**Cours 3wa Academy 2018**

* **Intégration**

**HTML**

***Débuter sur HTML***

Éditeur de texte : choix du prof > Atom

Accéder au serveur local (sur pc 3w Academy) > Google chrome > url > taper « localhost »

Enregistrer fichier sur Atom > Ctrl + S

Rafraichir une page web > F5

ou pour rafraichir avec le cache > Ctrl + F5

Revenir en arrière > ctrl Z

body{

background-color: tomato;background-repeat: no-repeat;

color: steelblue;

}background: linear-gradient(150deg, white 50%, #333 50%)

Faire un fichier index.html > quand on débute un projet pour que ça s’affiche directement (page d’acceuil)

**Pour commencer un document html :**

<!doctype html> ou <!DOCTYPE html>

<html lang="fr">

Marc4 h 07  background-repeat: no-repeat;

<head>http://legal-hiring.com/

body{

background-color: tomato;

color: steelblue;

}

<title> </title>

<meta charset="utf-8">

</head>

<main>

<body>

*du texte : Et ma policbox-sizing: border-box;e sera Arial*

</main>

<footer>

</footer>

</body>

</html>

***Vérifier le charset***

**Infos sur le charset :**

<meta(c’est le tag name) charset (c’est l’attribut)="utf-8"(c’est la valeur de l’attribut)>

***Vérifier son code***

Pour valider un site on va sur **W3C validator : étape obligatoire !**

***Les commentaires***

**< !-- ici le commentaire header – > c’est un commentaire**

raccourci clavier pour faire un **commentaire (sur un éditeur de texte) ctrl + maj + /**

***Les différentes balises HTML...***

Pour les titres ça va de H1 à H6 .

Pour les blocs sémantiques, depuis HTML 5, <section> a tendance à remplacer <div>. On peut mettre une section dans une autre section.

Pour le contenu d’une section, on peut (recommandé) utiliser la balise <article>. Dans article, on peut utiliser <p> pour les paragraphes.

Pour les images, on utilise <img(tag name) src(attribut)= « »(valeur de l’attribut)>

Pour les images, il faut obligatoirement un attribut alt

**commentaire**

Pour désindenter > shift Tab

Deux types de listes, ul (liste non ordonnée) et ol (liste ordonnée).

>Ol pour Orderedlist

>UL pour Underedlist

Pour faire un trait, la balise <hr> (littéralement horizontal rule).

**CSS**

***Mise en place du CSS…***

Pour faire du css on peut l’intégrer dans le corps du code HTML (ex: style="color:red") mais c’est pas recommandé car moins facilement modifiable. Le mieux c’est de faire du CSS dans un document CSS à part.

On met la balise link <link href=«css/hello.css» rel=«stylesheet»>.

C’est à dire, feuille de style .CSS spécifique.

***Les bases…***

Pour le contenu du CSS :

exemple :

body(c’est un sélecteur){

background-color:(c’est une propriété) tomato(c’est une valeur);

color(c’est une propriété):(espace) steelblue(c’est une valeur);

}

autre exemple :

h1{

color: olive;

text-align: center;

font-size: 60px;display

}

***Manipuler le texte…***

text-transform: uppercase; > majuscule **commentaire**

text-decoration: none; (enlève les soulignements)

***Les nav…***

nav a{

color: oliclear:both;

overflow: hidden;ve;

font-size: 24px; **commentaire**

} (pour modifier les a dans la nav)

h1, h3{

text-align: center;

} (deux sélecteurs en même temps)

Pour modifier certains trucs précis dans html on fait une class. Exemple :

<h4 class="important"> Le Html : </h4>

<h4 class="important bigtext"> Le CSS : </h4> (pour mettre deux class à la suite (important et bigtext) il faut mettre un espace entre les class entre les mêmes « »...

Pour centrer un élément, au lieu de faire appel à des px vaut mieux utiliser des % (responsive)

body{

width: 80%;

margin: auto;

}

Prof d’intégration (Erwin Calvez) > [erwincalvez@gmail.com](mailto:erwincalvez@gmail.com) > 06 12 40 26 23

Pour les couleurs différentes techniques :

- en hexadecimal *Exemple : background-color: #00009a;*

- en rgb. *Exemple : background-color: rgb(55, 123, 200);*(ici les valeurs varient entre 0 et 255)

- en rgba. *Exemple : background-color: rgba(55, 123, 200, 1);* (ici la dernière valeur (qui change l’opacité) varie entre 0 et 1)

Pour rallonger un paragraphe dans Atom, on met un « faux texte », on appelle ça le « Lorem ». Pour le faire, on va au début d’un paragraphe et on tape « Lorem ».

ackground-color: red

Interlignage en CSS : il faut utiliser ligne-height. Pour s’assurer que la ligne-height soit adaptée à la font-size de l’élément, on peut sur body faire une ligne-height avec une unité de mesure 1.5 (exemple) plutôt qu’une unité avec des pixels.

Pour sortir d’un dossier dans lequel on est (en tapant une url) on tape .. avant le /.

L'élément HTML <iframe> représente un contenu de navigation imbriqué qui permet en fait d'obtenir une page HTML intégrée dans la page courante. Le contexte de navigation qui contient le contenu intégré est appelé « contexte de navigation parent ». Le contexte de navigation le plus élevé (qui n'a pas de contexte parent) correspond généralement à la fenêtre du navigateur (cf. Window).

Pour éviter la répétition du background > background-repeat: no-repeat;

Pour la répétition sur l’axe vertical :background > background-repeat: repeat-y;

Pour la répétition sur l’axe horizontal  :background > background-repeat: repeat-x;

Pour choisir une police différente de celles pré-installées par HTML/CSS, on peut aller sur la biliothèque google font. Pour ce faire, on fait d’abord un link dans le head (ex : <link href="https://fonts.googleapis.com/css?family=Oswald" rel="stylesheet") puis on ajoute la police choisie dans le CSS (ex : font-family: 'Oswald', sans-serif;).

Display, il y a deux types principaux : les blocs (L’élément p,Les éléments h1, h2, Les éléments ol et ul ; l’élément form , l’élément div, … ) et les inline ( de la déco exemples : les éléments strong et em ; l’élément a ; l’élément img ; l’élément span, ….).

-Les éléments de type blocs vont prendre par défaut toute la largeur de leur élément. Il s’étale dans toute la largeur disponible. Un élément block prend donc toute la largeur. Si le block prend toute la place de l’élément, il y aura une marge automatique qui se mettra.

-Les éléments inline vont de gauche à droite. Quand il y a plus de place, ça revient à la ligne.

Les propriétés width, margin-top, margin-bottom ne marchent que sur les blocs et pas sur les inlines. Par contre, mar.fab{

margin-right: 10px;

}gin-right et margin-left marchent sur les deux.

display

La propriété height quasi interdite c’est le contenu qui donne la hauteur d’un élément. Utliser height comporte le risque d’un dépassement du texte de l’élément en cause.

clear:both;

overflow: hidden;

display:inline-block > permet de combiner les avantages de block et de inline.

On va mettre des éléments inline tout en pouvant maitrîser leurs width ce qui n’est normalement pas possible pour les éléments blocs.

clear:both;

overflow: hidden;

Vertical-align permet d’aligner les éléments.

Interligne > line-Height: 2;

Pour éviter une anamorphose. On combine width avec height : auto

header img{

width:175px;

height: auto;

text-align: center;

.fab{

margin-right: 10px;

}}

overflow: hidden; > pour faire en sorte que ce qui dépasse soit caché…

Pour le responsive, favoriser le % plutôt que le Px.

const FIRST\_CONSTANT = "Ceci est une constante";

document.write(FIRST\_CONSTANT);

Max-width > pour définir la valeur maximale d’un élément. Peut être placé après width: x%.

max-width: 100%; > pour faire en sorte que l’image ne soit pas plus grosse que le parent et on met ubn height:auto avec pour être sûr de pas les déformer.

Le padding est la marge à l’intérieur d’un bloc.

Pour être sur que les éléments en trop ne dépassent pas > clear:both; (qui permet de supprimer les effets du float) suivi de overflow: hidden;(qui permet de supprimer les éléments qui dépassent)

Pour mettre deux éléments côte à côte, on fait width : 48 % avec float:left ;

box-sizing: border-box > box-sizing: border-box. Ici l'élément fils s'inscrit correctement au sein de l'élément parent car width: 100% prend en compte l'épaisseur de la bordure et du remplissage. Ca permet donc de gérer les effets néfastes du padding…

Pour mettre en gras> font-weight: bold;

Pour mettre en majuscule> text-transform: uppercase;

Pour faire en sorte que les lettres se rapprochent ou s’espacent > letter-spacing:

Pour les mots > word-spacing

Pour faire un aliéna > on fait sur, par exemple, article p {text-indent: (averc une valeur qui peut être négative)}

Pour justifier un paragraphe > text-align: justify;

Pour les icones > [https://fontawesome.com](https://fontawesome.com/) > on copie le link puis on intègre les icones > on obtient des images en vectoriels. Dans le code, on peut modifier les caractéristiques de ces icones via >

.fab{ margin-right: 10px; }

aside c’est une balise de section qui veut dire contenu supplementaire

hover > pour modifier quand on survole

Transition > Les transitions CSS permettent de contrôler la vitesse d'animation lorsque les propriétés CSS sont modifiées. Plutôt que le changement soit immédiat, on peut l'étaler sur une certaine période. Exemple : transition : all 300ms ;

***Le FLOAT***

Pour mettre deux choses côte à côte il y a deux techniques, le float et le inline block

Le float il y a des limites c’est plutôt quand on veut occuper toute la largeur.

Pour un fond blanc transparent > background-color: rgba(255,255,255,0.2); (le 0,2 correspond au degré d’opacité)

Pour que l’image prenne tout l’espace > background-size: cover;

background-position: center bottom; > pour centrer l’image et voir le bas

padding: 15% 30%; > pour adapter la taille du padding en fonction de la taille de l’écran.

body{

clear:both;

overflow: hidden; background-color: tomato;

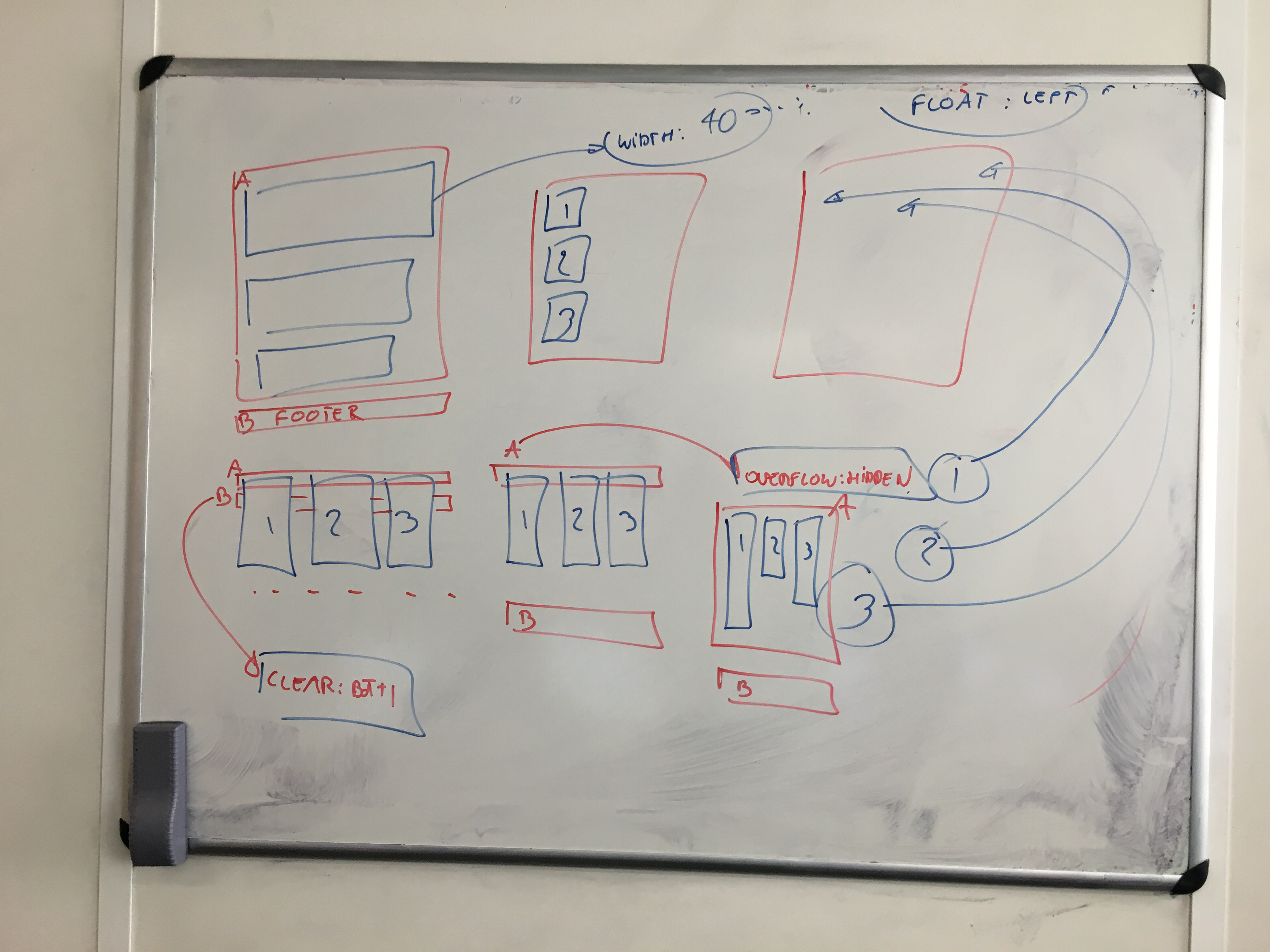
color: steelblue;

}

Pour être sur que deux éléments ne se colle pas (après utilisation d’un float) >

clear:both;

overflow: hidden;



***La spécificité en CSS***

# La spécificité en CSS

En dehors des Floats, la spécificité en CSS est l'un des concepts les plus difficiles à comprendre au sujet des feuilles de style en cascade (CSS). Les différents poids des sélecteurs (ex : h1, p, .bonjour, #bonjour,...) est généralement la raison pour laquelle vos règles CSS ne s'appliquent pas à certains éléments.

Afin de minimiser le temps de recherche des bugs, vous devez comprendre comment les navigateurs interprètent votre code CSS. Et pour comprendre cela, vous devez avoir une bonne compréhension de la façon dont fonctionne la spécificité. Dans la plupart des cas, les problèmes sont causés par le simple fait que quelque part entre vos règles CSS, vous avez défini un sélecteur plus spécifique qu'un autre.

La spécificité en CSS n'est pas simple. Cependant, il existe des méthodes pour l'expliquer de manière simple et intuitive. Et c'est justement le sujet de cet article.

Jetons un œil sur quelques problèmes importants liés à la spécificité en CSS ainsi que des exemples, des règles, des principes, des solutions et des ressources.

## Qu'est-ce que la spécificité ?

* La spécificité détermine quelles règles CSS sont appliquées par les navigateurs. “La spécificité est un type de pondération qui a une incidence sur la façon dont les règles de votre feuille de style en cascade (CSS) sont affichées.”
* Un sélecteur de spécificité est un processus utilisé pour déterminer quelles règles sont prioritaires en CSS lorsque plusieurs règles peuvent être appliquées à un même élément dans le balisage
* Chaque sélecteur a sa propre spécificité. "Toutes les règles CSS engendrent une notation de la spécificité indépendamment du type de sélecteur, bien que la pondération qui est donnée à chaque type de sélecteur varie et finira par affecter le style de vos documents Web."
* Si deux sélecteurs s'appliquent au même élément, celui qui a la plus grande spécificité l'emporte.

## Hiérarchie des spécificités

* Chaque sélecteur a sa place dans la hiérarchie de la spécificité. Il existe quatre catégories distinctes qui définissent le niveau de spécificité d'un sélecteur donné :

1. Les styles Inline (Présence du style dans le document et non dans la feuille CSS).  
Un style inline est situé à l'intérieur de votre document XHTML. Il est rattaché directement à l'élément à styliser. Par exemple <h1 style="color: #fff;">

2. Les IDs  
Un ID est un identifiant pour les éléments de vos pages, tels que #div.

3. Les classes, attributs et pseudo-classes.  
Ce groupe comprend les .classes, [attributes] et les pseudo-classes telles que :hover, :focus etc.

4. Les éléments et pseudo-éléments.  
Par exemple : :before et :after.

***Pour viser un élément particulier en CSS***

Pour viser uniquement un élément (le 2ème) parmi d’autres du même nom on peut écrire >

p:nth-of-type(2) {

text-align: left;

}

/\* Paragraphes impairs \*/

p:nth-of-type(2n+1) {

text-align: left;

}

/\* Paragraphes pairs \*/

p:nth-of-type(2n) {

text-align: right;

}

/\* Premier paragraphe \*/

p:nth-of-type(1){

font-weight: bold;

}

Pour viser le dernier élément d’un type =

.item-news:last-of-type {

border-bottom: 0px;

}

***Aligner des éléments par le haut***

Aligner des éléments par le haut (en display : inline-block) > vertical-align: top;

Pour enlever l’espace entre deux inline block > l’une des techniques est de faire font-size:initial ; (sur l’enfant) et du font-size:0 ; (sur le parent).

<td colspan="2"> 5</td>

background-attachment: fixed; > pour faire en sorte que l’image soit mobile en fonction du dépalcement de la souris <td colspan="2"> 5</td>

En CSS, ::after crée un [pseudo-élément](https://developer.mozilla.org/fr/docs/Web/CSS/Pseudo-éléments) qui sera le dernier enfant de l'élément sélectionné. Il est souvent utilisé pour ajouter du contenu cosmétique à un élément, en utilisant la propriété CSS [content](https://developer.mozilla.org/fr/docs/Web/CSS/content). Par défaut, ce contenu est de type « [en ligne](https://developer.mozilla.org/fr/Apprendre/CSS/Introduction_à_CSS/Le_modèle_de_boîte" \l "Les_types_de_boîte) ».

***Les tableaux***

Tableau > balise : table

En tête de tableau : thead

Corps du tableau : tbody

Pied de page de tableau : tfoot

Ligne du tableau : tr

Colonne du tableau : th (par défaut les th sont centrés)

Données de tableau : table data > td

<td **colspan**="2"> 5</td> > permet de fusionner **horizontalement** deux cellules d’un tableau (dans le html)

<td **rowspan**="2"> 5</td> ermet de fusionner **verticalement** deux cellules d’un tableau (dans le html)

Exemple (basique) de CSS pour un tableau :

/\* PARTIE TABLEAU \*/

table{

border-collapse: collapse;

width: 100%;

margin: 5px;

text-align: center;

}

th, td{

border:2px solid grey;

padding: 5px

}

***Les formulaires***

Les formulaires :

commencent par form…

resize: vertical; > pour empêcher qu’on modifie verticalement un text area

input[type="submit"]{

margin-bottom: 20px;

background-color: grey;

} > Pour sélectionner un input subtmit

[class^=bouton] > pour sélectionner tous les éléments qui commencent par button.

Pour rendre la page HTML compatible avec tous les navigateurs > normalize.css copier coller dans un document .css et un lien dans le head (comme pour le lien vers le css).

*Webkit-margin-before = margin-top*

***Media query pour le responsive***

*Toutes les fonctions CSS commencent par arobase « @ »*

*Pour être responsive il faut :*

*Sur le lien html (dans le head) <meta name="viewport" content="width=device-width">*

*+*

*Dans le CSS : @media screen and (min-width: 767px){*

*body{*

*background-color: red;*

*}*

*}*

***Vérifier la compatibilité entre différents navigateurs***

Pour vérifier la compatibilité de son code suivant les différents navigateurs > https://caniuse.com

***Iframe***

L'élément HTML <iframe> représente un contenu de navigation imbriqué qui permet en fait d'obtenir une page HTML intégrée dans la page courante. Le contexte de navigation qui contient le contenu intégré est appelé « contexte de navigation parent ». Le contexte de navigation le plus élevé (qui n'a pas de contexte parent) correspond généralement à la fenêtre du navigateur (cf. [Window](https://developer.mozilla.org/fr/docs/Web/API/Window)).

***Vidéos***

L'élément HTML <video> intègre un contenu vidéo dans un document.

<video src="fichiervideo.webm" autoplay poster="vignette.jpg">

Votre navigateur ne permet pas de lire les vidéos.

Mais vous pouvez toujours

<a href="fichiervideo.webm">la télécharger</a> !

</video>

\*{

box-sizing: border-box;

}

.container{

width:99%;

margin-left: auto;

margin-right: auto;

}

***Les positions***

La propriété position définit la façon dont un élément est positionné dans un document. Lesabsolut propriétés [top](https://developer.mozilla.org/fr/docs/Web/CSS/top), [right](https://developer.mozilla.org/fr/docs/Web/CSS/right), [bottom](https://developer.mozilla.org/fr/docs/Web/CSS/bottom) et [left](https://developer.mozilla.org/fr/docs/Web/CSS/left) déterminent l'emplacement final de l'élément positionné.

Par défaut, la position est définie en statique (position: static;).

Différents autre type de positions :

**position: relative;** > C ‘est un artifice visuel qui permet de faire bouger l’élément par rapport à son emplacement initial. l’élément est toujours dans le flux (dans l’emplacement initial). Il se positionne par rapport à lui-même.

**position: absolute;** > je sors du flux et donc l’espace que j’occupais est disponible de nouveau. Il se positionne par rapport à la page en elle-même (ou d’un autre élément définit en position relative ou en position absolute ou fixed (tout ce qui est autre que static)). Permet par exemple de mettre une croix en haut d’une fenêtre en faisant (sur l’élément enfant en absolute et à condition que le parent soit sur autre chose que static) : top : -10px ; right : -10px ;

**position: fixed;** > je sors du flux et l’élément ne bouge pas

Un élément positionné de façon relative est un élément dont la propriété de position calculée est relative. Dans ce cas, les propriétés [top](https://developer.mozilla.org/fr/docs/Web/CSS/top) ou [bottom](https://developer.mozilla.org/fr/docs/Web/CSS/bottom)indiquent le décalage vertical à appliquer et [left](https://developer.mozilla.org/fr/docs/Web/CSS/left) ou [right](https://developer.mozilla.org/fr/docs/Web/CSS/right) indiquent le décalage horizontal.

Un élément positionné de façon absolue est un élément dont la propriété de position calculée est absolute ou fixed. Dans ce cas, les propriétés [top](https://developer.mozilla.org/fr/docs/Web/CSS/top), [bottom](https://developer.mozilla.org/fr/docs/Web/CSS/bottom), [right](https://developer.mozilla.org/fr/docs/Web/CSS/right) et [left](https://developer.mozilla.org/fr/docs/Web/CSS/left) indiquent les distances entre les bords de l'élément et les bords du bloc englobant (c'est-à-dire l'ancêtre par rapport auquel l'élément est positionné). Si l'élément possède des marges, elles sont ajoutées aux décalages.

Cette valeur crée un nouveau [contexte d'empilement](https://developer.mozilla.org/fr/docs/Glossaire/Contexte_d_empilement) lorsque z-index ne vaut pas auto. Les éléments positionnés de façon absolue peuvent avoir des marges, ces marges ne fusionnent pas avec les autres marges (exemple : position: absolute;

z-index: 1;)

Un élément positionné en adhérence est un élément dont la propriété de position calculée vaut sticky. Un tel élément se comporte comme un élément positionné de façon relative jusqu'à ce que son bloc englobant dépasse un seuil donné (par exemple fourni par la valeur de [top](https://developer.mozilla.org/fr/docs/Web/CSS/top)) au sein du conteneur puis il se comporte ensuite comme un élément fixe jusqu'à atteindre le bord opposé du bloc englobant.

La propriété transform

La propriété transform permet de modifier l'espace de coordonnées utilisé pour la mise en forme visuelle. Grâce à cette propriété, il est possible de translater les éléments, de les tourner, d'appliquer des homothéties, de les distordre pour en changer la perspective.

Exemple pour afficher un menu avec transform:scaleY

.menucache{

/\*display: none;\*/

top:100%;

left: 0;

padding: 5px;

background-color: #30a894;

position: absolute;

transition: all 300ms;

transform: scaleY(0);

transform-origin: top;

}

li:hover .menucache{

display: block;

transform: scaleY(1);

}

Pour les dégradés de couleur :

background: linear-gradient(red,blue)

avec une ligne :

background: linear-gradient(150deg, white 50%, #333 50%)

Pour pouvoir manipuler des <a>, vaut mieux les mettre en inline-block

Pour permettre à CSS de revenir automatiquement à la ligne

sur \* :

word-break: break-word

Pour enregistrer une police (téléchargée autre que google font)

Dans le CSS :

@font-face{

font-family : « bodega »

src : url(../….) ;

}

Pour sélectionner que les enfants (syntaxe)

Exemple :

.products > \* {

code

}

autre exemple avec les petits enfants :

.products > \* > \* {

code

}

Pour vérifier la compatibilité avec les navigateurs

[http://shouldiprefix.com](http://shouldiprefix.com/)

<https://caniuse.com/>

Pour viser un élément précédé d’un autre élément

.advanced h2 + h2{

color: red;

}

Pour viser un H2 précédé d’un H2

.advanced h2 + h2 + h2{

color: red;

}

Pour viser un H2 précédé de deux H2

.advanced h2 + p{

color: red;

}

Pour viser un p précédé d’un H2

La FLEXBOX

La distribution, c’est à dire le sens d’affichage horizontal ou vertical des éléments "flex-items" est définie par la propriété flex-direction dont les valeurs peuvent être :

* row (distribution horizontale, valeur par défaut)
* row-reverse (distribution horizontale inversée)
* column (distribution verticale)
* column-reverse (distribution verticale inversée)

La propriété flex-wrap définit si le contenu sera distribué sur une seule ligne (ou colonne selon l’axe principal) ou sur plusieurs lignes. En clair, si les "flex-items" ont le droit de passer à la ligne ou non.

Les valeurs de flex-wrap sont :

* nowrap (les éléments ne passent pas à la ligne, valeur par défaut)
* wrap (les éléments passent à la ligne dans le sens de lecture)
* wrap-reverse (les éléments passent à la ligne dans le sens inverse)

#### Axe principal : justify-content

Les alignements dans l’axe de lecture principal sont définis à l’aide de la propriété justify-content, dont les valeurs possibles sont :

* flex-start (éléments positionnés au début du sens de lecture, valeur par défaut)
* flex-end (éléments positionnés à la fin)
* center (position centrale)
* space-between (répartition “justifiée”)
* space-around (variante de répartition “justifiée”)

#### Axe secondaire: align-items

Dans l’axe secondaire, les alignements sont régis via la propriété align-items, dont les valeurs sont :

* flex-start (au début)
* flex-end (à la fin)
* center (au centre)
* baseline (généralement identique à flex-start)
* stretch (étirés dans l’espace disponible, valeur par défaut)

#### Traiter les cas particuliers : align-self

La propriété align-self , permet de distinguer l’alignement d’un "flex-item" de ses frères. Les valeurs de cette propriété sont identiques à celles de align-items.

### Flexibilité

Cela ne devrait étonner personne, la notion de flexibilité constitue le fondement du module de positionnement Flexbox, et c’est là qu’intervient l’indispensable propriété flex.

La propriété flex est un raccourci de trois propriétés, flex-grow, flex-shrink et flex-basis, qui s’appliquent aux "flex-items" et dont les fonctionnalités sont:

* flex-grow : capacité pour un élément à s’étirer dans l’espace restant,
* flex-shrink : capacité pour un élément à se contracter si nécessaire,
* flex-basis : taille initiale de l’élément avant que l’espace restant ne soit distribué.

Par défaut, les valeurs de ces propriétés sont : flex-grow: 0, flex-shrink: 1 et flex-basis: auto.  
En clair, les flex-items n’occupent initialement que la taille minimale de leur contenu.

Pour rendre un élément flexible, il suffit de lui attribuer une valeur de flex-grow (ou flex en raccourci) supérieure à zéro.  
Cet élément occupera alors l’espace restant au sein de son conteneur :

Les Hover

nav i{

transition: all 1500ms;

}

nav a:hover i{

transform: translateY(-10px);

color: pink;

}

Les em

**em > la taille d’un caractère d’imprimerie > la taille du fontsize de l’élément**

**rem > la taille de la page HTML (par défaut, le plus souvent, 16 px)**

Vh

height: 100vh; > la taille de l’écran

Pour faire un carré au milieu de l’écran

.check-out{

background-color: red;

height: 100vh;

width: 100vw;

display: flex;

justify-content: center;

align-items: center;

}

.check-out-inner{

background-color: white;

height: 80vh;

width: 80vw;

}

Wordpress

Pour Wordpress, il faut :

Un serveur en local host via (WAMP,MAMP, …)

On peut utiliser MAMP même sur windows.

Après faut utiliser des fichiers wordpress sur wordpress.org

télécharger sur > <https://fr.wordpress.org/download/>

3 dossiers principaux :

Wp admin > l’administration (on touche pas)

Wp include > on touche pas

Wp content > on peut manipuler

Pour accéder à la base de données : dans localhost/phpmyadmin/

Par défaut, le mot de passe et login : root et root

Lors de la création d’une base de données choisir comme interclassement : utf8\_general\_ci

Ensuite on répercute les informations sur localhost > wordpress

wp template hierarchy > permet de savoir où sont placés les éléments wordpress (architecture)



* **Développement**

**JAVASCRIPT**

***Introduction***

[Thibaut.monesma@gmail.com](mailto:Thibaut.monesma@gmail.com)

<script src="exemple1.js"></script> Pour faire le lien entre la page html et la page Js. C’est à mettre systématiquement en bas de la page html. On évite ainsi les bugs.

document.write("HAHAHAH"); > écrit dans le document html

document.write("<p>bonjour</p>"); > il interprète les balises html

simple quote (guillemets) à favoriser car on peut intégrer dedans des guillemets double exemple : document.write('<p style="color:red">bonjour</p>');

/ permet de faire comprendre à Js que les « » sont du texte. Exemple : document.write('<p>j\'aime les fleurs</p>');

Pour écrire dans la console > console.log('ceci ne se voit que dans la console'); Très utile pour le débugage.

***Les tableaux (avec exemple date)***

var datedujour = new Date();

var weekday = new Array();

weekday[0] = "Dimanche";

weekday[1] = "Lundi";

weekday[2] = "Mardi";

weekday[3] = "Mercredi";

weekday[4] = "Jeudi";

weekday[5] = "Vendredi";

weekday[6] = "Samedi";

var j = weekday[datedujour.getDay()];

document.write(j + ' ');

var Auj = datedujour.getDate();

document.write(Auj + ' ');

var m = new Array();

m[0] = "Janvier";

m[1] = "Février";

m[2] = "Mars";

m[3] = "Avril";

m[4] = "Mai";

m[5] = "Juin";

m[6] = "Juillet";

m[7] = "Août";

m[8] = "Septembre";

m[9] = "Octobre";

m[10] = "Novembre";

m[11] = "Décembre";

var m = m[datedujour.getMonth()];

document.write(m + ' ');

document.write(datedujour.getUTCFullYear());

Autre exemple :

var name = 'Thibaut';

document.write('Je m\var datedujour = new Date();

var weekday = new Array();

weekday[0] = "Dimanche";

weekday[1] = "Lundi";

weekday[2] = "Mardi";

weekday[3] = "Mercredi";

weekday[4] = "Jeudi";

weekday[5] = "Vendredi";

weekday[6] = "Samedi";

var j = weekday[datedujour.getDay()];

document.write(j + ' ');

var Auj = datedujour.getDate();

document.write(Auj + ' ');

var m = new Array();

m[0] = "Janvier";

m[1] = "Février";

m[2] = "Mars";

m[3] = "Avril";

m[4] = "Mai";

m[5] = "Juin";

m[6] = "Juillet";

m[7] = "Août";

m[8] = "Septembre";

m[9] = "Octobre";

m[10] = "Novembre";

m[11] = "Décembre";

var m = m[datedujour.getMonth()];

document.write(m + ' ');

document.write(datedujour.getUTCFullYear());'appelle ' + name + ' je suis de la PA 121');

***Les variables***

La variable permet de stocker une donnée qui peut par la suite changé.

Pour déclarer la variable on fait VAR avant la valeur.

Pour l’utiliser ensuite pas besoin de mettre VAR avant.

La constant même principe qu’une variable mais elle évolue pas. Elle se note comme ça :

const FIRST\_CONSTANT = 'Cec'

Chaque variable a un type. Il y a trois types principaux. Le type string (chaine de caractères), le type nombre et enfin le type boolean (true ou false).

***ParseInt***

La méthode parseInt permet de transformer du string en nombre.

ParseInt> ne fonctionne qu’avec les nombres entiers (s’il y a une virgule il va arrondir le nombre) > pour faire avec des nombres décimaux on fait un parseFloat

***Incrémentation***

Pour incrémenter , il y a un raccourci le ++, exemple > nombre++;

Pour incrémenter de +3 on fait nombre+= 3

***Concaténéation***

Concaténation > c’est l’idée de permettre à js d’interpréter une variable et de la séparer d’une chaine de caractères.

exemple var name = 'Thibaut';

document.write('Je m\'appelle name') > ne fonctionne pas (il écrit name)

Par contre si on fait : var name = 'Thibaut';

document.write('Je m\'appelle ' + name) > il va interpréter la variable name

Dès qu’il y a concaténation avec une string l’ensemble de la variable devient une string. Pas d’opérations arithmétiques possibles...

***Window.prompt***

window.prompt > pour qu’une boîte de dialogue s’ouvre. > par défaut, ce qui a été tapé va être un string > il faut faire un parseInt derrière si on veut transformer en nombre.\

***Les opérateurs de comparaison***

> (supérieur à)

< (inférieur à)

### Inégalité simple (!=)

L'opérateur d'inégalité simple renvoie true si les deux opérandes ne sont pas égaux. Si les deux opérandes ne sont pas du même type, une conversion sera effectuée vers un type adéquat. Si les deux opérandes sont des objets, le moteur JavaScript comparera les références internes pour voir si elles réfèrent à des objets différents en mémoire.

Syntaxe

x != y

### Égalité simple (==)

L'opérateur d'égalité simple convertit les deux opérandes s'ils ne sont pas du même type, ensuite la comparaison stricte est appliquée. Si les deux opérandes sont des objets, le moteur JavaScript comparera les références internes pour voir si elles réfèrent au même objet en mémoire.

#### Syntaxe

x == y

#### Exemples

1 == 1; // true

"1" == 1; // true

### Inégalité simple (!=)

L'opérateur d'inégalité simple renvoie true si les deux opérandes ne sont pas égaux. Si les deux opérandes ne sont pas du même type, une conversion sera effectuée vers un type adéquat. Si les deux opérandes sont des objets, le moteur JavaScript comparera les références internes pour voir si elles réfèrent à des objets différents en mémoire.

#### Syntaxe

#### if (i!==j){

x != y

#### Exemples

1 != 2; // true

1 != "1"; // false

### Égalité stricte (===)

L'opérateur d'égalité stricte renvoie true si les opérandes sont strictement égaux (voir ci-avant), aucune conversion de type n'est effectuée.

#### Syntaxe

x === y

#### Exemples

3 === 3 // true

3 === '3' // false

### Inégalité stricte (!==)

L'opérateur d'inégalité stricte renvoie true si les opérandes sont de types différents ou ne sont pas égaux.

#### Syntaxe

x !== y

#### Exemples

3 !== '3' // true

4 !== 3 // true

***Exercice calcul TVA***

Calculatrice de TVA (corrigé) :

const TAUX\_DE\_TVA = 20;

var montantHT;

var montantTTC;

var montantTVA;

montantHT = window.prompt('Quel est le montant HT');

montantHT = parseFloat(montantHT);

montantTVA = montantHT \* (TAUX\_DE\_TVA / 100);

montantTTC = montantHT + montantTVA;

document.write

( '<p>Pour un montant HT de ' + montantHT + ' € il y a ' + montantTVA + ' € de TVA.</p>');

document.write('<p>Le montant TTC est donc de ' + montantTTC + ' €.</p>');

***Autre application des tableaux avec les dates***

Les tableaux sont dits arrays.

Pour les déclarer on tape « new ».

Le mode strict apporte quelques changements à la sémantique « normale » de JavaScript:

Le mode strict élimine quelques erreurs silencieuses de JavaScript en les changeant en erreurs explicites (une exception sera levée).

Le mode strict corrige les erreurs qui font qu'autrement il est difficile pour les moteurs JavaScript d'effectuer des optimisations. Le code sera donc exécuté plus rapidement en mode strict, sans changer une seule ligne si cela n'est pas nécessaire.

Le mode strict interdit les mot-clés susceptibles d'être définis dans les futures versions de ECMAScript.

Exemple : weekDays = new Array();

weekDays[0] = 'Lundi';

weekDays[1] = 'Mardi';

weekDays[2] = 'Mercredi';

weekDays[3] = 'Jeudi';

weekDays[4] = 'Vendredi';

weekDays[5] = 'Samedi';

weekDays[6] = 'Dimanche';

Le premier index dans un tableau est toujours zéro.

Si on veut afficher samedi dans le cas ci-dessus on fait document.write(weekDays[5]);

Autre écriture :

weekDays = [ 'Lundi', 'Mardi', 'Mercredi', 'Jeudi', 'Vendredi', 'Samedi', 'Dimanche' ]

On peut mettre une variable à l’intérieur des crochets (au lieu de l’index) mais ça peut être qu’une variable nombre.

Exemple  :

var sundayIndex = 6;

document.write('et le dernier jour de la semaine est le ' + weekDays[sundayIndex] + '.');

Corrigé :

var dayNames = new Array();

var monthNames = new Array();

dayNames[0] = 'Dimanche';

dayNames[1] = 'Lundi';

dayNames[2] = 'Mardi';

dayNames[3] = 'Mercredi';

dayNames[4] = 'Jeudi';

dayNames[5] = 'Vendredi';

dayNames[6] = 'Samedi';

monthNames[0] = 'Janvier';

monthNames[1] = 'Février';Le mode strict apporte quelques changements à la sémantique « normale » de JavaScript:

Le mode strict élimine quelques erreurs silencieuses de JavaScript en les changeant en erreurs explicites (une exception sera levée).

Le mode strict corrige les erreurs qui font qu'autrement il est difficile pour les moteurs JavaScript d'effectuer des optimisations. Le code sera donc exécuté plus rapidement en mode strict, sans changer une seule ligne si cela n'est pas nécessaire.

Le mode strict interdit les mot-clés susceptibles d'être définis dans les futures versions de ECMAScript.

monthNames[2] = 'Mars';

monthNames[3] = 'Avril';

monthNames[4] = 'Mai';

monthNames[5] = 'Juin';

monthNames[6] = 'Juillet';

monthNames[7] = 'Août';

monthNames[8] = 'Septembre';

monthNames[9] = 'Octobre';

monthNames[10] = 'Novembre';

monthNames[11] = 'Décembre';

today = new Date();

document.write('Nous sommes le ' + dayNames[today.getDay()] + ' '); // dayNames[5]

document.write(today.getDate() + ' ');

document.write(monthNames[today.getMonth()] + ' '); //monthNames[8];

document.write(today.getFullYear() + '.');

***Les objets***

Les objets permettent de stocker plusieurs informations. Plusieurs caractéristiques pour un même élément.

Deux manières de déclarer un objet :

var car = new Object();

ou \

var car = {} ;

Ensuite on peut désigner une caractéristique de l’objet comme ça :

car['brand'] = 'Alfa Roméo';

ou

car.brand = 'Alfa Roméo';

Enfin, on peut utiliser ces caractéristiques comme ça :

console.log(car['brand']);

ou

console.log(car.brand);

Exemple complet :

var car = new Object();

car.brand = 'Alfa Roméo';

car.color = 'red';

car.year = '1974';

document.write('Voici une : ' + car.brand + ' ' + car.color + ' de ' + car.year);

**Exemple de comparaison avec une valeur en boolean :**

var age;

var isThirtyYearsOld;

var isAdult;

var isNotCentennial;

age = window.prompt('Quel est votre âge ?');

isThirtyYearsOld = (age == 30);

isAdult = (age >= 18);

isNotCentennial = (age != 100);

document.write('<p>Vous avez trente ans : ' + isThirtyYearsOld + '.</p>');

Dans cet exemple, à la place de isThirtyYearsOld, il sera affiché false sauf si l’âge tapé est 30 ans.

***Les conditions***

**Une condition (syntaxe : if) pour afficher « Vous êtes majeur » en fonction du résultat de âge :**

if (age > 18) {

document.write('<p>Vous êtes majeur.</p>');

}

**Condition avec un else :**

if(age < 20)

{

document.write('<p>Vous avez moins de 20 ans.</p>');

}

else

{

document.write('<p>Vous avez plus de 20 ans.</p>');

}

**Condition avec trois cas de figure (else if) :**

if (firstName == 'Elsa')

{

document.write("<p>Vous vous appelez Elsa.</p>");

}

else if(firstName == 'Charlotte')

{

document.write("<p>Vous vous appelez Charlotte.</p>");

}

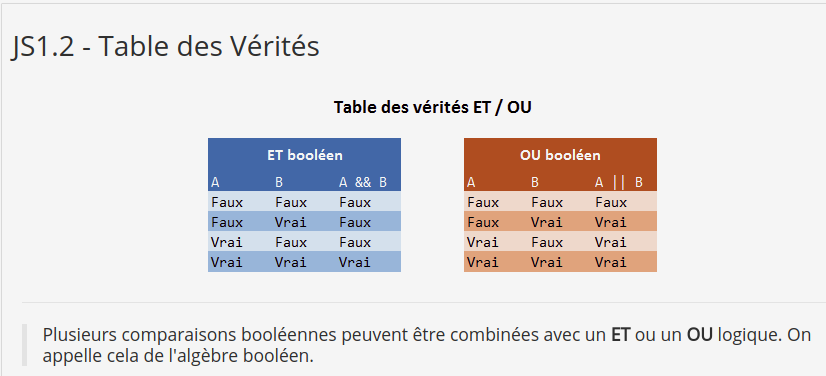
else

{

document.write("<p>Vous ne vous appelez ni Elsa ni Charlotte.</p>");

}

***Les booleans***



***Exemple de double condition (avec ou || et and)***

**Exemple de condition avec ou :**

if(firstName == 'Tom' || age != 40)

{

document.write("<p>Vous vous appelez Tom et vous n'êtes pas quadra.</p>");

}

**Autre exemple avec and :**

|  |  |  |
| --- | --- | --- |
| && | and | (x < 10 && y > 1) is true |

***Le mode strict***

Le mode strict apporte quelques changements à la sémantique « normale » de JavaScript:

1. Le mode strict élimine quelques erreurs silencieuses de JavaScript en les changeant en erreurs explicites (une exception sera levée).
2. Le mode strict corrige les erreurs qui font qu'autrement il est difficile pour les moteurs JavaScript d'effectuer des optimisations. Le code sera donc exécuté plus rapidement en mode strict, sans changer une seule ligne si cela n'est pas nécessaire.
3. Le mode strict interdit les mot-clés susceptibles d'être définis dans les futures versions de ECMAScript.

Pour invoquer le mode strict pour un script entier, on ajoutera l'instruction exacte "use strict"; (ou 'use strict';) avant toutes les autres instructions.

// Script entier en mode strict

"use strict";

var v = "Allo ! Je suis en mode strict !";

***Correction exercice TVA après remise***

<script>

'use strict';

const TAUX\_DE\_TVA = 20;

var pourcentage;

var montantTVA;

var montantTTC;

var montantHT = parseFloat(window.prompt('Quel est le montant HT ?'));

var remise = window.prompt('Souhaitez-vous une remise ?');

if (remise == 'oui' || remise == 'yes') {

pourcentage = parseFloat(window.prompt('Montant de la remise en % :'));

montantHT = montantHT - (montantHT \* pourcentage / 100);

}

montantTVA = montantHT \* TAUX\_DE\_TVA / 100;

montantTTC = montantHT + montantTVA;

document.write('<p>Pour un montant HT de ' + montantHT + ' € il y a ' + montantTVA + ' € de TVA.</p>');

document.write('<p>Le montant TTC est donc de ' + montantTTC + ' €.</p>');

if (remise == 'oui' || remise == 'yes') {

document.write('<p>Une remise de ' + pourcentage + '% a été appliquée sur le montant HT.</p>');

} else {

document.write("<p>Aucune remise n'a été appliquée.</p>");

}

</script>

<script>

var name = window.prompt('votre nom ?');

if (name == 'Paul') {

document.write('Vous êtes Paul');

} else if (name == 'Pierre') {

<script>

var name = window.prompt('votre nom ?');

if (name == 'Paul') {

document.write('Vous êtes Paul');

} else if (name == 'Pierre') {

document.write('Vous êtes Paul');

} else if (name == 'Jacque') {

document.write('Vous êtes Jacque');

} else if (name == 'Ginette') {

document.write('Vous êtes Ginette');

} else {

document.write('Je ne vous connais pas');

}

***Exemple de Else if (avec beaucoup de conditions)***

</script>

document.write('Vous êtes Paul');

} else if (name == 'Jacque') {

document.write('Vous êtes Jacque');

} else if (name == 'Ginette') {

document.write('Vous êtes Ginette');

} else {

document.write('Je ne vous connais pas');

}

</script>

***Switch (plus rapide que Else If quand on a beaucoup de conditions)***

L'instruction switch évalue une expression et, selon le résultat obtenu et le cas associé, exécute les instructions correspondantes.

Exemple :

switch(name) {

case 'Paul':

case 'PAUL':

document.write('Vous êtes Paul');

br+ "<br>"eak;

case '('Le resultat est de ' + operation);Pierre':

document.write('Vous êtes Pierre');

break;

case 'Jacque':

document.write('Vous êtes Jacque');

break;

case 'Ginette':

document.write('Vous êtes Ginette');

break;

}

## 

A noter : le break est un mot clé qui permet d’arrêter le code.

***La fonctionnalité isNaN()***

La fonction isNaN() permet de déterminer si une valeur est [NaN](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/NaN). On notera que cette fonction utilise des règles de conversion différentes de [Number.isNaN()](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Number/isNaN), défini avec ECMAScript 2015 (ES6).

Exemple :

if (isNaN(x)) {

return 'Not a Number!';

}

***Correction de l’exercice calculette***

<script>

var number1 = parseFloat(window.prompt('Saisissez un premier nombre :'));

var number2 = parseFloat(window.prompt('Saisissez un deuxième nombre :'));

var operation = window.prompt('Quelle opération mathématique souhaitez-vous effectuer ?');

var result;

switch(operation)

{

case '+':

case 'addition':

result = number1 + number2;

break;

case '-':

case 'soustraction':

result = number1 - number2;

break;

case '\*':

case 'multiplication':

result = number1 \* number2;

break;

La fonction Math.round() retourne la valeur d'un nombre arrondi à l'entier le plus proche.

case '/':La fonction Math.round() retourne la valeur d'un nombre arrondi à l'entier le plus proche.

case 'division':

if (number2 == 0) {

document.write('<p>Erreur : vous ne pouvez pas diviser un nombre par 0 !</p>');

break;

}

result = number1 / number2;

break;

case '^':

case 'puissance':

result = number1 \*\* number2;

break;

default:

document.write("<p>Erreur : vous avez indiqué une opération inconnue !</p>");

break;

}

if (isNaN(result) == true || isNaN(number1) == true) {

document.write("<p>Vous n'avez pas saisi deux nombres !</p>");

result = undefined;

}

if(result != undefined)

{

document.write("<p>Le résultat de l'opération est <strong>" + result + '</strong>.</p>');

}

</script>

***Math.round***

La fonction Math.round() retourne la valeur d'un nombre arrondi à l'entier le plus proche.

## Syntaxe

Math.round+ "<br>"(*x*)

***Math.random***

La fonction Math.random() renvoie un nombre flottant pseudo-aléatoire compris dans l'intervalle [0, 1[ (ce qui signifie que 0 est compris dans l'intervalle mais que 1 en est exclu) selon une distribution approximativement uniforme sur cet intervalle.

## Syntaxe

Math.random()

***Correction exercice chifoumi***

<script>

'use strict';

var computer;

var player;

var random;

player window.prompt('Choissez : Pierre, Feuille, Ciseau');

player = player.toLowerCase();

random = Math.random();

if(random < (1/3)) {

computer = 'pierre';

}

else if(random < 2 / 3) // Entre ~0.33 et ~0.66 : l'ordinateur sélectionne la feuille

{

computer = 'feuille';

}

else const PA121 = ['Jean', 'Alan', 'Max', 'Younes', 'Marc', 'Julien', 'Rémi', 'Saliou', 'Laurent', 'Marie', 'Camille', 'Céline', 'Pierre', 'Baptiste' ]

document.write('Dans la PA121 il y a :');

for(var index = 0; index < PA121.length; index++) {

document.write('<p> - '+PA121[index]+',</p>')

} // Au-delà de ~0.66 : l'ordinateur sélectionne le ciseau

{

computer = 'ciseau';

}

if (computer == player) {

document.write('<p>Vous avez choisi la même chose : égalité !</p>');

} else {

switch(computer) {

case 'ciseau':

if(player == 'pierre')

{

document.write('<p>La pierre écrase le ciseau : vous gagnez !</p>');

}

else // player == 'feuille'

{

document.write('<p>La feuille est découpée par le ciseau : vous perdez !</p>');

}

break;

case 'feuille':

if(player == 'pierre')

{

document.write('<p>La pierre est enveloppée par la feuille : vous perdez !</p>const PA121 = ['Jean', 'Alan', 'Max', 'Younes', 'Marc', 'Julien', 'Rémi', 'Saliou', 'Laurent', 'Marie', 'Camille', 'Céline', 'Pierre', 'Baptiste' ]

document.write('Dans la PA121 il y a :');

for(var index = 0; index < PA121.length; index++) {

document.write('<p> - '+PA121[index]+',</p>')

}');

}

else // player == 'ciseau'

{

document.write('<p>Le ciseau découpe la feuille : vous gagnez !</p>');

}

break;

case 'pierre':

if(player == 'feuille')

{

document.write('<p>La feuille enveloppe la pierre : vous gagnez !</p>');

}

else // player == 'ciseau'

{

document.write('<p>Le ciseau est écrasé par la pierre : vous perdez !</p>');

}

break;

default:

document.write('le puit n\'existe pas en chifoumi');

break;

}

}

</script>

***Les boucles***

Les boucles permettent de répéter des actions simplement et rapidement.

Il y a deux types de boucles : -while

-for

- et do while

***While***

Exemple while :

n= 0 ;

while (n < 50) {***While***

document.write(« Hello ») ;

n++;

}const PA121 = ['Jean', 'Alan', 'Max', 'Younes', 'Marc', 'Julien', 'Rémi', 'Saliou', 'Laurent', 'Marie', 'Camille', 'Céline', 'Pierre', 'Baptiste' ]

document.write('Dans la PA121 il y a :');***While***

for(var index = 0; index < PA121.length; index++) {

document.write('<p> - '+PA121[index]+',</p>')

}

document.write

***For***

## L'instruction for

Une [boucle for](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/statements/for) répète des instructions jusqu'à ce qu'une condition donnée ne soit plus vérifiée. La boucle for JavaScript ressemble beaucoup à celle utilisée en C ou en Java. Une boucle for s'utilise de la façon suivante :

for ([expressionInitiale]; [condition]; [expressionIncrément])

instruction

Voici ce qui se passe quand une boucle for s'exécute :

1. L'expression initiale expressionInitiale est exécutée si elle est présente. Généralement, on utilise cette expression pour initialiser un ou plusieurs compteurs dont on se servira dans la boucle. Il est possible d'utiliser des expressions plus complexes si besoin. Elle peut servir à déclarer des variables.
2. L'expression condition est évaluée, si elle vaut true, les instructions contenues dans la boucle sont exécutées. Si la valeur de condition est false, la boucle forse termine. Si la condition est absente, elle est considérée comme true.
3. L'instruction instruction est exécutée. Si l'on souhaite exécuter plusieurs instructions, on utilisera un bloc d'instructions ({ ... }) afin de les grouper.
4. Si elle est présente, l'expression de mise à jour expressionIncrément est exécutée.
5. On retourne ensuite à l'étape 2.

Exemple :

var count;

for (count = 0; count < 10; count++) {

document.write('<p>La boucle for</p>');

}

Autre exemple :

const PA121 = ['Jean', 'Alan', 'Max', 'Younes', 'Marc', 'Julien', 'Rémi', 'Saliou', 'Laurent', 'Marie', 'Camille', 'Céline', 'Pierre', 'Baptiste' ]

document.write('Dans la PA121 il y a :');

for(var index = 0; index < PA121.length; index++) {

document.write('<p> - '+PA121[index]+',</p>')

}

***Do While***

L'instruction do...while crée une boucle qui exécute une instruction jusqu'à ce qu'une condition de test ne soit plus vérifiée. La condition est testée après que l'instruction soit exécutée, l'instruction définie dans la boucle est donc exécutée au moins une fois.

do {

i += 1;

console.log(i);

} while (i < 5);

);

***Correction de l’exercice la phrase la plus longue***

<script>

'use strict'; // Mode strict du JavaScript

var phrases =

[

"Lorem ipsum dolor sit amet, consectetuer adipiscing elit.",

"Aenean commodo ligula eget dolor. Aenean massa.",

"Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.",

"Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem.",

"Nulla consequat massa quis enim.",

"Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu.",

"In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo.",

"Nullam dictum felis eu pede mollis pretium.",

"Integer tincidunt. Cras dapibus.",

"Vivamus elementum semper nisi.",

"Aenean vulputate eleifend tellus.",

"Aenean leo ligula, porttitor eu, consequat vitae, eleifend ac, enim.",

"Aliquam lorem ante, dapibus in, viverra quis, feugiat a, tellus.",

"Phasellus viverra nulla ut metus varius laoreet.",

"Quisque rutrum. Aenean imperdiet.",

"Etiam ultricies nisi vel augue.",

"Curabitur ullamcorper ultricies nisi. Nam eget dui. Etiam rhoncus.",

"Lorem ipsum dolor sit amet, consectetuer adipiscing elit.",

"Aenean commodo ligula eget dolor. Aenean massa.",

"Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus.",

"Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem.",

"Nulla consequat massa quis enim.",

"Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu.",

"In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo.",

"Nullam dictum felis eu pede mollis pretium.",

"Integer tincidunt. Cras dapibus.",

"Vivamus elementum semper nisi.",

"Aenean vulputate eleifend tellus.",

"Aenean leo ligula, porttitor eu, consequat vitae, eleifend ac, enim.",

"Aliquam lorem ante, dapibus in, viverra quis, feugiat a, tellus.",

"Phasellus viverra nulla ut metus varius laoreet.",

"Quisque rutrum. Aenean imperdiet.",

"Etiam ultricies nisi vel augue.",

"Curabitur ullamcorper ultricies nisi. Nam eget dui. Etiam rhoncus.",

"Maecenas tempus, tellus eget condimentum rhoncus, sem quam semper libero, sit ipsuminus max.",

"Nam quam nunc, blandit vel, luctus pulvinar, hendrerit id, lorem.",

"Maecenas nec odio et ante tincidunt tempus.",

"Donec vitae sapien ut libero venenatis faucibus. Nullam quis ante.",

"Etiam sit amet orci eget eros faucibus tincidunt. Duis leo.",

"Sed fringilla mauris sit amet nibh. Donec sodales sagittis magna.",

"Sed consequat, leo eget bibendum sodales, augue velit cursus nunc."

];

var indexLongestPhrase = 0;

for (var i = 1; i < phrases.length ; i++) {

if (phrases[i].length > phrases[indexLongestPhrase].length) {

indexLongestPhrase = i;

}

}

document.write

(

'<p>' +

"La phrase la plus longue est : '" + phrases[indexLongestPhrase] + "' " +

'et elle fait ' + phrases[indexLongestPhrase].length + ' caractère(s) !' +

'</p>'

);

***Correction de l’exercice s****aisie obligatoire*

## Enoncé

Demander à l'utilisateur de saisir obligatoirement un nombre quoiqu'il arrive, afficher ce nombre ensuite en HTML.

var number;

do {

number = parseFloat(window.prompt('Veuillez saisir un nombre :'));

} while (isNaN(number) == true )

document.write('<p>Merci, vous avez saisi <strong>' + number + '</strong>.</p>');

***Correction*** *pour l’exercice table de multiplication*

var size = parseInt(window.prompt('Taille de la table de multiplications ?'));

document.write('<table>');

for (var row = 1; row <= size; row ++) {

document.write('<tr>');

for(var column = 1; column <= size; column++) {

var result = row \* column;

if (row == column) {

document.write('<td class="same">'+ result +'</td>');

} else {

document.write('<td>'+ result +'</td>');

}

}

document.write('</tr>');

}

document.write('</table>');

*Les fonctions*

Une fonctionest une portion de code qui peut être appelée par d'autres codes ou par elle-même ou par une [variable](https://developer.mozilla.org/fr/docs/Glossaire/Variable)qui se réfère à la fonction. Lorsqu'une fonction est appelée, des [arguments](https://developer.mozilla.org/fr/docs/Glossaire/Argument)lui sont généralement donnés en entrée.

Exemple :

function sayHello2(firstName, lastName) {

document.write('<p>Bonjour ' + firstName + ' ' + lastName + ' !</p>');

}

sayHello2('Marie', 'MAYERS');

Output :

Bonjour Marie MAYERS !

*Return*

L'instruction return met fin à l'exécution d'une fonction et définit une valeur à renvoyer à la fonction appelante.

Exemple :

function carre(x) {

return x \* x;

}

var demo = carre(3);

// demo vaudra alors 9

C’est pour renvoyer une valeur. Quand on a besoin de récupérer une valeur. Pour la stocker ensuite dans une variable.

On peut utiliser ensuite la variable à sa guise.

Ça sert à renvoyer de la valeur.

Exemple :

function sayHello3()

{

return 'Bonjour à toutes et à tous !';

}

var message = sayHello3();

document.write(message);

*D’autres fonctions précodées*

# **Méthode :**[**Array**](https://www.toutjavascript.com/reference/ref-array.php)**.push()**

## **Ajoute de nouveaux éléments en fin de tableau**

**Syntaxe**  
**Integer tableau.push(element\_1 [, ..., element\_n])**  
  
**Description**  
Ajoute les éléments **element\_1** à **element\_n** en fin du tableau [**Array**](https://www.toutjavascript.com/reference/reference.php?ref=Array&parent=125) **tableau**.  
  
Voir [**unshift()**](https://www.toutjavascript.com/reference/reference.php?ref=unshift&parent=125) qui insère des éléments en début de tableau.  
Voir aussi [**concat()**](https://www.toutjavascript.com/reference/reference.php?ref=concat&parent=125) pour concaténer deux tableaux ensemble.  
  
Notez que **push()** ne retourne pas de nouveau tableau, mais seulement la nouvelle taille du tableau. C'est le tableau d'origine qui est directement modifié par l'appel.

Exemple1 :

var name = ['Gégé', 'Pedro', 'Beber'];

name.push('Cricri'); // ajoute Cricri dans le tableau name

**Exemple2**

**Code source**

<script type="text/javascript">  
[**var**](https://www.toutjavascript.com/reference/reference.php?ref=var) tab=[**new**](https://www.toutjavascript.com/reference/reference.php?ref=new) [**Array**](https://www.toutjavascript.com/reference/reference.php?ref=Array)("Pommes", "Poires", "Ananas", "Cerise");  
[**var**](https://www.toutjavascript.com/reference/reference.php?ref=var) nb=tab.[**push**](https://www.toutjavascript.com/reference/reference.php?ref=push)("Banane", "Fraise")  
[**document**](https://www.toutjavascript.com/reference/reference.php?ref=document).[**write**](https://www.toutjavascript.com/reference/reference.php?ref=write)(tab.[**join**](https://www.toutjavascript.com/reference/reference.php?ref=join)(", "));  
[**console**](https://www.toutjavascript.com/reference/reference.php?ref=console).[**log**](https://www.toutjavascript.com/reference/reference.php?ref=log)("Nombre d'éléments dans le tableau : "+nb);  
</script>

**Résultat**

Pommes, Poires, Ananas, Cerise, Banane, Fraise

Nombre d'éléments dans le tableau : 6

# **Méthode : indexof**

# name.indexOf('Gégé'); // renvoie l'index où est stocké la donnée dans le tableau name

# // si pas de donnée renvoie -1

# exemple :

# var beasts = ['ant', 'bison', 'camel', 'duck', 'bison'];

# console.log(beasts.indexOf('bison')); // renvoi 1

# console.log(beasts.indexOf('camel')); // renvoi 2

# console.log(beasts.indexOf('bison', 2)); // renvoi 4

# console.log(beasts.indexOf('giraffe')); // renvoi -1

# **Méthode : splice**

# name.splice(1, 1): // supprime pedro (argument 1 : index du premier élément à supprimer, argument 2 : nombre d'éléments à supprimer après l'index de l'argument 1

*Correction exercice listedecourses*

var shoppingList = new Array();

function addItem(item)

{

shoppingList.push(item);data

}

function displayShoppingList() {

console.log('La liste contient ' + shoppingList.length + ' produit(s).');

console.log(shoppingList);

}

function removeItem(item){

var index = shoppingList.indexOf(item);

if (index == -1) {

console.log("ERREUR : le produit " + item + " n'existe dans la liste de courses");

} else {

shoppingList.splice(index,1);

}

}

function removeAllItems()

{

shoppingList = new Array();

}

addItem('Fraises');

addItem('Poulet');

addItem('Tournevis');

addItem('Artichaut');

addItem('couteau pour le chat');

displayShoppingList();

removeItem(window.prompt('Quel produit de la liste de courses souhaitez-vous supprimer ?'));

displayShoppingList();

removeAllItems();

displayShoppingList();

*Notes DragonSlayer*

*Demander infos du joueur*

*HP player(1er window prompt)*

*HP dragon(1er window prompt)*

*> si facile :le dragon a entre 150hp et 200hp*

*> si facile :le chevalier a entre 200hp et 250hp*

*> si moyen :le dragon a entre 200hp et 250hp*

*> si moyen :le chevalier a entre 200hp et 250hp*

*> si difficile :le dragon a entre 200hp et 250hp*

*> si difficile :le chevalier a entre 150hp et 200hp*

*Armor ratio player(2me window prompt)*

*> si cuire : base du ratio est de : 1*

*> si fer :base du ratio est de : 1,25*

*> si magique :base du ratio est de : 2*

*Weapon ratio player(3me window prompt)*

*> si bois : base du ratio \* par 0,5*

*> si fer : base du ratio \* par 1*

*> si excalibur  :base du ratio \* par 2*

***CorrectionDragonSlayer***

var game = {};

function initializeGame()

{

game.difficulty = requestInteger( 'Niveau de difficulté ?\n' + '1. Facile - 2. Normal - 3. Difficile',1, 3);

switch(game.difficulty)

{

case 1:

game.hpDragon = getRandomInteger(150, 200);

game.hpPlayer = getRandomInteger(200, 250);

break;

case 2:

game.hpDragon = getRandomInteger(200, 250);

game.hpPlayer = getRandomInteger(200, 250);

break;

case 3:

game.hpDragon = getRandomInteger(200, 250);

game.hpPlayer = getRandomInteger(150, 200);

break;

}

game.armor = requestInteger('Armure ?\n' + '1. Cuivre - 2. Fer - 3. Magique', 1, 3);

if (game.armor == 1) {

game.armorRatio = 1;

} else if (game.armor == 2) {

game.armorRatio = 1.25;

} else {

game.armorRatio = 2;

}

game.sword = requestInteger('Epée ?\n' +'1. Bois - 2. Acier - 3. Excalibur',1, 3 );

switch(game.sword){

case 1:

game.swordRatio = 0.5;

break;

case 2:

game.swordRatio = 1;

break;

case 3:

game.swordRatio = 2;

break;

}

}

initializeGame();

console.log(game);

function gameLoop() {

var damagePoint;

var dragonSpeed;

var playerSpeed;

while (game.hpDragon > 0 && game.hpPlayer > 0) {

dragonSpeed = getRandomInteger(10, 20);

playerSpeed = getRandomInteger(10, 20);

if(dragonSpeed > playerSpeed)

{

damagePoint = computeDragonDamagePoint();

game.hpPlayer -= damagePoint; // game.hpPlayer = game.hpPlayer - damagePoint;

console.log

(

'Le dragon est plus rapide et vous brûle, il vous enlève ' +

damagePoint + ' PV'

);

} else {

damagePoint = computePlayerDamagePoint();

game.hpDragon -= damagePoint;

console.log

(

'Vous êtes plus rapide et frappez le dragon, vous lui enlevez ' +

damagePoint + ' PV'

);

}

showGameState();

}

}

function showGameState()

{

console.log

(

'Dragon : ' + game.hpDragon + ' PV, ' +

'joueur : ' + game.hpPlayer + ' PV'

);

}

function showGameWinner()

{

if(game.hpDragon <= 0)

{

document.write('<img src="images/knight.jpg">');

console.log("Vous avez gagné, vous êtes vraiment fort !");

console.log("Il vous restait " + game.hpPlayer + " PV");

}

else // if(game.hpPlayer <= 0)

{

document.write('<img src="images/dragon.jpg">');

console.log("Le dragon a gagné, vous avez été carbonisé !");

console.log("Il restait " + game.hpDragon + " PV au dragon");

}

}

function start() {

initializeGame();

showGameState();

gameLoop();

showGameWinner();

}

start();

## Document Object Model

Au fur et à mesure de la lecture du fichier **HTML** le navigateur web construit dans la mémoire une structure de données représentant chaque balise **HTML**.

Cette structure de données, le **Document Object Model** (DOM), est en forme d'arbre, comme une pyramide :

* la racine de l'arbre, ou haut de la pyramide, est la balise *html*,
* les descendants directs (ou enfants) les balises *head* et *body*,
* et ainsi de suite.

On dit également que les balises *head* et *body* sont frères (siblings).

Le **DOM** est la structure de données fondamentale du navigateur web et sert autant au **CSS** qu'au **JavaScript**.

Exemple :

var para1 = document.getElementById('para1');

para1.style.color = "red";

para1.style.backgroundColor = "green";

var myPara = document.querySelectorAll('.my-para');

myPara[0].style.color = "blue";

for (var i = 0; i < myPara.length; i++) {

myPara[i].style.backgroundColor = "pink";

}

para1.classList.add('blue');

para1.classList.remove('useless');

var event = document.getElementById('para2');

function myFunction() {

event.classList.toggle('blue');

}

event.addEventListener('click', myFunction);

console.log(myPara);

Tous les événements:

> <https://www.w3schools.com/js/js_htmldom_eventlistener.asp>

**Correction exercice sur les souris :**

var button = document.getElementById('toggle-rectangle');

var rectangle = document.querySelector('.rectangle');

function onClickButtonDisplay(){

rectangle.classList.toggle('hide');

}

function onMouseOverRectangle() {

rectangle.classList.add('important');

}

function onDoubleClickRectangle()

{

rectangle.classList.toggle('good');

}

function onMouseOutRectangle()

{

rectangle.classList.remove('good');

rectangle.classList.remove('important');

}

button.addEventListener('click', onClickButtonDisplay);

rectangle.addEventListener('mouseover', onMouseOverRectangle);

rectangle.addEventListener('dblclick', onDoubleClickRectangle);

rectangle.addEventListener('mouseout', onMouseOutRectangle);

C**orrection Exercice sélecteur de photos :**

<!DOCTYPE html>

<html lang="fr">

<head>

<meta charset="utf-8">

<title>JavaScript</title>

<link rel="stylesheet" href="css/main.css">

</head>

<body>

<header>

<h1>Sélecteur de Photos</h1>

</header>

<main>

<ul class="photo-list">

<li><img src="images/1.jpg"></li>

<li><img src="images/2.jpg"></li>

<li><img src="images/3.jpg"></li>

<li><img src="images/4.jpg"></li>

<li><img src="images/5.jpg"></li>

<li><img src="images/6.jpg"></li>

<li><img src="images/7.jpg"></li>

<li><img src="images/8.jpg"></li>

<li><img src="images/9.jpg"></li>

<li><img src="images/10.jpg"></li>

</ul>

<p id="total">Vous avez sélectionné <em>0</em> photo(s)</p>

</main>

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

</body>

</html>

<style>

body

{

margin: 0 auto;

width: 90%;

}

em

{

color: red;

font-weight: bold;

}

.photo-list

{

/\* structure \*/

padding-left: 0;

width: 800px;

/\* presentation \*/

list-style-type: none;

}

.photo-list li

{

/\* structure \*/

display: inline-block;

margin: 1em;

padding: 0.5em;

/\* presentation \*/

border-radius: 8px;

transition: background-color 0.5s;

}

.photo-list li:hover

{

background-color: silver;

}

.photo-list li.selected

{

background-color: forestgreen;

}

</style>

<script>

var photos = document.querySelectorAll('.photo-list li');

var total = document.querySelector('#total em');

function onClickSelected() {

this.classList.toggle('selected');

var selectedPhotos = document.querySelectorAll('.photo-list li.selected');

console.log(selectedPhotos.length);

total.textContent = selectedPhotos.length;

}

for (var i = 0; i < photos.length; i++) {

photos[i].addEventListener('click', onClickSelected);

}

</script>

Corrigé carré vert :

<body>

<h1> Déplacer un carré</h1>

<div style="width: 300px; height: 300px; border: 1px solid black">

<div id="square">

</div>

</div>

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

</body>

</html>

<script>

var square = document.getElementById("square");

var top\_margin = 0;

var left\_margin = 0;

const ARROW\_LEFT = 37;

const ARROW\_UP = 38;

const ARROW\_RIGHT = 39;

const ARROW\_DOWN = 40;

function move(event)

{

switch(event.keyCode)

{

case ARROW\_LEFT :

if(left\_margin >= 10) {

left\_margin-=10;

square.style.left = left\_margin +"px";

}

break;

case ARROW\_RIGHT :

if(left\_margin <= 240) {

left\_margin+=10;

square.style.left = left\_margin +"px";

}

break;

case ARROW\_UP :

if(top\_margin >= 10) {

top\_margin-=10;

square.style.top = top\_margin +"px";

}

break;

case ARROW\_DOWN :

if(top\_margin <= 240) {

top\_margin+=10;

square.style.top = top\_margin +"px";

}

break;

}

}

document.addEventListener("keydown",move)

</script>

**Localstorage**

Web Storage est une solution adaptée aux besoins actuels de stockage de données variées, dans le navigateur. C'est aussi une technique plus puissante que les cookies, qui sont limités en taille (quelques Ko contre plusieurs Mo pour Web Storage) et qui engendrent un trafic HTTP supplémentaire pour chaque requête (que ce soit pour demander la page web, une image, une feuille de styles, un fichier javascript, etc).

La méthode **JSON.stringify()** convertit une valeur JavaScript en chaîne JSON.

(transforme en string)

console.log(JSON.stringify({ x: 5, y: 6 }));

// expected output: "{"x":5,"y":6}"

La méthode **JSON.parse() parse** une chaîne de caractères JSON et construit la valeur JavaScript ou l'objet décrit par cette chaîne.

(transforme en array)

var json = '{"result":true, "count":42}';

obj = JSON.parse(json);

console.log(obj.count);

// expected output: 42

La méthode **setItem()**de l'interface [Storage](https://developer.mozilla.org/fr/docs/Web/API/Storage), lorsque lui sont passées le duo clé-valeur, les ajoute à l'emplacement de stockage, sinon elle met à jour la valeur si la clé existe déjà

window.localstorage.setItem(key, tabstring) (on enregistre des données)

function remplissageStockage() {

localStorage.setItem('bgcolor', 'red');

localStorage.setItem('font', 'Helvetica');

localStorage.setItem('image', 'myCat.png');

}

window.localstorage.getItem(key) ; (on récupère des données)

tab. JSON.parse(x) ; (pour passer de format string à tableau)

JSON est un format d’enregistrement de données adapté à javascript

**JQUERY**

var couleur = sessionStorage.getItem("couleur");

$('#para1').on('click', hello);

>>équivalent en natif >>

var para1 = document.getElementById('para1');

para1.addEventListener('click', hello);

$('.para').on('click', alert);

>>équivalent en natif >>

var para = document.querySelector('.para');

for (var i = 0; i < para.length; i++) {

para[i].addEventListener('click', alert);

}

$('#but').on('click', alertButton);

function alertButton() {

event.preventDefault(); (à mettre dès qu’on met un élément sur un bouton)

alert($('#inp1').val());

}

>>équivalent en natif >>

var inp1 = document.getElementById('inp1');

inp1.value (pour récupérer la valeur de value)

Pour écrire mais sans que le texte sans supprimer le texte antérieur.

affiche.append("<p>hello</p>") // natif

$('#affichage').append('<p>hello</p>'); // jQuery

$('input[name=what]:checked').val();>>>> récupérer la valeur d’un input

**DATASET**

Lire les valeurs de ces attributs avec du [JavaScript](https://developer.mozilla.org/fr/docs/Web/JavaScript) est également très simple. Vous pourriez utiliser [getAttribute()](https://developer.mozilla.org/fr/docs/Web/API/Element/getAttribute) avec leur nom HTML complet pour les lire, mais le standard les définit d'une manière plus simple : un [DOMStringMap](https://developer.mozilla.org/fr/docs/Web/API/DOMStringMap) peut être lu via une propriété [dataset](https://developer.mozilla.org/fr/docs/Web/API/HTMLElement/dataset).

Pour obtenir un attribut data avec l'objet dataset, repérez la propriété avec la partie du nom de l'attribut qui suit le préfixe data- (notez que les tirets sont convertis en [camelCase](https://fr.wikipedia.org/wiki/CamelCase)).

var article = document.getElementById('voitureelectrique');

article.dataset.columns // "3"

article.dataset.indexNumber // "12314"

article.dataset.parent // "voitures"

Chaque propriété est une chaîne et peut être en lecture et écriture. Dans le cas ci-dessus passer le paramètre article.dataset.columns = 5 mettrait l'attribut à "5".

**AJAX**

$.getJSON('mon url d\' api', myFunctionCallBack); (syntaxe raccourcie)

$.ajax({ (syntaxe longue)

url: 'mon url d\' api',

type: 'GET',

data: {

"countries": "FR",

},

dataType: 'json',

headers: {

"X-API-Key": "nce8u3Rq5yNq0jL9FjpmxZ8jWCzv9xvw",

}

})

.done( myFunctionCallBack )

.fail( console.log('la requête a échoué') );

window.location;

Renvoie un objet Location contenant des informations concernant l'URL actuelle du document et fournit des méthodes pour modifier cette URL. Cette propriété peut être utilisée pour charger une autre page.

## Syntaxe

Section

var adresseActuelle = window.location;

window.location = nouvelleAdresse;

où

* adresseActuelle est un objet de type Location, fournissant des informations à propos de l'URL courante et des méthodes pour la modifier. Ses propriétés et méthodes sont décrites ci-dessous.
* nouvelleAdresse est un objet Location ou une chaîne de caractères indiquant l'URL à charger.

Exemple pour récupérer un Url :

function extractUrlParams () {

var t = location.search.substring(1).split('&');

var f = [];

for (var i=0; i<t.length; i++) {

var x = t[ i ].split('=');

f[x[0]]=x[1];

}

return f;

}

var params = extractUrlParams();

console.log(params);

La méthode substring()

La méthode substring() retourne une sous-chaîne de la chaîne courante, entre un indice de début et un indice de fin.

var str = 'Mozilla';

console.log(str.substring(1, 3));

// expected output: "oz"

**La géolocalisation :**

function position(response){

console.log(response);

}

if(navigator.geolocation) {

console.log('il y a la géoloc');

navigator.geolocation.getCurrentPosition(position);

} else {

console.log('Pas de geoloc');

}

**Les objets en JAVASCRIPT :**

**LA SYNTAXE :**

**POUR ENREGISTER UN OBJET**

var obj = {

propriete\_1: valeur\_1,

propriete\_2: valeur\_2,

propriete\_3: valeur\_3

}

Exemple :

var maHonda = {

couleur: "rouge",

roue: 4

}

Pour afficher « rouge »: console.log (maHonde.couleur) ;

POUR METTRE UN OBJET DANS UN AUTRE OBJET

var maHonda = {

couleur: "rouge",

roue: 4,

moteur: {

cylindres: 6,

taille: 2.2

}

}

Pour afficher « 6»: console.log (maHonda.moteur.cylindres) ;

LES FONCTIONS CONSTRUCTEURS

Elles permettent d’automatiser la création d’objet sur un même modèle dit « constructeur »( Par convention , on met une majuscule quand on crée un constructeur).

EXEMPLE :

var Gadjo = function(name, caracteristique) {

this.nom = name;

this.signedistinctif = caracteristique;

}

A partir de cette fonction je vais, sur le même modèle, créer deux objects distincts.

var Ledoc = new Gadjo("Gyneco", "Défonceman");

var Trouduc = new Gadjo("Denis", « Tête de con »);

LES FONCTIONS DANS LES FONCTIONS CONSTRUCTEURS

Deux méthodes :

- soit intégrer la fonction directement dans la fonction constructeur,

- soit via l’outil « prototype ».

> EN INTEGRANT LA FONCTION DIRECTEMENT DANS LA FONCTION CONSTRUCTEUR CA DONNE :

var Gadjo = function(name, caracteristique) {

this.nom = name;

this.signedistinctif = caracteristique;

this.generermessage = function(){

alert(this.nom + " a comme signe distinctif d'être un " + this.signedistinctif)

};

}

Pour appeler la fonction on fait :

var Ledoc = new Gadjo("Gyneco", "Défonceman");

puis :

Ledoc.generermessage() ;

> EN INTEGRANT LA FONCTION VIA L’OUTIL PROTOTYPE CA DONNE

var Perso = function(name, hp, attack, defense, magie) {

this.name = name;

this.hp = hp;

this.attack = attack;

this.defense = defense;

this.magie = magie;

}

Perso.prototype.attaquer = function(perso) {

var degats = this.attack - perso.defense

console.log(this.name +' Attaque, il enlève '+ degats + ' hp a '+ perso.name )

perso.hp -= degats

console.log(perso.name +' a '+ perso.hp+ ' hp' );

}

***PHP***

Pour les variables : on met un dollar devant $

Beaucoup de similarité avec Js

Pas d’objet mais des tableaux associatifs

En php 3 fonctions natives (dites « magiques ») permettant de gérer des tableaux associatifs :

exemple

$var = if(!empty($\_POST)){

$motatraduire = $\_POST['word'];

$languechoisie = $\_POST['direction'];

}'Paul';

$tab = [

'name' => 'Gérard',

'age' => '34',

'enfants'=> 8

];

il y array\_key\_exists (pour savoir si y a colonne dans un tableau (renvoi true ou false)

array\_search(8, $tab); // renvoie la clef

et in\_array('34', $tab); // renvoie true ou false

include permet d’intégre du html dans une page : (exemple include '05-exercice-traducteur.phtml';)

p.html > il y a du php mais on va gérer le html (les élements à afficher)

Pour intégrer une variable php dans du html > <?= $var ?>

for each > est une bloucle php permettant de parcourir un tableau

exemple:

foreach $array in $arrays (synthaxe pas bonne)

Methodes get/post :

permettent d’envoyer les données d’un formulaire vers une page php

var\_dump : l’équivalent du console log en php

A la différence de post, get affiche les données dans l’url

**Fopen**

**Liste des modes possibles pour la fonction fopen() en utilisant le paramètre mode**

| mode | Description |
| --- | --- |
| *'r'* | Ouvre en lecture seule, et place le pointeur de fichier au début du fichier. |
| *'r+'* | Ouvre en lecture et écriture, et place le pointeur de fichier au début du fichier. |
| *'w'* | Ouvre en écriture seule ; place le pointeur de fichier au début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas, on tente de le créer. |
| *'w+'* | Ouvre en lecture et écriture ; place le pointeur de fichier au début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas, on tente de le créer. |
| *'a'* | Ouvre en écriture seule ; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, on tente de le créer. Dans ce mode, la fonction [fseek()](http://php.net/manual/fr/function.fseek.php) n'a aucun effet, les écritures surviennent toujours. |
| *'a+'* | Ouvre en lecture et écriture ; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, on tente de le créer. Dans ce mode, la fonction [fseek()](http://php.net/manual/fr/function.fseek.php) n'affecte que la position de lecture, les écritures surviennent toujours. |
| *'x'* | Crée et ouvre le fichier en écriture seulement ; place le pointeur de fichier au début du fichier. Si le fichier existe déjà, **fopen()** va échouer, en retournant **FALSE** et en générant une erreur de niveau **E\_WARNING**. Si le fichier n'existe pas, **fopen()** tente de le créer. Ce mode est l'équivalent des options *O\_EXCL|O\_CREAT* pour l'appel système *open(2)* sous-jacent. |
| *'x+'* | Crée et ouvre le fichier pour lecture et écriture; le comportement est le même que pour *'x'*. |
| *'c'* | Ouvre le fichier pour écriture seulement. Si le fichier n'existe pas, il sera crée, s'il existe, il n'est pas tronqué (contrairement à *'w'*) et l'appel à la fonction n'échoue pas (comme dans le cas de *'x'*). Le pointeur du fichier est positionné au début. Ce mode peut être utile pour obtenir un verrou (voyez [flock()](http://php.net/manual/fr/function.flock.php)) avant de tenter de modifier le fichier, utiliser *'w'* pourrait tronquer le fichier avant d'obtenir le verrou (vous pouvez toujours tronquer grâce à [ftruncate()](http://php.net/manual/fr/function.ftruncate.php)). |
| *'c+'* | Ouvre le fichier pour lecture et écriture, le comportement est le même que pour le mode *'c'*. |
| *'e'* | Défini l'indicateur close-on-exec sur le descripteur de fichier ouvert. Disponible uniquement en PHP compilé sur les systèmes conforme POSIX.1-2008. |

Pour écrire dans un fichier :

$file = fopen(file.txt, « r ») ;

$task = fgetscsv($file) > fgetcsv permet d’agir sur des fichiers format csv

fclose($file) ;

fgets($file)

fgetsc…

fputscsv($donnée)

## *foreach***[¶](http://php.net/manual/fr/control-structures.foreach.php" \l "control-structures.foreach)**

(PHP 4, PHP 5, PHP 7)

La structure de langage *foreach* fournit une façon simple de parcourir des tableaux. *foreach* ne fonctionne que pour les tableaux et les objets, et émettra une erreur si vous tentez de l'utiliser sur une variable de type différent ou une variable non initialisée. Il existe deux syntaxes :

foreach (array\_expression as $value){

//commandes

}

foreach (array\_expression as $key => $value){

//commandes

}

La première forme passe en revue le tableau *array\_expression*. À chaque itération, la valeur de l'élément courant est assignée à *$value* et le pointeur interne de tableau est avancé d'un élément (ce qui fait qu'à la prochaine itération, on accédera à l'élément suivant).

La seconde forme assignera en plus la clé de l'élément courant à la variable *$key* à chaque itération.

***MYSQL***

Exemple de requête de base :

/\* La liste des bureaux (adresse et ville) triés par pays décroissant puis par état \*/

SELECT addressLine1, addressLine2, city, country, state

FROM offices

ORDER BY country DESC, state

/\* RESULTAT ==> 7 lignes / 100 Market Street \*/

/\* La liste des avions (code et nom) triés par vendeur et par quantité en stock décroissants \*/

SELECT productCode, productName

FROM products

WHERE productLine = 'Planes'

ORDER BY productVendor DESC, quantityInStock DESC

/\* RESULTAT ==> 12 lignes / 1900s Vintage Tri-Plane \*/

/\* La liste des produits (code, nom, échelle et quantité) qui ont une échelle soit de 1:10, soit de 1:18 triés par quantité en stock décroissante \*/

SELECT productCode, productName, productScale, quantityInStock

FROM products

WHERE productScale IN ('1:10', '1:18') /\* équivalent à WHERE productScale = '1:10' OR productScale = '1:18' \*/

ORDER BY quantityInStock DESC

/\* RESULTAT ==> 48 lignes / 1995 Honda Civic \*/

/\* La liste des produits (nom, vendeur et prix de vente) qui sont vendus au moins 132$ triés par nom du produit \*/

SELECT productName, productVendor, MSRP

FROM products

WHERE MSRP >= 132

ORDER BY productName

/\* RESULTAT ==> 24 lignes / 1903 Ford Model A \*/

/\* La liste des produits (code, nom et prix d'achat) des produits achetés au moins 60$ au plus 90$ triés par prix d'achat \*/

SELECT productCode, productName, buyPrice

FROM products

WHERE buyPrice BETWEEN 60 AND 90 /\* équivalent à WHERE buyPrice >= 60 AND buyPrice <= 90 \*/

ORDER BY buyPrice

/\* RESULTAT ==> 34 lignes / 1937 Lincoln Berline \*/

/\* La liste des motos (nom, vendeur, quantité et marge) triés par marge décroissante \*/

SELECT productName, productVendor, quantityInStock, (MSRP - buyPrice) AS margin

FROM products

WHERE productLine = 'Motorcycles'

ORDER BY margin DESC

/\* RESULTAT ==> 13 lignes / 2003 Harley-Davidson Eagle Drag Bike \*/

/\* La liste des commandes (numéro, date de commande, date d'expédition, écart en jours entre les deux dates et statut) qui sont en cours de traitement ou qui ont été expédiées et ont un écart de plus de 10j triés par écart décroissant puis par date de commande \*/

SELECT orderNumber, orderDate, shippedDate, (shippedDate - orderDate) AS processTime, status

FROM orders

WHERE status = 'In Process' OR (status = 'Shipped' AND (shippedDate - orderDate) > 10)

ORDER BY processTime DESC, orderDate

/\* RESULTAT ==> 33 lignes / 10165 \*/

/\*La liste des produits (nom et valeur du stock à la vente) des années 1960 \*/

SELECT productName, (quantityInStock \* MSRP) AS stockValue

FROM products

WHERE productName LIKE '196%'

/\* RESULTAT ==> 16 lignes / 1969 Harley Davidson Ultimate Chopper \*/

/\*Le GrROUP BY (classement des infos par groupe) –-- AVG (calcule la moyenne)\*/

SELECT productVendor, AVG(MSRP) AS averagePrice

FROM products

GROUP BY productVendor

ORDER BY averagePrice DESC

/\*Le COUNT (équivalent du .length)\*/

En SQL, la fonction d’agrégation COUNT() permet de compter le nombre d’enregistrement dans une table. Connaître le nombre de lignes dans une table est très pratique dans de nombreux stockValue cas, par exemple pour savoir combien d’utilisateurs sont présents dans une table ou pour connaître le nombre de commentaires sur un article.

SELECT productLine, COUNT(productCode)

FROM products

GROUP BY productLine

/\*Le SUM \*/

SELECT productLine, SUM(quantityInStock) AS totalStock, SUM(quantityInStock \* MSRP) AS totalStockValue

FROM products

WHERE MSRP > 100

GROUP BY productLine

ORDER BY totalStockValue

/\*Le MAX \*/

SELECT productVendor, MAX(quantityInStock) AS maxInStock

FROM products

GROUP BY productVendor

ORDER BY productVendor

/\*Le HAVING (c’est comme WHERE)\*/

SELECT customerNumber, SUM(amount) AS totalCredit

FROM payments

WHERE paymentDate BETWEEN '2004-01-01' AND '2004-12-31'

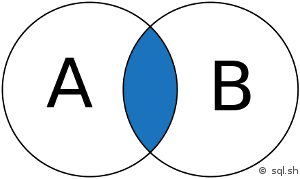
GROUP BY customerNumber

HAVING totalCredit > 20000

ORDER BY totalCredit DESC

/\*Le INNERJOIN\*/

Dans le langage SQL la commande INNER JOIN, aussi appelée EQUIJOIN, est un type de jointures très communes pour lier plusieurs tables entre-elles. Cette commande retourne les enregistrements lorsqu’il y a au moins une ligne dans chaque colonne qui correspond à la condition.



SELECT lastName, firstName, jobTitle, addressLine1, addressLine2, city

FROM employees

INNER JOIN offices ON offices.officeCode = employees.officeCode

/\* La liste des clients français ou américains (nom du client, nom, prénom du contact et pays) et de leur commercial dédié (nom et prénom) triés par nom et prénom du contact \*/

var\_dump($requeteProduct);

SELECT customerName, contactLastName, contactFirstName, country, lastName, firstName

FROM customers

INNER JOIN employees ON employees.employeeNumber = customers.salesRepEmployeeNumber

WHERE country IN ('France', 'USA')

ORDER BY contactLastName, contactFirstName

/\* La liste des lignes de commande (numéro de commande, code, nom et ligne de produit) et la remise appliquée aux voitures ou motos commandées triées par numéro de commande puis par remise décroissante \*/

SELECT orderNumber, orderdetails.productCode, productName, productLine, (MSRP - priceEach) AS discount

FROM orderdetails

INNER JOIN products ON products.productCode = orderdetails.productCode

WHERE productLine IN ('Classic Cars', 'Vintage Cars', 'Motorcycles')

ORDER BY orderNumber, discount DESC

/\* Le total des paiements effectués de chaque client (numéro, nom et pays) américain, allemand ou français de plus de 50000$ trié par pays puis par total des paiements décroissant \*/

SELECT customers.customerNumber, customerName, country, SUM(amount) AS totalPayment

FROM customers

INNER JOIN payments ON payments.customerNumber = customers.customerNumber

WHERE country IN ('France', 'Germany', 'USA')

GROUP BY customers.customerNumber, customerName, country

HAVING totalPayment > 50000

ORDER BY country, totalPayment DESC

/\* RESULTAT ==> 38 lignes / 146 / 130305.35 \*/

/\* Le montant total de chaque commande (numéro et date) des clients New-Yorkais (nom) trié par nom du client puis par date de commande \*/

SELECT customerName, orders.orderNumber, orderDate, SUM(quantityOrdered \* priceEach) AS totalAmount

FROM customers

INNER JOIN orders ON orders.customerNumber = customers.customerNumber

INNER JOIN orderdetails ON orders.orderNumber = orderdetails.orderNumber

WHERE city = 'NYC'

GROUP BY city, customerName, orderNumber, orderDate

ORDER BY customerName, orderDate

/\* RESULTAT ==> 16 lignes / Classic Legends / 10115 / 21665.98 \*/

INSERT INTO table ( ) VALUES ('valeur 1', 'valeur 2', ...)

INSERT INTO office (city, phone, addressLine1 )VALUES ('Paris', '0651203265', 'rue de la republique')

UPDATE Customers

SET contactName = 'Alfred Schmidt'

WHERE customerID = 1;

DELETE

DELETE FROM `customer`

WHERE `customerNumber` = 103

DELETE \* FROM \*

Se connecter à la base de données via PHP

$pdo = new PDO('mysql:host=localhost;dbname=classicmodels', 'root', 'troiswa');

$pdo->exec('SET NAMES UTF8');

$query = $pdo->prepare

(

'SELECT \* FROM customers'

);

$query->execute();

$customers = $query->fetchAll(PDO::FETCH\_ASSOC);

fetchAll > :fetchAll — Retourne un tableau contenant toutes les lignes du jeu d'enregistrements

:fetch — Récupère la ligne suivante d'un jeu de résultats PDO

Pour accélérer l’insertion d’une page html, on peut utiliser un layout (c’est à dire une page insérant (via include) une page insérant automatiquement le header et le footer.

**Htmlspecialchars :**

htmlspecialchars**—**Convertit les caractères spéciaux en entités HTML

Permet d’éviter le piratage

header('Location: index.php');

exit();

**Crypter un mot de passe (en php)**

<?php

// application/lib.php

function hashPassword($password)

{

$salt = '$2y$11$'.substr(bin2hex(openssl\_random\_pseudo\_bytes(32)), 0, 22);

// Voir la documentation de crypt() : http://devdocs.io/php/function.crypt

return crypt($password, $salt);

}

function verifyPassword($password, $hashedPassword)

{

// Si le mot de passe en clair est le même que la version hachée alors renvoie true.

return crypt($password, $hashedPassword) == $hashedPassword;

}

?>

Attention ça ne marche qu’avec des VARCHAR d’au moins 120 caractères