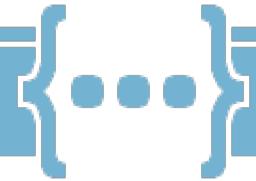


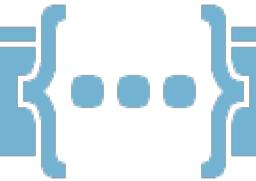
Semaine 2

Chaînes de caractères, DOM et introduction aux fonctions

Intro. à la programmation - Aut. 2022



- ❖ Chaînes de caractères
 - ◆ Affectation
 - ◆ Concaténation
- ❖ Introduction au DOM
 - ◆ QuerySelector
 - ◆ textContent
- ❖ Visual Studio Code
 - ◆ Structure d'un projet
 - ◆ Ouvrir / travailler sur un projet
- ❖ Introduction aux fonctions
- ❖ Petits plus
 - ◆ Commentaires en JavaScript
 - ◆ console.log() et alert()



❖ Chaînes de caractères

- ◆ Jusqu'ici, nous avons affecté des **nombres entiers** et des **nombres à virgule** à nos variables :

```
» let age = 17;  
let prixSalade = 9.99;
```

- ◆ On peut également affecter des « **chaînes de caractères** » à des variables :

```
» let commentaire1 = "Le cours est plate";  
let commentaire2 = "Sus";  
let commentaire3 = "420 chiens chassent 69 chats";
```

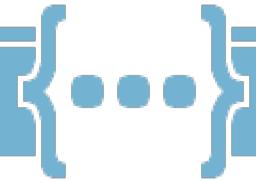
- ◆ Les **chaînes de caractères** sont **TOUJOURS** encadrées par des **guillemets** doubles (" ... ") ou simples (' .. '). Cela permet de les différencier des **noms de variables** ou des **nombres**.

```
» let commentaire4 = Salut; ← ❌
```

! ► Uncaught ReferenceError: Salut is not defined
<anonymous> debugger eval code:1

[En savoir plus]

Si on oublie les guillemets "...", ça ne marche pas ! 😞



❖ Chaînes de caractères

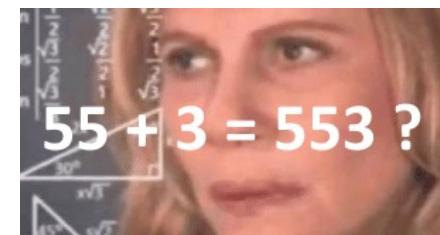
- ◆ Les deux variables ci-dessous contiennent des valeurs **complètement différentes** ! Même si `55` et `"55"` semblent identiques, `55` est un nombre et `"55"` est une chaîne de caractères composées du caractère "5" deux fois.

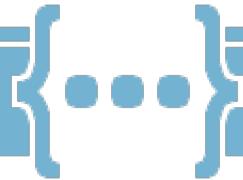
```
» let pasUnNombre = "55";
  let unNombre = 55;
```

- ◆ Qu'est-ce que ça change ?
 - Si on tente de faire des opérations mathématiques avec la variable `pasUnNombre`, on pourrait avoir des petites surprises ...

```
» pasUnNombre + 3
← "553"
```

```
» pasUnNombre + unNombre
← "5555"
```





❖ Concaténation

- ◆ L'opérateur **+** fonctionne différemment dès qu'une donnée de type **chaîne de caractères** fait partie de l'équation

```
» 5 + 5  
← 10
```

Si on additionne deux **nombre**s, une opération mathématique est faite.

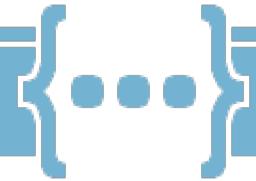
```
» 5 + "salut"  
← "5salut"
```

Si on additionne un **nombre** avec une **chaîne de caractères**, les deux valeurs sont tout simplement concaténées l'une à la suite de l'autre pour former une nouvelle **chaîne de caractères**.

```
» 5 + "3"  
← "53"
```

```
» "pika" + "chu"  
← "pikachu"
```

Bien entendu, si on additionne deux **chaînes de caractères**, elles sont concaténées également.



❖ Concaténation avancée

- ◆ On peut former des phrases en concaténant des chaînes de caractères

```
» let message = "Salut";
  let nom = "Simone";
← undefined

» message + nom
← "SalutSimone"
```

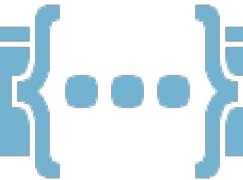
- Hmm... j'aurais préféré qu'il y ait une **espace** entre "Salut" et "Simone" ...

```
» "Salut" + " " + "Simone"
← "Salut Simone"
```

La valeur " " contient un seul caractère : une **espace** !
On a donc **concaténé** une espace entre les 2 autres mots.

```
» message + " " + nom
← "Salut Simone"
```

On peut également utiliser les variables déclarées plus haut et concaténer une espace " " entre les deux.



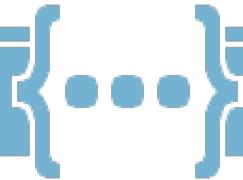
❖ Concaténation avancée

- ◆ On peut former des phrases en concaténant des chaînes de caractères

```
» let debutMessage = "Salut ";
  let finMessage = ", comment ça va ?";
  let nom = "Simone";
← undefined

» debutMessage + nom + finMessage
← "Salut Simone, comment ça va ?"
```

- Remarquez l'espace à la fin de "Salut " ! Sinon le message aurait contenu SalutSimone au lieu de Salut Simone.



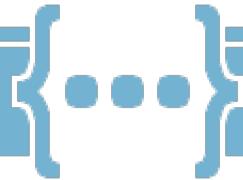
❖ Concaténation avancée

- ◆ Exemple de phrase encore plus complexe impliquant un calcul !

```
» let prixChat = 30;
  let nbChats = 4;
  let message = "Le prix pour " + nbChats + " chats est de " + (prixChat * nbChats) + " dollars.";
← undefined
» message
← "Le prix pour 4 chats est de 120 dollars."
```

"Le prix pour " + nbChats + " chats est de " + (prixChat * nbChats) + " dollars."

- ◆ Les parenthèses nous aident à garantir qu'on va effectuer le calcul entre **prixChat** et **nbChats** au lieu de concaténer les deux nombres.
 - Dans ce cas-ci, ce n'était pas obligatoire car le ***** a la priorité sur les **+**.



❖ Concaténation avancée

◆ Opérateur `+=`

- Avec des **nombres** : On augmente la valeur d'une variable.

On augmente **x** de 3, **x** contient maintenant 7.

```
» let x = 4;  
← undefined  
» x += 3;  
← 7
```

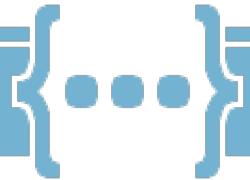
```
» let y = 7;  
← undefined  
» y += 3 * 2;  
← 13
```

On augmente **y** de $3 * 2$ (donc 6), **y** contient maintenant 13.

- Avec des **chaînes de caractères** : On ajoute du texte à la fin de la chaîne.

```
» let mot = "ni";  
← undefined  
» mot += "che";  
← "niche"
```

On ajoute les caractères "che" à la fin de la chaîne "ni". La variable **mot** contient maintenant "niche".



❖ Concaténation avancée

◆ Opérateur `+=`

- D'autres exemples de concaténation

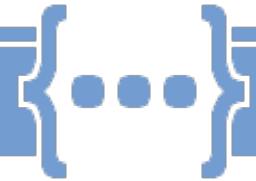
```
» let phrase = "Salut ";
← undefined

» phrase += "Simone.";
← "Salut Simone."

» phrase += " Quelle heure est-il ?"
← "Salut Simone. Quelle heure est-il ?"
```

```
» let phrase2 = "Regarde ";
let nom = "Claude";
← undefined

» phrase2 += nom + ". Il est tellement normal !";
← "Regarde Claude. Il est tellement normal !"
```



❖ Ouvrir une page Web

- ◆ À partir de maintenant, pour tester certaines notions, nous utiliserons nos propres pages Web.
- ◆ Pour les ouvrir dans le navigateur, choisissez un fichier **.html** et faites **clic-droit** -> **Ouvrir avec** -> **Firefox (ou Chrome)**

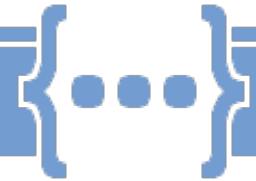


- ◆ Une fois la page ouverte, nous pourrons utiliser la **console** du navigateur, comme d'habitude.



❖ 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** ».

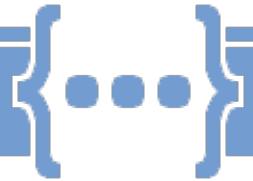
- Élément sans **id** :

```
<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 le modifier dans le code JavaScript :

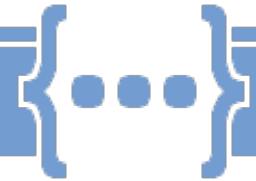
```
<div id="pikachu"> Pick a shoe </div>
```



```
document.querySelector("#pikachu").textContent = "Pikachu";
```

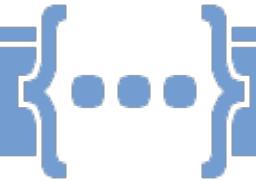
Ce bout de code veut dire « Pour l'élément dont l'id est **pikachu** », ...

- ◆ Pour utiliser le `<div>` (ex : le modifier, obtenir son contenu textuel, etc.), on utilise son **id**, qui ici est "**pikachu**".



❖ 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.



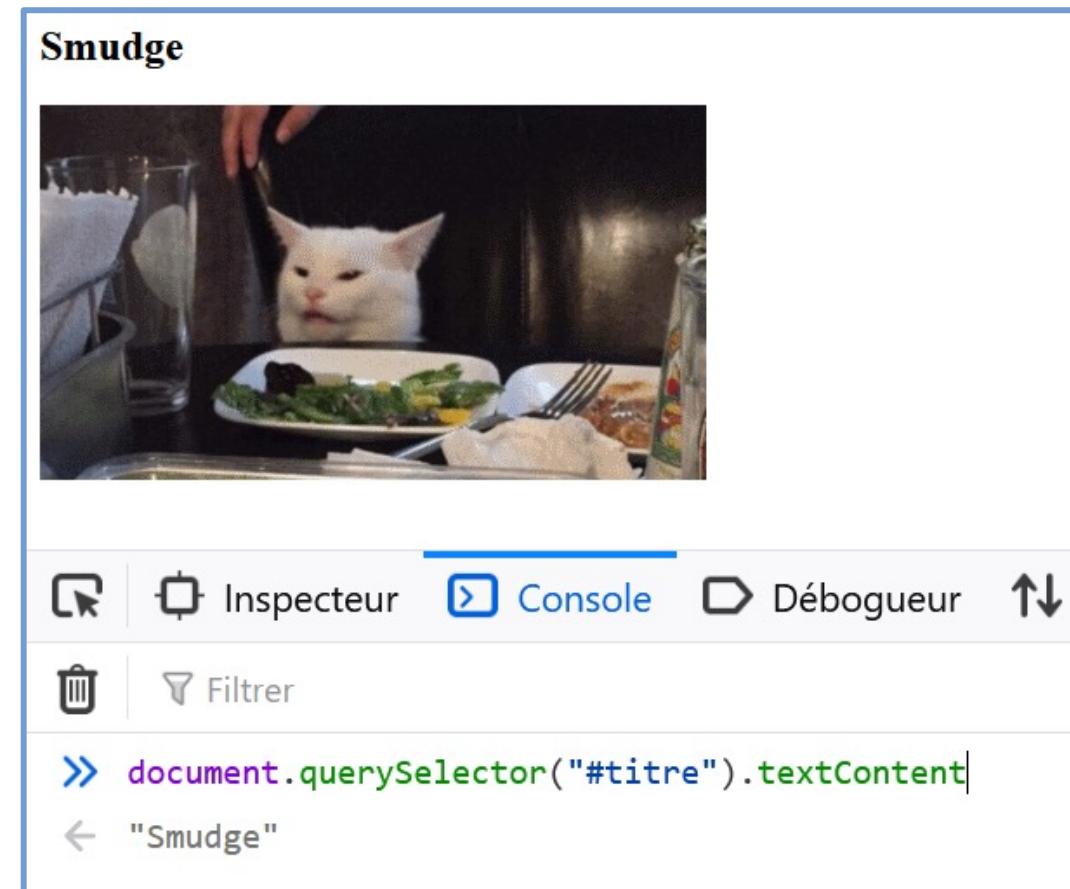
❖ Obtenir le contenu textuel

◆ `document.querySelector("#id").textContent`

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

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

```





❖ Modifier le contenu textuel

◆ `document.querySelector("#id").textContent = "nouveau texte";`

- Ici, on veut changer le titre « **Smudge** » pour « **Chat consterné** ». On doit utiliser l'**id** **titre** pour le faire.

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

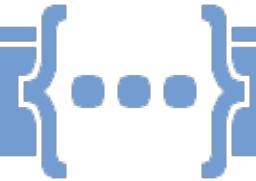
```

Smudge



Chat consterné





❖ Modifier le contenu textuel

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

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

```

Smudge

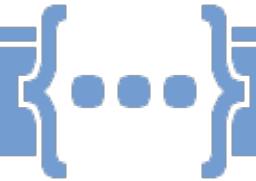


The screenshot shows the browser's developer tools with the 'Console' tab selected. The console output is as follows:

```
Inspecteur Console Débogueur Réseau Éditeur
Filtrer
» document.querySelector("#titre").textContent = "Chat consterné";
← "Chat consterné"
»
```

Chat consterné





❖ Modifier le contenu textuel

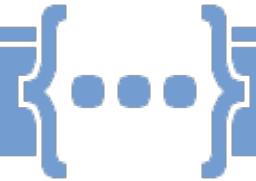
◆ `document.querySelector("#id").textContent += "texte supplémentaire";`

- Ici, on remarque l'usage de `+=` au lieu de `=`. Cet opérateur va permettre, bien entendu, d'ajouter du texte sans remplacer le texte déjà présent.

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

```





❖ Modifier le contenu textuel

- ◆ Plutôt que d'affecter une **chaîne de caractères** au **contenu textuel**, on peut également affecter n'importe quelle **variable** pour afficher sa valeur dans la page Web :

```
» let maVariable = "Imbroglio";
  document.querySelector("#titre").textContent = maVariable;
```

Ça peut même être un nombre !

```
» let maVariable = 5114;
  document.querySelector("#titre").textContent = maVariable;
```

Imbroglio

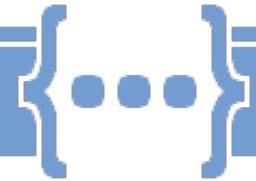
est le meilleur chat.



5114

est le meilleur chat.





❖ Modifier le contenu textuel

- ◆ Tentons quelque chose d'un peu plus complexe.

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

```



- 1) On va récupérer le nom du chat (« Smudge ») pour le mettre dans une variable :

```
» let nomChat = document.querySelector("#titre").textContent;
← undefined

» nomChat
← "Smudge"
```



❖ Modifier le contenu textuel

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

```



- 2) Nous allons récupérer le texte dans l'élément `#description` pour l'affecter à une variable également.

```
» let phrase = document.querySelector("#description").textContent;
← undefined
» phrase
← "est le meilleur chat."
```



❖ Modifier le contenu textuel

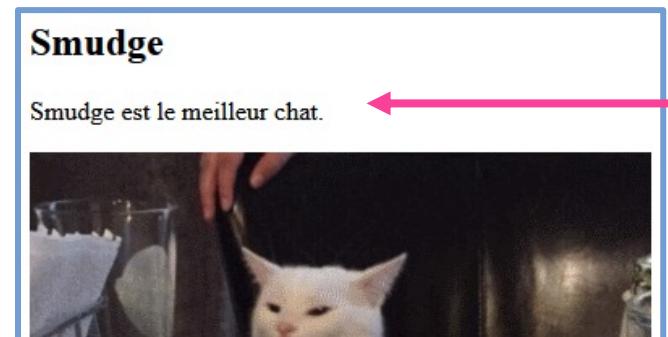
```
<h2 id="titre">Smudge</h2>
<p id="description">est le meilleur chat.</p>

```

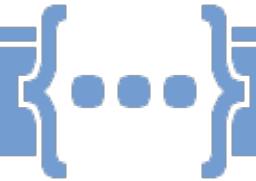


- 3) Finalement, nous allons modifier le texte de l'élément #description pour y inclure le nom du chat grâce à la concaténation 😊

```
» document.querySelector("#description").textContent = nomChat + " " + phrase;
← "Smudge est le meilleur chat."
```



Le texte dans la page a bel et bien changé.



❖ Modifier le contenu textuel

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

```



The screenshot shows a browser developer tools console with the following content:

```
Inspecteur  Console  Débogueur  Réseau  Éditeur de style
Filtrer

» let nomChat = document.querySelector("#titre").textContent;
← undefined

» let phrase = document.querySelector("#description").textContent;
← undefined

» document.querySelector("#description").textContent = nomChat + " " + phrase;
← "Smudge est le meilleur chat."
```

Smudge

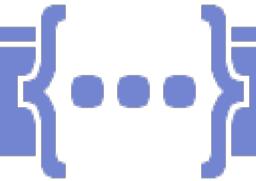
est le meilleur chat.



Smudge

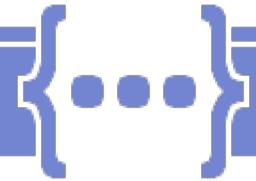
Smudge est le meilleur chat.





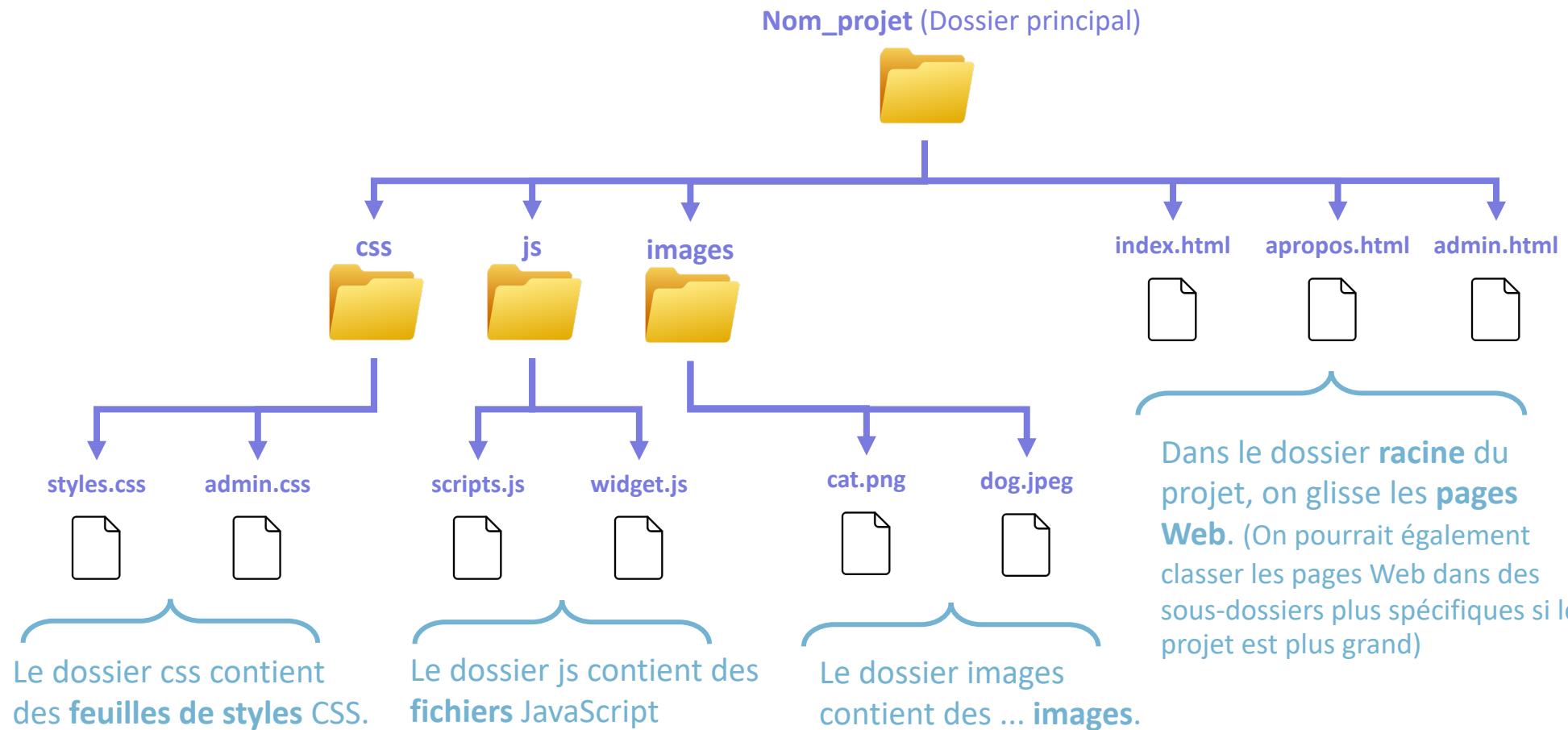
❖ Visual Studio Code et WebStorm

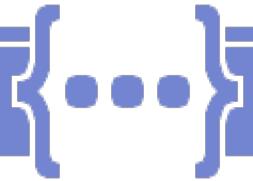
- ◆ « Éditeurs de code » pour plein de langages, dont HTML / CSS / JavaScript
 - Outils qui nous aideront à créer des projets Web avec JavaScript
- ◆ Visual Studio Code est gratuit 😊
 - Il est déjà installé sur les ordinateurs du cégep
 - Vous pouvez l'installer à la maison sans problème : [Lien de téléchargement](#)
- ◆ WebStorm est payant 😱, mais il est disponible sur les ordinateurs du cégep
 - À la maison, il est préférable d'utiliser Visual Studio Code. En classe, nous utiliserons Webstorm et/ou Visual Studio Code.



❖ Créer un nouveau projet Web

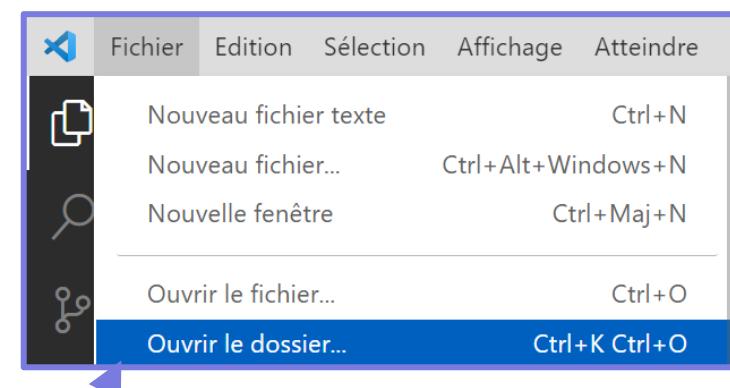
- ◆ Il faut respecter l'arborescence suivante pour le dossier de notre projet



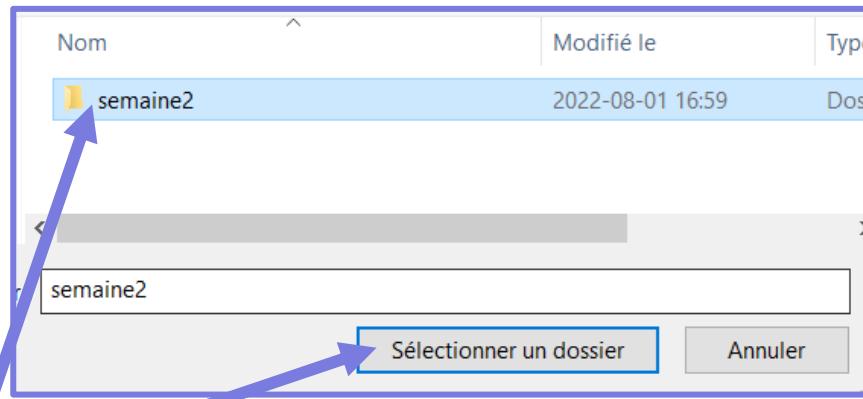


❖ Ouvrir un projet avec VS Code

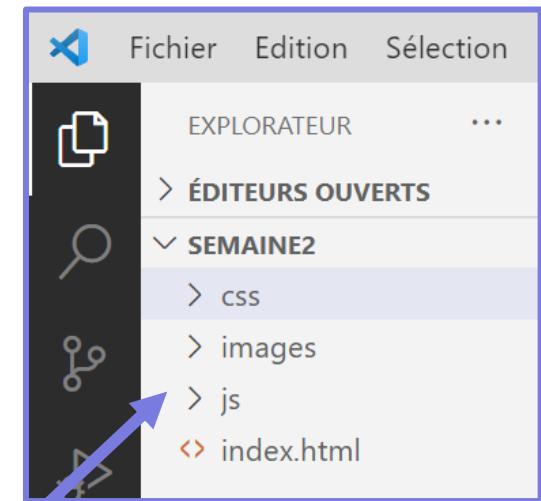
- ◆ Une fois le répertoire du projet créé, on peut ouvrir le dossier avec Visual Studio Code.



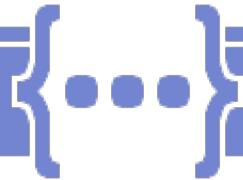
1 Fichier → Ouvrir le dossier...



2 Sélectionner le dossier racine du projet et appuyer sur Sélectionner un dossier

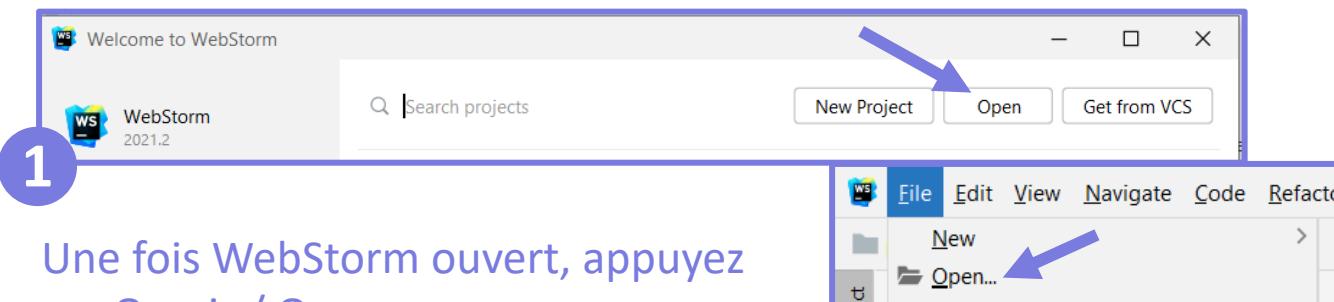


3 On a facilement accès à tous les dossiers / fichiers du projet dans **Visual Studio Code** et on peut éditer le code.



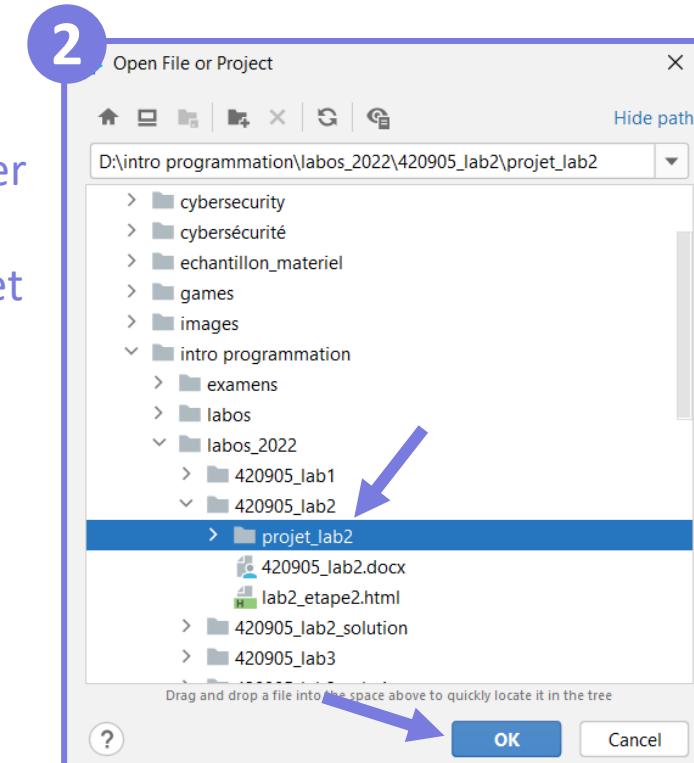
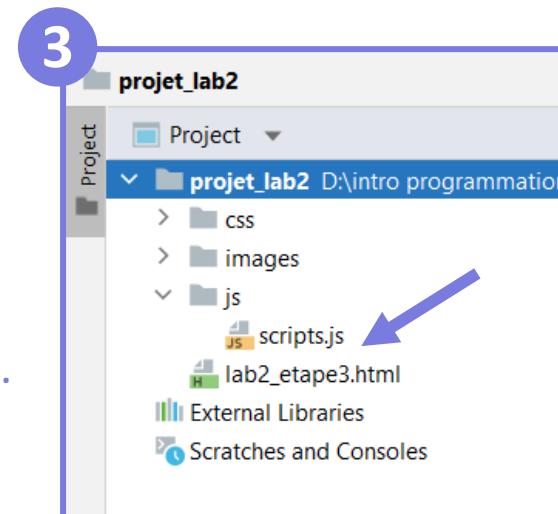
❖ Ouvrir un projet avec WebStorm

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



On a facilement accès à tous les dossiers / fichiers du projet dans WebStorm et on peut éditer le code.

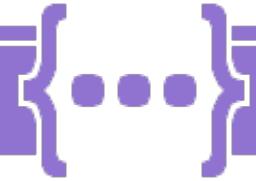
Trouvez le dossier de votre projet, sélectionnez-le et appuyez sur OK.





❖ Introduction aux **fonctions**

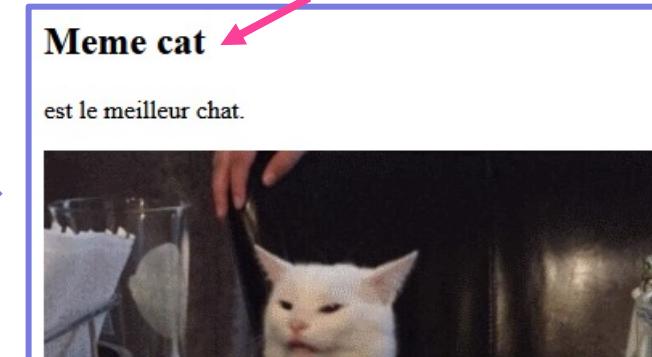
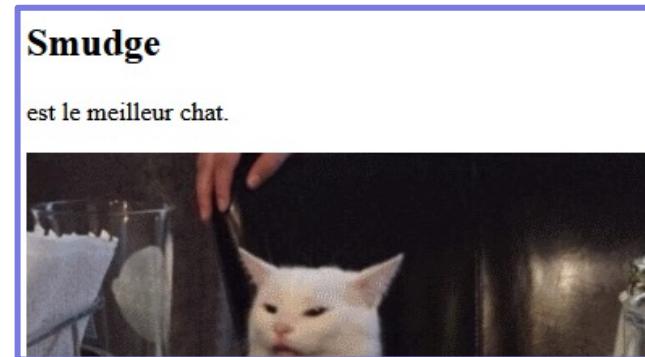
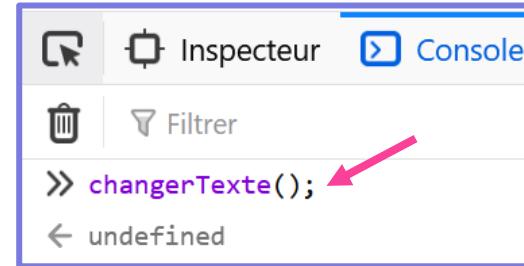
- ◆ Qu'est-ce qu'une **fonction**, grossièrement?
- ◆ Où déclarer une fonction?
- ◆ Créer une fonction



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

◆ Exemple

- Lorsqu'on écrit « `changerTexte()` » dans la console, le texte du titre change ! 😱



- Comment c'est possible? Nous n'avons même pas utilisé `.querySelector` ou `.textContent` !



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

◆ Déclarer une fonction

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

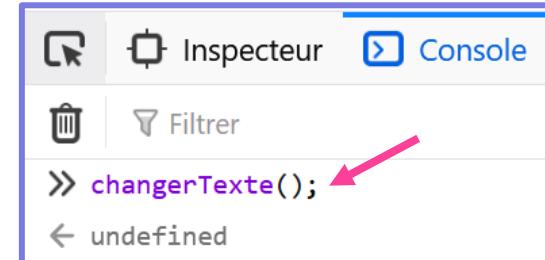
Ce mot-clé sert à **déclarer** une fonction

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

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

```
function changerTexte(){  
  document.querySelector("#titre").textContent = "Meme cat";  
}
```

- Le fait de « **déclarer** » cette fonction va nous permettre de « **l'appeler** » dans la console comme on l'a vu dans la diapositive précédente.





❖ Où déclarer la fonction ?

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

```
function changerTexte(){  
  document.querySelector("#titre").textContent = "Meme cat";  
}
```

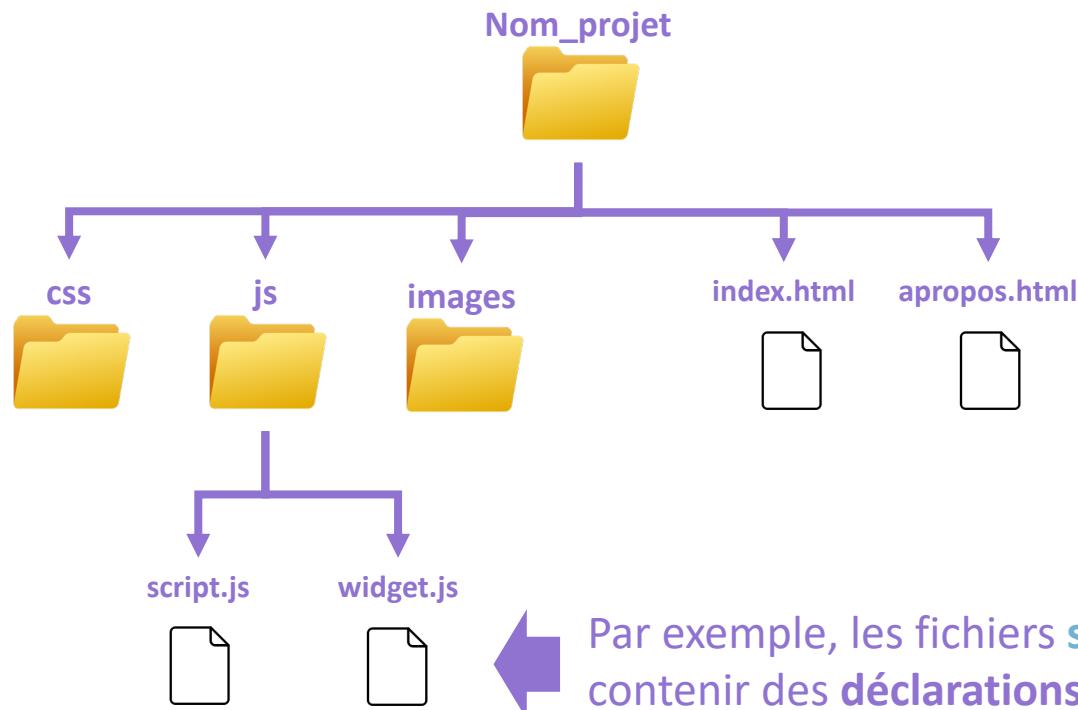
- Si on déclare la fonction dans la **console** du navigateur... la fonction n'existera plus quand nous réactualiserons la page. 😢 Pas très pratique.





❖ Où déclarer la fonction ?

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



A screenshot of a code editor showing a file named 'scripts.js'. The code is as follows:

```
js > JS scripts.js > ...
1  function changerTexte(){
2      |   document.querySelector("#titre").textContent = "Meme cat";
3  }
```

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 :

The image shows a code editor with two tabs: 'index.html' and 'script.js'. The 'index.html' tab is active, displaying the following code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Cours 2</title>
<script src="js/script.js"></script>
<style>
```

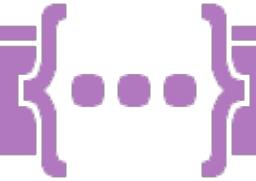
A purple arrow points from the line containing the `<script>` tag in the `index.html` code to the `script.js` file icon on the right. The `script.js` file icon is a white document with a blue outline.

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



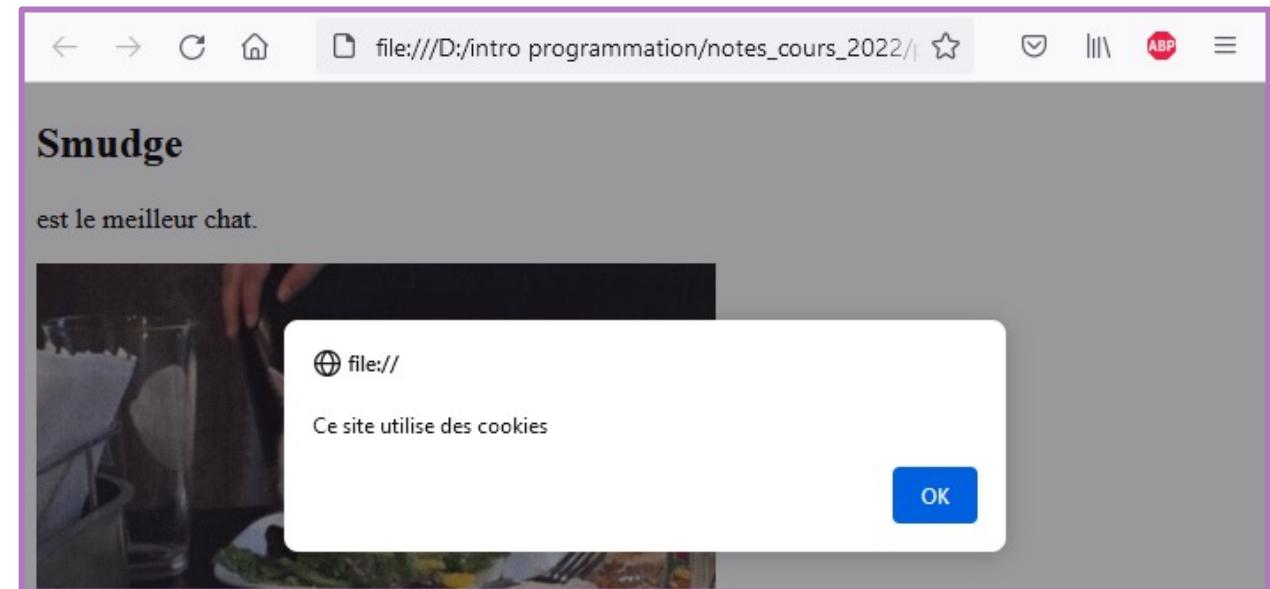
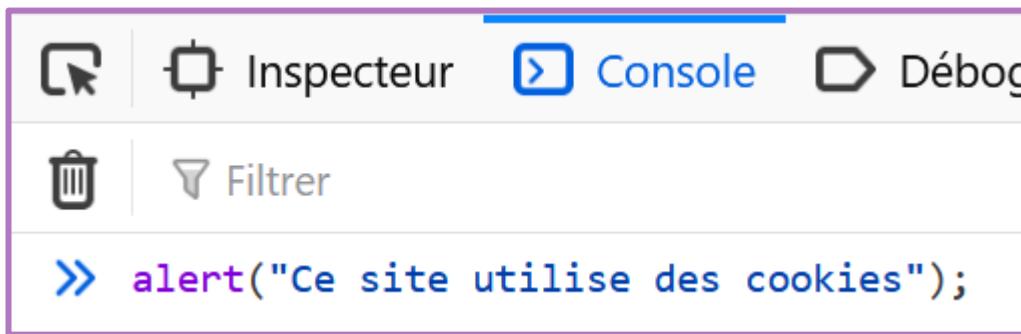
❖ Fonctions préexistantes

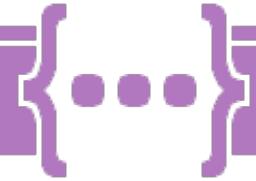
- ◆ Certaines **fonctions** existent déjà par défaut en JavaScript. Nous n'avons donc pas besoin de les **déclarer** nous-mêmes et on peut les utiliser n'importe quand.
- ◆ Quelques exemples
 - `document.querySelector()`
 - `alert()`
 - `console.log()`
- ◆ On connaît déjà `querySelector()`. Dans les prochaines diapositives, nous abordons `alert()` et `console.log()`.



❖ alert()

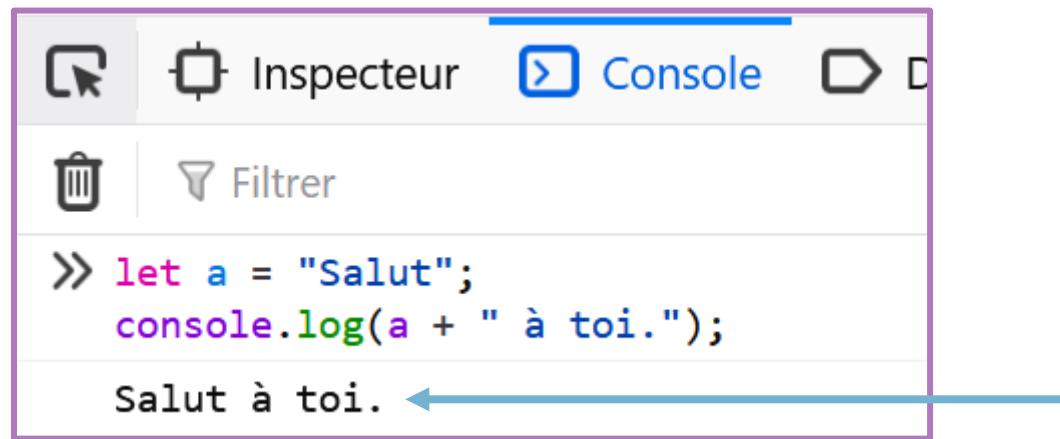
- ◆ La fonction **alert()** permet de créer un « pop-up » dans la page avec le message de notre choix.
 - Il suffit d'inclure une **chaîne de caractères** à l'intérieur des **parenthèses**.
 - Exemple :





❖ console.log()

- ◆ La fonction **console.log()** permet simplement d'afficher du texte dans la console du navigateur.
 - Il suffit d'inclure une **chaîne de caractères** ou une variable de notre choix à l'intérieur des **parenthèses**.
 - Exemple :



The screenshot shows a browser developer tools interface with the 'Console' tab selected. The console window contains the following text:
» `let a = "Salut";
console.log(a + " à toi.");`
The output of the code is:
Salut à toi.

La fonction **console.log()** a fait apparaître ce message dans la **console**.



❖ 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 **multiligne** (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 !



❖ Créer une fonction

- ◆ Exemple : J'aimerais coder une fonction qui fait les choses suivantes :

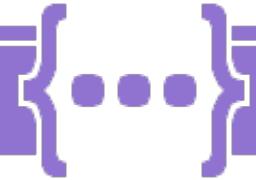
- 1) Remplace le texte « est le meilleur chat. » par « veut manger sa salade en paix. » dans la page.
- 2) Fait un **pop-up** avec le message « Texte changé ! ».
- 3) Affiche le message « Fonction terminée. » dans la **console**.

- ◆ Nous allons construire notre fonction étape par étape dans les prochaines diapositives.

Smudge

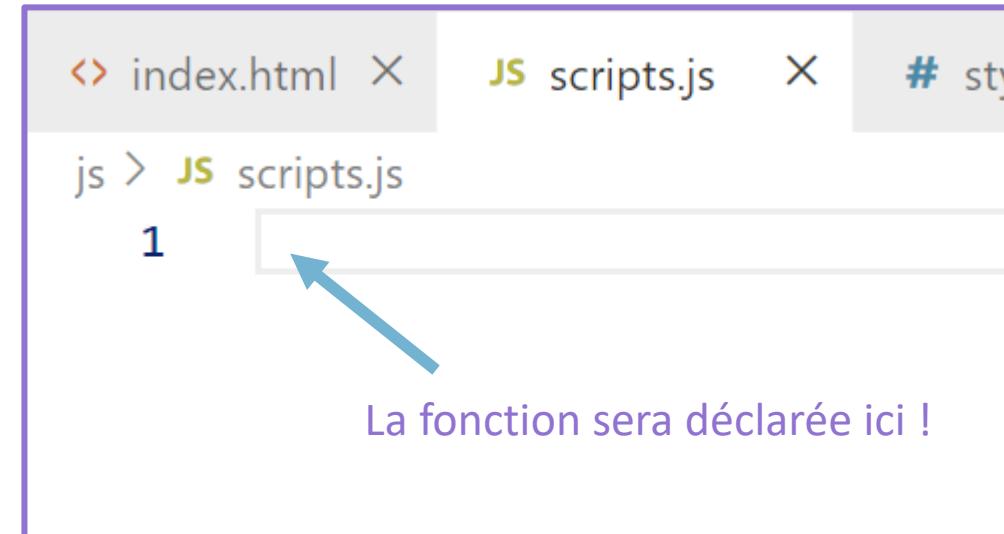
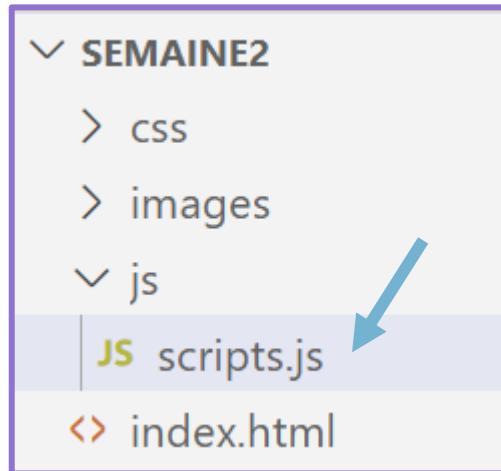
est le meilleur chat.





❖ Créer une fonction

- ◆ **Étape 1** : Trouver le fichier `scripts.js` dans notre projet Web. (Ou le créer s'il n'existe pas)
 - Ce fichier doit être situé dans le dossier « `js` » de notre projet Web.

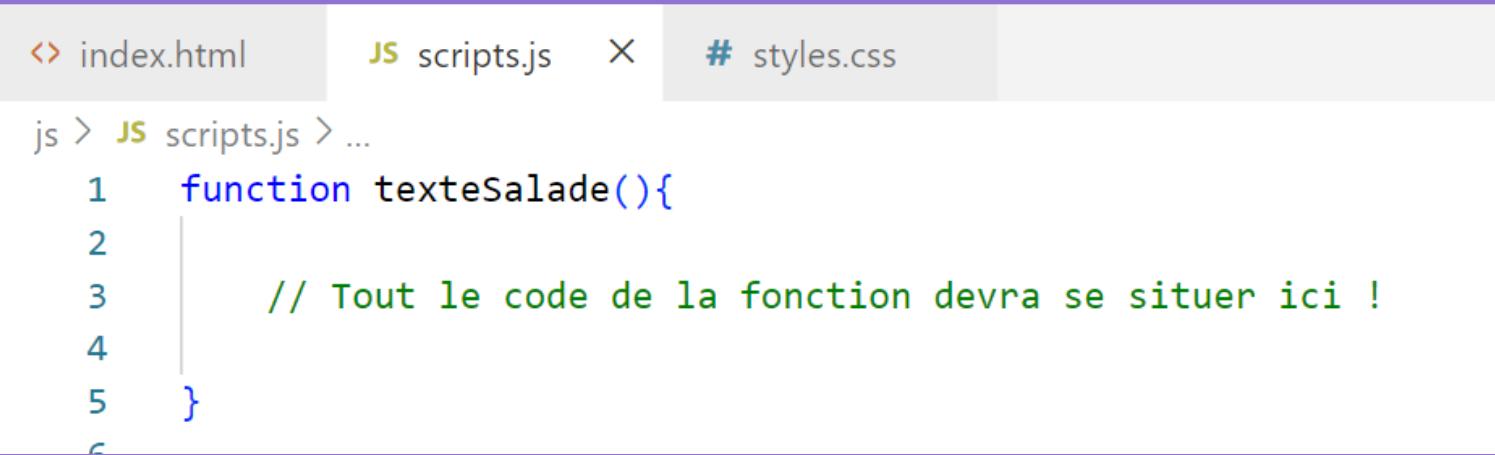




❖ Créer une fonction

◆ Étape 2 : Nommer la fonction et préparer sa structure

- Ici, la fonction a été nommée **texteSalade**. Pour l'instant, la fonction ne fait absolument rien. Il nous reste à ajouter des instructions à l'intérieur.



```
index.html  JS scripts.js  # styles.css
js > JS scripts.js > ...
1  function texteSalade(){
2
3      // Tout le code de la fonction devra se situer ici !
4
5  }
6
```



❖ Créer une fonction

◆ Étape 3 : Rédiger le code de la fonction

- Nous souhaitons que la fonction fasse trois choses :

- 1) Remplacer le texte « est le meilleur chat. » par « veut manger sa salade en paix. » dans la page.
- 2) Faire un **pop-up** avec le message « Texte changé ! ».
- 3) Afficher le message « Fonction terminée. » dans la **console**.

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



On jette un coup d'œil au code HTML pour trouver l'**id** de l'élément dont on souhaite changer le texte.

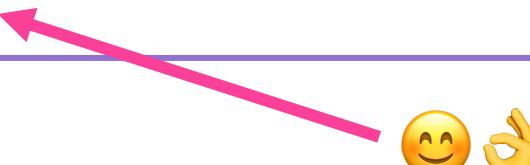
```
function texteSalade(){  
  
    document.querySelector("#description").textContent = "veut manger sa salade en paix.";  
    alert("Texte changé !");  
    console.log("Fonction terminée.");  
}
```



❖ Créer une fonction

- ◆ Étape 4 : S'assurer que la page Web avec laquelle on souhaite utiliser notre fonction est reliée à notre fichier JavaScript :

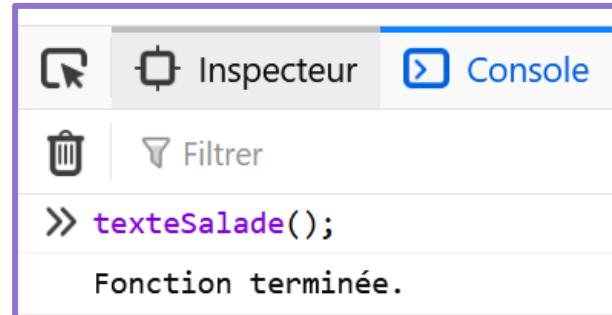
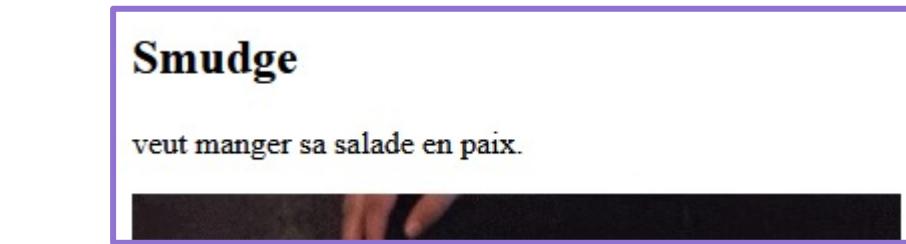
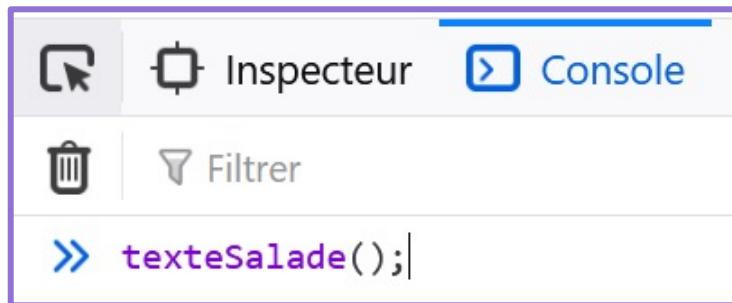
```
↳ index.html X JS scripts.js # styles.css
↳ index.html > ⚒ html > ⚒ body > ⚒ div > ⚒ img
1  <!DOCTYPE html>
2  <html lang="fr">
3  <head>
4  <meta charset="UTF-8">
5  <meta http-equiv="X-UA-Compatible" content="IE=edge">
6  <meta name="viewport" content="width=device-width, initial-scale=1.0">
7  <title>Semaine 2</title>
8  <link rel="stylesheet" href="css/styles.css">
9  <script src="js/scripts.js"></script>
10 </head>
```





❖ Créer une fonction

- ◆ **Étape 5 : Tester la fonction !** Il est possible qu'on ait fait des erreurs. Il faut s'assurer qu'elle fonctionne tel que prévu.



1) Le texte de la page a bien été modifié.

2) On a un pop-up avec le texte souhaité.

3) Un message s'affiche dans la console, comme prévu.