PENSER COMME UN ROBOT

JAVASCRIPT, LES CONDITIONS

COMMUNIQUER AVEC UN ROBOT

LES OPERATEURS D'AFFECTATION

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

NOM	OPÉRATEUR	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 ** y
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

LES OPERATEURS DE COMPARAISONS

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

(==) EGALITE SIMPLE
Converti les deux valeurs avant d'effectuer une comparaison d'égalité strict

(!=) INEGALITE SIMPLE
Converti les deux valeurs avant d'effectuer une comparaison d'inégalité strict

(===) EGALITE STRICTE
Renvoi true si le type de variable et la valeur sont égales

(!==) INEGALITE STRICTE
Renvoi true si le type de variable et les valeurs sont inégales

(>) SUPERIEUR A
(<) INFERIEUR A

(>=) SUPERIEUR OU EGAL A
(<=) INFERIEUR OU EGAL A

```
var number = 15;

number == 15;          // true
number == "15";        // true

number != 20;          // true
number != "15";        // false
number != 15;          // false

number === 15;         // true
number === "15";       // false

number !== 15;         // false
number !== "15";       // true

number > 15;           // false
number < 16;           // true

number >= 15;          // true
number <= 16;          // 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 s'ils sont utilisés sur des variables de type number ou string.

DECLARATION DE 3 VARIABLES

```
var number1 = 10, number2 = 10, number3 = 5;
```

ET LOGIQUE (&&)

Permet de comparer deux valeurs et renvoi true si les valeurs sont identiques

```
number1 == 10 && number2 == 10;    // Renvoi true
number1 == 10 && number3 == 10;    // Renvoi false
number2 == 5 && number3 == 5;      // Renvoi false
```

OU LOGIQUE (||)

Permet de comparer deux valeurs et renvoi true si une des deux est égales au résultat attendu

```
number1 == 10 || number2 == 10;    // Renvoi true
number1 == 10 || number3 == 10;    // Renvoi true
number2 == 5 || number3 == 5;      // Renvoi true
```

NON LOGIQUE (!)

Inverse la valeur d'une variable, true devient false et false devient true

```
var1 = !true;                      // Renvoi false
var2 = !false;                     // Renvoi true
```

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

LES CONDITIONS IF...ELSE ET ELSE IF

La condition if exécute une commande si une condition vérifiée est vraie, 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.

CHOISIR UNE COULEUR

TESTER UNE CONDITION
L'instruction if vérifi la condition entre parenthèses

TESTER D'AUTRES CONDITIONS
L'instruction else if permet de vérifier plusieurs conditions au sein d'une condition if

TERMINER LA CONDITION
L'instruction else permet de définir une réponse quand les conditions ne sont pas respectées

```
var userChoice = prompt("Quelle couleur veux-tu traduire en anglais ?");

if (userChoice === "rouge") {
    console.log("Rouge se dit Red en anglais.");
} else if (userChoice === "vert"){
    console.log("Vert ce dit Green an anglais.");
} else if (userChoice === "bleu"){
    console.log("Bleu se dit Blue en anglais.");
} else {
    console.log("Je ne connais pas cette couleur...");
}
```

SWITCH, UNE ALTERNATIVE AUX IF...ELSE ET ELSE IF

La condition if exécute une commande si une condition vérifiée est vraie, 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.

CHOISIR UNE COULEUR

SWITCHER LES CONDITIONS

TESTER LES CONDITIONS

Chaque case correspond à un comportement selon une condition précise

COMPORTEMENT PAR DEFAUT

Le cas par défaut et le comportement lorsqu'aucune condition n'est vérifiée

```
var userChoice = prompt("Quelle couleur veux-tu traduire en anglais ?");

switch (userChoice) {
    case "rouge":
        console.log("Rouge se dit Red en anglais.");
        break;
    case "vert":
        console.log("Rouge se dit Red en anglais.");
        break;
    case "bleu":
        console.log("Rouge se dit Red en anglais.");
        break;
    default:
        console.log("Je ne connais pas cette couleur...");
        break;
}
```

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 entrée 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++;
}
```

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'entrée du tableau users, exécute une action et incrémente i de un.

La boucle while (...) s'arrête quand i n'est plus inférieur 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]);
}
```

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 (...) ce stop quand i n'est plus inférieur au nombre dentrée 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.

LES 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 propriété de l'objet. Javascript a prévu une boucle particulière pour les objets qui s'appelle for...in.

CREATION D'UN OBJET

```
var julien = {
    firstName: "Julien",
    lastName: "Noyer",
    age: 36
};
```

BOUCLE FOR...IN

```
for (var prop in julien) {
  console.log("o." + prop + " = " + julien[prop]);
}

// Affiche dans la console :
// o.firstName = Julien
// o.lastName = Noyer
// o.age = 36
```

Le mot-clé prop permet de récupérer la propriété de l'objet et [prop] permet de récupérer la valeur de la propriété