**Section recap**

**Recap**

I know this lesson was pretty quick, but my intent was to give you some degree of familiarity with the low-level functionality behind interacting with websites in Web3.

If you're interested in expanding your full-stack skills, I encourage you to check out the html-fund-me repo in more depth. Some additional tools and frameworks you may want to investigate include:

* [**React**](https://react.dev/)
* [**Svelte**](https://svelte.dev/)

Let's do a refresher on the important things to know under the hood, when it comes to interacting using our wallets.

We learnt, in order to send a transaction, you need to connect your wallet.

The most popular way to connect our wallet to Web3 enabled applications is through browser injection. Our browser can check for the presence of a wallet by checking for the window.ethereum object.

Additionally, in order to send a transaction to our wallet, our browser needs an RPC URL or a provider this is derived from the ethereum.window object that our browser wallet creates.

const provider = new ethers.providers.Web3Provider(window.ethereum);

Our wallet also provides the browser with an account to use through this line.

const signer = provider.getSigner();

Once a wallet is connected, it's important to remember that the browser sends transactions *to* our wallet for signing/confirmation. The wallet does *not* provide private key information to the browser application.

We also learnt a basic way to verify the function calls being sent to our wallet through the use of function selectors and decoding calldata. We'll go over this in more detail later!

That's all there is to this lesson! With your deeper understanding of how transactions are handled, I'll see you in the next one!