



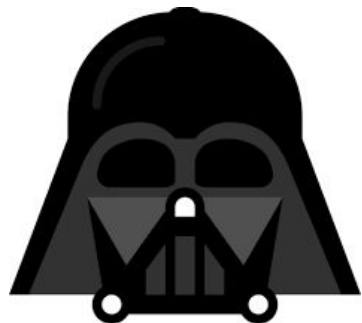
# *Brief Star Wars*

*Call Api, le retour*

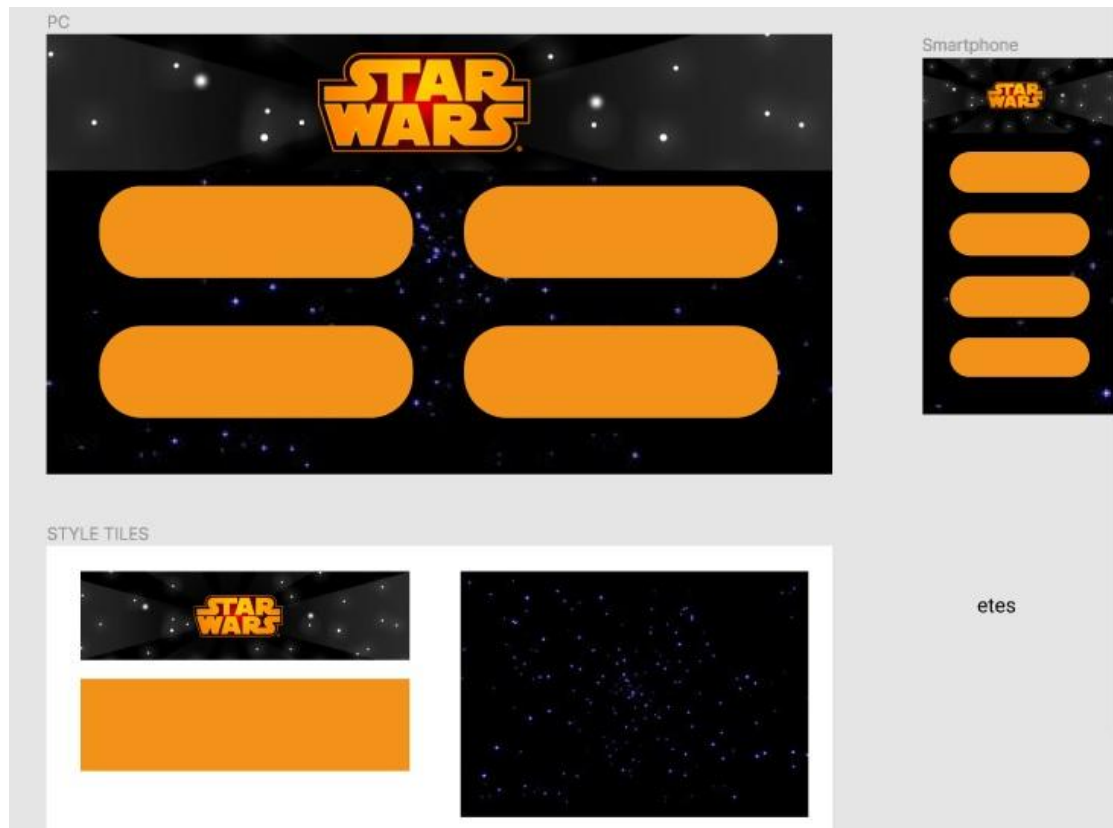
*Il y a bien longtemps dans une galaxie lointaine, très lointaine...  
genre vers Montreuil...*

*C'est un temps d'incertitude. De grands présages s'emparent de la  
galaxie avec l'influence croissante de Premier Ordre, mais une lueur  
d'espoir apparaît grâce aux force héroïques de la Résistance... Nos  
héros : Mohamed, Paul-Emmanuel, Thomas, Cécile....*

# Maquette Figma

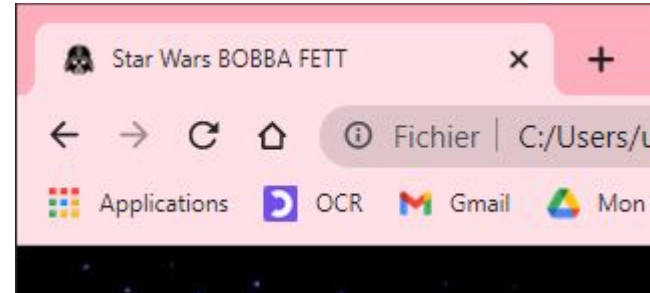


[Lien vers la maquette](#)



# Figma, maquettage

- Extension chromatique des couleurs du logo pour l'harmonie visuelle du site  
=> thème en noir et orange, fond d'écran noir avec étoiles
- Arrondi du titre reporté sur les angles des cadres et des photos  
Photos avec thème orangé également
- Ajout d'un favicon à l'onglet



# Github organisation

=> Création d'un Main que nous avons partagé, incluant une page html et un css commun, afin d'avoir une base de travail avec des balises communes

=> Création de 4 branches (films/personnages/planètes/véhicules) individuelles

=> Création d'une autre branche pour tester (dev) sur laquelle on a mergé

# Code

## HTML :

Nous sommes partis sur un HTML de base.

```
<div class="box" id="box">
  <h2></h2>

  <div class="data_base" id="data_base">
    <div class="data_second" id="data_second">
    </div>
    <div class="data_three" id="data_three">
    </div>
  </div>
</div>
```

# Premier Call API

Constante de l'adresse URL de l'API (sans le numéro de page).

Constante de l'élément HTML dans laquelle on va insérer l'affichage en fonction des retours Call API.

Initialisation d'un tableau vide allDats pour y insérer toutes nos données.

Premier appel API pour récupérer le nombre de pages.


Ensuite on boucle sur le nombre de pages pour récupérer TOUTES les données qu'on stocke dans le tableau allDats[[]].

```
async function getData(api){  
  // on récupère le nombre de pages de personnages  
  const resp = await fetch(api);  
  const d = await resp.json();  
  let nbPages = Math.ceil(d.count/10);  
  
  // on boucle sur le nombre total de pages  
  for(i=1; i <= nbPages; i++){  
  
    const response = await fetch(api + i);  
    const data = await response.json();  
  
    // les résultats disponibles en retour de l'appel  
    API  
    let results = data.results;  
  
    // on boucle sur tous les index donc tous les  
    personnages  
    for (let index = 0; index < results.length; index++)  
    {  
      let objetPerso= {};  
  
      allDats.push(objetPerso={  
        "name" : results[index].name,  
        "height" : results[index].height,  
        "mass" : results[index].mass,  
        "hair_color" : results[index].hair_color,  
        "skin_color" : results[index].skin_color,  
        "eye_color" : results[index].eye_color,  
        "birth_year" : results[index].birth_year,  
        "gender" : results[index].gender,  
        "films" : results[index].films,  
        "getDataFilm": false,  
        "datafilm" : [] ,  
        "planete" : results[index].homeworld,  
        "getHomeWorld" : false,  
        "homeworld" : []  
      });  
    }  
  }  
}
```

# Affichage des Titres dans un premier temps et des détails au click

1 fonction `afficherNom()` qui récupère les données de `allDatas`

1 fonction `afficherDétails()` qui se déclenchera au click sur le Titre



```
affichageNoms(allDatas);
```

```
affichageDetails();
```



Dans afficherDetails()

l'écouteur d'événement sur les <h2> permet le second et le troisième appels API pour récupérer les noms des films dans lesquels ils apparaissent et le nom de la planète de naissance et leur affichage ainsi que l'affichage des données de base déjà existantes dans le tableau allDatas.

```
function affichageDetails(){
  let buttons = document.querySelectorAll('h2');

  for (let i = 0; i < buttons.length; i++) {
    buttons[i].addEventListener("click", function(e) {
      const DIV = e.target.nextSibling.nextElementSibling;

      for(let index = 0 ; index < allDatas.length ; index++){
        if(allDatas[index].name === e.target.innerText){
          getPromiseFilms(allDatas[index].films).then(resp =>{

            getPromisePlanetes(allDatas[index].planete).then(respPlanete=>{
              DIV.innerHTML =
                `
                <div class="data_base">
                  <ul>
                    <li>La taille : ${allDatas[index].height}</li>
                    <li>Le poids : ${allDatas[index].mass}</li>
                    <li>La couleur des cheveux : ${allDatas[index].hair_color}</li>
                    <li>La couleur de la peau : ${allDatas[index].skin_color}</li>
                    <li>La couleur des yeux : ${allDatas[index].eye_color}</li>
                    <li>La date de naissance : ${allDatas[index].birth_year}</li>
                    <li>Le genre : ${allDatas[index].gender}</li>
                  </ul>
                </div>
                <div class="data_second">
                  <h2>Apparition Films</h2>
                  <p> ${resp.join(" , ")} </p>
                </div>
                <div class="data_three">
                  <h2>Nom de la Planète de naissance</h2>
                  <p> ${respPlanete} </p>
                </div>
                `
            });
          });
        }
      }
    });
  }
}
```

## *Des promesses, toujours des promesses...*

Nous avons inclus la notion d'asynchronie afin de gérer les latences d'affichage.

La “promesse” permet la récupération de toutes les données avant leur livraison à l'écran.

La notion de Promesse et de Fetch sur l'API se lient naturellement car l'appel à une api induit une notion de temporalité liée aux serveurs, aux pings, aux affichages, etc.

Les fonctions getPromise

permettent le second et troisième appel  
api.

On récupère les apparitions dans les films  
et planète de naissance (stockés  
uniquement en URL API dans le tableau  
allDatas).

```
async function getFilms(dataFilms){
  const responseFilms = await fetch(dataFilms);
  const dataFilm = await responseFilms.json();
  return dataFilm.title;
}

async function getPlanete(dataPlanetes){
  const responsePlanete = await fetch(dataPlanetes);
  const dataPlanete = await responsePlanete.json();
  return dataPlanete.name;
}

async function getPromiseFilms(datafilms){
  let arrayFilms = [];
  for(let i = 0; i < datafilms.length ; i++){
    arrayFilms.push(getFilms(datafilms[i]));
  }
  return Promise.all(arrayFilms);
}

async function getPromisePlanetes(dataplanetes){
  let dataPlanete = [];
  dataPlanete.push(getPlanete(dataplanetes));

  return Promise.all(dataPlanete);
}
```

# Accordéon

Élément dynamique qui permet d'optimiser la place sur le site.

Utilisation d'événements, de boucles et d'un toggle qui permet lors du clic de se déployer ou de remonter

```
const LIST_BUTTON = document.querySelectorAll('h2');

for(let i = 0; i < LIST_BUTTON.length; i++){
  LIST_BUTTON[i].addEventListener('click',
function() {
  let go = this.nextElementSibling
  go.classList.toggle('active');
});
}
```

# Barre de recherche

Création d'une barre de recherche à l'aide de l'événement

**addEventListener.**

L'utilisation d'une boucle FOR et d'une condition pour spécifier la recherche :

on met les deux comparaisons en petit caractère pour rendre la comparaison insensible à la casse.

Et si indexOf renvoie un chiffre différent à -1, c'est que la string renseignée dans l'input est bien présente dans le titre en cours dans la boucle.

Donc on affiche le résultat trouvé.

```
const text = document.getElementById("searchBar");
const button = document.getElementById("search");

button.addEventListener('click' , (e) => {
    megaBox.innerHTML = "";

    for(let index = 0; index < bigData.length; index++){

        if(bigData[index].title.toLowerCase().indexOf(text.value.toLowerCase()) !== -1){

            megaBox.innerHTML =`
```

# STAR WARS

Retour d'expérience....

Quand tu  
doutes de toi,  
rappelle-toi  
que même un  
nain vert  
dyslexique  
à grandes  
oreilles  
a réussi à  
devenir un  
maître Jedi.

