Projet 4

Créer une page landing avec Javascript

Sommaire

Présentation

Projet

Projet : Contexte et consignes à suivre

Contexte

Jason a réalisé les maquettes HTML et CSS de la page d'accueil et de la "modal".
Il a aussi commencé à écrire quelques lignes de code en Javascript.
Son travail est sur Github dans un repo Gameon-website-fr

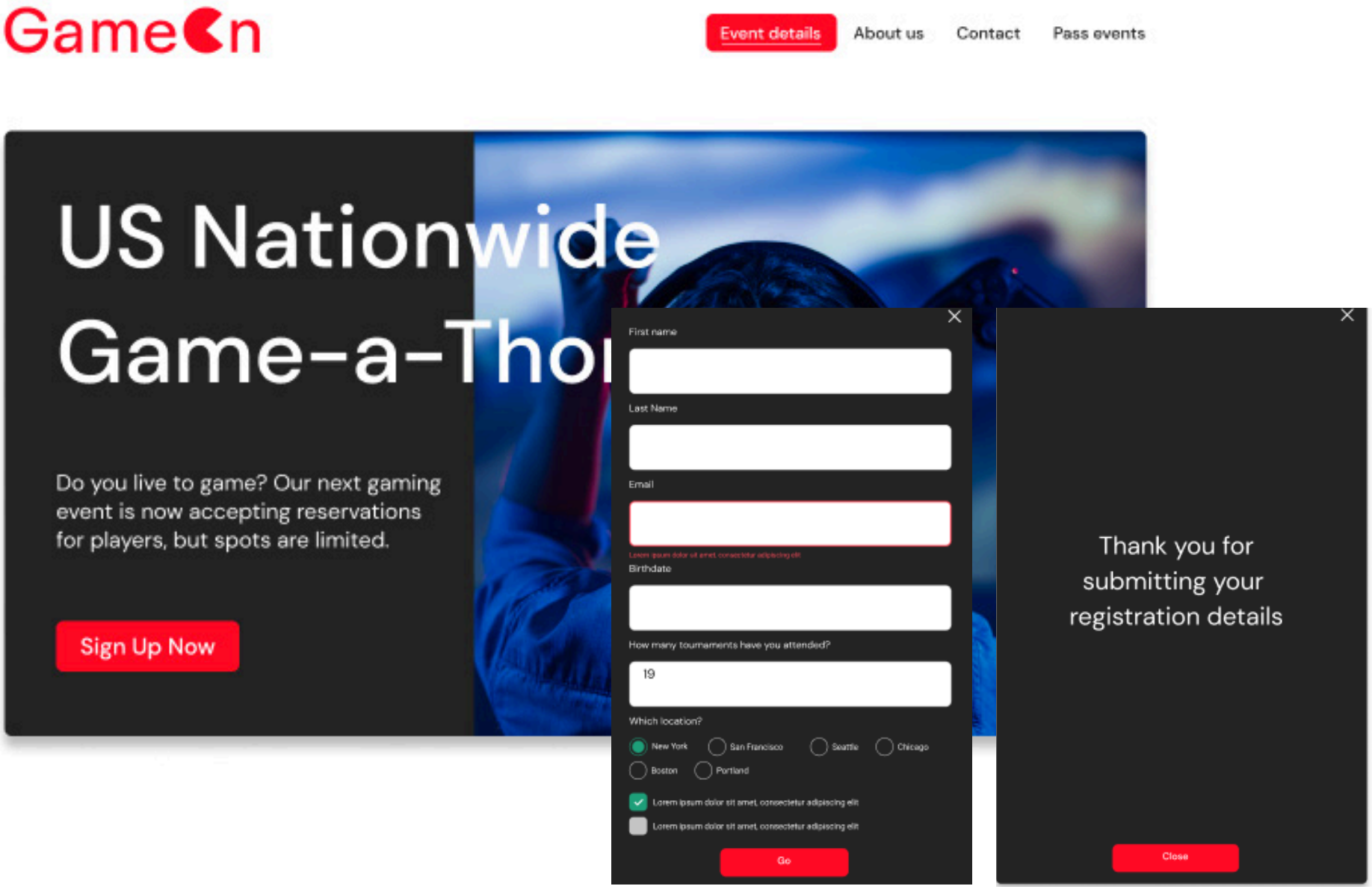
Le projet

Le projet consiste à ajouter le code JavaScript manquant pour que le formulaire soit pleinement fonctionnel, les consignes sont sur le repo GitHub qui décrivent la marche à suivre.

Les consignes

Travailler sur un repo GitHub forké ;
Utiliser des fichiers séparés pour le HTML, le CSS et le JavaScript;
Commenter le code;
Tester manuellement les fonctionnalités, les entrées de formulaire et l'affichage responsive.

Maquettes HTML / CSS



Issues à réparer

Issue 1 : Fermer la modal

Issue 2 : Implémenter les entrées du formulaire :

- Lier les labels aux entrées dans HTML avec les attributs “for” et “id”
- Le formulaire doit être valide quand l’utilisateur clique sur “Submit”
- Les champs Prénom / Nom ont un minimum de 2 caractères / ne sont pas vides.
- L’adresse électronique est valide
- Une date est sélectionnée dans le calendrier
- Nombre de concours, une valeur numérique est saisie.
- Un bouton radio est sélectionné
- La case des conditions générales est cochée

Issue 3 : Ajouter validation ou messages d’erreurs du type :

- “Veuillez entrer 2 caractères ou plus pour le champ du nom.”

Issue 4 : Ajouter un message de confirmation quand l’envoi est réussi

Issue 5 : Tests manuels

Projet

Modal & issues

Code commencé par Jason - Renommage

A gauche est le code commencé par Jason.
Dans un premier temps, j'ai renommé la variable x par myTopNav dans la fonction editNav pour qu'elle soit plus parlante.

```
function editNav() {
  var x = document.getElementById("myTopnav");
  if (x.className === "topnav") {
    x.className += " responsive";
  } else {
    x.className = "topnav";
  }
}

// DOM Elements
const modalbg = document.querySelector(".bground");
const modalBtn = document.querySelectorAll(".modal-btn");
const formData = document.querySelectorAll(".formData");

// launch modal event
modalBtn.forEach((btn) => btn.addEventListener("click", launchModal));

// launch modal form
function launchModal() {
  modalbg.style.display = "block";
```

Issue 1 : Fermer la modal

Dans devtool, je cherche quel élément représente l'icone de fermeture de la modal,

il s'agît de la classe close de la balise span

Je créé une variable close, dans laquelle je sélectionne cette balise (noeud HTML) à travers le DOM

avec la méthode querySelector en lui indiquant la classe close

La variable modalbg représente la modal globale (classe bground)

J'écoute un évenement avec addEventListener

Au click sur close qui représente la balise HTML <span class="close>

La modal se masque en activant style.display ="none" sur la balise HTML

```
10  // DOM Elements
11  const modalbg = document.querySelector(".bground")
12  const modalBtn = document.querySelectorAll(".modal-btn")
13  const formData = document.querySelectorAll(".formData")
14  const close = document.querySelector('.close')
15
16  // launch modal event
17  modalBtn.forEach((btn) => btn.addEventListener("click", launchModal))
18
19  // launch modal form
20  function launchModal() {
21    modalbg.style.display = "block";
22  }
23
24  close.addEventListener('click', () => {
25    modalbg.style.display = "none"
26  });
```

Issue 2 : Implémenter les entrées du formulaire (1)

Attributs “for” et “id”

```html
<div class="formData">
  <label for="email">E-mail</label><br>
  <input class="text-control"  name="email" id="email"/><br>
  <div class="email-error-message hidden">Veuillez saisir une adresse email valide</div>
</div>
```

Je lie mon label à mon input
avec les attributs for et id
qui doivent être identiques

Déclaration - Variables

Je déclare mes variables et leur assigne
un noeud HTML à chacune.

Je sélectionne ce noeud HTML
A travers le DOM avec document
Et la méthode querySelector()

document.querySelector()

```js
// Je déclare mes variables
const reservationForm = document.querySelector('.reservation-form')

const firstNameInput = document.querySelector('#first')
const firstNameErrorMessage = document.querySelector('.first-name-error-message')

const lastNameInput = document.querySelector('#last')
const lastNameErrorMessage = document.querySelector('.last-name-error-message')

const emailInput = document.querySelector('#email')
const emailInputErrorMessage = document.querySelector('.email-error-message')

const tournamentNumberInput = document.querySelector('#quantity')
const tournamentNumberErrorMessage = document.querySelector('.number-error-message')

const calendarInput = document.querySelector('#birthdate')
const calendarErrorMessage = document.querySelector('.error-message-birthdate')

const citiesLocation = document.querySelectorAll('.citiesLocation input')
const citiesLocationErrorMessage = document.querySelector('.citiesLocation-error-message')

const requiredCheckbox = document.querySelector('#checkbox1')
const requiredCheckboxErrorMessage = document.querySelector('.error-message-conditions')

const closeModal = document.querySelector('.modal-body')
```

Issue 2 : Implémenter les entrées du formulaire (2)

Mode opératoire pour chacune de mes inputs :

j'utilise une fonction dans laquelle je pose une condition que je vérifie avec if / else statement
Selon que la condition soit true ou false, j'applique un message d'erreur ou non

Pour afficher ou masquer un message d'erreur :
je joue avec la propriété classList.add ou classList.remove
La classe HTML en question est hidden, sa propriété CSS est display: none

Si la condition est vrai, le noeud HTML contenu dans la variable de message d'erreur ne s'affichera pas, sinon il s'affichera

A chacune de mes fonctions :
j'applique return pour définir une valeur à renvoyer à la fonction appelante.

CSS

```css
.hidden {
  display: none;
}
```

Fonctions JS

```js
// Fonction validation prénom
function checkFirstName() {
  const isFirstNameValid = firstNameInput.value.length > 2

  if (isFirstNameValid) {
    firstNameErrorMessage.classList.add('hidden')
  } else {
    firstNameErrorMessage.classList.remove('hidden')
  }
  return isFirstNameValid
}
```

HTML Message d'erreur

```html
<div class="first-name-error-message hidden">
  Veuillez entrer 2 caractères ou plus pour le champ du prénom.</div>
</div>
```

Le champ Prénom / nom a un minimum de 2 caractères

Function checkFirstName() - Function checkLastName()

const isFirstNameValid = firstNameInput.value.length > 2

Je verifie qu'il y a au moins deux caractères saisis dans les champs inputs first name & last name avec la propriété value.length > 2

If / else statement

Si la condition est vrai, la classe hidden s'applique et aucun message ne s'affiche et return isFirstNameValid sera renvoyé dans la fonction appelante.

Sinon la classe hidden se désactive et le message d'erreur s'affiche

```
// Fonction validation nom
function checkLastName () {
  const isLastNameValid = lastNameInput.value.length > 2

  if (isLastNameValid) {
    lastNameErrorMessage.classList.add('hidden')
  } else {
    lastNameErrorMessage.classList.remove('hidden')
  }
  return isLastNameValid
}
```

```
// Fonction validation prénom
function checkFirstName() {
  const isFirstNameValid = firstNameInput.value.length > 2

  if (isFirstNameValid) {
    firstNameErrorMessage.classList.add('hidden')
  } else {
    firstNameErrorMessage.classList.remove('hidden')
  }
  return isFirstNameValid
}
```

L’adresse est valide

Function checkEmailInput()

const re

J’utilise une regex pour vérifier que l’utilisateur entre un format valide d’adresse email

const isEmailValid = re.test(String(emailInput.value).toLowerCase())

Je test ma regex et je lui applique la méthode .toLowerCase() pour être en caractère minuscule.

If / else statement

Je vérifie si la condition définie dans isEmailValid est vrai, Si oui, la classe hidden s’applique et aucun message n’apparaît et return isEamilValid sera renvoyé dans la fonction appelante.

Sinon la classe hidden se désactive et le message d’erreur s’affiche

```
// Fonction validation adresse email
function checkEmailInput() {
  const re = /^(([^<>()[\]\\.,;:\s@"]+(\.[^<>()[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|
  const isEmailValid = re.test(String(emailInput.value).toLowerCase())

  if (isEmailValid) {
    emailInputErrorMessage.classList.add('hidden')
  } else {
    emailInputErrorMessage.classList.remove('hidden')
  }
  return isEmailValid
}
```

Une date est saisie dans le calendrier

Function checkCalendarInput()

const isCalendarValid = calendarInput.value

Je vérifie si l'utilisateur a selectionné une date dans le calendrier avec la propriété .value

If / else statement

Si la condition est vrai, la classe hidden s'applique et aucun message n'apparaît

return isCalendarValid sera renvoyé dans la fonction appelante

Sinon la classe hidden se désactive et le message d'erreur s'affiche

```
// Fonction validation date de calendrier
function checkCalendarInput(){
  const isCalendarValid = calendarInput.value;

  if (isCalendarValid){
    calendarErrorMessage.classList.add('hidden')
  } else {
    calendarErrorMessage.classList.remove('hidden')
  }
  return isCalendarValid
}
```

Nombre de concours, une valeur numérique est saisie.

Function checkTournamentNumberInput

Dans ma variable IsNumberValid je pose deux conditions séparées par un opérateur logique && (et)
Pour être vrai les deux conditions doivent être remplies.

Dans la première condition, j'utilise la fonction isNaN,
isNaN renvoie true quand la valeur n'est pas un nombre.
Mais je veux que la condition soit vrai, alors j'utilise l'opérateur logique ! (différent de)

Quand l'utilisateur rentre un nombre, isNaN devient false mais avec !, isNumberValid devient vrai
ParseInt transforme mon string en number

Dans la deuxième condition, j'indique que la valeur doit être < 99

Si mes conditions sont vrai alors aucun message d'erreur n'apparaît et return sera renvoyé

```
// Fonction validation nombre de tournois
function checkTournamentNumberInput(){
  const isNumberValid = !isNaN(parseInt(tournamentNumberInput.value)) && tournamentNumberInput.value < 99

  if(isNumberValid) {
    tournamentNumberErrorMessage.classList.add('hidden')
  } else {
    tournamentNumberErrorMessage.classList.remove('hidden')
  }
  return isNumberValid
}
```

Un bouton radio est sélectionné

Function checkCityLocation()

J'initialise ma variable isCityLocationValid sur false

J'utilise une boucle for car en HTML, citiesLocation est une collection d'objets

Dans ma boucle, j'initialise ma varibale i à 0; la condition est :

Tant que i < longueur total d'objets dans citiesLocation, j'incrémente de un.

A l'intérieur de ma loop, je pose une condition, dès que la loop rencontrera un élément checked.

On sortira de la boucle isCityLocationValid passera à true

La conditon est vrai et return sera renvoyé

Loop for

```
// Fonction radio input
function checkCityLocation() {
  let isCityLocationValid = false

  for (let i = 0; i < citiesLocation.length; i++) {
    if (citiesLocation[i].checked) {
      isCityLocationValid = true
      citiesLocationErrorMessage.classList.remove('hidden')
    } else {
      citiesLocationErrorMessage.classList.add('hidden')
    }
  }
  return isCityLocationValid
}
```

Collection HTML

```
<div class="formData citiesLocation">
  <input class="checkbox-input" type="radio" name="location" value="New York" id="location1"/>
  <label class="checkbox-label" for="location1"><span class="checkbox-icon"></span>New York</label>

  <input class="checkbox-input" type="radio" name="location" value="San Francisco" id="location2"/>
  <label class="checkbox-label" for="location2"><span class="checkbox-icon"></span>San Francisco</label>

  <input class="checkbox-input" type="radio" name="location" value="Seattle" id="location3"/>
  <label class="checkbox-label" for="location3"><span class="checkbox-icon"></span>Seattle</label>

  <input class="checkbox-input" type="radio" name="location" value="Chicago" id="location4"/>
  <label class="checkbox-label" for="location4"><span class="checkbox-icon"></span>Chicago</label>

  <input class="checkbox-input" type="radio" name="location" value="Boston" id="location5"/>
  <label class="checkbox-label" for="location5"><span class="checkbox-icon"></span>Boston</label>

  <input class="checkbox-input" type="radio" name="location" value="Portland" id="location6"/>
  <label class="checkbox-label" for="location6"><span class="checkbox-icon"></span>Portland</label>
  <div class="citiesLocation-error-message hidden">Veuillez sélectionner une ville</div>
</div>
```

la case des conditions générales est cochée

function checkconditionsInput()

Ici j'initialise ma variable isConditionvalid sur false
Dans cette condition, je vérifie si la checkbox est cochée.

Si la condition est vrai isConditionsValid passera à true
Aucun message d'erreurs ne s'affichera et return renverra isConditionsValid = true quand la fonction sera appelée

```
// Fonction conditions de vente
function checkconditionsInput(){
  let isConditionsValid = false

  if(requiredCheckbox.checked){
    isConditionsValid = true
    requiredCheckboxErrorMessage.classList.add('hidden')
  } else {
    requiredCheckboxErrorMessage.classList.remove('hidden')
}
return isConditionsValid
}
```

Le formulaire est valide quand l'utilisateur clique sur "Submit"

J'écoute l'événement "submit" avec addEventListener.
Au click sur "submit" je pose une condition qui vérifie si toutes les fonctions inputs sont valides.

C'est donc ici que les fonctions sont appelées et que les valeurs de return sont renvoyées pour passer la validation. Si une fonction n'est pas valide, elle ne renvoie pas le return, le formulaire ne passe donc pas la validation.

Dans la fonction on call, je paramètre (event) et déclare la méthode preventDefaut, pour empêcher le comportement par défaut de s'activer quand on clique sur un bouton.

Si ma condition est vrai, la console affiche "tout est ok"
et la modal se ferme avec modalbg.style.display ="none".

```
// J'écoute l'évenement submit pour valider mon formulaire
reservationForm.addEventListener('submit', function(event) {
  event.preventDefault();

  if (checkFirstName() && checkLastName() && checkEmailInput() && checkCityLocation() && checkTournamentNumberInput() && checkconditionsInput() && checkCalendarInput()){
    console.log("Tout est ok")
    modalbg.style.display = "none"

  } else {
    console.log("Il y a un problème")
  }

})
```

Issue 3 : Ajouter validation ou messages d'erreurs

Input HTML id="first"

```html
<div class="formData">
  <label for="first">Prénom</label><br>
  <input class="text-control" type="text" name="first" id="first" minlength="2"/><br>
  <div class="first-name-error-message hidden">Veuillez entrer 2 caractères ou plus pour le champ du prénom.</div>
</div>
```

Selector CSS .first-name-error-message

```css
.first-name-error-message, .last-name-error-message, .number-error-message,
.email-error-message, .error-message-birthdate, .citiesLocation-error-message, .error-message-conditions {
  font-size: 0.4em;
  color: #e54858;
  margin-top: 7px;
  margin-bottom: 7px;
  text-align: right;
  transition: 0.3s;
}
```

CSS .hidden

```css
.hidden {
  display: none;
}
```

ClassList.remove('hidden')

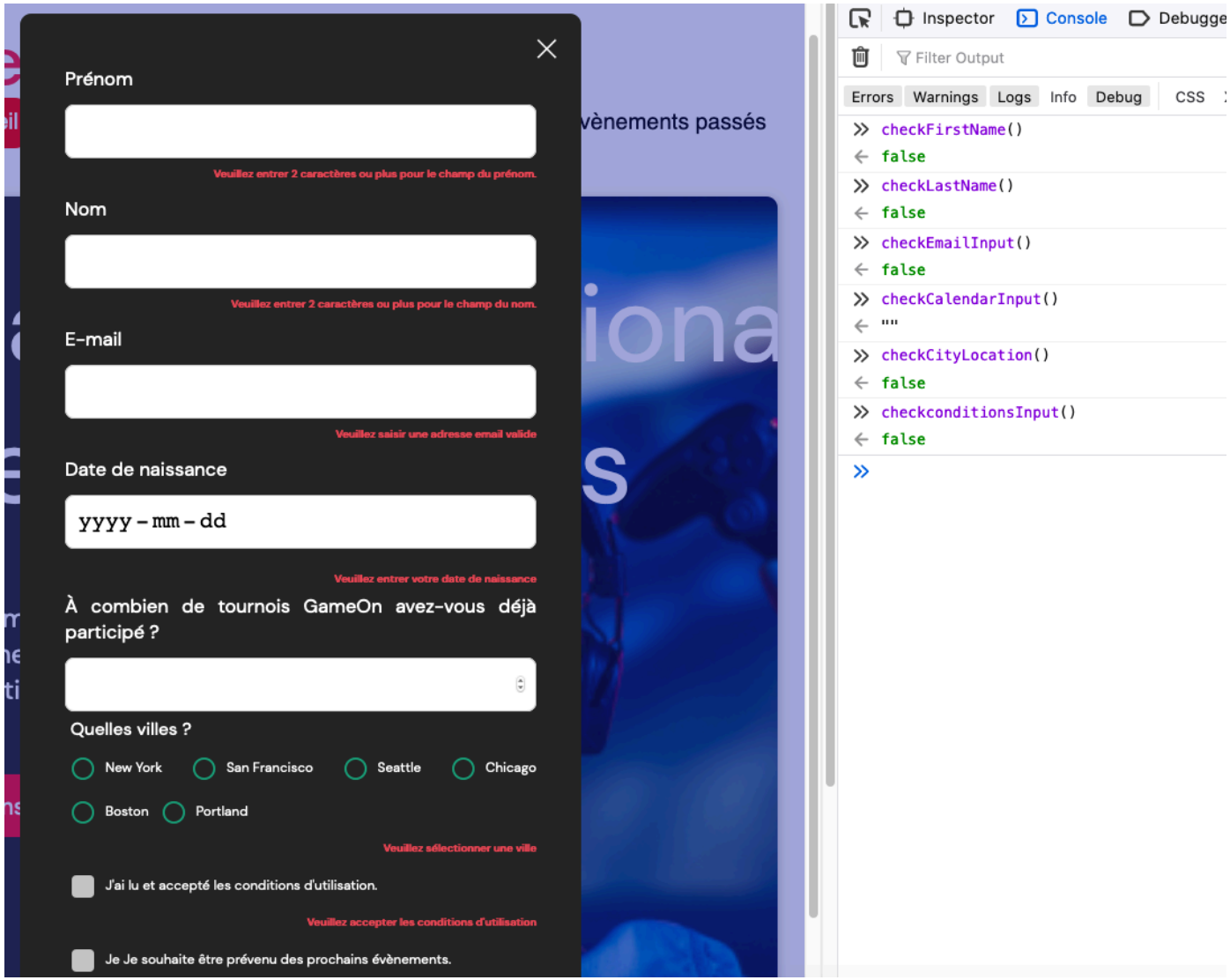
```js
// Fonction validation prénom
function checkFirstName() {
  const isFirstNameValid = firstNameInput.value.length > 2

  if (isFirstNameValid) {
    firstNameErrorMessage.classList.add('hidden')
  } else {
    firstNameErrorMessage.classList.remove('hidden')
  }
  return isFirstNameValid
}
```

Variable JS firstNameErrorMessage

```js
// Je déclare mes variables
const firstNameInput = document.querySelector('#first')
const firstNameErrorMessage = document.querySelector('.first-name-error-message')
```

Voici tous les messages d’erreurs déclenchés dans la modal grâce à la console en vérifiant chaque fonction une à autre. Aucune condition n’est remplie



Issue 4 : Ajouter la confirmation quand l'envoi est réussi

Après avoir vérifié que le formulaire est valide au click sur "submit" :

Le formulaire se masque avec reservationForm.style.display = "none"

Le message de confirmation apparaît avec successMessage.style.display = "flex"

Pour fermer la fenêtre de succès, j'écoute l'évenement "click" sur le bouton fermer

La modal se ferme

JS

```
// J'écoute l'évenement submit pour valider mon formulaire
reservationForm.addEventListener('submit', function(event) {
  event.preventDefault();

  if (checkFirstName() && checkLastName() && checkEmailInput() && checkCityLocation()
  && checkTournamentNumberInput() && checkconditionsInput() && checkCalendarInput()){
    console.log("Tout est ok") // Je vérifie dans ma console
    reservationForm.style.display = "none"  // Je masque le formulaire
    successMessage.style.display = "flex"  // J'affiche le message de validation

  } else {
    console.log("Il y a un problème") // Je vérifie dans ma console
  }
})

// J'écoute l'évenement click pour fermer la modal
closeButton.addEventListener('click', function() {
modalbg.style.display = 'none'
})
```

CSS

```
.reservation-form {
  display: block;
}

.success-message {
  display: flex;
  flex-direction: column;
  justify-content: center;
  height: 70vh;
}
```

```
<div class="success-message">
  <div class= "message-validation">
    <p>Merci !</p>
    <p>Votre formulaire a bien été envoyé</p>
  </div>
  <button class="btn-close">Fermer</button>
</div>
```

HTML

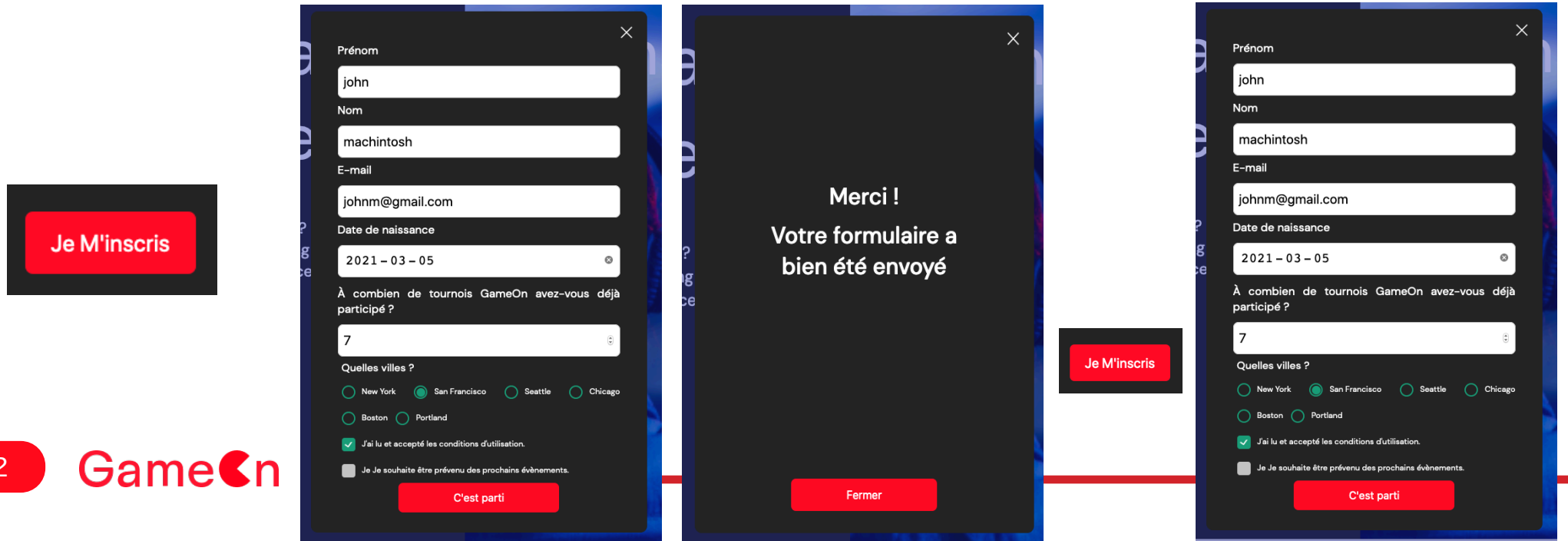
```
<form class="reservation-form" action="index.html" method="get">
```

Si je veux réouvrir à nouveau le formulaire

La modal s'ouvrira avec modal.style.display ="block".
Mais je dois aussi réafficher le formulaire avec reservation.style.display ="block" qui a été masqué à la soumission du formulaire

Ensuite je remasque le message de validation avec successMessage.style.display = "none"

```
// launch modal form
function launchModal() {
  modalbg.style.display = "block"   // J'affiche  le formulaire
  reservationForm.style.display = "block"   // Je réactive le formulaire après sa fermeture dès que je relance la modal
  successMessage.style.display = "none"   // Je masque le message de validation
}
```



Issue 5 : Tests manuels

Conclusion

Challenge & difficultés

- Javascript est un langage qui m'a demandé plus de temps à sa compréhension que HTML, CSS et SASS
- Comment bien relier les élements (objets) entre eux de façon logique
- Bien comprendre les propriétés et méthodes
- Les fonctions sont assez complexes à comprendre surtout quand il y a des paramètres et des arguments à gérer

MERCI!