CS 168: Blockchain and Cryptocurrencies

Distributed Apps (DApps)

Prof. Tom Austin

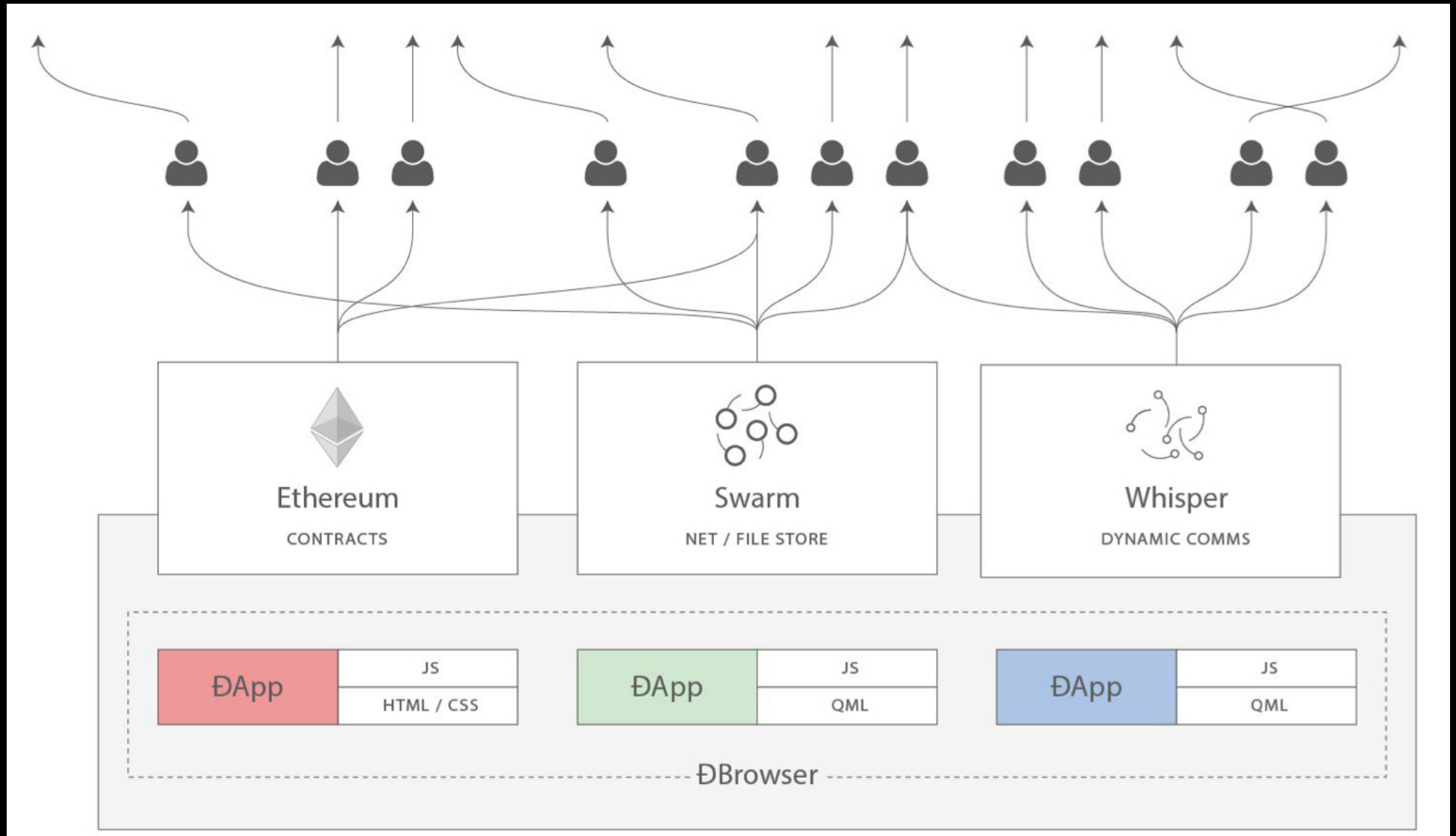
San José State University

Evolution of the Web

- Web 1.0
  - Few content creators
- Web 2.0
  - User-generated content
  - Social media
  - Applications still centralized
- Web 3.0
  - Decentralized applications

What Makes a DApp?

- Smart contracts
  - Business-logic, application state, etc.
  - Replace server-side layer
- Frontend
  - HTML, JavaScript, etc.
  - Web3.js – communication with smart contract
- Other components
  - Storage: IPFS or Swarm
  - Messaging: Whisper



Taken from *Mastering Ethereum*

Advantages of a DApp

- Advantages:
  - Censorship-resistance
  - Transparency
  - Resilience
- Disadvantages:
  - Pretty much everything else

DApp Development Process

1. Compile Solidity code
2. Run Ganache for testing
3. Deploy code
4. Interact through web3.js

Solidity Compiler

- Produces:
  - EVM bytecode
  - Application Binary Interface (ABI)
- Usage:
  ```
  $solcjs --bin --abi ContractFile.sol
  ```
- Installed through npm.
  - NOTE: There are other solidity compilers.

ABI

- Specifies interface for other tools to use.
- Produced by Solidity compiler.

Solidity Compiler Example

(in class)

Ganache

- "One-click blockchain"
  - Blockchain emulator
- Part of Truffle tool suite
- Install through npm
- Creates 10 accounts with 100 ether

Web3.js

- Interacts with Smart Contract
- Works with either node or browser
- WARNING: a little buggy

DApp example

(in class)

Note on Using Strings

- Many applications use `bytes32` rather than `string`.
  - Lower gas price.
- Web3.js offers utility functions to convert:
  - `web3.utils.hexToAscii` converts bytes to a string
  - `web3.utils.asciiToHex` converts a string to bytes

Lab: Auction DApp

- Build a DApp for an auction
- Details in Canvas