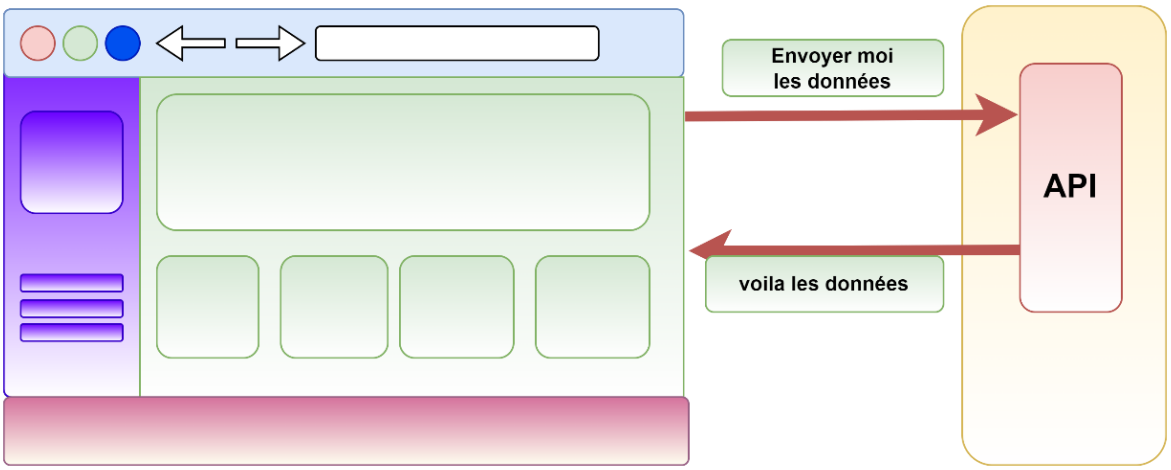
11.1. React et AJAX



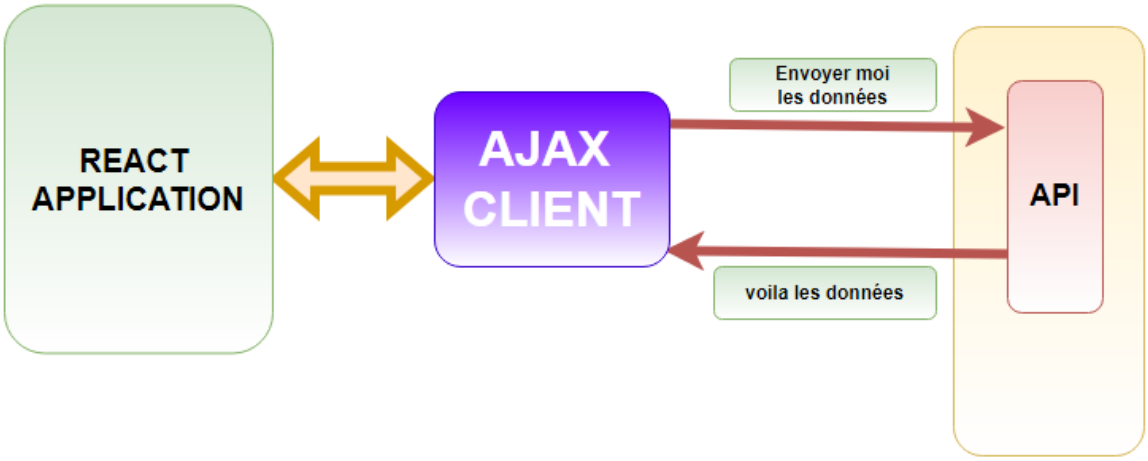
Ajax est un concept permettant de mettre à jour une partie de la page HTML affichée en effectuant une requête au serveur. Ce concept est très utilisé de nos jours dans la plupart des sites web, car il permet une mise à jour dynamique de la page sans avoir à la recharger totalement.

React n'intègre pas une API permettant d'effectuer des requêtes Ajax, mais il est possible d'utiliser d'autres API fournies par d'autres bibliothèques, voire d'utiliser l'API interne du navigateur.

Nous étudions ici ces deux possibilités :

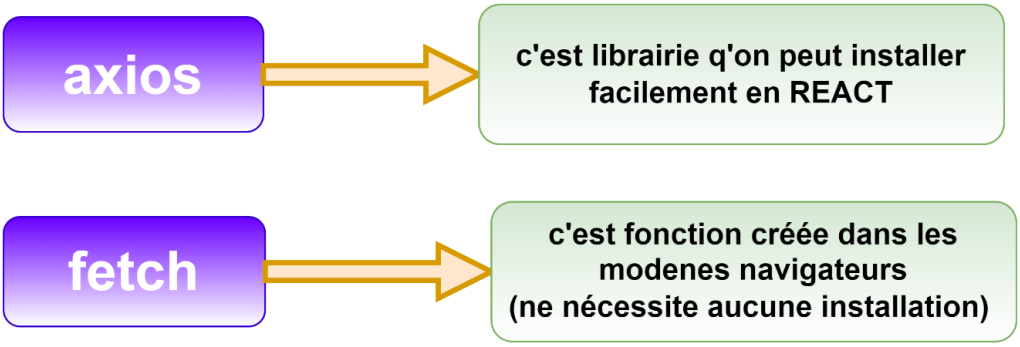
- utilisation de l'API interne du navigateur (en utilisant la méthode fetch()).
- Utilisation de l'API AXIOS

Pour le fournisseur d'API on va utiliser nos propres API sur un serveur PHP ou bien utiliser l'API en ligne :



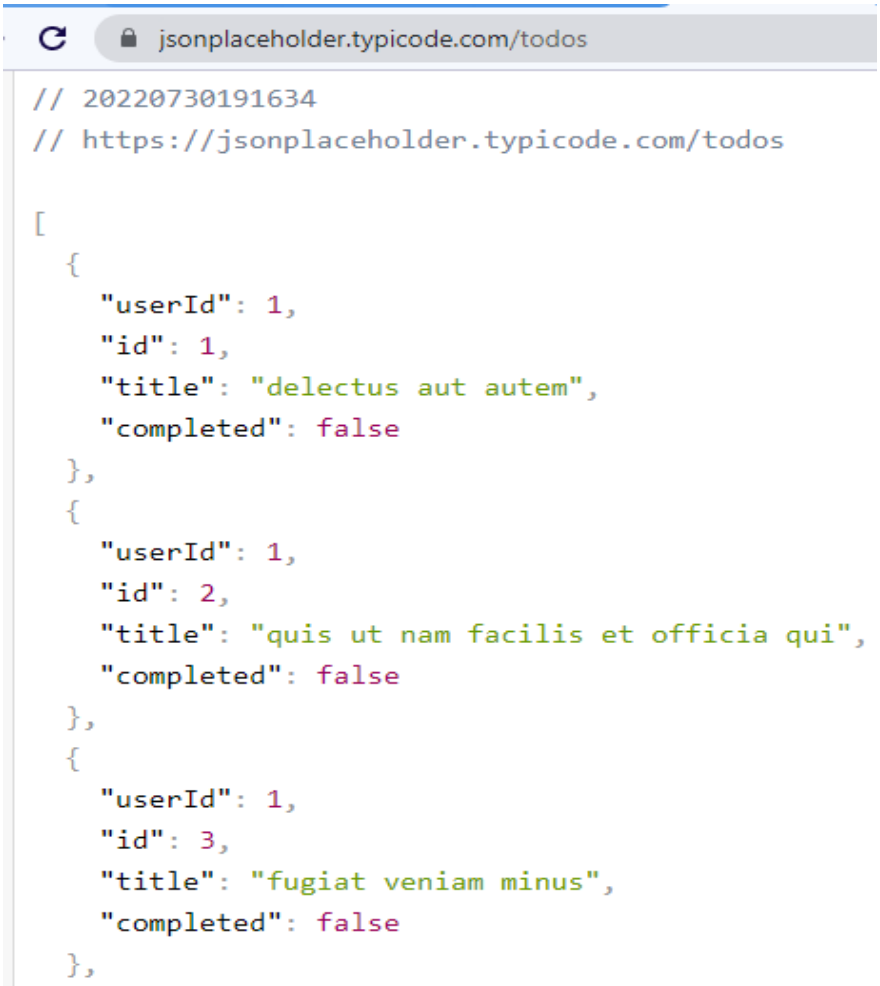
Il est à préciser que REACT est une librairie destinée pour faire l'affichage de contenu HTML au client, interagir avec le client, récupérer les requêtes client.

REACT n'est pas responsable pour consommer un API,c'est le rôle de développeur d'ajouter la partie du code responsable de cette operation (utiliser par exemple fetch,axios……..)



L'endpoint de l'API est sous la forme :
https://jsonplaceholder.typicode.com/todos

Quand on navigue à cette url on aura le rendu suivant :



Cet API retourne les données en format JSON

Les différents EndPoint de cet API

endpoint	Array de :
https://jsonplaceholder.typicode.com/todos	{ "userId": 1, "id": 1, "title": "delectus aut autem", "completed": false }

https://jsonplaceholder.typicode.com/comments	{
	"postId": 1,
	"id": 1,
	"name": "id labore ex et quam laborum",
	"email": "Eliseo@gardner.biz",
	"body": "laudantium enim quasi est quidem magnam voluptate ipsam eos\ntempora quo necessitatibus\ "
	}

https://jsonplaceholder.typicode.com/comments	{ "userId": 1, "id": 1, "title": "quidem molestiae enim" },
https://jsonplaceholder.typicode.com/photos	{ "albumId": 1, "id": 1, "title": "accusamus beatae ad facilis cum similique qui sunt", "url": "https://via.placeholder.com/600/92c952", "thumbnailUrl": "https://via.placeholder.com/150/92c952" },

https://jsonplaceholder.typicode.com/todos	{
	"userId": 1,
	"id": 1,
	"title": "delectus aut autem",
	"completed": false
	}

https://jsonplaceholder.typicode.com/users	{
	"id": 1,
	"name": "Leanne Graham",
	"username": "Bret",
	"email": "Sincere@april.biz",
	"address": {
	"street": "Kulas Light",
	"suite": "Apt. 556",
	"city": "Gwenborough",
	"zipcode": "92998-3874",
	"geo": {
	"lat": "-37.3159",
	"lng": "81.1496"
	}

Les exemples qui vont être utilisés dans le cous vont être traités par à la fois **FETCH** et **AXIOS**

11.2. Consommation d'un API par AXIOS fonctionnel composant

Axions Client HTTP basé sur les promesses pour navigateur et node.js

Axios est un client HTTP simple basé sur les promesses compatibles avec le navigateur et node.js. Il propose une librairie facile à utiliser et à étendre, le tout dans un tout petit package.

Installation de axios

```
npm install --save axios
```

Utilisation de axios pour consommer l'PI : https://jsonplaceholder.typicode.com/users

Il faut importer axios

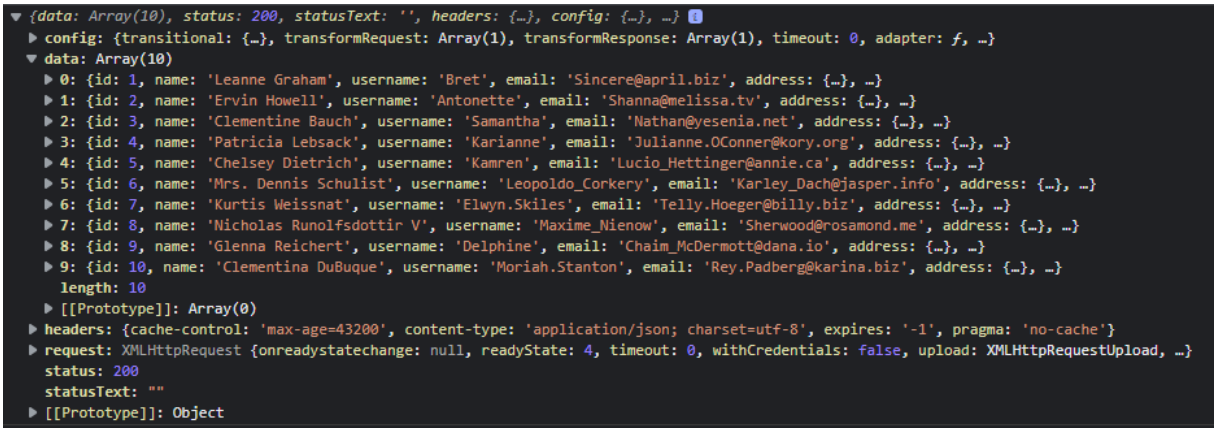
```
import axios from 'axios'
```

Le code suivant :

```
  axios.get('https://jsonplaceholder.typicode.com/users').then(
        (res)=>{ console.log(res)}
    )
```

Affiche le rendu console

```
▶ {data: Array(10), status: 200, statusText: '', headers: {…}, config: {…}, …}
```



Le rendu est un objet javascript qui contient les propriétés data, status, statusText, headers….

On peut remarquer que la propriété data contient un Array constitué par les objet javaScript user.

Par ailleurs si on écrit :

```
axios.get('https://jsonplaceholder.typicode.com/users').then(
        (res)=>{ console.log(res.data)}
    )
```

On aura le rendu suivant sur le console:

Par conséquent res.data est un Array de 10 users.

Maintenant on doit se demander ou nous devons placer le code de consommation de l'API

Si on est dans un fonctionnel composant, le code doit être place dans useEffect mais le deuxième argument doit contenir [],pour que l'exécution se lance uniquement au premier rendu

```
useEffect(()=>{
    axios.get('https://jsonplaceholder.typicode.com/users').then(
        (res)=>{ console.log(res.data)}
    )
    },[])
```

Si on ne met pas l'argument [],le code sera exécuté à chaque changement des propriété d'états useState

Affichant maintenant le nombre des users, pour ce faire il faut créer une propriété d'état useState utilisateurs destinée pour récupérer le résultat users de l'API

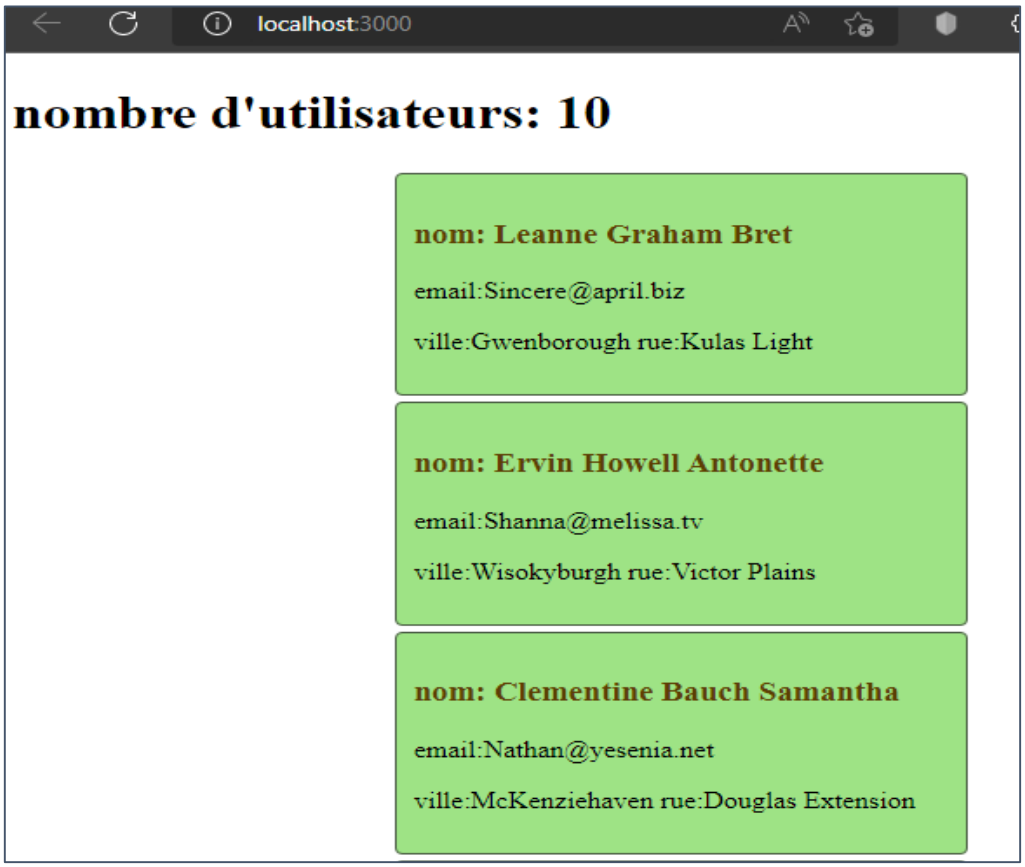
Code App.js:

```
import React, { useEffect, useState } from 'react'
import axios from 'axios'
export default function App(){
    const [utilisateurs,setUtilisateurs]=useState([])
    useEffect(()=>{
    axios.get('https://jsonplaceholder.typicode.com/users')
       .then((res)=>{ console.log(res.data);setUtilisateurs(res.data)})
    },[])
    return(
        <div>
            <h1>nombre d'utilisateurs: {utilisateurs.length}</h1>
        </div>
    )
}
```

Retenons que l'objet user est de la forme :


```
{
  "id": 1,
  "name": "Leanne Graham",
  "username": "Bret",
  "email": "Sincere@april.biz",
  "address": {
    "street": "Kulas Light",
    "suite": "Apt. 556",
    "city": "Gwenborough",
    "zipcode": "92998-3874",
    "geo": {
      "lat": "-37.3159",
      "lng": "81.1496"
    }
```


Nous allons afficher les utilisateurs ures sur l'écran, pour ce faire on ajoute le code jsx suivant



Code finale de App.js

```
import React, { useEffect, useState } from 'react'
import axios from 'axios'
export default function App(){
    const [utilisateurs,setUtilisateurs]=useState([])
    useEffect(()=>{
    axios.get('https://jsonplaceholder.typicode.com/users')
       .then((res)=>{setUtilisateurs(res.data)})
    },[])
  return(
        <div>
            <h3>liste utilisateurs</h3>
            <div>
            <h1>nombre d'utilisateurs: {utilisateurs.length}</h1>
            {utilisateurs.map(user=>{
                return(
                    <div className='child' key={user.id}>
                        <h3 style={{color:"rgb(92, 62, 3)"}}>nom:
{user.name}  {user.username}</h3>
                        <p>email:{user.email}</p>
                        <p> ville:{user.address.city}
rue:{user.address.street} </p>
                    </div>
                )
            })}</div>
        </div>
)}
```

On peut utiliser **async await**

```
 useEffect(()=>{
        const getData=async ()=>{
             const users=
await  axios.get('https://jsonplaceholder.typicode.com/users')
             setUtilisateurs(users.data)
        }
        getData()
},[])
```

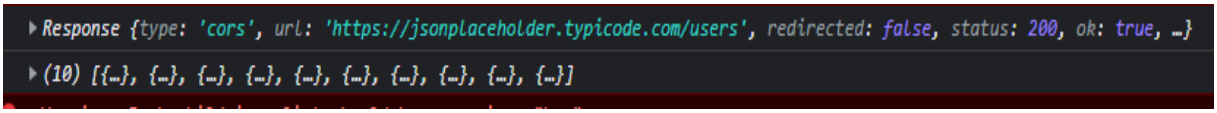
Le style utilisé pour mettre en forme le rendu

```
.child{
    background-color:rgb(158, 227, 133);
    width:40%;
    margin:4px auto;
    padding:10px;
    border: 1px solid rgb(70, 88, 64);
    border-radius: 4px;
    box-shadow:8px rgb(210, 217, 208); ;
}
```

11.3. Consommation d'un API par fetch fonctionnel composant

```
fetch('https://jsonplaceholder.typicode.com/users')
    .then((response)=>{ console.log(response); return response.json()})
    .then((users)=>{console.log(users);setUtilisateurs(users)})
```

Rendu console



Etant donné que les données response ne sont pas du json, on applique la méthode json à response qui retourne a une promesse d'où l'enchainement de la deuxième then.

La variable users contient cette fois ci un Array contenant des objet java script user , en suite on met à jour la variable d'état utilisateurs en utilisant le setter setUtilisateurs(users)

L'utilisation **async await** donne le même résultat (voir cours Java script Avancé):

```
const [utilisateurs,setUtilisateurs]=useState([])
   useEffect(()=>{
 const getData=async (rep)=>{
 const response= await  fetch('https://jsonplaceholder.typicode.com/users')
 const users=await response.json()
       setUtilisateurs(users)
       }
       getData()
   },[])
```

11.4. Consommation d'un API par Axios classe composant

Pour les classes composant on met le code de consommation API dans **componentDidMount()**

```
componentDidMount(){
   axios.get('https://jsonplaceholder.typicode.com/users')
        .then((res)=>{this.setState({utilisateurs:res.data})})
}
```

Code complet :

App.js

```
import React  from "react";
import axios from 'axios'
export default class App extends React.Component{
    constructor(props) {
      super(props)
      this.state = {
         utilisateurs:[]
      }
    }
    componentDidMount(){
```

```
        axios.get('https://jsonplaceholder.typicode.com/users')
        .then((res)=>{this.setState({utilisateurs:res.data})})
    }

    render(){
        return(
            <div>
                <h3>liste utilisateurs</h3>

                <div>
                <h1>nombre d'utilisateurs:
{this.state.utilisateurs.length}</h1>
                {this.state.utilisateurs.map(user=>{

                    return(
                        <div className='child' key={user.id}>
                            <h3 style={{color:"rgb(92, 62, 3)"}}>nom:
{user.name}  {user.username}</h3>
                            <p>email:{user.email}</p>
                            <p> ville:{user.address.city}
rue:{user.address.street} </p>
                        </div>
                    )

                })}</div>

            </div>
        )
    }
}
```

11.5. Consommation d'un API par fetch classe composant :

On change seulement le code de la méthode **componentDidMount**

```
componentDidMount(){

        fetch('https://jsonplaceholder.typicode.com/users')
        .then((response)=>{ console.log(response); return response.json()})
        .then((users)=>{this.setState({utilisateurs:users})})

    }
```

Exercice d'application 1 avec solution :

Sachons que le endpoint https://jsonplaceholder.typicode.com/users/3

Retourne l'objet user qui a le id égal à 3

```
{
  "id": 3,
  "name": "Clementine Bauch",
  "username": "Samantha",
  "email": "Nathan@yesenia.net",
  "address": {
    "street": "Douglas Extension",
    "suite": "Suite 847",
    "city": "McKenziehaven",
    "zipcode": "59590-4157",
    "geo": {
      "lat": "-68.6102",
      "lng": "-47.0653"
    }
  },
  "phone": "1-463-123-4447",
  "website": "ramiro.info",
  "company": {
    "name": "Romaguera-Jacobson",
    "catchPhrase": "Face to face bifurcated interface",
    "bs": "e-enable strategic applications"
  }
}
```


Créer l'application React

L'utilisateur saisie le id suite à l'événement onChange, les informations de l'utilisateur s'affichent

Le message 'svp choisir un id valide !!!!' S'affiche si l'utilisateur n'a rien saisie ou bien le id n'existe pas

Eléments de solution :

```
import React, { useEffect, useState } from 'react'
export default function App(){
  const [id,setId]=useState(1)
  const [utilisateur,setUtilisateur]=useState({})
  const [address,setAddress]=useState({})
  function handelChangeId(event){
    setId(event.target.value)
  }
  useEffect(
()=>{
fetch(`https://jsonplaceholder.typicode.com/users/${id}`)
.then((response)=>{ return response.json()})
.then((user)=>{setAddress(user.address);setUtilisateur(user);})
},[id] )
    return(
        <div>
         <h1>Details utilisateur</h1>
         <div>
            <label>donner le id:</label>
            <input type="text" onChange={handelChangeId} value={id}></input>
         </div>
          {utilisateur && address ?
         <div className='child' key={utilisateur.id}>
                        <h3 style={{color:"rgb(92, 62,
3)"}}>  id:{utilisateur.id}  nom:
{utilisateur.name}   {utilisateur.username}</h3>
                        <p>email:{utilisateur.email}</p>
                        <p> telephone:{utilisateur.phone} </p>
                        <p> site web:{utilisateur.website} </p>
                        <p> rue : {address.street} </p>
                        <p> ville : {address.city} </p>
        </div>:"svp choisir un id valide!!!!" }
        </div>
    )
}
```

CSS :

```css
.child{
    background-color:rgb(158, 227, 133);
    width:40%;
    margin:4px auto;
    padding:10px;
    border: 1px solid rgb(70, 88, 64);
    border-radius: 4px;
    box-shadow:8px rgb(210, 217, 208); ;
}
```