

Université Sidi Mohamed Ben Abdallah

Faculté des sciences Dhar El Mahraz
Licence professionnelle SIGL

Web dynamique: JavaScript

RÉALISÉ PAR: PR. MAHRAZ MED ADNANE

ANNÉE UNIVERSITAIRE:2017/2018

Expressions régulières

```
var exp = new RegExp('modèle',['options']);  
var exp = /modèle/[options]
```

► Options :

- **i** insensible à la casse
- **m** multi-ligne
- **g** toutes les occurrences

Méthodes

- **exp.test(ch)** true si correspondance
- **exp.match(ch)** tableau de correspondances et les groupes mémorisés
- **exp.exec(ch)** première occurrence capturées
- **ch.search(exp)** indice 1ere occurrence
- **ch.replace(exp, ch2)** remplace les occurrences par ch2

```
var exp = /(\d{2}) (\w+) (\d{4})/;  
var ch = "28 octobre 2011";  
if (exp.test(ch)){  
  alert(ch.replace(exp, "annee : $3, mois : $2, jour : $1"));  
  // annee : 2011, mois : octobre, jour : 28  
}
```

Expressions régulières

Indicateurs d'occurrence

<code>{n}</code>	exactement n fois	<code>*</code>	<code>{0,}</code>
<code>{n,}</code>	au moins n fois	<code>+</code>	<code>{1,}</code>
<code>{n,m}</code>	entre n et m fois	<code>?</code>	<code>{0,1}</code>

Caractères spéciaux

<code> </code>	ou	<code>.</code>	tout caractère sauf <code>\n</code>
<code>\t</code>	tabulation	<code>\n</code>	saut de ligne
<code>\0</code>	car. nul	<code>\</code>	car. d'échappement

Classes de caractères

[abc]	un caractère parmi a, b ou c		
[a-z]	intervalle : un caractère de a à z		
[^ab]	un caractère autre que a ou b		
\d	un chiffre	\D	tout sauf un chiffre
\w	[a-zA-Z0-9_]	\W	tout sauf mot
\s	espacement	\S	tout sauf car. esp.

Correspondances dans la chaîne

`^` début `$` fin

Mémorisation

`(x)` Mémoriser sous expression x

Expressions régulières: Exemples

```
<script type="text/javascript">
true      document.write(/l/.test('Hello')) ;
false     document.write(/^l/.test('Hello')) ;
false     document.write(/^h/.test('Hello')) ;
true      document.write(/^h/i.test('Hello')) ;
true      document.write(/^Hel.o/.test('Hello')) ;
true      document.write(/^Hel+o/.test('Hello')) ;
true      document.write(/^He+llo/.test('Hello')) ;
true      document.write(/^Hea*llo$/ .test('Hello')) ;
true      document.write(/^He(l|o)*$/ .test('Hello')) ;
true      document.write(/^H[leos]+/.test('Hello')) ;
false     document.write(/^H[^leo]+/.test('Hello')) ;
true      document.write(/^H[^kyz]+/.test('Hello')) ;
true      document.write(/^H[a-z]*/.test('Hello')) ;
true      document.write(/^H[a-z]*$/ .test('Hello')) ;
</script>
```



DOM

DOCUMENT OBJECT MODEL

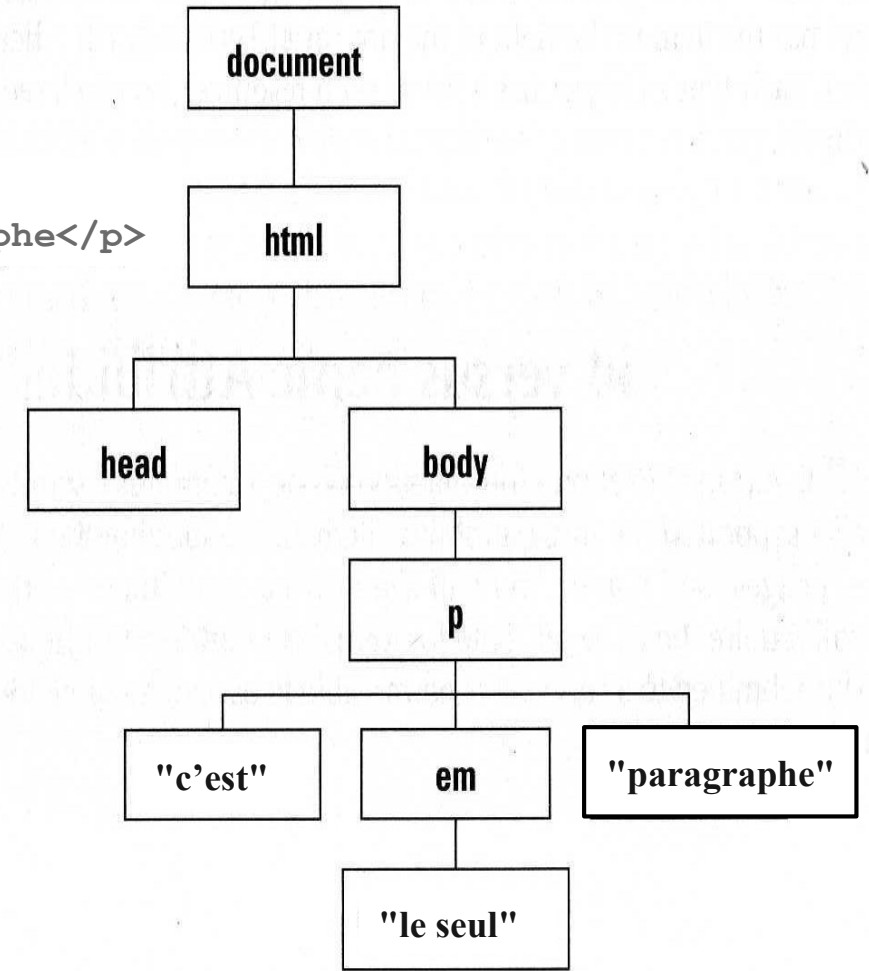
DOM = Document Object Model

- ▶ Le modèle d'objet du document (DOM) donne une représentation en mémoire des objets du document.
- ▶ Un objet est un élément HTML qui a été défini par une ID ou un nom.
- ▶ Le DOM est l'adresse par laquelle vous pouvez localiser un objet de la page HTML.
- ▶ Un objet peut être récupéré et exploité par le Javascript (ou un autre langage de programmation).

- Le DOM décrit le chemin partant de la fenêtre du navigateur pour descendre jusqu'à l'objet de la page Web.
- Le DOM est structuré comme un arbre et suit de près la structure hiérarchique du code HTML.
- L'arbre contient des nœuds, les nœuds peuvent avoir des fils, et tous les nœuds ont un parent (sauf la racine).

DOM(Document Object Model)

```
<html>  
  <head></head>  
  <body>  
    <p>c'est <em>le  
    seul</em> paragraphe</p>  
  </body>  
</html>
```



Utilisation du DOM

A l'aide de Javascript :

- ▶ On peut sélectionner un élément (<p> par exemple), et modifier sa couleur (**DOM document + DOM element**).
- ▶ On peut sélectionner un élément et lui assigner un événement (**DOM document + DOM events**).
- ▶ On peut sélectionner les attributs ("title" par exemple) et changer leur contenu (je remplace title="image2" par title="beau tigre") (**DOM document + DOM attribute**).

DOM « document »

- ▶ Le DOM « document » qui permet de sélectionner un objet au sein d'un document:

- `document.getElementById()` ; // par son ID
- `document.getElementsByName()` ; // par son attribut « name »
- `document.getElementsByTagName()` ; // par son nom de balise HTML
- `document.getElementsByClassName()` ; // par son nom de la classe

```
<input type="search" class="c1" name="rechercher" id="rechercher"/>
```

```
var zone1=document.getElementById("rechercher");  
var zone2=document.getElementsByName("rechercher");  
var zone3=document.getElementsByTagName("input");  
var zone3=document.getElementsByClassName("c1");
```

DOM « document »

- ▶ Deux autres méthodes basées sur les selecteurs CSS:
 - ▶ **querySelectorAll(selector)**: retourne tous les éléments correspondant au **selector css**.
 - ▶ **querySelector(selector)**: retourne uniquement le premier élément trouvé.

```
<input type="text" class="c1" name="text" id="text"/>  
<input type="search" class=" c1" name="recherche" id="rechercher"/>
```

```
document.querySelectorAll(". c1")[1]; //retourne le deuxième élément qui a la  
classe « c1 ».  
var zone2=document.querySelector("#text"); //retourne le premier élément qui a un  
id « text ».
```

DOM « element » + DOM « attribute »

- ▶ Le DOM « element » permet de faire une action sur les éléments sélectionnés.
- ▶ Le DOM « attribute » permet de modifier les attributs des éléments sélectionnés

```
<input type="search" name="recherche" id="rechercher"/>
```

```
var zone1=document.getElementById("rechercher"); // selectionne la zone de recherche
zone1.style.color="red"; //change la couleur du texte
Zone1.style.setProperty('color','red'); //changer une propriété
Var attribut=zone1.getAttribute("type"); // renvoie « search »
zone1.setAttribute("placeholder", " Texte à chercher "); // ajout l'attribut « placeholder »
zone1.removeAttribute("placeholder"); // supprime l'attribut « placeholder »
```

DOM « events »

- Le DOM « events » permet de faire une action lors d'un événement (exemple au clic de la souris)

```
<input type="search" placeholder="Texte à chercher" name="recherche" id="c"/>
```

```
<script>  
  var zone=document.getElementById("c");  
  zone.onfocus=function(){  
    zone.value="";  
  }  
</script>
```

```
<script>  
  var zone=document.getElementById("c");  
  zone.onfocus=vider;  
  function vider(){  
    zone.value="";  
  }  
</script>
```

DOM « events »: `addEventListener`

La méthode **`addEventListener`** permet d'abonner à l'objet sur lequel elle est invoquée une fonction pour l'événement précisé.

`objet.addEventListener(eventType, listenerFunction)`

- `objet` : l'objet ciblé (window, document ou un élément de la page).
- `eventType` : une chaîne de caractères désignant le type d'événement:
 - ▶ **`"click"`**, **`"load"`**, **`"change"`**, **`"mouseover"`**, **`"keypress"`** etc.
- `listenerFunction` : la fonction listener qui sera appelée lorsque l'événement se produit

DOM « events »: addEventListener (eventType)

- click
 - dblclick
 - mousedown
 - mousemove
 - mouseover
 - mouseout
 - mouseup
 - keydown
 - keypress
 - keyup
 - abort
- Error
 - load
 - resize
 - scroll
 - unload
 - blur
 - change
 - focus
 - reset
 - select
 - submit

DOM « events »: addEventListener

► Exemple 1:

```
window.addEventListener('load', function() {  
    console.log('la page est totalement chargée');  
});
```

➤ Exemple 2:

```
document.addEventListener('DOMContentLoaded', function() {  
    var img = document.getElementsByTagName("img")[0];  
    img.addEventListener('mouseover', function() {  
        img.style.opacity=0.5;  
    });  
    img.addEventListener('mouseout', function() {  
        img.style.setProperty('opacity', 1);  
    });  
});
```

DOM « events »:

addEventListener (listenerFunction)

- dans une fonction **listener**, la variable **this** est définie et désigne l'objet qui a déclenché l'événement typiquement l'élément du document.
- un objet **event** est créé pour chaque événement. Cet objet est passé en paramètre lors du déclenchement de la fonction **listener** associée. Le type d'objet **event** varie selon l'événement. Un objet **event** possède des propriétés qui informent sur l'événement.

DOM « events »: addEventListener

► Exemple:

```
document.addEventListener('DOMContentLoaded', function() {  
    var box=document.querySelector('#box');  
    box.addEventListener('click', function(event) {  
        //this.style.backgroundColor='red';  
        event.target.style.backgroundColor='red';  
        console.log(event.target);  
    });  
});
```



Event
interface

```
<div id="box">  
    <button>cliquer ici</button>  
</div>
```

DOM « events »: addEventListener

► Exemple

```
document.addEventListener('DOMContentLoaded', function() {  
    var box=document.querySelector('#box');  
    box.addEventListener('click',function(event) {  
        console.log('tu as cliqué sur la div');  
    });  
  
    var bt=document.querySelector('button');  
    bt.addEventListener('click',function(event) {  
        event.stopPropagation();  
        if (event.target.tagName=='BUTTON') {  
            console.log('tu as cliqué sur le bouton');  
        }  
    });  
});
```

Modifier le DOM

Deux Solutions pour modifier le DOM:

► **innerHTML**

- Construire une chaîne de caractères contenant du code HTML
- Affecter cette chaîne de caractères à l'attribut **innerHTML** d'un élément de la page
- Le navigateur interprète le contenu de la chaîne de caractères affectée pour modifier la structure du document

► **DOM « pur »**

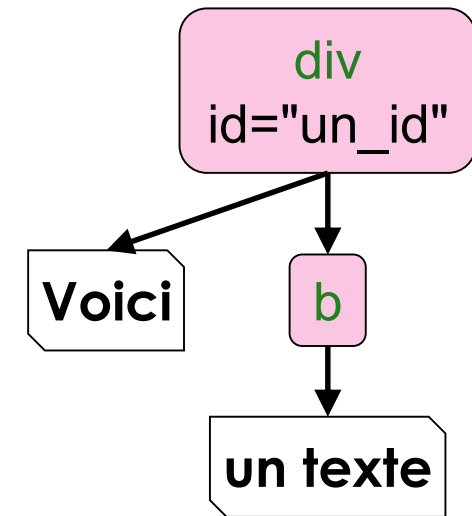
- Construire de nouveaux éléments logiques du document
- Construire les liens de parenté entre ces éléments

Modifier le DOM: **innerHTML**

- ▶ Identifier un élément HTML
`<div id="un_id"></div>`
- ▶ Accéder à un élément
`e = document.getElementById("un_id") ;`
- ▶ Construire une chaîne contenant du HTML
`s = "Voici un texte" ;`
- ▶ Modifier le contenu de l'élément
`e.innerHTML = s ;`
- ▶ Interprétation « automatique » par le navigateur du nouveau contenu pour modifier le document

Modifier le DOM « pur »

- ▶ Identifier un élément HTML
`<div id="un_id"></div>`
- ▶ Accéder à un élément
`e = document.getElementById("un_id");`
- ▶ Créer un nœud de type « texte »
`t1 = document.createTextNode('Voici ');`
`t2 = document.createTextNode('un texte');`
- ▶ Créer un nouveau nœud de type « balise »
`b = document.createElement('b');`
- ▶ Construire des liens de parenté
`e.appendChild(t1);`
`b.appendChild(t2); e.appendChild(b);`



Exercice

- Ecrire une page HTML qui contient un script qui verifie lors de la soumission du formulaire, si le champs n'est pas vide, sinon, un message d'erreur est affiché.
 1. Utiliser innerHTML pour insérer le message dans la balise span, en gras avec une couleur rouge.
 2. Utiliser une modification manuelle sur le DOM.

Entrer votre nom	Ce champs est obligatoire	↖	span
Entrer votre prénom			
Envoyer			

Correction

```
<script>
function verifier(objet){
    var text1=objet.nom.value;
    var text2=objet.prenom.value;
    if (objet.nom.value==""){
        var sp=document.getElementById("c1");
        var b=document.createElement("b");
        var text=document.createTextNode("Ce champs est obligatoire");
        b.appendChild(text);
        sp.appendChild(b);
        sp.style.color="red";
    }else{
        document.getElementById("c1").innerHTML="";
    }
}
```

```
        if (objet.prenom.value==""){
            var sp=document.getElementById("c2");
            var chaine="<b>Ce champs est obligatoire<b>";
            sp.innerHTML=chaine;
            sp.style.color="red";
        }else{
            document.getElementById("c2").innerHTML="";
            if (objet.nom.value=="" || objet.prenom.value=="")return false;
            return true;
        }
    }
</script>
```