Name: Johan Lajili
Website: http://lajili.com Recommended visit
LinkedIn:
https://uk.linkedin.com/in/johanlajili/en Mail:
johan.lajili@gmail.com

Looking for: Contract opportunities, JavaScript front-end / full-stack in London.

Personal profile:

I'm a Senior JavaScript developer, with more than 8 years of experience and a master's degree in "Game Design and Programming". I focus on highly interactive web applications, with a complex front-end.
I am now in the market for contracting opportunities.

Skills

- **JavaScript**: ES2015+, ES6, ES5. Thorough understanding of all the feature of the language, a variety of architecture patterns and at least basic knowledge of the most popular library/frameworks.
- **React Ecosystem**: Knowledge of ReactJS (including latest changes like hooks), ReactRouter, Redux, Flux, ImmutableJS, Preact etc.
- **Other runtime JS libraries** PixiJS, Three.js, AngularJS, KnockoutJS, Vue
- **Build time JS libraries:** gulp, webpack, browserify, babel, grunt, node filesystem API, Electron, templating library (jade, mustache), esLint (including writing custom rules)
- **Associated web technologies:** NodeJS, HTML, CSS, Sass
- **Git** (Git flow, merge vs rebase, command line as well as IDEs)
- **Soft skills:** Agile Development (Scrum master for two years in a team that takes scrum very seriously), writing DRY code, Unit Testing, Automation Testing, Communication, Teaching
- **Also had decent exposure to:** C# (Unity), Ruby On Rails
- **Softwares:** Adobe Design Suite, Blender, Esoteric software's Spine, 2D Texture Packer, Charles, Tiled Map Editor, Fusion 360:

Work experience

CMC Markets – 09/2018 – present
JavaScript Consultancy [Contracting Role]

About the Company
CMC Markets is a leading global provider of financial services. They have created an award-winning Trading platform used all over the world, using web technologies. Their platform has a gigantic amount of features, and satisfies the exigences of Traders needs.

Experience and Goals Achieved:

- I provided my expertise in JavaScript to help with their transition from Angular to React, ensuring good practices were met.
- I implemented the "Welcome Experience", which is what the user will see when they use the app for the first time when using the demo application. With its shortcuts to effective trading based on the user profile and needs and smooth interface, it is hoped to increase conversion massively.
- Improved performance and stability by chasing memory leaks, finding some core issue that had been present in the code base for more than 3 years. This series of fix saved dozens of megabytes of memory on very simple scenarios, in a very measurable way.
- Worked on new features that I cannot describe here as they are not rolled out in production yet.

Gamesys – 01/2016 – 08/2018
Games Developer (promoted to Senior in 10/2017)

About the Company
Gamesys is an online gambling operator with ventures around the world (UK, Spain, USA) and 10s of millions of users. They own websites such as Jackpotjoy, Heart Bingo, Virgin Games and many more. All their games are now developed in HTML5, and work on all platforms.

Experience and Goals Achieved:

- I joined as part of the Roulette team. Besides working on the game features, I also tackled a major issue. The backend was not installable on front-end dev machines, so they were all dependent on the environment working. This resulted in many man-days lost to a non-working backend. I decided to create a fake backend, using NodeJS and Socket.io, and from that point devs where completely independent of environment issues. And as an added bonus, they could work on new features before they were implemented by the backend.
- I then moved to a different team making more visually ambitious slot games. I had to update some of their oldest games with new features and I quickly saw the problem. While there was an intent to share some code in between games, it was shy and came short of solving any issue, actually creating more problem than it solved. We discussed it and decided to create a new engine from the ground up for
  our next game: the Warp Engine.
- The first thing I did on the Warp Engine was to create a building system worthy of the name. While in our previous games, you could wait a full minute for a build, by using Webpack DLLs, gulp in a smart way and browserify, I was able to make development builds pretty much instantaneous.
- I then created the first components and actually made the game. Implement along the way Spine Animations and particles, it was the most beautiful game made by our team so far.
- I tracked down every memory leak from the game to a point where you can leave it to spin on its own for a full weekend, and it would still work and have the same memory usage on Monday.
- As more devs joined our team, I took a role of “warp engine evangelist” and started helping them get up to speed with Reactive programming and the engine in general, all the while adding new features. Under my supervision, we created 7 now released game, and more that are still unreleased, all using the same engine, to a point where most of those game don’t have code at all. They are all around the million users monthly.
- Thanks to the dynamic nature of JavaScript, I was able to create a “plumbing” system that dealt with every external call, such as updating the balance or telling the player he needs to take a break. While those behaviours are slightly different per family of games, they are mainly common. A clever use of inheritance and the PubSub pattern allowed me to make those connections as common as they can be, from the ones that are used in every game to the ones that are unique to one specific title.
- I applied TDD practices to the development of the game, which proved useful many times in considering all the scenarios and preventing bugs.
- Halfway through I became Scrum Master. Previously, people barely knew what other people in the team where working on, sprint commitments were a joke and the morning stand-up depressing. By using positive reinforcement (rewards for finishing the sprint early), grooming the backlog properly and making the story clearer, as well as printing each task in color, with associated screenshot and tags, I was able to make the whole team care about the sprint. The morale went up a lot, so did the productivity of the team and our ability to specify deadlines.

URL: http://www.gamesyscorporate.com/

CINIME – 02/2015 – 12/2015
JavaScript Developer*

About the Company
Cinime was a relatively big startup (70 employees worldwide). Their mission is to create an application for cinema goers that allows them,
- Before the movie, to browse movies and book tickets in any Cinema,
- During the séance, to play interactive games that are in sync with an advertisement in the big screen in the same way a musical game is in sync with the music. Users can then win items based on their performance (pop-corn, cinema tickets or objects related to the advertisement),
- After the séance, to receive offers linked to the movie seen. Avengers themed hard drives after "Avengers" for instance.

Experience and Goals Achieved:

- In Cinime, I was the Subject Matter Expert for everything that is in a webview. That meant mostly HTML games and the movie feed. The rest of the application was made in Unity, though the new version (still in development) is in Native + HTML for games.
- I singled handedly developed 7 HTML games, all in sync with a content on screen. They range from quizzes (were the question are asked on the big screen and you have to answer on your phone) to spot-the-object games.
- They are all sharing a core repository that is used for every generic behaviour: scene management system, syncing with the advertisement, resizing, communication with the app, building…
- They use EcmaScript 6 (Babel), Browserify, GSAP (animation tool), Sass. The building system (gulp) is on the core so each game much follows a proper interface to work properly.
- The core also ships with CiniView, a tool I made that is pretty much an emulation of the Unity App. This way you don't have to build to the application to be able to test that the game is indeed in sync with the video. It's also a great tool for demos and presentations as it allows one to display the advertisement and the game on one screen, which helps a lot if you are trying to show the application to 20 persons.
- I have technically designed and spec all of this, implemented it and heavily participated in the game design of the games as well.
- I also have worked on a movie feed (list of movies, in which you can see the details for each movie and the showtimes). This uses React and Flux to optimise performance as it is an AJAX intensive application with a very big DOM tree. It features a gulp building system that allows one to test while being outside of the application (giving fake data) for staging and a clean release for production.
- Lastly, I also worked on a NodeJS / PhantomJS / CasperJS scraper system that return the booking page of a cinema based on a movie name, location and time.

URL: https://web.archive.org/web/20160322141859/http://www.cinime.com/ (the company has closed so to see the website we need the wayback machine)

** : The contract stated "HTML Developer", as a reference to the HTML ecosystem (JavaScript + HTML + CSS) to differentiate from other developers who were in the Unity Ecosystem. In practice the role was one of a Front-End JavaScript developer.*

INNES – 02/2014 – 02/2015
Software Engineer

About the Company
Innes is a medium sized company that focus on its own products for a B2B market.
Their main products are “players”, devices that allow one to display information on screens. Train Stations, Estate agents, Museums and other facilities use them to display various information, from itinerary and company news to exhibitions, advertisement and video clips.
INNES also create the software that allows to select what information you are going to display on screen and to publish it, even if you have 1000 players in the country and you want each one to have specificities.

Experience and Goals Achieved:

- During my time at INNES, I have worked on Plug’n Cast. This CMS basically allows one to manage a library of medias and screen layouts, and arrange all that in calendars. Once you have your calendar, you can publish it directly to your players with a single click.
- I worked mostly on the front-end written in JavaScript (using KnockoutJS and a jQueryUI plugin named IgniteUI). Although the project was already advanced when I arrived, I used my previous experiences to drive the refactoring of the application. For instance, I made a lot of Custom Bindings (kind of widget of the KnockoutJS world) to allow DRY module development. I also refactored a 5000 lines spaghetti code working as a fork of a library into several modules divided in business logic and view that extend the library, instead of hard coding inside it.
- I used my knowledge of CSS3 including media queries and flexbox to improve greatly the responsivity of the application. Even though it is not made for mobile phones, it can be stretch from a netbook screen to a Full HD screen and occupy all the space available each time. Using
  CSS3 allowed me to remove a lot of the JavaScript resize methods that were heavy in performance terms.
- The Widget library we use, IgniteUI, was choosed for its native compatibility with KnockoutJS. But since it was far from perfect, I extended it to correct some bugs, improve performances and develop new features.
- I also worked on the publication server that transform the database for the current project into a few xml files and scripts that will be read by the devices. That include resolving the relational dependences of each module, creating the manifest file and the XPF file that describe the different playlists, events and tasks, and managing the different devices. For instance, you can publish a playlist that will show different medias depending on the geographical region (using user-made variables) or on the current date (using XPath functions).
- I worked on the CMS backend written in XQuery 3.0 with a Zorba implementation. XQuery is a functional language that allows to interrogate XML based databases. Although I did not know of the existence of this language when I was hired, I learned from books and spec to replace a co-worker who had had personal issues, and quickly became the person in charge of the XQuery in the team.
- I suggested to change the version control system from SVN to Git, and helped the company doing so. It has bring a much better workflow, increasing the granularity of commits, helped people to work together instead of force them to work on different parts of the project, given a sense to branches, which were not used with SVN and are now used at their full potential (feature branches and alpha, beta, production branches) and allow to bring the building system into the versioning system, automating the compilation tasks.
- I implemented the migration system from end to end, starting with the shell script that replace the version number during the compilation to the xquery methods that perform any kind of migration we need.
- I learned how to follow a rigorous workflow when it comes to task management, following an agile methodology. That mean not to correct a bug as soon as I see it, but specify it in Bugzilla and decide of its importance.

URL: http://innes.fr

Toxicode – 02/2013 – 02/2014
Web Game Developer

About the Company
Toxicode is a recent startup with a quick expansion (10 employees after 2 years of activity). Its specificity is to be entirely based on teleworking. The employees are scattered around France and even China for one. Toxicode splits the work time in two main parts: services to other companies and experiments. Some of these experiments are a one day thing, others are developed for a year or more. They include games to teach programming to children, multiplayers online board games…

Experience and Goals Achieved:

- I worked on an immense variety of languages and platform, including DOM based games that had to work on both IE7 and tablets, a canvas based Multiplayer online game using SocketIO, a mobile statistic application using KineticJS, ruby on rails server sides and html CSS static websites…
- We had a lot of autonomy and it was the occasion to discover the life of a freelancer. We had to look for new business opportunities on social networks for instance, and I sometime handled every part of a project from the communication with the client to the development of its application, including the contract making.
- I realized a lean startup experiment, to test if making HTML5 mobile games with the idea to sell to publishers was financially interesting. So I made a casual mobile game in three days, got a list of publishers, and sent to each one a mail with a link to the demo (including a token in the URL that allowed us to trace the publisher) and an image from our server that was also traced. Though we concluded that it was not worth it (most publisher didn't even test the game, and the one who proposed to buy it aked for too much modifications for it to be financially worth it), it was an interesting experiment from which we have learned a lot.

URL: http://toxicode.fr

Technologies Used:
JavaScript OOP, Ajax, HTML5, HTML5 Canvas, jQuery, Requirejs, QUnit, CSS3, GIT, Ruby, REST, KineticJS, PIXI.JS, Impact.js, NodeJS, Socket.io,

Learnscaper – 09/2010 – 02/2013
Serious game manager / developer

About the Company
Learnscaper was a start-up founded a few month before I join it as an employee. Its goal was to create a serious game authoring system, allowing companies to create their own serious game, based on interactive videos. Unfortunately, the company had to close a month ago, but its project are maintained by its US alter ego.

Experience and Goals Achieved:

- Although I was young, I managed to scale the different project to the skills that were available in the company. In the first year, we created interactive videos using a product realized for us by partners, in Silverlight. We also created a serious board game, to teach negotiation.
- The second year, since I became more fluent in JavaScript as well as my co-worker, we created a serious game for Orange. The serious game was DOM based (IE6 compatibility was required) and quite long (1 hour of life time).
- The third year, I recreated the Silverlight product in HTML5 to add tablets / phone compatibility, allow streaming and remove the necessity of a plugin. We also started recreating the authoring tool in HTML5, but that task was outsourced in Bangladesh.
- I was in charge of the development in Bengladesh, including the specifications, technical choices, code review and team meeting.

URL: http://learnscaper.com

Technologies Used:
JavaScript OOP, Ajax, HTML5, NodeJS

Education / Awards:

Game Design and Programming – Master's degree – 2013
Studied creation of games using Web Technologies and Unity, as well as associated skills.
Isart Digital - France

PROJECTS

Next Gen Trading Platform – CMC Markets – Released
The trading platorm by CMC Markets is a gigantic application. The JavaScript version is over 5 years old, being maintained by a team of 8 senior developers. It is declined in two main families and over 20 branded version, which feature different style and behavior. The application is initially organized as an Angular 1 application, but is being slowly converted towards the React ecosystem, while keeping new features coming and of course without sacrificing on the quality of the application.

Technologies Used:
JavaScript ES2015, Angular, React, Webpack, Grunt

Warp Engine and associated Games – GAMESYS – Released
Jin's bouncing Wilds, Wonders Of The Deep, Paper Wins, In It For The Monet, Paper Wins Jackpot, Family
Feud, and more that are not public yet.

Each game is composed of a shared core (the WARP engine) as well as custom configuration files and more custom code if necessary. Using **React-type components** (both presentational and containers) and our custom implementation of a Virtual DOM made specifically for games, they are all fully tested by 4 QA and the automation system. The engine has a lot of features to help development (live-reload, live-refresh, accelerating or slowing down time, fake backend, unit testing, automation hooks, different level of logging etc.). It also is renderer agnostic, and tomorrow we could replace PIXI with the next best thing without having to reimplement a single game. We were also able to swap PIXI for three.js, and with very little modification to the game (specify 3d objects instead of sprite), we had the same game in 3D.

Technologies Used:
JavaScript ES2015, Pixi.js, Redux, ImmutableJS, Gulp, Babel, Webpack, Mocha, Waud.

Spine Toolkit – GAMESYS – Released
Spine is a animation software made by EsotericSoftware that is very powerful. But it also has a lot of flaws, usually dumbing down to the fact that playing an animation in game is not the same as playing it in the editor. To cut short on the endless back and forth between developers and animators, I developed a tool to allow animators to test their animation themselves. It's a native application, made using Electron, React and Javascript, and is packed with features to help animators. It can detect automatically most mistakes in animations, provides performance benchmark and is auto-updating. This is an application I made on my own, from the idea to the final implementation, and that is now used by the whole design department in Gamesys.

Technologies Used:
JavaScript ES2015, React, Electron, Pixi.js, gulp, babel.

Multiplayer Roulette – GAMESYS – Released
Multiplayer Roulette is the project I joined on route when I first joined the company. Unlike the other projects, I didn't have any say in the technical decisions as I arrived quite late in its development process. It's a multiplayer roulette game, using websockets for real time interaction between players. It uses Haxe, which is a type language compiling into JavaScript, similar to TypeScript. Besides fixing bugs, I created a fake backend, in NodeJS, to be able to play the game locally independently of the real backend actually working. This saved multiple man-days that would have been lost to environment problems.

Technologies Used:
Haxe, Pixi.js, grunt, NodeJS, SocketIO.

Live Roulette – GAMESYS – Released
Live Roulette is another roulette games, with the twist that it connects to an actual live studio where a real person throws the ball in the roulette and announce the results. I was there for the first couple month of the project and implemented the Responsive Design system, taking inspiration into CSS rules, but applying them to a PIXI game. I also implemented the video stream displayed on the screen.

Technologies Used:
Haxe, Pixi.js, grunt, NodeJS, SocketIO

"HTML Games" – CINIME – Released
They were 7 HTML games developed, but since they use the same technology stack, I prefer to put them in one project. The seven games are finished but new ones were created frequently. The games would be loaded via an iframe in a Unity application. The unity application, using audio-waterproofing technology, would help the game synchronise with an advertisement happening on a different device. Each games was ultimately a way to interact with the advertisement. Examples include musical quizzes, spot the object games and a space battle.

Technologies Used:
JavaScript OOP (EcmaScript 6), HTML, CSS3, Gulp, Browserify, jQuery, GSAP, Git, NPM, Sass

---

Movie Feed – CINIME – Released
The movie feed is the part of the application that allows you to browse movies (in the form of a series of posters you can click on, then see the movie details and the showtimes for the cinema around you.

Technologies Used:
JavaScript OOP HTML, CSS3, Gulp, Browserify, React, Flux, Sass

---

Plug'n Cast – INNES – Released
Plug'n Cast is a CMS that allow user to specify what content is to be played on Digital Signage screens (such as TVs in train stations). It allows to specify content to be played at specific time or under special conditions.
The application is divided in several layers:

- The front end, in HTML5, is a single page application. Using requireJS to load each module. It is divided in view modules, business layer and services that allows to communicate with the server. The view modules are each defined by a JavaScript controller file, a JavaScript ViewModel file, an HTML file and a CSS file, and can include any number of submodules or subsubmodules. The business models follow an interface that is known by the services and by the view. And the services are singleton with helper to populate the models from request to the server using AJAX methods. Depending on the type of data, we will communicate with the filesystem directly using WebDav or with the three different parts of the server:
- The CMS server is written in XQuery, and is basically composed of generic methods for each collection (calendars, layouts etc.) who are unaware of the content of those collection, and specific methods that allow for instance to delete any reference to a calendar in the same time we delete the calendar.
- The Publication server is written in JavaScript EcmaScript 6, and transform the information on the XQuery DataBase into the different files and script required by the player to play the publication.
- The hardware server is written in C++ and allows to communicate with the players directly, in order to copy the published files to them, to know their state and to check any error, as well as to manage the preferences.

I worked on every part of the application except for the C++ serverside.

Technologies Used:
JavaScript OOP, HTML5, CSS3, EcmaScript6, XQuery, XML, XPath, WebDav, Ajax, jQuery, KnockoutJS, jQueryUI, IgniteUI, RequireJS , promises, RPC, SVN, GIT

“Freaky Monster Show” – Slak games – Released
Freaky Monster show is an HTML5 multiplayer real-time online game. It features up to 8 players in a 2D platform based stage shooting at one another.

- The online feature works with nodeJS and socket.io. Each time a player makes an action, it sends a request to the server that redistributes it to the other players. In case of a difference in the outcome of each client, one of the player is considered the host. Event-driven programming is used to make that as transparent as possible.
  - Impact.js is used to display the sprites and calculate the collisions.

Technologies Used: JavaScript, HTML5 Canvas, Impact.js, NodeJS, Socket.io

---

“Bot Arena” – Toxicode – Released
Bot Arena is a JavaScript game where players must code an A.I in JavaScript to complete levels. The application is made to be extremely modular, and for instance, each “world” (set of levels) can implement its own rendering system and rules. The basic world is a grid system in DOM, but another one is a continuous map in Canvas. I created, for fun, a 3D world with a threeJS rendering and based on hexagonal grids.

Technologies Used: JavaScript, HTML5, ThreeJS, nodeJS

“Zeegzag” – Toxicode – Released
Zeegzag is the mobile HTML5 game I made alone while I was at toxicode to test if it was worth it. It follows an architecture I consider interesting, extension. Every class takes a single argument, which is the object that will extend them. Using _.extend and Compose.js, I’m able to create a really versatile architecture that is perfect for project of that envergure. It allows fast development and DRY code. Please check the project “fierysnake” on my github account here: https://github.com/iraldir/fierysnake which is the sequel of this game, based on the same architecture.

Technologies Used: JavaScript, HTML5, PIXI.js, Underscore.js

“Chatscaper” – Learnscaper- Released
Chatscaper is the authoring tool that allows to create interactive dialog. The user can create an arborescence of dialogues, specify voices or video for each node of the conversion, as well as choices and variables. I did not program the front-end myself, but I managed the outsourced developer and have written the specifications.
Technologies Used: JavaScript, HTML5 Canvas, jQuery.

“Gamescaper” – Learnscaper – Released
Gamescaper is the interactive video player. Basically, it’s able to read a conversional tree written in a JSON file, display the video/audio/images for each node, propose the choices to the player and go on till the end of the dialogue. It’s fully customisable with CSS and a JSON settings file. I realized it entirely myself.

Technologies Used: JavaScript, HTML5

“Gamescaper Online” – Learnscaper – Released
Gamescaper Online is what makes the link between the two applications. Even though you can export your project in Chatscaper, you have a whole bunch of file to copy, rename and modify to make it work in Gamescaper. Gamescaper Online is a server API that will compile the Chatscaper project into a Gamescaper Package available online.

Technologies Used: JavaScript, HTML5