兼容处理就很好搞定。

```
switch(event.button){
    case 0:
    case 1:
    case 3:
    case 5:
    case 7:
        return 0;
    case 2:
    case 6:
        return 2;
    case 4:
        return 1;
}
```

案例

- 右键弹出自定义菜单
- 屏蔽浏览器的右键菜单：oncontextmenu事件里面阻止默认行为即可，可以把事件加到document上就可以全兼容了。

鼠标事件对象的键盘信息

- altLeft: 布尔类型，是否按下了左边的alt键。altKey也为true。
- ctrlLeft： 布尔类型，是否按下了左侧的ctrl键
- shiftLeft：布尔类型

案例：如果按住ctrl键，则多选多个li标签进行高亮，否则排他高亮。

mousewheel（滚轮事件）

- ie、chrome支持mousewheel
- 火狐支持：DOMMouseWheel事件。
- 鼠标向上或者向下滚动的时候会触发mousewheel事件
- 事件可以在任何元素进行触发，而且会冒泡到顶层的window对象（Ie只能冒泡到document）
- event的特殊属性：wheelDelta.当用户向上滚动的时候是120的倍数，如果是向下滚动则是-120的倍数。
- 火狐支持：DOMMouseWheel事件。而且事件对象只有detial属性，值是 -3或者+3，向下滚动式3，上是-3

第七节：键盘和文本事件

- keydown：用户在键盘上按下某按键是发生。一直按着某按键则会不断触发。并触发默认行为。
- keypress：用户按下一个按键，并产生一个字符时发生（也就是不管类似shift、alt、ctrl之类的键，就是说用户按了一个能在屏幕上输出字符的按键keypress事件才会触发）。一直按着某按键则会不断触发。
- keyup：用户释放某一个按键时触发。
- 事件触发顺序：keydown - > keypress - > keyup
- keydown和keyup都是通过事件对象的keyCode获取按下键的字符编码。keypress是通过事件对象的charCode来获取按下的键的字符（ie8可以通过keyCode获取）。
- 键盘事件的对象中也包含shiftKey、ctrlKey、altKey和metaKey属性
- 键盘分为：可以打印的字符键和不可打印的功能键。常用功能键：Esc、Tab、Caps Lock、Shift、Ctrl、Alt、Enter、Backspace....
- change事件，input、textarea、select可以出发change事件。焦点离开文本框才会触发。

键盘文本事件

- 监控文本框输入文本超过3个字符，下一个文本框获得焦点。

第八节：事件高级

- 事件绑定封装
  - 标准浏览器：addEventListener（ie9+）
  - IE浏览器：attachEvent
  - 标准浏览器支持捕获和冒泡。
- 事件对象封装
  - ie下获取事件对象，通过window.event
  - 标准浏览器通过事件处理程序的参数。
  - 取消默认操作兼容
  - 取消事件冒泡兼容
- 文档加载完成封装
  - ie通过监听document的readystatechange事件，根据document的readyState的值来判断文档是否可进行交互访问。
  - 标准浏览器可以通过DOMContentLoaded事件
  - 最后也可以使用window.onload
- 事件委托
  - 当子标签非常多的时候，可以灵活运用事件的冒泡或者捕获的特性来实现在父容器上绑定事件，让所有的子元素都享受同一个事件处理程序。
  - 优势：动态添加的元素也可以享受此事件。较少的占用内存（更少的绑定了事件对象，提高了效率）

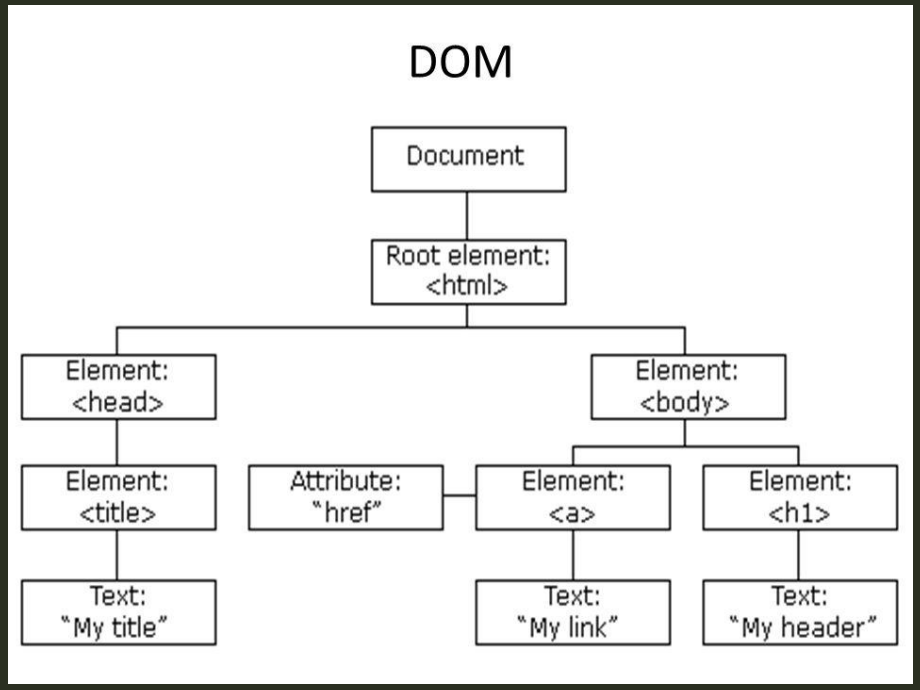
第九节：节点操作

- 节点对象Node
- 节点层级和关系
- 节点的类型
- 节点操作

节点层次

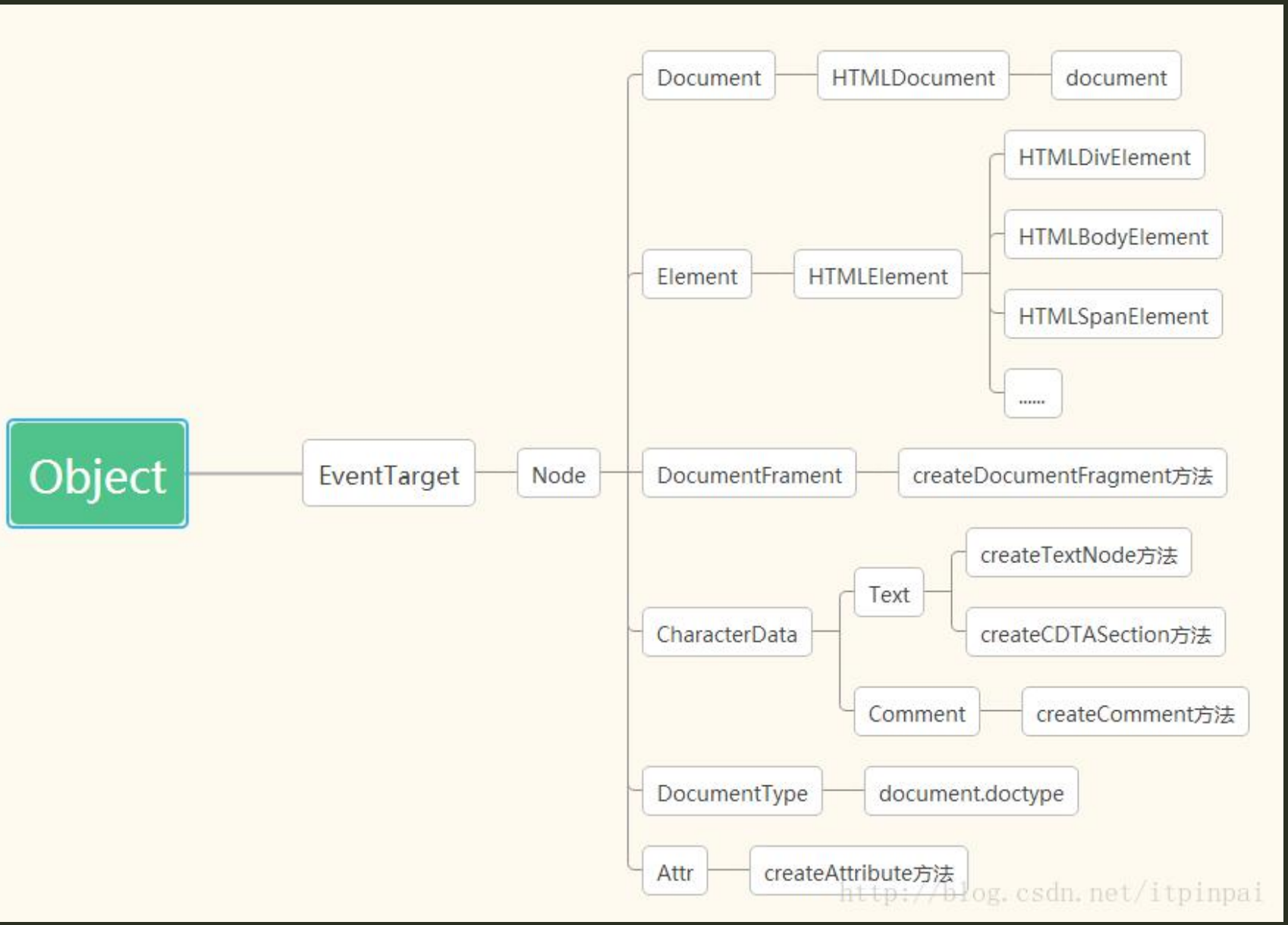
- 一个HTML文档分为很多个节点。节点有不同的类型：文档节点、元素节点、文本节点、注释节点、文档类型节点、属性节点等。
- 节点与节点形成了层级结构。也就是文档节点树
- 节点对象的DOM类型是Node

文档结构树




```
<!DOCTYPE html>
<html lang="en">
  <head>
        <title>My title</title>
  </head>
  <body>
        <h1>My header</h1>
        <a href="#">My link</a>
    </body>
</html>
```

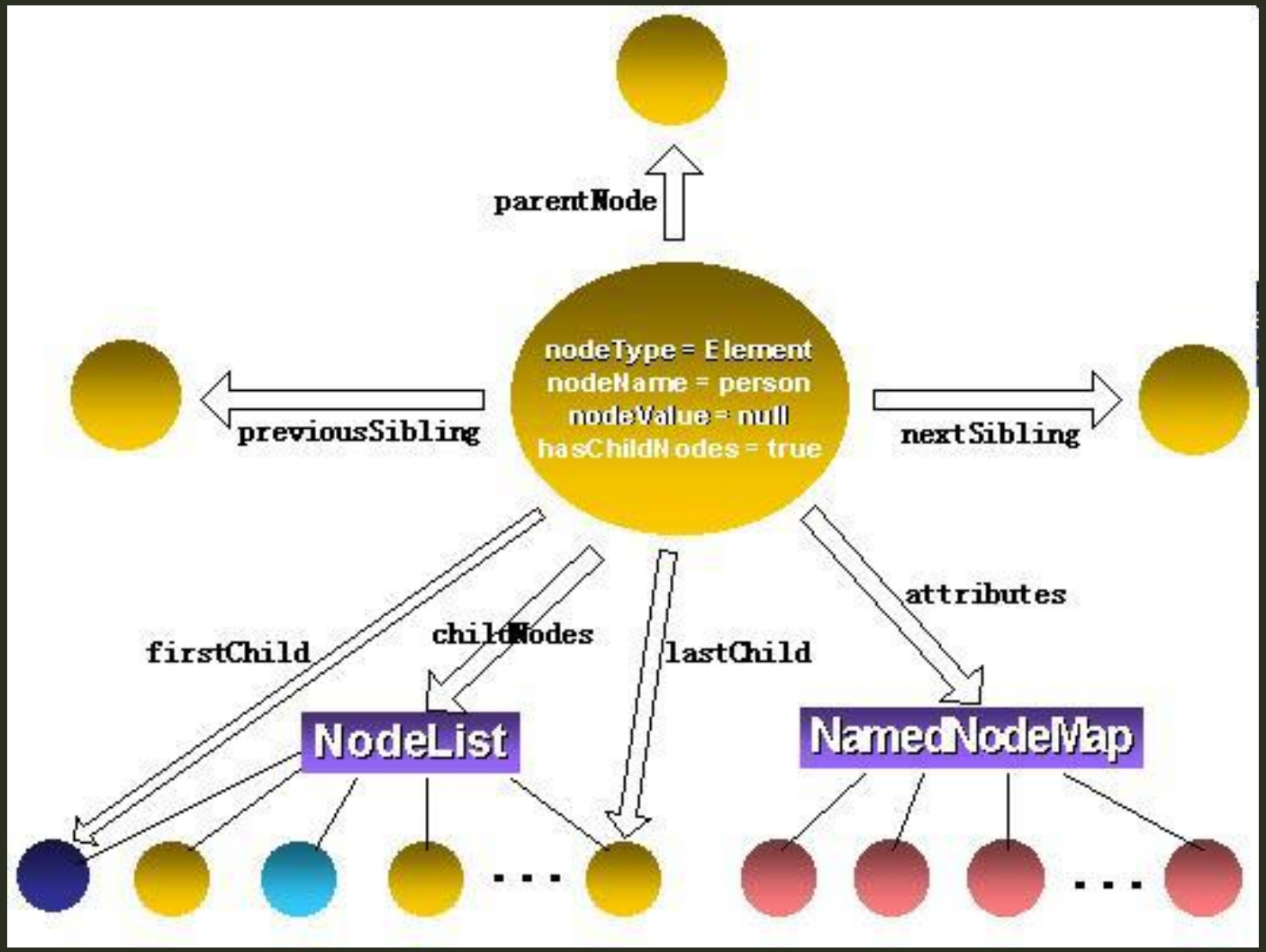
DOM继承关系图



Node接口

- Node接口是对节点的封装。继承自EventTarget。也就是说所有的Node节点对象都可以绑定事件。
- Node常用属性
  - nodeType
    - 标签 =1，属性=2，文本=3，注释=8，文档=9
    - 详情参考：https://developer.mozilla.org/zh-CN/docs/Web/API/Node/nodeType
  - nodeName
    - 标签 = tagName ， 属性=属性名， 文本=#text ，文档=#document
  - nodeValue
    - 文本节点返回文本值，属性节点返回属性内容，注释节点返回注释内容，其他都返回null
  - textContent 属性表示一个节点及其后代的文本内容。（ie9+）
  - innerText是一个表示一个节点及其后代的“渲染”文本内容的属性（ie6）
- 节点关系
  - Node.childNodes (NodeList类型） 返回子节点 ，子节点变化会自动更新。
  - Node.firstChild 返回第一个子节点或者 null。
  - Node.lastChild 返回最后一个子节点或者null。
  - Node.nextSibling 返回与该节点同级的下一个节点，如果没有返回null。
  - Node.previousSibling返回当前节点的前一个兄弟节点,没有则返回null.
  - Node.parentNode 返回父节点
  - Node.parentElement 返回父节点标签，如果不是标签返回null

节点关系


parentNode
nodeType = Element
nodeName = person
nodeValue = null
hasChildNodes = true
previousSibling
nextSibling
firstChild
childNodes
lastChild
attributes
NodeList
NamedNodeMap

Node接口的方法

- Node.appendChild(child) 添加子节点//***
- Node.insertBefore(newElement, referenceElement) 在当前节点的某个子节点之前再插入一个子节点。
- Node.removeChild(c) 从DOM中删除一个子节点。返回删除的节点。
- Node.replaceChild(newChild, oldChild) 用指定的节点替换当前节点的一个子节点，并返回被替换掉的节点
- Node.hasChildNodes( ) 如果有子节点返回true否则false
- Node.cloneNode(deep) 方法返回调用该方法的节点的一个副本，注意id必须设置不同的值。deep是否是深度拷贝。
- Node.contains(c)返回的是一个布尔值，来表示传入的节点是否为该节点的后代节点。

节点操作

- 创建节点
  - document.createElement(tagName) 创建元素节点
  - document.createAttribute(name) 方法创建并返回一个新的属性节点.可以通过nodeValue设置属性节点值。
  - document.createComment(data) 创建注释节点。
  - document.createTextNode创建一个新的文本节点.
- 删除节点
- 修改节点
- 遍历节点

节点操作案例

- 一个数组放着菜单信息，请将按钮的菜单信息动态生成为li标签添加到ul#menu的列表中。
- JSON数据格式：

```
var data = [{name:'首页', url:'/index.html', id: 1},
  {name:'关于', url:'/about.html', id: 2},
  {name:'产品', url:'/product.html', id: 3},
  {name:'案例', url:'/usecase.html', id: 4},
  {name:'联系', url:'/contact.html', id: 5}
  ];
```

补充：Element.setAttributeNode(attrNode);可以设置元素的属性节点。

动态创建标签的方式及注意事项

- 动态创建标签的方式
  - document.createElement方法
  - innerHTML方法
- 动态创建标签的注意事项
  - 尽量减少dom的操作，尽量批量操作。比如：HTMLElement的设置innerHTML的操作次数尽量减少，减少交互次数会提高DOM的显示和解析效率。
  - 设置标签内的文本尽量使用innerText或者textContent尽量减少使用TextNode节点对象。

关闭对话框案例

- 实现消息框上的按钮的关闭功能

第十节：元素样式操作

- 内联样式
  - Element.style属性可以获得元素的内联样式。并且style中已经把所有的样式都做了处理，可以直接处理访问各个样式规则。
  - 不能直接对Element.style做赋值操作。但是可以对它的属性做处理。
  - Element.style.cssText可以直接设置内联样式的文本。
  - Element.className 可以设置内联的样式类
  - 也可以通过Element的setAttribute来进行设置样式的属性值。
- 样式表操作
  - 自己扩展搜索：dom操作样式表。很少用到。了解可以这么做就行了。

动态添加样式表

```
// 动态引入link标签和css文件
var flag = true;
if (flag) {
  // 调用函数,引入路径;
  loadStyles('basic.css');
}

function loadStyles(url) {
  var link = document.createElement('link');
  link.rel = 'stylesheet';
  link.type = 'text/css';
  link.href = url;

document.getElementsByTagName('head')[0].
appendChild(link);
}
```

```
//动态创建样式脚本并执行
function loadStyleString(css) {
  var style = document.createElement("style");
  style.type = "text/css";
  try {
    // IE会报错;不允许向<style>元素添加子节点;
    style.appendChild(document.createTextNode(css));
  } catch (ex) {
    // 此时,访问元素的StyleSheet属性,该属性有有一个
cssText属性,可以接受CSS代码;
    style.styleSheet.cssText = css;
  }
  var head =
document.getElementsByTagName("head")[0];
  head.appendChild(style);
}
// 调用;
loadStyleString("body {background-color:red}");
```

动态创建js脚本

```
//动态引入js文件
var flag = true;
if (flag) {
  loadScript('browserdetect.js'); // 调用函数,引入路径;
}

function loadScript(url) {
  var script = document.createElement('script'); // 创建script标签;
  script.type = 'text/javascript'; // 设置type属性;
  script.src = url; // 引入url;

document.getElementsByTagName('head')[0].appendChild(script); // 将script引入<head>中;
}
```

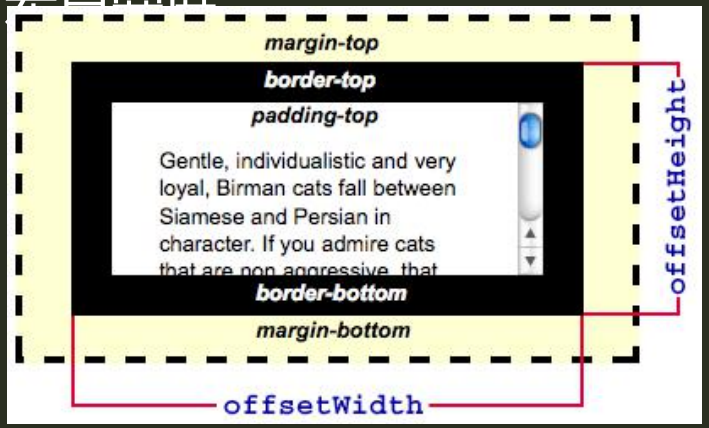
```
//动态创建script标签执行js
function loadScriptString(code) {
  var script = document.createElement("script");
  script.type = "text/javascript";
  try {
    // IE浏览器认为script是特殊元素,不能再访问子节点;报错;

script.appendChild(document.createTextNode(code));        // W3C方式;
  } catch (ex) {
    script.text = code;
// IE方式;
  }
  document.body.appendChild(script);
}
loadScriptString("function sayHi(){alert('hi')}");
```

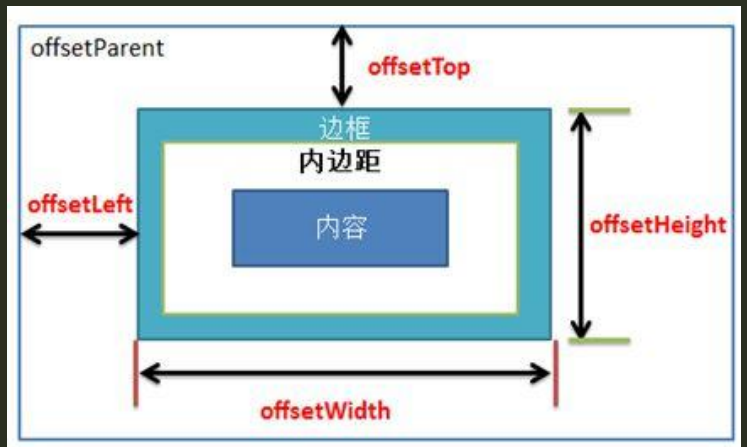
第十一节：HTMLElement元素

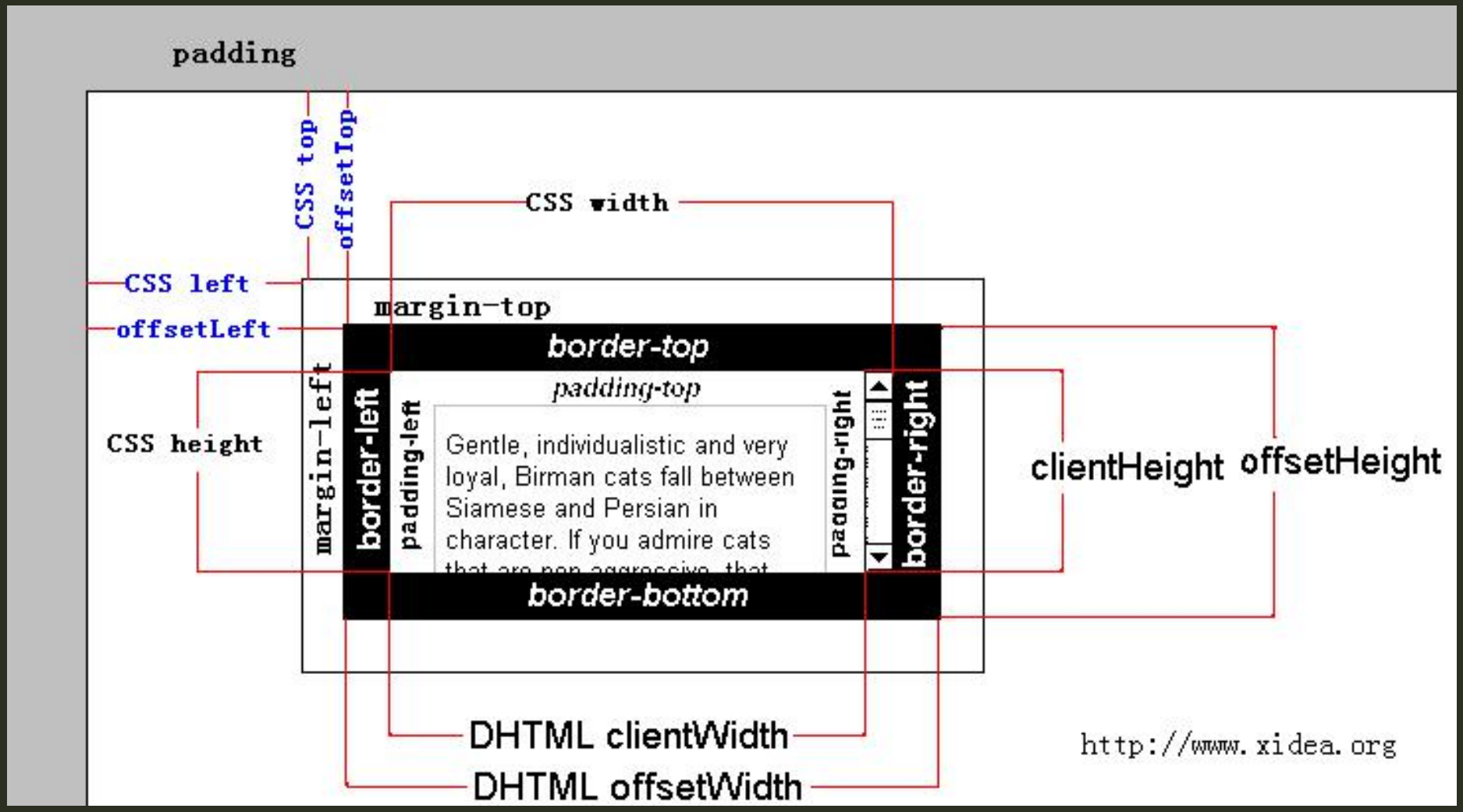
- 常用元素位置
- 表单处理

HTML元素的大小和位置

- HTML元素的偏移量
  - HTMLElement.offsetParent 返回一个指向最近的包含该元素的定位元素。如果没有定位的元素，则 offsetParent 为最近的 table, table cell 或根元素。
  - HTMLElement.offsetLeft返回当前元素左上角相对于 HTMLElement.offsetParent 节点的左边界偏移的像素值。.
  - HTMLElement.offsetTop 它返回当前元素相对于其 offsetParent 元素的顶部的距离。
  - HTMLElement.offsetHeight返回该元素的像素高度，高度包含该元素的垂直内边距和边框
  - HTMLElement.offsetWidth 是一个只读属性，返回一个元素的布局宽度。

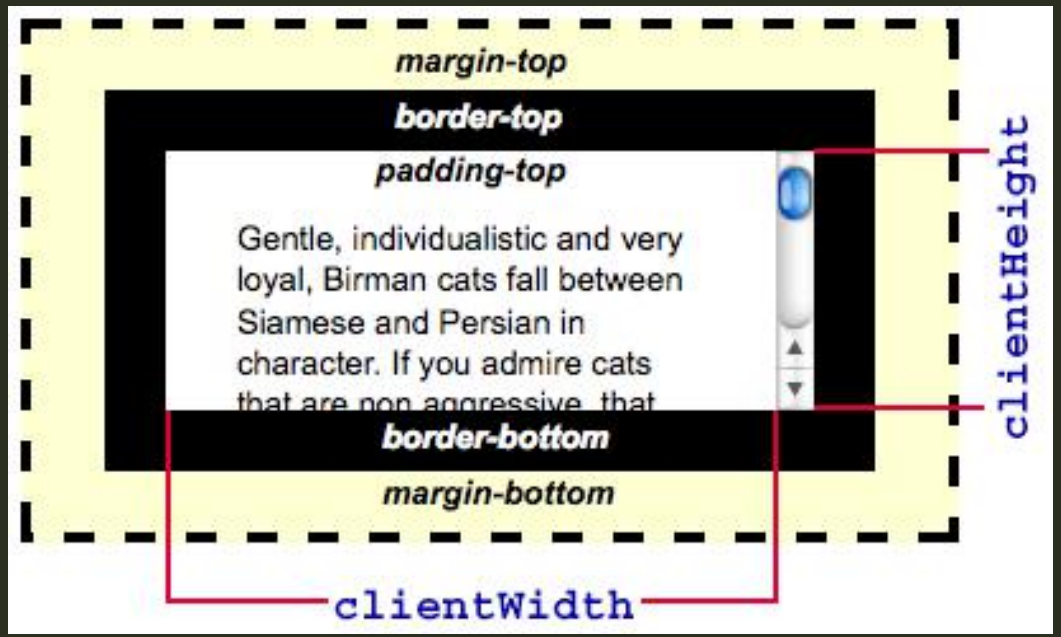





padding
CSS top
offsetTop
CSS width
CSS left
offsetLeft
margin-top
border-top
padding-top
margin-left
border-left
padding-left
Gentle, individualistic and very loyal, Birman cats fall between Siamese and Persian in character. If you admire cats
padding-right
border-right
CSS height
clientHeight
offsetHeight
border-bottom
DHTML clientWidth
DHTML offsetWidth
http://www.xidea.org

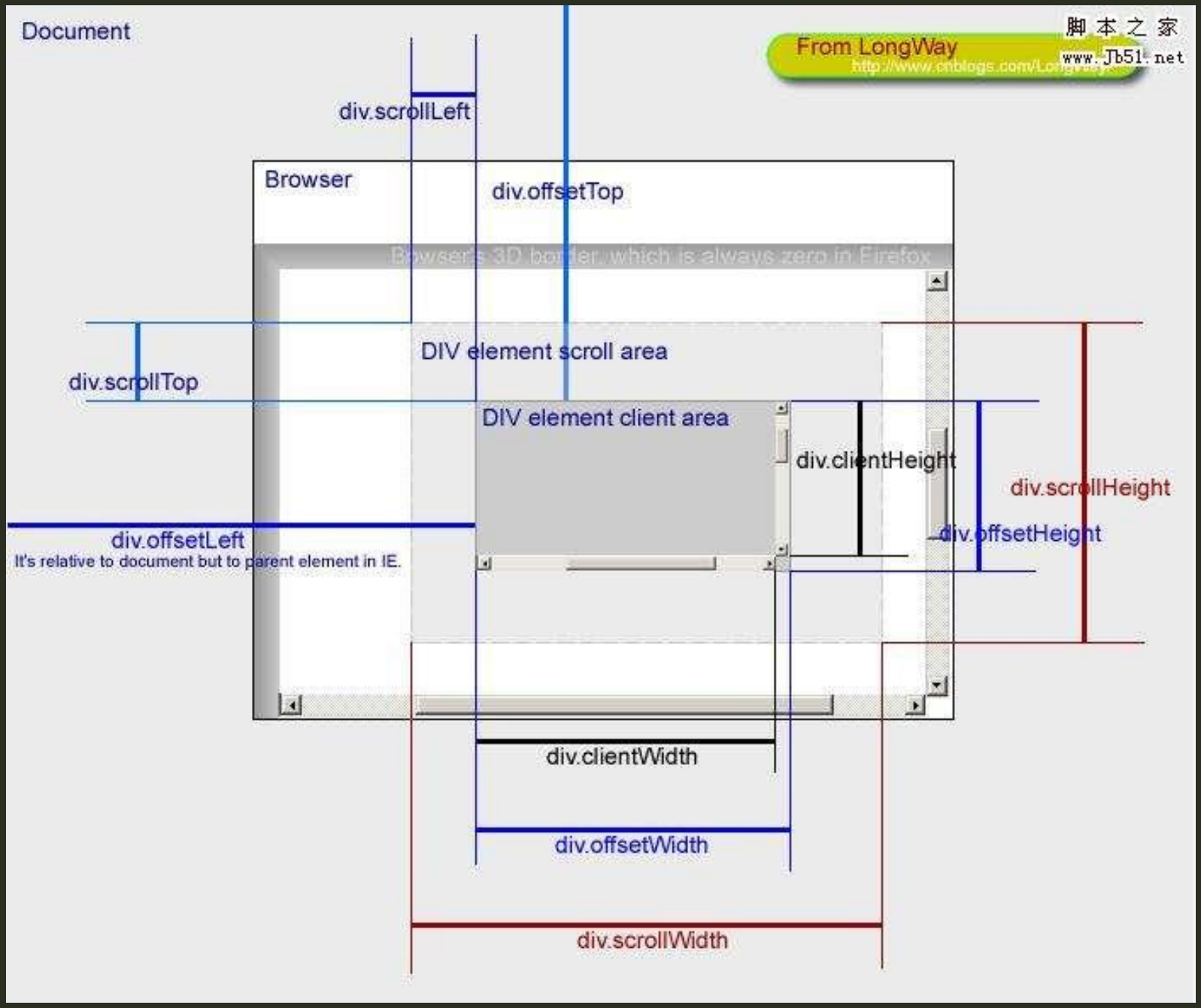
HTML元素的大小和位置

- Element.clientWidth 属性表示元素的内部宽度，以像素计。该属性包括内边距，但不包括垂直滚动条（如果有）、边框和外边距。
- Element.clientWidth 属性表示元素的内部宽度，以像素计。该属性包括内边距，但不包括垂直滚动条（如果有）、边框和外边距。
- Element.clientLeft： 表示一个元素的左边框的宽度，以像素表示。
- Element.clientTop：一个元素顶部边框的宽度（以像素表示）。不包括顶部外边距或内边距。clientTop 是只读的。



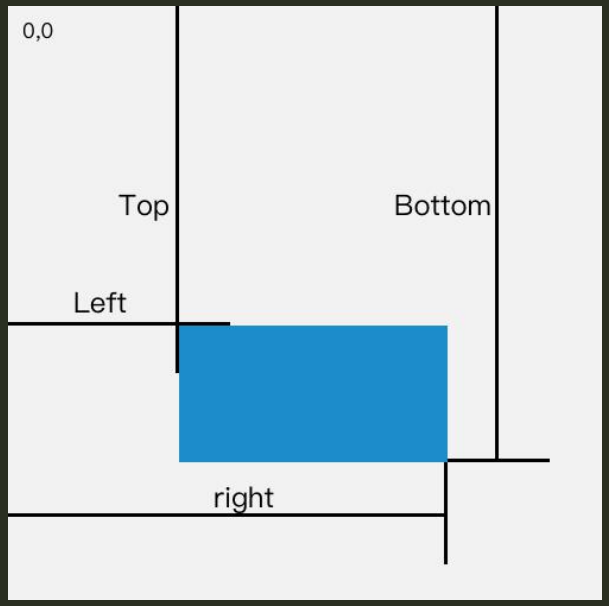
元素的scroll宽高

- Element.scrollHeight: 是计量元素内容高度的只读属性，包括overflow样式属性导致的视图中不可见内容。没有垂直滚动条的情况下，scrollHeight值与元素视图填充所有内容所需要的最小值clientHeight相同。包括元素的padding，但不包括元素的margin.
- Element.scrollWidth：元素的scrollWidth只读属性以px为单位返回元素的内容区域宽度或元素的本身的宽度中更大的那个值。若元素的宽度大于其内容的区域（例如，元素存在滚动条时），scrollWidth的值要大于clientWidth
- Element.scrollLeft 左侧滚动的距离。
- Element.scrollTop 顶部滚动的距离


Document
From LongWay
脚本之家
www.Jb51.net
div.scrollLeft
Browser
div.offsetTop
Browser's 3D border, which is always zero in Firefox
DIV element scroll area
div.scrollTop
DIV element client area
div.clientHeight
div.scrollHeight
div.offsetHeight
div.offsetLeft
It's relative to document but to parent element in IE.
div.clientWidth
div.offsetWidth
div.scrollWidth

Element.getBoundingClientRect()

- Element.getBoundingClientRect()方法返回元素的大小及其相对于视口的位置。
- 返回值是一个 DOMRect 对象。它包含了一组用于描述边框的只读属性——left、top、right和bottom，单位为像素。除了 width 和 height 外的属性都是相对于视口的左上角位置而言的。ie9以上才支持width和height



表单操作

- 文本框选中文本
- 表达提交、重置
- 选择框添加、移除、排序选项
- 单选按钮和多选按钮

HTMLInputElement对象的方法和属性

- HTMLInputElement →HTMLElment→Element→Node→EventTarget→Object
- 常用方法
  - focus() 获得焦点
  - blur() 失去焦点
  - select() 选中文本
  - click() 触发点击事件（类似鼠标点击)
- 常用属性
  - width：标签的宽度
  - height：标签的高度
  - value 字符串类型，返回标签的value属性值。
  - name 标签的name属性
  - type (字符串类型)，标签的type属性
  - disabled （布尔类型）是否禁用
  - checked （布尔类型）是否选中 （仅radio和checkbox）
  - defaultChecked （布尔类型）是否默认选中（仅radio和checkbox）
  - form 获得input标签所属的form表单

HTMLFormElement接口 表单

- 继承自HTMLElement
- 常用属性：
  - method 对应表单的method属性
  - action 对应表单的action属性
- 常用方法
  - submit() 调用表单的提交（类似提交按钮）
  - reset() 触发表的重置（类似重置按钮)

HTMLSelectElement接口

- 继承自HTMLElement
- 常用属性
  - selectedIndex ： 可读可设置。标志标签选择的具体选项的索引。索引从0开始，0代表选择第一个option选项。
  - type： 两个值：select-multiple　select-one　选择类型。
  - options 获得所有的选项DOM对象
- 常用方法
  - remove(index ）　删除索引位置的元素
  - add(item[, before]);
    - item 是 HTMLOptionElement 类型实例
    - before可以省略，可以是一个HTMLOptionElement或是索引，添加的元素在此元素或者索引前面插入。如果是null或者索引不存在则追加在最后。
- 臭名昭著的bug
  - IE8 设置select标签的innerHTML不能用！！！shit！

HTMLOptionElement接口

- 常用属性
  - defaultSelected 是否默认选中，boolean
  - selected 是否被选中
  - text 选项的文本
  - value 选项的value属性
- 如果通过代码设置selected=true则代表选中。
- 通过text可以获取option的innerText（textContent)

案例

- 省市选择案例
- 权限选择案例
- 全选、全不选案例
- 许愿墙
- tab页签效果

document总结

文档写入

document.write()

document.open()　document.close()

文档属性 ***

title、documentElement、body、forms、links

获取标签（查找元素)***

getElementBy***() querySelector()

文档事件*****

加载完成、各种鼠标、键盘、UI等事件

文档创建编辑***

createElement、createAttibute....

文档的样式处理

样式类的操作、样式的属性。

第十二节：BOM的详解

- window对象的属性
  - navigator对象
  - location对象
  - screen对象
  - history对象

- window对象的方法
  - alert
  - confirm
  - open
  - close
  - prompt
  - print

window对象概述

- window对象是对浏览器窗口（某些方法和属性对应的是页签）的功能api的封装。
- window是浏览器中的顶层对象（也就是说js中的global对象是window）
- window常用的属性
  - console ，用于浏览器控制台的访问。
    consoe.log();
  - innerHeight/innerWidth 浏览器的宽度和高度（包括滚动条）
  - name 设置或者读取窗口的名字
- window的常用方法
  - moveTo(x,y)/moveBy(w,h) 移动窗口　兼容性不好。
  - resizeTo(w,h)/resizeBy(w,h) 调整窗口大小
  - alert 进行提醒消息的消息框。
  - confirm 进行确认操作的消息框。
  - close 关闭窗口的方法。
  - print 调出打印窗口
  - prompt 交互信息输入

window的open方法

- open方法是根据指定的参数，将一个资源加载到一个新的浏览上下文（如一个窗口）或一个已经存在的浏览上下文中。
- 语法：

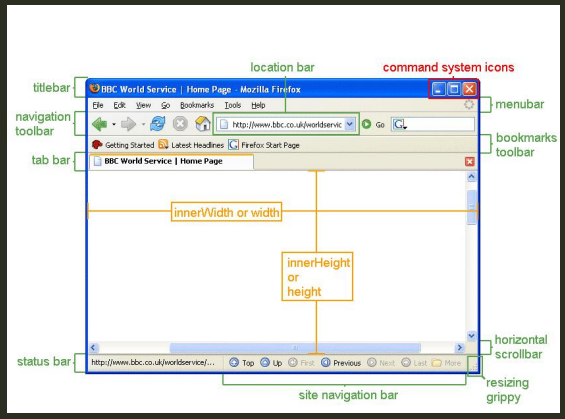
  window.open(*strUrl*, *strWindowName*, [*strWindowFeatures*]);
- 参数说明：

  •strUrl：新窗口需要载入的url地址。*strUrl*可以是web上的html页面也可以是图片文件或者其他任何浏览器支持的文件格式。

  •strWindowName：新窗口的名称。该字符串可以用来作为超链接 <a> 或表单 <form> 元素的目标属性值。字符串中不能含有空白字符。注意：*strWindowName* 并不是新窗口的标题。

  •strWindowFeatures：可选参数。是一个字符串值，这个值列出了将要打开的窗口的一些特性(窗口功能和工具栏)。 字符串中不能包含任何空白字符，特性之间用逗号分隔开。
- 例如：

  window.open("http://www.aicoder.com", " 老马博客",
  "width=420,height=230,resizable,scrollbars=yes");



screen对象

- Window.screen 返回screen对象，代表电脑的屏幕。
- 常用属性：
    - Screen.availHeight 屏幕可用的高度（去掉任务栏）
    - Screen.availWidth 屏幕可用的宽度
    - Screen.height 屏幕的高度（包括所有的）
    - Screen.width 屏幕的宽度

location对象

- window.location可用获得当前请求地址的对象。也可用document.location是一样的。
- 常用属性：
  - href ：可读可写，当前页面的url地址。
  - protocol： 协议类型 比如： “http:”
  - host：域名+端口 比如： aicoder.com aicoder.com:80
  - hostName：域名（不含端口）
  - port：端口
  - pathname： url中的路径部分， "/zh-CN/docs/Web/API/Location"
  - search：url地址中的请求参数 "?id=1&name=2"
- 常用方法：
  - Location.assign(url） 页面跳转
  - Location.reload() 重新加载页面。他有一个特殊的可选参数，类型为Boolean，该参数为true时会导致该方法引发的刷新一定会从服务器上加载数据。如果是 false或没有制定这个参数，浏览器可能从缓存当中加载页面。
  - Location.replace() 用给定的URL替换掉当前的资源。与 assign() 方法不同的是用 replace()替换的新页面不会被保存在会话的历史 History中，这意味着用户将不能用后退按钮转到该页面。
  - http://www.aicoder.com：80/index/ss.html

navigator对象

- navigator.userAgent 返回当前浏览器的一些信息。
  - chrome: "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.90 Safari/537.36"
  - IE8: "4.0(compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0)"
- onLine: 是否联网

属性或方法	说　明	IE	Firefox	Safari/Chrome	Opera
appCodeName	浏览器的名称。通常都是Mozilla，即使在非Mozilla浏览器中也是如此	3.0+	1.0+	1.0+	7.0+
appMinorVersion	次版本信息	4.0+	–	–	9.5+
appName	完整的浏览器名称	3.0+	1.0+	1.0+	7.0+
appVersion	浏览器的版本。一般不与实际的浏览器版本对应	3.0+	1.0+	1.0+	7.0+
buildID	浏览器编译版本	–	2.0+	–	–
cookieEnabled	表示cookie是否启用	4.0+	1.0+	1.0+	7.0+
cpuClass	客户端计算机中使用的CPU类型（x86、68K、Alpha、PPC或Other）	4.0+	–	–	–
javaEnabled()	表示当前浏览器中是否启用了Java	4.0+	1.0+	1.0+	7.0+
language	浏览器的主语言	–	1.0+	1.0+	7.0+
mimeTypes	在浏览器中注册的MIME类型数组	4.0+	1.0+	1.0+	7.0+
onLine	表示浏览器是否连接到了因特网	4.0+	1.0+	–	9.5+
opsProfile	似乎早就不用了。查不到相关文档	4.0+	–	–	–
oscpu	客户端计算机的操作系统或使用的CPU	–	1.0+	–	–
Platform	浏览器所在的系统平台	4.0+	1.0+	1.0+	7.0+
plugins	浏览器中安装的插件信息的数组	4.0+	1.0+	1.0+	7.0+
preference()	设置用户的首选项	–	1.5+	–	–
product	产品名称（如Gecko）	–	1.0+	1.0+	–
productSub	关于产品的次要信息（如Gecko的版本）	–	1.0+	1.0+	–

history对象

- window.history记录当前浏览器的浏览记录。
- History.back()
  - 前往上一页, 用户可点击浏览器左上角的返回按钮模拟此方法. 等价于 history.go(-1).
  - Note: 当浏览器会话历史记录处于第一页时调用此方法没有效果，而且也不会报错。
- History.forward()
  - 在浏览器历史记录里前往下一页，用户可点击浏览器左上角的前进按钮模拟此方法. 等价于 history.go(1).
  - Note: 当浏览器历史栈处于最顶端时(当前页面处于最后一页时)调用此方法没有效果也不报错。
- History.go()
  - 通过当前页面的相对位置从浏览器历史记录(会话记录)加载页面。比如：参数为-1的时候为上一页，参数为1的时候为下一页. 当整数参数超出界限时(译者注:原文为When integerDelta is out of bounds)，例如: 如果当前页为第一页，前面已经没有页面了，我传参的值为-1，那么这个方法没有任何效果也不会报错。调用没有参数的 go() 方法或者不是整数的参数时也没有效果。(这点与支持字符串作为url参数的IE有点不同)。

十三节：DOM特效封装

- 模态对话框
- 准备知识：
  - 半透明效果，标准浏览器直接用opacity。ie9+支持。值是0-1之间，1代表不透明，0代表完全透明。
  - ie8使用filter：alpha(opacity=50)；opacity是0-100之间的一个值。

DOM特效封装

- 动画系统
- setInterval定时的进行刷新页面。
- 由于时钟并不精确，必须根据时间进行校正。
- 每次进行绘制的间隔（一秒钟至少24帧才不会卡顿）

DOM特效封装

轮播图效果

DOM特效封装

手风琴效果

表单验证

- 表单的验证逻辑一般写在表单提交之前
- 两种方式校验表单：
  - 第一种：在表单的提交按钮点击事件中
  - 第二种：表单的submit事件中处理
- 表单校验可以使用正则表达式配合

动态创建表格

- 动态创建表格

老马自我介绍

老马联系方式

QQ：515154084

邮箱：malun666@126.com

微博：http://weibo.com/flydragon2010

百度传课：

https://chuanke.baidu.com/s5508922.html

Thanks

@马伦-flydragon