

JavaScript Graphics

Table des matières

I. Contexte	3
II. Présentation de Chart.js	4
A. Présentation de Chart.js	4
B. Exercice : Quiz	10
III. Présentation de Plotly.js	11
A. Présentation de Plotly.js	11
B. Exercice : Quiz	15
IV. Essentiel	16
V. Auto-évaluation	16
A. Exercice	16
B. Test	19
Solutions des exercices	20

I. Contexte

Durée : 1 h

Environnement de travail : ordinateur/tablette et Replit

Contexte

En tant que développeur web, il est important de maîtriser au moins un langage front end. Javascript est l'un deux. Votre expertise passe donc par la maîtrise d'un ensemble de composants, de bibliothèques comme React ou comme celles qui vont être présentées lors des chapitres suivants.

Rappelons qu'une bibliothèque est un ensemble d'outils qui permet aux développeurs web d'optimiser le temps de développement d'une application. Rappelons aussi que pour maîtriser une bibliothèque, il peut être intéressant pour vous, de vous rendre sur la documentation de celle-ci : en ce qui concerne Plotly et Chart.js, vous trouverez les liens de ces documentations dans la partie 4 de votre cours.

L'objectif de ce cours est donc de comprendre concrètement comment utiliser les différentes bibliothèques permettant de réaliser des graphiques dans des applications web. Nous évoquerons les modalités d'installation ainsi que leurs spécificités, ce qui les rapprochent et ce qui les séparent.

Dans le cadre de l'installation, il sera fait mention d'outils tels que NPM qui nécessitent d'installer Node.js. Vous trouverez le lien d'installation dans la dernière rubrique de ce cours.

Enfin, au titre des prérequis pour une bonne compréhension de ce cours, dans la mesure où il sera fait référence à des éléments tels que NPM, il est utile de disposer de bonnes connaissances sur Javascript.

[cf. 1_ Introduction à ChartJs _ Plotly Js.mp3]

Attention

Pour avoir accès au code et à l'IDE intégré de cette leçon, vous devez :

- 1) Vous connecter à votre compte sur <https://replit.com/> (ou créer gratuitement votre compte)
- 2) Rejoindre la Team Code Studi du module via ce lien : <https://replit.com/teams/join/mmurvlgippxuasordloklllbqskoim-programmer-avec-javascript>

Une fois ces étapes effectuées, nous vous conseillons de rafraîchir votre navigateur si le code ne s'affiche pas.

En cas de problème, redémarrez votre navigateur et vérifiez que vous avez bien accepté les cookies de connexion nécessaires avant de recommencer la procédure.

Pour accéder au code dans votre cours, cliquez sur le nom du lien Replit dans la fenêtre. Par exemple :



II. Présentation de Chart.js

A. Présentation de Chart.js

Dans l'environnement Javascript, Chart.js et Plotly.js sont des librairies très populaires dont l'objet est de permettre la mise en place de graphique. Le rendu de ces graphiques peut s'effectuer en front ou en back.

Régulièrement, vous serez amené à faire un choix dans l'utilisation des librairies que vous allez utiliser. Ces choix vous amèneront à vous interroger sur :

- Le spectre de la librairie
- Le poids de la librairie et la maintenabilité de votre code

Le choix d'une librairie se fait en fonction de critères spécifiques, notamment :

- Sa popularité
- La date de sa dernière mise à jour
- Son nombre de téléchargement
- La compatibilité de la librairie dans l'environnement

Au regard de ces critères précis et factuels se dégagent 2 grandes librairies : Chart.js et Plotly.js. La connaissance d'au moins une de ces librairies peut s'avérer nécessaire dans un certain nombre de situations. Ce cours va vous présenter Chart.js.

Pourquoi l'utiliser ?

Vous serez confronté dans votre carrière à l'obligation d'enrichir vos pages web avec des graphiques, des éléments dynamiques, etc. C'est donc le moment de voir le fonctionnement et les avantages d'une librairie.

En ce qui concerne Chart.js, elle vous permettra d'assurer le rendu de vos graphiques en évitant des heures de codes et donc cela vous permettra d'optimiser votre temps de production. Comme cela sera détaillé plus tard, il est temps de faire un point sur votre environnement de travail.

Son environnement

Chart.js est une librairie Javascript qui est compatible avec l'ensemble des frameworks modernes comme React, Vue, Svelte, etc.

Mais aussi dans l'environnement Chart.js nous retrouvons des éléments HTML avec la balise `<canvas>` `</canvas>`. Nous reviendrons sur la balise Canvas ultérieurement.

C'est donc un environnement essentiellement front end que vous aurez à mettre en place.

Avant de procéder à l'installation de la librairie il est important que vous ayez l'habitude de naviguer dans la documentation pour approfondir et continuer l'apprentissage de cette librairie.

Le document est disponible sur le lien suivant : [chartjs.org](https://www.chartjs.org/docs/latest/)¹

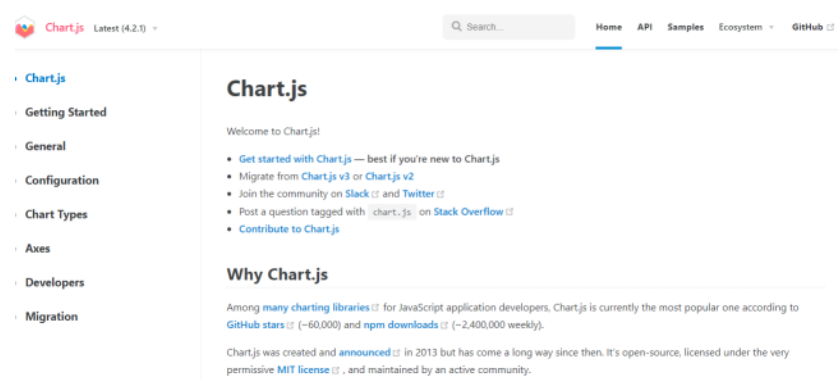
Tout d'abord à gauche de votre écran, vous vous apercevez qu'il existe une colonne dans laquelle figure un certain nombre d'onglets mais pour cette introduction nous nous bornerons aux onglets suivants :

- Getting Started
- Configuration
- Charts Types

¹ <https://www.chartjs.org/docs/latest/>

L'onglet Getting Started est celui qui vous permettra de commencer votre projet et d'aller chercher les cdn et autres éléments nécessaires à l'installation. L'onglet Configuration sera plus important car il contient toutes les options dont vous aurez besoin, affichage du nom du graphique, etc. Enfin le dernier onglet est l'onglet qui vous permettra de retrouver l'ensemble des types de graphiques qui sont possibles.

Une attention toute particulière doit être portée à la version que vous vous apprêtez à télécharger, c'est pour cela que si vous observez la documentation vous pourrez constater qu'il existe un onglet déroulant reprenant la dernière version mais aussi les versions précédentes.



Page d'accueil du site
Chart.js

Source : [chartjs.org](https://www.chartjs.org/)¹

Méthode

Il est possible d'installer Chart.js de différentes manières. Vous verrez dans les chapitres suivants comment effectuer cette installation.

Tout d'abord commençons par ce qui fait le quotidien des développeurs Javascript NPM. Comme vous l'avez appris, NPM est le gestionnaire de paquets pour Javascript.

Pour commencer l'installation, il est nécessaire de créer notre premier fichier HTML, pour cela vous devez ouvrir votre IDE et créer un fichier HTML.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Chart Js Project</title>
5 </head>
6
7 <body>
8 The content of the document.....
9 </body>
10
11 </html>
```

Dans la partie suivante, nous allons procéder à l'installation concrète de la librairie via NPM mais il sera nécessaire de vous assurer que Node.js est bien installé. Pour cela, vous devrez ouvrir votre invite de commande et taper `node -v..` Normalement cela devrait vous afficher la version installée sur votre ordinateur.

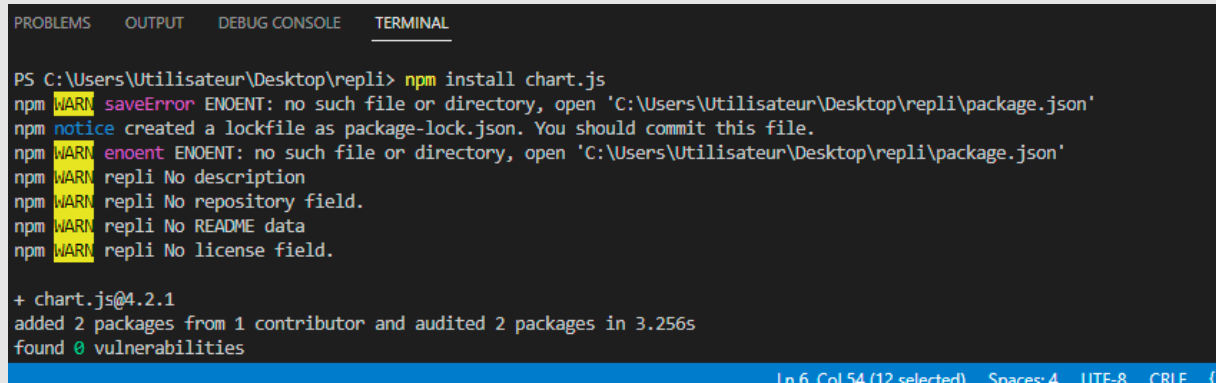
¹ <https://www.chartjs.org/>

Méthode Installer Chart.js avec NPM

Cette étape réalisée, il est maintenant impératif d'ouvrir votre terminal et de taper la commande suivante :

```
1 npm install chart.js
```

Vous devrez obtenir, si tout s'est bien passé, le visuel suivant :



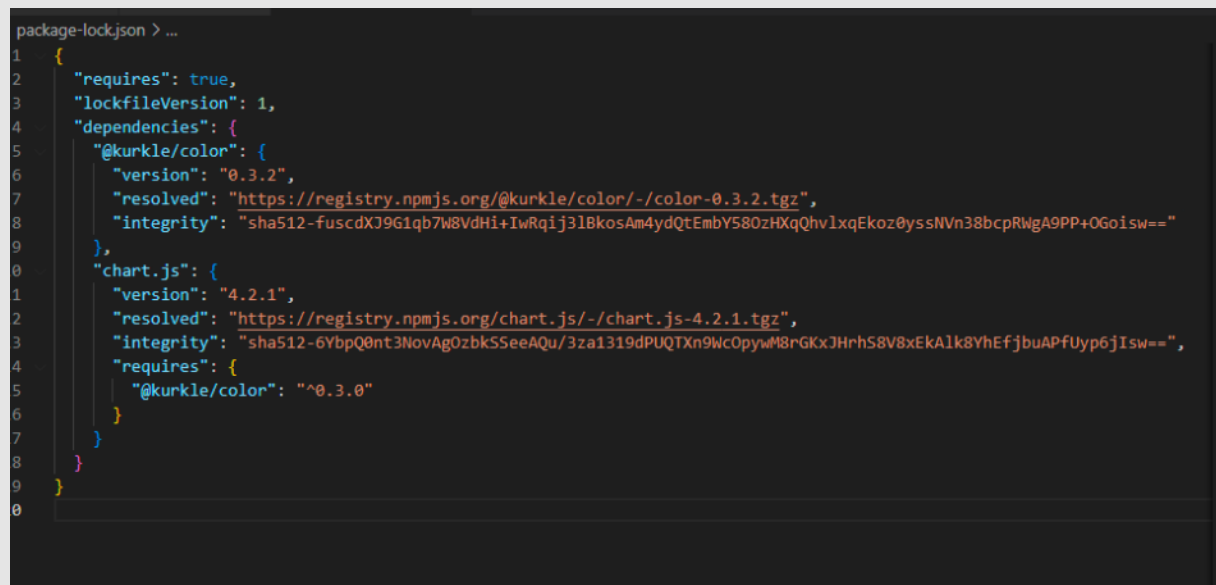
```
PS C:\Users\Utilisateur\Desktop\repli> npm install chart.js
npm WARN saveError ENOENT: no such file or directory, open 'C:\Users\Utilisateur\Desktop\repli\package.json'
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN enoent ENOENT: no such file or directory, open 'C:\Users\Utilisateur\Desktop\repli\package.json'
npm WARN repli No description
npm WARN repli No repository field.
npm WARN repli No README data
npm WARN repli No license field.

+ chart.js@4.2.1
added 2 packages from 1 contributor and audited 2 packages in 3.256s
found 0 vulnerabilities
```

Comme vous le voyez sur l'image ci-dessus, Chart.js est installé et 2 packages ont bien été installés. Vous constaterez d'ailleurs que dans l'arborescence de votre dossier, il existe maintenant un fichier node.

Méthode

Cette installation a d'ailleurs généré un fichier Json dans l'arborescence de votre dossier. Si vous l'ouvrez, vous aurez ce qui suit :



```
package-lock.json > ...
1 {
2   "requires": true,
3   "lockfileVersion": 1,
4   "dependencies": {
5     "@kurkle/color": {
6       "version": "0.3.2",
7       "resolved": "https://registry.npmjs.org/@kurkle/color/-/color-0.3.2.tgz",
8       "integrity": "sha512-fuscXJ9G1qb7W8VdHi+IwRqij3lBkosAm4ydQtEmbY580zHXqQhvlxqEkoZ0ySSNVn38bcpRWGA9PP+Ogoisw==",
9     },
10    "chart.js": {
11      "version": "4.2.1",
12      "resolved": "https://registry.npmjs.org/chart.js/-/chart.js-4.2.1.tgz",
13      "integrity": "sha512-6YbpQ0nt3NovAg0zbkSSeeAQu/3za1319dPUQTXn9WcOpywM8rGKxJHrhS8V8xEkAlk8YhEfjbuAPfUyp6jIsw==",
14      "requires": {
15        "@kurkle/color": "^0.3.0"
16      }
17    }
18  }
19 }
```

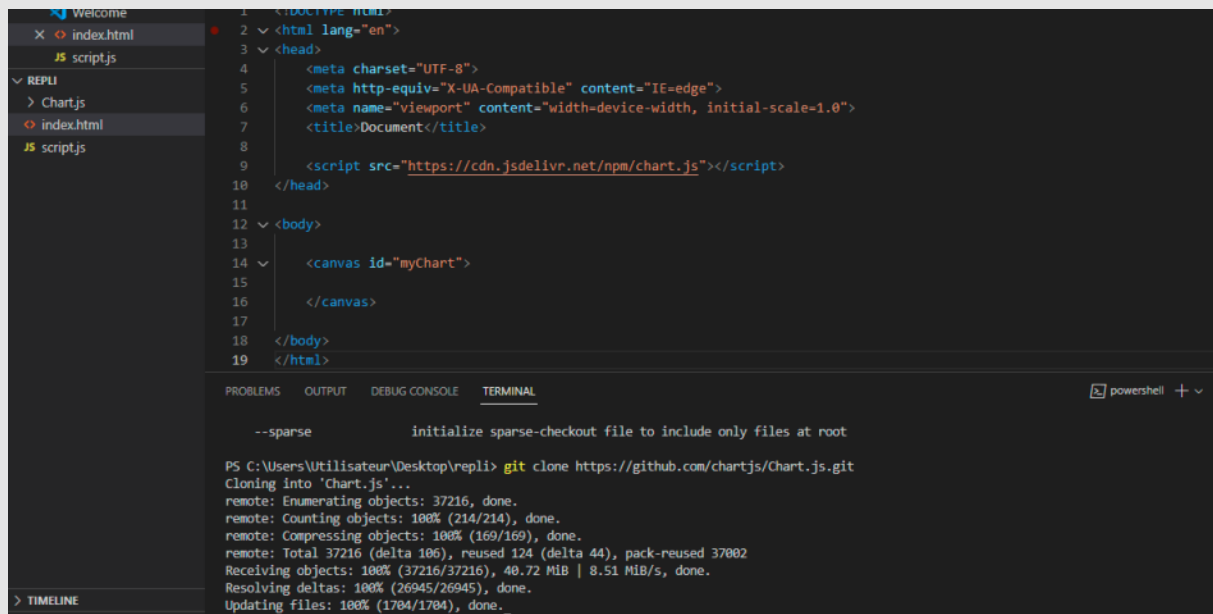
Méthode Installer Chart.js via un CDN

Cette installation ne présente aucune difficulté, car elle se limite à un simple copier-coller de liens figurant dans les bibliothèques suivantes. Ces liens doivent être copiés entre la balise **<head>** **</head>** et en fin de page avant la fermeture de la balise **<body>**. Pour disposer du lien, il est impératif d'aller sur la documentation et d'ouvrir l'onglet Getting Started.

Au titre des avantages de ce type d'installation, est soulignée la simplicité de l'installation qui se résume à un simple copier-coller du lien. L'inconvénient majeur est le fait que cela n'est qu'un lien, donc en cas de changement d'adresse du lien, votre application ne fonctionnera plus. Cela peut être particulièrement embêtant.

Méthode Installer Chart.js via GitHub

L'avantage d'une telle installation est la sécurité que cela engendre contrairement à un simple CDN, la librairie étant mise dans votre App, il n'y a pas de risque qu'elle soit déplacée et qu'elle impacte directement le rendu de vos graphiques. Pour procéder à l'installation via GitHub, il n'est pas nécessaire de disposer d'un compte GitHub, mais bien entendu, la maîtrise d'un instrument de versioning est un requis pour tout futur développeur. Pour télécharger le code, rendez-vous sur le lien contenu dans la documentation. Pour réaliser cette démarche, ouvrez un terminal comme le démontre l'image ci-après. L'installation via GitHub se fait par l'utilisation de la commande : **git clone**.



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8
9   <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
10 </head>
11
12 <body>
13   <canvas id="myChart">
14
15   </canvas>
16
17 </body>
18 </html>
```

```
--sparse          initialize sparse-checkout file to include only files at root

PS C:\Users\Utilisateur\Desktop\repli> git clone https://github.com/chartjs/Chart.js.git
Cloning into 'Chart.js'...
remote: Enumerating objects: 37216, done.
remote: Counting objects: 100% (214/214), done.
remote: Compressing objects: 100% (169/169), done.
remote: Total 37216 (delta 106), reused 124 (delta 44), pack-reused 37002
Receiving objects: 100% (37216/37216), 40.72 MiB | 8.51 MiB/s, done.
Resolving deltas: 100% (26945/26945), done.
Updating files: 100% (1704/1704), done.
```

Complément

Il est important de noter que lorsque Chart.js est téléchargé directement depuis le repository GitHub, celui-ci ne contient plus toutes les dépendances. Il est donc nécessaire de procéder à l'ajout d'un script autonome, en l'occurrence il s'agit de **pnpm**.

Pour cela il est nécessaire d'ouvrir un terminal et de taper la commande suivante :

```
1 iwr https://get.pnpm.io/install.ps1 -useb | iex
```

Cette commande vous permettra d'installer Chart.js et installera l'ensemble des dépendances nécessaires au bon fonctionnement de la librairie.

Méthode

Tout graphique réalisé grâce à Chart.js débutera par un certain nombre de nécessités et nous imposera un certain formalisme. Tout d'abord il nous faudra implanter une div sur notre fichier HTML qui contiendra un ID, que vous pourrez nommer comme vous le souhaitez. Pour la suite de ce cours et pour en faciliter la suite, l'ID sera identifié sous « *myChart* », mais il n'y aucune obligation de conserver la dénomination de l'ID.

```
1 <body>
2 <canvas id="myChart" width="400" height="400"></canvas>
3 </body>
```

Comme vous le constatez, le départ du graphique se fait par la mise en place d'une balise **<canvas>** **</canvas>**. C'est une balise qui s'ouvre et qui se ferme, dédiée à l'environnement des graphiques. Prise seule, cette balise est une image et elle ne fournit aucune aide pour l'accessibilité, il est donc nécessaire de ne pas se servir seulement de cette balise pour répondre aux besoins d'accessibilité. L'élément canvas est un nouvel élément HTML qui est associé généralement à des scripts en Javascript. Il permet aux développeurs front de composer des graphiques et s'avère plus rapide au chargement.

Complément Le Javascript dans notre fichier HTML

Pour simplifier la démarche et la compréhension, nous allons mettre en place Chart.js avec un lien CDN.

```
1 <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
```

Le lien CDN est à mettre entre les balises **<head>****</head>**, il s'agit de la même démarche que celle qui consiste à utiliser le lien bootstrap. Le lien copié, il est maintenant nécessaire de mettre en place notre Javascript. Pour cela et pour simplifier le cours, nous utiliserons la balise script dans notre fichier HTML. Néanmoins il aurait été tout aussi possible d'injecter notre Javascript en créant un fichier dédié, appelé script.js. Sur notre fichier js figurera le code qui nous permettra de mettre en action.

```
1 <script>
2 let ctx = document.querySelector("#myChart");
3     new Chart(ctx, {
4         type: "pie",
5         data: {
6             labels: ['Paris', 'New-York', 'Madrid', 'Moscou', 'Berlin', 'Auckland'],
7             datasets: [{
8                 label: ' Nombre d'habitants',
9                 data: [12, 19, 7, 5, 2, 3],
10                backgroundColor: ['red', 'blue', 'yellow', 'green', 'purple', 'orange']
11            }]
12        },
13        options: {
14            plugins: {
15                title: {
16                    display: true,
17                    text: 'Mon premier graphique avec Chart Js'
18                },
19            },
20            legend: {
21                position: 'bottom'
22            }
23        }
24    }
25 });
26 </script>
```

[cf.]

Une lecture de notre code nous donnera les clés pour comprendre ce que nous venons de faire. Tout d'abord nous avons passé un **ID « myChart »** dans la balise canvas. Cet ID a été récupéré grâce à la méthode du document.querySelector. Nous avons ensuite utilisé new chart. Cette syntaxe est un impératif imposé par Chart.js.

Dans cette variable nous avons mis en place la balise **ctx** (context = getContext). Cette syntaxe est là aussi imposée par Chart.js. En effet, cette méthode nous est imposée par l'utilisation de la balise **<canvas>**.

Enfin, nous avons créé un objet dans lequel nous avons défini l'ensemble des données de notre graphique, son type, ses datas, etc.

Configuration des options

Vous le remarquerez en lisant le code, il y a un ensemble d'options qui sont possibles, il existe d'autres types de graphiques possibles avec cette librairie.

Les types de graphiques possibles

Sans citer tous les graphiques, il est néanmoins possible d'en dégager trois ou quatre types que nous rencontrons régulièrement :

- Le radar
- Le graphique en ligne
- Le pie
- Le graphique en bar

Qu'est-ce que Data et Datasets ?

Data et datasets sont 2 éléments incontournables de Chart.js. Data est avant tout un objet qui contiendra un ensemble de données telles que décrit dans l'exemple ci-après. Cet exemple vous montre ce que sont les datas dans chart.js et ce qu'il contient.

```
1 type: 'line',
2   data: {
3     labels: ['Paris ', 'London', 'New York', 'Singapore', 'Shanghai', 'Sydney'],
```

Dans cet objet sera inclus l'ensemble des variables qui feront l'objet d'un graphique.

Pour ce qui est de Datasets, il s'agit de transcrire une donnée qui est un tableau qui contiendra des objets tels que vous le voyez ci-dessous.

```
1 datasets: [{
2   label: 'Nombre de résident en million',
3   data: [12, 19, 3, 5, 2, 3],
4   borderWidth: 1
5 }]
```

Datasets contiendra donc les variables suivantes :

- Les labels qui sont des chaînes de caractères,
- Les data qui sont des strings,
- Et pour finir des éléments de css comme la taille et la couleur d'une ligne de point.

Les options possibles à paramétrer

Une option est un objet dans lequel nous plaçons d'autres objets comme le montre le code ci-dessous.

Une option particulièrement intéressante est **chartContainer** qui permettra d'assurer les aspects responsive de votre graphique.

```
1 <div class="chart-container">
2   <canvas id="myChart"></canvas>
3 </div>
```

L'inconvénient du genre de structure proposé par Chart.js se situe au niveau de la maintenabilité du code, dès qu'un graphique s'avère plus complexe. En effet, chaque option se déclare dans un objet. De fait, si le graphique est complexe, le code peut vite devenir compliqué à lire. Comme le montre le code ci-après.

```

1 options: {
2     title: {
3         display: true,
4         text: 'Mon premier graphique avec Chart Js'
5     },
6     legend: {
7         position: 'bottom'
8     }
9 }
10 })

```

Remarque

Les graphiques développés via Chart.js ne commencent pas au zéro sur les axes des X et Y, il peut donc être important de paramétrer le début de la numération en paramétrant le y comme débutant à 0.

B. Exercice : Quiz

[solution n°1 p.21]

Question 1

Pour assurer les aspects responsifs sur Chart.js, vous pouvez :

- ☐ Utiliser la classe « *ChartContainer* »
- ☐ Utiliser le système des options avec un display True ?
- ☐ Les deux propositions précédentes sont nécessaires
- ☐ Rien du tout, Chart.js est nativement responsif

Question 2

Quelle est la syntaxe pour la balise canvas ?

- ☐ <canvas/>
- ☐ <canvas></canvas>
- ☐ canvas

Question 3

Quelle est la syntaxe pour configurer le type de graphique ?

- ☐

```
const cfg = {
  type: 'bar',
  data: {
    datasets: [{
      data: [20, 10],
    }],
    labels: ['a', 'b']
  }
}
```

- ☐

```
const cfg = {  
  type: 'bar',  
  data: [{  
    datasets: [{  
      data: [20, 10],  
    }],  
    labels: ['a', 'b']  
  }  
}
```
- ☐

```
const cfg = {  
  type: 'bar';  
  data: {  
    datasets: [{  
      data: [20, 10]  
    }];  
    labels: ['a', 'b']  
  }  
}
```

Question 4

Lors de l'installation de la librairie Chart.js via NPM, quelle commande devez-vous entrer ?

- ☐ Npm-install chart.js
- ☐ Npm_install chart.j
- ☐ npm install chart.js
- ☐ npm install -i -chartjs

Question 5

Le clonage du repository de GitHub permet de faire fonctionner la librairie avec toutes ses dépendances.

- ☐ Vrai
- ☐ Faux

III. Présentation de Plotly.js

A. Présentation de Plotly.js

[cf. 4_Présentation de Plotly.mp3]

Autre librairie spécialisée dans la création de graphiques : Plotly.js.

Installer Plotly.js

Comme Chart.js, Plotly.js est une librairie qui viendra vous aider à enrichir vos pages de graphiques responsives et dynamiques. Cette librairie est aussi open source. L'une des grandes différences se situera dans les modalités d'installation.

De plus, elle a la caractéristique d'être disponible pour de nombreux langages de programmation.

Il y a bien entendu la possibilité d'installer Plotly.js via NPM. Pour cela il est nécessaire d'ouvrir votre terminal et de taper la commande suivante :

```
1 npm i plotly.js
```

Elle s'installe aussi via le cdn, ainsi que via le repos de GitHub. Ces démarches ayant été vues en amont avec l'introduction de Chart.js, il est plus important de voir la dernière façon d'installer cette librairie, via webpack.

Conseil Webpack pour l'installation de Plotly.js dans un projet HTML

Dans le cadre de l'installation de Plotly.js, un élément de différenciation existe avec Chart.js : il est possible d'installer Plotly.js via webpack.

La procédure sera donc un peu plus complexe si vous souhaitez utiliser webpack pour installer cette librairie, néanmoins, une fois maîtrisée, elle est assez simple à mettre en place.

Méthode

L'installation via webpack nécessite un certain nombre de démarches, la première est l'installation de **ify-loader** en utilisant npm.

```
1 npm install --save ify-loader
```

L'installation de cette dépendance devrait vous donner le visuel suivant :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

npm WARN repli No README data
npm WARN repli No license field.

+ ify-loader@1.1.0
added 47 packages from 32 contributors and audited 47 packages in 72.526s

5 packages are looking for funding
  run 'npm fund' for details

found 0 vulnerabilities
```

Ify-loader téléchargé, il sera nécessaire d'installer webpack dans votre projet :

```
1 npm install --save-dev webpack
```

Si l'installation s'est bien déroulée, vous devriez avoir le visuel suivant :

```
S C:\Users\Utilisateur\Desktop\repli> npm install --save-dev webpack
npm WARN saveError ENOENT: no such file or directory, open 'C:\Users\Utilisateur\Desktop\repli\package.json'
npm WARN enoent ENOENT: no such file or directory, open 'C:\Users\Utilisateur\Desktop\repli\package.json'
npm WARN repli No description
npm WARN repli No repository field.
npm WARN repli No README data
npm WARN repli No license field.

webpack@5.76.2
added 75 packages from 123 contributors and audited 152 packages in 52.341s
```

```
1 npm install --save-dev webpack
```

Enfin, vous devrez configurer webpack en ajoutant un script à votre webpack.config.json. Il vous sera donc nécessaire d'ajouter le code ci-dessous.

```
1 ...
2     module: {
3         rules: [
4             {
5                 test: /\.js$/,
6                 loader: 'ify-loader'
7             }
8         ]
9     }
```

```

8      ]
9      },
10     ...

```

Les impératifs : une balise HTML pour le départ

Comme pour Chart.js, la première démarche est la création d'un fichier HTML. Entre les 2 balises **<body></body>**, il est nécessaire de mettre une balise **<div></div>**.

Méthode

```
1 <div id="tester" style="width:600px;height:250px;"></div>
```

Cette **<div>** installée, il sera nécessaire de créer notre fichier Javascript sur lequel l'ensemble des données sera injecté.

```

1 <script>
2   let graphDiv = document.getElementById('tester')
3
4   let data = [{
5     x: [1999, 2000, 2001, 2002],
6     y: [10, 15, 13, 17],
7     type: 'scatter'
8   }];
9
10  let layout = {
11    title: 'Sales Growth',
12    xaxis: {
13      title: 'Year',
14      showgrid: false,
15      zeroline: false
16    },
17    yaxis: {
18      title: 'Percent',
19      showline: false
20    }
21  };
22  Plotly.newPlot(graphDiv, data, layout);
23  let dataRetrievedLater = graphDiv.data;
24  let layoutRetrievedLater = graphDiv.layout;
25
26 </script>

```

[cf.]

Conseil

Plotly.newPlot(graphDiv, data, layout); cette classe est un impératif dans Plotly.js : elle s'introduit toujours dans une **<div>**. Elle contient l'ensemble des éléments du graphiques, comme les données, le titre, l'axe des x et des y, etc.

La variable layout est incluse dans un objet, l'objet layout servira à stocker des variables comme le titre, le départ à zéro du graphique, etc.

Comment configurer les options du graphique

Il est possible de paramétrer l'ensemble du graphique pour obtenir un rendu correspondant à nos attentes. La librairie comptant plus de 40 types de graphiques, il est plus intéressant de s'attarder sur la méthode pour ajouter une ou plusieurs options. Pour cela, il sera nécessaire de déclarer une variable qui dans un objet contiendra les options voulues, telle que le nom, le responsive, etc. Observez le code ci-dessous, il a pour objectif de permettre un zoom et un scroll.

Méthode

```
1 <script>
2   let graphDiv = document.getElementById('tester')
3
4   let data = [{
5     x: [1999, 2000, 2001, 2002],
6     y: [10, 15, 13, 17],
7     type: 'scatter'
8   }];
9
10  let layout = {
11    title: 'Sales Growth',
12    xaxis: {
13      title: 'Year',
14      showgrid: false,
15      zeroline: false
16    },
17    yaxis: {
18      title: 'Percent',
19      showline: false
20    }
21  };
22  Plotly.newPlot(graphDiv, data, layout, );
23  let dataRetrievedLater = graphDiv.data;
24  let layoutRetrievedLater = graphDiv.layout;
25
26 </script>
```

[cf.]

La mise en place des options se fait dans une variable « *Layout* », dans l'exemple au-dessus, il s'agit de l'option scroll et zoom mais la démarche serait la même avec des aspects responsifs.

Exemple

Voici un exemple de projet avec Plotly.js.

```
1 <!DOCTYPE html>
2 <html lang="fr">
3
4 <head>
5 <!-- cdn Plotly.js entre les balises head -->
6 <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
7 </head>
8
9 <!-- Plots s'installe dans une div.-->
10 <div id="tester" style="width:90%;height:250px;"></div>
11 <script src="script.js"></script>
12 </body>
13 </html>
```

```
1 <script>
2 let tester = document.getElementById('tester');
3 Plotly.newPlot( tester, [
4   {
5     x: [1, 2, 3, 4, 5],
6     y: [1, 2, 4, 8, 16]
7   }
8 ],
9   {
10    margin: { t: 0 }
11  });
12 </script>
13 Lien Replit :
```

[cf.]

L'utilisation de ces morceaux de codes vous permettront de réaliser votre premier graphique en ligne avec Plotly.js.

B. Exercice : Quiz

[solution n°2 p.22]

Question 1

Parmi les propositions suivantes, laquelle est la syntaxe utilisée par Plotly.js lors de la mise en place d'un projet ?

- ☐ Plotly.newPlot
- ☐ Plotlynew.Plot
- ☐ PlotlynewPlot

Question 2

Dans le cadre d'un projet faisant appel à Plotly.js, quelle balise devez-vous utiliser lors de l'initiation d'un projet Plotly ?

- ☐ Une div
- ☐ Une div avec la balise canvas
- ☐ Une div avec un IDE

Question 3

Quelle est la syntaxe pour configurer le type de graphique ?

- ☐ `let layout = {
 title: 'Scroll and Zoom',
 showlegend: false };`
- ☐ `let layout = [
 title: 'Scroll and Zoom',
 showlegend: false];`
- ☐ `let layout = [{
 title: 'Scroll and Zoom',
 showlegend: false }];`

Question 4

Lors de l'installation de la librairie Plotly.js via NPM, quelle commande devez-vous entrer ?

- ☐ Npm-install plotly.js
- ☐ npm i plotly.js
- ☐ npm install plotly.js-dist
- ☐ npm install plotly-dist.js

Question 5

Le clonage du repository de GitHub permet de faire fonctionner la librairie avec toutes ses dépendances.

- ☐ Vrai
- ☐ Faux

IV. Essentiel

Il est à retenir un certain nombre d'éléments de ces deux librairies. Elles sont toutes deux en open source et donc complètement gratuites. Elles sont aussi toutes deux totalement customisables grâce au système des options.

Elles s'installent de la même façon (NPM, GitHub, CDN), à l'exception près que Plotly.js peut s'installer via webpack. Elles se mettent dans une balise de type div dans un fichier HTML.

De plus, toutes les deux nécessitent la mise en place d'un ID qui est récupéré par `document.querySelector` ou `document.getElementById`.

Ces librairies sont totalement customisables et offrent l'opportunité de mettre en place rapidement et très simplement des graphiques que cela soit la partie front end ou la partie back end. Elles prennent aussi en compte les aspects responsives nécessaires compte tenu de l'évolution des appareils connectés.

Ces librairies prennent en considération l'expérience utilisateur et notamment l'accessibilité pour les personnes souffrant d'une déficience visuelle. Chart.js ajoute un rôle `` dans le canvas déclaratif du fichier HTML.

V. Auto-évaluation

A. Exercice

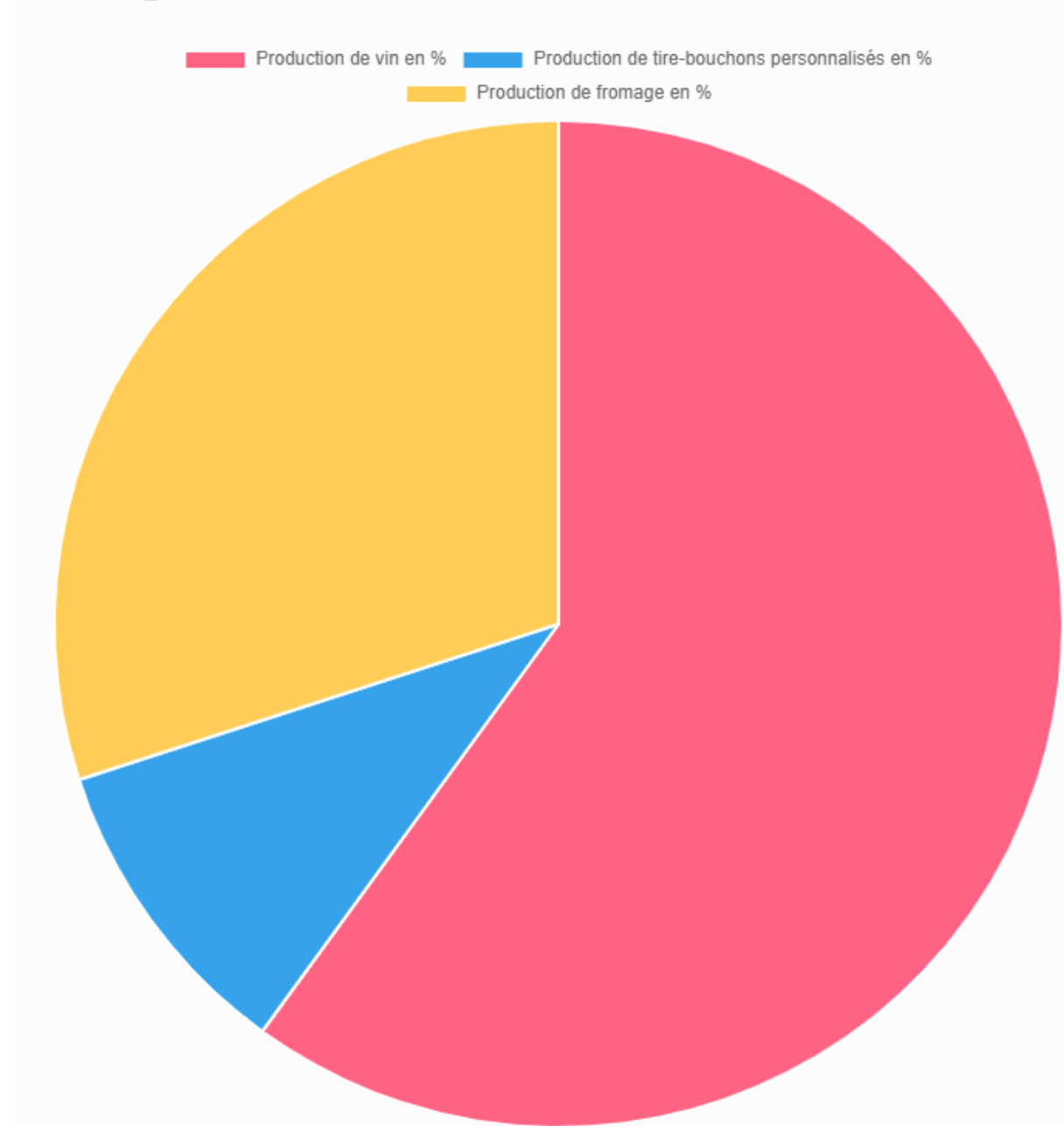
Vous êtes reçu par une agence web dans le cadre d'un entretien technique. Votre futur manager vous confie votre première mission. Il vous demande de réaliser deux graphiques avec deux librairies spécialisées : Chart.js et Plotly.js. Avec Chart.js, il vous demande de créer un graphique de type pie.

Question 1

[solution n°3 p.23]

Dans un premier temps, votre patron vous demande de créer un graphique de type pie avec Chart.js pour une société proposant des dégustations de vins et de fromages de la région. Ceux-ci souhaitent faire apparaître leur pourcentage de production de vin, de tire-bouchons personnalisés et de fromages.

Voici ce que vous devriez obtenir :

Notre production :

Voici les données à renseigner :

- 60 % de vin
- 10 % de tire-bouchons
- 30 % de fromage

Enfin, voici le code HTML :

```
1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4   <!-- Mis en place du cdn -->
5   <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
6 </head>
7 <body>
8   <h1>Notre production :</h1>
9   <!-- Chart s'installe entre des éléments de div.-->
10  <div>
11    <canvas id="myChart"></canvas>
12  </div>
13  <script src="script.js"></script>
14 </body>
15 </html>
16 </html>
```

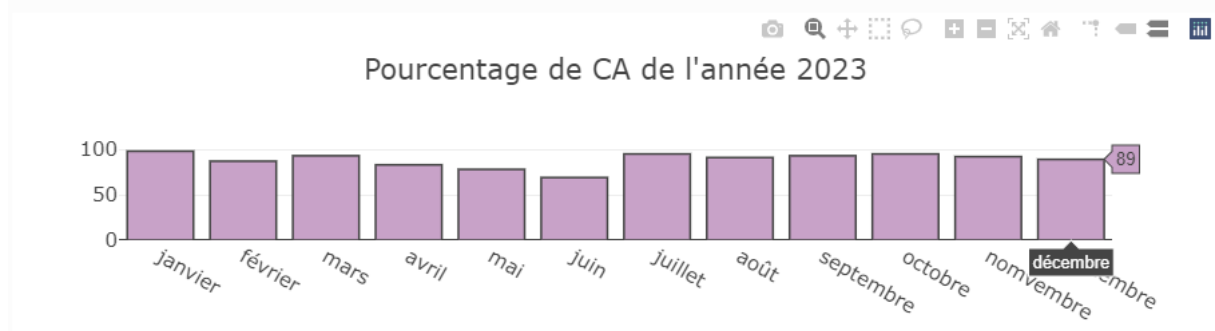
Question 2

[solution n°4 p.24]

Dans un second temps, votre patron vous demande de créer un graphique de type bar tout en prenant en compte l'aspect responsive dans le cadre du projet d'un client. Ce client souhaite indiquer sous la forme d'un diagramme le pourcentage de chiffre d'affaires atteint par l'entreprise Volea en fonction de son chiffre d'affaires prévisionnel.

Voici ce que vous devriez obtenir :

Les chiffres de Volea en 2022



Voici les données à renseigner :

- Janvier : 98 %
- Février : 87 %
- Mars : 93 %
- Avril : 83 %
- Mai : 78 %
- Juin : 69 %
- Juillet : 95 %
- Août : 91 %
- Septembre : 93 %
- Octobre : 95 %
- Novembre : 92 %
- Décembre : 89 %

Enfin, voici le code HTML :

```
1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4   <!-- Mis en place du cdn Plotly Js -->
5   <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
6 </head>
7 <h1>Les chiffres de Volea en 2022</h1>
8 <!-- Plots s'installe entre des éléments de div.
9 -->
10 <div id="tester" style="width:90%;height:250px;"></div>
11 <script src="script.js"></script>
12 </body>
13 </html>
```

B. Test

Exercice 1 : Quiz

[solution n°5 p.24]

Question 1

Il existe une différence majeure entre Plotly.js et Chart.js dans les méthodes d'installation.

- ☐ Vrai
- ☐ Faux

Question 2

Il n'est pas possible d'utiliser Plotly.js avec des frameworks modernes tels que Vue.js et React.

- ☐ Vrai
- ☐ Faux

Question 3

Dans le cas d'un projet Chart, la notion d'accessibilité peut poser problème, il est recommandé de doter la balise canvas...

- ☐ D'un width
- ☐ D'un height
- ☐ D'un width, d'un height, d'un rôle avec une balise img

Question 4

Dans quel élément Javascript se déclare le type de graphique ?

- ☐ Un tableau
- ☐ Un objet
- ☐ Un tableau contenant un objet

Question 5


C'est un inconvénient de déclarer un rôle dans l'insertion de la balise Chart.

- ☐ Vrai
- ☐ Faux

Solutions des exercices


Exercice p. 10 Solution n°1**Question 1**

Pour assurer les aspects responsifs sur Chart.js, vous pouvez :

- ☒ Utiliser la classe « *ChartContainer* »
- ☐ Utiliser le système des options avec un display True ?
- ☐ Les deux propositions précédentes sont nécessaires
- ☐ Rien du tout, Chart.js est nativement responsif
-  Dans le cadre d'un projet Chart.js, il est nécessaire d'utiliser la classe « *ChartContainer* » qui permettra de rendre le graphique responsive.

Question 2

Quelle est la syntaxe pour la balise canvas ?

- ☐ <canvas/>
- ☒ <canvas></canvas>
- ☐ canvas
-  Canvas est une balise HTML fermante de ce fait, la seule solution valide est <canvas></canvas>.

Question 3

Quelle est la syntaxe pour configurer le type de graphique ?

- ☒


```
const cfg = {
  type: 'bar',
  data: {
    datasets: [{
      data: [20, 10],
    }],
    labels: ['a', 'b']
  }
}
```
- ☐

```
const cfg = {
  type: 'bar',
  data: [{
    datasets: [{
      data: [20, 10],
    }],
    labels: ['a', 'b']
  }
}
```

```

○ const cfg = {
  type: 'bar';
  data: {
    datasets: [{
      data: [20, 10]
    }];
    labels: ['a', 'b']
  }
}


```

 Data est un objet, ce qui le diffère de datasets, qui est un tableau qui contient des objets.

Question 4

Lors de l'installation de la librairie Chart.js via NPM, quelle commande devez-vous entrer ?


- ☐ Npm-install chart.js
- ☐ Npm_install chart.j
- ☒ npm install chart.js
- ☐ npm install -i -chartjs

 La bonne réponse est npm install chart.js. Cette installation se fait en ligne de commande via le terminal de votre IDE. Cette commande est la seule permettant l'installation de Chart.js via NPM, elle est donc à connaître.

Question 5

Le clonage du repository de GitHub permet de faire fonctionner la librairie avec toutes ses dépendances.

- ☐ Vrai
- ☒ Faux

 Le simple clonage du repository de Chart.js ne permettra pas le fonctionnement de la librairie avec toutes ses dépendances. Il faudra installer un script autonome usuellement pnpm.

Exercice p. 15 Solution n°2

Question 1


Parmi les propositions suivantes, laquelle est la syntaxe utilisée par Plotly.js lors de la mise en place d'un projet ?

- ☒ Plotly.newPlot
- ☐ Plotlynew.Plot
- ☐ PlotlynewPlot

 C'est la syntaxe utilisée par Plotly.js pour créer un nouveau graphique dans une page HTML.

Question 2

Dans le cadre d'un projet faisant appel à Plotly.js, quelle balise devez-vous utiliser lors de l'initiation d'un projet Plotly ?

- ☐ Une div
- ☐ Une div avec la balise canvas
- ☒ Une div avec un IDE
-  Canvas est une balise HTML, malgré cela, elle n'est pas utilisée par Plotly.js qui préfère installer une simple div.


Question 3

Quelle est la syntaxe pour configurer le type de graphique ?

- ☒


```
let layout = {  
  title: 'Scroll and Zoom',  
  showlegend: false };
```
- ☐

```
let layout = [  
  title: 'Scroll and Zoom',  
  showlegend: false ];
```
- ☐

```
let layout = [{  
  title: 'Scroll and Zoom',  
  showlegend: false }];
```
-  La variable layout s'inscrit dans un objet.


Question 4

Lors de l'installation de la librairie Plotly.js s via NPM, quelle commande devez-vous entrer ?

- ☐ Npm-install plotly.js
- ☒ npm i plotly.js
- ☐ npm install plotly.js-dist
- ☐ npm install plotly-dist.js
-  La bonne réponse est npm i plotly.js. Cette installation se fera en ligne de commande via votre IDE. Là aussi vous pouvez retrouver la commande sur le lien suivant : plotly.js¹

Question 5

Le clonage du repository de GitHub permet de faire fonctionner la librairie avec toutes ses dépendances.

- ☒ Vrai
- ☐ Faux
-  Plotly.js s'installe aussi via GitHub sans difficulté particulière. Cette installation se fait en ligne de commande via votre IDE. Il sera nécessaire de taper la commande suivante, `git clone plotly/plotly.js`². Cette installation vous permettra de télécharger la librairie dans votre projet.

¹ <https://www.npmjs.com/package/plotly.js>

² <https://github.com/plotly/plotly.js.git>

```

1 <script>
2 let pieChart = document.querySelector("#myChart");
3     new Chart(pieChart, {
4         type: "pie",
5         data: {
6             labels: ['Production de vin en %', 'Production de tire-bouchons personnalisés
7 en %', 'Production de fromage en %'],
8             datasets: [{
9                 data: [60, 10, 30],
10                backgroundColor: [
11                    'rgb(255, 99, 132)',
12                    'rgb(54, 162, 235)',
13                    'rgb(255, 205, 86)'
14                ],
15                hoverOffset: 4
16            }]
17        });
18 </script>

```

[cf.]

p. 18 Solution n°4

```

1 <script>
2 let tester = document.getElementById('tester')
3 let data = [{
4     type: 'bar',
5     x: ['janvier', 'février', 'mars', 'avril', 'mai', 'juin', 'juillet', 'août', 'septembre',
6 'octobre', 'novembre', 'décembre'],
7     y: [98, 87, 93, 83, 78, 69, 95, 91, 93, 95, 92, 89],
8     marker: {
9         color: '#C8A2C8',
10        line: {
11            width: 1.5
12        }
13    }
14 }];
15 let layout = {
16     title: 'Pourcentage de CA de l\'année 2022',
17     font: {size: 15}
18 };
19 let config = {responsive: true}
20 Plotly.newPlot(tester, data, layout, config );
21 </script>

```

[cf.]

Exercice p. 19 Solution n°5

Question 1

Il existe une différence majeure entre Plotly.js et Chart.js dans les méthodes d'installation.

- ☒ Vrai
- ☐ Faux

Q Il est possible d'installer Plotlys.js via webpack alors que cela n'est pas possible avec Chart.js.

Question 2

Il n'est pas possible d'utiliser Plotly.js avec des frameworks modernes tels que Vue.js et React.

☐ Vrai

☒ Faux

Q Plotly et Chart sont compatibles avec les framework modernes. Pour Plotly, il faudra créer un composant.

Question 3

Dans le cas d'un projet Chart, la notion d'accessibilité peut poser problème, il est recommandé de doter la balise canvas...

☐ D'un width

☐ D'un height

☒ D'un width, d'un height, d'un rôle avec une balise img

Q Dans le cadre d'un projet sous Chart.js, l'accessibilité peut poser des soucis, donc pour un meilleur référencement, une bonne pratique est de doter la balise canvas d'un width, d'un height, d'un rôle avec une balise img.

Question 4

Dans quel élément Javascript se déclare le type de graphique ?

☐ Un tableau

☒ Un objet

☐ Un tableau contenant un objet

Q Il se déclare dans un objet qui pourra contenir un certain nombres de valeurs, comme un tableau qui lui-même contiendra les valeurs attribuées à x et y.

Question 5

C'est un inconvénient de déclarer un rôle dans l'insertion de la balise Chart.

☐ Vrai

☒ Faux

Q L'accessibilité pour les personnes non voyante ou ayant une déficience est un élément important pour le référencement d'un site. C'est pourquoi il est judicieux d'insérer un rôle dans le canvas.