Student 1

Exemples de correction :

Test ①

Ok.

```
function doubleInt(int) {
    return int * 2;
}
```

Test ②

La fonction et le raisonnement sont justes.
Pour aller plus loin, tu peux optimiser ta fonction davantage.

```
function isEvent( int )  {
    return int % 2 === 0;
}
```

Test ③

Ok.

```
function getExtension ( str )  {
    var parts = str.split('.');
    if (parts.length === 1) {
        return false;
    }

    var lastPart = parts[parts.length - 1];
    return lastPart;
}
```

Test ④

Ok.

```
function getLongestString(arr) {
    var strMax = '';

    for (var index in arr) {
        var str = arr[index];

        if (typeof str !== 'string') {
            continue;
        }

        if (str.length > strMax.length) {
            strMax = str;
        }
    }

    if (arr.length === 0) {
        return false;
    }
    return strMax;    }
```

Test ⑤

En écartant les expressions qui n'étaient pas des nombres, tu as également écarté les tableaux, alors qu'ils contenaient eux aussi des nombres. Ils manquent donc au calcul.
Attention au type des données que tu manipules.

```
function sum(arr) {
    var result = 0;

    for (var index in arr) {
        var item = arr[index];

        if (typeof item === 'number') {
            result += item;
        } else if (typeof item === 'object') {
            result += sum(item);    }
    }
return result;          }
```

Retour global :
Tu sembles avoir bien compris comment manipuler les expressions en JavaScript. Continue comme ça!

Student 2

Exemples de correction :

Test ①

Ok.

```
function doubleInt(int) {
        return int * 2;
}
```

Test ②

Ok.

```
function isEvent( int )   {
        return int % 2 === 0;
}
```

Test ③

Ta fonction est bien optimisée, très bien!

```
function getExtension ( str )  {
        var parts = str.split('.');
        if (parts.length === 1) {
                return false;
        }

        var lastPart = parts[parts.length - 1];
        return lastPart;
}
```

Test ④

Encore une manière audacieuse de résoudre l'exercice.

```
function getLongestString(arr) {
        var strMax = '';

        for (var index in arr) {
                var str = arr[index];

                if (typeof str !== 'string') {
                        continue;
                }

                if (str.length > strMax.length) {
                        strMax = str;
                }
        }

        if (arr.length === 0) {
                return false;
        }
        return strMax;       }
```

Test ⑤

Faire une première fonction pour régulariser les tableaux est une bonne idée.

```
function sum(arr) {
        var result = 0;

        for (var index in arr) {
            var item = arr[index];

            if (typeof item === 'number') {
                    result += item;
            } else if (typeof item === 'object') {
                    result += sum(item);     }
        }
return result;              }
```

Retour global :
Tu as bien compris les exercices. Tes fonctions sont optimisées, bien pensées. Bravo!

Student 3

Exemples de correction :

Test ①

Ok.

```
function doubleInt(int) {
        return int * 2;
}
```

Test ②

Ok. Tu aurais pu utiliser le comparateur « ===» car le résultat doit être égal en valeur et en type, à 0.

```
function isEvent( int )   {
        return int % 2 === 0;
}
```

Test ③

Ok. La fonction serait plus claire avec des variables nommées plus précisément. Notamment pour «j».

```
function getExtension ( str )  {
        var parts = str.split('.');
        if (parts.length === 1) {
                return false;
        }

        var lastPart = parts[parts.length - 1];
        return lastPart;
}
```

Test ④

Ok, la fonction marche.
Attention au nom de tes variables. «plus» et «nom» ne sont pas assez informatifs pour comprendre tout de suite leur utilité dans la fonction.

```
function getLongestString(arr) {
        var strMax = '';

        for (var index in arr) {
                var str = arr[index];

                if (typeof str !== 'string') {
                        continue;
                }

                if (str.length > strMax.length) {
                        strMax = str;
                }
        }

        if (arr.length === 0) {
                return false;
        }
        return strMax;       }
```

Test ⑤

La fonction marche. Elle pourrait être un peu plus optimisée, notamment en évitant d'imbriquer trois boucles for. Tu peux par exemple chercher du côté de «typeof» pour trier les éléments par type, comme dans l'exemple proposé.

```
function sum(arr) {
        var result = 0;

        for (var index in arr) {
            var item = arr[index];

            if (typeof item === 'number') {
                    result += item;
            } else if (typeof item === 'object') {
                    result += sum(item);     }
        }
return result;              }
```

Retour global :
Les exercices sont bien compris et les fonctions sont fonctionnelles. Attention en revanche, elles gagneraient parfois à être simplifiées en utilisant des méthodes natives JavaScript qui évitent d'enchainer les boucles for (ex.: sum(), split()…).
Attention aussi au nom des variables qui ne sont pas assez informatifs et rendent difficile la lisibilité du code. Tu es sur la bonne voie.

Student 4

Test ①

☑ Ok. Pense à nettoyer tes «console.log()» avant de présenter ton code, il sera plus propre. De plus, si tu veux faire un console.log de ta fonction doubleInt, n'oublie pas qu'elle prend un integer en paramètre.

Exemples de correction :

```
function doubleInt(int) {
        return int * 2;
}
```

Test ②

☑ Ok. La solution est propre. Tu aurais pu aussi utiliser le comparateur « ===» car le résultat doit être égal en valeur, mais aussi en type, à 0 (égalité stricte).

```
function isEvent( int )   {
        return int % 2 === 0;
}
```

Test ③

☐ Un problème de syntaxe rend impossible l'exécution de la fonction. Utilise la documentation du langage ou l'inspecteur du navigateur pour corriger les problèmes de syntaxe.
La fonction n'est pas terminée. Essaye d'aller plus loin après avoir corrigé la syntaxe (lastIndexOf() accepte en paramètre un tableau : ['a','b','c']).
Tu peux t'inspirer de cette solution possible =>

```
function getExtension ( str )  {
        var parts = str.split('.');
        if (parts.length === 1) {
                return false;
        }

        var lastPart = parts[parts.length - 1];
        return lastPart;
}
```

Test ④

☐ Essaie de reprendre l'exercice en t'aidant de ces explications et l'exemple :
Tu peux tout d'abord récupérer les longueurs des chaines de caractères que tu dois examiner. Par exemple avec la méthode «.length».
Pour analyser l'intérieur du tableau, il faudra boucler sur chaque index (boucle «for» ou «foreach» par exemple).
Restera ensuite à comparer les résultats obtenus, et à les stocker dans une variable pour la restituer avec 'return'.
Pour savoir où tu en es à tout moment, n'hésite pas à faire des consoles.log() à chaque étape.

```
function getLongestString(arr) {
        var strMax = '';

        for (var index in arr) {
                var str = arr[index];

                if (typeof str !== 'string') {
                        continue;
                }

                if (str.length > strMax.length) {
                        strMax = str;
                }
        }

        if (arr.length === 0) {
                return false;
        }
        return strMax;      }
```

Test ⑤

☐ Essaie de reprendre l'exercice en t'appuyant sur l'exemple. Il faudra pour cela : trier les éléments suivant leur type. Les difficultés sont que les chaines de caractère n'entrent pas dans le calcul, et il faut aller chercher les caractères à l'intérieur des tableaux.
Ensuite, tu pourras utiliser le comparateur « += » pour ajouter une valeur à la valeur précédente, ainsi que la méthode sum().
Courage! N'hésite pas à nous solliciter si tu n'y parviens pas.

```
function sum(arr) {
        var result = 0;

        for (var index in arr) {
            var item = arr[index];

            if (typeof item === 'number') {
                    result += item;
            } else if (typeof item === 'object') {
                    result += sum(item);    }
        }
return result;              }
```

Retour global :
Tu sembles avoir perdu la motivation au cours des exercices. Courage! Essaie de reprendre les exercices grâce aux solutions que nous proposons. N'hésite pas à repartir du début avec le JavaScript et à réviser les types de variables (integer, string, array...), les opérateurs (>,<, ==...) et quelques méthodes natives revenant souvent, comme sum(), reduce(), filter(), splice()... N'hésite pas à poser des questions à tes professeurs quand tu es coincé ou ne sais pas comment commencer.

Retour pour l'équipe pédagogique

Student ①

Student 1 a réussi les 4 premiers exercices. À l'aise avec la manipulation des expressions en JavaScript. Devrait travailler un peu la précision (ex. 5, il a négligé un type d'éléments, ce qui a faussé son calcul) et l'optimisation de ses fonctions.

Student ②

Tous les exercices sont réussis. Manipule le JavaScript avec aisance et fait des choix plutôt audacieux. Il est au top.

Student ③

Student 3 a réussi tous les exercices. A compris le principe mais a tendance à compliquer un peu les choses. Il semble avoir besoin d'un peu plus de clarté dans sa réflexion comme son code. Réfléchir à des noms de variables plus pertinents pourrait l'aider à mieux s'y retrouver.

Student ④

Après avoir réussi les deux premiers tests, a lâché l'affaire pendant le test 3. Une erreur de syntaxe empêchait sa fonction de s'exécuter et il n'a pas su dépasser ce blocage. Deux derniers exercices vides. Semble avoir besoin d'être remotivé ou d'être plus encadré.

En général

Les exercices sont adaptés au niveau de la promo. Tout le monde a réussi au moins 2 tests, la plupart a fait tous les exercices. Pour ceux qui ont encore du mal avec certaines notions, j'ai rédigé un récap' sur les points à maîtriser, qui résume mes observations à chacun.

Point général pour la promo

Quelques points à garder en tête:
- Faites bien attention aux types de données que vous devez manipuler (string, boolean, integer...), ils n'ont pas les mêmes caractéristiques.
- Soyez précis sur les comparateurs que vous utilisez. Strictement inférieur et inférieur ou égal ne rendent pas la même chose. De même que « == » (égal) n'est pas pareil que « === » (strictement égal, càd égal en valeur et en type de donnée). À ne pas confondre avec « = » qui est utilisé pour assigner une valeur à une variable.
- Faites attention aux noms que vous donnez à vos fonctions et à vos variables. Il faut que d'un coup d'œil, on comprenne à quoi elles servent.

Pour aller plus loin, vous pouvez explorer les méthodes natives de manipulation de données et de tableaux en JavaScript comme :
sort(), shift(), toString(), splice(), indexOf()...
Ce sont de précieux outils pour optimiser le traitement des données.

Aleth Arrias Camps

Notes sur mes choix pédagogiques

Pour cette première, j'ai pris le parti de me faire rapidement un template sur InDesign avec les cinq solutions proposées dans la correction, et la place à côté pour des commentaires personnalisés.
Je trouve que c'est plus propre et plus clair pour les apprenants, et ce sont deux points qui jouent en général dans la motivation pour se pencher sur une correction…

J'ai choisi cette solution pour qu'ils aient le code de la correction parce que dans le briefing, si j'ai bien compris, vous expliquez qu'ils n'y ont pas accès directement, c'est bien ça ?
Si j'ai mal compris et que les étudiants ont finalement accès à l'issue des tests aux corrections, je pense que je choisirais de leur envoyer mes commentaires par Google doc ou alors un fichier .txt ou .md «pushé» sur le repository GitHub.

J'ai mis environ une heure et demi à faire l'exercice. J'ai mis la majorité du temps à lire minutieusement le brief, à prendre connaissance des fichiers sur les différentes branches du dépot Git, à découvrir et faire les exercices, et surtout, à réfléchir au ton sur lequel rédiger mes commentaires aux apprenants.
J'ai mis ensuite entre 5 et 10 minutes par étudiant, en passant plus de temps sur les exercices ratés et le Student 4 qui visiblement est fâché avec le JavaScript, et qu'il faut remotiver en évitant de le froisser plus.

Aleth Arrias Camps