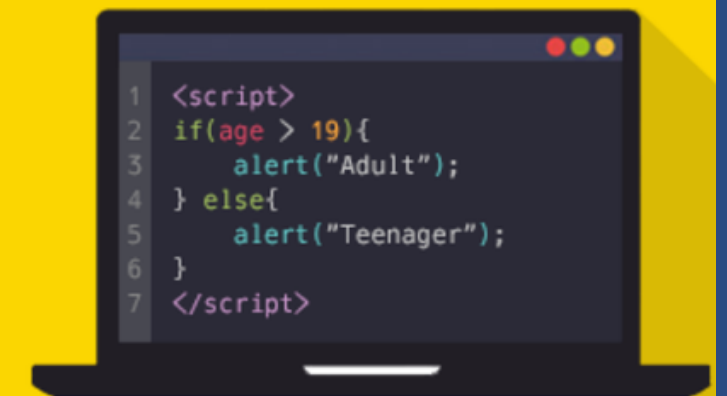


# Javascript

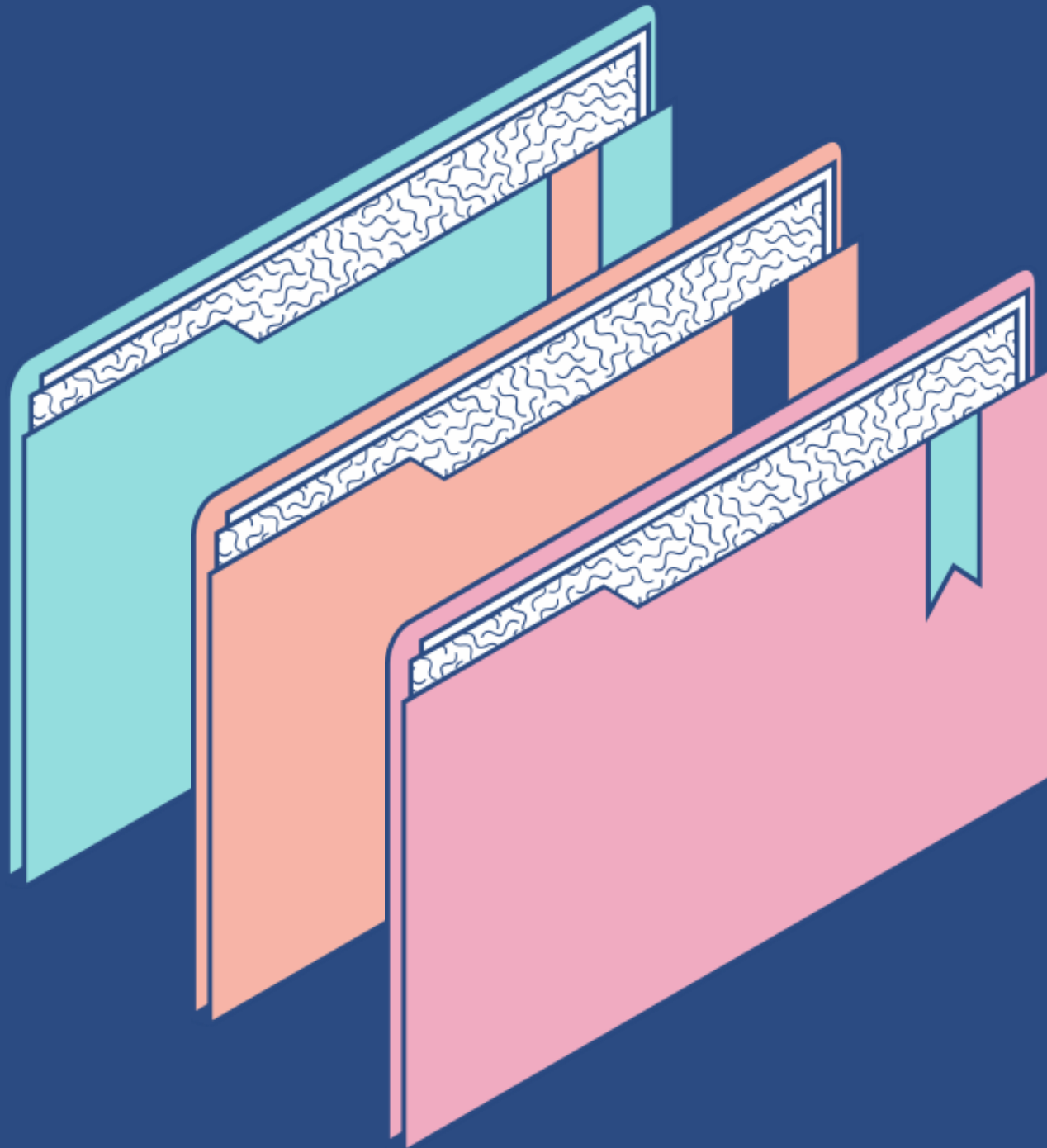


**JavaScript**



# SOMMAIRE

- Présentation du JS
- Variables
- Structures de contrôle
- Fonctions
- POO en JS
- Valeurs primitives
- Manipulation du BOM
- Manipulation du DOM
- Fonctions avancées
- Gestion des erreurs
- Stockage de données persistantes
- Canvas
- Asynchrone



# Variables et types JS

10

## JavaScript Variables

```
var title = 'Code Life';  
let publisher = 'House of Books';  
const author = 'Melvin Gray';
```

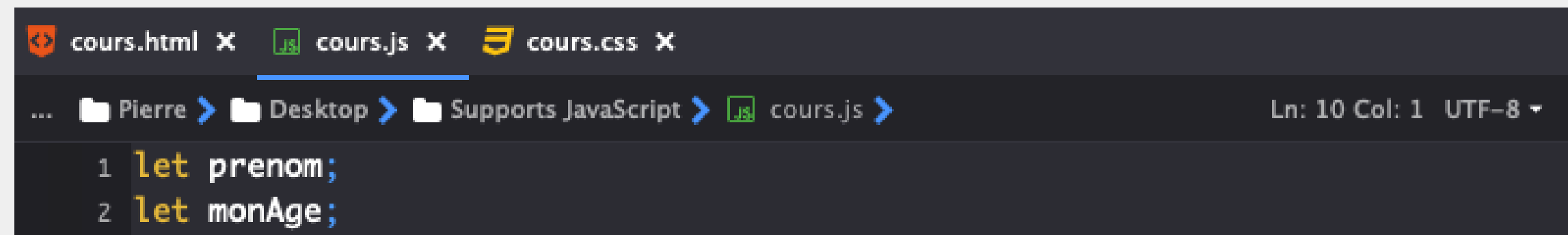


# Déclaration de variable

Une variable est un conteneur permettant de stocker une information de façon temporaire.

Pour créer une variable (déclaration), il faut utiliser le mot clé "var" ou "let" suivi de son nom.

Par convention, les variables s'écrivent en lowerCamelCase.



```
cours.html X cours.js X cours.css X
... Pierre Desktop Supports JavaScript cours.js Ln: 10 Col: 1 UTF-8
1 let prenom;
2 let monAge;
```



# Initialisation de variable

La première assignation de valeur à une variable s'appelle l'initialisation.

```
s.html X cours.js X cours.css X  
Pierre > Desktop > Supports JavaScript > cours.js > Ln: 9 Col: 15 UTF-8 JavaScript  
//On déclare et on initialise la variable en même temps  
let prenom = 'Pierre';  
  
//On déclare la variable puis on l'initialise ensuite  
let monAge;  
monAge = 29;
```

# Modifier la valeur d'une variable

La modification de valeur d'une variable s'effectue de la même façon que l'initialisation.

L'ancienne valeur sera alors supprimée, et remplacée par la nouvelle.

```
//On déclare et on initialise la variable en même temps  
let prenom = 'Pierre';  
  
//On déclare la variable puis on l'initialise ensuite  
let monAge;  
monAge = 29;  
  
/*On modifie la valeur stockée dans prenom.  
*Notre variable stocke désormais la valeur "Mathilde"*/  
prenom = 'Mathilde';
```



# Let vs Var

"let" est la nouvelle façon de déclarer les variables en JS, c'est donc la méthode à privilégier.

La déclaration des "let" doit obligatoirement se faire avant l'initialisation.  
Ce n'est pas le cas pour les "var" qui peuvent être déclarées ensuite.

```
//Ceci fonctionne  
prenom = 'Pierre';  
var prenom;  
  
//Ceci ne fonctionne pas et renvoie une erreur  
nom = 'Giraud';  
let nom;
```



# Types de données JS

Il existe 7 types de valeurs différentes en JS :

- String
- Number
- Boolean
- Null
- Undefined
- Symbol
- Object

```
let chaineDeCaractère = "toto"; //String
let nombre = 10; //Number
let bool = true; //Boolean
let nul = null; //null
let undef; //undefined
```



# Types de données JS : String

En JS, vous pouvez utiliser soit des guillemets " soit des apostrophes ' pour entourer vos chaînes de caractères.

Si votre chaîne de caractère contient elle-même l'un de ces caractères, il faudra l'échapper :

```
//Délimiteurs non trouvés dans la chaîne = rien à échapper  
let a = "Je m'appelle Pierre";  
  
//Délimiteurs non trouvés dans la chaîne (apostrophe non droite) = rien à échapper  
let b = 'Je m'appelle Pierre';  
  
//Délimiteurs trouvés dans la chaîne = on échappe le caractère en question  
let c = 'Je m\'appelle Pierre';  
  
//Délimiteurs non trouvés dans la chaîne = rien à échapper  
let d = "Je m'appelle « Pierre »";  
  
//Délimiteurs trouvés dans la chaîne = on échappe les caractères en question  
let e = "Je m'appelle \"Pierre\"";
```



# Types de données JS : Number

En JS, tous les nombres sont regroupés au sein du type unique Number

```
let x = 10; //x stocke un entier positif  
let y = -2; //y stocke un entier négatif  
let z = 3.14; //z stocke un nombre décimal positif
```



# Types de données JS : Boolean

Un boolean est une valeur binaire, soit true, soit false :

```
let vrai = true; //Stocke le booléen true
let faux = false; //Stocke le booléen false

/*On demande au JavaScript d'évaluer la comparaison "8 > 4". Comme 8 est bien
*strictement supérieur à 4, le JavaScript renvoie true en résultat. On
*stocke ensuite ce résultat (le booléen true) dans la variable let resultat*/
let resultat = 8 > 4;
```



# Types de données JS : Null & Undefined

Les types Null et Undefined n'ont qu'elles même comme valeurs respectives.

Le type undefined correspond à une variable qui n'a pas de valeur affectée.

Le type null correspond à une absence de valeur connue.

```
let nul = null;  
let ind;
```



# Opérateurs arithmétiques

```

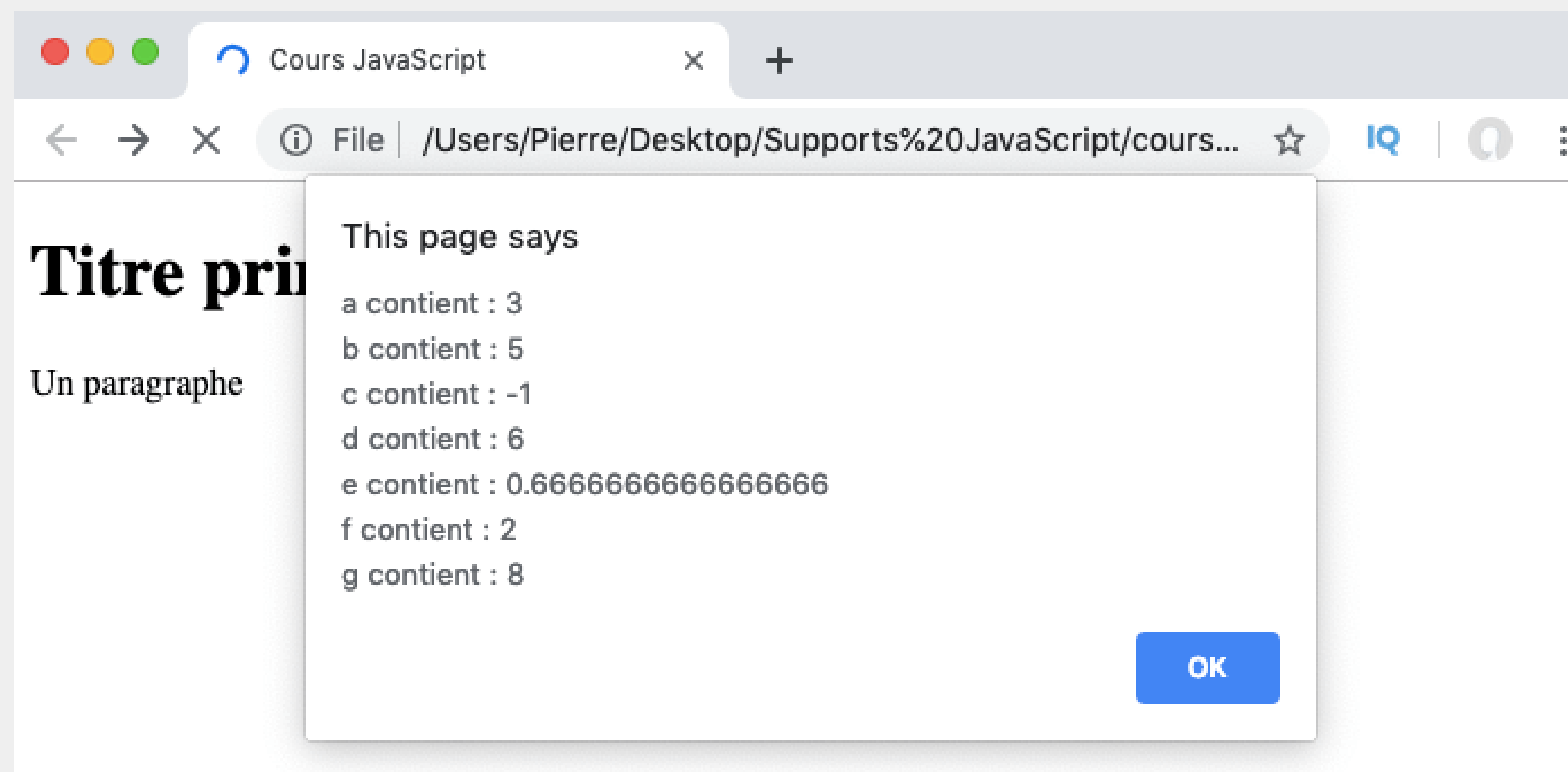
let x = 2;
let y = 3;
let z = 4;

let a = x + 1; //a stocke 2 + 1 = 3
let b = x + y; //b stocke 2 + 3 = 5
let c = x - y; //c stocke 2 - 3 = -1
let d = x * y; //d stocke 2 * 3 = 6
let e = x / y; //e stocke 2 / 3
let f = 5 % 3; //f stocke le reste de la division euclidienne de 5 par 3
let g = x ** 3; //g stocke 2^3 = 2 * 2 * 2 = 8

/*On affiche les résultats dans une boîte d'alerte en utilisant l'opérateur
 *de concaténation "+". On retourne à la ligne dans l'affichage avec "\n"*/
alert('a contient : ' + a +
      '\nb contient : ' + b +
      '\nc contient : ' + c +
      '\nd contient : ' + d +
      '\ne contient : ' + e +
      '\nf contient : ' + f +
      '\ng contient : ' + g);

```

# Opérateurs arithmétiques



# Priorité de calculs

La priorité de calcul est la même en JS qu'en arithmétique.  
Pensez à l'utilisation de parenthèses !

```
let x = 1 - 2 - 3; //Calcule (1 - 2) - 3 = -1 - 3 = -4
let y = 1 - (2 - 3); //Calcule 1 - (2 - 3) = 1 - (-1) = 1 + 1 = 2
let z = 2 ** 3 ** 2; //Calcule 3 ** 2 = 3 * 3 = 9 puis 2 ** 9 = 512

/*On affiche les résultats dans une boîte d'alerte en utilisant l'opérateur
 *de concaténation "+". On retourne à la ligne dans l'affichage avec "\n"*/
alert('x contient : ' + x +
      '\ny contient : ' + y +
      '\nz contient : ' + z);
```



# Opérateurs d'affectation

L'affectation s'effectue à l'aide de l'opérateur "=".

Vous pouvez cumuler opérateur arithmétique et d'affectation pour gagner du temps.

```
let x = 2; //x stocke 2
let y = 10; //y stocke 10

/*On ajoute 3 à la valeur stockée précédemment par x (2) puis on
*affecte le résultat à x. x stocke désormais 2 + 3 = 5*/
x += 3;

/*On multiplie la valeur de y (10) par celle de z (5) puis on affecte
*le résultat à y. y stocke désormais 10 * 5 = 50*/
y *= x;

alert('x stocke : ' + x + '\ny stocke : ' + y);
```





# Concaténation

La concaténation de chaînes se fait à l'aide de l'opérateur "+" en JS :

```
let x = 28 + 1; //Le signe "+" est ici un opérateur arithmétique  
let y = 'Bonjour';  
let z = x + 'ans'; //Le signe "+" est ici un opérateur de concaténation  
  
alert(y + ', je m\'appelle Pierre, j\'ai ' + z);
```



# Déclaration de constante

Une constante est identique à une variable, à la différence près que sa valeur sera immuable.

Une constante se déclare à l'aide du mot clé "const"

```
const prenom = 'Pierre';  
const age = 29;
```



# Exercice d'application n°1 : Création et opérations sur les variables

Créez 2 variables, contenant respectivement votre nom et votre prénom.

Concaténez ces 2 variables au sein d'une nouvelle variable name

Créez une nouvelle variable contenant votre age.

Affichez le résultat suivant : "Je m'appelle NAME, et j'avais XXX ans en l'an 2000"

*Vous pouvez visualiser vos résultats en utilisant la méthode alert()*