Explication de mon code:

Inscription/connexion:

HTML:

dans le head juste avant le body, on met le lien à la page css et le titre de mon document.

```
<link rel="stylesheet" href="connexion-inscription.css">
    <title>Global Jobs</title>
</head>
```

Dans le body nous mettons le contenue principal du html

```
<body>
```

Cela représente la configuration de la page (qu'on retrouve dans le css)

```
      <div class="container">
```

ici on ajoute le logo d'avion:

```
       <div>
              <h1> <img src="avion.png" id="avion" alt="avion">Global
<br> Jobs</h1>
```

ici on fait le formulaire de connexion
label sert à donner un intitulé à un champ.
et l'input à créer une zone de texte avec des conditions (ex type="email" nous devrions mettre @ dans le champ de texte pour que ce soit valide).
required est utilisé pour rendre le champ de texte obligatoire.
La balise small sert à mettre du texte en plus petit, ici c'est pour les messages d'erreurs.

```
            <form id="loginForm">
                <label for="mail">Email :</label>
                <input type="email" id="mail" name="user_mail"
required/>
                <small class="error" id="login-email-error"></small>
                <label for="password">Mot de passe :</label>
                <input type="password" id="password" name="password"
required/>
                <small class="error" id="login-password-error"></small>
```

Ici on ajoute le bouton de connexion, qui est fonctionnel grâce au js.
on a le bouton pour se connecter et celui pour s'inscrire. On ferme le formulaire.

```
                <div>
                    <button id="submit" type="submit">Se
connecter</button>
```

```
                    <button id="register"
type="button">S'inscrire</button>
                </div>

            </form>
```

On refait un formulaire, qui s'affiche lorsqu'on clique sur s'inscrire fait juste avant avec bouton. Et on ajoute deux boutons s'inscrire et retour.

```
            <form id="Formc" class="form">
                <label for="nom">Nom :</label>
                <input type="text" id="nom" name="nom" required/>
                <label for="Prenom">Prénom :</label>
                <input type="text" id="Prenom" name="Prenom" required/>
                <label for="mail">Email :</label>
                <input type="email" id="mail2" name="user_mail"
required/>
                <small class="error"
id="inscription-email-error"></small>

                <label for="phone">Numéro de téléphone :</label>
                <input type="tel" id="phone" name="phone">
                <small class="error"
id="inscription-phone-error"></small>

                <label for="username">Username :</label>
                <input type="text" id="username" name="username"
required/>
                <small class="error"
id="inscription-username-error"></small>

                <label for="password">Mot de passe :</label>
                <input type="password" id="password2" name="password"
required/>
                <small class="error"
id="inscription-password-error"></small>

                <div>
                    <button id="connexion"
type="submit">S'inscrire</button>
                    <button id="retour" type="button">Retour</button>
                </div>
            </form>
```

```
        </div>
```

Et pour finir on souhaite ajouter une image à droite de notre écran (possible grâce aux grid dans css)

```
         <div>
            <img src="paysage.png" data-info="paysage">
        </div>
    </div>
```

On finit par fermer notre body.
Avant la fermeture du body on a ajouté le lien au js

```
 <script src="connexion-inscription.js"></script>
</body>
```

CSS:

on utilise le container qu'on avait mis dans notre div, ici on créer une grid avec display (qui permet d'organiser nos éléments en lignes et colonnes) on lui créer deux colonnes avec `grid-template-columns` et une ligne `grid-template-rows;`
gap permet de définir les espaces entre les lignes et les colonnes dans une grid.
height permet de définir la hauteur de notre élément.
width permet de définir la largeur.
padding est utilisé pour définir les différents écarts de remplissage sur les quatre côtés d'un élément.
box-sizing définit la façon dont la hauteur et la largeur totale d'un élément est calculé, ici avec border box on dit de prendre en compte dans la valeur défini, la bordure et le remplissage, en gros la taille prend la bordure et le remplissage.

```
.container {
    display: grid;
    position: relative;
    grid-template-columns: repeat(2, 1fr);
    grid-template-rows: 100%;
    gap: 1rem;
    height: 100vh;
    width: 100%;
    padding: 1rem;
    box-sizing: border-box;
```

Ici on récupère dans la div container la dernière div(la deuxième partie de la grid) et donc l'image (la div avec l'image paysage).
en mettant width et height à 100 l'image prend toute la hauteur et la largeur;
background-size permet à l'image de couvrir tout l'arrière plan.

```
.container>div:last-of-type>img{
    width: 100%;
    height: 100vh;
    background-size: cover;
}
```

ici on prend dans container la première div, donc l'image d'avion, le titre et le form. On vient gérer leur emplacement dans la grid.
On lui donne une position flexible grâce au display flex, donc là on utilise les flex box, cela nous permet de positionner les éléments comme on le souhaite dans la page.
flex-direction : column, on dit de mettre notre form et img en colonne
et ensuite avec align-items: center, de placer cette colonne au centre de notre grid.

```
.container>div:first-of-type{
    display:flex;
    flex-direction: column;
    align-items: center;
}
```

On vient définir comment sera disposé le formulaire, les éléments du form en colonne et centré.

```
form{
    display: flex;
    flex-direction: column;
    align-items: center;
```

Dans le formulaire on vient modifier les input,
Avec margin-bottom on définit l'espace de marge en bas des éléments.
on a définit le fond de couleur des inputs avec background-color.

```
form>input{
    margin-bottom: 30px;
    border: none;
    background-color: rgb(206, 203, 203);
    border-radius: 10px;
    padding: 10px;
    width: 75%;
}
```

On vient ajuster les boutons se connecter et s'inscrir,
Font family permet de définir une liste, ordonnée par priorité, de polices à utiliser pour mettre en forme le texte de l'élément ciblé.
Font weight permet de définir l'épaisseur des caractères.

```
form>div:first-of-type{
    display: flex;
    align-items: center;
    justify-content: center;
    gap: 40px;
    font-family: Arial, Helvetica, sans-serif;
    font-weight: bolder;
}
```

On vient ensuite ajuster tous les boutons:
on donne un fond bleu et un texte blanc

```
button{
    background-color: #352B6B;
    color: white;
    padding: 10px;
    width: 150px;
    height: 40px;
    font-size: 18px;
    border: none;
    border-radius: 8px;
```

Pour le titre de notre site : on vient prendre h1 (qui définit le titre principale en html)
on met text shadow pour faire des ombres.

```
h1{
    color:#FFC533;
    font-weight: bolder;
    font-size: 50px;
    font-family: 'Montserrat', 'Poppins', sans-serif;
    text-align: center;
    text-shadow:#352B6B 4px 4px 4px;
```

On ajuste les labels de nos inputs (titre comme email nom etc….)

```
form>label{
    display: flex;
    color:darkgray;
    font-family: Arial, Helvetica, sans-serif;
    font-weight: bolder;
    text-align: left;
```

dans le h1 on vient ajouter notre image d’avion comme logo
on le place au dessus du titre, avec transform on le déplace vers l’image paysage on voulait faire une transition mais on a pas eu le temps;

```
h1>img{
    display:flex;
    width: 50vh;
    height: 25vh;
    align-items: center;
    padding: 10px;
    transform: translate(720px, 35px);
}
```

On utilise cela pour cacher le form d'inscription (il apparaît grâce au js quand on clique sur s'inscrire)

```
/*form d'inscrpition caché*/
#Formc {
    display: none;
}
```

et pour finir on rend la page responsive avec media screen:

```
@media screen and (max-width: 500px){
    body{
        margin: 0;
        padding: 0;
        overflow-x: hidden;
        width: 100%;
    }
    .container {
    display: grid;
    position: relative;
    grid-template-columns: 100%;
    grid-template-rows: repeat(2, 1fr);
    gap: 1rem;
    width: 100vw !important;
    padding: 0 !important;
}

    h1>img{
    transform: translate(0px, 0px) !important;
    display:flex;
    width: 90vw;
    height: 30%;
    align-items: center;
```

```
    padding: 25px;
}

.container>div:last-of-type{
   grid-row: 1;
   height: 140%;
}

.container>div:last-of-type>img{
    height : 100%;
}
form>div:first-of-type{
    display: flex;
    align-items: center;
    justify-content: center;
    gap: 3px;
    font-family: Arial, Helvetica, sans-serif;
    font-weight: bolder;
}
}
```

JAVASCRIPT:

La première ligne permet d'attendre que tout le code html soit chargé avant d'exécuter le code JS.
Ici on récupère des id qu'on vient mettre dans des variables (ex: buttonRegister) pour accéder facilement aux éléments HTML

```
    document.addEventListener("DOMContentLoaded", () => {
    const buttonRegister = document.getElementById("register");
    const formInscription = document.getElementById("Formc");
    const formConnexion = document.getElementById("loginForm");
    const retourBtn = document.getElementById("retour");
```

Dans la première ligne on ajoute un écouteur d'événement sur le buttonRegister donc quand on clique sur le bouton ce qu'il y'a entre parenthèse s'exécute, c'est à dire si on a un compte on clique sur se connecter et ca marche, si non on clique sur inscription et le form apparaît. Donc on empêche l'envoie du form ensuite on cache le form de connexion et on affiche celui d'inscription quand on clique sur s'inscrire.

```
buttonRegister.addEventListener("click", (event) => {
        event.preventDefault();
        formConnexion.style.display = "none";
        formInscription.style.display = "flex";
    });
```

Ensuite on refait la même chose avec le bouton retour, on clique sur retour ca cache le form d'inscription pour faire apparaître celui de connexion;
La même chose lorsqu'on fait s'inscrire, on enregistre et on retourne sur connexion pour que l'utilisateur se connecte.

```
const retourBtn = document.getElementById("retour");
    retourBtn.addEventListener("click", (event) => {
        event.preventDefault();
        formInscription.style.display = "none";
        formConnexion.style.display = "flex";
    });
    const registerBtn = document.getElementById("inscrire");
    registerBtn.addEventListener("click",(event)=> {
        event.preventDefault();
        formConnexion.style.display ="flex"
    });
```

Ici quand l'utilisateur se connecte, on vient vérifier tous les éléments avant l'envoi.
Quand on essaie à nouveau de se connecter on enlève les anciennes erreurs et on affiche que celles qui sont encore présentes.

```
document.getElementById("loginForm").addEventListener("submit",
function (e) {
  e.preventDefault();
  let valid = true;

  const email = document.getElementById("mail").value.trim();
  const password = document.getElementById("password").value.trim();

  document.getElementById("login-email-error").textContent = "";
  document.getElementById("login-password-error").textContent = "";

  if (!/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email)) {
    document.getElementById("login-email-error").textContent = "Email
invalide.";
    valid = false;
  }

  if (password.length < 6) {
    document.getElementById("login-password-error").textContent = "Mot
de passe trop court (min 8 caractères).";
    valid = false;
  }
```

```
  if (valid) alert("Connexion réussie");
});
```

On fait la même chose avec le form d'inscription pour les erreurs:

```
// ---- VALIDATION INSCRIPTION ----
document.getElementById("Formc").addEventListener("submit", function
(e) {
  e.preventDefault();
  let valid = true;

  const email = document.getElementById("mail2").value.trim();
  const phone = document.getElementById("phone").value.trim();
  const username = document.getElementById("username").value.trim();
  const password = document.getElementById("password2").value.trim();

  // Réinitialiser les erreurs
  document.querySelectorAll(".error").forEach(e => e.textContent = "");

  if (!/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email)) {
    document.getElementById("inscription-email-error").textContent =
"Email invalide.";
    valid = false;
  }

  if (!/^\d{10}$/.test(phone)) {
    document.getElementById("inscription-phone-error").textContent =
"Téléphone invalide (10 chiffres).";
    valid = false;
  }

  if (["admin", "user", "test"].includes(username.toLowerCase())) {
    document.getElementById("inscription-username-error").textContent =
"Ce nom d'utilisateur est déjà utilisé.";
    valid = false;
  }

  if (password.length < 8) {
    document.getElementById("inscription-password-error").textContent =
"Mot de passe trop court (min 6 caractères).";
    valid = false;
  }
```

```
  if (valid) {
    alert("Inscription réussie");
    this.reset();
  }
});
```