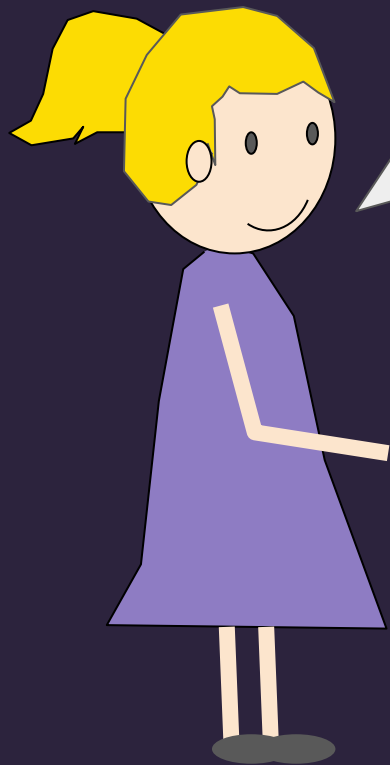


# MIXTe9N

Born to code



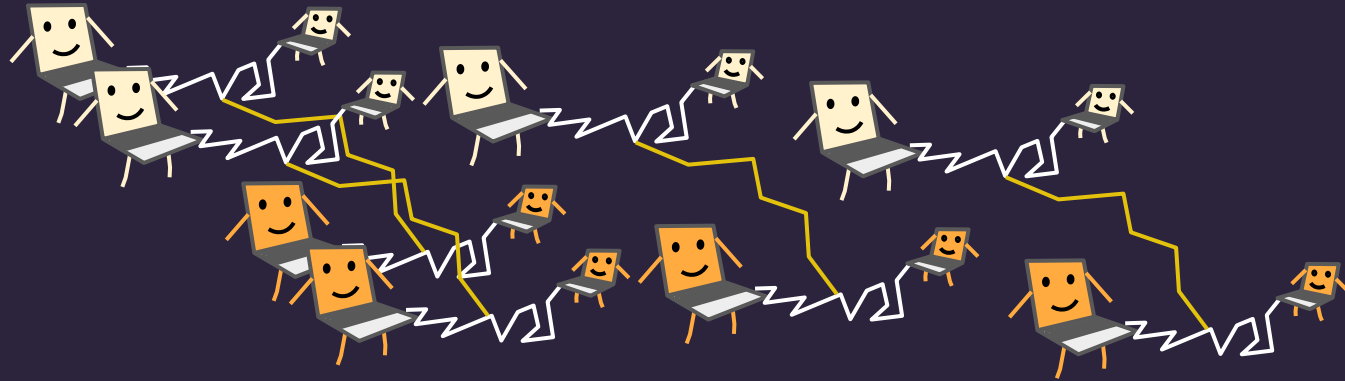
Avant d'attaquer ce chapitre je te conseille  
d'avoir vu ceux sur

1. Comment marche un ordinateur ?
2. Comment marche Internet ?
3. Comment marche le web ?
4. Comment programmer en HTML ?

# C'est quoi le JavaScript ?



# Internet est un ensemble de réseaux connectés entre eux

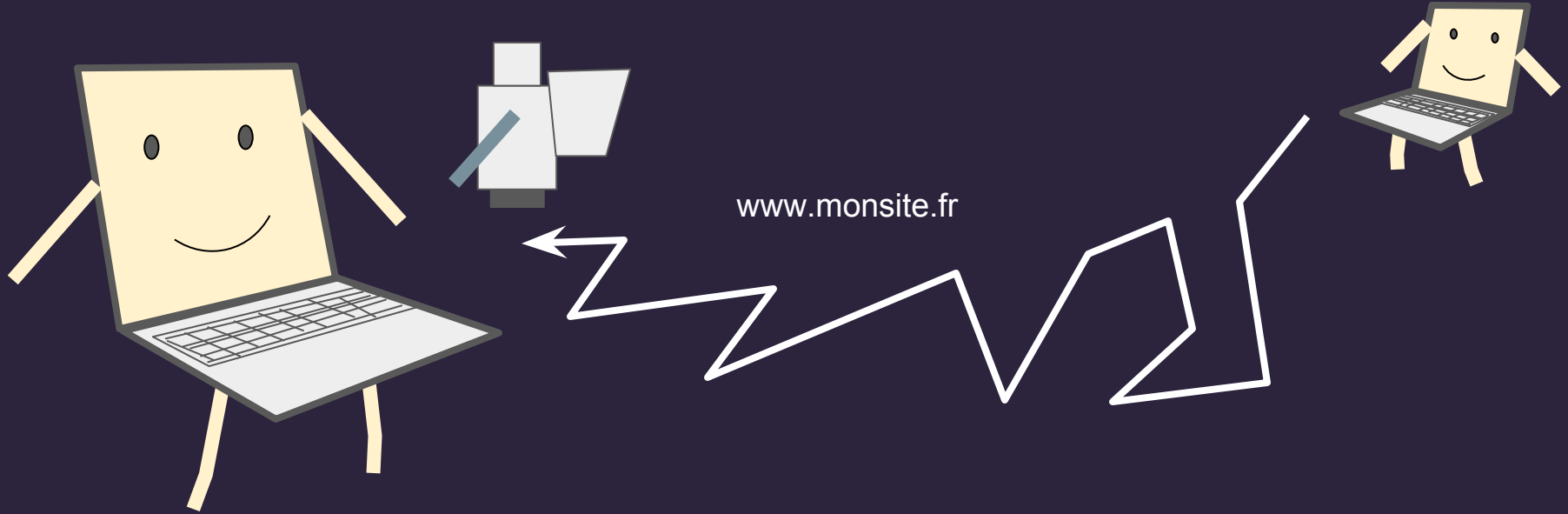


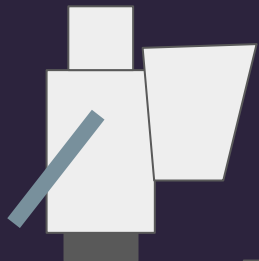
# Le world wide web est une application



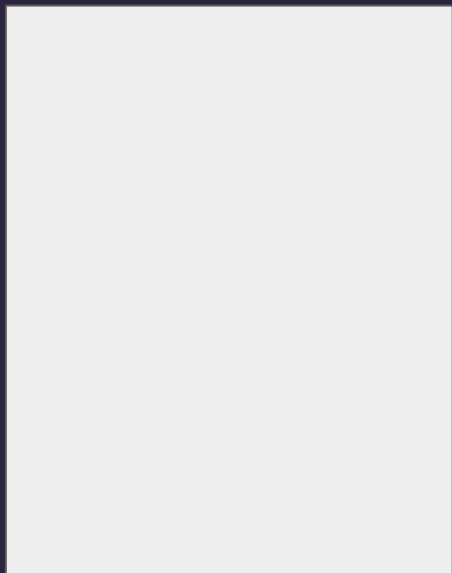
# qui se base sur Internet

# Qui te renvoie un site Internet

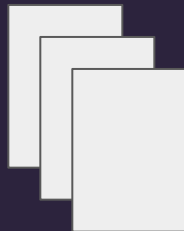
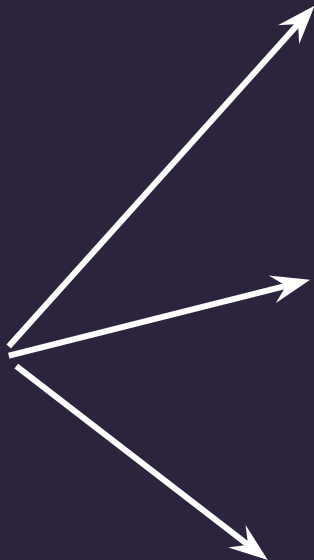




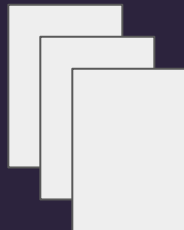
# Un site c'est...



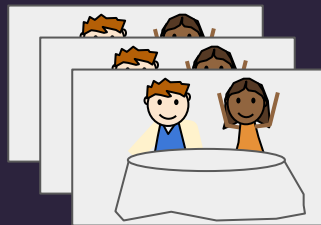
Un document HTML



Fichiers de style CSS pour mettre en forme la page et la rendre plus jolie

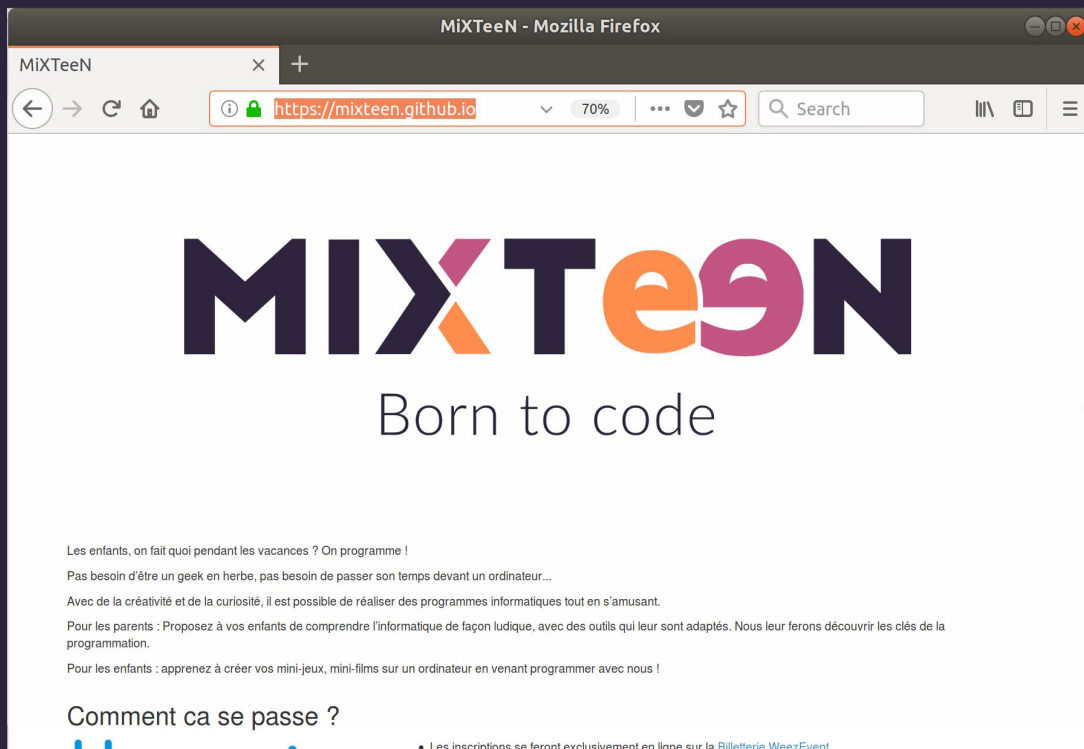


Fichiers JavaScript qui permettent d'ajouter des petits bouts de programme pour rendre la page plus dynamique



Images, vidéos, polices d'écritures...

# Un page HTML n'affiche que des images et des liens vers d'autres pages

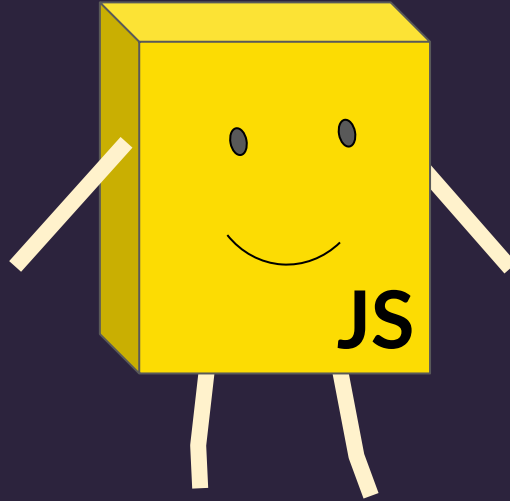




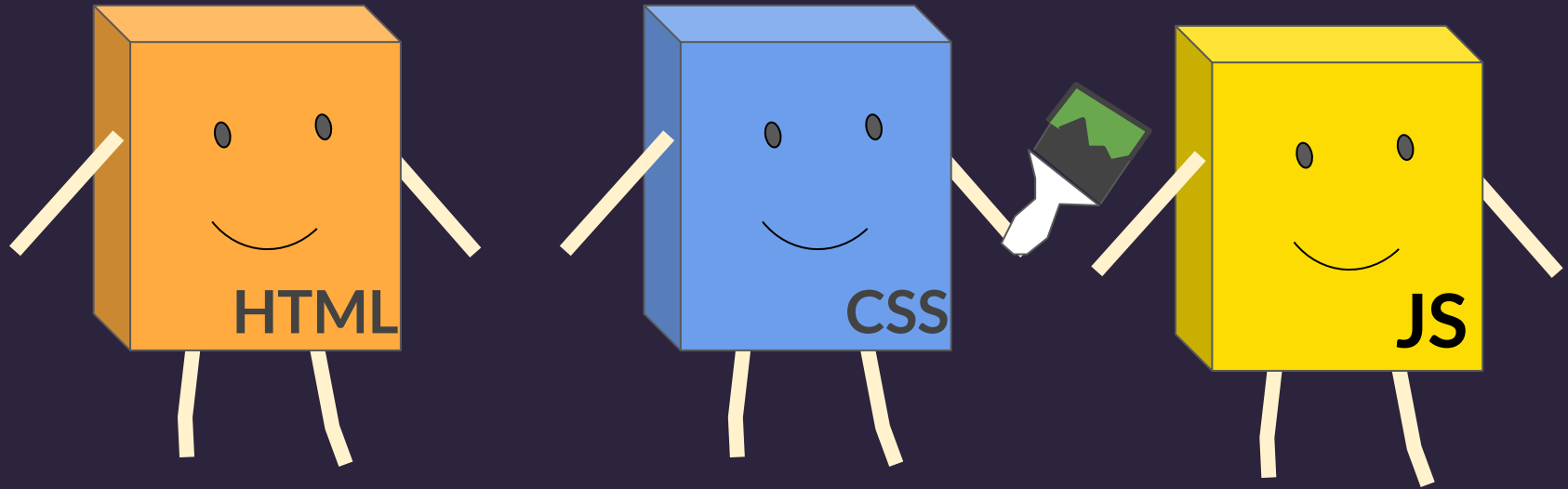
Rien n'est dynamique  
Pas d'animation  
Pas de petit programme

...

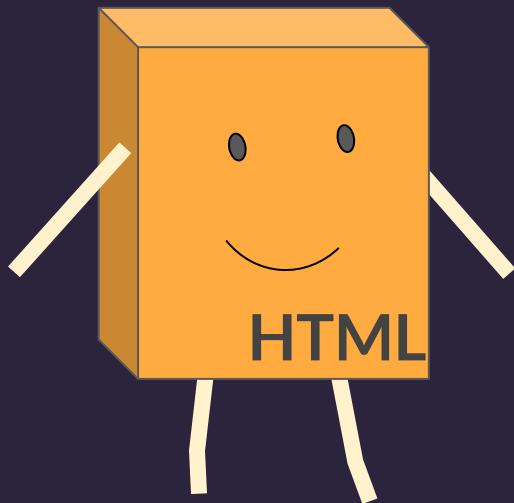
Pour changer tout ça  
JavaScript est apparu



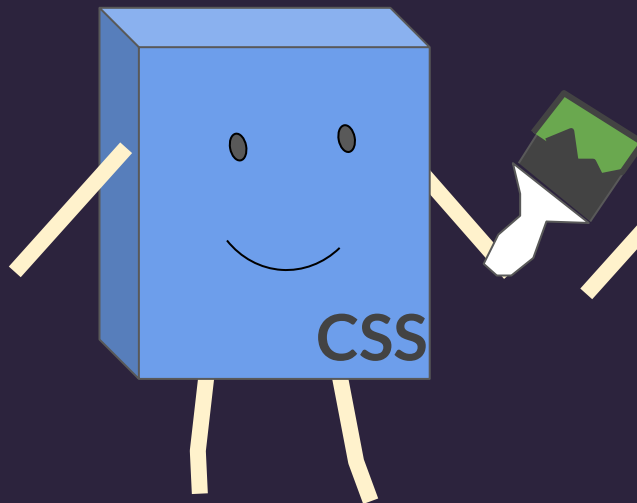
Pour faire de la programmation web tu  
peux donc utiliser



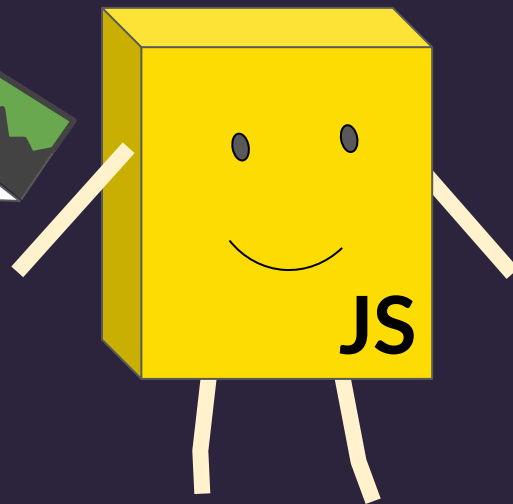
Le contenu



La peinture



La dynamique

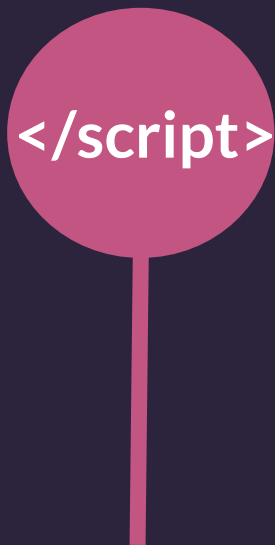


# Comment on fait du JavaScript ?





...



La balise `<script>` permet  
d'insérer du JavaScript dans une  
page HTML

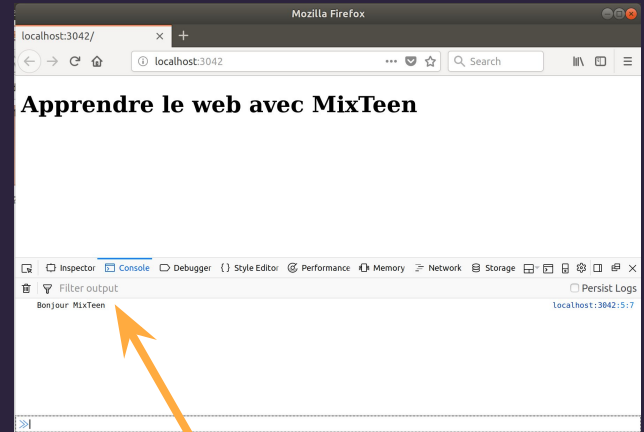
# Chaque ligne se termine par ;

```
<!doctype html>
<html>
<head>
  <script>
    console.log('Bonjour MixTeen');
  </script>
</head>
<body>
  <h1>Apprendre le web avec MixTeen</h1>
</body>
</html>
```

# L'instruction `console.log` permet d'écrire un message dans la console Développeur



Si tu lances l'exemple dans Firefox

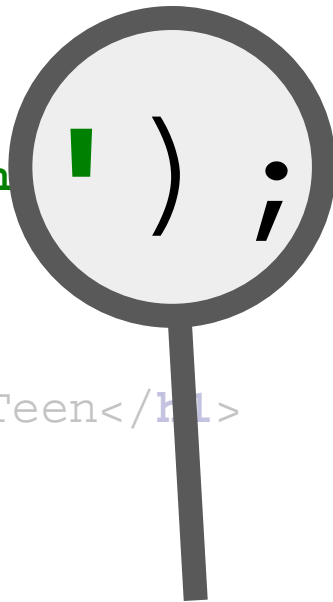


Ton message apparaît

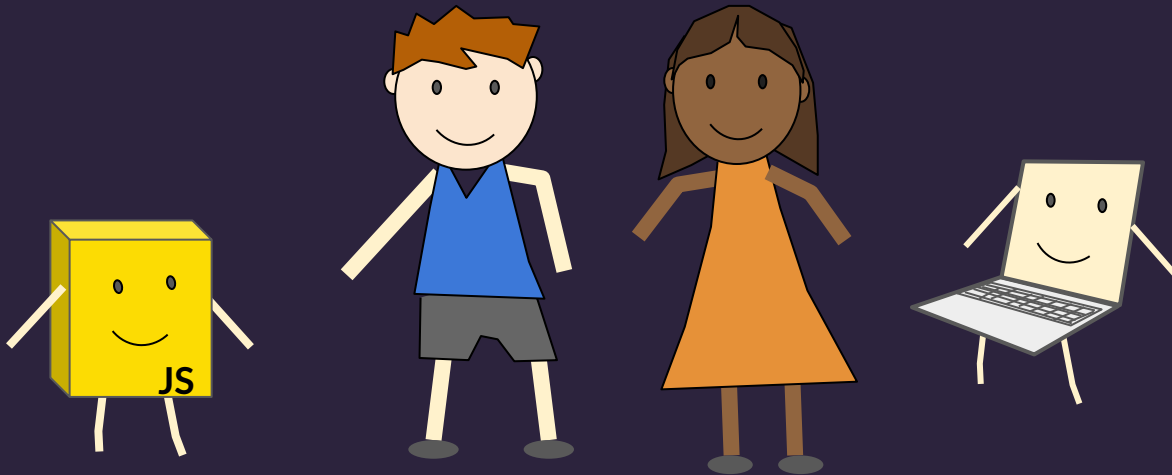


# Chaque ligne se termine par ;

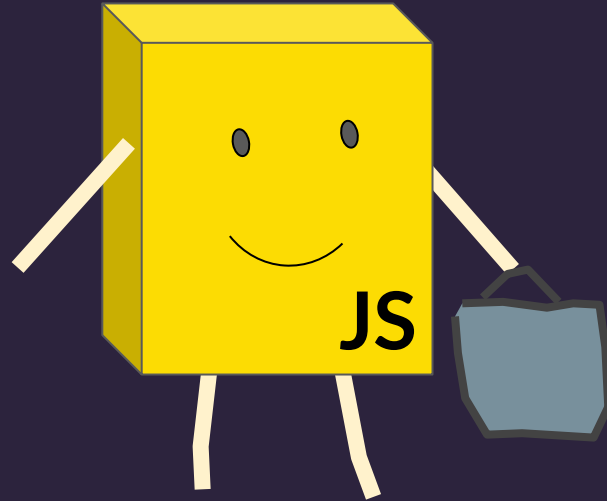
```
<!doctype html>
<html>
<head>
  <script>
    console.log('Bonjour MixTeen
  </script>
</head>
<body>
  <h1>Apprendre le web avec MixTeen</h1>
</body>
</html>
```



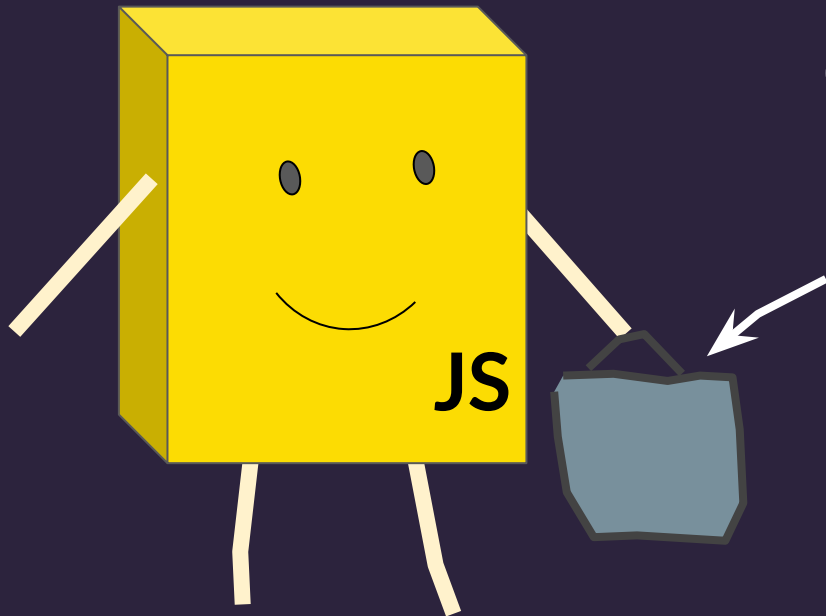
# Les variables



# Tu peux définir des variables en JavaScript



Une variable est comme un sac dans lequel tu peux stocker des choses et les changer



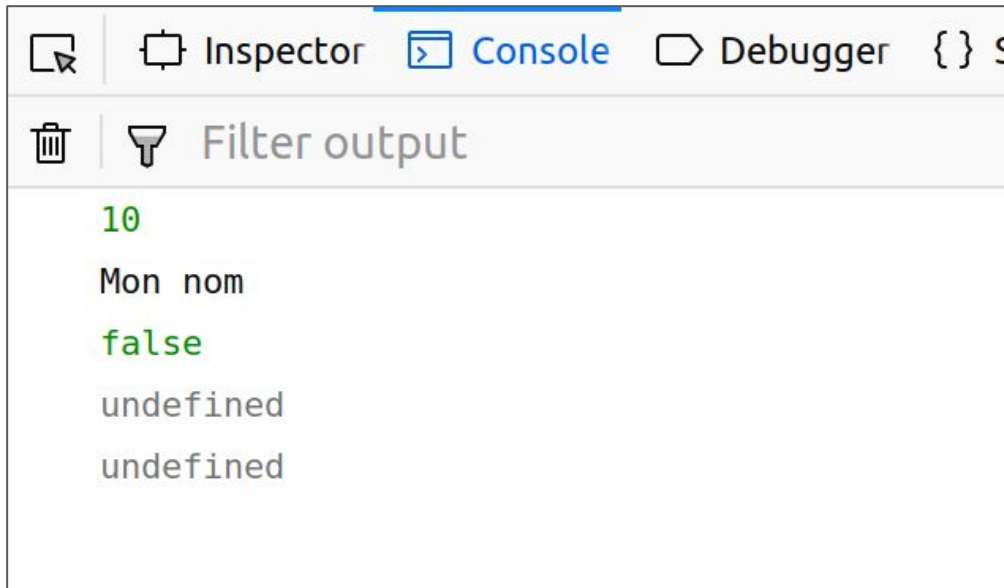
# On peut mettre à l'intérieur

- Un nombre
- Du texte
- Un booléen (**vrai** ou **faux**)
- Rien (**undefined** en JavaScript)

Pour créer une variable on utilise le mot clé **let**

# Définir des variables

```
let variable1 = 10;  
console.log(variable1);  
  
let variable2 = 'Mon nom';  
console.log(variable2);  
  
let variable3 = false;  
console.log(variable3);  
  
let variable4 = undefined;  
console.log(variable4);  
  
let variable5;  
console.log(variable5);
```



# Définir des variables

```
let variable = 10;
```

```
let variable = 30;
```

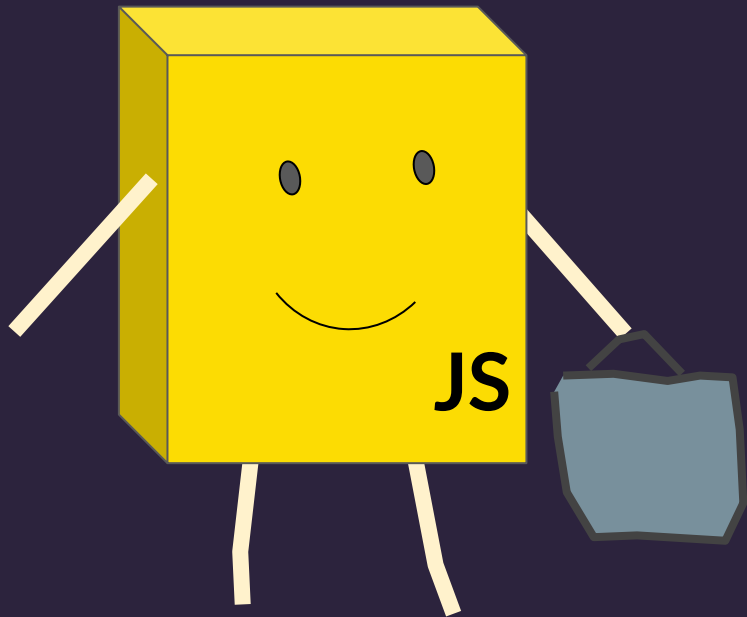


Si tu recrée une autre variable avec le même nom tu as une erreur

```
variable = 30;
```



Par contre tu as le droit de changer la valeur



Une variable a toujours un nom mais ce nom **ne doit pas avoir**

- D'espaces
- D'accents ou de caractères spéciaux

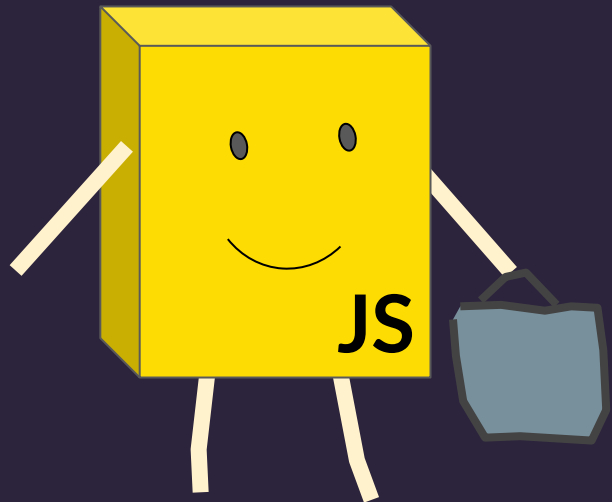
Ce nom est important car il te permet plus tard de retrouver l'information

# Définir des variables

```
let age = 10;  
let nom = 'Mon nom';  
let aimeLesMaths = true;
```



# JavaScript n'est pas toujours simple

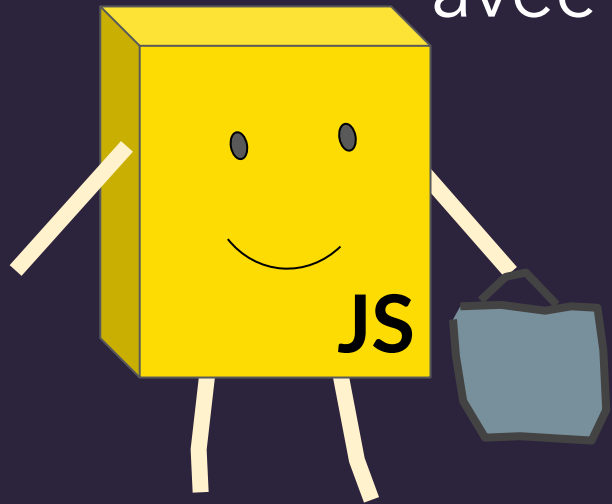


Par exemple pour définir du texte, une chaîne de caractères tu peux écrire

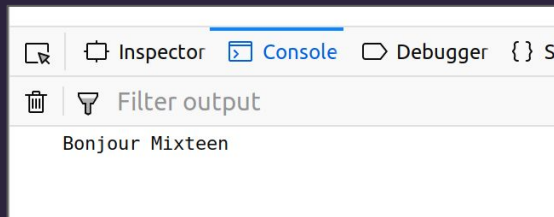
```
'Bonjour MixTeen'  
"Bonjour MixTeen"  
`Bonjour MixTeen`
```

On peut utiliser des quotes, double quotes, back ticks

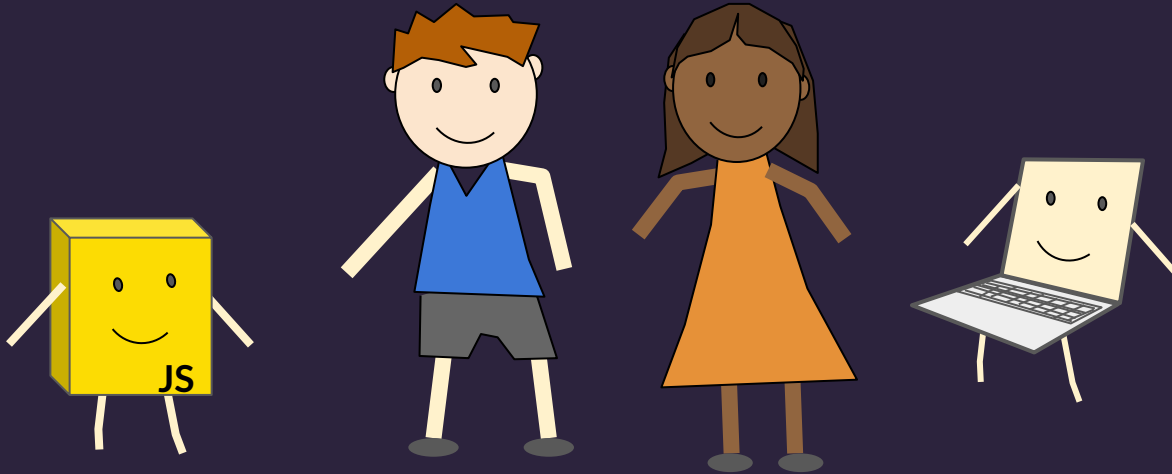
La dernière forme avec ``` est intéressante  
car elle permet l'interpolation de variables  
avec des moustaches `${}`



```
let nom = 'Mixteen';  
console.log(`Bonjour ${nom}`);
```



# Les commentaires



En JavaScript tu peux ajouter avec `//` des commentaires dans ton code

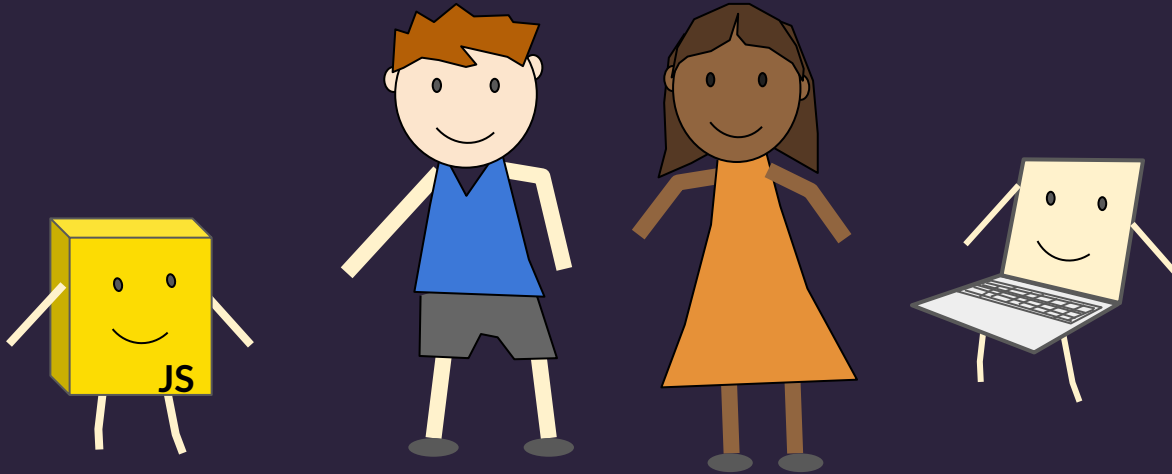
```
// Ceci est un exemple de moustache  
let nom = 'Mixteen';  
console.log(`Bonjour ${nom}`);
```

ou commenter plusieurs lignes avec `/*` et `*/`

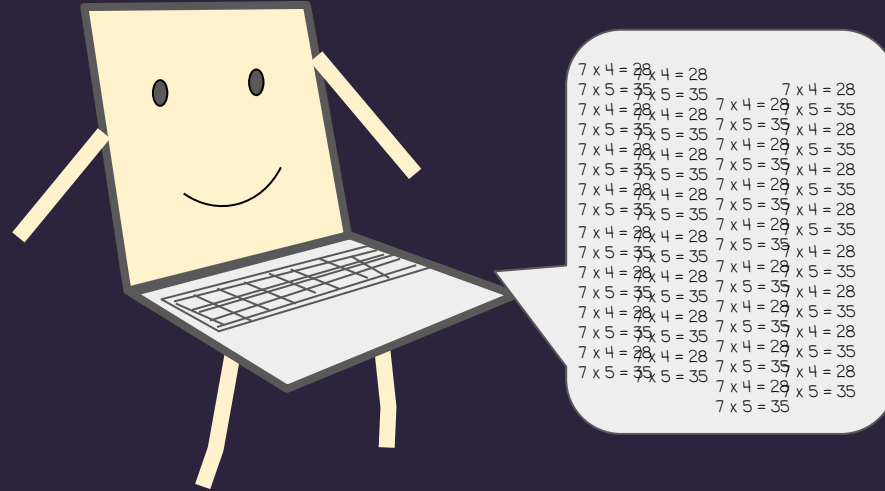
```
/*  
let nom = 'Mixteen';  
console.log(`Bonjour ${nom}`);  
*/
```

Les lignes commentées ne sont pas exécutées

# Les opérateurs



# Un ordinateur peut faire 2 milliards d'opérations par seconde



Une opération c'est

**variable** + **opérateur** + **variable**

# JavaScript possède donc plusieurs opérateurs

## Opérateurs arithmétiques

*	multiplication	<code>console.log(4 * 5)</code>	20
/	division	<code>console.log(40 / 5)</code> <code>console.log(40 / 0)</code>	8 Infinity
+	addition	<code>console.log(5 + 2)</code> <code>console.log('Mix' + 'Teen')</code>	7 MixTeen
-	soustraction	<code>console.log(4 - 5)</code> <code>console.log(8 - 5)</code>	-1 3
**	exponentiation	<code>console.log(5 ** 2)</code> <code>console.log(10 ** -1)</code>	25 0.1
=	égalité	<code>let resultat = 4 * 5</code> <code>console.log(resultat)</code>	20

# Tu peux aussi faire des comparaisons

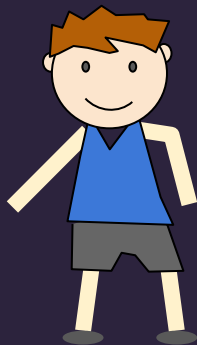
## Opérateurs de comparaison

===	égalité	<code>console.log(4 === 5)</code> <code>console.log('Mix' === 'Mix')</code>	false true
!==	inégalité	<code>console.log(4 !== 5)</code> <code>console.log('Mix' !== 'Mix')</code>	true false
>	addition	<code>console.log(5 &gt; 2)</code>	true
<	soustraction	<code>console.log(8 &lt; 5)</code>	false
<=	exponentiation	<code>console.log(5 &lt;= 5)</code> <code>console.log(10 &lt;= 11)</code>	true true
>=	égalité	<code>console.log(5 &gt;= 5)</code> <code>console.log(10 &gt;= 11)</code>	true false

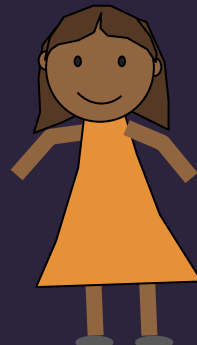


# Les opérateurs logiques

permettent de cumuler les comparaisons



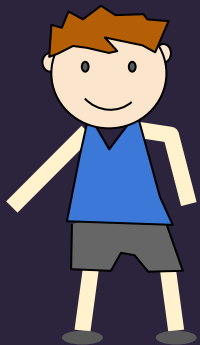
```
let sexe = 'garçon';  
let nom = 'Paul';  
let age = 11;  
let sport = 'foot';  
let aimeLesMaths = false;
```



```
let sexe = 'fille';  
let nom = 'Emma';  
let age = 12;  
let sport = 'basket';  
let aimeLesMaths = true;
```

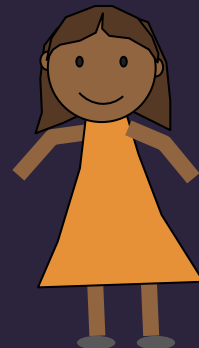
# Les opérateurs logiques

Jouons aux devinettes : quel opérateur utiliser ?



Je veux les ados qui ont un  
âge inférieur à 13 ans

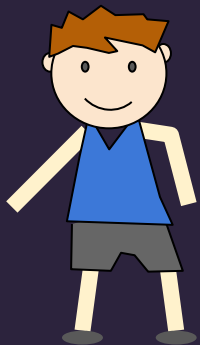
```
let sexe = 'garçon';  
let nom = 'Paul';  
let age = 11;  
let sport = 'foot';  
let aimeLesMaths = false;
```



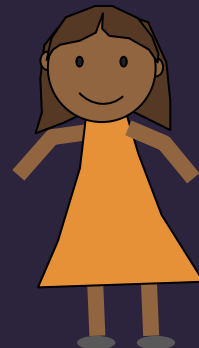
```
let sexe = 'fille';  
let nom = 'Emma';  
let age = 12;  
let sport = 'basket';  
let aimeLesMaths = true;
```

# Les opérateurs logiques

Jouons aux devinettes : quel opérateur utiliser ?

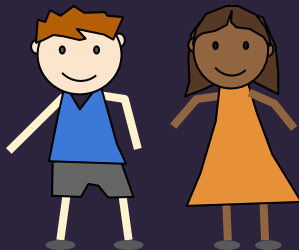


Je veux les ados qui ont un  
âge inférieur à 13 ans



```
let sexe = 'garçon';  
let nom = 'Paul';  
let age = 11;  
let sport = 'foot';  
let aimeLesMaths = false;
```

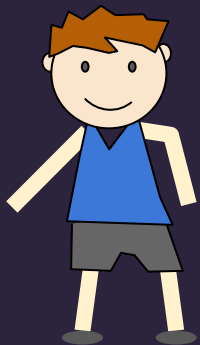
*age* < 13



```
let sexe = 'fille';  
let nom = 'Emma';  
let age = 12;  
let sport = 'basket';  
let aimeLesMaths = true;
```

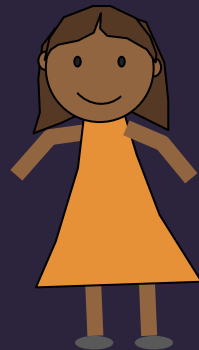
# Les opérateurs logiques

Jouons aux devinettes : quel opérateur utiliser ?



Je veux les ados qui ont un  
âge inférieur à 13 ans  
et  
qui sont une fille

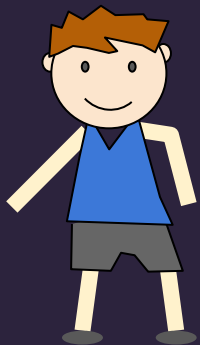
```
let sexe = 'garçon';  
let nom = 'Paul';  
let age = 11;  
let sport = 'foot';  
let aimeLesMaths = false;
```



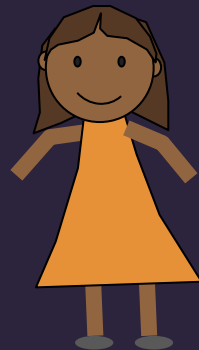
```
let sexe = 'fille';  
let nom = 'Emma';  
let age = 12;  
let sport = 'basket';  
let aimeLesMaths = true;
```

# Les opérateurs logiques

Jouons aux devinettes : quel opérateur utiliser ?



Je veux les ados qui ont un  
âge inférieur à 13 ans et  
qui sont une fille



```
let sexe = 'garçon';  
let nom = 'Paul';  
let age = 11;  
let sport = 'foot';  
let aimeLesMaths = false;
```




`age < 13` **ET**  
`sexe === 'fille'`

```
let sexe = 'fille';  
let nom = 'Emma';  
let age = 12;  
let sport = 'basket';  
let aimeLesMaths = true;
```

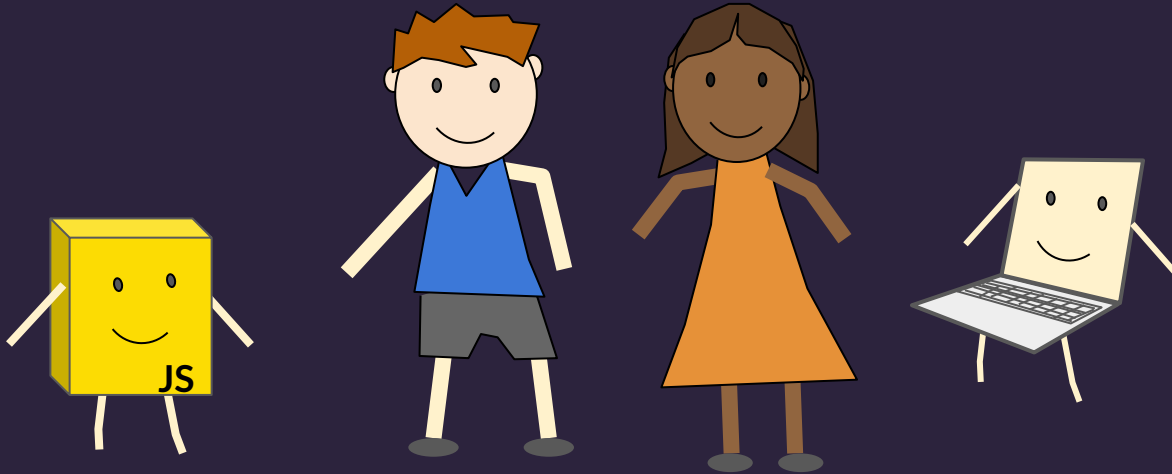
comment exprimer le ET ?

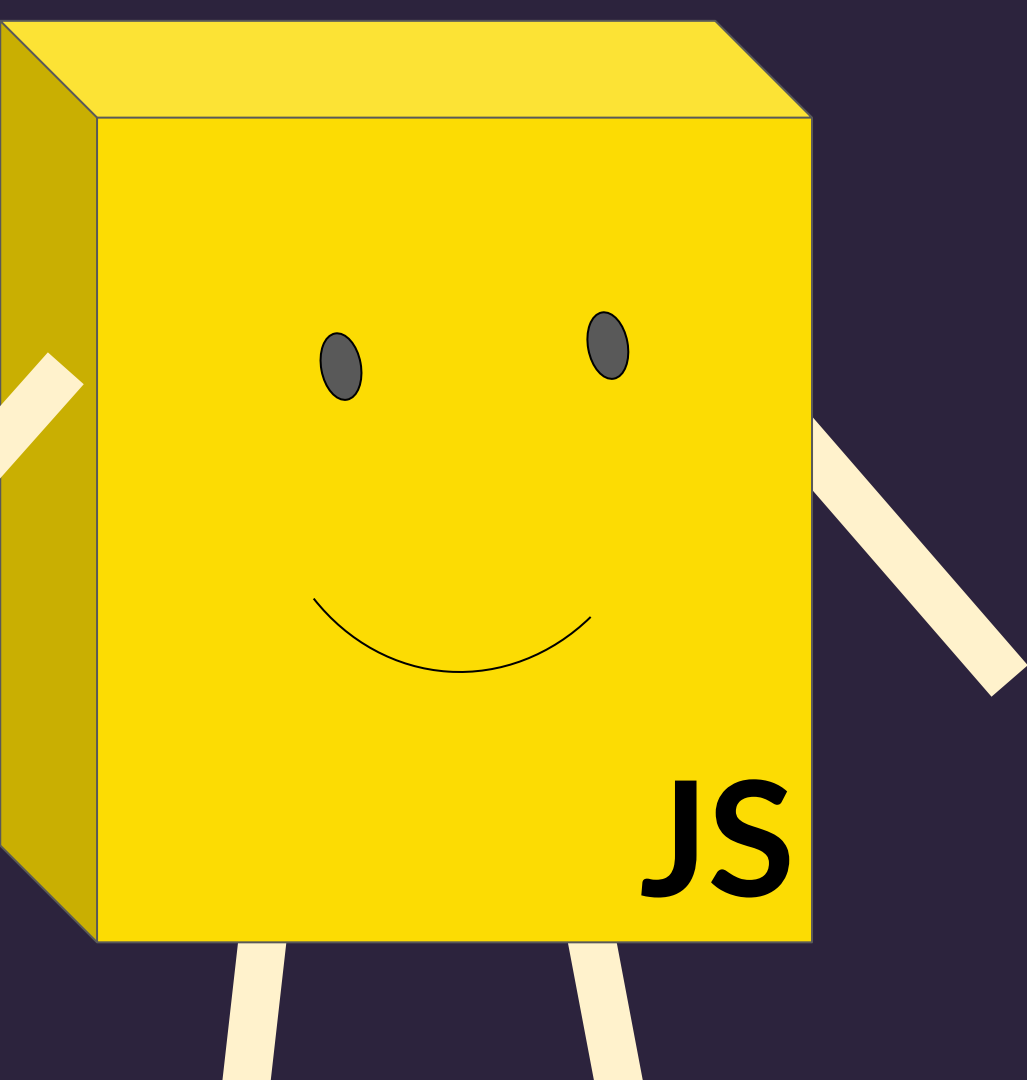
# Les opérateurs logiques

permettent de cumuler les comparaisons

Opérateurs logiques				
&&	ET	les ados qui ont un âge inférieur à 13 ans ET qui sont une fille	<code>age &lt; 13 &amp;&amp; sexe === 'fille'</code>	
	OU	les ados qui ont un âge supérieur à 10 ans et qui sont un garçon	<code>age &gt; 10 &amp;&amp; sexe === 'garçon'</code>	
!	NON	les ados n'aiment pas les maths	<code>!aimeLesMaths</code>	

# Les blocs d'instructions





Un programme  
JavaScript peut être  
très gros. On peut  
organiser les  
instructions en es  
regroupant par blocs  
avec { et }



# Définir des blocs

```
{  
  // Bloc 1  
  // liste des instructions  
}  
{  
  // Bloc 2  
  // liste des instructions  
  {  
    // Sous Bloc  
    // liste des instructions  
  }  
}
```

J'ai défini 3 blocs

- Bloc 1 et 2 qui se suivent
- Et sous bloc qui est imbriqué dans le bloc 2

# Définir des blocs

```
let nom = 'Paul';  
console.log(`Bonjour ${nom}`);  
{  
  // Bloc 1  
  let nom = 'Elodie';  
  console.log(`Bonjour ${nom}`);  
}  
console.log(`Bonjour ${nom}`);
```

Affiche => Bonjour Paul

# Définir des blocs

```
let nom = 'Paul';
console.log(`Bonjour ${nom}`);
{
  // Bloc 1
  let nom = 'Elodie';
  console.log(`Bonjour ${nom}`);
}
console.log(`Bonjour ${nom}`);
```

Affiche => Bonjour Paul

J'ai le droit de redéfinir une variable *nom* car nous ne sommes dans un sous bloc  
Affiche => Bonjour Elodie

# Définir des blocs

```
let nom = 'Paul';  
console.log(`Bonjour ${nom}`);  
{  
  // Bloc 1  
  let nom = 'Elodie';  
  console.log(`Bonjour ${nom}`);  
}  
console.log(`Bonjour ${nom}`);
```

Affiche => Bonjour Paul

J'ai le droit de redéfinir une variable *nom* car nous ne sommes dans un sous bloc  
Affiche => Bonjour Elodie

Affiche => Bonjour Paul  
Car ce qui est défini dans le sous bloc n'est pas connu du bloc principal

# Définir des blocs

```
let nom = 'Paul';  
console.log(`Bonjour ${nom}`);  
{  
  // Bloc 1  
  let nom = 'Elodie';  
  console.log(`Bonjour ${nom}`);  
}  
console.log(`Bonjour ${nom}`);
```

Affiche => Bonjour Paul

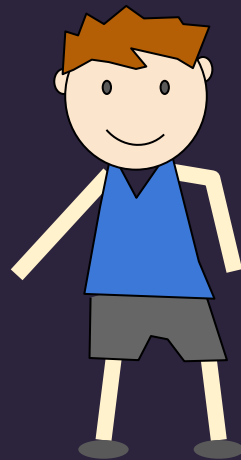
J'ai le droit de redéfinir une variable *nom* car nous ne sommes dans un sous bloc  
Affiche => Bonjour Elodie

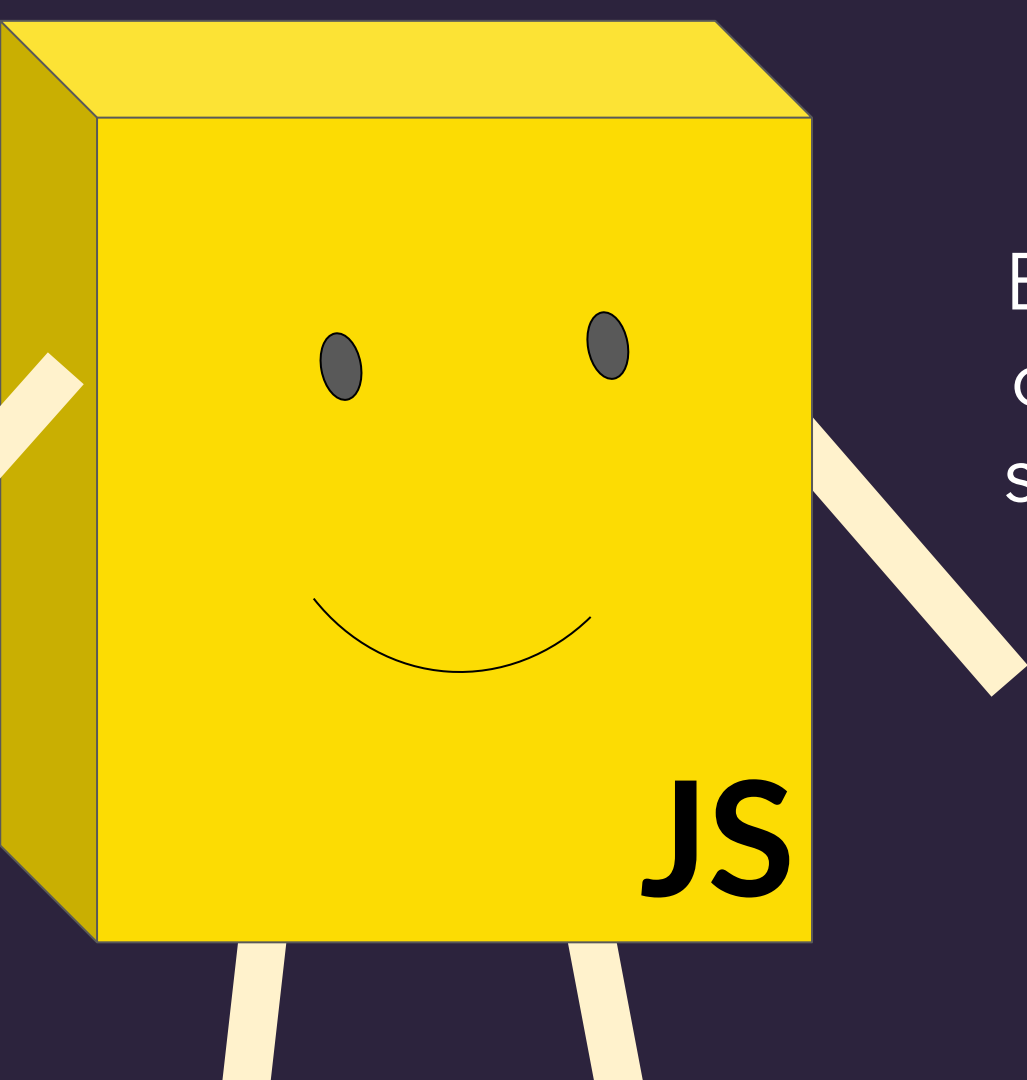
Affiche => Bonjour Paul  
Car ce qui est défini dans le sous bloc n'est pas connu du bloc principal

Si on appelle le bloc principal le  
parent et le sous bloc le fils

Le fils connaît tout ce qui a été fait  
dans le parent mais le parent ne  
connait pas ce que le fils a fait...

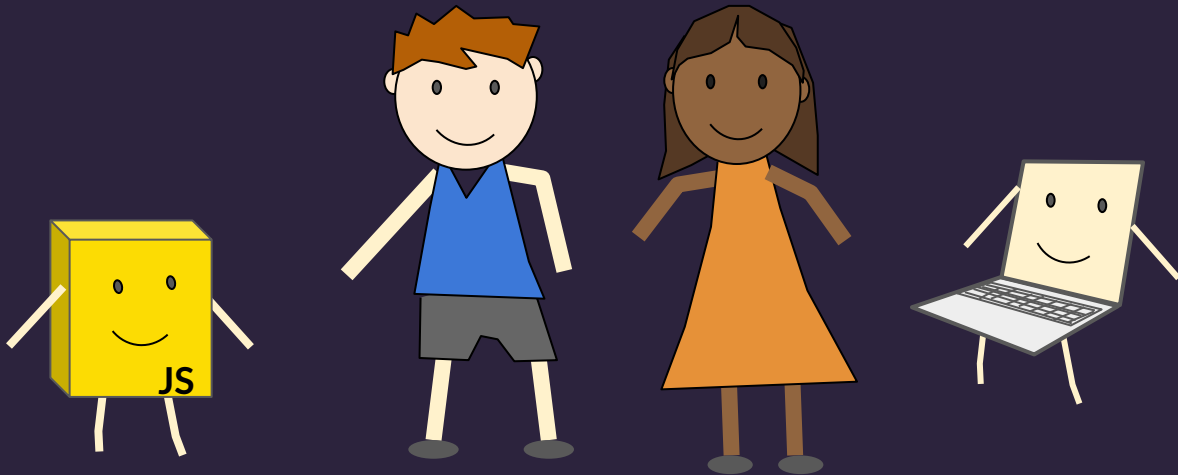
Cool non ?



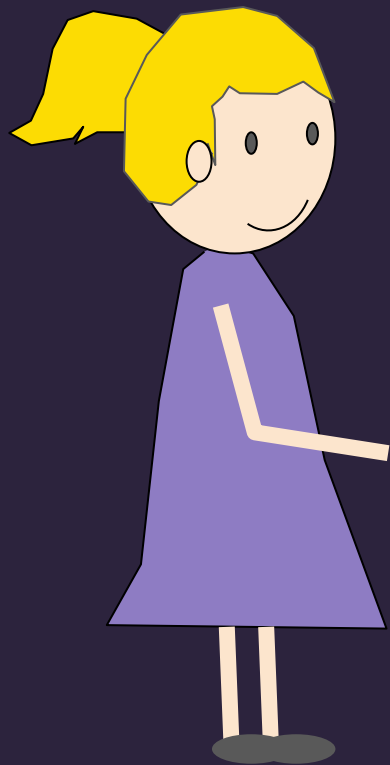


En JavaScript les blocs  
définis avec { et } sont  
surtout utiliser avec les  
conditions et les  
fonctions

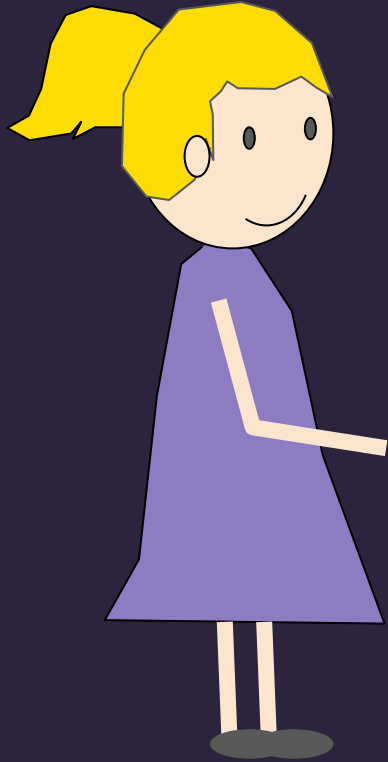
# Les conditions





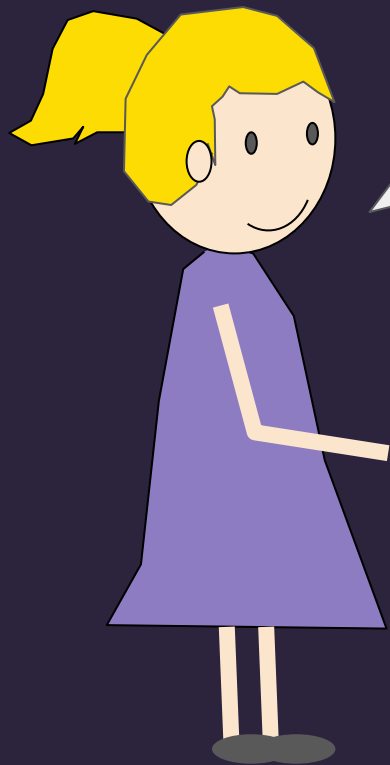


Comment exprime  
t-on une condition ?

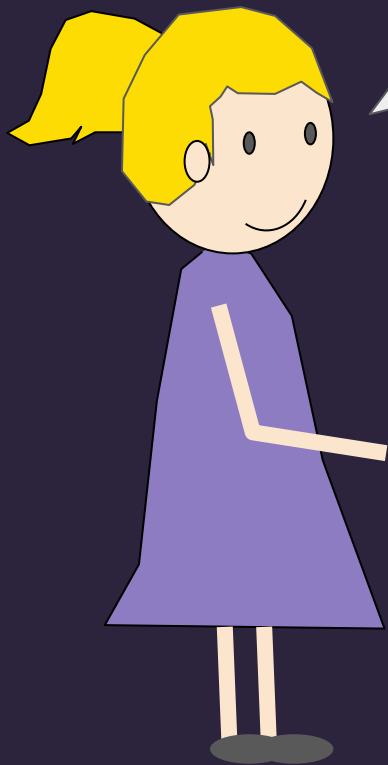


Pour exprimer une condition  
on utilise le mot SI

Et SINON quand la condition  
n'est pas remplie



Si tu as plus de 13 ans ALORS  
tu pourras avoir une tablette  
SINON  
tu n'auras pas de tablette



Si tu as plus de 13 ans ALORS  
tu pourras avoir une tablette  
SINON  
tu n'auras pas de tablette

Se traduit en JavaScript par

```
let asUneTablette = undefined;  
if (age > 13) {  
    asUneTablette = true;  
}  
else {  
    asUneTablette = false;  
}
```

# D'autres exemples

```
if (nom === 'Paul' && !aimeLesMaths) {  
    console.log(`Je préfère jouer au ${foot}`);  
}  
  
if (sexe === 'fille') {  
    if (age >= 18) {  
        console.log(`Je suis une fille majeure`);  
    }  
    else {  
        console.log(`Je suis une fille mineure`);  
    }  
}
```

# Les fonctions



# Je veux afficher l'âge d'une personne

```
let annee = 2018;  
let anneeNaissance;  
let prenom;  
let age;  
  
anneeNaissance = 2008;  
prenom = 'Paul';  
age = annee - anneeNaissancePaul;  
console.log(`En ${annee} ${prenom} a ${age} ans`);
```

Affiche => En 2018 Paul a 10 ans

# Je veux afficher l'âge d'une personne

```
let annee = 2018;;  
let anneeNaissance;  
let prenom;  
let age;
```

```
anneeNaissance = 2008;  
prenom = 'Paul';  
age = annee - anneeNaissancePaul;  
console.log(`En ${annee} ${prenom} a ${age} ans`);
```

Affiche => En 2018 Paul a 10 ans

```
anneeNaissance = 2006;  
prenom = 'Emma';  
age = annee - anneeNaissancePaul;  
console.log(`En ${annee} ${prenom} a ${age} ans`);
```

Affiche => En 2018 Emma a 12 ans



# Je veux afficher l'âge d'une personne

```
let annee = 2018;;  
let anneeNaissance;  
let prenom;  
let age;
```

```
anneeNaissance = 2008;  
prenom = 'Paul';  
age = annee - anneeNaissancePaul;  
console.log(`En ${annee} ${prenom} a ${age} ans`);
```

```
anneeNaissance = 2006;  
prenom = 'Emma';  
age = annee - anneeNaissancePaul;  
console.log(`En ${annee} ${prenom} a ${age} ans`);
```

# Une fonction permet de regrouper des instructions dans un bloc

```
function nomFonction(parametre1, parametre2) {  
    let result = ...;  
    return result;  
}
```

Elle peut avoir des paramètres et retourner un résultat mais ce n'est pas obligatoire

# Une fonction

Se déclare avec le mot clé **function**

a un nom qui ne contient ni accent, ni caractères spéciaux ni espaces

```
function nomFonction(parametres) {  
    let result = ...;  
    return result;  
}
```

a un bloc d'instructions délimité par { et }

Peut avoir des paramètres, ce sont des variables

Peut renvoyer un résultat avec **return**

# Pour appeler la fonction

```
function nomFonction(parametres) {  
    let result = ...;  
    return result;  
}
```

on reprend son nom et on indique les  
paramètres

```
nomFonction(parametres) ;
```

Si pas de paramètres ce sera

```
nomFonction() ;
```

# Je veux afficher l'âge d'une personne

```
let annee = 2018;;  
let anneeNaissance;  
let prenom;  
let age;
```

```
anneeNaissance = 2008;  
prenom = 'Paul';  
age = annee - anneeNaissancePaul;  
console.log(`En ${annee} ${prenom} a ${age} ans`);
```

```
anneeNaissance = 2006;  
prenom = 'Emma';  
age = annee - anneeNaissancePaul;  
console.log(`En ${annee} ${prenom} a ${age} ans`);
```

# Je veux afficher l'âge d'une personne

```
function calculerAge(prenom, anneeNaissance) {  
  let annee = 2018;  
  let age = annee - anneeNaissancePaul;  
  console.log(`En ${annee} ${prenom} a ${age} ans`);  
  return age;  
}
```

```
calculerAge('Paul', 2008);
```

```
calculerAge('Emma', 2006);
```

# Lire et écrire des éléments de la page HTML



# Une page avec un champ, un bouton, un message

```
<input type="number" placeholder="Saisir un nombre A">  
<input type="number" placeholder="Saisir un nombre B">  
<button>Additionner</button>  
<div>A + B = ?</div>
```

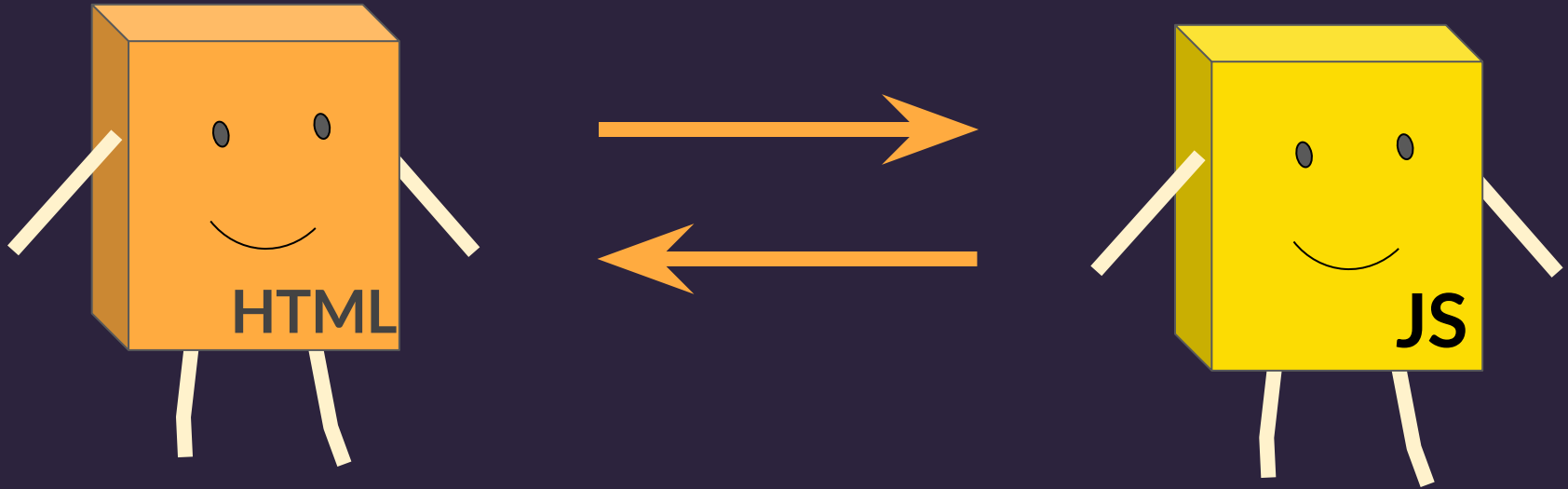
Saisir un nombre A	Saisir un nombre B	Additionner
--------------------	--------------------	-------------

A + B = ?

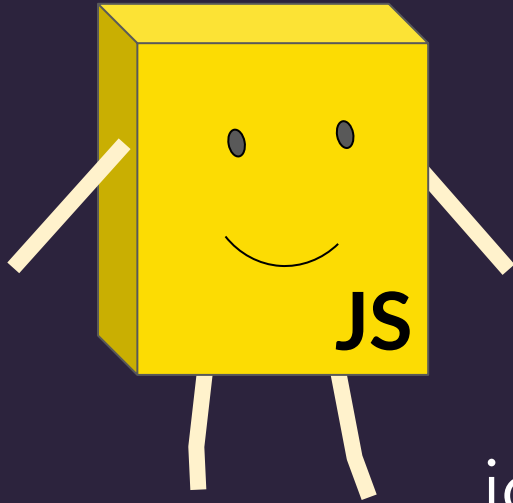
Le but est de mettre à jour le message contenu dans le div avec le résultat



# Il existe plusieurs moyens d'échanger des informations entre le HTML et JavaScript



En JavaScript tu peux utiliser `document.getElementById(id)` pour sélectionner un élément de la page HTML.



Pour que ça fonctionne ton élément HTML doit avoir un identifiant défini avec la propriété `id`

# Lire une donnée en JavaScript

```
<input id="nombreA" type="number" placeholder="Saisir un nombre A" value="10">
```

```
<input id="nombreB" type="number" placeholder="Saisir un nombre B">
```

```
<button>Additionner</button>
```

```
<div id="resultat">A + B = ?</div>
```

Des propriétés id on été ajoutées aux champs nombre et au texte correspondant au résultat

```
<script>
```

```
console.log(document.getElementById('nombreA').value);
```

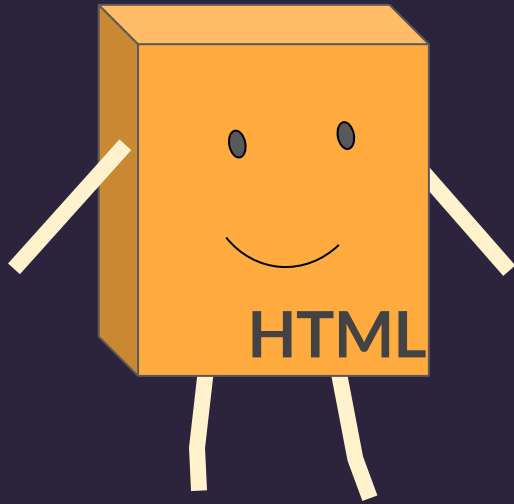
```
console.log(document.getElementById('resultat').innerHTML);
```

```
</script>
```

Affiche dans la console

10

A + B = ?



Plusieurs balises ont des propriétés permettant de pointer vers une fonction JavaScript quand l'utilisateur interagit avec cette balise

# Propriétés pour lier une balise à du code JS

click	<pre>&lt;button onclick = "clickBouton()"&gt; Additioner&lt;/button&gt;</pre>	Appelle la fonction <code>clickBouton</code> quand une personne clique sur le bouton
change	<pre>&lt;input type = "number" onchange = "changeNombre(this.value)"&gt;</pre>	Appelle la fonction <code>changeNombre</code> en envoyant la valeur quand le champ a été modifié (quand on sort du champ)
keyup	<pre>&lt;input type = "number" onkeyup = "changeNombre(this.value)"&gt;</pre>	Appelle la fonction <code>changeNombre</code> en envoyant la valeur quand le champ a été modifié (à chaque fois qu'une touche est relâchée)
load	<pre>&lt;body load = "initialiser()"&gt;&lt;/body&gt;</pre>	S'utilise sur la balise <code>&lt;body&gt;</code> pour lancer du code JavaScript une fois que la page HTML a été chargée

# Ajout de onclick sur button

```
<input type="number" placeholder="Saisir un nombre A">  
<input type="number" placeholder="Saisir un nombre B">  
<button onclick="additionner()">Additionner</button>  
<div>A + B = ?</div>
```

Nous devons maintenant écrire la fonction additionner. Elle sera lancée à chaque fois que quelqu'un clique sur le bouton

# Ajout de onclick sur button

```
<script>
function additionner() {
    let a = document.getElementById('nombreA').value;
    let b = document.getElementById('nombreB').value;

    document.getElementById('resultat').innerHTML =
        `A + B = ${a + b}`;
}
</script>
```

On peut tester maintenant la page. Saisit 10 et 5, le résultat est...

# Ajout de onclick sur button

```
<script>
function additionner() {
    let a = document.getElementById('nombreA').value;
    let b = document.getElementById('nombreB').value;

    document.getElementById('resultat').innerHTML =
        `A + B = ${a + b}`;
}
</script>
```

On peut tester maintenant la page. Saisit 10 et 5, le résultat est... **105**  
**getElementById retourne toujours une chaîne de caractères** et quand on additionne 2 chaînes de caractères, elles sont concaténées (les caractères de la deuxième sont ajoutés à droite de la première)



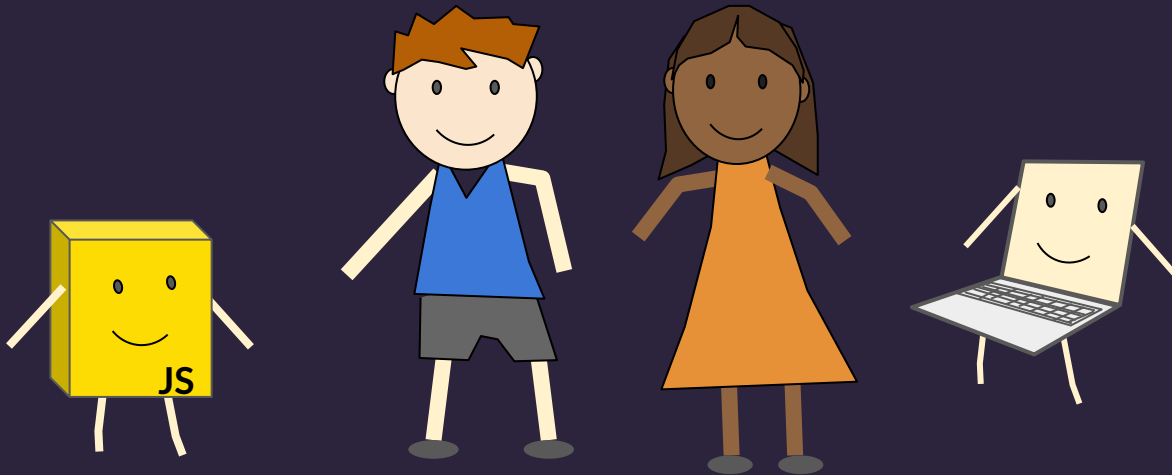
# parseInt convertit une chaîne en nombre

```
function additionner() {  
    let a = 0;  
    let b = 0;  
  
    if(document.getElementById('nombreA').value) {  
        a = parseInt(document.getElementById('nombreA').value);  
    }  
    if(document.getElementById('nombreB').value) {  
        b = parseInt(document.getElementById('nombreB').value);  
    }  
  
    document.getElementById('resultat').innerHTML =  
    `A + B = ${a + b}`;  
}
```

# on peut aussi faire plus joli

```
function lireNombre(id){  
    if(document.getElementById(id).value){  
        return parseInt(document.getElementById(id).value);  
    }  
    return 0;  
}  
  
function additionner() {  
    document.getElementById('resultat').innerHTML =  
    `A + B = ${lireNombre('nombreA') + lireNombre('nombreB')}`;  
}
```

# Les boucles



# Les objets

