

COMPARAISONS ET CONDITIONNELLES

Objectifs

- Créer différents types d'embranchements conditionnels.
- Revoir les opérateurs de comparaison classiques
- Savoir utiliser les opérateurs de comparaison “strictes”.

```
1 > 2 // false
2 >= 2 // true
1 < 2 // true
2 <= 2 // true
```

Une comparaison retourne systématiquement une valeur booléenne.

JavaScript utilise les opérateurs de comparaison classiques, sauf pour les égalités.

```
"sympa" == "sympa" // true  
"1" == 1 // true  
0 == false // true
```


Le **==** vérifie l'égalité de valeur, mais pas de type.

Les deux valeurs sont converties dans le même type, puis comparées.

Ceci peut produire des résultats innatendus.

```
"sympa" === "sympa" // true  
"1" === 1 // false  
0 === false // false
```

Le **===** vérifie l'égalité de
valeur et de type.

Utilisez  dans
99.99% des cas!

```
let password = "sympa";  
  
if(password.length ≥ 6) {  
    console.log("valide");  
} else {  
    console.log("invalide");  
}
```

**On fait des comparaisons pour
créer des embranchements
dans notre programme.**

```
let password = "sympa";

if(password.length ≥ 6) {
    if(password.indexOf(" ") === -1) {
        console.log("valide");
    } else {
        console.log("contient des espaces");
    }
} else {
    console.log("doit être plus long");
}
```

Les structures conditionnelles peuvent être imbriquées et contrôlées par les classiques **if**, **else if**, et **else**

Les opérateurs logiques **||** (OR), **&&** (AND) et **!** (NOT) sont valides en JavaScript

L'opérateur de coalescence des nuls **??**,
permet d'attribuer facilement une valeur qui
serait de type null ou undefined.

```
let user;  
console.log(user ?? "anonymous"); // Anonymous  
  
// Définissons user  
let user = "Bruce";  
console.log(user ?? "anonymous"); // Bruce
```

condition ? value1 : value2;

EXPRESSION À ÉVALUER

VALEUR SI TRUE

VALEUR SI FALSE

The diagram illustrates the syntax of a ternary operator. The text 'condition ? value1 : value2;' is displayed in a monospaced font. The question mark '?' and colon ':' are highlighted in pink. Three yellow vertical lines with horizontal caps at the top point to specific parts of the syntax: one points to 'condition', another points to 'value1', and a third points to 'value2'. Below each line is a label in French: 'EXPRESSION À ÉVALUER' (Expression to evaluate) under 'condition', 'VALEUR SI TRUE' (Value if true) under 'value1', and 'VALEUR SI FALSE' (Value if false) under 'value2'.

```
let age = 2;  
let message = (age > 1) ? "ans" : "an";  
console.log(`${age} ${message}`)
```

L'opérateur ternaire **?** permet d'attribuer facilement une valeur conditionnelle à une variable.

On peut également les enchaîner.

“Falsy” / “Truthy”

L’instruction `if (...)` évalue l’expression entre parenthèses et la converti en type booléen. Une valeur non comparée a donc une valeur booléenne intrinsèque.

“ ”	FALSE
null	FALSE
undefined	FALSE
NaN	FALSE
0	FALSE
N’importe quelle autre valeur	TRUE

```
let a = 2 + 2;

switch (a) {
  case 3:
    alert( 'Too small' );
    break;
  case 4:
    alert( 'Exactly!' );
    break;
  case 5:
    alert( 'Too big' );
    break;
  default:
    alert( "🙄" );
}
```

**Si vous devez comparer
une variable avec plein de
variantes, privilégiez
l'instruction `switch`**