encode CLUB

Solidity Foundations

Lesson 4

Anatomy of a dApp - Part 2
Interfaces and Frontend

Topics

1. Review of dApps and Interfaces
2. Overview of frontend applications
3. Web development frameworks
4. Practical experiment
5. Next steps

dApps (review)

A **Decentralized application** is an application that can operate **autonomously**, typically through the use of **smart contracts**, that run on a **decentralized computing**, **blockchain** or other **distributed ledger** system.

dApps (review)

Any Smart Contract in the Blockchain can be interacted with, so long we use an **Interface** to do so.

Interfaces (review)

encode CLUB

ERC20 Interface

https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/IERC20.sol

https://github.com/bokkypoobah/WeenusTokenFaucet

https://goerli.etherscan.io/token/0x022E292b44B5a146F2e8ee36Ff44D3dd863C915c#writeContract

Interfaces

Even though **ABIs** and **Contract Interfaces** may allow for solidity developers to easily build and execute smart contract calls, this is not so accessible for the common public.

User Interfaces (UIs) can greatly reduce the distance between a smart contract in a blockchain and the common average user or consumer.

Practical example

encode CLUB

Try it out

Let’s use a smart contract by interface, but let’s do it using a better UX for non developers

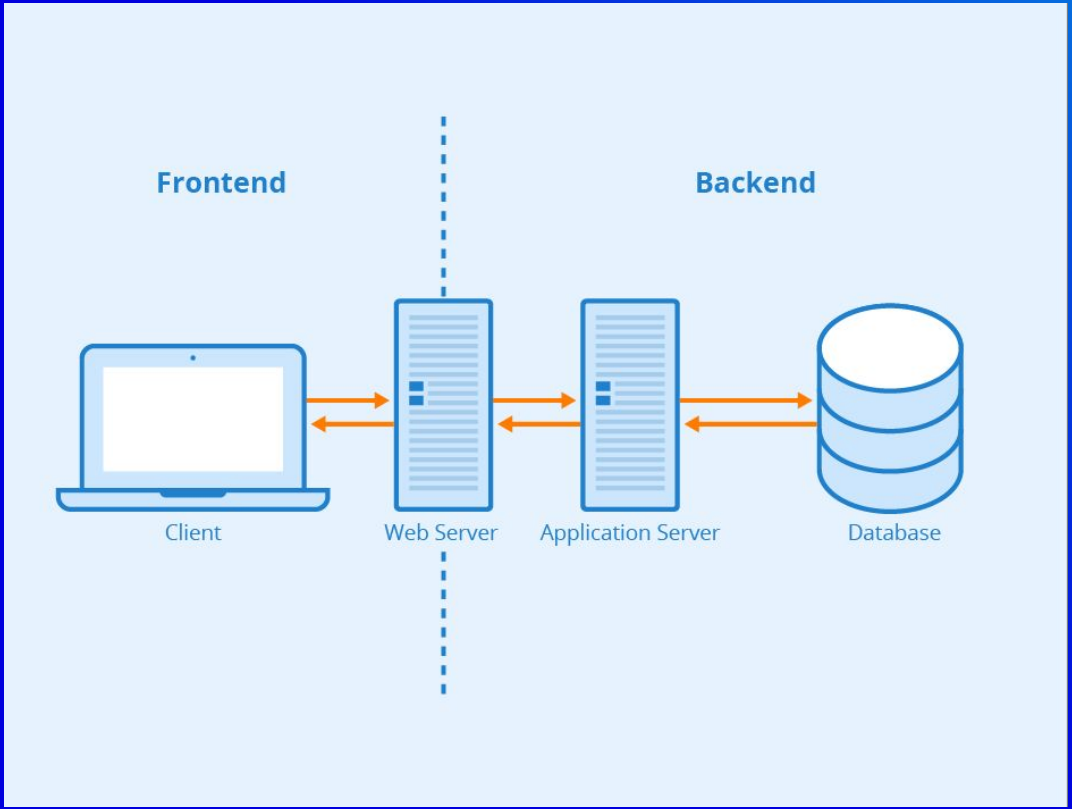
https://www.cryptokitties.co/

https://app.uniswap.org

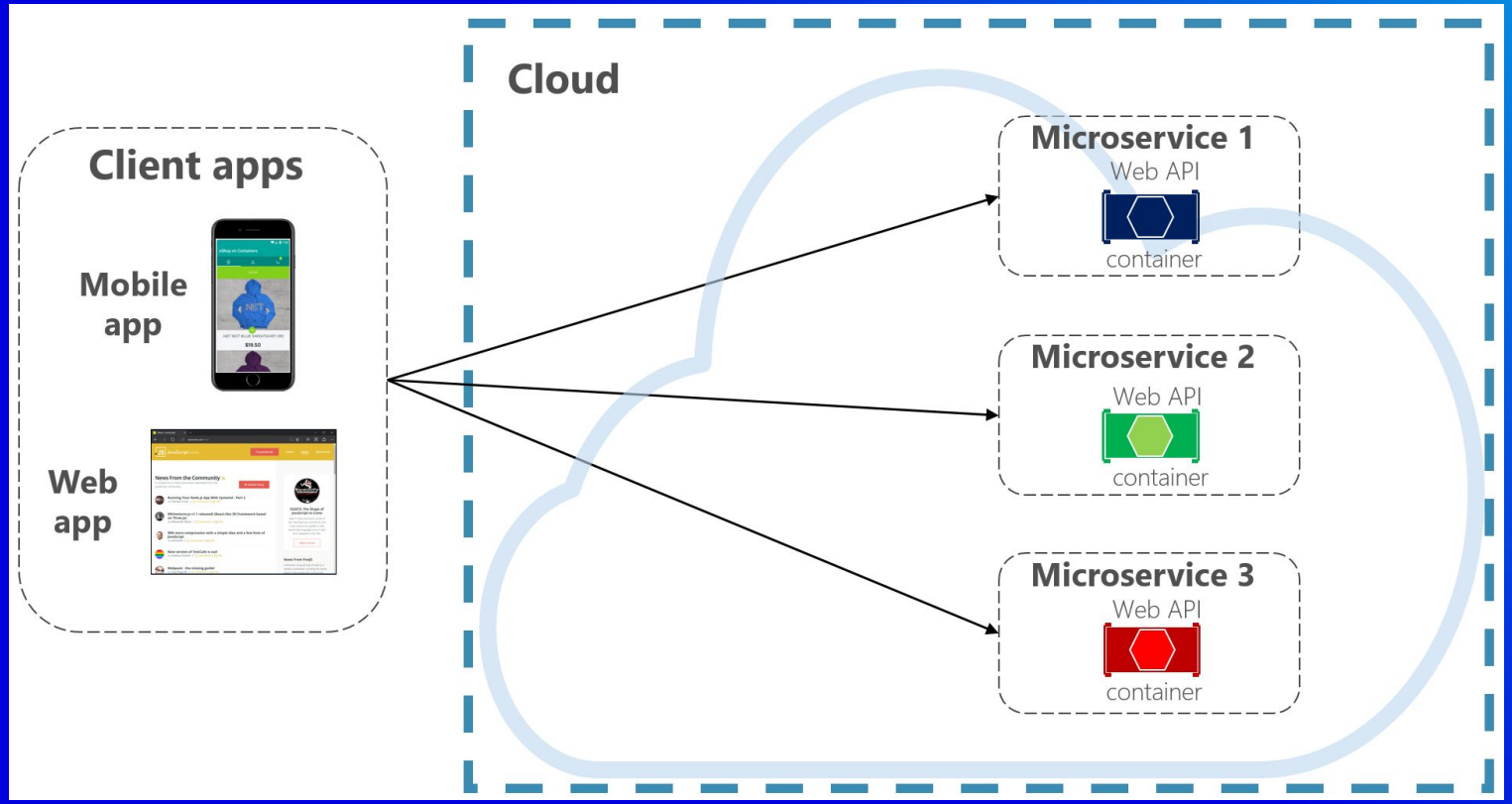
Frontend applications

A **front-end application**, commonly known as the **interface** of an application, is the layer or element that the user has the ability to **use**, **see**, and **interact** with through buttons, images, interactive elements, navigational menus, and text.

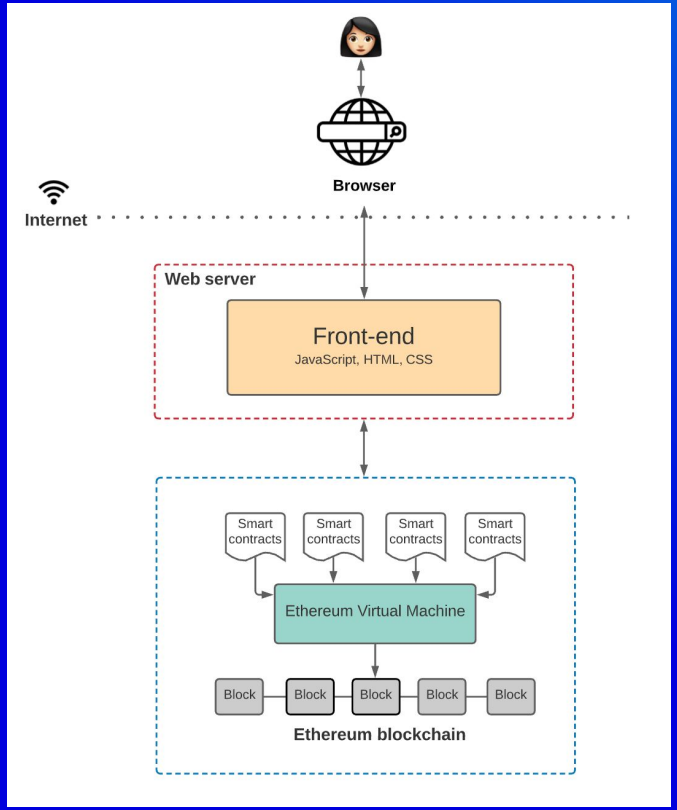
Frontend applications



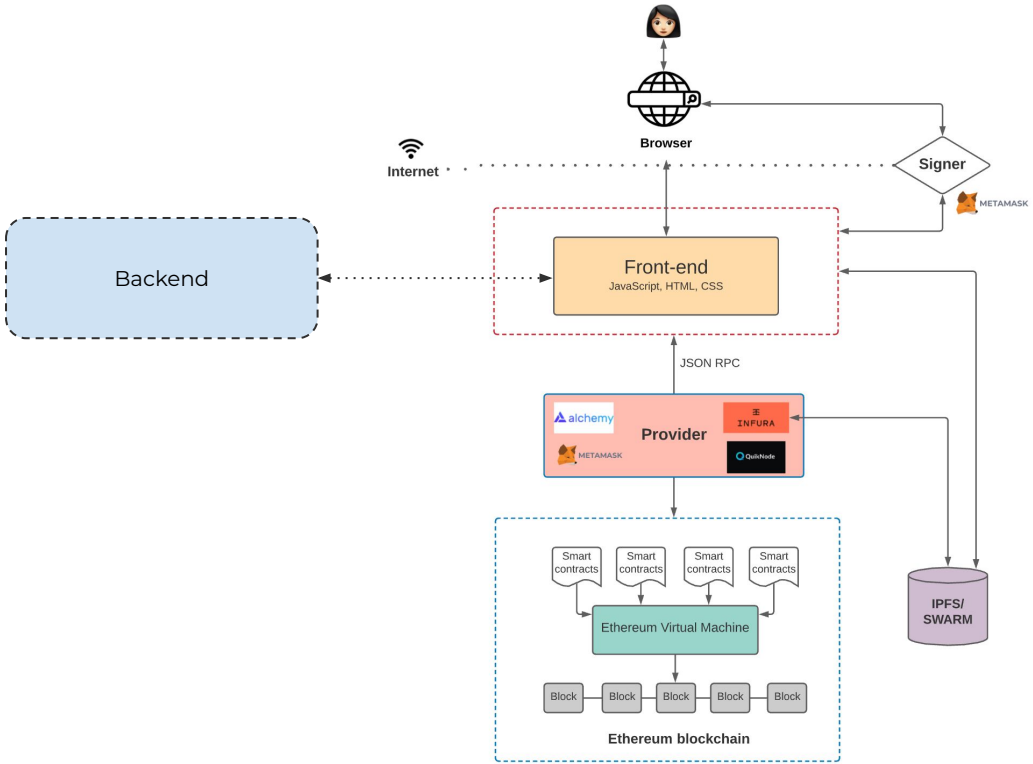
Frontend applications



Frontend applications - Web 3.0



Frontend applications - Web 3.0



Web development frameworks

A **Web Application Framework** is a software framework that is designed to support the **development** of web applications.

Web development frameworks

Web Application Frameworks provide a **standard way** to build and deploy web applications. They aim to **automate** the **overhead** associated with common activities performed in web development.

Web development frameworks

For example, many web frameworks provide libraries and/or tooling for **database access**, pages **routing**, creating and reusing **templates** and **components**, managing **themes**, handling **environment variables**, **session management**, and much more.

They often promote **code reuse** and reduce the **average effort** for building web applications.

Web development frameworks

encode CLUB

Comparison

https://www.simform.com/blog/best-frontend-frameworks/

Statistics

https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190

React

React is a Javascript library for building **user interfaces**.

It is not an **opinionated framework**, like most other popular options.

It is intended to be used in **gradual adoption**, and developers can decide to use more or less of it, as needed.

React

encode CLUB

React library can be used to provide **script** functionality to any **html** file served together with the proper **javascript code**.

The most common way to use React for web development is using the ***Create React App*** tool, that creates an environment that comes pre-configured with everything needed for creating a React application.

React

encode CLUB

Try it out

https://codepen.io/gaearon/pen/MjrdWg?editors=1010

https://codepen.io/mercuryworks/pen/EyQaBO

React

encode CLUB

Uniswap Interface example

https://github.com/Uniswap/interface

https://github.com/Uniswap/interface/blob/main/src/index.tsx

https://github.com/Uniswap/interface/blob/main/src/pages/App.tsx

React

The development of a web application using React requires some software development prerequisites, like **setting up the environment**, basic understanding of **Javascript** and **programming logic**, using **terminal** or **shell applications** to run programs and scripts, basic understanding of **networks** and **data structures**, coding using **IDE**s, using **git** for software version control, **installing** and **managing** dependencies and packages and, possibly, at least a very basic understanding of how your **operational system** works.

Frontend for dApps

Any web development framework that build **static websites** are perfect matches for the Web 3.0 development needs.

If you find a web development framework that requires each page to be **computed** and then **fetched** from a centralized server, this is **not** suitable for Web 3.0, since this could be **exploited** and possibly could weaken the **censorship resistance** of its users.

Frontend for dApps

Even if a dApp frontend is **hosted** at a centralized structure, this should be done only **for convenience**.

Anyone should be able to **clone** the code, **compile**, **deploy** and **run** the application by itself.

On top of that, the dApp frontend and code could also be hosted on **Decentralized Storages** and/or **P2P Networks** for increased **decentralization** and **censorship resistance**.

Practical example

encode CLUB

Try it out

Let's use the ***One Click dApp*** to build a frontend for our HelloWorldOwnable.sol

https://oneclickdapp.com/

Next steps

- Get involved in a community;
- Learn more of Solidity;
- Learn the basics of Javascript and Typescript;
- Sort out your development environment;
- Take part in hackathons;
- Build personal projects;
- Set and track your goals;
- Seek and provide help at forums and chats;
- Get familiar with the documentations;
- Enroll in more bootcamps!