Cours 4 : Les conditions ?

Alors, commençons par le cœur de notre application. Son mécanisme principal est la vérification d'un mot tapé au clavier par l'utilisateur. Le mot tapé doit correspondre au mot proposé par l'application. Pour résoudre ce problème, nous allons utiliser des **structures conditionnelles**, et plus particulièrement : **les conditions**.

Découvrez les conditions

Une **condition** est une **structure conditionnelle** qui contient un **test** dont le résultat sera vrai ou faux. Elle permet d'exécuter des instructions en fonction du résultat de ce test. On parle donc de structure conditionnelle, car un code ne s'exécutera qu'**à condition** que le test soit vrai ou faux.

Par exemple, lorsque vous allez à la boulangerie et que vous vous dites :

"S'il y a du pain aux noix, j'en achèterai, sinon, je vais juste prendre une baguette."

Pour transformer cette phrase en langage de programmation, on écrira donc :

Si "présence de pain aux noix" == "vrai"

Alors "j'achète du pain au noix"

Sinon "j'achète une baguette"

La première ligne [Si "présence de pain aux noix" == vrai] est un **test** : c'est-à-dire, ce que l'ordinateur va vérifier si la condition est vraie.

Rédigez une condition en JavaScript

Pour rédiger une condition, vous devez :

- utiliser des structures conditionnelles ;
- rédiger un test ;
- rédiger un bloc de code.

Il existe deux principaux types de conditions en JavaScript :

- les **conditions if / else** permettent d'exécuter du code selon une réponse unique à un test ;
- les **conditions switch** permettent d'exécuter du code si notre test peut avoir plusieurs réponses.

Utilisez des conditions if / else pour gérer une seule réponse

Commençons par découvrir la syntaxe d'une condition en JavaScript :

```
if (condition) {

   // Code exécuté si la condition est vraie

} else {

   // Code exécuté si la condition est fausse

}
```

Ce morceau de code signifie : **Si** (if, en angais) la condition est vraie, alors j'exécute le premier bloc de code, **sinon** (else, en anglais) j'exécute le second.

La condition utilisée peut être un booléen (valant *true* ou *false*), ou une comparaison (exemple : *variable === 42*).

Rédigez un test avec des booléens

Revenons maintenant à notre projet pour rédiger notre premier test. Dans notre cas, nous cherchons à comparer le mot tapé par l'utilisateur à celui choisi par l'application.

Je crée ainsi une variable *motTapeOk* qui contiendra true ou false, et j'écris mon test en fonction :

```
let motTapeOk = true // Essayez de mettre false à la place de true


if  motTapeOk) {

   console log "Bravo, vous avez correctement tapé le mot"

} else {

   console log "Échec, le mot n'est pas correct"

}
```

Ici, *motTapeOk* est une variable de type booléen. Comme la variable vaut *true* (vrai), alors JavaScript a exécuté le premier bloc de code, car la condition est validée. Le mot tapé est correct, j'affiche donc le message correspondant.

Je vous invite à modifier la valeur de *motTapeOk*, et remplacer *true* par *false*. Vous verrez alors l'autre portion de code s'exécuter !

Si je ne veux pas afficher de message quand l'utilisateur fait une erreur, je mets un else vide ?

En fait, le else est optionnel. Vous pouvez donc simplement écrire :

```
let motTapeOk = true // Essayez de mettre false à la place de true
```

```
if  motTapeOk) {

   console log "Bravo, vous avez correctement tapé le mot"

}
```

Si l'utilisateur a correctement tapé le mot, le premier bloc sera exécuté, sinon… eh bien pas de sinon. Le code s'arrête là.

Rédigez un test avec des opérateurs de comparaison

Pour l'instant, nous avons manipulé un code simple. Nous ne comparons pas vraiment le mot de l'utilisateur avec le mot suggéré par l'application.

La première étape va donc être de **demander un mot à l'utilisateur et de mettre ce mot dans une variable**. Pour cela, le plus simple est d'utiliser une nouvelle instruction : ***prompt***.

Voilà ce que cela donne dans le code :

```
let motUtilisateur = prompt("Entrez un mot :")

console log(motUtilisateur)
```

Dans ce morceau de code :

- nous déclarons une variable *motUtilisateur* ;
- à l'intérieur nous mettons le résultat de l'instruction *prompt("Entrez un mot :").* Cette instruction fera apparaître une petite popup sur la page ;
- l'utilisateur n'a plus qu'à répondre à la question, et ce mot se retrouve à l'intérieur de la variable *motUtilisateur*.

Vérifions ensemble le résultat de cette opération avec *console.log* :

Affichage du prompt avec le texte "Entrez un mot :"

Super, nous sommes désormais capables d'interagir avec l'utilisateur !

L'étape suivante est de **réellement comparer le mot de l'utilisateur**, qui est stocké dans la variable *motUtilisateur*, avec le mot de l'application que nous allons stocker dans la variable *motApplication*.

Pour comparer ces deux mots, nous allons utiliser des opérateurs de comparaison :

<	inférieur à
<=	inférieur ou égal à
===	égal à
>=	supérieur ou égal à
>	supérieur à
!==	différent de

Il existe également les opérateurs == et != pour comparer des valeurs entre elles. Cependant, il n'est pas recommandé de les utiliser, car ils ne permettent pas de tester en une seule opération la valeur et le type de données de la valeur. Vous pourrez néanmoins être amené à en trouver dans le code d'autres développeurs.

Notez bien également la différence entre = et === :

- a = 42 signifie que l'on met la valeur 42 dans la variable a ;
- a === 42 signifie que l'on compare la valeur a et la valeur 42, pour savoir si ce sont les mêmes ou pas.

Dans notre cas, nous allons utiliser l'opérateur === . Il va nous permettre de comparer si deux éléments ont exactement la même valeur. Nous allons donc vérifier si les deux mots sont les mêmes.

```
const motApplication = "Bonjour" // Essayez de mettre un autre mot ici !

let motUtilisateur = prompt("Entrez le mot : " + motApplication)



if (motUtilisateur === motApplication) {

    console log "Bravo !"

} else {

    console log "Vous avez fait une erreur de frappe."

}
```

Utilisez la condition switch/case pour gérer plusieurs réponses

Il arrive parfois que l'on souhaite imbriquer beaucoup de *if*. Typiquement, cela arrive quand on veut faire un traitement différent pour chaque valeur d'une même variable, par exemple. Comme cela rend le code difficile à lire, nous utiliserons dans ce cas la **structure conditionnelle *switch/case***.

Prenons un exemple concret pour illustrer cela. Imaginons que nous voulions empêcher l'utilisateur d'écrire des insultes, comme par exemple "Gredin", "Mécréant" ou encore "Vilain".

Pour écrire cette instruction, nous pourrions imbriquer des *if* et des *else* pour chacun de ces mots :

```
if (motUtilisateur === motApplication) {

   console log "Bravo !"

} else {

   if (motUtilisateur === "Gredin") {

      console log "Restez correct !"

   } else {

      if (motUtilisateur === "Mécréant") {

         console log "Restez correct !"

      } else {

         if (motUtilisateur === "Vilain") {

            console log("Soyez gentil !")

         } else {

            console log("Vous avez fait une erreur de frappe.")

         }

      }

   }

}
```

Mais le résultat n'est pas facile à lire. Heureusement, on peut utiliser ***switch*** en combinaison avec ***case*** pour remédier à cela. On procède alors en deux étapes :

- définir le test avec ***switch****(laValeurATester)* ;
- lister les valeurs possibles avec ***case***.

Dans notre exemple, on écrira donc :

```
switch  motUtilisateur) {
   case motApplication:
      console log "Bravo !")
      break
   case "Gredin"
      console log "Restez correct !")
      break
   case "Mécréant"
      console log "Restez correct !")
      break
   case "Vilain"
      console log "Soyez gentil !")
      break
   default
      console log "Vous avez fait une erreur de frappe.")
}
```

Ici, je teste *motUtilisateur* :

- si l'utilisateur a tapé "Gredin", alors c'est le premier *console.log* qui va s'exécuter ;
- s'il a tapé "Mécréant", c'est le second *console.log* qui s'exécute. ;
- s'il a tapé "Vilain", c'est le troisième ;
- s'il a rentré autre chose (*default*), alors c'est le dernier *console.log* qui s'exécute.

Le *break* ("casser", en anglais) sert à arrêter le code.

Dans notre exemple, si l'utilisateur à écrit "Mécréant" et qu'il n'y a pas de break, alors le *console.log("Soyez gentil !")* qui est lié au mot "Vilain", sera exécuté aussi.

En résumé

- Une condition est un type de structure conditionnelle qui contient un test dont le résultat sera vrai ou faux.
- Les conditions ***if / else*** permettent d'exécuter du code selon une réponse unique à un test.

- Les conditions ***switch*** permettent d’exécuter du code si notre test peut avoir plusieurs réponses.
- Vous pouvez utiliser des booléens pour les tests de vos conditions, ou des opérateurs de comparaison, en fonction de ce que vous souhaitez tester.