JavaScript
DOM

420-JJA-JQ
Programmation mobile

Gestion des évènements

- Transformer la page statique en page dynamique
  - **Côté serveur** : il est possible d'utiliser un langage comme PHP associé à une base de données SQL ;
  - **Côté client** : ce sont les actions de l'utilisateur, puis la modification du document en fonction de ces actions et à l'aide de JavaScript, qui produisent une page web dynamique.

Web dynamique…

- Utiliser JavaScript pour modifier le document côté utilisateur
- Le **Modèle Objet de Document**, ou **DOM** est un outil permettant l'accès aux documents HTML et XML.
  - Il fournit une représentation structurée du document ;
  - Il codifie la manière dont un script peut accéder à cette structure.
- Il s'agit donc essentiellement d'un moyen de lier une page Web, par exemple, à un langage de programmation ou de script.

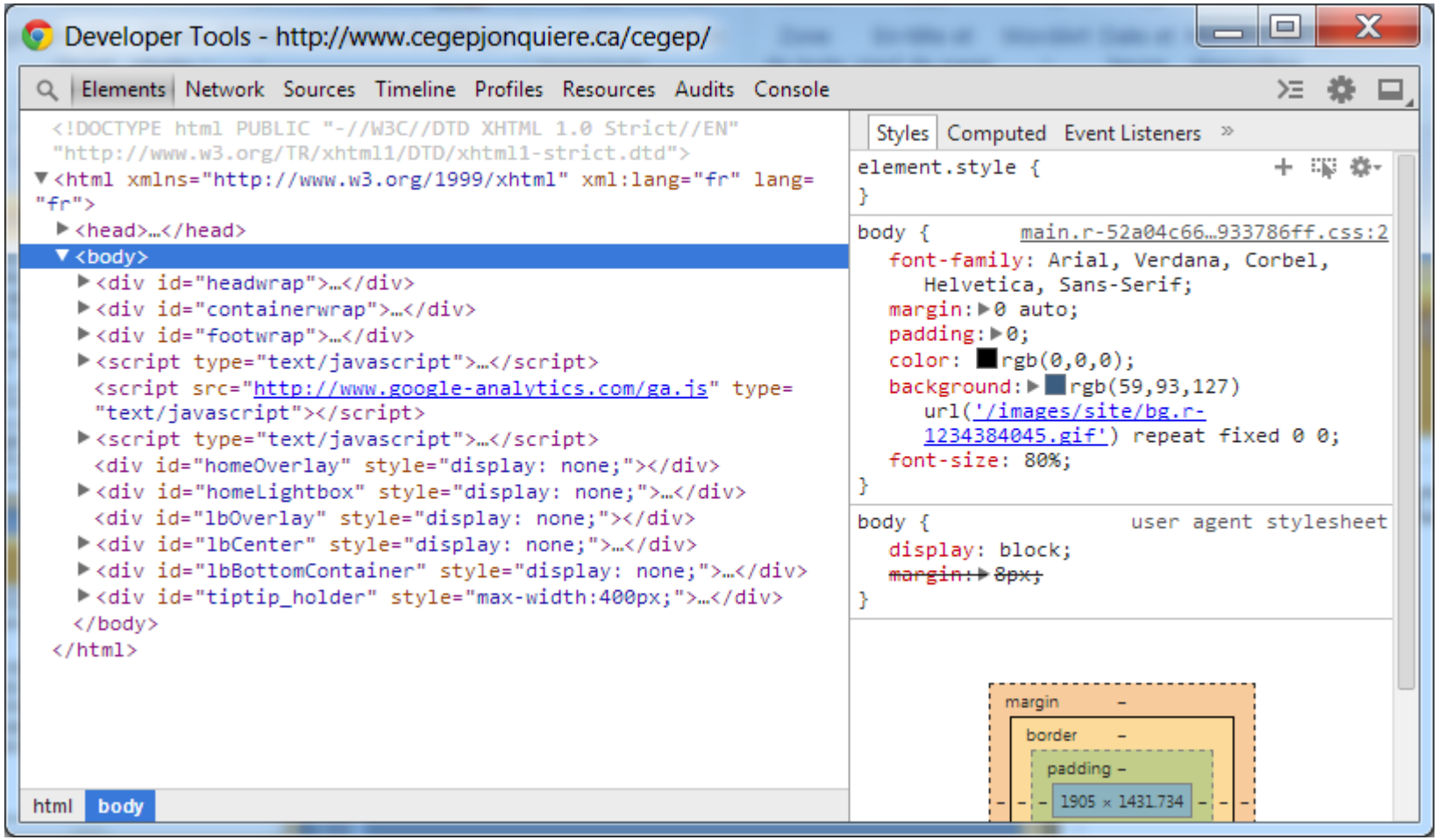
DOM

- DOM : structure d'arbre représentant le document et présente dans la mémoire du navigateur.
- DOM : API (*Application Programming Interface*) qui nous fournit les moyens d'interagir avec l'arbre-document. C'est une recommandation du W3C pour gérer le contenu de documents XML, HTML en particulier.

Le DOM pourquoi faire ?

- Rendre visible/invisible une partie du document,
- Modifier un élément de style de la page,
- Changer une image,
- Remplissage automatique de formulaires ou test de validité des données saisies dans un formulaire avant envoi au serveur, etc.
- **Notons que, dans tous les cas, les identifiants HTML (attributs ID) vont jouer un rôle crucial pour repérer les nœuds de l'arbre.**

Une arborescence ou un ensemble de nœuds



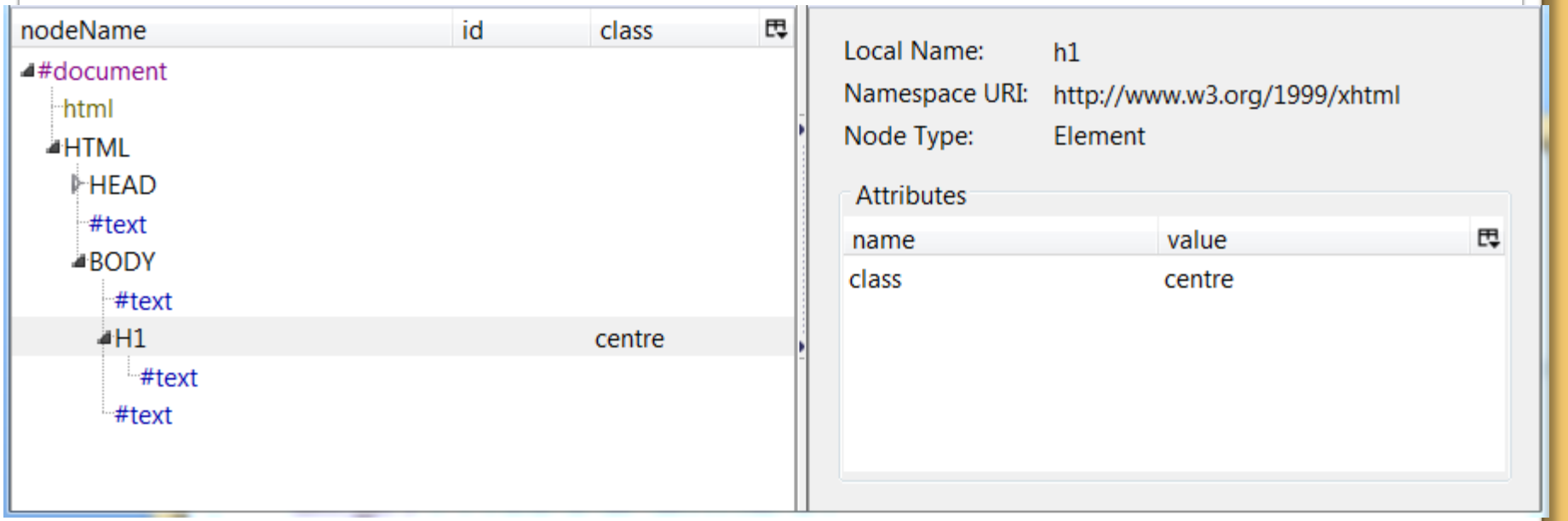
Nœud

- Chaque élément de la structure HTML représente un nœud
- Dans un document HTML, trois types :
  - les nœuds-élément
  - les nœuds-attribut
  - les nœuds-texte.

Exemple Nœud

```
<h1 class="centre">Bonjour les jeunes !</h1>
```

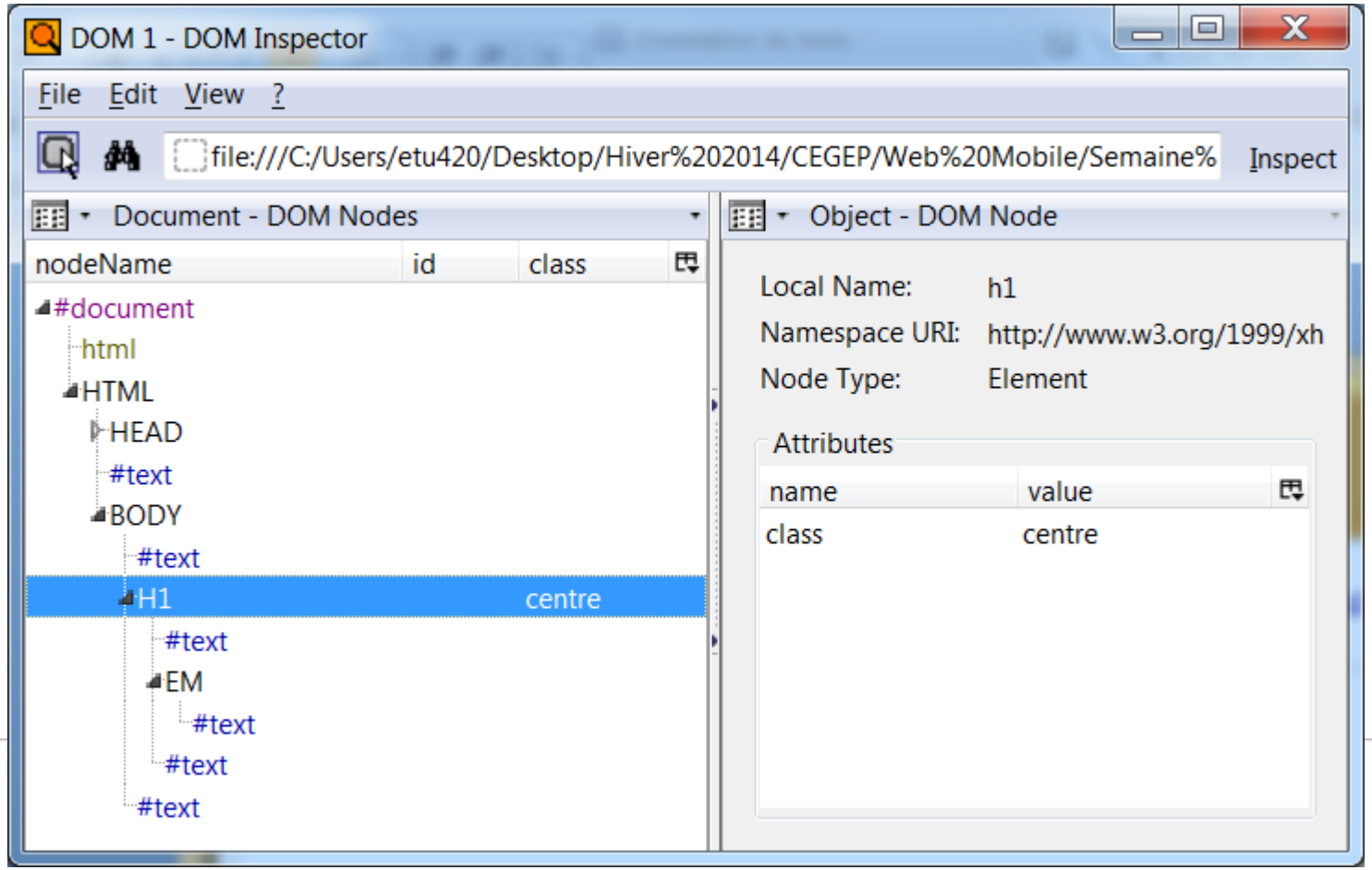
- h1 : nœud élément
- class: nœud attribut
- #text: nœuds-texte



Exemple Nœud 2

```
<h1 class="centre">Bonjour <em>les jeunes</em> !</h1>
```

- h1 a 3 nœuds maintenant !



Et JavaScript dans tout cela ?

- Il permet de manipuler les éléments du document HTML.

Accès aux éléments

- L'objet **document** est un objet qui représente l'ensemble de l'arborescence du document
  - **getElementById** permet de sélectionner un élément d'identifiant donné dans une page.
    - Par exemple, si on a dans la page (...) , document.getElementById("intro") permettra de sélectionner précisément l'élément p en question.
  - **getElementsByName** permet de sélectionner les éléments portant un nom donné dans une page (pas supportée par Internet Explorer)
  - **getElementsByTagName** permet de sélectionner les éléments portant un nom de balise donné dans une page.

getElementById

- Accéder au contenu grâce à **`innerHTML`**

```
<!DOCTYPE html>
<html>
<head>
<meta content="text/html; charset=utf-8" >
<title>getElementById</title>
</head>
<body>
<p id=para1 > Mon premier paragraphe!<p>
</body>
</html>

<script>
var p = document.getElementById("para1");
alert(p.innerHTML); // Mon premier paragraphe
</script>
```

Accès à un élément via sélecteur

- **`document.querySelectorAll(...)`**
  - Tableau d'éléments de DOM avec le sélecteur
  - Possibilité de restreindre une partie de l'arborescence

```
<!DOCTYPE html>
<html>
<head>
<meta content="text/html; charset=utf-8" >
<title>Selecteur</title>
</head>
<body>
<p id=para1 >
<em> Mon premier paragraphe ! </em>
<em> et des élémnets em </em>
<p>

<p id=para2 >
<em> Mon deuxième paragraphe ! </em>
<em> et des élémnets em </em>
<p>
</body>
</html>
```

document.querySlectorAll(…)

```
<script>
var p = document.getElementById("para2");
var ems = p.querySelectorAll("em:first-child");
alert(ems[0].innerHTML); // Mon deuxième
paragraphe!
</script>
```

- Pour retourner tous les éléments

```
var ems = document.querySelectorAll("em:first-child");
```

- `:nth-child, :last-child, …`

Création d'un élément

- Deux étapes
  - Créer l'élément

    `document.createElement(nom_Balise)`
  - Ajouter l'élément comme fils d'une autre balise

    `parent.appendChild(Fils)`

    `parent.insertBefore(Fils,frère)`

Exemple insertion

```
<!DOCTYPE html>
<html>
<head>
<meta content="text/html; charset=utf-8" >
<title>Insertion Élément</title>
</head>
<body>
<p id=para1 >
<em> Mon premier paragraphe ! </em>
<em> et des élémnets em </em>
<p>
</body>
</html>


<script>
var p = document.createElement("p");
p.innerHTML = "<em> Mon deuxième paragraphe ! </em> "+
                    "<em> et des élémnets em </em>";
p.id = "para2";
document.body.appendChild(p);
</script>
```

Évènement de Javascript

- **load**, **unload** : événements déclenchés à l'arrivée et au départ de la page,
- **click**, **mousedown**, **mouseup**, **mousemove**, **mouseover**, **mouseout** : événements associés aux clics et déplacements de la souris,
- **keypress**, **keydown**, **keyup** : événements provoqués par l'appui d'une touche au clavier,
- **submit**, **change** : événements associés à la manipulation d'un formulaire par l'utilisateur.
- **Abort**, **Error**, **Move**, **Resize**, **KeyPress**, **KeyUp**, **DblClick**, **MouseDown**, **MouseUp**, **MouseMove**, **Reset**, ...

Évènement

- Pour gérer un évènement en JavaScript, il faut installer un **gestionnaire d'événement :**
  - Un gestionnaire d'événement sera l'action déclenchée automatiquement lorsque l'évènement associé se produit.
  - La syntaxe courante est la suivante : **`onEvenement=fonction()`** où **`Evenement`** est le nom de l'évènement géré.

Exemple Évènement

```
<!DOCTYPE html>
<html>
<head>
<meta content="text/html; charset=utf-8" >
<title>Bouton</title>
</head>
<body>

<button onclick="alert('Vous avez bien cliqué ici !')">Cliquez ici</button>

</body>
</html>
```

Suppression élément

```
<!DOCTYPE html>
<html>
<head>
<meta content="text/html; charset=utf-8" >
<title>Suppression Élément</title>
</head>
<body>
<p> Cliquer ici pour supprimer 1</p>
<p> Cliquer ici pour supprimer 2</p>
<p> Cliquer ici pour supprimer 3</p>
<p> Cliquer ici pour supprimer 4</p>
<p> Cliquer ici pour supprimer 5</p>
<p> Cliquer ici pour supprimer 6</p>
<p> Cliquer ici pour supprimer 7</p>
<p> Cliquer ici pour supprimer 8</p>
<p> Cliquer ici pour supprimer 9</p>
<p> Cliquer ici pour supprimer 10</p>

</body>
```

Exemple Suppression (suite)

```
<script>
var ps = document.querySelectorAll('p');

for(var i in ps)
{
        ps[i].onclick = function()
        {
                this.parentNode.removeChild(this);
        }
}
</script>
```

Gérer des listeners

- Intercepter des évènements
- Intercepter un **`onclick`** sur un paragraphe

```
<p onclick = "this.parentNode.removeChild(this);"> Cliquer ici pour supprimer 10</p>
```

- Mais si je veux ajouter un autre traitement ?
- Utiliser la fonction : **`addEventListener()`**

Exemple

```
<!DOCTYPE html>
<html>
<head>
<meta content="text/html; charset=utf-8" >
<title>Evenement</title>
</head>
<body>
<p id=para1 onclick = "this.parentNode.removeChild(this);"> Cliquer ici pour supprimer le paragraphe</p >

</body>
</html>
```

Exemple (suite)

```
<script>
document.getElementById(id).addEventListener("click", function(event)
{
        var p = document.createElement("p");
        p.innerHTML = "<em> Mon deuxième paragraphe ! </em> "+
                        "<em> et des éléments em </em>";
        p.id = "para2";
        document.body.appendChild(p);
});
</script>
```

Exemple Move

```
<script>
var position = document.getElementById('plan');

    document.addEventListener('mousemove', function(e)
            {
        position.innerHTML = 'Abscisse X : ' + e.clientX + 'px<br
/>Ordonné Y : ' + e.clientY + 'px';
    }, false);
</script>
```

Modifier le contenu

```
<script>
var p = document.getElementById("para2");
var ems = p.querySelectorAll("em:first-child");
var str = "Mon deuxième paragraphe que je viens de
remplacer" + "<br>";

ems[0].innerHTML += str; // Mon deuxième paragraphe!
</script>
```

Autres possibilités

- **`cloneNode(bool)`** : avec ou sans les fils
- **`replaceChild(nœud1, nœud2)`**
- **`hasChildNodes()`**
- **`insertBefore()`**