Cody Lamson

Full Stack Blockchain Web Developer

Phone: +49 176 8656 2888 **Email:** me@codylamson.com **Telegram:** @TovarishFin

Github: https://github.com/TovarishFin **Site:** https://codylamson.com **Location:** Berlin

Languages	Experience	Information
Javascript	Expert	Have built servers, clients, APIs using many different frameworks. Have used extensively for integration testing of solidity smart contracts. Very familiar with ES6. If there is one thing I know, it is JavaScript.
Solidity	Expert	Built 90% of the Brickblock smart contract ecosystem. Currently on mainnet. Have also built a variety of other open source smart contracts, including CryptoWeddings which is also on mainnet. Have had several auditing companies audit my code, none of them have shown severe or even medium issues. This is my passion. I work on smart contracts both professionally and for fun.
Rust	Beginner	Currently spending all of my free time getting better at rust. Currently working on an ethereum vanity address generator. Hoping to soon get my hands dirty with some lower level blockchain client code. Also spending time learning how to compile to web assembly, as this seems to have a future in many blockchain spaces.
Python	Basics	Know enough to play around with some python only blockchain projects.
Go	Basics	Have stopped pursuing go in favor of rust. Would still be happy to write some Go code professionally though.
SQL	Proficient	Have designed and implemented database structures for various apps including a prototype for Adidas. Experienced in writing complex queries for application consumption.

Frameworks etc.	Experience	Information
Truffle	Expert	Used for integration tests in JavaScript and unit tests in Solidity. I have a ton of experience writing tests with truffle. I like to think of it as the best way of getting to know and exploring the smart contract code that you write. Testing is especially important with smart contracts given the immutable nature of them.
Waffle	Proficient	Have been spending more time with Waffle lately instead of Truffle due to Ethers integration.

Frameworks etc.	Experience	Information
OpenZeppelin	Proficient	I use OpenZeppelin smart contracts whenever possible. I am more than happy to take advantage of code which has been extensively audited when working with smart contracts. I have plenty of experience extending and overriding various OpenZeppelin smart contracts. I generally know what each contract does and when to use them.
Web3.js	Proficient	I have used Web3 with node, Vue, React, inside Redux Sagas, in Vuex actions, and in pretty much any other place you could imagine where you need to connect to Ethereum.
Ethers	Profficient	I have used this on several professional projects as well as in most personal projects recently. It seems to have less bugs and be a bit easier to use as well.
Geth/Parity	Proficient	Have used in combination with Truffle to deploy smart contracts to various Ethereum networks including: ropsten, kovan, and mainnet. Have also done cold store deployments using these clients.
Node	Expert	I use node for all sorts of tooling setup as well as for servers. I have built fast and efficient APIs as well as servers meeting various unique needs. I have experience with server side rendering React apps. I have also contributed to a Node tool for generating/signing offline ethereum transactions.
React	Expert	Have created more dapps than I can remember at this point using react. Also familiar with newer hooks/context api.
Redux & Redux Saga	Expert	I reach for Redux whenever the React app is getting more complex. Redux Saga keeps things sane and organized when dealing with async actions.
Mocha & Chai	Proficient	Used for testing smart contracts as well as servers, clients, and other web applications. Testing Testing Testing. Test all the things!
Material-UI	Proficient	I reach for this component library every time when a React project that has a quick deadline.
Vue/Vuex/Router	Expert	I have been using Vue for any personal projects lately and have managed to also work it into my professional life. I generally like it due to the ease of use and lack of decisions you need to make. It feels like the good old days of AngularJS with batteries included. The slick functional style of React might be missing, but it definitely gets the job done.
Vuetify	Expert	My go to component library for Vue.

Recent Achievements

- Have written 15 smart contracts which have passed audits by several auditors, including Consensys Diligence and SmartDec. No Critical or Moderate vulnerabilities were found. They are now all deployed to mainnet without incident.
- Have several projects with mainnet deployed contracts.
- Have an article published by coinmonks on how solidity handles storage.

- Have completed all hacking challenges on capturetheether.com. Not many in this field have the low level knowledge necessary to do this.
- Created a new pattern for going over the gas limit for smart contract deployment via multi stage delegate calls using what I like to call "non-sequential storage".
- Was featured on BlockTV.com with an interview for my CryptoWeddings dapp, where my wife and I got married on the blockchain and streamed it live on twitch using my dapp.

LINKS:

- audits: https://github.com/brickblock-io/smart-contracts/tree/master/audits
- Hacking challenges: https://capturetheether.com/leaderboard (I am TovarishFin)
- coinmonks article: https://medium.com/coinmonks/a-practical-walkthrough-smart-contract-storage-d3383360ea1b
- blockTV interview: https://blocktv.com/watch/2019-07-18/5d307a2c6f719

Recent Projects:

Porsche Digital Lab:

I have been working at Porsche Digital Lab as a full-time employee for the first time in many, many years. I am responsible for working on various experimental blockchain projects as well as educating others on best practices when it comes to smart contracts and web related code. I was hired as a full time employee after working freelance in order to help bring different experimental blockchain projects to production.

Brickblock:

Wrote 90% of the code for the smart contract ecosystem which comprises of 15 smart contracts. Wrote over 500 tests for said smart contracts. Contributed to a cold store solution which creates and signs transactions offline in order to prevent private keys from ever touching an internet connected computer.

Deployed contracts via cold-store solution. Performed 1000's of transactions interacting with Ethereum mainnet. Was a significant contributor to the DApp which will act as a platform for interacting with the smart contracts after the audit is complete.

main page: https://brickblock.io
GitHub: https://github.com/brickblock-io/smart-contracts

CryptoWeddings:

A personal project I made with my wife. Allows two addresses to get married on the blockchain. The smart contracts facilitate marriage, divorce, receiving wedding gifts in the form of ether, and uploading wedding photo.

The DApp is built on Vue, Vuex, Vue Router, Vuetify, and Ethers. I was entirely responsible for everything technical.

The contracts have been deployed to mainnet and the dapp is available for anyone to use. My wife and I are married on there and were the first users.

MetaMask is optional due to the web based wallet I built as part of CryptoWeddings.

site: https://cryptoweddings.io
GitHub: https://github.com/TovarishFin/crypto-weddings-smart-contracts

BitcoinHex:

Took over writing the smart contracts at an early stage. Updated and finished contracts according to needed specifications and security requirements. Wrote the tests for said contracts.

Contracts use bitcoin block root merkle hash and elliptic curve recovery in order to facilitate claiming tokens based on Bitcoin UTXOs at a specified block number. The contracts also facilitate trustless interest. Special care was needed in order to ensure no integer overflows or gas limit issues were encountered when performing compound interest.

The contracts have since been completely rewritten due to changing requirements.
main page: https://bitcoinhex.com
GitHub: https://github.com/BitcoinHEX/contract

TV-TWO:

Was responsible for creating an ERC20 token which would use µRaiden for token transfers upon a user consuming or creating content. Advertisers would also use tokens in order to advertise. Created the start of the project and handed over to a friend who had matching competencies due to my own time constraints due to my project load at the time.

site: https://tv-two.com
GitHub: https://github.com/tvtwocom/contracts