

# DIU Enseignement de l'Informatique au Lycée

-

## Programmation Web (côté client)

Pierre Pompidor

21 juin 2019

### Table des matières

<b>1</b>	<b>Cours introductif sur la programmation web</b>	<b>3</b>
1.1	Introduction à la programmation web . . . . .	3
1.1.1	Différence entre internet et web . . . . .	3
1.2	Architectures de bases . . . . .	3
1.2.1	Architectures client/serveur . . . . .	3
1.2.2	Architectures multipages . . . . .	4
1.2.3	Architectures monopages . . . . .	4
1.3	Introduction au langage HTML . . . . .	4
1.3.1	Structure d'une page HTML . . . . .	5
1.3.2	Les balises HTML essentielles . . . . .	5
1.3.3	Utilisation de balises sémantiques . . . . .	6
1.3.4	Structure d'un document utilisant les balises sémantiques . . . . .	6
1.4	Les feuilles de styles CSS . . . . .	6
1.4.1	Application d'un style à un type de balise . . . . .	7
1.4.2	Application d'un style à des balises associées à une classe . . . . .	7
1.4.3	Application d'un style à une balise désignée par un identifiant . . . . .	7
1.4.4	Règles de priorité de l'application des styles CSS . . . . .	8
1.5	Analyse d'une page . . . . .	8
1.5.1	Mode d'emploi . . . . .	8
1.5.2	Le CV de James Bond . . . . .	9
1.5.3	Découverte des outils du navigateur . . . . .	11
<b>2</b>	<b>TP : Création de votre CV</b>	<b>11</b>
<b>3</b>	<b>Cours sur les liens hypertextes et les formulaires</b>	<b>12</b>
3.1	Liens hypertextes . . . . .	12
3.1.1	Ouvertures de pages dans d'autres onglets ou fenêtres . . . . .	12
3.1.2	Enrichissement du CV de James Bond . . . . .	12
3.2	Formulaires . . . . .	13
3.2.1	Appel d'un programme avec paramètres . . . . .	14
3.2.2	Liste des balises les plus usuelles . . . . .	14
<b>4</b>	<b>TP : liens et formulaire</b>	<b>15</b>
4.1	Modification du CV personnel par l'insertion de liens . . . . .	15
4.2	Création d'un formulaire . . . . .	15

<b>5</b>	<b>Par l'exemple, un aperçu du langage JavaScript</b>	<b>16</b>
5.1	Où coder du code javascript et comment le déboguer ? . . . . .	16
5.2	Exemple d'un code JavaScript associé au CV de James qui zoome sa photo . . . . .	16
5.2.1	Mise en place de l'écouteur d'événements (survol de la souris) . . . . .	17
5.2.2	Fonctions JavaScript qui zoome et dézoome la photo . . . . .	17
<b>6</b>	<b>Informations sur l'évaluation du sous-module</b>	<b>18</b>
<b>7</b>	<b>Cours JavaScript</b>	<b>19</b>
7.1	Rôle de JavaScript comme accesseur du DOM . . . . .	19
7.1.1	Pourquoi accéder au DOM . . . . .	19
7.1.2	Sélection d'éléments dans le DOM . . . . .	19
7.2	Présentation des éléments de base de la syntaxe de JavaScript . . . . .	19
7.2.1	Déclaration des variables . . . . .	19
7.2.2	Les structures de données usuelles . . . . .	20
7.2.3	Les listes . . . . .	20
7.2.4	Les objets et les fonctions constructrices . . . . .	20
7.2.5	Les blocs d'instructions et la structure conditionnelle . . . . .	21
7.2.6	Les structures itératives . . . . .	21
<b>8</b>	<b>TP Enrichissement du CV avec un carrousel</b>	<b>22</b>
<b>9</b>	<b>Cours AJAX</b>	<b>22</b>
9.1	Principe de l'importation de données (AJAX) . . . . .	22
9.1.1	Fichier de données JSON . . . . .	22
9.1.2	Code JQuery d'importation des données et leur encapsulation dans du HTML . . . . .	22
9.1.3	Fonctions anonymes passées en paramètres : les fonctions de rappel . . . . .	23
9.2	Exemple d'un code AJAX qui via JQuery charge des données complémentaires au CV de James . . . . .	24
<b>10</b>	<b>TP AJAX</b>	<b>25</b>

# 1 Cours introductif sur la programmation web

## 1.1 Introduction à la programmation web

### 1.1.1 Différence entre internet et web

Le point essentiel est de ne pas confondre **Internet** et **Web**.

**Internet** désigne une **interconnexion de réseaux** de différentes granularités. Ce réseau mondial permet que n'importe quel ordinateur<sup>1</sup> puisse communiquer avec n'importe quel autre ordinateur (à condition bien entendu que ces deux ordinateurs soient connectés au réseau et qu'ils aient les permissions d'établir cette communication). Internet fait donc référence aux matériels et logiciels permettant la transmission de données entre deux ordinateurs situés n'importe où dans le monde. La première connexion utilisant Internet a eu lieu en 1969 aux Etats-Unis.

Le **World Wide Web**, ou plus simplement le **Web** (la Toile), désigne l'ensemble des technologies (protocoles de communication, programmes...) qui permettent d'utiliser Internet pour y retrouver une information. Cette recherche d'information (et la visualisation qui l'accompagne) s'appelle la **navigation** internet. Parmi ces programmes nous pouvons citer les **navigateurs** et les **serveurs web** dont les rôles sont présentés dans les paragraphes suivants.

Les ressources désignées en fonction de leur emplacement sont adressées par des **URL** (Uniform Resource Locator) nommées également **adresses web** (exemple : `http://advanse.lirmm.fr/~pompidor/DUI/cours_DUI_web.html`). Le protocole de communication usuel permettant la navigation est **HTTP** (HyperText Transfert Protocol).

Le Web a été créé par Tim Berners-Lee en 1990 à Genève en Suisse.

## 1.2 Architectures de bases

### 1.2.1 Architectures client/serveur

Les **applications web** (dites en anglais "web app") sont fondées sur une **architecture client/serveur** dans laquelle le **serveur** délivre des ressources, et notamment des pages web au **navigateur**, le **client**, en fonction de la navigation effectuée par l'internaute. Le mot "serveur" confond deux réalités différentes : l'ordinateur distant sur lequel sont situées les ressources demandées, et sur celui-ci, un logiciel qui réceptionne les messages envoyés par le client et qui identifie et renvoie ces ressources. Ce logiciel est plus précisément nommé **serveur web** ou **serveur HTTP**.

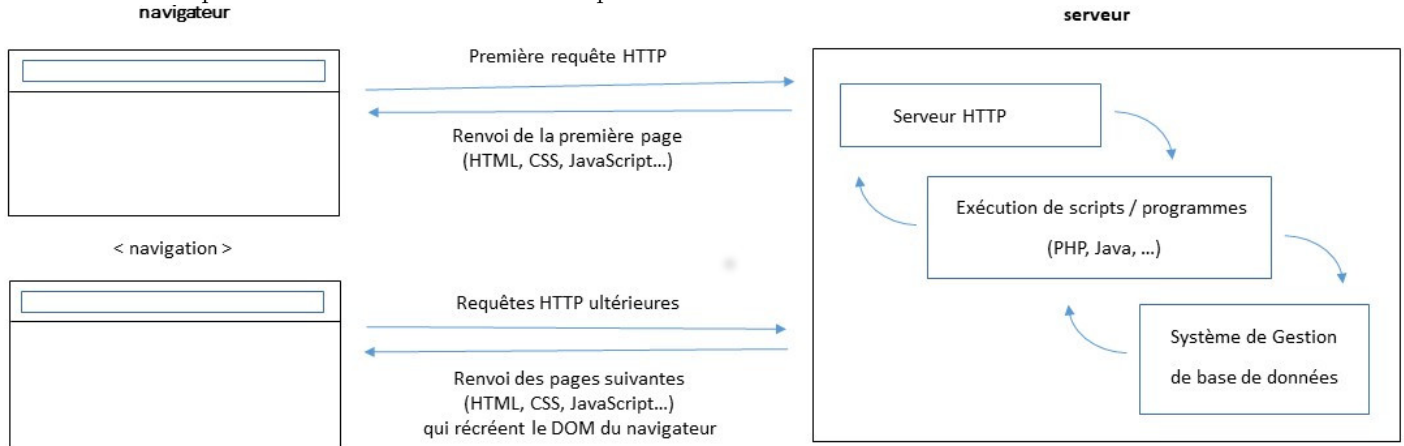
Ces pages web sont fondamentalement structurées grâce à des **balises HTML** qui mettent en page les informations qui y sont présentées, des **styles CSS** (*Cascading Style Sheets*) pour factoriser des styles graphiques entre différents éléments des pages, et enfin des **codes JavaScript** pour gérer du côté client - c'est-à-dire au sein du navigateur - certaines actions qui n'exigent pas du serveur le renvoi d'une nouvelle page.

---

1. Le terme ordinateur est un peu restrictif en sachant que de plus en plus d'**objets connectés**, de fonctionnalités très spécifiques, sont aussi connectés à Internet.

## 1.2.2 Architectures multipages

Voici un schéma présentant une architecture classique :



Il faut noter que dans le schéma ci-dessus, la machine distante (le serveur) regroupe à la fois le serveur HTTP (l'application qui réceptionne les requêtes HTTP et invoque les traitements à effectuer), l'évaluateur qui exécute les programmes, ainsi que le système de gestion de bases de données. Bien entendu d'autres configurations sont aussi possibles : par exemple le système de gestion de bases de données est très fréquemment délocalisé sur un serveur dédié.

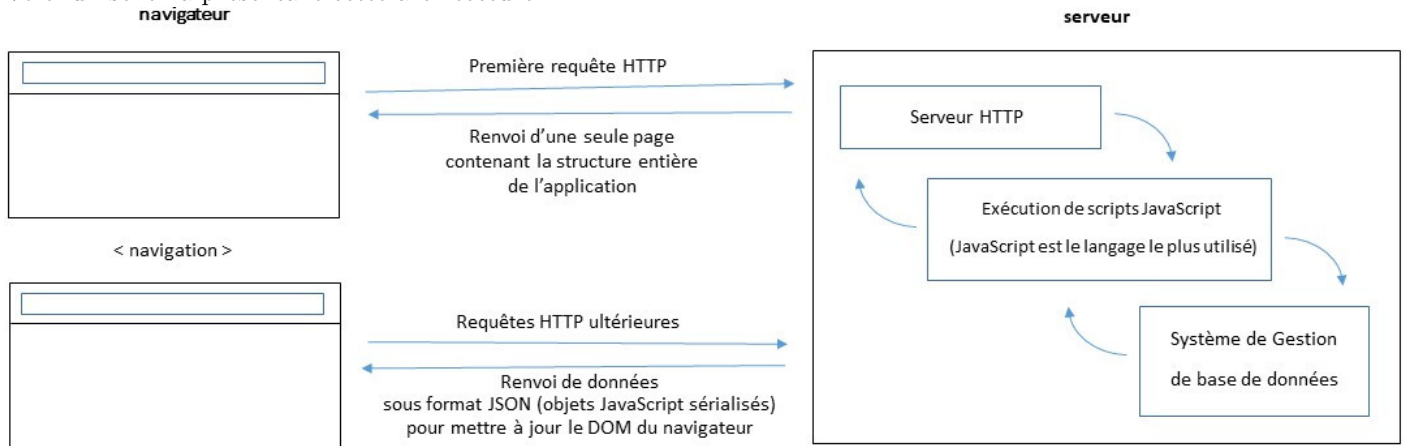
Dans cette architecture très commune, chaque chargement d'une page implique l'allocation en mémoire vive de l'ordinateur qui la reçoit, d'informations correspondantes à chaque élément de cette page. Ces informations sont associées à la mémoire du navigateur et correspondent au modèle objet du document (désigné par l'acronyme anglais DOM). Cette opération peut être très lourde en terme de temps d'exécution.

Attention : ce cours n'introduira que les technologies majeures relatives au côté client (langage HTML, feuilles de style CSS et exemples d'utilisation du langage JavaScript). La mise en œuvre des technologies serveur (avec par exemple l'utilisation du langage PHP et d'un système de gestion de bases de données n'aura lieu que lors de la seconde session d'enseignement).

## 1.2.3 Architectures monopages

Pour optimiser les temps de navigation, une autre architecture est possible. La structure de l'application web - et donc toutes les balises HTML qui la composent - est chargée avant toute navigation, ce qui implique que seules les données à exploiter ne sont ultérieurement transmises du serveur au client.

Voici un schéma présentant cette architecture :



*Attention : cette architecture n'est donnée ici qu'à titre de référence et n'est pas à enseigner aux élèves à moins que ceux-ci l'évoque.*

## 1.3 Introduction au langage HTML

La présentation des données dans la fenêtre d'un navigateur est organisée grâce à l'utilisation d'un langage de description nommé **HTML** (pour HyperText Markup Language). Ce langage, très facile d'accès, mobilise généralement des duos de **balises**, une balise ouvrante et une balise fermante, encadrant les données ou les autres balises sur lesquelles la

directive est portée.<sup>2</sup>

Par exemple, un texte centré et cliquable, renvoyant sur l'article HTML de Wikipedia, peut être codé ainsi :

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

Dans cet exemple, les balises `<center>` et `</center>` portent la directive "centrer horizontalement" sur tout ce qu'elles circonscrivent ; les balises `<a>` et `</a>` portent la directive "créer l'hyperlien affiché sous le nom 'Article sur HTML de Wikipedia'".

Nous remarquons que la balise ouvrante `<a>` qui crée dans notre exemple une ancre sur une chaîne de caractères qui peut être sélectionnée (un hypertexte), est accompagnée d'un **attribut href** qui spécifie l'**URL** (l'adresse internet) où la ressource désignée par cette ancre est stockée. La majorité des balises sont ainsi enrichies d'attributs qui modifient leurs comportements.

Il est important de noter que le langage HTML (accompagné par des feuilles de style CSS qui sont introduites dans le paragraphe suivant) est un langage de programmation dont les fonctionnalités sont **très spécifiques et réduites** par rapport aux principaux langages de programmations (C, C++, Java, Python, PHP, ...). Il ne sert qu'à :

- mettre en forme des informations (disposition dans la fenêtre du navigateur, présentation graphique) ;
- créer des liens hypertextes (comme vu dans l'exemple précédent) ;
- créer des formulaires qui permettent d'invoquer des programmes (cette invocation pouvant être accompagnée de paramètres).

### 1.3.1 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>`

Nous aurons donc quasiment toujours la structure suivante :

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

Le bloc "body" contient les éléments à afficher dans la fenêtre du navigateur.

Le bloc "head" contient optionnellement :

- la spécification de l'encodage des caractères : UTF-8 (à privilégier) ou ISO-8859-1
- le nom de la page qui apparaîtra dans l'onglet du navigateur entre les balises `<title>` et `</title>`
- le(s) lien(s) sur le(s) fichier(s) contenant les styles CSS (ou directement leurs spécifications entre les balises `<style>` et `</style>`)
- le(s) lien(s) sur le(s) fichier(s) contenant les codes JavaScript associés (ou directement du code JavaScript entre les balises `<script>` et `</script>`)

### 1.3.2 Les balises HTML essentielles

*L'enseignement d'HTML aux élèves ne devant pas se transformer en un inventaire à la Prévert (les élèves ayant tout loisir de les découvrir par eux-mêmes), seules les principales balises devront être introduites en cours.*

Le langage HTML nous offre un certain nombre de balises qui permettent de disposer et de structurer les données à afficher :

- la balise `<p>` permet de créer un paragraphe (avec une indentation au début et un espacement terminal) ;
- les balises `<ul>` et `<li>` permettent de créer respectivement une liste et un item (non numéroté) dans la liste<sup>3</sup> ;
- les balises `<table>`, `<tr>` et `<td>` permettent de créer respectivement un tableau, une ligne dans un tableau et une cellule dans la ligne ;

---

2. Le langage HTML donne aussi la possibilité d'utiliser des **balises autofermantes** quand aucune donnée n'est assujettie à son action : par exemple la balise `<br/>` crée un retour à la ligne (remarquez la position du slash).

3. Utilisez la balise `<ol>` pour créer une liste possédant des items numérotés.

- la balise `<div>` permet de créer un bloc qui pourra être placé à n'importe quel endroit de la page via un style CSS ;
- la balise `<center>` permet un centrage horizontal ;
- la balise `<br/>` permet de créer un saut de ligne ;

Des exemples de mises en œuvre de listes et d'un tableau sont donnés dans le paragraphe "Le CV de James Bond".

Le langage HTML propose également des balises permettant d'associer des propriétés graphiques à des chaînes de caractères, les plus usuelles sont :

- les balises `<h1>` à `<h7>` qui permettent de régler leurs tailles et leurs poids (de la plus "grosse" à la plus "petite") ;
- la balise `<font>` qui permet de régler (entre autres) leurs polices et leurs couleurs d'avant et d'arrière plan ;
- la balise `<b>` qui permet de mettre en gras ;
- la balise `<i>` qui permet de mettre en italique ;
- la balise `<span>` qui permet de distinguer un fragment de texte en lui associant un style CSS.

**Hormis la balise `<span>`, il est vraiment déconseillé d'utiliser ces balises car leur mise en œuvre va confondre dans le même document les données (et leur structuration), et leurs caractéristiques graphiques, ce qui empêche une bonne évolutivité des codes HTML. Ainsi tout ce qui concerne l'apparence graphique (emplacements, tailles, couleurs, ...) doit être délégué à une feuille de style CSS.**

Une évolution du langage HTML propose des balises sémantiques qui peuvent être considérées comme des balises `<div>` typées. Même si ce typage n'implique pas un positionnement automatique (ce positionnement devant être assuré par des styles CSS), la sémantique associée à leur nom rend le code HTML plus lisible. Ces balises sont explicitées dans le paragraphe suivant.

### 1.3.3 Utilisation de balises sémantiques

Les balises sémantiques sont particulièrement adaptées pour la publication d'articles. Dans l'exemple qui suivra, elles seront utilisées pour la création d'un CV bien que les balises `<div>` plus génériques pourraient être également employées dans ce but. Les principales balises sémantiques sont :

- la balise `<header>` qui structure l'entête de page (généralement en affichage permanent) ;
- la balise `<main>` qui structure le document en articles ;
- la balise `<article>` qui structure un article ;
- la balise `<section>` qui définit une section d'un article ;
- la balise `<footer>` qui structure le pied de page (généralement en affichage permanent).

### 1.3.4 Structure d'un document utilisant les balises sémantiques

Voici une structure simplifiée d'une page HTML utilisant les balises sémantiques :

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

## 1.4 Les feuilles de styles CSS

Les **feuilles de styles CSS** (Cascading Style Sheets) sont des fichiers textuels qui permettent d'associer des styles graphiques (emplacements, tailles, couleurs...) aux éléments d'une page HTML.

L'association entre une page HTML et sa (ou ses) feuille(s) de style(s) est assurée par la balise `<link>` dans le bloc `head` de la page HTML :

```
<html>
  <head>
    <link rel="stylesheet" href="<nom de la feuille de style>" />
  </head>
  ...
</html>
```

Un **style graphique** est lui même décomposé en **propriétés graphiques**.

Les styles graphiques peuvent s'appliquer<sup>4</sup> :

- sur un type de balises (par exemple toutes les balises `<p>`);
- à des balises associées à une classe<sup>5</sup> (par exemple tous les articles qui doivent être justifiés à droite);
- à une balise désignée par un identifiant unique (un "id");

#### 1.4.1 Application d'un style à un type de balise

Voici l'exemple d'un style graphique associé aux balises `<p>` (qui intègrent les textes associés dans un paragraphe).

Dans le CV que nous allons étudier, un paragraphe est créé pour spécifier le nom du personnage à présenter ainsi que sa photo :

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

Voici le style graphique associé :

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

Et ainsi les caractères de tous les paragraphes (\*), et donc de celui-ci, ont une de taille de 18 points, ils sont mis en gras et pris dans la police de caractères "Courier New".

A moins qu'un style de portée plus spécifique, par exemple sur une partie du texte du paragraphe, ne soit appliqué.

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

Des styles peuvent être associés à différentes balises associées à une classe.

Dans le CV que nous allons étudier, certains blocs d'informations (des articles) doivent être justifiés à droite de la fenêtre (dans un but purement esthétique). C'est le cas pour celui décrivant les compétences de James Bond :

```
<article class="right">
  <p> Compétences : </p>
  <ul>
    <li> Permis B (avec certificat d'aptitude à la conduite rapide) </li>
    <li> Langues étrangères : russe, japonais, danois (notions) </li>
  </ul>
</article>
```

Voici le style associé qui définit cette justification (remarquez le point qui désigne un nom de classe) :

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

6

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

Des styles peuvent être associés à des balises spécifiquement désignées par des identifiants uniques.

Dans le paragraphe vu précédemment, la balise `<img>` permet d'afficher la photo d'identité de James Bond. Un identifiant unique ("photo") peut être associé à cette image grâce à l'attribut `id` (nous verrons l'usage d'un tel identifiant dans la partie sur AJAX).

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

Voici le style associé à cette balise (remarquez le dièse qui désigne un nom d'identifiant).

4. D'autres types de sélections utilisant d'autres sélecteurs existent.

5. Le mot *classe* renvoie à un regroupement arbitraire de balises.

6. Une extension à CSS, appelée FlexBox, permet également de positionner des éléments dans une page HTML.

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

Ainsi la taille de l'image est fixée à 100 pixels de largeur sur 120 pixels de hauteur, et celle-ci est positionnée (elle flotte...) à droite du paragraphe.

#### 1.4.4 Règles de priorité de l'application des styles CSS

Le fait que plusieurs styles CSS puissent être appliqués à une même balise, implique l'application de différentes règles de priorités : la principale est que le style le plus spécifique à la balise est prioritaire.

Par exemple dans la feuille de style suivante :

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

la taille du texte assujetti à une balise `p` emboîtée (quel que soit son niveau) dans une balise `header` est potentiellement réglée par deux styles : celui qui s'applique (en cascade) à partir de la balise `<body>`, et celui qui s'applique (toujours en cascade) à partir de la balise `<header>`.

C'est donc la taille de ce second style qui est choisie.

### 1.5 Analyse d'une page

Analysons un CV simplifié de James Bond, c'est à dire le code HTML qui structure les informations et la feuille de style CSS qui associe à ces informations des styles graphiques.

#### 1.5.1 Mode d'emploi

Ouvrez un terminal ; vous êtes positionné au niveau de votre répertoire (dossier) d'accueil.

Créez un dossier nommé par exemple WEB et déplacez-vous-y :

```
mkdir WEB
cd WEB
```

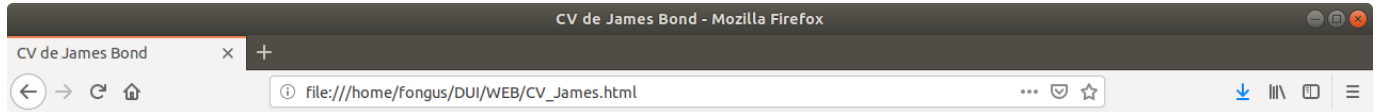
Copier dans votre dossier les fichiers `CV_James.html`, `styles_CV_James.css` et `photo_James.jpg` à partir d'un des sites suivants :

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



## 1.5.2 Le CV de James Bond

Nous voulons faire apparaître le CV de James Bond que voici :



### 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

Voici le code HTML correspondant :

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

  <body>
    <header>
      <p> James Bond  </p>
      Nationalité britannique <br/>
      Profession : agent secret (dernier employeur : MI6)
    </header>

    <main>
      <article>
        <p> Expérience professionnelle : </p>
        <table border="1">
          <tr> <th> Année </th> <th> Nature de la mission </th> </tr>
          <tr> <td> 1954 </td> <td> Neutralisation d'un dangereux amnésique (le Chiffre) </td> </tr>
          <tr> <td> 1961 </td> <td> Prévention d'une catastrophe nucléaire </td> </tr>
          <tr> <td> 1962 </td> <td> Démantèlement d'une organisation criminelle </td> </tr>
          <tr> <td> ... </td> <td> ... </td> </tr>
        </table>
      </article>

      <article class="right">
        <p> Compétences : </p>
        <ul>
          <li> Permis B (avec certificat d'aptitude à la conduite rapide) </li>
          <li> Langues étrangères : russe, japonais, danois (notions) </li>
        </ul>
      </article>

      <article>
        <p> Hobbies : </p>
        <ul>
          <li> Dégustation des vodka-martinis au shaker </li>
          <li> Conversation mondaine en présence de public féminin </li>
        </ul>
      </article>
    </main>
  </body>
</html>
```

Cet exemple synthétise les différentes informations qui ont été données dans ce magnifique support :

- la balise `<head>` importe bien la feuille de style associée à la page;
- les balises sémantiques `<header>`, `<main>` et `<article>` ont été mises en œuvre : nous aurions aussi pu utiliser des balises `<div>`;
- un tableau de deux colonnes a été créé pour lister les exploits de James;
- une liste a été créée pour lister les hobbies de James.

Nous voyons aussi apparaître quelques notions supplémentaires :

- la balise SGML<sup>7</sup> initiale `<!doctype html>` signe le type de la ressource (ici une page HTML) : elle n'est pas obligatoire mais recommandée;
- la balise `<th>` (*table header*) permet de mettre en gras les éléments d'entête d'un tableau.

---

7. SGML (*Standard Generalized Markup Language*) est le langage ancêtre et normatif de tous les langages balisés.

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; }

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; }
```

Nous voyons aussi ici apparaître quelques notions supplémentaires sur les styles CSS :

- la propriété de style **margin** définit un espacement extérieur;
- les deux spécialisations **margin-left** et **margin-right** de cette propriété, avec la valeur **auto**, permettent de centrer le CV au milieu de la fenêtre du navigateur.

### 1.5.3 Découverte des outils du navigateur

Si vous utilisez Firefox, Chrome ou Chromium, vous pouvez faire apparaître les outils du navigateur en appuyant sur le touche **F12** (si vous utilisez Safari, ouvrez le menu du navigateur).

Quatre de ces outils sont d'une importance capitale : l'inspecteur DOM, l'éditeur de style, le réseau et la console.

- l'inspecteur DOM dévoile ce qui a été alloué en mémoire du navigateur à partir de l'analyse de la page HTML qu'il a reçue;
- l'éditeur de style permet de visualiser et modifier les styles CSS;
- le réseau permet de vérifier que toutes les ressources utiles à la page ont bien été chargées par le navigateur;
- la console (qui nous sera très utile ultérieurement) affiche les erreurs des codes JavaScript associés à la page.

Parcourez l'arborescence des éléments du DOM via l'inspecteur DOM, puis grâce à l'éditeur de style modifiez "in vivo" les styles CSS pour exprimer votre bon goût.

## 2 TP : Création de votre CV

A partir des éléments de cours et de votre créativité artistique, créez un CV simplifié.

Pour ce faire, vous devez :

- sur papier (ou dans votre tête) définir un gabarit, c'est à dire créer une maquette grossière des éléments à afficher ainsi que leurs emplacements;
- implémenter un code HTML qui structure les blocs d'informations en articles (et éventuellement sections), tableaux, listes, ...;
- implémenter une feuille CSS qui associe aux différents éléments à afficher des styles graphiques.

Si vous n'avez pas d'idée de gabarit, voici un site qui peut vous stimuler : <https://www.modeles-de-cv.com/>

## 3 Cours sur les liens hypertextes et les formulaires

### 3.1 Liens hypertextes

Outre l’affichage de données, le langage HTML permet de créer des **liens hypertextes** (avec la balise `<a>`) dont les sélections permettent :

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

Nous nous intéresserons ici qu’à la navigation sur des liens externes. Dans ce cadre, la page que nous voulons accéder est désignée par une URL en valeur de l’attribut `href` de la balise `<a>` (nommée ainsi pour ”anchor”). Cette nouvelle page pourra, remplacer la page courante (comportement par défaut), être ouverte dans un nouvel onglet du navigateur, voire même apparaître dans une pop-up (mais l’utilisation de pop-ups n’est pas recommandée car leur usage est souvent perturbant et elles peuvent être par défaut refusées par le navigateur).

#### 3.1.1 Ouvertures de pages dans d’autres onglets ou fenêtres

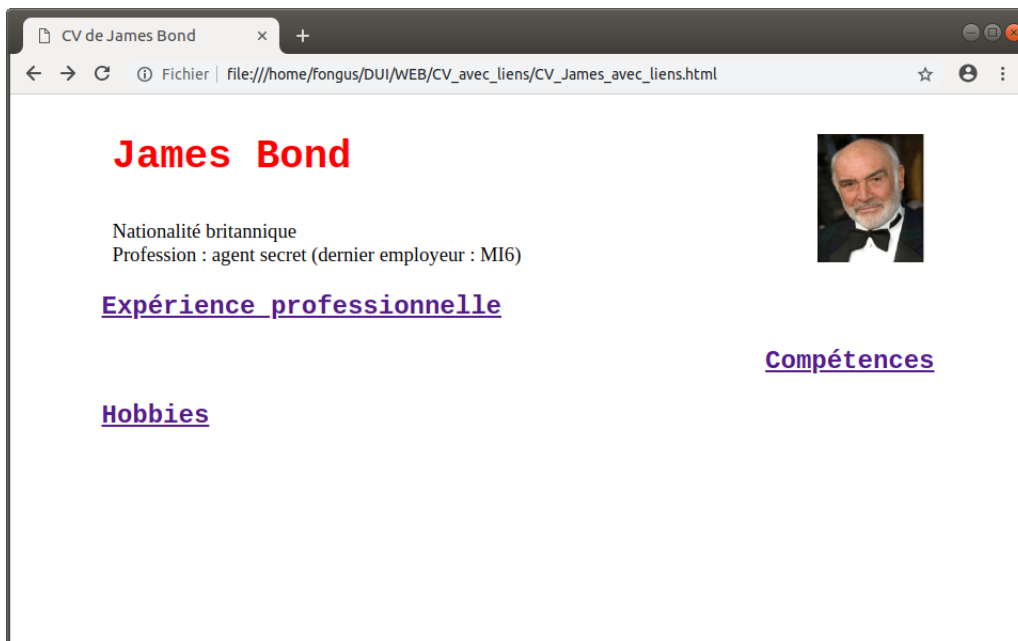
Voici le schéma d’utilisation de la balise `<a>` pour ouvrir la ressource dans un nouvel onglet du navigateur :

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

La propriété `target` avec la valeur `_blank` provoque l’ouverture de la ressource dans un nouvel onglet.

#### 3.1.2 Enrichissement du CV de James Bond

Modifions le CV de James pour conditionner l’affichage de son expérience professionnelle, de ses compétences et de ses hobbies à la sélection de liens :



Le code HTML devient donc comme suit :

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

  <body>
    <header>
      <p> James Bond  </p>
      Nationalité britannique <br/>
      Profession : agent secret (dernier employeur : MI6)
    </header>

    <main>
      <article>
        <p> <a href="CV_James_experiences.html" target="_blank"> Expérience 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>
    </main>
  </body>
</html>
```

Et voici le fichier *CV\_James\_experiences.html* invoqué sur le premier lien :

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

  <body>
    <p> Expérience professionnelle : </p>
    <table border="1">
      <tr> <th> Année </th> <th> Nature de la mission </th> </tr>
      <tr> <td> 1954 </td> <td> Neutralisation d'un dangereux amnésique (le Chiffre) </td> </tr>
      <tr> <td> 1961 </td> <td> Prévention d'une catastrophe nucléaire </td> </tr>
      <tr> <td> 1962 </td> <td> Démantèlement d'une organisation criminelle </td> </tr>
      <tr> <td> ... </td> <td> ... </td> </tr>
    </table>
  </body>
</html>
```

Vous remarquerez la factorisation de la feuille de style entre les différents fichiers HTML.

## 3.2 Formulaires

Avec la mise en forme de données à afficher et la création de liens hypertextes, la création de **formulaires** est la troisième fonction du langage HTML.

Un formulaire permet d'invoquer sur Internet un programme en lui passant optionnellement des paramètres.

### 3.2.1 Appel d'un programme avec paramètres

La mise en œuvre d'un formulaire repose sur la balise `<form>`.

Voici un schéma d'utilisation de cette balise, en utilisant ici un bouton de validation créé par la balise `<input>` :

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

Les ... désignent l'emplacement des différents éléments du formulaire (zones de saisies de texte, boutons radion, cases à cocher...) que l'on nomme "contrôles", et qui sont explicités dans le paragraphe suivant.

Lors de la sélection du bouton de validation, une **requête HTTP** est envoyée par le navigateur au serveur HTTP localisé sur la machine distante. L'adresse de la machine distante (*remote host*) est connue grâce à l'URL donnée en valeur de l'attribut `action` de la balise `<form>`.

Cette requête peut être envoyée de différentes façons (*method*) :

- par la **méthode GET** : les paramètres sont associés à l'URL et donc visibles dans la barre du navigateur (ce qui est bien pour mettre au point un programme mais peut être par la suite gênant) ;
- par la **méthode POST** : les paramètres sont insérés dans le corps du message.

Par défaut c'est la méthode GET qui est appliquée. Pour appliquer la méthode POST plus sécuritaire, vous devez rajouter l'attribut `method` avec la valeur POST à la balise `action` :

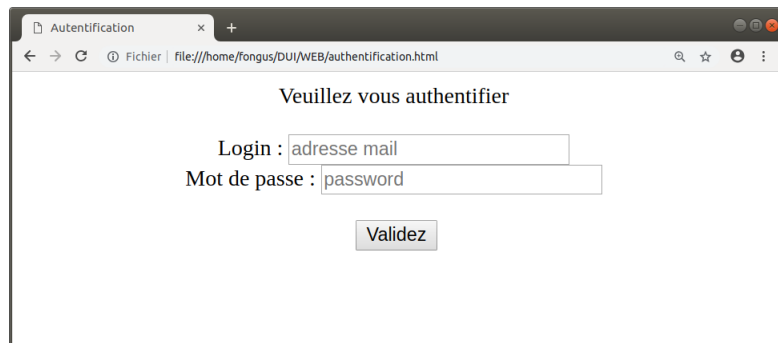
```
<form action="<URL>" method="POST">
...
  <input type="submit" value="Validez" />
</form>
```

### 3.2.2 Liste des balises les plus usuelles

Les balises permettant de créer des contrôles sont :

- la balise `<input>` qui permet de créer :
  - des zones de saisie de chaînes de caractères avec `text`
  - des zones de saisie de chaînes de caractères masquées avec `password`
  - des cases à cocher avec `checkbox`
  - des boutons radio avec `radio`comme valeur de l'attribut `type` ;
- la balise `<select>` qui permet de créer des listes déroulantes (de conserve avec la balise `<option>` qui crée chaque item de la liste déroulante ;
- la balise `<textarea>` qui permet de créer des zones de saisie de textes multi-lignes

Voici l'exemple d'un formulaire d'authentification typique (mais sans "décoration") :



créé par le code suivant :

```

<!doctype html>
<html>
  <head>
    <meta charset="UTF8" />
    <title> Authentification </title>
  </head>

  <body>
    <center>
      Veuillez vous authentifier <br/><br/>
      <form action="<URL>" method="GET">
        Login : <input type="text" name="login" placeholder="votre nom" required /> <br/>
        Mot de passe : <input type="password" name="pass" placeholder="mot de passe" required />
        <br/> <br/>
        <input type="submit" value="Validez">
      </form>
    </center>
  </body>
</html>

```

Vous remarquerez la présence :

- de l'attribut **placeholder** qui permet d'afficher une indication dans la zone de saisie ;
- de l'attribut **required** qui rend obligatoire la saisie d'une information.
- et surtout de l'attribut **name** qui en nommant chaque contrôle du formulaire permet d'émettre un paramètre de même nom ; par exemple si dans l'exemple ci-dessus
  - l'URL a pour valeur *https://monsite.portaildesloosers.fr/identification.php* (la soumission du formulaire permettrait d'invoquer un programme écrit dans le langage de programmation PHP) ;
  - la méthode est **GET** (pour que les paramètres soient accolés à l'URL bien que cela ne soit pas du tout une bonne idée dans ce cas) ;
  - le login a pour valeur *pompidor* ;
  - le password a pour valeur *2fast4U* ;

l'URL créée lors de la soumission du formulaire aura la valeur suivante :

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

Dans le cas où un choix exclusif doit être exprimé par plusieurs boutons radio, ceux-ci doivent être nommés avec le même nom (pour que la sélection d'un de ceux-ci, désélectionne le bouton précédemment sélectionné).

## 4 TP : liens et formulaire

### 4.1 Modification du CV personnel par l'insertion de liens

Le but de ce TP est d'ouvrir quelques rubriques de votre CV, par exemple celles concernant votre cursus universitaire ou vos délicieux passe-temps, dans d'autres onglets du navigateur lors de la sélection du lien correspondant.

Pour cela vous devez :

- créer de nouvelles pages HTML contenant les informations des sous-rubriques
- et créer dans le fichier principal les liens désignant ces nouvelles pages.

### 4.2 Création d'un formulaire

Créez un formulaire qui :

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

Ce formulaire doit mettre en œuvre la méthode **GET** pour que l'URL accompagnée de ses paramètres soit visible dans la barre du navigateur.

## 5 Par l'exemple, un aperçu du langage JavaScript

**JavaScript** est un langage de programmation créé en 1995 par *Brendan Eich* qui travaillait pour la société *Netscape*. Le but originel du langage est de créer des scripts (c'est à dire des programmes interprétés), qui sont exécutés par un navigateur, principalement pour manipuler les données du **DOM** (*Document Object Model*), c'est à dire les objets (au sens informatique du terme) représentant les éléments d'un document balisé (par exemple ceux d'une page HTML) alloués en mémoire du navigateur.

JavaScript a ensuite beaucoup évolué fonctionnellement (par exemple en permettant l'accès asynchrone (non bloquant) à des données fournies par le serveur) et a même récemment investi "le côté serveur" avec l'environnement *Node.js*.

Malgré son nom qui peut porter à confusion, JavaScript a très peu de liens avec la langage *Java* (en fait seulement quelques structures syntaxiques qui proviennent en fait du langage *C*).

Le langage JavaScript se situe à la confluence de deux paradigmes de programmation : la paradigme de la **programmation fonctionnelle** et celui de la **programmation par objets**, et cela le rend difficile à maîtriser.

*Le but de ce cours n'est pas d'apprendre en profondeur ce langage, mais de montrer quelques fonctionnalités simples à mettre en œuvre.*

### 5.1 Où coder du code javascript et comment le déboguer ?

Pour une utilisation côté client (*client-side*) hors framework, les codes JavaScript doivent être externalisés dans des fichiers d'extension **.js** et liés aux codes HTML via la balise **<script>**.

Soit un fichier JavaScript nommé *bonjour.js* ne contenant que la ligne suivante : `console.log("Bonjour !");`

Et la page HTML *test.html* mettant en œuvre ce script :

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

Cet exemple de code est exécutable en étant chargé (ouvert) dans votre navigateur.

Les messages générés par la méthode *log()* de l'objet *console* seront affichés dans la console qui fait partie des **outils du navigateur** (appuyez sur la touche <F12> pour la faire apparaître).

Remarque : surtout n'utilisez jamais une balise **<script>** auto-fermante !

### 5.2 Exemple d'un code JavaScript associé au CV de James qui zoome sa photo

Une fonctionnalité très usuelle de JavaScript est la mise en place d'actions suite aux actions de l'internaute.

Différents **gestionnaires d'événements** peuvent être associés aux éléments HTML. Dès qu'un gestionnaire est mis en œuvre, il scrute les événements produits par les actions de l'internaute, ou le changement de l'état de la page web, correspondants à sa raison d'être.

Par exemple, quand l'internaute clique, ou survole avec le pointeur de souris, une image ou une chaîne de caractères, un événement est déclenché ce qui provoque l'exécution de la fonction JavaScript associée.

Voici les principaux gestionnaires d'événements qui correspondent à des attributs à ajouter aux balises HTML concernées :

- **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 contrôle (par exemple un item de liste déroulante)
- **onload** : chargement de la page dans le navigateur (cet attribut doit accompagner la balise **<body>**)
- **onunload** : l'internaute quitte la page (cet attribut doit accompagner la balise **<body>**)

Voici le schéma de programmation qui permet d'associer un gestionnaire d'événements qui traite les clics souris effectués sur une image :

```
" />
```



### 5.2.1 Mise en place de l'écouteur d'événements (survol de la souris)

Notre but est de pouvoir agrandir à sa taille originelle la photo de James quand celle-ci est survolée par le pointeur de la souris. Voici le code mettant en œuvre les deux gestionnaires d'événements :

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF8" />
    <title> CV de James Bond avec zoom </title>
    <link rel="stylesheet" href="styles_CV_James.css" />
    <script src="CV_James.js"></script>
  </head>

  <body>
    <header>
      <p> James Bond 
      Nationalité britannique <br/>
      Profession : agent secret (dernier employeur : MI6)
    </header>
    ...
  </body>
</html>
```

Les deux gestionnaires d'événements `onmouseover` et `onmouseout` appellent respectivement une fonction JavaScript qui zoome la photo et une autre qui la remet dans sa taille réduite.

`this` représente l'élément du DOM sur lequel l'événement a porté (ici l'image) (techniquement cette variable contient l'adresse en mémoire de cet élément).

### 5.2.2 Fonctions JavaScript qui zoome et dézoome la photo

Voici le contenu du fichier *CV\_James\_avec\_zoom\_photo.js* qui contient le code JavaScript permettant d'agrandir la photo quand celle-ci est survolée par le pointeur de la souris :

```
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;
}
```

Ce code se décompose en trois parties :

Deux **variables** `largeurImage` et `hauteurImage` sont déclarées et évaluées avec la largeur et la hauteur de l'image (définies dans la feuille de style) pour pouvoir remettre l'image dans sa dimension initiale quand le pointeur de la souris la quittera.

Le **typage des variables est dynamique** en JavaScript, c'est à dire que comme en Python, il n'est pas besoin de préciser un typage (entier, réel, caractère, chaîne de caractères, ...) lors de leurs déclarations.

Le mot réservé **var** permet de déclarer une nouvelle variable globalement (elle sera connue dans tout le code), à la différence de **let** qui restreint sa visibilité (sa "portée") au bloc d'instructions dans lequel elle apparaît.

La fonction `zoom()` qui 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 (ouvrante et fermante) qui délimitent un **bloc d'instructions**.

Son paramètre correspond à la référence de l'image : il contient l'adresse en mémoire de l'objet (une zone mémoire implémentant une liste de duos clefs/valeurs) qui contient les différents attributs (ou propriétés) de l'image. Ainsi la largeur et la hauteur de l'image sont stockées comme valeurs des deux propriétés **width** et **height** contenues dans un sous-objet en valeur de la propriété **style** de l'objet **image**.

La notion d'**objet** liée au paradigme de la programmation par objets ne sera pas approfondie ici. Il faut juste signaler que contrairement aux autres langages objets majeurs (C++, Java, ...) la programmation par objets en JavaScript ne nécessite pas la définition de classes (même s'il est possible depuis sa dernière évolution d'en utiliser).

Cette fonction recopie la largeur et la hauteur de l'image dans les variables globales **largeurImage** et **hauteurImage**, puis value les propriétés de l'image en mémoire du navigateur à **auto**, c'est à dire aux valeurs intrinsèques de l'image.

La fonction **dezoom()** qui est invoquée par le gestionnaire d'événements **onmouseout** (le pointeur de souris a quitté l'image). Elle remet les valeurs de largeur et de hauteur de l'image à ses valeurs initiales.

*Le choix des noms de variable est à la discrétion du programmeur, mais il est fortement recommandée que leur sémantique soit forte pour une bonne lisibilité du code, et qu'elles soient définies soient en français, soient en anglais, mais en évitant un mélange des deux.*

## 6 Informations sur l'évaluation du sous-module

L'évaluation de ce module consistera en la présentation par binôme d'un exercice que vous proposeriez à vos élèves.

Cette évaluation aura lieu lors de la seconde partie du second et dernier créneau de cet enseignement.

Cet exercice durera au plus 10 minutes.

## 7 Cours JavaScript

### 7.1 Rôle de JavaScript comme accesseur du DOM

Le **DOM** (Document Object Model) représente toutes les données mises en mémoire du navigateur suite au chargement d'une page.

En effet, quand du code HTML est transmis au navigateur, celui-ci va utiliser un outillage spécifique pour allouer en mémoire de l'ordinateur les différentes entités de ce code (balises, attributs, informations textuelles, ...). Cet outillage est constitué de classes au sens de la programmation par objets - ce point ne sera pas détaillé dans ce support - et est nommé le Document Object Model (le DOM). Par extension le DOM désigne également toutes les données mises en mémoire et associées à une page affichée dans une fenêtre du navigateur.

L'allocation d'une balise HTML est appelé un **élément**.

#### 7.1.1 Pourquoi accéder au DOM

L'accès au DOM est inévitable si on veut modifier le contenu d'une page sans être obligé de demander au serveur de nous renvoyer une nouvelle page. C'est par exemple le cas vu précédemment, où nous voulions agrandir une image en la survolant.

JavaScript a été initialement conçu pour assurer cette fonctionnalité. Il possède donc des fonctions spécifiques pour sélectionner un ou plusieurs éléments du DOM :

- via le nom d'un type de balises (par exemple toutes les balises `<img>`);
- via un identifiant (exprimé par un attribut `id`).

#### 7.1.2 Sélection d'éléments dans le DOM

Pour sélectionner des éléments du DOM (c'est à dire des balises allouées en mémoire du navigateur), il y a deux possibilités :

- utiliser les fonctions JavaScript de l'API DOM "de base" (comme `getElementById()` ou `getElementsByName()`);
- utiliser une bibliothèque JavaScript de plus haut niveau qui offre une syntaxe plus concise : la bibliothèque la plus populaire est **JQuery**.

Voici comment par exemple sélectionner l'élément correspondant à l'image identifiée par l'identifiant `photo`, pour afficher le nom du fichier image dans la console JavaScript :

Avec l'API DOM :

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

Avec la bibliothèque JQuery :

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

### 7.2 Présentation des éléments de base de la syntaxe de JavaScript

#### 7.2.1 Déclaration des variables

Les variables sont **typées dynamiquement** (et non pas lors de leurs déclarations). Elles sont déclarées par :

- le mot réservé **var** : la portée de la variable est globale, mais si elle est déclarée dans une fonction, la variable n'est pas visible dans les fonctions incluses) : depuis la création du mot réservé *let*, ce typage est beaucoup moins employé;
- 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.

JavaScript va typer (en interne) les variables suivant six types primitifs :

- un type **booléen** : **true** et **false**;
- un type **nul** : **null** (qui représente une adresse en mémoire qui ne pointe vers rien);
- un type **indéfini** (en résultat de l'accès à une variable qui n'existe pas ou qui n'a pas de valeur) : **undefined** ; exemple de variable créée mais de valeur indéfinie : `var i`;
- un type pour les **nombres** qu'ils soient entiers ou réels : **number** ;
- un type pour les **chaînes de caractères** : **string**  
les chaînes de caractères peuvent être circonscrites par des simples quotes ou des doubles quotes ; nous prendrons le parti dans nos exemples, de généralement les encadrer par des doubles quotes.

- un type pour les **symboles** (ce type est introduit par ECMAScript 6 et ne sera pas développé ici).
- et le type **Object** dans les autres cas de figures : nous voyons déjà que JavaScript est bien un langage à objets.

Voici quelques exemples de création de variables scalaires (en rangeant par simplicité dans cette catégorie les chaînes de caractères) :

```
var bizarre; // sa valeur est : undefined
var i = 0;
let bool = true;
const nom = "Pierre l'immuable";
```

## 7.2.2 Les structures de données usuelles

En JavaScript, deux structures de données sont omniprésentes :

- les **objets** ;
- les **listes** (qui correspondent à des tableaux dynamiques et qui sont par ailleurs des objets).

*Nous présenterons d'abord les listes avant les objets.*

## 7.2.3 Les listes

Les listes sont créées soit en utilisant la fonction constructrice `Array()` (les fonctions constructrices ne seront pas détaillées dans ce support), soit directement en employant les crochets (qui ne sont que du sucre syntaxique).

Par exemple, si nous voulons créer une liste nommée `maListe` qui contiendra les chiffres 1, 2 et 3 et la chaîne de caractères "partez", voici les deux façons de faire :

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

Le nombre d'éléments d'une liste est donné par l'attribut `length`

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

## 7.2.4 Les objets et les fonctions constructrices

Contrairement à de nombreux autres langages de programmation, les objets peuvent être créés littéralement en JavaScript (et ne sont pas originellement des instances de classes). La syntaxe qui permet de formater ces objets lorsque ceux-ci sont sérialisés (càd écrits dans un fichier texte) est appelée **JSON** (*JavaScript Object Notation*). Cette syntaxe est devenue extrêmement populaire dans l'échange de données (en rendant XML beaucoup moins utilisé).

Pour information, la gestion des classes en JavaScript a été introduite via la norme ECMAScript 6 et des extensions "de plus haut niveau" comme **TypeScript** (<http://www.typescriptlang.org/>) ou **Dart** (<https://www.dartlang.org/>).

Dans cet exemple, un objet littéral *monObjet* est défini comme suit :

```
let monObjet = { "nom": "Bond",
                 "prénom": "James",
                 "acteurs": ["Sean Connery", "George Lazenby", "Roger Moore",
                             "Timothy Dalton", "Pierce Brosnan", "Daniel Craig"],
                 "films": {"1962" : "James Bond 007 contre Dr. No",
                           "1963" : "Bons baisers de Russie",
                           "1964" : "Goldfinger"}
               };
```

Voici quelques exemples de manipulation de cet objet (vous remarquerez que la valeur de la troisième propriété est une liste et celle de la quatrième un objet) :

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

Les objets peuvent être aussi instanciés par des **fonctions constructrices**.

Voici un exemple d'une fonction constructrice *personne()* qui crée un objet *bond*.

Le mot réservé **new** alloue une zone mémoire dans laquelle seront associées variables (ici *nom* et *prenom*) et méthodes (ici *cv()*) qui utilisent ces variables. Le mot réservé **this** représente dans la classe la référence à la zone mémoire dévolue à l'objet en cours de construction.

```
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
```

### 7.2.5 Les blocs d'instructions et la structure conditionnelle

En JavaScript les blocs d'instructions sont généralement circonscrits par des **accolades**. Dans le cas où une seule instruction est assujettie au bloc, les accolades peuvent être omises. Mais même dans ces cas-là, il est préférable de garder les accolades ce qui est une sécurité lors de l'ajout d'une nouvelle instruction au bloc.

Les structures conditionnelles et itératives créent des blocs d'instructions.

La structure conditionnelle présente le schéma suivant :

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

### 7.2.6 Les structures itératives

Différentes structures itératives existent en JavaScript :

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

Nous ne présenterons pas de manière détaillée les structures itératives dans ce support hormis une proposée par la bibliothèque JQuery dans le chapitre concernant AJAX.

Cela-dit, voici deux exemples de mise en œuvre des deux structures *for* qui sont les plus usitées.

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); }
```

## 8 TP Enrichissement du CV avec un carrousel

Le but de ce TP est de pouvoir faire défiler les photos des différents acteurs (en tout cas des principaux) ayant joué le rôle de James en cliquant plusieurs fois sur la photo.

Dans un premier temps, mettez en œuvre le zoom sur la photo accompagnant votre CV lors d'un survol du pointeur souris (si vous ne l'avez donc fait, ajoutez une photo à votre CV).

Dans un second temps, faites défiler un carrousel circulaire de photos en cliquant sur la photo d'identité de James. Pour cela vous devez :

- associer un gestionnaire d'événements `onclick` à la balise `<img>` pour appeler une fonction JavaScript `carrousel()`
- récupérer sur internet une photo de chaque James Bond et les placer dans votre dossier
- créer dans le code JavaScript :
  - une variable `numImage` initialisée à 1
  - une liste contenant les noms des différentes photos (la première étant celle affichée par défaut)
- créer la fonction JavaScript `carrousel()` qui :
  - affiche la photo d'indice `numImage` (en utilisant la fonction `getElementById()`)
  - incrémente `numImage`
  - si `numImage` est égal au nombre de photos, lui redonne 0 comme valeur (un modulo pourrait aussi être utilisé) :  
`if ( ... ) { numImage = 0; }`

## 9 Cours AJAX

L'acronyme **AJAX** (Asynchronous Javascript And XML) désigne une fonctionnalité maintenant ancienne de JavaScript qui permet de récupérer d'un serveur – de celui qui a délivré la page ou d'un serveur tiers - des **données** (et non du HTML) qui permettent de mettre à jour une page sans modifier sa morphologie (un exemple classique est de faire apparaître le nom d'une ville suite à la spécification par l'internaute de son adresse postale).

Les données importées étaient initialement formatées en **XML** (Extensible Mark-up Language), mais sont de plus en plus formatées en **JSON** (JavaScript Objet Notation) qui correspond à la sérialisation (c'est à dire la notation textuelle) d'objets JavaScript.

Nous allons utiliser la bibliothèque JavaScript JQuery pour avoir un code plus concis.

*Attention : utilisez préférentiellement Firefox pour exécuter du code JavaScript AJAX (Chrome ou Chromium ont besoin de permissions spéciales pour le faire).*

### 9.1 Principe de l'importation de données (AJAX)

#### 9.1.1 Fichier de données JSON

Des données doivent être créées dans un fichier texte.

Pour les formater, la structure la plus usuelle (mais pas la seule) est de créer une liste d'objets :

```
[{
  "<nom de propriété>" : <valeur de propriété>,
  "<nom de propriété>" : <valeur de propriété>,
  ...
},
...
]
```

Comme nous l'avons vu, une liste est encadrée par des crochets, et un objet par des accolades.

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

Voici la structure d'une fonction JavaScript qui appelée lors de la sélection du 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 += "<création d'un fragment HTML>";
        });
        $("<point d'attachement du nouveau code HTML>").append(html);
    });
}
```

\$ désigne l'objet de plus niveau de la bibliothèque JQuery : à cet objet sont attachées des fonctions (appelées, dans le contexte de la programmation par objets, méthodes).

\$.getJSON() est la méthode qui permet de télécharger un fichier JSON et d'allouer en mémoire la liste qui y est décrite (variable `data`).

\$.each() est la méthode qui permet de parcourir une liste et en extraire chaque objet (variable `objet`, la variable `index` qui numérote chaque objet à partir de 0 n'est pas utilisée dans cet exemple).

Les méthodes de la bibliothèques *JQuery* font couramment usage de fonctions de rappel (ou fonctions de callback). Ces fonctions sont des fonctions anonymes passées en paramètres à des fonctions qui les appellent en retour pour renvoyer des données (souvent de manière asynchrones).

### 9.1.3 Fonctions anonymes passées en paramètres : les fonctions de rappel

L'exemple précédent met en exergue que JavaScript - et ici JQuery - fait usage de fonctions de rappel (ou fonctions de callback) qui sont des fonctions anonymes (*lambda fonctions*) passées en paramètres à une fonction.

Pour illustrer ce point, voici ci-après deux exemples de l'implémentation d'une fonction/méthode `myForEach()` qui :

- soit prend en paramètre une liste
- soit est associée à une liste (la fonction est donc une méthode)

et prend en paramètre une fonction de rappel qui sera exécutée autant de fois que cette liste comprend d'éléments (en affichant l'indice et la valeur de chaque élément de la liste).

Première illustration de la mise en œuvre d'une fonction de rappel :

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

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

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

Seconde illustration de la mise en œuvre d'une fonction de rappel :

```
let liste = new Array(1, 2, 3, "partez !");

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

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

8

9

---

8. La fonction constructrice *Array* permet de créer des objets "liste" : la programmation objets et l'utilisation de fonctions constructrices ont été sommairement abordées précédemment.

9. L'objet *prototype* associé à la fonction constructrice *Array()* permet de mettre en place un mécanisme d'"héritage" entre objets qui ne sera pas abordé dans ce support de cours (c'est un point très sensible de JavaScript qui l'a fait détester ou adorer).

## 9.2 Exemple d'un code AJAX qui via JQuery charge des données complémentaires au CV de James

Modifions le code HTML du CV de James pour :

- insérer un lien hypertexte dans la rubrique "hobbies" qui va permettre d'invoquer la fonction JavaScript d'importation des données ;
- créer une balise <ul> identifiée par un identifiant qui nous permettra d'accrocher le code HTML formé à partir des données importées ;
- et sans oublier de créer un lien sur la bibliothèque JQuery dans le head de la page HTML.  
Cette bibliothèque peut être téléchargée du site JQuery : <https://code.jquery.com/>.

Voici les fragments de codes ainsi modifiés :

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF8" />
    <title> CV de James Bond </title>
    <link rel="stylesheet" href="styles_CV_James.css" />
    <script src="jquery-3.4.1.min.js" type="text/javascript"></script>
    <script src="CV_James_AJAX.js"></script>
  </head>

  <body>
    <header>
      <p> James Bond  </p>
      Nationalité britannique <br/>
      Profession : agent secret (dernier employeur : MI6)
    </header>

    <main>
      ...
      <article>
        <p> Hobbies : </p>
        <ul>
          <li> Dégustation de <a onclick="afficherAlcools()" href="#">nombreux alcools</a>
            <ul id="listeAlcools"></ul>
          </li>
          <li> Conversation mondaine en présence de public féminin </li>
        </ul>
      </article>
    </main>
  </body>
</html>
```

Voici maintenant le fichier JSON *alcools.json* qui liste quelques cocktails prisés par les différents alcooliques que nous connaissons bien :

```
[{"nom": "Vodka Martini",
  "amateurs": ["Sean Connery", "George Lazenby", "Roger Moore",
    "Timothy Dalton", "Pierce Brosnan", "Daniel Craig"]
},
{"nom": "Vesper Martini",
  "amateurs": ["Daniel Craig"]
},
{"nom": "Rum Collins",
  "amateurs": ["Sean Connery"]
},
{"nom": "Mint Julep",
  "amateurs": ["Sean Connery", "George Lazenby"]
}
]
```



Ce fichier décrit une liste (encadrée par des crochets) contenant des objets (encadrés par des accolades).

Voici la fonction JavaScript qui utilise du code JQuery pour importer le fichier de données, extraire de chaque objet le nom du cocktail, et l'intégrer dans du code HTML :

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

La chaîne de caractères html aura finalement la valeur suivante :

<li>Vodka Martini</li><li>Vesper Martini</li><li>Rum Collins</li><li>Mint Julep</li>

JQuery fait usage de fonctions de rappel (fonctions anonymes passées en paramètres) qui délivrent les données mises à disposition. Leur syntaxe n'est pas à maîtriser.

*Nous rappelons que L'ABUS D'ALCOOL EST DANGEREUX POUR LA SANTÉ, A CONSOMMER AVEC MODÉRATION.*

Et voici le résultat une fois le lien sélectionné :

CV de James Bond avec AJAX - Mozilla Firefox

CV de James Bond avec AJAX

file:///home/fongus/DUI/WEB/CV\_avec AJAX/CV\_James AJAX.html#

## 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 :**

- Dégustation de nombreux alcools
  - Vodka Martini
  - Vesper Martini
  - Rum Collins
  - Mint Julep
- Conversation mondaine en présence de public féminin

## 10 TP AJAX

Créez un fichier JSON contenant les données relatives à l'une des rubriques de votre CV.

Modifiez votre fichier HTML pour qu'à partir d'un lien, les informations qui étaient auparavant présentées sur une autre page soient directement insérées dans la page.

Pour ce faire, créez un fichier JavaScript contenant la fonction JavaScript qui va importer les données et créer le fragment HTML. Cette fonction sera calquée sur celle donnée en exemple.

# Index

<a>, 12  
<article>, 6  
<body>, 5  
<br/>, 6  
<center>, 6  
<div>, 6  
<footer>, 6  
<form>, 14  
<head>, 5  
<header>, 6  
<html>, 5  
<img>, 7  
<input>, 14  
<li>, 5  
<link>, 6  
<main>, 6  
<ol>, 5  
<option>, 14  
<p>, 5  
<section>, 6  
<select>, 14  
<span>, 6  
<table>, 5  
<td>, 5  
<textarea>, 14  
<th>, 10  
<tr>, 5  
<ul>, 5  
éditeur de style, 11  
élément (DOM), 19  
  
action (attribut), 14  
adresse web, 3  
AJAX, 22  
applications web, 3  
architecture client/serveur, 3  
Array(), 20  
attribut, 5  
  
balise, 4  
bibliothèque JavaScript, 19  
bloc d'instructions, 18  
  
checkbox, 14  
client, 3  
console, 11  
const (JavaScript), 19  
CSS, 6  
  
Document Object Model, 19  
DOM, 19  
  
feuilles de styles CSS, 6  
Fonction constructrice, 21  
Fonction de rappel, 23  
font-family, 7  
font-size, 7  
  
font-weight, 7  
for (... in ...), 21  
for (... of ...), 21  
formulaire, 5, 13  
function (JavaScript), 17  
  
gestionnaire d'événement, 16  
  
href (attribut), 12  
HTML, 4  
HTTP, 3  
  
inspecteur DOM, 11  
Internet, 3  
  
JavaScript, 16  
jQuery, 19  
JSON, 20, 22  
  
length (JavaScript), 20  
let (JavaScript), 17, 19  
lien hypertexte, 5, 12  
liste (JavaScript), 20  
  
méthode GET, 14  
méthode POST, 14  
margin (CSS), 11  
method (attribut), 14  
  
name, 15  
navigateur, 3  
new (JavaScript), 21  
  
objet (JavaScript), 18  
onclick (attribut), 16  
onload (attribut), 16  
onmouseout (attribut), 16  
onmouseover (attribut), 16  
onselect (attribut), 16  
onunload (attribut), 16  
outils du navigateur, 11  
  
password, 14  
  
réseau, 11  
radio, 14  
  
serveur, 3  
serveur web, 3  
style graphique, 7  
  
target (attribut), 12  
text, 14  
this (JavaScript), 17, 21  
type (attribut), 14  
TypeScript, 20  
  
URL, 3, 5

var (JavaScript), 17, 19  
variable, 17

web, 3  
World Wide Web, 3

XML, 22