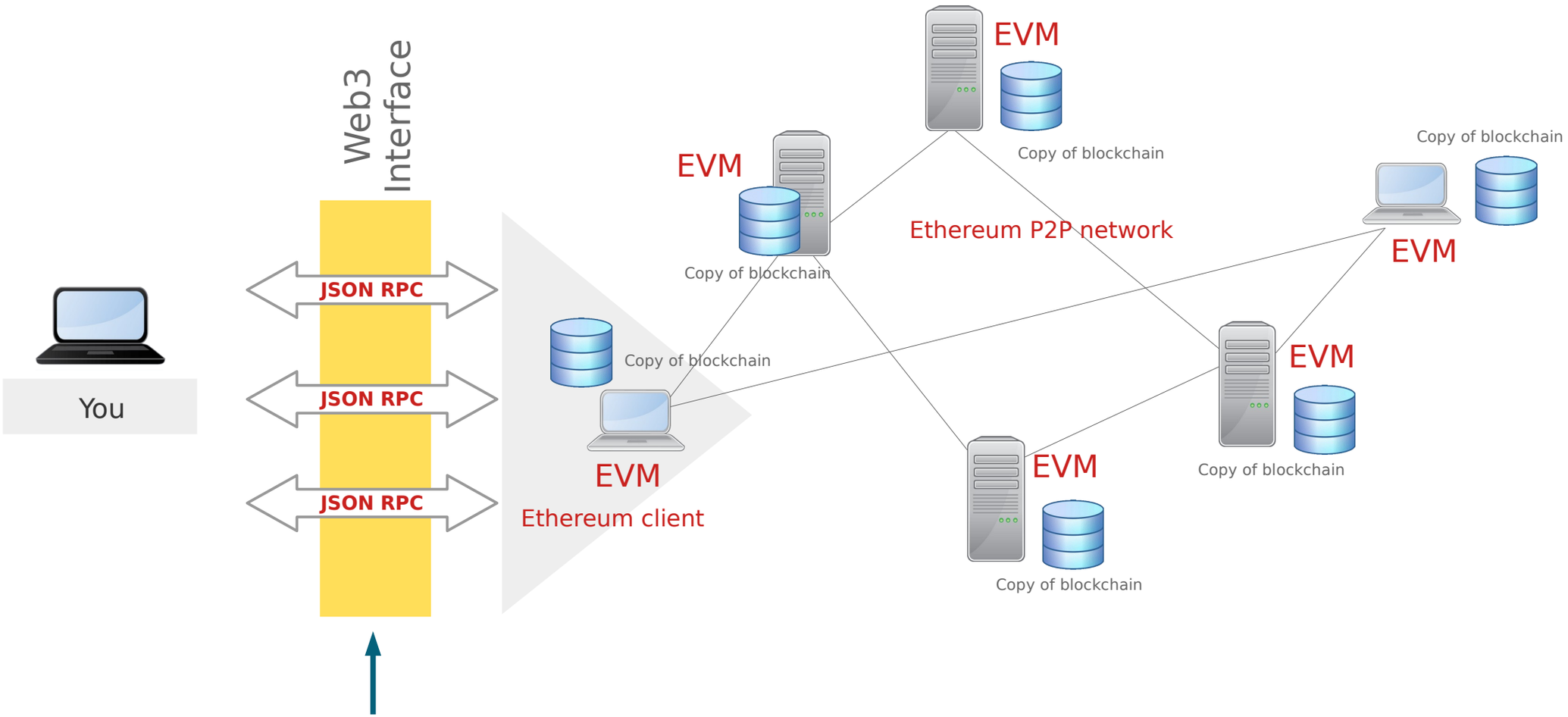
Decentralized Systems

Web3 (cnt)

Ethereum ecosystem



https://ethereum.org/en/developers/docs/apis/json-rpc/

Examples of today

- Ethers.js and Node.js
  - Sending ETH to other accounts
  - Writing on the blockchain
  - Reading events from the blockchain
- Ethers.js and JavaScript
  - Interacting via browser with WishOfDay smart contract

Solidity: events (recall)

- Smart contract WishOfDay: we can add one event emitted each time a new wish is written in the blockchain

1. **Declaration**

```
event WishAdded(uint256 _data, string _message, string indexed _author, address indexed _from);
```

The keyword indexed is used to make authors and addresses filterable when querying the event logs

2. Usage

```
function setOneWish(string memory _message, string memory _author)
public {
 ...
 ...
  emit WishAdded(block.timestamp, _message, _author, msg.sender);
}
```

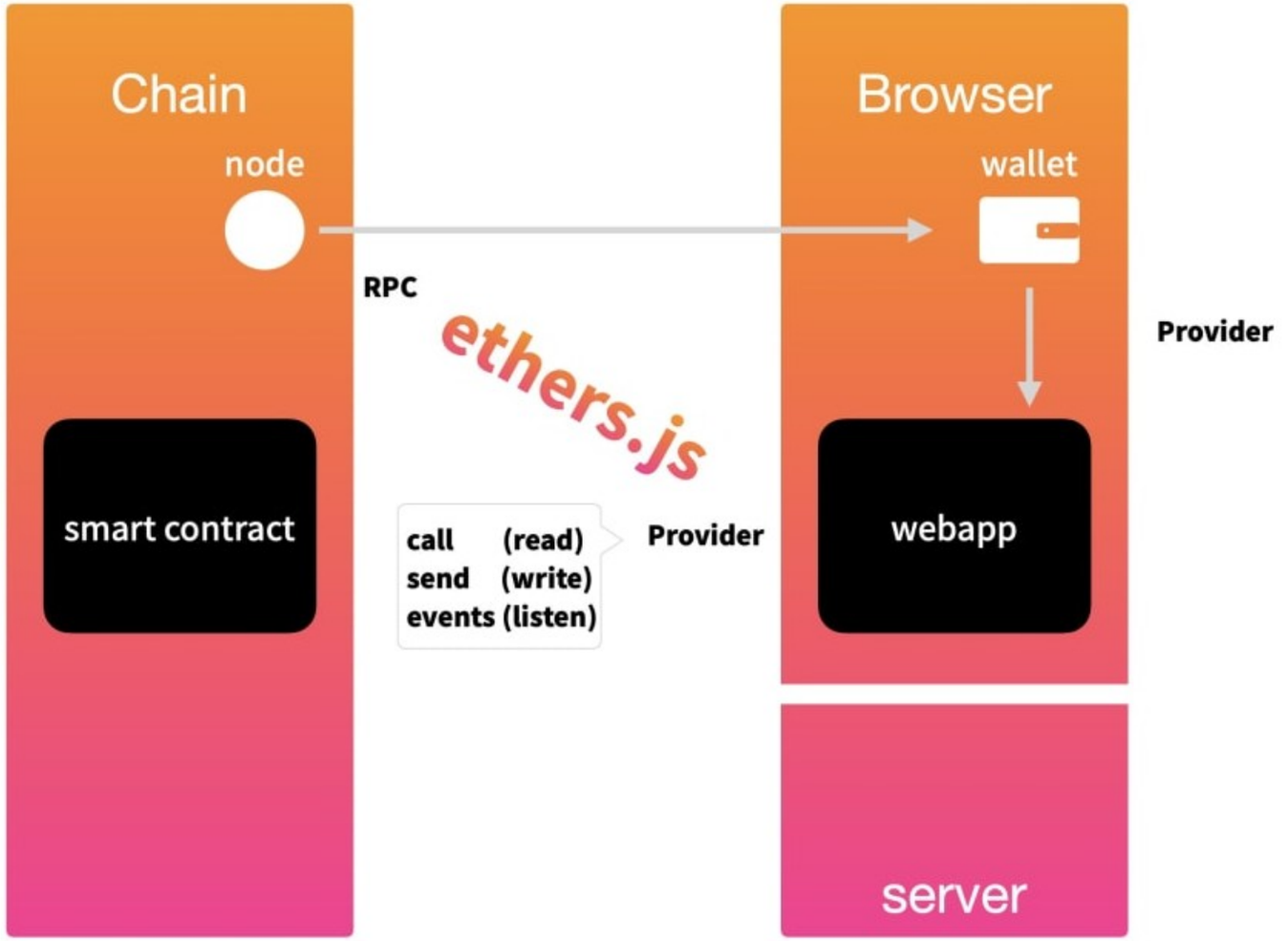
Solidity: events

- Emitted events are used for **logging information** that external web3-based applications can listen to
- Used to **store data** on the blockchain, outside the contract storage, that **can be queried later**
- When an event is emitted, the **topics** are the **indexed parameters of the event** (max 3) and they are the key to enabling efficient searching and filtering of blockchain logs
- The **first topic** always represents the hash of the **event signature** (event name and parameter types), allowing for a quick identification of the event

Solidity: events

- EventLog typically includes
  - **address** of the contract that emitted the event
  - **topics**
  - **data**, e.g., the actual payload, an hex value which can be decoded using the ABI, and contains the values that were emitted with the event

Connect MM with a web3-app



https://dev.to/yakult/a-tutorial-build-dapp-with-hardhat-react-and-ethersjs-1gmi

Connect MM with a web3-app

- In the user interface of a web3-app, wallets can connect the application with the blockchain, using ethers.js or other libraries

- The wallet becomes the **Provider**, thanks to the **window.ethereum** object injected into the browser when the wallet is installed

- With ethers.js it is possible to read and write on smart contracts, thanks to this provider

Connect MM with a web3-app

- MetaMask can be used by
  - Users 

Connect MM with a web3-app

- MetaMask can be used by
  - Users 
  - Developers 

  The **window.ethereum object** API provides a standardized way for web3 applications to interact with Ethereum-compatible wallets

  Bridge between the browser environment and the Ethereum blockchain

Connect MM with a web3-app

- MetaMask can be used by
  - Users 
  - Developers 

  Injected into the browser when the wallet is installed

  Interaction through **asynchronous methods** (e.g., Promises) for blockchain queries and wallet actions

  https://docs.metamask.io/wallet/reference/provider-api/#ethereum-provider-api


Start
DEMO