**JS Text Exercise Documentation**

# Overview

The purpose of the test is to demonstrate good architectural design. The main aspects of such design will be:

* Document Object Model (DOM)
* Able to modify the content of an an HTML Tag Element dynamically using JavaScript
* Able to use a JS function that can ‘prompt’ user and provide data into a dialog box to be used in the page
* Able to add an event to an HTML element
* Look up HTML forms on the web to figure out how to also capture information and how to use JS to reset and submit to the form
* JavaScript’s ability to launch a script when page is loaded
* Ability to validate if an input can be safely used as a date
* JavaScript String escape characters for hard return, etc..
* Events for text field

# Constraints

* Do your research and if necessary, workout mini proof of concepts sample code (PoC), you can use <https://jsfiddle.net/> or I would recommend you actually archives your PoCs as well to increase your online portfolio in GitHub
* Ensure your scaffolding works
* Ensure the most basic requirements are met
* Ensure your unit test(s) are functional
* Always ensure your unit tests are up-to-date
* Have fun!!

# Background

I’ve never developed web based solutions.Only working on small school projects and PCS

Considering the amount of time I had for this particular challenge, I tackled it based on the order of block on the picture for project provided by the teacher.

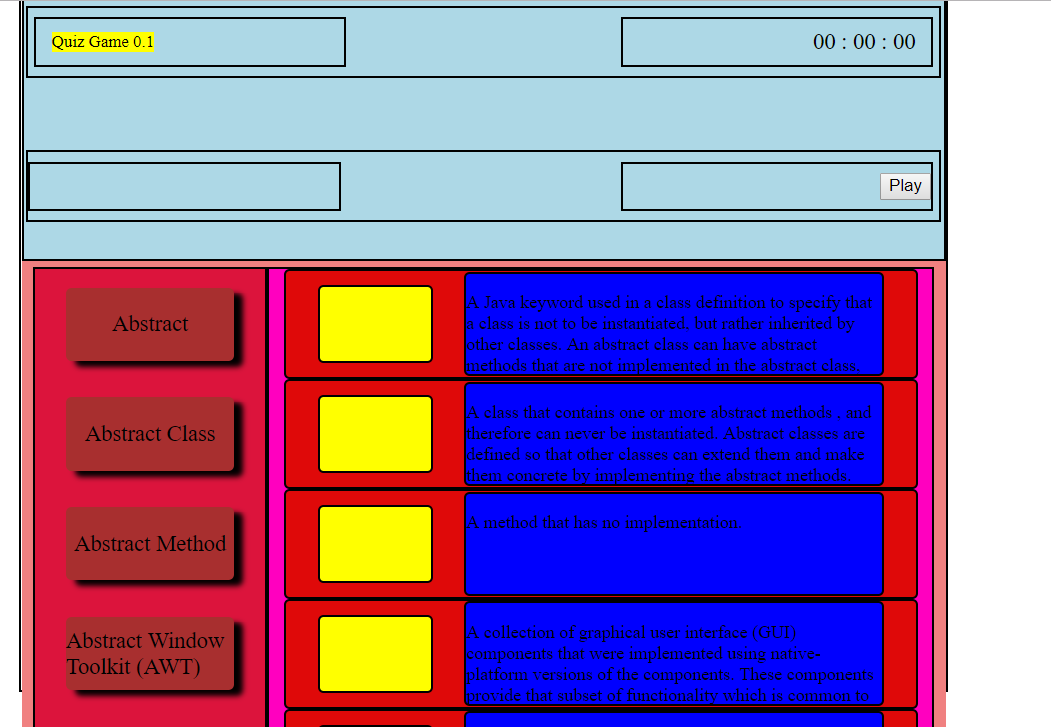
1. HTML scaffolding
2. Game title
3. Timer
4. Button
5. And all respective gameboard area.

As such, I used an building block like approach in the design and development of this project.

I tackled the project as follows:

1. index.html (1 day)
2. Timer feature (4 hours)
3. DHTML(2 days)
4. Drag and drop (30 minutes)
5. styles.css (30 minutes)

# App snip



# Potential Improvement

There are many things that can be done from this work in order to improve upon it. The list is far from exhaustive by any means.

1. Getting a development course that can actually teach me the tools and development methods available for developer and being able to properly access elements of the project that I was not able to reach even when I have working pseudocode and just couldn’t implement because I didn’t know how.
2. Better css
3. adding unit test and functional test
4. improving on the API provided, the techniques for adding html and refreshing views are very rudimentary
5. Design patterns
6. Look and feel of the app..
7. Usability.
8. Code optimization

Joel Reyes