***JavaScript :***

On peut intégrer JS de 4 manières différentes :

**Inline** (avec une balise <script>

**Internal** (dans le head)

**External** (dans un fichier externe)

<script src="script.js"> </script> script.js est le document (comme style.css) et se rentre tel quel dans VS CODE.

**Remote** (un fichier extérieur d’un autre serveur qu’on appelle).

L’avantage de passer par un fichier externe (external ou remote) est qu’il reste dans le cache. En plus, on peut le partager entre plusieurs pages.

3 façons d’intégrer le résultat du JS dans une page :

La première dans le head:

**<script>**

**document.write(« hello.world. »)**

**</script>**

le browser va afficher ce qu’il y a entre guillemets qui sont dans les parenthèses. Même si c’est dans le head, ça va apparaître dans le body et être affiché avant ce qu’il y a dans le body. On peut aussi mettre la balise dans le body. Cette méthode est bien pour essayer ou tester un programme.

**<Script >**

**Alert(« hello.world »)**

**</script>**

Permet de faire apparaître une fenêtre pop-up avec le message.

***En général, on va mettre le lien de la page du JS juste avant la fin du body (avant le </body> ) pour que le contenu soit généré avant le JS afin qu’il puisse à coup sûr tout interpréter.***

**Document.getElementById(« nomdelID »)** permet de cibler un ID pour modifier son contenu.

<div id="content">

<div id="test"> Lorem ipsum dolor, sit amet consectetur adipisicing elit. Omnis reiciendis hic voluptates

quas aspernatur harum? Consequatur eligendi iste veritatis at quos. Labore eius quidem ipsum itaque

libero! Unde, cupiditate molestiae.</div>

<div class="test"></div>

<script>

alert("stop, regarde ici!");

document.getElementById("test").innerHTML = "J'aime les frites et les merguez"

</script>

</div>

Ici, le lorem ne s’affiche pas à la génération de la page.

Quand on lance la page, ***l’alerte*** apparaît. Quand on clique sur OK, c’est le message du **document.getElementById** qui apparaît à la place.

**.innerHTML** permet de modifier ce qu’il y a dans le contenu de la balise. On peut le remplacer, le compléter…

**.innerHTML+=** va ajouter le contenu du .innerHTML à la div existante et pas le remplacer.

**.style= « color :red ; font-size :.. »** pour modifier le style comme CSS.

Les variables :

***Elles ne doivent ni commencer par un chiffre ni par un symbole. Elles ne doivent pas contenir d’espace.***

***Elles doivent toujours commencer par une lettre ou un underscore ( \_ )***

Les minuscules et majuscules sont importantes. Il ne faut pas se tromper.

On peut déclarer a=  3  ou var a=3. Pour un chiffre on ne met pas de guillemets, on en met pour les mots.

<script>

alert("stop, regarde ici!");

a=5;

b=7;

c=a+b;

document.getElementById("test").innerHTML =c;

</script>

Après l’alerte, le viewport va afficher 12.

<script>

alert("stop, regarde ici!");

a=5;

b=7;

c=a+b;

document.getElementById("test").innerHTML ="5+7="+c;

</script>

La ça va afficher ce qu’il y a entre guillemets tel quel et le c qui équivaut à 12. Donc ***5+7=12.***  Le + avant le C ne sera pas affiché, il signifie juste qu’il doit afficher la var C en plus de ce qu’il y a entre les guillemets.

document.getElementById("test").innerHTML ="5+7="+(a+b);

Donnera la même chose.

Pour commenter sous JS : **// commentaire //** pour une ligne ou **/\* commentaire \*/** pour un bloc.

**++ sera toujours une incrémentation de 1 et -- toujours une décrémentation de 1.**

**a=a+1** doit se comprendre a(valeur) signifie cette valeur+1. Ce n’est pas une équation mathématique mais une incrémentation. On dit aussi **a++** .

**a=a-1** se dit aussi **a-- .** C’est une décrémentation de 1.

A++ dans un calcul va utiliser la valeur de a (1) et seulement après le calcul dire a= 2.

++a dans un calcul va faire l’incrémentation avant la fin du calcul.

<script>

a=1;

b=5+(a++);

c=5+(++a);

document.getElementById("test").innerHTML =c;

</script>

Ici b=6 et c=8.

a=5;

b=a++ + a++;

alert(a);

c=++a + ++a;

alert(a);

document.getElementById("test").innerHTML =c;

A la sortie de b a=7.

Donc b+11. (5 + 6)

A la sortie de c a=9 donc c=17. (8+9).

On peut simplifier la notation :

A=a+3 🡺 a+=3.

A=a/2 🡺 a/=2.

A=a-5 🡺 a-=5.

Donc pour :**.**

<script>

a=6;

document.getElementById("test").innerHTML =a/=2;

</script>

**A=3** au sortir du calcul.

Quelques notions à prendre en compte :

N= nombre entier (1,2…) positif ou négatif

R= nombre flottants ( 1.5, 2.56,… les nombres a virgule) positif ou négatif.

***String***= chaîne de caractères.

***Char***= un caractère

Boolear : true ou false.

Les boolears permettent de montrer si un truc est vrai ou faux :

**>**Supérieur à

**<** inférieur à

**==** égal à (il faut mettre deux égal car sinon c’est une incrémentation de variable)

**>=** est supérieur ou égal à

**<=** est inférieur ou égal à

**!=** est différent de

<script>

a=6;

document.getElementById("test").innerHTML= a!=8 //true//

</script>

Pour comparer des variables et faire des tests logiques :

**&& permet de vérifier que la variable est entre deux valeurs :**

<script>

a=6;

document.getElementById("test").innerHTML= a<7 && a>4; //true//

</script>

Si les deux sont vrai : true

Si les deux sont faux : false

Si seulement un des deux est vrai : false.

*Ici, tout doit être vrai pour avoir un résultat positif*

**| = (altgr + -) pour avoir la barre**

**|| permet de vérifier si une valeur fait partie d’une ou l’autre des catégories.**

<script>

a=6;

document.getElementById("test").innerHTML= a<3 || a>47;

//false//

</script>

si les deux sont vraies : true

si une seule des deux est vraie : true.

Si tout est faux : false.

*Ici, une seule affirmation doit être vraie vu que c’est un « ou ».*

**!A sert a inverser un résultat.**

Si a= true alors !a= false.

On peut par exemple dire :

***si résultat=a alors on va appliquer telle chose.***

***Si résultat= !a alors on appliquera autre chose.***

Faire un traitement selon une condition :

Il faut utiliser la fonction ***if :***

***If (x>5) {***

***a=3 ;***

***}***

***If (condition) alors***

***Faire un traitement***

***Sinon***

***Faire un autre traitement***

***If(condition){***

***Msg= « moyenne » ;***

***}***

***Else {***

***Msg= « pas la moyenne » ;***

***}***

On peut faire **else if {** si on veut enchaîner plusieurs conditions.

Le ternaire :

Le ternaire permet de prendre une condition et de faire comme un if/else, mais ça sert surtout pour les affectations :

X=(condition) ? val1 :val2 ;

***Msg=***

***(x>10) ? « bon » : « moyen » ;***

Donc si x est supérieur à 10 ça affichera bon, sinon ça affichera moyen. Contrairement aux if/else, on peut utiliser les deux points qu’une seule fois.

C’est utile pour les intégrer à des expressions, comme par exemple :

***Msg= « Bonjour  »+(s==m ? « monsieur » : « madame ») ;***

Pour dire si c’est un homme on dira bonjour monsieur, sinon bonjour madame.

<script>

var gender="m";

document.getElementById("test").innerHTML ="Bonjour "+((gender=="m")?"monsieur.":"madame.");

</script>

<script>

var note=5;

if(note<=5){

document.getElementById("test").innerHTML ="Médiocre";

}

else if(note>5 && note<=10){

document.getElementById("test").innerHTML ="Passable";

}

else if(note>10 && note<=15){

document.getElementById("test").innerHTML ="Bon";

}

else if(note>15 && note<=20){

document.getElementById("test").innerHTML ="Très Bon";

}

else if(note>20){

document.getElementById("test").innerHTML ="Dieu Vivant";

}

/\*L'appréciation dépendra de la valeur de la note entrée dans la variable. Ici, le résultat est égal à 5 donc l’appréciation est « Médiocre ». \*/

</script

<script>

var note = 17;

if (note < 0 || note > 20) {

document.getElementById("test").innerHTML = "Erreur de notation";

}

else if (note <= 5) {

document.getElementById("test").innerHTML = "Médiocre";

}

else if (note <= 10) {

document.getElementById("test").innerHTML = "Passable";

}

else if (note <= 15) {

document.getElementById("test").innerHTML = "Bon";

}

else {

document.getElementById("test").innerHTML = "Très Bon";

}

/\* Ici, nous avons une autre manière de vérifier la notation qui demandera moins de calculs.

la première solution est correcte, mais cette solution est plus optimisée dans les tests du

navigateur car il y a moins de calculs à faire. Sur des calculs de milliers de lignes, les

performances peuvent êtres grandement améliorées si on optimise les méthodes de vérifications. \*/

</script>

<script>

var degres=-2;

if(degres<0){

temperature ="gelées";

couleur = "color:white; font-size:45px; text-shadow: 2px 2px 1px black";

}

else if(degres>=0 && degres<=10){

temperature = "températures basses";

couleur = "color:green; font-size:45px";

}

else if(degres>10 && degres<=15){

temperature = "températures moyennes";

couleur = "color:orange; font-size:45px";

}

else if(degres>15 && degres<=30) {

temperature = "chaleurs";

couleur = "color:yellow; font-size:45px";

}

else{

temperature = "canicule";

couleur = "color:red; font-size:45px";

}

document.getElementById("test").innerHTML = temperature

document.getElementById("test").style = couleur;

</script>

On peut prendre la ligne document et l’écrire qu’une seule fois à la fin en donnant un nom à un élément, comme par exemple la température. Ici, température= « gelées » ; suffit pour afficher gelées sur le viewport car la phrase entière est notée après le if/else avec le innerHTML identifié comme étant le « température ».

Créer un bouton ou une div cliquable :

Dans la div que l’on veut rendre cliquable on a juste à ajouter ***onclick= « nomdevariable() »***

<div onclick="test3()">coucou</div>

Ensuite, dans le bas du body on met :

function test3() {

document.getElementById("test").innerHTML = "Test3";

document.getElementById("test").style = "background-color:blue; color:red;"

}

Ici, la div cliquable est dans le header.

La div qui sera modifiée est la div avec l’id « test » qui se trouve dans le content. Au clic, la div affichera « test3 » puisque l’**innerHTML** est modifié et aura un style appliqué.

Le plus important est de retenir qu’il faut nommer une fonction pour pouvoir la cibler, et qu’on peut modifier plusieurs id d’un coup avec un seul clic. Il suffit de cibler deux **.getElementById** différents.

Créer un mode jour/nuit ou avoir deux variables qui switch (par exemple deux CSS qui switch onclick) :

Dans le head, on vient créer le script :

<script>

nstyle=0;

</script>

On vient donner in ID à notre CSS :

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

Ensuite on créé cette variable :

function logo() {

if (nstyle==0){

document.getElementById("styleid").href="style2.css";

nstyle=1;

}

else{

document.getElementById("styleid").href="style.css";

nstyle=0;

}

}

L’idée est de donnée à la page un nom de variable et une valeur de variable (ici nstyle=0).

Ensuite, on créé un if/else qui dit que si nstyle=0 alors quand on joue sur l’ID styleid alors ça appliquera le style2.css.

Ensuite le style sera égal à 1. Donc au reclic, le nstyle qui sera égal = 1 lira le deuxième css et après l’avoir lu repassera à zéro.

Dans l’exemple, à chaque fois qu’on clique sur le logo, le CSS change.

Les switch :

Note=’c’ ;

**Switch(Note){**

**Case ‘a’ :**

**Document.write(« excellent travail ») ;**

**Break ;**

Case ‘b’ :

Document.write(« bon travail ») ;

Break ;

Case ‘c’ :

Document.write(« passable) » ;

Break ;

Case ‘d’ :

Document.write(« mauvais ») ;

Break ;

Default :

Document.write(« erreur de notation »)

Break ;

}

Le switch sert à faire du cas par cas. On définit les cas pour chaque valeur potentielle d’une variable.

Si une valeur n’est pas incluse dans les cas spécifiés, c’est le ***default*** qui sera affiché. Il ne faut pas oublier les ***break*** sinon le navigateur va afficher chaque valeur jusqu’à trouver un ***break***.

Si plusieurs résultats ont le même résultat, on peut les regrouper de telle sorte :

case 3:

case 4:

case 5:

case 6:

document.write("passable")

break;

on met juste case « valeur » : et on passe à la ligne d’après qui aura le même résultat. Le break regroupera toutes les valeurs qui auront le même résultat.

Les incrémentations :

Une incrémentation est un compteur. On va répéter une action jusqu’à avoir un résultat.

var vie=5;

while(vie>0){

//Jouer jusqu'à la mort du personnage

vie--;

on utilise le mot while. Ici, on peut jouer tant qu’on a des vies. A chaque mort, le vie--va enlever une vie. A zéro, ça sera fini.

var connexion="no";

while(connexion=="no"){

//Essayer de se connecter au serveur

connexion=verifier();

}

Ici, si on est pas connecté, ça va vérifier en permanence si on est co ou pas.

var compteur=1;

while(compteur<=12){

document.write("le compteur vaut"+ compteur+"<br>");

compteur++;

}

Ici, il va afficher « le compteur vaut 1 », passer une ligne et le compteur va augmenter donc il va afficher « le compteur vaut 2 », … jusqu’à 12 et ensuite il va s’arrêter.

Do while :

On va appliquer une chose avec do et ensuite le vérifier avec le while :

var vie=5;

do{

//jouer la partie jusqu'à la mort du personnage

vie--;

}

while(vie>0);

comme le ***do*** est avant, on est sûr que l’itération se fera une fois. Avec juste le ***while*** en premier, si la condition n’est pas remplie dès le départ la ligne de code ne s’appliquera pas.

<script>

var compteur=1;

do{

document.write("le compteur vaut"+compteur+ "<br>");

compteur++;

}

while(compteur<=5);

</script>

Ici, la phrase « le compteur vaut x » va s’afficher jusqu’à atteindre 5.

*Si on met une valeur supérieure à 5, la phrase va quand même s’afficher une fois avec le nombre supérieur. Avec juste un while, rien ne s’afficherait.*

For :

***For(initialisation ;condition ;itération){***

***//bloc de traitement***

***}***

Ici, on va dire la valeur est égale à tant, elle doit appliquer telle condition (par exemple ne pas être supérieure à tant) ; et va augmenter de 1 à chaque fois.

<script>

var i;

for(i=1;i<=5;i++) {

document.write("i est égal à"+i+("<br>"));

}

</script>

Ici, la valeur de la variable est incluse directement dans le code. Le navigateur va afficher « i est égal à x » de 1 jusqu’à 5.

Ici, comme pour le ***while/do***, la condition est testée avant d’exécuter le bloc.

<script>

var compteur;

for(compteur=0; compteur<13; compteur +=2) {

document.write("i est égal à"+compteur+("<br>"));

}

</script>

Les compteurs sont affichés avec une itération de 2 (0/2/4/…) jusqu’à 12 puisque 13 est impair et qu’on part de 0.

On aura tendance à utiliser le ***for*** quand on connait à l’avance le nombre d’itérations qu’on devra faire.

<script>

var compteur;

for(compteur=0; compteur<13; compteur +=1) {

document.write("i est égal à"+compteur+("<br> <br>"));

if(compteur==5){

break;

}

}

</script>

On peut sortir d’une boucle prématurément en donnant une condition et ***break.*** Ici, il affichera donc que de 1 à 5.

<script>

var compteur;

for(compteur=0; compteur<13; compteur +=1) {

if(compteur==5 || compteur==6 ){

continue;

}

document.write("i est égal à "+compteur+("<br>"));

}

</script>

Ici, si compteur est égal à 5 ou à 6 , le continue va revenir sur le début de la boucle et donc ne pas afficher le 5 et le 6. Si c’est 5 ou 6 le navigateur va remonter à for sans lire le ***document.write*** donc il va tout afficher sauf 5 et 6.

Si le compteur n’est pas égal à 5 ou 6 alors la boucle ne remontera pas et va ***continue***r comme le demande la commande.

Les modulos :

Les **modulos** permettent de déterminer une variable pour les multuples d’une valeur. Par exemple :

else if (compteur % 3 == 0) {

document.write("triplette <br> <br>");

}

Ici, si la variable compteur modulo 3 (la touche est % dans JS) =0 ça veut dire que la valeur de la variable compteur est un multiple de 3 puisque quand on la divise par 3 on obtient zéro.

Par exemple : 18= (3x6)+0.

Donc le document va marquer « triplette » dès que la variable sera un multiple de 3.

else if (compteur % 3 == 1) {

document.write("triplette + 1 <br> <br>");

}

Ici, si le ==1 signifie que le reste après la division est égal à 1.

Par exemple : 33= (3x8)+1. Donc ***« triplette +1 »*** s’affichera à chaque valeur qui est un multiple de 3 +1, ou dit autrement à chaque fois qu’une division par 3 donne un reste=1.

<script>

var compteur = 1;

while (compteur <= 36) {

if (compteur % 5 == 0 || compteur % 7 == 0) {

document.write("ne rien imprimer <br> <br>");

}

else if (compteur % 3 == 0) {

document.write("triplette <br> <br>");

}

else {

document.write(compteur + "<br> <br>");

}

if (compteur == 27) {

break;

}

compteur++;

}

</script>

Ici, à chaque multiple de 5 ou 7 ça affichera « ne rien imprimer »

A chaque multiple de 3 ça affichera « triplette »

Le 27 ne sera pas affiché.

Tous les autres chiffres entre 1 et 27seront affichés normalement puisqu’après 27 la fonction s’arrête.

Les function :

Les fonctions sont un bloc de traitements qu’on va regrouper sous un nom pour que tout s’applique en même temps.

function logo() {

if (nstyle == 0) {

document.getElementById("styleid").href = ("style.css");

nstyle = 1;

}

else {

document.getElementById("styleid").href = ("style2.css");

nstyle = 0;

}

Pour se servir d’une fonction, il faut la saisir où on veut de la manière suivante :

logo();

L’appel de la fonction appliquera toutes les modifications qui sont entrées dans le ***function nomdefonction()***

Par exemple, pour se servir du logo pour changer le css en cliquant dessus, on utilisera la fonction appelée logo :

<img src="logo.png" width="150px" height="100px" alt="" onclick="logo()">

Donc, si on modifie une fonction, tous les endroits où elle a été appelée seront modifiés en même temps.

<div id="test" style="text-align: center;">

<h2>Titre </h2>

<script>

function dostart(){

document.write("=======Start======="+"<br>");

}

function doend(){

document.write("=======End========"+"<br>");

}

function dobr(){

document.write("<br>");

}

var i;

for(var i=1;i<=7;i++){

dostart();

document.write(i+"<br>");

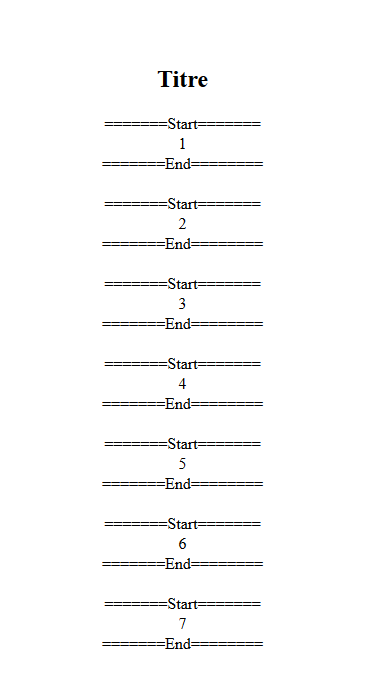
doend();

dobr();

}

</script>

</div>



Définir une fonction :

On peut définir une fonction. Une fonction utilisée normalement sera utilisée telle qu’elle dès qu’elle sera appelée. En mettant un mot ou une lettre dans les parenthèses, on pourra personnaliser :

<script>

function mafonction(x){

document.write("Hello "+x+" welcome! <br>");

}

</script>

<script>

mafonction("Jean");

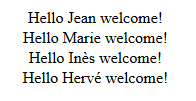
mafonction("Marie");

mafonction("Inès");

mafonction("Hervé");

</script>

Donnera :



Chaque prénom remplace le x dans la fonction.

On peut les cumuler. Il suffit de mettre des virgules entre les paramètres et bien sûr les recopier dans le bon ordre. :

<script>

function mafonction(x,y){

document.write("Hello "+x+" welcome! <br>");

document.write("numero "+ y+ "<br>")

}

</script>

<script>

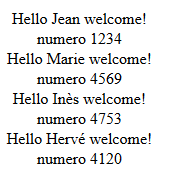
mafonction("Jean",1234);

mafonction("Marie", 4569);

mafonction("Inès", 4753);

mafonction("Hervé", 4120);

</script>



On peut imbriquer des fonctions dans des fonctions :

<script>

function doDW(msg){

document.write(msg);

}

function mafonction(x,y){

doDW("Hello "+x+" welcome! <br>");

doDW("numero "+y+ "<br>");

}

</script>

<script>

mafonction("Jean",1234);

mafonction("Marie", 4569);

mafonction("Inès", 4753);

mafonction("Hervé", 4120);

</script>

***Ce qui est entre les parenthèses devient le paramètre.***

Les fonctions qui retournent un résultat :

Certaines fonctions ont pour but de faire un calcul afin de calculer un résultat. Quand on appellera la fonction, uniquement le résultat final sera envoyé :

function somme(x, y, z) {

var som;

som = x + y + z;

return (som);

}

function moyenne(x, y, z) {

return ((x + y + z) / 3);

}

var m1;

m1 = somme(5, 6, 7);

document.write("la somme de (5,6,7) est " + m1 + "<br>");

m1 = somme(15, 17, 10);

document.write("La somme de (15,17,10) est " + m1 + "<br>");

var moy1;

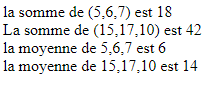
moy1 = moyenne(5, 6, 7);

document.write("la moyenne de 5,6,7 est " + moy1 + "<br>");

moy1=moyenne(15,17,10);

document.write("la moyenne de 15,17,10 est " + moy1 + "<br>");

Ici, uniquement ce qui est indiqué par le mot ***return*** sera envoyé.



On peut cumuler les return :

function salut(s) {

if (s == "m" || s == "M"){

return "Monsieur";

}

else if (s == "f" || s == "F"){

return "Madame";

}

else{

return "";

}

}

Par contre, si le premier return est validé alors rien de ce qu’il n’y a après ne sera lu, donc il faut faire attention au placement.

function salut(s) {

if (s == "m" || s == "M"){

return "Monsieur";

}

else if (s == "f" || s == "F"){

return "Madame";

}

else{

return "";

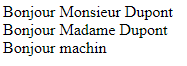
}

}

document.write("Bonjour "+salut("m")+" Dupont <br>");

document.write("Bonjour "+salut("f")+" Dupont <br>");

document.write("Bonjour "+salut()+" machin <br>");



On peut appeler une fonction dans un ***return*** :

function max(a,b){

return((a>b)? a:b);

}

function max3(a,b,c){

return((max(a,b)>c)? max(a,b):c);

}

var nombres;

nombres=max3(10,12,7);

document.write("la valeur la plus élevée est:"+nombres);

Ici, l’idée est d’afficher uniquement la valeur la plus élevée.



function max(a,b){

return((a>b)? a:b);

}

function max3(a,b,c){

return max(max(a,b),c);

}

Est aussi une solution. Ici, le max3 fait un ***return*** de la première fonction en la mettant elle-même en comparaison avec la lettre c. Vu que la fonction max a déjà calculé qui de a ou b est le plus grand avant, il peut comparer le résultat avec c.

function tableau(h,l){

document.write("<table>");

for(n=0;n<h;n++){

document.write("<tr>");

for(N=1;N<l;N++){

document.write("<td>"+N+"</td>");

}

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

}

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

}

tableau(5,6);

Cette fonction exemple permet de créer un tableau qui a des variables qui se modifient. Il n’est pas rigide et on peut modifier le nombre de colonnes et de lignes en modifiant les variables entre les parenthèses de la fonction appelée.

Refresh une page :

function rejouer(){

location.reload();

}

Créer un timer :

setTimeout(

function () {

document.getElementById("result").style.display = "none"

}, 2000);

La fonction set timeout englobe tout ce qu’elle modifie dans ses parenthèses. Le 2000 est le temps exprimé en millisecondes. 2000 équivaut à 2secondes.