

# Semaine 2

---

Chaînes de caractères, DOM et introduction aux fonctions

[Intro. à la programmation](#)



- ❖ 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** » (mots, phrases...) à des variables :

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

- ◆ Les **chaînes de caractères** sont encadrées par des **guillemets** doubles (" ... ") ou simples (' .. ') ou des accents graves (` ... `) 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 composée du caractère "5" deux fois.

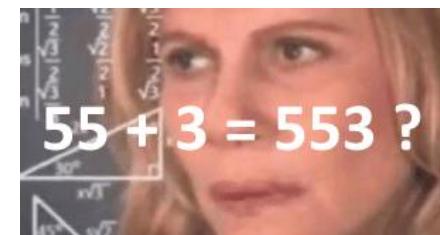
```
>> let pasUnNombre = "55";
      let unNombre = 55;
```

- ◆ 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.

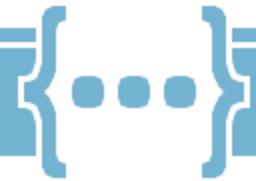


## ❖ Littéraux de gabarits (Template strings)

- ◆ Les **littéraux de gabarits** permettent de glisser la valeur d'une variable (ou un calcul) dans une chaîne de caractères.
- ◆ Exemple SANS template string :

```
» let nb = 5;
← undefined
» let phrase = "J'ai nb conjoints";
← undefined
» phrase
← "J'ai nb conjoints"
```

- Hélas, nous sommes fâchés 😡 de ce résultat : nous souhaitons que la variable **nb** soit remplacée par sa valeur dans la phrase, c'est-à-dire **5**.
- La phrase souhaitée est "**J'ai 5 conjoints**".



## ❖ Littéraux de gabarits (Template strings)

- ◆ Les **littéraux de gabarits** permettent de glisser la valeur d'une variable (ou un calcul) dans une chaîne de caractères.

- ◆ Exemple AVEC **template string** :

```
» let nb = 5;  
← undefined  
» let phrase = `J'ai ${nb} conjoints`;  
← undefined  
» phrase  
← "J'ai 5 conjoints"
```

- Hourra 😊, **nb** a été remplacée par sa valeur, c'est-à-dire **5**. On obtient la phrase voulue.
- Remarquez que la variable **nb** est encadrée par **`\${...}`**. Ça permet de dire à JavaScript « remplace le **nom de la variable** par la **valeur** qu'elle contient ».



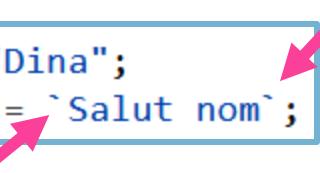
## ❖ Littéraux de gabarits (Template strings)

- ◆ Lorsqu'on construit un **template string**, il y a 2 étapes importantes :
  - 1 - Remplacer les guillemets "..." par des accents graves ` ... `

```
let nom = "Dina";
let phrase = "Salut nom";
```



```
let nom = "Dina";
let phrase = `Salut nom`;
```



- 2 - Encadrer les noms de variables avec \${ ... }

```
let nom = "Dina";
let phrase = `Salut nom`;
```



```
let nom = "Dina";
let phrase = `Salut ${nom}`;
```



```
» let nom = "Dina";
  let phrase = `Salut ${nom}`;
← undefined

» phrase
← "Salut Dina" 
```



## ❖ Littéraux de gabarits (Template strings)

- ◆ Un template string peut contenir plusieurs variables :

```
» let nom = "Roland";
  let objet = "chaises";
  let phrase = `Je suis ${nom} et j'aime les ${objet}`;
← undefined
» phrase
← "Je suis [Roland] et j'aime les [chaises]"
```

- ◆ Un template string peut contenir des calculs :

```
» let article = "rhododendron";
  let qte = 3;
  let prix = 3.99;
  let phrase = `${qte} ${article} coûtent ${qte * prix} dollars.`;
← undefined
» phrase
← "[3] [rhododendron] coûtent [11.97] dollars."
```



## ❖ Concaténation

### ◆ 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 = "caca";  
← undefined  
» mot += "huète";  
← "cacahuète"
```

On ajoute les caractères "huète" à la fin de la chaîne "caca". La variable **mot** contient maintenant "cacahuète".





## ❖ Concaténation

### ◆ 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.



## ❖ Le **DOM** (Document Object Model)

- ◆ 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 « **classes** ».

- Élément sans **classe** :

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

- Élément avec une **classe** :

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

- Ci-dessus, on dit que « La **classe** de cet élément HTML est "**gris**" »



## ❖ Utiliser **DOM** avec un élément HTML

- ◆ Nous pouvons nous servir de la **classe** d'un élément HTML afin de pouvoir le modifier dans le code JavaScript :

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



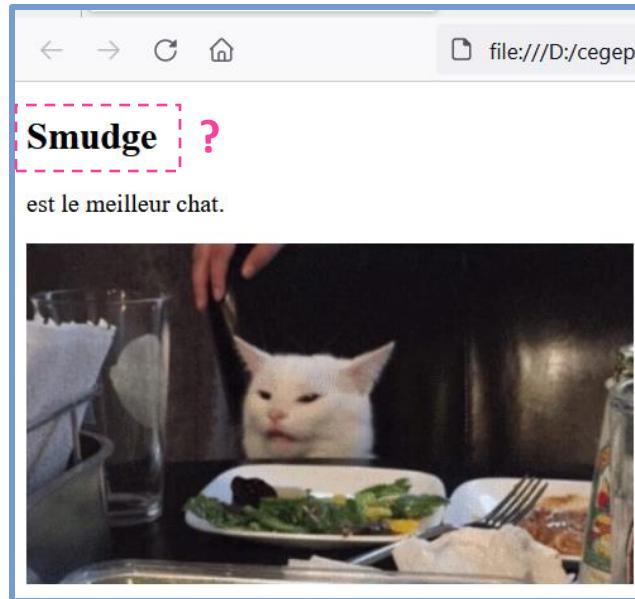
```
document.querySelector(".pikachu").textContent = "Pikachu";
```

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

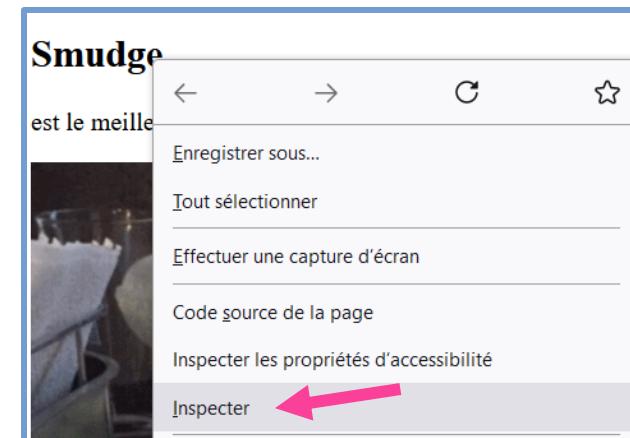
- ◆ Pour utiliser le `<div>` (ex : le modifier, obtenir son contenu textuel, etc.), on utilise sa **classe**, qui ici est "**pikachu**".



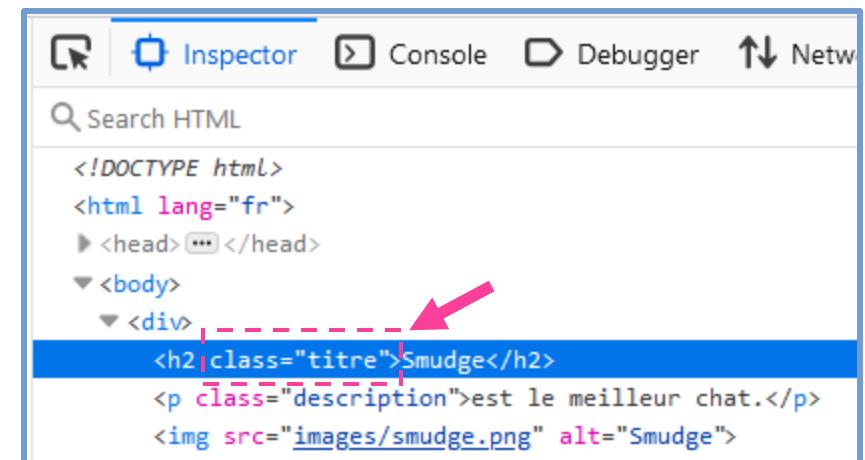
- ❖ Astuce : inspecter la page pour trouver la classe d'un élément
  - ◆ Ex : je me demande quelle est la classe du titre « Smudge ».



1) Faites un clic-droit sur l'élément (sur son texte) et choisissez l'option **Inspecter**



2) Le code HTML de la page sera affiché, mais surtout, l'élément choisi sera surligné (ici, en bleu) et on peut voir sa **classe** !





## ❖ Utiliser le **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 supprime tous les changements faits avec le DOM dans la console.



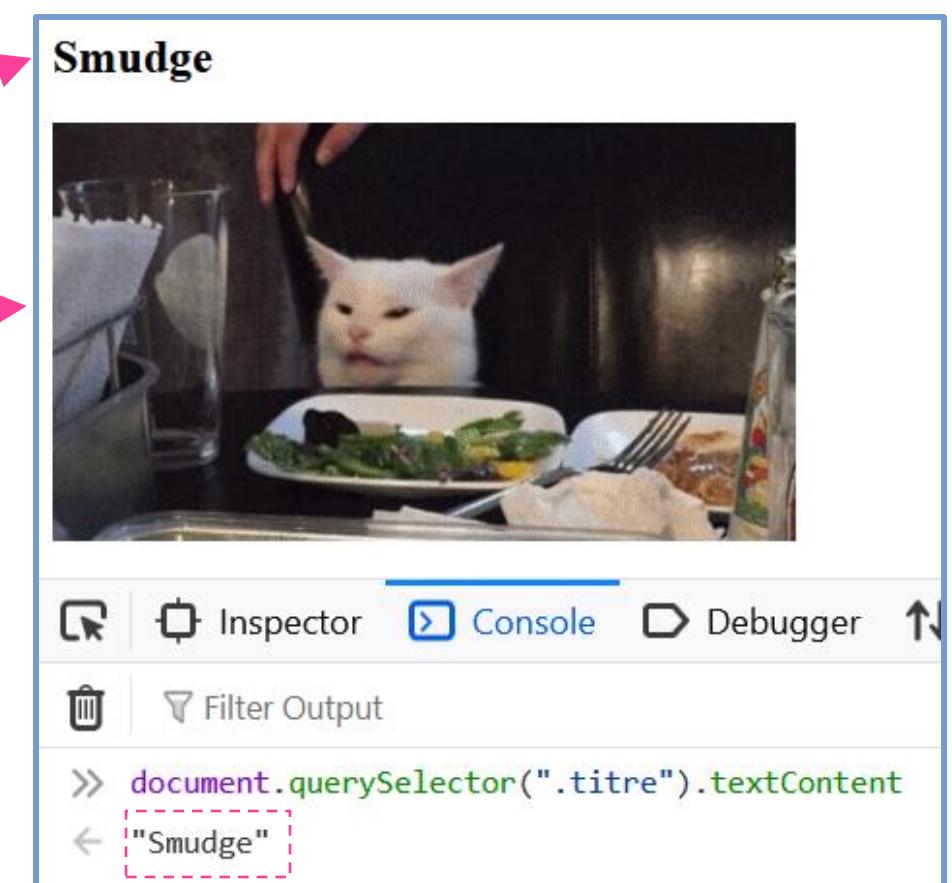
## ❖ Obtenir le contenu textuel

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

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

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

```





## ❖ Modifier le contenu textuel

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

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

```

Smudge



The screenshot shows the browser's developer tools with the 'Console' tab selected. The command `document.querySelector(".titre").textContent = "Chat consterné";` is entered and executed, resulting in the output `← "Chat consterné"`. The 'Console' tab is highlighted with a blue border.

```
Inspector Console Debugger Network Styles
Filter Output
» document.querySelector(".titre").textContent = "Chat consterné";
← "Chat consterné"
» |
```

Chat consterné





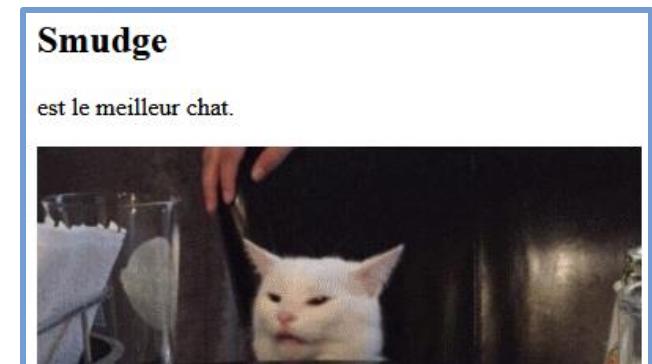
## ❖ Modifier le contenu textuel

◆ `document.querySelector(".classe").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 class="titre">Smudge</h2>
<p class="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 nouveauTitre = "Imbroglio";
  document.querySelector(".titre").textContent = nouveauTitre;
```

Ça peut même être un nombre !

```
» let nouveauTitre = 1109;
  document.querySelector(".titre").textContent = nouveauTitre;
```

**Imbroglio**

est le meilleur chat.



**1109**

est le meilleur chat.





## ❖ Modifier le contenu textuel

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

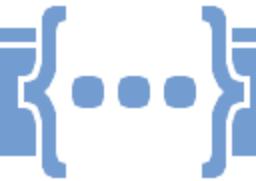
```
<h2 class="titre">Smudge</h2>
<p class="description">est le meilleur chat.</p>

```



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

```
Inspector Console Debugger Network {  
  Filter Output  
  » let nomChat = document.querySelector('.titre').textContent;  
  ← undefined On vérifie ce que contient la variable  
  » nomChat ← "Smudge" nomChat... et c'est bon !  
  ← "Smudge"
```



## ❖ Modifier le contenu textuel

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

```



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



```
Inspector Console Debugger Network Styles
Filter Output
» let phrase = document.querySelector(".description").textContent;
← undefined
» phrase
← "est le meilleur chat."
```



## ❖ Modifier le contenu textuel

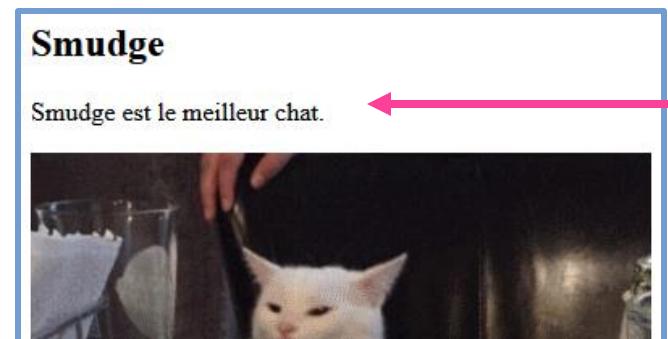
```
<h2 class="titre">Smudge</h2>
<p class="description">est le meilleur chat.</p>

```



- 3) Finalement, nous allons modifier le texte de l'élément `.description` pour y inclure le nom du chat grâce à un **template string** 😊

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



Le texte dans la page a bel et bien changé.

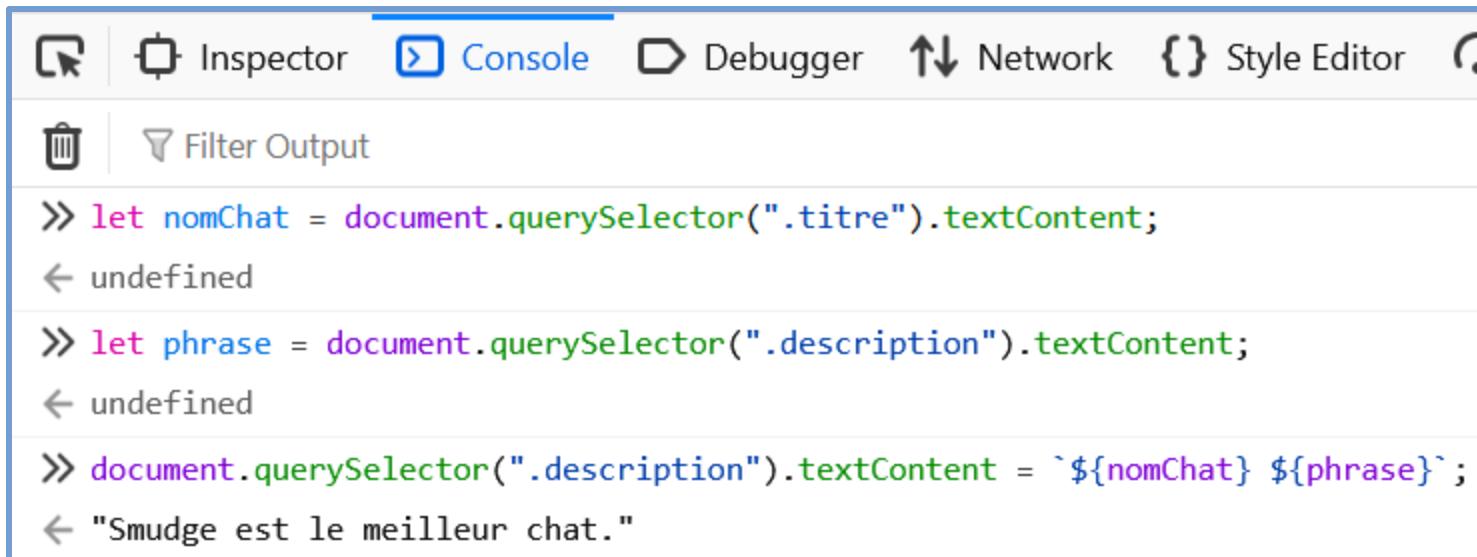


## ❖ Modifier le contenu textuel

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

```

Récapitulatif :



The screenshot shows the developer tools console with the following code:

```
» 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

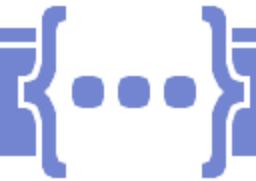
est le meilleur chat.



Smudge

Smudge est le meilleur chat.





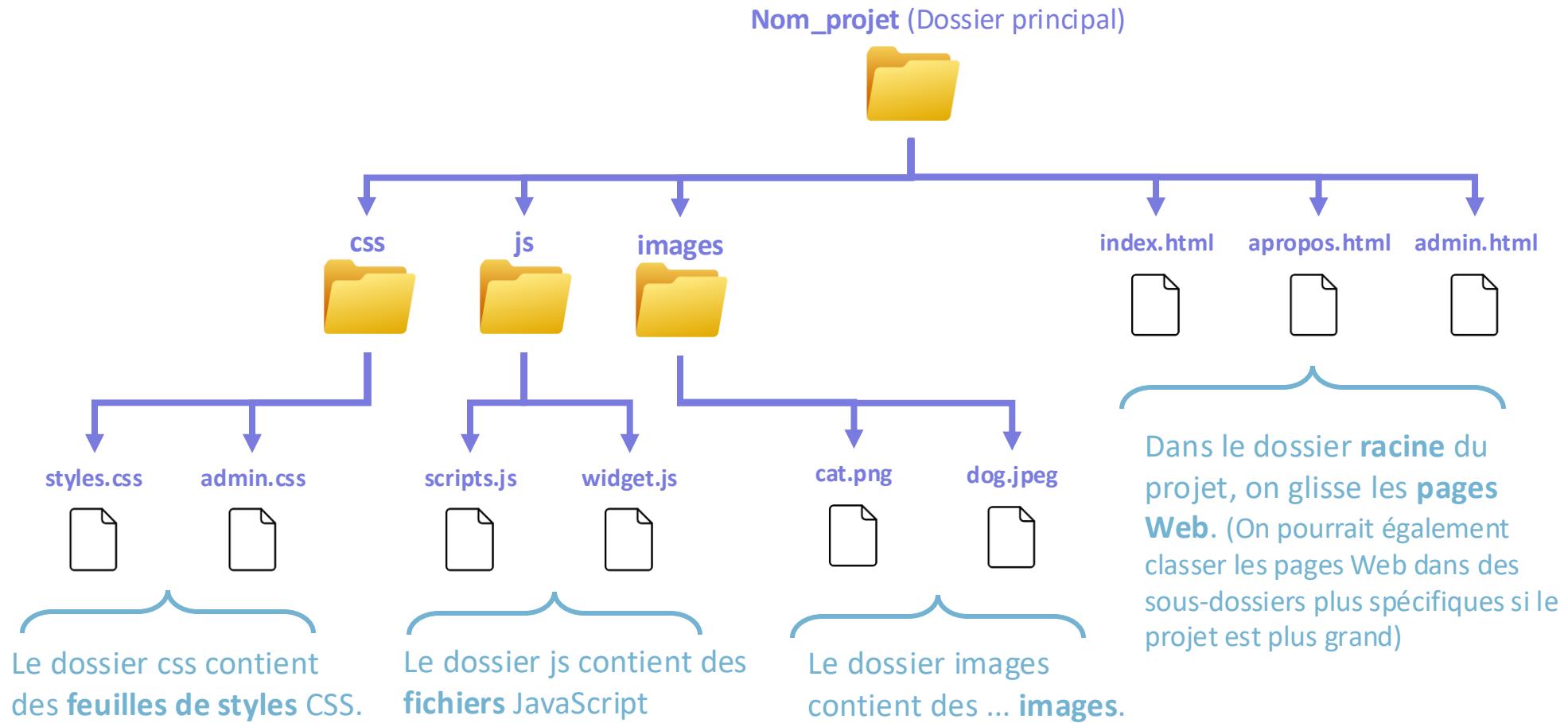
## ❖ Visual Studio code

- ◆ « Éditeur de code » pour plein de langages, dont HTML / CSS / JavaScript
  - Outil qui nous aidera 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](#)
    - Un guide d'installation vous est également fourni dans les notes de cours sur Léa.



## ❖ Créer un nouveau projet Web

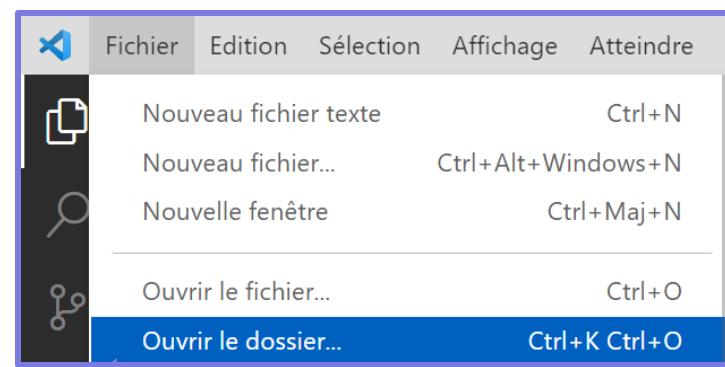
- ◆ Nous utiliserons la structure suivante pour le dossier de nos projets



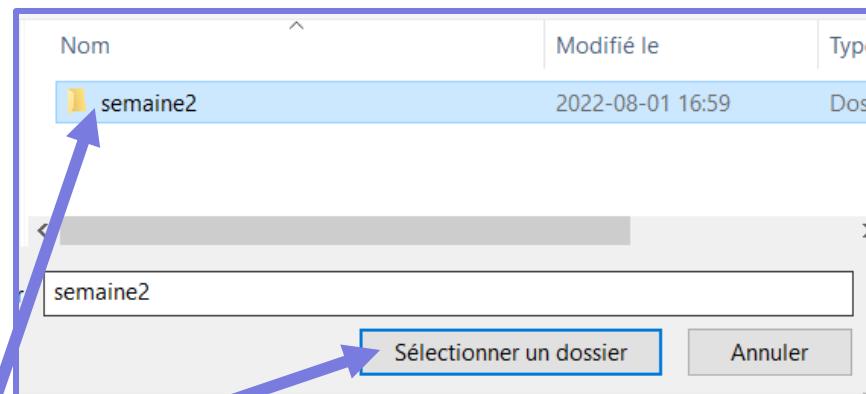


## ❖ Ouvrir un projet avec VS Code

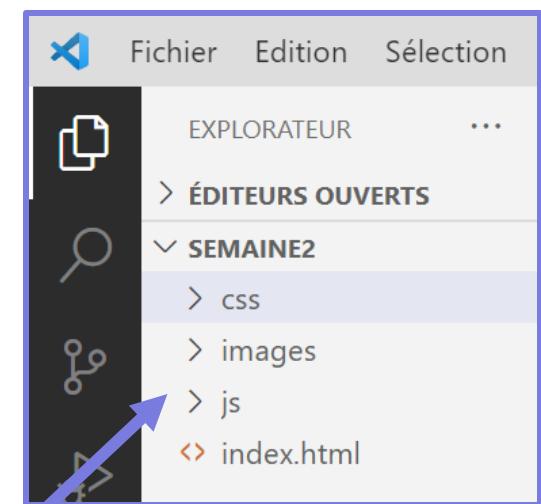
- ◆ Une fois le répertoire du projet créé, on peut ouvrir le dossier avec Visual Studio Code.



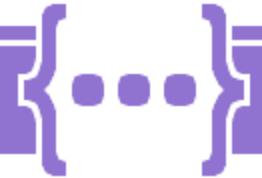
1 Fichier → Ouvrir le dossier...



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

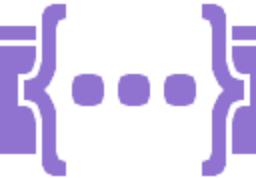


3 On a facilement accès à tous les dossiers / fichiers du projet dans **Visual Studio Code** et on peut éditer le code.



## ❖ Introduction aux **fonctions**

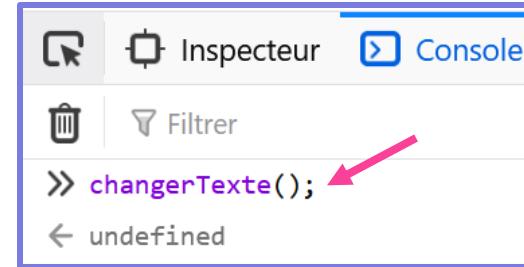
- ◆ Qu'est-ce qu'une **fonction**, grossièrement ?
- ◆ Où déclarer une fonction ?
- ◆ Créer une fonction



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

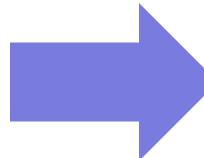
### ◆ Exemple

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



**Smudge**

est le meilleur chat.



**Meme cat**

est le meilleur chat.



- 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 la classe **titre**.

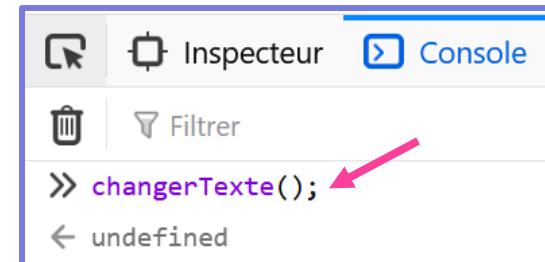
Ce mot-clé sert à **déclarer** une fonction

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";
}
```

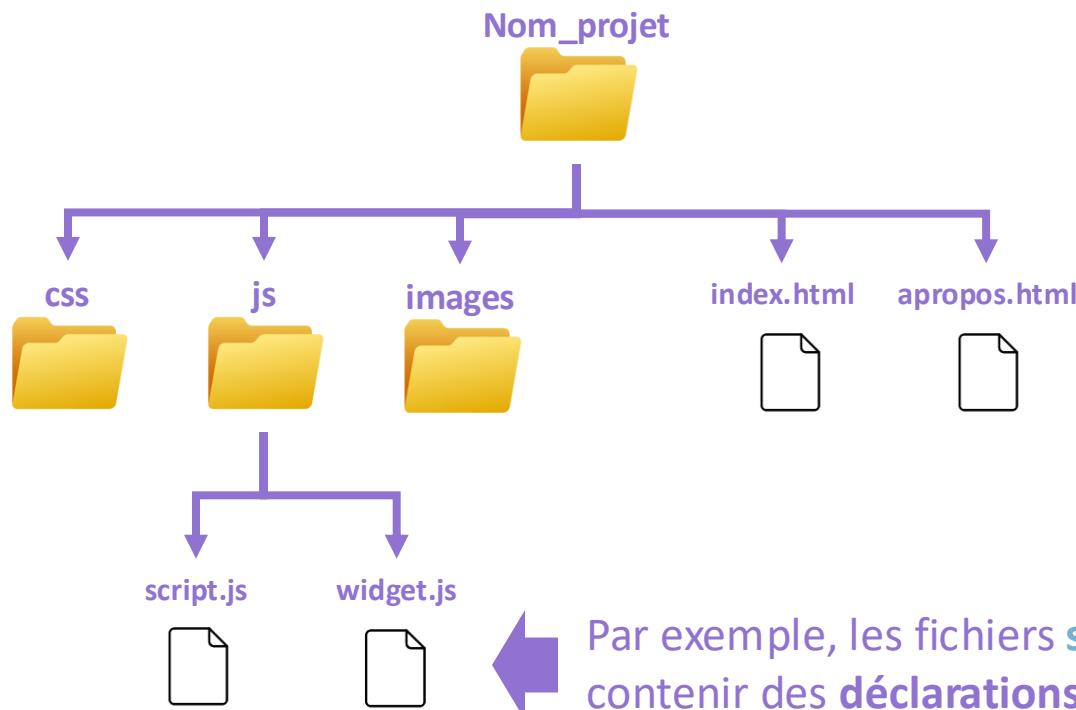
- Si on déclare la fonction dans la **console** du navigateur... la fonction n'existera plus quand nous réactualiserons la page. 😢 Pas très pratique.





## ❖ Où déclarer la fonction ?

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



```
index.html    JS scripts.js  X
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.



## ❖ Où déclarer la fonction ?

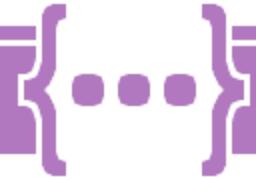
- ◆ De plus, il faut ajouter une ligne de code HTML dans la page Web où l'on souhaite pouvoir utiliser cette fonction :

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.

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



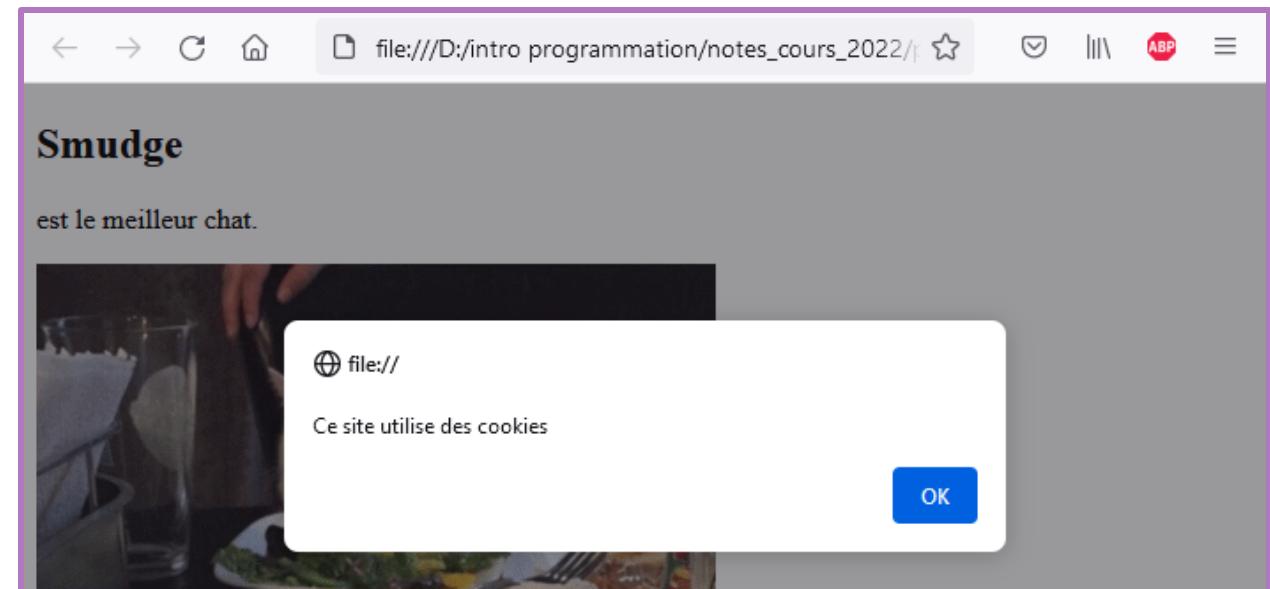
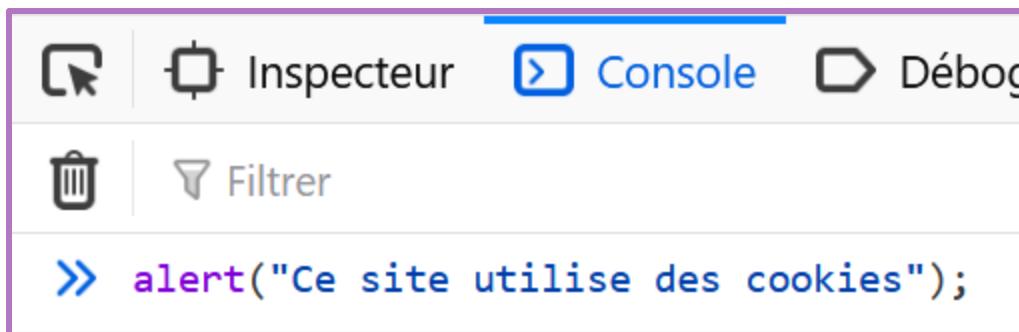
## ❖ Fonctions préexistantes

- ◆ 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()

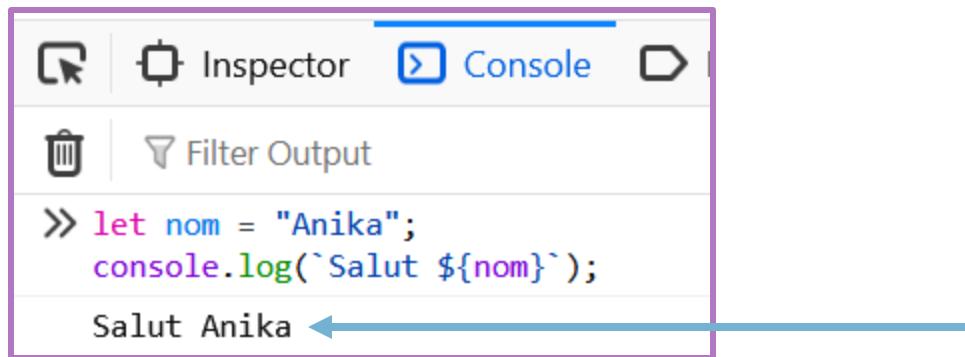
- ◆ La fonction **alert()** permet de créer un « pop-up » dans la page avec le message de notre choix.
  - Il suffit d'inclure une **chaîne de caractères** à l'intérieur des **parenthèses**.
  - Exemple :



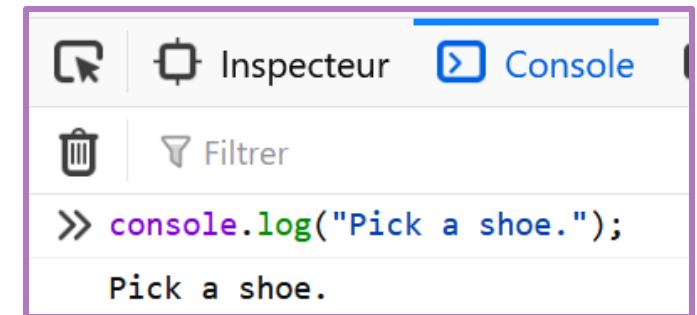


## ❖ console.log()

- ◆ 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 :



```
» let nom = "Anika";
  console.log(`Salut ${nom}`);
  Salut Anika
```



```
» console.log("Pick a shoe.");
  Pick a shoe.
```

(Exemple plus simple)

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 **trois** choses suivantes :

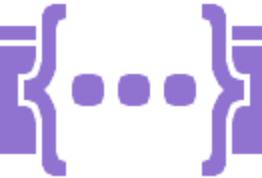
- 1) Remplace le texte « est le meilleur chat. » par « veut manger sa salade en paix. » **dans la page**.
- 2) Fait un **pop-up** avec le message « Texte changé ! ».
- 3) Affiche le message « Fonction terminée. » **dans la console**.

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

### Smudge

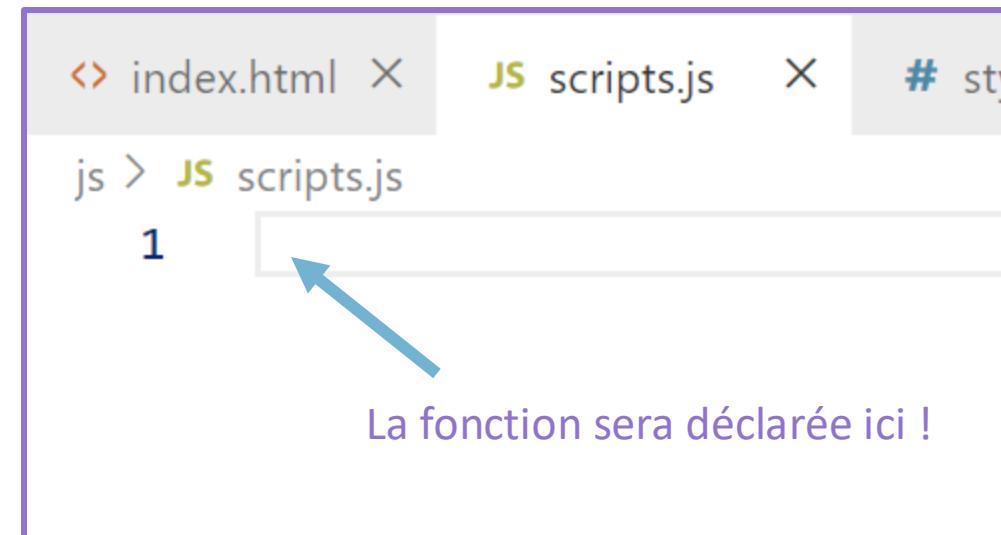
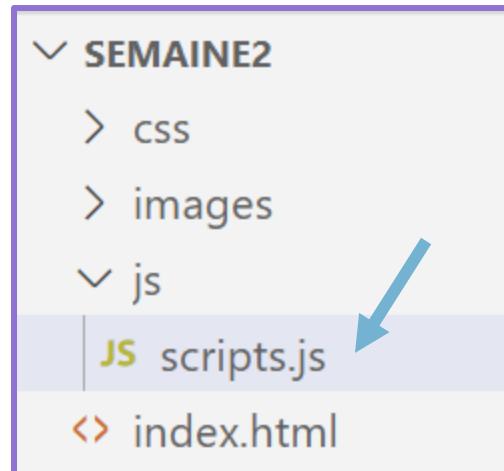
est le meilleur chat.





## ❖ 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.

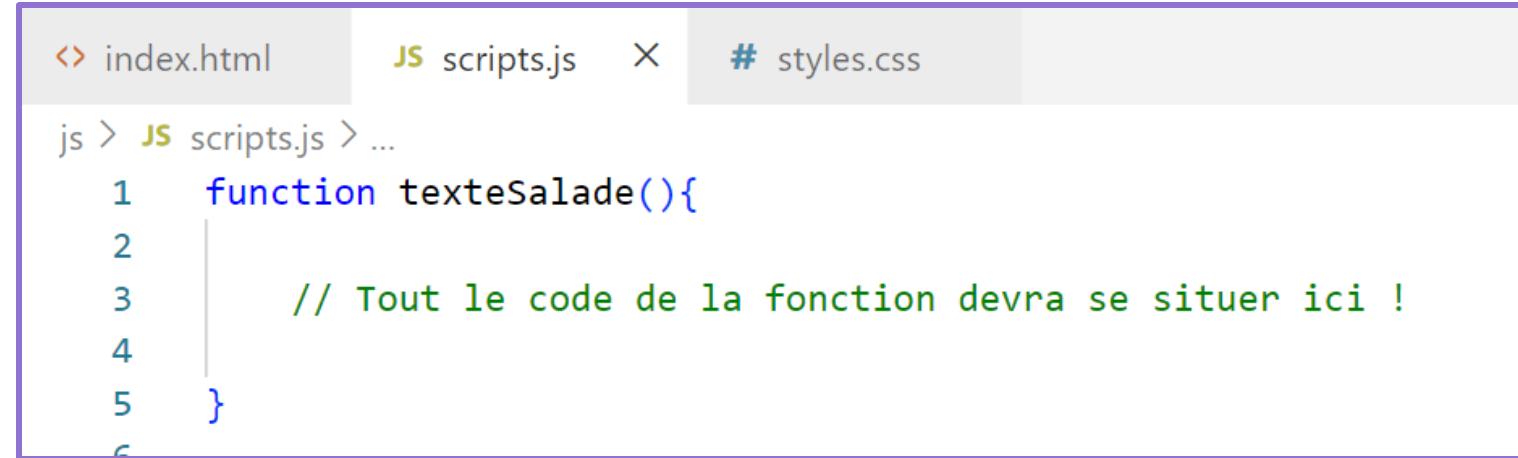




## ❖ Créer une fonction

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

- **N'oubliez jamais** l'accolade ouvrante **{** et l'accolade fermante **}** ! Si on oublie l'une des deux, **la fonction ne marchera pas** ! Ces deux symboles sont indispensables pour que JavaScript sache où la fonction **commence** et où la fonction se **termine**.



## ❖ 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 class="description">est le meilleur chat.</p>
```

On jette un coup d'œil au code HTML pour trouver la **classe** 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.");  
}
```

pop up

message dans la console

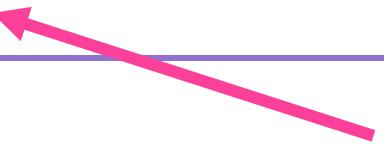
Modification du texte dans la page



## ❖ Créer une fonction

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

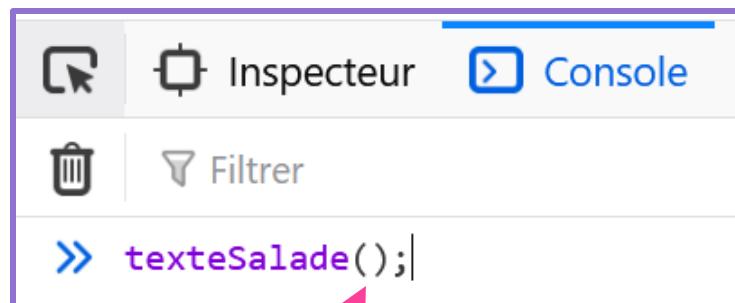
```
↳ index.html X JS scripts.js # styles.css
↳ index.html > ⚒ html > ⚒ body > ⚒ div > ⚒ img
1  <!DOCTYPE html>
2  <html lang="fr">
3  <head>
4  |  <meta charset="UTF-8">
5  |  <meta http-equiv="X-UA-Compatible" content="IE=edge">
6  |  <meta name="viewport" content="width=device-width, initial-scale=1.0">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



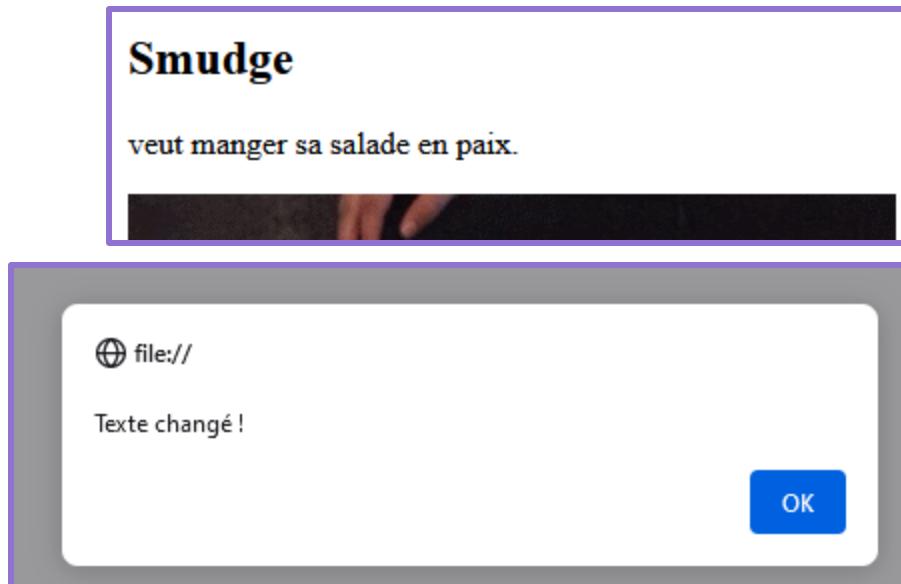


## ❖ Créer une fonction

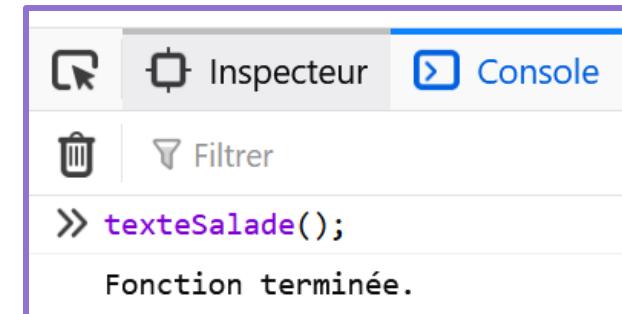
- ◆ **Étape 5 : Tester la fonction !** Il est possible qu'on ait fait des erreurs. Il faut s'assurer qu'elle fonctionne tel que prévu.



Ne pas oublier de mettre des parenthèses vides quand on appelle notre fonction !



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.



## ❖ Attention ! Une fonction à la fois

- ◆ Une **erreur fréquente** lorsqu'on déclare une **fonction** est d'oublier une accolade **}** ou de mal placer les accolades.

- L'accolade ouvrante **{** est en rouge et il y a un **trait rouge** en bas de la fonction : c'est parce qu'il manque l'**accolade de fermeture ! }**

- Ici, le problème est qu'on a déclaré la fonction... **dans une autre fonction** ! Ce n'est pas bon.
- On doit « sortir » la **fonction** et la déclarer en-dessous.

```
function texteJeu(){  
  document.querySelector(".description").textContent = "veut jouer";  
}  
  
function texteSalade(){  
  document.querySelector(".description").textContent = "veut manger sa salade en paix";  
  function texteJeu(){  
    document.querySelector(".description").textContent = "veut jouer";  
  }  
}  
}
```