

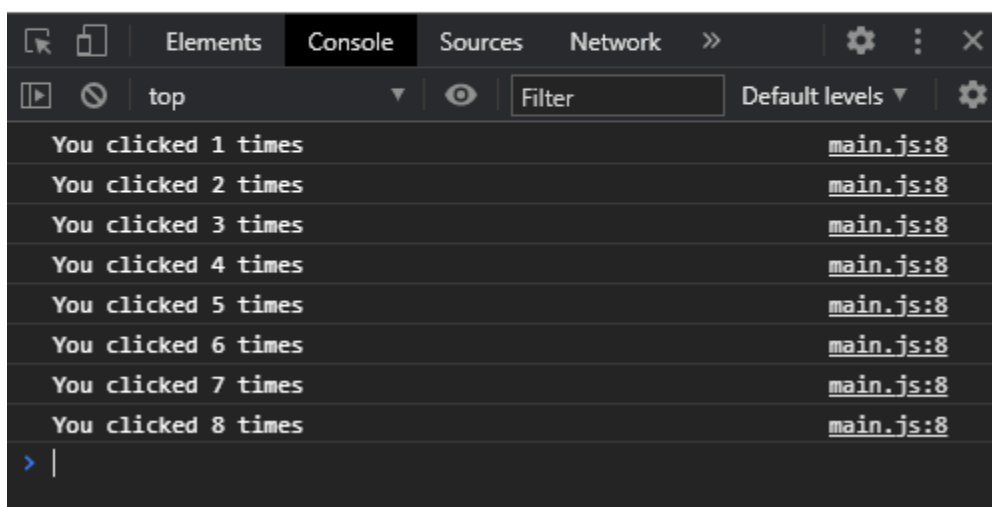
Exercices d'application en Javascript

[1] Cliquez sur le bouton pour incrémenter

Objectif : À chaque clic de l'utilisateur sur le bouton « *Increment!* », le contenu de la div « *cntClic* » s'incrémente de 1.

Increment !

You clicked 8 times



Cadre :

Créez un fichier « *index.html* » avec le contenu suivant :

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Click on the button to increment!</title>
  </head>
  <body>
    <button id="myBtn">Increment!</button>
    <p id="cntClic">Click on the button to increment!</p>
    <script src="main.js"></script>
```

```
</body>
</html>
```

⚠ Vous n'avez plus à toucher au HTML, tout ce que vous coderez pour réussir cet exercice se passe dans le fichier JavaScript.

Créez le fichier « *main.js* », déclarez une variable pour le compteur, et utilisez des « *getElementById* » pour récupérer les deux ids et les stocker dans des constantes :

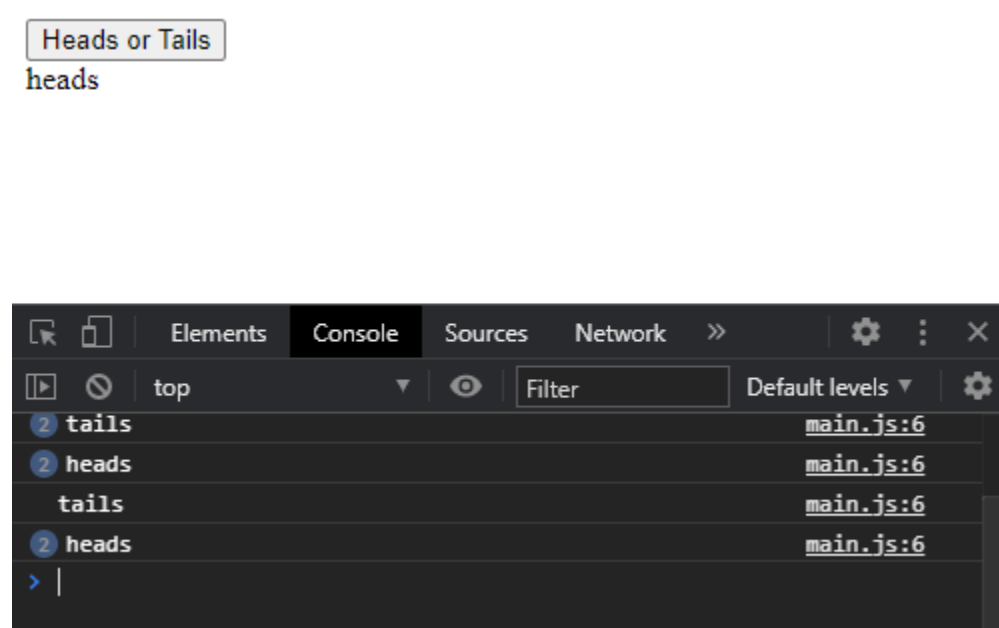
```
const myBtn = document.getElementById('myBtn') ;
const cntClic = document.getElementById('cntClic') ;

let nbClic = 0 ;
```

Consigne : Utilisez un « *addEventListener* » sur « *myBtn* » pour développer la fonction. Ajoutez un « *console.log* » sur « *nbClic* » dans la fonction afin de pouvoir contrôler le résultat dans la console.

[2] Pile ou Face

Objectif : À chaque clic de l'utilisateur sur le bouton « *Heads or Tails* », le contenu de la div « *result* » affiche aléatoirement soit « *Heads* », soit « *Tails* ».



Créez un fichier « *index.html* » avec le contenu suivant :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>
    <button id="btn">Pile ou Face</button>
    <div id="result">Cliquez ici !</div>
    <script src="main.js"></script>
  </body>
</html>
```

Créez ensuite le fichier « *main.js* », déclarez des variables « *myBtn* » et « *result* » pour récupérer les deux ids avec des « *getElementById* » :

```
var btn = document.getElementById('btn') ;  
var result = document.getElementById('result') ;
```

Utilisez un « *addEventListener* » sur « *myBtn* » qui remplace le contenu de « *result* » avec le résultat d'une fonction « *PileOuFace()* ».

Développez cette fonction « *PileOuFace()* » à l'extérieur de l'écouteur d'évènement.

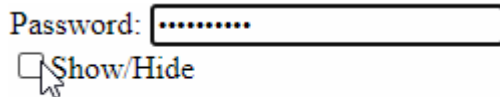
Ajoutez un « *console.log* » sur la fonction « *PileOuFace()* » dans la fonction qui écoute l'évènement au clic afin de pouvoir contrôler le résultat dans la console.

Utilisez la fonction « *Math.random()* ».

Si le résultat est inférieur à 0,5 alors c'est pile sinon, c'est face.

[3] - Afficher / cacher un mot de passe

Objectif : Lorsque l'utilisateur entre un mot de passe, il est masqué par défaut. L'idée est de lui proposer une option pour afficher/masquer son mot de passe.



Cadre de travail :

Créez un fichier « *index.html* » avec le contenu suivant :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <title>Mot de passe masqué/visible</title>
  </head>
  <body>

    <form action="">
      <label for="password">Mot de passe</label>
      <input type="password" name="password" id="input">
      <input id="checkbox" type="checkbox">
      <span id="display">Visible</span>
    </form>

    <script type="text/javascript" src="main.js"></script>
  </body>
</html>
```

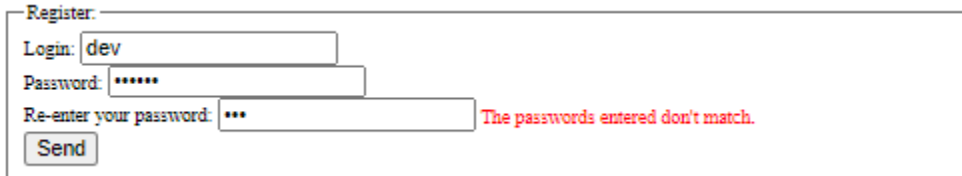
Créez un fichier « main.js » :

```
const input = document.getElementById('input');  
const display = document.getElementById('display');  
const checkbox = document.getElementById('checkbox') ;  
  
// suite du code ici
```

En HTML, les « *input* » ont un « type ». Écrivez une condition qui vérifie la valeur de « input.type » et la change en fonction du résultat !

[4] Comparer les mots de passe

Objectif : Lorsque l'utilisateur re-tape son mot de passe au moment d'une l'inscription, le script indique si les deux mots de passe saisis se correspondent.



Cadre de travail :

Créez un fichier « *index.html* » avec le contenu suivant :

```
<!DOCTYPE html>

<html lang="fr">

  <head>

    <meta charset="utf-8">

    <title>Vérifier le mot de passe</title>

  </head>

  <body>

    <form action="">

      <fieldset>

        <legend>Inscription :</legend>

        <label for="login">Login: </label>

        <input type="text" name="login"><br>

        <label for="password">Mot de passe : </label>

        <input type="password" name="password"
id="password"><br>

        <label for="check-password">Re-saisir votre mot de passe
: </label>
```



```

        <input type="password" name="check-password" id="check-
password">
        <span id="message"></span><br>
        <input type="button" value="Send">
    </fieldset>
</form>

<script type="text/javascript" src="main.js"></script>
</body>
</html>

```

Créez enfin un fichier « main.js » :

```

let password = document.getElementById("password") ;
let checkPassword = document.getElementById("check-password") ;
let alert = document.getElementById("alert") ;

checkPassword.addEventListener('keyup', () => {
    // Votre code ici
})

```

Un écouteur d'évènement surveille ce qui est entré dans le champ de formulaire qui a l'id « check-password » : si le second mot de passe entré correspond au premier, un message de succès apparaît (en vert). Tant que les deux mots de passe saisis ne correspondent pas, un message apparaît en rouge pour l'indiquer. Utilisez l'évènement keyup pour récupérer les caractères saisis au clavier par l'utilisateur.

- Dans votre condition, utilisez la propriété « .value » sur les éléments html que vous récupérez dans des constantes en JS ;
- Utilisez la propriété « style.color » pour changer la couleur du message d'alerte ;

Utilisez la propriété « innerHTML » pour changer le contenu du message d'alerte.

[5] Calcul de distance

Objectif : Ecrire un programme qui, connaissant un ensemble de villes et la distance les séparant d'un point de référence, soit capable de dire, après que l'utilisateur ait saisi une distance parcourue depuis le point de référence, quelles villes auraient pu être atteintes.

Cadre de travail :

Créez un fichier « *index.html* » avec le contenu suivant :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <title>Calcul de distance</title>
  </head>
  <body>

    <form action="">
      <fieldset>
        <legend>Entrez la distance parcourue :</legend>
        <label for="login">Distance (km) : </label>
        <input type="text" name="check-distance" id="check-distance"><br>
        <input type="button" value="Send" id="btn">
      </fieldset>
    </form>

    <script type="text/javascript" src="main.js"></script>
  </body>
</html>
```

Créez un fichier « *main.js* » avec le contenu suivant :

```
let villes = ["Bordeaux", "Nantes", "Lyon", "Marseille",
  "Monptellier", "Paris", "Rennes", "Strasbourg"];

let distance = [950, 850, 450, 800, 1000, 460, 840, 0];

// Suite du code ici
```

[6] Convertisseur d'unités de temps

Objectif : L'utilisateur entre un nombre d'années, le script convertit ce nombre en secondes, en minutes, en heures et en nombre de jours.

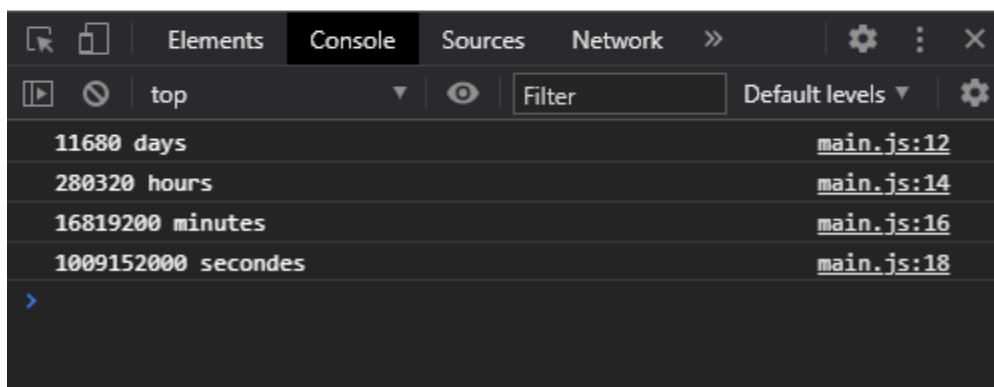
Enter a number of years here:

1009152000 secondes

16819200 minutes

280320 hours

11680 days



Créez un fichier « *index.html* » avec le contenu suivant :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <title>Convertisseur temps</title>
  </head>
  <body>
    <label for="numberOfYears">Entrez un nombre d'années</label>
```

```

<input type="number" name="numberOfYears" id="numberOfYears">
<input type="submit" value="convert" id="convert">
<p id="resultSeconds"></p>
<p id="resultMinutes"></p>
<p id="resultHours"></p>
<p id="resultDays"></p>
<script src="main.js"></script>
</body>
</html>

```

Créez le fichier « *main.js* », déclarez une variable « *convert* » qui sélectionne au moyen d'un « *getElementById* » l'*input* de type « *submit* » qui a l'id « *convert* ».

Déclarez aussi des variables contenant les valeurs des unités temporelles que vous allez utiliser.

Écoutez l'évènement « au clic » sur l'*input*, déclarez une *variable* qui récupère la valeur soumise par l'utilisateur, et développez la fonction dans cet écouteur d'évènement :

```

let convert = document.getElementById("convert") ;
const secondsInMinute = 60;
const minutesInHour = 60;
const hoursInDay = 24;
const daysInYear = 365;

convert.addEventListener("click", () => {
    let numberOfYears =
document.getElementById("numberOfYears").value ;
    // Votre code ici
});

```

Réfléchissez au moyen d'écrire le moins d'opérations possible en utilisant les résultats de calculs précédents stockés dans des variables.

Bonus :

- Gérer les années bissextiles ;
- L'utilisateur choisit l'unité d'entrée (année, jours, heures, minutes, secondes) et l'unité de sortie. Pour réussir cette étape, vous devez modifier le fichier HTML.

[7] Roue de chargement

Objectif : Au lancement de la page, une roue de chargement tourne durant 3 secondes avant de disparaître pour laisser la place au contenu de la page.



Créez un fichier « *index.html* » avec le contenu suivant :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Roue de chargement</title>
    <link rel="stylesheet" href="main.css">
  </head>
  <body>
    <div id="loadingAnimation"></div>
    <div id="displayContent">
      
    </div>
    <script type="text/javascript" src="main.js"></script>
```

```
</body>
</html>
```

Créez ensuite un fichier « *main.css* » dans lequel vous mettrez le contenu suivant :

```
#displayContent {
    display: none;
    text-align: center;
}

#LoadingAnimation {
    position: absolute;
    left: 50%;
    top: 50%;
    width: 100px;
    height: 100px;
    border: 16px solid #f3f3f3;
    border-radius: 50%;
    border-top: 16px solid #d31f43;
    animation: spin 3s linear infinite;
}

@keyframes spin {
    // TODO
}
```

Enregistrez l'image logo-accs.png à la racine du répertoire.

Enfin, créez un fichier « *main.js* » :

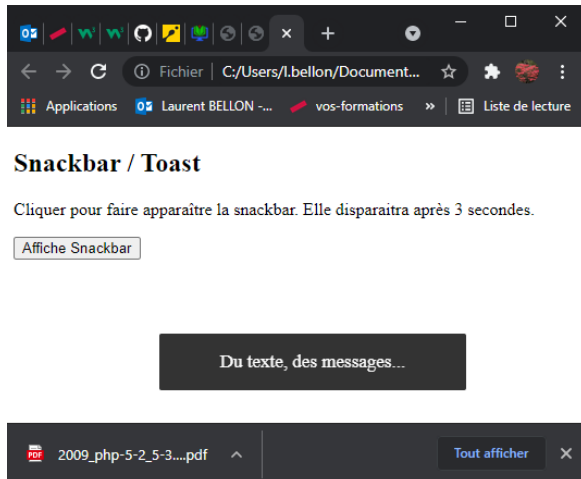
```
let loading = document.getElementById("loadingAnimation") ;
let content = document.getElementById("displayContent") ;
```

L'exercice consiste à changer les propriétés CSS « *display* » des 2 divs après un laps de temps (par exemple 3 secondes) lancé dès que la page est entièrement chargée, événement qui est détecté par `window.onload`.

Utilisez la fonction [`.setTimeout`](#) pour définir le laps de temps.

[8] – Afficher une snackbar

Objectif : Lorsque l'utilisateur clique sur un bouton, une snackbar (aussi appelée toast) apparaît en bas de l'écran. Et disparaît au bout de 3 secondes



Créez un fichier « *index.html* » avec le contenu suivant :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="main.css">
    <script src="./main.js" defer></script>
  </head>
  <body>
    <h2>Snackbar / Toast</h2>
    <p>Cliquer pour faire apparaître la snackbar. Elle disparaîtra après 3
secondes.</p>
    <button id="btn">Affiche Snackbar</button>
    <div id="snackbar">Du texte, des messages...</div>
  </body>
</html>
```

Créez un fichier « *main.css* » avec le contenu suivant :

```
#snackbar {
    visibility: hidden;
    width: 250px;
    background-color: #333;
    color: #fff;
    text-align: center;
    padding: 16px;
    border-radius: 10px;
    position: absolute;
    z-index: 1;
    left: 50%;
    margin-left: -125px;
    bottom: 30px;
    font-size: 17px;
}

#snackbar.show {
    visibility: visible;
    animation: appear 0.5s, disappear 0.5s 2.5s;
}

@keyframes appear {
    from {
        bottom: 0; opacity: 0;
    }
    to {
        bottom: 30px; opacity: 1;
    }
}

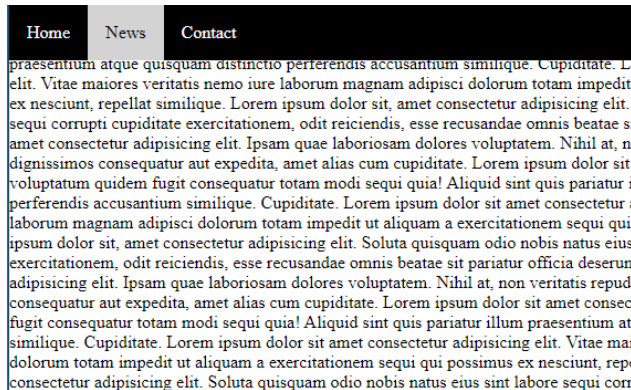
@keyframes disappear {
    from {
        bottom: 30px; opacity: 1;
```

```
}  
to {  
    bottom: 0; opacity: 0;  
}  
}
```

Dans un fichier « main.js »

- On utilise un écouteur sur l'événement « click » sur la balise button.
- A la suite du 'click', faire apparaître la snackbar.
- Au bout de 3 secondes, la snackbar doit disparaître

[9] Apparition d'une navbar au défilement vers le bas



Objectif : Lorsque l'utilisateur *scroll* vers le bas sur la page web, une *navbar* apparaît en haut de l'écran.

Commencez par [récupérer du Lorem Ipsum en utilisant un générateur](#). Partez sur 30 paragraphes, pour avoir de la marge lors du *scroll*. Créez ensuite un fichier « *index.html* » avec le contenu suivant, dans lequel vous insérerez votre *Lorem Ipsum* :

```
<!DOCTYPE html>

<html lang="fr">

  <head>

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1.0">

    <title>Défilement</title>

    <link rel="stylesheet" href="main.css">

  </head>

  <body>

    <div id="navbar">

      <a href="#home">Accueil</a>

      <a href="#news">Nouveautés</a>

      <a href="#contact">Contact</a>

    </div>

    <div class="lorem">
```

```
        <!-- Insérez ici du Lorem Ipsum -->
    </div>
    <script type="text/javascript" src="main.js"></script>
</body>
</html>
```

Créez ensuite un fichier « *main.css* » dans lequel vous mettrez le contenu suivant :

```
body {
    margin:0;
}

#navbar {
    background-color: black;
    position: fixed;
    display: flex;
    flex-direction: row;
    top: -50px;
    width: 100%;
    transition: top 0.8s;
}

#navbar a {
    color: white;
    padding: 16px;
    text-decoration: none;
}

#navbar a:hover {
    background-color: lightgrey;
    color: black;
}
```

Enfin, créez un fichier « main.js » :

```
window.onscroll = () => {  
    // Votre code ici  
}
```

- Dans le CSS, vous constatez que la position Top de la *navbar* est à -50px. L'idée de cet exercice est de mettre une condition qui passe la position Top de la *navbar* à 0 quand le défilement atteint une certaine hauteur.
- Faites une recherche pour savoir dans quel objet JS se trouve la propriété « scrollTop » qui donne la hauteur de défilement de la page.

[10] Années bissextiles

Objectif : L'utilisateur saisit une année, le script vérifie s'il s'agit d'une année bissextile ou non.

Créez un fichier « *index.html* » avec le contenu suivant :

```
<!DOCTYPE html>

<html lang="fr">

  <head>

    <meta charset="utf-8">

    <title>Année bissextile</title>

  </head>

  <body>

    <label for="year">Entrez une année :</label><br>

    <input type="year" name="year" id="year">

    <input type="submit" value="check" id="check">

    <p id="result"></p>

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

  </body>

</html>
```

Créez ensuite le fichier « *main.js* », déclarez une variable « *check* » qui sélectionne au moyen d'un « *getElementById* » l'*input* de type « *submit* » qui a l'id « *check* »..

Écoutez l'évènement « au clic » sur l'*input*, déclarez une *variable* qui récupère la valeur soumise par l'utilisateur, et développez la fonction dans cet écouteur d'évènement :

```
let check = document.getElementById("check") ;

check.addEventListener("click", () => {

  let year = document.getElementById("year").value ;

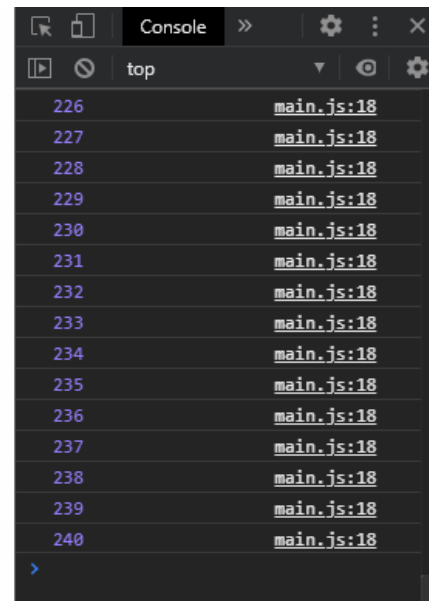
  // Votre code ici
```

```
});
```

Écrivez une condition if/else qui vérifie si l'année saisie par l'utilisateur est bissextile ou non, si elle l'est, la balise html qui a l'id « *result* » l'indique, sinon elle indique que l'année n'est pas bissextile. Utilisez le modulo %.

[11] Animation Logo

Objectif : Au chargement de la page, un minuscule logo surgit et grandit jusqu'à atteindre une largeur et une hauteur de 240px.



Créez un fichier « *index.html* » avec le contenu suivant :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title> Logo animé</title>
    <link rel="stylesheet" href="main.css">
  </head>
  <body>
    <div id="logo">ACCESS CODE<br>SCHOOL</div>
```

```
    <script type="text/javascript" src="main.js"></script>
  </body>
</html>
```

Créez ensuite un fichier « main.css » :

```
body {
  padding: 0;
  margin: 0;
  height: 100vh;
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
}

#logo {
  /* Style du cercle */
  position: fixed;
  background-color: #d21242;
  border-radius: 50%;
  height: 0;
  width: 0;
  /* Style du texte */
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  text-align: center;
  color: white;
}
```

```
}
```

Créez enfin un fichier « main.js » :

```
var logo = document.getElementById('logo');  
var value = 0 ;  
function animLogo(){  
    // Votre code ici}  
  
setInterval(animLogo, 10);
```

- Initialisez une variable à 0 et créez une condition dans le setInterval(), si la variable est inférieure à 240, incrémentez là ;
- Le texte et le rond rouge grandissent en même temps. Utilisez style.height, style.width, style.fontSize pour augmenter les valeurs des propriétés css dans le code javascript (vous pouvez faire des opérations mathématiques).

[12] Animer une icône

Objectif : Utiliser des icônes de fontawesome pour créer une animation.



Créez un fichier « *index.html* » avec le contenu suivant :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Icon Animation</title>
    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.15.1/css/all.css"
integrity="sha384-
vp86vTRFVJgpjF9jiIGPEEqYqlDwgyBgEF109VFjmqGmIY/Y4HV4d3Gp2irVfcrp "
crossorigin="anonymous">
    <link rel="stylesheet" href="main.css">
  </head>
  <body>
    <div id="balanceIcon" class="icon fa"></div>
    <script type="text/javascript" src="main.js"></script>
  </body>
</html>
```

Créez ensuite un fichier « *main.css* » :

```
.icon{
```

```
font-size:100px;
}
```

Créez enfin un fichier « main.js » :

```
function balanceScale() {
// Développez la fonction
}

balanceScale();
setInterval(balanceScale, 4000) ;
```

- Tout d'abord, vous allez avoir besoin de consulter la [Cheatsheet de Font Awesome](#) pour récupérer les entités html de vos icônes.

Dans le code javascript, le code est précédé des trois caractères `&#x`. L'icône « balance avec les plateaux horizontaux » est donc appelée par « `` »

- Dans la fonction `balanceScale()`, vous allez avoir besoin de :
 - Récupérer l'élément qui a l'id « `balanceIcon` » ;
 - Utiliser `innerHTML` pour définir le contenu de départ de votre div, dans cet exemple ce sera donc

```
balanceIcon.innerHTML = "&#xf24e;";
```

- Faire des `setTimeout()` pour chacune des étapes de votre animation, au cours de laquelle l'entité html de la balance est remplacée ;
- Paramétrer le temps entre chaque étape à une seconde, normalement vous avez un état initial et 3 modifications de cet état (la balance repasse par le stade horizontal entre chaque abaissement du plateau), donc les 4 secondes du `setInterval()` (si vous développez d'autres animations avec plus d'étapes, adaptez le temps pour que ça soit fluide) ;

Bonus :

- Développez une animation cadenas ouvert / fermé, il y a 2 icones de fontawesome qui vous permettent de le faire ;
- Développez une animation de chargement de batterie, il y a 5 icones de fontawesome qui vous permettent de le faire ;
- Inventez votre propre animation 😊

[13] Animer un titre

Objectif : Réaliser un effet d'actualisation d'un message.



Cadre de travail :

Créez un fichier « *index.html* » avec le contenu suivant :

```
<!DOCTYPE html>

<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Animate Title</title>
    <link rel="stylesheet" href="main.css">
  </head>
  <body>
    <div class="container">
      <div class="image-wrapper left"></div>
      <hr>
```

```
<h1 id="title">Salut !</h1>
<div class="image-wrapper right"></div>
</div>

<script type="text/javascript" src="main.js"></script>
</body>
</html>
```

Créez ensuite un fichier « main.css » :

```
* {
    padding: 0;
    margin: 0;
}

.container {
    width: 100vw;
    height: 100vh;
    display: flex;
}

.image-wrapper {
    width: 50%;
    height: 100%;
    background-size: cover;
    position: absolute;
}

.left {
    background-image: url('left.jpg');
    background-position: right;
    z-index: 4;
}
```

```
}
```

```
.right {  
    background-image: url('right.jpg');  
    background-position: left;  
    left: 50%;  
    z-index: 1;  
}
```

```
hr {  
    height: 180px;  
    border: none;  
    border-left: solid 10px #76a89d;  
    position: absolute;  
    z-index: 4;  
    left: 50%;  
    top: 50%;  
    transform: translate(-48%, -48%)  
}
```

```
#title {  
    font-size: 3rem;  
    text-align: center;  
    position: absolute;  
    z-index: 2;  
    left: 50%;  
    top: 50%;  
    margin-left: 2rem;  
    transform: translate(-120%, -56%);  
    animation: showTitle 3s infinite ;  
}
```



```
}
```

```
@keyframes showTitle {  
  0% {  
    transform: translate(-120%, -56%)  
  }  
  15% {  
    transform: translate(0%, -56%)  
  }  
  80% {  
    transform: translate(0%, -56%)  
  }  
  94% {  
    transform: translate(-120%, -56%)  
  }  
  100% {  
    transform: translate(-120%, -56%)  
  }  
}
```

```
@media screen and (max-width: 600px) {  
  hr {  
    height: 136px;  
  }  
  
  .title {  
    font-size: 2rem  
  }  
}
```

Dans le fichier main.css, vous constaterez qu'il y a des z-index, l'idée, c'est que l'écran est divisé en deux parties égales qui contiennent chacune une image. Le texte passe sous celle de gauche en disparaissant, et se superpose à celle de droite pour réapparaître. Enregistrez ces deux images dans votre répertoire :

Créez enfin un fichier « main.js » :

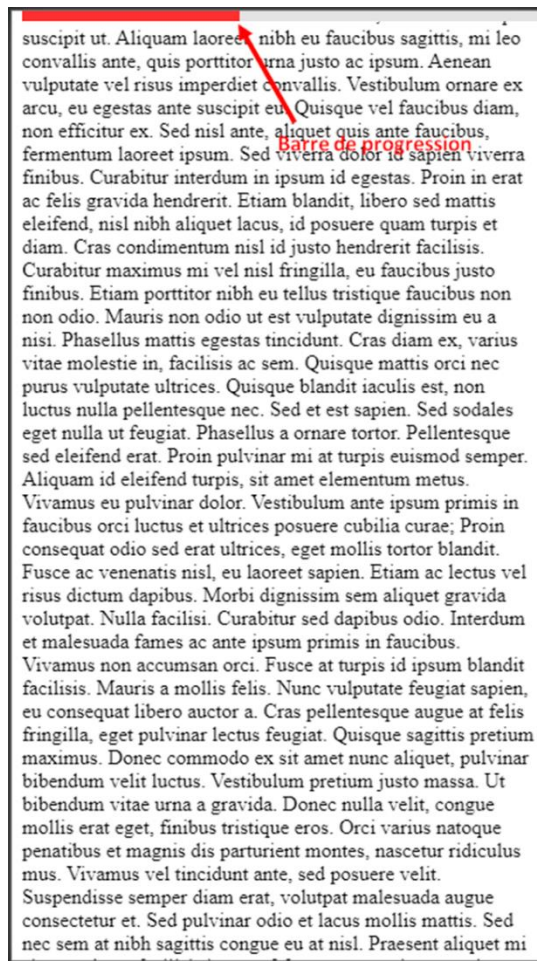
```
const title = document.getElementById('title');  
function changeTitle() {  
    // Votre code ici  
}  
  
setInterval(changeTitle, 3000);
```

Consigne :

- Ce sont les keyframes dans le css qui gèrent la fluidité de l'animation. Il suffit de faire une série de conditions pour chaque cas et de remplacer le texte dans le HTML, on peut le faire avec un switch mais aussi avec une variable compteur qu'on incrémente et des if/else if/else.

[14] Indicateur de défilement

Objectif : Lorsque l'utilisateur *scroll* vers le bas sur la page web, une barre de progression placée en haut de la fenêtre lui indique le pourcentage de la page qu'il a déjà lu et qu'il lui reste à lire.



Commencez par [récupérer du Lorem Ipsum en utilisant un générateur](#). Partez sur 30 paragraphes, pour avoir de la marge lors du *scroll*.

Créez ensuite un fichier « *index.html* » avec le contenu suivant, dans lequel vous insérerez votre *Lorem Ipsum* :

```
<!DOCTYPE html>

<html lang="fr">

  <head>

    <meta charset="utf-8">
```

```
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<title>Indicateur défilement</title>
<link rel="stylesheet" href="main.css">
</head>
<body>
  <div class="progress-container">
    <div class="progress-bar" id="bar"></div>
  </div>
  <div class="content">
    <!-- Insérez ici du Lorem Ipsum -->
  </div>
  <script type="text/javascript" src="main.js"></script>
</body>
</html>
```

Créez ensuite un fichier « *main.css* » dans lequel vous mettrez le contenu suivant :

```
.progress-container {
  position: fixed;
  top: 0;
  z-index: 1;
  width: 100%;
  width: 100%;
  height: 8px;
  background: #E4E4E4;
}

.progress-bar {
  height: 8px;
  background: #FF3131;
```

```
width: 0%;  
}
```

Enfin, créez un fichier « main.js » :

```
window.onscroll = function() { scrollIndicator() } ;
```

```
function scrollIndicator() {  
    // Votre code ici  
}
```

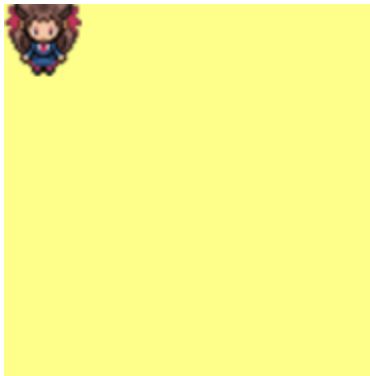
- Déclarez une variable « windowScroll » qui indique le nombre de pixels dont le contenu de l'élément `<html>` a défilé vers le haut ;
- Déclarez une variable « height » qui soustrait la hauteur totale en pixels de l'élément `<html>` de la hauteur totale de l'écran `<html>` ;
- Déclarez une variable qui contient le calcul du pourcentage de votre *défilement* : le nombre de pixels dont le contenu de l'élément `<html>` a défilé vers le haut divisé par la variable height, et multipliez le résultat par 100.
- Appliquez la valeur obtenue à la *width* de la *div* qui a l'id « bar ».

Ressources utiles à connaître pour l'exercice :

- [window](#)
- [onscroll](#)
- [document.documentElement](#)
- [scrollTop](#)
- [scrollHeight](#)
- [clientHeight](#)
- [style.width](#)

[15] Animer un personnage

Objectif : Les flèches directionnelles du clavier permettent de déplacer le personnage en bas, en haut, à gauche et à droite. Lorsqu'il se déplace, le personnage simule un mouvement de marche vers le haut, le bas, la gauche ou la droite.



Créez un fichier « *index.html* » avec le contenu suivant :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <title>Animation personnage</title>
    <link rel="stylesheet" href="main.css">
  </head>
  <body>
    <div id="gameContainer">
      <div id="player"></div>
    </div>
    <script src="player.js" type="text/javascript"></script>
  </body>
</html>
```

Et le fichier « *main.css* » :

```
#gameContainer {  
  width: 800px;  
  height: 800px;  
  margin: 0 auto;  
  background-color: rgba(255, 255, 24, 0.5);  
  position: relative;  
  overflow: hidden;  
}
```

```
#player {  
  width: 40px;  
  height: 40px;  
  background-image: url('img/face.png');  
  background-size: 100%;  
  background-position: center;  
  position: absolute;  
  z-index: 1;  
}
```

Enfin, créez un répertoire « *img* » dans lequel vous placerez les éléments suivants, issus d'une *spritesheet* (préalablement découpée pour les besoins de l'exercice) :

Créez ensuite le fichier « *player.js* » avec le script suivant :

```
let player = document.getElementById('player');  
  
document.addEventListener('keydown', (event) => {  
  if (event.code == 'ArrowUp') {  
    // Votre code ici  
  }  
})
```

```
});
```

Définir une constante « `moveSize` » qui définit de combien de pixels le personnage se déplace à chaque mouvement.

Définir une variable qui compte le nombre de mouvements du personnage. Cette variable va permettre de déterminer pour chaque paire de visuels lequel afficher.

Pour chaque évènement (`ArrowUp`, `ArrowRight`, `ArrowDown`, `ArrowLeft`) afficher le visuel du personnage correct.

Définir la position du personnage sur le plateau au moyen des directives `player.style.top` et `player.style.left`

Indices :

- la méthode « `style.backgroundImage` »
- la méthode « `player.offsetTop` »

Bonus :

- Limitez les mouvements du personnage à la taille du plateau de jeu ;
- Changez de *spritesheet* pour personnaliser un peu tout ça ;
- [pour les plus guerriers] Développez un jeu sur cette base !