*HTML est responsable de la structure du contenu, CSS gère l'apparence visuelle et la mise en forme, tandis que JavaScript permet d'ajouter des interactions et des fonctionnalités dynamiques aux pages web.* **HTML** est le langage de balisage de base - N’est pas un langage de programmation - utilisé pour structurer le contenu d'une page web. Il définit la structure sémantique d'une page en utilisant des balises pour décrire différents types d'éléments tels que les titres, les paragraphes, les images, les liens, les listes,… **Contenu**: Ce que les navigateurs ignorent Plusieurs espaces blancs, Sauts de ligne (retour chariot), Tabs, Balisage inconnu. **Anatomie d'un élément HTML:**

Tag: Le nom de l'élément entre crochets

<link rel="stylesheet" href="…">

Element: Le contenu et son balisage (balises de début et de fin)

**<html>:** est l'élément racine qui contient tous les éléments du document.

**<ul> </ul>** : Définir la liste complète. Les éléments peuvent apparaître dans n'importequel ordre **<ol> </ol>** : Définir la liste complèt. Les éléments apparaissent dans un ordre particulier **<li> </li>** : Définir chaque élément de la liste

**<dl> </dl>** : Définir la liste complète **<dt> </dt>** : Définir un nom, tel qu'un terme **<dd> </dd>** : Définir une valeur, telle qu’une définition

**<q> </q>** : Les navigateurs ajoutent automatiquement des guillemets appropriés**.**

**<abbr> </abbr>** : est utilisée pour marquer une abréviation ou un acronyme dans une page web.

Le texte sélectionné est affiché dans une taille plus petite et légèrement au-dessus **<sup>** ou au-dessous **<sub>** de la base.

**<cite> </cite>:** Identifie une référence à un autre document.

**<dfn></dfn>** en HTML sont utilisées pour marquer et définir un terme dans un document web

**<code> </code>** Code dans le flux de texte**. <var> </var>** Variables

**<samp> </samp>** Exemple de programme**. <kbd> </kbd>** Coups du clavier entrés par l'utilisateurs

**<b> </b>** Le texte test mis en gras. **<i> </i>** Le texte est mis en italique. **<s> </s>** Texte incorrect (barré). **<u> </u>** Texte souligné, lorsque le soulignement a un but sémantique. **<small> </small>** Annexe ou note complémentaire (taille de texte inférieure). **<mark> </mark>** est utilisée pour mettre en évidence ou surligner un texte dans une page web. **<time> </time>** Fournit des équivalents lisibles par la machine pour les dates et les heures.

*<form action="URL" method="POST ou GET"> <!-- Form content and inputs here --></form>*

**Attributs**: Ce sont des instructions qui clarifient ou modifient un élément. La plupart des attributs prennent des valeurs qui suivent un signe =. Il peut avoir plus d’un attribut dans une balise

**Commentaires**: <!-- start global navigation -->

**<!DOCTYPE>:** indique aux navigateurs que le document est écrit conformément à la spécification HTML5.

L'attribut action fournit l'emplacement du script ou de l'application qui traitera les données de formulaire collectées. Method spécifie comment les informations encodées doivent être envoyées au serveur (GET ou POST):

– **GET**: Les données encodées sont clouées sur l'URL envoyée au serveur

– **POST**: les données sont envoyées dans une transaction distincte et peuvent être cryptées avec HTTPS.

**Saisie de texte** : *<input type="text" name="color" value="Red" maxlength="24">*

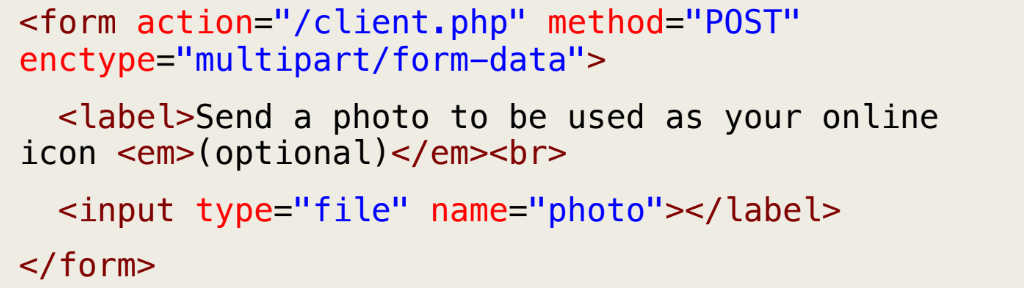
**Saisie de texte multiligne:** *<textarea name="entry" rows="6" cols="64"> This band is totally awesome!</textarea>*

**Type** : search, email, tel, url, date, number, range,color : Les navigateurs peuvent fournir des claviers adaptés au type d'entrée.

Submit : Soumet les données de formulaire collectées au serveur. Ne nécessite pas de nom de variable (attribut de name)

Reset : Réinitialise le formulaire à ses valeurs par défaut

File : Le type d'entrée de fichier permet à un utilisateur desélectionner un document de son disque dur à soumettre avec les données du formulaire. Hidden : Les contrôles masqués ne s'affichent pas dans le navigateur.



**Radio**: Un seul bouton radio peut être sélectionné à la fois.

*<****ol****> <****li****><input type="radio" name="age" value="under24" checked> …</****li****> <****li****><input type="radio" name="age" value="25-34"> 25 to 34</****li****> <****ol****>*

Checkbox : Plusieurs boutons de case à cocher peuvent être sélectionnés.

*<****ul****> <****li****><input type="checkbox" name="genre" value="punk" checked> Punk rock</****li****>*

*<****li****><input type="checkbox" name="genre" value="indie" checked>*

*Indie rock</****li****> </****ul****>*

**Select :** contient un certain nombre d'éléments d'option.

**Optgroup**: L'élément optgroup crée un groupe conceptuel d'options. L'attribut label fournit l'en-tête du groupe.

*<****select*** *name="EightiesFave"><****optgroup*** *label="traditional"> <****option****>The Cure</****option****> … </****optgroup****></****select****>*

**Fieldset** : Indique un regroupement logique de contrôles

**Legend** : Fournit une légende pour les champs fermés.

*<fieldset> <legend>Customer Information</legend>*

*<ul> <li><label>Full name: <input type="text" name="fullname"></label></li></ul> </fieldset>*

<a href="URL"> </a> : L'attribut href fournit l'emplacement (URL) de la ressource.

Chemin d’image src : *<img src = "/images/icons/next.svg" />*

Liens externes: vont vers des pages qui ne sont pas sur votre serveur.

mailto : créer une adresse électronique liée dans un programme de messagerie: *<a href="mailto:wonderwoman@example.com"> Envoyer un message à WonderWoman </a>*

**Tableau:** Identifie le matériel tabulaire

<tr>: Elle est utilisée pour définir une ligne dans un tableau HTML.

<td>: Elle est utilisée pour définir une cellule de données à l'intérieur d'une ligne de tableau (<tr>).

<th>: Elle est utilisée pour définir une cellule d'en-tête à l'intérieur d'une ligne de tableau (<tr>).

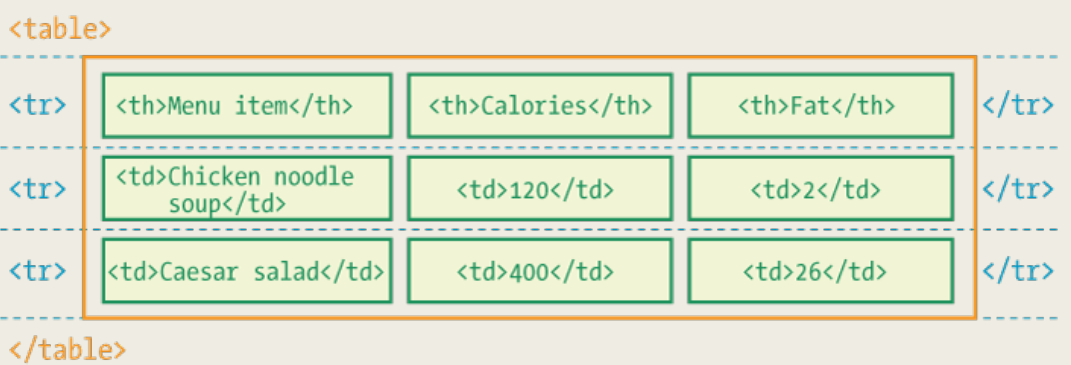
<thead>/<tbody>/<tfoot>: Cette balise est utilisée pour définir la section d'en-tête/de corps/de pied d'un tableau.

<colgroup> et <col> sont utilisées en HTML pour définir des règles de mise en forme des colonnes d'un tableau

*<table><colgroup> <col style="background-color: lightgray;">*

*<col style="background-color: lightblue;">*

*<col style="background-color: lightblue;"></colgroup> <tr>….. </tr> </table>*

****

Colspan : spécifier le nombre de colonnes sur lesquelles la cellule doit s'étendre

Rowspan : spécifie le nombre de lignes étendues.

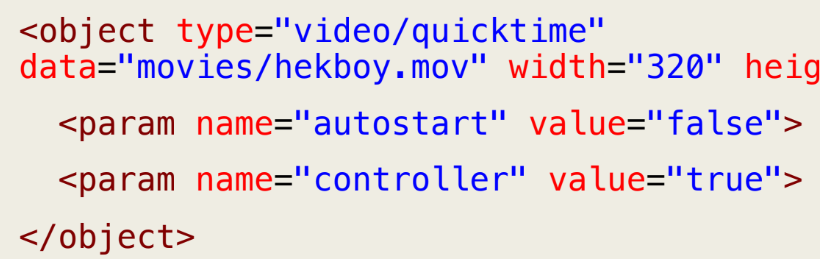
Caption : Fournit un titre ou une description pour le tableau. Améliore l'accessibilité au tableau

**Media**

<iframe src=””> : permet d'incorporer un document HTML distinct sur une page.

<object data="picture.svg" type="image/svg+xml"> : utilisée pour intégrer des contenus externes dans une page web.

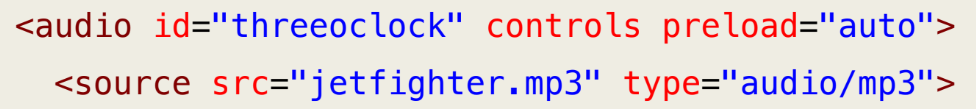
<param> : fournir des paramètres supplémentaires au contenu incorporé.



<video src=””> <img src=””> <source> : fournir plusieurs options de format vidéo.

A screen shot of a computer

Description automatically generated



L'élément <**canvas**> intègre une zone de dessin bitmap 2D contrôlée par les fonctions JavaScript.

**CSS** est utilisé pour définir la présentation visuelle et la mise en forme d'une page web. Il permet de contrôler les couleurs, les polices, les marges, les espacements, les dispositions et d'autres aspects visuels d'une page. En utilisant des sélecteurs, des propriétés et des valeurs, les développeurs peuvent appliquer des styles à différents éléments HTML.

Comment : /\* comment goes here \*/

**JavaScript** est un langage de programmation de script côté client - pas sur le serveur- qui permet d'ajouter des fonctionnalités interactives et dynamiques aux pages web. Avec JavaScript, vous pouvez manipuler le contenu HTML et les styles CSS après que la page a été chargée dans le navigateur. <script async/defer src="my\_script.js"></script>.

Si vos scripts n'ont pas besoin d'attendre l'analyse et peuvent s'exécuter indépendamment sans dépendances, utilisez **async**.

Si vos scripts doivent attendre l'analyse et dépendent d'autres scripts, chargez-les à l'aide de **defer** et placez leurs éléments <script> correspondants dans l'ordre dans lequel vous souhaitez que le navigateur les exécute.

Si vos scripts n'ont pas besoin d'attendre l'analyse et peuvent s'exécuter indépendamment sans dépendances, utilisez async.Si vos scripts doivent attendre l'analyse et dépendent d'autres scripts, chargez-les à l'aide de defer et placez leurs éléments <script> correspondants dans l'ordre dans lequel vous souhaitez que le navigateur les exécute.

Comment : // commentaire - /\* sur deux lignes\*/

**Variables**: Ne peut pas contenir d'espaces de caractères. Utilisez des sous-tiret \_ ou “CamelCase” à la place.

DOM: Document Object Model est une interface de programmation qui permet d’accéder au contenu d’un document et de le manipuler.

***getElementById*** spécifie un élément avec l'id "…".***innerHTML*** représente le contenu HTML dans cet élément.

***document.getElementsByTagName(name)*** Trouver les éléments par son nom de la balise name. ***document.getElementsByClassName(name)*** Trouver les éléments par son nom de la class name.

***querySelectorAll():*** Accède aux nœuds basés sur un sélecteur CSS.

***element.setAttribute(attribute, value)/ element.attribute = new value:*** Change la valeur d’attribut d’un élément HTML

***element.style.property = new style:*** Change le style d’un élément HTML

***document.createElement(element)*** Créer un élément HTML

***document.appendChild(element)*** Ajouter un élément HTML

***document.replaceChild(new, old)*** Remplacer un élément HTML

*var para = document.createElement("p");*

*var node = document.createTextNode("This is new.");*

*para.appendChild(node);*

*var parent = document.getElementById("div1");*

*var child = document.getElementById("p1");*

*parent.replaceChild(para, child);*

***document.removeChild(element)*** Enlever un élélent HTML

*var parent = document.getElementById("div1");*

*var child = document.getElementById("p1");*

*parent.removeChild(child);*

***document.write(text)*** Écrire dans le flux de sortie HTML

A screen shot of a computer code

Description automatically generated

*myButton.****addEventListener****('click’, function(){alert('Hello world');});*

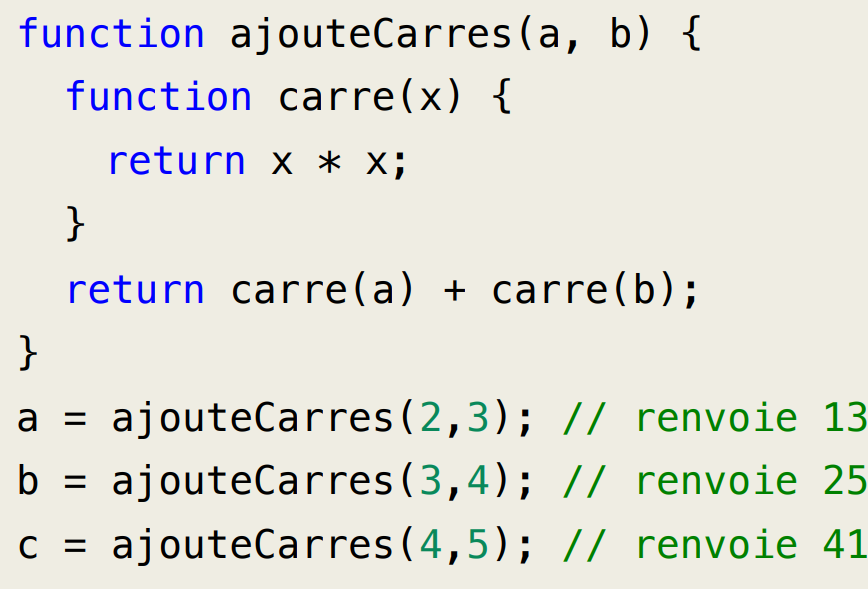
Les **onload** et les **onunload** événements sont déclenchés lorsque l'utilisateur entre ou quitte la page.

**onmouseover**/ **onmouseout** se déclenche lorsque l'utilisateur déplace la souris au dessus d'un élément particulier/ lorsque le pointeur de la souris quitte un élément.

**onmouseup** renvoie le gestionnaire de l'évènement mouseup. L'évènement mouseup se déclenche lorsque l'utilisateur relâche le bouton gauche de la souris. **onmousedown** renvoie le gestionnaire de l'évènement mousedown. L'évènement mousedown se déclenche lorsque l'utilisateur appuie sur le bouton gauche de la souris.

**Fonction** est un morceau de code permettant d’effectuer une tâche afin de l’appeler plus tard. Les parenthèses contiennent parfois des arguments.

A screen shot of a computer code

Description automatically generated 

Il est possible d'imbriquer une fonction dans une autre fonction. **La fonction imbriquée** ne peut être utilisée qu’à l’intérieur de la fonction parente. elle peut utiliser les arguments et les variables de la fonction parente. En revanche, la fonction parente ne peut pas utiliser les arguments et les variables de la fonction fille

Deux arguments ou variables ont le même nom à un conflit de noms. JavaScript permet d'imbriquer des fonctions et la fonction interne aura accès aux variables et paramètres de la fonction parente.

*function* ***multiplier****(facteur, ...lesArgs) { return lesArgs.map(x => facteur \* x);} var arr = multiplier(2, 1, 2, 3); // [2, 4, 6]*

**Créer de nouveaux objets:**

function Person(first, eye) { this.firstName = first; this.eyeColor = eye;}

**Class**: Les classes sont juste des fonctions spéciales sont définies de la même façon que les fonctions : par déclaration, ou par expression.

*class Rectangle { constructor(hauteur, largeur) {this.hauteur = hauteur;this.largeur = largeur;}}*

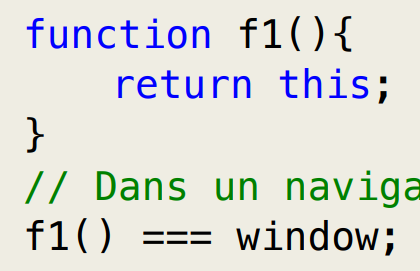
Le **corps**: la partie contenue entre les accolades.**Constructeur** est une méthode qui est utilisée pour créer et initialiser un objet lorsqu'on utilise le mot clé class : constructor([arguments]) { ... }

Le mot-clé **static** permet de définir une méthode statique d'une classe. Les méthodes statiques ne sont pas disponibles sur les instances d'une classe mais sont appelées sur la classe elle-même. **Déclarations :** class Rectangle { hauteur = 0; // publiques #largeur; // privés}

Mot-cles **This** comportement variera également légèrement selon qu'on utilise le mode strict ou le mode non-strict. Dans la plupart des cas, la valeur de this sera déterminée à partir de la façon dont une fonction est appelée.

// Si l'environnement de script est un navigateur, l'objet window sera l'objet global : *console.log(this === window); // true*

this.a = 37; console.log(window.a); // 37

 A white screen with black text

Description automatically generated

La fonction **bind()** crée une nouvelle fonction, lorsqu'elle est appelée, dans laquelle la valveur de this est passée en paramètre et éventuellement une suite d'arguments qui précéderont ceux fournis à l'appel de la fonction créée. Pour passer this d'un contexte à un autre, on pourra utiliser **call()** ou **apply().**

A screen shot of a computer code

Description automatically generated A screen shot of a computer code

Description automatically generated

**Callback :** Une fonction de rappel est une fonction passée dans une autre fonction en tant qu'argument, qui est ensuite invoquée à l'intérieur de la fonction externe pour accomplir une sorte de routine ou d'action.

**A screen shot of a computer

Description automatically generated**

**Promises :** qui représente la complétion ou l'échec d'une opération asynchrone.

**A screen shot of a computer code

Description automatically generated**

**fetch() :** Fonction pour charger des ressources en JavaScript

*fetch(pathToResource) .then(onResponse).then(onResourceReady);*

**Json**: Une syntaxe pour stocker et échanger des données.

Avantages : Facile à lire ou à écrire pour des humains, Aisément analysable ou générable par des machines. Un format texte complètement indépendant de tout langage. Un texte écrit avec la notation objet JavaScript.

**REST** : REpresentational State Transfer

**RESTful API** : L’API basée sur une URL ayant les propriétés suivantes:

**Les six contraintes sont:** Interface uniforme, Sans état, Avec mise en cache, Client- serveur, Système en couches, Avec code à la demande.

**1/L’objective 2/Envoyer Data 3/Caching 4/ Securité**

**GET**: 1/Obtenir des informations à partir de la ressource. 2/Pas de données envoyées avec la requête. Les informations sont transmises dans l'URL. 3/Peut être mis en cache. Lorsque vous envoyez une requête GET avec les mêmes informations, la réponse peut être renvoyée depuis le cache. 4/Niveau de sécurité faible. Les informations sont transmises dans l'URL et peuvent être exposées publiquement.

**POST:** 1/Créer / Effectuer une action sur la ressource. 2/Envoyer des données avec la requête. Les données sont généralement envoyées dans le corps (body) de la requête. 3/Généralement pas mis en cache. Les réponses ne sont généralement pas marquées pour le cache. 4/Niveau de sécurité moyen. Les données envoyées ne sont pas affichées directement dans l'URL, mais peuvent être vulnérables au vol.

**DELETE :** 1/Supprimer la ressource existante.2/Pas de données envoyées avec la requête, ou très peu. Les informations nécessaires pour identifier la ressource à supprimer sont généralement transmises dans l'URL. 3/Généralement pas mis en cache. Les réponses ne sont généralement pas marquées pour le cache. 4/Niveau de sécurité faible pour la suppression de données.

**PUT** : 1/Mise à jour de la ressource existante ou création si inexistante. 2/ Envoyer des données avec la requête, similaire à POST. Les données sont généralement envoyées dans le corps (body) de la requête. 3/Généralement pas mis en cache. 4/Niveau de sécurité élevé. Les données envoyées ne sont pas affichées directement dans l'URL.

**Serveurs** : Exécutent des programmes pour générer des pages Web et

autres ressources Web. Stocker et manipuler le base de donné.

**Sockets**: Un socket est un canal de communication. Vous pouvez

envoyer et recevoir des données sur des sockets.

**Nodejs** est un environnement d'exécution open-source basé sur le moteur JavaScript V8 de Google Chrome. Il permet d'exécuter du code JavaScript côté serveur.

**MongoDB** est une base de données orientée documents, également classée dans la catégorie des bases de données NoSQL.