

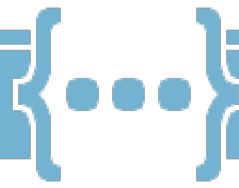
Cours 2

Opérateurs logiques, DOM, conditions, Webstorm et fonctions

Intro. à la programmation - Aut. 2021



- ❖ Type booléen
- ❖ Opérateurs de comparaison
- ❖ Opérateurs logiques
- ❖ Introduction au DOM
- ❖ Instructions conditionnelles
- ❖ WebStorm
- ❖ Introduction aux fonctions

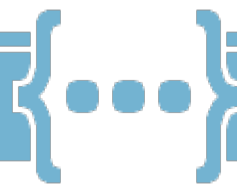


❖ Booléens

- ◆ C'est un autre **type de données**. (Nous avons vu **nombres**, **nombres décimaux** et **chaînes de caractères** pour le moment)
- ◆ Les *booléens* ont seulement 2 valeurs possibles
 - **true**
 - **false**
- ◆ Ils permettent d'exprimer que quelque chose est **vrai** ou **faux**
- ◆ Ce sont des **valeurs** qu'on peut affecter à des **variables** :
 - Attention ! Ce ne sont PAS des **chaînes de caractères** !

let a = **true**;

let b = **false**;



❖ Opérateurs de comparaison

◆ Donnent un résultat qui est **true** ou **false**

◆ Plus grand que > 5.5 > 6.5 (false)

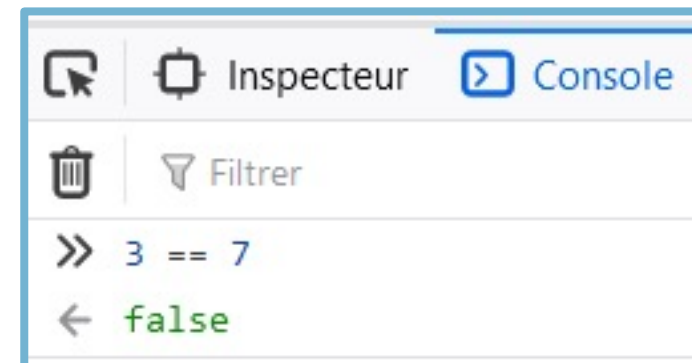
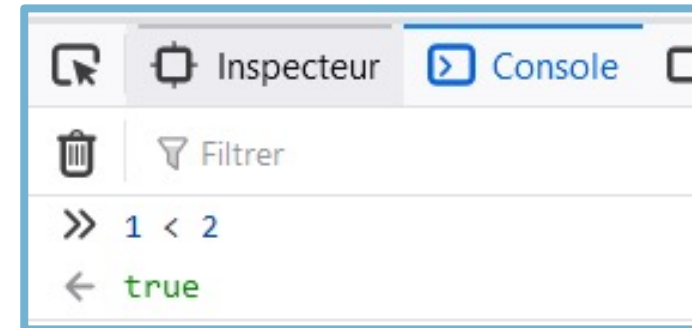
◆ Plus grand ou égal >= 5 >= 5 (true)

◆ Plus petit < 5 < 7 (true)

◆ Plus petit ou égal <= 5 <= 7 (true)

◆ Égal == 5 == 7 (false)

◆ Pas égal != 5 != 7 (true)





❖ Opérateurs de comparaison (Avec des chaînes de caractère)

◆ Donnent un résultat qui est **true** ou **false**

- | | | |
|----------------------|----|--|
| ◆ Plus grand que | > | "bon" > "arbre" (true , ordre alphabétique) |
| ◆ Plus grand ou égal | >= | "ball" >= "car" (false , ordre alphabétique) |
| ◆ Plus petit | < | "car" < "ball" (false , ordre alphabétique) |
| ◆ Plus petit ou égal | <= | "ball" <= "ball" (true , identiques) |
| ◆ Égal | == | "allo" == "allo_" (false , différents) |
| ◆ Pas égal | != | "allo" != "allo_" (true , différents) |



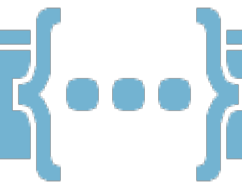
❖ Priorité des opérateurs

◆ Ordre de priorité (Du premier au dernier)

- **Parenthèses**
- ***, /, %**
- **+, -**
- **<, <=, >, >=**
- **==, !=**
- **=**

let a = 4 + 6 > 2 * 3 + 5; → let a = 10 > 11; → let a = false;

let b = 4 + 6 == 2 + 8; → let b = 10 == 10; → let b = true;



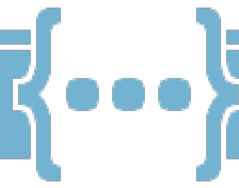
❖ Booléens

- ◆ Nous connaissons donc maintenant les « données de type **booléen** » (**true** et **false**) et les **opérateurs de comparaison** **>**, **<**, **>=**, **<=**, **==**, **!=**
 - Exemples

5 **>=** 5 true

"car" **<** "ball" false

"allo" **!=** "allo " true



❖ Opérateurs logiques

◆ Permettent de combiner plusieurs expressions de comparaison !

◆ AND &&

- Les 2 conditions doivent être **true**

`1 < 2 && 2 > 3` (**false**, car `2 > 3` n'est pas **true**)

◆ OR* ||

- Au moins une condition doit être **true**

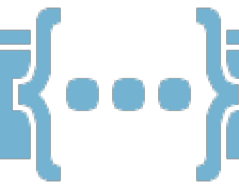
`1 < 2 || 2 > 3` (**true**, car `1 < 2` est **true**)

◆ NOT !

- Le booléen est **inversé** (true devient false, false devient true)

`!(1 < 2)` (**false**, car `1 < 2` était **true**, mais on inverse)

* || représente deux barres verticales, et non deux L minuscules.



❖ Opérateurs logiques

◆ Exemples concrets



let **prixPerruche** = 5;



let **prixBaudruche** = 10;



let **prixRuche** = 15;

- Le prix d'une **ruche** 🐝 est à la fois plus élevé que le prix d'une **perruche** 🐦 ET que le prix d'une **baudruche** 🎈

`prixRuche > prixPerruche && prixRuche > prixBaudruche`

Attention ! On ne doit pas écrire l'expression comme ceci :



`prixRuche > prixPerruche && prixBaudruche`



❖ Opérateurs logiques

◆ Exemples concrets



let **prixPerruche** = 5;

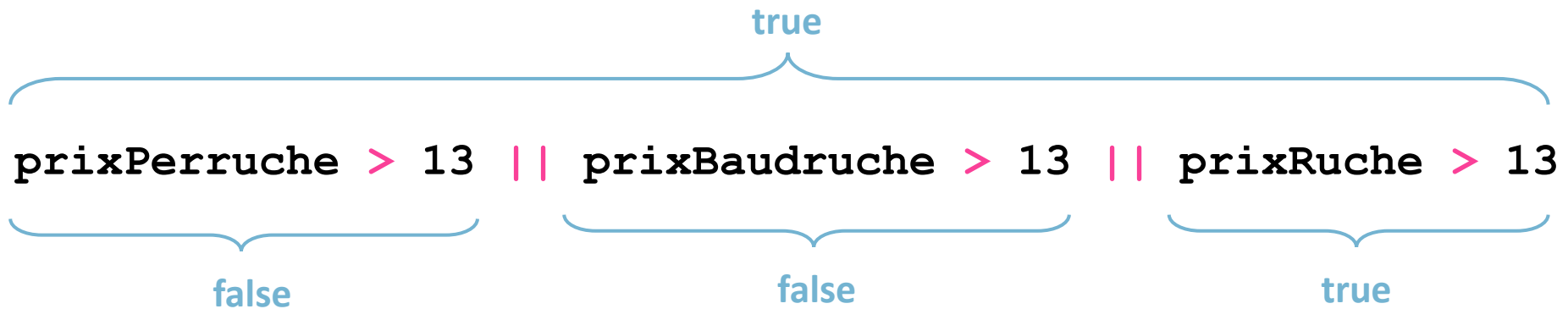


let **prixBaudruche** = 10;



let **prixRuche** = 15;

- Au moins un des trois prix est **plus élevé** que 13\$



Attention ! On ne doit pas écrire l'expression comme ceci :



`prixPerruche || prixBaudruche || prixRuche > 13`



❖ Opérateurs logiques

◆ Exemples concrets



let **prixPerruche** = 5;



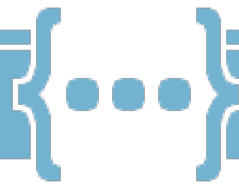
let **prixBaudruche** = 10;



let **prixRuche** = 15;

- Le prix d'une perruche 🐦 N'EST PAS égal à 5

true
! (prixPerruche == 5) ou encore **prixPerruche != 5**
false **false**



❖ Opérateurs logiques

◆ Exemples concrets



let **prixPerruche** = 5;



let **prixBaudruche** = 10;



let **prixRuche** = 15;

- Le prix d'une ruche 🐝 N'EST PAS plus élevé que le prix de 2 perruches 🐦 🐦.

true

! (**prixRuche** > 2 * **prixPerruche**)

false, à cause du « ! »



❖ JavaScript HTML **DOM**

- ◆ **DOM** = **D**ocument **O**bject **M**odel
- ◆ Le **DOM** nous permet, à l'aide de **Javascript**, de modifier le code HTML et CSS d'une page Web grâce à des instructions que nous allons apprendre.
 - Exemples
 - Couleur du texte
 - Contenu textuel d'un élément HTML
 - Taille du texte
 - Police du texte
 - etc.



❖ Utiliser **DOM** avec un élément HTML

◆ En HTML, les éléments / balises peuvent avoir des « **id** ».

- Élément sans **id** :

```
<p> Petit tapis rouge.</p>
```

- Élément avec un **id** :

```
<p id="gris"> Petit tapis gris.</p>
```

- Ci-dessus, on dit que « L'**id** de cet élément HTML est "**gris**" »
- Deux éléments HTML n'ont pas le droit d'avoir le même **id** !



❖ Utiliser **DOM** avec un élément HTML

- ◆ Nous pouvons nous servir de l'**id** d'un élément HTML afin de pouvoir l'utiliser dans le code JavaScript :

```
<div id="pikachu"> Pick a shoe </div>
```




```
document.getElementById("pikachu").textContent = "Pikachu";
```

Ce bout de code veut dire « Pour l'élément dont l'id est **pikachu** », ...

- ◆ Pour utiliser le `<div>` (ex : le modifier, obtenir son contenu textuel, etc.), on utilise son **id**, qui ici est "**pikachu**".
 - On peut ajouter soi-même un id à un élément qu'on souhaite modifier s'il n'en a pas.



❖ Utiliser **DOM** avec un élément HTML

- ◆ Notons que pour le moment, nous utilisons la **console** du navigateur Web pour écrire notre code **JavaScript**, alors tous les changements que nous faisons sont temporaires !
 - **Réactualiser**  la page Web réinitialise les changements faits avec DOM dans la console.



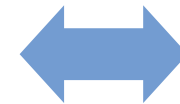
❖ Obtenir le contenu textuel

◆ `document.getElementById("id").textContent`

- Par exemple ici, on veut le contenu textuel de l'élément avec l'id "title".
 - La console nous retourne « **Smudge** ».

```
<h2 id="title">Smudge</h2>  

```



Smudge





❖ Modifier le contenu textuel

- ◆ `document.getElementById("id").textContent = "nouveau texte";`
 - Ici, on veut changer le titre « Smudge » pour « Chat consterné ». On doit utiliser l'id **title** pour le faire.

```
<h2 id="title">Smudge</h2>  

```



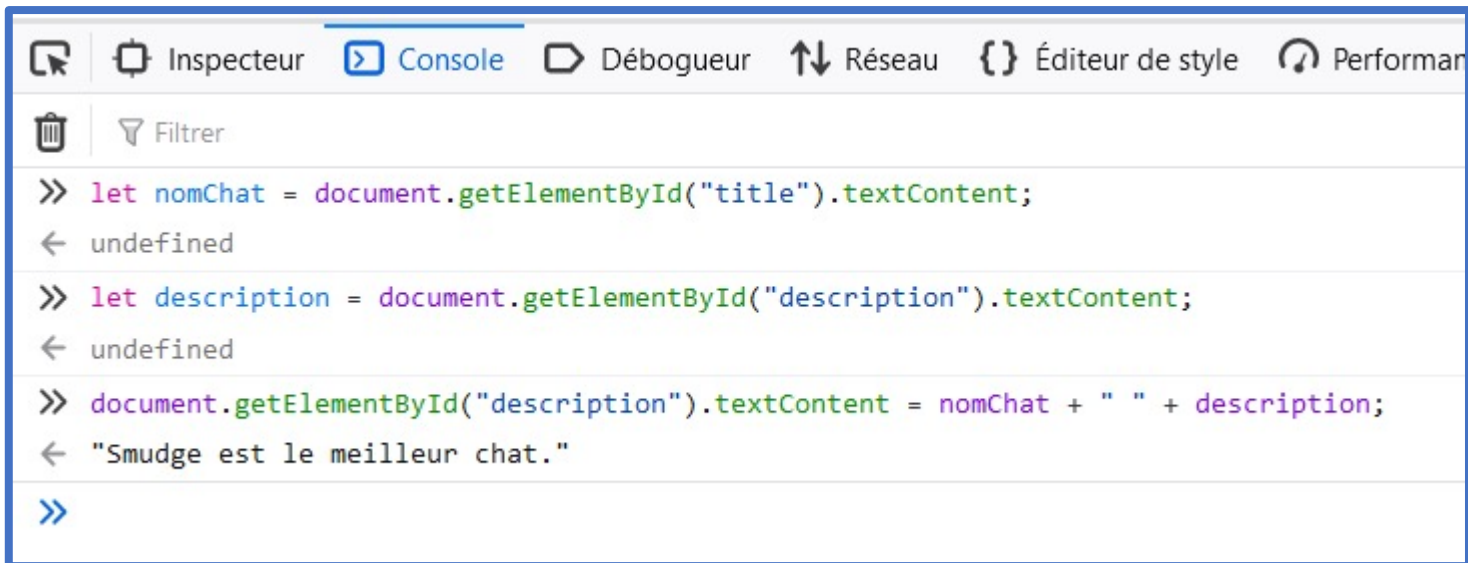


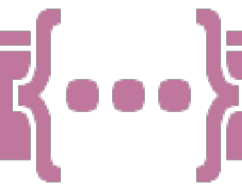
❖ Modifier le contenu textuel

- ◆ `document.getElementById("id").textContent = "nouveau texte";`
 - On peut même se servir du texte présent dans d'autres éléments pour modifier le texte d'un élément.

```
<h2 id="title">Smudge</h2>
<p id="description">est le meilleur chat.</p>

```





❖ Commentaires en JavaScript

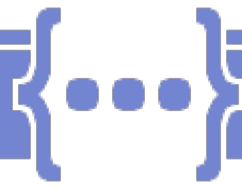
- ◆ Commentaires **mono-ligne** (Avec `// . . .`)
 - Tout ce qui est à droite des `//` est un commentaire.

```
// Ceci est un commentaire  
let a = 1;  
let b = 2;  
// let c = 3;
```

- ◆ Commentaires **multi-ligne** (Avec `/* . . . */`)
 - Le commentaire débute à `/*` et se termine à `*/`

```
/*  
  Commentaire sur  
  plusieurs lignes  
  let d = 50;  
*/
```

- ◆ Les commentaires permettent de faire des annotations dans le code. Ils sont ignorés lorsque l'application est exécutée.
 - Ça sert à laisser des notes / descriptions dans le code pour se retrouver !



❖ Bloc **if**

- ◆ Exécute un morceau de code seulement si une **condition** est respectée
- ◆ Syntaxe :

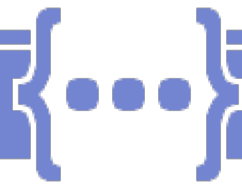
```
if (/*...Condition...*/)
{
    // Code à exécuter si la condition est « true »
}
```

- La **condition** est entourée de parenthèses (...)
- Le **bloc de code conditionnel** est entouré d'accolades {...}

Exemple :

- ◆ Ici, on écrit "majeur" dans l'élément avec l'id **personne** seulement si la variable **age** est supérieure ou égale à **18**.
- ◆ Sinon, on saute le bloc de code du **if**.

```
let age = 19;
if (age >= 18)
{
    document.getElementById("personne").textContent = "Majeur(e)";
}
```

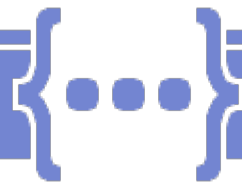


❖ Bloc **else**

- ◆ Les blocs **if** peuvent être accompagnés d'un bloc **else**.
- ◆ Syntaxe :

```
if (/*...Condition...*/)
{
    // Code à exécuter si la condition est « true »
}
else
{
    // Code à exécuter si la condition est « false »
}
```

- Le **else** n'est pas suivi d'une **condition**, car il est associé à la même condition que le **if**.
- Le bloc de code sous le **else** s'exécute seulement si la condition est **false**.
 - Ainsi, il y a forcément **un seul des deux** blocs de code qui s'exécutera.

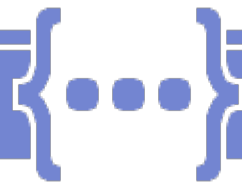


❖ Bloc **else**

◆ Exemples

- ◆ Ici, la condition du **if** est **true**. On va donc seulement exécuter le code du bloc **if**.
 - On met le texte « Enseignant(e) » à l'élément **personne**.

```
let nom = "Maxime";
if(nom == "Maxime" || nom == "Maude" || nom == "Mathieu")
{
    document.getElementById("personne").textContent = "Enseignant(e)";
}
else
{
    document.getElementById("personne").textContent = "Étudiant(e)";
}
```

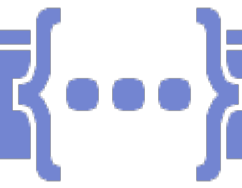


❖ Bloc **else**

◆ Exemples

- ◆ Ici, la condition du **if** est **false**. On va donc ignorer le code du bloc **if** et exécuter le bloc **else**.
 - On met le texte « Étudiant(e) » à l'élément **personne**.

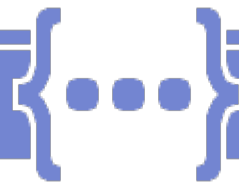
```
let nom = "Thérèse";
if(nom == "Maxime" || nom == "Maude" || nom == "Mathieu")
{
    document.getElementById("personne").textContent = "Enseignant(e)";
}
else
{
    document.getElementById("personne").textContent = "Étudiant(e)";
}
```

❖ Bloc **else if**




- ◆ Permet d'insérer une ou plusieurs conditions alternatives
- ◆ Syntaxe :

```
if(/*...Condition 1...*/)
{
    // Code à exécuter si la condition 1 est « true »
}
else if(/*...Condition 2...*/)
{
    // Code à exécuter si la condition 1 est « false » et la condition 2 est « true »
}
else
{
    // Code à exécuter si les conditions 1 et 2 sont « false »
}
```

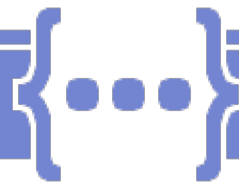


❖ Bloc **else if**

◆ Exemple 1

- ◆ La première condition est **false**. On saute le bloc **if**. 
- ◆ La deuxième condition est **true**. On exécute le bloc **else if** ! 
- ◆ On saute le **else** puisqu'un des blocs au-dessus était **true**. 
 - On met le texte « Grand » à l'**élément**.

```
let a = 6;
if(a < 3)
{
    document.getElementById("element").textContent = "Petit";
}
else if(a > 5)
{
    document.getElementById("element").textContent = "Grand";
}
else
{
    document.getElementById("element").textContent = "Moyen";
}
```



❖ Bloc **else if**

◆ Exemple 2

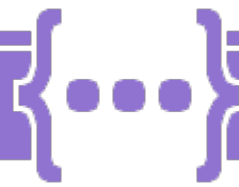
◆ La première condition est **false**. On saute le bloc **if**.

◆ La deuxième condition est **false**. On saute le bloc **else if**.

◆ On exécute le **else** puisque les deux blocs au-dessus était **false**.

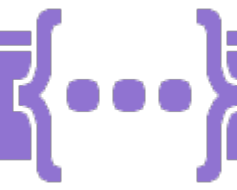
- On met le texte « Moyen » à l'**élément**.

```
let a = 4;
if(a < 3)
{
    document.getElementById("element").textContent = "Petit";
}
else if(a > 5)
{
    document.getElementById("element").textContent = "Grand";
}
else
{
    document.getElementById("element").textContent = "Moyen";
}
```



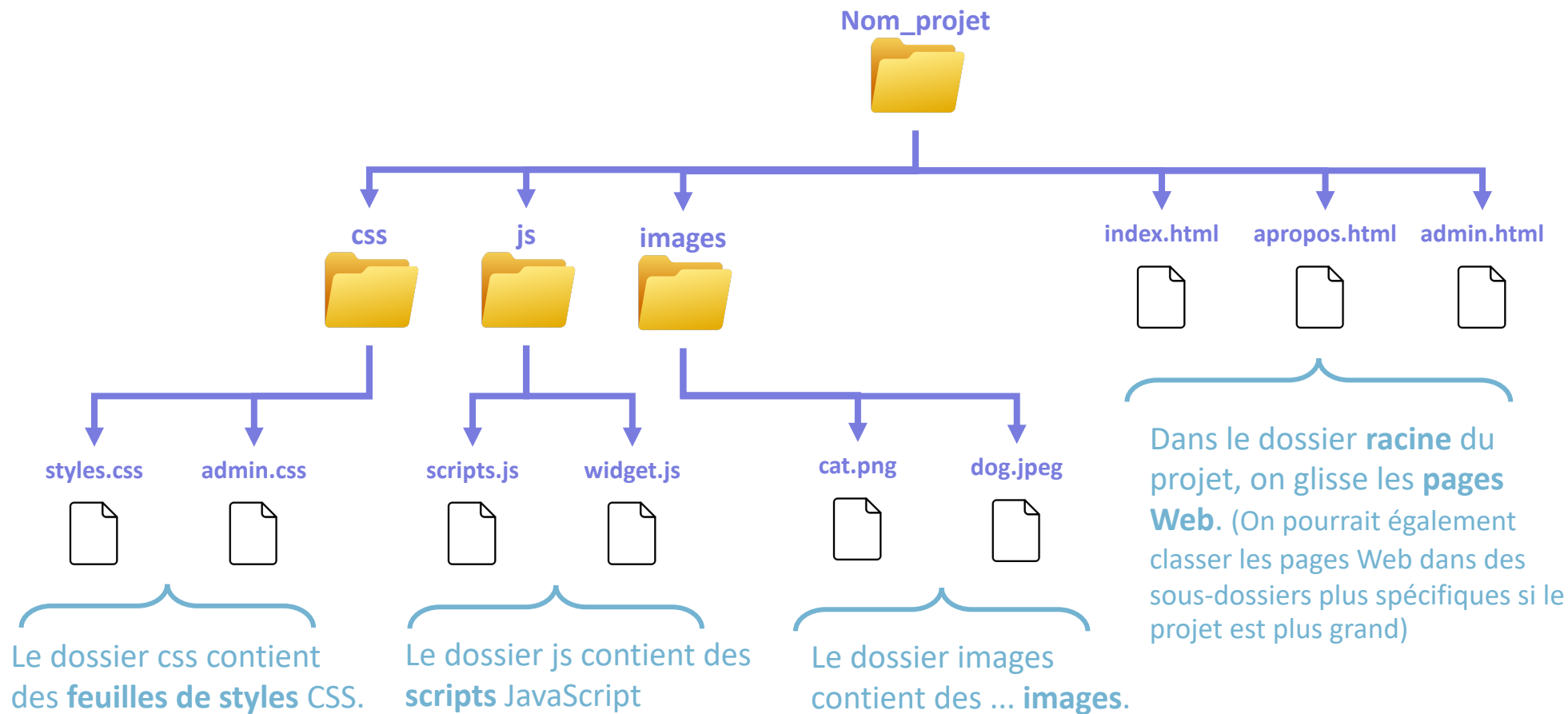
❖ WebStorm

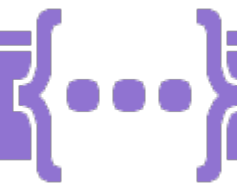
- ◆ « Environnement de développement » pour JavaScript
 - Outil qui nous aidera à créer des projets avec **JavaScript**
- ◆ C'est un logiciel payant 💰
 - Il est disponible sur les ordinateurs au cégep
 - Il est gratuit si vous avez une licence étudiante (Voir document « **WebStorm à la maison** » sur la page du cours. (Léa ou Github)
- ◆ WebStorm nous aide en détectant les **erreurs de syntaxe** dans notre code.
 - Permet aussi l'autocomplétion de code
 - C'est un outil que vous réutiliserez au cours de la technique !



❖ Créer un nouveau projet Web

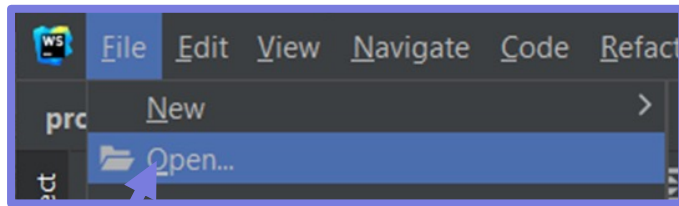
- ◆ Il faut respecter l'arborescence suivante pour le dossier de notre projet





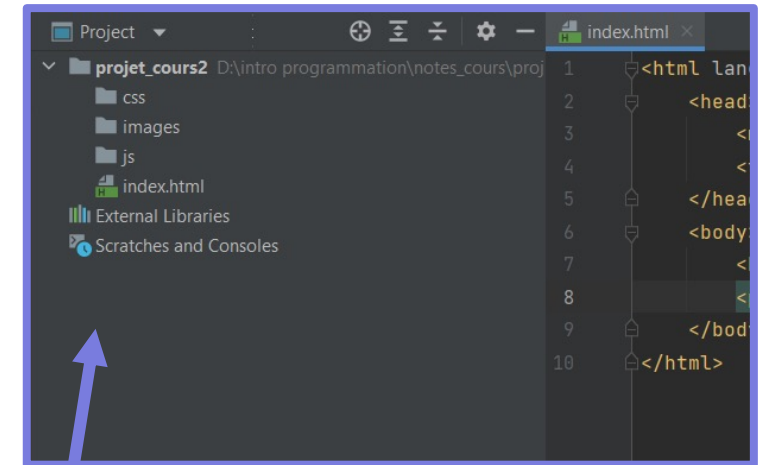
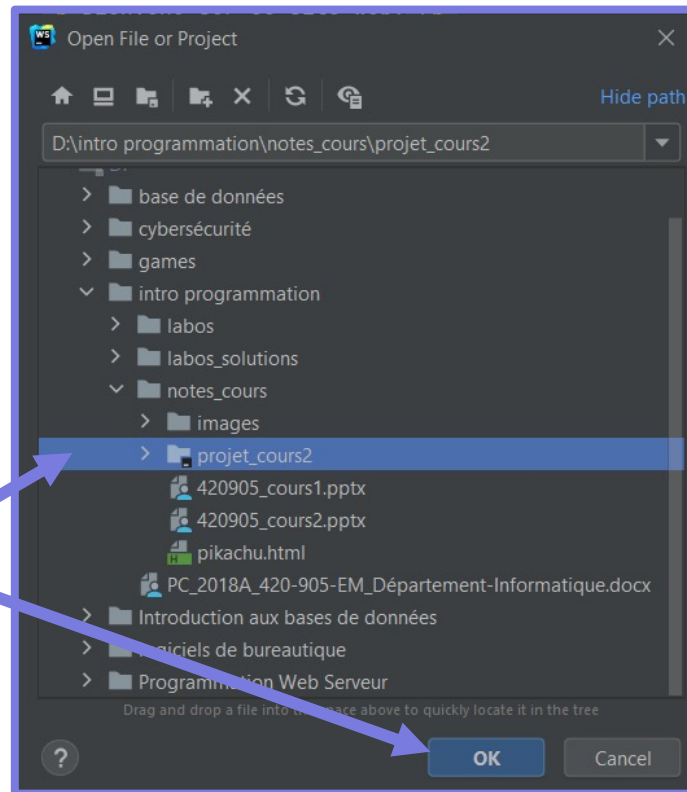
❖ Ouvrir un projet

- ◆ Une fois le répertoire du projet créé, on peut ouvrir le dossier avec WebStorm.

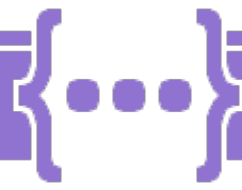


1 Fichier → Ouvrir

Sélectionner le dossier racine du projet et appuyer sur OK.



3 On a facilement accès à tous les fichiers du projet dans **WebStorm** et on peut éditer le code à droite.



❖ Ouvrir un projet



- ◆ Notez que dès qu'un projet / répertoire est ouvert avec WebStorm, un petit dossier nommé « .idea » est généré automatiquement. **Pas de panique** : c'est normal et il ne faut pas le toucher !
 - Ce dossier contient des paramètres, préférences et configurations pour le projet Web.



❖ Introduction aux **fonctions**

- ◆ Qu'est-ce qu'une **fonction**, grossièrement ?
- ◆ Où déclarer / mettre les fonctions ?



❖ Qu'est-ce qu'une **fonction**, grossièrement ? 🤔

- ◆ Pour le moment, disons que c'est un « **morceau de code** » qui peut être réutilisé à volonté sans qu'on ait à le réécrire en entier à chaque fois.

◆ Exemple

- Ici, on a une **fonction** nommée « **message** » qui contient un morceau de code réutilisable qui modifie le **contenu textuel** de l'élément avec l'id **log**.

Ce mot-clé sert à **déclarer** une fonction

Ceci est le **nom** de la fonction. C'est ce qui l'identifie.

Le morceau de code réutilisable est situé entre des **accolades** { ... }

```
function message() {  
    document.getElementById("log").textContent = "Il est " + new Date().toString().slice(0,8);  
}
```

```
<div id="log">Il est 19:55:47.</div>
```

Il est 19:55:47.

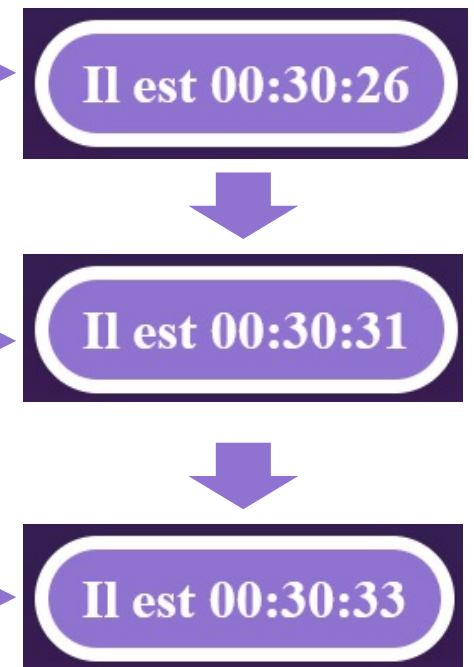


❖ Exemple sans fonction

- ◆ On souhaite écrire l'heure dans l'élément avec l'id « **log** ».
 - Imaginons qu'on le fait sans utiliser de fonction. À chaque fois qu'on veut mettre l'heure à jour, on doit réécrire `document.getElementById("log").textContent = ...` en entier !

```
>> document.getElementById("log").textContent = "Il est " + new Date().toString().slice(0,8);  
← "Il est 00:30:26"  
  
>> document.getElementById("log").textContent = "Il est " + new Date().toString().slice(0,8);  
← "Il est 00:30:31"  
  
>> document.getElementById("log").textContent = "Il est " + new Date().toString().slice(0,8);  
← "Il est 00:30:33"  
  
>> |
```

Pas besoin de comprendre ça !
C'est juste pour afficher la date de
manière lisible.



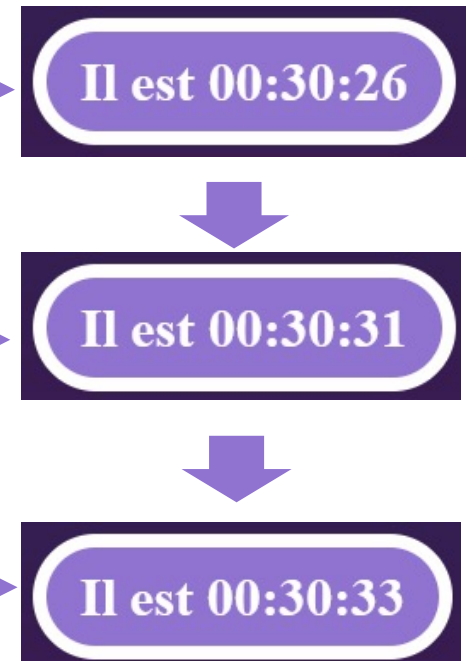
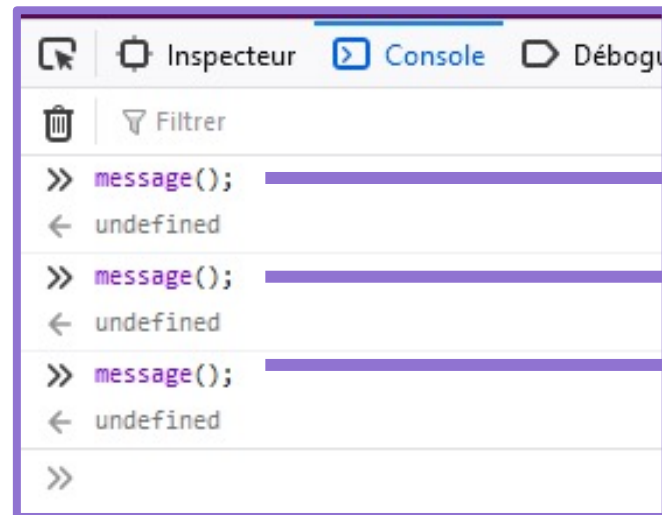


❖ Exemple avec fonction

- ◆ Quand on veut modifier le contenu textuel de l'élément **log** pour indiquer l'heure actuelle, on « appelle » la fonction nommée « **message** ».

```
function message(){  
    document.getElementById( elementId: "log").textContent = "Il est " + new Date().toString().slice(0,8);  
}
```

Pour **appeler une fonction**, on doit simplement écrire son **nom**, suivi de **parenthèses vides**.



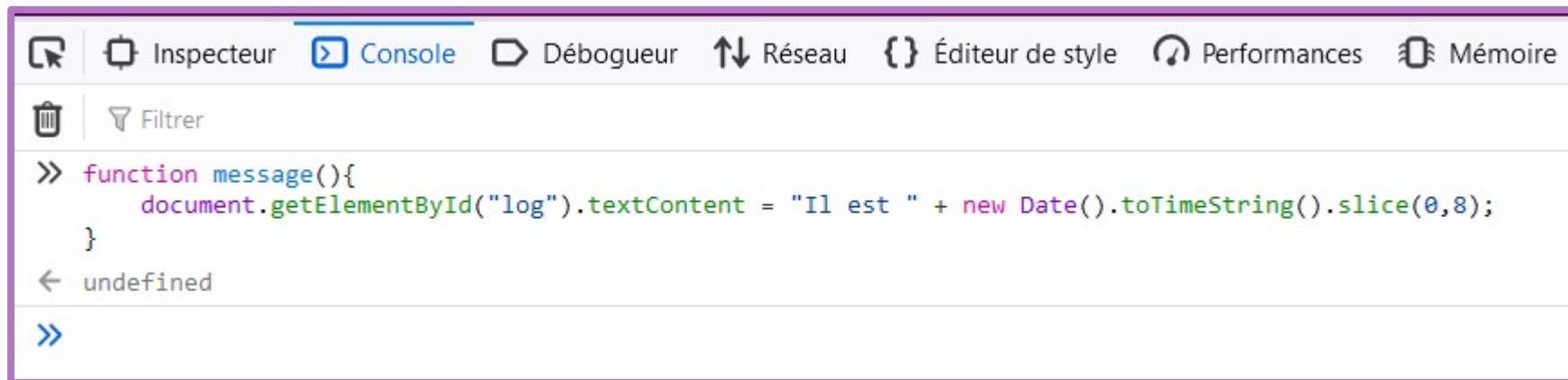


❖ Où déclarer la fonction ?

- ◆ Où faut-il écrire le morceau de code qui sert à déclarer la fonction ?

```
function message(){  
    document.getElementById( elementId: "log").textContent = "Il est " + new Date().toString().slice(0,8);  
}
```

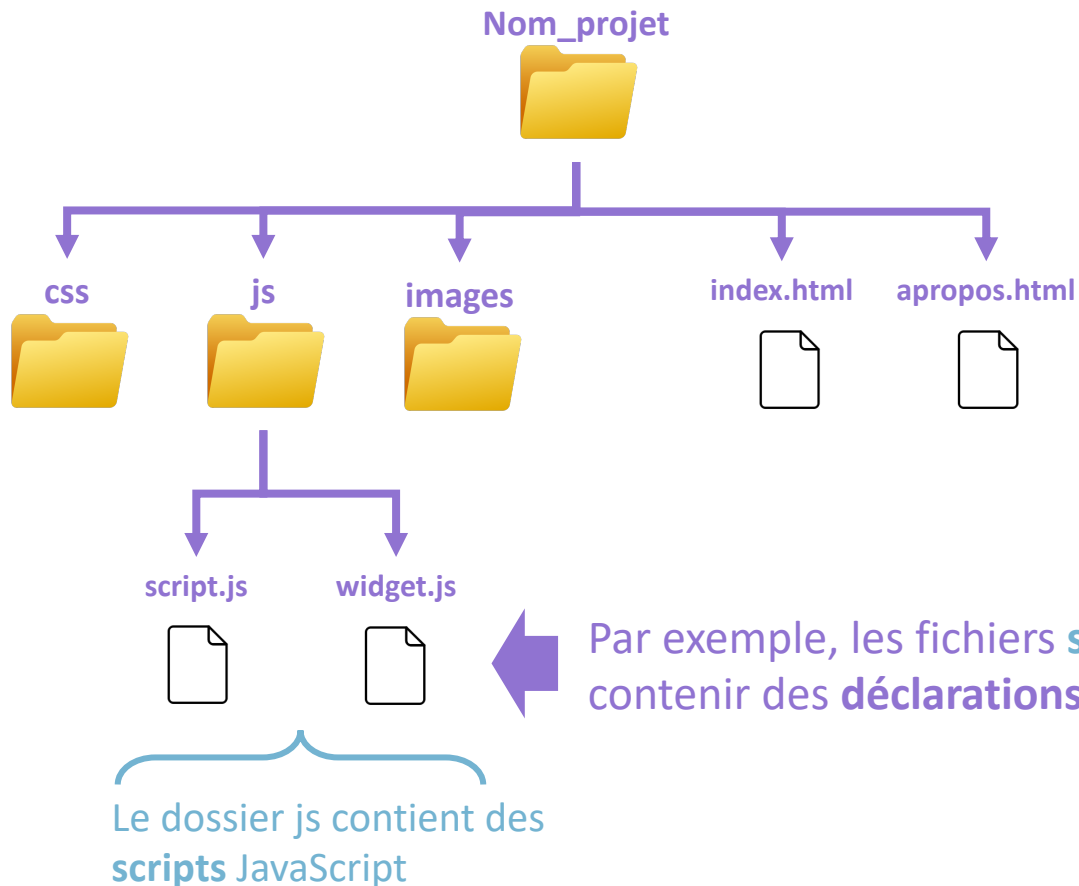
- Si on écrit la déclaration de fonction dans la **console** du navigateur.. la fonction n'existera plus quand nous réactualiserons la page. 😞





❖ Où déclarer la fonction ?

- ◆ La fonction doit être déclarée dans un fichier avec l'extension **.js**, dans le dossier **js** de notre **projet Web**.



```
1 function message(){
2     document.getElementById( elementId: "log").textContent =
3         "Il est " + new Date().toString().slice(0,8);
4 }
```

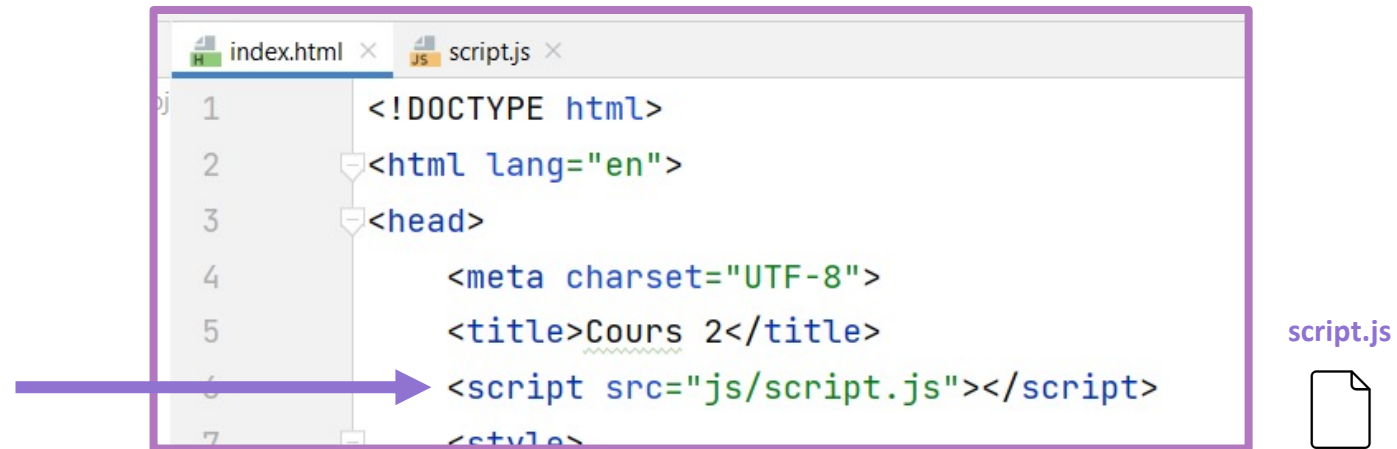
Ceci est un aperçu du fichier **script.js**, qui contient une déclaration de fonction.

Par exemple, les fichiers **script.js** et **widget.js** pourraient contenir des **déclarations de fonction** en JavaScript !



❖ Où déclarer la fonction ?

- ◆ De plus, il faut ajouter une ligne de code HTML dans la page Web où l'on souhaite pouvoir utiliser cette fonction :



- ◆ La portion « **script.js** » correspond au nom du fichier qui contient la / les déclaration(s) de fonction.