Cody Lamson

Full Stack Blockchain Web Developer

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

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

Languages	Experience	Information
Javascript	6 years	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	1.5 years	Have built 90% of the upcoming Brickblock smart contract ecosystem. Have also built a variety of other open source smart contracts, some of them using oraclize. 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.
Go	4 months	This is a language that I really want to get into but haven't had the time to create any significant projects with it. Currently I am using it to create a bitcoin blockchain clone. I want to eventually look into rebuilding geth from scratch. It would be a great learning experience to see how the Ethereum EVM works and get a better grasp on go as well.
Python	6 months	Most recently I have used it for using μRaiden in order to make off-chain ERC20 token transfers on the Ethereum blockchain.
SQL	1.5 years	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	1.5 years	Truffle has many uses, but the main thing I use this for is 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.
OpenZeppelin	1.5 years	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, including ERC20 and Ownable contracts to name a few.
Web3	1.5 years	Have used in various React related projects, both personal and professional. Most Notably for DApp related functionality relating to smart contracts and blockchain interaction using web3.

Geth/Parity	1.5 years	Have used in combination with truffle to deploy smart contracts to various ethereum networks in including: ropsten, kovan, and mainnet
Node	5 years	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 offline transactions.
React	2 years	Have created several DApps (distributed applications) running on Ethereum blockchain using es6 syntax.
Redux/Redux-Saga	2 years	Have used in various React related projects, both personal and professional. Most Notably for DApp related functionality relating to smart contracts and blockchain interaction using web3.
Mocha & Chai	3 years	Used for testing smart contracts as well as servers, clients, and other web applications.
Material-UI	2.5 years	My go to for personal projects.

Core Competencies	**Information**
Upgradeable Contracts	Implemented a delegate call proxy factory pattern for deploying multiple instances of a contract in a gas efficient manner. Took the concept further and chained delegate calls in order to work around gas limits for a huge smart contract.
Inline Assembly	Have gotten pretty deep into inline assembly in order to save on gas costs for smart contracts when absolutely needed. Was also needed for delegate call proxies. Have used for many other things that just cant be done with regular solidity.
ERC20 Tokens	Plenty of experience with this industry standard. Have an extended ERC20 token that I wrote audited and deployed.
ERC721 Tokens	Have been playing around with this in my free time lately. Quite familiar with the concept and am working on a personal project involving this standard in my free time for fun.
Dividend Patterns	Implemented a gas efficient method of distributing dividends which was not common knowledge when implemented.
Crowdsale Contracts	Also very familiar with this pattern and have written different versions with different requirements.
Oraclze Integration	Have used for random numbers as well as complex API calls requiring inline assembly to parse.
µRaiden	Have implemented ERC20 tokens which use µRaiden for off-chain token transfers. Was my reason for getting back into python.
Merkle Trees	Have used Bitcoin block root merkle hashes for verifying Bitcoin UTXOs in an Ethereum smart contract. Have a pretty good understanding on how they work. Am also currently working on creating a Bitcoin clone in Go which also requires merkle tree knowledge.
DApps	Have built/contributed to 3 different DApps so far.
IPFS	Have used for large data storage requirements for smart contracts at Brickblock.

Recent Achievements

- Have written 2 smart contracts passing audits by multiple auditors, including Consensys Diligence.
- 15 more contracts are currently being audited by Consensys Diligence. So far nothing but good feedback.
- 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".

LINKS:
audits: https://github.com/brickblock-io/smart-contracts/tree/master/audits
Hacking challenges: https://capturetheether.com (I am TovarishFin)
coinmonks article: https://medium.com/coinmonks/a-practical-walkthrough-smart-contract-storage-d3383360ea1b

Recent Projects:

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 an ERC20 token to mainnet 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.

LINKS:
main page: https://brickblock.io
platform: https://platform.brickblock.io
github: https://github.com/brickblock-io/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.

LINKS:
main page: https://bitcoinhex.com
github: https://github.com/BitcoinHEX/contract

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 via IPFS.

The DApp is built on React, Redux, Redux-Saga, Material-UI, and Web3. I was entirely responsible for everything technical.

Due to GDPR issues, the project's future is uncertain.

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

---

Loop:

A short term project where I was in charge of creating an ERC20 token as well as facilitating transfers via µRaiden. Also built a simple proof of concept tool to create and store private keys via node or the client. All work is closed source sadly and the project got put on hold. Nothing to really show publicly here. Can show parts of the code upon request.

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.

LINKS:
site: https://tv-two.com/
github: https://github.com/tvtwocom/contracts/graphs/contributors

---

Adidas:

Was responsible for creating a modern API using Node and MongoDB which used data from a previous Adidas project in order to provide and store all data needed for an android fitness application. Also created a few web applications for data entry. This code is closed source.

GUUEY:

Created a care home management software solution which incorporates many of the newest features of web apps into it's design. The database runs in real time, allowing users to see all updates to service user's records in real time. Other interesting features of this web app include: notification system along with push notifications, access levels, privacy settings, care home reporting, subscription payments, medication scheduling, version control, and visualized statistics. The company has since gone under and is no longer available online. The code is closed source. Sample code can be shown upon request.