Applications Support Engineer Tech Test

Learnosity's APIs are extensible through the use of Custom Questions. Custom Questions allow a developer to build a JavaScript-based question type that accomplishes one or more goals our existing question types don't. To evaluate your critical thinking, ability to follow documentation, and coding skills, **we'd like you to create a custom question type.**

There are two essential parts to a custom question.

1. The logic behind the question itself. Our example builds an analog clock and validates a matching student's 24-hour time answer.
2. The integration of the custom question into our API stacks. For this tech test, we'll skip authoring and reporting, and focus exclusively on assessment. You will create the custom question and then use the Questions API to create a single-question assessment for testing.

To support this effort, we'll provide you with two levels of scaffolding.

The first will be the infrastructure of the custom question definition, in JavaScript and CSS. You'll add your question logic and styling to these files. The second is a simple, pre-written assessment using the Questions API. The Questions API is our lowest-level API and is responsible for all question rendering within our authoring, assessment, and reporting stacks. However, it can also be initialized directly in rare use-cases. While not common for our average customer, it provides a very direct route to assessment, so it can be ideal for testing scenarios like this. Essentially, it allows us to set aside the higher-level APIs for assessment, such as the Items API, which a customer would typically initialize.

We've included some documentation links below, so you can get a general idea of how the Questions API works. It will be useful to review these links, but the scaffolding provided will not require you to fully understand the API. What we're really focusing on is the custom question as a means of looking at your JavaScript work. So feel free to use these links where needed.

Questions API

Quick Start Guide:
https://help.learnosity.com/hc/en-us/articles/360000755498-Getting-Started-With-the-Items-API
Demo: https://demos.learnosity.com/usecases/customquestions/custom_shorttext.php

The second group of documentation links is for custom questions. In addition to the commented template in this archive, we've also provided an unrelated, fully-functional sample for you to use as a guide. As stated above, use these for extra help but don't feel compelled to study them exhaustively.

Custom Questions

Docs: https://help.learnosity.com/hc/en-us/articles/360000758817-Creating-Custom-Questions
Demos: https://demos.learnosity.com/usecases/customquestions/custom_box_whisker.php

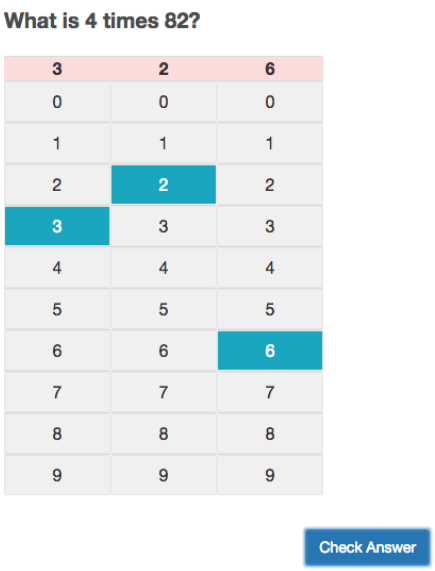
How the Question Should Work

We've provided the raw materials for a custom number pad question. You can pursue this question, or you can come up with a different question type that you might find more interesting, if you prefer.

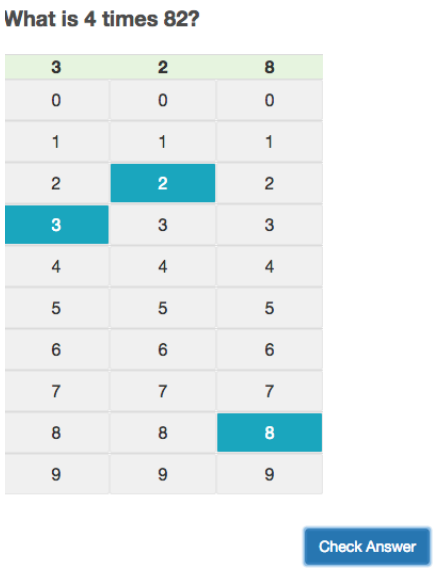
In the number pad example, there should be 3 columns, with clickable buttons 0-9 in each. The top row in each column should populate with the number clicked by the user.

Clicking the Check Answer button should fire a validation event, and you should change the UI as a result. Something like this, which shows two attempts to illustrate incorrect and correct responses:

Incorrect:



Correct:



Coding the Question Type

There are 3 main areas to code:

1. Function `buildHtmlAndInteractions`
   This is the place to build the HTML layout, and the interactive event handling that you want. For example, if you wanted purely a text box, the returned object would either be a string of HTML (containing “<input type="text" />”) or a jQuery element ($htmlObj) with similar code.
2. Function `addValidationUI`
   This is where you can alter, or add to, the UI upon validation, based on whether the supplied response is correct or not.
3. Function `CustomNumberPadScorer.prototype.isValid`
   This should return true or false, depending on whether the student answer matches the `valid_response` set in the question JSON.

Obviously, you can also do anything else you like here, extend out the functionality, do it in a different manner. We’d really like to see what you can think of in this area.

jQuery (http://api.jquery.com/) and Bootstrap’s CSS (http://getbootstrap.com/) are also available to you without further steps on your part, and you can modify the main PHP file as you want for layout, look and feel etc. You could also extend the “require” call in `custom_numberpad_stubbed.js` if you want, to include things like Angular JS, etc. Note that we’ve added the minimum here for maximum flexibility.

We recommend you start by getting the QuickStart Guide working in your preferred server-side language, starting with one of our SDKs:
https://help.learnosity.com/hc/en-us/sections/360000194318-Server-side-development-SDKs
PHP is provided here, but you can use any language found in the link above. Currently, the bulk of our customers use PHP and JS. Depth in more of these languages would be a big bonus, so speak up if you have any such experience.

KEY POINTS TO REMEMBER:

1. Please comment your code as though you’re presenting it to someone who’s never seen this before. You may prepare sample files for customers, or collaborate with other Application Support Engineers or our API engineering teams. It’s crucial that your code be clear and easy to understand.
2. If there are gaps here, or things you’re unsure of, it’s ok to ask questions. We don’t want you completely stuck because we missed something in this overview or because of a gap in our documentation. Contact rich.shupe@learnosity.com any time. Sincerely, this is more about how you’d fare in a support role, and less about a locked down test. Use all resources available as you would if you already worked for Learnosity, including other members of the support team, within the scope of this exercise.

Zip File Contents

examples/analog_clock (dir)
Start here. This directory contains an example that builds an analog clock. These files will help you get up to speed, but are for reference only.

> **analog_clock.php**
> This is the “host page” that loads the Learnosity Questions API to build the assessment. This is the simple, one-question assessment that will display the custom question type you create.
>
> **analog_clock.js**
> This JS file contains the question logic, events, and scoring hooks that allow you to tap into our back-end delivery and scoring mechanisms, as well as the logic of the question type itself. This file will be referenced by the above host page.
>
> **analog_clock.css**
> Similar to the JS file above, this CSS document is where you can add any styling for the custom question type.

custom_numberpad.php
This is your suggested “host page”. Modify as needed. Remember that you don’t have to use this example, you can develop your own, if preferred.

custom_numberpad_stubbed.js
This is the general infrastructure for the number pad example, without any custom question logic. You will build your own logic here.

custom_numberpad.css
This is your CSS, with default styles provided. Add your own.

helpers (dir)
This directory contains helper files and should not be altered. In particular, it contains the PHP Learnosity SDK for the provided example. The SDK is used to sign the API request, which you can learn more about in the aforementioned QuickStart Guides. Feel free to use a different language, if you prefer, as explained above.