***JavaScript :***

On peut intégrer JS de 4 manières différentes :

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

**Internal** (dans le head)

**External** (dans un fichier externe)

**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…

**.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 »