

Université de Sfax



République Tunisienne
Ministère de l'Enseignement
Supérieur et de la Recherche
Scientifique

Ecole Nationale d'Electronique
et des Télécommunications de
Sfax



Département Mathématique et Informatique

Décisionnelle

Niveau : 2-IDSD

Cours Développement Web Dynamique

Préparé par

Sonda Ammar

Maître Assistant à l'Ecole Nationale d'Electronique et des Télécommunications de Sfax

Email : sonda.ammar@enetcom.usf.tn

Sommaire

AVANT-PROPOS	1
CHAPITRE I : LE LANGAGE HTML	2
I. INTRODUCTION	3
II. HISTORIQUE	3
III. LES BALISES ET LES ATTRIBUTS	4
IV. STRUCTURE D'UNE PAGE WEB.....	5
V. CREATION DES LIENS HYPERTEXTES	6
1. ATTRIBUTS DE LA BALISE <A>	6
2. TYPES DE LIENS HYPERTEXTES	6
VI. INSERTION D'ELEMENTS HYPERMEDIA.....	7
1. LES IMAGES	7
2. LES FIGURES	8
3. LES ELEMENTS AUDIO	8
4. LES ELEMENTS VIDEO	9
VII. CREATION DE TABLEAUX	9
1. LES TABLEAUX STRUCTURES	9
2. FUSION DE CELLULES D'UN TABLEAU	10
VIII. CREATION DES FORMULAIRES	10
1. LES ZONES DE SAISIE	10
2. LES ELEMENTS D'OPTIONS	11
3. REGROUPEMENT DE CHAMPS	13
4. LES BOUTONS D'ENVOI	13
IX. CREATION DES IMAGES REACTIVES	13
CHAPITRE II : LE STYLE DE PRESENTATION CSS.....	16
I. INTRODUCTION	17
II. HISTORIQUE	17
III. DEFINITION DE STYLE CSS	18
IV. INSERTION DE CODE CSS DANS UN DOCUMENT HTML.....	19
1. AJOUT D'UNE CSS A UNE BALISE HTML (ATTRIBUT STYLE).....	19
2. AJOUT DE CSS A UNE PAGE (BALISE STYLE).....	19
3. LIER UN SITE A UN FICHIER CSS EXTERNE (BALISE LINK)	20
V. APPLIQUER UN STYLE A UNE OU PLUSIEURS BALISES.....	20
1. APPLIQUER UN STYLE A PLUSIEURS BALISES	20
2. APPLIQUER UN STYLE AVEC DES CLASSES ET DES IDS	21
3. APPLIQUER UN STYLE A UNE BALISE QUI SUIV UNE AUTRE	21
4. APPLIQUER UN STYLE A UNE BALISE QUI POSSEDE UN ATTRIBUT	21
VI. LES PROPRIETES ET LES VALEURS	21
1. FORMATAGE DU TEXTE	22
2. LA COULEUR ET LE FOND :	22
3. LES BORDURES ET LES OMBRES	22
4. TRANSPARENCE ET OPACITE	23
5. ARRIERE-PLANS MULTIPLES	23
6. LES DEGRADES.....	24
7. LES ELEMENTS FLOTTANTS	25
CHAPITRE III : LE LANGAGE JAVASCRIPT	26

I.	INTRODUCTION	27
II.	INTEGRATION DE JAVASCRIPT DANS HTML	27
III.	LES PREMIERES METHODES JAVASCRIPT POUR L'INTERACTION UTILISATEUR	27
1.	METHODE WRITE() :	27
2.	METHODE ALERT() :	28
3.	METHODE PROMPT() :	28
4.	METHODE CONFIRM() :	28
IV.	LES ELEMENTS FONDAMENTAUX DU LANGAGE JAVASCRIPT	28
1.	LES COMMENTAIRES EN JAVASCRIPT	28
2.	LES VARIABLES EN JAVASCRIPT	29
3.	LA PORTEE DES VARIABLES	30
4.	LA CONCATENATION DES VARIABLES	31
V.	LES TABLEAUX	31
VI.	LES FONCTIONS	33
1.	LES FONCTIONS PRE-PROGRAMMEES	33
2.	LES FONCTIONS UTILISATEUR	34
VII.	LE CONCEPT OBJET EN JAVASCRIPT	34
VIII.	LA GESTION DES EVENEMENTS	35
	EXEMPLES :	37
	EXERCICE :	39

CHAPITRE IV : LE PHP..... 42

I.	INTRODUCTION	43
II.	LA VITALITE PERSISTANTE DE PHP DANS LE PAYSAGE DU DEVELOPPEMENT WEB.....	43
III.	LE FONCTIONNEMENT DU WORLD WIDE WEB	44
IV.	LES SCRIPTS PHP	45
1.	LES VARIABLES ET LES CONSTANTES	45
2.	LES CHAINES DE CARACTERES.....	46
3.	LES TABLEAUX	47
V.	LES FONCTIONS	48
VI.	INCLURE DES BIBLIOTHEQUES OU DES FICHIERS.....	49
VII.	RECUPERATION DES DONNEES D'UN FORMULAIRE	50
VIII.	LES SESSIONS	50
1.	OUVERTURE D'UNE SESSION	51
2.	ENREGISTREMENT DES VARIABLES DE SESSION	51
3.	UTILISATION DES VARIABLES DE SESSION	51
4.	SUPPRESSION DES VARIABLES DE SESSION	51
5.	SUPPRESSION DE LA SESSION	52
IX.	PHP ET MYSQL.....	52
	EXERCICE 1 :	56
	EXERCICE 2 :	58
	REFERENCES	61

AVANT-PROPOS

Ce cours a pour but l'exploration des langages fondamentaux utilisés pour créer des sites web interactifs et dynamiques. Ce cours vise à fournir aux étudiants une solide compréhension des langages front-end tels que HTML (HyperText Markup Language), CSS (Cascading Style Sheets) et JavaScript, ainsi que du langage back-end PHP (PHP : Hypertext Preprocessor). À la fin de ce cours, les étudiants doivent maîtriser les bases de HTML, ils seront capables de styliser efficacement les éléments HTML avec CSS, et comprendront les concepts fondamentaux de la programmation JavaScript pour rendre les pages web interactives et réactives. De plus, ils seront initiés au développement côté serveur avec PHP, apprenant à créer des pages web dynamiques, à gérer les formulaires et à interagir avec les bases de données. En intégrant ces langages, ils seront en mesure de développer des applications web complètes et fonctionnelles. Enfin, tout au long de ce cours, ils auront l'occasion de mettre en pratique leurs connaissances à travers des projets pratiques, pour développer une expérience pratique du développement web.

CHAPITRE I : Le langage HTML

I. Introduction

Le HTML, acronyme pour HyperText Markup Language, est un langage fondamental pour créer des pages web. Il repose sur une syntaxe de balisage simple et intuitive, où les éléments du contenu sont encadrés par des balises pour les identifier et les structurer. L'objectif principal de HTML est de définir la structure et le contenu d'une page web, permettant ainsi de décrire son apparence et son fonctionnement. Grâce à HTML, il est possible d'inclure une variété d'informations dans une page web, notamment du texte, des images, des vidéos, des liens hypertexte et d'autres types de médias. Les balises HTML fournissent des instructions aux navigateurs web sur la manière d'afficher et d'interpréter le contenu, ce qui permet aux utilisateurs de naviguer facilement à travers les documents. En outre, HTML facilite l'établissement de relations entre les documents en permettant l'insertion de liens hypertexte vers d'autres pages web ou ressources en ligne. Cela favorise la navigation fluide des utilisateurs d'un document à un autre, ce qui contribue à la connectivité et à l'accessibilité du contenu sur le web.

II. Historique

Depuis les débuts du langage HTML dans les années 1990, il a connu plusieurs versions majeures, chacune apporte des améliorations significatives et des fonctionnalités nouvelles pour répondre aux besoins évolutifs du web. Voici un aperçu des principales versions de HTML :

- **HTML 1** : La première version de HTML est apparue en 1991. À cette époque, il s'agissait d'une version simple qui définissait les bases du langage de balisage pour la création de documents hypertextes. Les premiers éléments du langage HTML comprennent, le titre du document, les hyperliens, la structuration du texte en titres, sous-titres, listes ou texte brut, etc.
- **HTML 2** : La deuxième version de HTML est proposée en 1994. Elle a introduit des fonctionnalités telles que les tables, permettant de structurer les données de manière tabulaire, ainsi que de nombreux éléments de présentation tels que le positionnement du texte autour des images, améliorant ainsi la mise en page des pages web, le clignotement, le centrage, etc.
- **HTML 3** : En 1996, HTML 3 est apparu, offrant de nouvelles possibilités telles que la prise en charge des feuilles de style en cascade (CSS), qui permettent de séparer la

présentation du contenu. Cela a ouvert la voie à des designs plus flexibles et esthétiques pour les sites web.

- **HTML 4** : Lancé en 1998, HTML 4 a introduit des fonctionnalités avancées telles que l'utilisation de cadres (frames) pour diviser une page en plusieurs zones indépendantes, ainsi que des améliorations sur les formulaires et les tableaux. Il a également permis une meilleure gestion de la présentation grâce à l'exploitation des feuilles de style.
- **HTML 5** : La version la plus récente et la plus avancée, HTML 5, est sortie en 2014. Elle a apporté un grand nombre d'améliorations, notamment la possibilité d'inclure facilement des éléments multimédias tels que des vidéos et de l'audio, un meilleur arrangement du contenu avec de nouvelles balises sémantiques, et des fonctionnalités étendues pour les formulaires, le stockage local et la géolocalisation.

III. Les balises et les attributs

Les balises est l'un des éléments fondamentaux du langage HTML. Elles sont utilisées pour définir la structure et le contenu des documents web en utilisant des délimiteurs '<' et '>'. Les balises HTML se caractérisent par :

- **Balises en paires** : Les balises en paires sont des éléments qui s'ouvrent avec une balise ouvrante, contiennent du texte ou d'autres balises, puis se ferment avec une balise fermante correspondante.
 - Par exemple, la balise `<p>` est utilisée pour indiquer le début d'un paragraphe, et `</p>` pour le fermer.
- **Balises orphelines** : Contrairement aux balises en paires, les balises orphelines ne nécessitent pas de balise fermante. Elles sont souvent utilisées pour insérer des éléments spécifiques à un endroit précis dans le document, tels que des images ou des liens.
 - Par exemple, la balise `` est utilisée pour insérer une image et ne nécessite pas de balise fermante.
- **Les attributs** : Les attributs sont des options associées aux balises qui permettent de fournir des informations supplémentaires sur les éléments HTML. Ils complètent les balises en leur donnant des caractéristiques spécifiques. Les attributs sont placés à l'intérieur de la balise ouvrante et ont généralement une valeur associée. Chaque attribut se compose d'un nom et éventuellement d'une valeur.

- Par exemple, dans la balise ****, l'**attribut src** est utilisé pour spécifier l'emplacement de l'image à afficher. Il est défini comme suit : ****, où "chemin/vers/image.jpg" est la valeur de l'attribut src.

IV. Structure d'une page Web

La structure d'une page HTML est composée de plusieurs balises qui organisent efficacement le contenu et la présentation d'un site web. Les balises structurantes sont :

- **Balise DOCTYPE** : Cette balise est indispensable car elle indique au navigateur qu'il s'agit d'une page HTML. Elle est placée au tout début du document et est suivie par la balise **<html>**.
- **Balise HTML** : Cette balise englobe tout le contenu de la page HTML et fournit des informations sur le langage utilisé. Les informations mentionnées dans cette balise n'apparaissent pas sur le document final généré.
- **Entête du document HTML** : La balise **<head>** contient des informations qui ne seront pas affichées sur la page, telles que le titre du document **<title>** et d'autres métadonnées comme le nom de l'auteur.
- **Corps du document HTML** : La balise **<body>** définit le contenu et la présentation visibles de la page web. Elle contient toutes les balises nécessaires pour structurer le contenu, y compris les en-têtes **<header>**, les pieds de page **<footer>**, les sections **<section>**, les informations complémentaires **<aside>**, etc.
- **En-tête (<header>)** : Cette balise se trouve généralement en haut de la page et comprend des éléments tels qu'un logo, une bannière ou le slogan du site.
- **Pied de page (<footer>)** : Cette balise se trouve en bas de la page et contient des informations telles que des liens de contact, le nom de l'auteur, les mentions légales, etc.
- **Principaux liens de navigation (<nav>)** : Cette balise est utilisée pour regrouper tous les principaux liens de navigation du site, tels que le menu principal.
- **Section de page (<section>)** : Cette balise est utilisée pour regrouper des contenus en fonction de leur thématique, ce qui facilite l'organisation du contenu sur la page.
- **Informations complémentaires (<aside>)** : Cette balise contient des informations complémentaires au document principal. Elle peut être utilisée pour des éléments autonomes de la page qui pourraient être repris sur d'autres sites, comme des actualités ou des articles de blogs.

V. Création des liens hypertextes

Les liens hypertextes sont des éléments essentiels pour relier différentes parties d'une page web ou pour pointer vers d'autres documents sur le Web. Pour créer un lien hypertexte, on utilise la balise `<a>` avec l'attribut **href** pour spécifier l'URL du document vers lequel le lien doit pointer.

Exemple

```
<a href="http://www.example.com">Un texte</a>
```

1. Attributs de la balise `<a>`

La balise `<a>` peut inclure d'autres attributs, facultatifs, pour contrôler le comportement du lien.

- L'attribut **title** permet de définir un texte qui s'affiche lorsque l'utilisateur survole le lien avec la souris.
- L'attribut **target** indique dans quelle fenêtre ou onglet le lien doit s'ouvrir, avec des valeurs telles que `_blank` pour ouvrir dans une nouvelle fenêtre ou `_top` pour ouvrir dans la fenêtre parente.

2. Types de liens hypertextes

On distingue deux types de liens hypertextes : les liens internes et les liens externes.

- **Les liens internes** permettent de naviguer à l'intérieur du même document, en utilisant des ancres pour définir des points spécifiques dans la page. Pour créer un lien interne, il faut définir des points d'ancrage dans la page à l'aide de l'élément `<a>` et de l'attribut **href**. Ensuite, on peut référencer ces points d'ancrage dans le lien en utilisant l'URL de la page suivie du symbole `#` suivi du nom de l'ancre.

Exemple

```
<a href="#label1">Référence à mon point d'ancrage 1</a><br>
<a href="#label2">Référence à mon point d'ancrage 2</a><br>
<a href="#label3">Référence à mon point d'ancrage 3</a><br>

<a name="label1"> Point d'ancrage 2</a><br>
<!-- le contenu de cette première partie -->
<a name="label2"> Point d'ancrage 2</a><br>
<!-- le contenu de cette deuxième partie -->
<a name="label3"> Point d'ancrage 3</a><br>
<!-- le contenu de cette troisième partie -->
```

- **Les liens externes**, quant à eux, pointent vers des documents situés sur d'autres sites web en spécifiant simplement l'URL complète. Les liens externes utilisent simplement l'attribut href pour spécifier l'URL complète du document cible.

Exemple

```
<a href="http://www.example.com">Un texte</a>
```

VI. Insertion d'éléments hypermédia

L'intégration d'éléments hypermédia constitue une pratique fréquente dans la conception de pages web, visant à enrichir le contenu et à offrir une expérience utilisateur plus interactive et captivante. Cette approche permet d'incorporer divers types de médias, tels que les images, les figures, l'audio et la vidéo, dans une page web. Voici une analyse approfondie de la manière d'insérer ces différents types d'éléments hypermédia.

1. Les images

L'image est une composante essentielle dans la création de pages web pour enrichir le contenu visuel. La balise **** est utilisée pour insérer des images dans une page web. Elle ne nécessite pas de balise de fermeture. Cette balise possède divers attributs, permettant de contrôler l'apparence et le comportement des images dans les pages web. Les Attributs de la balise **** sont :

- **src** : Cet attribut spécifie l'URL de l'image à afficher. Il indique l'emplacement et le nom du fichier image.

Exemple

```

```

- **align** : Cet attribut permet d'aligner l'image par rapport au texte. Les valeurs possibles sont pour cet attribut sont :
 - **top** pour aligner en haut,
 - **middle** pour aligner au milieu,
 - **bottom** pour aligner en bas,
 - **left** pour aligner à gauche et
 - **right** pour aligner à droite.
- **width** : Cet attribut définit la largeur de l'image en pixels. **Exemple : width="300"**.
- **height** : Cet attribut définit la hauteur de l'image en pixels. **Exemple : height="200"**.
- **border** : Cet attribut spécifie l'épaisseur de la bordure autour de l'image en pixels. **Exemple : border="1"**.

- **alt** : Cet attribut fournit un texte alternatif qui s'affiche lorsque l'image ne peut pas être chargée ou lorsque la souris reste pointée sur l'image. La description spécifiée par cet attribut est utile pour l'accessibilité, la compréhension du contenu de l'image et ainsi pour les moteurs de recherche.
- **title** : Cet attribut permet d'ajouter une infobulle qui s'affiche lorsque l'utilisateur survole l'image avec la souris. Il peut contenir un texte descriptif ou informatif.

Exemple

```
L'Ecole Nationale d'Electronique et des TeleCommunications de Sfax

```

2. Les figures

La balise **<figure>** est utilisée pour encapsuler tout type de contenu multimédia, comme une image, une vidéo, une illustration, etc., avec une légende associée. Elle est généralement suivie de la balise **<figcaption>** qui contient la légende de l'élément encapsulé.

Exemple

```
<figure>
  
  <figcaption>Figure 1 : Caption </figcaption>
</figure>
```

3. Les éléments audio

L'intégration de fichiers audio dans une page web est réalisé utilisant la balise **<audio>**. Cette balise est dotée de plusieurs attributs qui offrent un contrôle et une personnalisation flexibles sur la manière dont le fichier audio est présenté et lu. Elle permet de spécifier la source du fichier audio à l'aide de l'attribut **src** et peut inclure des attributs supplémentaires tels que **controls** pour afficher les contrôles de lecture.

- L'attribut **controls** est utilisé pour afficher les contrôles de lecture standard tels que Lecture, Pause et Barre de défilement, facilitant l'interaction avec le contenu audio.
- L'attribut **width** est utilisé pour définir la largeur de l'élément de lecture audio, ce qui permet d'adapter sa taille à la mise en page de la page web.
- L'attribut **loop** indique que la piste audio doit être lue en boucle une fois qu'elle est terminée. Cet attribut est généralement utilisé pour créer des arrière-plans sonores ou des effets audio répétitifs.
- L'attribut **autoplay** permet de démarrer automatiquement la lecture de la piste audio dès le chargement de la page.

- L'attribut **preload** détermine si le navigateur doit pré-charger la piste audio lors du chargement initial de la page. Les valeurs possibles sont auto, metadata et none, chacune offrant un contrôle précis sur le moment et la manière dont les données audio sont chargées.

Exemple

```
<audio src= "son.mp3" controls />
```

4. Les éléments Vidéo

L'intégration de contenu vidéo dans une page web se fait en utilisant la balise **<video>**. Cette balise est munie de plusieurs attributs qui permettent de contrôler et personnaliser le comportement de lecture de la vidéo. Elle partage les mêmes attributs que la balise **<audio>**, avec l'ajout d'un attribut supplémentaire appelé **poster**. Cet attribut permet de spécifier une image à afficher avant le démarrage de la vidéo.

Exemple

```
<video src="chemin_vers_la_vidéo" controls width="600"></video>
```

VII. Création de tableaux

La création d'un tableau en HTML est réalisée utilisant la balise **<table>**. Cette balise définit la zone qui contiendra les objets du tableau. Elle est utilisée pour organiser les données de manière tabulaire. Divers attributs sont couramment utilisés avec cette balise, tels que l'attribut **'border'** qui définit l'épaisseur de la bordure du tableau, l'attribut **'align'** qui définit l'alignement horizontal, l'attribut **'width'** spécifie la largeur du tableau, l'attribut **'cellspacing'** définit l'espace entre les cellules, l'attribut **'cellpadding'** utilisé pour préciser l'espace entre le contenu et la bordure des cellules, etc.

La création d'un tableau se fait ligne par ligne utilisant la balise **<tr>**. Chaque ligne peut contenir plusieurs colonnes, ajoutées utilisant la balise **<th>** ou **<td>**. Chaque cellule peut contenir du texte, une image, un formulaire, une liste, etc. Les cellules **<th>** sont généralement utilisées pour l'en-tête de colonne ou de ligne, alors que les cellules **<td>** contiennent généralement les données du tableau.

1. Les tableaux structurés

Des tableaux, dites structurés, peuvent être créés en HTML. La création de ce type de tableaux nécessite l'utilisation d'autres balises structurantes définissant l'en-tête, le corps et le pied du tableau. La balise **<thead>** définit l'en-tête du tableau, situé généralement en

haut du tableau. La balise **<tbody>** contient le corps principal du tableau, où se trouvent les données. Enfin, la balise **<tfoot>** définit le pied du tableau, généralement situé en bas du tableau et contenant des informations récapitulatives.

2. Fusion de cellules d'un tableau

Dans un tableau, plusieurs cellules peuvent être fusionnées pour les regrouper en une seule, améliorant ainsi la présentation. La fusion des cellules est faite horizontalement avec l'attribut **'colspan'** et verticalement avec l'attribut **'rowspan'**. Cela permet de créer des mises en page complexes et structurées.

VIII. Création des formulaires

Les formulaires en HTML constituent des outils fondamentaux qui permettent aux utilisateurs de saisir et de soumettre des informations. Ils sont utilisés dans une variété de contextes, notamment pour les inscriptions, l'authentification, les enquêtes et bien d'autres interactions en ligne. Ces formulaires jouent un rôle crucial dans l'interaction avec les utilisateurs et la collecte des données sur le web. Selon le contexte, un formulaire peut contenir une variété de champs qui sont enveloppés par la balise **<form>**. Cette balise comporte deux attributs essentiels, l'attribut **'action'**, qui spécifie l'URL vers laquelle les données saisies dans le formulaire seront envoyées, et l'attribut **'method'**, qui détermine la méthode d'envoi des données, soit la méthode POST, soit la méthode GET.

1. Les zones de saisie

Les formulaires peuvent inclure une variété de zones de saisie pour recueillir différentes informations. Parmi celles-ci, on trouve les zones de texte mono-ligne, utiles pour la saisie de données telles que les identifiants de connexion, les noms et prénoms ; les zones de texte multi-lignes, idéales pour la saisie de commentaires et de messages plus longs ; ainsi que les zones de saisie de mots de passe, conçues pour masquer les caractères saisis. De plus, il existe des zones spécifiques pour la saisie d'adresses e-mail et d'URL, qui permettent souvent une vérification automatique des données sans nécessiter d'intervention de l'utilisateur. De plus, la zone de saisie de type "number" est utilisée pour recueillir des valeurs numériques. Le Tableau 1 présente les différentes zones de saisie d'un formulaire ainsi que leur syntaxe.

Tableau 1. Liste des zones de saisies

Type de zone de saisie	Syntaxe
Texte mono-ligne	<code><input type="text" name="pseudo" id="pseudo" /></code>
Texte multi-ligne	<code><textarea name="motivation" id="motivation" rows="5" cols="30"> Votre motivation ci </textarea></code>
Mot de passe	<code><input type="password" name="pass" id="pass" /></code>
E-mail	<code><input type="email" name="adr" id="adr"/></code>
URL	<code><input type="url" name="adr" id="adr"/></code>
Numéro de téléphone	<code><input type="tel" name="num" id="num"/></code>
Les nombres	<code><input type="number" name="num" id="num" min="0" max="10" step="2"/></code>
Un curseur	<code><input type="range" name="num" id="num" min="0" max="10" step="2"/></code>
La couleur	<code><input type="color" name="col" id="col"/></code>
La date	<code><input type="time" name="d" id="d"/></code> <code><input type="date" name="d" id="d"/></code> <code><input type="week" name="d" id="d"/></code> <code><input type="month" name="d" id="d"/></code>
La recherche	<code><input type="search" name="rech" id="rech"/></code>

2. Les éléments d'options

Les formulaires incluent des éléments d'options, à savoir les cases à cocher, les boutons radio et les listes déroulantes.

- **Cases à cocher** : Elles permettent à l'utilisateur de sélectionner plusieurs options parmi un ensemble. Chaque case est indépendante des autres, permettant des choix multiples. Par exemple, dans un formulaire, l'utilisateur peut cocher plusieurs cases correspondant à ses intérêts pour recevoir des notifications par e-mail sur différents sujets.
- **Boutons radio** : Ils permettent à l'utilisateur de choisir une seule option parmi plusieurs alternatives. Contrairement aux cases à cocher, les boutons radio offrent un choix exclusif. Dans un formulaire de sondage, par exemple, l'utilisateur peut sélectionner une seule réponse parmi plusieurs propositions en utilisant des boutons radio.

- **Listes déroulantes** : Aussi appelées menus déroulants, offrent à l'utilisateur une liste d'options parmi lesquelles choisir. Les listes déroulantes sont souvent utilisées pour des sélections uniques parmi des alternatives mutuellement exclusives ou pour des listes longues. Par exemple, dans un formulaire de sélection de pays, l'utilisateur peut choisir son pays dans une liste déroulante plutôt que de le saisir manuellement.

Le Tableau 2 présente les différents éléments d'options d'un formulaire ainsi que leur syntaxe.

Tableau 2. Liste des éléments d'options

Type de zone de saisie	Syntaxe - Exemple
Cases à cocher	<p>Cochez les aliments :</p> <pre> <input type="checkbox" name="frites" id="frites" checked /> <label for="frites">Frites</label>
 <input type="checkbox" name="steak" id="steak" /> <label for="steak">Steak haché</label>
 <input type="checkbox" name="epinards" id="epinards" /> <label for="epinards">Epinards</label>
 </pre>
Zones d'options	<p>Veuillez indiquer la tranche d'âge :</p> <pre> <input type="radio" name="age" value="moins15" id="moins15" /> <label for="moins15">Moins de 15 ans</label>
 <input type="radio" name="age" value="medium15-25" id="medium15-25" /> <label for="medium15-25">15-25 ans</label>
 <input type="radio" name="age" value="medium25-40" id="medium25-40" /> <label for="medium25-40">25-40 ans</label>
 <input type="radio" name="age" value="plus40" id="plus40" /> <label for="plus40">Encore plus vieux que ça ?!</label> </pre>
Les listes	<pre> <label for="pays">Pays :</label>
 <select name="pays" id="pays"> <option value="france"> France </option> <option value="espagne"> Espagne </option> <option value="italie"> Italie </option> <option value="japon"> Japon </option> </select> </pre>

3. Regroupement de champs

Pour améliorer la présentation et la lisibilité d'un formulaire HTML, il est courant de regrouper les différents champs en ensembles logiques. Cette pratique permet aux utilisateurs de mieux comprendre le contenu du formulaire. Chaque ensemble est encapsulé dans une balise `<fieldset>`, accompagné d'une légende définie par la balise `<legend>`. Ainsi, les utilisateurs peuvent facilement identifier et naviguer entre les différents groupes de champs.

4. Les boutons d'envoi

Les boutons dans un formulaire HTML offrent à l'utilisateur la possibilité de transmettre les informations saisies au serveur, d'annuler ce qui engendre la suppression des informations saisies, ou d'appeler une fonction.

- **Bouton de soumission du formulaire (`<input type="submit">`)** : Ce bouton, lorsqu'actionné, envoie les données du formulaire au serveur pour traitement. Il est couramment utilisé pour valider et transmettre les informations saisies par l'utilisateur.
- **Bouton de réinitialisation du formulaire (`<input type="reset">`)** : En cliquant sur ce bouton, tous les champs du formulaire sont réinitialisés à leurs valeurs par défaut. Il permet à l'utilisateur d'effacer les informations saisies et de recommencer à remplir le formulaire.
- **Bouton d'action personnalisée (`<input type="button">`)** : Ce bouton peut être programmé pour déclencher des actions spécifiques côté client, comme la validation des informations avant l'envoi ou l'affichage de messages contextuels. Contrairement au bouton de soumission, il n'envoie pas automatiquement les informations au serveur.
- **Bouton de soumission du formulaire sous forme d'image (`<input type="image">`)** : Avec ce type de bouton, l'utilisateur peut soumettre le formulaire en cliquant sur une image plutôt qu'un bouton standard. Lorsque l'image est sélectionnée, les informations insérées dans le formulaire sont transmises au serveur.

IX. Création des images réactives

Chaque image insérée dans une page HTML peut être rendue cliquable, c'est à dire qu'elle devienne un lien vers une page web. Les images réactives visent à associer plusieurs liens hypertextes à un seul objet graphique. Dans ce contexte, la balise `<map>` sert à définir des zones de liens appelées zones réactives sur une image. Son utilisation est couplée à la balise ``.

La création des images réactives suit les étapes suivantes :

➤ Etape 1: Insertion de l'image : L'image est tout d'abord insérée dans la page HTML, avec tous les attributs nécessaires:

Exemple

```

```

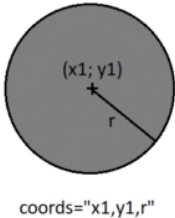
➤ Etape 2: Création de la carte : Il faut ensuite créer la carte qui va contenir les différentes zones réactives. Cette carte est créée à l'aide de la balise <map>. Cette balise possède l'attribut **name=nom-de-la-carte**, où nom-de-la-carte correspond au usemap de l'image. Le conteneur <map> contient autant de balises <area /> qu'il y a de zones réactives.

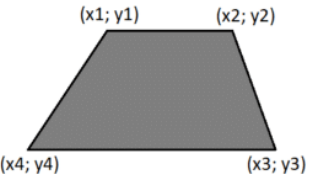

➤ Etape 3: Zones réactives : A chaque zone réactive est associée une balise <area /> dont les attributs sont:

- **name** : le nom de la zone,
- **shape** : la forme géométrique (cercle, rectangle ou polygone),
- **coords** : les coordonnées de la zone, séparées par des virgules,
- **href** : lien hypertexte de la zone,
- **alt** : le texte alternatif.

Le Tableau 2 présente les différentes zones réactives ainsi que leur syntaxe.

Tableau 3. Liste des zones réactives

Zones réactives	Syntaxe
<p>Circulaire</p>  <p>coords="x1,y1,r"</p>	<pre><area shape="circle" coords="x1,y1,r" href="....." /></pre>
<p>Polygonale</p>	<pre><area shape="poly" coords="x1,y1,x2,y2,x3,y3,x4,y4,x1,y1" href="....." /></pre>

 <p>coords="x1,y1,x2,y2,x3,y3,x4,y4,x1,y1"</p>	
<p>Rectangulaire</p>  <p>coords="x1,y1,x2,y2"</p>	<pre><area shape="rect" coords="x1,y1,x2,y2" href="....." /></pre>

CHAPITRE II : Le style de présentation CSS

I. Introduction

Le CSS, ou Cascading Style Sheets, est un pilier fondamental de la conception et la mise en forme des pages web. Alors que le HTML définit la structure et le contenu d'une page web, le CSS intervient pour façonner leur apparence visuelle et leur mise en page. En d'autres termes, il indique aux balises HTML comment se comporter et comment être présentées à l'utilisateur. De plus, les feuilles de style CSS permettent de regrouper dans un seul document les règles de mise en forme qui seront appliquées à l'ensemble des éléments d'une page web. Cette approche présente plusieurs avantages significatifs. La centralisation des styles en regroupant toutes les caractéristiques de mise en forme dans une feuille de style externe. Ainsi, il devient possible de maintenir la cohérence visuelle sur l'ensemble d'un site web. Plutôt que de définir les styles individuellement pour chaque élément sur chaque page, les concepteurs peuvent définir une seule fois les règles de style dans une feuille de style et les appliquer de manière cohérente sur toutes les pages du site. De plus, le CSS assure la consistance et homogénéité. En fait, l'utilisation d'une feuille de style commune assure une présentation uniforme sur l'ensemble du site. Cela garantit une expérience utilisateur cohérente, où les éléments visuels sont prévisibles et où la navigation est intuitive. Les visiteurs du site peuvent ainsi facilement s'orienter et trouver les informations qu'ils recherchent. Finalement, le CSS garantit la flexibilité et la facilité de gestion. Les feuilles de style permettent également une gestion efficace des mises à jour et des modifications de l'aspect visuel d'un site. En modifiant simplement les règles de style dans la feuille de style, les concepteurs peuvent apporter des changements significatifs à l'apparence du site sans avoir à modifier individuellement chaque élément sur chaque page. Cela simplifie grandement le processus de maintenance et de mise à jour du site.

II. Historique

Les feuilles de style CSS ont connu une évolution significative depuis ses débuts dans les années 1990.

- **CSS 1 :** Lancé en 1996. CSS 1 a établi les bases du langage pour la présentation des pages web. Il a introduit des fonctionnalités telles que la définition des couleurs, des marges et des polices de caractères, permettant aux concepteurs de contrôler l'apparence visuelle des sites web de manière plus précise.

- **CSS 2** : apparu en 1999. CSS 2 a apporté de nombreuses améliorations et options supplémentaires par rapport à CSS 1. Il a introduit des fonctionnalités avancées telles que le positionnement des éléments sur la page, la gestion des couches (z-index), et les sélecteurs avancés, offrant ainsi aux concepteurs une plus grande flexibilité dans la création de mises en page complexes et sophistiquées.
- **CSS 3** : a commencé à émerger au début des années 2000. Il a introduit une multitude de nouvelles fonctionnalités et de propriétés pour améliorer encore plus la conception des sites web. Parmi ces nouveautés, on trouve les bordures arrondies, les ombres, les dégradés, les transitions et les animations, offrant aux concepteurs des outils plus avancés pour créer des interfaces web modernes et esthétiques.

III. Définition de style CSS

La définition d'un style en CSS repose sur l'utilisation de règles CSS, qui spécifient comment les éléments HTML doivent être présentés sur une page web. Chaque règle CSS est composée de plusieurs éléments clés :

- **Sélecteur** : Le sélecteur CSS indique à quelles balises HTML le style doit être appliqué. Il peut s'agir d'un nom de balise, d'une classe, d'un ID ou d'un sélecteur universel. Par exemple, le sélecteur `h1` cible tous les éléments `<h1>` sur la page, tandis que le sélecteur `'.classe'` cible tous les éléments ayant la classe spécifiée.
- **Déclaration de style** : Une déclaration de style est composée d'une ou de plusieurs propriétés et de leurs valeurs, délimitées par des accolades `{ }`. Chaque déclaration de style est associée à un sélecteur et contient les instructions sur la façon dont les éléments sélectionnés doivent être stylisés.

Exemple

```
h1 {  
  color: blue;  
  font-size: 24px;  
}
```

Dans cet exemple, le sélecteur `h1` est associé à deux déclarations de style : une pour définir la couleur du texte en bleu et une pour définir la taille de la police à 24 pixels.

- **Propriété** : Une propriété CSS définit l'aspect visuel d'un élément sélectionné. Il existe de nombreuses propriétés CSS différentes pour contrôler des aspects tels que la couleur, la taille, la police, la marge, le remplissage, etc.
- **Valeur** : La valeur d'une propriété CSS spécifie la manière dont la propriété doit être appliquée à l'élément sélectionné. Par exemple, la valeur blue spécifie la couleur bleue pour la propriété color, tandis que 24px spécifie une taille de police de 24 pixels pour la propriété font-size.

En combinant tous ces éléments, sélecteurs, déclarations de style, propriétés et valeurs, permet de contrôler l'apparence de chaque élément sur la page.

IV. Insertion de code CSS dans un document HTML

L'insertion de code CSS dans un document HTML peut se faire de plusieurs manières, chacune avec ses propres avantages et utilisations spécifiques :

1. Ajout d'une CSS à une balise HTML (Attribut Style)

Cette première méthode d'insertion de style CSS dans un document HTML consiste à utiliser l'attribut **style** dans une balise HTML spécifique pour appliquer des styles directement à cette balise. Cette méthode est utile pour des styles spécifiques à un élément.

Exemple

```
<div style="color: blue; font-size: 16px;">
  Ceci est un exemple de texte avec un style en ligne.
</div>
```

Dans cet exemple, les styles sont spécifiés directement dans l'attribut style de la balise <div>, ce qui affecte uniquement cette balise.

2. Ajout de CSS à une page (Balise Style)

Cette deuxième méthode d'insertion de style CSS dans un document HTML consiste à utiliser la balise <style> dans l'en-tête HTML pour inclure directement du code CSS dans une page HTML spécifique. Cette méthode est utile pour des styles spécifiques à une page ou des modifications rapides.

Exemple

```
<head>
  <style>
    /* Vos règles de style CSS ici */
    body {
      font-family: Arial, sans-serif;
      background-color: #f0f0f0;
    }
  </style>
</head>
```

Dans cet exemple, les règles de style sont définies directement entre les balises `<style>` et `</style>`.

3. Lier un site à un fichier CSS externe (Balise Link)

Cette méthode d'insertion de style CSS dans un document HTML consiste à utiliser la balise `<link>` dans l'en-tête HTML pour lier une page HTML à un fichier CSS externe. Cette méthode est couramment utilisée pour appliquer un style cohérent à tout un site web.

Exemple

```
<head>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
```

Dans cet exemple, le fichier CSS externe **styles.css** contient toutes les règles de style pour le site web, et la balise `<link>` spécifie l'emplacement de ce fichier CSS.

V. Appliquer un style à une ou plusieurs balises

Un style CSS peut être appliqué à une ou plusieurs balises HTML, ainsi qu'à des balises spécifiques selon différents critères. Ainsi, diverses techniques en CSS sont définies :

1. Appliquer un style à plusieurs balises

Pour appliquer le même style CSS à plusieurs balises différentes, il faut séparer les noms de balises par une virgule. Dans l'exemple suivant, les titres de niveau 1, 2 et 3 auront la même famille de polices.

Exemple

```
h1, h2, h3
{
  font-family: Arial, sans-serif;
}
```


2. Appliquer un style avec des classes et des IDs

Pour cibler des balises spécifiques et leur appliquer des styles il est possible d'utiliser des classes et des IDs.

Exemple

```
.important
{
    font-weight: bold;
}

#header
{
    background-color: #333;
    color: white;
}
```

```
<p class="important">Ce paragraphe est important.</p>
<div id="header">Ceci est l'en-tête du site.</div>
```

3. Appliquer un style à une balise qui suit une autre

Utiliser le sélecteur + pour cibler une balise qui suit immédiatement une autre balise spécifique. Dans l'exemple suivant, tous les paragraphes <p> qui suivent immédiatement un titre de niveau 2 <h2> auront leur texte en italique.

Exemple

```
h2 + p {
    font-style: italic;
}
```

4. Appliquer un style à une balise qui possède un attribut

Utiliser le sélecteur d'attribut [attribut] pour cibler une balise qui possède un attribut spécifique. Dans cet exemple, tous les paragraphes centrés auront une couleur rouge.

Exemple

```
p[align="center"] {
    color : red;
}
```

VI. Les propriétés et les valeurs

Dans un fichier CSS, les propriétés et les valeurs sont utilisées pour définir différents aspects de la mise en forme d'une page web. Voici quelques propriétés couramment utilisées avec leurs exemples d'utilisation :

1. Formatage du texte

- **font-style** : Définit le style de l'écriture, comme l'italique.
Exemple : `h3 {font-style: italic;}`
- **font-weight** : Définit l'épaisseur de la police, comme le gras.
Exemple : `p {font-weight: bold;}`
- **font-size** : Définit la taille du texte en pixels.
Exemple : `p { font-size: 14px;}`
- **font-family** : Définit la police à utiliser pour le texte.
Exemple : `p { font-family: Verdana;}`
- **text-align** : Définit l'alignement du texte, comme à gauche, au centre ou à droite.
Exemple : `h1 {text-align: center;}`
- **text-transform** : Définit la casse du texte, comme en majuscules, en minuscules ou avec la première lettre en majuscule.
Exemple : `p {text-transform: uppercase;}`
- **text-decoration** : Permet de décorer le texte, comme avec un soulignement, un texte barré ou un clignotement.
Exemple : `h1 { text-decoration: blink;}`

2. La couleur et le fond :

- **color** : Définit la couleur du texte.
Exemple : `h3 {color: #000080;}`
- **background-color** : Définit la couleur de fond d'un élément.
Exemple : `body {background-color: black; color: white;}`
- **background-image** : Indique une image de fond.
- **background-attachment** : Fixe le fond par rapport à l'élément, avec des valeurs comme `fixed` ou `scroll`.
- **background-repeat** : Définit la répétition de l'image de fond.
- **background-position** : Définit la position de l'image de fond par rapport à l'élément.

3. Les bordures et les ombres

- **border-radius** : Arrondir les angles de n'importe quel élément
- **border-image** : Utiliser une image pour tracer une bordure

- **box-shadow** : définir des ombres autour des boîtes d'éléments HTML, comme des divs ou des blocs de texte. Elle prend quatre valeurs, le décalage horizontal de l'ombre, le décalage vertical de l'ombre, l'adoucissement du dégradé de l'ombre et la couleur de l'ombre.
- **text-shadow** : définir des ombres autour du texte. Elle prend également quatre valeurs similaires à box-shadow.

4. Transparence et opacité

Dans le langage CSS, il est possible de contrôler la transparence et l'opacité des éléments en utilisant différentes méthodes. Deux approches courantes sont largement utilisées :

a) Utilisation de rgba()

La fonction **rgba()** permet de spécifier une couleur en utilisant les valeurs de rouge, vert, bleu et un canal alpha pour la transparence. Ce canal alpha est un nombre compris entre 0 (complètement transparent) et 1 (complètement opaque). Dans l'exemple suivant, l'élément HTML avec la classe **.element-transparent** aura un fond rouge semi-transparent avec une opacité de 50%.

Exemple

```
.element-transparent {
  /* Rouge semi-transparent */
  background-color: rgba(255, 0, 0, 0.5);
}
```

b) Utilisation de l'opacité avec opacity

La propriété **opacity** permet de définir le niveau global de transparence pour un élément et son contenu, indépendamment de sa couleur. Elle prend une valeur entre 0 (complètement transparent) et 1 (complètement opaque). Dans l'exemple suivant, l'élément HTML avec la classe **.element-opaque** aura un niveau d'opacité de 70%, rendant à la fois son fond et son contenu partiellement transparents.

Exemple

```
.element-opaque {
  opacity: 0.7; /* Opacité à 70% */
}
```

5. Arrière-plans multiples

Avant CSS3, la propriété background-image était limitée à l'utilisation d'une seule image comme arrière-plan pour un élément. Ensuite, avec CSS3, il est désormais possible

d'utiliser plusieurs images comme arrière-plan pour un même élément, offrant ainsi plus de flexibilité dans la conception. Dans l'exemple suivant, l'élément HTML avec la classe **.element-multiple-backgrounds** aura deux images d'arrière-plan, positionnées respectivement en haut à gauche et en bas à droite, avec la première image ne se répétant pas et la deuxième se répétant horizontalement.

Exemple

```
.element-multiple-backgrounds {  
    background-image: url('image1.jpg'), url('image2.jpg');  
    /* Positionnement des images */  
    background-position: left top, right bottom;  
    /* Répétition des images */  
    background-repeat: no-repeat, repeat-x;  
}
```

6. Les dégradés

Le CSS3 permet de créer des dégradés directement dans les pages, sans avoir besoin de charger une image depuis le réseau ou faire recours à un logiciel spécialisé. Il existe deux types de dégradés : les dégradés linéaires et les dégradés radiaux.

- Les dégradés linéaires peuvent être horizontaux, verticaux, diagonaux, centrés ou non, et peuvent contenir deux ou plusieurs couleurs. Dans l'exemple suivant, l'élément HTML avec la classe **.element-gradient** aura un arrière-plan avec un dégradé linéaire allant du rouge au bleu de gauche à droite.

Exemple

```
.element-gradient {  
    /* Dégradé linéaire de rouge à bleu, de gauche à droite */  
    background-image: linear-gradient(to right, red, blue);  
}
```

- Les dégradés radiaux sont une fonctionnalité avancée offerte par CSS3, permettant de créer des effets de transition en douceur à partir d'une couleur centrale vers une couleur périphérique, ou vice versa. Les dégradés radiaux peuvent être concentriques, elliptiques ou axiaux. Un dégradé radial concentrique part d'un point central et se propage uniformément dans toutes les directions. Un dégradé radial elliptique suit une forme elliptique définie. Un dégradé radial axial suit une direction axiale définie. Dans l'exemple suivant, l'élément HTML avec la classe **.element-radial-gradient** aura un arrière-plan avec un dégradé radial, commençant par la couleur rouge au centre et se fondant progressivement en bleu vers l'extérieur.

Exemple

```
.element-radial-gradient {  
    background-image: radial-gradient(circle, red, blue);  
}
```

7. Les éléments flottants

En CSS, les éléments flottants sont des éléments qui ne suivent pas le flux normal du document. L'utilisation de la propriété **float** en CSS offre la possibilité de créer des mises en page flexibles où les éléments peuvent se disposer côte à côte ou s'envelopper les uns autour des autres. La propriété float est ainsi utilisée pour déplacer un élément vers la gauche ou la droite de son conteneur, en le retirant du flux normal du document. Les valeurs courantes pour la propriété float sont :

- **left** : L'élément flotte vers la gauche, permettant aux éléments suivants de s'écouler autour de lui sur le côté droit.
- **right** : L'élément flotte vers la droite, permettant aux éléments suivants de s'écouler autour de lui sur le côté gauche.
- **none** : Aucun flottement n'est appliqué à l'élément. C'est la valeur par défaut.
- **inherit** : L'élément hérite de la valeur float de son parent.

CHAPITRE III : Le langage JavaScript

I. Introduction

JavaScript est un langage de programmation coté client dans le développement web. Intégré dans le langage HTML, il offre des fonctionnalités interactives aux pages web. Avec JavaScript, les concepteurs peuvent ajouter des éléments de contrôle, des animations, et des fonctionnalités interactives qui améliorent à la fois la présentation et l'interactivité des sites web. Dans ce chapitre, nous explorerons en détail les concepts fondamentaux, les fonctionnalités et les applications pratiques de JavaScript dans le développement web.

II. Intégration de JavaScript dans HTML

Pour inclure des scripts JavaScript dans une page HTML, la balise `<script>` est utilisée. Cette balise peut être insérée directement dans le code HTML entre les balises `<script>...</script>`. Alternativement, si le script est stocké dans un fichier externe, l'attribut `src` de la balise `<script>` peut être utilisé pour faire référence à l'URL de ce fichier.

L'avantage de cette approche réside dans sa flexibilité : les balises `<script>` peuvent être placées n'importe où dans le document HTML, et autant de fois que nécessaire. Ainsi, les développeurs ont la liberté d'organiser leur code JavaScript de manière logique et efficace, en le répartissant selon les besoins et les exigences de la page web. Cette flexibilité permet également de maintenir un code HTML propre et lisible, en séparant la structure de la page de sa fonctionnalité interactive.

III. Les premières méthodes JavaScript pour l'interaction utilisateur

Dans le monde du web, l'interaction avec l'utilisateur est essentielle pour renforcer l'interactivité. JavaScript, en tant que langage de programmation côté client, offre une gamme de méthodes puissantes pour interagir avec les utilisateurs directement depuis les pages web. Cette section se concentre sur les différentes méthodes d'interaction JavaScript, telles que `write()`, `alert()`, `prompt()` et `confirm()`, qui permettent aux développeurs de créer des dialogues, des messages d'alerte, des invitations à saisir des données et des confirmations d'action.

1. Méthode `write()` :

La méthode `write()` est utilisée pour écrire du texte directement dans le document HTML à l'emplacement où le script est appelé. Elle est principalement utilisée pour ajouter du contenu dynamique obtenu par exemple par une variable ou une fonction à une page

web en utilisant JavaScript. Lorsque cette méthode est appelée, le texte spécifié est inséré à l'endroit où se trouve le script dans le code HTML.

2. Méthode `alert()` :

La méthode `alert()` est utilisée pour afficher des boîtes de dialogue modales simples à l'utilisateur avec un message texte. Elle est souvent utilisée pour fournir des informations importantes à l'utilisateur ou pour demander une confirmation avant de continuer une action. Lorsque cette méthode est appelée, une boîte de dialogue de type alerte apparaît avec le message spécifié, et l'utilisateur doit cliquer sur "OK" pour la fermer.

3. Méthode `prompt()` :

La méthode `prompt()` permet à l'utilisateur de saisir des données via une boîte de dialogue. Lorsque cette méthode est appelée, une boîte de dialogue avec un champ de saisie et des boutons "OK" et "Annuler" apparaît. Généralement, elle est utilisée pour demander à l'utilisateur de fournir une entrée, comme son nom ou une valeur numérique. Une fois que l'utilisateur saisi les données et appuie sur le bouton "OK", ces données peuvent être récupérées pour être utilisées dans le script.

4. Méthode `confirm()` :

La méthode `confirm()` est utilisée pour demander à l'utilisateur de confirmer ou d'annuler une action. Elle affiche une boîte de dialogue avec deux boutons : "OK" pour confirmer et "Annuler" pour annuler. Cette méthode renvoie `true` si l'utilisateur clique sur "OK" et `false` s'il clique sur "Annuler".

IV. Les éléments fondamentaux du langage JavaScript

Le langage JavaScript comprend plusieurs éléments fondamentaux, tels que les commentaires, les variables, portée et typage dynamique des variables, etc.

1. Les commentaires en JavaScript

Les commentaires en JavaScript permettent aux développeurs d'ajouter des annotations et des explications à leur code pour faciliter la compréhension et la maintenance. Deux types de commentaires couramment utilisés en javascript sont les commentaires sur une seule ligne et les commentaires sur plusieurs lignes.

2. Les variables en JavaScript

Les variables sont des composants fondamentaux de JavaScript, permettant aux développeurs de stocker et de manipuler des données dans leurs scripts. En JavaScript, une variable peut être déclarer explicitement avec le mot-clé 'var' ou implicitement en attribuant directement une valeur. Les variables sont soumises à un typage dynamique qui est une caractéristique essentielle. Elle permet aux variables de changer de type dynamiquement pendant l'exécution du programme. Contrairement à d'autres langages de programmation, où les variables sont typées statiquement et doivent être déclarées avec un type spécifique, JavaScript permet aux développeurs de déclarer des variables sans spécifier leur type. Cela signifie qu'une même variable peut contenir différentes valeurs de types différents à différents moments du programme. Cependant, outil fondamental, l'opérateur **'typeof'**, est offert par le langage JavaScript permettant la définition du type actuel d'une variable. Cet opérateur renvoie une chaîne de caractères indiquant le type de la valeur stockée dans la variable. Par exemple, typeof peut être utilisé pour déterminer si une variable contient un nombre, une chaîne de caractères, un objet, une fonction, ou autre. L'utilisation de l'opérateur typeof est utile dans les situations où le type d'une variable peut être ambigu ou lorsqu'il est nécessaire de prendre des décisions conditionnelles en fonction du type d'une variable. L'exemple suivant présente l'utilisation de l'opérateur typeof :

Exemple

```
<script type="text/javascript" >
  var x = 80;
  console.log(typeof x); // Affiche "number"

  var y = "Bonjour";
  console.log(typeof y); // Affiche "string"

  var jour = new Date();
  console.log(typeof jour); //Affiche "object"

  var choix = true;
  typeof choix; //retourne "boolean"

  typeof parseFloat; //retourne "function"
</script>
```

3. La portée des variables

En JavaScript, la portée des variables est un concept fondamental qui détermine la visibilité et l'accessibilité des variables dans différentes parties d'un script. Deux types de portée sont couramment utilisés : les variables globales et les variables locales.

a) Les variables globales

Une variable globale est déclarée en dehors de toutes les fonctions, généralement au début du script. Une fois déclarée, elle est accessible partout dans le script, à la fois à l'intérieur et à l'extérieur des fonctions. Cela signifie que sa portée s'étend à l'ensemble du script. Les variables globales sont souvent utilisées pour stocker des données qui doivent être accessibles et modifiables à partir de plusieurs parties du script.

Exemple

```
<script type="text/javascript" >
  var GV = "Bonjour à vous tous";
  function VGFunction() {
    console.log(GV); // La variable globale VG est accessible ici
  }
  console.log(GV); // La variable globale VG est aussi accessible ici
</script>
```

b) Les variables locales

Une variable locale est déclarée à l'intérieur d'une fonction et elle n'est accessible qu'à l'intérieur de la fonction dans laquelle elle est déclarée. En dehors de cette fonction, la variable locale n'est pas accessible et ne peut pas être référencée. Les variables locales sont utiles pour limiter la portée des variables aux fonctions dans lesquelles elles sont déclarées et ainsi éviter les conflits entre les noms de variables utilisés dans différentes fonctions et parties du script.

Exemple

```
<script type="text/javascript" >
  function VLFunction()
  {
    var LV = "Bonjour à vous tous";
    console.log(LV); //La variable locale LG est bien accessible ici
  }

  console.log(LV); // Erreur : LV n'est pas défini
</script>
```

4. La concaténation des variables

La concaténation de variables est un processus qui consiste à combiner plusieurs chaînes de caractères en une seule ou une chaîne de caractère et une valeur numérique. Cela peut être fait en utilisant l'opérateur de concaténation +.

a) Concaténer deux chaînes de caractères

La concaténation de deux chaînes de caractères consiste simplement à les joindre l'une à l'autre pour former une seule chaîne.

Exemple

```
<script type="text/javascript" >
  var prenom = "Sonda";
  var nom = "Ammar";
  var nomComple = prenom + " " + nom;
  console.log(nomComple); // Résultat: "Sonda Ammar"
</script>
```

b) Concaténer des valeurs numériques et une chaîne de caractères

La concaténation de valeurs numériques avec une chaîne de caractères engendre la conversion automatique des valeurs numériques en chaîne de caractères avant leur concaténation.

Exemple

```
<script type="text/javascript" >
  var age = 37;
  var ch = "Votre âge est : ";
  var message = ch + " " + age;
  console.log(message); // Résultat: "Votre âge est 37"
</script>
```

V. Les tableaux

Les tableaux sont des objets qui permettent de stocker et d'organiser des collections d'éléments. Ils peuvent contenir différents types de données, tels que des nombres, des chaînes de caractères, des objets, et même d'autres tableaux. En JavaScript, la taille des tableaux est dynamique. Elle peut être modifiée pendant l'exécution lorsqu'on ajoute un nouvel élément au tableau ou on retire un élément. La déclaration d'un tableau vide peut se faire de deux syntaxes différentes :

Exemple

```
<script type="text/javascript" >
  // Déclaration de tableau vide avec la classe Array
  var tab = new Array();

  // Déclaration de tableau vide avec des crochets
  var tab = [ ];
</script>
```

Un tableau peut être déclaré et initialisé en utilisant les crochets [...].

Exemple

```
<script type="text/javascript" >
  // Déclaration et initialisation ultérieure
  var fruits = ["Pomme", "Fraise", "Orange"];

  // Déclaration et initialisation ultérieure
  var fruits = [];
  fruits.push("Pomme");
  fruits.push("Fraise");
  fruits.push("Orange");
</script>
```

Les tableaux possèdent la propriété **length** qui renvoie le nombre d'éléments.

Exemple

```
<script type="text/javascript">
  var fruits = ["Pomme", "Banane", "Orange"];
  console.log(fruits.length); // Affiche : 3
</script>
```

Le langage JavaScript offre plusieurs méthodes prédéfinies qui peuvent être utilisées pour faciliter la manipulation des tableaux.

Méthode	Description -Exemple
unshift()	Ajouter un élément au début du tableau fruits.unshift("Poivre");
push()	Ajouter un élément à la fin du tableau fruits.push("Poivre");
pop()	Supprimer le dernier élément du tableau

	<code>fruits.pop();</code>
shift()	Supprimer le premier élément du tableau <code>fruits.shift();</code>
indexOf()	Trouver l'index d'un élément dans le tableau <code>fruits.indexOf("Orange");</code>
join()	Convertir un tableau en chaîne de caractères <code>fruits.join();</code> // "Pomme,Banane,Orange" <code>fruits.join(", ");</code> // " Pomme, Banane, Orange "
slice()	Retourner une section du tableau <code>fruits.slice(0, 2);</code> // ["Pomme", "Banane"]

VI. Les fonctions

JavaScript supporte deux types de fonctions, les fonctions pré-programmées et les fonctions définies par le programmeur (selon les besoins du script).

1. Les fonctions pré-programmées

Les fonctions pré-programmées sont des fonctions fournies par le langage JavaScript. Elles sont disponibles dès le chargement de la page ou de l'application web et peuvent être utilisées sans avoir besoin de les définir explicitement.

- **console.log()** : Cette fonction envoie un message à la console du navigateur, généralement utilisée à des fins de débogage pour afficher des valeurs de variables ou des messages d'état.
- **parseInt()** et **parseFloat()** : Ces fonctions convertissent une chaîne de caractères représentant un nombre en un nombre entier ou flottant respectivement. Elles sont utiles pour traiter les données saisies par l'utilisateur.
- **isFinite()** et **isNaN()** : Ces fonctions déterminent si le paramètre est un nombre fini ou ce n'est pas un nombre (NaN).
- **Eval()** : Cette fonction évalue une chaîne de caractères sous forme de valeur numérique.
- **Date()** : La fonction Date() crée un nouvel objet de date représentant la date et l'heure actuelles. Elle est souvent utilisée pour manipuler et afficher des dates dans les applications web.

2. Les fonctions utilisateur

Les fonctions définies par l'utilisateur sont créées par les développeurs pour effectuer des tâches spécifiques à l'application ou pour encapsuler des séquences d'instructions répétitives. Elles sont déclarées en utilisant le mot-clé `function`, suivi du nom de la fonction et de parenthèses contenant éventuellement des paramètres, puis des accolades délimitant le corps de la fonction. Une fois déclarées, les fonctions peuvent être appelées en utilisant leur nom suivi de parenthèses contenant les arguments, le cas échéant. L'appel de fonction peut être effectué à partir du code principal ou à partir d'autres fonctions.

Syntaxe	<pre><script type="text/javascript"> function nomDeLaFonction(paramètre1, paramètre2, ...) { // Corps de la fonction // Instructions à exécuter lorsque la fonction est appelée Instruction_1; ... Instruction_n; } </script></pre>
Exemple	<pre><!DOCTYPE html> <html> <head> <title>Fonction</title> <script language="Javascript"> function produit(a,b) { return a*b; } </script> </head> <body> <script language="Javascript"> var nb1, nb2; nb1= window.prompt("Entrez un premier entier"); nb2= window.prompt("Entrez un deuxième entier"); document.write(nb1+" * "+nb2+" = " +produit(parseInt(nb1),parseInt(nb2))); </script> </body> </html></pre>

VII. Le concept Objet en JavaScript

Les objets en JavaScript, sont soit des entités pré-définies du langage, soit créés par le développeur pour répondre à des besoins spécifiques de l'application. Voici quelques exemples d'objets couramment utilisés en JavaScript :

- **navigator** : Cet objet représente le navigateur web utilisé par l'utilisateur. Il fournit des informations sur le navigateur telles que le nom, la version et les fonctionnalités prises en charge.
- **window** : L'objet window représente la fenêtre du navigateur dans laquelle le contenu web est affiché. Il fournit des fonctionnalités pour manipuler la fenêtre du navigateur, telles que la modification de la taille, la navigation entre les pages et la gestion des événements.
- **document** : L'objet document représente le contenu HTML chargé dans la fenêtre du navigateur. Il fournit des méthodes pour accéder et manipuler les éléments HTML de la page, tels que la modification du contenu, l'ajout ou la suppression d'éléments, et la gestion des événements.
- **Formulaires** : Les formulaires HTML ainsi que leurs contenus sont également des objets en JavaScript. Chaque formulaire sur une page web est représenté par un objet qui permet d'accéder à ses éléments et de récupérer les valeurs saisies par l'utilisateur.

L'accès aux objets en JavaScript se fait généralement par une notation par points, où chaque point représente une descente dans la hiérarchie des objets. Par exemple, pour accéder à la première zone de texte du premier formulaire d'une page web, on peut utiliser **window.document.forms[0].elements[0]**. Il est à noter que certains objets, comme les formulaires et les éléments de formulaire, peuvent être automatiquement numérotés pour faciliter l'accès. En outre, il est possible d'accéder aux objets directement par leur nom, ce qui rend le code plus lisible et compréhensible. Par exemple, si un formulaire est nommé "personnel" et que sa première zone de texte est nommée "nom", on peut y accéder via **window.document.personnel.nom**.

Enfin, une autre méthode d'accès consiste à utiliser la notation entre crochets, ce qui permet de référencer les objets en utilisant des chaînes de caractères. Par exemple, **window.document.form["personnel"].elements["nom"]** accède à la première zone de texte du formulaire nommé "personnel" par le biais de son attribut nom.

VIII. La gestion des événements

La gestion des événements en JavaScript est essentielle pour rendre une page web interactive et réactive aux actions de l'utilisateur. Contrairement à HTML classique, où la gestion des événements est limitée, JavaScript offre la possibilité de contrôler et de réagir à divers événements, tels que les clics de souris, les pressions de touche du clavier, et bien plus encore. Les événements sont généralement associés à des actions de l'utilisateur, telles

que l'appui sur une touche, un clic de souris ou un déplacement de la souris. En HTML classique, les événements sont automatiquement gérés pour des actions telles que les clics sur des liens ou des boutons de formulaire, mais leur comportement est souvent limité et non personnalisable. JavaScript étend cette capacité en permettant aux développeurs de contrôler et de réagir à une gamme plus large d'événements. Par exemple, les événements liés à la souris permettent de détecter les mouvements de la souris, les clics et les relâchements de bouton. De même, les événements clavier permettent de réagir aux frappes de touche sur le clavier. En outre, JavaScript permet de gérer des événements liés aux objets de la page web, tels que les fenêtres du navigateur ou les éléments de formulaire. Cela offre aux développeurs un contrôle précis sur le comportement de la page en réponse aux actions de l'utilisateur. Le Tableau 4 présente les événements associés à la souris, au clavier, aux objets de la page web et les fenêtres du navigateur.

Tableau 4. Les événements en JavaScript

<i>Les événements sur la souris</i>	
<i>onclick</i>	Click sur l'élément
<i>ondblclick</i>	Double-click sur l'élément
<i>onmousedown</i>	Appui de la souris sur l'élément
<i>onmousemove</i>	Déplacement de la souris au-dessus de l'élément
<i>onmouseover</i>	Arrivée de la souris sur l'élément
<i>onmouseout</i>	Sortie de la souris de l'élément
<i>onmouseup</i>	Relâchement de la souris au-dessus de l'élément
<i>Les événements sur le clavier</i>	
<i>onkeydown</i>	Appui sur une touche
<i>onkeypress</i>	Appui puis relâchement de la touche
<i>onkeyup</i>	Relâchement de la touche
<i>Les événements sur les objets/fenêtre</i>	
<i>onabort</i>	Chargement d'une image interrompu
<i>onerror</i>	Chargement d'une image qui échoue
<i>onload</i>	Chargement d'un élément, d'un document

<i>onresize</i>	Redimensionnement du document
<i>onunload</i>	Fermeture de la page
<i>Les évènements sur les formulaires</i>	
<i>onblur</i>	Un élément de formulaire perd le focus
<i>onchange</i>	Le contenu d'un élément change
<i>onfocus</i>	Un élément du formulaire prend le focus
<i>onreset</i>	Le formulaire est réinitialisé
<i>onselect</i>	Sélection de texte dans un formulaire
<i>onsubmit</i>	Soumission de formulaire

Exemples :

Exemple 1 : Modification de valeur d'une zone de texte

Enoncé : Écrivez un programme JavaScript qui permet de changer la valeur d'un champ de texte lorsqu'un utilisateur survole un lien hypertexte.

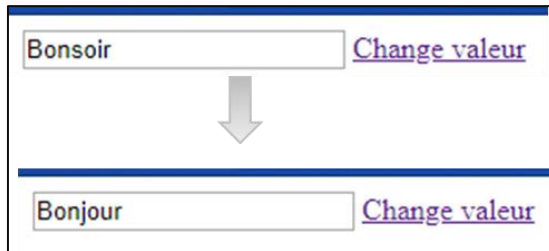
Instructions :

- Créez un fichier HTML avec la structure de base.
- Ajoutez un formulaire avec un champ de texte et un lien hypertexte à l'intérieur.
- Utilisez l'événement `onMouseOver` pour détecter le survol du lien.
- Définissez une fonction JavaScript qui change la valeur du champ de texte lorsque le lien est survolé.
- Utilisez l'événement `onMouseOut` pour détecter lorsque le curseur de la souris quitte le lien.
- Définissez une fonction JavaScript qui rétablit la valeur d'origine du champ de texte lorsque le lien n'est plus survolé.

Solution proposée

```
<body>
  <form name="f">
    <input type="text" name="ztm">
    <a href="#" onMouseOver = "document.forms.f.ztm.value = 'Bonjour'"
      onMouseOut = "document.forms.f.ztm.value = 'Bonsoir'">Change valeur</a>
  </form>
</body>
```

Résultat d’affichage



Exemple 2 : Bloquer l’écriture dans la balise HTML Input

Enoncé : Écrivez un programme JavaScript qui empêche l’utilisateur d’écrire dans un champ de texte lorsqu’il est en focus.

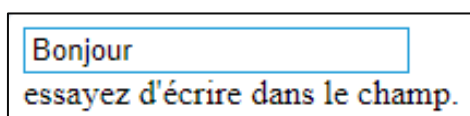
Instructions :

- Créez un fichier HTML avec la structure de base.
- Ajoutez un formulaire avec un champ de texte à l’intérieur.
- Utilisez l’événement onfocus pour détecter lorsque le champ de texte est en focus.
- Définissez une fonction JavaScript qui appelle la méthode blur() sur le champ de texte lorsqu’il est en focus.

Solution proposée

```
<body>
  <form name="f">
    <input type="text" name="ztb" value="Bonjour"
      onfocus = "document.forms.f.ztb.blur()"><br>essayez d’écrire dans le champ.
  </form>
</body>
```

Résultat d’affichage



Exemple 3 : Copier le contenu d'un champ de texte dans un autre champ de texte

Enoncé : Écrivez un programme JavaScript qui permet de copier le contenu d'un champ de texte dans un autre champ de texte lorsqu'un utilisateur clique sur un lien hypertexte.

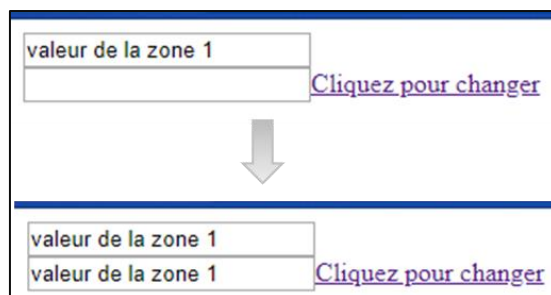
Instructions :

- Créez un fichier HTML avec la structure de base.
- Ajoutez un formulaire avec deux champs de texte à l'intérieur.
- Ajoutez un lien hypertexte à côté du deuxième champ de texte.
- Utilisez l'événement onClick sur le lien pour détecter le clic de l'utilisateur.
- Définissez une fonction JavaScript qui copie la valeur du premier champ de texte dans le deuxième champ de texte lorsque le lien est cliqué.

Solution proposée

```
<body>
  <form name="f">
    <input type="text" name="ztr" value="valeur de la zone 1"> <br/>
    <input type="text" name="cmd" value="">
    <a href="#"
      onClick = "document.forms.f.cmd.value = document.forms.f.ztr.value">
      Cliquez pour changer
    </a>
  </form>
```

Résultat d'affichage



Exercice :

Objectif : Mise en place d'un processus d'authentification sécurisé et un formulaire d'inscription intuitif pour les utilisateurs d'un site web, garantissant un accès contrôlé aux fonctionnalités du site tout en permettant aux utilisateurs de fournir leurs informations personnelles de manière simple et efficace.

Instructions :

Page d'accueil (Accueil.html) :

- Créez une page d'accueil nommée "Accueil.html".
- Cette page contient deux liens hypertexte :
 - Le premier lien "Authentification" redirige l'utilisateur vers la page "Auth_User.html".
 - Le deuxième lien "Créer un compte" redirige l'utilisateur vers la page "Formulaire.html".

Page d'authentification (Auth_User.html) :

- Créez une page "Auth_User.html" avec deux champs de saisie : "Login" et "Mot de passe".
- Ajoutez deux boutons :
 - Un bouton "Valider" pour soumettre les informations saisies vers la page "Verif_Login_Mp.js".
 - Un bouton "Annuler" pour réinitialiser le formulaire.

Script de vérification des informations d'authentification (Verif_Login_Mp.js) :

- Créez un script nommé "Verif_Login_Mp.js" qui récupère les saisies de la page "Auth_User.html".
- Vérifiez si les informations d'authentification saisies correspondent à celles autorisées (par exemple, login=0000 et mot de passe=1234).
- Si les informations ne sont pas correctes, redirigez l'utilisateur vers la page "Auth_User.html".

Page de formulaire d'inscription (Formulaire.html) :

- Créez une page "Formulaire.html" présentant un formulaire d'inscription avec plusieurs champs obligatoires.
- Ajoutez deux boutons :
 - Un bouton de validation pour soumettre les informations saisies à la page "Verif_Info.js".
 - Un bouton de réinitialisation pour effacer les champs du formulaire.

Script de vérification des informations du formulaire (Verif_Info.js) :

- Créez un script nommé "Verif_Info.js" qui définit des règles de vérification pour les informations du formulaire.
- Définissez les règles de validation pour les champs obligatoires selon vos besoins.

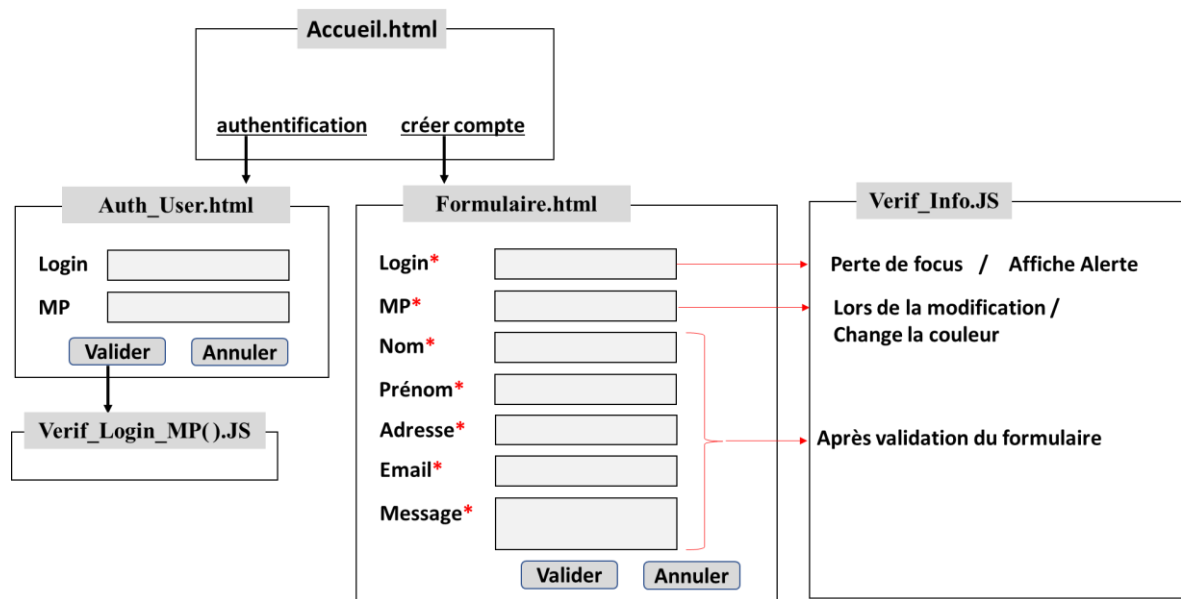


Figure 1. Structure des différentes pages

CHAPITRE IV : Le PHP

I. Introduction

PHP, acronyme de "Hypertext Preprocessor", est un langage de script côté serveur largement utilisé pour la création de pages Web dynamiques. Depuis son introduction, PHP a révolutionné le paysage du développement web en permettant la construction de sites interactifs et évolutifs. Des géants de l'Internet tels que Facebook, YouTube et Wikipedia ont largement profité de la puissance de PHP pour offrir des expériences utilisateur riches et dynamiques.

Ce langage dynamique présente de nombreux avantages, notamment sa gratuité, sa disponibilité en open source et sa simplicité d'écriture de scripts. Une caractéristique clé de PHP est sa capacité à interagir avec des bases de données, permettant ainsi aux développeurs de créer des applications web qui stockent et récupèrent des données de manière transparente. Un des avantages de PHP réside également dans sa facilité d'intégration au sein de pages HTML, offrant ainsi une flexibilité maximale aux développeurs pour créer des sites web à la fois dynamiques et esthétiques. En outre, PHP offre une compatibilité avec de nombreux systèmes de gestion de bases de données (SGBD), le plus populaire étant MySQL, qui est largement utilisé dans l'écosystème PHP pour son accessibilité et sa robustesse.

II. La vitalité persistante de PHP dans le paysage du développement web

Bien que certaines voix puissent laisser entendre que PHP est en déclin, les données révèlent tout autre chose. Selon W3Techs (voir Figure 2), PHP reste le choix de 78,1 % des sites web dont le langage de programmation côté serveur est identifié. Cela signifie que près de 4 sites web sur 5 reposent sur PHP pour leur fonctionnement.

Plutôt que de disparaître, PHP continue de prospérer, évoluant constamment pour répondre aux besoins changeants du développement web. Son adoption massive et sa popularité continue attestent de sa robustesse et de sa pertinence dans le paysage technologique actuel.

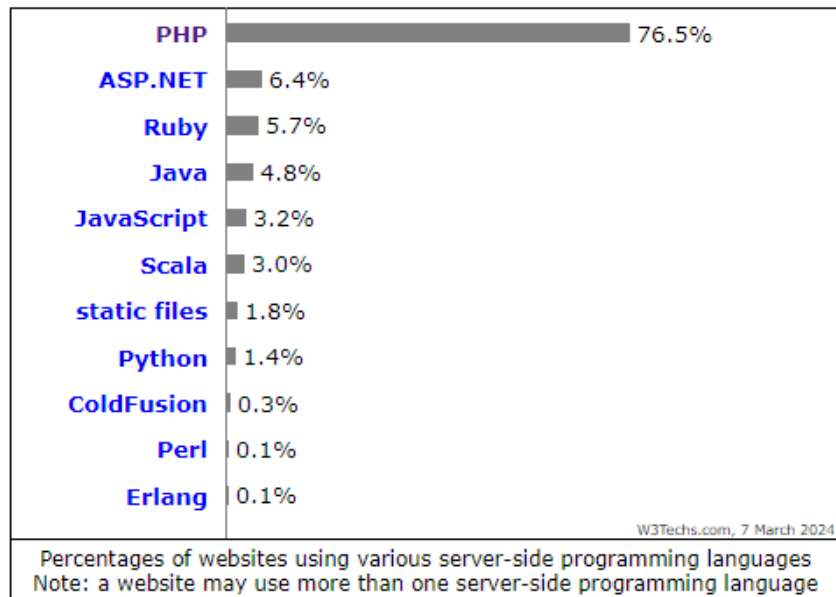


Figure 2. Comparaison des langages de développement web coté serveur ([Mars 2024](#))¹

III. Le fonctionnement du World Wide Web

Le fonctionnement du World Wide Web est une interaction complexe entre les utilisateurs, les serveurs web et les ressources dynamiques telles que les pages PHP et les bases de données. Lorsqu'un utilisateur accède à une URL via un navigateur, celui-ci envoie une requête HTTP au serveur web. Cette requête peut demander une page PHP qui nécessite un traitement dynamique, comme l'accès à une base de données pour récupérer des informations spécifiques. Le serveur web reçoit cette requête et, en fonction de son contenu, détermine la réponse appropriée. S'il s'agit d'une page PHP, le serveur exécute le code PHP inclus dans la page pour générer du contenu personnalisé. Cela peut impliquer la récupération de données à partir d'une base de données, le traitement de formulaires soumis par l'utilisateur, ou d'autres actions nécessaires pour répondre au besoin du client de manière dynamique. Une fois que le serveur a généré la réponse, il la renvoie au navigateur de l'utilisateur sous forme de réponse HTTP. Le navigateur interprète ensuite cette réponse et affiche le contenu de la page générée, qu'il s'agisse de texte, d'images, de formulaires interactifs ou d'autres éléments.

¹ [Usage Statistics and Market Share of Server-side Programming Languages for Websites, March 2024 \(w3techs.com\)](#)

IV. Les scripts PHP

Les instructions PHP sont encapsulées entre les balises spéciales `<?php` et `?>`. Cela permet de délimiter clairement le code PHP du reste du code HTML. Un fichier HTML contenant des scripts PHP doit avoir l'extension **.php** pour que le serveur web puisse interpréter le code PHP contenu à l'intérieur.

Chaque instruction PHP se termine par un point-virgule `;`. Cela indique la fin de l'instruction et permet au parseur PHP de savoir où se termine chaque instruction.

Syntaxe

```
<html>
  <head>
    <title>Exemple de page PHP</title>
  </head>
  <body>
    <h1>Exemple de page PHP</h1>
    <?php
      $prenom = "Sonda";
      $nom = "Ammar";
      echo "Bonjour, $prenom $nom !";
    ?>
  </body>
</html>
```

1. Les variables et les constantes

En PHP, toutes les variables sont préfixées par le symbole dollar `"$"` suivi du nom de la variable qui est sensible à la casse. La déclaration d'une variable peut se faire en même temps avec l'affectation d'une valeur à cette variable. Il n'est pas nécessaire de spécifier explicitement le type de données lors de la déclaration.

Les constantes se déclarent par la commande **define**, ne portent pas le préfixe `$` en début d'identificateur et ne sont pas modifiables.

Syntaxe

```
define("var", valeur) //define la constante var (sans $) de valuer valeur
```

En PHP, il est possible d'effectuer des conversions de type explicites, également connues sous le nom de "casting", similaires à celles que l'on trouve dans d'autres langages

de programmation tels que le langage C. Cela permet de convertir une variable d'un type de données à un autre. Le casting explicite permet de convertir une variable d'un type de données à un autre en utilisant les fonctions de conversion de type intégrées de PHP.

Syntaxe

```
$x = (int) $variable; // Convertit $variable en un entier  
$y = (float) $autreVariable; // Convertit $autreVariable en un flottant  
$z = (string) $encoreUneAutreVariable; // Convertit  
$encoreUneAutreVariable en une chaîne de caractères  
$tableau = (array) $variable; // Convertit $variable en un tableau
```

Plusieurs fonctions prédéfinies en PHP sont proposées facilitant la manipulation des variables, telles que :

- **Empty(\$var)** : renvoi vrai si la variable est vide
- **isset(\$var)** : renvoi vrai si la variable existe
- **unset(\$var)** : détruit une variable
- **gettype(\$var)** : retourne le type de la variable
- **is_long(), is_double(), is_string(), is_array(), is_object(), is_bool(), is_float(), is_numeric(), is_integer(), is_int()...**

2. Les chaînes de caractères

En PHP, une variable chaîne de caractères peut être délimitée par des guillemets simples (') ou des guillemets doubles ("). Voici un exemple illustrant la différence entre les deux :

Syntaxe

```
echo "Nom : $nom"; // affiche Nom : AMMAR  
echo 'Nom : $nom'; // affiche Nom : $nom  
$j=10; $m=5; $a=2018;  
echo "La date choisie: $j-$m-$a"; // affiche La date
```

• Quelques caractères spéciaux:

- **\n** : retour à la ligne

- `\t` : tabulation
- `\\` : antislash
- `\$` : caractère dollars
- `\"` : double quote
- `.` (**points**) : opérateur de concaténation de chaînes

3. Les tableaux

Une variable de type tableau est déclarée avec le mot-clé **array**. Un tableau peut être initialisé de plusieurs façons, notamment en énumérant ses éléments entre crochets et en les séparant par des virgules. Les tableaux en PHP peuvent contenir des éléments de différents types de données, tels que des entiers, des chaînes de caractères, des booléens, voire d'autres tableaux ou objets. Pour accéder à un élément spécifique du tableau, on utilise son indice, qui commence généralement à zéro.

Exemple

```
$monTableau = array(10, "texte", true, 3.14);
```

En PHP, il existe deux types principaux de tableaux : les tableaux indicés et les tableaux associatifs.

a) Les tableaux indicés

Les tableaux indicés utilisent des clés numériques pour accéder à leurs éléments. Pour accéder à un élément spécifique du tableau, on utilise son indice, qui commence généralement à zéro.

Exemple

```
<?php
//déclaration d'un tableau indexé numériquement
$jours=array('Lundi', 'Mardi', 'Mercredi');
// Ajout d'un jour au tableau
$jours[]='Jeudi';
// Affichage d'un élément du tableau
echo $jours[1]; // Affiche Mardi
?>
```

b) Les tableaux associatifs

Un tableau associatif est un tableau dont l'index est une chaîne de caractère au lieu d'un nombre. L'initialisation d'un tableau associatif est similaire à celle d'un tableau indicé. L'accès aux éléments du tableau peut être direct sans passer par les clés.

Exemples

```
<?php
    $auteur=array("Nom" => "AMMAR", "Prenom" => "Sonda");
?>
```

```
<?php
    $auteur["Nom"] = "AMMAR";
    $auteur["Prenom"] = "Sonda";
?>
```

V. Les fonctions

Les fonctions en PHP sont des blocs d'instructions regroupées et nommées, conçues pour effectuer une tâche spécifique et pouvant être appelées et réutilisées à plusieurs endroits dans le code. La déclaration d'une fonction se fait en utilisant le mot-clé **function**, suivi du nom de la fonction et de parenthèses contenant éventuellement des paramètres d'entrée.

Lors de la déclaration d'une fonction, les parenthèses sont obligatoires même s'il n'y a pas de paramètres. Contrairement aux variables, les identificateurs de fonctions en PHP sont insensibles à la casse, ce qui signifie que `maFonction()`, `mafonction()` et `MAFONCTION()` font référence à la même fonction. Les fonctions peuvent être définies n'importe où dans le code. Ils peuvent prendre des arguments, pour lesquels il n'est pas nécessaire de spécifier le type. Ces arguments peuvent être passés par valeur ou par référence. La fonction, en PHP, peut, de manière optionnelle, retourner une valeur à l'aide du mot-clé `return`. Dès que `return` est exécuté, PHP sort de la fonction. Il est possible de retourner plusieurs valeurs en retournant un tableau contenant ces valeurs.

Exemple

```
<?php
function fonction_list ($nb)
{
    Return ($nb/2, $nb*2, $nb+2; $nb-2);
}
list($d, $m, $a, $s)= FONction_List (41);
echo "$nb/2=$d, $nb*2=$m, $nb+2=$a, $nb-2=$s";
?>
```

VI. Inclure des bibliothèques ou des fichiers

PHP offre des fonctions très simples et très utiles pour la gestion de l'inclusion de fichiers, ce qui permet une réutilisation efficace du code et une meilleure organisation des scripts. Les fonctions **require()**, **include()** et leurs variantes **include_once()** et **require_once()**, permettent le chargement des fichiers PHP dans un script PHP.

La principale distinction entre **include** et **require** réside dans leur comportement lorsque le fichier à inclure ne peut pas être trouvé ou chargé pour une raison quelconque, telle qu'une erreur de chemin d'accès ou une indisponibilité du fichier. En fait, lors de l'utilisation de la fonction **include()**, si l'inclusion échoue, PHP émettra simplement un avertissement (warning) mais l'exécution du script se poursuit. En revanche, si la fonction **require()** est utilisée et que l'inclusion échoue, PHP générera une erreur fatale et interrompra immédiatement l'exécution du script. Ainsi, **require** est considéré comme plus "strict" que **include**.

La différence entre les fonctions **require()** et **include()** ainsi que leurs variantes **include_once()** et **require_once()**, réside dans le fait qu'un même fichier peut être inclus plusieurs fois avec **require()** et **include()**, alors qu'avec **include_once()** et **require_once()**, un même fichier ne peut être inclus qu'une seule fois dans un même script. Cela garantit que les dépendances ne sont chargées qu'une seule fois, évitant ainsi les conflits et les erreurs de redondance.

VII. Récupération des données d'un formulaire

La récupération des données d'un formulaire est une étape cruciale dans le développement de pages Web interactives. Les formulaires permettent aux utilisateurs de saisir et de transmettre des données à une page PHP pour traitement ultérieur. Dans le code HTML, un formulaire est défini à l'aide de la balise `<form>`. Cette balise comprend deux attributs principaux : **action** et **method**. L'attribut **action** spécifie l'URL de la page PHP qui traitera les données du formulaire une fois soumis, tandis que l'attribut **method** spécifie la méthode de transmission des données, qui peut être soit **get**, soit **post**.

Exemple

```
<form action="traitement.php" method="post">
  <!-- Les éléments du formulaire vont ici -->
</form>
```

Une fois que l'utilisateur soumet le formulaire, les données sont envoyées à la page spécifiée dans l'attribut **action** en utilisant la méthode spécifiée dans l'attribut **method**. En PHP, les données du formulaire sont accessibles à travers deux superglobales : `$_GET` et `$_POST`. La superglobale `$_GET` contient les données envoyées par la méthode **get**, tandis que `$_POST` contient les données envoyées par la méthode **post**. En utilisant ces superglobales, les données du formulaire peuvent être récupérées et manipulées dans le script PHP pour effectuer diverses opérations, telles que l'insertion dans une base de données, la validation des entrées utilisateur, ou la génération de réponses dynamiques.

Exemple

```
<?php
// Accès aux données envoyées par la méthode POST
$code = $_POST['code'];
$id = $_POST['id'];

// Accès aux données envoyées par la méthode GET
$nom = $_GET['nom'];
$categorie = $_GET['categorie'];
?>
```

VIII. Les sessions

Les sessions PHP constituent un mécanisme essentiel pour stocker et maintenir des données utilisateur tout au long de sa navigation sur un site Web. Introduites dans la version 4 de PHP, les sessions offrent une solution pratique pour transmettre des informations entre

différentes pages d'un site sans avoir à recourir à des mécanismes complexes de transmission de données. En fait, elles stockent des variables qui restent accessibles à partir des différentes pages du site, même lorsque l'utilisateur navigue entre les pages. En revanche, la session d'un utilisateur est associée à une durée de vie limitée. Si l'utilisateur ferme son navigateur ou quitte le site, la session sera détruite, ce qui garantit la sécurité et la confidentialité des données. Ceci explique l'utilisation des sessions pour des fonctionnalités telles que l'identification des utilisateurs via un système de login et de mot de passe, ainsi que la gestion des profils utilisateur. Contrairement aux cookies qui sont stockés côté client, les données des sessions sont conservées sur le serveur, ce qui les rend plus sécurisées car elles ne peuvent pas être manipulées par l'utilisateur.

Pour gérer les sessions de manière transparente, PHP fournit un ensemble de fonctions. Par exemple, la fonction **session_name()** permet de récupérer ou de définir le nom de la session en cours, tandis que **session_id()** renvoie l'identifiant unique de la session. Les principales étapes d'utilisation des sessions sont :

1. Ouverture d'une session

Une session est initiée en appelant la fonction `session_start()`. Cette fonction vérifie si un identifiant de session existe déjà. Si ce n'est pas le cas, elle crée une nouvelle session et attribue un identifiant unique à l'utilisateur.

2. Enregistrement des variables de session

Les données utilisateur sont enregistrées dans la session en les ajoutant au tableau super-global `$_SESSION`. Par exemple, `$_SESSION['nom'] = Sonda;` stocke le nom d'utilisateur dans la session.

3. Utilisation des variables de session

Une fois les données sont enregistrées dans la session, les variables de session peuvent être utilisées sur toutes les pages du site en accédant au tableau `$_SESSION`. Par exemple, `echo $_SESSION['nom'];` affichera " Sonda".

4. Suppression des variables de session

Pour supprimer une variable de session, on utilise la fonction **unset()**. Par exemple, `unset($_SESSION['nom']);` supprimera la variable 'nom' de la session.

5. Suppression de la session

Pour détruire une session, on utilise la fonction `session_destroy()`. Cette fonction supprime le fichier de données de session du serveur. Cependant, elle ne supprime pas les variables de session actuellement en mémoire. Pour effacer complètement la session, il est recommandé de vider le tableau `$_SESSION` à l'aide de `unset()` avant d'appeler `session_destroy()`.

Exemple

```
<?php
// Initialisation de la session
session_start() ;
// Sauvegarde de variable de la session
$_SESSION['prenom'] = 'SONDA';
// Affichage de la variable de session
echo $_SESSION['prenom'];
// Divers traitements
...
// Destruction de la session
session_destroy();
unset($_SESSION);
echo $_SESSION['prenom'] ; // N'affiche rien
?>
```

IX. PHP et MySQL

MySQL est un système de gestion de bases de données relationnelles largement utilisé. Il permet la création de plusieurs bases de données sur un même serveur, offrant ainsi une grande flexibilité dans la gestion des données. Chaque base de données est composée de tables qui contiennent des enregistrements structurés. L'accès aux bases de données MySQL se déroule généralement en plusieurs étapes :

- Ouvrir une connexion avec le serveur MySQL.
- Sélectionner la base de données sur laquelle travailler.
- Définir la requête SQL à exécuter, telle que SELECT, INSERT, UPDATE ou DELETE.
- Exécuter la requête et récupérer les résultats dans un objet appelé "recordset".
- Parcourir les lignes de résultats du recordset et extraire les données nécessaires.
- Fermer la connexion avec la base de données une fois toutes les opérations terminées.

a) Connexion au serveur

La fonction **mysqli_connect(\$server,\$user,\$password);** permet de se connecter au serveur **\$server** en tant qu'utilisateur **\$user** avec le mot de passe **\$password**. Cette fonction renvoie l'identifiant de connexion si la connexion est bien établie, FALSE sinon.

La fonction **mysqli_pconnect** est une fonction équivalente à **mysqli_connect** sauf que la connexion est persistante, il n'y a donc pas besoin de ré-ouvrir la connexion à chaque script qui travaille sur la même base.

Exemple

```
<?php
$connexion=mysqli_connect("localhost", "root", "");
if ($connexion)
{ // Connexion au serveur effectuée
  echo "<p>connexion réussie</p>";
}
else
  echo "<p>Erreur de connexion</p>";
?>
```

b) Connexion à la Base de Données : Sélection d'une base de données

La connexion à la base de donnée est basée sur l'utilisation de la fonction **mysqli_select_db([\$connexion,] base);**. Elle renvoie TRUE en cas de succès, sinon FALSE. Une fois la communication avec la base de données arrive à sa fin, on utilise la fonction **mysqli_close([\$connection])** pour fermer la connexion avec la base de données.

Exemple

```
<?php
$bd = mysqli_select_db($connexion, "Test");
if ($bd)
{ // Connexion à la base de données effectuée
  echo "<p>connexion à la BD réussie</p>";
}
else
  echo "<p> BD inconnue</p>";
?>
```

• Créer un fichier de configuration

Lorsqu'on développe une application web, il est courant d'avoir besoin d'accéder à une base de données à différents endroits du code. Pour éviter la redondance et simplifier la maintenance, il est recommandé de factoriser les informations de connexion dans un fichier de configuration dédié. Ainsi, un fichier de configuration doit être créé pour stocker

les informations de connexion à la base de données. Dans ce fichier de configuration, des constantes ou des variables sont définies pour stocker les informations de connexion, telles que l'hôte, le nom de la base de données, le nom d'utilisateur et le mot de passe. Une fois créée, dans les scripts PHP où on a besoin d'accéder à la base de données, on inclut simplement le fichier de configuration en utilisant la directive `require_once` ou `include_once`. Cela rendra les informations de connexion disponibles pour une utilisation ultérieure dans le script. Ensuite, les constantes ou les variables définies dans le fichier de configuration peuvent être utilisés pour établir une connexion à la base de données. De cette manière, la gestion des informations de connexion est centralisée dans un même fichier, ce qui facilite les mises à jour et améliore la sécurité de l'application.

Exemple

Configuration.php

```
<?php
$mysql_host = 'localhost';
$mysql_login = 'login';
$mysql_pass = 'secret';
$mysql_db = 'mabase';
$link = mysqli_connect($mysql_host, $mysql_login, $mysql_pass, $mysql_db);
?>
```

Traitement.php

```
<?php
include_once 'configuration.php';
// traitements suite à la connexion
?>
```

c) Interrogation d'une base de données

Une étape importante dans l'interaction avec une base de données MySQL en PHP est de définir la requête SQL à exécuter. Cette requête peut prendre différentes formes, telles que `SELECT` pour récupérer des données, `INSERT` pour ajouter de nouvelles données, `UPDATE` pour mettre à jour des données existantes, ou `DELETE` pour supprimer des données.

Une fois que la requête SQL est définie, elle est exécutée à l'aide de la fonction **`mysqli_query()`**. Lors de l'exécution de la requête, les résultats sont généralement récupérés dans un objet appelé "recordset". Cet objet contient les résultats de la requête sous une forme structurée, telle qu'un tableau associatif ou un objet. Il suffit ensuite de

parcourir cet objet pour accéder aux données récupérées et les manipuler selon les besoins de l'application.

Exemple

```
<?php
include_once 'configuration.php';
$requete = "Select nom, prenom From employe where deptno = 10 ";
$resultat = mysqli_query($link, $requete);
if ($resultat)
{
    // requête exécutée
    echo "<p>Requête correcte</p>";
}
else
    echo "<p>Requête incorrecte</p>";
?>
```

d) Extraire et exploiter les données de chaque ligne de résultats

Pour récupérer le résultat d'une requête, trois fonctions peuvent être exploiter

- **mysqli_fetch_assoc()** : Récupère le résultat sous forme de tableau associatif

Exemple

```
<?php
include_once 'configuration.php';
$requete = "Select nom, prenom From employe where deptno = 10 ";
$resultat = mysqli_query($link, $requete);

while ($enregistrement = mysqli_fetch_assoc($resultat))
{
    // Affiche le champ prenom
    echo $enregistrement['prenom'], ' ';
    // Affiche le champ nom
    echo $enregistrement['nom'], '<br>';
}
?>
```

- **mysqli_fetch_row()** : Récupère le résultat sous forme de tableau indexé. Chaque ligne est renvoyée sous forme de tableau. Les éléments du tableau sont les valeurs des attributs de la ligne. Si aucune ligne n'est trouvée, alors cette fonction renvoie false.

Exemple

```
<?php
if($resultat = mysqli_query($requete))
{
    while($ligne = mysqli_fetch_row($resultat))
    {
        $nom = $ligne[0];
        $prenom = $ligne[1];
        echo "$nom - $prenom <br />";
    }
}
else
{
    echo "Erreur de la requête de base de données.";
}
?>
```

- **mysqli_fetch_object()** : Récupère le résultat sous forme d'objet. Les attributs de l'objet et leurs valeurs correspondent à ceux de la ligne de résultat. Si aucune ligne n'est trouvée, alors cette fonction renvoie false.

Exemple

```
<?php
$requete = "Select nom, prenom From employe";
if($resultat = mysqli_query($requete))
{
    while($ligne = mysqli_fetch_object($resultat))
    {
        $nom = $ligne->nom;
        $prenom = $ligne->prenom;
        echo "$nom - $prenom <br />";
    }
}
else
{
    echo "Erreur de requête de base de données.";
}
?>
```

e) Fermer une connexion

La fonction **mysqli_close()** permet de fermer la connexion établie avec la base de données. Elle prend en argument l'identifiant de la connexion.

Exercice 1 :

On désire mettre en place un site web dynamique pour la gestion d'usine (**Figure 3**). Dans cet exercice, on s'intéresse uniquement à la partie administrateur qui consiste à la gestion des employés.

La base de données, utilisée dans ce site, est appelée 'Usine'. Cette base comporte les tables suivantes :

- Employé (CIN, login, mp, nom, prenom, adresse, email, grade)
- Admin (ID, login, mp)

On suppose que le compte de l'administrateur est déjà créé avec le login 'admin' et le mot de passe 'admin'. L'admin s'authentifie dans la page **auth_admin.html**. Le login et

le mot de passe seront envoyés vers la page **accueil.php** pour les vérifier. Si l'admin est correctement authentifié, le contenu de la page d'accueil (**accueil.php**) s'affiche, sinon il sera redirigé vers la page **auth_admin.html**.

La page d'accueil comporte les deux liens suivants :

- **Ajouter employé** : lorsque l'admin clique sur ce lien, il sera redirigé vers la page '**ajout_employe.php**'. Cette page affiche le formulaire d'ajout d'un nouvel employé. La validation du formulaire, renvoie l'administrateur vers la page '**liste_employe.php**'. Cette page insère les informations saisies dans la table 'Employé' de la base 'Usine'. Ensuite, elle affiche une alerte pour informer l'administrateur de la réussite de l'ajout. Enfin, elle affiche la nouvelle liste des employés.
- **Supprimer employé** : lorsque l'admin clique sur ce lien, il sera redirigé vers la page '**liste_employe.php**'. Cette page affiche la liste de tous les employés de la table 'Employé' sous la forme d'un tableau de quatre colonnes. La 1^{ère} colonne affiche le CIN de l'employé, la 2^{ème} colonne affiche le nom et le prénom de l'employé et la 3^{ème} affiche son grade. Une 4^{ème} colonne comporte un lien 'supp'. Ce lien renvoi l'administrateur vers la page '**supp_employe.php**' qui permet de supprimer l'employé correspondant de la base de données et elle affiche la nouvelle liste des employés.

Travail demandé :

Ecrivez le code des différentes pages : **auth_admin.html**, **accueil.php**, **ajout_employe.php**, **liste_employe.php** et **supp_employe.php**.

NB. Le message « Bonjour 'admin' » affiché dans certaines pages est basé sur l'utilisation de la session lancée après la connexion de l'administrateur.

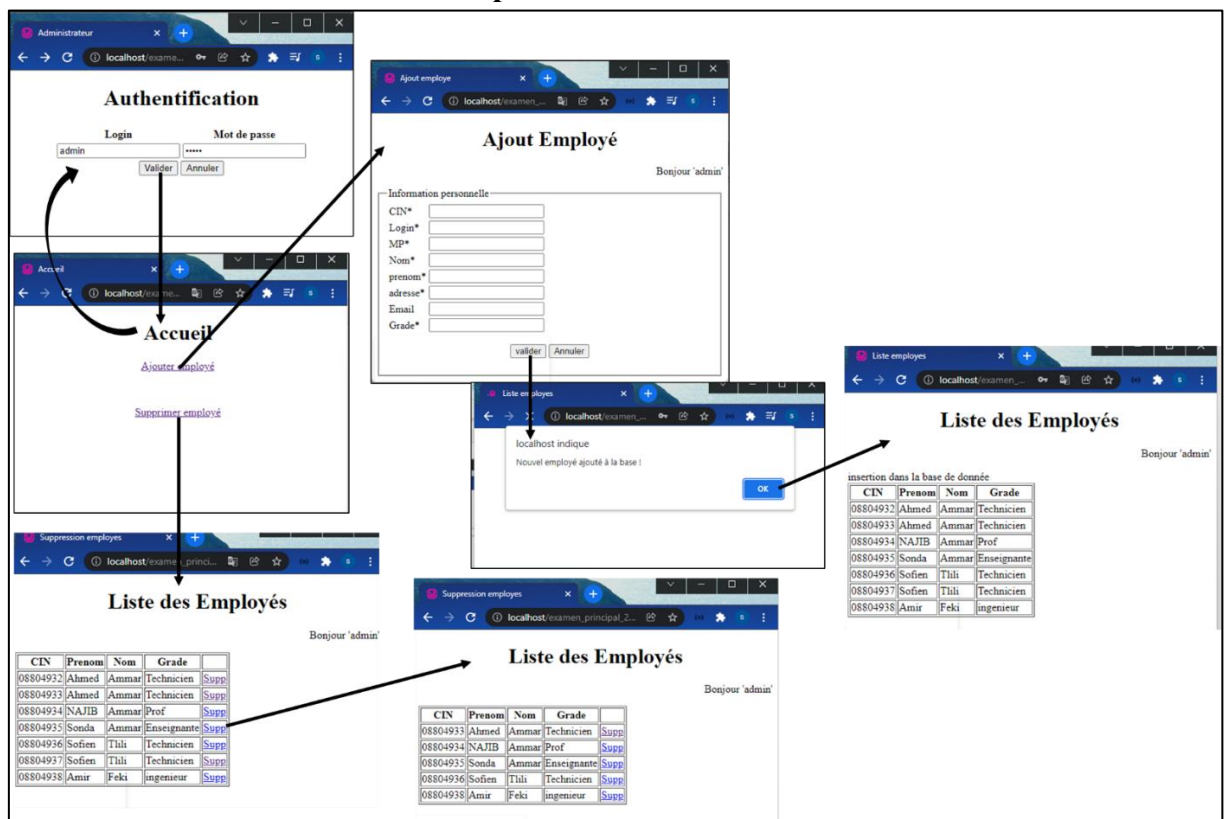


Figure 3. Structure du site Gestion des employés

Exercice 2 :

On désire mettre en place un site web dynamique permettant de prendre un RDV pour le vaccin Covid (**Figure 4**). Le site sera divisé en deux parties :

- Partie pour le citoyen.
- Partie pour l'administrateur

La base de données, utilisée dans ce site, est appelée 'Covid'. Cette base est composée des tables suivantes :

- Citoyen (CIN, login, motdepasse, nom, prenom, datenaissance, sexe, region, adresse, numtelephone, email, datepremiervaccin, datedeuxiemevaccin)

Les champs 'datepremiervaccin' et 'datedeuxiemevaccin' déterminent respectivement la date du premier vaccin et la date du deuxième vaccin (Rappel). Il faut savoir que la date du deuxième vaccin est calculée à partir de la date du premier vaccin en ajoutant une semaine.

- Admin (ID,login, motdepasse)

Partie 1 : Citoyen

Un nouveau citoyen doit créer un compte '**enregistrement.php**'. La création de compte d'un citoyen consiste à saisir les informations suivantes : Login, mot de passe, nom, prénom, numéro de la CIN, date de naissance, sexe, région, adresse, numéro de téléphone, et son email. Tous les champs sont obligatoires sauf l'adresse email qui est facultatif.

Les champs doivent respecter les exigences suivantes :

- Pour le **champ sexe**, deux boutons radio sont affichés avec les valeurs Femme et Homme.
- Pour le **champ région**, un menu déroulant doit être affiché au citoyen. Ce menu comporte 8 choix à savoir : Tunis, El Kef, Sousse, Sfax, Gafsa et Medenine. Le citoyen doit choisir la région la plus proche.
- Le **champ numéro de téléphone** n'accepte que 6 chiffres.

Si une de ces exigences n'est pas respectée par le citoyen, une alerte sera affichée. L'alerte indique au citoyen la cause de l'erreur et l'invite à corriger les informations saisies.

Une fois le formulaire est correctement rempli, le citoyen clique sur le bouton **valider**. Après la validation, les informations du nouveau citoyen sont enregistrées dans la table 'Citoyen' de la base de données 'Covid'. Enfin, le citoyen sera redirigé vers une page 'Info.php'.

La page **Info.php** affiche les informations du citoyen sous forme d'un tableau de deux colonnes. La première colonne affiche les champs et la deuxième affiche la valeur correspondante à chaque champ. Les champs 'date premier vaccin' et 'date deuxième vaccin' s'affichent sans valeurs tant que le dossier du citoyen n'est pas encore traité par l'administrateur. Dans les jours qui suit, le citoyen doit s'authentifier, via la page '**auth_citoyen.php**', pour voir la date des deux rdv.

Partie 2 : Administrateur

On considère que le compte de l'administrateur est déjà créé avec le login 'admin' et le mot de passe 'admin'. Une fois l'admin est correctement authentifié via la page **auth_admin.php**, la page d'accueil (**accueil.php**) sera affichée. Cette page affiche les propositions suivantes sous formes de lien hypertexte :

1. **Afficher tous les citoyens** : lorsque l'admin clique sur ce lien, il sera redirigé vers la page '**tous_citoyen.php**'. Cette page affiche la liste de tous les citoyens de la table 'Citoyen' sous la forme d'un tableau de trois colonnes. La 1^{ère} colonne affiche le nom et le prénom du citoyen, la 2^{ème} affiche la région du citoyen et la 3^{ème} affiche une note (n'a pas encore de rdv, vacciné 0 fois, vacciné 1 fois, vacciné 2 fois).
2. **Afficher les citoyens par région** : lorsque l'admin clique sur ce lien, il sera redirigé vers la page '**region_citoyen.php**'. Cette page affiche, pour chaque région, la liste de tous les citoyens de la table 'Citoyen' sous la forme d'un tableau de deux colonnes. La 1^{ère} affiche le nom et le prénom du citoyen, la 2^{ème} affiche une note (n'a pas encore de rdv, vacciné 0 fois, vacciné 1 fois, vacciné 2 fois).
3. **Afficher les citoyens qui n'ont pas encore de rdv** : lorsque l'admin clique sur ce lien, il sera redirigé vers la page '**rdv_citoyen.php**'. Cette page affiche la liste de tous les citoyens de la table 'Citoyen' qui n'ont pas encore de rdv. La liste est ordonnée par âge des citoyens, du plus âgé au moins âgé. L'affichage sera sous la forme d'un tableau de cinq colonnes. La 1^{ère} affiche le nom et le prénom du citoyen, la 2^{ème} affiche la région, la 3^{ème} affiche l'âge du citoyen, la 4^{ème} comporte un bouton '**rdv**' et la 5^{ème} comporte un bouton de confirmation '**confirmer**'. Lorsque l'admin clique sur le bouton '**rdv**', un calendrier sera affiché. L'admin choisit une date et il confirme ensuite le rdv en cliquant sur le bouton '**confirmer**'. Dès que l'admin confirme la date du rdv, les champs 'datepremiervaccin', 'datedeuxiemevaccin', de la table 'Citoyen', du citoyen correspondant seront mis à jour.
4. **Afficher les citoyens qui ont fait uniquement le premier vaccin** : lorsque l'admin clique sur ce lien, il sera redirigé vers la page '**rdv_1_citoyen.php**'. Cette page affiche la liste de tous les citoyens de la table 'Citoyen' qui ont fait uniquement le premier vaccin. L'affichage dans cette page sera sous la forme d'un tableau de trois colonnes. La 1^{ère} affiche le nom et le prénom du citoyen, la 2^{ème} affiche la région et la 3^{ème} affiche la date du 2^{ème} vaccin.
5. **Afficher les statistiques** : lorsque l'admin clique sur ce lien, il sera redirigé vers la page '**statistiques.php**'. Cette page affiche sous la forme d'un tableau les chiffres et les pourcentages suivants :
 - le nombre total des citoyens,
 - le nombre de citoyen qui ont effectué un seul vaccin et le pourcentage par rapport au nombre total des citoyens inscrits,

- le nombre de citoyen qui ont effectué les deux vaccins et le pourcentage par rapport au nombre total des citoyens inscrits,

Mise en forme du site :

Créer une feuille de style « style.css » avec les propriétés suivantes :

1. La couleur d'arrière-plan de la première page est noire, le lien pour la page auth_admin.php est de couleur rouge, le lien pour la page auth_citoyen.php est de couleur blanc et le lien pour la page enregistrement.php est de couleur jaune. Le style des liens est Arial.
2. La couleur d'arrière-plan des autres pages est blanche. La bordure des tableaux des différentes pages est d'épaisseur 2, de couleur bleu, et avec un ombre.
3. Les nombres affichés dans la page '**statistiques.php**' sont en gras et de couleur vert. Les pourcentages de la même page sont soulignés et de couleur rouge.

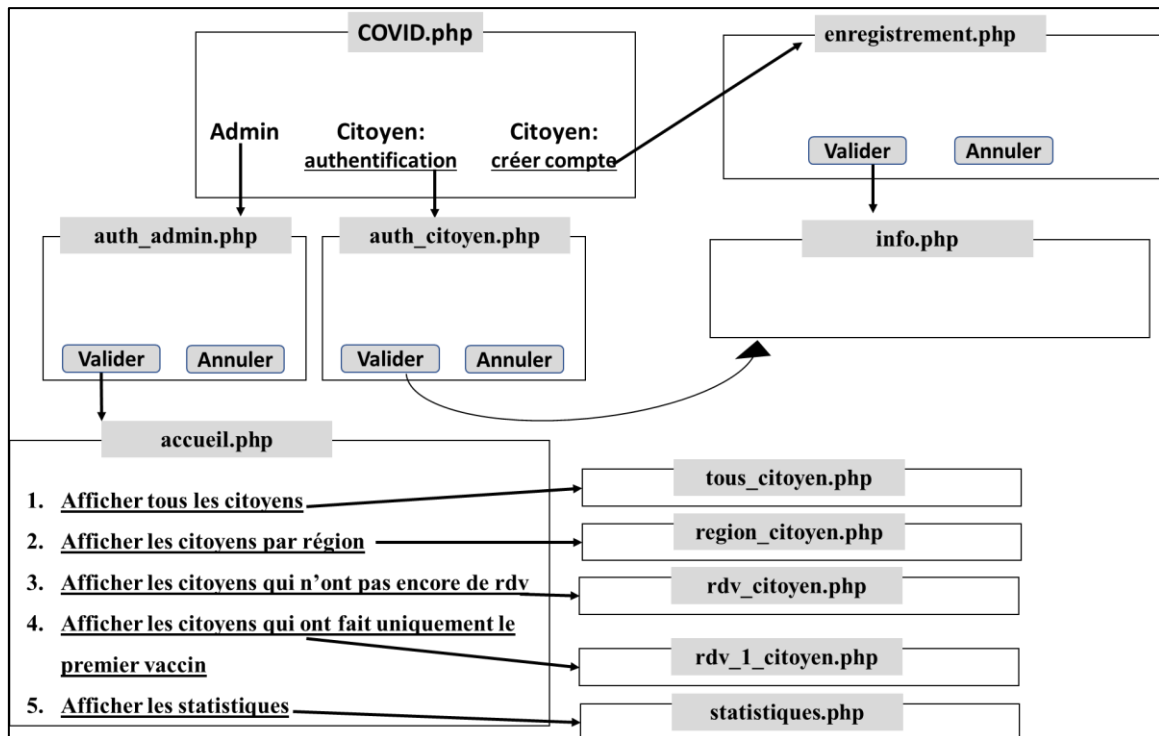


Figure 4. Structure du site Gestion de Rdv Covid

Références

- ◆ J. P. Vincent, J. Verrecchia, "HTML5: De la page web à l'application web", Series. Etude, développement et intégration, Dunod, 256 pages, isbn : 9782100568567, 2011.
<https://books.google.tn/books?id=eAHrCgAAQBAJ>
- ◆ R. Goetter, H. Giraudel, " CSS3 pratique du design web", Collection. Blanche, 4^{ème} édition, 372 pages, isbn : 978-2-212-67896-3, 2019.
<https://www.eyrolles.com/Informatique/Livre/css3-9782212678963/>
- ◆ M. Nebra, "Réussir son site web avec XHTML et CSS", Series. Accès libre, 3^{ème} édition 318 pages, isbn : 221209969X, 9782212099690, 2011.
<https://books.google.tn/books?id=QgCB-kgsoZUC>
- ◆ É. Daspet, C. P. De Geyer, "PHP 5 Avancé", Collection. Blanche, 6^{ème} édition, 857 pages, isbn : 978-2-212-134335-3, 2012.
<https://www.eyrolles.com/Informatique/Livre/php-5-avance-9782212113235/>