Les conditions

Description Sommaire

À propos de ce tutoriel

Lorsque l'on souhaite écrire des algorithmes il est important d'être capable de tester si une valeur est bien celle attendue. On va donc découvrir maintenant les conditions.

Les conditions s'écrivent de la manière suivante :

```
if (<booleen>) {
    <code si vrai>
}
```

Si par exemple on souhaite afficher à l'utilisateur une phrase si un nombre est pair :

```
// % donne le reste de la division de $nombre par 2
if (nombre % 2 == 0) {
    console.log(`Bravo !`)
}
```

Il est aussi possible de mettre un code à exécuter si la condition n'est pas remplie :

```
if (<booleen>) {
    <code si vrai>
} else {
    <code si faux>
}
```

Enfin on est parfois amené à utiliser des conditions multiples :

```
if (<booleen>) {
    <code si vrai>
} else if (<booleen>) {
    <code>
} else {
    <code>
}
```

Opérateur de comparaison

Parfois on souhaitera tester si une variable est bien celle attendue, ou faire une condition plus complexe. Dans ce cas là il faudra utiliser les opérateurs de comparaison

```
a == b // a égale à b
a === b // a == b et a est de même "type" que b
a >= b // a supérieur OU égal à b
a > b   // a strictement supérieur à b
a <= b // a inférieur OU égal à b
a < b   // a strictement inférieur à b
a != b // a est différent de b
a !== b // a est strictement différent de b
```

On essaiera tant que possible d'éviter les `==` et `!=` car ces comparaisons se basent sur une conversion implicite des types qui peut être surprenante.

```
'1' == 1 // true
'1' === 1 // false
```

Attention aussi lorsque l'on compare des objets, ils ne sont pas considérés comme égaux même si leurs propriétés sont identiques.

```
{a: 1} == {a: 1} // false
{} == {} // false
```

```
[] == [] // false
NaN == NaN // false
```

On pourra aussi utiliser les opérateurs booléens, qui nous permettront de combiner plusieurs conditions ensembles

```
// && ET
true && true // true
true && false // false
false && true // false
false && false // false

// || OU
true || true // true
true || false // true
false || true // true
false || false // false

// ! NON
!true // false
!false // true
```

Le switch / case

Le switch case permet d'effectuer une opération suivant la valeur que prendra une expression.

```
switch (expression) {
  case valeur1:
    // Instructions à exécuter lorsque le résultat
    // de l'expression correspond à valeur1
    instructions1
    break
  case valeur2:
    // Instructions à exécuter lorsque le résultat
    // de l'expression correspond à valeur2
    break
  default:
    // Instructions à exécuter lorsqu'aucune des valeurs
    // ne correspond
```

```
    break
}
```

C'est une notation qui est plus spécifique que les conditions avec les **if** et **else** mais qui peut s'avérer plus simple à écrire dans certains cas spécifiques

Le ternaire

Le ternaire, ou opérateur conditionnel est un outil qui permet de racourcir une condition en une seule ligne

```
// condition ? <expression si vrai> : <expression si faux>
const age = 19
const phrase = "Je suis " + (age >= 18 ? "majeur" : "mineur")
```

Exercices

Système de recommandation

20:56 - Créer un système de recommandation qui conseille le bon film en fonction de l'âge de l'utilisateur

- Si l'utilisateur a moins de 13 ans (13 ans inclu) on lui affichera "Lilo & Stitch"
- Si l'utilisateur a plus de 13 ans et moins de 18 ans (strictement) on lui affichera "Matrix" (je sais c'est un peu jeune)
- Si l'utilisateur a plus de 18 ans on lui affichera "Evil Dead"

On commencera avec le code suivant

```
const currentYear = 2022
const birthyear = prompt('Quel est votre année de naissance ?')
// Ecrire votre code ici, afficher le film à l'aide de console.log('votre réponse')
```

Voir la réponse

Multiplication

25:40 - On souhaite créer une calculatrice simplifiée qui est capable de multiplier 2 nombres et de nous donner le signe du résultat. L'objectif est d'afficher

`nombre1 x nombre2 = resultat est positif`

On commencera avec la base suivante

```
const a = prompt('Entrez un premier nombre')
const b = prompt('Entrez un second nombre')
```

Voir la réponse