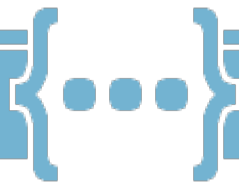


# Semaine 3

Variables globales, événements, DOM (styles), this

Intro. à la programmation - Aut. 2022



- ❖ Révision
- ❖ Variables globales / locales et constantes
- ❖ Événements
- ❖ DOM (styles)
- ❖ this



## ❖ Semaine 1 :

- ◆ Opérateurs mathématiques `+`, `-`, `*`, `/`, `()`
- ◆ Opérateurs d'affectation `=`, `+=`, `-=`, `++`, `--`
- ◆ Déclarer une variable : `let a = 3;`



## ❖ Semaine 2 :

### ◆ Chaînes de caractères

```
>> let energie = "Je m'endors déjà";
```

### ◆ Concaténation avec +

```
>> "sa" + "lut"
```

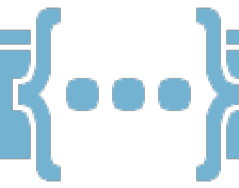
```
← "salut"
```

### ◆ Concaténation avec +=

```
>> let mot = "per";
```

```
    mot += "ruche";
```

```
← "perruche"
```



## ❖ Semaine 2 (suite)

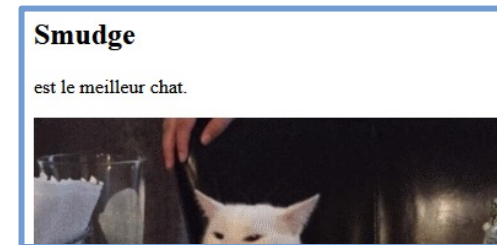
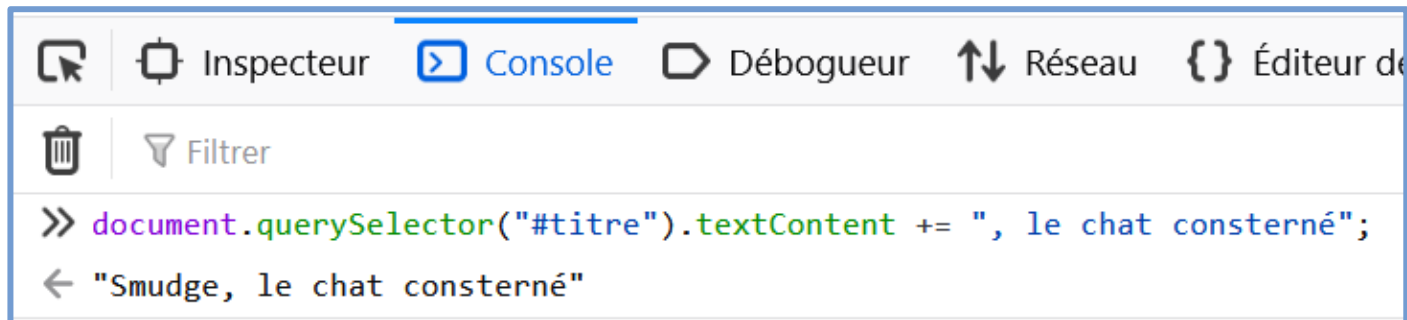
### ◆ document.querySelector() et .textContent

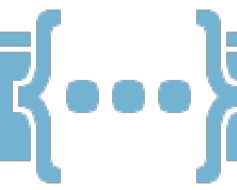
```
<div id="pikachu"> Pick a shoe </div>
```



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

### ◆ .textContent et concaténation





## ❖ Semaine 2 (suite)

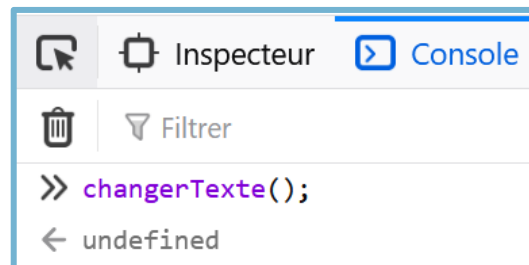
### ◆ Les fonctions

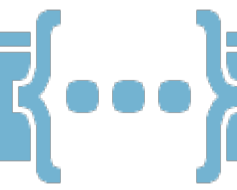
Ce mot-clé sert à **déclarer** une fonction

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";  
}
```

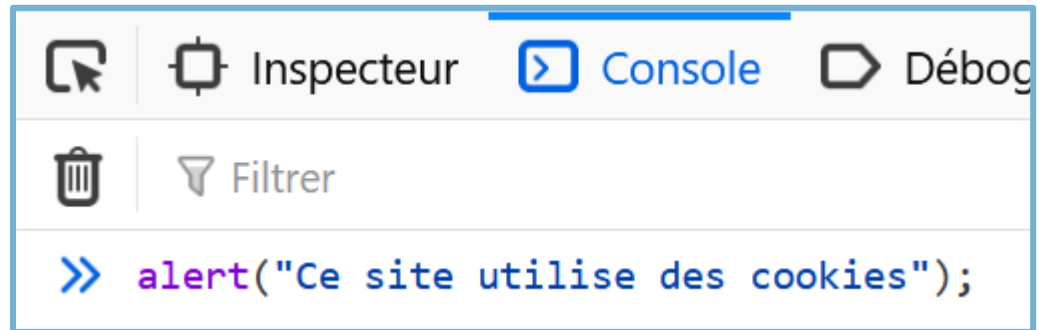




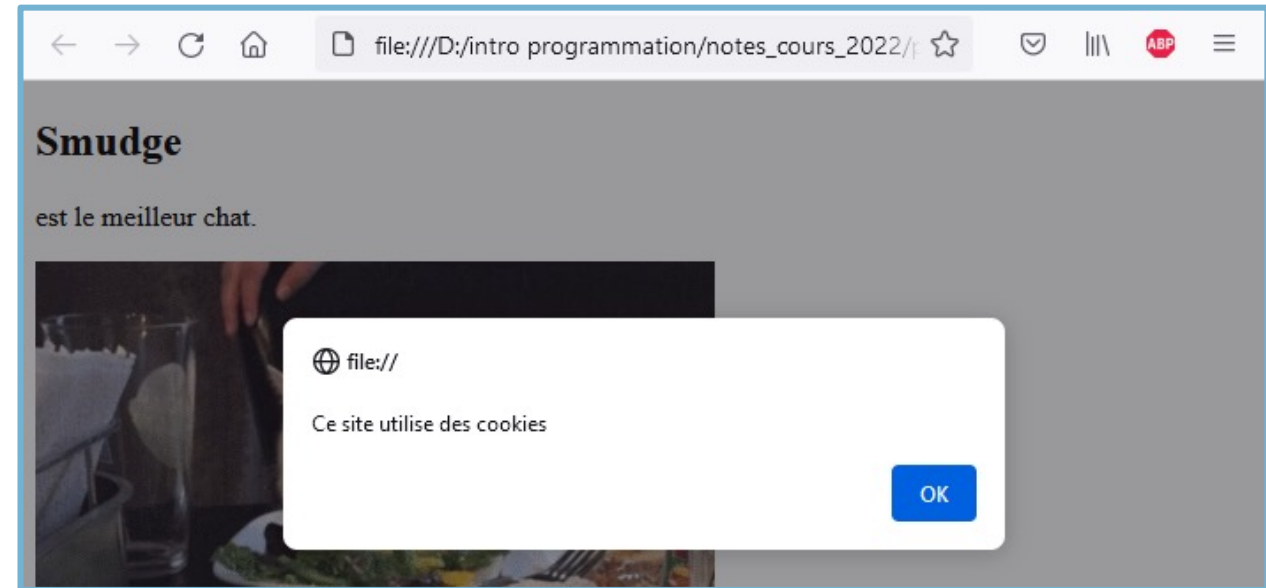
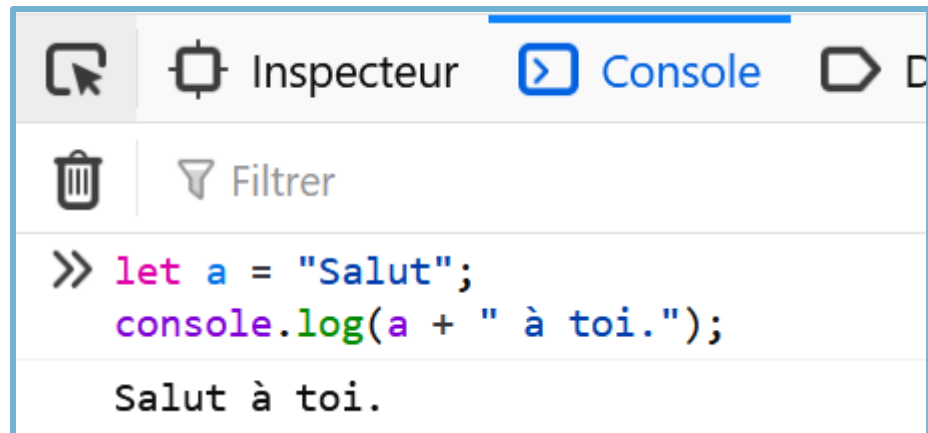
## ❖ Semaine 2 (suite)

### ◆ Fonctions préexistantes

- alert()



- console.log()





## ❖ Constantes

- ◆ Nous avons vu comment déclarer des **variables** ...

```
>> let nom = "Simone";  
    let nombre = -3.5;
```

- ◆ Nous pouvons également déclarer des **variables constantes**

- Il suffit de remplacer le mot-clé **let** par « **const** »

```
>> const PI = 3.1415;  
    const USD_TO_CAD = 0.79;
```

**Convention** : Les variables constantes sont nommées en **MAJUSCULES** pour les reconnaître facilement !

- Ces **variables constantes** peuvent être utilisées comme n'importe quelle autre

```
let circonference = PI * 1.65;
```





## ❖ Constantes

- ◆ **Particularité** : Leur valeur **ne peut pas** être **modifiée**. (Cela provoque une erreur dans le programme)

```
const PI = 3.1415;
```



```
PI = 3.14;
```



```
PI = PI + 1;
```

```
let PI = 3.1415;
```



```
PI = 3.14;
```



```
PI = PI + 1;
```

## ◆ Pourquoi les utiliser ?

- Lorsqu'on sait qu'une **variable** n'est jamais censée changer de valeur, il est préférable d'en faire une **constante**. Comme ça, si jamais on change sa valeur par erreur, le programme nous avertit immédiatement de cette bétise.



❖ Nous avons déjà vu comment déclarer une variable

◆ Ex : `let a = 4;`

❖ Toutefois, l'**emplacement** dans le code (dans le fichier `scripts.js`, par exemple) où cette variable est **déclarée** est important.

◆ La variable n'existe qu'à l'intérieur de la fonction où elle est déclarée. On ne peut pas l'utiliser ailleurs.

La variable `texte` est déclarée dans la fonction `titre1()`

```
function titre1(){  
  let texte = "Natacha n'attache pas son chat";  
  document.querySelector("#titre").textContent = texte;  
}  
  
function titre2(){  
  document.querySelector("#soustitre").textContent = texte;  
}
```

On peut utiliser `texte` ici sans problème.

On ne peut pas réutiliser la variable `texte` ici, puisqu'on n'est pas dans la fonction `titre1()`. Cela provoque une erreur.



## ❖ Variables locales

- ◆ Dans cette situation, **texte** est une variable **locale**. Elle ne peut être utilisée que « localement », c'est-à-dire seulement à l'intérieur de la **fonction** ou du « **bloc** » de code où elle est déclarée.

```
function titre1(){  
  let texte = "Natacha n'attache pas son chat";  
  document.querySelector("#titre").textContent = texte;  
}  
  
function titre2(){  
  document.querySelector("#soustitre").textContent = texte;  
}
```





## ❖ Variables **globales**

- ◆ Une variable dite « **globale** » peut être utilisée n'importe où dans le code.
- ◆ Les variables **globales** doivent être déclarées en dehors de toute fonction, dans n'importe quel fichier JavaScript du projet Web.

- ◆ Ici, la variable **gTexte** est déclarée en dehors de toute fonction, au début du code.

```
let gTexte = "Ces seize chaises sont sèches";
```

```
function titre1(){  
  document.querySelector("#titre").textContent = gTexte;  
}
```

- ◆ Elle est donc utilisable n'importe où dans les **fonctions** qui suivent.

```
function titre2(){  
  document.querySelector("#soustitre").textContent = gTexte;  
}
```



## ❖ Convention de nommage

- ◆ Pour dissiper le doute 🕵️, nous ajouterons toujours un **g** devant le nom d'une variable globale.
  - De plus, nous déclarerons toujours les variables globales tout en haut du fichier .js

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

scripts.js > 📦 calculerAireCercle
{
  let gScore = 15;
  let gTexte = "Ces seize chaises sont sèches";
  let gCouleur = "magenta";
}

5  function titre1(){
6  →  let message = "Salut";
7      document.querySelector("#titre").textContent = gTexte;
8      console.log(message);
9  }
```

Variables globales

Variable locale



## ❖ Constantes

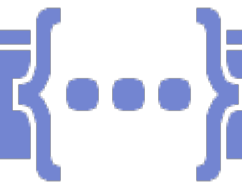
### ◆ Globales ou locales

- Tout comme les variables déclarées avec « **let** », une variable **constante** peut être **globale** ou **locale**, selon l'endroit où on l'a déclarée.

```
const gUSD_TO_CAD = 0.79;

function calculerAireCercle(){
  const PI = 3.1415;
  let resultat = PI * PI * 3;
}
```

- **gUSD\_TO\_CAD** : Déclarée hors fonction, donc **globale**. Peut être utilisée n'importe où dans le code.
- **PI** : Déclarée dans une fonction, donc **locale**. Peut seulement être utilisée dans la fonction où elle a été déclarée.
  - **À éviter** : en général on veut que nos constantes soient globales. **!**



## ❖ Écouteurs d'événements

◆ Les écouteurs d'événements, permettent, entre autres, d'appeler des **fonctions** suite à la détection d'un **déclencheur**.

○ Exemples simples :

- En **cliquant** sur un **élément**... son **texte** change !

Cliquez-moi délicatement



Tu as cliqué trop fort 😞

- En **cliquant** sur un **élément**... une **alerte** apparait dans la page !

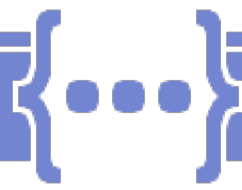
Ne me touche pas 😡



🌐 file://

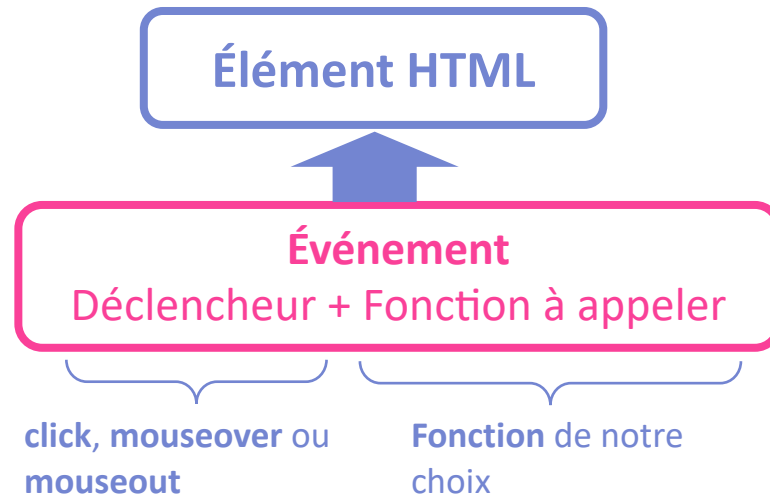
IL NE FALLAIT PAS CLIQUER 🚗⚠️

OK

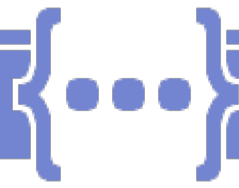


## ❖ Écouteurs d'événements

- ◆ Les écouteurs d'événements, permettent, entre autres, d'**appeler des fonctions** suite à la détection d'un **déclencheur**.
- ◆ Voici 3 **déclencheurs** :
  - **click** : Appelle une fonction lorsque l'élément HTML **est cliqué**.
  - **mouseover** : Appelle une fonction lorsque l'élément **est survolé**.
  - **mouseout** : Appelle une fonction lorsque l'élément **n'est plus survolé**. (La souris le quitte)







## ❖ Écouteurs d'événements

### ◆ Comment ajouter un écouteur d'événement

#### ○ Syntaxe :

```
document.querySelector("#id").addEventListener("type", nom_fonction)
```

Élément associé à l'événement

Déclencheur et fonction

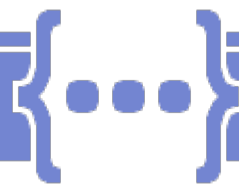
#### ○ Exemple

```
document.querySelector("#bouton1").addEventListener("click", changerTexte);
```

Id de l'élément interactif

Type d'événement

Fonction



## ❖ Écouteurs d'événements

### ◆ Exemple complet

- On a un **élément** avec l'id "**bouton1**". Il est associé à un **événement** de type « **click** » qui exécute la fonction « **changerTexte()** » lorsque déclenché.

```
<button id="bouton1">Cliquez-moi délicatement</button>
```

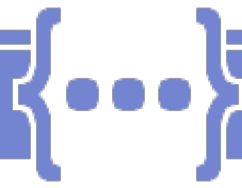
```
document.querySelector("#bouton1").addEventListener("click", changerTexte);
```

```
function changerTexte(){  
  document.querySelector("#bouton1").textContent = "Tu as cliqué trop fort 😞";  
}
```

Cliquez-moi délicatement



Tu as cliqué trop fort 😞



## ❖ Écouteurs d'événements

### ◆ Où ajouter les événements

- Dans le cadre du cours, nous placerons toujours les **déclarations d'écouteurs d'événements** dans une fonction nommée **init()**, qui sera automatiquement appelée lorsqu'un projet Web est chargé dans le **navigateur**.

```
function init(){  
  
    document.querySelector("#bouton1").addEventListener("click", changerTexte);  
    document.querySelector("#bouton2").addEventListener("click", lancerAlerte);  
  
}
```

- Par exemple, ci-dessus, on peut voir que 2 **écouteurs d'événements** sont déclarés.



## ❖ Changer un **style** avec DOM

- ◆ Changer un style correspond à modifier le **CSS** d'un élément **HTML**. Pour cela, on utilise la syntaxe **document.querySelector("id").style**
- ◆ Nous allons voir comment changer...
  - La **couleur du texte**
  - La **couleur de fond**
  - La **couleur de la bordure**
  - La **largeur de la bordure**
  - La **largeur / la hauteur** de l'élément
  - L'**opacité** d'un élément
  - La **visibilité** d'un élément
  - L'**espacement** depuis la gauche / le haut d'un élément

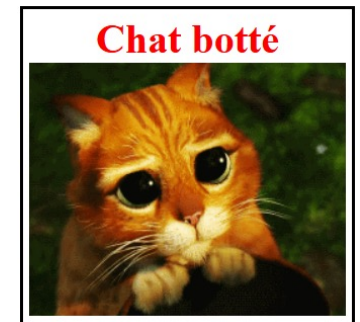
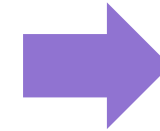


## ❖ Changer la couleur du texte

◆ Syntaxe : `document.querySelector("#id").style.color = "nom_de_la_couleur";`

```
<h1 id="titre">Chat botté</h1>
```

```
>> document.querySelector("#titre").style.color = "red";
```



## ❖ Changer la couleur de fond

◆ Syntaxe : `document.querySelector("#id").style.backgroundColor = "nom_de_la_couleur";`

```
<h1 id="titre">Chat botté</h1>
```

```
>> document.querySelector("#titre").style.backgroundColor = "lightblue";
```





## ❖ Changer la couleur de la bordure

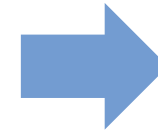
◆ `document.querySelector("#id").style.borderColor = "nouvelle_couleur";`

## ❖ Changer la largeur de la bordure

◆ `document.querySelector("#id").style.borderWidth = "taille_en_pixels";`

```
<div id="boite">
  <h1 id="titre">Fée marraine</h1>
  
</div>
```

```
>> document.querySelector("#boite").style.borderColor = "gold";
     document.querySelector("#boite").style.borderWidth = "20px";
```

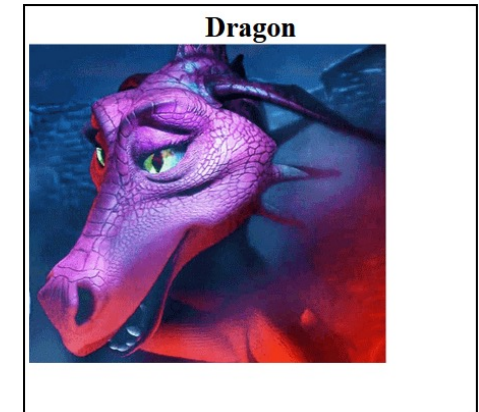
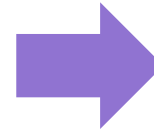
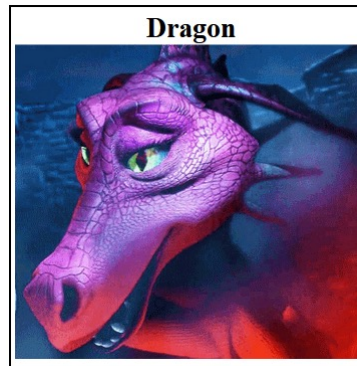




## ❖ Changer la largeur / hauteur d'un élément

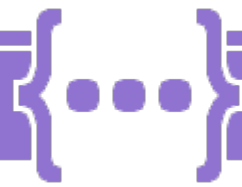
- ◆ `document.querySelector("#id").style.width = "largeur_en_pixels";`
- ◆ `document.querySelector("#id").style.height = "hauteur_en_pixels";`

```
>> document.querySelector("#boite").style.width = "500px";  
document.querySelector("#boite").style.height = "450px";
```



## ❖ Changer la visibilité d'un élément

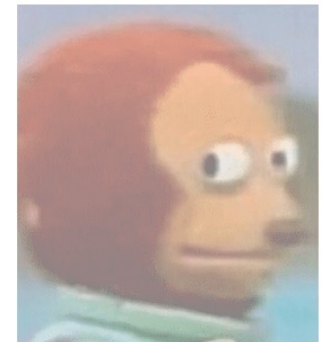
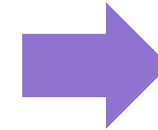
- ◆ `document.querySelector("#id").style.display = "none";`
  - Permet de masquer l'élément : Il deviendra invisible.
  - Alternativement, les valeurs "**block**", "**inline**" et "**inline-block**" rendront l'élément visible.



## ❖ Changer l'opacité d'un élément

- ◆ `document.querySelector("#id").style.opacity = "0.5";`
- ◆ Valeur de **0** à **1**.
  - **0** -> totalement transparent
  - **1** -> totalement opaque.

```
>> document.querySelector("#suspect").style.opacity = "0.5";
```







## ❖ Changer l'espacement à gauche / en haut d'un élément

- ◆ `document.querySelector("#id").style.left = "taille_en_pixels";`
- ◆ `document.querySelector("#id").style.top = "taille_en_pixels";`

```

```

```
>> document.querySelector("#doris").style.left = "200px";  
document.querySelector("#doris").style.top = "50px";
```



On peut voir que l'image s'est éloignée de la gauche de 200 pixels et du haut de 50 pixels.



## ❖ Plus de **couleurs** s'il vous plaît 🎨

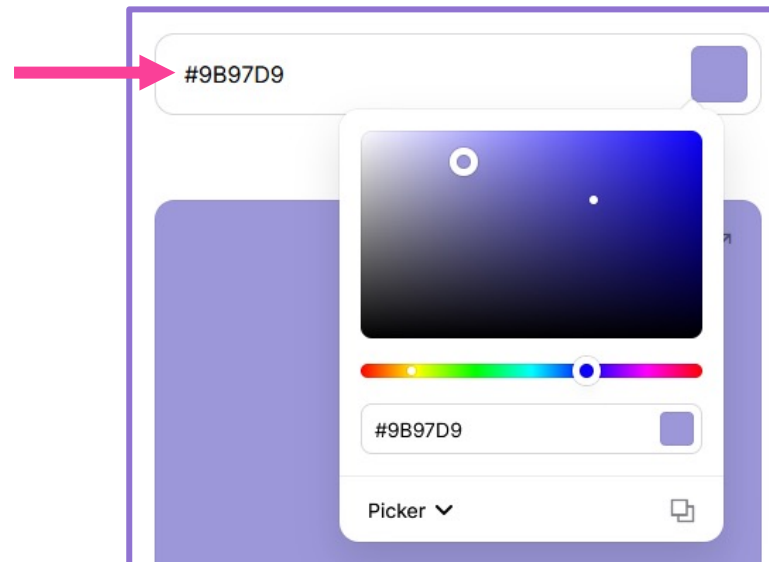
- ◆ Les navigateurs Web connaissent 140 couleurs en lettres comme ceci :

```
document.querySelector("#id").style.color = "red";
```

- ◆ C'est plutôt limité. Afin de pouvoir utiliser des couleurs personnalisées, on doit utiliser les couleurs « hexadécimales » :

```
document.querySelector("#id").style.color = "#DC143C";
```

- <https://colors.co/fe0313> Exemple de **roue chromatique** qui nous permet d'obtenir le code **hexadécimal** d'une couleur de notre choix.



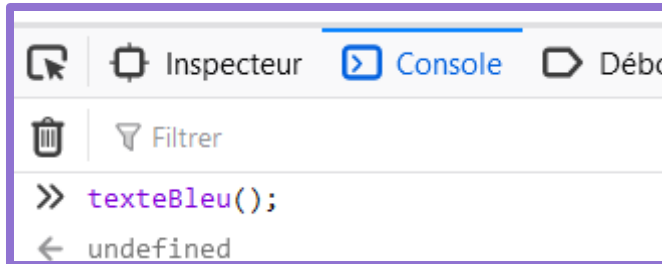


❖ On peut également glisser ces instructions (qui se servent du **DOM**) dans des **fonctions**. (Plutôt que d'écrire ces instructions en entier dans la **console**)

◆ Exemple

```
function texteBleu(){  
    document.querySelector("#bouton1").style.color = "blue";  
}
```

- Bien entendu, on pourra appeler cette **fonction** dans la **console** ou avec un **événement** par la suite pour l'utiliser !



```
<button id="bouton1">Cliquez-moi délicatement</button>
```

Cliquez-moi délicatement



Cliquez-moi délicatement



❖ Jusqu'à maintenant, nous avons vu trois modifications que l'on peut faire à un élément HTML :

- ◆ Modifier / accéder à son contenu textuel
- ◆ Modifier un style
- ◆ Lui ajouter un écouteur d'événements

```
document.querySelector("#bouton").addEventListener("type", maFonction);
```

```
document.querySelector("#bouton").textContent = "Texte";
```

```
document.querySelector("#bouton").style.propriété = "valeur de style";
```




## ❖ Mot-clé **this**

- ◆ Un aspect de la **fonction texteBleu()** est embêtant : cette fonction ne marche que pour l'élément avec l'id « **bouton1** » !

```
function texteBleu(){  
  document.querySelector("#bouton1").style.color = "blue";  
}
```

- On ne peut pas utiliser la fonction pour un autre élément.
  - Une solution pourrait être de créer une fonction pour chaque élément dont le fond peut changer de couleur... mais cela implique de la **répétition inutile de code** ! 😞

```
function texteBleu(){  
  document.querySelector("#bouton1").style.color = "blue";  
}  
  
function texteBleu2(){  
  document.querySelector("#bouton2").style.color = "blue";  
}
```





## ❖ Mot-clé **this**

- ◆ C'est ici que le mot-clé « **this** » est pratique.
  - Si on remplace **document.querySelector("#id")** dans la fonction **texteBleu()** par **this**, c'est automatiquement **l'élément HTML qui appelle la fonction** (suite au déclenchement d'un événement) qui sera affecté par la **fonction**.

```
function texteBleu(){  
  |   this.style.color = "blue";  
}
```


- Donc présentement, cette fonction marchera pour tous les éléments pour lesquels nous avons configuré un **événement** qui appelle la fonction « **texteBleu()** ».



## ❖ Mot-clé **this**

### ◆ Exemple

```
function init(){  
    document.querySelector("#bouton1").addEventListener("click", texteBleu);  
    document.querySelector("#bouton2").addEventListener("click", texteBleu);  
}
```



```
function texteBleu(){  
    this.style.color = "blue";  
}
```

- Si on clique sur l'élément avec l'id **#bouton1**, la fonction `texteBleu` rendra le texte de l'élément avec l'id **#bouton1** bleu.
- Si on clique sur l'élément avec l'id **#bouton2**, la fonction `texteBleu` rendra le texte de l'élément avec l'id **#bouton2** bleu.