Exemples

Cette page présente quelques exemples plus détaillés de développement Web et XML utilisant le DOM. Partout où c'est possible, les exemples utilisent des API courantes, des astuces et des modèles en JavaScript pour manipuler l'objet de document.

Exemple 1 : *height* (hauteur) et width *(largeur)*

L'exemple qui suit montre l'utilisation des propriétés `height` et `width` pour dimensionner des images de diverses tailles :

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>width/height example</title>
<script>
function init() {
  var arrImages = new Array(3);

  arrImages[0] = document.getElementById("image1");
  arrImages[1] = document.getElementById("image2");
  arrImages[2] = document.getElementById("image3");

  var objOutput = document.getElementById("output");
  var strHtml = "<ul>";
```

```
14
15    for (var i = 0; i < arrImages.length; i++) {
16      strHtml += "<li>image" + (i+1) +
17              ": height=" + arrImages[i].height +
18              ", width=" + arrImages[i].width +
19              ", style.height=" + arrImages[i].style.height +
20              ", style.width=" + arrImages[i].style.width +
21              "<\/li>";
22    }
23
24    strHtml += "<\/ul>";
25
26    objOutput.innerHTML = strHtml;
27  }
28  </script>
29  </head>
30  <body onload="init();">
31
32  <p>Image 1: no height, width, or style
33    <img id="image1" src="http://www.mozilla.org/images/mozilla-banner.
34  </p>
35
36  <p>Image 2: height="50", width="500", but no style
37    <img id="image2"
38         src="http://www.mozilla.org/images/mozilla-banner.gif"
39         height="50" width="500">
40  </p>
41
42  <p>Image 3: no height, width, but style="height: 50px; width: 500px;"
43    <img id="image3"
44         src="http://www.mozilla.org/images/mozilla-banner.gif"
45         style="height: 50px; width: 500px;">
46  </p>
47
48  <div id="output"> </div>
49  </body>
50  </html>
51
```

Exemple 2 : attributs d'image

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Modifying an image border</title>

<script>
function setBorderWidth(width) {
  document.getElementById("img1").style.borderWidth = width + "px";
}
</script>
</head>

<body>
<p>
  <img id="img1"
       src="image1.gif"
       style="border: 5px solid green;"
       width="100" height="100" alt="border test">
</p>

<form name="FormName">
  <input type="button" value="Make border 20px-wide" onclick="setBorc
  <input type="button" value="Make border 5px-wide"  onclick="setBorc
</form>

</body>
</html>
```

Exemple 3 : manipulation de styles

Dans cet exemple simple, on accède à certaines propriétés basiques de style d'un élément de paragraphe HTML à l'aide de son objet style. L'objet style de l'élément et ses propriétés de style CSS peuvent être récupérés et définis depuis le DOM. Dans ce cas-ci, les styles individuels sont manipulés directement. Dans l'exemple suivant (l'exemple 4), on utilisera les feuilles de style et leurs règles pour changer les styles de documents entiers.

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
<title>Changing color and font-size example</title>

<script>
function changeText() {
  var p = document.getElementById("pid");

  p.style.color = "blue"
  p.style.fontSize = "18pt"
}
</script>
</head>
<body>

<p id="pid" onclick="window.location.href = 'http://www.cnn.com/';">l

<form>
  <p><input value="rec" type="button" onclick="changeText();" /></p>
</form>

</body>
</html>
```

Exemple 4 : utilisation de feuilles de style

La propriété `styleSheets` de l'objet `document` renvoie une liste des feuilles de style qui ont été chargées pour ce document. On peut accéder à ces feuilles de style et leurs règles individuelles à l'aide des objets `stylesheet`, `style` et `CSSRule`, comme montré dans cet exemple qui affiche tous les sélecteurs de règles de style dans la console.

```
var ss = document.styleSheets;

for(var i = 0; i < ss.length; i++) {
  for(var j = 0; j < ss[i].cssRules.length; j++) {
    dump( ss[i].cssRules[j].selectorText + "\n" );
  }
}
```

Pour un document avec une seule feuille de style dans laquelle les trois règles suivantes sont définies :

```
BODY { background-color: darkblue; }
P { font-face: Arial; font-size: 10pt; margin-left: .125in; }
#lumpy { display: none; }
```

Ce script affiche les lignes suivantes :

```
BODY
P
#LUMPY
```

Exemple 5 : propagation d'évènements

Cet exemple montre comment les évènements se déclenchent et sont gérés dans le DOM d'une manière très simple. Lorsque l'élément BODY de ce document HTML est chargé, un écouteur d'évènement est enregistré sur la ligne supérieure de l'élément TABLE. Celui-ci gère l'évènement en exécutant la fonction `stopEvent`, qui modifie la valeur de la cellule inférieure du tableau.

Cependant, `stopEvent` appelle également une méthode d'objet `event`, `event.stopPropagation` , qui empêche l'évènement de se propager (bubbling) plus haut dans le DOM. Notez que le tableau lui-même dispose d'un gestionnaire d'évènement `onclick` qui devrait afficher un message lorsqu'on clique sur le tableau. Mais comme la méthode `stopEvent` a interrompu la propagation, après que les données du tableau aient été mises à jour, la phase évènementielle est effectivement arrêtée et un message d'alerte est affiché pour le confirmer.

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Event Propagation</title>

<style>
#t-daddy { border: 1px solid red }
```

```
#c1 { background-color: pink; }
</style>

<script>
function stopEvent(ev) {
  c2 = document.getElementById("c2");
  c2.innerHTML = "hello";

  // cela devrait empêcher t-daddy d'obtenir le click.
  ev.stopPropagation();
  alert("event propagation halted.");
}

function load() {
  elem = document.getElementById("tbl1");
  elem.addEventListener("click", stopEvent, false);
}
</script>
</head>

<body onload="load();">

<table id="t-daddy" onclick="alert('hi');">
  <tr id="tbl1">
    <td id="c1">one</td>
  </tr>
  <tr>
    <td id="c2">two</td>
  </tr>
</table>

</body>
</html>
```

Exemple 6 : getComputedStyle

Cet exemple montre comment la méthode `window.getComputedStyle` peut être utilisée pour obtenir les styles d'un élément qui ne sont pas définis dans l'attribut `style` ou à l'aide de JavaScript (c'est-à-dire, `elem.style.backgroundColor="rgb(173, 216, 230)"`). Ces

derniers types de styles peuvent être récupérés avec la propriété plus directe `elt.style` , dont les propriétés sont listées dans la liste des propriétés DOM CSS.

`getComputedStyle()` renvoie un objet `ComputedCSSStyleDeclaration`, dont les propriétés de style individuelles peuvent être référencées à l'aide de sa méthode `getPropertyValue()` comme montré dans l'exemple suivant.

```
<!DOCTYPE html>
<html lang="en">
<head>

<title>getComputedStyle example</title>

<script>
function cStyles() {
  var RefDiv = document.getElementById("d1");
  var txtHeight = document.getElementById("t1");
  var h_style = document.defaultView.getComputedStyle(RefDiv, null).g

  txtHeight.value = h_style;

  var txtWidth = document.getElementById("t2");
  var w_style = document.defaultView.getComputedStyle(RefDiv, null).g

  txtWidth.value = w_style;

  var txtBackgroundColor = document.getElementById("t3");
  var b_style = document.defaultView.getComputedStyle(RefDiv, null).g

  txtBackgroundColor.value = b_style;
}
</script>

<style>
#d1 {
  margin-left: 10px;
  background-color: rgb(173, 216, 230);
  height: 20px;
  max-width: 20px;
}
</style>

```

```
</head>

<body>

<div id="d1"> </div>

<form action="">
  <p>
    <button type="button" onclick="cStyles();">getComputedStyle</butt
    height<input id="t1" type="text" value="1" />
    max-width<input id="t2" type="text" value="2" />
    bg-color<input id="t3" type="text" value="3" />
  </p>
</form>

</body>
</html>
```

Exemple 7 : affichage des propriétés d'un objet `event`

Cet exemple utilise des méthodes DOM pour afficher les propriétés d'un objet `window.onload` `event` et leurs valeurs dans un tableau. Il montre également une technique utile utilisant une boucle `for..in` pour parcourir les propriétés d'un objet et obtenir leurs valeurs.

Les propriétés des objets `event` diffèrent sensiblement entre les différents navigateurs, la spécification norme DOM liste les propriétés standard, mais beaucoup de navigateurs ont ajouté un bon nombre de propriétés supplémentaires.

Placez le code qui suit dans une fichier texte vide et chargez-le dans différents navigateurs, vous serez surpris des différences entre le nombre et le nom des propriétés. Vous pouvez également ajouter quelques éléments à la page et appeler cette fonction depuis d'autres gestionnaires d'évènements.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8"/>
```

```
<title>Show Event properties</title>

<style>
table { border-collapse: collapse; }
thead { font-weight: bold; }
td { padding: 2px 10px 2px 10px; }

.odd { background-color: #efdfef; }
.even { background-color: #ffffff; }
</style>

<script>

function showEventProperties(e) {
  function addCell(row, text) {
    var cell = row.insertCell(-1);
    cell.appendChild(document.createTextNode(text));
  }

  var e = e || window.event;
  document.getElementById('eventType').innerHTML = e.type;

  var table = document.createElement('table');
  var thead = table.createTHead();
  var row = thead.insertRow(-1);
  var lableList = ['#', 'Property', 'Value'];
  var len = lableList.length;

  for (var i=0; i<len; i++) {
    addCell(row, lableList[i]);
  }

  var tbody = document.createElement('tbody');
  table.appendChild(tbody);

  for (var p in e) {
    row = tbody.insertRow(-1);
    row.className = (row.rowIndex % 2)? 'odd':'even';
    addCell(row, row.rowIndex);
    addCell(row, p);
    addCell(row, e[p]);
  }

```

```
  document.body.appendChild(table);
}

window.onload = function(event){
  showEventProperties(event);
}
</script>
</head>

<body>
<h1>Properties of the DOM <span id="eventType"></span> Event Object</
</body>

</html>
```

Exemple 8 : utilisation de l'interface DOM Table

L'interface DOM HTMLTableElement fournit certaines méthodes utilitaires permettant de créer et de manipuler des tableaux. Deux méthodes utilisées fréquemment sont `HTMLTableElement.insertRow` et `tableRow.insertCell`

Pour ajouter une ligne et quelques cellules à un tableau existant :

```
<table id="table0">
 <tr>
  <td>Row 0 Cell 0</td>
  <td>Row 0 Cell 1</td>
 </tr>
</table>

<script>
var table = document.getElementById('table0');
var row = table.insertRow(-1);
var cell,
    text;

for (var i = 0; i < 2; i++) {
  cell = row.insertCell(-1);
```

```
    text = 'Row ' + row.rowIndex + ' Cell ' + i;
    cell.appendChild(document.createTextNode(text));
  }
</script>
```

Notes

- N'utilisez jamais la propriété `innerHTML` d'un tableau pour le modifier, même si vous pouvez l'utiliser pour créer un tableau entier ou le contenu d'une cellule.
- Si vous utilisez les méthodes DOM Core `document.createElement` et `Node.appendChild` pour créer des lignes et cellules de tableau, il est nécessaire de les ajouter à un élément `tbody` dans Internet Explorer, tandis que les autres navigateurs vous permettront de les ajouter à un élément `table` (les lignes seront ajoutées au dernier élément `tbody`).
- Un certain nombre d'autres méthodes utilitaires faisant partie de l'interface table peuvent être utilisées pour créer et modifier des tableaux.