

Structures de contrôle en JS

Table des matières

I. Structure de contrôle if/else	3
II. Exercice : Quiz	6
III. La structure de contrôle switch	8
IV. Exercice : Quiz	10
V. Essentiel	12
VI. Auto-évaluation	13
A. Exercice	13
B. Test	13
Solutions des exercices	14

I. Structure de contrôle if/else

Durée : 1 h 30

Prérequis : avoir suivi le cours jusqu'ici

Environnement de travail : Replit

Contexte

En programmation, un point fondamental, quelle que soit la technologie, est la possibilité d'adapter le comportement d'un programme en fonction d'interactions et d'évènements. Cette possibilité est rendue accessible grâce aux structures de contrôle. On trouve des structures de contrôle dans tous les langages de programmation. Vous pourrez constater que la syntaxe en JS est proche de celle de nombreuses autres technologies, comme le Java, le PHP, etc.

Les structures de contrôle sont des instructions permettant de vérifier une condition, et d'adapter le comportement du programme en fonction du résultat obtenu. La base des structures conditionnelles, c'est la condition « **if** » en anglais qui signifie « *si* » en français. Cette instruction va nous permettre de contrôler si une condition est respectée, et si c'est le cas, d'exécuter des blocs de codes. Mais les structures de contrôle ne s'arrêtent pas là. D'autres mots clés comme « **else** » ou « **switch** » permettent de créer des instructions conditionnelles plus poussées et adaptées aux besoins du développeur.

Dans ce cours, nous aborderons les structures de contrôles fondamentales en JavaScript. L'objectif sera de voir à travers des exemples de codes concrets comment les utiliser et quelle est leur syntaxe. Bien évidemment, vous pourrez tester chaque code dans votre IDE. N'hésitez pas à leur apporter des modifications pour vous familiariser avec les conditions en JS. Voyons dans un premier lieu la structure de base **if/else**, puis parlons ensuite de la structure **switch**.

Utilisation de la condition if

Comme vous le savez, des opérateurs de comparaison peuvent renvoyer un booléen : **true** ou **false**. On peut les utiliser pour vérifier si une valeur est égale, inégale, supérieure, etc., à une autre.

Finalement, si l'on écrit :

```
1 let a = 1;
2 let b = 1;
3 console.log(a == b);
```

L'expression **a == b** renverra un booléen, ici **true**, puisque les deux variables sont égales. Si ce n'était pas le cas, elle **renverrait** false. Mais quel rapport avec la condition **if** ?

Eh bien, nous allons pouvoir utiliser cette structure de contrôle en spécifiant en paramètre une condition, c'est-à-dire une expression qui va renvoyer **true** ou **false**. Si et uniquement si cette expression renvoie **true**, alors l'instruction ou les instructions spécifiées seront exécutées. Mais comment écrire une condition **if** ?

Voici la syntaxe :

```
1 if (/*condition*/) {
2   //instructions;
3 }
```

Vous voyez, la syntaxe est assez simple. Prenons un exemple tout simple de condition :

```
1 const nombre = 11;
2
3 if (nombre > 10) {
4   console.log("Le nombre " + nombre + " est plus grand que 10");
5 }
```

On peut voir que l'on spécifie comme condition l'expression **nombre > 10**. Comme la constante **nombre** a pour valeur 11, et que 11 est plus grand que 10, alors l'instruction **console.log("Le nombre " + nombre + " est plus grand que 10");** est exécutée. Donc la console affiche : **le nombre 11 est plus grand que 10**.

En revanche, on peut voir que, si on définit **nombre** sur 10 par exemple :

```
1 const nombre = 10;
2
3 if (nombre > 10) {
4   console.log("Le nombre " + nombre + " est plus grand que 10");
5 }
```

Étant donné que **nombre** n'est pas strictement supérieur à 10, alors l'expression **nombre > 10** va renvoyer **false**. Donc l'instruction **console.log("Le nombre " + nombre + " est plus grand que 10");** ne sera pas exécutée. La console n'affichera rien dans ce cas. On peut voir qu'on peut insérer plusieurs lignes de codes qui seront exécutées si la condition renvoie **true**. Prenons un autre exemple :

```
1 const marque = "Peugeot";
2 const cv = 120;
3 const portes = 5;
4 let prix;
5
6 if (marque == "Peugeot" || marque == "BMW") {
7   prix = (cv * 100 + portes * 30);
8   console.log ("Prix = " + prix);
9 }
```

Dans ce cas, l'expression **marque == "Peugeot" || marque == "BMW"** va renvoyer **true** puisque la variable **marque** est définie sur la chaîne de caractères "Peugeot". Les deux lignes dans le corps du **if** (c'est-à-dire les lignes présentes dans les accolades du **if**) sont donc exécutées, le prix est calculé et affiché dans une chaîne, via la console.

Finalement, cette condition permet d'exécuter les deux lignes du corps de la condition **if** uniquement si la marque est Peugeot ou BMW. Si on prend la marque Renault par exemple :

```
1 const marque = "Renault";
2 const cv = 120;
3 const portes = 5;
4 let prix;
5
6 if (marque == "Peugeot" || marque == "BMW") {
7   prix = (cv * 100 + portes * 30);
8   console.log ("Prix = " + prix);
9 }
```

On peut voir qu'aucune des deux instructions du corps de la condition **if** ne sont exécutées.

Utilisation du mot clé else

Le mot clé **else** (qui signifie littéralement « *sinon* ») permet de dire en quelques sortes : « *Si (if) une condition est vérifiée, exécuter les premières instructions, sinon (else), exécuter les secondes instructions* ». Pour faire simple, le mot clé **else** va permettre d'indiquer les instructions à exécuter si l'expression de la condition précédente renvoie **false**. La syntaxe est très simple :

```
1 if (/*condition*/) {
2   //instructions;
3 }
4 else {
5   //instructions;
6 }
```

Reprenons l'exemple précédent et rajoutons un **else** pour afficher dans la console une chaîne de caractères si la marque n'est ni Peugeot, ni BMW.

```
1 const marque = "Renault";
2 const cv = 120;
3 const portes = 5;
4 let prix;
5
6 if (marque == "Peugeot" || marque == "BMW") {
7   prix = (cv * 100 + portes * 30);
8   console.log ("Prix = " + prix);
9 }
10 else {
11   console.log("La marque " + marque + " n'est pas valide");
12 }
```

Dans ce cas, comme l'expression de la condition renvoie **false**, c'est l'instruction du **else** qui est exécutée. Donc la console affiche :

```
1 La marque Renault n'est pas valide
```

Si l'on change la marque et que l'on écrit BMW :

```
1 const marque = "BMW";
2 const cv = 120;
3 const portes = 5;
4 let prix;
5
6 if (marque == "Peugeot" || marque == "BMW") {
7   prix = (cv * 100 + portes * 30);
8   console.log ("Prix = " + prix);
9 }
10 else {
11   console.log("La marque " + marque + " n'est pas valide");
12 }
```

Alors, ce sont les premières instructions (celles du **if**) qui sont exécutées. La console affiche :

```
1 Prix = 12150
```

Donc le script fonctionne correctement.

Utilisation de else if

Maintenant, nous pouvons parler de l'instruction **else if**, littéralement « *sinon si* ». La structure **else if** va nous permettre de rajouter une condition si l'expression de la condition précédente renvoie **false**.

La syntaxe n'est pas bien plus compliquée :

```
1 if (/*condition 1*/) {
2   //instructions;
3 }
4 else if (/*condition 2*/) {
5   //instructions;
6 }
7 else {
8   //instructions;
9 }
```

Pour prendre un exemple, on peut améliorer notre script précédent pour que la structure de contrôle exécute :

« **Si (if)** la marque est "Peugeot" ou "BMW" et si le nombre de cv est inférieur ou égal à 150, calculer le prix et afficher le prix, **sinon si (else if)** la marque est "Peugeot" ou "BMW", calculer le prix en ajoutant une taxe de 2 000 €, et afficher le prix taxé, **sinon (else)**, afficher que la marque n'est pas valide. » Cette condition est cohérente, nous pouvons donc réaliser notre script :

```
1 const marque = "Peugeot";
2 const cv = 180;
3 const portes = 5;
4 let prix;
5
6 if ((marque == "Peugeot" || marque == "BMW") && cv <= 150) {
7   prix = (cv * 100 + portes * 30);
8   console.log ("Prix = " + prix);
9 }
10 else if (marque == "Peugeot" || marque == "BMW") {
11   prix = ((cv * 100 + portes * 30) + 2000);
12   console.log ("Prix taxé = " + prix);
13 }
14 else {
15   console.log("La marque " + marque + " n'est pas valide");
16 }
```

Dans cet exemple, la condition vérifie si la marque est « Peugeot » ou « BMW », et si le nombre de cv est inférieur ou égal à 150. Comme le nombre de cv n'est pas inférieur ou égal à 150, l'expression de la condition renvoie **false**. Les instructions du **if** ne sont pas exécutées et la condition du **else if** est donc traitée. Comme la marque est égale à **Peugeot**, l'expression de la condition (du **else if**) est validée, elle renvoie **true** et les instructions du corps du **else if** sont exécutées : le prix est calculé avec une taxe de 2 000 et le prix taxé est affiché.

Le **else** aurait été appelé uniquement si la condition du **else if** avait été fausse, donc dans le cas présent si la marque n'avait été ni Peugeot, ni BMW.

Bien évidemment, on peut rajouter autant de **else if** que l'on veut, comme on peut le voir dans cette vidéo :

Exercice : Quiz

[solution n°1 p.15]

Question 1

Voici un extrait de code :

```
1 const nombre = 18;
2
3 if (nombre != 18) {
4   console.log("Le nombre n'est pas égal à 18");
5 }
```

Que va afficher la console ?

- ☐ "Le nombre n'est pas égal à 18"
- ☐ Une erreur
- ☐ Rien

Question 2

Voici un extrait de code :

```
1 const nombre = 18;  
2  
3 if nombre == 18 {  
4   console.log("Le nombre est égal à 18");  
5 }
```

Que va afficher la console ?

- ☐ "Le nombre est égal à 18"
- ☐ Une erreur
- ☐ Rien

Question 3

Voici un extrait de code :

```
1 const typeForme = "cercle";  
2  
3 const rayon = 4;  
4  
5 let aire;  
6  
7 if (typeForme == "cercle") {  
8   aire = Math.PI * (rayon ** 2);  
9 }  
10 else if (typeForme == "sphere") {  
11   aire = 4 * (Math.PI * (rayon ** 2));  
12 }  
13 else {  
14   console.log ("Forme non pris en charge");  
15 }  
16  
17 console.log("L'aire est de " + aire + " cm carré");
```

Que va afficher la console ?

- ☐ Un entier
- ☐ L'aire est de 50.26548245743669 cm carré
- ☐ Une erreur
- ☐ L'aire d'une sphère de rayon 4 cm

Question 4

Voici un extrait de code :

```
1 const typeForme = "carre";  
2  
3 const rayon = 4;  
4  
5 let aire;  
6  
7 if (typeForme == "cercle") {  
8   aire = Math.PI * (rayon ** 2);  
9 }  
10 else if (typeForme == "sphere") {  
11   aire = 4 * (Math.PI * (rayon ** 2));  
12 }
```

```
12 }
13 else {
14   console.log ("Forme non pris en charge");
15 }
```

Que va afficher la console ?

- ☐ Rien
- ☐ Une erreur
- ☐ "Forme non pris en charge"

Question 5

Voici un extrait de code :

```
1 const a = true;
2
3 if (! a) {
4   console.log(1);
5 }
6 else {
7   console.log(2);
8 }
```

Que va afficher la console ?

- ☐ 1
- ☐ a
- ☐ 2

III. La structure de contrôle switch

Introduction à switch

En programmation, la structure de contrôle **switch** existe dans de nombreuses technologies. Mais qu'est-ce que c'est au juste ? La dernière console de Nintendo ? Non évidemment, le terme **switch** peut se traduire littéralement par « interrupteur » ou le verbe « changer ». En programmation, la structure **switch**, aussi souvent appelée **switch/case**, permet de facilement répertorier plusieurs valeurs possibles renvoyées par une expression et de définir les instructions pour chaque cas.

```
1 switch (/*expression*/) {
2   case /*valeur 1*/:
3     //instructions 1;
4     break;
5   case /*valeur 2*/:
6     //instructions 2;
7     break;
8   case /*valeur 3*/:
9     //instructions 3;
10    break;
11  case /*valeur 4*/:
12    //instructions 4;
13    break;
14 }
```


Prenons un exemple très simple. Nous avons une variable **marque** qui correspond à la marque d'un téléphone et on souhaite générer un **console.log** spécifique pour chaque marque. Voici un script qui fonctionne :

```
1 const marque = "Huawei";
2
3 switch (marque) {
4   case "Apple":
5     console.log ("Smartphone haut de gamme avec système IOS");
6     break;
7   case "Samsung":
8     console.log ("Smartphone haut de gamme avec système Android");
9     break;
10  case "Xiaomi":
11    console.log ("Smartphone bon marché avec système Android");
12    break;
13  case "Huawei":
14    console.log ("Smartphone bon marché avec système Android");
15    break;
16 }
```

Dans ce cas, le **switch** va vérifier à chaque **case** (à chaque cas) si **marque** est égale à "Huawei". Si ce n'est pas le cas, elle passe au traitement du **case** suivant. C'est donc pour le **case "Huawei"** que l'instruction va être déclenchée, donc c'est l'instruction **console.log "Smartphone bon marché avec système Android;"** qui est exécutée.

Pour faire simple, le script revient à dire : « Si la marque est "Apple", afficher "Smartphone haut de gamme avec système IOS", si la marque est "Samsung", afficher "Smartphone haut de gamme avec système Android", etc. ». On note que la présence du mot clé **break** est indispensable pour éviter des erreurs. Le **break** permet de terminer le **switch** en cours pour passer aux instructions suivantes du script dans le cas où les instructions d'un **case** sont exécutées. Finalement, une fois qu'un **case** correspond à la valeur renvoyée par la variable **marque**, les instructions du **case** sont exécutées et le **break** permet de sortir du **switch** pour ne pas traiter les **case** qui peuvent être présents après.

Maintenant, qu'est-ce qui se passe si on définit la variable **marque** sur une marque qui n'apparaît pas dans le **switch** ? Par exemple, « Motorola » ? Eh bien tout simplement, rien. Admettons que l'on veuille qu'une instruction soit exécutée pour chaque nom de marque non défini comme **case** dans le **switch**. Pour cela, il faut définir la clause **default**.

La clause default

La clause **default** permet d'indiquer les instructions qui seront exécutées « *par défaut* », c'est-à-dire dans le cas où aucun cas spécifié ne correspond à la valeur renvoyée par l'expression de référence. Bien qu'elle soit facultative, il est fortement préconisé de l'utiliser dans un switch. Pour bien comprendre, appliquons cette notion à notre exemple :

```
1 const marque = "Motorola";
2
3 switch (marque) {
4   case "Apple":
5     console.log ("Smartphone haut de gamme avec système IOS");
6     break;
7   case "Samsung":
8     console.log ("Smartphone haut de gamme avec système Android");
9     break;
10  case "Xiaomi":
11    console.log ("Smartphone bon marché avec système Android");
12    break;
13  case "Huawei":
14    console.log ("Smartphone bon marché avec système Android");
15    break;
16 }
```

```
16 default:
17     console.log("Marque non référencée");
18     break;
19 }
```

On peut voir qu'en spécifiant la clause **default**, on indique quelles instructions va exécuter le **switch** dans le cas où aucun des **case** ne correspond à la valeur de **marque**.

On peut cependant constater que notre **switch** pourrait être amélioré. En effet, pour les marques « *Xiaomi* » et « *Huawei* », les instructions sont exactement les mêmes. Il n'y a donc pas besoin de les répéter deux fois. Comment appliquer la même instruction à deux **case** ?

Appliquer une même suite d'instructions pour plusieurs cas

Pour appliquer une même suite d'instructions pour plusieurs cas, il nous faut simplement préciser chaque **case** concerné avant d'écrire la suite d'instructions, en respectant la syntaxe. Pour bien comprendre, appliquons cette notion à notre exemple :

```
1 const marque = "Huawei";
2
3 switch (marque) {
4     case "Apple":
5         console.log("Smartphone haut de gamme avec système IOS");
6         break;
7     case "Samsung":
8         console.log("Smartphone haut de gamme avec système Android");
9         break;
10    case "Xiaomi":
11    case "Huawei":
12        console.log("Smartphone bon marché avec système Android");
13        break;
14    default:
15        console.log("Marque non référencée");
16        break;
17 }
```

On peut voir que ça fonctionne, les instructions exécutées seront les mêmes pour les marques « *Xiaomi* » et « *Huawei* », à savoir :

```
1     console.log("Smartphone bon marché avec système Android");
2     break;
```

Voilà, vous connaissez maintenant la structure **switch** en JavaScript. La vidéo explique comment utiliser le **switch**, mais nous utiliserons un autre exemple. Nous y reprenons la majorité des points abordés.

Exercice : Quiz

[solution n°2 p.17]

Question 1

Dans un **switch**, la clause **default** est :

- ☐ Facultative
- ☐ Obligatoire
- ☐ Identique à la clause case

Question 2

Quel mot clé est indispensable afin de mettre fin aux instructions d'un **case** ?

- ☐ alert
- ☐ exit
- ☐ break

Question 3

Voici un extrait de code :

```
1 const MARQUE = "BMW";
2
3 let affichage;
4
5 switch (MARQUE) {
6   case "Peugeot":
7     affichage = "La marque est Peugeot";
8     break;
9   case "Renault":
10    affichage = "La marque est Renault";
11    break;
12   case "Audi":
13     affichage = "La marque est Audi";
14     break;
15 }
16
17 console.log(affichage);
```

Que va afficher la console ?

- ☐ "La marque est Peugeot"
- ☐ "undefined"
- ☐ "La marque est BMW"

Question 4

Voici un extrait de code différent :

```
1 const MARQUE = "BMW";
2
3 let affichage;
4
5 switch (MARQUE) {
6   case "Peugeot":
7     affichage = "La marque est Peugeot";
8     break;
9   case "Renault":
10    affichage = "La marque est Renault";
11    break;
12   case "Audi":
13     affichage = "La marque est Audi";
14     break;
15   default:
16     affichage = "La marque n'est pas référencée";
17     break;
18 }
19
20 console.log(affichage);
```

Que va afficher la console ?

- ☐ "undefined"
- ☐ Une erreur
- ☐ "La marque n'est pas référencée"

Question 5

Voici un extrait de code différent :

```
1 const MARQUE = "BMW";
2
3 let affichage;
4
5 switch (MARQUE) {
6   case "Peugeot":
7     affichage = "La marque est Peugeot";
8     break;
9   case "Renault":
10    affichage = "La marque est Renault";
11    break;
12   case "Audi":
13     affichage = "La marque est Audi";
14     break;
15   default:
16     affichage = "La marque n'est pas référencée";
17     break;
18   case "BMW":
19     affichage = "La marque est BMW";
20     break;
21 }
22
23 console.log(affichage);
```

Que va afficher la console ?

- ☐ "undefined"
- ☐ "La marque est BMW"
- ☐ Une erreur
- ☐ "La marque n'est pas référencée"

V. Essentiel

Les structures de contrôles sont des ensembles d'instructions permettant d'adapter le comportement d'un programme au résultat de conditions. On comprend donc que les structures de contrôle vont jouer un rôle plus qu'essentiel dans la mise en place d'interactivité sur les sites web avec JavaScript.

La structure de base est **if**. Les instructions d'un **if** seront exécutées uniquement dans le cas où la condition renvoie **true**. Le mot clé **else** (qui signifie sinon) permet de définir des instructions qui seront exécutées dans le cas où l'instruction précédente n'est pas exécutée. Entre un **if** et un **else**, nous pouvons rajouter des **else if** (sinon si). Bien évidemment, les structures conditionnelles peuvent être imbriquées et construire ainsi des mécanismes complexes de contrôle.

La structure **switch** aussi appelée **switch/case** permet de définir rapidement le comportement spécifique du programme pour chaque valeur renvoyée par une expression. La clause **case** permet de définir des instructions pour une valeur renvoyée spécifique. La clause **default** (facultative) permet quant à elle de spécifier une suite

d'instructions par défaut, c'est-à-dire qui sera exécutée dans le cas où aucun **case** ne spécifie la valeur renvoyée de l'expression actuelle. Le mot **break** est indispensable pour que le programme sorte du **switch** au moment où les instructions d'un **case** sont exécutées.

Voilà, vous connaissez maintenant les structures de contrôle fondamentales en JS. Ces systèmes sont syntaxiquement très ressemblants dans de nombreux autres langages comme Java ou PHP.

VI. Auto-évaluation

A. Exercice

Vous cherchez à réaliser un programme permettant à un réparateur d'ordinateurs de savoir si un modèle spécifique est pris en charge par ses équipes. Chaque ordinateur à plusieurs caractéristiques. Voici votre script de base :

```
1 const marque;  
2  
3 const processeur;  
4  
5 const ram;  
6  
7 const stockage;  
8  
9 //conditions
```

Le réparateur ne prend en charge que les ordinateurs de marque Apple, HP et Microsoft. La ram doit aussi être supérieure ou égale à 4 GO pour que l'ordinateur soit pris en charge. Les ordinateurs de moins de 256 GO de stockage ne sont pas pris en charge.

Question 1

[solution n°3 p.19]

Construire un système permettant d'indiquer dans la console si l'ordinateur est pris en charge ou non, en fonction de ces critères. Tester le script en définissant les variables sur des valeurs spécifiques.

Question 2

[solution n°4 p.19]

Si la marque est **Apple**, le réparateur demande que pour plusieurs processeur, l'utilisateur soit informé du temps de réparation maximum :

Intel core i7 : "Le temps de réparation est de 25 jours max"

M1 : "Le temps de réparation est de 10 jours max"

M2 : "Le temps de réparation est de 20 jours max"

Autre processeur : "Le temps de réparation est de 35 jours max"

Pour les ordinateurs des autres marques, le temps de réparation sera le même : 20 jours max. L'utilisateur devra aussi en être informé.

Modifier le script pour que ces changements soient pris en compte.

B. Test

Exercice 1 : Quiz

[solution n°5 p.20]

Question 1

Quelle est la syntaxe d'un **if** en JavaScript (sans l'indentation) ?

- ☐ if (/*condition*/) { /*instructions;*/ }
- ☐ if /*condition*/ { /*instructions;*/ }
- ☐ if /*condition*/ : /*instructions;*/

Question 2

Quelle est la syntaxe du switch en JavaScript (sans l'indentation) ?

- ☐ switch /*expression*/: case /*value 1*/: /*instructions*/
- ☐ switch /*expression*/ {case /*value 1*/: /*instructions*/}
- ☐ switch (/*expression*/) {case /*value 1*/: /*instructions*/; break;}

Question 3

Que signifie le mot clé **else** :

- ☐ Sinon
- ☐ Si
- ☐ Sinon si

Question 4

L'instruction **else if** peut être utilisée dans une structure de contrôle :

- ☐ Autant de fois que nécessaire
- ☐ Avant le **if**
- ☐ 2 fois maximum

Question 5

Dans une structure de contrôle avec **if**, le mot clé **else** est :

- ☐ Facultatif
- ☐ Obligatoire
- ☐ Utilisé avant le **if**

Solutions des exercices

Exercice p. 6 Solution n°1


Question 1

Voici un extrait de code :

```
1 const nombre = 18;  
2  
3 if (nombre !== 18) {  
4   console.log("Le nombre n'est pas égal à 18");  
5 }
```

Que va afficher la console ?

- ☐ "Le nombre n'est pas égal à 18"
- ☐ Une erreur
- ☒ Rien

 Ici, il n'y a aucune erreur de syntaxe. L'expression **nombre !== 18** renvoie **false** car **nombre = 18**. Donc l'instruction inscrite dans le corps du **if** ne s'exécute pas. La console n'affiche donc rien.


Question 2

Voici un extrait de code :

```
1 const nombre = 18;  
2  
3 if nombre == 18 {  
4   console.log("Le nombre est égal à 18");  
5 }
```

Que va afficher la console ?

- ☐ "Le nombre est égal à 18"
- ☒ Une erreur
- ☐ Rien

 Ici, il y a une erreur de syntaxe car l'expression de la condition n'est pas inscrite entre parenthèses. La console affiche donc une erreur.

Question 3


Voici un extrait de code :

```
1 const typeForme = "cercle";  
2  
3 const rayon = 4;  
4  
5 let aire;  
6  
7 if (typeForme == "cercle") {  
8   aire = Math.PI * (rayon ** 2);  
9 }  
10 else if (typeForme == "sphere") {  
11   aire = 4 * (Math.PI * (rayon ** 2));  
12 }  
13 else {  
14   console.log ("Forme non pris en charge");  
15 }
```

```
15 }
16
17 console.log("L'aire est de " + aire + " cm carré");
```

Que va afficher la console ?

- ☐ Un entier
- ☒ L'aire est de 50.26548245743669 cm carré
- ☐ Une erreur
- ☐ L'aire d'une sphère de rayon 4 cm

 Ici, c'est l'instruction du **if** qui est exécutée car **typeForme** est bien égal à « *cercle* ». Cette instruction correspond au calcul d'une aire d'un cercle : **aire = Math.PI * (rayon ** 2)**. On utilise la constante **Math.PI** qui permet d'obtenir la valeur de π et l'opérateur ****** permettant d'élever une valeur à une puissance.


Question 4

Voici un extrait de code :

```
1 const typeForme = "carre";
2
3 const rayon = 4;
4
5 let aire;
6
7 if (typeForme == "cercle") {
8   aire = Math.PI * (rayon ** 2);
9 }
10 else if (typeForme == "sphere") {
11   aire = 4 * (Math.PI * (rayon ** 2));
12 }
13 else {
14   console.log ("Forme non pris en charge");
15 }
```

Que va afficher la console ?

- ☐ Rien
- ☐ Une erreur
- ☒ "Forme non pris en charge"

 Ici, les deux premières conditions renvoient **false**. C'est donc l'instruction du **else** qui est exécutée, c'est-à-dire **console.log ("Forme non pris en charge")**.

Question 5

Voici un extrait de code :

```
1 const a = true;
2
3 if (! a) {
4   console.log(1);
5 }
6 else {
7   console.log(2);
8 }
```

Que va afficher la console ?

- ☐ 1
- ☐ a
- ☒ 2

Q Ici, la variable **a** est définie sur le booléen **true**. Elle renvoie donc **true**. Or, nous utilisons dans l'expression en paramètre de la condition l'opérateur NON : **!**. Donc, si **a = true**, alors **! a = false**. L'expression renvoyant **false**, c'est l'instruction du **else** qui est exécutée : **console.log(2)**.

Exercice p. 10 Solution n°2

Question 1

Dans un **switch**, la clause **default** est :

- ☒ Facultative
- ☐ Obligatoire
- ☐ Identique à la clause case

Q La clause **default** qui permet de définir des instructions par défaut est facultative. Les instructions du **default** seront exécutées pour tous les cas qui ne sont pas référencés dans le **switch**.

Question 2

Quel mot clé est indispensable afin de mettre fin aux instructions d'un **case** ?

- ☐ alert
- ☐ exit
- ☒ break

Q Le mot clé **break** permet, dans le cas où un **case** est exécuté, de terminer le **switch** pour passer aux instructions suivantes, sans analyser les autres cas.

De plus, il serait intéressant de parler de la notion de Fall-Through : <https://www.freecodecamp.org/news/fall-through-in-javascript-switch-statements/>

Question 3

Voici un extrait de code :

```
1 const MARQUE = "BMW";
2
3 let affichage;
4
5 switch (MARQUE) {
6   case "Peugeot":
7     affichage = "La marque est Peugeot";
8     break;
9   case "Renault":
10    affichage = "La marque est Renault";
11    break;
12   case "Audi":
13     affichage = "La marque est Audi";
14     break;
15 }
16
17 console.log(affichage);
```

Que va afficher la console ?

- ☐ "La marque est Peugeot"
- ☒ "undefined"
- ☐ "La marque est BMW"

Q Il n'y a aucune erreur de syntaxe, mais le cas où **marque** serait égal à **BMW** n'est pas référencé dans le **switch**, et la clause **default** n'est pas définie. La variable **affichage** n'est donc pas définie, elle n'a aucune valeur. La console affiche donc "undefined".

Question 4

Voici un extrait de code différent :

```
1 const MARQUE = "BMW";
2
3 let affichage;
4
5 switch (MARQUE) {
6   case "Peugeot":
7     affichage = "La marque est Peugeot";
8     break;
9   case "Renault":
10    affichage = "La marque est Renault";
11    break;
12   case "Audi":
13    affichage = "La marque est Audi";
14    break;
15   default:
16    affichage = "La marque n'est pas référencée";
17    break;
18 }
19
20 console.log(affichage);
```

Que va afficher la console ?

- ☐ "undefined"
- ☐ Une erreur
- ☒ "La marque n'est pas référencée"

Q Dans ce cas, il n'y a pas d'erreur de syntaxe. Le cas où **marque** serait égale à "BMW" n'est pas défini dans le **switch**, mais la clause **default** l'est. On indique que par défaut, le **switch** devra définir la variable **affichage** sur la chaîne de caractères : "La marque n'est pas référencée". La constante **marque** étant égale à "BMW", alors **affichage** est définie sur la chaîne "La marque n'est pas référencée".

Question 5

Voici un extrait de code différent :

```
1 const MARQUE = "BMW";
2
3 let affichage;
4
5 switch (MARQUE) {
6   case "Peugeot":
7     affichage = "La marque est Peugeot";
8     break;
```


```

9  case "Renault":
10     affichage = "La marque est Renault";
11     break;
12  case "Audi":
13     affichage = "La marque est Audi";
14     break;
15  default:
16     affichage = "La marque n'est pas référencée";
17     break;
18  case "BMW":
19     affichage = "La marque est BMW";
20     break;
21 }
22
23 console.log(affichage);

```

Que va afficher la console ?

- ☐ "undefined"
- ☒ "La marque est BMW"
- ☐ Une erreur
- ☐ "La marque n'est pas référencée"

 Ici, le cas ou **marque** = "BMW" est référencé dans le **switch**. Dans ce cas, **affichage** sera définie sur la chaîne de caractères : "La marque est BMW". Donc, **affichage** est égal à la fin du script à : "La marque est BMW".

p. 13 Solution n°3

Voici un script qui fonctionne, en utilisant la structure if/else :

```

1  const marque = "Apple";
2
3  const processeur = "M1";
4
5  const ram = 8;
6
7  const stockage = 256;
8
9  if ((marque == "Apple" || marque == "HP" || marque == "Microsoft") && ram >= 4 && stockage >=
10     256) {
11     console.log ("L'ordinateur est pris en charge");
12 }
13 else {
14     console.log("L'ordinateur n'est pas pris en charge");
15 }

```

p. 13 Solution n°4

Voici un script qui fonctionne, en utilisant la structure **if/else** et le **switch** :

```

1  const marque = "Apple";
2
3  const processeur = "M2";
4
5  const ram = 8;
6

```

```

7 const stockage = 256;
8
9 if ((marque == "Apple" || marque == "HP" || marque == "Microsoft") && ram >= 4 && stockage >=
10 256) {
11     console.log ("L'ordinateur est pris en charge");
12     if (marque == "Apple") {
13         switch (processeur) {
14             case "Intel core I7":
15                 console.log ("Le temps de réparation est de 25 jours max");
16                 break;
17             case "M1":
18                 console.log ("Le temps de réparation est de 10 jours max");
19                 break;
20             case "M2":
21                 console.log ("Le temps de réparation est de 20 jours max");
22                 break;
23             default:
24                 console.log ("Le temps de réparation est de 35 jours max");
25                 break;
26         }
27     }
28     else {
29         console.log ("Le temps de réparation est de 20 jours max");
30     }
31 }
32 else {
33     console.log ("L'ordinateur n'est pas pris en charge");
34 }

```

N'hésitez pas à tester le code et à opérer des modifications pour bien comprendre comment il fonctionne.

Exercice p. 13 Solution n°5

Question 1

Quelle est la syntaxe d'un **if** en JavaScript (sans l'indentation) ?

- ☒ if (/*condition*/) { /*instructions;*/ }
- ☐ if /*condition*/ { /*instructions;*/ }
- ☐ if /*condition*/ : /*instructions;*/
- ☐ La syntaxe pour utiliser la structure **if** est **if (/*condition*/) { /*instructions;*/ }**.

Question 2


Quelle est la syntaxe du **switch** en JavaScript (sans l'indentation) ?

- ☐ switch /*expression*/ : case /*value 1*/ : /*instructions*/
- ☐ switch /*expression*/ { case /*value 1*/ : /*instructions*/ }
- ☒ switch (/*expression*/) { case /*value 1*/ : /*instructions*/ ; break; }
- ☐ La syntaxe pour utiliser la structure **switch** est sans indentation : **switch (/*expression*/) { case /*value 1*/ : /*instructions*/ ; break; }**.

Question 3

Que signifie le mot clé **else** :


- ☒ Sinon
- ☐ Si
- ☐ Sinon si

 Le mot clé **else** signifie « *sinon* ». Il est utilisé pour exécuter des instructions dans le cas où l'expression de la condition précédente n'est pas validée (renvoie **false**).

Question 4

L'instruction **else if** peut être utilisée dans une structure de contrôle :


- ☒ Autant de fois que nécessaire
- ☐ Avant le **if**
- ☐ 2 fois maximum

 On peut insérer autant de fois que nécessaire des **else if** dans une structure de contrôle. Ils signifient littéralement « *sinon si* ». On peut donc l'utiliser après un **if**.

Question 5

Dans une structure de contrôle avec **if**, le mot clé **else** est :

- ☒ Facultatif
- ☐ Obligatoire
- ☐ Utilisé avant le **if**

 Dans une structure de contrôle, le mot clé **else** est facultatif. S'il est utilisé, il doit obligatoirement être placé après le **if**, et c'est logique (on dit « *si* » avant « *sinon* » et pas l'inverse).