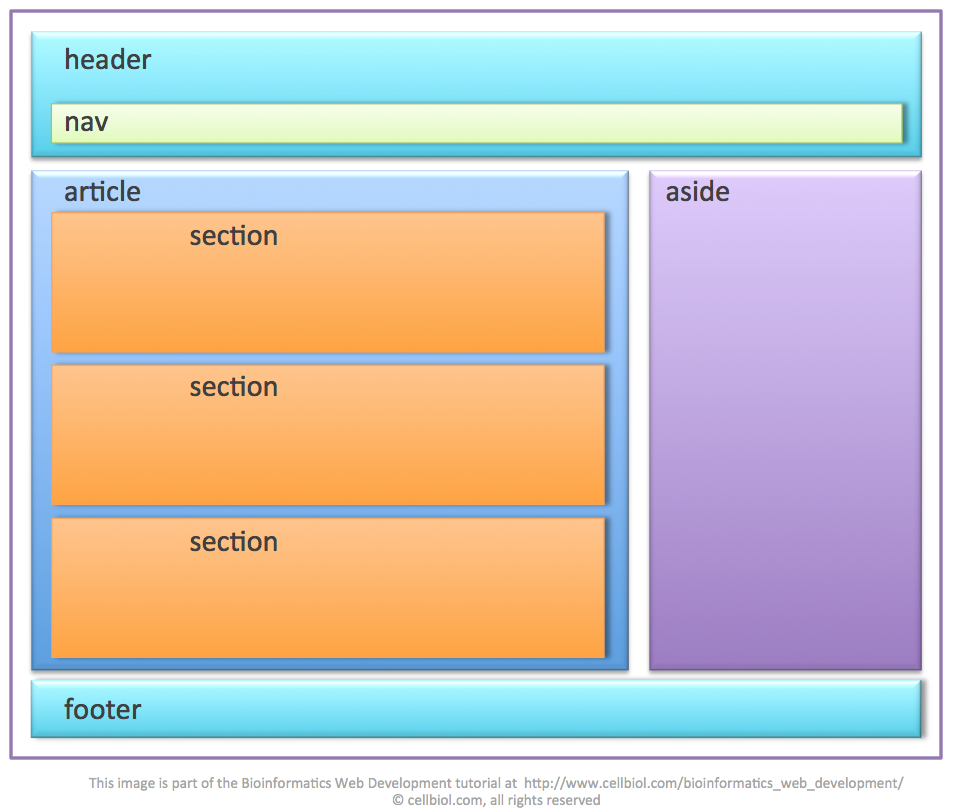
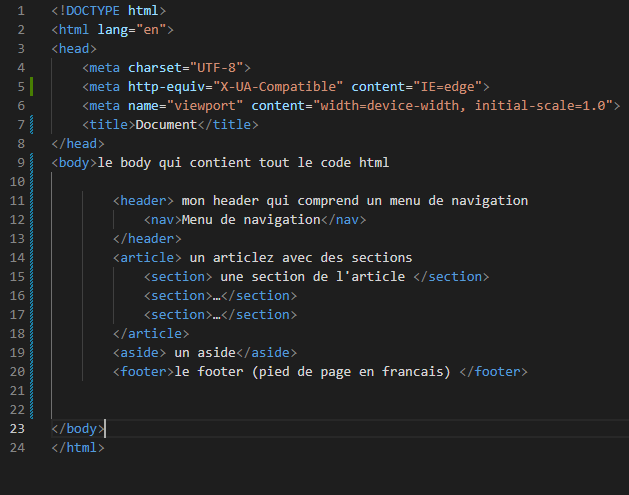
Cours HTML CSS JAVASCRIPT

Votre ECF "Front-end" se déroulera le mercredi 06/07/2022 ce sera l'occasion de faire le point sur la maitrise du HTML5 et CSS3 pour faire des composants d'interface comme un menu animé à partir de liste puce, ou un menu par Onglet par exemple ou encore un formulaire bien mise en forme, un diaporama. La partie JS portera sur les contrôles de saisies, manipulation du DOM, et la consommation d'API JSON (méthode JS Fetch() ou Objet XHR) pour alimenter un diaporama ou une liste d'annonces ou de produits( générés dynamiquement). Durée de l’épreuve de 07h00. ( 4 heures le matin, 3 heures l'après-midi)

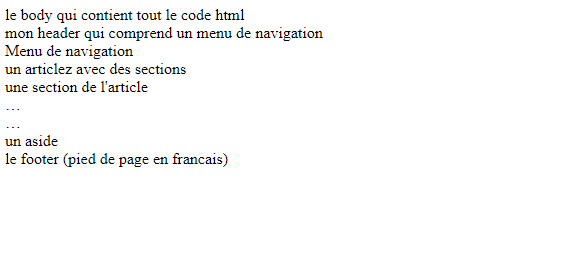
Plan type d’une page html5 :



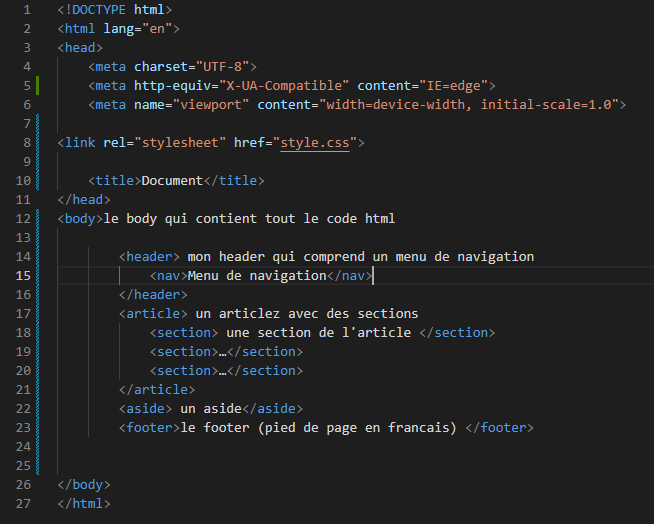
Voici un plan de page html5 qu’on pourrait écrire en code HTML5 comme décrit ci-dessous :



Regardez le résultat cela est moche non ?



En effet aucun style CSS n’est appliqué. Pour l’appliqué il faut relier un fichier CSS au html Reprenons cette exemple : comme vous pouvez le voir j’ai rajoutez une balise link qui sert à relier un fichier css du même nom ici (style.css).



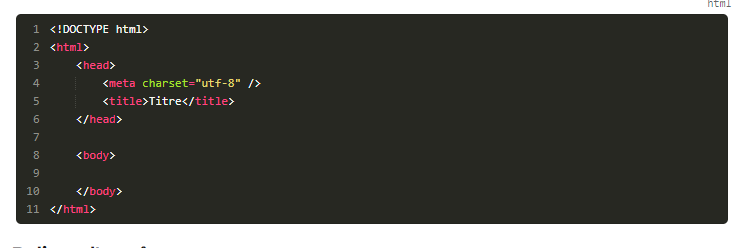
**Mémento HTML5**

**Balises de premier niveau**

Les balises de premier niveau sont les principales balises qui structurent une page HTML. Elles sont indispensables pour réaliser le « code minimal » d'une page web.

| **Balise** | **Description** |
| --- | --- |
| <html> | Balise principale |
| <head> | En-tête de la page |
| <body> | Corps de la page |

Code minimal d'une page HTML :



**Balises d'en-tête**

Ces balises sont toutes situées dans l'en-tête de la page web, c'est-à-dire entre <head>  et </head>  :

| **Balise** | **Description** |
| --- | --- |
| <link /> | Liaison avec une feuille de style |
| <meta /> | Métadonnées de la page web (charset, mots-clés, etc.) |
| <script> | Code JavaScript |
| <style> | Code CSS |
| <title> | Titre de la page |

**Balises de structuration du texte**

| **Balise** | **Description** |
| --- | --- |
| <abbr> | Abréviation |
| <blockquote> | Citation (longue) |
| <cite> | Citation du titre d'une œuvre ou d'un évènement |
| <q> | Citation (courte) |
| <sup> | Exposant |
| <sub> | Indice |
| <strong> | Mise en valeur forte |
| <em> | Mise en valeur normale |
| <mark> | Mise en valeur visuelle |
| <h1> | Titre de niveau 1 |
| <h2> | Titre de niveau 2 |
| <h3> | Titre de niveau 3 |
| <h4> | Titre de niveau 4 |
| <h5> | Titre de niveau 5 |
| <h6> | Titre de niveau 6 |
| <img /> | Image |
| <figure> | Figure (image, code, etc.) |
| <figcaption> | Description de la figure |
| <audio> | Son |
| <video> | Vidéo |
| <source> | Format source pour les balises <audio>  et <video> |
| <a> | Lien hypertexte |
| <br /> | Retour à la ligne |
| <p> | Paragraphe |
| <hr /> | Ligne de séparation horizontale |
| <address> | Adresse de contact |
| <del> | Texte supprimé |
| <ins> | Texte inséré |
| <dfn> | Définition |
| <kbd> | Saisie clavier |
| <pre> | Affichage formaté (pour les codes sources) |
| <progress> | Barre de progression |
| <time> | Date ou heure |

**Balises de listes**

Cette section énumère toutes les balises HTML permettant de créer des listes (listes à puces, listes numérotées, listes de définitions…)

| **Balise** | **Description** |
| --- | --- |
| <ul> | Liste à puces, non numérotée |
| <ol> | Liste numérotée |
| <li> | Élément de la liste à puces |
| <dl> | Liste de définitions |
| <dt> | Terme à définir |
| <dd> | Définition du terme |

**Balises de tableau**

| **Balise** | **Description** |
| --- | --- |
| <table> | Tableau |
| <caption> | Titre du tableau |
| <tr> | Ligne de tableau |
| <th> | Cellule d'en-tête |
| <td> | Cellule |
| <thead> | Section de l'en-tête du tableau |
| <tbody> | Section du corps du tableau |
| <tfoot> | Section du pied du tableau |

**Balises de formulaire**

| **Balise** | **Description** |
| --- | --- |
| <form> | Formulaire |
| <fieldset> | Groupe de champs |
| <legend> | Titre d'un groupe de champs |
| <label> | Libellé d'un champ |
| <input /> | Champ de formulaire (texte, mot de passe, case à cocher, bouton, etc.) |
| <textarea> | Zone de saisie multiligne |
| <select> | Liste déroulante |
| <option> | Élément d'une liste déroulante |
| <optgroup> | Groupe d'éléments d'une liste déroulante |

**Balises sectionnantes**

Ces balises permettent de construire le squelette de notre site web.

| **Balise** | **Description** |
| --- | --- |
| <header> | En-tête |
| <nav> | Liens principaux de navigation |
| <footer> | Pied de page |
| <section> | Section de page |
| <article> | Article (contenu autonome) |
| <aside> | Informations complémentaires |

**Balises génériques**

Les balises génériques sont des balises qui n'ont pas de sens sémantique.

En effet, toutes les autres balises HTML ont un *sens* : <p>  signifie « paragraphe », <h2>  signifie « sous-titre », etc.  
Parfois, on a besoin d'utiliser des balises génériques (aussi appelées ***balises universelles***) car aucune des autres balises ne convient. On utilise le plus souvent des balises génériques pour construire son design.

Il y a deux balises génériques : l'une est inline, l'autre est block.

| **Balise** | **Description** |
| --- | --- |
| <span> | Balise générique de type inline |
| <div> | Balise générique de type block |

Ces balises ont un intérêt uniquement si vous leur associez un attribut class  , id  ou style  :

* **class**  : indique le nom de la classe CSS à utiliser.
* **id**  : donne un nom à la balise. Ce nom doit être unique sur toute la page car il permet d'identifier la balise. Vous pouvez vous servir de l'ID pour de nombreuses choses, par exemple pour créer un lien vers une ancre, pour un style CSS de type ID, pour des manipulations en JavaScript, etc.
* **style**  : cet attribut vous permet d'indiquer directement le code CSS à appliquer. Vous n'êtes donc pas obligé d'avoir une feuille de style à part, vous pouvez mettre directement les attributs CSS. Notez qu'il est préférable de ne pas utiliser cet attribut et de passer à la place par une feuille de style externe, car cela rend votre site plus facile à mettre à jour par la suite.

Ces trois attributs ne sont pas réservés aux balises génériques : vous pouvez aussi les utiliser sans aucun problème dans la plupart des autres balises.

**Memento css**

Je résume ici la plupart des propriétés de **mise en forme du texte**.

Qu'est-ce que la mise en forme de texte ? C'est tout ce qui touche à la présentation du texte proprement dit : le gras, l'italique, le souligné, la police, l'alignement, etc.

| **Propriété** | **Valeurs (exemples)** | **Description** |
| --- | --- | --- |
| font-family | *police1, police2, police3*, serif, sans-serif, monospace | Nom de police |
| @font-face | *Nom et source de la police* | Police personnalisée |
| font-size | 1.3em, 16px, 120%... | Taille du texte |
| font-weight | bold, normal | Gras |
| font-style | italic, oblique, normal | Italique |
| text-decoration | underline, overline, line-through, blink, none | Soulignement, ligne au-dessus, barré ou clignotant |
| font-variant | small-caps, normal | Petites capitales |
| text-transform | capitalize, lowercase, uppercase | Capitales |
| font | - | Super propriété de police. Combine : font-weight  , font-style  , font-size  , font-variant  , font-family  . |
| text-align | left, center, right, justify | Alignement horizontal |
| vertical-align | baseline, middle, sub, super, top, bottom | Alignement vertical (cellules de tableau ou éléments inline-block  uniquement) |
| line-height | 18px, 120%, normal... | Hauteur de ligne |
| text-indent | 25px | Alinéa |
| white-space | pre, nowrap, normal | Césure |
| word-wrap | break-word, normal | Césure forcée |
| text-shadow | 5px 5px 2px blue *(horizontale, verticale, fondu, couleur)* | Ombre de texte |

**Propriétés de couleur et de fond**

| **Propriété** | **Valeurs (exemples)** | **Description** |
| --- | --- | --- |
| color | *nom*, rgb(rouge,vert,bleu), rgba(rouge,vert,bleu,transparence), #CF1A20... | Couleur du texte |
| background-color | *Identique à color* | Couleur de fond |
| background-image | url('image.png') | Image de fond |
| background-attachment | fixed, scroll | Fond fixe |
| background-repeat | repeat-x, repeat-y, no-repeat, repeat | Répétition du fond |
| background-position | *(x y)*, top, center, bottom, left, right | Position du fond |
| background | - | Super propriété du fond. Combine :  background-image  , background-repeat  , background-attachment  , background-position |
| opacity | 0.5 | Transparence |

**Propriétés des boîtes**

| **Propriété** | **Valeurs (exemples)** | **Description** |
| --- | --- | --- |
| width | 150px, 80%... | Largeur |
| height | 150px, 80%... | Hauteur |
| min-width | 150px, 80%... | Largeur minimale |
| max-width | 150px, 80%... | Largeur maximale |
| min-height | 150px, 80%... | Hauteur minimale |
| max-height | 150px, 80%... | Hauteur maximale |
| margin-top | 23px | Marge en haut |
| margin-left | 23px | Marge à gauche |
| margin-right | 23px | Marge à droite |
| margin-bottom | 23px | Marge en bas |
| margin | 23px 5px 23px 5px *(haut, droite, bas, gauche)* | Super-propriété de marge. Combine : margin-top  , margin-right  , margin-bottom  , margin-left  . |
| padding-left | 23px | Marge intérieure à gauche |
| padding-right | 23px | Marge intérieure à droite |
| padding-bottom | 23px | Marge intérieure en bas |
| padding-top | 23px | Marge intérieure en haut |
| padding | 23px 5px 23px 5px *(haut, droite, bas, gauche)* | Super-propriété de marge intérieure. Combine : padding-top  , padding-right  , padding-bottom  , padding-left  . |
| border-width | 3px | Épaisseur de bordure |
| border-color | *nom*, rgb(rouge,vert,bleu), rgba(rouge,vert,bleu,transparence), #CF1A20... | Couleur de bordure |
| border-style | solid, dotted, dashed, double, groove, ridge, inset, outset | Type de bordure |
| border | 3px solid black | Super-propriété de bordure. Combine ,border-width  , border-color  , border-style  . Existe aussi en versionborder-top  , border-right  , border-bottom  , border-left  . |
| border-radius | 5px | Bordure arrondie |
| box-shadow | 6px 6px 0px black *(horizontale, verticale, fondu, couleur)* | Ombre de boîte |

**Propriétés de positionnement et d'affichage**

| **Propriété** | **Valeurs (exemples)** | **Description** |
| --- | --- | --- |
| display | block, inline, inline-block, table, table-cell, none... | Type d'élément ( block  , inline  , inline-block  , none  …) |
| visibility | visible, hidden | Visibilité |
| clip | rect (0px, 60px, 30px, 0px) *rect (haut, droite, bas, gauche)* | Affichage d'une partie de l'élément |
| overflow | auto, scroll, visible, hidden | Comportement en cas de dépassement |
| float | left, right, none | Flottant |
| clear | left, right, both, none | Arrêt d'un flottant |
| position | relative, absolute, static | Positionnement |
| top | 20px | Position par rapport au haut |
| bottom | 20px | Position par rapport au bas |
| left | 20px | Position par rapport à la gauche |
| right | 20px | Position par rapport à la droite |
| z-index | 10 | Ordre d'affichage en cas de superposition. La plus grande valeur est affichée par-dessus les autres. |

**Propriétés des listes**

| **Propriété** | **Valeurs (exemples)** | **Description** |
| --- | --- | --- |
| list-style-type | disc, circle, square, decimal, lower-roman, upper-roman, lower-alpha, upper-alpha, none | Type de liste |
| list-style-position | inside, outside | Position en retrait |
| list-style-image | url('puce.png') | Puce personnalisée |
| list-style | - | Super-propriété de liste. Combine list-style-type  , ,list-style-position  , list-style-image  . |

**Propriétés des tableaux**

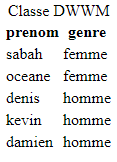
| **Propriété** | **Valeurs (exemples)** | **Description** |
| --- | --- | --- |
| border-collapse | collapse, separate | Fusion des bordures |
| empty-cells | hide, show | Affichage des cellules vides |
| caption-side | bottom, top | Position du titre du tableau |

**Autres propriétés**

| **Propriété** | **Valeurs (exemple)** | **Description** |
| --- | --- | --- |
| cursor | crosshair, default, help, move, pointer, progress, text, wait, e-resize, ne-resize, auto... | Curseur de souris |

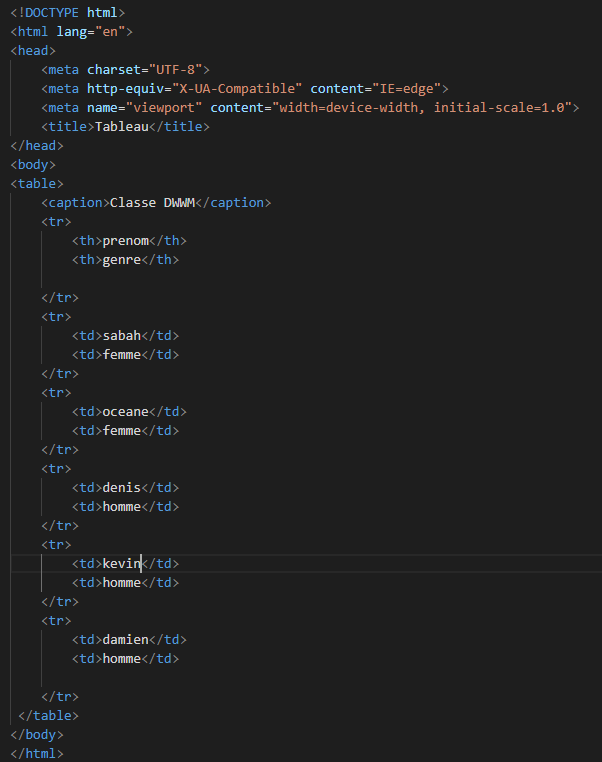
**Création de tableau en html css**

Voici un tableau simple dans un navigateur internet :



C’est bien mais vous allez me dire <<comment on réalise cela ?>>

C’est simple et sans danger analysons le code html suivant:



Un tableau en HTML ce place dans la balise **<table>** Le code du tableau **</table>**

La balise <caption> permet de créer un titre global de la table ici Classe dwwm.

<tr> </tr> : indique le début et la fin d'une ligne du tableau .

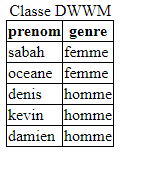
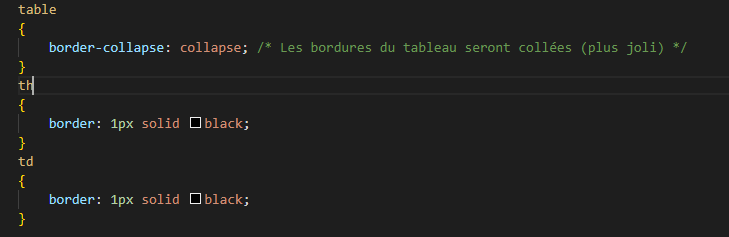
<th> </th> : indique le début et la fin d’une cellule titre du tableau. C’est-à-dire de la première ligne du tableau ou on placera les titres des colonnes.

<td> </td> : indique le début et la fin du contenu d'une cellule.

En HTML, un tableau se construit ligne par ligne. Dans chaque ligne ( <tr> ), on indique le contenu des différentes cellules ( <td> ).

C’est bien mais ça serais pas mieux avec une belle mise en page ( un peu de css quoi ).

Rajoutons ce code css dans un fichier css :

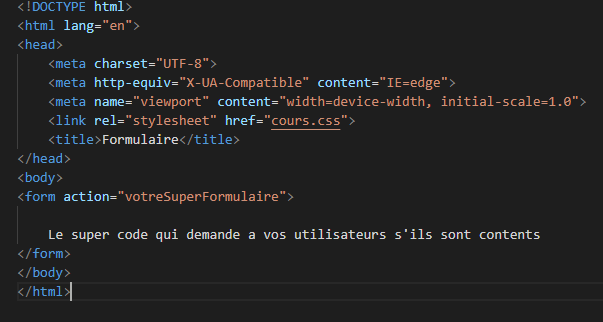


Ça a quand même plus d’allure non ?

**Création formulaire**

Et si on demandait à vos visiteurs de votre génial super site web s’ils ont été contents ?

Comment me direz-vous. Justement avec formulaire.

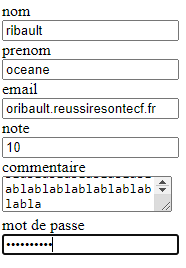


Vous devez pour commencer écrire une balise <form> </form> . C'est la balise principale du formulaire, elle permet d'en indiquer le début et la fin.

Remplissons notre formulaire et voyons de quoi il retourne :



Ce qui donne dans un navigateur web :



Détaillons un peut se code html sans vous faire peur :

<label>  : Sert a indiqué au visiteur ce que il doit écrire dans votre formulaire.

<input type=  "text" > : Pour rentrer un texte

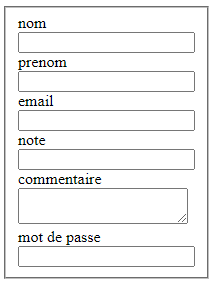
<input type=  "email" > : Afin de saisir un email

<input type=  "number" > : Pour sélectionner un nombre

<textarea > : Pour rentrer un long texte

<input type=  "password" > : Pour rentrer un mot de passe

Et si on encapsule la balise <form> dans une balise <fieldset> cela rassemble les éléments du formulaire ce qui donne dans un navigateur web:



**Memo javascript**

Une variable permet d’associer un nom à une valeur. La valeur peut prendre plusieurs formes (texte, nombre, boolean, etc.). On la déclare avec le mot clé  let .

*Exemple*:



**Constante**

Une constante est une variable qui a la particularité de ne pas pouvoir changer de valeur dans le temps. On la déclare avec le mot clé  const  .

*Exemple*:

C:\Users\Stagiaire\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Const.png

**Type**

Un type correspond à la nature des valeurs que peut prendre une donnée (string, number, array, etc).

*Exemple*:



Selon son type on peut appliquer divers actions sur une variable :

* Les variables  number  pourront s’additionner, multiplier, etc.
* Les variables  string  pourront être converti en majuscule avec la fonction  maString.toUpperCase()
* Les autres types ont chacun des fonctionnalités associées.

**Programmation orientée Objet (POO)**

La programmation orientée objet est un paradigme (concept) de programmation qui consiste à définir et faire interagir des éléments qu’on appelle “objets”. Un “objet” est un concept, une représentation, une idée qui se rattache au monde physique. Par exemple, un livre, une page de livre, une lettre, etc.

**Classe**

En POO, une classe regroupe les méthodes et propriétés (attributs) qui se rapportent à un objet. Elle définit ce qu’est un objet et ce que l’on peut faire avec.

*Exemple*:*C:\Users\Stagiaire\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Class.png*

**Instance de classe**

L’instance de classe est la création d’un objet unique basée sur la définition d’une classe. Une instance est donc un élément unique créé à partir d’une classe. Par exemple, à partir d’une classe Maison, on peut créer 2 instances qui seront 2 maisons différentes mais basées sur les mêmes plans de construction.

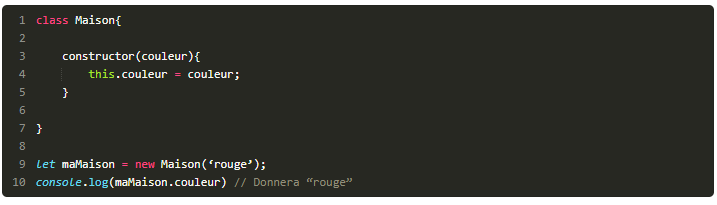
*Exemple*:



**Propriété/attribut de classe**

La propriété (dite aussi *attribut*) de classe est une variable interne à une classe qui pourra évoluer. Par exemple, un objet maison a un nombre de portes de base, mais cette valeur peut évoluer selon les actions appliquées sur cette maison (des travaux, par exemple).

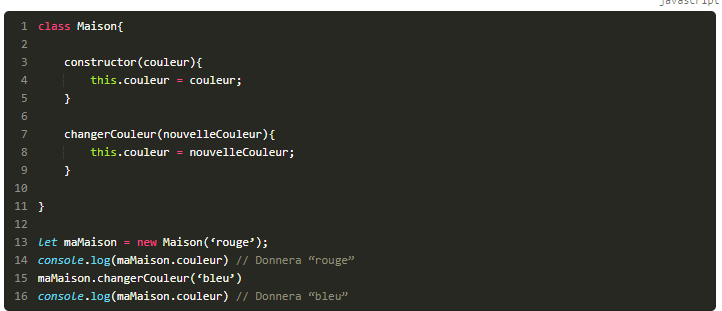
*Exemple*:



**Méthode de classe**

La méthode de classe est une fonction interne à la classe qui permet d'exécuter des actions au sein de la classe instanciée. Par exemple, la méthode “peindre(couleur)” peut changer la propriété “couleur” de l’objet maison.

*Exemple*:



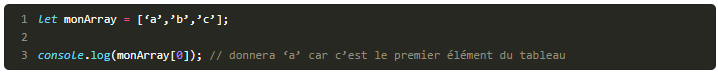
**Collection**

Les collections sont des types de données qui permettent de ranger un ensemble de données dans une seule variable (comme ranger de la donnée dans un tableau). On trouve principalement deux types de collection : les Array et les Object.

**Collection de type Array**

Une collection de type Array permet de créer une liste ordonnée d’éléments. On accède à l’un de ces éléments par son index. Il faut savoir que le premier index est 0 et non pas 1.

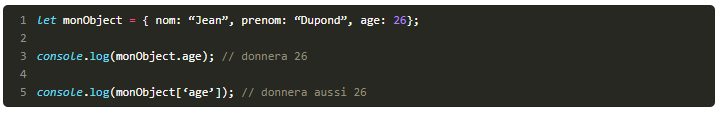
*Exemple*:

**

**Collection de type Object**

Une collection de type Object est une liste non ordonnée qui ne fonctionne pas à partir d’index mais à partir de clé à laquelle est rattachée une valeur. Donc pour récupérer une valeur, on doit renseigner la clé de cette valeur.

*Exemple*:



**Bloc de code**

Un bloc de code est une partie du code commençant par un crochet ouvrant  {  et terminant par un crochet fermant}.

*Exemple*:

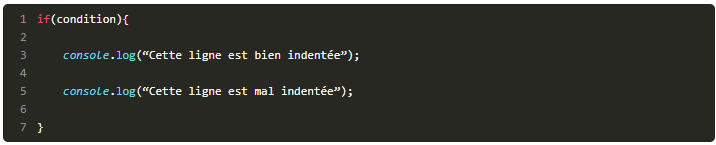


**Indentation**

C’est la mise en forme des lignes de code pour rendre la lecture plus simple et explicite.

La règle est de mettre le code en retrait par rapport à la ligne précédente pour signifier que la ligne d’un niveau inférieur. Dans le cas de l’écriture d’un bloc de code, cela permet de mieux comprendre où commence et où s'arrête chaque bloc.

*Exemple*:

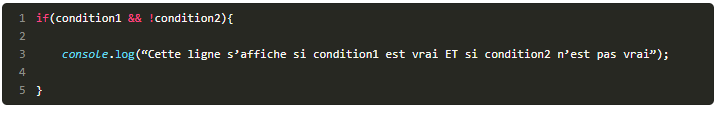


**Opérateurs logiques**

Ce sont des caractères spéciaux qui permettent de faire des conditions plus complexes en liant plusieurs conditions, ou en faisant des comparaisons de valeurs. On retrouve les caractères suivants : &&, ||, <, >, >=, <=, ==, ===, != et !.

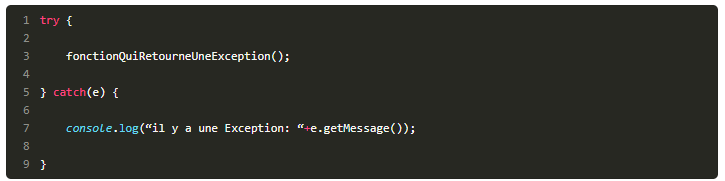
*Exemple*:

**Exception**



Dès qu’il y a une erreur liée à un défaut d'exécution, une erreur est soulevée en JavaScript, on appelle cela une Exception. On le remarque souvent dans la console en rouge. Une Exception contient des informations permettant de comprendre les raisons du problème, et on peut facilement récupérer cette Exception au sein du code pour faire un traitement personnalisé grâce au bloc  try    { }     catch(e){  }  .

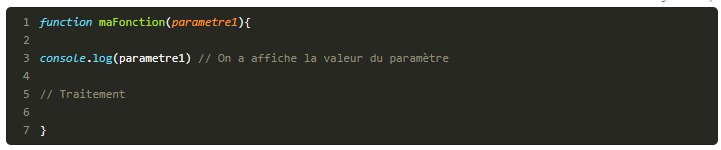
*Exemple*:



**Paramètre de fonction**

Un paramètre est une valeur qu’une fonction attend en entrée. On lui donne un nom comme une variable qui servira au sein de la fonction pour le traitement. Il peut bien sûr y avoir plusieurs paramètres dans une fonction.

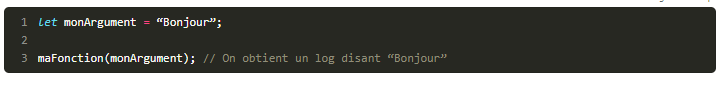
*Exemple*:



**Argument de fonction**

L’argument d’une fonction est la valeur qu’on va injecter en entrée d’une fonction au moment de l'exécution. Ce concept est lié directement à la notion de paramètre de fonction, car en général pour chaque paramètre d’une fonction, on devra apporter un argument.

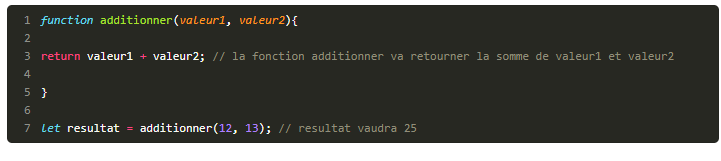
*Exemple*:



**Valeur de retour**

Lors de l'exécution d’une fonction, il y a plusieurs traitements réalisés en son sein, et une valeur finale pourra être retournée avec le mot clé “return”. Cela permet de récupérer la valeur retournée par la fonction dans une variable, par exemple, et de l’exploiter dans le reste du code.

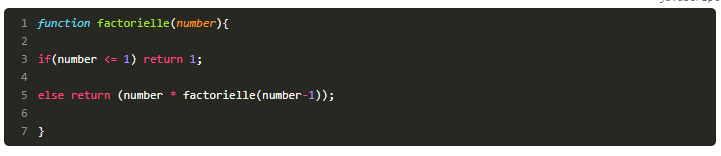
*Exemple*:



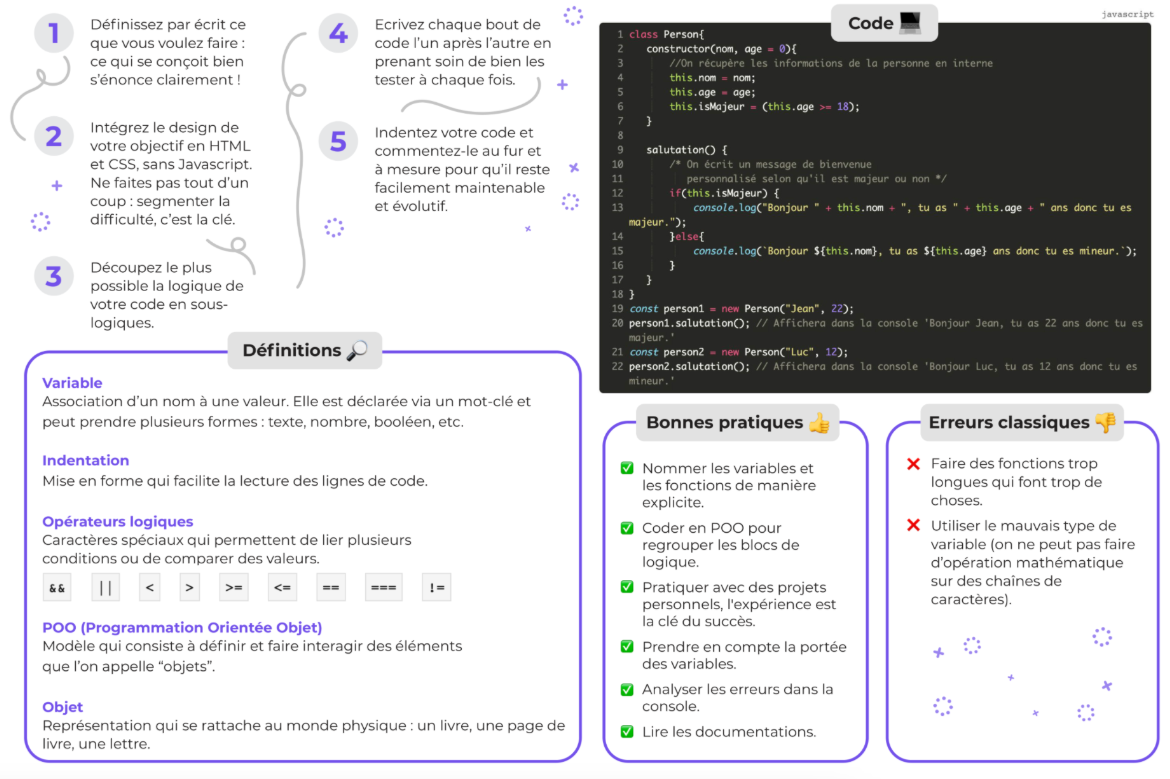
**Récursivité**

La récursivité est un concept dans lequel un élément fait appel à lui-même. Par exemple, une fonction récursive est une fonction qui s'appelle elle-même d'une façon ou d'une autre.

*Exemple*:



***Résumé:***



**Manipulation du DOM en javascript:**

Au-delà de jouer avec des boucles et des chaînes de caractères, JavaScript permet de manipuler un document et de jouer avec le HTML et/ou son CSS, le tout en temps réel.

Le document que nous allons pouvoir manipuler, appelé le DOM *(Document Object Model)* est, par exemple, la page sur laquelle vous vous trouvez à l'instant même ! Il ne s'agit pas de modifier un ensemble de pages, mais bien la page en cours.  
Évidemment, dans le cadre de cet article, nous ne modifierons pas cette page. Cependant, nous supposerons que vous avez une page en cours de développement, sur laquelle nous pourrons appliquer quelques essais.

**DOM**

C'est une arborescence créée par votre navigateur contenant la structure HTML de la page Web chargée, ce qui permet à JavaScript de manipuler ce document.



Le code afficher ici c’est le **DOM**

**querySelector() & querySelectorAll()**

Grâce aux méthodes querySelector() et querySelectorAll(), nous allons pouvoir sélectionner un ou plusieurs éléments du DOM et y appliquer par la suite nos différents ajouts ou modifications.

Elles attendent un argument afin de sélectionner l'élément voulu. Vous pouvez leur passer le nom du tag HTML (a, section, img, div…), le nom d'un sélecteur CSS (.ma\_classe, .container…) ou l'ID de l'élément désiré (#mon\_id, #first, ...).

**querySelector()**

Cette méthode sélectionnera le premier élément trouvé selon l'argument passé.

*let* paragraph = *document*.querySelector('p');

Dans cet exemple, le premier élément HTML p repéré sera sélectionné et les suivants seront ignorés.

**querySelectorAll()**

Celle-ci sélectionnera tous les éléments repérés selon l'argument passé. Il faudra utiliser une boucle afin de retrouver chaque élément un à un.

*let* buttons = *document*.querySelectorAll('.btn .btn-danger');

**utilisation du getElementsByTagName(), getElementsByClassName() & getElementById()**

**getElementsByTagName()**

Placer purement le nom du tag HTML que vous souhaitez récupérer. Cette méthode vous retournera une collection HTML, soit un tableau. À lire avec une boucle si vous voulez intervenir sur les différents éléments retrouvés.

*let* paragraphs = *document*.getElementsByTagName('p');

**getElementsByClassName()**

Utilisez la classe CSS afin de retrouver les éléments HTML qui l'utiliseraient. Cette dernière vous retournera une collection HTML, soit un tableau. À lire avec une boucle si vous souhaitez intervenir sur les différents éléments retrouvés.

*let* containers = *document*.getElementsByClassName('container');

**getElementById()**

Utilisez l'ID dans le but de retrouver l'élément HTML qui l'utiliserait. Cette dernière vous retournera un seul élément, le premier trouvé. Pas besoin de boucle.

*let* list = *document*.getElementById('listUser');

**Les propriétés**

Une fois votre ou vos éléments retrouvés, vous avez une multitude de propriétés permettant d'y apporter des modifications. Nous allons en voir quelques-unes, les plus utilisées :

**innerHTML**  
Reprends ou change la syntaxe HTML à l'intérieur de l'élément.

**outerHTML**  
Reprends ou modifie la syntaxe HTML complète de l'élément.

**innertText**  
Reprends ou modifie la syntaxe textuelle à l'intérieur de l'élément.

**textContent**  
Reprends ou modifie la syntaxe textuelle à l'intérieur de l'élément ainsi que des ascendants.

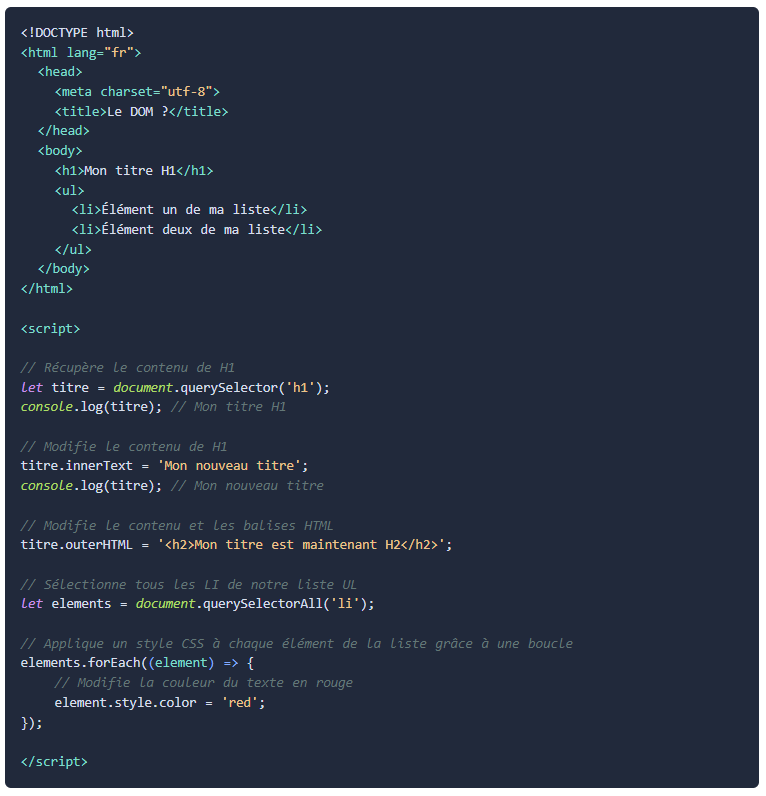
**dataset**  
Reprends, ajoute ou modifie une valeur contenue dans un attribut "data" d'un élément.

**setAttribute/getAttribute**  
Reprends ou modifie n'importe quel attribut d'un élément.

**classList**  
Permet de "jouer" sur les classes CSS d'un élément.

**style**  
Aide à jouer sur le style CSS d'un élément.

Exemple :



**Créer un nouveau nœud**

Jouer avec le DOM pour en modifier des valeurs, ajouter des classes CSS… c'est bien, mais on peut aussi créer de nouveaux nœuds et cela facilement !

Avant de créer un nouveau nœud, nous allons retrouver un élément dans lequel nous mettrons justement cette nouvelle création. Pour l'exemple, j'ajoute une nouvelle li à ma liste.

Dans un premier temps, je reprends ma ul :

*let* liste = *document*.querySelector('ul');

Maintenant, je crée mon nouvel élément :

*let* li = *document*.createElement('li');

J'ajoute le texte que je souhaite dans ce nouvel élément :

li.innerText = 'Element trois de ma liste';

Enfin, je peux l'ajouter à la fin de ma liste :

liste.appendChild(li);

**Supprimer un nœud**

Cette action est assez simple. À partir du moment où vous avez repris un élément, la suppression est facile :

*let* liste = *document*.querySelector('ul');

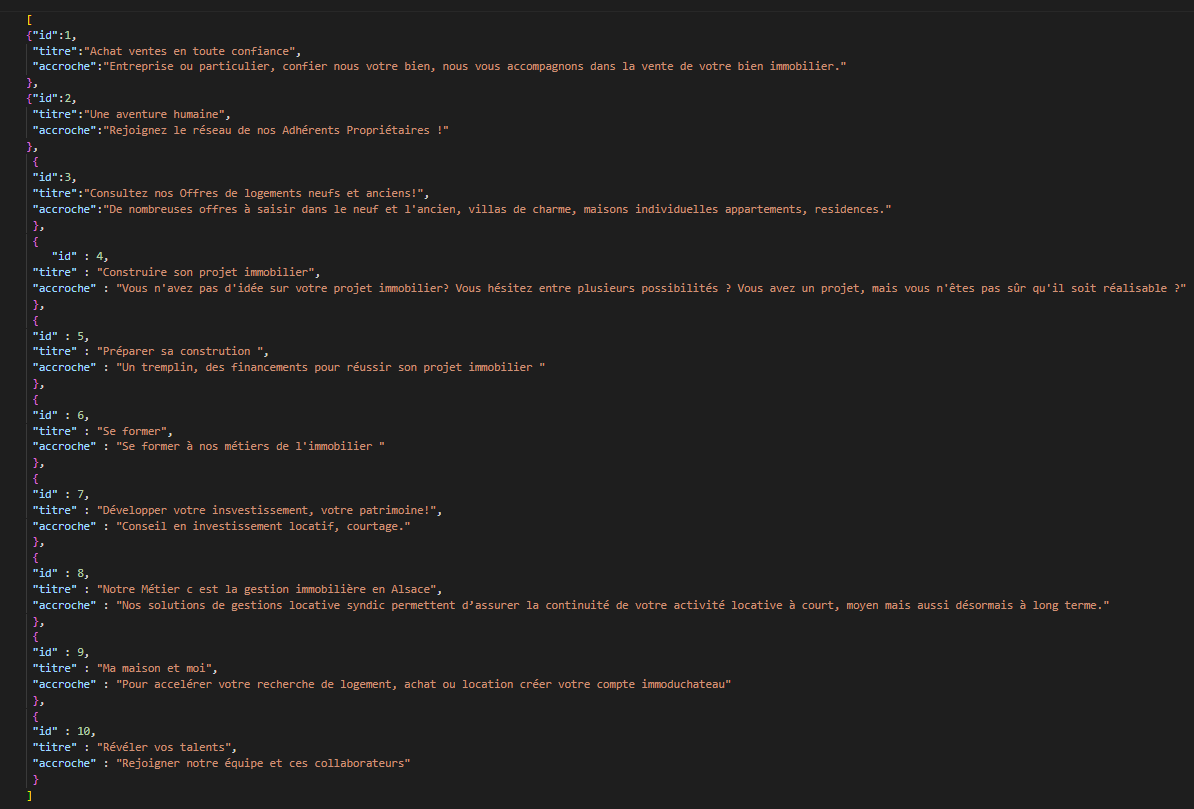
liste.remove();

**Le javascript avec la manipulation du json et chargement de données**

**Le json ajax et javascript**

.Pour les ordinateurs, ce format se génère et s’analyse facilement. Pour les humains, il est pratique à écrire et à lire grâce à une syntaxe simple et à une structure en arborescence. JSON permet de représenter des données structurées.

Mais concrètement voilà à quoi ça ressemble:



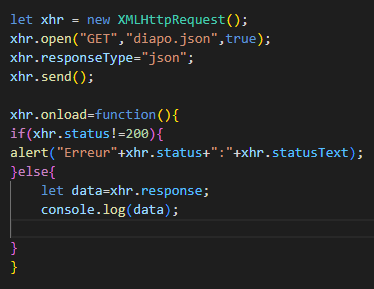
Capture d'écran d'un fichier tableau json contenue dans le fichier que j'ai nommé diapo.json.

Pour le charger il existe 2 méthodes :

**xhr**

Cette méthode et la plus ancienne.

Exemple de la méthode xhr permettant de charger notre fichier diapo.json:



XMLHttpRequest est conçu pour lire la source de données de l'URL de manière synchrone (synchronous) ou asynchrone (asynchronous). La lecture asynchrone des données permet aux utilisateurs d'être toujours en mesure de manipuler le navigateur pendant que XMLHttpRequest lit à distance la source de données.

Le constructeur ici et let xhr= new XMLHttpRequest();

Properties

|  |  |
| --- | --- |
| Propriété | Description |
| readyState | Renvoie un numéro, décrit l'état d'avancement de la requête (request). |
| timeout | Définie la durée timeout maximale accordée à une requête  pour quelle s'effectue complètement,   si elle n’est pas reçue dans ce délai,  elle est considérée comme un échec et se termine automatiquement. |
| withCredentials | Sa valeur est true ou false (valeur par défaut est false).  Si true signifie que cette requête peut utiliser des cookies,  des autorisations headers mais qu'elle doit toujours respecter la même politique  d'origine. |
| responseType | Définie la valeur que vous voulez recevoir. La valeur par défaut est "text"  . (Voir les valeurs possibles de cette propriété ci-dessous) |
| response | Renvoie un objet ArrayBuffer, Blob, Document, ou DOM String, selon la valeur de XMLHttpRequest.responseType qui contient la valeur de response. |
| responseText | Renvoie une DOM String qui contient le contenu de la réponse (response)  sous forme de texte ou null nếu yêu cầu (request)  si la demande a été rejetée ou n’a pas encore été envoyée. |
| responseURL | Renvoie une serialized URL de la réponse (response)  la chaîne vide si l’URL est nulle. |
| responseXML | Renvoie un objet Document contenant la réponse à la demande,  ou null si la demande a échoué, n’a pas encore été envoyé,  ou ne peut pas être analysé (parse) comme XML ou HTML. |
|  | |
| status | Renvoie un numéro indiquant l’état de la réponse à la demande. |
| statusText | Renvoie une DOM String contenant la chaîne de réponse retournée  par le serveur Http Server. Par exemple "200 OK". |

readyState

Les valeurs possibles de readyState :

|  |  |  |
| --- | --- | --- |
| État | Valeur | Description |
| UNSENT | 0 | XMLHttpRequest a été créé mais open() n'a pas encore été appelé. |
| OPENED | 1 | open() a été appelé. |
| HEADERS\_RECEIVED | 2 | send() a été appelé et headers & status sont disponibles. |
| LOADING | 3 | Téléchargement; responseText contient des données partielles. |
| DONE | 4 | L'opération est complète. |

responseType

Les valeurs possibles de responseType :

| **Valeur** | **Description** |
| --- | --- |
| "" | Si aucune valeur n’est spécifiée pour le responseType,  ou la valeur vide, elle est traitée comme "texte". |
| "arraybuffer" | La réponse (response) est un ArrayBuffer  contenant des données binaires (binary data). |
| "blob" | La réponse est un objet Blob contenant des données binaires. |
| "document" | La réponse est un HTML Document ou XMLDocument,  selon le type MIME (MIME type) des données reçues. |
| "json" | La réponse est un objet JavaScript  créé en analysant le contenu des données reçues en JSON. |
| "text" | La réponse est du texte dans un objet DOMString. |

Events

Lors de l'envoi d'une demande de lecture d'une source de données à partir d'une URL, XMLHttpRequest déclenche (fire) des événements décrits dans l'interface ProgressEvent.

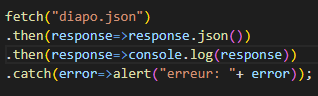
Remarque : Les événements marqués par (?) sont standards de test, qui ne sont pas pris en charge par la plupart des navigateurs.

|  |  |  |
| --- | --- | --- |
|  | **Événement** | **Description** |
| ? | loadstart | Indique que le processus de chargement (load) des données a commencé.  Cet événement se déclenche toujours en premier. |
|  | progress | L’événement se déclenche plusieurs fois lorsque les données sont chargées,  ce qui donne accès aux données intermédiaires. |
| ? | error | L’événement se déclenche lorsque le chargement est échoué. |
| ? | abort | L'événement se déclenche lorsque le chargement des données a été annulé  en appelant la méthode abort()  (méthode disponible sur XMLHttpRequest & FileReader). |
|  | load | L'événement ne se déclenche que lorsque toutes les données  ont été chargées avec succès. |
| ? | loadend | L'événement se déclenche lorsque l'objet a terminé  le transfert des données.   L'événement se déclenche toujours après l'événement error, abort, ou load. |

**Fetch**

La deuxième méthode la plus récente est fetch qui affichera sur le navigateur la même chose que la méthode xhr vue précédemment.

Etudions un peu sont code que voici:



La premières ligne fetch("diapo.json") permet de relier le fetch avec notre fichier json.

La seconde ligne .then(response=>response.json()) indique le format de la réponse ici du json.

.then(response=>console.log(response)) la troisième ligne affiche le json dans la console du navigateur comme avec le xhr vue au-dessus.

Et enfin la dernière ligne .catch(error=>alert("erreur: "+ error)); affiche une erreur si le code s'exécute mal.

Voilà ce petit cours est terminer voyons maintenant un cas concret avec un petit exo expliqué reprenant ce que nous avons appris dans ce cours avec un exercice. Ne vous inquiété pas je vous donnerais la solution expliqué.

Pour cet exercice, vous devez créer une application web affichant la liste des employés d’une société. Les informations proviennent d’une API fictive et sont rendues disponibles dans le fichier JSON employees.json.

Les employés doivent être listés dans un tableau, avec les informations suivantes :

ID (fourni par l’API)

Full name (fourni par l’API)

Email : (à calculer) Exemple pour John Doe => “j.doe@email.com”

Salaire mensuel : (à calculer)

Year of birth : (à calculer)

Une colonne d’actions contenant des boutons d’actions (voir la capture ci-dessous)

Le décompte des employés est affiché en-pied de la colonne ID.

Le total des salaires mensuels est affiché en-pied de la colonne Salaire mensuel.

Un bouton d’action permet de dupliquer un employé dans le tableau.

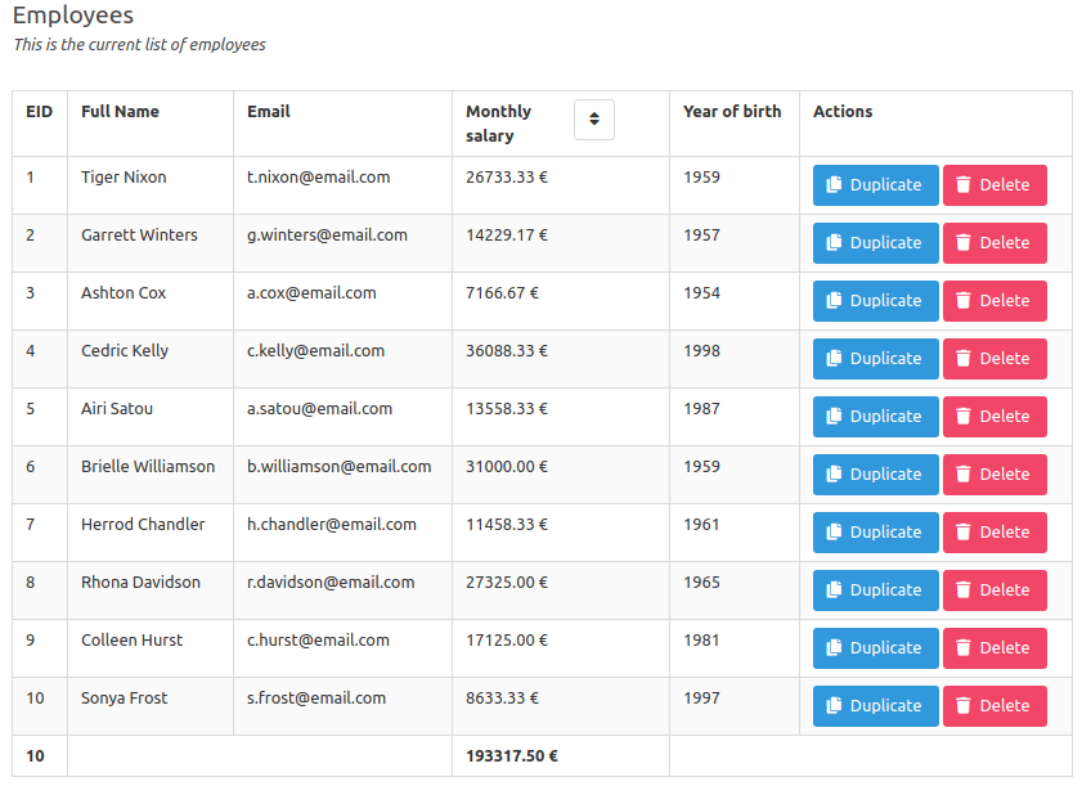
Note: L’ID de l’employé dupliqué doit être redéfini (les identifiants sont uniques)

Un bouton d’action permet de supprimer un employé du tableau.

Pour ces boutons d’actions, ne manipulez que les données du JSON chargé.

Un bouton de tri dans l’en-tête de colonne Monthly salary permet le tri croissant / décroissant des valeurs de cette colonne.

Si aucun employé n’est dans la liste, un message signale l’absence d’informations dans le tableau.

L’affichage « en anglais » devrait ressembler à la capture suivante : 

On vous donne ce json:

[

  {

    "employee\_id": 1,

    "employee\_manager": null,

    "employee\_lastname": "Holems",

    "employee\_firstname": "Cathy",

    "employee\_role": "Directrice",

    "employee\_salary": 100000,

    "employee\_hiredate": "2010-01-09"

  },

  {

    "employee\_id": 2,

    "employee\_manager": 1,

    "employee\_lastname": "Mannheim",

    "employee\_firstname": "Luc",

    "employee\_role": "DRH",

    "employee\_salary": 87500,

    "employee\_hiredate": "2017-06-17"

  },

  {

    "employee\_id": 3,

    "employee\_manager": 2,

    "employee\_lastname": "Kipré",

    "employee\_firstname": "Abdou",

    "employee\_role": "RH",

    "employee\_salary": 42800,

    "employee\_hiredate": "2017-10-09"

  },

  {

    "employee\_id": 4,

    "employee\_manager": 2,

    "employee\_lastname": "Martin",

    "employee\_firstname": "Valérie",

    "employee\_role": "Comptable",

    "employee\_salary": 39500,

    "employee\_hiredate": "2018-03-30"

  },

  {

    "employee\_id": 5,

    "employee\_manager": 1,

    "employee\_lastname": "Slezak",

    "employee\_firstname": "Daniel",

    "employee\_role": "DSI",

    "employee\_salary": 75000,

    "employee\_hiredate": "2011-09-09"

  },

  {

    "employee\_id": 6,

    "employee\_manager": 5,

    "employee\_lastname": "Bahl",

    "employee\_firstname": "Tarik",

    "employee\_role": "Administrateur Système",

    "employee\_salary": 60000,

    "employee\_hiredate": "2014-04-08"

  },

  {

    "employee\_id": 7,

    "employee\_manager": 5,

    "employee\_lastname": "Armanetti",

    "employee\_firstname": "Michaël",

    "employee\_role": "Administrateur Réseaux",

    "employee\_salary": 60000,

    "employee\_hiredate": "2014-05-06"

  },

  {

    "employee\_id": 8,

    "employee\_manager": 5,

    "employee\_lastname": "Goldman",

    "employee\_firstname": "Estelle",

    "employee\_role": "Cheffe de projet",

    "employee\_salary": 55000,

    "employee\_hiredate": "2016-04-20"

  },

  {

    "employee\_id": 9,

    "employee\_manager": 5,

    "employee\_lastname": "Durand",

    "employee\_firstname": "Gabriel",

    "employee\_role": "Chef de projet",

    "employee\_salary": 55000,

    "employee\_hiredate": "2016-12-02"

  },

  {

    "employee\_id": 10,

    "employee\_manager": 8,

    "employee\_lastname": "Morel",

    "employee\_firstname": "Audrey",

    "employee\_role": "Développeuse",

    "employee\_salary": 46500,

    "employee\_hiredate": "2012-07-01"

  },

  {

    "employee\_id": 11,

    "employee\_manager": 8,

    "employee\_lastname": "Carpentier",

    "employee\_firstname": "Guillaume",

    "employee\_role": "Développeur Sénior",

    "employee\_salary": 58500,

    "employee\_hiredate": "2010-02-03"

  },

  {

    "employee\_id": 12,

    "employee\_manager": 9,

    "employee\_lastname": "Lefebvre",

    "employee\_firstname": "Hugo",

    "employee\_role": "Développeur",

    "employee\_salary": 42000,

    "employee\_hiredate": "2015-10-11"

  },

  {

    "employee\_id": 13,

    "employee\_manager": 9,

    "employee\_lastname": "Sharif",

    "employee\_firstname": "Sonia",

    "employee\_role": "Développeuse Sénior",

    "employee\_salary": 54500,

    "employee\_hiredate": "2011-01-23"

  },

  {

    "employee\_id": 14,

    "employee\_manager": 15,

    "employee\_lastname": "Fournier",

    "employee\_firstname": "Sabrina",

    "employee\_role": "Commerciale",

    "employee\_salary": 42000,

    "employee\_hiredate": "2017-10-27"

  },

  {

    "employee\_id": 15,

    "employee\_manager": 5,

    "employee\_lastname": "Bower",

    "employee\_firstname": "Sarah",

    "employee\_role": "Responsable Produit",

    "employee\_salary": 49500,

    "employee\_hiredate": "2018-05-22"

  },

  {

    "employee\_id": 16,

    "employee\_manager": 8,

    "employee\_lastname": "Dimario",

    "employee\_firstname": "Jordan",

    "employee\_role": "Développeur Junior",

    "employee\_salary": 32500,

    "employee\_hiredate": "2019-06-30"

  },

  {

    "employee\_id": 17,

    "employee\_manager": 9,

    "employee\_lastname": "Ribault",

    "employee\_firstname": "Oceane",

    "employee\_role": "Développeuse Junior",

    "employee\_salary": 25000,

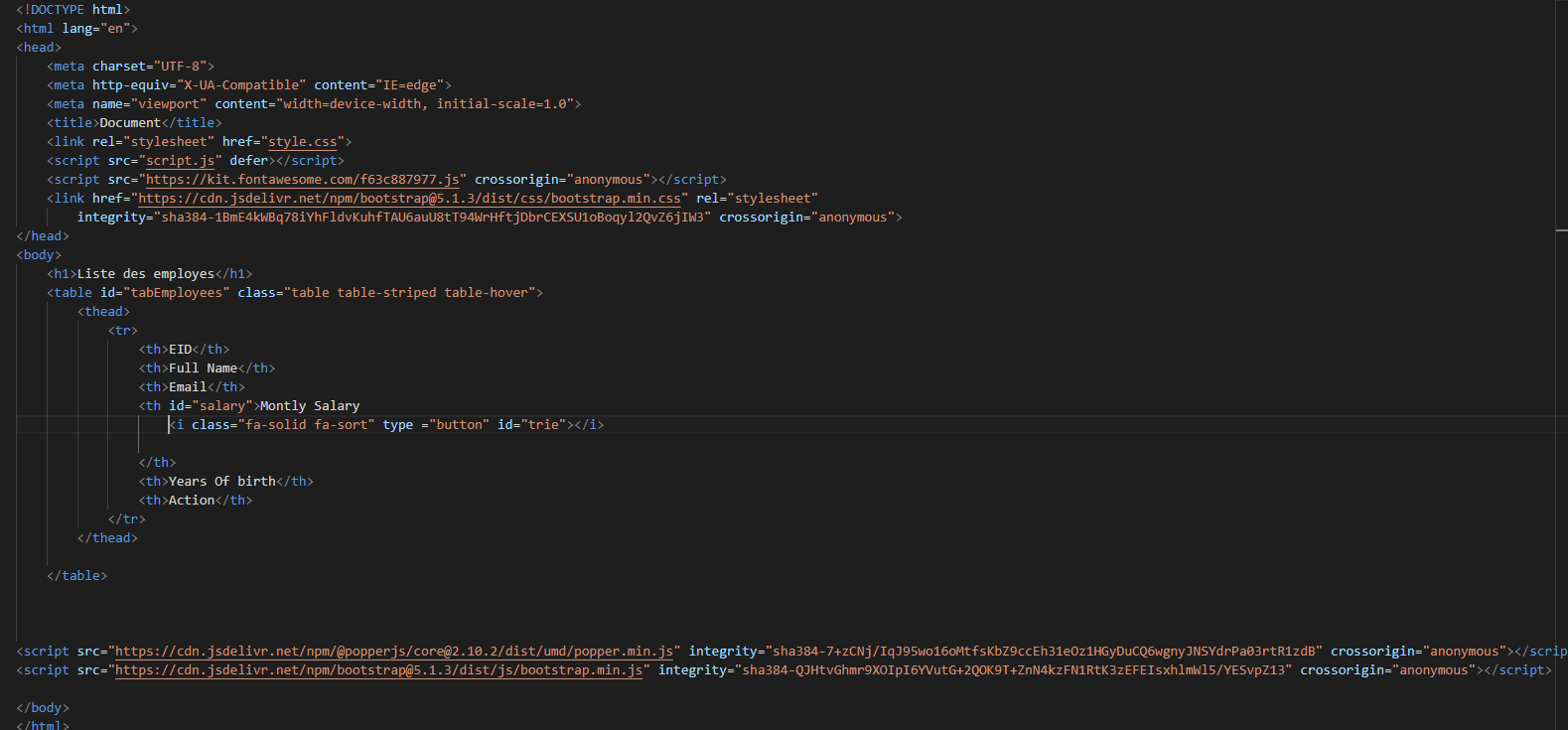
    "employee\_hiredate": "2019-11-04"

  }

]

**Solution et explications:**

Pour cet exercice il faut en premier réaliser la page HTML suivante:



Dans un premier temps il nous faut écrire le doctype de la page html avec le raccourci clavier ! Sur vscode à l'intérieur j'ai relié un link pour qu'il prenne en compte mon futur fichier CSS puis un script pour le relier au js

     <link rel="stylesheet" href="style.css">

    <script src="script.js" defer></script>

Les balises suivantes sont:

-la premières ligne pour chercher un logo sort qui et placé dans le tableau⇳ (juste à côté de Montly Salarydans le tableau)

Les autres lignes relie le html avec du bootstrap

    <script src="https://kit.fontawesome.com/f63c887977.js" crossorigin="anonymous"></script>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet"

        integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">

Ensuite nous avons rempli le corps de la page html:

<body>

    <h1>Liste des employes</h1>

    <table id="tabEmployees" class="table table-striped table-hover">

        <thead>

            <tr>

                <th>EID</th>

                <th>Full Name</th>

                <th>Email</th>

                <th id="salary">Montly Salary

                    <i class="fa-solid fa-sort" type ="button" id="trie"></i>

                </th>

                <th>Years Of birth</th>

                <th>Action</th>

            </tr>

        </thead>

    </table>

<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.10.2/dist/umd/popper.min.js" integrity="sha384-7+zCNj/IqJ95wo16oMtfsKbZ9ccEh31eOz1HGyDuCQ6wgnyJNSYdrPa03rtR1zdB" crossorigin="anonymous"></script>

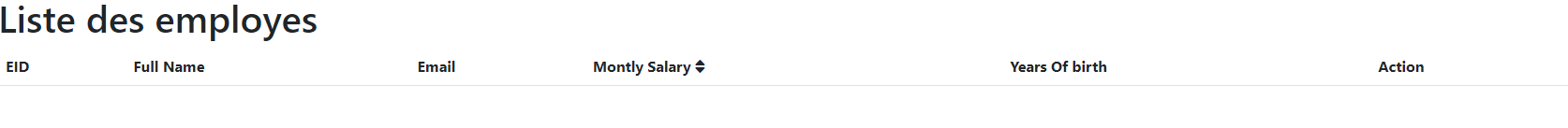
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.min.js" integrity="sha384-QJHtvGhmr9XOIpI6YVutG+2QOK9T+ZnN4kzFN1RtK3zEFEIsxhlmWl5/YESvpZ13" crossorigin="anonymous"></script>

</body>

En html nous avons simplement fait le h1 qui est le titre de la page puis la balise <table> qui contient les éléments du tableau ici le thead ou on a placé les titre des colonnes du tableau.

Et a la fin nous plaçons des balise script du bootstap qui permet d'agencé le tableau.

Nous obtenons en html ce resultat:



Pour rendre le tableau plus attrayant nous y ajoutons des balises CSS:

h1{

    text-align: center;

    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;

}

#tabEmployees{

    width: 70%;

    margin-left: auto;

    margin-right: auto;

    margin-top: 50px;

}

.btn-primary{

    margin-right: 10px;

Comme vous étiez gentils jusque-là je vais vous l'expliquer:

h1{

    text-align: center; /\* centre le texte \*/

    font-family:Impact,Haettenschweiler,'Arial Narrow Bold',sans-serif;

    /\*ajoute la police d'ecriture\*/

}

#tabEmployees{/\* selectionne le tableau\*/

    width: 70%;/\* ajoute une taille maximum au tableau\*/

    margin-left: auto;/\* margin a gauche  \*/

    margin-right: auto;/\* marging a droite\*/

    margin-top: 50px;/\* margint en haut\*/

}

.btn-primary{/\*selectionne les bouton \*/

    margin-right: 10px;/\* marging a gauche\*/

}

Et pour finir on s'occupe du code javascript (je sais c'est la partie la plus complexe mais je vais vous l'expliquer Vous verrez ce n'est pas bien méchant).

Je vais vous montrer le code final (ne vous enfuyez pas c'est moins compliqué que sa en a l'air)

function creerCellTitre(\_contenu, \_element) {

    let maCell = document.createElement("th");

    \_element.appendChild(maCell);

}

let mesDonnees = [];

let init = false;

function fillTable(\_donnees) {

    mesDonnees = \_donnees;

    let montab = document.querySelector("#tabEmployees");

    //Création de la partie "Mon Body               ;"

    let monBody = montab.createTBody();

    monBody.setAttribute("id", "corps");

    for (let i = 0; i < \_donnees.length; i++) {

        let monObjet = \_donnees[i];

        let maLigne = monBody.insertRow();

        maLigne.setAttribute("id", "maLigne" + monObjet.employee\_id);

        for (let i = 0; i <= 5; i++) {

            switch (i) {

                case 0:

                    let cell1 = maLigne.insertCell();

                    cell1.innerText = monObjet.employee\_id;

                    break;

                case 1: let cell2 = maLigne.insertCell();

                    cell2.innerText = monObjet.employee\_firstname + " " + monObjet.employee\_lastname;

                    break;

                case 2:

                    let cell3 = maLigne.insertCell();

                    cell3.innerText = (monObjet.employee\_firstname.substring(0, 1)).toLowerCase() + "." + (monObjet.employee\_lastname).toLowerCase() + "@email.com";

                    break;

                case 3: let cell4 = maLigne.insertCell();

                    cell4.innerText = ((monObjet.employee\_salary / 12).toFixed(2));

                    break;

                case 4:

                    let cell5 = maLigne.insertCell();

                    cell5.innerText = monObjet.employee\_hiredate.substring(0, 4);

                    break;

                case 5:

                    let cell6 = maLigne.insertCell();

                    let monBtnd = document.createElement("input");

                    monBtnd.setAttribute("type", "button");

                    monBtnd.setAttribute("class", "btn btn-primary");

                    monBtnd.setAttribute("id", "btnd" + monObjet.employee\_id);

                    monBtnd.value = "Duplicate";

                    cell6.appendChild(monBtnd);

                    let monbtnde = document.createElement("input");

                    monbtnde.setAttribute("type", "button");

                    monbtnde.setAttribute("id", "btnde" + monObjet.employee\_id);

                    monbtnde.setAttribute("class", "btn btn-danger");

                    monbtnde.value = "Delete";

                    cell6.appendChild(monbtnde);

                    break;

                default:

                    break;

            }

        }

        document.getElementById("btnd" + monObjet.employee\_id).addEventListener("click", function () {

            let row = document.getElementById("maLigne" + monObjet.employee\_id);

            console.log(row)

            let clone = row.cloneNode(true);

            clone.setAttribute("id", "clone" + monObjet.employee\_id);

            monBody.appendChild(clone);

        })

        document.getElementById("btnde" + monObjet.employee\_id).addEventListener("click", function () {

            let remove1 = document.getElementById("maLigne" + monObjet.employee\_id);

            remove1.remove();

        })

    }

    let footer = montab.createTFoot();

    footer.setAttribute("id", "pied");

    let derligne = footer.insertRow();

    for (let i = 0; i <= 5; i++) {

        if (i == 0) {

            let total = \_donnees.length;

            let maCell = derligne.insertCell();

            maCell.innerText = total;

        } else if (i == 3) {

            let somme = 0;

            for (let j = 0; j < \_donnees.length; j++) {

                somme += \_donnees[j].employee\_salary;

            }

            let cellsomme = derligne.insertCell();

            cellsomme.innerText = somme + " €";

        } else {

            let cellvide = derligne.insertCell();

            cellvide.innerText = " ";

        }

    }

}

function trierData(\_mesDonnees) {

    let sorties = \_mesDonnees.sort(function (a, b) {

        return a.employee\_salary - b.employee\_salary;

    })

    return sorties;

}

function trierDatadecroissant(\_mesDonnees) {

    let sorties = mesDonnees.sort(function (b, a) {

        return a.employee\_salary - b.employee\_salary;

    })

    return sorties;

}

let xhr = new XMLHttpRequest();

xhr.open("GET", "test.json", true);

xhr.responseType = "json";

xhr.send();

xhr.onload = function () {

    if (xhr.status != 200) {

        alert("Erreur" + xhr.status + ":" + xhr.statusText);

    } else {

        let data = xhr.response;

        // console.log(data);

        fillTable(data);

    }

}

let btntrie = document.getElementById("trie");

btntrie.addEventListener("click", function () {

    let reservoir = trierData(mesDonnees);

    let tabBody = document.querySelector("#corps");

    tabBody.remove();

    let tabPied = document.querySelector("#pied");

    tabPied.remove();

    fillTable(reservoir);

});

btntrie.addEventListener("dblclick", function () {

    let reservoir = trierDatadecroissant(mesDonnees);

    let tabBody = document.querySelector("#corps");

    tabBody.remove();

    let tabPied = document.querySelector("#pied");

    tabPied.remove();

    fillTable(reservoir);

});

Vous êtes toujours là?

Je vous explique mon code.

// une fonction pour créer des cellule qui contient des parametre s

function creerCellTitre(\_contenu, \_element) {

    // on instancie une nouvelle variable

    //dans cette variable on créer l'element th

    let maCell = document.createElement("th");

    // on indique que cellule et l'enfant de \_elemlent

    \_element.appendChild(maCell);

}

//on créer une variable mesDonnées

//on a indiquer que cetais un tableau avec les []

let mesDonnees = [];

// on indique que les donnée son false(booleen)

let init = false;

// nouvelle fonction avec \_donnees en parametre

function fillTable(\_donnees){

// on indique que le parametre \_donnees =

// la variable mesDonnees créer au dessu qui du coup devien une donees tableau

    mesDonnees = \_donnees;

    // on accroche montab a l'id #tabemployées present dans le fichier html

    let montab = document.querySelector("#tabEmployees");

    //Création de la partie "Mon Body  avec un tbody

    let monBody = montab.createTBody();

//on y ajoute un id Corp au tbody

    monBody.setAttribute("id", "corps");

    //on parcour le tableau et avec le for a chaque fois

    //que des donnée json sont trouvé il créer une nouvelle ligne

    for (let i = 0; i < \_donnees.length; i++) {

        let monObjet = \_donnees[i];

        //grace a la boucle for on créer une ligne inserrow()

        let maLigne = monBody.insertRow();

        //sur cette ligne on insert un id et on ajoute le numero id present dans le json

        maLigne.setAttribute("id", "maLigne" + monObjet.employee\_id);

        //2eme boucle for pour creer les cellule du tableau

        for (let i = 0; i <= 5; i++) {

            //avec le swuitch case on remplace les donner qui ne sont pas dans l'ordre

            //a chaque fin d'instruction de la case on y mets un break pour la finir

            switch (i) {

                case 0:

                    let cell1 = maLigne.insertCell();

                    cell1.innerText = monObjet.employee\_id;

                    break;

                case 1: let cell2 = maLigne.insertCell();

                    cell2.innerText = monObjet.employee\_firstname + " " + monObjet.employee\_lastname;

                    break;

                case 2:

                    let cell3 = maLigne.insertCell();

                    cell3.innerText = (monObjet.employee\_firstname.substring(0, 1)).toLowerCase() + "." + (monObjet.employee\_lastname).toLowerCase() + "@email.com";

                    break;

                case 3: let cell4 = maLigne.insertCell();

                    cell4.innerText = ((monObjet.employee\_salary / 12).toFixed(2));

                    break;

                case 4:

                    let cell5 = maLigne.insertCell();

                    cell5.innerText = monObjet.employee\_hiredate.substring(0, 4);

                    break;

                case 5:

                    // creation de la cellule 5 un tableau commence toujours a zero

                    let cell6 = maLigne.insertCell();

                    // creation de l'element input

                    let monBtnd = document.createElement("input");

                    //ajoute du type bouton a l'input

                    monBtnd.setAttribute("type", "button");

                    //ajout de l'attribut classe btn-primary

                    monBtnd.setAttribute("class", "btn btn-primary");

                    //ajout de l'id

                    monBtnd.setAttribute("id", "btnd" + monObjet.employee\_id);

                    //ajout de la value du bouton

                    monBtnd.value = "Duplicate";

                    // la variable monBtn devient l'enfant de la variable cell6

                    cell6.appendChild(monBtnd);

                    let monbtnde = document.createElement("input");

                    monbtnde.setAttribute("type", "button");

                    monbtnde.setAttribute("id", "btnde" + monObjet.employee\_id);

                    monbtnde.setAttribute("class", "btn btn-danger");

                    monbtnde.value = "Delete";

                    cell6.appendChild(monbtnde);

                    break;

                default:

                    break;

            }

        }

        // ici on reprent le bouton par l'id et on rajoute le numero de l'id present en json

        //puis aon a fait yn event qui execute la fontion au click sur la page

        document.getElementById("btnd" + monObjet.employee\_id).addEventListener("click", function () {

            //ce bouton créer un nouvelle id au bouton créer

            let row = document.getElementById("maLigne" + monObjet.employee\_id);

            console.log(row)

            // nousq avons créer un clone de la ligne au click sur le bouton

            let clone = row.cloneNode(true);

            // le clone a un nouveau id

            clone.setAttribute("id", "clone" + monObjet.employee\_id);

            // le clone deviens l'enfant de monBody

            monBody.appendChild(clone);

        })

        //même procedé pour la fonction remove (supprimer)

        document.getElementById("btnde" + monObjet.employee\_id).addEventListener("click", function () {

            let remove1 = document.getElementById("maLigne" + monObjet.employee\_id);

            // a la place de créer une ligne on la suprime avec un .remove()

            remove1.remove();

        })

    }

    //créatopn du footer du tableau

    let footer = montab.createTFoot();

    // ajout de l'id au footer

    footer.setAttribute("id", "pied");

    //ajout de la ligne footer avec inserRow()

    let derligne = footer.insertRow();

    //realisation d'un for qui vas caculé la somme des ligne ici 17  ligne

    for (let i = 0; i <= 5; i++) {

        if (i == 0) {

            let total = \_donnees.length;

            let maCell = derligne.insertCell();

            maCell.innerText = total;

        } else if (i == 3) {

            let somme = 0;

            // creation d'un 2eme for qui calcule la sommes des salaires

            for (let j = 0; j < \_donnees.length; j++) {

                somme += \_donnees[j].employee\_salary;

            }

            let cellsomme = derligne.insertCell();

            cellsomme.innerText = somme + " €";

        } else {

            let cellvide = derligne.insertCell();

            cellvide.innerText = " ";

        }

    }

}

//fonction pour trier les données.

function trierData(\_mesDonnees) {

    let sorties = \_mesDonnees.sort(function (a, b) {

        return a.employee\_salary - b.employee\_salary;

    })

    return sorties;

}

function trierDatadecroissant(\_mesDonnees) {

    let sorties = mesDonnees.sort(function (b, a) {

        return a.employee\_salary - b.employee\_salary;

    })

    return sorties;

}

//xhr qui charge les données du json

let xhr = new XMLHttpRequest();

xhr.open("GET", "test.json", true);

xhr.responseType = "json";

xhr.send();

xhr.onload = function () {

    if (xhr.status != 200) {

        alert("Erreur" + xhr.status + ":" + xhr.statusText);

    } else {

        let data = xhr.response;

        // console.log(data);

        fillTable(data);

    }

}

//fonction qui trie en ordre croissant avec un click

let btntrie = document.getElementById("trie");

btntrie.addEventListener("click", function () {

    let reservoir = trierData(mesDonnees);

    let tabBody = document.querySelector("#corps");

    tabBody.remove();

    let tabPied = document.querySelector("#pied");

    tabPied.remove();

    fillTable(reservoir);

});

//Fonction qui trie en ordre décroissant au double click

btntrie.addEventListener("dblclick", function () {

    let reservoir = trierDatadecroissant(mesDonnees);

    let tabBody = document.querySelector("#corps");

    tabBody.remove();

    let tabPied = document.querySelector("#pied");

    tabPied.remove();

    fillTable(reservoir);

});

**Fin du document**