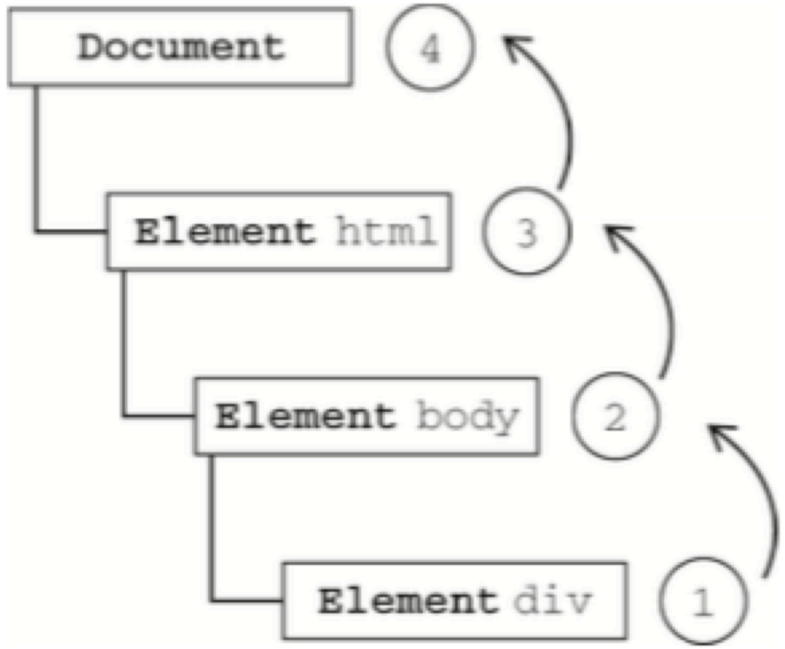
事件

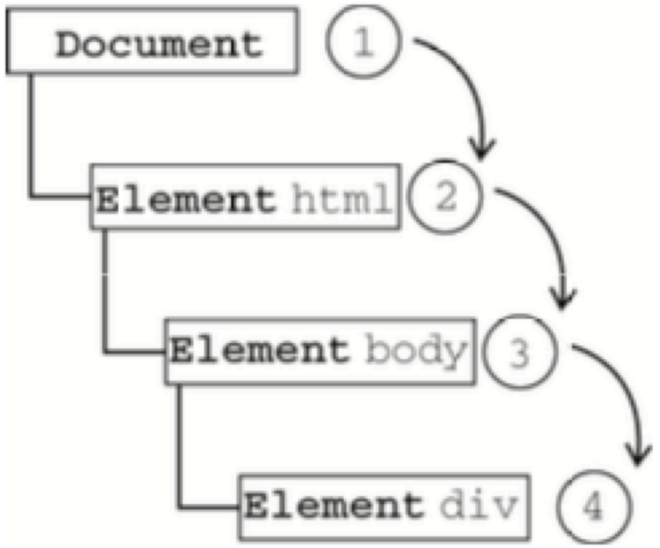
事件流
事件处理
事件对象
事件类型
事件代理

事件流

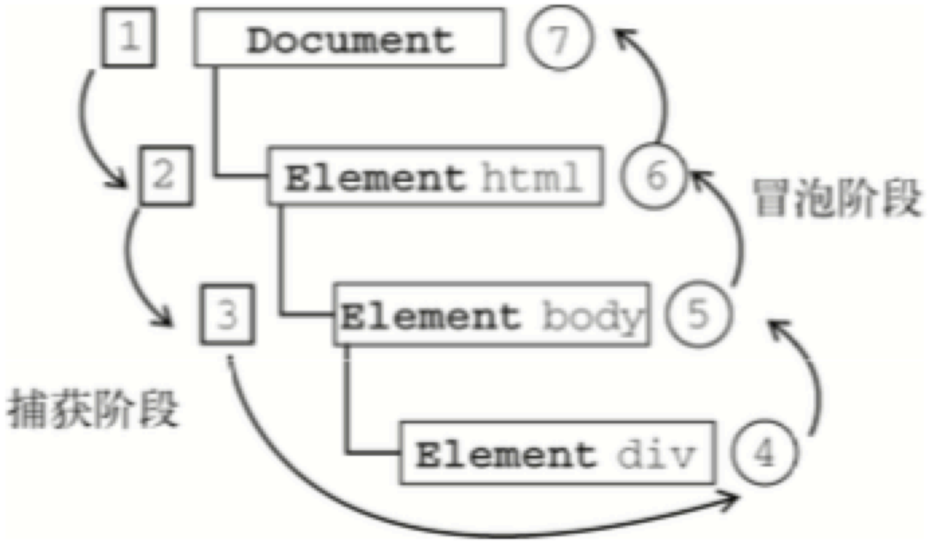
事件流 – 冒泡



事件流 – 捕获



事件流 – DOM事件流


1
Document
7
2
Element html
6
冒泡阶段
3
Element body
5
捕获阶段
Element div
4

事件处理

事件处理 – HTML

```
<input type="button" value="Click Me" onclick="alert('Clicked')" />
```

事件处理 – HTML

```
<script type="text/javascript">
    function showMessage(){
        alert("Hello world!");
    }
</script>
<input type="button" value="Click Me" onclick="showMessage()" />
```

事件处理 – DOM0

```
var btn = document.getElementById("myBtn");
btn.onclick = function(){
    alert("Clicked");
};
```

事件处理 – DOM2

```
var btn = document.getElementById("myBtn");
btn.addEventListener("click", function(){
    alert("Hello world!");
}, false);
```

事件处理 – DOM2

```
var btn = document.getElementById("myBtn");
btn.addEventListener("click", function(){
    alert("Hello world!");
}, false);
btn.addEventListener("click", function(){
    alert("Hello world2!");
}, false);
```

事件处理 – IE

```
var btn = document.getElementById("myBtn");
btn.attachEvent("onclick", function(){
    alert("Clicked");
});
```

事件处理 – IE

```
var btn = document.getElementById("myBtn");
btn.attachEvent("onclick", function(){
    alert("Clicked");
});
btn.attachEvent("onclick", function(){
    alert("Hello world!");
});
```

事件处理 – 删除事件处理

```
var btn = document.getElementById("myBtn");
btn.addEventListener("click", function(){
    alert("Hello world!");
}, false);
btn.removeEventListener("click", function(){
    alert("Hello world!");
}, false);
```

```
var btn = document.getElementById("myBtn");
btn.attachEvent("onclick", function(){
    alert("Hello world!");
});
btn.detachEvent("onclick", function(){
    alert("Hello world!");
});
```

事件处理 – 删除事件处理

```
var btn = document.getElementById("myBtn");
var handler = function(){
    alert("Hello world!");
};
btn.addEventListener("click", handler, false);
btn.removeEventListener("click", handler, false);
```

```
var btn = document.getElementById("myBtn");
var handler = function(){
    alert("Hello world!");
};
btn.attachEvent("onclick", handler);
btn.detachEvent("onclick", handler);
```

事件处理 – 作用域

```
<div title="hello" onclick="alert(this.title)">Show Title HTML</div>
<div title="hello" onclick="showTitle()">Show Title HTML Handler</div>
<div title="hello" id="titleDiv">Show Title DOM0</div>
<div title="hello" id="titleDiv2">Show Title DOM2</div>
<script>
    function showTitle() {
        alert(this.title);
    }
    var targetDiv = document.getElementById('titleDiv');
    targetDiv.onclick = function() {
        alert(this.title);
    };
    var targetDiv2 = document.getElementById('titleDiv2');
    targetDiv2.addEventListener("click", function() {
        alert(this.title);
    }, false);
</script>
```

hello

undefined

hello

hello

事件处理 – 作用域

```
var targetDiv = document.getElementById('titleDiv');
targetDiv.attachEvent("onclick", function() {
    alert(this.title);
});
```

undefined

事件处理 – 跨浏览器

```
var EventUtil = {
    addHandler: function(element, type, handler){
        if (element.addEventListener){
            element.addEventListener(type, handler, false);
        } else if (element.attachEvent){
            element.attachEvent("on" + type, handler);
        } else {
            element["on" + type] = handler;
        }
    },
    removeHandler: function(element, type, handler){
        if (element.removeEventListener){
            element.removeEventListener(type, handler, false);
        } else if (element.detachEvent){
            element.detachEvent("on" + type, handler);
        } else {
            element["on" + type] = null;
        }
    }
};
```

事件对象

事件对象

```
var btn = document.getElementById("myBtn");
btn.onclick = function(event){
    alert(event.type); //"click"
};
btn.addEventListener("click", function(event){
    alert(event.type); //"click"
}, false);
```

事件对象 – target

```html
<body>
    <input type="button" id="myBtn"/>
</body>
```

```js
var btn = document.getElementById("myBtn");
btn.onclick = function(event){
    alert(event.currentTarget === this); //true
    alert(event.target === this); //true
};

document.body.onclick = function(event){
    alert(event.currentTarget === document.body); //true
    alert(this === document.body); //true
    alert(event.target === document.getElementById("myBtn")); //true
};
```

事件对象 – type

```
var btn = document.getElementById("myBtn");
var handler = function(event){
    switch(event.type){
        case "click":
            alert("Clicked");
            break;
        case "mouseover":
            event.target.style.backgroundColor = "red";
            break;
        case "mouseout":
            event.target.style.backgroundColor = "";
            break;
    }
};
btn.onclick = handler;
btn.onmouseover = handler;
btn.onmouseout = handler;
```

事件对象 – preventDefault

```
var link = document.getElementById("myLink");
link.onclick = function(event){
    event.preventDefault();
};
```

事件对象 – stopPropagation

```
var btn = document.getElementById("myBtn");
    btn.onclick = function(event){
        alert("Clicked");
        event.stopPropagation();
};
document.body.onclick = function(event){
    alert("Body clicked");
};
```

事件对象 – eventPhase

```
var btn = document.getElementById("myBtn");
btn.onclick = function(event){
    alert(event.eventPhase); //2
};
document.body.addEventListener("click", function(event){
    alert(event.eventPhase); //1
}, true);
document.body.onclick = function(event){
    alert(event.eventPhase); //3
};
```

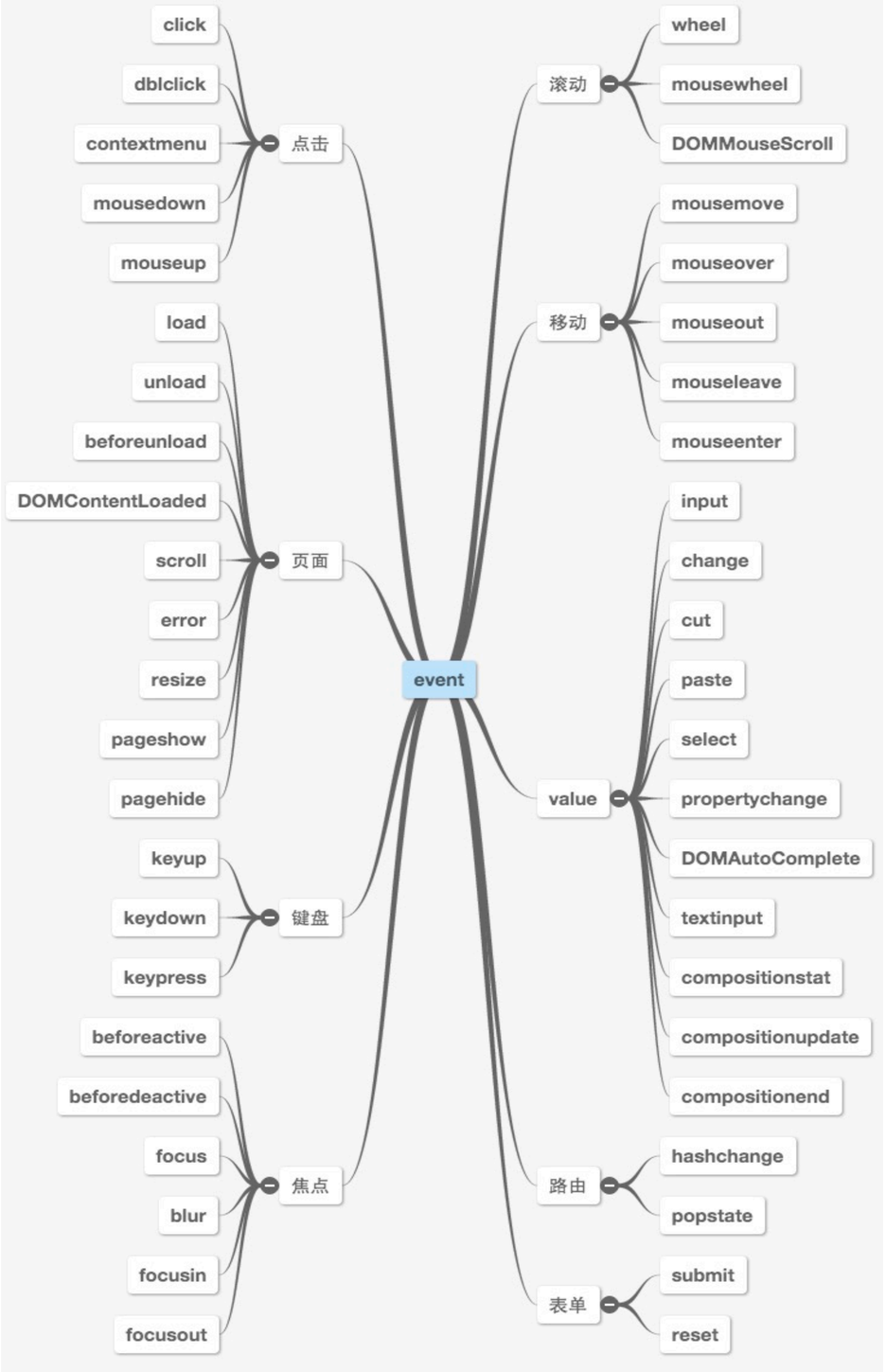
事件对象 – 跨浏览器

略

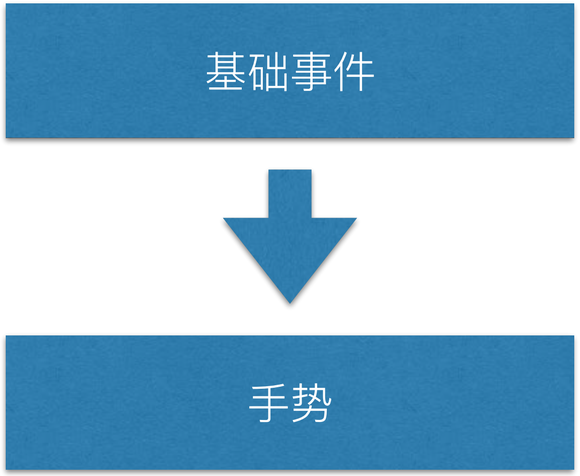
事件类型

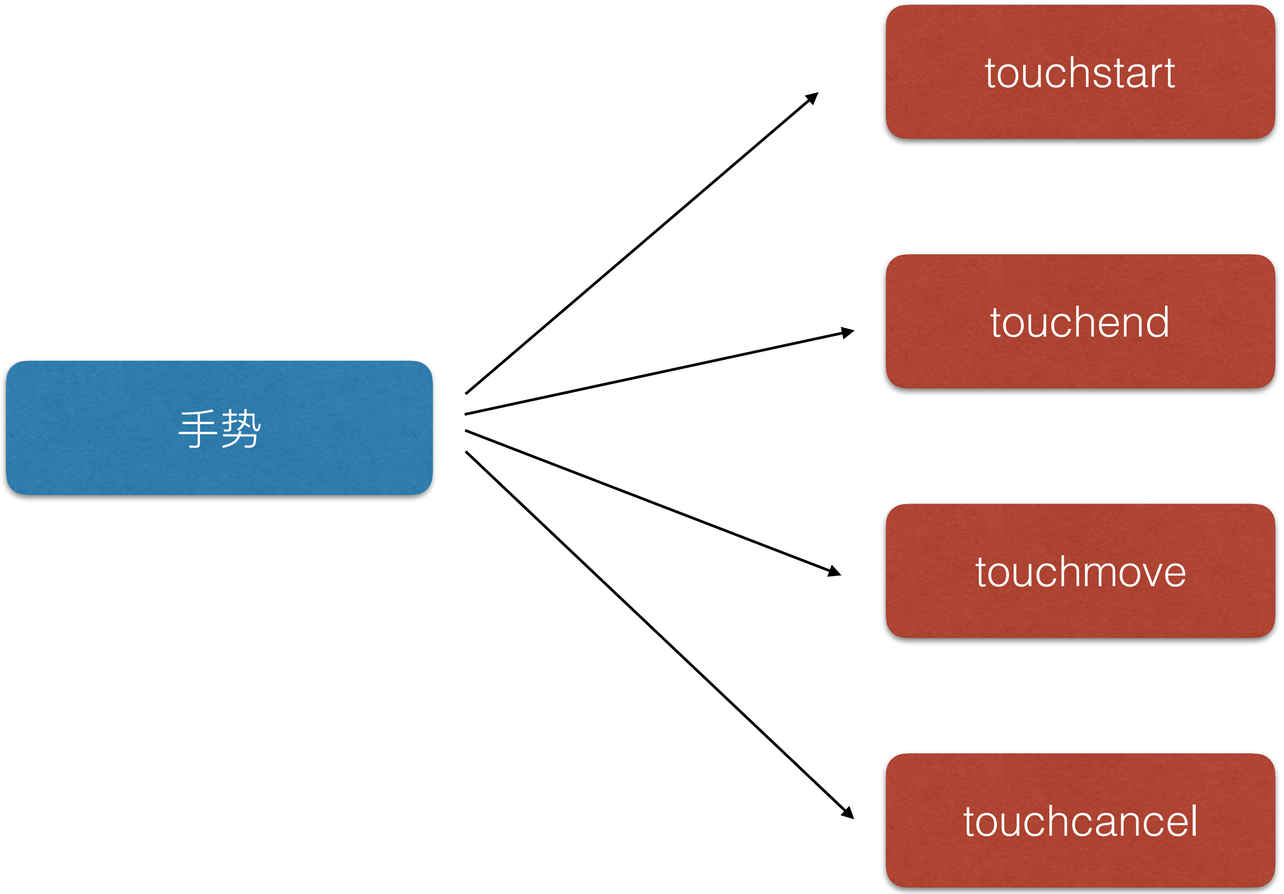
事件类型 – PC

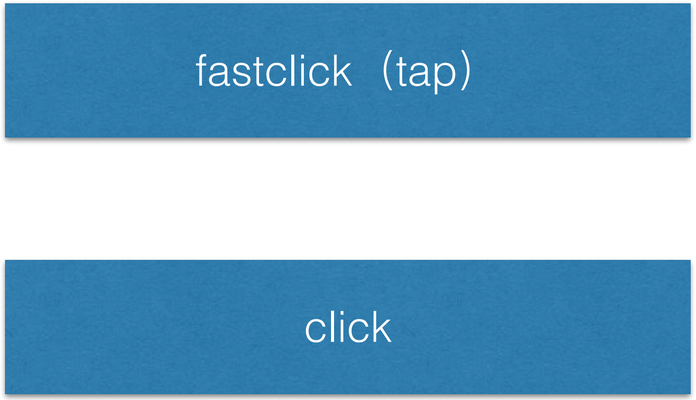
- UI（User Interface，用户界面）事件，当用户与页面上的元素交互时触发；
- 焦点事件，当元素获得或失去焦点时触发；
- 鼠标事件，当用户通过鼠标在页面上执行操作时触发；
- 滚轮事件，当使用鼠标滚轮（或类似设备）时触发；
- 文本事件，当在文档中输入文本时触发；
- 键盘事件，当用户通过键盘在页面上执行操作时触发；
- 合成事件，当为 IME（Input Method Editor，输入法编辑器）输入字符时触发；
- 变动（mutation）事件，当底层 DOM 结构发生变化时触发。


event
点击
click
dblclick
contextmenu
mousedown
mouseup
页面
load
unload
beforeunload
DOMContentLoaded
scroll
error
resize
pageshow
pagehide
键盘
keyup
keydown
keypress
焦点
beforeactive
beforedeactive
focus
blur
focusin
focusout
滚动
wheel
mousewheel
DOMMouseScroll
移动
mousemove
mouseover
mouseout
mouseleave
mouseenter
value
input
change
cut
paste
select
propertychange
DOMAutoComplete
textinput
compositionstat
compositionupdate
compositionend
路由
hashchange
popstate
表单
submit
reset

事件类型 – 移动设备




手势
touchstart
touchend
touchmove
touchcancel


fastclick（tap）
click

事件类型 - 自定义

```
document.createEvent
```

```
new Event
```

```
new CustomEvent
```

事件代理

事件代理

```
<ul id="myLinks">
    <li id="goSomewhere">Go somewhere</li>
    <li id="doSomething">Do something</li>
    <li id="sayHi">Say hi</li>
</ul>
```

事件代理 – 传统方式

```
var item1 = document.getElementById("goSomewhere");
var item2 = document.getElementById("doSomething");
var item3 = document.getElementById("sayHi");
EventUtil.addHandler(item1, "click", function(event){
    location.href = "http://www.wrox.com";
});
EventUtil.addHandler(item2, "click", function(event){
    document.title = "I changed the document's title";
});
EventUtil.addHandler(item3, "click", function(event){
    alert("hi");
});
```

事件代理 – 统一处理

```
var list = document.getElementById("myLinks");
EventUtil.addHandler(list, "click", function(event){
    event = EventUtil.getEvent(event);
    var target = EventUtil.getTarget(event);
    switch(target.id) {
        case "doSomething":
            document.title = "I changed the document's title";
            break;
        case "goSomewhere":
            location.href = "http://www.wrox.com";
            break;
        case "sayHi":
            alert("hi");
            break;
    }
});
```