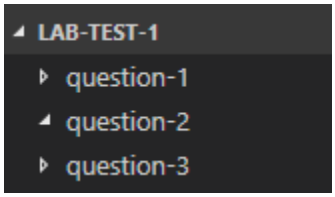
++++Full Stack– Lab 6 – Node Test

Developer Note:

When working on your questions, please create a separate folder for your work. This way you won't putting all your code in the same file, which can pollute the global namespace. In short, it will prevent you from overwriting your own work and causing your code to compile incorrectly.

Tip: Organize your folder structure in this way



Lab Test Submission

Please submit your test at the end of the lab, on Black Board Ultra

Question 1: Modules, Require and Export (10 Marks)

1. Open a command prompt and create a directory for **question-1.**
2. Open Visual Studio Code and open the folder **question-1**
3. Create a file named **calculator.js**.
4. In the **calculator.js** file create two methods named **multiplyTwoNumbers** and **evenDoubler** with the following requirements.
    - **mulitplyTwoNumbers**
        - will take two parameters x and y
        - will multiply the two parameters and return the result
        - will throw an error if either x or y is not an integer.
    - **evenDoubler**
        - will take one parameter x
        - will multiple x by x, if x is an even number
        - will return zero, if x is even
        - will throw an error x is not an integer
5. **Export** from your calculator module, the two methods **multipyTwoNumbers** and **evenDoubler.**
6. Create an **app.js** file and **import** the calculator module.
7. Use the calculator module to output the following when running **app.js** on the command line.

```
multiply 6 * 6 equals: 36
even doubler 4 equals: 0
```

Question 2: Event Handlers and Emitters

Task:

Build a simple ping pong simulation using **Node.js** with event emitters and asynchronous methods.

Stage 1: Event Emitter (5 Marks)

- Using event emitters do the following
    - create a function named **playGame** that will take the number of rounds param num
    - create an event listener for the '**ping**' event, this will trigger the '**pong**' event
    - create an event listener for the '**pong**' event, this will trigger the '**ping**' event
    - invoke the **playGame** function to trigger the first 'ping' event
    - keep track of the number of rounds and stop triggering to **exit** the program
- **Invoke** the **playGame** function in your script, the following way

```
playGame(3);
```

- The expected output is as follows

```
ping..
pong..
ping..
pong..
ping..
pong..
```

Stage 2: Asynchronous Behavior (5 Marks)

- Add some asynchronous behavior to add a 1-2 second delay before each event is triggered.
- Add an output for the **Round** number to the console.
    - log to console **game is over**..

- The expected output

```
Round 1
ping..
pong..
Round 2
ping..
pong..
Round 3
ping..
pong..
game completed..
```

Question 3: NPM packages and Asynchronous coding

Task:

Build a simple progress bar simulation using **Node.js** with NPM packages and asynchronous methods.

Demo Link:

https://www.youtube.com/watch?v=bUnoE-ySnTQ

Stage 1: Progress Bar (5 Marks)

- Download the NPM package progress and install.
  https://www.npmjs.com/package/progress

- Create a main file **app.js**
- Create a **progress-bar.js** module file and import in the **NPM progress module**
- In the app.js import in the **progress-bar.js** module

The **progress-bar.js** will have the following requirements:

1. Create a reference to the **progress bar object**
2. Configure the progress bar to have 10 steps

```
var downloadBar = new progressBar(':bar', { total: 10 });
```

3. Create a function timer that uses setTimeOut or setInterval
4. The timer function will call a callback function every 500 miliseconds
5. The callback function will progress through the 10 steps
6. The callback function will message to the console **'Download has completed'** when the progress bar is completed and clear the timer
7. Create a function named **startProgress** that will message to console that the **'Download has started'** and trigger or start the timer
8. Export the startProgress method to be used in **app.js**
9. In the **app.js** use the **progress-bar** module reference to call the **startProgress** method.

The expected output will be as follows:

```
Downloaded --> Started.
==========
Downloaded --> Completed.
```

Stage 2: Chalk (5 Marks)

- Download the NPM package **Chalk** and install.
  https://www.npmjs.com/package/chalk

- Using NPM at the command line, install the chalk module

Update **progress-bar.js** module file with the following requirements

1. Import the **chalk NPM module**
2. Configure the progress bar object to use the **percentage** and **estimated time attributes**
   (refer to progress documentation)
3. Configure **width** of the progress bar to be 20 and **total ticks** to be 100
   (refer to progress documentation)
4. Configurer the **complete** and **incomplete** characters attributes to use the **chalk**
   (tip you configure the chalk color and using an empty space to build the colored bar)
5. Update the **console message** to use the **chalk** coloring.

The expected output will be as follows:

```
Downloaded --> Started.
Downloading [                    ]  100% 0.0s
Downloaded --> Completed.
```

Submission

1. The exam question will uploaded and submitted to Black Board

Specification	Points
Question 1: Calculator	10 Marks
Question 2: Ping Pong	10 Marks
Question 3: Progress Bar	10 Marks
Clean Code and Clarity	5 Marks
Total	**35 Marks**