* **Drag**드래그하고 싶은 엘리먼트에 드래그 가능여부 속석을 추가하고 드래그시 발생할 이벤트를 등록한다.  
  + 먼저 드래그 하고 싶은 엘리먼트에 **draggable 속성을 true**로 작성한다.   
    <div id="draggable" draggable="true"></div>  
    아직 draggable는 드래그 되는것 외에는 어떠한 기능도 없다.
  + **ondragstart** 이벤트를 등록한다.  
    <div id="draggable" draggable="true" ondragstart="drag()"></div>  
    ondragstart이벤트를 통해 앨리먼트가 드래그 되면 drag() 함수를 호출한다.
* **Drop**  
  드래그한 엘리먼트를 드롭받을 엘리먼트를 정할 수 있다.  
  드롭받는 엘리먼트에는 **ondrop, ondragover** 두가지 이벤트를 등록할 수 있다.  
  + ondragover는 드래그된 엘리먼트가 드랍될 수 있을때 발생한다.  
    <div id="droppable" ondragover="ondrag(event)"></div>  
    기본적으로 엘리먼트는 다른 엘리먼트로 드롭될 수 없기때문에 기본 이벤트를 제한해야한다.  
    ondrag함수내에 event.preventDefault()를 작성하면 ondragover의 디폴트 이벤트가 호출되지 않는다.
  + 드래그한 엘리먼트를 drop할때 발생한 이벤트를 등록한다.  
    <div id="droppable" ondrop="drop(event)"></div>  
    드래그된 엘리먼트가 droppable에 drop되면 drop()함수가 호출된다.
* javascript 만으로 drag&drop자체의 구현은 어렵지 않지만 다양한 기능을 사용하기위해서는 브라우져별로 구현해줘야 할 것들이 많다. 브라우저에 영향없이 다양하고 강력한 기능이 있는 jQueryui의 dragNdrop에 대해서 알아보겠다.

Make an Element Draggable

First of all: To make an element draggable, set the draggable attribute to true:

<img draggable="true">

## What to Drag - ondragstart and setData()

The dataTransfer.setData() method sets the data type and the value of the dragged data:

function drag(ev) {  
    ev.dataTransfer.setData("text", ev.target.id);  
}

In this case, the data type is "text" and the value is the id of the draggable element ("drag1").

## Where to Drop - ondragover

The ondragover event specifies where the dragged data can be dropped.

By default, data/elements cannot be dropped in other elements. To allow a drop, we must prevent the default handling of the element.

This is done by calling the event.preventDefault() method for the ondragover event:

*event*.preventDefault()

## Do the Drop - ondrop

When the dragged data is dropped, a drop event occurs.

In the example above, the ondrop attribute calls a function, drop(event)

function drop(ev) {  
    ev.preventDefault();  
    var data = ev.dataTransfer.getData("text");  
    ev.target.appendChild(document.getElementById(data));  
}

Execute a JavaScript when the user has finished dragging a <p> element:

**Ex01)**

<!DOCTYPE HTML>

<html>

<head>

<style>

#div1 {

width: 350px;

height: 70px;

padding: 10px;

border: 1px solid #aaaaaa;

}

</style>

<script>

function allowDrop(ev) {

ev.preventDefault();

}

function drag(ev) {

ev.dataTransfer.setData("Text", ev.target.id);

}

function drop(ev) {

var data = ev.dataTransfer.getData("Text");

ev.target.appendChild(document.getElementById(data));

ev.preventDefault();

}

</script>

</head>

<body>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>

<br>

<p id="drag1" draggable="true" ondragstart="drag(event)">This is a draggable paragraph. Drag this element into the rectangle.</p>

</body>

</html>

Ex02)

<!DOCTYPE HTML>

<html>

<head>

<style>

#div1, #div2

{float:left; width:100px; height:35px; margin:10px;padding:10px;border:1px solid #aaaaaa;}

</style>

<script>

function allowDrop(ev) {

ev.preventDefault();

}

function drag(ev) {

ev.dataTransfer.setData("text", ev.target.id);

}

function drop(ev) {

ev.preventDefault();

var data = ev.dataTransfer.getData("text");

ev.target.appendChild(document.getElementById(data));

}

</script>

</head>

<body>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)">

<img src="img\_w3slogo.gif" draggable="true" ondragstart="drag(event)" id="drag1" width="88" height="31">

</div>

<div id="div2" ondrop="drop(event)" ondragover="allowDrop(event)"></div>

</body>

</html>

**Ex03)**

<!DOCTYPE HTML>

<html>

<head>

<style>

.droptarget {

float: left;

width: 100px;

height: 35px;

margin: 15px;

padding: 10px;

border: 1px solid #aaaaaa;

}

</style>

</head>

<body>

<p>Drag the p element back and forth between the two rectangles:</p>

<div class="droptarget" ondrop="drop(event)" ondragover="allowDrop(event)">

<p ondragstart="dragStart(event)" draggable="true" id="dragtarget">Drag me!</p>

</div>

<div class="droptarget" ondrop="drop(event)" ondragover="allowDrop(event)"></div>

<p style="clear:both;"><strong>Note:</strong> drag events are not supported in Internet Explorer 8 and earlier versions or Safari 5.1 and earlier versions.</p>

<p id="demo"></p>

<script>

function dragStart(event) {

event.dataTransfer.setData("Text", event.target.id);

document.getElementById("demo").innerHTML = "Started to drag the p element";

}

function allowDrop(event) {

event.preventDefault();

}

function drop(event) {

event.preventDefault();

var data = event.dataTransfer.getData("Text");

event.target.appendChild(document.getElementById(data));

}

</script>

</body>

</html>

## DOM Nodes

**D**ocument **O**bject **M**odel of the page. 웹페이지가 로드될 때 브라우저는 페이지의 DOM를 만든다. HTML DOM 모델은 오브젝트 (OBJECT)의 트리로 구성된다.



DOM 에서 모든 HTML 엘리먼트는 object로서 정의된다. 프로그래밍 인터페이스는 각 오브젝트의 속성 (properties) 과 메서드 ( method) 이다.

여기서 property 는 a value 이다. 이것은 html 엘리먼트의 내용을 바꾸거나 얻는 등의 일을 한다.(예를 들면)

메서드 (method) 는 html 엘리먼트의 추가하거나 삭제하는 등의 일을 한다. 예를 들면.

Finding HTML Elements

|  |  |
| --- | --- |
| **Method** | **Description** |
| document.getElementById(*id*) | Find an element by element id |
| document.getElementsByTagName(*name*) | Find elements by tag name |
| document.getElementsByClassName(*name*) | Find elements by class name |

Changing HTML Elements

|  |  |
| --- | --- |
| **Method** | **Description** |
| *element*.innerHTML =  *new html content* | Change the inner HTML of an element |
| *element*.*attribute = new value* | Change the attribute value of an HTML element |
| *element*.setAttribute*(attribute, value)* | Change the attribute value of an HTML element |
| *element*.style.*property = new style* | Change the style of an HTML element |

Adding and Deleting Elements

|  |  |
| --- | --- |
| **Method** | **Description** |
| document.createElement(*element*) | Create an HTML element |
| document.removeChild(*element*) | Remove an HTML element |
| document.appendChild(*element*) | Add an HTML element |
| document.replaceChild(*element*) | Replace an HTML element |
| document.write(*text*) | Write into the HTML output stream |

Adding Events Handlers

|  |  |
| --- | --- |
| **Method** | **Description** |
| document.getElementById(*id*).onclick = function(){*code*} | Adding event handler code to an onclick event |

## Finding HTML Elements by Tag Name

var x = document.getElementsByTagName("p"); 이것은 <p> 엘리먼트를 찾아낸다

아래것은 main 안에 있는 p 태그를 찾느다.

var **x** = document.getElementById("main");  
var y = **x**.getElementsByTagName("p");

## Finding HTML Elements by Class Name

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

이것은 class=”intro”의 모든 엘리먼트의 리스트를 리턴한다.

## Finding HTML Elements by CSS Selectors

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

이것은 class=”intro”의 모든 <p>엘리먼트의 리스트를 리턴한다.

예제01)

<!DOCTYPE html>

<html>

<body>

<p>Hello World!</p>

<p class="intro">The DOM is very useful.</p>

<p class="intro">This example demonstrates the <b>querySelectorAll</b> method.</p>

<p id="demo"></p>

<script>

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

document.getElementById("demo").innerHTML =

'The first paragraph (index 0) with class="intro": ' + x[0].innerHTML;

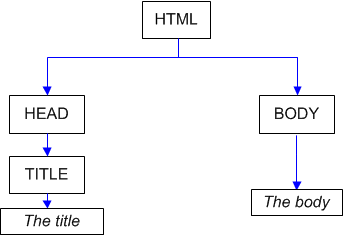
</script>

</body>

</html>

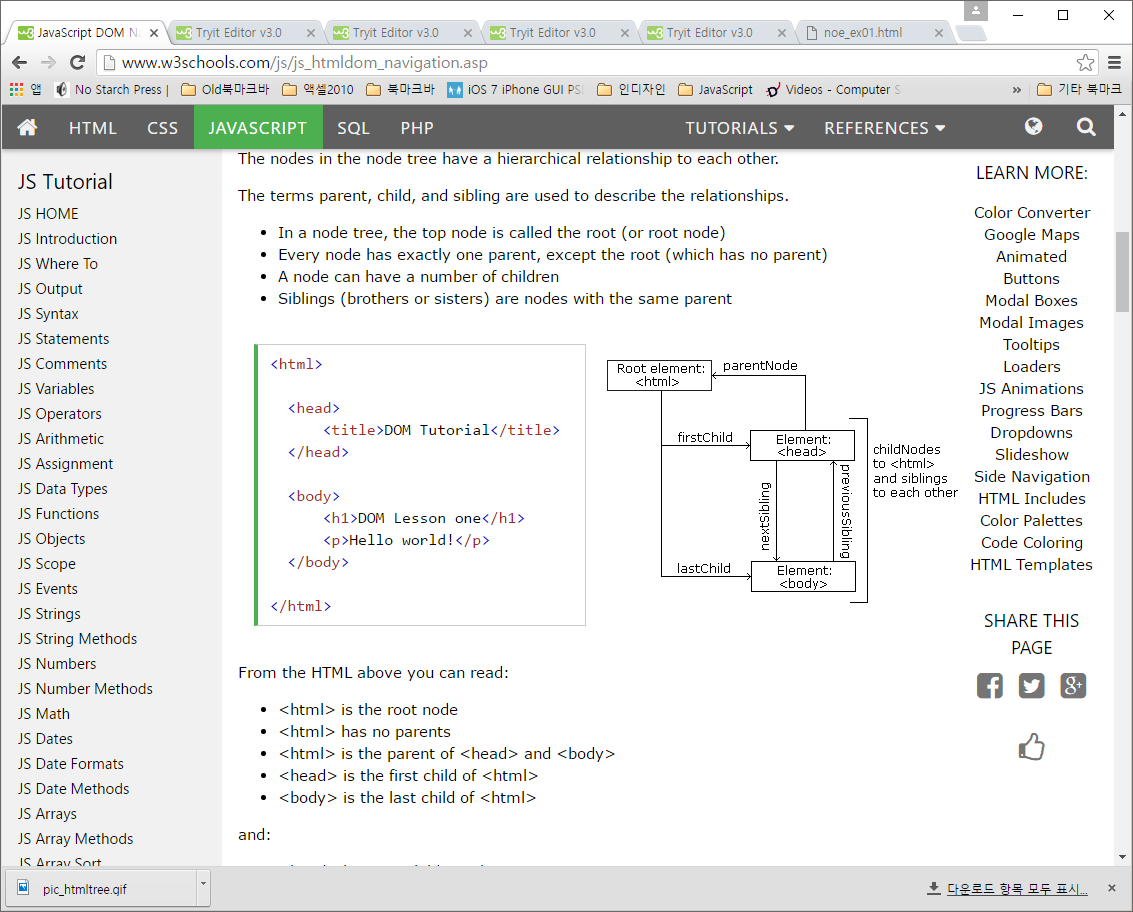
DOM 은 document를 나무(tree) 로 비유할 수 있다. Tree는 부모-자식의 관계로 만들어지는데 부모는 하나 또는 여러 개의 많은 자식 nodes 를 가질 수 있다.

예를 들어



맨 위의 상위에는 html 이라는 node 가 있고 두개의 자식 node ( head 와 body)를 갖고 있다. Head는 하나의 title 이라는 자식 node를 하나 갖고 있다.

html태그들은 DOM tree에서 엘리먼트 ( element) node 들이다. Text 의 조각들은 text node들이 된다. 둘 다 node 이며 단지 type (형태)가 다른다.



## HTML DOM Node List

The getElementsByTagName() method returns a **node list**. A node list is an array-like collection of nodes.

아래의 예제는 document에 있는 모든 <p> node를 수집한다.

예제01)

<!DOCTYPE html>

<html>

<body>

<p>Hello World!</p>

<p>The DOM is very useful!</p>

<p id="demo"></p>

<script>

var myNodelist = document.getElementsByTagName("p");

document.getElementById("demo").innerHTML =

"The innerHTML of the second paragraph is: " +

myNodelist[1].innerHTML;

</script>

</body>

</html>

## HTML DOM Node List Length

The getElementsByTagName() method는  **node list를 리턴하는데** node list는 마치 배열과 같은 node들의 집합체이다.

The length property defines the number of nodes in a node list:

예제01)

<!DOCTYPE html>

<html>

<body>

<p>Hello World!</p>

<p>How many paragraphs in this document?</p>

<p>This example demonstrates the length property of a nodelist.</p>

<p id="demo"></p>

<script>

var myNodelist = document.getElementsByTagName("p");

document.getElementById("demo").innerHTML = myNodelist.length;

</script>

</body>

</html>

Ex02)

<!DOCTYPE html>

<html>

<body>

<p>This is a p element</p>

<p>This is also a p element.</p>

<p>This is also a p element - Click the button to change the background color of all p elements in this document.</p>

<button onclick="myFunction()">Try it</button>

<script>

function myFunction() {

var myNodelist = document.getElementsByTagName("p");

var i;

for (i = 0; i < myNodelist.length; i++) {

myNodelist[i].style.backgroundColor = "red";

}

}

</script>

</body>

</html>

## Child Nodes and Node Values

아래의 예제는 <h1> element 의 value 와 이것을 <p> element에 복사해 넣는다.

<!DOCTYPE html>

<html>

<body>

<h1 id="intro">My First Page</h1>

<p id="demo">Hello World!</p>

<script>

var myText = document.getElementById("intro").childNodes[1].nodeValue;

document.getElementById("demo").innerHTML = myText;

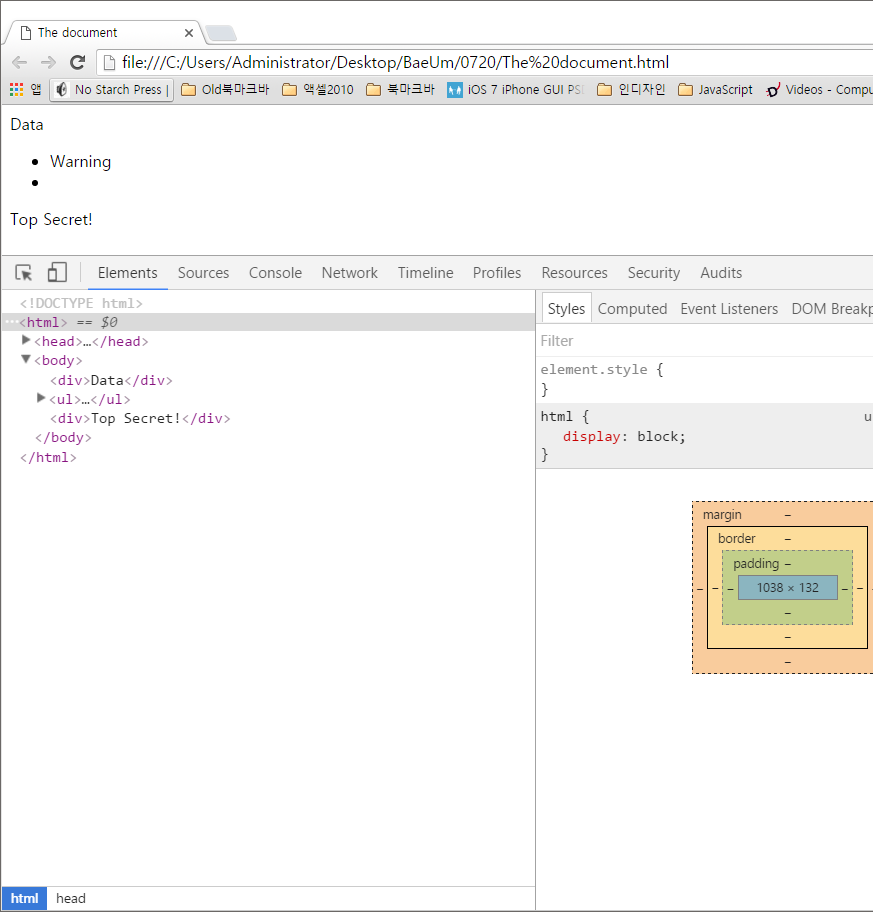
</script>

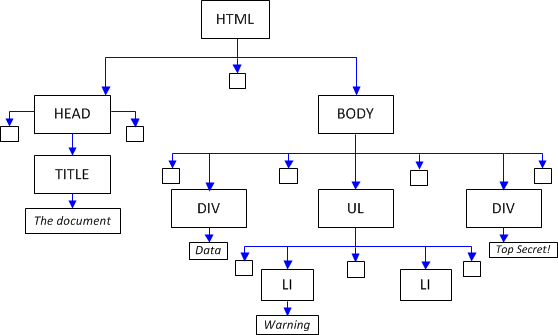
</body>

</html>

위의 예제에서 getElementById is a method, while childNodes and nodeValue are properties.

|  |
| --- |
| <!DOCTYPE HTML>  <html>  <head>  <title>The document</title>  </head>  <body>  <div>Data</div>  <ul>  <li>Warning</li>  <li></li>  </ul>  <div>Top Secret!</div>  </body>  </html>  이것을 DOM tree 로 표현하면, |





여기서 html 안에 있는 whitespace 심볼은 text (텍스트)로 인식되면 text node로 간주된다.

위의 그림은 whitespace를 포함한 text node까지 보여준다.

(여기서 whitespacesms developer tools 에서는 보이지 않는다. )

Whitespace는 node 사이의 공간를 나타낸다. 그래서 태그들 사이에 공간을 없앤다면 whitespace도 사라진다.

예를 들어 <!DOCTYPE HTML><html><head><title>Title</title></head><body></body></html> 이것은 whitespace 가 하나도 없다