JavaScript

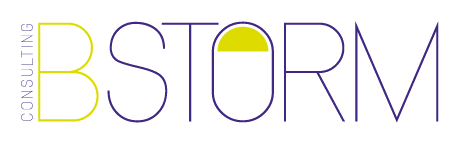


Table des matières

[Introduction 4](#_Toc525572173)

[Afficher le fichier « bonjour.html » 4](#_Toc525572174)

[NB 4](#_Toc525572175)

[Chapitre 1 : Les bases 5](#_Toc525572176)

[Introduction 5](#_Toc525572177)

[Afficher du texte 5](#_Toc525572178)

[Les commentaires 5](#_Toc525572179)

[Les commentaires mono-lignes 5](#_Toc525572180)

[Les commentaires multi-lignes 5](#_Toc525572181)

[Les variables 6](#_Toc525572182)

[Un nom et une valeur 6](#_Toc525572183)

[Les différents types de variables 6](#_Toc525572184)

[Conversion et transformation 6](#_Toc525572185)

[Affecter une valeur à une variable 7](#_Toc525572186)

[Utiliser les types de données 7](#_Toc525572187)

[Le type string (chaine de caractères) 7](#_Toc525572188)

[Le type number : entier (nombre entier) 7](#_Toc525572189)

[Le type number : float (nombre décimal) 7](#_Toc525572190)

[Le type boolean (booléen) 7](#_Toc525572191)

[Le type object (vide) 7](#_Toc525572192)

[Les constantes 8](#_Toc525572193)

[Afficher et concaténer des variables 8](#_Toc525572194)

[Afficher le contenu d’une variable 8](#_Toc525572195)

[La concaténation 8](#_Toc525572196)

[Interagir avec l'utilisateur 8](#_Toc525572197)

[Afficher un message et le contenu d'une variable 8](#_Toc525572198)

[Afficher un message et récupérer une valeur 8](#_Toc525572199)

[Chapitre 2 : Les opérateurs 9](#_Toc525572200)

[Les opérations de base 9](#_Toc525572201)

[Le modulo 9](#_Toc525572202)

[Concaténation d'affectation 9](#_Toc525572203)

[Chapitre 3 : Les opérateurs conditionnels 10](#_Toc525572204)

[Les symboles de comparaison 10](#_Toc525572205)

[Chapitre 4 : Les structures conditionnelles 10](#_Toc525572206)

[La structure if… else 10](#_Toc525572207)

[Les ternaires 10](#_Toc525572208)

[La structure if… else if… else 11](#_Toc525572209)

[La structure switch 11](#_Toc525572210)

[Chapitre 5 : Les structures itératives 12](#_Toc525572211)

[while 12](#_Toc525572212)

[for 12](#_Toc525572213)

[Chapitre 6 : Les tableaux 13](#_Toc525572214)

[Les tableaux scalaires 13](#_Toc525572215)

[Les tableaux associatifs 13](#_Toc525572216)

[Parcourir un tableau 13](#_Toc525572217)

[La boucle for 13](#_Toc525572218)

[La boucle for in 13](#_Toc525572219)

[La boucle forEach 13](#_Toc525572220)

[Quelques fonctions 13](#_Toc525572221)

# Introduction

Enregistrez le fichier : File > Save ou ctrl + S. Appelez le « bonjour.html»

Sublime Text nous offre encore un raccourci : dès que vous écrivez un fichier html, tapez « script » et puis appuyez sur la touche « Tab ».

Il ajoute lui-même : <script></script>

Il nous reste à écrire entre <script></script> la commande :

console.log('Hello');

Votre fichier doit donc contenir :

<body>

<script>

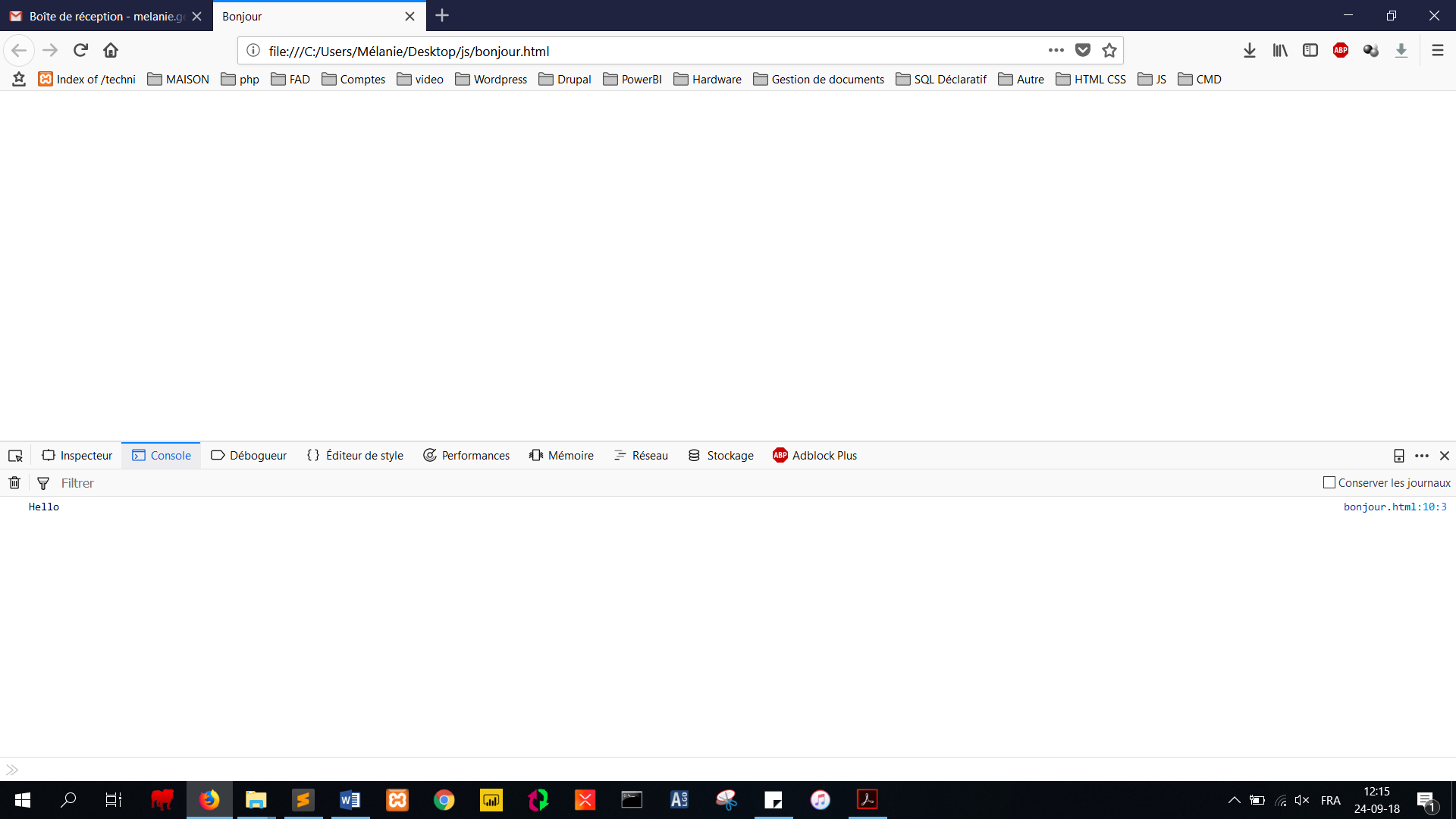
console.log('Hello');

</script>

</body>

### Afficher le fichier « bonjour.html »

Ouvrez votre navigateur et tapez sur la touche F12, rendez-vous dans l'onglet Console



Dans un premier temps, nous allons apprendre les fondamentaux de JS uniquement à l'aide de la console. Après nous verrons comment utiliser JS pour modifier la mise en page du site.

### NB

On peut placer les balises <script> </script> dans la balise <head></head> mais attention à l'ordre d'exécution. Il va essayer de lancer le JS même si la page n'est pas encore chargée.

Il est donc préférable de placer le JS en toutes dernières lignes avant la balise </body> ainsi, il connait l'entièreté du document et tous les appels seront possibles.

Il est également possible de placer tout le JS dans un fichier .js

Il suffira d'intégrer la balise juste avant la balise </body>

|  |  |
| --- | --- |
| **bonjour.html** | **hello.js** |
| <script src="hello.js"></script> | console.log('Hello'); |

# Chapitre 1 : Les bases

## Introduction

JS est un langage de programmation. Chaque instruction doit se terminer par « ; ».

## Afficher du texte

Pour afficher du texte, on utilise l’instruction « console.log » suivie du texte à afficher entre "mon texte" entre les parenthèses.

<script>

console.log('Hello');

</script>



N’oubliez pas le « ; » à la fin de votre instruction.

## Les commentaires

### Les commentaires mono-lignes

Pour indiquer un commentaire sur une ligne, on utilise les caractères « // » suivis du texte en commentaire.

<script>

//un commentaire sur une ligne

</script>

### Les commentaires multi-lignes

Pour indiquer un commentaire sur plusieurs lignes, on utilise les caractères « /\* » suivis du texte en commentaire terminé par « \*/ ».

<script>

/\*un commentaire

sur plusieurs ligne\*/

</script>

## Les variables

### Un nom et une valeur

Une variable a un nom. Elle est déclarée par le mot "var" suivi du nom de la variable composé de lettres.

#### Déclarer

let age; // pas d’accent, en un mot

let lieu\_de\_naissance; //pas d’espace

#### Affecter

age = 20;

lieu\_de\_naissance = "Nivelles" ;

#### Déclarer et affecter

Une variable a une valeur.

let age = 20;

let lieu\_de\_naissance = "Nivelles" ;

### Les différents types de données

|  |  |  |  |
| --- | --- | --- | --- |
| **Symbole** | **Nom** | **Signification** | **Exemple** |
| *string* | Chaine de caractères | C’est une suite de caractères | "Mon texte " |
| *number* | Nombres | C’est un nombre | 5, -5, 12.74, 3.141592 |
| *boolean* | Booléens | 2 valeurs possibles : « true » ou « false » | true  false |
| *object* | Objet | C'est un objet | new Objet() |
| *function* | Fonction | C'est une fonction | function nom(param) {} |
| *undefined* | sans valeur | Ne contient rien |  |

Pour pouvoir connaitre le type d'une variable, on utilise la fonction typeof(nom\_variable);

### Conversion et transformation

|  |  |  |
| --- | --- | --- |
| **DE** | **EN** | **FONCTION** |
| *string* | *number* | Number(nom\_variable);  Si conversion impossible, NaN  (Not a Number) |
| *string* | *entier* | parseInt(nom\_variable)  Si conversion impossible, NaN |
| *string* | *float (nombre à virgule)* | parseFloat(nom\_variable)  Si conversion impossible, NaN |
| *string* | *string en minuscule* | nom\_variable.toLowerCase(); |
| *string* | *string en majuscule* | nom\_variable.toUpperCase(); |
| *string* | *une partie de string* | nom\_var.substring(1,4)  preCar = 0;  derCar = string.length(); |
| *string* | *1 caractère (position\_car)* | nom\_var.charAt(1) |

### Affecter une valeur à une variable

Pour affecter (donner) une valeur à une variable, on utilise le signe « = » suivi de sa valeur.

age = 20;

## Utiliser les types de données

### Le type string (chaine de caractères)

Une chaine de caractères doit être entourée de soit :

* " " : des guillemets
* ' ' : des apostrophes

nomUtilisateur1 = "Roméo";

nomUtilisateur2 = 'Juliette';

Pour intégrer des guillemets ou des apostrophes au texte, il faut :

* utiliser un caractère d’échappement « \ ».

texte\_a\_afficher = "Mon \"nom\" est Romeo";

texte = 'Je m\'appelle Juliette';

OU

* inverser les guillemets et les apostrophes :

texte\_a\_afficher = 'Mon "nom" est Romeo';

texte = "Je m'appelle Juliette";

### Le type number : entier (nombre entier)

Pour stocker un nombre entier dans une variable, on indique simplement le nombre.

age = 20;

### Le type number : float (nombre décimal)

Pour stocker un nombre décimal dans une variable, on indique simplement le nombre en remplaçant la virgule par un point.

poids = 53.7;

### Le type boolean (booléen)

Pour stocker une valeur « vraie » ou « fausse », on utilise respectivement « true » et « false » :

* true = vrai ;
* false = faux.

connecte = true;

sexeF = false;

### Le type object (vide)

Pour stocker une valeur sans valeur, on utilise « NULL ».

pas\_de\_valeur = null;

## Les constantes

Les constantes ne sont accessibles qu'en lecture.

const taux\_tva = 0.21;

## Afficher et concaténer des variables

### Afficher le contenu d’une variable

Pour afficher du texte, on utilise l’instruction « console.log() » suivie du texte à afficher entre "mon texte".

nomUtilisateur = "Juliette";

console.log(nomUtilisateur);

### La concaténation

#### Avec des guillemets +

Pour assembler plusieurs contenus sur une ligne, on utilise :

nomUtilisateur = "Juliette";

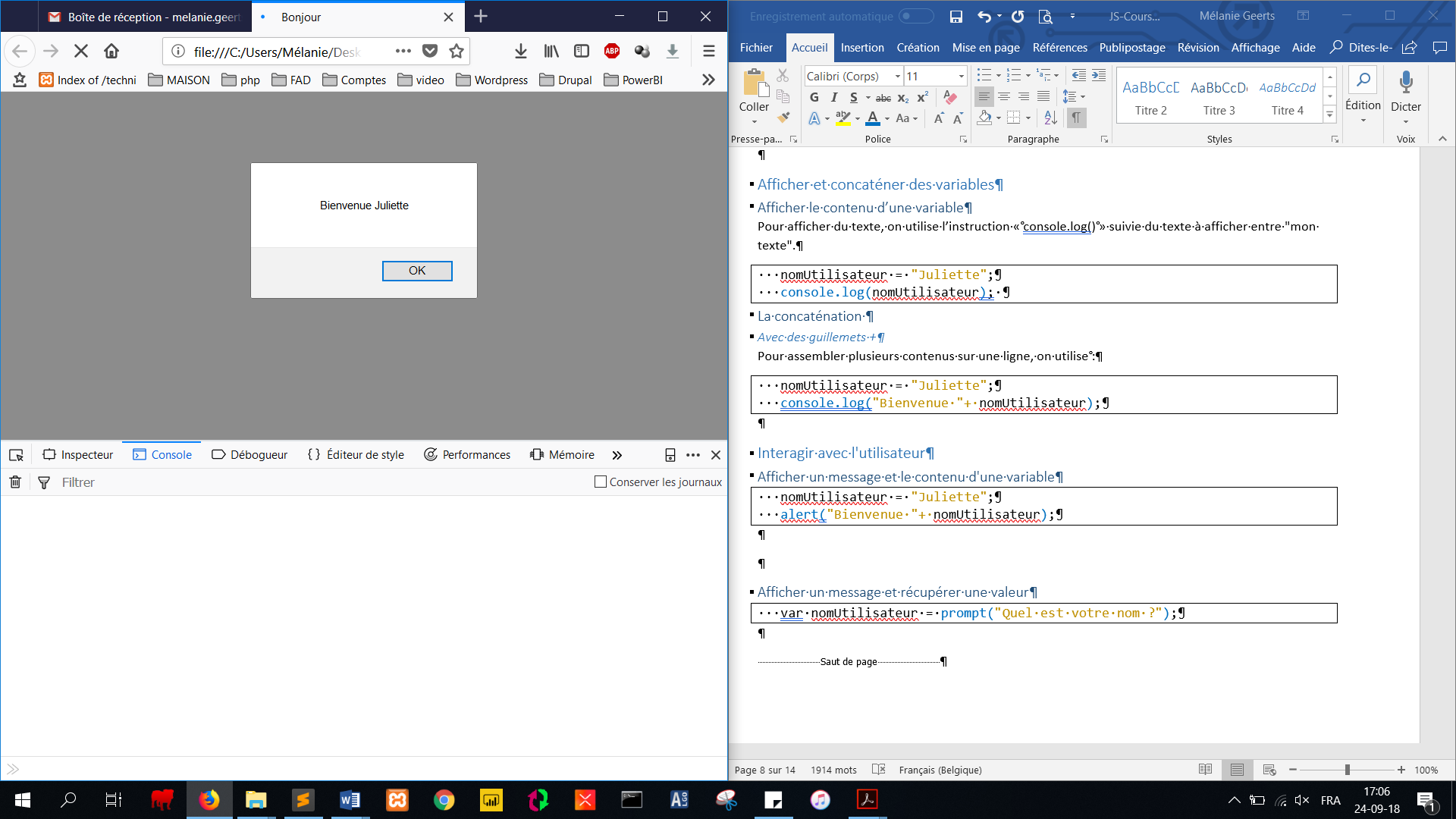
console.log("Bienvenue "+ nomUtilisateur);

## Interagir avec l'utilisateur

### Afficher un message et le contenu d'une variable

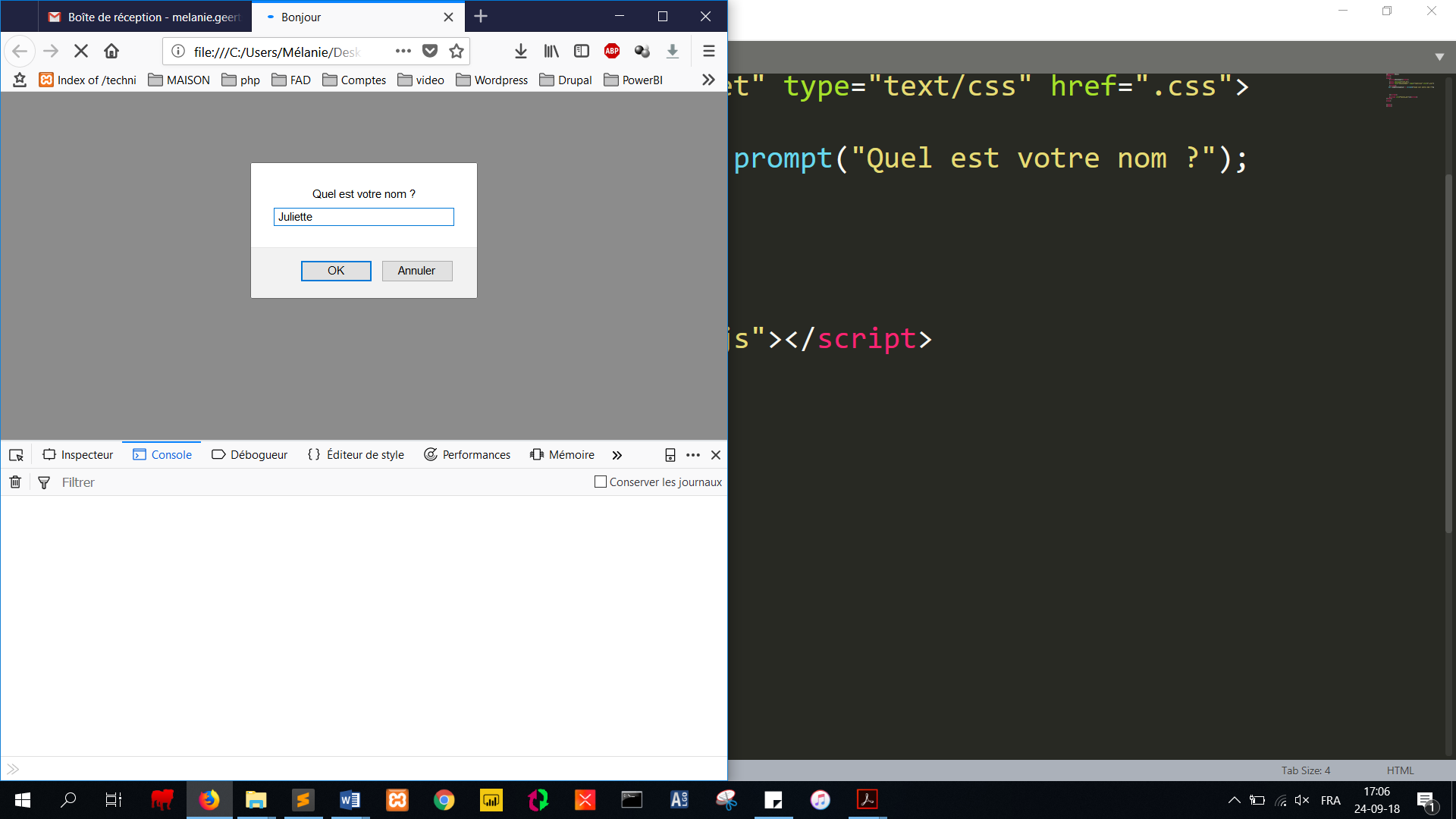
nomUtilisateur = "Juliette";

alert("Bienvenue "+ nomUtilisateur);



### Afficher un message et récupérer une valeur

let nomUtilisateur = prompt("Quel est votre nom ?");



# Chapitre 2 : Les opérateurs

## Les opérations de base

|  |  |
| --- | --- |
| **Symbole** | **Signification** |
| + | Addition |
| - | Soustraction |
| \* | Multiplication |
| / | Division |
| % | Modulo |
| ( ) | Parenthèses |

addition = 2 + 4; //6

soustraction = 4 – 2; //2

multiplication = 2 \* 4; //8

division = 4 / 2; //2

calculParenthese = (1 + 2) \* 2; //6

## Le modulo

C’est le reste de la division entière. Pour récupérer le résultat d’un modulo, on utilise le signe « % ».

modulo = 4 % 2 ; //0 car il reste 0

modulo = 9 % 2 ; //1 car il reste 1

## Concaténation d'affectation

|  |  |
| --- | --- |
| resultat += 4; | resultat = resultat + 4; |
| let txt1 = "What a very "; txt1 += "nice day"; | // txt1 = "What a very nice day"; |
| resultat -= 4; | resultat = resultat – 4; |
| **resultat \*=** 4; | resultat = resultat \* 4; |
| resultat **/= 4;** | resultat = resulat / 4; |
| **resultat %= 2** | resultat = resultat % 2; |
| resultat++; | resultat = resultat + 1; |
| resultat --; | resultat = resultat – 1; |

# Chapitre 3 : Les opérateurs conditionnels

## Les symboles de comparaison

|  |  |
| --- | --- |
| **Symbole** | **Signification** |
| *==* | est égal à |
| *===* | est égal et de même type |
| *>* | est supérieur à |
| *<* | est inférieur à |
| *>=* | est supérieur ou égal à |
| *<=* | est inférieur ou égal à |
| *!=* | est différent de |
| *!==* | est différent et de type différent |
| ***!*** | négation logique |
| *&&* | le ‘et’ logique |
| *||* | le ‘ou’ inclusif logique |

# Chapitre 4 : Les structures conditionnelles

## La structure if… else

Pour ajouter une condition, on utilise le mot « if » qui signifie « si », on ajoute ensuite la condition entre parenthèses. Ensuite, entre accolades, on place l’action à exécuter si la condition est vraie.

let age = 18;

if (age >= 18)

{

console.log("Bienvenue");

}

else

{

console.log("Accès interdit");

}

if (pays == "France" **||** pays == "Belgique")

{

console.log("Bienvenue sur notre site !");

}

else

{

console.log("Désolés, notre service n'est pas encore disponible dans votre pays !");

}

## Les ternaires

S’il n’y a qu’une condition, on l’écrit sur une seule ligne :

let age = 24;

let message = (age >= 18) ? "Bienvenue" : "Accès interdit";

console.log(message);

## La structure if… else if… else

S’il y a plus que 2 conditions, on utilise :

let jour = 4;

if (jour == 1)

{

console.log("Lundi");

}

else if (jour == 2)

{

console.log("Mardi");

}

else if (jour == 3)

{

console.log("Mercredi");

}

else if (jour == 4)

{

console.log("Jeudi");

}

else if (jour == 5)

{

console.log("Vendredi");

}

else if (jour == 6)

{

console.log("Samedi");

}

else if (jour == 7)

{

console.log("Dimache");

}

else

{

console.log("Erreur");

}

## La structure switch

let jour = 4;

switch (jour)

{

case 1 :

console.log("Lundi");

break;

case 2 :

console.log("Mardi");;

break;

case 3 :

console.log("Mercredi");;

break;

case 4 :

console.log("Jeudi");

break;

case 5 :

console.log("Vendredi");

break;

case 6 :

console.log("Samedi");

break;

case 7 :

console.log("Dimanche");

break;

default :

console.log("Erreur");

break;

}

# Chapitre 5 : Les structures itératives

## while

Tant que la condition est vraie, les instructions s’exécutent.

let nombre = 0;

while ((nombre < 1) || (nombre > 10))

{

console.log("Nombre incorrect ! Réessayez.");

nombre++; //est équivalent à nombre = nombre + 1;

}

let nombre = 1;

while (nombre <= 10)

{

console.log('Ligne n°'+nombre+'\n');

nombre++; //est équivalent à nombre = nombre + 1;

}

## for

Elle contient 3 éléments :

* départ : « Initialisation » : valeur que l’on donne au départ à la variable.
* condition : tant que la condition est remplie, la boucle est exécutée.
* pas : « incrémentation » : le nombre que l’on ajoute à chaque passage.

for(nombre =1;nombre <10; nombre++)

*{*

console.log('Ceci est la ligne n°'+*nombre+*'\n')*;*

}

# Chapitre 6 : Les tableaux

## Les tableaux scalaires

On peut créer un tableau comme suit :

let tableau = ["Mercure", "Vénus", "Terre"];

Ou de la manière suivante :

let tableau = new Array("Mercure", "Vénus", "Terre");

## Les tableaux associatifs

Un tableau associatif est un tableau où l'indice n'est pas un nombre mais un mot clé.

On peut créer un tableau associatif comme suit :

let tableau = {"planete1" : "Mercure"};

Ou de la manière suivante :

let tableau = new Object();

tableau["planete1"] = "Mercure";

tableau["planete2"] = "Venus";

tableau["planete3"] = "Terre";

## Parcourir un tableau

### La boucle for

let tableau = ["Mercure", "Vénus", "Terre"];

for(nombre =0; nombre <tableau.length; nombre++)

*{*

console.log('Planète n°'*+(nombre+1)+*' : '+*tableau[nombre]+* '\n')*;*

}

### La boucle for of

let tableau = ["Mercure", "Vénus", "Terre"];

for(let value of tableau)

*{*

console.log(value);

}

### La boucle forEach

var tab = ['html', 'css', 'js'];

tab.forEach(element => {

console.log(element);

});

### Quelques fonctions

|  |  |
| --- | --- |
| tab.push('JQuery'); | ajoute un élément à la fin du tableau |
| tab.unshift('Ajax'); | ajoute un élément au début du tableau |
| tab.shift(); | retire le premier élément du tableau |
| tab.pop(); | retire le dernier élément du tableau |
| var tab = chaine\_car.split(' '); | renvoie un tableau dont l'élément séparateur est ' ' |
| var chaine\_car = tab.join('-'); | renvoie une chaine de caractères composée des différents éléments du tableau séparé par - |