***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>");

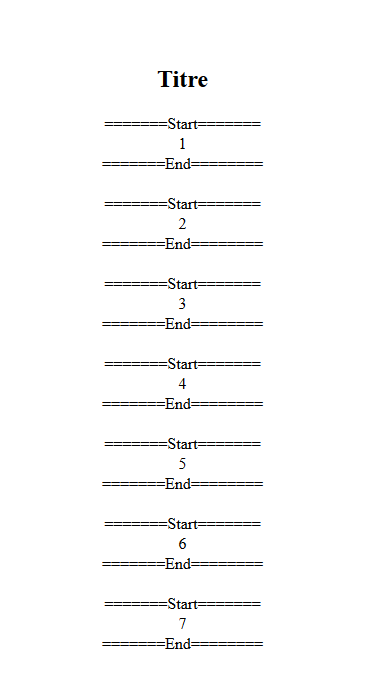
doend();

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");

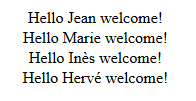
mafonction("Marie");

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>

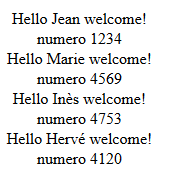
mafonction("Jean",1234);

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;

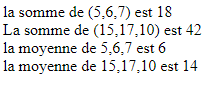
moy1 = moyenne(5, 6, 7);

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 "";

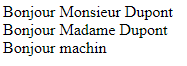
}

}

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.

Les boites de dialogue :

Il existe des boîtes de dialogues qui permettent de faire pas mal de choses :

alert("Message d'alerte");

//Indique un message d'alerte.

confirm("oui ou non");

//Demande confirmation de quelque chose (on répond toujours par true ou false sur le nav, donc oui ou non)

prompt("Entrez votre nom");

//Sert à entrer une valeur.

Les dates :

var d = new Date();

document.write("getDay:"+ d.getDay()+"<br>");// 0:dimanche...6:samedi

document.write("getdate"+d.getDate()+"<br>"); //retourne la date du jour. (ex: le 6)

document.write("getMonth:"+d.getMonth()+"<br>");//0=janvier... 11=décembre.

document.write("getFullYear:"+d.getFullYear()+"<br>");//Indique l'année en cours (ex:2018)

document.write("getYear:"+d.getFullYear()+"<br>");//A éviter, ça indiquera un chiffre mais depuis 1900. ex:01=1901

document.write("getHours:"+d.getHours()+"<br>");//Indique l'heure ex:13

document.write("getMinutes:"+d.getMinutes()+"<br>");//Indique la minute ex:43

document.write("getSeconds:"+d.getSeconds()+"<br>");//Indique la seconde

document.write("getTime:"+d.getTime()+"<br>");//Indique le temps passé depuis le 01/01/1970. (utile pour comparer).

document.write(d.toLocaleDateString()+"<br>");//Indique la date dans le format utilisé par le pays.

document.write(d.toLocaleTimeString()+"<br>");//Indique l'heure dans le format utilisé par le pays.

document.write(d.toLocaleString()+"<br>");//Convertis l'ensemble dans le format du pays.

document.write(d.toDateString()+"<br>");//Indique la date en format Américain.

document.write(d.toTimeString()+"<br>");//Indique l'heure en format Américain.

document.write(d.toString()+"<br>");//Convertis l'ensemble en format Américain.

setInterval(() => {

var d= new Date();

document.getElementsByClassName("heure")[0].innerHTML=(d.toLocaleTimeString());

}, 1000); //affiche l’heure en temps dynamique.

Les tableaux :

Il existe deux manières de créer un tableau (elles n’ont aucune différence):

var personnes = ['marc','paul','James'];

var personnes2 =new Array('marc','paul','James');

console.log(personnes);//Affiche le tableau personnes dans la console.

Pour afficher une donnée en particulier d’un tableau dans la console:

console.log(personnes[2]); affichera dans la console du F12

***Ici, on appellera la troisième valeur (donc James). 0 compte comme étant la première valeur.***

console.log(personnes.length);

Permet de savoir combien il y a de valeurs dans le tableau.

personnes[1]="Micheline";

On peut modifier une variable directement dans le tableau en tapant la variable du tableau, la place dans le tableau= la donnée du tableau à remplacer.

document.write(personnes[1]);

***Une variable de tableau peut être appelée directement.***

personnes.sort(); permet de trier un tableau (par ordre alphabétique ou de grandeur). Les majuscules sont prioritaires sur les minuscules. Par ordre on a les chiffres, après les majuscules et après les minuscules.

Si on trie comme ça, JS triera de manière non naturelle pour nous. Il va par exemple mettre tous les nombres commençant par 1 avant ceux commençant pas deux, ect. Pour les avoir dans l’ordre croissant il faut faire :

function compare(x, y) {

return x - y; // return y – x pour l’ordre décroissant.

}

//Ensuite :

document.write( "Tri par ordre de grandeurs: "+tab1.sort(compare));

Il existe deux types de variables :

**Les variables primitives** qui représentent qu’une seule valeur à la fois. On peut les comparer. ***On peut dire P1=P2. Si on modifie P2 par la suite, P1 ne sera pas modifiée.***

**Les variables objets** peuvent être par exemple un tableau ou un regroupement de plusieurs données. Elles sont propres à elles-mêmes et ne peuvent pas être comparées car elles ont des adresses différentes. Si on affecte un objet sur un autre ***(ex :T3=T2🡺 T3[0]=500 alors T2[0] vaudra aussi 500.)***

personnes.push("Alex");//ajoutera la donnée en fin de tableau.

On peut aussi mettre :

personnes[3]="Alex";//Va ajouter la donnée à la position 4 qui n'est pas encore occupée.

personnes[personnes.length]="Bob";//Ajoute automatiquement à la dernière place pour être sûr de ne rien remplacer.

personnes.pop();//retire la dernière valeur du tableau. C'est l'inverse de push.

var item = personnes.pop();//la donnée enlevée devient la variable item.

delete personnes[1];//supprime et une valeur et la remplace par un vide.

personnes.splice(1,1,"Hervé");//Le premier chiffre va dire à partir de où on se place. La deuxième combien de données on delete. On peut en ajouter une troisième pour ajouter des données à partir de l'endroit où on a delete. A partir du moment où on ajoute, on peut rajouter autant de données qu'on veut en séparant par une virgule.

var str = personnes.join(" $ "); permet d’ajouter ce qu’on met entre les parenthèses pour créer un string unique (d’où le nom de variable str.

personnes.reverse();affiche le tableau mais avec les données à l’envers.

var existe = personnes.includes("Hervé");

console.log(existe); Permet de vérifier si le tableau contient ou pas la donnée de l’includes. La valeur sera exprimée en true ou false.

var existe = personnes.includes("Hervé");

console.log(existe);

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

console.log(personnes[i]);

}

//Permet de tout extraire dans une boucle.

var people= personnes2.concat(personnes);

console.log(people); //concat permet d’ajouter un tableau à un autre tableau. Ici, personnes va s’ajouter au tableau personnes2. Les données du tableau entre parenthèses vont s’ajouter après celles du premier tableau.

Comparer des tableaux :

var res;

var tab1 = [19, 72, 33, 5, 66, 2, 14, 9, 10];

var tab2 = [19, 72, 33, 15, 16, 8, 17, 39, 50];

var tab3 = [19, 72, 33, 15, 16, 8, 39, 50];

var tab4 = [19, 72, 33, 5, 66, 2, 14, 9, 10];

res = fcompare(tab1, tab2);

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

res = fcompare(tab1, tab3);

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

res = fcompare(tab1, tab4);

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

function fcompare(param1, param2) {

if (param1.length != param2.length) {

return false;

}

for (var i = 0; i < param1.length; i++) {

if (param1[i] != param2[i]) {

return false;

}

}

return true;

}

//On compare d’abord la longueur. Si elle est différente alors le résultat sera forcément différent. Ensuite si les longueurs sont les mêmes on compare les données des tableaux une par une. Si une donnée diffère alors ça sera également faux. Si tout correspond (même nombre de données et données similaires) alors le résultat sera vrai.

Pour par exemple changer toutes les divs paires d’un tableau d’une couleur et les impaires d’une autre couleur :

var div = document.getElementsByTagName("div");

function color() {

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

if (i % 2 == 0) {

document.getElementsByTagName("div")[i].style = "color:red ;font-size:large;";

document.getElementsByTagName("div")[i].style.backgroundColor = "blue";

document.getElementsByTagName("div")[i].style.textAlign = "center";

}

else {

document.getElementsByTagName("div")[i].style = "color:yellow ;font-size:large;";

document.getElementsByTagName("div")[i].style.textAlign = "right";

}

}

}

color();

Afficher l’heure en dynamique :

function date() {

document.getElementsByTagName("div")[5].style.textAlign="center";

setInterval(() => {

var d = new Date();

document.getElementsByTagName("div")[5].innerHTML=(d.toLocaleTimeString());

}, 1000);

}

date();

Les objets :

***Il existe trois types d’objets :***

*Les premiers sont les objets universels : par exemple*

Math.

*Existe dans tout ce qui utilisera JS. Aussi bien le navigateur qu’une montre connectée…*

*Les seconds sont les objets hôtes. Certains objets sont existants dans certains domaines (par exemple le serveur) mais n’existera pas dans le navigateur ou la montre connectée.*

*Enfin les derniers sont ceux que l’on va créer nous-même.*

Créer un objet :

var personne=new Object();

ou :

var personne={}; Les accolades restent vides.

Y affecter des variables :

personne.nom="Alex";

personne.age="30";

personne.email="alex@gmail.com";

Ou :

personne={

nom:"Alex", //On met : et pas = et les éléments sont séparés par une ,

age:"30",

email:"alex@gmail.com"

} ;

Pour récupérer les données :

document.write(personne["nom"]); //Ou bien:

document.write(personne.nom); //cette version est de loin la plus utilisée car plus simple et logique.

On peut créer ou enlever une donnée à un objet n’importe quand :

personne.ville="paris";

delete personne.age;

Un objet créé comme ça est unique. Il faut en créer autant qu’il y a de personnes. Il est possible de créer une fonction qui va créer les objets à la volée :

function Employe(param1,param2,param3){

this.nom=param1;

this.age=param2;

this.email=param3;

}

var personne1= new Employe("Pierre","42","pierre@gmail.com");

var personne2=new Employe("Cyril", "27","cyril@gmail.com");

document.write("Nom: "+ personne1.nom+"<br>");

document.write("Nom: "+ personne1.age+"<br>");

document.write("Nom: "+ personne1.email+"<br> <br>");

document.write("Nom: "+ personne2.nom+"<br>");

document.write("Nom: "+ personne2.age+"<br>");

document.write("Nom: "+ personne2.email+"<br>");

On peut intégrer une fonction à un objet :

function Employe(param1,param2,param3){

this.nom=param1;

this.age=param2;

this.email=param3;

this.presentemoi=function(){ //ici presentemoi n'a pas de parenthèses car on définit ce mot comme étant une fonction.

document.write("Je suis "+this.nom+", J'ai "+this.age+" et mon email est: "+this.email);

}

}

var personne1= new Employe("Pierre","42","pierre@gmail.com");

personne1.presentemoi(); //Ici, il y en a car on appelle la fonction qu'on a créé.

On peut regrouper un traitement à une fonction :

this.setAge = function (param2) {

if (param2 < 0 || param2 > 110) {

this.age = 0;

alert("Erreur. L'âge indiqué n'est pas accepté.")

}

else {

this.age = param2

}

}

this.getAge=function(){

return(this.age);

}

Ici, par exemple on définit que si l’âge indiqué est inférieur à 0 ou supérieur 110 alors c’est une erreur, sinon l’âge sera appliqué.

Après il suffit de créer une autre fonction qui appellera l’âge.

Pour parcourir un objet, trouver ses membres et même afficher ses valeurs :

L’objet va être affiché par rapport à la classe qui le définit.

for (var variable in personne1) {

document.write("variable= " + variable + "---> valeur ="+ personne1[variable]+"<br>"); //Ici, même les fonctions dans la variable seront affichées. Ça affichera la variable qui définit l’objet qui donnent les résultats de cet objet en particulier.

}

On peut recréer la même fonction mais en plus complexe. CI-dessous, on va créer une liste de dix chiens qui seront rangés dans un tableau. L’idée est qu’un nouveau propriétaire aura peut-être envie de changer le nom de son chien. Si on le change, dans ce cas le changement sera pris en temps réel.

Il est toujours mieux de mettre this.name au lieu de param1 parce que le param1 fera référence à rien de concret alors que this.name si.

function chien(param1, param2, param3) {

this.nom = param1;

this.race = param2;

this.age = param3;

this.setName = function (param1) {

this.nom = param1;

alert("Le nom du chien a changé");

}

this.getName = function () {

return (param1);

}

this.annonce = function () {

return ("<strong>" + this.nom + ", </strong>Très gentil chien de race <strong>" + this.age + "</strong>. Il a <strong>" + this.race + "</strong> et cherche un gentil maître pour une nouvelle vie. <br> <br>");

}

}

var chien1 = new chien("rocky", "Husky", "3 ans");

var chien2 = new chien("Oslo", "Berger Belge", "6 ans");

var chien3 = new chien("Mike", "Rottweiller", "2 ans");

var chien3 = new chien("Schadée", "Rottweiller", "3 ans");

var chien4 = new chien("Oliver", "Dalmatien", "7 ans");

var chien5 = new chien("Pampa", "Berger Australien", "11 ans");

var chien6 = new chien("Hector", "Bulldog Français", "13 ans");

var chien7 = new chien("Iliana", "Caniche Royal", "1.5 an");

var chien8 = new chien("Lotus", "Labrador", "4 ans");

var chien9 = new chien("Zelda", "Cocker", "5 ans");

var chien10 = new chien("Vaïana", "Fox Terrier", "2.5 ans");

var chenil = [chien1, chien2, chien3, chien4, chien5, chien6, chien7, chien8, chien9, chien10];

chenil[0].setName("John");

document.write("<table> <br> <tr> <td> Nom: </td> <td> Race: </td> <td> Age: </td> <td> Description:</td> </tr>");

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

document.write('<tr> <td>' + chenil[i].nom + '</td>' + '<td>' + chenil[i].race + '</td>' + '<td>' + chenil[i].age + '</td>' + '<td>' + chenil[i].annonce() + " </tr>");

}

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

Ici, on mélange plusieurs fonctions pour définir quel joueur aura choisi quelle arme le temps qu’il s’agit d’un marteau, d’une épée ou d’un arc.:

function arme(param1, param2) {

this.name = param1;

if (param2 == "épée" || param2 == "marteau" || param2 == "arc") {

this.type = param2;

}

else {

alert("L'arme sélectionnée n'est pas valable");

}

this.getarme = function () {

return this.type;

}

}

function player(param1, param2, param3, arme1, arme2) {

this.pseudo = param1;

this.email = param3;

if (param2 == "Warrior" || param2 == "Mage" || param2 == "Ranger") {

this.class = param2;

}

else {

alert("La classe n'existe pas.");

}

this.arme = [arme1, arme2];

this.setclass = function (param2) {

if (this.class == "Warrior" || this.class == "Mage" || this.class == "Ranger") {

this.class = param2;

}

else {

alert("La classe n'existe pas.");

}

}

this.getclass = function () {

return this.class;

}

this.presentation = function () {

return ("Je suis " + this.pseudo + " " + this.class + " de niveau max et super classe! J'utilise " + this.arme[0].name + " mon " + this.arme[0].getarme() + " et " + this.arme[1].name + " mon " + this.arme[1].getarme() + " .Mon mail est: " + this.email + " <br> ")

}

}

var arsenal = [];

arsenal[0] = new arme("Excalibur", "épée");

arsenal[1] = new arme("Bifrost", "bâton");

arsenal[2] = new arme("Whisper", "dague");

arsenal[3] = new arme("Earthquake", "marteau");

arsenal[4] = new arme("Bloodrain", "arc");

var serveur = [];

serveur[0] = new player("Stein", "Warrior", "Alexis@gmail.com", arsenal[0], arsenal[3]);

serveur[1] = new player("Sophia", "Mage", "Julie@gmail.com", arsenal[1], arsenal[2]);

serveur[2] = new player("Snow", "Ranger", "Jonhattan@gmail.com", arsenal[4], arsenal[0]);

document.write("<ol>");

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

document.write("<li>" + serveur[i].presentation() + "</li>");

}

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

Faire un compte à rebours :

var s=8;

var rebours=setInterval(() => {

s--;

document.getElementById("rebours").innerHTML=("Veuillez attendre "+ s+" secondes, merci.");

if(s<=0){

clearInterval(rebours);

document.getElementById("rebours").innerHTML=("Merci d'avoir attendu.");

}

}, 1000);

Queryselector et queryselectorall :

document.querySelectorAll(".myclass")[5];

Queryselectorall permet de remplacer le getelementby et de retrouver ce qu’on veut avec ce qu’on a rentré sur CSS. Le ALL renvoie à toutes les balises contenant par exemple la classe ou toutes les balises tout court.

document.querySelector("#myID");

Renvoie uniquement à la première balise possédant la chose entre parenthèses. Si on met ça pour une div, on verra uniquement la première div du HTML.

Créer du contenu à ajouter directement via JS :

var objet= document.createElement("<div>"); // JS sait qu'il faudra refermer la div car c'est un objet.

objet.innerHTML="Coucou"; //On ajoute du contenu à la div qu'on vient de créer.

document.querySelector("body").insertBefore(3,objet); //va insérer ma div dans le body avant la 4ème div, donc en troisième position.

document.querySelectorAll("div")[3].appendChild(objet); // Ici, la div sera un enfant de la div4, et donc sera à l'intérieur.

Aller chercher un lien non défini à l’avance avec une base de données:

Si on a un lien non défini tel que :

<div id="container">

Lien vers une page <a href="#"> Contact</a> avec un formulaire pour laisser vos coordonnées.

</div>

Ici, le lien est indéfini pour permettre de travailler avec une base de données.

var x = document.getElementById("container");

var a = x.querySelectorAll("a")[0];

a.href="page/contact.html";

De cette manière, on peut par exemple rajouter une classe à une balise :

var a = document.querySelector("#note");

a.className="note\_importante";

On peut par exemple modifier un CSS en temps réel si on coche quelque chose grâce à cette technique.

Créer un contenu dynamique :

Ici, l’idée est de créer des nodes et de les imbriquer les uns aux autres pour avoir un résultat dynamique. Il suffira d’imbriquer dans l’ordre qu’on veut les éléments et nodes qu’on créé.

var x = document.getElementById("container");

var p1 = document.createElement("p");

p1.appendChild(document.createTextNode("La liste des grands joueurs:"));

//Premier node. On créé un paragraphe et on ajoute du contenu à l'intérieur dans la foulée.

var ul = document.createElement("ul"); //On créé un ul qui contiendra 4 li.

var li1 = document.createElement("li");

li1.appendChild(document.createTextNode("Zidane")); //Le premier LI.

var li2 = document.createElement("li");

li2.appendChild(document.createTextNode("Ronaldo")); //Le second.

var li3 = document.createElement("li");

li3.appendChild(document.createTextNode("Messi")); // Le troisième.

var li4 = document.createElement("li");

li4.appendChild(document.createTextNode("Barthez")); //Le dernier.

ul.appendChild(li1);

ul.appendChild(li2);

ul.appendChild(li3);

ul.appendChild(li4);

// La liste est crée.

var p2= document.createElement("p");

p2.appendChild(document.createTextNode("Voir: ")); //Paragraphe 2 créé.

var link1=document.createElement("a"); // On va créer un lien.

link1.href="https://www.youtube.com/watch?v=z4VpJL3bNk4";//Lien lié à la balise.

link1.appendChild(document.createTextNode(" Ici!")); //On créé le mot cliquable du lien.

p2.appendChild(link1);// On vient l'ajouter à la suite du contenu du paragraphe.

//Le paragraphe contenant le lien est créé.

x.appendChild(p1);

x.appendChild(ul);

x.appendChild(p2);

//On ajoute tout à la div container dans l'ordre souhaité.

Il faut toujours recréer des variables pour chaque chose qu’on veut ajouter.

Il est possible de simplifier le code en mettant un node directement dans le appendchild :

var link1= document.createElement("a");

link1.href="https:// www.google.com/";

link1.appendChild(document.createTextNode("Google, c'est ici!"));

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

p.appendChild(document.createTextNode("J'aime la pizza, mais je dois faire attention!"));

Les insert before :

var li6=document.createElement("li");

li6.appendChild(document.createTextNode("Iniesta"));

ul.insertBefore(li6,ul.children[2]);

En premier vient la variable que l’on veut ajouter, ensuite l’endroit où il va être placé avant.

Les cookies :

Ce sont des fichiers textes stockés qui vont accompagner chaque requête. Il y a deux types :

Les cookies de session : sont valables que le temps de la session.

Les cookies stockés sur notre ordinateur : Qui sont valables plus longtemps, la durée varie.

Créer un cookie :

document.cookie="nomducookie"="la valeur qu'on veut lui donner."

document.cookie="email\_user"="antoine.tancrez@gmail.com";

Les cookies créés de la sorte sont des cookies de session parce qu’on ne leur a pas donné de valeur.

Chaque cookie ne doit contenir qu’une seule donnée, donc il est tout à fait possible d’en avoir plusieurs.

Si on a deux cookies qui ont le même nom, alors le deuxième remplacera le premier.

Lui donner une date :

document.cookie="email\_user"="antoine.tancrez@gmail.com";expires="th, 18 dec 2018 09:24:00 UTC";

Le mieux est de lui donner une date via JS.

Pour supprimer un cookie on recréé le même en lui donnant une date du passé.

Il est possible de faire une fonction pour créer un cookie.

function setCookie(cname, cvalue, exdays) {

var d = new Date();

d.setTime(d.getTime() + (exdays\*24\*60\*60\*1000));

var expires = "expires="+ d.toUTCString();

document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";

}

Lire un cookie :

Pour lire un cookie, il faut créer une variable :

var x=document.cookie;

Cette variable permet de lire l’ensemble des cookies. Si on veut en trouver un parmi les autres, il faut créer une variable :

function getCookie(cname) {

var name = cname + "=";

var decodedCookie = decodeURIComponent(document.cookie);

var ca = decodedCookie.split(';');

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

var c = ca[i];

while (c.charAt(0) == ' ') {

c = c.substring(1);

}

if (c.indexOf(name) == 0) {

return c.substring(name.length, c.length);

}

}

return "";

}

Un exemple concret :

function setCookie(cname, cvalue, exdays) {

var d = new Date();

d.setTime(d.getTime() + (exdays \* 24 \* 60 \* 60 \* 1000));

var expires = "expires=" + d.toUTCString();

document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";

}

var x = document.cookie;

function getCookie(cname) {

var name = cname + "=";

var decodedCookie = decodeURIComponent(document.cookie);

var ca = decodedCookie.split(';');

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

var c = ca[i];

while (c.charAt(0) == ' ') {

c = c.substring(1);

}

if (c.indexOf(name) == 0) {

return c.substring(name.length, c.length);

}

}

return "";

}

var bc = 0;

function change() {

if (bc == 0) {

document.getElementById("bc").href = "style1.css";

setCookie("background", "black", 2);// On vient préciser que le cookie est celui-ci quand bc==0

bc++;

}

else {

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

setCookie("background", "darkred", 2);//Ici, on précise l'autre choix, à savoir darkred.

bc=0;

}

//Le nom (la clé) du cookie est la même, seule le contenu change pour qu'il soit remplacé à la relance de la fonction.

}

//Ici, ce if/else if vient définir ce que le nav fera quand on relancera la page.

if(getCookie("background")=="black"){ //Si c'est le cookie "black" qui est valide, alors:

bc=0; // Précise que bc=0 pour quand on relance la page.

change(); // lance la fonction change sans avoir besoin de cliquer pour quand on relance la page.

}

else if(getCookie("background")=="darkred"){ // Ici, c'est ce qui se passe si le dernier cookie utilisé était le darkred

bc=1; // précise que bc=1 pour quand on relance la page.

change();// Toujours la relance de la fonction pour quand la page se lance.

}