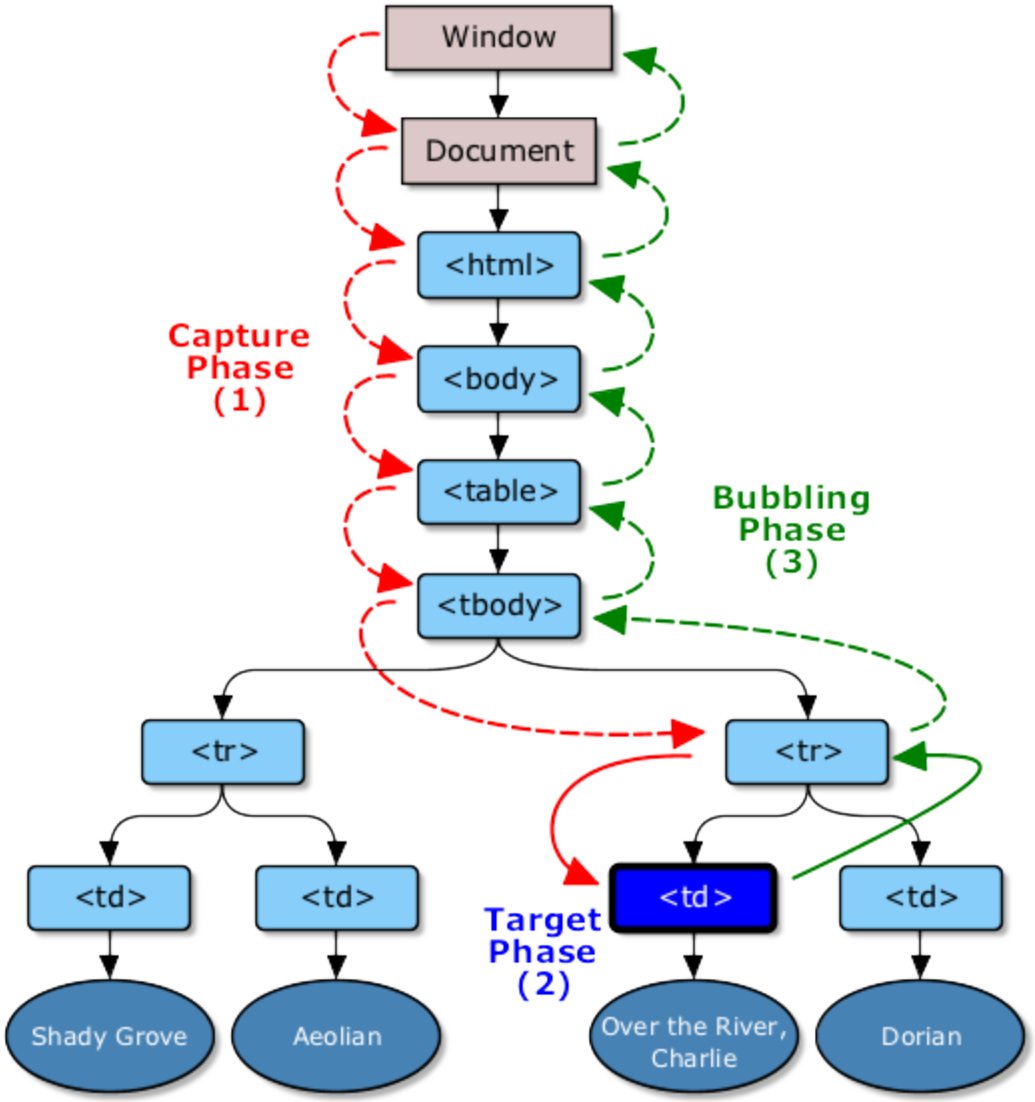
Javascript_3

Event 컨트롤

Event란?

- javascript 에서 이벤트란 브라우져안에서 취해지는 행동들의 집합체
- 브라우져에서 프로그래밍(core)한것을 사용자가 브라우져에서 액션을 취하면(EVENT) 페이지가 업데이트(DOM)된다.
- Event flow



Evnet 종류

마우스 이벤트	키보드 이벤트	인터페이스 이벤트
마우스에서 발행하는 액션	키에서 발행하는 액션	페이지나 어떤 특정요소에서 발행하는 액션
click, mousedown/up, mouseover/out, mousemove, mouseenter/leave	keydown/up, keypress	focus/blur, change, submit, reset, load/unload, resize, scroll

mouseover는 지정한 곳에 마우스가 올라가면 계속발생하지만

mouseenter는 지정한 곳에 마우스가 올라가면 한번 발생하고, 지정한 곳을 나가는 순간 mouseleave 이벤트가 발생한다

Event handling

- 기본적으로 사용자가 설정한 이벤트를 먼저 실행한 후 기본 브라우져동작이 이루어진다.

```
<a href="http://daum.net" onclick="return confirm('go to Daum?');">go to Daum</a>
```

- inline handing 과 Dom EventListen handling 방법이 있습니다.

inline Event handling

```
<html>
<head>
<title>An event</title></head>
<body onload="alert('페이지 로딩완료');">
<form onsubmit="custumSubmit(); return false;">
  <select onchange="if( this.value === 'red' ) { alert( 'red!' ); }">
    <option value="blue">blue</option>
    <option value="red">red</option>
  </select>
<input type="button" value="Click here to be alerted" onclick="customFunc(); return false;" />
</form>
</body>
</html>
```

Dom EventListen handling

- Dom Level1 (전통모델)

```
<a id="daumLink" href="http://daum.net">Daum</a>

var link = document.getElementById('daumLink');

function clickTest() {
    alert('daum');
}

link.onclick = testClick;

link.onclick = function() {
  alert('daum');
}

// 이벤트 삭제
link.onclick = null;
```

- Dom Level2

w3c모델과 MS 모델의 차이

```
<a id="daumLink" href="http://daum.net">Daum</a>

function clickTest() {
    alert('daum');
}
var link = document.getElementById('daumLink');
// W3C 모델(비IE)
link.addEventListener('click', clickTest, false );
link.removeEventListener('click', clickTest, false );

// MS 모델
link.attachEvent('onclick', clickTest);
link.detachEvent('onclick', clickTest);
```

```
<a href="http://daum.net">Daum</a>
<a href="http://tistory.com">Tistory</a>

<script type="text/javascript">

function addEvent( obj, evt, func ) {
  obj.addEventListener ? obj.addEventListener(evt, func, false) : obj.attachEvent('on' + evt, func);
}

function testClick() {
  alert('link');
}

var links = document.getElementsByTagName('a');
for (var i = 0, l = links.length; i < l; i++) {
    addEvent(links[i], 'click', testClick);
}

// 이벤트 삭제(w3c 모델)
links[0].removeEventListener('click', testClick, false);
// 이벤트 삭제(MS 모델)
links[0].detachEvent('onclick', testClick);
</script>
```

- jQuery 방식의 이벤트 탑제 및 해제

```
<div class="wrap">
  <span class="daum">Daum</span>
  <span class="tistory">Tistory</span>
</div>

<script type="text/javascript">

function testClick(event) {
  console.log(this);
  console.log(event.data.tag);
}

jQuery('.wrap span').on('click', testClick);
jQuery('.wrap').on('click', 'span', {tag: 'span'}, testClick);
jQuery('.wrap span').off(); // span에 대한 모든 이벤트 해제
jQuery('.wrap span').off('click'); // span에 대한 click 이벤트 해제


jQuery('.wrap').on('click', testClick);// .clickGroupClass 이라는 클래스 그룹화
jQuery('.wrap').on('mouseover.clickGroupClass', testClick);// .clickGroupClass 이라는 클래스 그룹화
jQuery('.wrap').on('mouseout.clickGroupClass', testClick);// .clickGroupClass 이라는 클래스 그룹화
jQuery('.wrap').off('.clickGroupClass');//.clickGroupClass 이라는 클래스 그룹의 이벤트만 제거


jQuery(function() {console.log('Dom reloaded')});// document가 모두 로딩됐을때 실행 onload와 타이밍이
다르다.
```

전통방식과 DOM Level2와의 차이점과 알아둘것

- 전통방식은 크로스 브라우징이 필요없다. (알아서 인식)
- 이벤트 핸들링의 간섭이 예상되면 DOM Level2를 사용하고 이벤트 핸들링을 총괄해서 사용한다면 전통방식을 사용한다.(현재 많은 부분은 Level2를 사용)
- MS 모델은 기본적으로 캡쳐링을 지원하지 않는다.

Event bubbling 을 통한 개발방법

```
<html>
  <body>
    <div id="div">Click Me</div>
    <ul id="service">
      <li class="daum">daum</li>
      <li class="google">google</li>
      <li class="facebook">facebook</li>
    </ul>
  </body>
</html>

var body = document.body;
var div = document.getElementById('div');
var service = document.getElementById('service');

addEvent(body, 'click', function() { alert('body'); });
addEvent(div, 'click', function() { alert('click me'); });

function alertService(e) {
  var e = e || window.event;
  if (e.target.className = 'daum') {
    alert(target.innerHTML);
  } else if () {
  .....
}
addEvent(service, 'click', alertService);
```

이벤트 객체

- 브라우저는 이벤트가 발생하면 이벤트객체를 자동으로 생성한다.
- W3C모델에서 이벤트 객체는 이벤트 핸들러 함수의 첫번째 인자로 전달되고, MS모델에서 이벤트 객체는 window.event이다.

```
function EventHandler(e) {
  var e = e || window.event;
  console.log(e);
  console.log(e.type);
  console.log(e.target);
}
```

Event 전파/기본동작 취소

```
function preventDefault(e) {
 if (e.preventDefault) {
  e.preventDefault(); // w3c 기본동작 취소
 } else {
  e.returnValue = false; // MS 기본동작 취소
 }
}

function stopPropagation(e) {
 if (e.stopPropagation) {
  e.stopPropagation();  // w3c Event 전파 취소
 } else {
  e.cancelBubble = true; // MS Event 전파 취소
 }
}

function EventHandler(e) {
   var e = e || window.event;
   preventDefault(e);
   stopPropagation(e);
}
```

- jQuery 방식

```
function EventHandler(e) {// 항상 첫번째 인자로 이벤트를 전달한다.
   e.preventDefault();
   e.stopPropagation();
   alert('body');
}

jQuery('body').on('click', EventHandler);
```

마우스 이벤트 객체

좌표정보	기준점
clientX, clientY	브라우져창
layerX, layerY	절대위치가 가까운 부모요소, 없으면 document
offsetX, offsetY	이벤트타겟
pageX, pageY	문서
screenX, screenY	컴퓨터 화면

- 마우스 위치 알아내기

```
<div style="position:absolute; top:1000px;left:0; width:300px;height:300px;background:red;"
onclick="getCoords(event); return false;">문서로부터의 거리</div>

<script type="text/javascript">
function getCoords(e) {
  var x = y = 0;
  var e = e || window.event;
  if (e.pageX || e.pageY){ //W3C
    x = e.pageX;  y = e.pageY;
  } else if (e.clientX || e.clientY) { //IE
    x = e.clientX + document.body.scrollLeft + document.documentElement.scrollLeft;
    y = e.clientY + document.body.scrollTop + document.documentElement.scrollTop;
  }
  alert(x + ', ' + y);
}
</script>
```

Key Event

- 키 코드
  - http://unixpapa.com/js/testkey.html

onload, DOMContentLoaded의 차이

- onload 는 document의 모든 요소의 로딩이 완료된상태
- DOMContentLoaded document의 모든 dom 요소가 호출을 완료한상태
- http://ie.microsoft.com/testdrive/HTML5/DOMContentLoaded/