

# Plan

- 1 Introduction
- 2 Récupérer un élément HTML
- 3 Récupérer/modifier l'attribut d'un élément HTML
- 4 Récupérer/modifier le contenu d'un élément HTML
- 5 Le parent et les enfants d'un élément HTML
- 6 Ajouter un nouvel élément HTML
- 7 Autres opérations sur les éléments HTML
- 8 Modifier les propriétés CSS d'un élément HTML
- 9 Les évènements

# JavaScript

## DOM : Document Object Model

- Interface de programmation (API) pour les documents XML et HTML
- Graphiquement, ça correspond à un arbre dont les nœuds sont les balises HTML
- Avant standardisation par le W3C, chaque navigateur avait son propre DOM.
- Utilisé par les scripts pour modifier un document HTML en
  - ajoutant un nœud
  - modifiant un autre
  - remplaçant un premier par un deuxième
  - supprimant un autre

# JavaScript

## Considérons la page HTML suivante

```
<!DOCTYPE html>
<html>
  <head>
    <title>My JS Page</title>
  </head>
  <body>
    <div id = container>
      Considerons les paragraphes suivants :
      <p class=blue> bonjour </p>
      <p class=red> bonsoir, voici <a href="http://www
        .lsis.org/elmouelhia/"> ma page </a></p>
      <p class="blue"> salut </p>
    </div>
    <script src="fichier.js"></script>
  </body>
</html>
```

# JavaScript

## Récupérer un élément selon son `id`

```
var container = document.getElementById('container');  
console.log(container.innerHTML);
```

# JavaScript

## Récupérer un élément selon son id

```
var container = document.getElementById('container');  
console.log(container.innerHTML);
```

## Récupérer des éléments selon le nom de la balise

```
var paragraphes = document.getElementsByTagName("p");  
for (elt of paragraphes)  
    console.log(elt.innerHTML);
```

# JavaScript

## Récupérer un élément selon son `id`

```
var container = document.getElementById('container');  
console.log(container.innerHTML);
```

## Récupérer des éléments selon le nom de la balise

```
var paragraphes = document.getElementsByTagName("p");  
for (elt of paragraphes)  
    console.log(elt.innerHTML);
```

## Ou encore

```
var paragraphes = container.getElementsByTagName("p");  
for (elt of paragraphes)  
    console.log(elt.innerHTML);
```

# JavaScript

## Récupérer un élément selon sa classe

```
var bleus = document.getElementsByClassName("blue");  
for (bleu of bleus)  
    console.log(bleu.innerHTML);
```

# JavaScript

## Récupérer un élément selon sa classe

```
var bleus = document.getElementsByClassName("blue");  
for (bleu of bleus)  
    console.log(bleu.innerHTML);
```

## Ou aussi

```
var bleus = container.getElementsByClassName("blue")  
;  
for (bleu of bleus)  
    console.log(bleu.innerHTML);
```



# JavaScript

## Récupérer un élément selon un sélecteur CSS 3

```
var pbleus = document.querySelectorAll("p.blue");  
for (bleu of pbleus)  
    console.log(bleu.innerHTML);
```

# JavaScript

## Récupérer un élément selon un sélecteur CSS 3

```
var pbleus = document.querySelectorAll("p.blue");  
for (bleu of pbleus)  
    console.log(bleu.innerHTML);
```

## Récupérer le premier élément selon un sélecteur CSS 3

```
var pbleu = document.querySelector("p.blue");  
console.log(pbleu.innerHTML);
```

# JavaScript

## Récupérer un élément selon un sélecteur CSS 3

```
var pbleus = document.querySelectorAll("p.blue");  
for (bleu of pbleus)  
    console.log(bleu.innerHTML);
```

## Récupérer le premier élément selon un sélecteur CSS 3

```
var pbleu = document.querySelector("p.blue");  
console.log(pbleu.innerHTML);
```

## Ceci déclenche une erreur

```
var prouges = document.querySelector("p.red");  
for (rouge of prouges)  
    console.log(rouge.innerHTML);
```

# JavaScript

## Récupérer l'attribut d'un élément HTML

```
var lien = document.getElementsByTagName('a');  
var href = lien[0].getAttribute('href');  
console.log(href);
```

# JavaScript

## Récupérer l'attribut d'un élément HTML

```
var lien = document.getElementsByTagName('a');  
var href = lien[0].getAttribute('href');  
console.log(href);
```

## Modifier l'attribut d'un élément HTML

```
lien[0].setAttribute('href', 'http://www.  
francefootball.fr');  
console.log(lien[0]);
```

# JavaScript

**Pour certains attributs comme `href`, on peut accéder directement à la valeur via son nom**

```
var lien = document.getElementsByTagName('a');  
var href = lien[0].href;  
console.log(href);
```

# JavaScript

**Pour certains attributs comme `href`, on peut accéder directement à la valeur via son nom**

```
var lien = document.getElementsByTagName('a');  
var href = lien[0].href;  
console.log(href);
```

**Modifier l'attribut d'un élément HTML**

```
lien[0].href='http://www.francefootball.fr';  
console.log(lien[0]);
```

# JavaScript

## Remarques

- Les attributs `class` et `for` sont des mots clés réservés en JS, on ne peut donc les utiliser pour modifier leurs valeurs HTML.



# JavaScript

## Remarques

- Les attributs `class` et `for` sont des mots clés réservés en JS, on ne peut donc les utiliser pour modifier leurs valeurs HTML.

## Solution

- Utiliser `className` ou `classList` pour l'attribut `class`
- Et `forHtml` pour l'attribut `for`

# JavaScript

## Récupérer la classe d'un élément HTML

```
var container = document.getElementById('container')  
;  
console.log(container.className);
```

# JavaScript

## Récupérer la classe d'un élément HTML

```
var container = document.getElementById('container')  
;  
console.log(container.className);
```

## Modifier une classe d'un élément HTML

```
container.className = "blue";  
console.log(container.className);
```

# JavaScript

## Récupérer la classe d'un élément HTML

```
var container = document.getElementById('container')  
;  
console.log(container.className);
```

## Modifier une classe d'un élément HTML

```
container.className = "blue";  
console.log(container.className);
```

## Ajouter une nouvelle classe à un élément HTML

```
container.className += "_red";  
console.log(container.className);
```

# JavaScript

## Récupérer la liste des classes d'un élément HTML

```
var container = document.getElementById('container');  
var list = container.classList;  
console.log(list);
```

# JavaScript

## Récupérer la liste des classes d'un élément HTML

```
var container = document.getElementById('container');  
var list = container.classList;  
console.log(list);
```

## Ajouter une classe à un élément HTML

```
var container = document.getElementById('container');  
var list = container.classList;  
list.add('green');  
console.log(list);
```

# JavaScript

## Récupérer la liste des classes d'un élément HTML

```
var container = document.getElementById('container');  
var list = container.classList;  
console.log(list);
```

## Ajouter une classe à un élément HTML

```
var container = document.getElementById('container');  
var list = container.classList;  
list.add('green');  
console.log(list);
```

## Ajouter des classes à un élément HTML

```
var container = document.getElementById('container');  
var list = container.classList;  
list.add('red', 'blue');  
console.log(list);
```

# JavaScript

## Supprimer une classe d'un élément HTML

```
list.remove('red');  
console.log(list);
```



# JavaScript

## Supprimer une classe d'un élément HTML

```
list.remove('red');  
console.log(list);
```

## Supprimer plusieurs classes d'un élément HTML

```
list.remove('red', 'green');  
console.log(list);
```

# JavaScript

## Supprimer une classe d'un élément HTML

```
list.remove('red');  
console.log(list);
```

## Supprimer plusieurs classes d'un élément HTML

```
list.remove('red', 'green');  
console.log(list);
```

## Supprimer une classe si elle existe, sinon l'ajouter

```
list.toggle('red');  
console.log(list);
```

# JavaScript

## Autres méthodes de `classList`

- `length` : retourne le nombre de classes.
- `contains(classe)` : retourne `true` si classe appartient à `classList`, `false` sinon.
- `item(i)` : retourne la classe d'indice `i` si `i` est inférieur à la longueur de la liste, `null` sinon.

# JavaScript

## Récupérer le contenu d'un élément HTML (y compris les balises)

```
var container = document.getElementById('container')  
;  
console.log(container.innerHTML);
```

# JavaScript

## Récupérer le contenu d'un élément HTML (y compris les balises)

```
var container = document.getElementById('container')  
;  
console.log(container.innerHTML);
```

## Récupérer le contenu d'un élément HTML (sans les balises)

```
var container = document.getElementById('container')  
;  
console.log(container.textContent);
```

# JavaScript

## Modifier le contenu d'un élément HTML avec `innerHTML`

```
var container = document.getElementById('container')
;
container.innerHTML += "<p>_hello_</p>";
console.log(container.innerHTML);
```

# JavaScript

## Modifier le contenu d'un élément HTML avec `innerHTML`

```
var container = document.getElementById('container')
;
container.innerHTML += "<p>_hello_</p>";
console.log(container.innerHTML);
```

## Récupérer le contenu d'un élément HTML avec `textContent`

```
var container = document.getElementById('container')
;
container.textContent += "<p>_hello_</p>";
console.log(container.textContent);
console.log(container.innerHTML);
```

# JavaScript

## Récupérer le parent d'un élément HTML (l'objet)

```
var container = document.getElementById('container')  
;  
var parent = container.parentNode;  
console.log(parent);
```



# JavaScript

## Récupérer le parent d'un élément HTML (l'objet)

```
var container = document.getElementById('container')  
;  
var parent = container.parentNode;  
console.log(parent);
```

## Récupérer le parent d'un élément HTML (le nom)

```
var container = document.getElementById('container')  
;  
var parent = container.parentNode;  
console.log(parent.nodeName);
```

# JavaScript

## Récupérer le premier élément fils d'un élément HTML

```
var container = document.getElementById('container')  
;  
var premierFils = container.firstElementChild;  
console.log(premierFils.nodeName);
```

# JavaScript

## Récupérer le premier élément fils d'un élément HTML

```
var container = document.getElementById('container')  
;  
var premierFils = container.firstElementChild;  
console.log(premierFils.nodeName);
```

## Récupérer le dernier élément fils d'un élément HTML

```
var container = document.getElementById('container')  
;  
var dernierFils = container.lastElementChild;  
console.log(dernierFils.nodeName);
```

# JavaScript

## Récupérer le premier fils (élément, texte ou autre) d'un élément HTML

```
var container = document.getElementById('container')  
;  
var premierFils = container.firstChild;  
console.log(premierFils.nodeName);
```

# JavaScript

## Récupérer le premier fils (élément, texte ou autre) d'un élément HTML

```
var container = document.getElementById('container')
;
var premierFils = container.firstChild;
console.log(premierFils.nodeName);
```

## Récupérer le dernier fils (élément, texte ou autre) d'un élément HTML

```
var container = document.getElementById('container')
;
var dernierFils = container.lastChild;
console.log(dernierFils.nodeName);
```

# JavaScript

## Récupérer tous les enfants (élément, texte ou autre) d'un élément HTML

```
var enfants = container.childNodes;  
for(enfant of enfants)  
    console.log(enfant);
```

# JavaScript

## Récupérer tous les enfants (élément, texte ou autre) d'un élément HTML

```
var enfants = container.childNodes;  
for(enfant of enfants)  
    console.log(enfant);
```

## Récupérer tous les éléments enfants d'un élément HTML

```
var enfants = container.children;  
for(enfant of enfants)  
    console.log(enfant);
```

# JavaScript

## Enfants suivant et précédent

- `previousSibling`
- `nextSibling`
- `previousElementSibling`
- `nextElementSibling`



# JavaScript

## Enfants suivant et précédent

- `previousSibling`
- `nextSibling`
- `previousElementSibling`
- `nextElementSibling`

## Attention

À chaque appel d'une des méthode précédentes, le pointeur passe à l'élément suivant (ou précédent).

# JavaScript

## Étapes

- Créer l'élément
- Préparer ses attributs [et valeurs]
- L'ajouter au document

# JavaScript

## Créer un élément de type `p`

```
var par = document.createElement('p');
```

# JavaScript

## Créer un élément de type `p`

```
var par = document.createElement('p');
```

## Préparer ses attributs [et valeurs]

```
par.id = 'intro';  
par.setAttribute('class', 'blue');  
var text = document.createTextNode("JS_paragraph");  
par.appendChild(text);
```

# JavaScript

## Créer un élément de type `p`

```
var par = document.createElement('p');
```

## Préparer ses attributs [et valeurs]

```
par.id = 'intro';  
par.setAttribute('class', 'blue');  
var text = document.createTextNode("JS_paragraph");  
par.appendChild(text);
```

## Ajouter l'élément au DOM

```
var container = document.getElementById('container')  
;  
container.appendChild(par);
```

# JavaScript

## Créer un élément de type `p`

```
var par = document.createElement('p');
```

## Préparer ses attributs [et valeurs]

```
par.id = 'intro';  
par.setAttribute('class', 'blue');  
var text = document.createTextNode("JS_paragraph");  
par.appendChild(text);
```

## Ajouter l'élément au DOM

```
var container = document.getElementById('container')  
;  
container.appendChild(par);
```

`appendChild()` ajoute l'élément à la fin

# JavaScript

## On peut aussi utiliser

- `insertBefore(newnode, existingnode)` : pour insérer newnode **avant** existingnode

# JavaScript

## La méthode `insertAdjacentHTML(position, text)`

permet d'insérer un texte HTML (`text`), dans une position spécifiée (`position`)

- `afterbegin`,
- `afterend`,
- `beforebegin`,
- `beforeend`.



# JavaScript

## Exemple : HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title> float et clear </title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <p id=first>
      bonjour
    </p>
    <script src="file.js"></script>
  </body>
</html>
```

## Le fichier style.css

```
.skyblue {
  background-color:
    skyblue;
}
.pink {
  background-color: pink;
}
.green {
  background-color: green;
}
```

## Le script script.js

```
var p = document.getElementById('first');
p.insertAdjacentHTML("afterbegin", "<p_class=skyblue>_afterbegin_</p>");
p.insertAdjacentHTML("afterend", "<p_class=pink>_afterend_</p>");
p.insertAdjacentHTML("beforebegin", "<p_class=pink>_beforebegin_</p>");
p.insertAdjacentHTML("beforeend", "<p_class=skyblue>_beforeend_</p>");
```

# JavaScript

## Le résultat

<b>beforebegin</b>
<b>afterbegin</b>
<b>bonjour</b>
<b>beforeend</b>
<b>afterend</b>

# JavaScript

## Quelques opérations

- `replaceChild()` : pour remplacer un nœud
- `cloneNode()` : pour cloner un nœud
- `removeChild()` : pour supprimer un nœud fils
- `hasChildNodes()` : pour vérifier l'existence d'au moins un nœud enfant

# JavaScript

## Modifier la couleur

```
var container = document.getElementById('container')  
;  
container.style.color="red";
```

# JavaScript

## Modifier la couleur

```
var container = document.getElementById('container')  
;  
container.style.color="red";
```

**Pour les propriétés CSS composées de deux mots séparés par –, il faut supprimer – mettre la propriété en camelCase**

```
var container = document.getElementById('container')  
;  
container.style.backgroundColor="red";
```

# JavaScript

## Quelques évènements

- `click` **et** `dblclick` : clic et double clic
- `mouseover`, `mouseout`, `mousedown`, `mouseup` **et** `mousemove` : mouvement de la souris
- `keyup`, `keydown` **et** `keypress` : touches de clavier
- `focus` **et** `blur` : ciblage et annulation de ciblage
- `change`, `input` **et** `select` : pour les éléments d'un formulaire
- `reset` **et** `submit` : pour les formulaires
- `load` : chargement de la fenêtre

# JavaScript

## Deux façons différentes pour installer un écouteur d'évènement

- comme attribut d'une balise HTML en précédant le nom d'évènement par le préfixe `on` ayant comme valeur le nom d'une fonction JavaScript
- dans le fichier JavaScript avec la méthode `addEventListener()`

# JavaScript

## Première méthode (la page HTML)

```
<!DOCTYPE html>
<html>
  <head>
    <title>My JS Page</title>
  </head>
  <body>
    <div id = container>
      <input type=text id = nom >
      <button onclick ="direBonjour()" id=cliquer>
        cliquer    </button>
    </div>
    <script src="fichier.js"></script>
  </body>
</html>
```



# JavaScript

## Contenu du fichier.js

```
function direBonjour(){  
    var nom = document.getElementById('nom').value;  
    alert ("Bonjour_" + nom);  
}
```

# JavaScript

## Deuxième méthode (la page HTML)

```
<!DOCTYPE html>
<html>
  <head>
    <title>My JS Page</title>
  </head>
  <body>
    <div id = container>
      <input type=text id = nom >
      <button id=clicquer> cliquer    </button>
    </div>
    <script src="fichier.js"></script>
  </body>
</html>
```

# JavaScript

## Contenu du fichier.js

```
var cliquer = document.getElementById('cliquer');  
cliquer.addEventListener('click', function() {  
    var nom = document.getElementById('nom').  
        value;  
    alert ("Bonjour_" + nom);  
});
```

**Pour supprimer un événement associé à un élément HTML**

```
element.removeEventListener(event, nomFunction);
```