JavaScript QCM 4

Bonjour, Adrien JOLY

Votre copie a bien été rendue. Merci !

Date/heure de rendu: Mon Dec 12 2016 15:15:01 GMT+0100 (CET)

Sign out

QCM

Question 1

```
function maFonction(param) {
  return param + 2;
}
```

Ceci est:

- ○ un appel de fonction
- ◉ une définition de fonction
- ○ une affectation de fonction
- ○ une fonction qui ne fonctionne pas

C'est une définition de fonction.

On la reconnait à l'usage du mot clé `function` et des accolades entourant le code qui sera exécuté lorsque cette fonction sera appelée.

Question 2

```
maFonction(4);
```

Ceci est:

- [x] un appel de fonction
- [ ] une définition de fonction
- [ ] une affectation de fonction
- [ ] une fonction qui ne fonctionne pas

C'est un appel de fonction.

Un appel de fonction = le nom de la fonction, suivi par les paramètres entre parenthèses. Sans le mot clé `function`.

Cette instruction va exécuter le code défini dans la fonction, et affecter les valeurs fournies à chaque paramètre.

Question 3

```
// cette fonction concatène un zéro à la fin de la valeur passée en paramè
function maFonction(param) {
  return param + '0';
}
```

Comment savoir si cette fonction fonctionne bien ? (c.a.d. sans bug)

- [ ] il suffit de la copier-coller dans la console
- [ ] il faut taper maFonction dans la console
- [x] vérifier que le test passe: maFonction(1) === '10';
- [ ] vérifier que maFonction(1) renvoie bien true

Pour vérifier le bon fonctionnement il faut définir et exécuter des tests unitaires.

Ceux-ci permettent de comparer le résultat attendu d'une fonction, à celui effectivement retourné par l'implémentation actuelle de cette fonction.

`maFonction(1) === '10';` est un bon test unitaire car son exécution retourne `true` si la fonction retourne le résultat attendu (`10`) lorsqu'on lui passe `1` en

paramètre.

Question 4

Supposons que nous avons défini une fonction `doubler()` qui retourne le double du nombre passé en paramètre, lors de son appel.

Que se passe-t-il si on exécute l'instruction suivante:

```
var maVariable = doubler(3);
```

- (●) le résultat va être affecté à maVariable
- ( ) le résultat va s'afficher dans la console
- ( ) maVariable contient la définition de la fonction
- ( ) maVariable contient l'appel de la fonction

Il s'agit ici d'un appel de fonction. De la même façon que pour une opération élémentaire (ex: `2 + 2`), tout appel de fonction sera remplacé par la valeur retourné par l'exécution de cette fonction.

Ici, le résultat de l'exécution de la fonction `doubler` avec le paramètre `3`, soit la valeur `6`, va être affectée à `maVariable`.

Exercices de codage

Question 1

Définir une fonction `soustraire` qui retourne le résultat de la soustraction `a - b`, `a` et `b` étant des paramètres de cette fonction.

Respecter les conventions et règles d'indentation vues en cours.

Saisissez votre code Javascript ici

```
function soustraire(a, b) {
  return a - b;
}
```

Solution:

```
function soustraire(a, b) {
  return a - b;
}
```

Question 2

Définir une fonction `repeter` qui affiche `n` fois `'Bonjour!'` dans la console, puis qui retourne `n`, `n` étant un paramètre de cette fonction.

Respecter les conventions et règles d'indentation vues en cours.

Saisissez votre code Javascript ici

```
function repeter(n) {
  for (var i = 0; i < n; i++) {
    console.log('Bonjour!');
  }
  return n;
}
```

Solution:

```
function repeter(n) {
  for (var i = 0; i < n; i++) {
    console.log('Bonjour!');
  }
  return n;
}
```