

Lecture 6-2

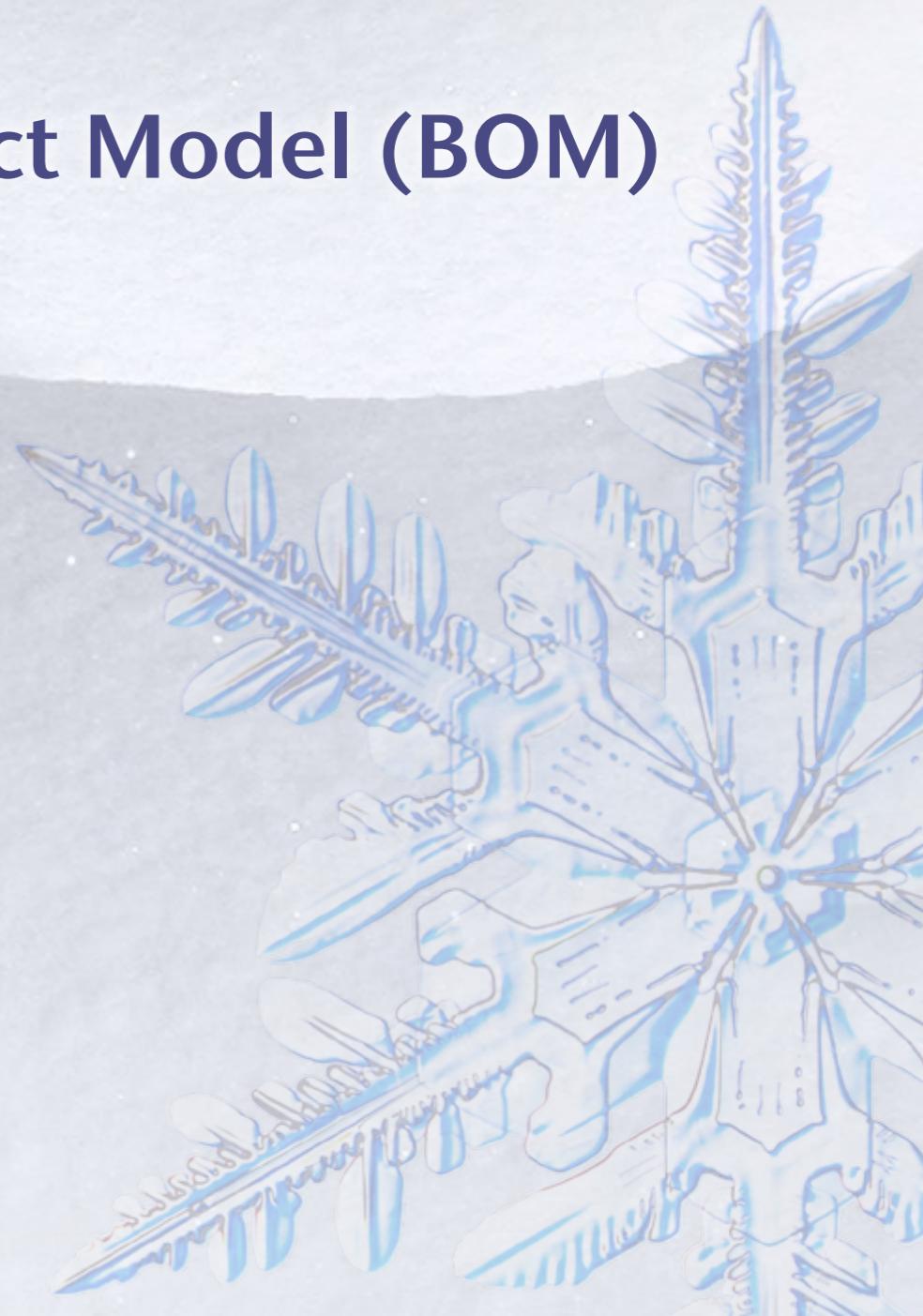
JavaScript DOM和BOM



目录

❖ ***DOM***

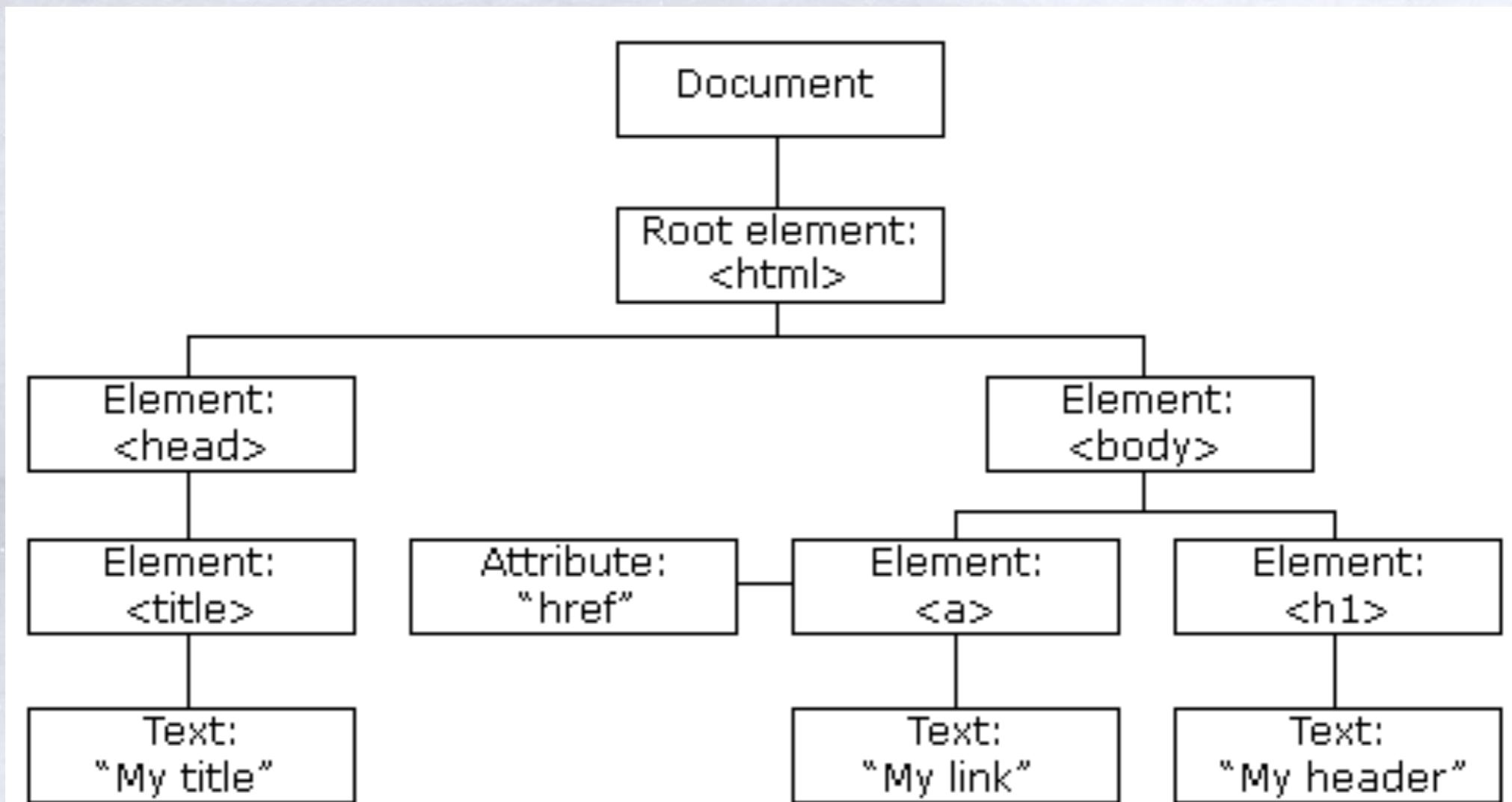
❖ **The Browser Object Model (BOM)**



DOM

* 当一个网页被加载时，浏览器会创建一个页面的文档对象模型（Document Object Model, DOM）

* HTML DOM模型被构造成一个对象树：



功能

❄ 通过对象模型，JavaScript获得了创建动态HTML所需的所有功能

- * JavaScript可以更改页面中的所有HTML元素
- * JavaScript可以更改页面中的所有HTML属性
- * JavaScript可以更改页面中的所有CSS样式
- * JavaScript可以删除现有的HTML元素和属性
- * JavaScript可以添加新的HTML元素和属性
- * JavaScript可以对页面中所有现有的HTML事件做出反应
- * JavaScript可以在页面中创建新的HTML事件

DOM是什么？

❄ DOM是W3C(万维网联盟)标准

❄ DOM定义了一个访问文档的标准：

- * W3C文档对象模型(DOM)是一个平台和语言无关的接口，它允许程序和脚本动态访问和更新文档的内容、结构和样式。

❄ W3C DOM标准分为3个不同的部分：

- * 核心DOM——所有文档类型的标准模型
- * XML DOM——XML文档的标准模型
- * HTML DOM——HTML文档的标准模型

HTML DOM

- ❖ HTML DOM是HTML的标准对象模型和编程接口。它定义了：
 - * HTML元素作为对象
 - * 所有HTML元素的属性
 - * 访问所有HTML元素的方法
 - * 所有HTML元素的事件
- ❖ 换句话说：HTML DOM是获取、更改、添加或删除HTML元素的标准。

DOM编程接口

- ❄ 可以使用JavaScript(以及其他编程语言)访问HTML DOM。
- ❄ 在DOM中，所有HTML元素都定义为对象。
- ❄ 编程接口是每个对象的属性和方法。
- ❄ 属性是可以获取或设置的值(如更改HTML元素的内容)。
- ❄ 方法是可以执行的操作(如添加或删除HTML元素)。

DOM元素

- ❄ 页面上的每个元素都有一个相应的DOM对象
- ❄ 使用`objectName.attributeName`访问/修改DOM对象的属性
- ❄ 事实上，浏览器在运行时将Web页面计算为相应的DOM对象

```
<script>
  document.getElementById("demo").innerHTML = "Hello World!";
</script>
```

查找HTML元素

❄ 通过id查找HTML元素

* `var myElement = document.getElementById("intro");`

❄ 根据标记名查找HTML元素

* `var x = document.getElementsByTagName("p");`

❄ 根据类名查找HTML元素

* `var x = document.getElementsByClassName("intro");`

❄ 通过CSS选择器查找HTML元素

* `var x = document.querySelectorAll("p.intro");`

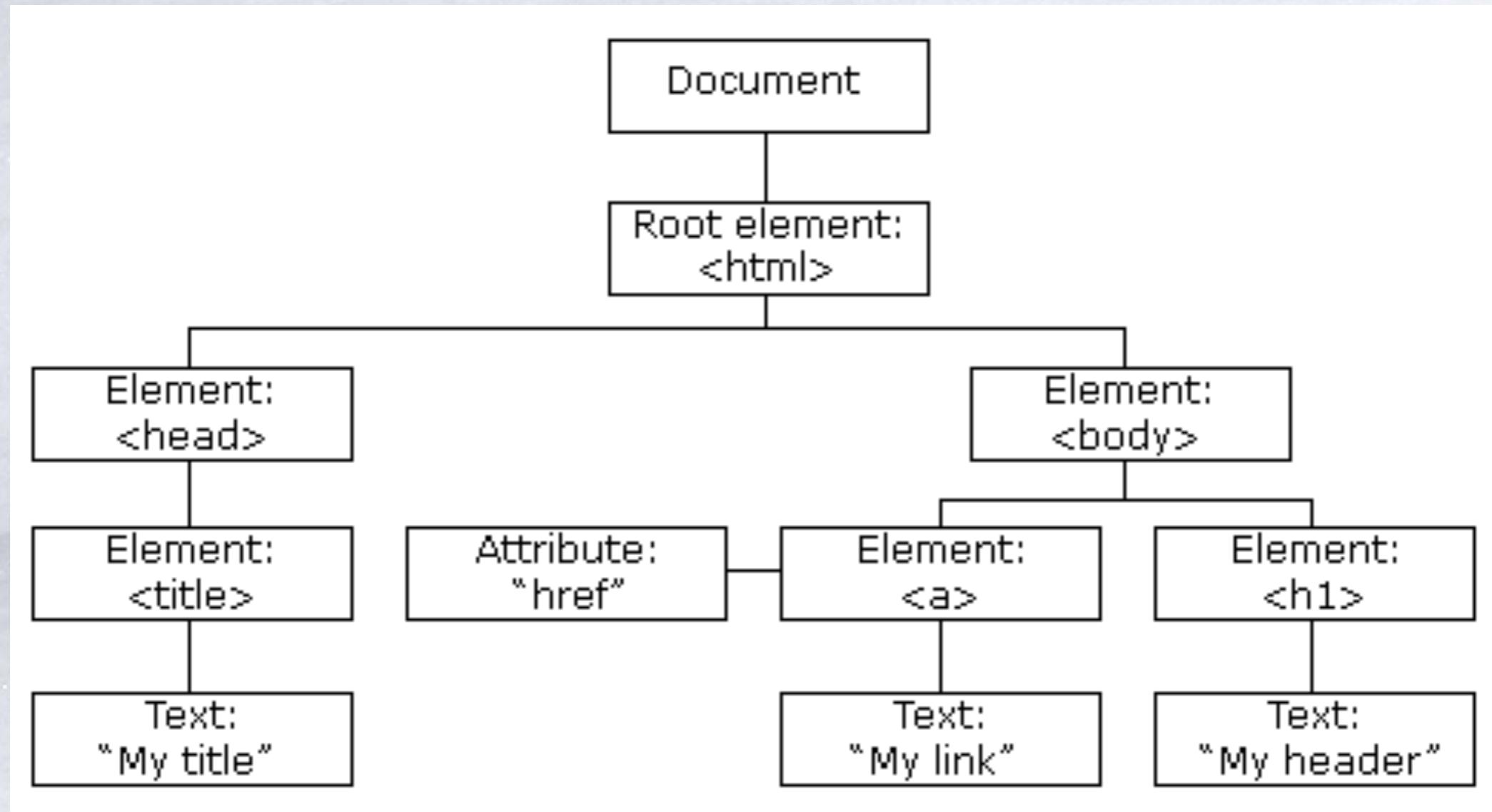
❄ 通过HTML对象集合查找HTML元素

* `Var x = document.forms["frm1"];`

JavaScript HTML DOM 导航

* 页面的元素嵌套在对象的树状结构中——DOM树

* DOM具有遍历该树的属性和方法



DOM节点类型

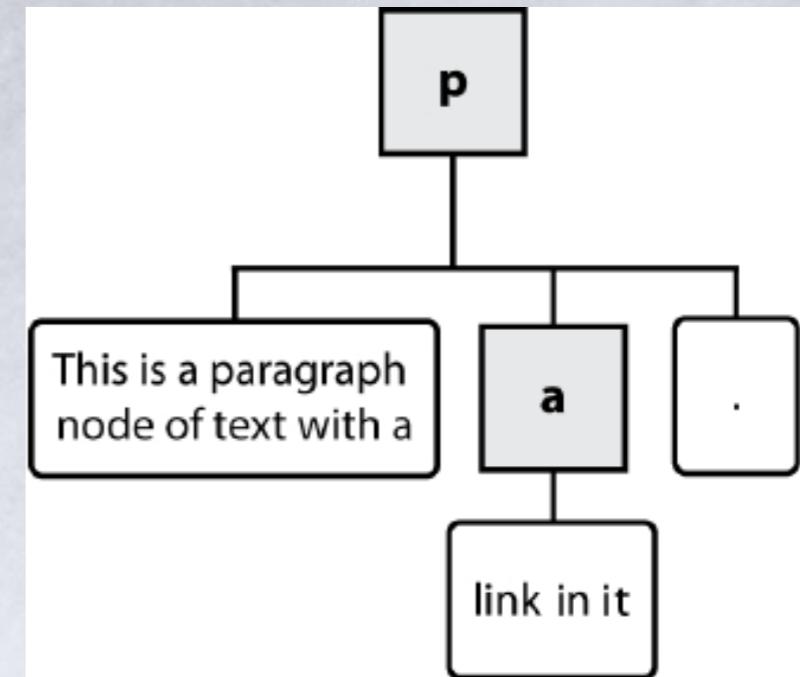
```
<p>
  This is a paragraph of text with a
  <a href="/path/page.html">link in it</a>.
</p>
```

HTML

- 每个节点都有 `nodeType` (节点类型) 、
`nodeName` (节点名称) 、`nodeValue` (节点
值) 属性

html DOM中常出现的类型

- * `Element`(元素节点)
- * `Text`(文本节点)
- * `Attr`(属性节点)
- * `Comment`(注释节点)
- * `Document`(文档节点)
- * `DocumentFragment`(文档片段节点)



遍历DOM树

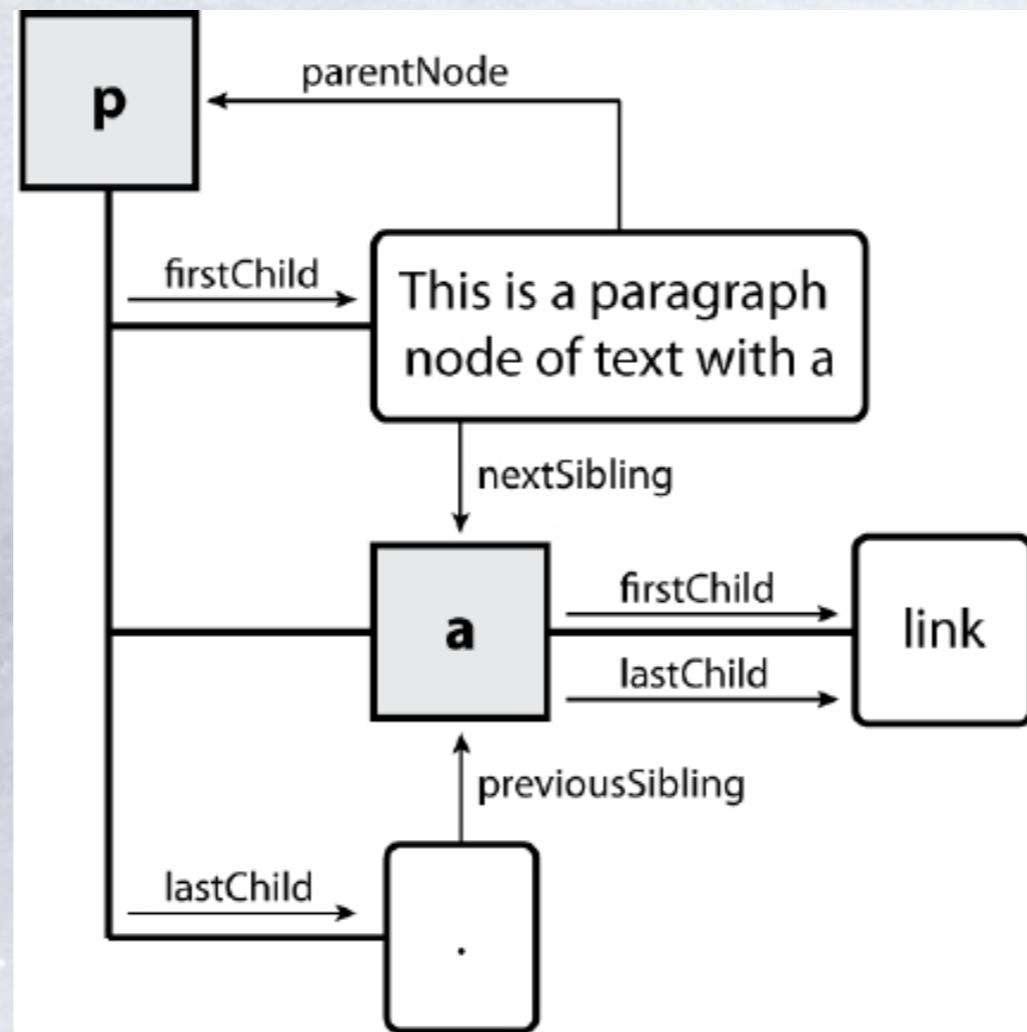
每个节点的DOM对象都有以下属性：

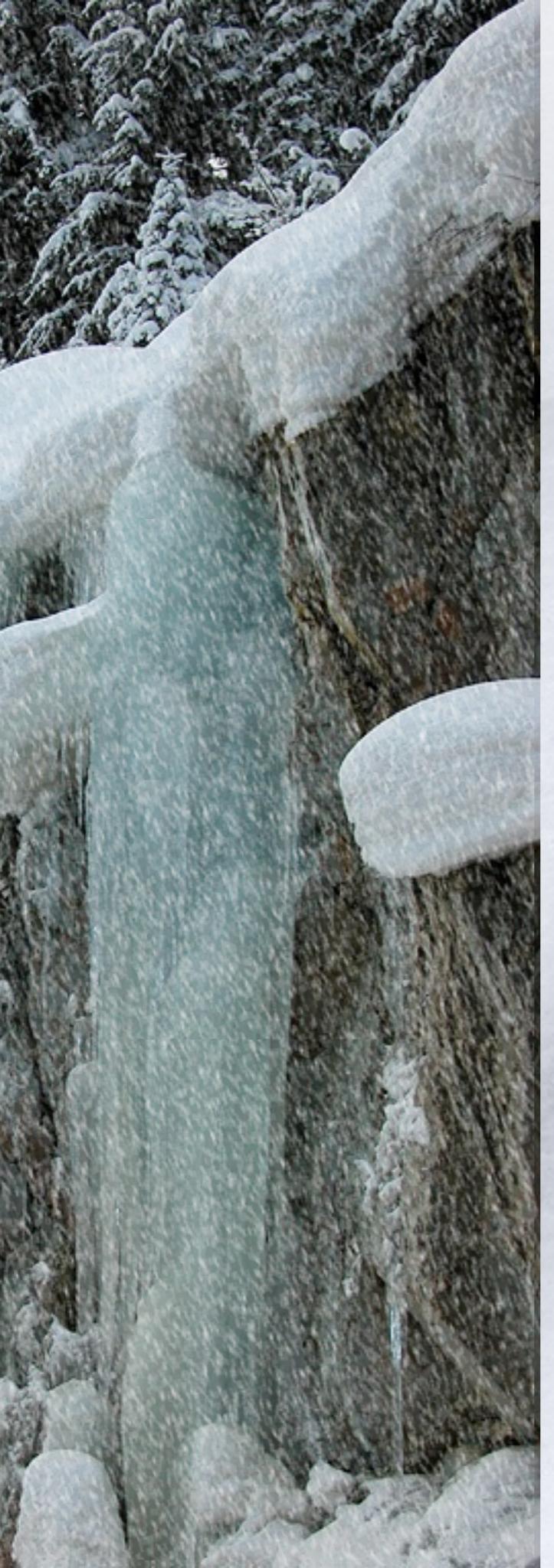
name(s)	description
<code>firstChild, lastChild</code>	<code>start/end of this node's list of children</code>
<code>childNodes</code>	<code>array of all this node's children</code>
<code>nextSibling, previousSibling</code>	<code>neighboring nodes with the same parent</code>
<code>parentNode</code>	<code>the element that contains this node</code>

浏览器不兼容问题(IE很糟糕)

DOM 树遍历实例

```
<p id="foo">This is a paragraph of text with a  
  <a href="/path/to/another/page.html">link</a>.</p>      HTML
```

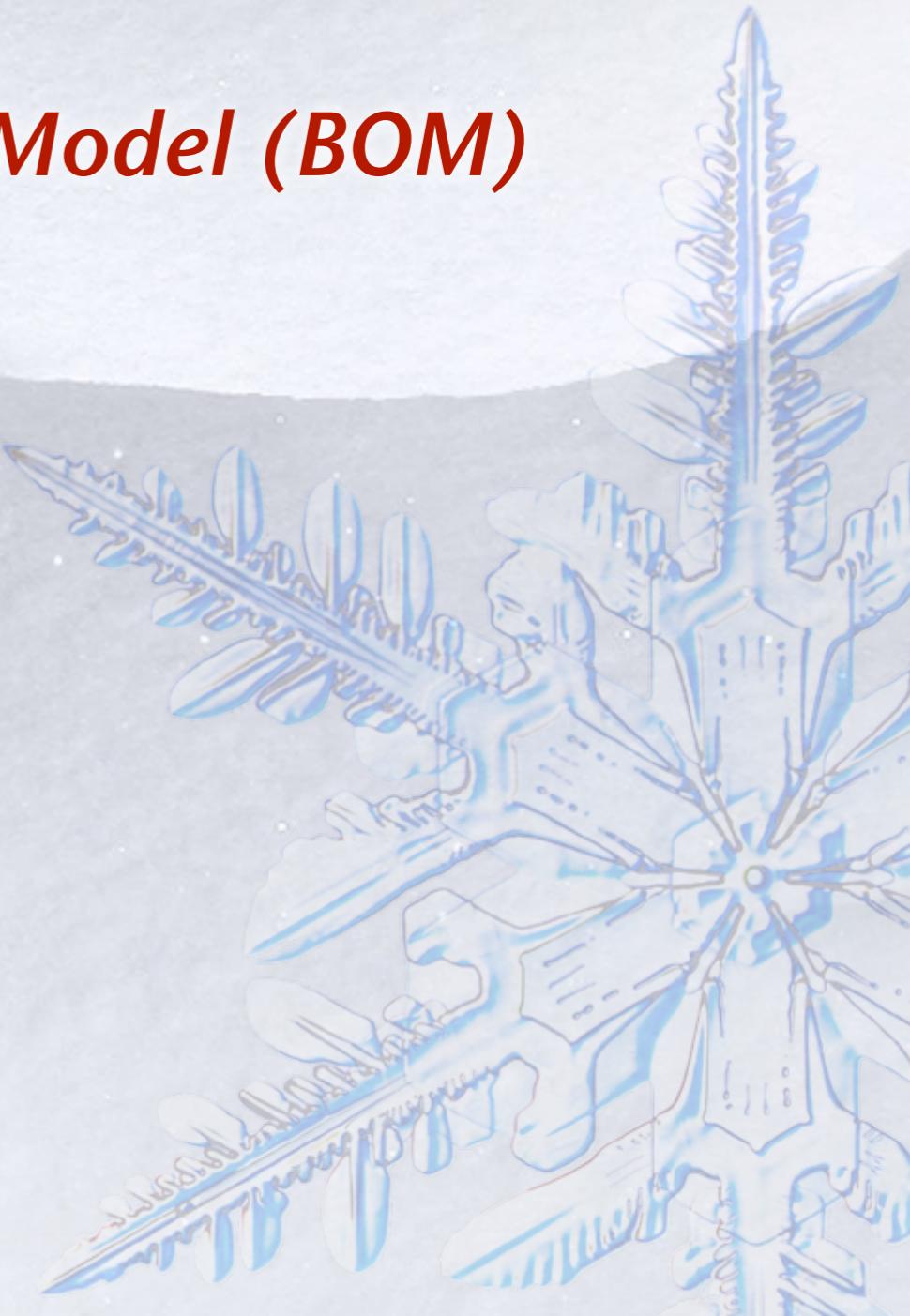




Outline

❖ DOM

❖ *The Browser Object Model (BOM)*



BOM

- ✿ 浏览器对象模型（Browser Object Model，简称 BOM）是 JavaScript 的组成部分之一，BOM 赋予了 JavaScript 程序与浏览器交互的能力。
- ✿ 每个浏览器的Javascript程序都可以引用以下全局对象：

name	description
document	current HTML page and its content
history	list of pages the user has visited
location	URL of the current HTML page
navigator	info about the web browser you are using
screen	info about the screen area occupied by the browser
window	the browser window

window 对象

- ※ 所有浏览器都支持window对象。它表示浏览器的窗口
- ※ 所有JavaScript全局对象、函数和变量都自动成为窗口对象的成员
- ※ 在客户端 JavaScript 中，Window 对象是全局对象，所有的表达式都在当前的环境中计算。也就是说，要引用当前窗口根本不需要特殊的语法，可以把那个窗口的属性作为全局变量来使用：
 - * `window.document.getElementById("header");`
 - * `document.getElementById("header");`

document 对象

- ❄ 每个载入浏览器的 HTML 文档都会成为 Document 对象
- ❄ Document 对象使我们可以从脚本中对 HTML 页面中的所有元素进行访问
- ❄ 提示：Document 对象是 Window 对象的一部分，可通过 `window.document` 属性对其进行访问
- ❄ Document 对象是 HTML 文档的根节点
- ❄ 属性：
 - * `anchors`, `body`, `cookie`, `domain`, `forms`, `images`, `links`, `referrer`, `title`, `URL`
- ❄ 方法：
 - * `getElementById`
 - * `getElementsByName`
 - * `getElementsByTagName`
 - * `close`, `open`, `write`, `writeln`

location 对象

- ✿ Location 对象包含有关当前 URL 的信息。
- ✿ Location 对象是 Window 对象的一个部分，可通过 window.location 属性来访问。
- ✿ 例子：
 - * window.location.href 设置或返回完整的 URL
 - * window.location.hostname 设置或返回当前 URL 的主机名
 - * window.location.pathname 设置或返回当前 URL 的路径部分
 - * window.location.protocol 设置或返回当前 URL 的协议。
(http:// 或 https://)
 - * window.location.assign() 加载新的文档

navigator 对象

- ❄ window.navigator 对象包含有关浏览器的信息。
- ❄ 注释：没有应用于 navigator 对象的公开标准，不过所有浏览器都支持该对象。
- ❄ 例子：
 - * navigator.appName 返回浏览器的名称
 - * navigator.onLine 返回指明系统是否处于脱机模式的布尔值
 - * navigator.appCodeName 返回浏览器的代码名
 - * navigator.platform 返回运行浏览器的操作系统平台

```
<p id="demo"></p>
<script>
  document.getElementById("demo").innerHTML =
  "Name is " + navigator.appName + ". Code name is " +
  navigator.appCodeName;
</script>
```

注意

❄ 来自navigator对象的信息通常会产生误导，不应该用于检测浏览器版本，因为：

- * 不同的浏览器可以使用相同的名称
- * 浏览器所有者可以更改浏览器数据
- * 一些浏览器为了绕过站点测试而故意错误标识自身
- * 浏览器不能报告比浏览器晚发布的新操作系统

screen 对象

- ❄ window.screen 对象包含有关客户端显示屏幕的信息
- ❄ 注释：没有应用于 screen 对象的公开标准，不过所有浏览器都支持该对象
- ❄ 属性：
 - * `screen.width` 返回显示器屏幕的宽度
 - * `screen.height` 返回显示屏幕的高度
 - * `screen.availWidth` 返回显示屏幕的宽度（除 Windows 任务栏之外）
 - * `screen.availHeight` 返回显示屏幕的高度（除 Windows 任务栏之外）
 - * `screen.colorDepth` 返回目标设备或缓冲器上的调色板的比特深度
 - * `screen.pixelDepth` 返回显示屏幕的颜色分辨率（比特每像素）

history 对象

- ❄ window.history 对象包含用户（在浏览器窗口中）访问过的 URL
- ❄ History 对象是 window 对象的一部分，可通过 window.history 属性对其进行访问。
- ❄ 为了保护用户的隐私，对 JavaScript 访问该对象的方式有限制
- ❄ 注释：没有应用于 History 对象的公开标准，不过所有浏览器都支持该对象。
- ❄ 方法：
 - * history.back() - 加载 history 列表中的前一个 URL
 - * history.forward() - 加载 history 列表中的下一个 URL

Cookies

- ✿ Cookies可以让你在网页中存储用户信息
- ✿ cookie是存储在计算机上的小文本文件中的数据
- ✿ Cookies的发明是为了解决“如何记住用户信息”的问题:
 - * 当一个用户访问一个网页时，他的名字会被存储在一个cookie中。下次用户访问该页面时，cookie就会“记住”他的名字。
- ✿ cookie以名称-值对的形式保存，例如
 - * username=Tom
- ✿ 当浏览器从服务器请求一个网页时，属于该网页的cookies被添加到请求中。通过这种方式，服务器获得必要的数据来“记住”关于用户的信息。

用JavaScript创建一个Cookie

- ❄ JavaScript可以使用 `document.cookie`属性来创建、读取和删除cookie
- ❄ 比如:
 - * `document.cookie="username=Tom";`
- ❄ 还可以添加一个过期日期(以UTC时间表示)。缺省情况下,关闭浏览器时删除cookie:
 - * `document.cookie="username=Tom; expires=Thu, 18 Sep 2015 10:00:00 UTC";`
- ❄ 通过路径参数, 可以告诉浏览器cookie属于哪个路径。缺省情况下, cookie属于当前页面。
 - * `document.cookie="username=Tom; expires=Thu, 18 Sep 2015 10:00:00 UTC; path=/";`

如何使用cookies

❄ 使用JavaScript读取一个cookie

- * `var x = document.cookie;`
- * Note `document.cookie` will return all cookies in one string much like: `cookie1=value1; cookie2=value2; cookie3=value3;`

❄ 使用JavaScript修改

- * `document.cookie="username=Tom; expires=Thu, 18 Sep 2015 10:00:00 UTC; path=/";`

❄ 用JavaScript删除Cookie，非常简单，只需将expires参数设置为一个过去的日期:

- * `document.cookie = "username=; expires=Thu, 01 Jan 1970 00:00:00 UTC";`
- * 注意，在删除cookie时不必指定cookie值。

Example

```
<script>
function setCookie(cname,cvalue,exdays) {
    var d = new Date();
    d.setTime(d.getTime() + (exdays*24*60*60*1000));
    var expires = "expires=" + d.toGMTString();
    document.cookie = cname+"="+cvalue+"; "+expires;
}
function getCookie(cname) {
    var name = cname + "=";
    var ca = document.cookie.split(';');
    for(var i=0; i<ca.length; i++) {
        var c = ca[i];
        while (c.charAt(0)==' ') c = c.substring(1);
        if (c.indexOf(name) == 0) {
            return c.substring(name.length, c.length);
        }
    }
    return "";
}
function checkCookie() {
    var user=getCookie("username");
    if (user != "") {
        alert("Welcome again " + user);
    } else {
        user = prompt("Please enter your name:","");
        if (user != "" && user != null) {
            setCookie("username", user, 30);
        }
    }
}
</script>
```

References

- ✿ JavaScript编程精解（原书第3版） [Eloquent JavaScript: A Modern Introduction to Prog]. [美] 马尔奇·哈弗贝克 (Marijn Haverbeke) 著, 卢涛 李颖 译. 机械工业出版社, 2020.
- ✿ <http://www.w3schools.com/>
- ✿ Javascript权威指南

Thanks!!!

