codecademy

Conditionnels

Flux de contrôle

Le flux de contrôle est l'ordre dans lequel les instructions sont exécutées dans un programme. Le flux de contrôle par défaut permet aux instructions d'être lues et exécutées dans l'ordre de gauche à droite, de haut en bas dans un fichier programme.
Les structures de contrôle telles que les conditions (`if` instructions et autres) modifient le flux de contrôle en n'exécutant des blocs de code que si certaines conditions sont remplies. Ces structures permettent essentiellement à un programme de prendre des décisions sur le code à exécuter pendant l'exécution du programme.

Opérateur logique | |

L'opérateur logique OU `||` vérifie deux valeurs et renvoie un booléen. Si une ou les deux valeurs sont véridiques, elle renvoie `true` . Si les deux valeurs sont fausses, elle renvoie `false` .

UNE	B	Un || B
faux	faux	faux
faux	vrai	vrai
vrai	faux	vrai
vrai	vrai	vrai

```
true || false;         // true
10 > 5 || 10 > 20;     // true
false || false;        // false
10 > 100 || 10 > 20;   // false
```

Opérateur ternaire

L'opérateur ternaire permet une syntaxe compacte dans le cas de décisions binaires (choix entre deux choix). Il accepte une condition suivie d'un `?` opérateur, puis deux expressions séparées par un `:` . Si la condition est évaluée comme véridique, la première expression est exécutée, sinon, la deuxième expression est exécutée.

```
let price = 10.5;
let day = "Monday";

day === "Monday" ? price -= 1.5 : price
+= 1.5;
```

`else`Déclaration

Un `else` bloc peut être ajouté à un `if` bloc ou à une série de `if` - `else if` blocs. Le `else` bloc ne sera exécuté que si la `if` condition échoue.

```
const isTaskCompleted = false;

if (isTaskCompleted) {
  console.log('Task completed');
} else {
  console.log('Task incomplete');
}
```

Opérateur logique&&

L'opérateur logique AND `&&` vérifie deux valeurs et renvoie un booléen. Si *les deux* valeurs sont véridiques, alors elle renvoie `true` . Si l'une des valeurs, ou les deux, est fausse, elle renvoie `false` .

```
true && true;      // true
1 > 2 && 2 > 1;    // false
true && false;     // false
4 === 4 && 3 > 1;  // true
```

`switch`Déclaration

Les `switch` instructions fournissent un moyen de vérifier une expression par rapport à plusieurs `case` clauses. Si un cas correspond, le code à l'intérieur de cette clause est exécuté.
La `case` clause doit se terminer par un mot- `break` clé. Si aucun cas ne correspond mais qu'une `default` clause est incluse, le code à l'intérieur `default` sera exécuté.
Remarque : Si `break` est omis dans le bloc d'un `case` , l' `switch` instruction continuera à vérifier les `case` valeurs jusqu'à ce qu'une rupture soit rencontrée ou que le flux soit interrompu.

```
const food = 'salad';

switch (food) {
  case 'oyster':
    console.log('The taste of the sea 🦪');
    break;
  case 'pizza':
    console.log('A delicious pie 🍕');
    break;
  default:
    console.log('Enjoy your meal');
}

// Prints: Enjoy your meal
```

`if`Déclaration

Une `if` instruction accepte une expression avec un ensemble de parenthèses :

> Si l'expression est évaluée à une valeur véridique, le code dans son corps de code s'exécute.
>
> Si l'expression donne une valeur fausse, son corps de code ne s'exécutera pas.

```
const isMailSent = true;

if (isMailSent) {
  console.log('Mail sent to recipient');
}
```

Opérateur logique !

L'opérateur logique NOT `!` peut être utilisé pour effectuer l'une des opérations suivantes :

- Inverser une valeur booléenne.
- Inverser la véracité des valeurs non booléennes.

```
let lateToWork = true;
let oppositeValue = !lateToWork;

console.log(oppositeValue);
// Prints: false
```

Opérateurs de comparaison

Les opérateurs de comparaison permettent de comparer deux valeurs et renvoient `true` ou `false` selon la validité de la comparaison :

- `===` strictement égal
- `!==` strict pas égal
- `>` plus grand que
- `>=` Meilleur que ou égal
- `<` moins que
- `<=` inférieur ou égal

```
1 > 3       // false
3 > 1       // true
250 >= 250  // true
1 === 1     // true
1 === 2     // false
1 === '1'   // false
```

`else if`Clause

`if` Après un bloc initial , les `else if` blocs peuvent chacun vérifier une condition supplémentaire. Un bloc facultatif `else` peut être ajouté après le `else if` (s) bloc(s) pour s'exécuter par défaut si aucune des conditions n'est évaluée comme véridique.

```
const size = 10;

if (size > 100) {
  console.log('Big');
} else if (size > 20) {
  console.log('Medium');
} else if (size > 4) {
  console.log('Small');
} else {
  console.log('Tiny');
}
// Print: Small
```

Vrai et faux

En JavaScript, les valeurs évaluent `true` ou `false` lorsqu'elles sont évaluées comme des booléens.

- Les valeurs évaluées à `true` sont connues sous le nom de *vérité*
- Les valeurs évaluées à `false` sont appelées *fausses*

Les valeurs fausses incluent `false` , `0` , chaînes vides `null` `undefined` , et `NaN` . Toutes les autres valeurs sont véridiques.