Javascript : instructions conditionnelles et alternatives

Contenu

L'action conditionnée

L'action conditionnée est une instruction élémentaire ou une suite d'instructions **exécutées** en séquence **si l'état du système l'autorise**. Le(s) critère(s) à respecter pour exécuter l'action s'exprime(nt) à l'aide d'une condition (ou **prédicat**) évaluable au moment précis où l'action doit, le cas échéant, intervenir.

Lors de l'exécution du programme, le processeur est donc amené à évaluer la condition. La condition évaluée constitue alors un énoncé (ou **proposition**) vrai ou faux.

Schéma :

```
Si prédicat alors
   Instruction 1
   Instruction 2
   …
   Instruction N
Fin si
```

Exemples :

Si Température > 38 **alors** **Écrire** "Le patient a de la fièvre" **Fin si**	**Si** Température > 41 **et** Tension > 25 **alors** **Écrire** "Le patient va perdre patience" **Fin si**
Si non Patient **alors** **Écrire** "Éconduire l'olibrius" **Fin si**	**Si** Température > 42 **ou** (Tension < 25 **et** Pouls > 180) **alors** **Écrire** "Prévenir la famille" **Fin si**
Si Température > 40 **ou** Tension ≥ 25 **alors** **Écrire** "Hospitaliser le patient" **Fin si**	**Si** Patient **et** Pouls = 0 **alors** Patient ← **non** Patient **Fin si**

Syntaxe Javascript :

```
if (prédicat)
{
   Instruction 1
   Instruction 2
   …
   Instruction N
}
```

Les conditions (expressions if, else, switch)

A un moment ou à un autre de la programmation, on aura besoin de tester une condition. Ce qui permettra d'exécuter ou non une série d'instructions.

Condition simple

Dans sa formulation la plus simple, l'expression `if` se présente comme suit :

```
if (condition vraie)
{
   une (ou plusieurs) instructions;
}
```

Ainsi, si la condition est vérifiée, les instructions s'exécutent. Si elle ne l'est pas, les instructions ne s'exécutent pas et le programme passe à la commande suivant l'accolade de fermeture.

Exemple :

```
var reponse == "oui";

if (reponse == "oui")
{
   console.log("Bonne réponse !");
}
```

Dans le cas où il n'y a qu'une seule instruction, les accolades peuvent être omises :

```
if (reponse == "oui")
   console.log("Bonne réponse !");
```

Dans une condition, l'opérateur d'égalité se note avec le signe double égal ==

Conditions multiples : les opérateurs logiques

Les opérateurs logiques "et" (signe `&&`) et "ou" (signe `||`) permettent de tester une association de conditions :

- Utilisation du « ET » *:* `if ((condition1) && (condition2))` teste si la condition 1 ET la condition 2 sont réalisées (les deux à la fois)
- Utilisation du « OU » *:* `if ((condition1) || (condition2))` teste si au moins UNE des 2 conditions est vraie. Le fonctionnement est le suivant : si la 1ère condition est vraie, on

ne teste pas la 2ème ; si la 1ère est fausse, on teste si la deuxième est vraie. Si aucune des 2 n'est vraie, alors « faux » est renvoyé.

Exemples :

+++ TODO : EXEMPLE PEU PROBANT +++

```
// Condition avec ET
if (region == "Ile de France" && ville == "Paris")
{
   console.log("Pays : France");
}

// Condition avec OU
if (region == "Ile de France" || pays == "France")
{
   console.log("Pays : France");
}
```

Autres opérateurs logiques de comparaison

Pour rappel : voir le document *04 - Les opérateurs*, paragraphe *Les opérateurs de comparaison*.

Conditions avec `else`

Si l'on reprend l'exemple du paragraphe condition simple, on voit qu'un autre choix, pour « non », est possible. Pour traiter ce second cas, on pourrait logiquement ajouter une seconde condition :

```
var reponse == "oui";

// 1er cas
if (reponse == "oui")
{
   console.log("Bonne réponse !");
}

// 2ème cas
if (reponse == "non")
{
   console.log("Mauvaise réponse !");
}
```

Ce code n'est pas faux mais pour un choix simple à deux possibilités comme ici, c'est-à-dire booléen, il existe une instruction permettant de le simplifier : `else` que l'on traduit par « sinon » : Si la condi-

tion est vérifiée (*true*), le bloc d'instructions 1 s'exécute. Si elle ne l'est pas (*false*), le bloc d'instructions 2 s'exécute.

```
if (condition vraie)
{
   instructions 1;
}
else
{
   instructions 2;
}
```

Exemple :

```
if (reponse == "oui")
{
   console.log("Bonne réponse !");
}
else
{
   console.log("Mauvaise réponse !");
}
```

Cet exemple est le plus simple mais on peut néanmoins appliquer une condition sur le `else` :

Exemple :

```
if (reponse == "A")
{
   console.log("Bonne réponse !");
}
else if (reponse == "B")
{
   console.log("Mauvaise réponse !");
}
```

Si on peut la rencontrer dans d'autres langage (PHP notamment), l'alternative `elseif` (sans espace) n'existe pas en Javascript.

- ⇨ Tester l'exemple avec les 2 possibilités.

Il se pose cependant un problème : si aucun des deux cas n'est avéré, il ne se passera rien ou plutôt on ne rentrera dans aucune des 2 conditions.

Reprenons l'exemple et testons le avec `reponse = C` :

```
var reponse = "C";

if (reponse == "A")
{
```

```
   console.log("Bonne réponse !");
}
else if (reponse == "B")
{
   console.log("Mauvaise réponse !");
}
```

Pour éviter cela et gérer tous les cas possibles, il est recommandé d'ajouter un `else` sans condition qui permettra donc de traiter tous les autres cas (notez bien que ce `else` final reste facultatif) :

```
var reponse = "C";

if (reponse == "A")
{
   console.log("Bonne réponse !");
}
else if (reponse == "B")
{
   console.log("Mauvaise réponse !");
}
else
{
   console.log("Réponse inconnue.");
}
```

L'instruction `switch`

L'instruction `switch` permet d'écrire un ensemble des conditions sous une autre forme. Elle est introduite par le mot-clé `switch` qui reçoit en argument la variable à tester, puis, entre accolades, les mot-clés `case` reçoivent les différentes valeurs attendues pour cette variable. Pour chaque cas on exécute alors des instructions. Chaque bloc `case` doit se terminer obligatoirement par l'instruction `break` qui permet de sortir du `switch` une fois la condition réalisée (si bsentesinon toutes les cas sont exécutés !).

> **Si l'instruction `break` est absente d'un bloc, tous les cas (suivants) seront exécutés !**

```
var variable = "Valeur";

switch (variable)
{
  case "1" :
     console.log("Cas 1");
     break;

  case "2" :
     console.log("Cas 2");
     break;
```

```
  case "3":
     console.log("Cas 3");
     break;
}
```

Un exemple :

```
var modele = "Laguna";

switch (modele)
{
  case "508" :
     console.log("Modèle 508 : marque Peugeot");
     break;

  case "Laguna" :
     console.log("Modèle Laguna : marque Renault");
     break;

  case "C5" :
     console.log("Modèle C5 : marque Citroën");
     break;
}
```

L'instruction `default` peut être ajoutée de façon facultative. Elle se place à la fin du bloc `switch` après tous les `case`. Son rôle est similaire au `else` des conditions avec `if`, c'est-à-dire que si aucune condition n'est réalisée dans les `case` (on n'est pas sorti du `switch` car on n'a pas rencontré de `break`) ce sont les instructions contenues dans ce bloc `default` qui s'appliqueront :

```
var modele = "A4";
{
  case "508" :
     console.log("Modèle 508 : marque Peugeot");
     break;

  case "Laguna" :
     console.log("Modèle Laguna : marque Renault");
     break;

  case "C5" :
     console.log("Modèle C5 : marque Citroën");
     break;

  default:
     console.log("Modèle "+modele+": marque inconnue");
}
```

Notez qu'il n'y a pas d'instruction `break` pour terminer le bloc `default`.

Enfin, sachez qu'il est possible de grouper des cas auxquelles on appliquera des instructions communes. Dans l'exemple ci-après, les *Laguna* et *Clio* sont 2 modèles de la marque *Renault* :

```
var modele = "Laguna";
{
  case "508" :
     console.log("Modèle 508 : marque Peugeot");
     break;

  case "Laguna" :
  case "Clio" :
     console.log("Modèle Laguna : marque Renault");
     break;

  case "C5" :
     console.log("Modèle C5 : marque Citroën");
     break;

  default:
     console.log("Modèle "+modele+": marque inconnue");
}
```

Imbrication de conditions

Un bloc de conditions peut contenir d'autres blocs de conditions (le nombre est illimité mais il y a un moment où le code devient plus très lisible et surtout cela signifie que votre programme est mal pensé : il existe probablement d'autres alternatives):

```
if (reponse == "oui")
{
   console.log("Bonne réponse!");

   score++;

   if (score == 20)
   {
       console.log("Vous avez tout bon !");
   } // fin du 2ème if
} // fin du 1er if
```

Condition ternaire

Il existe une autre forme d'écriture des conditions. Cette forme est dite « ternaire » et revêt diverses appellations : condition ternaire, forme ternaire, écriture ternaire ou encore opérateur (ternaire) conditionnel.

```
(condition) ? instruction 1 : instruction 2
```

Si la condition entre parenthèses est vraie (évaluée à `TRUE`), l'instruction 1 est exécutée, sinon (condition entre parenthèses est vraie (évaluée à `TRUE`), sinon - condition entre parenthèses évaluée à `FALSE` - c'est l'instruction 2 qui est exécutée.

Exemple :

```
var age = 19;

(age >= 18) ? console.log("Vous êtes majeur") : console.log("Vous êtes
mineur");
```

> **Bien que l'on puisse rencontrer parfois cette forme d'écriture, son utilisation est en général déconseillée pour des raisons de lisibilité du code.**

Exercices

1 - Parité

Ecrivez un programme qui demande un nombre à l'utilisateur puis qui teste si ce nombre est pair. Le programme doit afficher le résultat « nombre pair » ou « nombre impair ». Vous devez utiliser l'opérateur modulo « % » qui donne le reste d'une division. `a%2` donne le reste de la division de `a` par 2, si ce reste est égale à zéro, `a` est divisible par 2.

2 - Age

Ecrivez un programme qui demande l'année de naissance à l'utilisateur.

En réponse votre programme doit afficher l'âge de l'utilisateur et indiquer si l'utilisateur est majeur ou mineur.

3 - Calculette

Faire la saisie de 2 nombres entiers, puis la saisie d'un opérateur '+', '-', '*' ou '/'.

Si l'utilisateur entre un opérateur erroné, le programme affichera un message d'erreur.

Dans le cas contraire, le programme effectuera l'opération demandée (en prévoyant le cas d'erreur "division par 0"), puis affichera le résultat.

4 - Remise

A partir de la saisie du prix unitaire PU d'un produit et de la quantité commandée QTECOM, afficher le prix à payer PAP, en détaillant le port PORT et la remise REM, sachant que :

- le port est gratuit si le prix des produits TOT est supérieur à 500 €. Dans le cas contraire, le port est de 2% de TOT
- la valeur minimale du port à payer est de 6 €
- la remise est de 5% si TOT est compris entre 100 et 200 € et de 10% au-delà

5 - Participation

Un patron décide de calculer le montant de sa participation au prix du repas de ses employés de la façon suivante :

- si il est célibataire : participation de 20%
- si il est marié : participation de 25%
- si il a des enfants : participation de 10% supplémentaires par enfant

La participation est plafonnée à 50%

Si le salaire mensuel est inférieur à 1200 € la participation est majorée de 10%.

Ecrire le programme qui lit les informations au clavier et affiche pour chaque salarié, la participation à laquelle il a droit.