L'objet console

Table des matières

I. Contexte

Durée : 45 min

Environnement de travail : Repl.it

Pré-requis : Connaître les bases de JavaScript

Contexte

Vous allez apprendre dans ce cours les différentes possibilités d'entrées et de sorties, que ce soit pour déboguer votre code JavaScript, afficher un message à l'internaute, lui demander une confirmation ou lui faire saisir une valeur.

II. L'objet console

Objectifs

- Découvrir l'objet `console`.
- Savoir l'utiliser pour répondre à votre besoin.

Mise en situation

Pour visualiser la console de votre navigateur et ainsi voir ses messages, il existe deux solutions. Effectuez un clic droit sur la page web choisie afin d'accéder aux outils du navigateur, puis cliquez sur "Examiner" ou "Inspecter" et enfin rendez-vous sur l'onglet "Console", ou bien, appuyez directement sur la touche "F12" et rejoignez ce même onglet.

Rappel **Afficher la console de votre navigateur web**

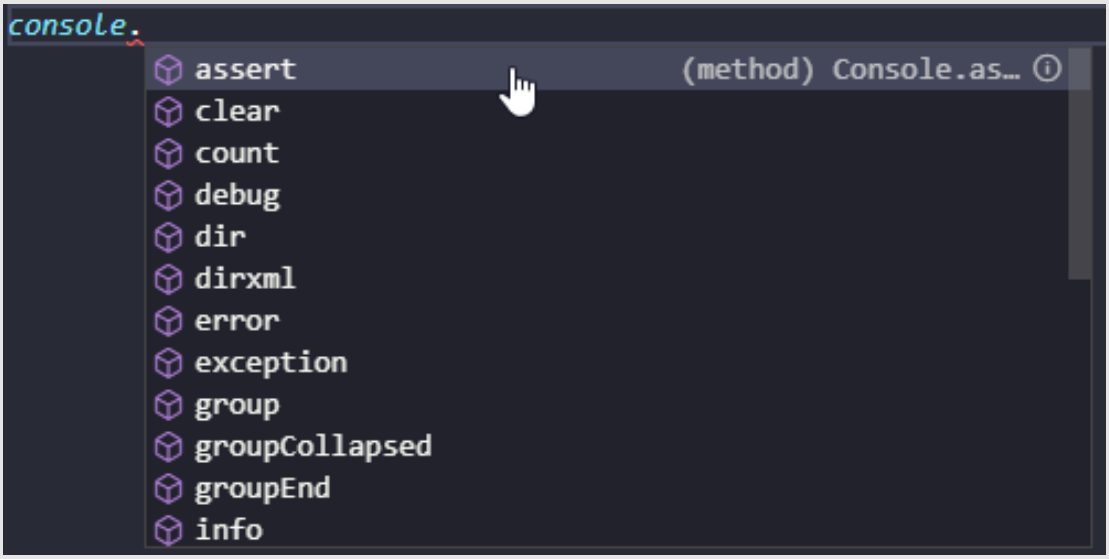
Pour visualiser la console de votre navigateur et ainsi voir ses messages, il existe deux solutions.

Effectuez un clic droit sur la page web choisie afin d'accéder aux outils du navigateur, puis cliquez sur "Examiner" ou "Inspecter" et enfin rendez-vous sur l'onglet "Console", ou bien, appuyez directement sur la touche "F12" et rejoignez ce même onglet.

Méthode **Dans le code JavaScript**

Pour utiliser l'objet `console`, il suffit de taper `console` dans le code, puis d'y ajouter un point afin d'appeler la méthode désirée.

Les bons éditeurs de code vous affichent alors une liste des méthodes disponibles pour cet objet, comme dans l'exemple ci-dessous avec Visual Studio Code.




```
console.info("Hello world !")
```

Méthode **La méthode info()**

Affiche un message d'information dans la console et prend en paramètre un message ou une liste d'objets à afficher.

Le message est affiché sans mise en forme particulière (avec une icône "Information" dans Firefox).

```
const val = 4
const min = 8
const infoMessage = 'Début de la fonction'

console.info('GO !')
console.info({val, min, infoMessage})
```

La console va afficher :

```
GO !
{val: 4, min: 8, infoMessage: "Début de la fonction"}
```

Méthode **La méthode log()**

La méthode log() affiche un message dans la console : **c'est une des fonctionnalités qui vous sera le plus utile pour afficher les valeurs de vos variables lors des phases de développement ou de débogage**.

Elle prend en paramètre un message ou une liste d'objets à afficher.

```
const val = 4
const numbers = [4, 8, 14, 2]

console.log('Debug')
console.log(val)
console.log(numbers)
```

Le code ci-dessus va afficher dans la console :

```
Debug
4
```

```
Array(4) [4, 8, 14, 2]
```

Méthode **La méthode error()**

Cette méthode affiche une erreur dans la console.

Elle aide à trouver l'origine de l'erreur en indiquant la trace d'appel (*stack trace*), c'est-à-dire le chemin parcouru depuis le début du programme pour arriver jusqu'à l'erreur avec, à chaque étape, le fichier et la ligne concernés.

console.error prend en paramètre un message ou une liste d'objets à afficher.

Le message est affiché en rouge avec une icône d'erreur rouge.

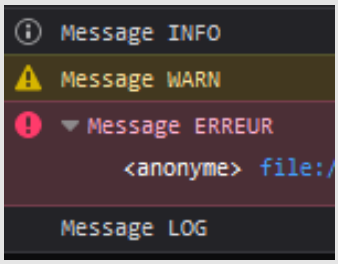
```
1 console.error('Une erreur est survenue')
```

Méthode **La méthode warn()**

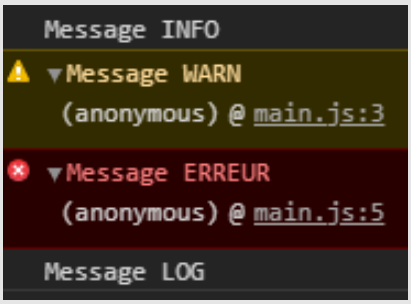
Affiche un message d'avertissement dans la console et prend en paramètre un message ou une liste d'objets à afficher.

Le message est affiché en jaune avec une icône d'avertissement jaune.

```
1 console.info('Message INFO')
2
3 console.warn('Message WARN')
4
5 console.error('Message ERREUR')
6
7 console.log('Message LOG')
```



Affichage d'un message warn() sur Firefox



Affichage d'un message warn() sur Chrome

Méthode **La méthode assert()**

Cette méthode affiche un message d'erreur donné dans la console, uniquement si la condition définie est fausse.

- Le premier paramètre est la condition à vérifier.
- Le deuxième paramètre est une liste d'objets JavaScript à afficher.

```
const age = 16
const majority = 18
const assertReason = 'La valeur âge doit être supérieure à la valeur de la majorité'

console.assert(age > majority, {age, majority, assertReason})
```

Dans l'exemple ci-dessus, on souhaite vérifier que `age` est bien supérieur à `majority` et si ce n'est pas le cas on souhaite afficher un message d'erreur dans la console.

Comme ce n'est pas le cas, la console va afficher : `Assertion failed: Object { âge: 16, majority: 18, assertReason: 'La valeur âge doit être supérieure à la valeur de la majorité' }`

Méthode La méthode clear()

Cette méthode va tout simplement vider la console.

```
const age = 16
const majority = 18
const assertReason = 'La valeur âge doit être supérieure à la valeur de la majorité'

console.assert(age > majority, {age, majority, assertReason})

console.clear()
```

Dans l'exemple ci-dessus, le message de la méthode `assert()` ne sera pas affiché, car la méthode `clear()` est déclenchée juste après et va donc vider la console, qui affichera uniquement : `La console a été effacée.`

Méthode La méthode count()

Cette méthode permet d'afficher dans la console le nombre de fois où elle a été appelée.

Elle accepte éventuellement un paramètre label qui permet de savoir combien de fois count() a été appelée avec ce label.

```
  console.count()
    for(let i=0; i<4;i++) {
    console.count('Dans la boucle');
  }
     console.count()
```

Dans l'exemple ci-dessus, on parcourt une boucle. Celle-ci commence à 0, s'incrémente de 1 pas et se termine lorsque le résultat est strictement inférieur à 4. La console affiche donc :

```
default: 1
Dans la boucle : 1
Dans la boucle : 2
Dans la boucle : 3
Dans la boucle : 4
default: 2
```

Au total, on a donc bien quatre `count()` effectués dans la boucle et deux count() effectués en dehors de celle-ci.

Méthode La méthode countReset()

Cette méthode va simplement remettre le compteur vu précédemment à zéro.

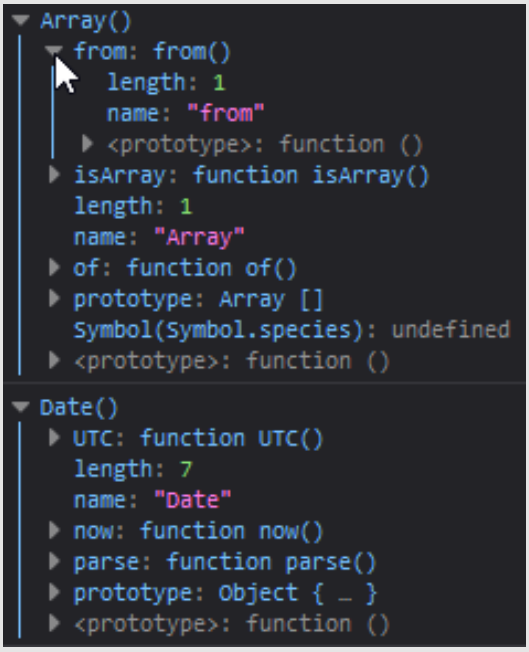
Elle peut également prendre `label` en paramètre afin de pouvoir remettre à zéro un compteur spécifique déclaré avec un `label`.

```
console.countReset()
console.countReset('supérieur')
```

Méthode **La méthode dir()**

Cette méthode affiche la liste des propriétés de l'objet passé en paramètre. Des flèches permettent de déplier les informations.

```
// Array object
console.dir(Array)

// Date object
console.dir(Date)
```



Complément

La méthode `dirxml()` affiche un arbre des éléments enfants de l'élément XML spécifié, mais il est également possible de spécifier un élément HTML.

Exemple avec le fichier HTML ci-dessous :

```
<!DOCTYPE html>
<html lang="fr">

<head>
  <meta charset="UTF-8">
  <title>Entrées, sorties</title>

  <script src="main.js"></script>
</head>

<body>
</body>

</html>
```

```
console.dirxml(document.head)
```

Le résultat dans la console de Chrome sera :

```
<head>
  <meta charset="UTF-8">
  <title>Entrées, sorties</title>

  <script src="main.js"></script>
</head>
```

Méthode **Les méthodes group(), groupCollapsed() et groupEnd()**

Ces trois méthodes fonctionnent de concert :

- `group()` permet de grouper plusieurs messages de la console dans un élément rétractable pour améliorer la lisibilité,
- `groupCollapsed()` fait la même chose que `group()`, mais le groupe est rétracté par défaut, l'utilisateur devra cliquer sur le bouton pour voir le contenu,
- `groupEnd()` ferme le groupe en cours, tous les messages de console affichés après cette ligne ne seront pas affichés dans l'élément rétractable.

Méthode **La méthode table()**

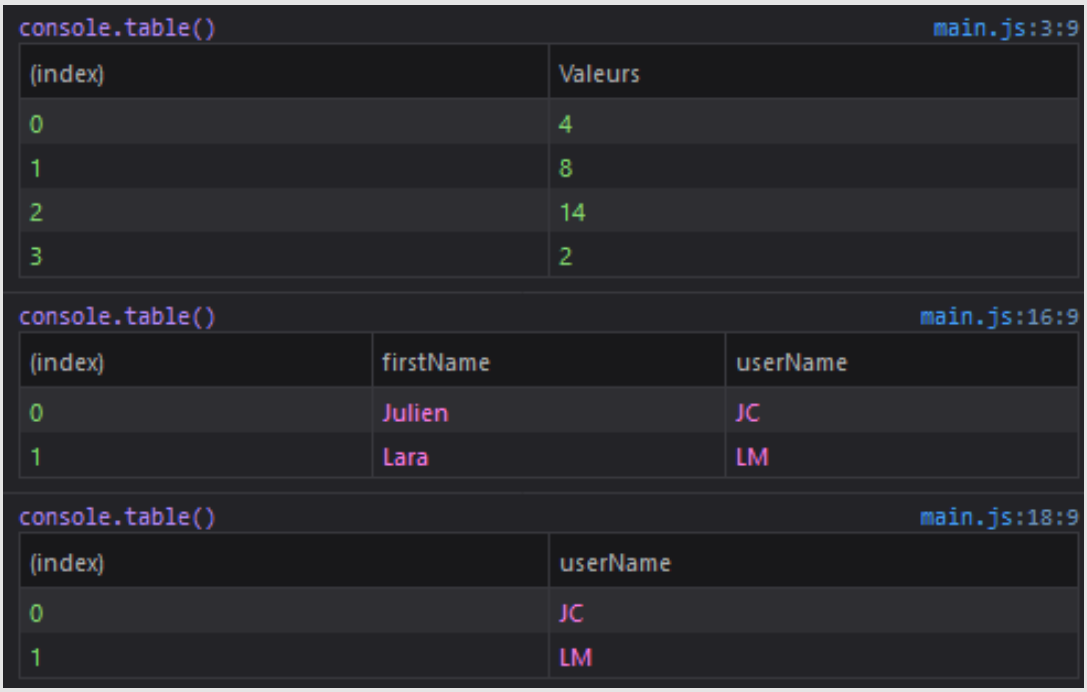
Affiche les données des tableaux JavaScript (Array()) ou des objets sous forme d'un tableau dans la console.

Le premier paramètre est obligatoire : ce doit être un tableau ou un objet.

Le deuxième paramètre est facultatif : le tableau contient les noms des colonnes à afficher dans la sortie.

```
const numbers = [4, 8, 14, 2]

console.table(numbers)

const persons = [
  {
    firstName: 'Julien',
    userName: 'JC',
  },
  {
    firstName: 'Lara',
    userName: 'LM',
  },
]

console.table(persons)

console.table(persons, ['userName'])
```

Le code ci-dessus va afficher :



Méthode Les méthodes time(), timeLog() et timeEnd()

La méthode `time()` démarre un chronomètre : il peut être utile pour mesurer le temps pris par une opération.

La méthode `timeEnd()` arrête ce même chronomètre : la console affiche alors le temps en millisecondes.

La méthode `timeLog()` affiche la valeur actuelle du chronomètre. Vous pouvez utiliser `time()` et `timeEnd()` sans utiliser `timeLog()` : cette méthode permet d'afficher la valeur actuelle du chronomètre à des emplacements choisis.

Il est possible de passer un paramètre "`label`" à ces trois fonctions afin de démarrer, afficher et arrêter un chronomètre précis. Vous pourrez ainsi lancer plusieurs chronomètres simultanément, les afficher et les arrêter via leurs labels.

Méthode La méthode trace()

La méthode suivante affiche la trace d'appel (*stack trace*) dans la console.

Cette fonction est notamment intéressante, tout comme *console.error / log*, pour du débogage.

```
first()

function first() {
  second()
}

function second() {
  third()
}

function third() {
  console.trace()
}
```

Dans le code ci-dessus, on lance la méthode `first()`, qui lance `second()`, qui lance `third()`, dans laquelle on appelle `trace()`.

La console nous donne bien le détail du chemin parcouru :

- `console.trace`
- `third @ main.js:12`
- `second @ main.js:8`
- `first @ main.js:4`
- `(anonymous) @ main.js:1`

Syntaxe **À retenir**

La méthode `log()` est **une des fonctionnalités qui vous sera le plus utile pour afficher les valeurs de vos variables lors des phases de développement ou de débogage**.

`assert()` : affiche un message d'erreur donné dans la console, uniquement si la condition définie est fausse.

`table()` : affiche les données des tableaux JavaScript (Array()) ou des objets sous forme d'un tableau dans la console.

`trace()` : affiche la trace d'appel (stack trace) dans la console, c'est-à-dire le chemin parcouru pour atteindre l'emplacement du code où vous avez appelé cette méthode.

Attention

L'objet console est très utile durant le développement pour vous permettre de déboguer rapidement ou de visualiser le comportement de votre application. Cependant, quand votre application est prête pour fonctionner en production, il convient de supprimer toutes les lignes console de votre code. En effet, elles constituent une perte – certes, minime – de performance, mais c'est également autant de pistes que vous laissez à un potentiel attaquant pour comprendre comment votre code fonctionne.

Complément

Console (MDN web docs)[1]

Console API Reference (Chrome DevTools)[2]

Traces d'appel (Stack traces)[3]

III. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail Repl.it.

[4]

1 https://developer.mozilla.org/fr/docs/Web/API/Console
2 https://developers.google.com/web/tools/chrome-devtools/console/api
3 https://developer.mozilla.org/fr/docs/Web/API/console#Traces_d'appel
4 https://repl.it/

Question

[solution n°1 p.15]

En utilisant le tableau des nombres ci-dessous, vous allez devoir :

1. Utiliser la méthode vous permettant de l'afficher sous la forme d'un tableau dans la console.
2. Utiliser la méthode permettant d'afficher son contenu dans la console comme si vous deviez déboguer.

```
const numbers = [4, 8, 14, 2]
```

3. À partir du code JavaScript ci-dessous, il faut remplacer **XX** par une valeur permettant de faire afficher le message dans la console.

```
const min = 10
const assertReason = 'La valeur doit être supérieure à 10'
console.assert(XX > min, assertReason)
```

4. Dans l'exemple de code ci-dessous, nous voulons connaître le chemin parcouru pour arriver jusqu'à la fonction `logged()`. Il faut copier/coller ce code et placer la méthode adéquate au bon endroit afin de faire afficher la trace d'appels.

```
// 4)
initialize()

function logged() {
  console.log('Hello')
}

function login() {
  logged()
}

function initialize() {
  login()
}
```

IV. Auto-évaluation

A. Exercice final

Exercice 1

[solution n°2 p.15]

Exercice

L'utilisation de la console du navigateur est destinée aux développeurs lors des phases de création et de débogage de code.

- ○ Vrai
- ○ Faux

Exercice

La méthode `assert()` de l'objet console permet d'afficher un message si une condition donnée est vraie.

- ○ Vrai
- ○ Faux

Exercice

La méthode `clear()` de l'objet console permet :

- O De vider les variables JavaScript.
- O D'effacer la console du navigateur web.

Exercice

La méthode `count()` de l'objet console permet :

- O De déclencher un compteur qui s'incrémente de 1 (+1) à chaque appel de la méthode.
- O De retourner le nombre d'éléments du tableau qu'on lui passe en paramètre.

Exercice

La méthode `dir()` de l'objet console :

- O Liste le contenu d'un dossier dont le chemin est passé en paramètre.
- O Affiche la liste des propriétés de l'objet passé en paramètre.

Exercice

Quelles sont les méthodes qui permettent d'afficher des messages dans la console ?

- ☐ info()
- ☐ warn()
- ☐ error()
- ☐ log()

Exercice

La méthode `table()` de l'objet console :

- O Crée un tableau visuel dans la console à partir d'un tableau ou d'un objet JavaScript passé en paramètre.
- O Permet de créer un tableau (balise <table>) dans la page HTML.

B. Exercice : Défi

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail Repl.it :

1

Question

[solution n°3 p.16]

Marion souhaite fêter son anniversaire dans une salle des fêtes pouvant contenir 50 personnes.

Vous allez devoir, à partir du code source ci-après :

1. Vérifiez que le nombre d'invités n'excède pas 50 personnes

2. Dans le cas où le nombre de personnes invitées dépasse la capacité de la salle, affichez "Le nombre d'invités est trop important"

3. Affichez le nombre d'invités dans la console

1 https://repl.it/

Supprimez les lignes "`// Write your code here instead of this comment`" et écrivez votre code à la place.

```
    // 1) Créez une variable pour le nombre max de personnes pouvant être contenues dans la salle ainsi que le nombre d'invités.
    // Write your code here instead of this comment
    // Write your code here instead of this comment
    if (// Write your code here instead of this comment) {
      // Write your code here instead of this comment
  } else {
        // Write your code here instead of this comment
    }

    //2)
    // Write your code here instead of this comment
    // Write your code here instead of this comment
    // 3) The user name is displayed in the console.
    // Write your code here instead of this comment
```

Indice :

Vous aurez besoin d'utiliser console.assert et console.log.

Solutions des exercices

p. 11 Solution n°1

```
const numbers = [4, 8, 14, 2]

// 1) Display the array of numbers as an array in the console
console.table(numbers)

// 2) Display the contents of the variable for debugging
console.log(numbers)

// 3) Display the error message because 4 is less than 10
const min = 10
const assertReason = 'La valeur doit être supérieure à 10'
console.assert(4 > min, assertReason)

// 4)
initialize()

function logged() {
  console.trace()
  console.log('Hello')
}

function login() {
  logged()
}

function initialize() {
  login()
}
```

Exercice p. 11 Solution n°2

Exercice

L'utilisation de la console du navigateur est destinée aux développeurs lors des phases de création et de débogage de code.

- ◉ Vrai
- ○ Faux

C'est vrai.

Exercice

La méthode `assert()` de l'objet console permet d'afficher un message si une condition donnée est vraie.

- ○ Vrai
- ◉ Faux

C'est faux, cette méthode affiche un message uniquement si la condition est fausse.

Exercice

La méthode `clear()` de l'objet console permet :

- ○ De vider les variables JavaScript.
- ◉ D'effacer la console du navigateur web.

La méthode `clear()` permet d'effacer la console du navigateur web.

Exercice

La méthode `count()` de l'objet console permet :

- ◉ De déclencher un compteur qui s'incrémente de 1 (+1) à chaque appel de la méthode.
- ○ De retourner le nombre d'éléments du tableau qu'on lui passe en paramètre.

La méthode `count()` de l'objet console permet de déclencher un compteur qui s'incrémente de 1 (+1) à chaque appel de la méthode.

Exercice

La méthode `dir()` de l'objet console :

- ○ Liste le contenu d'un dossier dont le chemin est passé en paramètre.
- ◉ Affiche la liste des propriétés de l'objet passé en paramètre.

La méthode `dir()` de l'objet console affiche la liste des propriétés de l'objet passé en paramètre.

Exercice

Quelles sont les méthodes qui permettent d'afficher des messages dans la console ?

- ☑ info()
- ☑ warn()
- ☑ error()
- ☑ log()

Toutes ces méthodes permettent d'afficher un message dans la console.

Exercice

La méthode `table()` de l'objet console :

- ◉ Crée un tableau visuel dans la console à partir d'un tableau ou d'un objet JavaScript passé en paramètre.
- ○ Permet de créer un tableau (balise <table>) dans la page HTML.

La méthode `table()` de l'objet console crée un tableau visuel dans la console à partir d'un tableau ou d'un objet JavaScript passé en paramètre.

p. 12 **Solution n°3**

```
    // 1) Créez une variable pour le nombre max de personnes pouvant être contenues dans la salle ainsi que le nombre d'invités.
   const peakNumber = 50;
    let guestsNumber = 50;
    if (guestsNumber <= peakNumber ) {
      console.log(true);
  } else {
      console.log(false);
    }

    //2)
  const message = "Le nombre d'invités est trop important";
  console.assert(guestsNumber <= peakNumber , {guestsNumber , peakNumber , message});

    // 3)
    console.log(guestsNumber);

```