

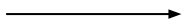
From HTML to React with NextJS

Initialisation du frontend du
projet The Good Corner





Objectifs



- Intégrer le code HTML
- Créer des composants React
- Ajouter du style avec CSS



Récupération du boilerplate

Un projet pré-configuré pour nos besoins



Récupération du boilerplate

- ➔ Récupère le fichier zip de ce repo et intègre-le dans le dossier de ton projet: https://github.com/karimmakhloufi/frontend_boilerplate

frontend_boilerplate Public

main 1 branch 0 tags

Go to file Add file <> Code

Karim Makhloufi Initial commit

public	Initial commit
src	Initial commit
.eslintrc.json	Initial commit
.gitignore	Initial commit
README.md	Initial commit
next.config.js	Initial commit

Local Codespaces

Clone

HTTPS SSH GitHub CLI

git@github.com:karimmakhloufi/frontend_bo

Use a password-protected SSH key.

Download ZIP



Installation

- Rends-toi dans le dossier et exécute les commandes `npm install` & `npm run dev` pour démarrer le projet en mode développement
- Clique sur le lien dans la console ou rends toi avec ton navigateur sur l'adresse `localhost:3000` pour vérifier que le projet démarre

```
karim@avalanche ~/Desktop/frontend boilerplate main npm run dev
> frontend@0.1.0 dev
> next dev

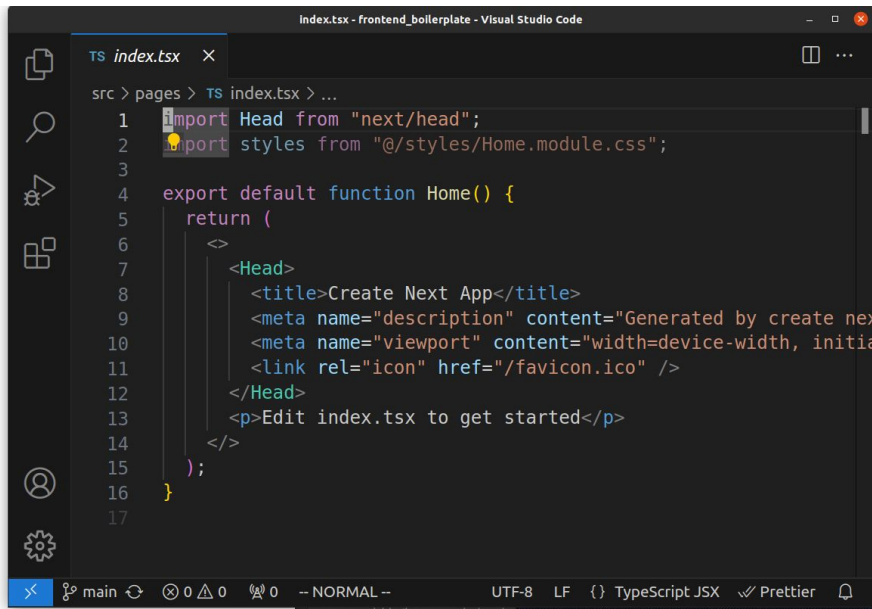
- ready started server on [::]:3000, url: http://localhost:3000
- event compiled client and server successfully in 151 ms (18 modules)
- wait compiling /_error (client and server)...
- event compiled client and server successfully in 168 ms (18 modules)
- wait compiling /_error (client and server)...
- event compiled client and server successfully in 369 ms (188 modules)
- warn Fast Refresh had to perform a full reload. Read more: https://nextjs.org/docs/messages/fast-refresh-reload
- wait compiling / (client and server)...
- event compiled client and server successfully in 144 ms (194 modules)
- warn Fast Refresh had to perform a full reload. Read more: https://nextjs.org/docs/messages/fast-refresh-reload
```



Test du Hot reload

- Modifie le texte du fichier `index.tsx` et vérifie que les changements sont visibles directement dans le navigateur sans avoir besoin de recharger la page

Edit index.tsx to get started



```
TS index.tsx x
src > pages > TS index.tsx > ...
1 import Head from "next/head";
2 import styles from "@styles/Home.module.css";
3
4 export default function Home() {
5   return (
6     <>
7       <Head>
8         <title>Create Next App</title>
9         <meta name="description" content="Generated by create next app" />
10        <meta name="viewport" content="width=device-width, initial-scale=1" />
11        <link rel="icon" href="/favicon.ico" />
12      </Head>
13      <p>Edit index.tsx to get started</p>
14    </>
15  );
16 }
17
```



Intégration du code html

La partie sémantique du frontend



Code HTML

Copie-colle ce code html dans le fichier `index.tsx` :

<https://pastebin.com/Ts9uak8t>

The image shows a development environment with two windows. The left window displays a "Failed to compile" error message. The right window shows the `index.tsx` file with the corresponding JSX code.

Error Message:

```
Failed to compile

./src/pages/index.tsx
Error:
  x JSX Namespace is disabled by default because react does not support it yet. You can specify jsc.transform.react.throwIfNamespace to false to override default behavior
```

Code in `index.tsx`:

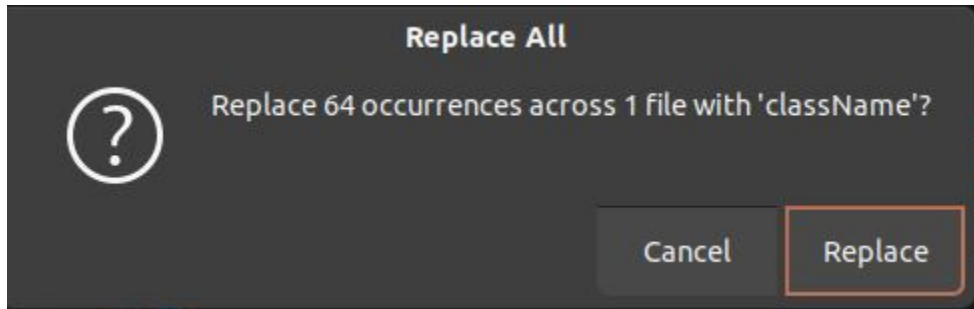
```
1 import Head from "next/head";
2 import styles from "@styles/Home.module.css";
3
4 export default function Home() {
5   return (
6     <div>
7       <body>
8         <header class="header">
9           <div class="main-menu">
10             <h1>
11               <a href="/" class="button logo link-button">
12                 <span class="mobile-short-label">T60
13                 <span class="desktop-long-label">THE
14               </a>
15             </h1>
16             <form class="text-field-with-button">
17               <input class="text-field main-search" type="text" />
18               <button class="button button-primary">
19                 <svg
20                   aria-hidden="true"
21                   width="16"
22                   height="16"
23                   xmlns="http://www.w3.org/2000/svg"
24                   viewBox="-50 -50 530 550"
25                   style="transform: scale(-1, 1); fill: #000;"/>
```




Du HTML au JSX

Afin que le code HTML soit du code JSX valable il faut l'adapter :

- **class** est un mot clé réservé en Javascript pour créer des classes au sens POO du terme. En JSX, une classe javascript s'écrit **className**. Remplace donc **class** par **className**

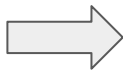




Du HTML au JSX

En **JSX**, une icône **SVG** n'accepte pas de balise style contenant plusieurs instructions. Chaque instruction a sa propre balise. Les balises ne s'écrivent pas en **kebab-case** mais en **camelCase**

```
<svg
  aria-hidden="true"
  width="16"
  height="16"
  xmlns="http://www.w3.org/2000/svg"
  viewBox="-50 -50 530 550"
  style="transform: scale(-1, 1); fill: currentColor"
  xml:space="preserve"
  className="styled__BaseIcon-sc-1jsm4qr-0 l1mHhT"
>
  <path d="m464.524 412.846-97.929-97.925c23.6-34.068
</svg>
```



```
<svg
  aria-hidden="true"
  width="16"
  height="16"
  xmlns="http://www.w3.org/2000/svg"
  viewBox="-50 -50 530 550"
  transform="scale(-1, 1)"
  fill="currentColor"
  xmlSpace="preserve"
  className="styled__BaseIcon-sc-1jsm4qr-0 l1mHhT"
>
  <path d="m464.524 412.846-97.929-97.925c23.6-34.0
</svg>
```



Ajout du CSS

Le projet est maintenant fonctionnel, cependant la page manque cruellement de CSS

TGCTHE GOOD CORNER

[Publier](#)[Publier une annonce](#)

[Ameublement](#) • [Electroménager](#) • [Photographie](#) • [Informatique](#) • [Téléphonie](#) • [Vélos](#) • [Véhicules](#) • [Sport](#) • [Habilleement](#) • [Bébé](#) • [Outillage](#) • [Services](#) • [Vacances](#)

Annonces récentes



[Table](#)

120 €



[Dame-jeanne](#)

75 €



[Vide-poche](#)

4 €



[Vaisselier](#)

900 €



[Bougie](#)

8 €



[Porte-magazine](#)

45 €



CSS Global

Remplace le fichier `globals.css` par le contenu de ce lien:

<https://pastebin.com/E6CVKSfx>

THE GOOD CORNER



Publier une annonce

Ameublement • Électroménager • Photographie • Informatique • Téléphonie • Vélos • Véhicules • Sport • Habillement • Bébé • Outillage • Services • Vacances

Annonces récentes



Table

120 €



Dame-jeanne

75 €



Vide-poche

4 €



Vaisselle

900 €



Bougie

8 €



Porte-magazine

45 €



CSS Global

NextJS permet d'utiliser du **CSS** de manière classique en utilisant le fichier **globals.css**

Nous verrons par la suite qu'il est également possible d'utiliser des **CSS modules**

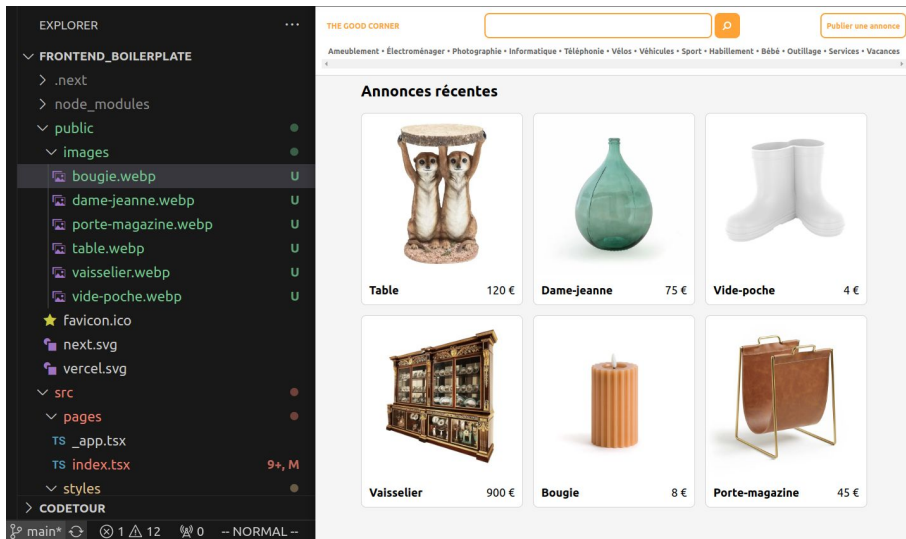


Ajout des images

Télécharge les images de ce dossier

<https://github.com/arnaudrenauxd/wcs-tgc-html-css/tree/main/images>

et ajoute-les dans le répertoire **public** du projet dans un sous-dossier **images**





Découpage en composants







Nous allons maintenant découper notre projet en composants:

THE GOOD CORNER

[Publier une annonce](#)

[Ameublement](#) • [Électroménager](#) • [Photographie](#) • [Informatique](#) • [Téléphonie](#) • [Vélos](#) • [Véhicules](#) • [Sport](#) • [Habille ment](#) • [Bébé](#) • [Outillage](#) • [Services](#) • [Vacances](#)

Annonces récentes

		
Table 120 €	Dame-jeanne 75 €	Vide-poche 4 €
		
Vaisselle 900 €	Bougie 8 €	Porte-magazine 45 €



Découpage en composants

Ajoutons un dossier **components** dans le dossier **src**

Créons maintenant 2 composants React. Un pour le header, l'autre pour les annonces récentes

The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORER' sidebar displays a file tree for a project named 'FRONTEND_B...'. The tree includes files like 'next.svg', 'vercel.svg', and a 'src' directory. Inside 'src', there is a 'components' folder which is expanded, showing 'TS Header.tsx' (highlighted with a blue bar and a 'U' icon) and 'TS RecentAds.tsx' (with a 'U' icon). Other files like 'pages/TS _app.tsx', 'pages/TS index.tsx', 'styles/TS _app.tsx', and 'globals.css' are also visible. The main editor area shows the code for 'TS Header.tsx'. The code defines a constant 'Header' as a function that returns a JSX element. The JSX element consists of a 'header' with a 'className' of 'header', containing a 'main-menu' div with a 'className' of 'main-menu'. Inside the 'main-menu' div, there is an 'h1' tag with two links: one with 'href' and 'className' attributes, and another with 'className' attributes. Below the 'h1' is a 'form' with a 'className' of 'text-field-with-button', containing an 'input' with 'className' and 'type' attributes, and a 'button' with a 'className' attribute.

```
1  const Header = () => {
2    return (
3      <header className="header">
4        <div className="main-menu">
5          <h1>
6            <a href="" className="button logo link-button">
7              <span className="mobile-short-label">TGC</span>
8              <span className="desktop-long-label">THE GOOD CORNER</span>
9            </a>
10          </h1>
11          <form className="text-field-with-button">
12            <input className="text-field main-search-field" type="search" />
13            <button className="button button-primary">
```




Découpage en composants

Importe maintenant ces 2 composants dans le fichier **index.tsx**

The screenshot shows a code editor with a dark theme. On the left is the 'EXPLORER' sidebar showing a file tree for 'FRONTEND_BOILERPLATE'. The tree includes files like 'next.svg', 'vercel.svg', and a 'src' directory. Inside 'src', there are 'components' and 'pages' subdirectories. 'components' contains 'Header.tsx' and 'RecentAds.tsx', both marked with a 'U' icon. 'pages' contains '_app.tsx' and 'index.tsx', with 'index.tsx' marked with an 'M' icon. Below 'src' are 'styles' (with 'globals.css' and 'Home.module.css') and configuration files like '.eslintrc.json' and '.gitignore'. The main editor area on the right shows the 'TS index.tsx' file. It contains the following code:

```
1 import Header from "@components/Header";
2 import RecentAds from "@components/RecentAds";
3
4 export default function Home() {
5   return (
6     <>
7       <body>
8         <main className="main-content">
9           <Header />
10          <RecentAds />
11        </main>
12      </body>
13    </>
14  );
15 }
16
```



Découpage en composants

On remarque qu'une carte contenant une annonce se répète, nous allons en faire un composant réutilisable





AdCard Component

Afin de rendre réutilisable un composant, nous allons définir ses **props**
Ce sont les éléments qui varient d'un composant à l'autre d'une même famille

Ici on peut définir 4 propriétés qui varient d'une carte à l'autre:

- L'url de l'image
- Le titre
- Le prix
- Le lien vers lequel un clic sur la carte nous amène

Nous allons donc créer un composant à partir du html et remplacer ces éléments par des props



AdCard Component

Voici notre composant AdCard:

```
1  const AdCard = ({ title, imgUrl, price, link }) => {
2    return (
3      <div className="ad-card-container">
4        <a className="ad-card-link" href={link}>
5          <img className="ad-card-image" src={imgUrl} />
6          <div className="ad-card-text">
7            <div className="ad-card-title">{title}</div>
8            <div className="ad-card-price">{price} €</div>
9          </div>
10         </a>
11       </div>
12     );
13   };
14
```

Il faut typer les props !



AdCard Component

Voici notre composant AdCard typé :

TS AdCard.tsx 2, U X

src > components > TS AdCard.tsx > ...

```
1  type AdCardProps = {  
2    title: string;  
3    imgUrl: string;  
4    price: number;  
5    link: string;  
6  };  
7  
8  const AdCard = ({ title, imgUrl, price, link }: AdCardProps) => {
```

On peut aussi typer les props avec une interface



Réutilisation de composants

Utilisons maintenant notre composant AdCard plusieurs fois en lui passant différentes props :

```
TS RecentAds.tsx U X
src > components > TS RecentAds.tsx > [🔍] RecentAds
1  import AdCard from "../AdCard";
2
3  const RecentAds = () => {
4    return (
5      <>
6        <h2>Annonces récentes</h2>
7        <section className="recent-ads">
8          <AdCard
9            imgUrl="/images/table.webp"
10             link="/ads/table"
11             price={120}
12             title="Table"
13          />
14          <AdCard
15            imgUrl="/images/dame-jeanne.webp"
16             link="/ads/dame-jeanne"
17             price={75}
18             title="Dame-jeanne"
19          />

```



Map sur un tableau d'objets

En développement web, on récupère souvent un tableau d'objets de la part du backend. Créons un tableau d'objets contenant les informations de nos annonces:

```
TS RecentAds.tsx U X
src > components > TS RecentAds.tsx > [🔗] RecentAds > [🔗] ads
1  import AdCard, { AdCardProps } from "../AdCard";
2
3  const RecentAds = () => {
4    const ads: AdCardProps[] = [
5      {
6        imgUrl: "/images/table.webp",
7        link: "/ads/table",
8        price: 120,
9        title: "Table",
10     },
11     {
12       imgUrl: "/images/dame-jeanne.webp",
13       link: "/ads/dame-jeanne",
14       price: 75,
15       title: "Dame-jeanne",
16     },
17   ],
18 }
```



Map sur un tableau d'objets

La fonction `.map()` des tableaux JS nous permet de générer un tableau de composants React à partir d'un tableau de données.

```
42   return (  
43     <>  
44       <h2>Annonces récentes</h2>  
45       <section className="recent-ads">  
46         {ads.map((ad) => (  
47           <AdCard  
48             imgUrl={ad.imgUrl}  
49             link={ad.link}  
50             price={ad.price}  
51             title={ad.title}  
52             key={ad.title}  
53           />  
54         )}  
55       </section>  
56     </>  
  )
```

Il faut spécifier une prop key unique pour chaque élément, en général on utilise l'id de la base de données



A toi de jouer !

- Met en place le frontend du projet à partir du boilerplate
- Ajoute le HTML et le CSS
- Découpe le HTML en composants
- Crée un tableau d'objets et map dessus pour afficher les annonces
- Crée un tableau d'objets et map dessus pour afficher les catégories
- **Bonus** : Utilise les CSS Modules pour le CSS
<https://nextjs.org/docs/app/building-your-application/styling/css-modules>