



Projet 4

Créer une page landing avec Javascript



Sommaire

Presentation du projet

1. Consignes & éléments fournis
2. Issues : Travail à réaliser

Projet JS - Modal & Issues

3. Issue 1 : Fermer la modal
4. Issue 2 : Implémenter les entrées du formulaire
5. Issue 3 : Ajouter validation ou messages d'erreurs
6. Issue 4 : Ajouter la confirmation quand l'envoi est réussi
7. Issue 5 : Tests manuels

Conclusion

8. Challenges & difficultés
- 



Présentation

Projet

Projet : Créer une page landing avec JS

Éléments fournis

- Maquettes HTML et CSS de la page d'accueil et de la “modal”.
- Quelques lignes de code en Javascript.
- Travail de Jason et consignes sur Github dans un repo Gameon-website-fr

Consignes

- Travailler sur un repo GitHub forké
- Utiliser des fichiers séparés pour le HTML, le CSS et le JavaScript
- Ajouter le code JavaScript manquant pour que le formulaire soit pleinement fonctionnel
- Commenter le code
- Tester manuellement les fonctionnalités, les entrées de formulaire et l’affichage responsive.

Issues : Travail à réaliser

Issue 1 : Fermer la modal

Issue 2 : Implémenter les entrées du formulaire :

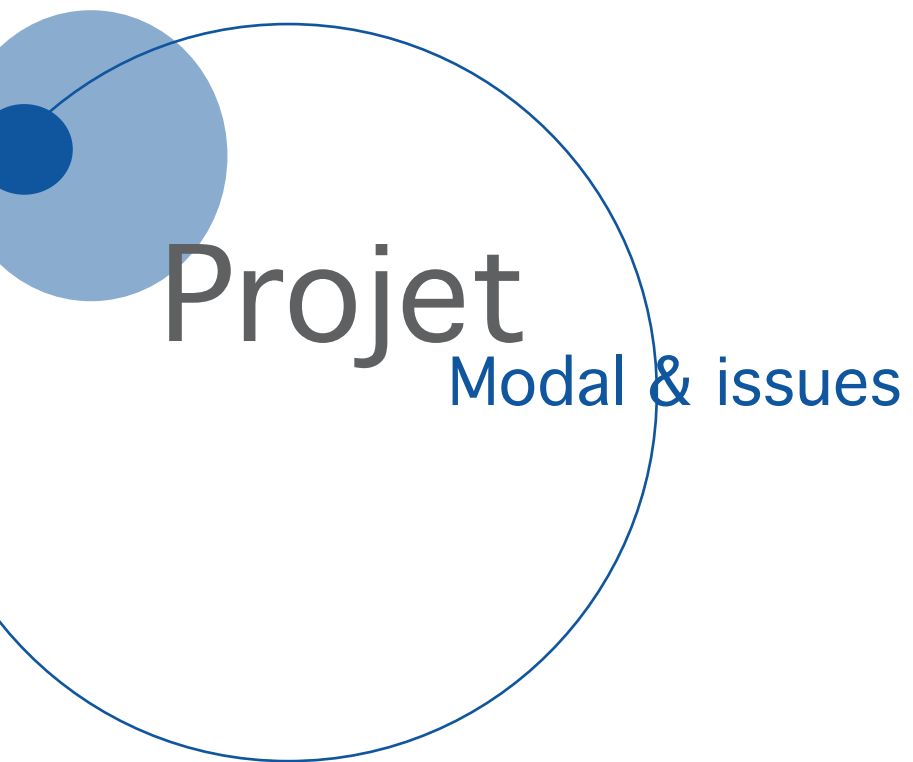
- Lier les labels aux entrées dans HTML avec les attributs “for” et “id”
- Le formulaire doit être valide quand l'utilisateur clique sur “Submit”
- Les champs Prénom / Nom ont un minimum de 2 caractères / ne sont pas vides.
- L'adresse électronique est valide
- Une date est sélectionnée dans le calendrier
- 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

Issue 1 : Fermer la modal

- HTML : class="close" représente l'élément de fermeture de la modal
- Const close : sélection du noeud HTML close avec la méthode `querySelector()`
- Const modalbg : représente la modal (".bground")
- J'écoute un événement avec `addEventListener`
- Au click sur close => la modal se masque

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

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

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

close.addEventListener('click', () => {
  modalbg.style.display = "none"
});
```

Issue 2 : Implémenter les entrées du formulaire

Attributs “for” et “id”

```
<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

```
<div class="formData">
  <label for="birthdate">Date de naissance</label><br>
  <input class="text-control" type="date" name="birthdate" id="birthdate"/><br>
  <div class="error-message-birthdate hidden">Veuillez entrer votre date de naissance</div>
</div>
```

Je fais cela pour tous les inputs de mon formulaire

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

```
<div class="formData">
  <label for="quantity">À combien de tournois GameOn avez-vous déjà participé ?</label><br>
  <input type="number" class="text-control" name="quantity" id="quantity">
  <div class="number-error-message hidden">Veuillez entrer un nombre</div>
</div>
```


Les champ Prénom / Nom ont un minimum de 2 caractères

`const isFirstNameValid`

Au moins deux caractères sont saisis dans l'input first name avec la propriété `value.length > 2`

If / else statement

Condition = true

- Le message d'erreur se masque avec la propriété => `.classList.add('hidden')`
- return renvoie `isFirstNameValid => true`

Condition = false

- Le message d'erreur apparaît avec la propriété => `.classList.remove('hidden')`
- return renvoie `isFirstNameValid => false`

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

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

Variables input & message d'erreur

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

HTML Message d'erreur

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

CSS

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

L'adresse email est valide

const re

- J'assigne une regex à ma variable
- cette regex représente le format email à respecter

const isValidEmail

- J'applique la regex à l'input de l'email et la vérifie

if / else

- Condition respectée => pas de message d'erreur => return true
- Condition non respectée => message d'erreur => return false

```
// 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 isValidEmail = re.test(String(emailInput.value).toLowerCase())

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

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

Une date est saisie dans le calendrier

const isCalendarValid

- Je vérifie si l'input du calendrier a une valeur saisie avec la propriété `.value.length > 0`

If / else statement

- Condition respectée => pas de message d'erreur => return true
- Condition non respectée => message d'erreur => return false

```
// Fonction validation date de calendrier
function checkCalendarInput(){
  const isCalendarValid = calendarInput.value.length > 0

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

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

Nombre de concours, une valeur numérique est saisie.

Const `isNumberValid` => Deux conditions séparées par un opérateur logique `&&`

Condition 1

- valeur n'est pas un nombre : `isNaN` => `true` .
- l'opérateur logique `!` : `isNaN` => `false`
- `parseInt` : `string` => `number`

Condition 2

- La valeur doit être `< 99`

If / else statement

Pour être vrai, mes deux conditions doivent être valides

```
// 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
}
```

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

Un bouton radio est sélectionné

const isCityLocationValid

Initialisation de la variable sur false

Loop for

- Parcourir le tableau dans citiesLocation

If / else

- Dès que la boucle rencontre un élément checked => on sort de la boucle
- La variable isCityLocationValid => true
- Aucun message ne s'affiche
- return isCityLocationValid => true

HTML

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.add('hidden')
      return isCityLocationValid
    } else {
      citiesLocationErrorMessage.classList.remove('hidden')
    }
  }
}
```

```
<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>
```

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

la case des conditions générales est cochée

Const isConditionvalid

- Initialisation de la variable sur false
- Je vérifie si la checkbox est cochée.

if / else statement

- Si la condition est vrai isConditionsValid => true
- Aucun message d'erreurs ne s'affiche et return => true

```
// 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
}
```

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

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 if / else

- J'appelle les fonctions de chaque inputs
- Si elles sont toutes valides => le formulaire passe la validation
- Si une seule est fausse => le formulaire ne peut pas être soumis

```
// J'écoute l'événement 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 les messages d'erreurs

Input 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>
```

Selectors 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;
}
```

ClassList.remove('hidden')

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

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

CSS .hidden

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

Variable JS firstNameErrorMessage

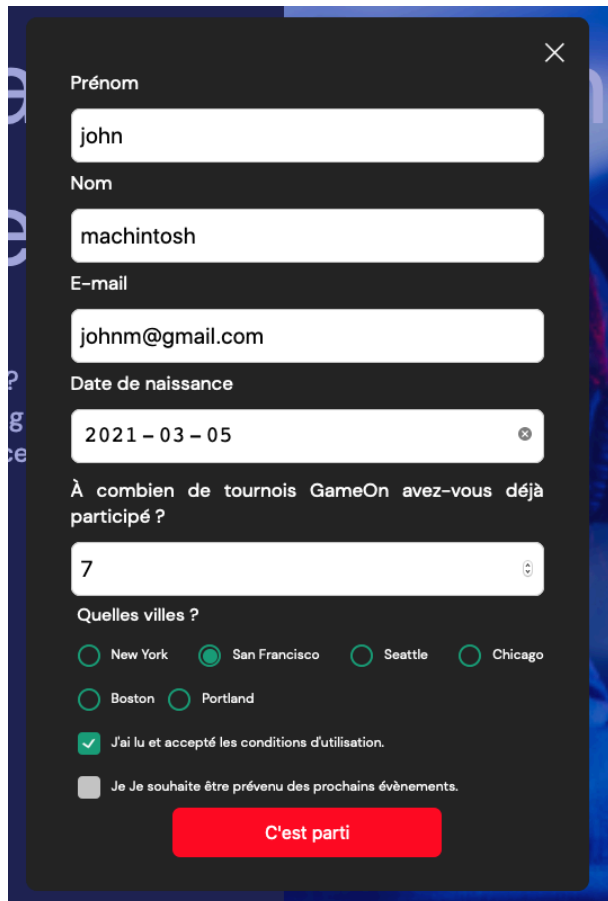
```
// Je déclare mes variables
const firstNameInput = document.querySelector('#first')
const firstNameErrorMessage = document.querySelector('.first-name-error-message')
```

The form contains the following fields and messages:

- Prénom**: Text input with error message "Veuillez entrer 2 caractères ou plus pour le champ du prénom."
- Nom**: Text input with error message "Veuillez entrer 2 caractères ou plus pour le champ du nom."
- E-mail**: Text input with error message "Veuillez saisir une adresse email valide"
- Date de naissance**: Text input with placeholder "yyyy-mm-dd" and error message "Veuillez entrer votre date de naissance"
- À combien de tournois GameOn avez-vous déjà participé ?**: Text input with a spinner and error message "Veuillez entrer un nombre"
- Quelles villes ?**: Radio buttons for New York, San Francisco, Seattle, Chicago, Boston, and Portland. Error message: "Veuillez sélectionner une ville"
- Conditions**: Two checkboxes: "J'ai lu et accepté les conditions d'utilisation." and "Je souhaite être prévenu des prochains événements."
- Buttons**: A red "C'est parti" button at the bottom.

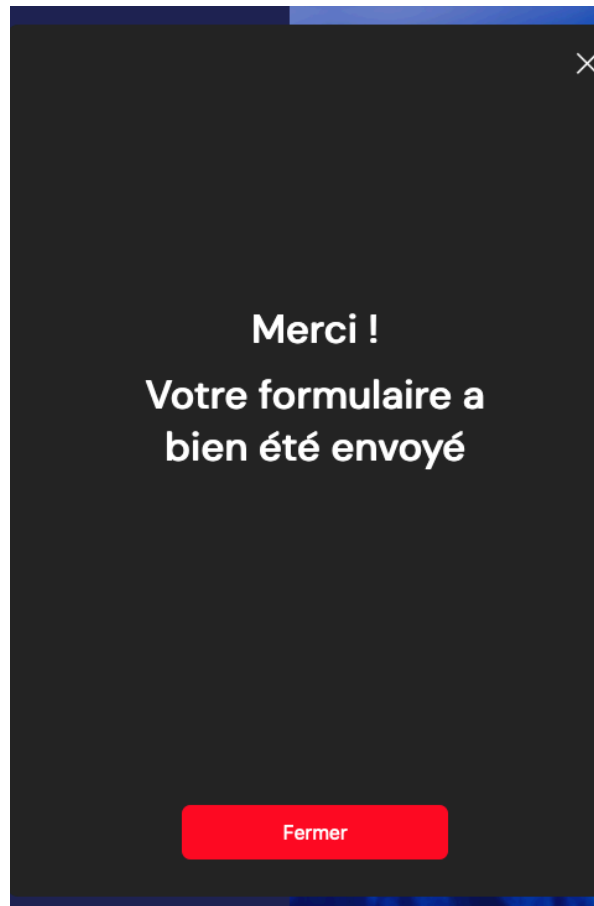
Issue 4 : Ajouter la confirmation quand l'envoi est réussi

Soumission modal

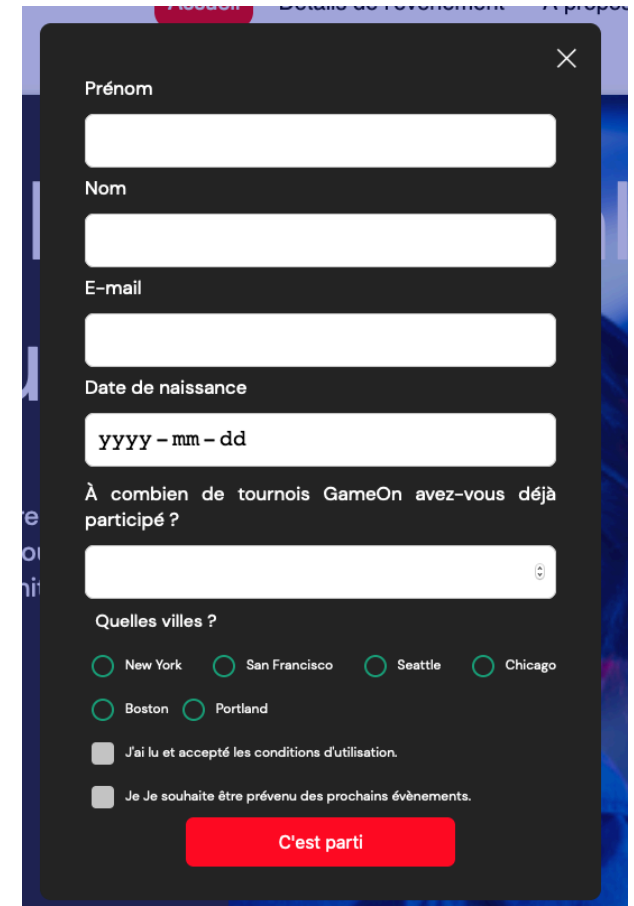


A dark-themed modal form for user registration. It contains several input fields: 'Prénom' (first name) with 'john', 'Nom' (last name) with 'machintosh', 'E-mail' with 'johnm@gmail.com', and 'Date de naissance' (date of birth) with '2021-03-05'. Below these is a question 'À combien de tournois GameOn avez-vous déjà participé ?' with a numeric input field containing '7'. A section titled 'Quelles villes ?' (Which cities?) has radio buttons for 'New York', 'San Francisco' (selected), 'Seattle', 'Chicago', 'Boston', and 'Portland'. At the bottom, there are two checkboxes: 'J'ai lu et accepté les conditions d'utilisation.' (checked) and 'Je souhaite être prévenu des prochains événements.' (unchecked). A red button labeled 'C'est parti' is at the bottom right.

Message de succès



Réinitialisation modal



A dark-themed modal form for resetting the registration. It contains empty input fields for 'Prénom', 'Nom', 'E-mail', and 'Date de naissance' (placeholder 'yyyy-mm-dd'). Below is the same question 'À combien de tournois GameOn avez-vous déjà participé ?' with an empty numeric input field. The 'Quelles villes ?' section has radio buttons for 'New York', 'San Francisco', 'Seattle', 'Chicago', 'Boston', and 'Portland'. At the bottom, there are two unchecked checkboxes: 'J'ai lu et accepté les conditions d'utilisation.' and 'Je souhaite être prévenu des prochains événements.' A red button labeled 'C'est parti' is at the bottom right.

Soumission modal

- Masquage du formulaire avec la propriété `style.display = "none"`
- Affichage du message de succès avec la propriété `style.display = "flex"`

Fermeture message de succès

- Pour fermer la modal, j'écoute l'événement "click" sur le bouton fermer

JS

HTML

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

```
<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>
```

CSS

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

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

Réinitialisation de la modal

- Réinitialisation de toutes les inputs
- Réaffichage du formulaire avec la propriété `style.display = "block"`
- Remasquage du message de succès avec la propriété `style.display = "none"`

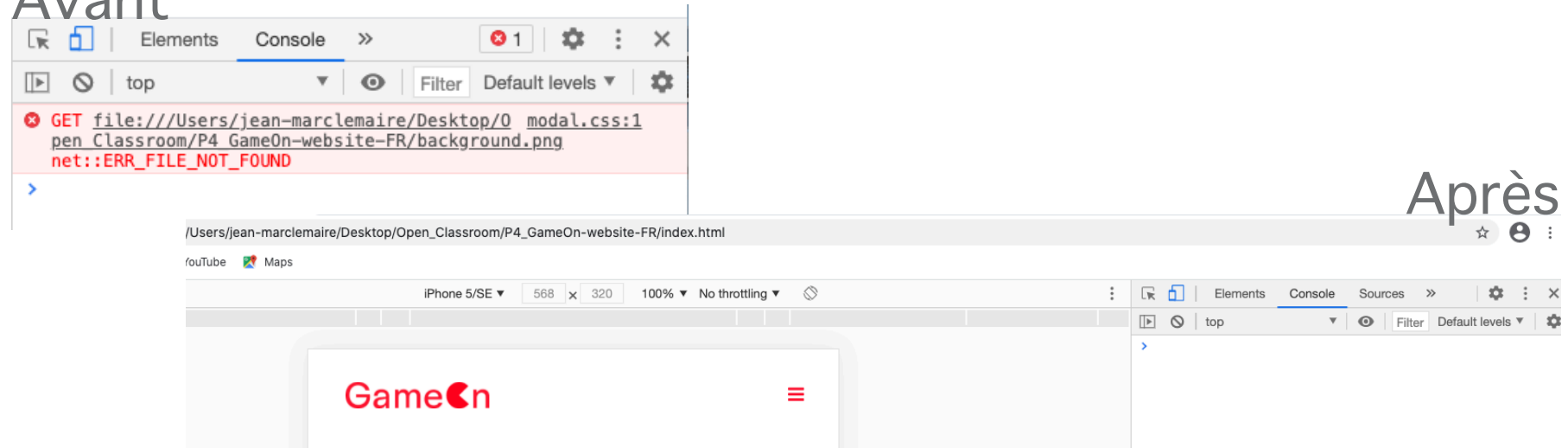
```
// Reset form
function cleanForm() {
  firstNameInput.value = ""
  lastNameInput.value = ""
  emailInput.value = ""
  calendarInput.value = ""
  tournamentNumberInput.value = ""
  for (let i = 0; i < citiesLocation.length; i++) {
    citiesLocation[i].checked = false
  }
  requiredCheckbox.checked = false
}
```

```
// 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 - erreur

Erreur Console Google Chrome : background.png (file not found)

Avant



Après

Avant

```
body {  
  margin: 0;  
  display: flex;  
  flex-direction: column;  
  background-image: url("background.png");  
  font-family: var(--font-default);  
  font-size: 18px;  
  max-width: 1300px;  
  margin: 0 auto;  
}
```

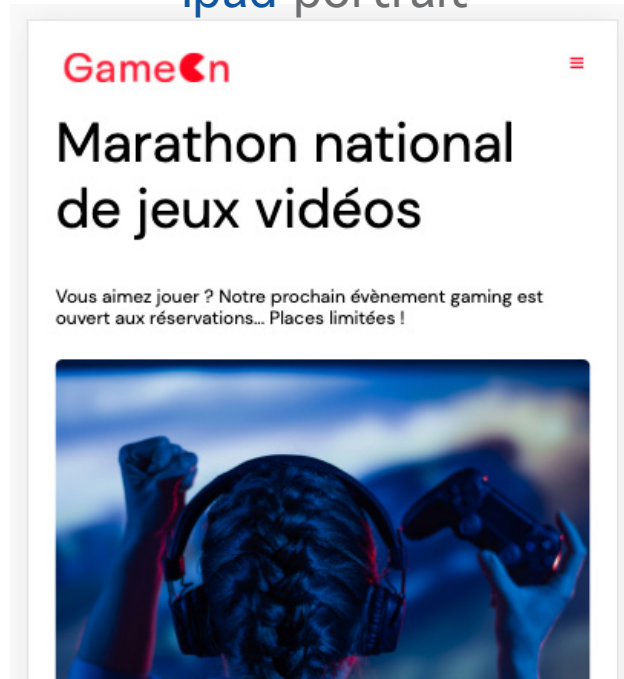
Après

```
body {  
  margin: 0;  
  display: flex;  
  flex-direction: column;  
  font-family: var(--font-default);  
  font-size: 18px;  
  max-width: 1300px;  
  margin: 0 auto;  
}
```

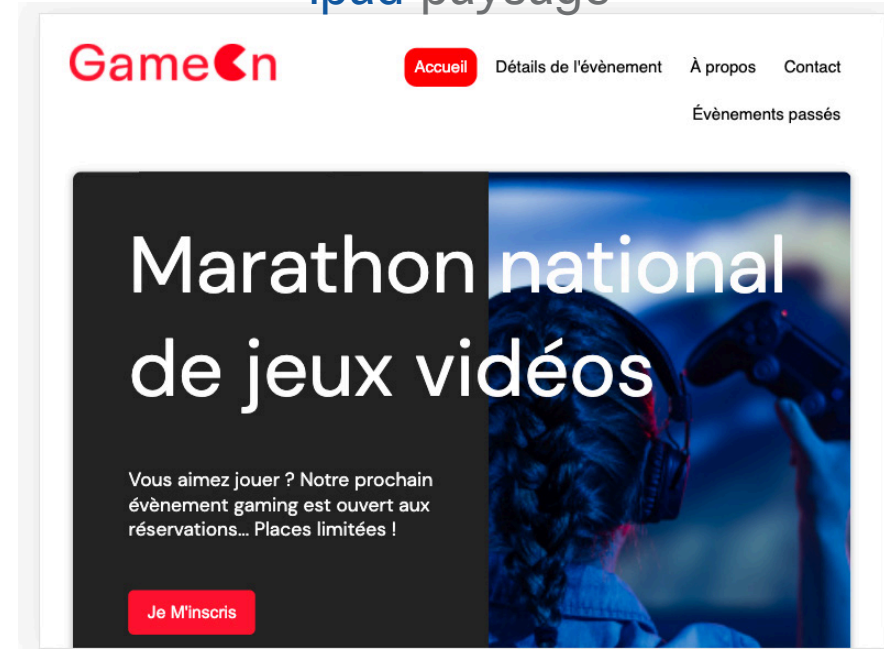
Iphone portrait



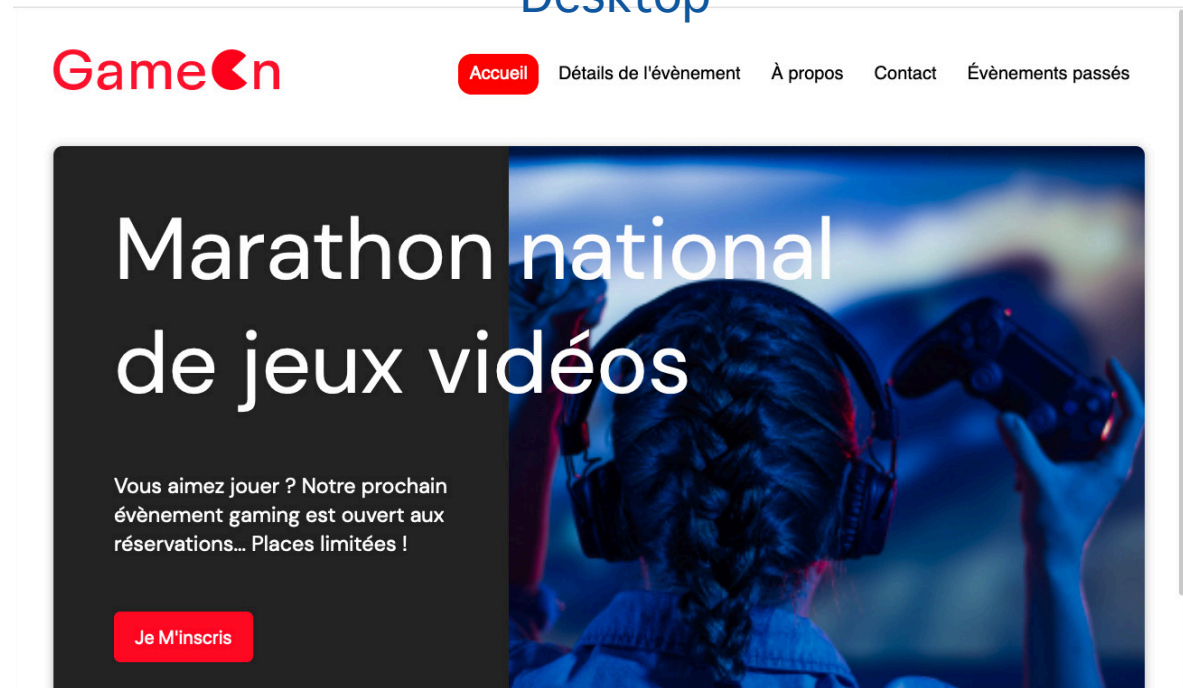
Ipad portrait



Ipad paysage



Desktop



Iphone paysage



Code d'origine

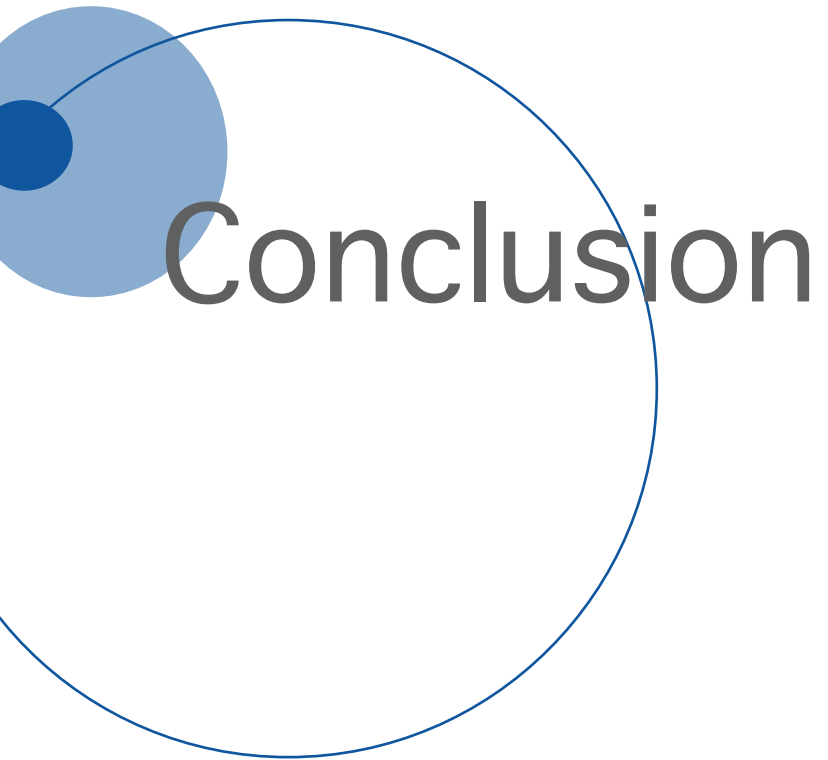
```
71 @media screen and (max-width: 768px) {
72   .topnav a {display: none;}
73   .topnav a.icon {
74     float: right;
75     display: block;
76   }
77 }
78
79 @media screen and (max-width: 768px) {
80   .topnav.responsive {position: relative;}
81   .topnav.responsive .icon {
82     position: absolute;
83     right: 0;
84     top: 0;
85   }
86   .topnav.responsive a {
87     float: none;
88     display: block;
89     text-align: left;
90   }
91 }
92
93
94
95 @media screen and (max-width: 540px) {
96   .topnav a {display: none;}
97   .topnav a.icon {
98     float: right;
99     display: block;
100     margin-top: -15px;
101   }
102 }
103
```

Refactorisation

```
78
79 @media screen and (max-width: 768px) {
80   .main-navbar {
81     flex-direction: column;
82     flex-wrap: nowrap;
83   }
84   .topnav a {display: none;}
85   .topnav a.icon {
86     display: flex;
87     align-self: flex-end;
88     display: block;
89     color: #fe142f;
90   }
91   .topnav.responsive {position: relative;}
92   .topnav.responsive .icon {
93     position: absolute;
94     right: 0;
95     top: 0;
96   }
97   .topnav.responsive a {
98     float: none;
99     display: block;
100     text-align: left;
101   }
102   .header-logo img {
103     width: 30vw;
104     margin-top: 15px;
105     margin-left: 20px;
106   }
107   .btn-signup-desktop {
108     display: none;
109     outline: none;
110     text-transform: capitalize;
111     font-size: 1.3rem;
112     padding: 15px 23px;
113     margin: 0;
114     margin-top: 59px;
115   }
116 }
```

```
17
18 @media screen and (max-width: 540px) {
19   .topnav a {display: none;}
20   .topnav a.icon {
21     margin-top: 0px;
22     color: #fe142f;
23   }
24   .hero-headline {
25     font-size: 2.5rem;
26   }
27 }
```

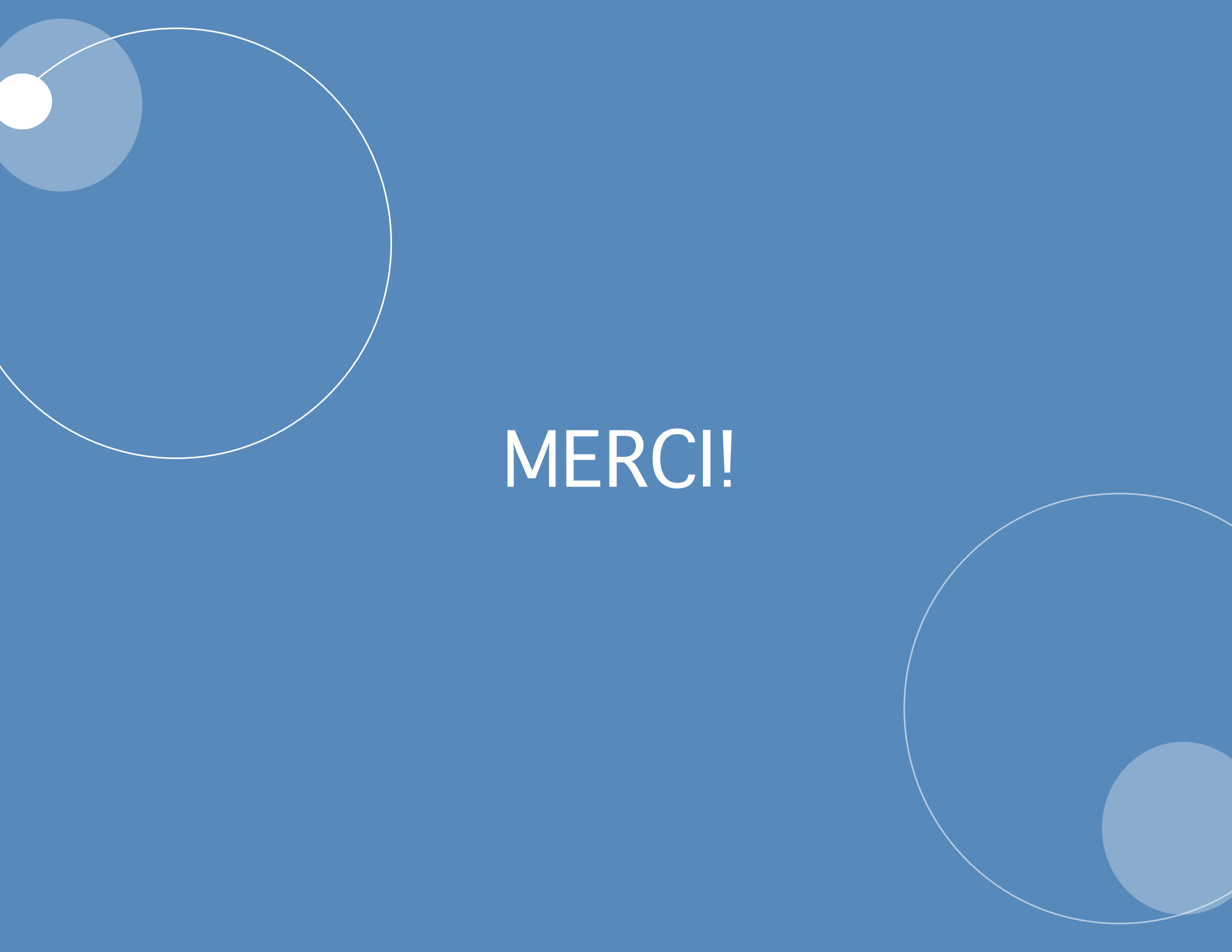
```
.main-navbar {
  display: flex;
  justify-content: flex-end;
  flex-wrap: wrap;
}
```



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 éléments entre eux de façon logique
- Bien comprendre les propriétés et méthodes
- Comprendre la logique javascript
- Se familiariser avec le vocabulaire javascript



MERCI!