JAVASCRIPT

POUR LES NULS

Inspiré par Javascripting

INTRODUCTION

Pour ce TP, nous allons avoir besoin d'un outil pour exécuter notre code Javascript.

Nous utiliserons JSbin.

Dans l'onglet Javascript, nous écrivons le code à exécuter.

L'onglet Console contient l'affichage du programme, ou les éventuelles erreurs.

EXERCICE

- Ouvrez JSbin
- Dans l'onglet Javascript, entrez le code suivant :

```
console.log('hello');
```

- Dans l'onglet Console, cliquez sur run.

La console doit afficher `"hello"`

VARIABLES

Une variable est un nom qui fait référence à une valeur spécifique. Les variables sont déclarées en utilisant le mot clé var suivi du nom de la variable.

```
var example;
```

La variable ci-dessus est déclarée, mais elle n'est pas définie (elle ne référence aucune valeur pour le moment).

Voici un exemple de définition de variable, lui faisant contenir une valeur spécifique :

```
var example = 'some string';
```

NOTE

Une variable est déclarée en utilisant var et utilise le signe égal pour assigner la valeur qu'elle référence. Nous utilisons communément l'expression "Assigner une valeur à une variable".

EXERCICE

- Déclarez une variable nommée `example`
- Assignez la valeur `'some string'` à la variable `example`.
- Utilisez ensuite `console.log()` pour afficher la variable `example` dans la console.

La console doit afficher `'some string'`

CHAÎNES DE CARACTÈRES

Une chaine de caractères peut être n'importe quelle valeur entourée par des guillemets.

Il peut s'agir de guillemets simples ou doubles :

```
'this is a string'
"this is also a string"
```

Vous allez assez souvent avoir besoin de savoir combien de caractères sont contenus dans une chaîne de caractères.

Pour cela vous allez utiliser la propriété `.length`.
Voici un exemple :

```
var example = 'example string';
example.length
```

EXERCICE

- Créez une variable nommée `example`.
- Assignez la chaîne de caractères `'example string'` à la variable `example`.
- Utilisez `console.log` pour afficher la longueur de la chaîne de caractères dans la console.

La console doit afficher `14`

Vous allez souvent avoir besoin de changer le contenu d'une chaîne de caractères.

Les chaînes de caractères ont des fonctionnalités directement intégrées qui vous permettent de manipuler leur contenu.

Voici un exemple qui utilise la méthode `.replace()` :

```
var example = 'this example exists';
example = example.replace('exists', 'is awesome');
console.log(example);
```

Notez que pour modifier la valeur contenue dans la variable exemple, nous devons utiliser encore une fois le signe égal, mais cette fois avec la méthode example.replace() à la droite du égal.

EXERCICE

- Définissez une variable nommée `pizza` qui contient cette chaîne de caractères : `'pizza is alright'`
- Utilisez la méthode `.replace()` pour modifier `alright` en `wonderful`.
- Utilisez `console.log()` pour afficher le résultat de la méthode `.replace()` dans la console.

La console doit afficher `"pizza is wonderful"`

NOMBRES

Les nombres peuvent être des entiers, comme `2`, `14`, ou `4353`, ou décimaux (aussi appelés flottants), comme `3.14`, `1.5`, ou `100.7893423`. Contrairement aux chaînes de caractères, les nombres ne sont pas entourés par des guillemets.

EXERCICE

- Définissez une variable nommée `example` qui contient l'entier `123456789`.
- Utilisez `console.log()` pour afficher ce nombre dans la console.

La console doit afficher `123456789`

Nous pouvons faire des mathématiques basiques avec les operateurs tels que `+`, `-`, `*`, `/`, et `%`.

```
var example = (1 + 3) / 2;
```

Pour des maths plus complexes, nous pouvons utiliser l'objet `Math`.

EXERCICE

- Définissez une variable nommée `roundUp` qui contient le flottant `1.5` multiplié par `3`.

Nous allons utiliser la méthode Math.round() pour arrondir ce nombre. Cette méthode retourne l'arrondi entier le plus proche. Exemple : `Math.round(0.5);`

- Définissez une seconde variable nommée `rounded` qui contient le résultat de la méthode `Math.round()`, en lui passant la variable `roundUp` en argument.
- Utilisez `console.log()` pour afficher ce nombre dans la console.

La console doit afficher `5`

CONDITIONS

Les instructions conditionnelles servent à changer le flux d'exécution d'un programme selon des conditions booléennes spécifiques.

Une instruction conditionnelle ressemble à ça :

```
if (n < 1) {
  console.log('the variable n is less than 1.');
} else {
  console.log('the variable n is greater than or equal to 1.');
}
```

Dans les parenthèses vous devez entrer une expression logique, ce qui veut dire que le résultat de l'instruction est soit vrai (`true`) soit faux (`false`).

Le bloc `else` est optionnel et contient le code qui sera exécuté si la condition est fausse.

Opérateurs logiques :

- `<` : inférieur
- `>` : supérieur
- `<=` : inférieur ou égal
- `>=` : supérieur ou égal
- `===` : strictement égal
- `!==` : différent

EXERCICE

- Déclarez une variable `fruit`.
- Assignez à la variable `fruit` la chaîne de caractères `orange`.
- Utilisez ensuite `console.log()` pour afficher « The fruit name has more than five characters. » si la longueur du contenu de la variable `fruit` est supérieure à cinq. Sinon, affichez « The fruit name has five characters or less. »

La console doit afficher `"The fruit name has more than five characters."`

BOUCLE FOR

Les boucles for vous permettent de répéter l'exécution d'un bloc de code un certain nombre de fois.

Cette boucle for affiche dans la console dix fois :

```
for (var i = 0; i < 10; i++) {
        // affiche les nombres de 0 a 9
        console.log(i)
}
```

La première partie, var i = 0, n'est exécutée qu'une fois au début de la boucle. La variable i est utilisée pour compter le nombre d'exécutions de la boucle.

La seconde partie, i < 10, est vérifiée au début de chaque itération de la boucle avant que le code contenu ne s'exécute. Si la condition est valide, le code contenu dans la boucle est exécuté. Sinon, la boucle est terminée. La condition i < 10; indique que la boucle va continuer de s'exécuter tant que i est inférieur à 10.

La partie finale, i++, est exécutée à la fin de chaque boucle. Elle incrémente la variable i après chaque itération. Dès que i atteint 10, la boucle est terminée.

EXERCICE

- Définissez une variable nommée `total` et assignez lui la valeur `0`.
- Créez une seconde variable nommée `limit` et assignez lui la valeur `10`.
- Créez une boucle `for` avec une variable `i` commençant à `0` et s'incrémentant à chaque itération de la boucle. La boucle doit s'exécuter aussi longtemps que `i` est strictement inférieur à `limit`.
- À chaque itération de la boucle, ajoutez le nombre i à la variable `total`. Pour faire cela, vous pouvez utiliser l'instruction suivante : `total += i;`
- Après la boucle, utilisez `console.log()` pour afficher la variable `total` dans le terminal.

La console doit afficher `45`

TABLEAUX

Un tableau est une liste de valeurs.

Voici un exemple :

```
var pets = ['cat', 'dog', 'rat'];
```

EXERCICE

- Définissez une variable nommée `pizzaToppings` qui contient un tableau composé de trois chaînes de caractères dans cet ordre : `tomato sauce`, `cheese`, `pepperoni`.
- Utilisez `console.log()` pour afficher le tableau `pizzaToppings` dans la console.

La console doit afficher `["tomato sauce", "cheese", "pepperoni"]`

On peut accéder aux cases du tableau via leurs index.

Les index doivent être des nombres allant de zero à la longueur du tableaux moins un.

Voici un exemple :

```
var pets = ['cat', 'dog', 'rat'];
console.log(pets[0]);
```

Le code ci-dessus affichera le premier élément du tableau `pets` : la chaine de caractères `cat`.

On ne doit accéder aux éléments de tableaux qu'au travers de la notation « crochets » : la notation en point est invalide.

Notation valide :

```
console.log(pets[0]);
```

Notation invalide :

```
console.log(pets.1);
```

EXERCICE

- Définissez un tableau `food` :

```
var food = ['apple', 'pizza', 'pear'];
```

- Utilisez `console.log()` pour afficher la deuxième valeur du tableau dans la console.

La console doit afficher `"pizza"`

OBJETS

Les objets sont des listes de valeurs similaires aux tableaux, sauf que les valeurs sont identifiées par une clé au lieu d'un entier.

Voici un exemple :

```
var foodPreferences = {
  pizza: 'yum',
  salad: 'gross'
};
```

EXERCICE

- Définissez une variable nommée `pizza` comme celà :

```
var pizza = {
  toppings: ['cheese', 'sauce', 'pepperoni'],
  crust: 'deep dish',
  serves: 2
};
```

- Utilisez `console.log()` pour afficher l'objet `pizza` dans la console.

Vous pouvez manipuler les propriétés d'objets — les clés et valeurs qu'un objet contient — en utilisant des méthodes très similaires aux tableaux.

Voici un exemple utilisant des crochets :

```
var example = {
  pizza: 'yummy'
};

console.log(example['pizza']);
```

Le code ci-dessus va afficher la chaine de caractères `yummy` dans la console.

Une alternative consiste à utiliser la notation en point pour avoir le même résultat :

```
example.pizza;
example['pizza'];
```

Les deux lignes de code ci-dessus renverront `yummy`.

EXERCICE - 1

- Définissez une variable nommée `food` comme ceci :

```
var food = {
  types: 'only pizza'
};
```

- Utilisez `console.log()` pour afficher la propriété `types` de l'objet `food` dans la console.

La console doit afficher `"only pizza"`

EXERCICE - 2

- Assignez la chaîne de caractères `pizza and beer` à la propriété `types` de l'objet `food`.
- Utilisez `console.log()` pour afficher la propriété `types` de l'objet `food` dans la console.

La console doit afficher `"pizza and beer"`

FONCTIONS

Une fonction est un bloc de code qui prend des entrées, traite ces entrées, et produit une sortie.

Voici un exemple :

```
function example(x) {
  return x * 2;
}
```

Nous pouvons appeler cette fonction comme cela pour récupérer le nombre 10 :

```
example(5);
```

EXERCICE

- Définissez une fonction nommée `eat` qui prend un argument nommé `food` qui est considéré comme étant une chaîne de caractères.
- Dans cette fonction, retournez l'argument `food` comme cela :

```
return food + ' tasted really good.';
```

- Dans les parenthèses de `console.log()`, appelez la fonction `eat()` avec la chaîne de caractères `bananas` comme argument.

La console doit afficher `"bananas tasted really good."`

On peut déclarer qu'une fonction reçoit n'importe quel nombre d'arguments. Les arguments peuvent être de n'importe quel type : une chaîne de caractères, un nombre, un tableau, un objet et même une autre fonction.

Voici un exemple :

```
function example(firstArg, secondArg) {
  console.log(firstArg, secondArg);
}
```

Nous pouvons appeler cette fonction avec deux arguments comme cela :

```
example('hello', 'world');
```

L'exemple ci-dessus va afficher `hello world` dans la console.

EXERCICE

- Définissez une fonction nommée `math` qui prend trois arguments. Il est important que vous compreniez que les noms d'arguments ne sont seulement utilisés que pour y faire référence.
- Nommez chaque argument comme vous le souhaitez.
- Dans la fonction `math`, renvoyez la valeur obtenue de la multiplication du second argument avec le troisième et en ajoutant le premier argument au résultat.
- Après cela, dans les parenthèses de `console.log()`, appelez la fonction `math()` avec le nombre `53` comme premier argument, le nombre `61` comme second et le nombre `67` en troisième argument.

La console doit afficher `4140`

C'EST TOUT

POUR LE MOMENT

Merci d'avoir suivi !