Table des matières

Chapitre 2 : Initiation au DOM

Partie 1 - Introduction au DOM

I. Introduction

Quand une page Web est chargée, le navigateur crée un modèle objet de document (Document Object Model) de la page. Le modèle DOM HTML est construit comme un arbre d'objets.

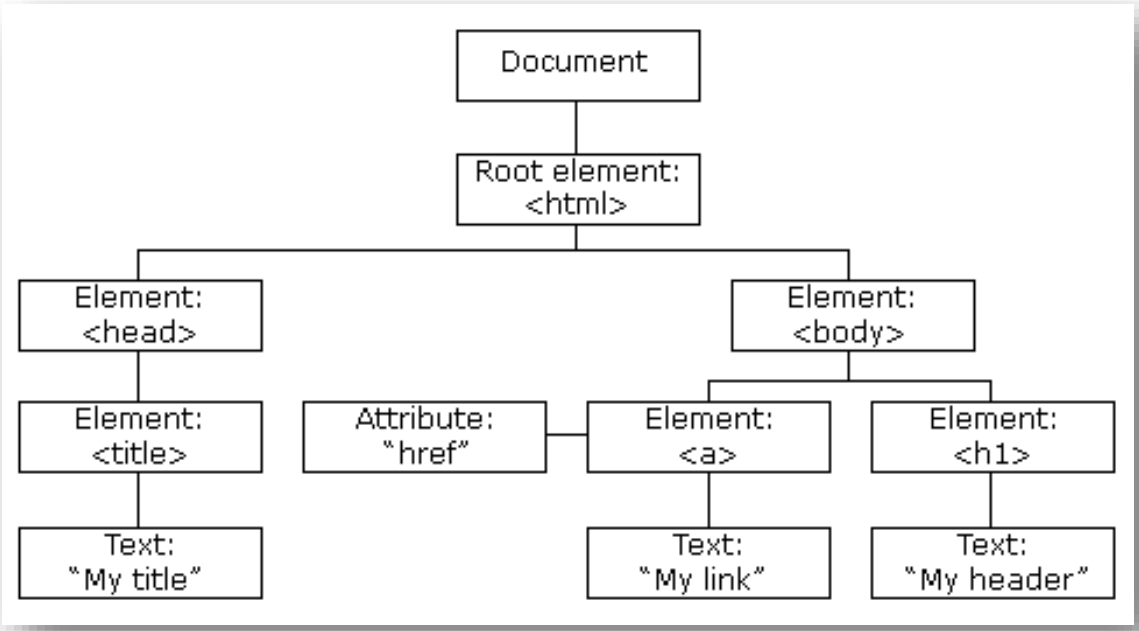


Figure 1 : Arbre d'objets DOM

Avec le modèle objet, JavaScript obtient toute la puissance dont il a besoin pour créer HTML dynamique :

- JavaScript peut changer tous les éléments HTML dans la page
- JavaScript peut changer tous les attributs HTML dans la page
- JavaScript peut changer tous les styles CSS dans la page
- JavaScript peut supprimer des éléments et attributs HTML existants
- JavaScript peut ajouter de nouveaux éléments et attributs HTML
- JavaScript peut réagir à tous les événements de HTML existants dans la page
- JavaScript peut créer de nouveaux événements HTML dans la page

II. Qu'est-ce que le DOM ?

Le DOM est un **W3C** (**W**orld **W**ide **W**eb **C**onsortium) standard. Le DOM définit une norme pour l'accès aux documents, La norme DOM W3C est séparée en trois parties distinctes :

- Noyau DOM - modèle standard pour tous les types de documents
- DOM XML - modèle standard pour les documents XML
- DOM HTML - modèle standard pour les documents HTML

1. Qu'est-ce que le DOM de HTML

Le code HTML DOM est une interface de programmation objet et un modèle standard pour HTML. Il définit :

- Les éléments HTML comme des objets
- Les propriétés de l'ensemble des éléments HTML
- Les méthodes pour accéder à tous les éléments HTML
- Les événements pour tous les éléments HTML

En d'autres termes : Le DOM HTML est un standard pour savoir comment obtenir, modifier, ajouter ou supprimer des éléments HTML

2. L'interface de programmation DOM

L'interface de programmation est les propriétés et méthodes de chaque objet.

- Une propriété est une valeur que vous pouvez obtenir ou définir (comme changer le contenu d'un élément HTML).
- Une méthode est une action que vous pouvez faire (comme ajouter ou de supprimer un élément HTML)

Exemple :

```
<html>
<body>
        <p id="intro">Hello World!</p>
        <script>
                var txt=document.getElementById("intro").innerHTML;
                document.write(txt);
        </script>
</body>
</html>
```

Dans l'exemple ci-dessus, ***getElementById*** est une méthode, tandis qu'***innerHTML*** est une propriété.

Dans l'exemple ci-dessus la méthode utilisée ***getElementById("intro")*** pour trouver l'élément dont l'id est « *intro* ».

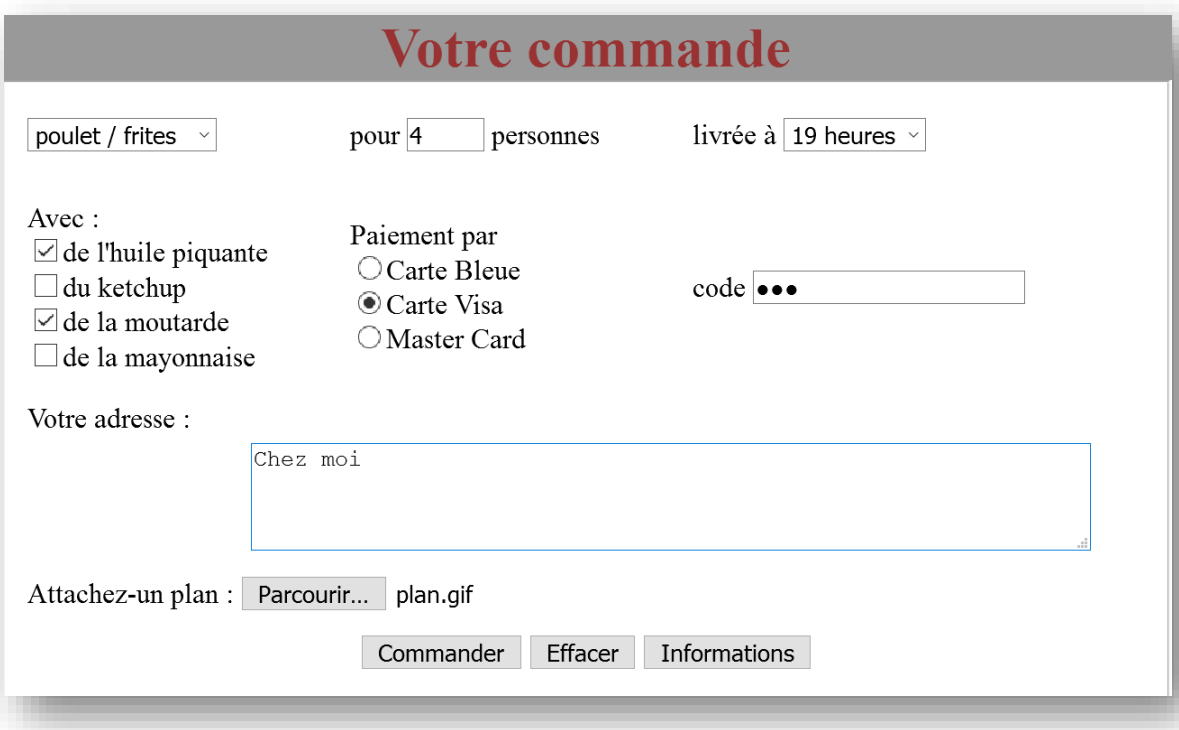
La propriété ***innerHTML*** est utile pour obtenir ou remplacer le contenu des éléments HTML.

Partie 2 - Accès aux données d'un Formulaire

I. Introduction

Au niveau de cette partie, nous montrons comment accéder aux données d'un formulaire en JavaScript.

Nous allons prendre l'exemple du formulaire ci-dessous afin d'illustrer l'accès aux champs pour lire les données ou pour les modifier. Ce formulaire est nommé "*commande*" (<**FORM name="commande" action=...>**).



II. Accès aux textes contenus dans les champs

Les éléments de formulaire concernés sont :

- Les boîtes de saisie de texte, nombre, email, recherche ... *(INPUT type="text", "number", "research", ...)*
- Les zones saisies de textes sur plusieurs lignes *(TEXTAREA)*
- Les boîtes de saisie masquées *(INPUT type="password")*
- Les boîtes de soumission de fichier *(INPUT type="file")*
- Les données masquées *(INPUT type="hidden")*

3. Lecture

document.*getElementById*(***identifiant_element***).*value*

Exemple 1 : pour la boîte de saisie de texte

```
// Code HTML dans l'élément FORM
pour <INPUT type="text" size="2" maxlength="2"  name="nb_pers" id="nbP" /> personnes
// Instruction JavaScript pour lire
nb = document.getElementById("nbP").value;
// nb prend dans l'exemple la valeur 4
```

Exemple 2 : exemple pour la soumission de fichier

```
// Code HTML dans l'élément FORM
Attachez-un plan : <INPUT type="file" name="fichier" id="fichier"/>
// Instruction JavaScript pour lire
fichier = document.getElementById("fichier").value;
// fichier prend dans l'exemple la valeur chemin_sur_disque/plan.gif
```

4. Écriture

window.document.nom_formulaire.nom_element.value = texte;

Exemple : La saisie de textes sur plusieurs lignes

```
// Code HTML dans l'élément FORM
<textarea name="adresse" rows="3" cols="40" id="adresse"></textarea>
// Instruction JavaScript pour écrire
document.getElementById('adresse').value = "Adresse non conforme";
```

5. Lire le texte par défaut

window.document.nom_formulaire.nom_element.defaultValue

Exemple : Affichage de la valeur par défaut

```
// Code HTML dans l'élément FORM
<input type="text" size="2" maxlength="2"  name="nb_pers" id="nbP" Value="5" />
// Instruction JavaScript pour écrire
nb = document.getElementById('nbP').defaultValue;
```

III. Accès aux choix cochés par le client

Les éléments de formulaire concernés sont :

- Les cases à cocher *(INPUT type="checkbox")*
- Les boutons radio *(INPUT type="radio")*

1. Recherche des choix cochés

```
document.getElementById(identifiant_element).checked
        // Retourne une valeur true/false
document.getElementById(identifiant_element).value
        // Retourne le nom donné à la boîte
```

Exemple 1 : Pour les boutons radio

```
//Code HTML dans l'élément Form
Paiement par<BR>
<INPUT type="radio" name="paiement" value="carte_bleue" />Carte Bleue<br />
<INPUT type="radio" name="paiement" value="visa" />Carte Visa<br />
<INPUT type="radio" name="paiement" value="master_card" checked />Master Card
// Instruction JavaScript pour rechercher qui est coché (un seul)
var listeR = document.getElementsByName('paiement');
for (i = 0; i < listeR.length; i++) {
   if (listeR[i].checked) {
      alert(listeR[i].value);
   }
}
// La fenêtre d'alerte donne radio = visa
```

Exemple 2 : pour les boîtes à cocher

```
// Code HTML dans l'élément FORM
Avec :<BR>
<input type="checkbox" name="assaisoner" value="huile" />de l'huile piquante<br />
<input type="checkbox" name="assaisoner" value="ketchup" />du ketchup<br />
<input type="checkbox" name="assaisoner" value="moutarde" checked />de la moutarde
<br />
<INPUT type="checkbox" name="assaisoner" value="mayonnaise" />de la mayonnaise
// Instruction JavaScript pour rechercher qui est coché (une ou plusieurs)
var listeCHK = document.getElementsByName('assaisoner');
var liste = "";
for (i = 0; i < listeCHK.length; i++) {
   if (listeCHK[i].checked) {
      liste += listeCHK[i].value + ', ';
   }
}
alert(liste) ;
 }
```

```
// La fenêtre d'alerte donne checkbox = huile, moutarde,
```

2. Cocher ou décocher un choix

document.*getElementById*(***identifiant_element***).**checked = true/false**

Exemple : pour les boîtes à cocher

```
// Instruction JavaScript qui décoche ce qui est coché et coche ce qui ne l'est pas
listeCHK = document.getElementsByName('assaisoner');
for (i = 0; i < listeCHK.length; i++) {
   if (listeCHK[i].checked) {
      listeCHK[i].checked = false;
   }else {
      listeCHK[i].checked = true;
   }
}
// Décoche huile, moutarde et coche ketchup, mayonnaise.
```

3. Obtenir le choix coché par défaut au chargement

document.*getElementById*(***identifiant_element***).**defaultChecked**

IV. Accès aux choix dans les listes de choix

L'élément de formulaire concerné est ***SELECT***

1. Lire les choix

document.*getElementById*(***identifiant_element***)**[num_element].selected**
// Retourne une valeur true/false
document.*getElementById*(***identifiant_element***)**[num_element].value**
// Retourne le nom donné au choix
document.*getElementById*(***identifiant_element***)**[num_element].length**
// Retourne le nombre de choix possibles dans la liste
document.*getElementById*(***identifiant_element***)**[num_element].text**
// Retourne le texte HTML associé

Exemple : pour une liste à choix unique (qui fonctionne pour les multiples également)

```
// Code HTML dans l'élément FORM
<select name="nourriture" id="nourriture">
        <option value="pizza">pizza</option>
        <option value="poulet_frites">poulet/frites</option>
        <option value="moules_frites">moules/frites</option>
</select>
// Instruction JavaScript
var select = document.getElementById('nourriture');
```

```
   liste = "";
   for (i=0; i<select.length; i++) {
      if (select[i].selected) {
         liste += select[i].value;
         liste += ' (Texte HTML : ' + select[i].text + ')';       }
   }
   alert(liste);
}
// La fenêtre d'alerte donne liste de choix = poulet_frites (texte HTML : poulet/frites),
```

✍✍: Pour les listes à choix unique il est possible d'obtenir directement quel est le choix sélectionné avec :

document.*getElementById(**identifiant_element**)*.**selectedIndex**

Exemple : pour une liste à choix unique

```
// Instruction JavaScript
var select = document.getElementById('nourriture');
choix = select[select.selectedIndex].value;
alert('Choix sélectionné : ' + choix);
// La fenêtre d'alerte donne choix sélectionné = poulet_frites
```

2. Modifier les choix

document.*getElementById(**identifiant_element**)*.**selected = true/false**

Dans l'exemple ci-dessous nous supposons qu'on peut choisir plusieurs plats (attribut multiple dans SELECT)

Exemple : pour une liste à choix multiples

```
// Code HTML dans l'élément FORM
<select name="nourriture" id="nourriture" multiple>
          <option value="pizza">pizza</option>
          <option value="poulet_frites">poulet/frites</option>
           <option value="moules_frites">moules/frites</option>
</select>
```

```
// Instruction JavaScript qui désélectionne ce qui était choisi
// et sélectionne ce qui ne l'était pas
var select = document.getElementById('nourriture');
for (i=0; i<select.length; i++) {
   if (select[i].selected)
      select[i].selected = false;
  else
      select[i].selected = true;
 }
```

```
// si pizza et poulet/frites étaient sélectionnés,
// alors ils sont désélectionnés et moules/frites devient sélectionné
```

V. Accès aux textes sur les boutons

Les éléments de formulaire concernés sont :

- Les boutons de soumission (*INPUT type="submit"*)
- Les boutons de soumission avec image (*INPUT type="image"*)
- Les boutons d'annulation (*INPUT type="reset"*)
- Les boutons simples (*INPUT type="button"*)

```
document.getElementById(identifiant_element).value
```

Exemple : *pour le bouton Informations.*

```
// Code HTML dans l'élément Form
<Input type="button" value="Informations" name ="binfo" id="binfo"/>
// Instruction JavaScript pour lire
texte = document.getElementById('binfo').value;
// texte contient Informations.
```

Partie 3 - Evènements dans un formulaire

I. Introduction

Au niveau de cette partie, nous allons traiter « les Événements dans les formulaires » et nous allons prendre comme exemple le formulaire de la partie 2.

Lorsque le client clique dans le formulaire, modifie des champs, coche des boutons radio..., il est possible d'exécuter du code JavaScript. Ce code sert généralement à :

- vérifier la validité des données entrées (pas de lettres dans un numéro de fax)
- donner des informations (affichage d'un message dans la barre d'état lorsqu'on clique dans un champ de formulaire)

Les événements se placent dans l'élément de formulaire comme des attributs, ils sont de la forme nom_evenement = "code JavaScript", le code JavaScript étant la plupart du temps un appel à une fonction JavaScript définie plus haut dans la page.

Exemple :

```
<INPUT type="button" value="Informations" name="binfo" onClick="Info()" >
```

II. onClick (clic sur un élément)

Éléments concernés :

- Les cases à cocher *(INPUT type="checkbox")*
- Les boutons radio *(INPUT type="radio")*
- Les boutons de soumission *(INPUT type="submit")*
- Les boutons de soumission avec image *(INPUT type="image")*
- Les boutons d'annulation *(INPUT type="reset")*
- Les boutons simples *(INPUT type="button")*

Événement onClick pour un bouton Informations

```
<HTML>
<HEAD>...</HEAD>
<SCRIPT type="text/javascript">
        function Info()
        {
            window.alert("Nos produits sont en général non périmés");
        }
</SCRIPT>
<BODY>
<FORM ...>
...
<INPUT type="button" value="Informations" name="binfo" onClick="Info()" />
...
</FORM>
</BODY>
</HTML>
```

✍ Lorsque le client clique sur le bouton Informations, une fenêtre d'alerte apparaît avec le texte : Nos produits sont en général non périmés.

Evénement onClick pour un bouton submit

```
// fonction JavaScript déclarée dans <script...>...</script>
function Confirmer()
{
   return window.confirm("Voulez-vous vraiment soumettre ?");
}
<!-- Elément de formulaire dans l'élément FORM -->
<INPUT type="submit" value="Commander" onClick="return Confirmer()" />
```

✍ Lorsque le client clique sur le bouton Commander, une fenêtre de confirmation apparaît avec le texte, s'il choisit d'annuler, alors la soumission est annulée grâce aux deux return.

Evénement onClick pour les boutons radio

```
// fonction JavaScript déclarée dans <script...>...</script>
function AffStatus(choix)
{
   switch (choix)
   {
      case 1 : window.status = "avez-vous vraiment une carte bleue ?"; break;
      case 2 : window.status = "avez-vous vraiment une carte visa ?"; break;
      case 3 : window.status = "avez-vous vraiment une master card ?"; break;
      default : ;
   }
}
<!-- Eléments de formulaire dans l'élément FORM -->
<input type="radio" name="paiement" value="carte_bleue" onClick="AffStatus(1)" >Carte Bleue
<input type="radio" name="paiement" value="visa" onClick="AffStatus(2)"> Carte Visa
<input type="radio" name="paiement" value="master_card" checked onClick="AffStatus(3)">Master Card
```

Lorsque le client clique sur le bouton radio carte bleue, le texte "avez-vous vraiment une carte bleue ?" est affiché dans la barre d'état.

III. onChange (modification dans l'élément)

Éléments concernés :

- Les boîtes de saisie de texte (*INPUT type="text"*)
- La saisie de textes sur plusieurs lignes (*TEXTAREA*)
- Les boîtes de saisie masquées (*INPUT type="password"*)
- Les boîtes de soumission de fichier (*INPUT type="file"*)
- Les listes (*SELECT*)

Evénement onChange pour un champ masqué

```
// Élément de formulaire dans l'élément FORM
<input type="password" size="6" maxlength="6" name="pwd"
        onChange="window.status=this.value" />
```

Lorsque le client a donné son code et qu'il quitte le champ, son code apparaît en clair dans la barre d'état.

IV. onFocus, onBlur (attribution, perte du focus)

Éléments concernés :

- Les boîtes de saisie de texte (*INPUT type="text"*)
- La saisie de textes sur plusieurs lignes (*TEXTAREA*)
- Les boîtes de saisie masquées (*INPUT type="password"*)
- Les boîtes de soumission de fichier (*INPUT type="file"*)
- Les listes (*SELECT*)

Evénement onFocus pour un champ masqué

```
// Élément de formulaire dans l'élément FORM
<input type="password" size="6" maxlength="6" name="pwd"
        onFocus="window.status='Entrez votre code'" />
```

Lorsque le client clique dans le champ, le message Entrez votre code apparaît dans la barre d'état.

V. onSelect (sélection dans un champ)

Éléments concernés :

- Les boîtes de saisie de texte (*INPUT type="text"*)
- La saisie de textes sur plusieurs lignes (*TEXTAREA*)
- Les boîtes de saisie masquées (*INPUT type="password"*)
- Les boîtes de soumission de fichier (*INPUT type="file"*)

Evénement onSelect pour un champ masqué

```
// Elément de formulaire dans l'élément FORM
<input type="password" size="6" maxlength="6" name="pwd"
        onSelect="window.status='champ password sélectionné'" />
```

Lorsque le client sélectionne dans le champ, le message champ password sélectionne apparaît dans la barre d'état.

VI. onReset, onSubmit

- Lorsque le formulaire est soumis en appuyant sur le bouton de soumission il est possible d'appeler un événement *onSubmit* (pour vérifier par exemple la validité des données avant d'envoyer le formulaire, pour demander la confirmation de la soumission).
- L'événement reset est appelé lorsqu'on a cliqué sur le bouton annuler.

Evénement onSubmit

```
<form name="commande" action="script.php" onSubmit="return Verif()" />
```

Lorsque le client clique sur le bouton Commander, une fonction de vérification Verif() est appelée, si elle retourne la valeur true, le formulaire est envoyé au serveur, sinon la soumission est annulée.

VII. Méthodes

nom	rôle
blur()	enlève le focus d'un champ du formulaire
click()	simule un clic de souris sur le champ du formulaire
focus()	attribue le focus à un champ du formulaire
select()	sélectionne le contenu d'un champ du formulaire

```
window.document.nom_form.nom_element.focus()
```

Il existe également les méthodes *submit()* et *reset()*

VIII. Activer ou désactiver des contrôles

Il est possible de désactiver ou d'activer des contrôles d'un formulaire pour les éléments ***INPUT***, ***BUTTON***, ***TEXTAREA*** et ***SELECT*** en réponse à un événement.
Prenons l'exemple dessiné ci-dessous qui active le champ pour entrer le code de la carte bancaire uniquement si on clique sur les boutons radio qui correspondent aux cartes et qui désactive le champ si on clique sur le bouton chèque :

Voici le code HTML qui permet d'effectuer ces actions :

```
<form name="formu1" method="" action="">
 <table width="40%">
  <tr>
    <td width="40%">Paiement par :</td>
    <td>Votre code :</td>
  </tr>
  <tr>
    <td>
   <input type="radio" name="mode" value="cb" onClick="Activer()" />Carte Bleue<br />
  <input type="radio" name="mode" value="ae" onClick="Activer()" />American Express<br>
  <input type="radio" name="mode" value="ch" onClick="Desactiver()" checked />Chèque
    </td>
    <td><input type="password" name="codeCarte" id="codeCarte" placeholder="code"
disabled /></td>
  </tr>
 </table>
</form>
```

Voici le contenu des fonctions JavaScript :

```
<script type="text/javascript">
	function Activer() {
		document.getElementById('codeCarte').disabled = false;
	}

	function Desactiver() {
		document.getElementById('codeCarte').disabled = true;
	}
</script>
```