PREMIERE GENERALE

NUMERIQUE ET SCIENCES INFORMATIQUES

INTERACTIONS ENTRE l'HOMME ET LA MACHINE SUR LE WEB : LANGAGE JAVASCRIPT

- ▷ **Histoire de l'informatique**
- ▷ Représentation des données : types et valeurs de base
- ▷ Représentation des données : types construits
- ▷ Traitement de données en tables
- ▷ **Interactions entre l'homme et la machine sur le Web**
- ▷ Architectures matérielles et systèmes d'exploitation
- ▷ Langages et programmation
- ▷ Algorithmique

JavaScript est un langage créé en **1995** par **Brendan Eich** qui travaille chez Netscape Communication Corporation principal concurrent à l'époque d'Internet Explorer. JavaScript est alors transmis à l'ECMA (European Computer Manufacturers Association) pour standardisation sous la dénomination d'ECMAScript abrégé en ES. JavaScript dont le nom est une référence à Java de Sun Microsystems, ne doit pas être confondu avec Java.

1 – FONCTIONNEMENT D'UN SITE WEB DE BOUT EN BOUT

1.1 – Le WEB

Le WEB est un service d'internet. Il s'agit de **l'ensembles des données** (**pages Web**) accessibles et modifiables à partir du réseau internet. Les pages des sites Web sont reliées entre elles via des **liens hypertextes** qui permettent de passer d'un site à un autre.

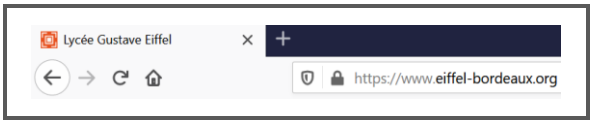
Le « Web 2.0 » est un concept d'un Web dans lequel les **utilisateurs sont en interaction**. A l'opposé du web dit « Web 1.0 », constitué seulement de pages consultables, le « Web 2.0 », favorise les échanges, la personnalisation des applications, la dimension sociale de l'Internet au travers notamment des réseaux sociaux et des sites collaboratifs. La prochaine évolution du Web, le « Web 3.0 », sera tournée vers la mobilité, les **objets connectés** et les données. Ce qui lui vaut d'ailleurs son appellation de **Web sémantique**.

1.2 – Site WEB

Un site Web est constitué de plusieurs ressources : pages HTML, feuilles de style CSS, scripts JavaScript, images, vidéo, son, etc. Ces ressources sont gérées par le serveur Web qui les rend accessibles via le **protocole HTTP**. Un utilisateur utilise son navigateur et interagit avec celui-ci. Le navigateur présente (affiche) les ressources à l'utilisateur. Selon les interactions réalisées par l'utilisateur, le navigateur interagit avec le serveur pour accéder à de nouvelles ressources et les afficher.

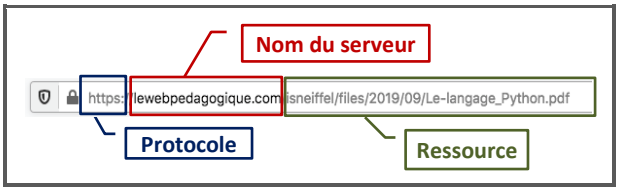
1.3 – URL

Pour accéder à un site Web, l'utilisateur doit de saisir l'**URL** (**U**niform **R**esource **L**ocator) du site dans la barre de navigation du navigateur.



Une URL c'est une chaîne de caractères. Elle est composée de 4 parties :

- **protocole :** http ou https.
- **nom du serveur web** (ou son adresse IP).
- **numéro du port** du serveur (par défaut 80).
- **nom de la ressource** à afficher.



1.4 – Le Protocole

Sur le web, seuls les **protocoles HTTP et HTTPS** sont utilisés. Ces protocoles sont standards. Ils précisent comment sont réalisés les échanges entre un navigateur et un serveur web. Le protocole HTTPS est la **version sécurisée** de HTTP. Il n'est pas toujours, nécessaire de saisir le type de protocole dans la barre de navigateur, le navigateur adaptera le protocole à l'adresse du serveur.

1.5 – Le nom du serveur

Un serveur Web est connecté en permanence à internet et possède donc une **adresse IP**. Pour accéder à un serveur il nécessaire de préciser cette adresse. Pour ne pas avoir à retenir son adresse IP, les serveur Web possèdent également nom. C'est le système DNS (Domain Naming Service) qui fait le lien entre une adresse IP et le nom des serveurs. Par exemple l'adresse IP d'un des serveurs « google.com » est 216.58.206.238.

Exercice 1

1. **Saisir** dans la barre de navigation de votre navigateur l'adresse IP du serveur Web de « google.com »

2. **Vérifier** que cette adresse permet d'accéder au site « google.com ».

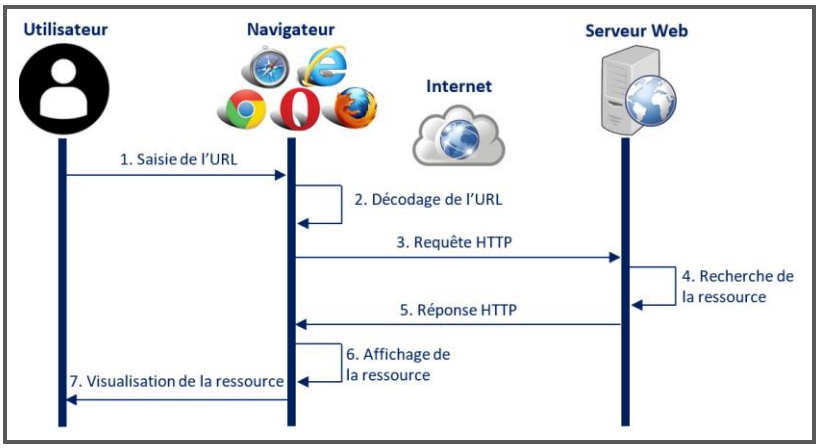
1.6 – Le nom de la ressource

Les serveurs Web disposent tous d'une ressource par défaut qui très souvent est la **page principale** du site Web « **index.html** ». Si seul le nom du serveur est saisi, le serveur transmettra cette page principale.

1.7 – Chargement de la ressource

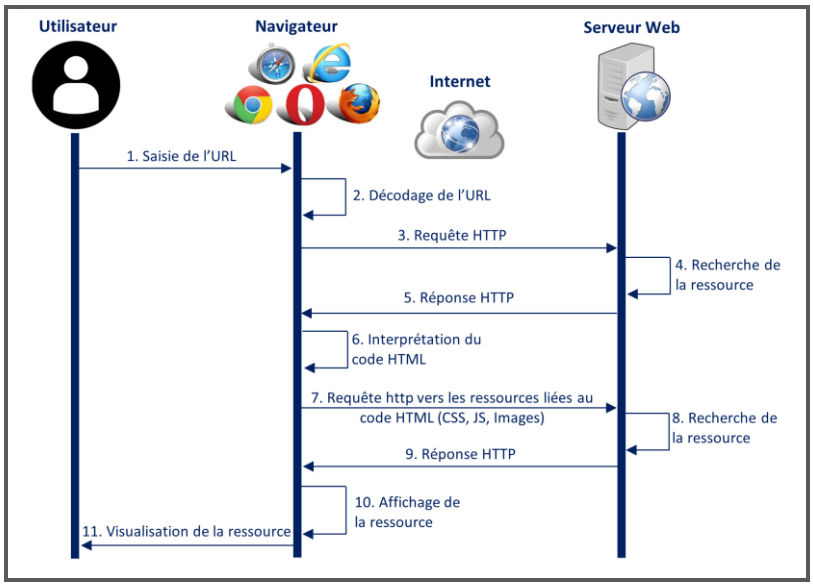
Une fois la saisie de l'URL réalisée, le navigateur va décoder l'URL afin d'envoyer une **requête HTTP** vers le serveur. Le serveur web reçoit cette requête et y répond en **envoyant la ressource correspondante** à l'URL. Dès qu'il aura reçu la réponse, le navigateur affiche la ressource afin que l'utilisateur puisse la voir.

Si les ressources sont des images ou des vidéos, le navigateur se contente de les afficher. Si les ressources sont des fichiers texte, le navigateur affiche le texte. Si les ressources sont des fichiers binaires (fichier .zip ou autre), le navigateur propose de les sauvegarder.



1.8 – Chargement d'une page HTML

Si le navigateur reçoit une page HTML, il va récupérer toutes les autres ressources liées à la page (feuille de style CSS, images, vidéo, script JavaScript). Pour ce faire, il va envoyer **plusieurs requêtes au serveur** et mettre à jour l'affichage. Le navigateur fera un premier affichage dès qu'il aura reçu la page HTML et les feuilles de styles CSS liées. En parallèle, il enverra les requêtes pour récupérer les autres éléments liés à la page et fera des mises à jour de l'affichage lorsqu'il les recevra.



2 – PROTOCOLE HTTP

2.1 – Protocole HTTP

Un site web est composé de plusieurs ressources : pages HTML, images, films, script JavaScript, feuilles CSS, etc. qui sont demandées au serveur par le navigateur Web auprès d'un serveur Web par l'intermédiaire d'un requête http.

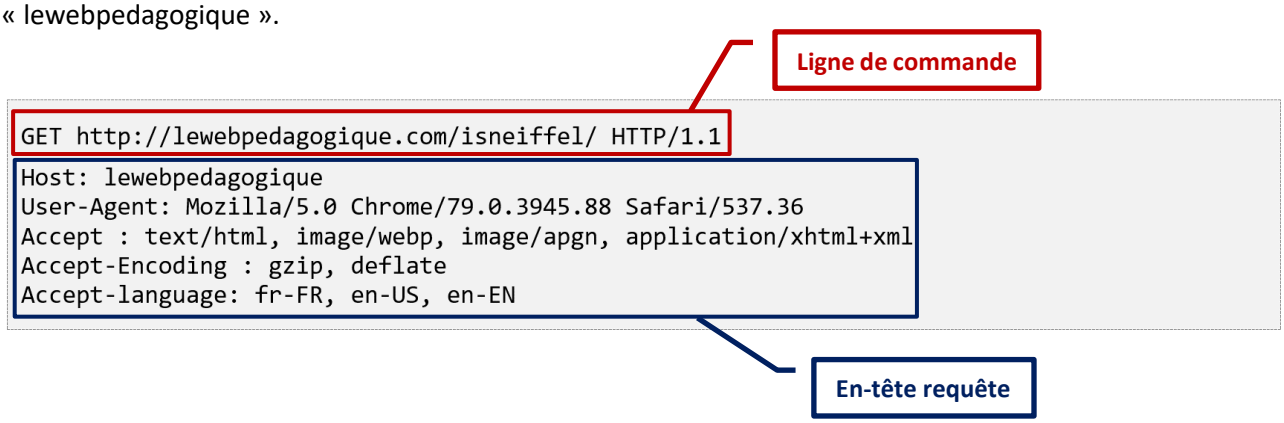
Le protocole HTTP définit la façon dont sont réalisés les requêtes et les réponses entre le navigateur et le serveur web. HTTP date de 1990 et la version 2.0 est sortie en 2015. Il s'agit d'un protocole requête / réponse : le client (le navigateur web) envoie une requête et le serveur web y répond.

2.2 – Requête HTTP

Une requête HTTP présente la syntaxe suivante :

```
Ligne de commande
En-tête de la requête
<Nouvelle ligne> (ligne vide)
Corps de la requête
```

L'exemple suivant présente la demande d'affichage de la page d'accueil du site « isneiffel » hébergé dans le serveur « lewebpedagogique ».



La ligne de commande contient :

- La **méthode** qui précise l'intention de la requête :
  - **GET** : pour obtenir une ressource (lire une page web),
  - **POST** : pour ajouter une information concernant une ressource,
  - **PUT** : pour ajouter une ressource,
  - **DELETE** : pour supprimer une ressource (cette commande est souvent interdite),
  - **PATCH** : pour modifier une ressource existante.
- L'**URL** qui cible la ressource.
- La **version** du protocole.

Dans l'exemple ci-dessus, il s'agit d'une requête **GET** qui cible la page d'accueil du site « **isneiffel** » hébergé par le serveur « **lewebpedagogique** ». La version du protocole http utilisée est la version **1.1**.

L'en-tête de la requête est composé de plusieurs champs (clé / valeur) parmi lesquels on trouve :

- **host** : **nom du domaine du site web**, ce qui est nécessaire quand le serveur web héberge plusieurs sites web (ce qui n'est pas rare),
- **User-Agent** : **nom du navigateur** web (ce qui permet d'envoyer du contenu adapté au navigateur).
- **Accept** : types de ressources acceptée par le navigateur.
- **Accept-Encoding** : types de ressources acceptée par le navigateur.
- **Accept-Language** : types de langues attendues par le navigateur.

Cet entête est ensuite suivi d'une **ligne vide** puis du **corps de la requête** qui contient des données envoyées avec la requête.

2.3 – Méthode GET ou POST

La **méthode GET** est la méthode utilisée par défaut. Il s'agit d'une simple requête de **demande d'une ressource**. Le corps d'une requête GET est vide. Il est toutefois possible de **transmettre des données dans l'URL** sous forme de paramètres. L'envoi de formulaires par la méthode GET pose deux grands problèmes :

- **Quantité limitée de données** car l'URL possède une taille limitée (2083 caractères).
- **Problème de sécurité** car les données sont visibles dans la barre d'URL et stockées dans l'historique du navigateur.

La **méthode POST** est utilisée pour **transmettre des données au serveur** afin qu'il les traite. Le plus souvent ces données sont récupérées dans le navigateur à partir d'un **formulaire HTML**. Ces données sont ajoutées dans le corps de la requête HTTP.

2.4 – Envoi d'une requête HTTP

L'envoi d'une requête peut se faire :

- En **HTML** grâce à la balise `<a>`. Si une page HTML contient une balise `<a>`dont l'attribut de lien `href` est une ressource externe à la page, alors un clic sur le lien émettra une requête GET par le navigateur avec le lien comme URL. Dans l'exemple ci-dessous, un clic sur « Lycée Gustave Eiffel » lancera une requête GET vers https://www.eiffel-bordeaux.org

```
<a href='https://www.eiffel-bordeaux.org'>Lycée Gustave Eiffel</a>
```

- En **HTML** grâce à la balise `<form>` : Si une page HTML contient un formulaire (balises `<form>` et `</form>`) alors, la soumission du formulaire émettra une requête par le navigateur. Le verbe de la requête correspondra à l'attribut `method` du formulaire. L'URL de la requête correspondra à l'attribut `action`. Dans l'exemple ci-dessous, remplir le formulaire et le soumettre émettra une requête GET vers l'URL « ./login ». Si le verbe est GET, les données du formulaire seront encodées dans l'URL. Dans le cas d'un POST elles seront encodées dans le corps de la requête.

```
<form method='GET' action='./login'>
   <input name='user' type='text' required>
   <input name='password' type='password' required>
   <button type='submit'>login</button>
</form>
```

- En **JavaScript** grâce à la fonction **`fetch`**. Cette fonction permet d'envoyer des requêtes et de recevoir des réponses. Cette méthode prend comme argument l'URL de la requête et, optionnellement, un objet qui permet de préciser les options de la requêtes (son verbe, son en-tête, etc.).

2.5 – Réponse à une requête HTTP

La syntaxe de la réponse envoyée par le serveur vers le client présente la syntaxe suivante :

```
Ligne de statut
En-tête de la réponse
<Nouvelle ligne> (ligne vide)
Corps de la réponse
```

La ligne de Statut contient, la version du protocole HTTP, le code de la réponse et un message explicatif. Par exemple, la réponse suivante indique la version du protocole HTTP utilisé est la **version 1.1** et que la **requête a été traitée avec succès**.

```
HTTP/1.1 200 OK
```

Le code de la réponse est normalisé est peut prendre les valeurs suivantes (voir la liste des valeurs possibles pour ce code) :

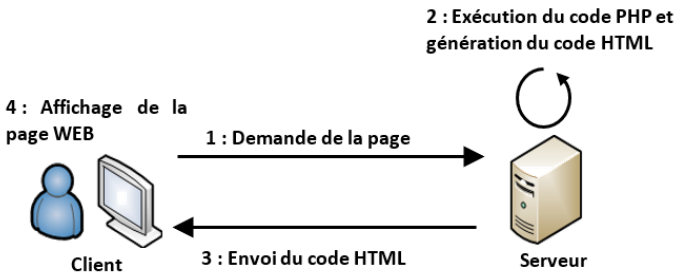
- **1xx** : la requête est en cours de traitement.
- **2xx** : la requête est traitée : succès (**200**), créée (**201**), etc.
- **3xx** : la requête a été transférée vers un autre serveur : déplacée définitivement sur un autre serveur (**301**), pas modifiée donc le cache est à jours (**304**), etc.
- **4xx** : erreur client : pas autorisé (**401**), interdit (**403**), (404) pas trouvé (**404**), etc.
- **5xx** : erreur serveur : pas implanté (**501**), service down (**503**), etc.

3 – LE LANGAGE JAVASCRIPT

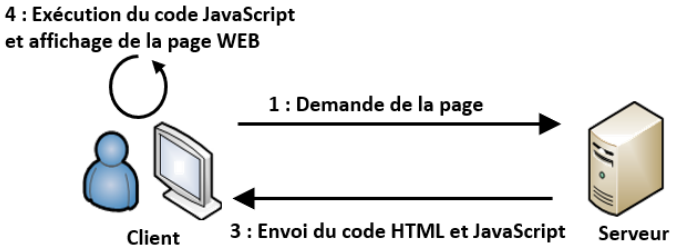
3.1 – Présentation

Les pages créées à l'aide des langages HTML et CSS sont des **pages WEB statiques** c'est-à-dire que l'interaction avec l'utilisateur est réduite à la possibilité de cliquer sur un lien hypertexte présent dans la page. Il est possible de rendre une page internet dynamique, de deux manières :

- soit du **côté serveur** avec le **langage PHP** (ou Hypertext Preprocessor) qui peut, par exemple, ajouter le résultat d'une requête à une base de données dans la page qui sera fournie au navigateur ;



- soit du **côté client** avec **JavaScript** qui peut, par exemple, faire apparaître des info-bulles contextuelles ou réaliser des animations.



Le langage JavaScript est donc un langage dit **client-side**, c'est-à-dire que les scripts sont exécutés par le navigateur chez l'internaute (le **client**).

D'un point de vue historique, JavaScript a été principalement défini pour être exécuté dans un navigateur web. Pour autant, depuis quelques années, il a gagné en popularité et est aujourd'hui utilisé dans bien d'autres contextes (programmation côté serveur, scripts et macros dans des applications, etc.).

JavaScript est un **langage interprété**. Cela veut dire qu'il est exécuté par un interpréteur sans qu'une phase de compilation soit nécessaire. **NodeJS** est l'interpréteur le plus populaire de JavaScript. Tout navigateur web dispose d'un interpréteur JavaScript et peut ainsi exécuter du code JavaScript. Lorsqu'une page HTML référence du code JavaScript, celui-ci est téléchargé par le navigateur et interprété par l'interpréteur.

3.2 – Intégration de code javascript dans une page HTML

Le code javascript peut être intégré directement dans le document html grâce à la balise `<script>` au sein de l'entête (entre les balises `<head>` et `</head>`). Il peut également être placé dans le corps (entre les balises `<body>` et `</body>`) de la page HTML.

Exercice 2

1. **Lancer** le logiciel **Visual Studio Code**. **Créer** un nouveau document nommé **`exercice2.html`**. **Editer** le code ci-après. **Sauvegarder** le fichier.

Exercice 2

exercice2.html :

```
<html>
   <head>
      <meta charset="utf-8" />
      <title>Un peu de javascript</title>
      <script type="text/javascript" charset="utf-8">
         // Je suis un commentaire javascript!
   // Alert crée une fenêtre d'affichage sur l'écran pour l'utilisateur (popup)
         alert('Bienvenue en NSI !') ;
      </script>
   </head>
   <body>
      <h1>Découverte du langage JavaScript</h1>
      <p>Ma première page dynamique (quoique !!!).</p>
   </body>
</html>
```

2. **Ouvrir** le fichier **`exercice2.html`** à l'aide d'un navigateur WEB. **Vérifier** l'affichage obtenu.

Le code javascript peut être également intégré dans un fichier séparé possédant l'extension **`.js`**. Il est alors nécessaire de l'importer dans la page html à l'aide de l'attribut `src` la balise `<script>`.

Exercice 3

1. **Editer** le code html et le code JavaScript ci-dessous. **Sauvegarder** les fichiers.

exercice3.html :

```
<html>
   <head>
      <meta charset="utf-8" />
      <title>Un peu de javascript</title>
      <script src="scriptEx3.js"></script>
   </head>
   <body>
      <h1>Découverte du langage JavaScript</h1>
      <p>Ma deuxième page dynamique (quoique !!!).</p>
   </body>
</html>
```

scriptEx3.js :

```
alert('Bienvenue en NSI !') ;
```

2. **Ouvrir** le fichier **`exercice3.html`** à l'aide d'un navigateur WEB. **Vérifier** que l'affichage obtenu correspond à celui attendu.

3.3 – Syntaxe de JavaScript

Voici quelques bases de la syntaxe de JavaScript (site Mozilla **Developper Network MDN**).

- Les variables doivent être déclarées grâce au mot clé `var`.
- Tout comme en python, les **chaînes de caractères** sont entourées de **guillemets**.
- Les instructions simples sont terminées par un **point-virgule ;**
- Les **blocs d'instructions** sont entourés d'**accolades**.
- Les **commentaires** sont notés précédés de deux barres obliques: `//`.
- L'indentation des blocs d'instruction n'est pas obligatoire comme en Python, mais souhaitable.

4 – MODIFIER UNE PAGE WEB AVEC JAVASCRIPT

Exercice 4

1. **Créer** dans, **Visual Studio Code**, un nouveau document nommé **`exercice4.html`** et **éditer** le code HTML ci-dessous. **Sauvegarder** le fichier.

exercice4.html :

```
<html>
   <head>
      <meta charset="UTF-8">
      <link rel="stylesheet" href="style.css">
      <title>Exercice 4</title>
      <script>
         var prenom = prompt('Quel est votre prénom ?');
         alert('Bonjour ' + prenom);
      </script>
   </head>
   <body>
      <h1>Page de salutations</h1>
   </body>
</html>
```

2. **Créer** un nouveau document nommé **`style.css`** et **éditer** le code afin d'avoir le titre **`h1`** en rouge et centré.
3. **Ouvrir** le fichier **`exercice4.html`** à l'aide du navigateur WEB.
4. **Vérifier** et **justifier** l'affichage obtenu.

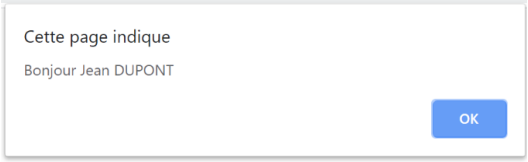
Exercice 5

1. **Editer** le code html ci-dessous. **Sauvegarder** le fichier.

Exercice5.html :

```
<html>
   <head>
      <meta charset="UTF-8">
      <link rel="stylesheet" href="style.css">
      <title> Exercice 5</title>
      <script src="scriptEx5.js"></script>
   </head>
   <body>
      <h1>Page de salutations bis</h1>
   </body>
</html>
```

2. **Créer** un nouveau document nommé **`scriptEx5.js`** et **éditer** le code afin de demander à l'utilisateur son prénom et son nom et de le saluer.



3. **Ouvrir** le fichier **`exercice5.html`** à l'aide d'un navigateur WEB.
4. **Vérifier** et **justifier** l'affichage obtenu.

Exercice 5

5. **Ajouter** les lignes de code suivantes au fichier **scriptEx5.js**.

```
var age = prompt('Quel est votre age');
var age_futur = age + 1 ;
alert('Dans un an vous aurez ' + age_futur + ' ans');
```

6. **Actualiser** la page WEB (Touche F5) et **justifier** pourquoi l'affichage obtenu n'est pas satisfaisant.

7. **Modifier** la troisième ligne du code javascript de la manière suivante.

```
var age = parseInt(prompt('Quel est votre age'));
```

8. **Actualiser** la page WEB et **vérifier** que l'affichage obtenu est, cette fois-ci, satisfaisant. **Donner** le rôle de la fonction **parseInt**.

5 – GESTION D'EVENEMENTS AVEC JAVASCRIPT

5.1 – Fonctions

Comme dans les autres langages, il peut être intéressant de placer les instructions dans des fonctions. Pour cela il faut utiliser le mot clé `function`. Ces fonctions sont particulièrement intéressantes pour la gestion des évènements.

Exercice 6

1. **Editer** le code html et le code JavaScript ci-dessous. **Sauvegarder** les fichiers.

scriptEx6.js :

```
function salutation() {
   var prenom = prompt('Quel est votre prénom ?');
   var nom = prompt('Quel est votre nom ?');
   alert('Bonjour ' + prenom + ' ' + nom);
   var age = parseInt(prompt('Quel est votre age'));
   var age_futur = age + 1 ;
   alert('Dans un an vous aurez ' + age_futur + ' ans');
}
```

exercice6.html :

```
<html>
   <head>
      <meta charset="UTF-8">
      <link rel="stylesheet" href="style.css">
      <script src="scriptEx6.js"></script>
      <title>Exercice 6</title>
   </head>
   <body onload="salutation();">
      <h1>Encore une page de salutations</h1>
   </body>
</html>
```

2. **Ouvrir** le fichier **exercice6.html** à l'aide d'un navigateur WEB. **Vérifier** et **justifier** l'affichage obtenu.

3. **Préciser** à quel moment est exécutée la fonction javascript **salutation()**.

5.2 – Evènements

Un évènement est par exemple le chargement d'une page, le passage du pointeur sur la page ou un élément de la page, un clic de la souris. Un événement permet de déclencher l'exécution d'une fonction. Pour cela un évènement est ajouté à un élément de la page, une balise par exemple, et lorsque l'utilisateur agit sur l'évènement d'une certaine manière, il déclenche le code JavaScript.

Dans l'exercice précédent, la fonction JavaScript **`salutation()`** est exécutée au chargement de la page grâce à l'attribut **`onLoad`** correspondant à l'évènement **`Load`** (Chargement).

Il existe un certain nombre d'évènements dont les principaux sont les suivants :

`Load`	Chargement d'un page
`Click`	Clic de la souris, sélection d'un élément
`Dblclick`	Double-clic sur l'élément
`Mouseover`	Entrée du pointeur sur un élément
`Mouseout`	Sortie du pointeur de l'élément
`Keypress`	Appui sur une touche produisant un caractère
`Focus`	Ciblage de l'élément
`Blur`	Annulation du ciblage de l'élément
`Submit`	Envoi d'un formulaire
`Reset`	Réinitialisation d'un formulaire
`Change`	Modification de la valeur d'un élément d'un formulaire

5.3 – Gestion d'un évènement

L'utilisateur déclenche un évènement qui réagit en appelant une fonction JavaScript. Le code s'exécute et provoque une modification dans la page. Les attributs associés aux éléments sont formés du mot « on » et du nom de l'événement : `onLoad`, `onClick`, `onMouseover`…

Exercice 7

1. **Editer** le code HTML et le code JavaScript ci-dessus. **Sauvegarder** les fichiers.

exercice7.html :

```
<html>
   <head>
      <meta charset="UTF-8">
      <script src="scriptEx7.js"></script>
      <title>Exercice 7</title>
   </head>
   <body>
      <h1 onClick="message()">Cliquez ici</h1>
      <h2 onDblclick="message()">Double-cliquez ici</h2>
      <h3 onMouseover="message()">Approchez le curseur</h3>
      <h3 onMouseout="message()">Eloignez le curseur</h3>
      <p>Tapez sur une touche</p><input type="text" onkeypress="message()">
   </body>
</html>
```

scriptEx7.js :

```
function message() {
   alert("Un évènement a été détecté");
}
```

Exercice 7

2. **Ouvrir** le fichier **exercice7.html** à l'aide d'un navigateur WEB.
3. **Testez** les différentes possibilités de détection d'un évènement.

6 – RECUPERATION DES DONNEES UTILISATEUR

La balise html `<input>` permet de récupérer les entrées utilisateur, il en existe de divers types :

- `<input type="text"…>` : entrée de texte (par défaut).
- `<input type="button"…>` : bouton.
- `<input type="checkbox"…>` : case à cocher.
- `<input type="radio"…>` : bouton radio qui permet de sélectionner une seule valeur parmi un groupe de différentes valeurs.

Exercice 8

1. **Editer** le code HTML et le code JavaScript ci-dessus. **Sauvegarder** les fichiers.

exercice8.html :

```
<html>
   <head>
      <meta charset="utf-8" />
      <title>Exercice 8</title>
      <script src="scriptEx8.js"></script>
   </head>
   <body>
      <!-- Zone de texte -->
      <input id="id1" type="text" placeholder="Taper votre texte ici" />
      <!-- Bouton à cliquer -->
      <input type="button" onclick="afficheEntrée();" value="Clique ici" />
   </body>
</html>
```

scriptEx8.js :

```
function afficheEntrée() {
   // On récupère la valeur du texte rentré sur l'élément input avec l'id "id1"
   const texte = document.querySelector('#id1').value
   alert('Affichage avec alert: ' + texte)
}
```

2. **Ouvrir** le fichier **exercice8.html** à l'aide d'un navigateur WEB.
3. **Vérifier** et **justifier** l'affichage obtenu.

8 – MODIFIER UN ELEMENT HTML

8.1 – Modifier le contenu

Il existe plusieurs méthodes pour avoir accès à un élément HTML. La méthode **querySelector()**, appliquée à partir de la racine document, permet d'accéder au premier élément rencontré.Pour modifier le contenu d'un élément il faut utiliser la propriété **innerHTML**.

Exercice 9

1. **Editer** le code HTML et le code JavaScript ci-dessus. **Sauvegarder** les fichiers.

exercice9.html :

```
<html>
   <head>
      <meta charset="utf-8" />
      <title>Exercice 9</title>
      <script src="scriptEx9.js"></script>
   </head>
   <body>
      <h1> Modification du contenu d'un élément HTML</h1>
      <p id="texte"> Ce texte ne me plait pas, je vais le modifier </p>
      <input type="button" onclick="modifTexte();" value="Cliquer ici pour modifier le
texte"/>
   </body>
</html>
```

scriptEx9.js :

```
function modifTexte() {
   document.querySelector('#texte').innerHTML = "Je n'aimais pas l'ancien
   texte,  je l'ai changé."
   }
```

2. **Ouvrir** le fichier **`exercice9.html`** à l'aide d'un navigateur WEB.
3. **Vérifier** et **justifier** l'affichage obtenu.

8.2 – Modifier le style

Pour modifier les propriétés de style d'un élément HTML il faut utiliser la propriété `style`.

Exercice 10

1. **Editer** les codes HTML, CSS et JavaScript ci-dessus. **Sauvegarder** les fichiers.

exercice10.html :

```
<html>
   <head>
      <meta charset="utf-8" />
      <title>Exercice 10</title>
      <link rel="stylesheet" href="style.css">
      <script src="scriptEx10.js"></script>
   </head>
   <body>
      <h1 id="titre"> Modification du style d'un élément HTML </h1>
      <p id="texte"> Ces couleurs ne me plaisent pas, je vais les modifier </p>
      <input type="button" onclick="modifStyle();" value="Clique ici pour modifier les
couleurs" />
   </body>
</html>
```

style.css :

```
h1{color: red;text-align: center;}
p{color: blue;text-align: justify;}
```

Exercice 10

scriptEx10.js :

```
function modifStyle() {
   document.querySelector('#titre').style.color = "#17657D"
   document.querySelector('#texte').style.color = "#B233FF"
   }
```

2. **Ouvrir** le fichier **exercice10.html** à l'aide d'un navigateur WEB.
3. **Vérifier** et **justifier** l'affichage obtenu.

Il est possible de travailler plus "proprement" en utilisant les classes CSS.

Exercice 11

1. **Editer** les codes HTML, CSS et JavaScript ci-dessus. **Sauvegarder** les fichiers.

exercice11.html :

```
<html>
   <head>
      <meta charset="utf-8" />
      <title>Exercice 11</title>
      <link rel="stylesheet" href="style.css">
      <script src="scriptEx11.js"></script>
   </head>
   <body>
      <h1> Modification du style d'un élément HTML </h1>
      <p id="texte"> Ces couleurs ne me plaisent pas, je vais les modifier </p>
      <input type="button" onclick="foncRouge();" value="Rouge" />
      <input type="button" onclick="foncVert();" value="Vert" />
   </body>
</html>
```

style.css :

```
h1{ text-align: center;}
.rouge { color:red; font-size:20px;}
.vert { color:green; font-size:30px;}
```

scriptEx11.js :

```
function foncRouge() {
   document.querySelector("#texte").classList.remove("vert");
   document.querySelector("#texte").classList.add("rouge");
}
function foncVert() {
   document.querySelector("#texte").classList.remove("rouge");
   document.querySelector("#texte").classList.add("vert");
}
```

2. **Ouvrir** le fichier **exercice11.html** à l'aide d'un navigateur WEB.
3. **Vérifier** et **justifier** l'affichage obtenu.

Exercice 12

1. **Editer** les codes HTML, CSS et JavaScript ci-dessus. **Sauvegarder** les fichiers.

exercice12.html :

```
<html>
   <head>
      <meta charset="utf-8" />
      <title>Exercice 12</title>
      <link rel="stylesheet" href="style.css">
      <script src="scriptEx12.js"></script>
   </head>
   <body>
      <h1> Modification du style d'un élément HTML </h1>
      <div onmouseover="foncEntre()" onmouseout="foncQuitte()" id="id1"><p>
      Survolez moi</p></div>
   </body>
</html>
```

style.css :

```
h1{ text-align: center; }
p{ text-align : center; }
#id1 {  width : 200px; height : 100px;
        margin : 0 auto; border : 2px solid black; }
.rouge { background-color:red; }
.blanc { background-color : white; }
```

scriptEx12.html :

```
function foncEntre(){
   document.querySelector("#id1").classList.remove("blanc");
   document.querySelector("#id1").classList.add("rouge");
}
function foncQuitte() {
   document.querySelector("#id1").classList.remove("rouge");
   document.querySelector("#id1").classList.add("blanc");
}
```

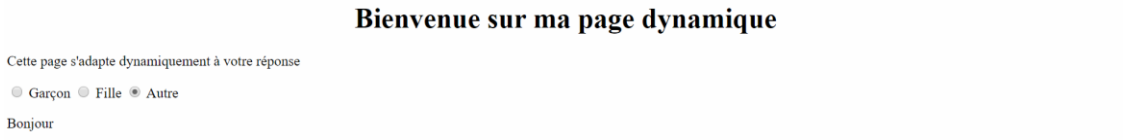
2. **Ouvrir** le fichier **`exercice12.html`** à l'aide d'un navigateur WEB.
3. **Vérifier** et **justifier** l'affichage obtenu.

9 – EXERCICE

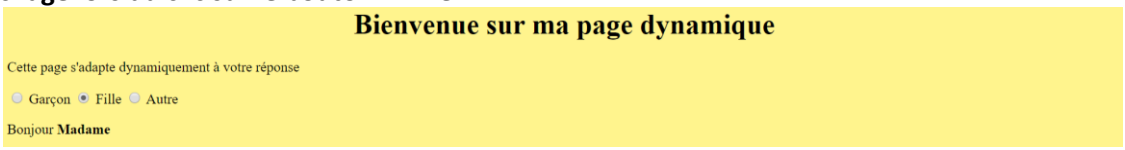
Exercice 13

On veut réaliser une page dynamique avec le fonctionnement suivant :

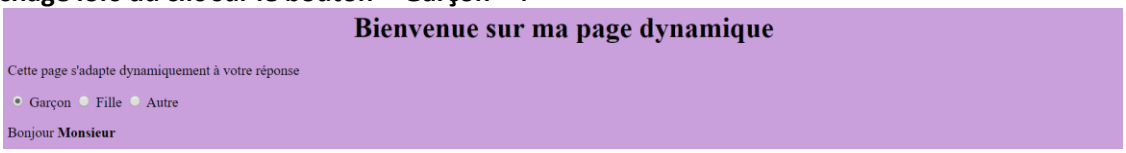
- **Affichage au démarrage ou lors du clic sur le bouton « Autre »** :



- **Affichage lors du clic sur le bouton « Fille »** :



- **Affichage lors du clic sur le bouton « Garçon » :**



1. **Editer** les fichiers HTML, CSS et JavaScript permettant d'obtenir la page dynamique ci-dessus.
2. **Vérifier** le fonctionnement de la page WEB.