

Architecture et Programmation de la toile (HLIN510)

Cours JavaScript (à partir de la diapo 44)

Pierre Pompidor

Différence entre internet et web

Internet (1969 aux Etats-Unis)

- ▶ Interconnexion mondiale de réseaux de différentes granularités
- ▶ Matériels et logiciels permettant la transmission de données entre deux ordinateurs situés n'importe où dans le monde

Différence entre internet et web

Internet (1969 aux Etats-Unis)

- ▶ Interconnexion mondiale de réseaux de différentes granularités
- ▶ Matériels et logiciels permettant la transmission de données entre deux ordinateurs situés n'importe où dans le monde

World Wide Web (Tim Berners-Lee en 1990 à Genève)

- ▶ Ensemble des technologies nécessaires à la recherche d'information sur Internet
- ▶ Visualisation d'informations impliquée par la navigation Internet
- ▶ Logiciels : navigateurs, serveurs web

Différence entre internet et web

Internet (1969 aux Etats-Unis)

- ▶ Interconnexion mondiale de réseaux de différentes granularités
- ▶ Matériels et logiciels permettant la transmission de données entre deux ordinateurs situés n'importe où dans le monde

World Wide Web (Tim Berners-Lee en 1990 à Genève)

- ▶ Ensemble des technologies nécessaires à la recherche d'information sur Internet
- ▶ Visualisation d'informations impliquée par la navigation Internet
- ▶ Logiciels : navigateurs, serveurs web

Protocole de communication HTTP (HyperText Transfert Protocol)

Architectures client/serveur

Les applications web ("web app") reposent sur :

- ▶ un **serveur** qui délivre des ressources (notamment des pages web)
- ▶ un **navigateur** (le **client**) qui les affiche

Architectures client/serveur

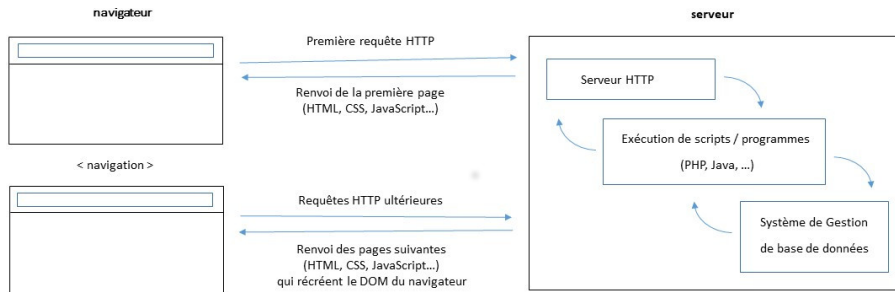
Les applications web ("web app") reposent sur :

- ▶ un **serveur** qui délivre des ressources (notamment des pages web)
- ▶ un **navigateur** (le **client**) qui les affiche

Attention au mot "serveur"

- ▶ l'ordinateur distant
- ▶ un logiciel (serveur web/HTTP) qui :
réceptionne les messages envoyés par le client
identifie et renvoie les ressources demandées

Architectures multipages



Architectures multipages (suite)

Le navigateur analyse ("parse") les documents balisés
qui lui sont envoyés (comme les pages HTML)

Architectures multipages (suite)

Le navigateur analyse ("parse") les documents balisés
qui lui sont envoyés (comme les pages HTML)

Un premier (et mauvais) exemple de fragment de code HTML :

```
<center>  
  <font color="red">  
    Un message en rouge  
  </font>  
</center>
```

Architectures multipages (suite)

Le navigateur analyse ("parse") les documents balisés
qui lui sont envoyés (comme les pages HTML)

Un premier (et mauvais) exemple de fragment de code HTML :

```
<center>  
  <font color="red">  
    Un message en rouge  
  </font>  
</center>
```

DOM (Document Object Model)

Allocation en mémoire du navigateur d'informations pour chaque élément d'une page (ici les balises `<center>` et `` et l'attribut *color*)

Architectures monopages

Optimisation des temps de navigation

Chargement des vues (les codes HTML) de l'application web avant toute navigation

Chargement ultérieur des données à exploiter

Architectures monopages

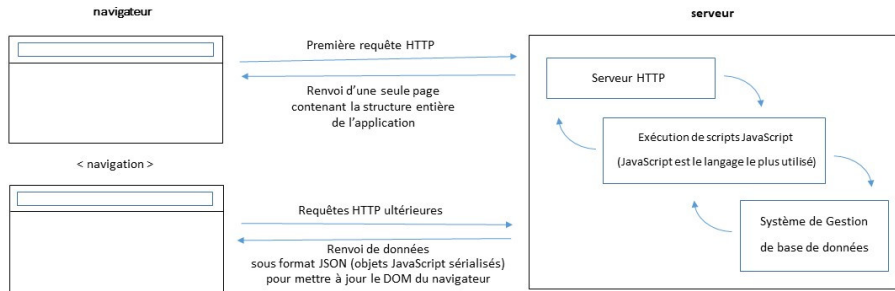
Optimisation des temps de navigation

Chargement des vues (les codes HTML) de l'application web avant toute navigation

Chargement ultérieur des données à exploiter

Premier temps de chargement plus long

Architectures monopages (suite)



HTML (HyperText Markup Language)

Caractéristiques

- ▶ Très facile d'accès
- ▶ Duos de **balises** (balise ouvrante et balise fermante)
(possibilité de balises auto-fermantes)
- ▶ Attributs précisant le comportement des balises

HTML (HyperText Markup Language)

Caractéristiques

- ▶ Très facile d'accès
- ▶ Duos de **balises** (balise ouvrante et balise fermante)
(possibilité de balises auto-fermantes)
- ▶ Attributs précisant le comportement des balises

Un second exemple :

```
<center>  
  <a href="https://fr.wikipedia.org/wiki/  
Hypertext_Markup_Language">  
    Article sur HTML de Wikipedia  
  </a>  
</center>
```

HTML : langage de programmation spécifique

Fonctionnalités très spécifiques et réduites

- ▶ Mise en forme des informations
(mais plus sur leur structuration que leur présentation)
- ▶ Création de liens hypertextes (vers d'autres pages web)
- ▶ Création de formulaires (appels paramétrés de programmes)

HTML : langage de programmation spécifique

Fonctionnalités très spécifiques et réduites

- ▶ Mise en forme des informations
(mais plus sur leur structuration que leur présentation)
- ▶ Création de liens hypertextes (vers d'autres pages web)
- ▶ Création de formulaires (appels paramétrés de programmes)

Association à des feuilles de style CSS

notamment pour la présentation graphique des informations :
emplacements, tailles, couleurs...

Structure d'une page HTML

Une page HTML est

- ▶ circonscrite par les balises `<html>` et `</html>`
- ▶ structurée en deux blocs :
 - ▶ un bloc circonscrit par les balises `<head>` et `</head>`
 - ▶ un bloc circonscrit par les balises `<body>` et `</body>`

Structure d'une page HTML

Une page HTML est

- ▶ circonscrite par les balises `<html>` et `</html>`
- ▶ structurée en deux blocs :
 - ▶ un bloc circonscrit par les balises `<head>` et `</head>`
 - ▶ un bloc circonscrit par les balises `<body>` et `</body>`

Schéma de programmation :

```
<html>  
  <head> ... </head>  
  <body> ... </body>  
</html>
```

Structure d'une page HTML (suite)

Le bloc head

contient optionnellement :

- ▶ la spécification de l'encodage des caractères
- ▶ le nom de la page dans l'onglet du navigateur
- ▶ les liens sur les fichier(s) contenant les styles CSS
- ▶ les liens sur les fichier(s) contenant les codes JavaScript

Structure d'une page HTML (suite)

Le bloc head

contient optionnellement :

- ▶ la spécification de l'encodage des caractères
- ▶ le nom de la page dans l'onglet du navigateur
- ▶ les liens sur les fichier(s) contenant les styles CSS
- ▶ les liens sur les fichier(s) contenant les codes JavaScript

Exemple de bloc head

```
<head>  
  <meta charset="UTF8" />  
  <title> CV de James Bond </title>  
</head>
```

Structure d'une page HTML (suite)

Le bloc head

contient optionnellement :

- ▶ la spécification de l'encodage des caractères
- ▶ le nom de la page dans l'onglet du navigateur
- ▶ les liens sur les fichier(s) contenant les styles CSS
- ▶ les liens sur les fichier(s) contenant les codes JavaScript

Exemple de bloc head

```
<head>  
  <meta charset="UTF8" />  
  <title> CV de James Bond </title>  
</head>
```

Le bloc body

contient les éléments à afficher dans la fenêtre du navigateur

Les balises HTML à éviter

Association de propriétés graphiques à des chaînes

- ▶ `<h1>` à `<h7>` : tailles et poids
(de la plus "grosse" à la plus "petite")
- ▶ `` avec différents attributs : polices et couleurs d'avant et d'arrière plan
- ▶ `` : **gras**
- ▶ `<i>` : *italique*

Les balises HTML à éviter

Association de propriétés graphiques à des chaînes

- ▶ `<h1>` à `<h7>` : tailles et poids
(de la plus "grosse" à la plus "petite")
- ▶ `` avec différents attributs : polices et couleurs d'avant et d'arrière plan
- ▶ `` : **gras**
- ▶ `<i>` : *italique*

Exemple :

`` James Bond ``

Les balises HTML à éviter

Association de propriétés graphiques à des chaînes

- ▶ `<h1>` à `<h7>` : tailles et poids
(de la plus "grosse" à la plus "petite")
- ▶ `` avec différents attributs : polices et couleurs d'avant et d'arrière plan
- ▶ `` : **gras**
- ▶ `<i>` : *italique*

Exemple :

`` James Bond ``

Une exception

`` : association d'un fragment de texte à un style CSS

Utilisation de balises sémantiques

Particulièrement adaptées pour la publication d'articles
mais que nous mettrons en œuvre pour la création d'un CV

Utilisation de balises sémantiques

Particulièrement adaptées pour la publication d'articles

mais que nous mettrons en œuvre pour la création d'un CV

Principales balises sémantiques :

- ▶ `<header>` : entête de page
- ▶ `<main>` : ensemble des articles
- ▶ `<article>` : un article
- ▶ `<section>` : une section d'un article
- ▶ `<footer>` : pied de page

Utilisation de balises sémantiques

Particulièrement adaptées pour la publication d'articles

mais que nous mettrons en œuvre pour la création d'un CV

Principales balises sémantiques :

- ▶ `<header>` : entête de page
- ▶ `<main>` : ensemble des articles
- ▶ `<article>` : un article
- ▶ `<section>` : une section d'un article
- ▶ `<footer>` : pied de page

Balises `<div>` plus génériques également employables

Structure d'un document utilisant les balises sémantiques

```
<html>
  <head> ... </head>
  <body>
    <header> ... </header>
    <main>
      <article>
        <section> ... </section>
        <section> ... </section>
        ...
      </article>
      <article> ... </article>
      ...
    </main>
    <footer> ... </footer>
  </body>
</html>
```

Les feuilles de styles CSS (Cascading Style Sheets)

Fichiers textuels associant des styles graphiques aux éléments d'une page HTML

Les feuilles de styles CSS (Cascading Style Sheets)

Fichiers textuels associant des styles graphiques aux éléments d'une page HTML

Un exemple de style CSS

```
body { font-size: 18pt; font-weight: bold; }
```


Les feuilles de styles CSS (Cascading Style Sheets)

Fichiers textuels associant des styles graphiques aux éléments d'une page HTML

Un exemple de style CSS

```
body { font-size: 18pt; font-weight: bold; }
```

Association assurée par la balise <link> dans le bloc head

```
<html>
  <head>
    <link rel="stylesheet"
          href="<nom_de_la_feuille_de_style>" />
  </head>
  ...
</html>
```

Les feuilles de styles CSS (suite)

Un style graphique

se décompose en **propriétés graphiques**

Un exemple de propriété graphique

```
font-size : 18pt ;
```

Les feuilles de styles CSS (suite)

Un style graphique

se décompose en **propriétés graphiques**

Un exemple de propriété graphique

```
font-size : 18pt ;
```

Les styles graphiques peuvent s'appliquer :

- ▶ sur un type de balises (par exemple toutes les balises `<p>`)
- ▶ à des balises associées à une classe (par exemple tous les articles (balises `<article>`) qui doivent être justifiés à droite)
- ▶ à une balise désignée par un identifiant unique (un "id")

Application d'un style à un type de balise

La balise <p> crée un paragraphe

```
<p> James Bond  </p>
```

Application d'un style à un type de balise

La balise <p> crée un paragraphe

```
<p> James Bond  </p>
```

Style associé :

```
p { font-size: 18pt;  
    font-weight: bold;  
    font-family: 'Courier New',  
    Courier, monospace;  
}
```

Application d'un style à des balises associées à une classe

Une classe regroupe arbitrairement des balises

Application d'un style à des balises associées à une classe

Une classe regroupe arbitrairement des balises

Dans le CV des articles sont justifiés à droite de la fenêtre

```
<article class="right">
  <p> Competences : </p>
  <ul>
    <li> Permis B (avec certificat d'aptitude ...)
    <li> Langues etrangeres : russe , ... </li>
  <ul>
</article>
```

Application d'un style à des balises associées à une classe

Une classe regroupe arbitrairement des balises

Dans le CV des articles sont justifiés à droite de la fenêtre

```
<article class="right">  
  <p> Competences : </p>  
  <ul>  
    <li> Permis B (avec certificat d'aptitude ...)  
    <li> Langues etrangeres : russe , ... </li>  
  </ul>  
</article>
```

style associé :

```
.right {text-align: right; }
```


Application d'un style à une balise désignée par un identifiant

Exemple de code HTML :

```
<p> James Bond  
    
</p>
```

Application d'un style à une balise désignée par un identifiant

Exemple de code HTML :

```
<p> James Bond  
    
</p>
```

Style associé :

```
#photo { width: 100px;  
         height: 120px;  
         float: right; }
```

Règle de priorité sur l'application des styles CSS

Plusieurs styles CSS peuvent être appliqués à une même balise

Le style le plus spécifique à une balise donnée est prioritaire

Règle de priorité sur l'application des styles CSS

Plusieurs styles CSS peuvent être appliqués à une même balise

Le style le plus spécifique à une balise donnée est prioritaire

Deux styles graphiques peuvent concerner les mêmes balises :

```
body { font-size: 14pt; }  
header p { font-size: 28pt; }
```

Règle de priorité sur l'application des styles CSS

Plusieurs styles CSS peuvent être appliqués à une même balise

Le style le plus spécifique à une balise donnée est prioritaire

Deux styles graphiques peuvent concerner les mêmes balises :

```
body { font-size: 14pt; }  
header p { font-size: 28pt; }
```

Pour notre exemple, le contenu d'une balise `<p>` affiliée à une balise `<header>` est assujetti à la taille du second style

Analyse d'une page HTML

Ouvrez un terminal

Vous êtes positionné sur votre répertoire d'accueil

Analyse d'une page HTML

Ouvrez un terminal

Vous êtes positionné sur votre répertoire d'accueil

Créez un dossier et déplacez-vous-y :

```
mkdir WEB
```

```
cd WEB
```

Analyse d'une page HTML

Ouvrez un terminal

Vous êtes positionné sur votre répertoire d'accueil

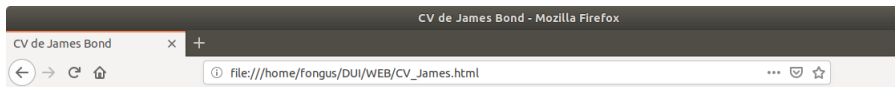
Créez un dossier et déplacez-vous-y :

```
mkdir WEB  
cd WEB
```

Copier les fichiers CV_James.html, styles_CV_James.css et photo_James.jpg :

```
https://moodle.umontpellier.fr/course/view.php?id=11006  
https://advanse.lirimm.fr/~pompidor/DIU
```


Le CV de James Bond



James Bond



Nationalité britannique

Profession : agent secret (dernier employeur : MI6)

Expérience professionnelle :

Année	Nature de la mission
1954	Neutralisation d'un dangereux amnésique (le Chiffre)
1961	Prévention d'une catastrophe nucléaire
1962	Démantèlement d'une organisation criminelle
...	...

Compétences :

- Permis B (avec certificat d'aptitude à la conduite rapide)
- Langues étrangères : russe, japonais, danois (notions)

Hobbies :

- Spécialiste mondial du vodka-martini au shaker
- Conversation mondaine en présence de public féminin

Code HTML du CV de James

```
<!doctype html>
<html>

  <head>
    <meta charset="UTF8" />
    <title> CV de James Bond </title>
    <link rel="stylesheet"
          href="styles_CV_James.css" />
  </head>
```

Code HTML du CV de James (suite)

```
<body>
  <header>
    <p> James Bond
      
    </p>
    Nationalite britannique <br/>
    Profession : agent secret (...)
  </header>
```

Code HTML du CV de James (suite)

```
<main>
```

```
<article>
```

```
<p> Experience professionnelle : </p>
```

```
<table border="1">
```

```
<tr> <th> Annee </th> <th> Nature ... </th></tr>
```

```
<tr> <td> 1954 </td> <td> ... </td> </tr>
```

```
<tr> <td> 1961 </td> <td> ... </td> </tr>
```

```
<tr> <td> 1962 </td> <td> ... </td> </tr>
```

```
<tr> <td> ... </td> <td> ... </td> </tr>
```

```
</table>
```

```
</article>
```

Code HTML du CV de James (suite)

```
<article class="right">
  <p> Competences : </p>
  <ul>
    <li> Permis B (avec certificat ...) </li>
    <li> Langues etrangeres : russe , ... </li>
  </ul>
</article>
```

```
<article>
  <p> Hobbies : </p>
  <ul>
    <li> Degustation des ... </li>
    <li> Conversation mondaine ... </li>
  </ul>
</article>
</main>
```

Et la feuille de style associée :

```
body { font-size: 14pt;  
        width: 800px;  
        margin-left: auto;  
        margin-right: auto; }
```

```
header {margin: 20px; }  
header p {font-size: 28pt;  
          color: red; }
```

```
article { margin: 10px; }
```

Code HTML du CV de James (suite)

```
p { font-size: 18pt;  
    font-weight: bold;  
    font-family: 'Courier New',  
                Courier, monospace; }
```

```
table { font-size: 12pt; }
```

```
li { list-style-position: inside; }
```

```
.right {text-align: right; }
```

```
#photo { width: 100px;  
         height: 120px;  
         float: right; }
```

Découverte des outils du navigateur

Découvrez les outils du navigateur (touche **F12**) :

- ▶ l'inspecteur DOM : allocation des éléments en mémoire du navigateur
- ▶ l'éditeur de style : visualisation et modification des styles CSS
- ▶ le réseau : vérification du chargement toutes les ressources utiles à la page
- ▶ la console : affichage des erreurs des codes JavaScript

Découverte des outils du navigateur

Découvrez les outils du navigateur (touche **F12**) :

- ▶ l'inspecteur DOM : allocation des éléments en mémoire du navigateur
- ▶ l'éditeur de style : visualisation et modification des styles CSS
- ▶ le réseau : vérification du chargement toutes les ressources utiles à la page
- ▶ la console : affichage des erreurs des codes JavaScript

Parcourez l'arborescence des éléments du DOM via l'inspecteur DOM

Découverte des outils du navigateur

Découvrez les outils du navigateur (touche **F12**) :

- ▶ l'inspecteur DOM : allocation des éléments en mémoire du navigateur
- ▶ l'éditeur de style : visualisation et modification des styles CSS
- ▶ le réseau : vérification du chargement toutes les ressources utiles à la page
- ▶ la console : affichage des erreurs des codes JavaScript

Parcourez l'arborescence des éléments du DOM via l'inspecteur DOM

Modifiez "in vivo" les styles CSS pour exprimer votre bon goût

Exercice : Création de votre CV (45 minutes)

Définissez un gabarit

Exemples sur <https://www.modeles-de-cv.com/>

Implémentez un code HTML structurant les blocs d'informations en articles, tableaux, listes...

Implémentez une feuille CSS associant aux différents éléments des styles graphiques

Liens hypertextes

```
<a href="<URL>"> ... </a>
```

```
<a href="<URL>" target="_blank"> ... </a>
```

Liens hypertextes

```
<a href="<URL>"> ... </a>  
<a href="<URL>" target="_blank"> ... </a>
```

Lien créé avec la balise `<a>` qui permet :

- ▶ soit de naviguer vers une autre page (lien externe)
- ▶ soit de se positionner à un autre endroit de la page courante (lien interne)

Liens hypertextes

```
<a href="<URL>"> ... </a>  
<a href="<URL>" target="_blank"> ... </a>
```

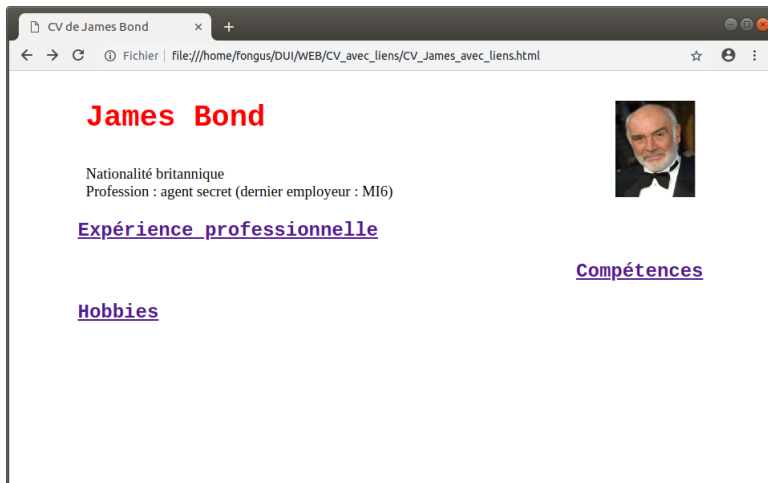
Lien créé avec la balise `<a>` qui permet :

- ▶ soit de naviguer vers une autre page (lien externe)
- ▶ soit de se positionner à un autre endroit de la page courante (lien interne)

URL en valeur de l'attribut href :

- ▶ Remplacement de la page courante
- ▶ Ouverture dans un nouvel onglet
- ▶ Ouverture dans une pop-up (non recommandée) via un autre mécanisme

Liens dans le CV de James



Liens dans le CV de James (suite)

```
<article>
  <p> <a href="CV_James_experiences.html"
      target="_blank">
    Experience professionnelle </a> </p>
</article>
```

```
<article class="right">
  <p> <a href="CV_James_compétences.html"
      target="_blank">
    Compétences </a> </p>
</article>
```

```
<article>
  <p> <a href="CV_James_hobbies.html"
      target="_blank">
    Hobbies </a> </p>
</article>
```


le fichier *CV_James_experiences.html*

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF8" />
    <title> CV de James Bond / Exp. pro </title>
    <link rel="stylesheet" href="styles_CV_James.css" />
  </head>

  <body>
    <p> Experience professionnelle : </p>
    <table border="1">
      <tr> <th> Annee </th> <th> Nature ... </th> </tr>
      <tr> <td> 1954 </td> <td> ... </td> </tr>
      ...
    </table>
  </body>
</html>
```

Formulaires : appel d'un programme avec paramètres

Utilisation de la balise `<form>`

Formulaire : appel d'un programme avec paramètres

Utilisation de la balise <form>

Schéma de programmation :

```
<form action="<URL>">  
  ... <!-- ici se placent les autres elements  
  ...      du formulaire (les controles) -->  
  <input type="submit" value="Validez" />  
</form>
```

Formulaire : appel d'un programme avec paramètres

Utilisation de la balise <form>

Schéma de programmation :

```
<form action="<URL>">  
    ... <!-- ici se placent les autres elements  
    ...      du formulaire (les controles) -->  
    <input type="submit" value="Validez" />  
</form>
```

Bouton de soumission pour émettre la requête HTTP

Formulaires : appel d'un programme avec paramètres (suite)

Envoi d'une **requête HTTP** par le navigateur

au serveur HTTP localisé sur la machine distante identifiée par la valeur de l'attribut **action**

Formulaires : appel d'un programme avec paramètres (suite)

Envoi d'une **requête HTTP** par le navigateur

au serveur HTTP localisé sur la machine distante identifiée par la valeur de l'attribut **action**

Méthodes d'envoi :

- ▶ **méthode GET** : les paramètres sont associés à l'URL
- ▶ **méthode POST** : les paramètres sont insérés dans le corps du message

Formulaires : appel d'un programme avec paramètres (suite)

Envoi d'une **requête HTTP** par le navigateur

au serveur HTTP localisé sur la machine distante identifiée par la valeur de l'attribut **action**

Méthodes d'envoi :

- ▶ **méthode GET** : les paramètres sont associés à l'URL
- ▶ **méthode POST** : les paramètres sont insérés dans le corps du message

Méthode POST plus sécuritaire :

```
<form action="<URL>" method="POST">  
  ...  
  <input type="submit" value="Validez" />  
</form>
```

Liste des balises usuelles créant des contrôles

La balise `<input>` pour des :

- ▶ zones de saisie : `<input type="text" ...></input>`
- ▶ zones de saisie masquées : `<input type="password"... />`
- ▶ cases à cocher : `<input type="checkbox" ... />`
- ▶ boutons radio : `<input type="radio" ... />`

Liste des balises usuelles créant des contrôles

La balise `<input>` pour des :

- ▶ zones de saisie : `<input type="text" ...></input>`
- ▶ zones de saisie masquées : `<input type="password"... />`
- ▶ cases à cocher : `<input type="checkbox" ... />`
- ▶ boutons radio : `<input type="radio" ... />`

La balise `<select>` : listes déroulantes

de conserve avec la balise `<option>` qui crée chaque item

```
<select name="Cocktail">  
  <option value="martini">Martini au shaker</option>  
  ...  
</select>
```

Liste des balises usuelles créant des contrôles

La balise `<input>` pour des :

- ▶ zones de saisie : `<input type="text" ...></input>`
- ▶ zones de saisie masquées : `<input type="password"... />`
- ▶ cases à cocher : `<input type="checkbox" ... />`
- ▶ boutons radio : `<input type="radio" ... />`

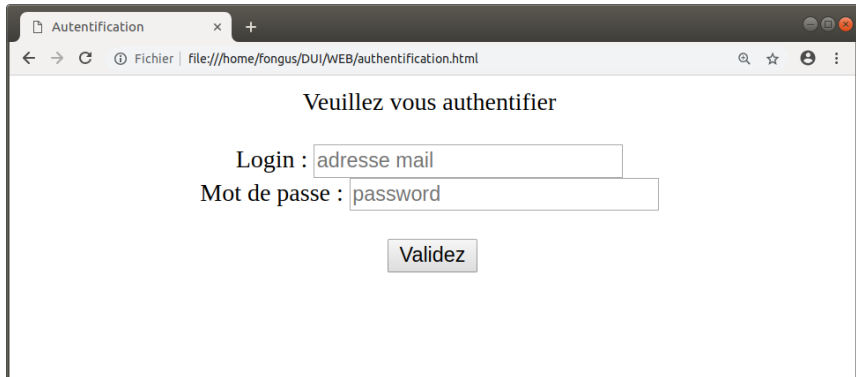
La balise `<select>` : listes déroulantes

de conserve avec la balise `<option>` qui crée chaque item

```
<select name="Cocktail">  
  <option value="martini">Martini au shaker</option>  
  ...  
</select>
```

La balise `<textarea>` : zones de saisie de textes multi-lignes

Formulaire d'authentification typique



The image shows a web browser window with a single tab titled "Autentification". The address bar displays the file path "file:///home/fongus/DUI/WEB/authentication.html". The page content is centered and features the heading "Veuillez vous authentifier". Below this, there are two input fields: the first is labeled "Login :" and contains the text "adresse mail"; the second is labeled "Mot de passe :" and contains the text "password". A "Validez" button is positioned below the password field. The browser's interface includes standard navigation buttons (back, forward, refresh) and a menu icon on the left, and search, star, and user profile icons on the right.

Autentification

file:///home/fongus/DUI/WEB/authentication.html

Veuillez vous authentifier

Login :

Mot de passe :

Validez

Formulaire d'authentification typique (suite)

```
Veuillez vous authentifier <br/><br/>
<form action="<URL>" method="POST">
    Login : <input type="text"
              name="login"
              placeholder="votre_nom"
              required />

    <br/>
    Mot de passe : <input type="password"
                        name="pass"
                        placeholder="password"
                        required />

    <br/> <br/>
    <input type="submit" value="Validez">
</form>
```

Formulaire d'authentification typique (suite)

Dans les exemples précédents

remarquez bien l'attribut *name* qui accompagne les balises

Formulaire d'authentification typique (suite)

Dans les exemples précédents

remarquez bien l'attribut *name* qui accompagne les balises

Exemple de formulaire d'authentification

- ▶ l'URL
<https://monsite.portaildesloosers.fr/identification.php>
- ▶ la méthode est *GET*
- ▶ le login a pour valeur *pompidor*
- ▶ le password a pour valeur *2fast4U*

Formulaire d'authentification typique (suite)

Dans les exemples précédents

remarquez bien l'attribut *name* qui accompagne les balises

Exemple de formulaire d'authentification

- ▶ l'URL
https://monsite.portaildesloosers.fr/identification.php
- ▶ la méthode est *GET*
- ▶ le login a pour valeur *pompidor*
- ▶ le password a pour valeur *2fast4U*

URL complète créée lors de la soumission du formulaire :

```
https://monsite.portaildesloosers.fr/  
identification.php?login=pompidor&password=2fast4U
```

Modification du CV personnel par l'insertion de liens (30 minutes)

Le but est de pouvoir ouvrir quelques rubriques de votre CV dans d'autres onglets du navigateur :

- ▶ créez de nouvelles pages HTML contenant les informations des sous-rubriques
- ▶ créez dans le fichier principal les liens désignant ces nouvelles pages

Création d'un formulaire (15 minutes)

Créez un formulaire qui :

- ▶ invoque lors de sa soumission un programme nommé *virtuel.php* situé sur le serveur *serveur.inconnu.fr* ;
(par la méthode GET)
- ▶ affiche horizontalement deux boutons radio (de nom *preparation*) permettant de choisir soit "*shaker*" ou "*cuillère*" ;
- ▶ affiche une liste déroulante (de nom *decoration*) permettant de choisir soit "*parasol*" ou "*belle tranche d'orange*".

Soumettez le formulaire

et regardez les paramètres (et leur encodage) qui se dévoilent dans la barre d'URL du navigateur
(et évidemment le message d'erreur qui apparaît)

Aperçu du langage JavaScript

Langage de programmation créé en 1995 :

- ▶ Scripts exécutés par un navigateur pour manipuler les données du **DOM** (Document Object Model)
- ▶ Accès asynchrone à des données fournies par le serveur
- ▶ Environnement serveur *Node.js* très populaire

Aperçu du langage JavaScript

Langage de programmation créé en 1995 :

- ▶ Scripts exécutés par un navigateur pour manipuler les données du **DOM** (Document Object Model)
- ▶ Accès asynchrone à des données fournies par le serveur
- ▶ Environnement serveur *Node.js* très populaire

Confluence de deux paradigmes de programmation :

Paradigme de la **programmation fonctionnelle**

Paradigme de la **programmation par objets**

Où coder du code javascript et comment le déboguer ?

Codes JavaScript dans des fichiers d'extension **.js** :

Liés aux codes HTML dans le bloc head
via la balise `<script>`

Où coder du code javascript et comment le déboguer ?

Codes JavaScript dans des fichiers d'extension **.js** :

Liés aux codes HTML dans le bloc head
via la balise `<script>`

Exemple du contenu d'un fichier JavaScript nommé *bonjour.js* :

```
console.log("Bonjour !");
```

Où coder du code javascript et comment le déboguer ?

Codes JavaScript dans des fichiers d'extension **.js** :

Liés aux codes HTML dans le bloc head
via la balise `<script>`

Exemple du contenu d'un fichier JavaScript nommé *bonjour.js* :

```
console.log("Bonjour_!");
```

Les messages générés par la méthode *log()* de l'objet *console* sont affichés dans la console (outils du navigateur)

Page HTML *test.html* mettant en œuvre ce script :

Lien sur le fichier JavaScript :

```
<html>
  <head>
    <script src="bonjour.js"
            type="text/javascript"
            language="javascript">
    </script>
  </head>
  <body> JavaScript vous dit bonjour </body>
</html>
```

Page HTML *test.html* mettant en œuvre ce script :

Lien sur le fichier JavaScript :

```
<html>
  <head>
    <script src="bonjour.js"
           type="text/javascript"
           language="javascript">
    </script>
  </head>
  <body> JavaScript vous dit bonjour </body>
</html>
```

Surtout n'utilisez jamais une balise `<script>` auto-fermante

Mise en place d'actions suite aux actions de l'internaute

Association de **gestionnaires d'événements** aux balises/éléments HTML

- ▶ onclick : clic sur un élément
- ▶ onmouseover : entrée du pointeur de la souris sur un élément
- ▶ onmouseout : sortie du pointeur de la souris d'un élément
- ▶ onselect : sélection d'un item d'un contrôle
(par exemple un item de liste déroulante)
- ▶ onload : chargement de la page dans le navigateur
- ▶ onunload : départ de la page

Appels des fonctions JavaScript pour zoomer et dézoomer la photo de James

```
<!doctype html>
<html>
  <head>
    <script src="CV_James.js"></script>
  </head>

  <body>
    <header>
      <p> James Bond
        
      </p>
    </header>
    ...
  </body>
```

Fonctions JavaScript pour zoomer et dézoomer la photo

```
var largeurImage;  
var hauteurImage;  
  
function zoom(image) {  
    largeurImage = image.style.width;  
    hauteurImage = image.style.height;  
    image.style.width = "auto";  
    image.style.height = "auto";  
}  
  
function dezoom(image) {  
    image.style.width = largeurImage;  
    image.style.height = hauteurImage;  
}
```

Commentaires sur les fonctions JavaScript

La fonction *zoom()* est invoquée par le gestionnaire d'événements *onmouseover*

Une fonction JavaScript est déclarée par le mot réservé *function*
Les instructions assujetties à la fonction sont circonscrites par ses accolades

Commentaires sur les fonctions JavaScript

La fonction *zoom()* est invoquée par le gestionnaire d'événements *onmouseover*

Une fonction JavaScript est déclarée par le mot réservé *function*
Les instructions assujetties à la fonction sont circonscrites par ses accolades

Le paramètre *image* contient l'adresse en mémoire d'un objet

Un objet implémente une liste de duos clefs/valeurs
Ici l'objet contient les différents attributs (ou propriétés) de l'image

Commentaires sur les fonctions JavaScript

La fonction *zoom()* est invoquée par le gestionnaire d'événements *onmouseover*

Une fonction JavaScript est déclarée par le mot réservé *function*
Les instructions assujetties à la fonction sont circonscrites par ses accolades

Le paramètre *image* contient l'adresse en mémoire d'un objet

Un objet implémente une liste de duos clefs/valeurs
Ici l'objet contient les différents attributs (ou propriétés) de l'image

Contrairement aux autres langages objets majeurs

la programmation par objets en JavaScript ne nécessite pas la définition de classes
même s'il est possible d'en utiliser (normalisation ECMA6 et extensions)

Rôle de JavaScript comme accesseur du DOM

Le **DOM** (Document Object Model)

représente toutes les données mises en mémoire du navigateur
(l'allocation d'une balise HTML est appelé un **élément**)

Rôle de JavaScript comme accesseur du DOM

Le **DOM** (Document Object Model)

représente toutes les données mises en mémoire du navigateur
(l'allocation d'une balise HTML est appelé un **élément**)

L'accès au DOM est essentiel

pour modifier le contenu d'une page sans passer par le serveur

Rôle de JavaScript comme accesseur du DOM

Le **DOM** (Document Object Model)

représente toutes les données mises en mémoire du navigateur
(l'allocation d'une balise HTML est appelé un **élément**)

L'accès au DOM est essentiel

pour modifier le contenu d'une page sans passer par le serveur

Des fonctions spécifiques pour sélectionner un ou plusieurs éléments du DOM :

- ▶ via le nom d'un type de balises
(par exemple toutes les balises ``)
- ▶ via un identifiant (exprimé par un attribut *id*)

Sélection d'éléments dans le DOM

Pour sélectionner des éléments du DOM, il y a deux possibilités :

- ▶ utiliser les fonctions JavaScript de l'API DOM "de base"
- ▶ utiliser une bibliothèque JavaScript de plus haut niveau

Sélection d'éléments dans le DOM

Pour sélectionner des éléments du DOM, il y a deux possibilités :

- ▶ utiliser les fonctions JavaScript de l'API DOM "de base"
- ▶ utiliser une bibliothèque JavaScript de plus haut niveau

Exemple avec l'API DOM :

```
console.log(document.getElementById("photo").src);
```

Sélection d'éléments dans le DOM

Pour sélectionner des éléments du DOM, il y a deux possibilités :

- ▶ utiliser les fonctions JavaScript de l'API DOM "de base"
- ▶ utiliser une bibliothèque JavaScript de plus haut niveau

Exemple avec l'API DOM :

```
console.log(document.getElementById("photo").src);
```

Exemple avec la bibliothèque JQuery :

```
console.log($("#photo").attr("src"));
```

Sélection d'éléments dans le DOM - suite

Une introspection avec l'API DOM :

```
function analyseDOM(noeud) {  
  switch (noeud.nodeType) {  
    case 1 : ... // noeud de type element  
    case 3 : ... // noeud de type text  
    ...  
  }  
  for (let n=0; n < noeud.attributes.length; n++) {  
    ...  
  }  
  for (let n=0; n < noeud.childNodes.length; n++) {  
    analyseDOM(noeud.childNodes[n])  
  }  
}
```

Sélection d'éléments dans le DOM - suite

Une introspection avec l'API DOM :

```
function analyseDOM(noeud) {  
  switch (noeud.nodeType) {  
    case 1 : ... // noeud de type element  
    case 3 : ... // noeud de type text  
    ...  
  }  
  for (let n=0; n < noeud.attributes.length; n++) {  
    ...  
  }  
  for (let n=0; n < noeud.childNodes.length; n++) {  
    analyseDOM(noeud.childNodes[n])  
  }  
}
```

noeud.nodeType noeud.nodeName noeud.nodeValue

JavaScript / syntaxe de base : déclaration des variables

Variables **typées dynamiquement** et déclarées par :

Le mot réservé **var** :

la portée de la variable est (presque) globale

JavaScript / syntaxe de base : déclaration des variables

Variables **typées dynamiquement** et déclarées par :

Le mot réservé **var** :

la portée de la variable est (presque) globale

Le mot réservé **let** :

la portée de la variable est alors le bloc d'instructions

JavaScript / syntaxe de base : déclaration des variables

Variables **typées dynamiquement** et déclarées par :

Le mot réservé **var** :

la portée de la variable est (presque) globale

Le mot réservé **let** :

la portée de la variable est alors le bloc d'instructions

Le mot réservé **const** :

la variable n'est accessible qu'en lecture

Syntaxe de base : déclaration des variables

Javascript type (en interne) les variables suivant six types primitifs :

- ▶ un type **booléen** : **true** et **false** ;
- ▶ un type **nul** : **null**
- ▶ un type **indéfini** : **undefined**
- ▶ un type pour les **nombres** : **number**
- ▶ un type pour les **chaînes de caractères** : **string**
- ▶ un type pour les **symboles**

Syntaxe de base : déclaration des variables

Javascript type (en interne) les variables suivant six types primitifs :

- ▶ un type **booléen** : **true** et **false** ;
- ▶ un type **nul** : **null**
- ▶ un type **indéfini** : **undefined**
- ▶ un type pour les **nombres** : **number**
- ▶ un type pour les **chaînes de caractères** : **string**
- ▶ un type pour les **symboles**

Type **Object** dans les autres cas de figures

Syntaxe de base : déclaration des variables

Voici quelques exemples de création de variables scalaires :

```
var bizarre; // sa valeur est : undefined  
var i = 0;  
let bool = true;
```

Syntaxe de base : déclaration des variables

Voici quelques exemples de création de variables scalaires :

```
var bizarre; // sa valeur est : undefined  
var i = 0;  
let bool = true;
```

Exemple de création d'une chaîne de caractères :

```
let nom = "Bond";
```

Syntaxe de base : les structures de données usuelles

Deux structures de données sont omniprésentes

Syntaxe de base : les structures de données usuelles

Deux structures de données sont omniprésentes

Les **objets** :

Regroupement de données et des traitements qui y sont appliqués :
l'utilisation d'objets permet une meilleure structuration du code

Syntaxe de base : les structures de données usuelles

Deux structures de données sont omniprésentes

Les **objets** :

Regroupement de données et des traitements qui y sont appliqués :
l'utilisation d'objets permet une meilleure structuration du code

Les **listes** :

- ▶ qui correspondent à des tableaux dynamiques
- ▶ qui sont par ailleurs des objets

JavaScript / syntaxe de base : les listes

Création d'une liste :

soit en utilisant la fonction constructrice *Array()*
soit directement en employant les crochets (qui ne sont que du sucre syntaxique).

JavaScript / syntaxe de base : les listes

Création d'une liste :

soit en utilisant la fonction constructrice *Array()*
soit directement en employant les crochets (qui ne sont que du sucre syntaxique).

Exemple de création d'une liste :

```
let maListe = new Array(1, 2, 3, "partez");  
let maListe = [1, 2, 3, "partez"];
```

JavaScript / syntaxe de base : les listes

Création d'une liste :

soit en utilisant la fonction constructrice *Array()*
soit directement en employant les crochets (qui ne sont que du sucre syntaxique).

Exemple de création d'une liste :

```
let maListe = new Array(1, 2, 3, "partez");  
let maListe = [1, 2, 3, "partez"];
```

Nombre d'éléments d'une liste :

```
console.log(maListe.length);
```

JavaScript / syntaxe de base : les objets

Les objets ne sont pas originellement des instances de classes
contrairement à de nombreux autres langages de programmation

JavaScript / syntaxe de base : les objets

Les objets ne sont pas originellement des instances de classes
contrairement à de nombreux autres langages de programmation

Formatage de ces objets lorsque ceux-ci sont sérialisés (c'est à dire sauvegardés sous un format textuel)

appelé **JSON** (JavaScript Object Notation)

JavaScript / syntaxe de base : les objets

Les objets ne sont pas originellement des instances de classes contrairement à de nombreux autres langages de programmation

Formatage de ces objets lorsque ceux-ci sont sérialisés (c'est à dire sauvegardés sous un format textuel)

appelé **JSON** (JavaScript Object Notation)

Gestion des classes en JavaScript :

- ▶ via la norme ECMAScript 6
- ▶ des extensions "de plus haut niveau" comme **TypeScript**

Syntaxe de base : les objets (suite)

Exemple de création d'un objet littéral :

```
let monObjet =  
  {"nom": "Bond", "prenom": "James",  
   "acteurs": ["Sean_Connery", "George_Lazenby",  
               "Roger_Moore", "Timothy_Dalton",  
               "Pierce_Brosnan", "Daniel_Craig"],  
   "films": {"1962" : "James_Bond_007_contre...",  
             "1963" : "Bons_baisers_de_Russie",  
             "1964" : "Goldfinger"}}  
};
```

Syntaxe de base : les objets (suite)

Exemple de création d'un objet littéral :

```
let monObjet =  
  {"nom": "Bond", "prenom": "James",  
   "acteurs": ["Sean_Connery", "George_Lazenby",  
               "Roger_Moore", "Timothy_Dalton",  
               "Pierce_Brosnan", "Daniel_Craig"],  
   "films": {"1962" : "James_Bond_007_contre...",  
             "1963" : "Bons_baisers_de_Russie",  
             "1964" : "Goldfinger"}}  
};
```

Quelques exemples de manipulation de cet objet :

```
console.log(monObjet.nom);           // Bond  
console.log(monObjet.acteurs[0]);    // Sean Connery  
console.log(monObjet.films[1964]);    // Goldfinger
```


Les objets instanciés par une fonction constructrice

```
function personne(nom, prenom) {  
  this.nom = nom;  
  this.prenom = prenom;  
  this.cv = function() {  
    console.log("Je_suis_" + this.nom  
               + "_" + this.prenom);  
  };  
}
```

```
let bond = new personne("Bond", "James");  
bond.cv(); // Je suis Bond James
```

Les objets instanciés par une fonction constructrice

```
function personne(nom, prenom) {  
  this.nom = nom;  
  this.prenom = prenom;  
  this.cv = function() {  
    console.log("Je_suis_" + this.nom  
               + "_" + this.prenom);  
  };  
}
```

```
let bond = new personne("Bond", "James");  
bond.cv(); // Je suis Bond James
```

this est la référence de l'objet en construction

Cette variable contient l'adresse mémoire de l'objet

Les blocs d'instructions et la structure conditionnelle

Blocs d'instructions circonscrits par des **accolades**

Création de blocs d'instructions par les structures conditionnelles et itératives

Les blocs d'instructions et la structure conditionnelle

Blocs d'instructions circonscrits par des accolades

Création de blocs d'instructions par les structures conditionnelles et itératives

La structure conditionnelle présente le schéma suivant :

```
if ( <expression conditionnelle> ) {  
    ...  
}  
else { // ce bloc est facultatif  
    ...  
}
```

Les structures itératives

Plusieurs structures itératives :

- ▶ la structure **while** (...) ...
- ▶ la structure **for** (...) **in** ...)
- ▶ la structure **for** (...) **of** ...)
- ▶ la méthode **forEach()** s'appliquant aux listes

Les structures itératives

Plusieurs structures itératives :

- ▶ la structure **while** (...) ...
- ▶ la structure **for** (...) **in** ...)
- ▶ la structure **for** (...) **of** ...)
- ▶ la méthode **forEach()** s'appliquant aux listes

Structure for (... in ...)

```
let voitures = ["Aston_Martin", "Ford_Mustang", ...]  
for (let i in voitures) {  
    console.log(i + "_:" + voitures[i]); }  
}
```

Les structures itératives

Plusieurs structures itératives :

- ▶ la structure **while** (...) ...
- ▶ la structure **for** (...) **in** ...)
- ▶ la structure **for** (...) **of** ...)
- ▶ la méthode **forEach()** s'appliquant aux listes

Structure for (... in ...)

```
let voitures = ["Aston_Martin", "Ford_Mustang", ...]  
for (let i in voitures) {  
    console.log(i + "_:" + voitures[i]); }
```

Structure for (... of ...)

```
for (let v of voitures) { console.log(v); }
```

AJAX (Asynchronous Javascript And XML)

Récupération (du serveur) de **données** et non de code HTML

- ▶ Fonctionnalité maintenant ancienne de JavaScript
- ▶ Mise à jour une page sans modifier sa morphologie

AJAX (Asynchronous Javascript And XML)

Récupération (du serveur) de **données** et non de code HTML

- ▶ Fonctionnalité maintenant ancienne de JavaScript
- ▶ Mise à jour une page sans modifier sa morphologie

Formatage des données

- ▶ Initialement en **XML** (Extensible Mark-up Language)
- ▶ Actuellement en **JSON** (JavaScript Objet Notation) :
sérialisation d'objets JavaScript

AJAX (Asynchronous Javascript And XML)

Récupération (du serveur) de **données** et non de code HTML

- ▶ Fonctionnalité maintenant ancienne de JavaScript
- ▶ Mise à jour une page sans modifier sa morphologie

Formatage des données

- ▶ Initialement en **XML** (Extensible Mark-up Language)
- ▶ Actuellement en **JSON** (JavaScript Objet Notation) :
sérialisation d'objets JavaScript

Utilisation de la bibliothèque JavaScript JQuery

AJAX (Asynchronous Javascript And XML)

Récupération (du serveur) de **données** et non de code HTML

- ▶ Fonctionnalité maintenant ancienne de JavaScript
- ▶ Mise à jour une page sans modifier sa morphologie

Formatage des données

- ▶ Initialement en **XML** (Extensible Mark-up Language)
- ▶ Actuellement en **JSON** (JavaScript Object Notation) :
sérialisation d'objets JavaScript

Utilisation de la bibliothèque JavaScript JQuery

Utilisation préférentielle de Firefox pour exécuter du code
JavaScript AJAX

AJAX - Fichier de données JSON

Sérialisation d'objets JavaScript

Les données sont spécifiées dans un fichier texte

AJAX - Fichier de données JSON

Sérialisation d'objets JavaScript

Les données sont spécifiées dans un fichier texte

Pour les formater, la structure la plus usuelle est la liste d'objets :

```
[{  
  "<nom_de_propriete>" : <valeur de propriete>,  
  "<nom_de_propriete>" : <valeur de propriete>,  
  ...  
},  
...  
]
```

Fonction JQuery d'importation des données (sur le lien)

- ▶ importe un fichier JSON qui contient une liste d'objets ;
- ▶ parcourt la liste pour accéder à chaque objet ;
inclut la valeur d'une propriété de l'objet courant pour
l'intégrer dans une chaîne de caractères HTML ;
- ▶ attache cette chaîne de caractère à un élément du DOM.

Fonction JQuery d'importation des données (sur le lien)

- ▶ importe un fichier JSON qui contient une liste d'objets ;
- ▶ parcourt la liste pour accéder à chaque objet ;
inclut la valeur d'une propriété de l'objet courant pour
l'intégrer dans une chaîne de caractères HTML :
- ▶ attache cette chaîne de caractère à un élément du DOM.

```
function afficherAlcools() {  
    $.getJSON("<nom_du_fichier>", function(data) {  
        let html = "";  
        $.each(data, function(index, objet) {  
            html += "<creation_d'un_fragment_HTML>";  
        });  
        $("<point_d'attachement>").append(html);  
    });  
}
```

Code JQuery d'importation des données et leur encapsulation dans du HTML

\$ désigne l'objet de plus niveau de la bibliothèque JQuery
à cet objet sont attachées des fonctions (appelées méthodes)

Code JQuery d'importation des données et leur encapsulation dans du HTML

\$ désigne l'objet de plus niveau de la bibliothèque JQuery
à cet objet sont attachées des fonctions (appelées méthodes)

`$.getJSON(...)`

Méthode qui permet de télécharger un fichier JSON et d'allouer en mémoire la liste en paramètre de la fonction de rappel (*data*)

Code JQuery d'importation des données et leur encapsulation dans du HTML

\$ désigne l'objet de plus niveau de la bibliothèque JQuery
à cet objet sont attachées des fonctions (appelées méthodes)

`$.getJSON(...)`

Méthode qui permet de télécharger un fichier JSON et d'allouer en mémoire la liste en paramètre de la fonction de rappel (*data*)

`$.each(...)`

Méthode qui permet de parcourir une liste et en extraire chaque objet (variable *objet*) (*index* indice chaque objet à partir de 0)

Code JQuery d'importation des données et leur encapsulation dans du HTML

\$ désigne l'objet de plus niveau de la bibliothèque JQuery
à cet objet sont attachées des fonctions (appelées méthodes)

`$.getJSON(...)`

Méthode qui permet de télécharger un fichier JSON et d'allouer en mémoire la liste en paramètre de la fonction de rappel (*data*)

`$.each(...)`

Méthode qui permet de parcourir une liste et en extraire chaque objet (variable *objet*) (*index* indice chaque objet à partir de 0)

JQuery fait usage de fonctions de rappel qui délivrent les données mises à disposition

fonctions anonymes (lambda fonctions) passées en paramètres

Focus sur les fonctions de rappel (suite)

Création d'une liste

```
let liste = [1, 2, 3, "partez_!"];
```

Focus sur les fonctions de rappel (suite)

Création d'une liste

```
let liste = [1, 2, 3, "partez_!"];
```

Création d'une fonction myForEach() :

```
let myForEach = function(liste, callback) {  
    for (let i in liste) callback(i, liste[i]);  
}
```

Focus sur les fonctions de rappel (suite)

Création d'une liste

```
let liste = [1, 2, 3, "partez_!"];
```

Création d'une fonction myForEach() :

```
let myForEach = function(liste, callback) {  
    for (let i in liste) callback(i, liste[i]);  
}
```

Appel de la fonction myForEach() :

```
myForEach([1, 2, 3, "partez_!"],  
    function(i, element) {  
        console.log(i, ":", element);  
    });
```

Focus sur les fonctions de rappel (suite)

Création d'une liste

```
let liste = [1, 2, 3, "partez_!"];  
let liste = new Array(1, 2, 3, "partez_!");
```

Focus sur les fonctions de rappel (suite)

Création d'une liste

```
let liste = [1, 2, 3, "partez_!"];  
let liste = new Array(1, 2, 3, "partez_!");
```

Création d'une fonction myForEach() :

```
Array.prototype.myForEach = function(callback) {  
    for (let i in this) callback(i, this[i]);  
}
```


Focus sur les fonctions de rappel (suite)

Création d'une liste

```
let liste = [1, 2, 3, "partez_!"];  
let liste = new Array(1, 2, 3, "partez_!");
```

Création d'une fonction myForEach() :

```
Array.prototype.myForEach = function(callback) {  
    for (let i in this) callback(i, this[i]);  
}
```

Appel de la méthode myForEach() :

```
liste.myForEach(function(i, element) {  
    console.log(i, ":", element);  
});
```

Retour sur les objets : l'héritage

L'objet prototype est "l'interface publique" d'un objet

```
function creationObjet(v1) { this.p1 = v1; }  
objet1 = new creationObjet("v1");  
objet1.prototype = {"p2": "v2"};  
  
objet2 = {"p3": "v3",  
          __proto__: objet1.prototype};  
  
for (let p in objet2) {  
    console.log(p+" : "+objet2[p]);  
}
```

AJAX - Chargement de données complémentaires au CV

Modifions le code HTML du CV de James pour :

- ▶ insérer un lien hypertexte dans la rubrique "hobbies" :
appel de la fonction JavaScript d'importation des données
- ▶ créer une balise `` identifiée par un identifiant :
accroche du code HTML formé à partir des données importées
- ▶ lien sur la bibliothèque JQuery dans le head de la page HTML :
bibliothèque téléchargée du site JQuery :
<https://code.jquery.com/>.

AJAX - Chargement de données complémentaires au CV

```
<head>
  <script src="jquery - 3.5.1.min.js"></script>
  <script src="CV_James_AJAX.js"></script>
</head>
<body>
  ...
  <article>
    <p> Hobbies : </p>
    <ul>
      <li> Degustation de
        <a onclick="afficherAlcools()"
          href="#">nombreux alcools</a>
        <ul id="listeAlcools"></ul>
      </li>
      <li> Conversation mondaine ... </li>
    <ul>
  </article>
```

AJAX - Fichier JSON *alcools.json*

```
[{ "nom": "Vodka_Martini",  
  "amateurs": [ "Sean_Connery", "Roger_Moore", ,  
                "Pierce_Brosnan", "Daniel_Craig" ]  
},  
{ "nom": "Vesper_Martini",  
  "amateurs": [ "Daniel_Craig" ]  
},  
{ "nom": "Rum_Collins",  
  "amateurs": [ "Sean_Connery" ]  
},  
{ "nom": "Mint_Julep",  
  "amateurs": [ "Sean_Connery", "George_Lazenby" ]  
}  
]
```

Liste (encadrée par des crochets) contenant des objets
(encadrés par des accolades)

AJAX : intégration des données dans le DOM

Exemple de fonction JavaScript utilisant du code JQuery :

```
function afficherAlcools() {  
    $.getJSON("alcools.json", function(data) {  
        let html = "";  
        $.each(data, function(index, objet) {  
            html += "<li>" + objet.nom + "</li>";  
        });  
        $("#listeAlcools").append(html);  
    });  
}
```

AJAX : intégration des données dans le DOM

Exemple de fonction JavaScript utilisant du code JQuery :

```
function afficherAlcools() {  
    $.getJSON("alcools.json", function(data) {  
        let html = "";  
        $.each(data, function(index, objet) {  
            html += "<li>" + objet.nom + "</li>";  
        });  
        $("#listeAlcools").append(html);  
    });  
}
```

La chaîne de caractères *html* de valeur finale :

```
<li>Vodka Martini</li><li>Vesper Martini</li>...
```

Autre version plus simple...

... pour éviter la méthode each()

```
function afficherAlcools() {  
    $.getJSON("alcools.json", function(data) {  
        let html = "";  
        for (let objet of data) {  
            html += "<li>" + objet.nom + "</li>";  
        }  
        $("#listeAlcools").append(html);  
    });  
}
```


jQuery : sélection d'éléments en utilisant la syntaxe CSS

<code>element (balise)</code>	<code>\$("td")</code>
<code>#id</code>	<code>\$("#maDiv")</code>
<code>.class</code>	<code>\$(".highlight")</code>
<code>parent>enfant</code>	<code>\$("td>table")</code>
<code>*</code>	<code>\$("td>*")</code>
<code>:first</code>	<code>\$("a:first")</code>
<code>:last</code>	<code>\$("div:last")</code>
<code>:eq()</code>	<code>\$("div:eq(1)")</code>
<code>:contains</code>	<code>\$("a:contains('important')")</code>
<code>element[attribut=valeur]</code>	<code>\$("[name='info']")</code> <code>\$("input[type=text]")</code>
<code>element[attribut!=valeur]</code>	<code>\$("[id!='param']")</code>
<code>element[attribut^=valeur]</code>	<code>\$("[id^='param']")</code>
<code>element[attribut*=valeur]</code>	<code>\$("[id*='param']")</code>
<code>element[attribut\$=valeur]</code>	<code>\$("[id\$='param']")</code>

JQuery : insertion d'éléments dans le DOM

Insertion avant la balise fermante

```
$("#selecteur").append("nouveau texte");  
$("#nouveau texte").appendTo("selecteur");
```

Insertion après la balise ouvrante

```
$("#selecteur").prepend("nouveau texte");  
$("#nouveau texte").prependTo("selecteur");
```