Tutorial 09 Challenges:

This assignment just brings it all together and let's you have fun with the **design** part! Try to think less about code syntax, and more about ORGANIZING your code to best fit your goals.

A template is provided, but you can replace the code in the www/ folder with your completed sound tutorial from last week if you would like.

Today, we'll turn the bouncing ball in to a "race against time" click game where the user

a) starts (and restarts) the game by clicking the button,
b) has to click a moving object as many times as possible within a fixed amount of time. When the time is up, the animation stops, and a dialog box pops up reporting the score.
c) can restart the game by clicking the button again.
d) can choose a level of difficulty that affects some aspect of the game.
e) Here is the goal movie: http://nm2207.org/creativeweb/tutorials/Challenge09.TimerGame_Goal.mp4

Try to write down the steps you will need to solve this challenge, starting from your existing code, before reading the detailed challenge steps provided below!!

What variables are you going to need?
What functions will you write?
What are the key things you need to do, and where (in which functions) will you do them?

You should have a little sketch that has the structure and components you need in kind of half English/ half code ("pseudo code"). This will really save you time later!

AFTER you have sketched out your strategy, variables, and functions, then see how they correspond to the steps below and code it up!

Steps to make the game:

Reminder: Check to make sure things are working properly *at every step* in your development with console messages, etc.

You'll want variables to store the game startTime, the number object clicks, and whether the game is running.

Of course, put a click listener on your object and count clicks.

Put functionality together in functions that belong together in concept:

Change your toggle button text, and its behavior so that it "restarts your game" when it is pushed. The callback function should:

a) use the Date() object to store the **new game start time**
b) show the current time in the footer section.
c) start calling your draw() function so that the object animates
d) reset your click counter to 0

e) (It should do these things only if the game isn't already playing!)

Create an endGame function that

a) stops the animation by stopping the periodic calls to your draw() function
b) opens a confirm box to report how many clicks were made during the game
c) resets the graphical object back to the center

Stop the game after the time limit is reached by calling endGame(). Where will you do that? That is, where in your code will you know when the game is over? Point to it with your finger, and then call endGame there!

Hint: Comment your code thoroughly so that you and others will know what each part of the code does and how you were thinking.

Once the above is working:

1) Serve this game from a node server

- A node server (exactly like the helloWorldFiles server from this week's vid lecture/code along) is provided in the root directory of the template code. You may have to "npm install express" like we did on the vid before your can run your server.
- First just run the server from the command line with node to make sure it works.
- Currently, the server is "hardcoded" to use 3000. We want to be able to specify the port it should use on the command line. For example, if we want it to use port 4000, we would type:
  - > node myNodeServer.js 4000
- **Change the server code to get a port number from the command line, using process.argv (google it!).**
- Point your browser to the game using your IP address and port number and play!

2) Show your game to your TA being **served** and played. Your node server must obviously be running for you to serve your game, and the address will clearly reflect whether you are serving your game or not.

Bonus round:

3) If you are on the same network as a friend (e.g. the NUS network or you home LAN), give your ipaddress and port number to your neighbor so they can play your game. (You won't be able to do this with your friend on their own home LAN. You should understand why.

- b) If your phone is on the same wireless network as your computer (and your server is running), you can also use your computer's IP address and port number you gave the server in your phone browser to play your game there! Cool, eh?

Double bonus round (to add coolness to your game):

- Allow the user to choose different levels of difficulty
  - For example, you could add radio buttons under your 'play' button to select difficulty – what might you change to make you game more/less difficult?
- Play a sound when the graphical object is clicked
- Play a sound when the game completes
- Add artwork (background, animated object)
- Make the game more complex

- Show the timer counting down to zero. Ooh, ooh – or how about a timer progress bar? (https://www.w3schools.com/howto/howto_js_progressbar.asp)