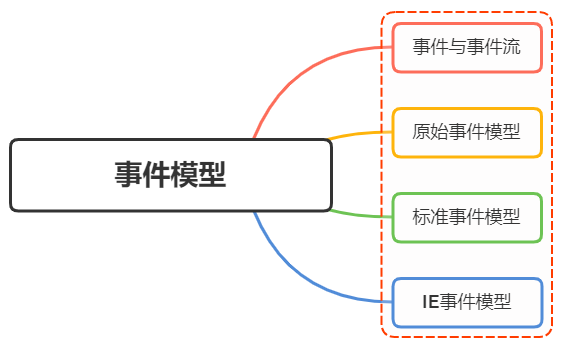
# 面试官：说说JavaScript中的事件模型



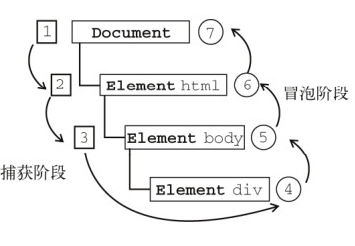
## 一、事件与事件流

javascript中的事件，可以理解就是在HTML文档或者浏览器中发生的一种交互操作，使得网页具备互动性， 常见的有加载事件、鼠标事件、自定义事件等

由于DOM是一个树结构，如果在父子节点绑定事件时候，当触发子节点的时候，就存在一个顺序问题，这就涉及到了事件流的概念

事件流都会经历三个阶段：

* 事件捕获阶段(capture phase)
* 处于目标阶段(target phase)
* 事件冒泡阶段(bubbling phase)



事件冒泡是一种从下往上的传播方式，由最具体的元素（触发节点）然后逐渐向上传播到最不具体的那个节点，也就是DOM中最高层的父节点

<!DOCTYPE html>  
<html lang="en">  
 <head>  
 <meta charset="UTF-8">  
 <title>Event Bubbling</title>  
 </head>  
 <body>  
 <button id="clickMe">Click Me</button>  
 </body>  
</html>

然后，我们给button和它的父元素，加入点击事件

var button = document.getElementById('clickMe');  
  
button.onclick = function() {  
 console.log('1.Button');  
};  
document.body.onclick = function() {  
 console.log('2.body');  
};  
document.onclick = function() {  
 console.log('3.document');  
};  
window.onclick = function() {  
 console.log('4.window');  
};

点击按钮，输出如下

1.button  
2.body  
3.document  
4.window

点击事件首先在button元素上发生，然后逐级向上传播

事件捕获与事件冒泡相反，事件最开始由不太具体的节点最早接受事件, 而最具体的节点（触发节点）最后接受事件

## 二、事件模型

事件模型可以分为三种：

* 原始事件模型（DOM0级）
* 标准事件模型（DOM2级）
* IE事件模型（基本不用）

### 原始事件模型

事件绑定监听函数比较简单, 有两种方式：

* HTML代码中直接绑定

<input type="button" onclick="fun()">

* 通过JS代码绑定

var btn = document.getElementById('.btn');  
btn.onclick = fun;

#### 特性

* 绑定速度快

DOM0级事件具有很好的跨浏览器优势，会以最快的速度绑定，但由于绑定速度太快，可能页面还未完全加载出来，以至于事件可能无法正常运行

* 只支持冒泡，不支持捕获
* 同一个类型的事件只能绑定一次

<input type="button" id="btn" onclick="fun1()">  
  
var btn = document.getElementById('.btn');  
btn.onclick = fun2;

如上，当希望为同一个元素绑定多个同类型事件的时候（上面的这个btn元素绑定2个点击事件），是不被允许的，后绑定的事件会覆盖之前的事件

删除 DOM0 级事件处理程序只要将对应事件属性置为null即可

btn.onclick = null;

### 标准事件模型

在该事件模型中，一次事件共有三个过程:

* 事件捕获阶段：事件从document一直向下传播到目标元素, 依次检查经过的节点是否绑定了事件监听函数，如果有则执行
* 事件处理阶段：事件到达目标元素, 触发目标元素的监听函数
* 事件冒泡阶段：事件从目标元素冒泡到document, 依次检查经过的节点是否绑定了事件监听函数，如果有则执行

事件绑定监听函数的方式如下:

addEventListener(eventType, handler, useCapture)

事件移除监听函数的方式如下:

removeEventListener(eventType, handler, useCapture)

参数如下：

* eventType指定事件类型(不要加on)
* handler是事件处理函数
* useCapture是一个boolean用于指定是否在捕获阶段进行处理，一般设置为false与IE浏览器保持一致

举个例子：

var btn = document.getElementById('.btn');  
btn.addEventListener(‘click’, showMessage, false);  
btn.removeEventListener(‘click’, showMessage, false);

#### 特性

* 可以在一个DOM元素上绑定多个事件处理器，各自并不会冲突

btn.addEventListener(‘click’, showMessage1, false);  
btn.addEventListener(‘click’, showMessage2, false);  
btn.addEventListener(‘click’, showMessage3, false);

* 执行时机

当第三个参数(useCapture)设置为true就在捕获过程中执行，反之在冒泡过程中执行处理函数

下面举个例子：

<div id='div'>  
 <p id='p'>  
 <span id='span'>Click Me!</span>  
 </p >  
</div>

设置点击事件

var div = document.getElementById('div');  
var p = document.getElementById('p');  
  
function onClickFn (event) {  
 var tagName = event.currentTarget.tagName;  
 var phase = event.eventPhase;  
 console.log(tagName, phase);  
}  
  
div.addEventListener('click', onClickFn, false);  
p.addEventListener('click', onClickFn, false);

上述使用了eventPhase，返回一个代表当前执行阶段的整数值。1为捕获阶段、2为事件对象触发阶段、3为冒泡阶段

点击Click Me!，输出如下

P 3  
DIV 3

可以看到，p和div都是在冒泡阶段响应了事件，由于冒泡的特性，裹在里层的p率先做出响应

如果把第三个参数都改为true

div.addEventListener('click', onClickFn, true);  
p.addEventListener('click', onClickFn, true);

输出如下

DIV 1  
P 1

两者都是在捕获阶段响应事件，所以div比p标签先做出响应

### IE事件模型

IE事件模型共有两个过程:

* 事件处理阶段：事件到达目标元素, 触发目标元素的监听函数。
* 事件冒泡阶段：事件从目标元素冒泡到document, 依次检查经过的节点是否绑定了事件监听函数，如果有则执行

事件绑定监听函数的方式如下:

attachEvent(eventType, handler)

事件移除监听函数的方式如下:

detachEvent(eventType, handler)

举个例子：

var btn = document.getElementById('.btn');  
btn.attachEvent(‘onclick’, showMessage);  
btn.detachEvent(‘onclick’, showMessage);