

# Penser comme un robot

JAVASCRIPT, LES CONDITIONS

#### JAVASCRIPT, LES CONDITIONS

# COMMUNIQUER AVEC UN ROBOT

#Les opérateurs d'affectation	P.3
#Les opérateurs de comparaisons	P.4
#Les opérateurs logiques	P.5
#Les conditions ifelse et else if	P.6
#Switch, une alternative aux ifelse et else if	P.7
#Les tableaux et les boucles JavaScript	P.8
#Les objets et les boucles JavaScript	P.9

# LES OPERATEURS D'AFFECTATION

Un opérateur d'affectation permet de modifier la valeur d'un variable. L'opérateur que nous avons déjà utilisé et l'opérateur égale (=) qui permet de définir la valeur d'un variable mais il est possible de la modifier ensuite avec l'utilisation d'autres opérateurs.

NOM	<i>OPÉRATEUR</i>	SIGNIFICATION
Affectation	x = y	x = y
Affectation après addition	x += y	x = x + y
Affectation après soustraction	x -= y	x = x - y
Affectation après multiplication	x *= y	x = x * y
Affectation après division	x /= y	x = x / y
Affectation du reste	x %= y	x = x % y
Affectation après exponentiation	x **= y	x = x *
Affectation après décalage à gauche	x <<= y	x = x << y
Affectation après décalage à droite	x >>= y	x = x >> y
Affectation après décalage à droite non-signé	x >>>= y	x = x >>> y
Affectation après ET binaire	x &= y	x = x & y
Affectation après OU exclusif binaire	x ^= y	$x = x ^ y$
Affectation après OU binaire	x  = y	x = x   y

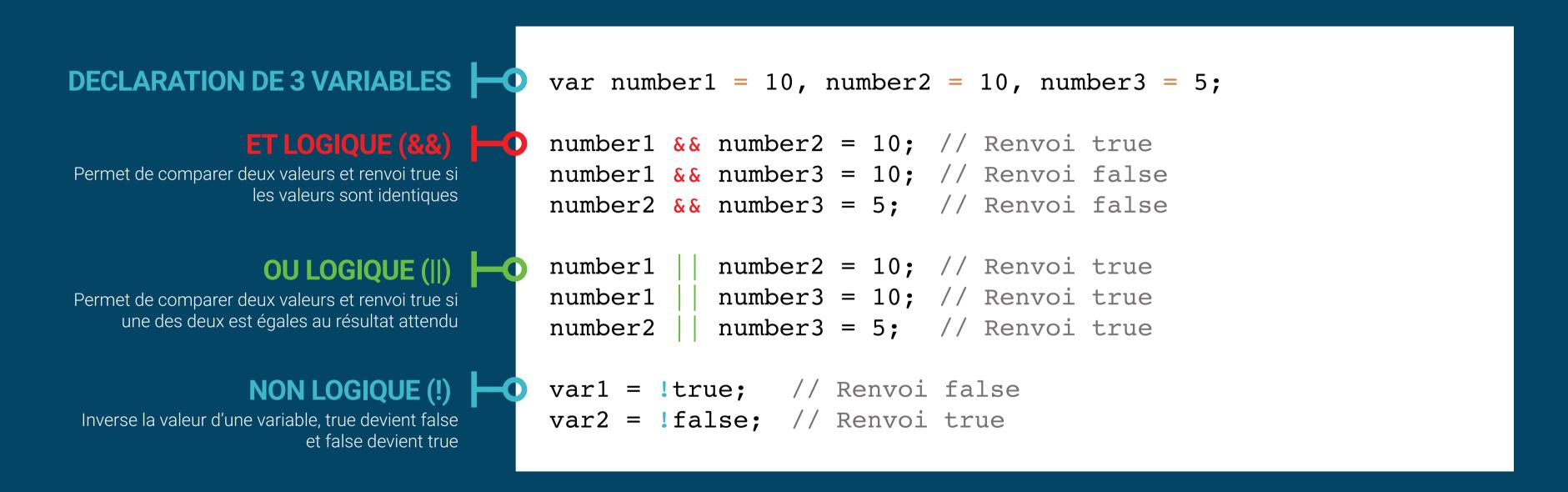
### PERATEURS DE COMPARAISONS

Les opérateurs de comparaisons permettent de vérifier la similitude ou non-similitude entre plusieurs valeurs. Ils sont utiliser pour effectuer des comparaisons au sein d'un programme pour pour réaliser des actions spécifiques.

```
var number = 15;
                             number == 15;  // true
                                 number == "15"; // true
 Converti les deux valeurs avant d'effectuer
       une comparaison d'égalité strict
                                 number != 20; // true
    (!=) INEGALITE SIMPLE —O
                                 number != "15"; // false
 Converti les deux valeurs avant d'effectuer
      une comparaison d'inégalité strict
                                 number != 15;  // false
                                 number === 15;  // true
   (===) EGALITE STRICTE
      Renvoi true si le type de variable et
                                 number === "15"; // false
              la valeur sont égales
                                 number !== 15; // false
  (!==) INEGALITE STRICTE
      Renvoi true si le type de variable et
                                 number !== "15"; // true
            les valeurs sont inégales
          (>) SUPERIEURA — number > 15; // false
           (<) INFERIEUR A
                                 number < 16;
                                                     // true
(>=) SUPERIEUR OU EGAL A —
                                 number >= 15;  // true
(<=) INFERIEUR OU EGAL A
                                 number <= 16;</pre>
                                                      // true
```

## LES OPERATEURS LOGIQUES

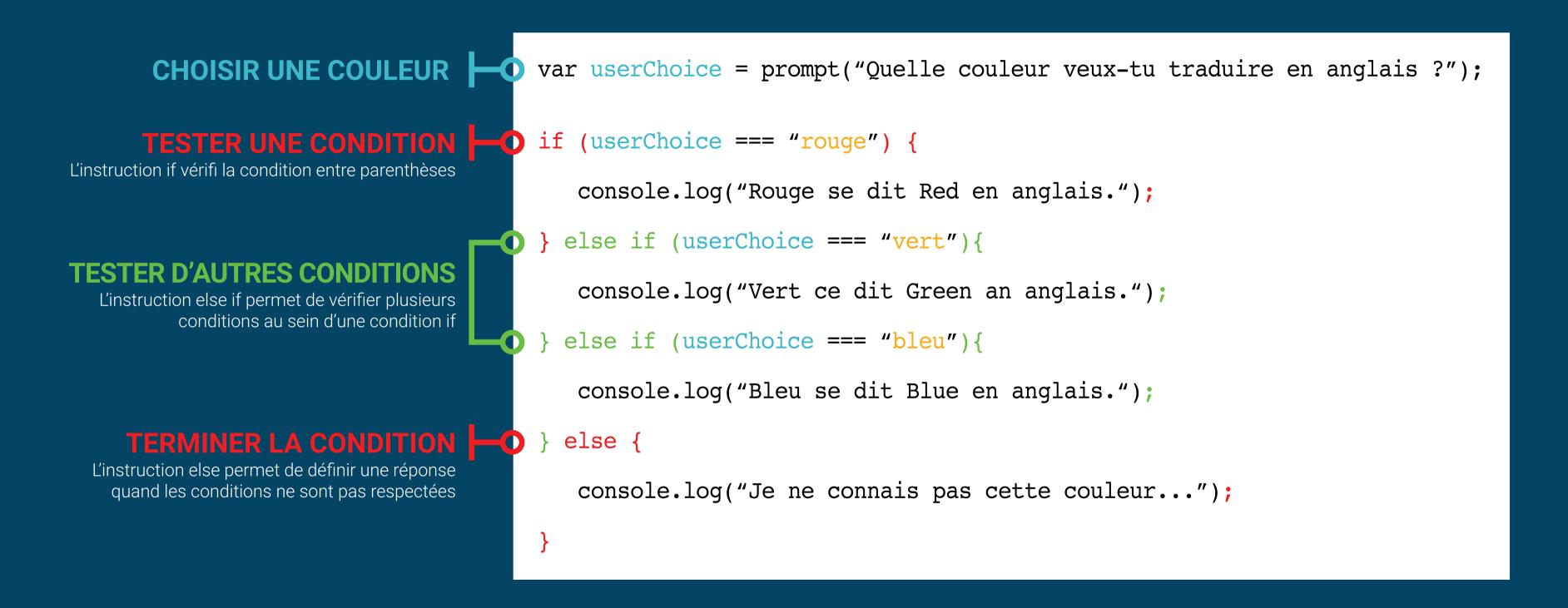
Principalement utilisés sur des valeurs bollean, les opérateurs logiques permettent de comparer deux valeurs pour effectuer des opérations quand la logique est respectée (true) ou non (false). Certains opérateurs peuvent renvoyer des valeurs non boolean si il sont utilisés sur des valirables de type number ou string.



Les opérateurs logiques permettent de comparer des variables comme les opérateurs de comparaison mais de façon logique

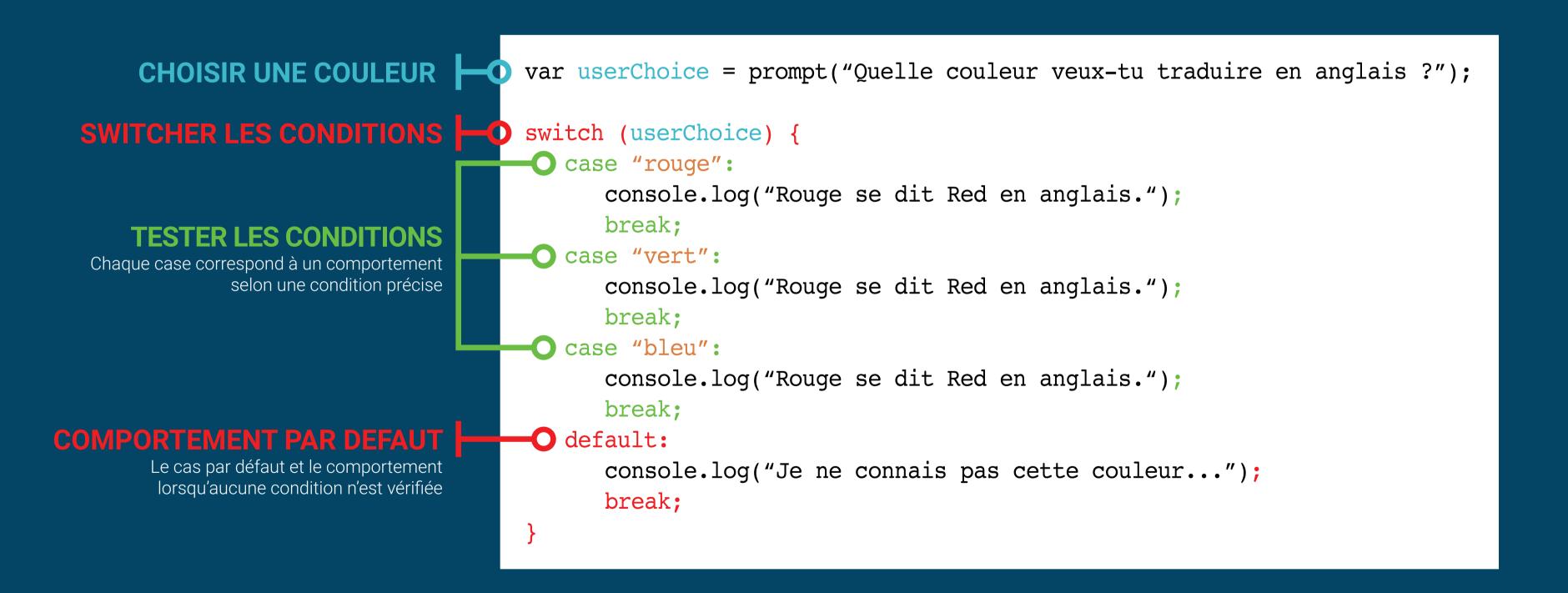
### ES CONDITIONS IF...ELSE ET ELSE IF

La condition if exécute une commande si une condition vérifiée est vrai, il est suivi d'une condition else qui exécute une commande si la condition vérifiée est fausse. Il est possible de définir plusieurs conditions avec l'opérateur else if pour décliner les commande à effectuer.



# VITCH, UNE ALTERNATIVE AUX IF...ELSE ET ELSE

La condition if exécute une commande si une condition vérifiée est vrai, il est suivi d'une condition else qui exécute une commande si la condition vérifiée est fausse. Il est possible de définir plusieurs conditions avec l'opérateur else if pour décliner les commande à effectuer.



### LES TABLEAUX ET LES BOUCLES JAVASCRIPT

Une fois que le tableau contient plusieurs entrées, il est possible d'effectuer des actions sur toutes les entrées du tableau. Le principe en programmation est de ne jamais répéter une ligne de code, c'est pour cette raison qu'il faut créer des boucles sur les tableaux plutôt que de répéter le même code pour chaque entrées du tableau. Il existe deux types de boucle : while et for qui doivent respecter une orthographe stricte.

#### **BOUCLE WHILE (...)**

```
var users = ["Frédérique", "Pascal, "Matthieu"];

// Initialisation de la boucle
var i = 0;

// Lancement de la boucle while
while (i < users.length) {
   console.log(users[i]);
   i++;
}</pre>
```

Pour lancer une boucle while (...) il faut initier une variable i valant zéro, la boucle se lit ensuite de la façon suivante : à chaque fois que i est inférieure au nombre d'entreés du tableau users, exécute une action et incrémente i de un.

La boucle while (...) s'arrête quand i n'est plus inférieure au nombre d'entrées du tableau, la boucle ci-dessus affichera dans la console chacun des prénoms et se stoppera.

#### **BOUCLE FOR (...)**

```
var users = ["Frédérique", "Pascal, "Matthieu"];

// Lancement de la boucle for
for (var i = 0; i < users.length; i++) {
  console.log(users[i]);
}</pre>
```

Dans une boucle for (...), la variable i est initiée et incrémentée directement dans la boucle, l'action à faire est ensuite placée entre accolades. La boucle for (...) se stop quand i n'est plus inférieur au nombre dentrèes du tableau.

La variable i est primordiale dans l'utilisation des deux boucles : il faut absolument incrémenter i à la fin de la boucle pour ne pas exécuter des boucles infinies. Car si i vaut toujours zéro, alors il ne sera jamais égale au nombre d'entrées du tableau.

Support de formation Julien Noyer 09/09/2015

### ES OBJETS ET LES BOUCLES JAVASCRIPT

Tout comme un tableau, il est possible d'effectuer une boucle sur un objet pour effectuer une opération similaire sur chaque propriétés de l'objet. Javascript à prévu une boubcle particulière pour les objet qui s'appelle for...in.

#### firstName: "Julien", lastName: "Noyer", age: 36 **}**; BOUCLE FOR...IN —O for (var prop in julien) { console.log("o." + prop + " = " + julien[prop]); Le mot-clé prop permet de récupérer la propriété de l'objet et [prop] permet de récupérer la valeur de la propriété // Affiche dans la console : // o.firstName = Julien // o.lastName = Noyer // o.age = 36