Interragir en js avec les smart contrats

Installation des dépendances

- Npm install ethers

La librairie Ethers.js

- Son but est d’assurer la communication avec les blockchains evm
- Par l’intermédiaire d’un connecteur RPC
- La liste des RPC des différentes blockchains peut être trouvé ici :
  - https://chainlist.org/
  - un fournisseur comme alchemy ou Infura.
-

Se connecter a un blockchain avec ethers.js

- 1 importer les dépendances
- 2 créer un provider en instanciant éthers.provider :

```
const provider = new ethers.providers.JsonRpcProvider('http://small.tixier.org:8545')
```

- 3 tester en récupérant le bloc courant :

```
const bloc = await provider.getBlockNumber()
```

Détecter les ethers sur un compte :

- En premier lieu il faut charger la librairie éthers en utilisant :
- const { ethers } = require(« ethers");

- Puis d’instancier éthers providers :

```
const provider = new ethers.providers.JsonRpcProvider('http://small.tixier.org:8545')
```

- Au final de contacter le provider pour utiliser la fonction getBalance.

Envoi d’ethers

- L’envoi d’ethers se fait en signant une transaction avec la clé privée de l’utilisateur dans le fichier 4-envoiEthers.
- Wallet utilise la clé privée et le provider que nous avons crée plus haut.
- Wallet nous permettra donc de créer une transaction qui attend les paramètres to: (Address) , value: montant à envoyer.

Communication avec les smart contrats

- Un smart contract une fois compilé ne nous permet pas d’acceder à ses fonctions.
- Pour retrouver ses fonctions nous devons connaitre l’ABI (Application Binary Interface)
- L’ABI est disponible dans le volet compilation de remix

Communication avec les smartcontracts partie 2

- On interroge le contrat avec la fonction éthers.Contract

```
const contract = new ethers.Contract(erc20Address,abi, provider)
```

- On interagit avec le contrat avec un signer (contract.connect)

```
const contractSigner = contract.connect(wallet)
```

-

Pour résumer avec éthers.js 1/2

- Pour consulter la blockchain nous avons besoin d'un provider
- Le provider permet de connaitre l'état des variables génériques de la blockchain et le solde des comptes en éther
- Les transformations d'unités se font par l'intermédiaire de parseEther pour convertir d'éther en wei, et format éther de wei en éther.
- Pour envoyer des éthers nous avons besoin de la clé privée qui signera les transactions avec la fonction sendTransaction qui est dispo avec Wallet.

Pour résumer avec éthers.js 2/2

- Pour communiquer avec un contrat nous avons besoin de son ABI et de son adresse.
- Instanciation de wallet avec clé privé et provider
- Instanciation de contract avec adresse ERC20 abi et provider
- Et d’un contract signer pour écrire dans le contrat (contract.connect(wallet))