



Cours 8 : Le DOM HTML

O.R. Merad Boudia

Université d'Oran 1 Ahmed Ben Bella

L2 : 2017/2018

rafik.merad@gmail.com

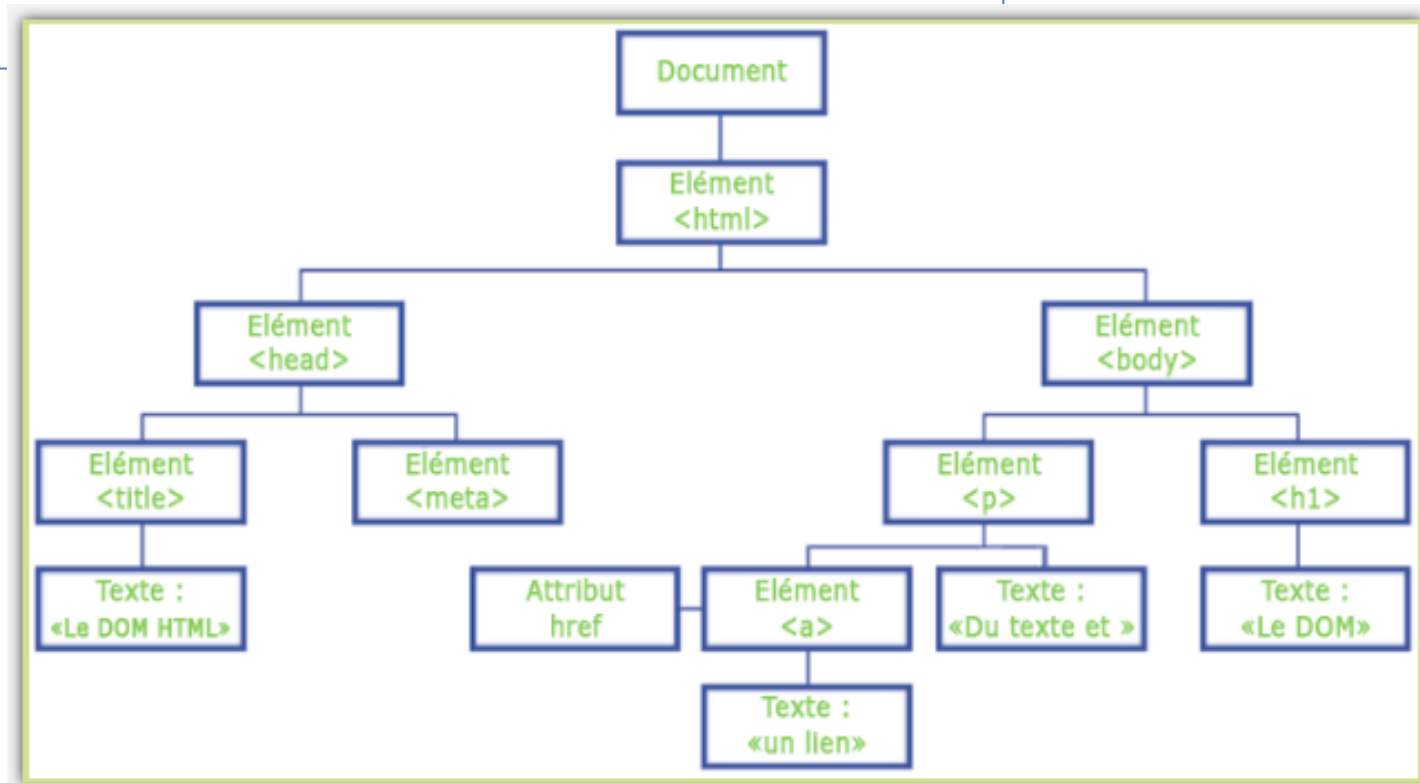
Développement Web

La structure du DOM HTML

- ✓ Dans cette structure, le document HTML, les éléments HTML, les attributs HTML, le texte à l'intérieur des éléments sont considérés comme des nœuds, ou « nodes » en anglais.
- ✓ Le terme « node » est un terme générique qui désigne tous les objets contenus dans le DOM. En un mot : tout objet appartenant au DOM est un node.
- ✓ Il y a plusieurs types de nœuds correspondant aux différents objets HTML : le type de nœud **ELEMENT_NODE** (pour les éléments HTML), le type de nœud **TEXT_NODE** (pour le texte), etc.
- ✓ Le DOM HTML peut être considéré comme une hiérarchie de nœuds. De cette manière, on prend souvent l'image d'un arbre pour représenter une page HTML du point de vue du DOM.

Exemple d'une structure DOM HTML

```
<!doctype html>
<html>
  <head>
    <title> Le DOM HTML </title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>Le DOM</h1>
    <p>Du texte et <a href="http://www.univ-oran1.dz">un lien</a></p>
  </body>
</html>
```



Remarques

- ✓ Le DOM est une interface de programmation qui permet de manipuler le code HTML d'une page de façon dynamique en utilisant le langage JavaScript.
- ✓ Le DOM est créé automatiquement lors du chargement d'une page web par le navigateur.
- ✓ Aujourd'hui, le DOM est un standard et a été unifié : tous les navigateurs récents vont créer le même DOM à partir d'une même page et nous pouvons utiliser les mêmes propriétés et méthodes.
- ✓ Dans ce cours, nous allons beaucoup travailler avec les objets **Document** et **Element** car c'est grâce à eux que nous allons pouvoir accéder au contenu de nos pages web et les modifier de façon dynamique.

L'objet document et ses méthodes

- ❑ Les différentes méthodes de l'objet Document :
 - ✓ La méthode **getElementById()** ;
 - ✓ La méthode **getElementsByTagName()** ;
 - ✓ La méthode **getElementsByClassName()** ;
 - ✓ La méthode **querySelector()** ;
 - ✓ La méthode **querySelectorAll()**.

La méthode getElementById()

Cette méthode permet de cibler un élément HTML possédant un attribut id en particulier. C'est la méthode la plus utilisée.

```
<body>
  <h1 id="gros_titre">Le DOM</h1>
  <p>Du texte et <a href="http://www.univ-oran1.dz/">un lien</a></p>

  <script>
    /*On utilise la méthode getElementById de l'objet document.
    *On enferme le résultat renvoyé dans une variable "titre"*/
    var titre = document.getElementById('gros_titre');
    alert(titre);
  </script>
</body>
```

Source : cette page

[object HTMLHeadingElement]

OK

La méthode `getElementsByTagName()`

Cette méthode retourne des informations relatives à tous les éléments HTML d'un même « genre » (tous les éléments `p` par exemple) dans un tableau.

```
<body>
  <h1 id="gros_titre">Le DOM</h1>
  <p>Du texte et <a href="http://www.univ-oran1.dz/">un lien</a></p>
  <p>Un deuxième paragraphe</p>

  <script>
    var tableau = document.getElementsByTagName('p');
    alert('Notre page contient ' + tableau.length + ' paragraphes');
  </script>
</body>
```

Source : cette page

Notre page contient 2 paragraphes

OK

La méthode `getElementsByClassName`

Cette méthode permet d'accéder aux éléments HTML disposant d'un attribut `class` en particulier. Cette méthode est à nouveau une méthode de l'objet `Document`.

```
<body>
  <h1 id="gros_titre">Le DOM</h1>
  <p>Du texte et <a href="http://www.univ-oran1.dz/">un lien</a></p>
  <p class="para">Un deuxième paragraphe</p>

  <script>
    var tableau = document.getElementsByClassName('para');
    alert('Notre page contient ' + tableau.length + ' paragraphe portant la classe para');
  </script>
</body>
```

Source : cette page

Notre page contient 1 paragraphe portant la classe para

OK

querySelector() et querySelectorAll()

Ces méthodes vont nous permettre d'accéder à des éléments HTML correspondant à un certain sélecteur CSS, que ce soit un id, une class, un type d'élément, un attribut, etc.

```
<p class="para">Du texte et <a href="http://www.univ-oran1.dz/">un lien</a></p>
<p class="para">Un deuxième paragraphe</p>

<script>
    //On accède au premier élément de type lien contenu dans un paragraphe
    var lien = document.querySelector('p a');

    //On accède à tous les paragraphes portant la class "para"
    var p = document.querySelectorAll('.para');

    //On affiche les résultats obtenus
    alert('Informations dans lien : ' + lien +
        '\nDans p[0] : ' + p[0] +
        '\nDans p[1] : ' + p[1])
</script>
```

Source : cette page

Informations dans lien : http://www.univ-oran1.dz/
Dans p[0] : [object HTMLParagraphElement]
Dans p[1] : [object HTMLParagraphElement]

OK

Accéder au contenu des éléments (1)

Pour cela, il faut utiliser la propriété `innerHTML` sur le résultat renvoyé par nos méthodes.

```
<p class="para">Du texte et <a href="http://www.univ-oran1.dz/">un lien</a></p>  
<p class="para">Un deuxième paragraphe</p>
```

```
<script>
```

```
/*On accède au premier paragraphe portant la class "para" et  
on récupère ce qui se trouve à l'intérieur*/
```

```
var p = document.querySelector('.para').innerHTML;  
alert(p);
```

```
</script>
```

Source : cette page

Du texte et un lien

OK

Accéder au contenu des éléments (2)

Si l'on souhaite ne récupérer que le contenu textuel présent à l'intérieur d'un élément, on préférera alors utiliser la propriété **textContent**.

```
<p class="para">Du texte et <a href="http://www.univ-oran1.dz/">un lien</a></p>
<p class="para">Un deuxième paragraphe</p>
```

```
<script>
    /*La différence entre innerHTML et textContent*/
    var p = document.querySelector('.para').innerHTML;
    var t = document.querySelector('.para').textContent;

    alert('Contenu récupéré avec innerHTML : \n' + p +
        '\n\nContenu récupéré avec textContent : \n' + t);
</script>
```

Source : cette page

Contenu récupéré avec innerHTML :

Du texte et un lien

Contenu récupéré avec textContent :

Du texte et un lien

OK

Accès direct à des types d'éléments

```
<p class="para">Du texte et <a href="http://www.univ-oran1.dz/">un lien</a></p>  
<p class="para">Un deuxième paragraphe</p>
```

```
<script>  
  var titre_page = document.title;  
  var page = document.body;  
  var lien = document.links;  
  
  alert('Contenu de titre_page : ' + titre_page +  
        '\nContenu de page : ' + page +  
        '\nContenu de lien : ' + lien);  
</script>
```

Source : cette page

Contenu de titre_page : Le DOM HTML

Contenu de page : [object HTMLBodyElement]

Contenu de lien : [object HTMLCollection]

OK

Modifier le contenu d'un élément HTML

La propriété `innerHTML` permet aussi de modifier le contenu d'un élément HTML. Pour cela, il suffit d'utiliser `innerHTML` sur un élément et de lui affecter une nouvelle valeur textuelle.

```
<body>
  <h1 id="gros_titre">Le DOM</h1>
  <p class="para">Du texte et <a href="http://www.univ-oran1.dz/">un lien</a></p>
  <p class="para">Un deuxième paragraphe</p>

  <script>
    //On modifie le contenu textuel de notre élément h1 avec innerHTML
    document.getElementById('gros_titre').innerHTML = 'Titre modifié !';
  </script>
</body>
```

Modifier la valeur d'un attribut HTML

```
<body>
  <h1 id="gros_titre">Le DOM</h1>
  <p class="para">Du texte et <a href="http://www.univ-oran1.dz/">un lien</a></p>
  <p class="para">Un deuxième paragraphe</p>

  <script>
    //On modifie la valeur de l'attribut href de notre lien
    document.querySelector('a').href = 'http://wikipedia.org';
  </script>
```

Modifier le CSS d'un élément HTML

```
<body>
  <h1 id="gros_titre">Le DOM</h1>
  <p class="para">Du texte et <a href="http://www.univ-oran1.dz/">un lien</a></p>
  <p class="para">Un deuxième paragraphe</p>

  <script>
    // On veut que notre titre s'affiche en orange
    document.getElementById('gros_titre').style.color = 'orange';
    // On donne une taille de 60px à notre titre
    document.getElementById('gros_titre').style.fontSize = '60px';
  </script>
</body>
```

Le DOM

Du texte et [un lien](http://www.univ-oran1.dz/)

Un deuxième paragraphe

Créer un nouvel élément HTML

```
<body>
  <h1 id="gros_titre">Le DOM</h1>
  <p class="para">Du texte et <a href="http://www.univ-oran1.dz/">un lien</a></p>
  <p class="para">Un deuxième paragraphe</p>

  <script>
    //On crée un élément de type p
    var newPara = document.createElement('p');

    //On ajoute un attribut id à notre paragraphe
    newPara.id = 'nouveau';

    //On crée un noeud de type texte
    var texte = document.createTextNode('Inséré !');
  </script>
</body>
```


Insérer texte et élément dans une page

```
<body>
  <h1 id="gros_titre">Le DOM</h1>
  <p class="para">Du texte et <a href="http://
www.univ-oran1.dz/">un lien</a></p>
  <p class="para">Un deuxième paragraphe</p>

  <script>
    //On crée un élément de type p
    var newPara = document.createElement('p');

    //On ajoute un attribut id à notre paragraphe
    newPara.id = 'nouveau';

    //On crée un noeud de type texte
    var texte = document.createTextNode('Inséré !');
    //On insère le texte dans notre paragraphe
    newPara.appendChild(texte);

    /*On insère finalement notre élément en tant que
    *dernier enfant de body (auquel on accède
    *directement
    *avec "document.body", tout simplement)*/
    document.body.appendChild(newPara);
  </script>
</body>
```

La méthode `appendChild()` va insérer un objet en tant que dernier enfant d'un autre objet

Le DOM

Du texte et [un lien](#)

Un deuxième paragraphe

Inséré !

Insérer un élément à un endroit précis

```
<body>
  <h1 id="gros_titre">Le DOM</h1>
  <p class="para">Du texte</p>
  <p class="para">Un deuxième paragraphe</p>

  <script>
    //On crée un élément de type p
    var newPara = document.createElement('p');

    //On ajoute un attribut id à notre paragraphe
    newPara.id = 'nouveau';

    //On crée un noeud de type texte
    var texte = document.createTextNode('Inséré !');

    //On insère le texte dans notre paragraphe
    newPara.appendChild(texte);

    //On accède à notre premier paragraphe
    var para1 = document.querySelector('.para');

    //On insère notre nouveau paragraphe juste avant
    document.body.insertBefore(newPara, para1);
  </script>
</body>
```

Supprimer un élément HTML en JS

```
<body>
  <h1 id="gros_titre">Le DOM</h1>
  <p class="para">Du texte</p>
  <p class="para">Un deuxième paragraphe</p>

  <script>
    /*On veut supprimer notre titre, on commence donc
    *par y accéder*/
    var titre = document.getElementById('gros_titre');

    //On accède ensuite à l'élément parent de h1 (body)
    var parent = document.body;

    //On supprime finalement notre titre avec removeChild
    parent.removeChild(titre);
  </script>
</body>
```

Du texte

Un deuxième paragraphe

Modifier des éléments HTML en JS

```
<body>
  <h1 id="gros_titre">Le DOM</h1>
  <p class="para">Du texte</p>
  <p class="para">Un deuxième paragraphe</p>
  <script>
    //On accède à notre élément h1 en JavaScript
    var titre = document.getElementById('gros_titre');

    //On accède ensuite à l'élément parent de h1 (body)
    var parent = document.body;

    //On crée une valeur de remplacement
    var nouveauTitre = document.createElement('h2');

    //On ajoute du texte et un id à notre h2
    nouveauTitre.id = 'titre_moyen';
    nouveauTitre.innerHTML = 'Titre modifié en Js !'

    //On remplace finalement h1 par h2
    parent.replaceChild(nouveauTitre, titre);
  </script>
</body>
```

Titre modifié en Js !

Du texte

Un deuxième paragraphe

La propriété parentNode

La propriété parentNode permet d'accéder au nœud parent d'un certain nœud, et donc de nous déplacer dans le DOM de notre page HTML. Cette propriété est une propriété de l'objet Element.

```
<body>
  <h1 id="gros_titre">Le DOM</h1>

  <div>
    <p class="para">Du texte</p>
    <p class="para">Un deuxième paragraphe</p>
  </div>

  <script>
    //On accède à notre premier élément p
    var p = document.querySelector('.para');

    //On accède ensuite à notre div avec parentNode
    var div = p.parentNode;

    //On peut ensuite manipuler notre div à loisir
    div.style.color = 'orange';
  </script>
</body>
```

Le DOM

Du texte

Un deuxième paragraphe

Les propriétés childNodes et nodeValue

ChildNodes, à l'inverse de parentNode, va nous permettre d'accéder aux nœuds enfants d'un certain nœud HTML.

```
<body>
  <h1 id="gros_titre">Le DOM</h1>

  <div>
    <p class="para">Du texte</p>
    <p class="para">Un deuxième paragraphe</p>
  </div>

  <script>
    //On accède à notre body
    var b = document.body;

    /*On accède ensuite à notre div avec childNodes.
    *On se rappelle que le premier élément d'un tableau
    *possède la clef 0, le second la clef 1, etc.*/
    var div = b.childNodes[3];

    //On accède ensuite au premier paragraphe dans notre div
    var p1 = div.childNodes[1];

    //Et enfin au texte dans notre paragraphe, qu'on affiche
    var texte = p1.childNodes[0].nodeValue;
    alert(texte);
  </script>
</body>
```

Source : cette page

Du texte

OK

Les propriétés firstChild et lastChild

```
<body>
  <h1 id="gros_titre">Le DOM</h1>
  <div>
    <p class="para">Du texte</p>
    <p class="para">Un deuxième <strong>paragraphe</strong></p>
  </div>
  <script>
    //On accède à notre deuxième paragraphe
    var p2 = document.querySelectorAll('.para')[1];

    //On accède aux premier et dernier enfants de p2
    var premier = p2.firstChild;
    var dernier = p2.lastChild;

    //On récupère et affiche ce qu'ils contiennent
    var inner1 = premier.nodeValue;
    var inner2 = dernier.innerHTML;

    alert('Contenu premier enfant : ' + inner1 +
          '\nContenu dernier enfant : ' + inner2);
  </script>
</body>
```

Source : cette page

Contenu premier enfant : Un deuxième

Contenu dernier enfant : paragraphe

OK

Les évènements en JS

Exemple

On souhaite afficher une boîte de dialogue de type alert() lorsqu'un utilisateur clique sur un paragraphe dans notre page web.

```
<body>
  <h1 id="gros_titre">Les évènements</h1>

  <p onclick="alert('Bravo !');">Cliquez-moi, cliquez-moi !</p>
  <p>Un deuxième <strong>paragraphe</strong></p>
</body>
```

Source : cette page

Bravo !

OK

L'utilisation du mot clé this

```
<body>  
  <h1 id="gros_titre">Les évènements</h1>  
  
  <p onclick="this.textContent='Merci !';">Cliquez-moi, cliquez-moi !</p>  
  <p>Un deuxième <strong>paragraphe</strong></p>  
</body>
```

Avant le clic

Les évènements

Cliquez-moi, cliquez-moi !

Un deuxième **paragraphe**

Après le clic

Les évènements

Merci !

Un deuxième **paragraphe**

Les évènements et le DOM

```
<body>
  <h1 id="gros_titre">Les évènements</h1>

  <p>Cliquez-moi, cliquez-moi !</p>
  <p>Un deuxième <strong>paragraphe</strong></p>
<script>
  //On accède à notre premier paragraphe
  var p1 = document.querySelector('p');

  /*Création d'un gestionnaire d'évènements pour
  *l'évènement "onclick"*/
  p1.onclick = function(){
    this.innerHTML = '<strong>Bravo !</strong>';
    this.style.color = 'orange';
  };
</script>
</body>
```

Avant le clic

Les évènements

Cliquez-moi, cliquez-moi !

Un deuxième **paragraphe**

Avant le clic

Les évènements

Bravo !

Un deuxième **paragraphe**

La méthode « `addEventListener()` »

Cette méthode permet de lier du code à un évènement. On parlera alors de gestionnaire d'évènements.

```
<body>
  <h1 id="gros_titre">Les évènements</h1>

  <p>Cliquez-moi, cliquez-moi !</p>
  <p>Un deuxième <strong>paragraphe</strong></p>
<script>
  //On accède à notre premier paragraphe
  var p1 = document.querySelector('p');

  //On accroche un gestionnaire d'évènements à p1
  p1.addEventListener('click',changeTexte);

  /*On construit notre fonction changeTexte qui ne sera
  *exécutée que lors du déclenchement de l'évènement*/
  function changeTexte(){
    this.innerHTML = '<strong>Bravo !</strong>';
    this.style.color = 'orange';
  };
</script>
</body>
```

Pourquoi utiliser « addEventListener() » ?

L'un des grands avantages de la méthode addEventListener() est de pouvoir lier plusieurs gestionnaires d'évènements de même type sur un élément HTML.

```
<body>
  <h1 id="gros_titre">Les évènements</h1>

  <p>Cliquez-moi, cliquez-moi !</p>
  <p>Un deuxième <strong>paragraphe</strong></p>
<script>
  //On accède à notre premier paragraphe
  var p1 = document.querySelector('p');

  //On accroche deux gestionnaires d'évènements à p1
  p1.addEventListener('click',Message1);
  p1.addEventListener('click',Message2);

  //Création des fonctions
  function Message1(){
    alert('Première boîte !');
  };

  function Message2(){
    alert('Deuxième boîte !');
  };
</script>
</body>
```

Source : cette page

Première boîte !

OK

Source : cette page

Deuxième boîte !

OK

Un autre exemple

On peut aussi exécuter un code lorsqu'un utilisateur déplace le curseur de sa souris sur un élément et un autre code lorsqu'il reste cliqué dessus.

```
<body>
  <h1 id="gros_titre">Les évènements</h1>
  <p>Passez sur moi svp</p>
  <p>Un deuxième <strong>paragraphe</strong></p>
<script>
  //On accède à notre premier paragraphe
  var p1 = document.querySelector('p');
  //On accroche deux gestionnaires d'évènements à p1
  p1.addEventListener('mouseover',Fonction1);
  p1.addEventListener('mousedown',Fonction2);
  //Création des fonctions
  function Fonction1(){
    this.innerHTML = 'Cliquez moi maintenant !';
    this.style.backgroundColor = 'orange';
  };
  function Fonction2(){
    this.innerHTML = 'Bravo !';
    this.style.color = '#26C';
    this.style.fontWeight = 'bold';
    this.style.fontSize= '24px';
  };
</script>
</body>
```

Avant le survol

Les évènements

Passez sur moi svp

Un deuxième paragraphe

Lors du survol

Les évènements

Cliquez moi maintenant !

Un deuxième paragraphe

Après le clic

Les évènements

Bravo !

Un deuxième paragraphe

Propagation des évènements

```
<body>
  <h1 id="gros_titre">Les évènements</h1>

  <div id="div">
    <p id="p">Un premier paragraphe</p>
    <p>Un deuxième paragraphe</p>
  </div>
<script>
  //On accède à notre div et à premier paragraphe
  var div = document.getElementById('div');
  var p1 = document.getElementById('p');

  //On crée deux gestionnaires d'évènements pour click
  div.addEventListener('click', MessageDiv);
  p1.addEventListener('click', MessageP);

  function MessageDiv(){
    alert('Evènement du div');
  };

  function MessageP(){
    alert('Evènement du paragraphe');
  };
</script>
</body>
```

(1)

Source : cette page

Evènement du paragraphe

OK

(2)

Source : cette page

Evènement du div

OK

Dans cet exemple, l'évènement du paragraphe se déclenche avant celui du div

Le BOM (Browser Object Model)

Introduction au BOM

- ✓ Le BOM permet d'accéder au navigateur, et notamment à la fenêtre ouverte actuellement en utilisant le JS.
- ✓ L'objet le plus célèbre et le plus utilisé du BOM est l'objet **Window**. L'objet **Window** va tout simplement représenter la fenêtre du navigateur.
- ✓ l'objet **Window** est dit implicite. Cela signifie que nous n'aurons généralement pas besoin de le mentionner pour utiliser les méthodes (ou fonctions globales) et propriétés (ou variables globales) lui appartenant.
- ✓ Par exemple, la fonction **alert()** est également une méthode de l'objet Window. Cependant, on ne précise jamais Window lorsqu'on utilise alert().

La méthode open() de Window

La méthode open() va nous servir à ouvrir un nouvel onglet ou une nouvelle fenêtre.

```
<body>
  <h1 id="gros_titre">Le BOM</h1>
  <p>Cliquez pour ouvrir un nouvel onglet !</p>

  <script>
    //On accède à notre paragraphe
    var para = document.querySelector('p');

    //On attache un gestionnaire d'évènement click
    para.addEventListener('click', fenetre);

    //On utilise open()
    function fenetre(){
      window.open('http://www.univ-oran1.dz/', '
        _blank', 'width = 500, height = 300');
    }
  </script>
</body>
```

La méthode close() de Window

```
<body>
  <h1 id="gros_titre">Le BOM</h1>
  <button id = 'ouvrir'>Ouvrir une nouvelle fenêtre</button>
  <button id = 'fermer'>Refermer la fenêtre</button>

  <script>
    var ouvrir = document.getElementById('ouvrir');
    var fermer = document.getElementById('fermer');
    var fenetre = '';

    ouvrir.addEventListener('click', fOuvrir);
    fermer.addEventListener('click', fFermer);

    function fOuvrir(){
      fenetre = window.open('http://
        www.univ-oran1.dz/', '_blank', 'width=500');
    }

    function fFermer(){
      fenetre.close();
    }
  </script>
</body>
```

Utilisation des propriétés de Screen

L'objet Screen nous donne accès à des informations concernant l'écran de vos visiteurs, comme la taille ou la résolution de l'écran par exemple.

```
<body>
  <h1 id="gros_titre">Le BOM</h1>
  <p></p>

  <script>
    var hauteur = screen.height;
    var hauteurDispo = screen.availHeight;
    var reso = screen.pixelDepth;

    var para = document.querySelector('p');
    para.innerHTML =
      'Hauteur de l\'écran : ' + hauteur +
      '<br>Hauteur dispo : ' + hauteurDispo +
      '<br>Résolution : ' + reso + ' bits/px';
  </script>
</body>
```

Le BOM

Hauteur de l'écran : 768

Hauteur dispo : 728

Résolution : 24 bits/px

Utilisation des propriétés de Navigator

L'objet Navigator va nous donner des informations sur le navigateur de vos visiteurs en soi ainsi que sur les préférences enregistrées (langue, etc.).

```
<body>
  <h1 id="gros_titre">Le BOM</h1>
  <p></p>

  <script>
    var langue = navigator.language;
    var navigateur = navigator.appName;
    var version = navigator.appVersion;
    var moteur = navigator.product;
    var cookieAutorise = navigator.cookieEnabled;

    var para = document.querySelector('p');
    para.innerHTML =
      'Langue utilisée : ' + langue +
      '<br>Nom du navigateur : ' + navigateur +
      '<br>Version : ' + version +
      '<br>Moteur : ' + moteur +
      '<br>Cookie ? : ' + cookieAutorise;
  </script>
</body>
```

Le BOM

Langue utilisée : fr-FR
Nom du navigateur : Netscape
Version : 5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36
Moteur : Gecko
Cookie ? : true

Utilisation de History (1/2)

L'objet History appartient à Window. Là encore, Window est implicite et nous n'avons pas besoin de le mentionner. History permet de nous déplacer dans l'historique de nos visiteurs et nous donner des informations relatives à celui-ci.

Utilisation de History (2/2)

```
<body>
  <h1 id="gros_titre">Le BOM</h1>
  <button id='precedent'>Charge la dernière page visitée</button>
  <button id='suivant'>Page suivante dans l'historique</button>
  <button id='specifique'>5 pages en arrière dans l'historique !</button>
  <p></p>
  <script>
    var para = document.querySelector('p');
    var precedent = document.getElementById('precedent');
    var suivant = document.getElementById('suivant');
    var specifique = document.getElementById('specifique');
    precedent.addEventListener('click', arriere);
    suivant.addEventListener('click', avant);
    specifique.addEventListener('click', spec);
    function arriere(){
      history.back();}
    function avant(){
      history.forward();}
    function spec(){
      history.go(-5);}
    //Nombre d'URL dans l'historique
    var histo = history.length;
    para.innerHTML = 'URLs : ' + histo;
  </script>
</body>
```

Le BOM

Charge la dernière page visitée

Page suivante dans l'historique

5 pages en arrière dans l'historique !

URLs : 2

Les expressions régulières (regex)

Introduction aux « regex »

- ✓ Une **expression régulière** va consister en une suite de caractères qui vont former, ensemble, un schéma de recherche.
- ✓ Les **expressions régulières** vont nous permettre de vérifier la présence de certains caractères ou suites de caractères dans une expression.
- ✓ Le grand intérêt des **expressions régulières** est qu'on peut les utiliser conjointement avec de nombreux langages, comme le JavaScript ou le PHP par exemple.
- ✓ Les **regex** vont nous permettre de vérifier, par exemple, qu'un utilisateur a bien mentionné une suite de dix chiffres dans le champ « numéro de téléphone » d'un formulaire d'inscription.

Syntaxe des « regex »

```
<body>
```

```
  <h1>regex et JS</h1>
```

```
  <script>
```

```
    //Pour construire une expression régulière,  
    //il suffit de préciser une séquence de caractères  
    // et d'entourer cette séquence par un slash "/"
```

```
    var r1 = /Mohammed/;
```

```
    // Le caractère i signifie que notre regex va  
    // ignorer la casse de notre séquence de caractères  
    // (i signifie ici « case-insensitive »).
```

```
    var r2 = /Mohammed/i;
```

```
  </script>
```

```
</body>
```

La méthode match() de l'objet String

La méthode match() de l'objet String permet de rechercher des caractères dans une séquence de caractères.

```
<body>
  <h1>Expressions régulières et JavaScript</h1>
  <p>Engagez-vous qu'ils disaient, rengagez-vous qu'ils disaient !</p>
  <p id="reg"></p>

  <script>
    /*On accède à notre paragraphe dans lequel on souhaite
    *faire notre recherche*/
    var expr = document.querySelector('p');

    //Première regex, avec l'option i
    var r1 = /engagez/i;

    //Deuxième regex, avec les options i et g
    var r2 = /engagez/ig;

    //On fait nos recherches avec match()
    var res1 = expr.textContent.match(r1);
    var res2 = expr.textContent.match(r2);

    //On affiche les résultats
    var resultat = document.getElementById('reg');
    resultat.innerHTML =
      'Résultat match() sur regex 1 : ' + res1 +
      '<br>Résultat match() sur regex 2 : ' + res2;
  </script>
</body>
```

Engagez-vous qu'ils disaient, rengagez-vous qu'ils disaient !

Résultat match() sur regex 1 : Engagez

Résultat match() sur regex 2 : Engagez,engagez

La méthode search() de l'objet String

La méthode search() retourne la position à laquelle a été trouvée la première occurrence de l'expression recherchée dans une chaîne de caractères ou -1 si elle n'a pas été trouvée.

```
<script>
/*On accède à notre paragraphe dans lequel on souhaite
 *faire notre recherche*/
var expr = document.querySelector('p');

//Première regex, avec l'option i
var r1 = /engagez/i;

//Deuxième regex, sans option
var r2 = /engagez/;

//On fait nos recherches avec search()
var res1 = expr.textContent.search(r1);
var res2 = expr.textContent.search(r2);

//On affiche les résultats
var resultat = document.getElementById('reg');
resultat.innerHTML =
    'Résultat search() sur regex 1 : ' + res1 +
    '<br>Résultat search() sur regex 2 : ' + res2;
</script>
```

Engagez-vous qu'ils disaient, reengagez-vous qu'ils disaient !

Résultat search() sur regex 1 : 0

Résultat search() sur regex 2 : 31

La méthode test() de l'objet regExp

La méthode test() va rechercher une séquence de caractères dans une chaîne de caractères.

```
<script>
//On accède au texte de notre paragraphe
var rec = document.querySelector('p').textContent;

//Première regex
var r1 = /ENGAGEZ/;

//Deuxième regex, avec l'option i (case insensitive)
var r2 = /ENGAGEZ/i;

//On fait nos recherches avec test()
var res1 = r1.test(rec);
var res2 = r2.test(rec);

//On affiche les résultats
var resultat = document.getElementById('reg');
resultat.innerHTML =
    'Résultat 1 : ' + res1 +
    '<br>Résultat 2 : ' + res2;
</script>
```

Engagez-vous qu'ils disaient, rengagez-vous qu'ils disaient !

Résultat 1 : false

Résultat 2 : true

Quantificateurs

Les **quantifieurs** sont des signes qui permettent de définir une « quantité » d'une séquence à rechercher.

Par exemple, le quantifieur **+** doit être placé juste après le caractère ou la séquence sur lesquels il doit agir.

```
<body>
  <h1>Expressions régulières et JavaScript</h1>
  <p>1, 10, 100</p>
  <p id="reg"></p>

  <script>
    //On accède au texte de notre paragraphe
    var rec = document.querySelector('p').textContent;

    //On recherche "1" suivi d'au moins un "0" dans rec
    var r1 = /10+/g;

    //On fait notre recherche avec match()
    var res1 = rec.match(r1);

    //On affiche le résultat dans notre deuxième paragraphe
    var resultat = document.getElementById('reg');
    resultat.innerHTML = 'Résultat : ' + res1;
  </script>
</body>
```

Expressions régulières et JavaScript

1, 10, 100

Résultat : 10,100

Liste des quantifieurs disponibles

QUANTIFIEUR	SENS
$a?$	On veut 0 ou 1 « a »
a^+	On veut au moins un « a »
a^*	On veut 0, 1 ou plusieurs « a »
a	On veut un « a » en début de chaîne
$a\$$	On veut un « a » en fin de chaîne
$a\{X\}$	On veut une séquence de X « a »
$a\{X,Y\}$	On veut une séquence de X à Y fois « a »
$a\{X,\}$	On veut une séquence d'au moins X fois « a »
$a(?:=b)$	On veut un « a » suivi d'un « b »
$a(?:!b)$	On veut un « a » non suivi d'un « b »

Exemples d'utilisation des quantifieurs

```
<body>
  <h1>Expressions régulières et JavaScript</h1>
  <p>On apprend à utiliser les regex en JavaScript</p>
  <p id="reg"></p>
  <script>
    //On accède au texte de notre paragraphe
    var rec = document.querySelector('p').textContent;
    //On veut savoir si notre paragraphe contient un "x"
    var r1 = /x+/;
    var res1 = r1.test(rec);
    /*On cherche "e" ou "end" (on utilise les parenthèses)
    *pour faire porter "?" sur "nd"*/
    var r2 = /e(nd)?/g;
    var res2 = rec.match(r2);
    //Notre paragraphe commence par "o" et finit par "e" ?
    var r3 = /^oe$/i;
    var res3 = r3.test(rec);
    var resultat = document.getElementById('reg');
    resultat.innerHTML =
      'Présence d\'un "x" : ' + res1 +
      '<br>Occurrences de "e" ou "end" : ' + res2 +
      '<br>Commence par "o", finit par "e" ? ' + res3;
  </script>
</body>
```

Expressions régulières

On apprend à utiliser les regex en JavaScript

Présence d'un "x" : true

Occurrences de "e" ou "end" : end,e,e,e,e,e

Commence par "o", finit par "e" ? false

Les options ou modificateurs

- ❑ Les options, qu'on appelle également modificateurs, vont servir à ajouter des options à notre recherche.
- ❑ Il existe trois options :
 - ✓ **L'option i** permet d'accepter toutes les combinaisons minuscules / majuscules représentant la séquence de notre regex.
 - ✓ **L'option g** sert à exécuter des recherches globales : plutôt que de s'arrêter au premier résultat trouvé, on va chercher tous les résultats disponibles.
 - ✓ **L'option m** permet d'effectuer notre recherche sur plusieurs lignes.

Exemple

```
<h1>Expressions régulières et JavaScript</h1>
<p>On apprend à utiliser les regex en JavaScript</p>
<p id="reg"></p>
<script>
    //On accède au texte de notre paragraphe
    var rec = document.querySelector('p').textContent;
    var rec2 = 'Je suis sur \ndeux lignes en Js'
    //On cherche "on", "On", "oN" ou "ON"
    var r1 = /on/i;
    var res1 = rec.match(r1);
    //On cherche tous les 'e'
    var r2 = /e/g;
    var res2 = rec.match(r2);
    //On cherche si une ligne commence par "d" (sans m) dans rec2
    var r3 = /^d/;
    var res3 = r3.test(rec2);
    //On cherche si une ligne commence par "d" (avec m) dans rec2
    var r4 = /^d/m;
    var res4 = r4.test(rec2);
    var resultat = document.getElementById('reg');
    resultat.innerHTML =
        'Résultat regex 1 : ' + res1 +
        '<br>Résultat regex 2 : ' + res2 +
        '<br>Résultat regex 3 : ' + res3 +
        '<br>Résultat regex 4 : ' + res4;
</script>
```

Expressions régulières

On apprend à utiliser les regex en JavaScript

Résultat regex 1 : On
Résultat regex 2 : e,e,e,e,e,e
Résultat regex 3 : false
Résultat regex 4 : true

Les classes de caractères

- ✓ Les classes de caractères vont nous permettre de mentionner des plages de caractères pour effectuer nos recherches.
- ✓ Pour déclarer une classe de caractères, nous allons devoir utiliser une paire de crochets ouvrant et fermant, sauf dans un cas particulier où nous utiliserons des parenthèses.

```
<script>
//On accède au texte de notre paragraphe
var rec = document.querySelector('p').textContent;

//On cherche "aeiouy"
var r1 = /aeiouy/g;
var res1 = rec.match(r1);

//On cherche tous les 'a', 'e', 'i', 'o', 'u' et 'y'
var r2 = /[aeiouy]/g;
var res2 = rec.match(r2);

var resultat = document.getElementById('reg');
resultat.innerHTML =
    'Recherche sans classe : ' + res1 +
    '<br>Recherche avec classe : ' + res2;
</script>
```

On apprend à utiliser les regex en JavaScript

Recherche sans classe : null

Recherche avec classe : a,e,u,i,i,e,e,e,e,a,i

Exemples d'écriture de classes

CLASSE	SENS
[abcde]	Trouve tous les caractères à l'intérieur des crochets
[^abcde]	Trouve tout caractère ne se situant pas entre les crochets
[a-z]	Trouve n'importe quelle lettre entre a et z
[^a-z]	Trouve n'importe quel caractère qui n'est pas une lettre minuscule de l'alphabet
[0-9]	Trouve n'importe quel nombre entre 0 et 9
[^0-9]	Trouve n'importe quel caractère qui n'est pas un nombre compris entre 0 et 9
(jour soir)	Trouve jour et soir

Java Script et les formulaires

Exemple d'un formulaire

```
<body>
  <h1>Les formulaires HTML</h1>

  <!--Nous ne créerons pas la page "traitement.php" dans ce cours.
  Cette page doit normalement effectuer le traitement des données.
  Le bouton "valider" ne fonctionnera donc pas ici.-->
  <form metod="post" action="traitement.php">
    <label for='prenom'>Entrez votre prénom svp : </label>
    <input type='text' name='prenom' id='prenom'><br><br>

    <label for='mail'>Entrez votre mail : </label>
    <input type='email' name='mail' id='mail'><br><br>

    <label for='tel'>Numéro de téléphone :</label>
    <input type='tel' name='tel' id='tel'><br><br>

    <input type='submit' value='Valider'>
  </form>
</body>
```

Les formulaires HTML

Entrez votre prénom svp :

Entrez votre mail :

Numéro de téléphone :

Valider

Une première validation en HTML

```
<body>
  <h1>Les formulaires HTML</h1>

  <!--Nous ne créerons pas la page "traitement.php" dans ce cours.
  Cette page doit normalement effectuer le traitement des données.
  Le bouton "valider" ne fonctionnera donc pas ici.-->
  <form metod="post" action="traitement.php">
    <label for='prenom'>Entrez votre prénom svp : </label>
    <input type='text' name='prenom' id='prenom' maxlength="20" required><br><br>

    <label for='mail'>Entrez votre mail : </label>
    <input type='email' name='mail' id='mail' required><br><br>


    <label for='tel'>Numéro de téléphone :</label>
    <input type='tel' name='tel' id='tel' required><br><br>

    <input type='submit' value='Valider'>
  </form>
</body>
```

-> Limitée !!!

Les formulaires HTML

Entrez votre prénom svp :

Entrez votre mail :  Veuillez renseigner ce champ.

Numéro de téléphone :

Validation en JS

```
<body>
  <h1>Les formulaires HTML</h1>

  <!--Nous ne créerons pas la page "traitement.php" dans ce cours.
  Cette page doit normalement effectuer le traitement des données.
  Le bouton "valider" ne fonctionnera donc pas ici.-->
  <form metod="post" action="traitement.php">
    <label for='prenom'>Entrez votre prénom svp : </label>
    <input type='text' name='prenom' id='prenom' maxlength="20" required><br><br>
    <label for='mail'>Entrez votre mail : </label>
    <input type='email' name='mail' id='mail' required><br><br>
    <label for='tel'>Numéro de téléphone :</label>
    <input type='tel' name='tel' id='tel' required><br><br>
    <input type='submit' value='Valider' id='bouton_envoi'>
  </form>
  <script>
    var formValid = document.getElementById('bouton_envoi');
    formValid.addEventListener('click', validation);

    function validation(){

    }
  </script>
</body>
```


Vérifier la présence d'une valeur

```
<input type='submit' value='Valider' id='bouton_envoi'>
</form>

<script>
var formValid = document.getElementById('bouton_envoi');
var prenom = document.getElementById('prenom');
var missPrenom = document.getElementById('missPrenom');

formValid.addEventListener('click', validation);

function validation(event){
    //Si le champ est vide
    if (prenom.validity.valueMissing){
        event.preventDefault();
        missPrenom.textContent = 'Prénom manquant';
        missPrenom.style.color = 'red';
    }
}
</script>
</body>
```

Les formulaires HTML

Entrez votre prénom svp : Prénom manquant

Entrez votre mail :

Numéro de téléphone :

Vérifier la qualité des données envoyées

Une adresse e-mail contient trois parties distinctes :

- ✓ La partie locale, avant l'arobase;
- ✓ L'arobase @;
- ✓ Le domaine, lui-même composé du label « ex: gmail » et de l'extension « ex: com ».

Vérifier la qualité des données envoyées

```
<script>
  var formValid = document.getElementById('bouton_envoi');
  var mail = document.getElementById('mail');
  var missMail = document.getElementById('missMail');
  var mailValid = /^[a-z0-9._-]+@[a-z0-9._-]+\.[a-z]{2,6}$/;

  formValid.addEventListener('click', validation);

  function validation(event){
    //Si le champ est vide
    if (mail.validity.valueMissing){
      event.preventDefault();
      missMail.textContent = 'Mail manquant';
      missMail.style.color = 'red';
    }
    //Si le format de données est incorrect
    }else if (mailValid.test(mail.value) == false){
      event.preventDefault();
      missMail.textContent = 'Format incorrect';
      missMail.style.color = 'orange';
    }else{
    }
  }
}</script>
```

Explication de la regex

La partie locale : composée de lettres, de chiffres et d'un tiret, un trait de soulignement et un point : `/^[a-z0-9._-]+$`

On ajoute l'arobase : `/^[a-z0-9._-]+@$/`

le nom de domaine : composé de lettres, de chiffres, de tirets et de traits de soulignement : `/^[a-z0-9._-]+@[a-z0-9._-]+$`

Puis vient le point de l'extension du domaine : attention à ne pas oublier de l'échapper, car il s'agit d'un métacaractère. Et pour finir, l'extension ! Une extension de nom de domaine ne contient que des lettres, au minimum 2, au maximum 6. Ce qui nous fait :

`/^[a-z0-9._-]+@[a-z0-9._-]+\.[a-z]{2,6}$/`

Notions avancées JS

La méthode setInterval()

Permet d'exécuter un script en boucle en précisant un intervalle de temps entre chaque répétition

```
<body>
  <p>Horloge actualisée en direct ! Il est <span id='t'></span></p>

  <script>
    var heure = document.getElementById('t');
    var tempsReel = setInterval(horloge, 1000);

    function horloge(){
      var d = new Date();
      heure.innerHTML = d.toLocaleTimeString();
    }
  </script>
</body>
```

Horloge actualisée en direct ! Il est 22:49:32

Créer un cookie en JavaScript

Les cookies sont très pratiques car ils permettent de conserver des informations et donc de pouvoir s'en resservir.

```
<body>
  <p>Horloge actualisée en direct ! Il est <span id='t'></span></p>

  <script>
    // On crée un cookie prenom contenant la valeur Mohammed
    // qui expire le 15 Dec 2018 à midi sauf si l'utilisateur
    // le supprime de son ordinateur avant
    document.cookie = 'prenom=Mohammed; expires=Mon, 31 Dec 2018 12:00:00 UTC; path='/';

  </script>
</body>
```

Récupérer et lire un cookie

```
<body>
  <h1>Les cookies en JavaScript</h1>
  <p id='cook'></p>

  <script>
    document.cookie='prenom1=Mohammed; expires=Mon, 31 Dec 2018 12:00:00 UTC; path='/';
    document.cookie='prenom2=Reda; expires=Mon, 24 Dec 2018 12:00:00 UTC; path='/';
    var c = document.cookie;
    document.getElementById('cook').innerHTML = c;
  </script>
</body>
```

Les cookies en JavaScript

prenom1=Mohammed; prenom2=Reda

Questions ?