Online Game Generator Documentation

## Purpose of the development

The reason the app concept was created was because of the last quizzes created by the co-op student before me. He had built a very nice web ‘app’ that did the same basics that I have managed to accomplish with my system, however there were a few things that made me feel that the previous system was not the platform that I should work from continuing on. The drag and drop had an issue where the items would disappear (they would be placed into other items). This issue I resolved for the previously built apps and was not the main reason. The main reason I decided to rebuild it was because of optimization. The previous dev built the quizzes off of bootstrap and jQuery, which are both great. However the reasoning for using them in this case was very debatable to me. He used jQuery to have fade in animations on elements (2-5 seconds each, was asked to remove) and for a single function call. I decided that animations and a single function were not worth another 90Kb of download to the client for a true and false question. The file size of bootstrap was the main reason for deciding on ditching it. It fell just short of 800Kb in file size with a 98Kb download to the client and was not serving any important purpose to the system. The final reason of why I decided to redo the project was because of duplication. Each game created had its own copy of bootstrap and jQuery. So when I was asked to reduce bootstraps default padding and to remove the fade in’s, I had to go into each game and modify the exact same files in the same locations for the same thing.

So the plan was to rewrite take the html, and replace bootstrap with my own css and patch the JavaScript without jQuery. What ended up happening was me salvaging the JavaScript that was decent from his old code (He is a pretty good coder, credit where credit is due), and modularizing the code with my own plan in mind. The goal was to enhance the functionality available in the system and making the main engine able to accept new game modes that anyone decides to build. The plan continued with the idea of storing the common JavaScript on a server. This way only the html and the specific quiz content are stored locally.

The reason the app came into the mix is because the quizzes were only able to made when someone with JavaScript experience. I wanted to make it accessible to anyone who thinks that a quiz would improve a course. So the idea of generating the games with a more accessible interface was the way I wished to continue.

## Things you should do to maintain/enhance the project:

READ THE README’s IN THE DEV AND PRODUCTION FOLDERS.

Build into the jsFramework.js for new game modes. It is extremely flexible and should provide you with a stable platform to run off of.

If there is a function that you feel that has a purpose across multiple game modes and seems like a core feature, please feel free to include it into the jsFramework. However if it is not a core function, add it as a plugin.

Update this document with your changes below and indicate that there is a change in developer, this way there is a good record of changes and what each dev does.

Please comment your code, and try to explain what is going on. If I fail to express the code properly, I apologize I have tried to express the code without writing a novel. If something is not commented, the code should either be simple enough to extract what it is doing or the function name explains the purpose clearly. If you feel the code should have been commented, if you figure it out, please fill in on where my decision was not correct.

Place loaded file name in form title? (Would prove useful)

## Reference guide to what things are called

Throughout development I have managed to swap between a few different names for the same things. So I figured I would make a quick list of what names refer to what elements.

MULTIPLE CHOICE:

* Question Simple
* T&F
* MC

DRAG AND DROP:

* Drag n drop
* Question Complex
* Dnd

QUIZ = GAME

## Chosen Optimizations

I have chosen to move the LessonGame.js and the unique styles.css into the index.html directly to remove files to download to the client in the production version of the quizzes as this improves load times by about 15%.

I have removed all whitespace from the index.html pages contents as I have found it creates a rough 2.5% file reduction and makes load times about 50% faster.

I have merged the popup.js and the mcf.js into a single mpf.js file and minified all the root js and css files to reduce file sizes and to also remove files to download, reducing both required bandwidth and requests needed to download the files. Increasing performance by a reasonable amount.

Running with html 5 components and not leveraging any external libraries has reduced client download sizes dramatically.

Using CSS3 animations to add some movement to the quizzes without bloating the system.

I am downloading the JavaScript asynchronous to allow the rest of the document load as the javascript is downloaded. It has shown no noticeable difference when all files are local but once the files are updated server side I think this will make some difference.