

# EMPOWERING EDUCATORS TO CREATE AI-GENERATED WEB APPS

A Sharing by Tan Seng Kwang, ST/Physics  
(Temasek Junior College)

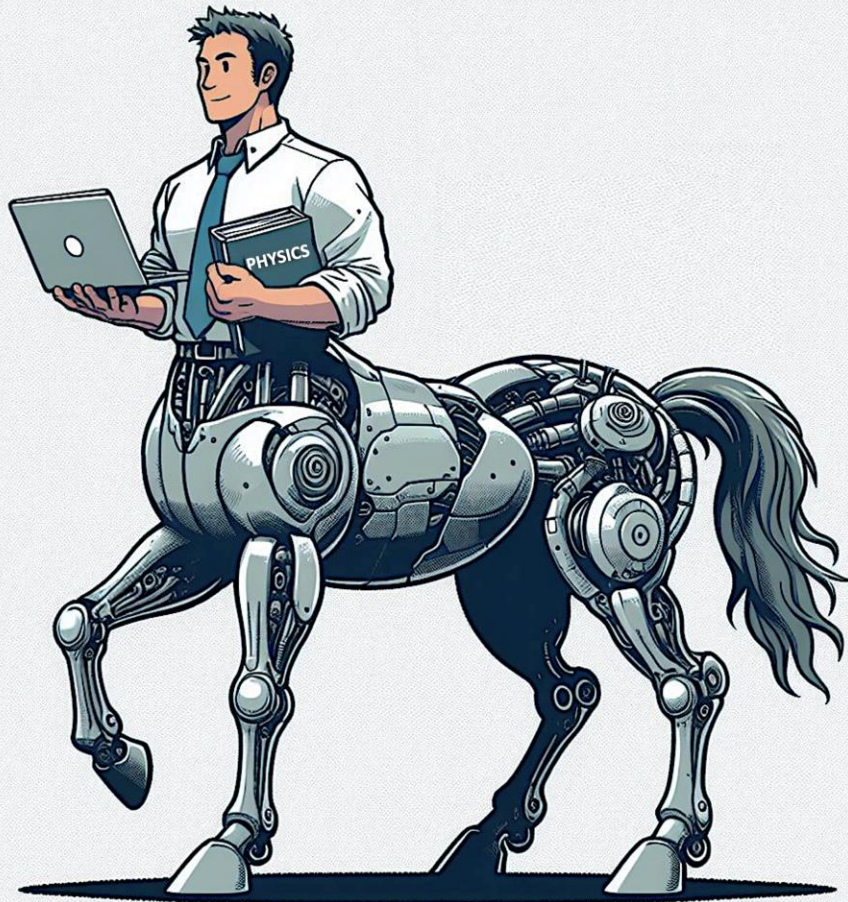


TEMASEK  
JUNIOR COLLEGE



TEMASEK  
JUNIOR COLLEGE

# CENTAUR > MAN OR MACHINE



While experts will always be in demand, more intelligent machines are continually **lowering the bar** to creating with new technology.

*Kasparov, 2017*

Lower barrier of entry for app development; allow for **focus on higher levels of problem solving**

*Bull and Kharrufa, 2023*

# AI-Generated Apps for Education

## Useful links

Repository:  
[for.edu.sg/apps](https://for.edu.sg/apps)

Editor:  
[for.edu.sg/html](https://for.edu.sg/html)

Home General Tools Games Assessment Visualisation Sensors Blog

## AI-Generated Apps for Student Learning Space

This is a series of apps that were produced using codes given by [ChatGPT 3.5 \(the free version\)](#). You can do so with other GAI like [Microsoft Copilot](#) too. The apps are usable directly from here or can be downloaded and included into an SLS package.

The prompts given to ChatGPT are provided for your reference. However, you may still have to refine the prompts as the output may not always be consistent. While it is possible to generate a working app without any changes to the code, you can customise the app much faster if you know a bit of html or javascript. You may even ask ChatGPT to teach you how to do so!

Check out the [step-by-step guide](#) to produce and deploy your app.

How to: [Generate your own apps](#) | [Upload to SLS as Media Objects](#)

### General Tools Back to Top

#### Code to HTML Converter

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Code to HTML Converter</title>
<style>
body {
font-family: Arial, sans-serif;
}
</style>
</head>
<body>
</body>
</html>
```

Download as HTML Run in New Tab

#### HTML File Generator

This serves as a web-based code editor

#### Pomodoro Timer

4:32

Start Pause Stop

Students can use this timer as a focusing strategy. For the first 25

#### Countdown Timer

00:00:00

Start Pause Stop

For use in class. Set a duration and start the timer. When the time is

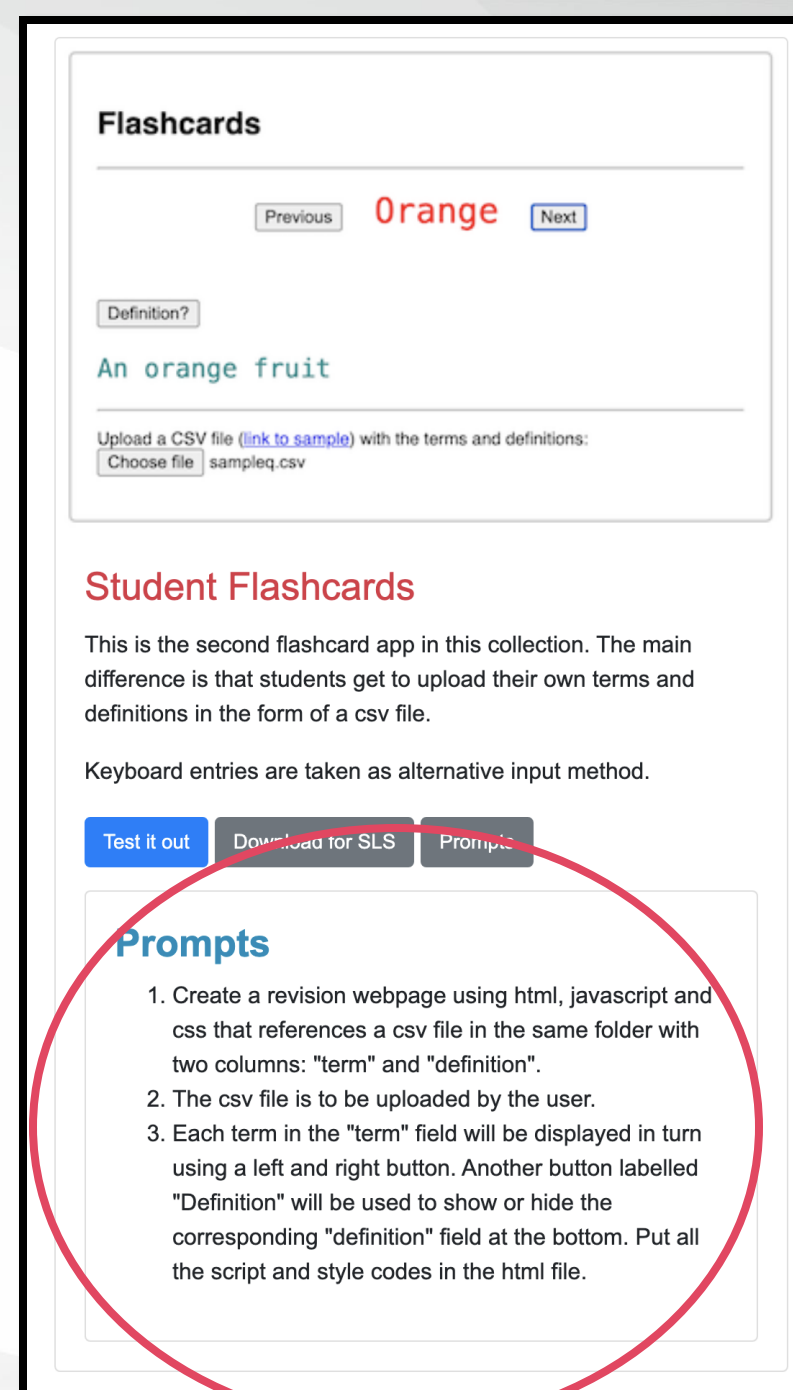
## About the repository...

- Can use directly on browser (online or offline) or embedded into SLS
- Prompts used are shown for each app.

## Example 1: Productivity Tools

- Use: Student or teacher can upload a csv file of terms and definitions to create digital flashcards

<https://physicstjc.github.io/sls/student-flashcards>



**Flashcards**

Previous **Orange** Next

Definition?

An orange fruit

Upload a CSV file ([link to sample](#)) with the terms and definitions:  
Choose file sampleq.csv

**Student Flashcards**

This is the second flashcard app in this collection. The main difference is that students get to upload their own terms and definitions in the form of a csv file.

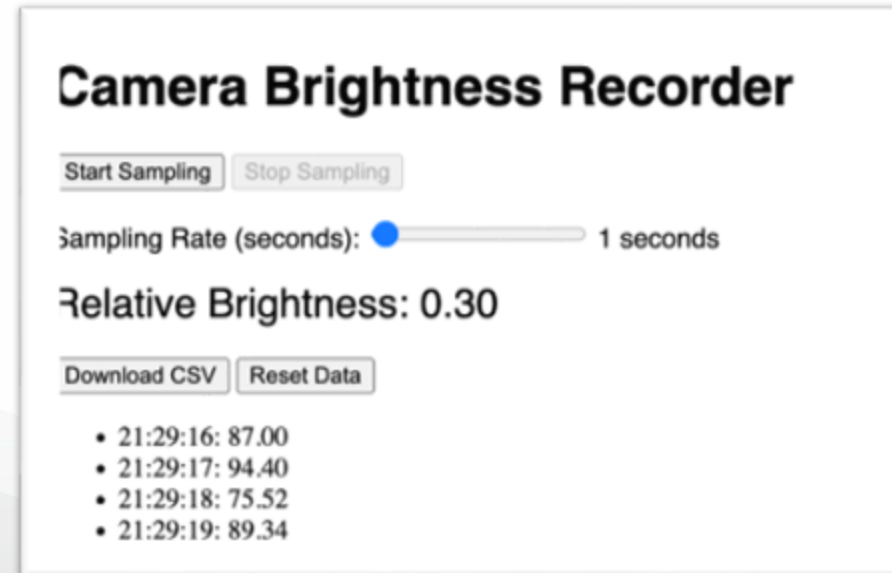
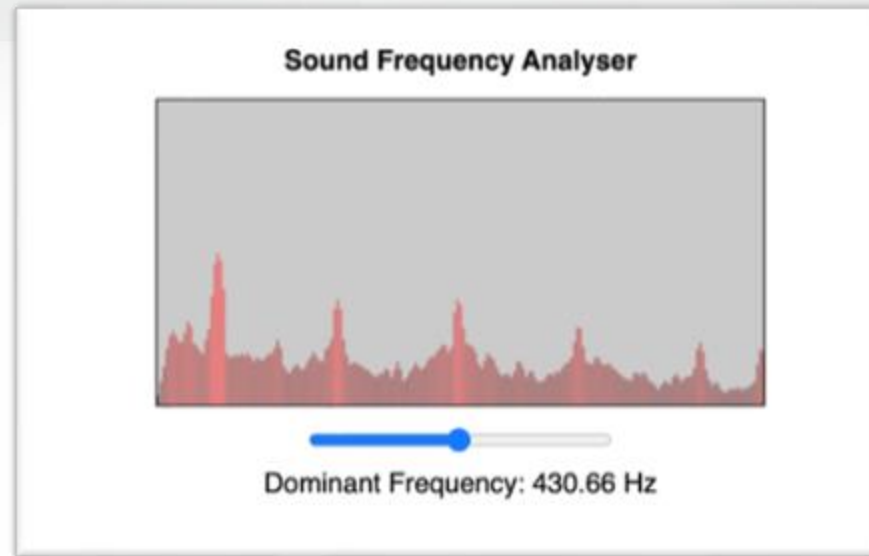
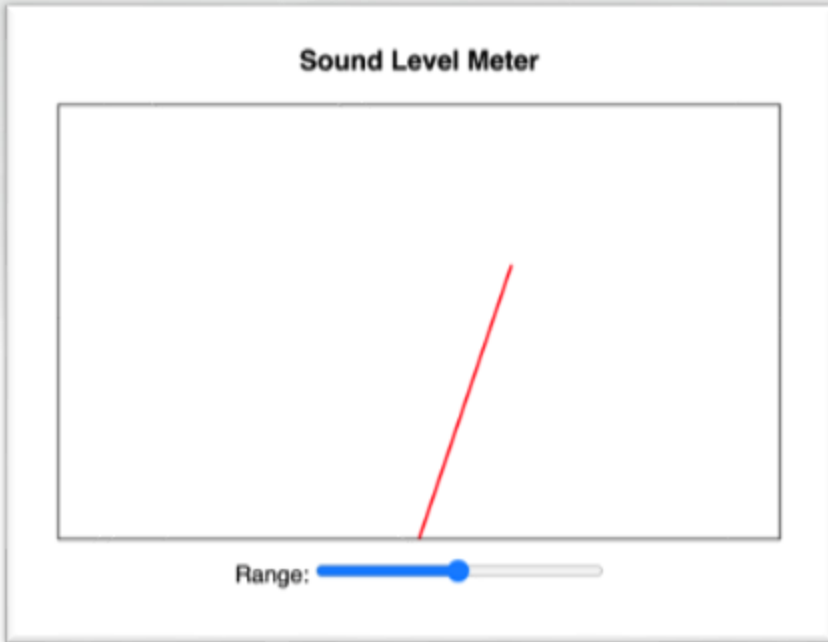
Keyboard entries are taken as alternative input method.

Test it out Download for SLS Prompts

**Prompts**

1. Create a revision webpage using html, javascript and css that references a csv file in the same folder with two columns: "term" and "definition".
2. The csv file is to be uploaded by the user.
3. Each term in the "term" field will be displayed in turn using a left and right button. Another button labelled "Definition" will be used to show or hide the corresponding "definition" field at the bottom. Put all the script and style codes in the html file.

# Examples 2-5: Built-in Sensors for Science Experiments



An example in the  
SLS Community  
Gallery

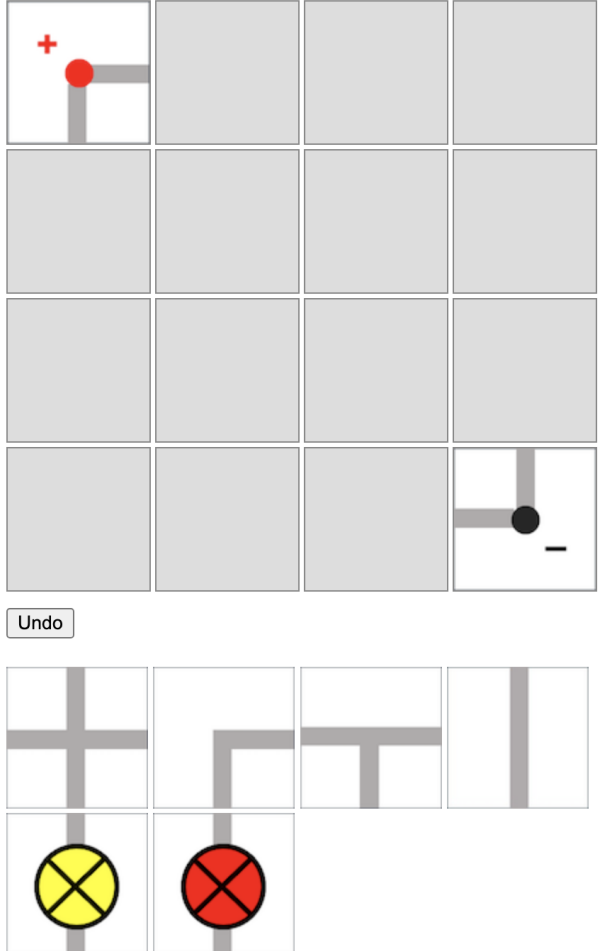
<https://vle.learning.moe.edu.sg/community-gallery/lesson/view/f9c45429-0261-4999-aaeb-f8ded90fe8eb/page/73124376>

# Examples 6: Build Games!

- Turn-based game to place wires to light up your bulb
- Sabotage your opponent by bypassing or blocking current

<https://physicstjc.github.io/sls/delight/>

**DELIGHT - electrifying fun**



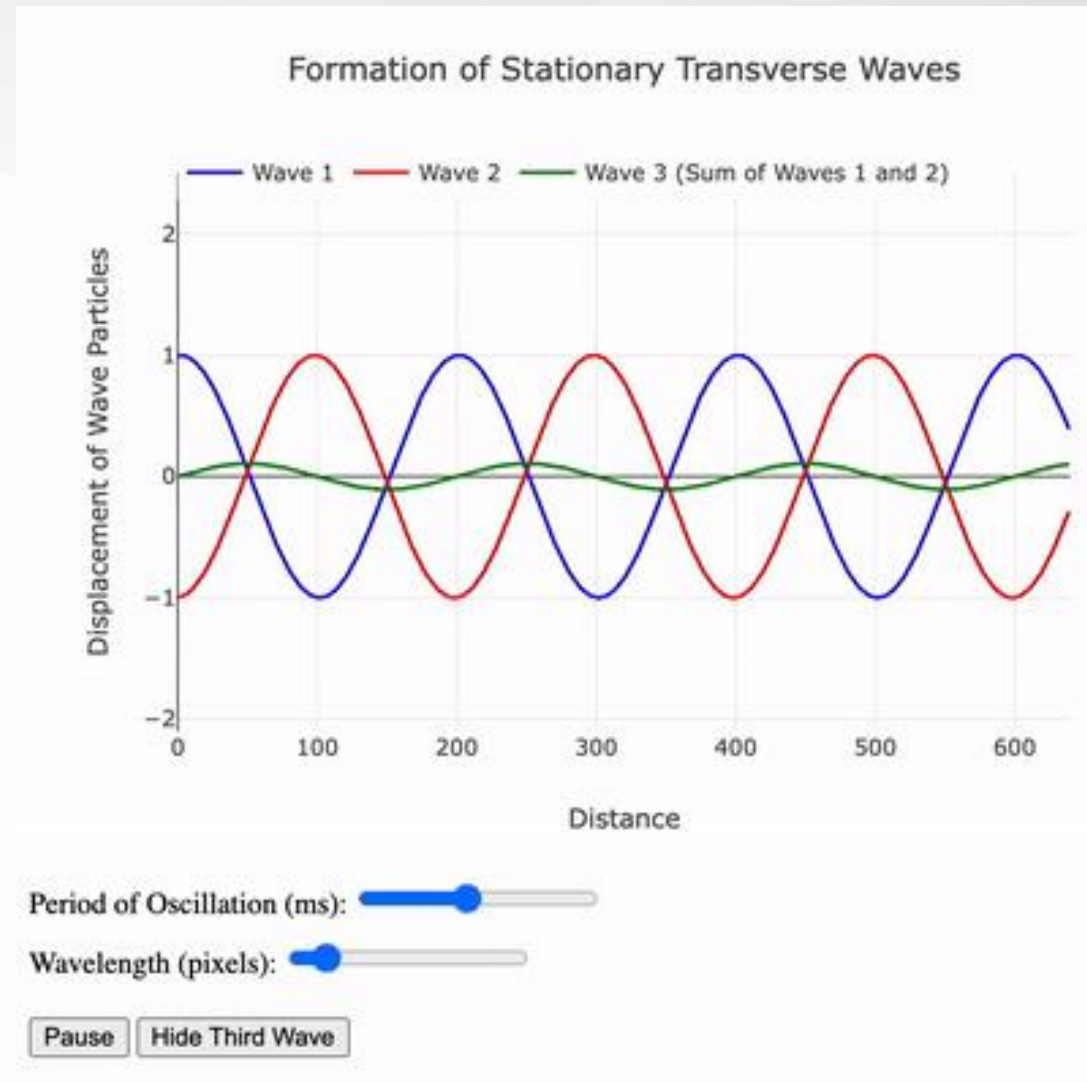
**Rules**

01. The aim is to light up as many bulbs of your colour as possible.
02. Players will take turns to place one piece on the 4-by-4 game board by clicking to select the electrical component and clicking on the square on the board to place it.
03. Upon placing the piece, the player can also turn that piece in any orientation (by clicking on it) within the same turn.
04. Players can choose to use up to two turns at any point in the game to rotate any piece that had been placed by any player.
05. In other words, each player has 9 turns: 7 placement turns and 2 rotation turns.

# Design Consideration

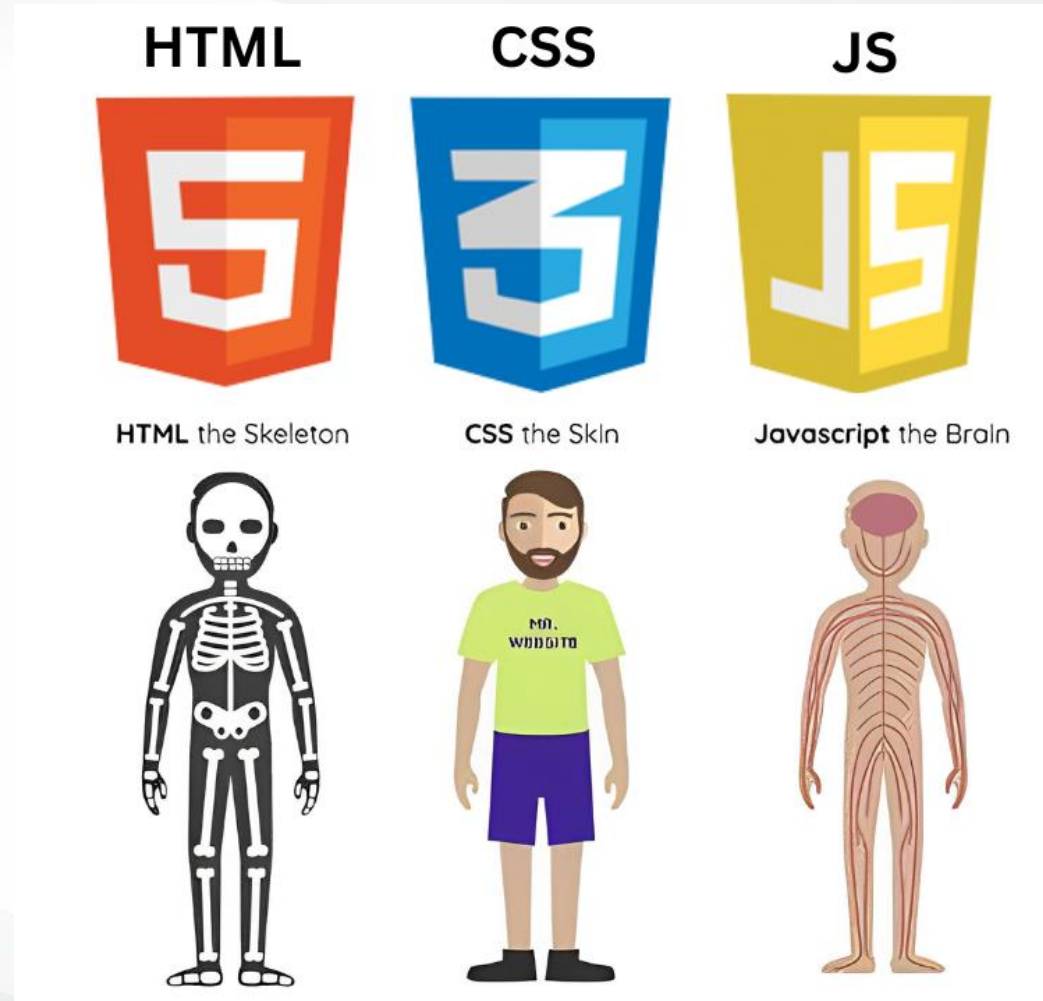
- Is a suitable app already out there?
- Allow for student inquiry
- Minimise extraneous cognitive load
- Interactive elements
  - Buttons or checkboxes – clicking to start/stop animation or hide objects
  - Sliders – quick change of variable
  - Input boxes – to key in variables
- Display elements
  - Canvas – where the animation takes place
  - Graph – multimodal representation

## Example 7: Interactive Animation



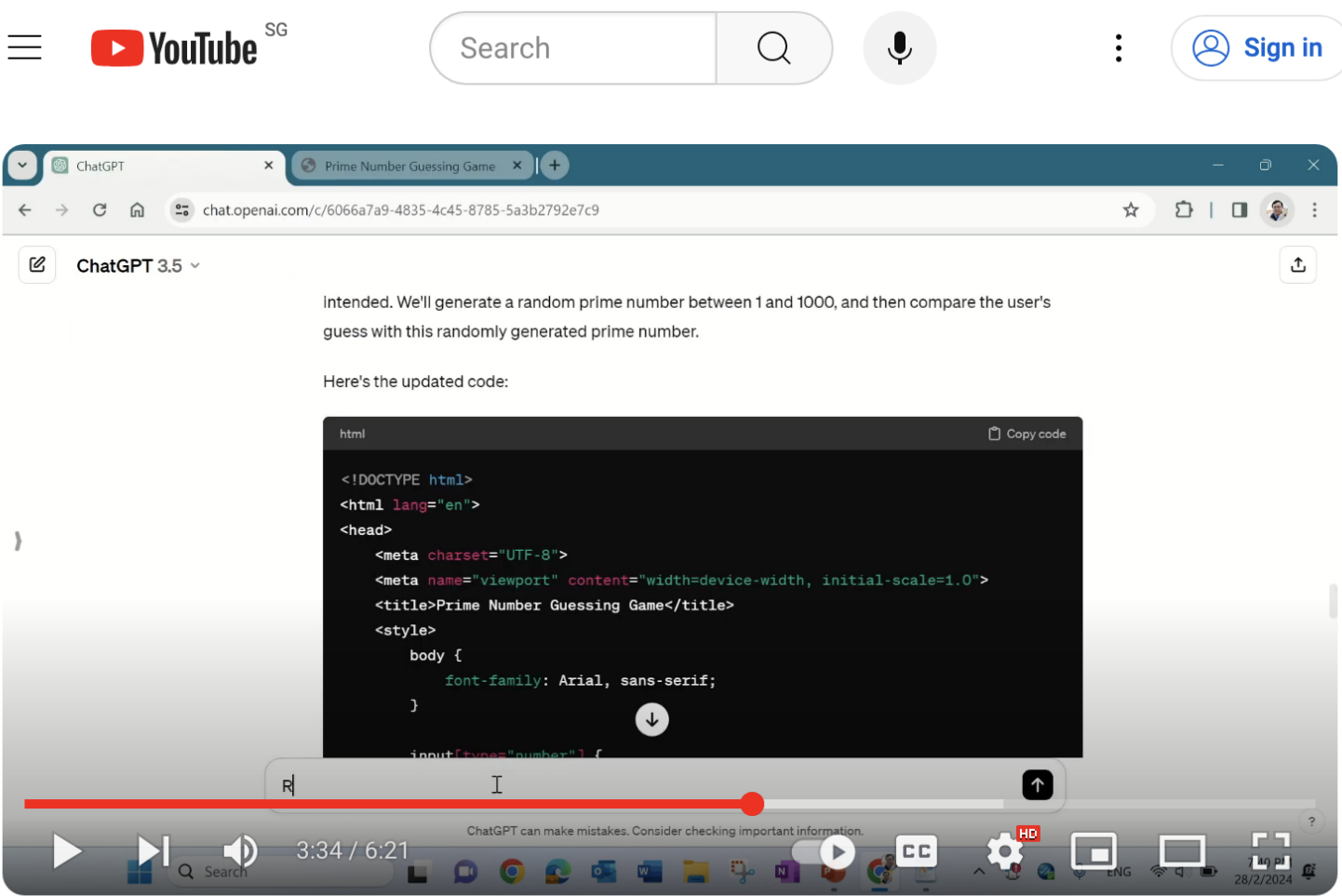
# Basics of Interactive Web Apps

- For large sites, these codes are usually placed in separate files for universal use.
- To simplify workflow, we will put all 3 sets of codes in one file.
- Possible to code in python for <https://www.glowscript.org>



Source:  
<https://www.linkedin.com/pulse/fundamental-elements-web-development-html-css-an-assadikine-byn7e/>

# Video Tutorial: <https://youtu.be/1YCtr2AyAko>



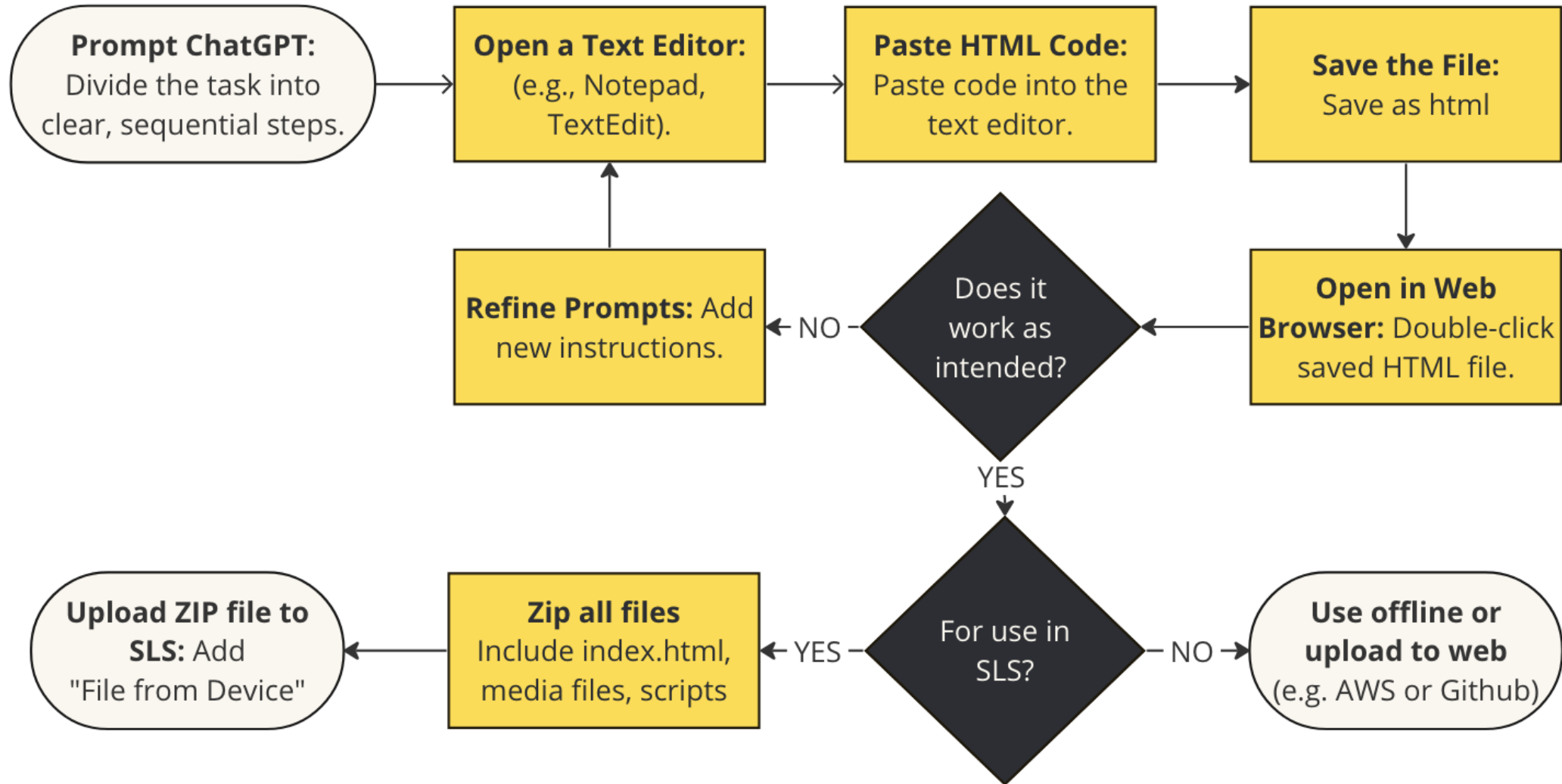
The screenshot displays a YouTube video player. At the top, the YouTube logo and search bar are visible. The video content shows a browser window with two tabs: 'ChatGPT' and 'Prime Number Guessing Game'. The ChatGPT interface shows a conversation where the user asks for a prime number guessing game, and ChatGPT provides the code. The code is displayed in a dark-themed editor with the following content:

```
html
Copy code
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Prime Number Guessing Game</title>
  <style>
    body {
      font-family: Arial, sans-serif;
    }
  </style>
  <input type="number" />
  </html>
```

The video player shows the video is at 3:34 / 6:21. Below the video, the title 'Generating standalone web apps for SLS using ChatGPT' is displayed. The channel name is 'Physics Lens' with 554 subscribers. The video has 2 likes and a share button is visible.

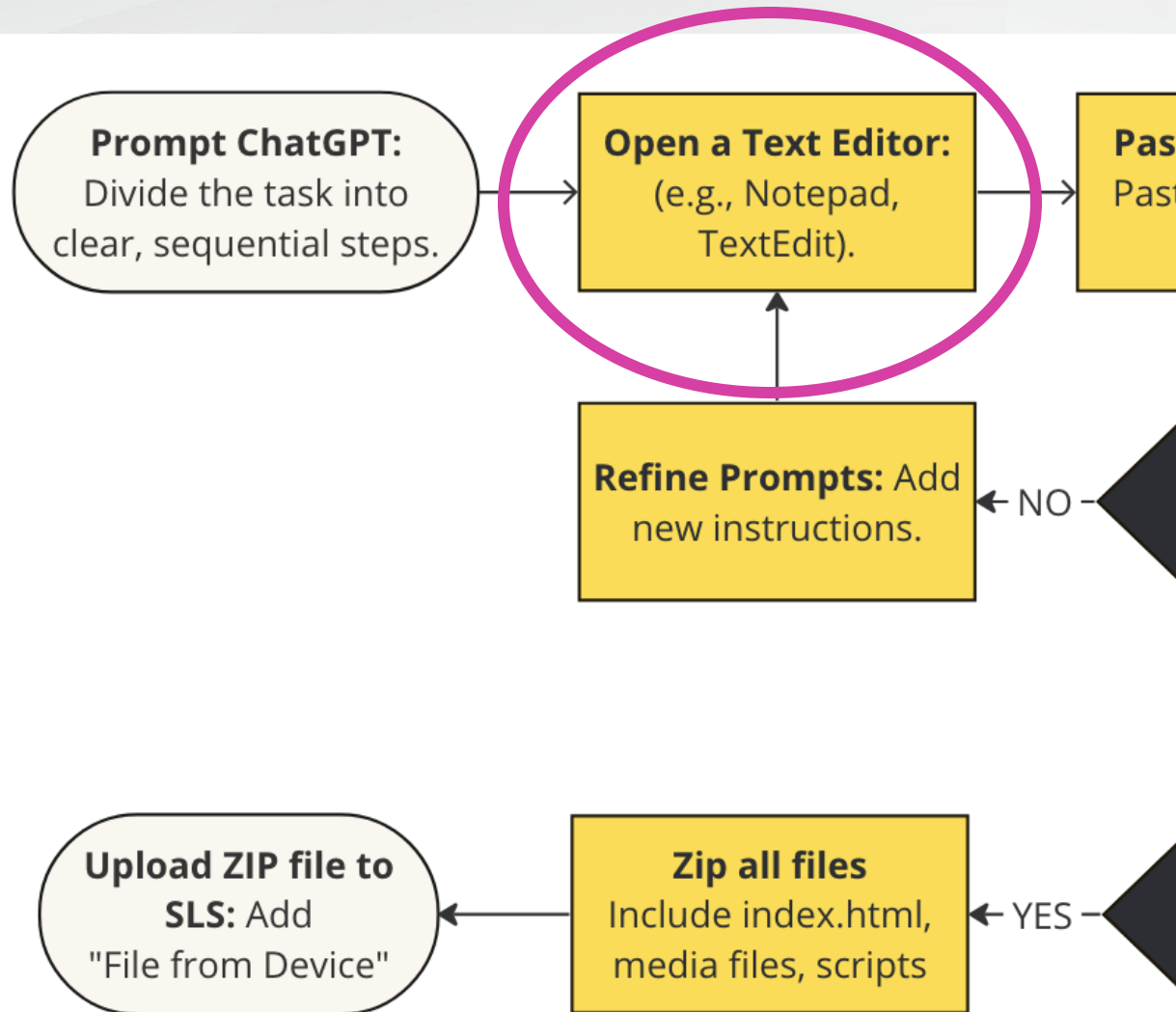


# Workflow



# Workflow

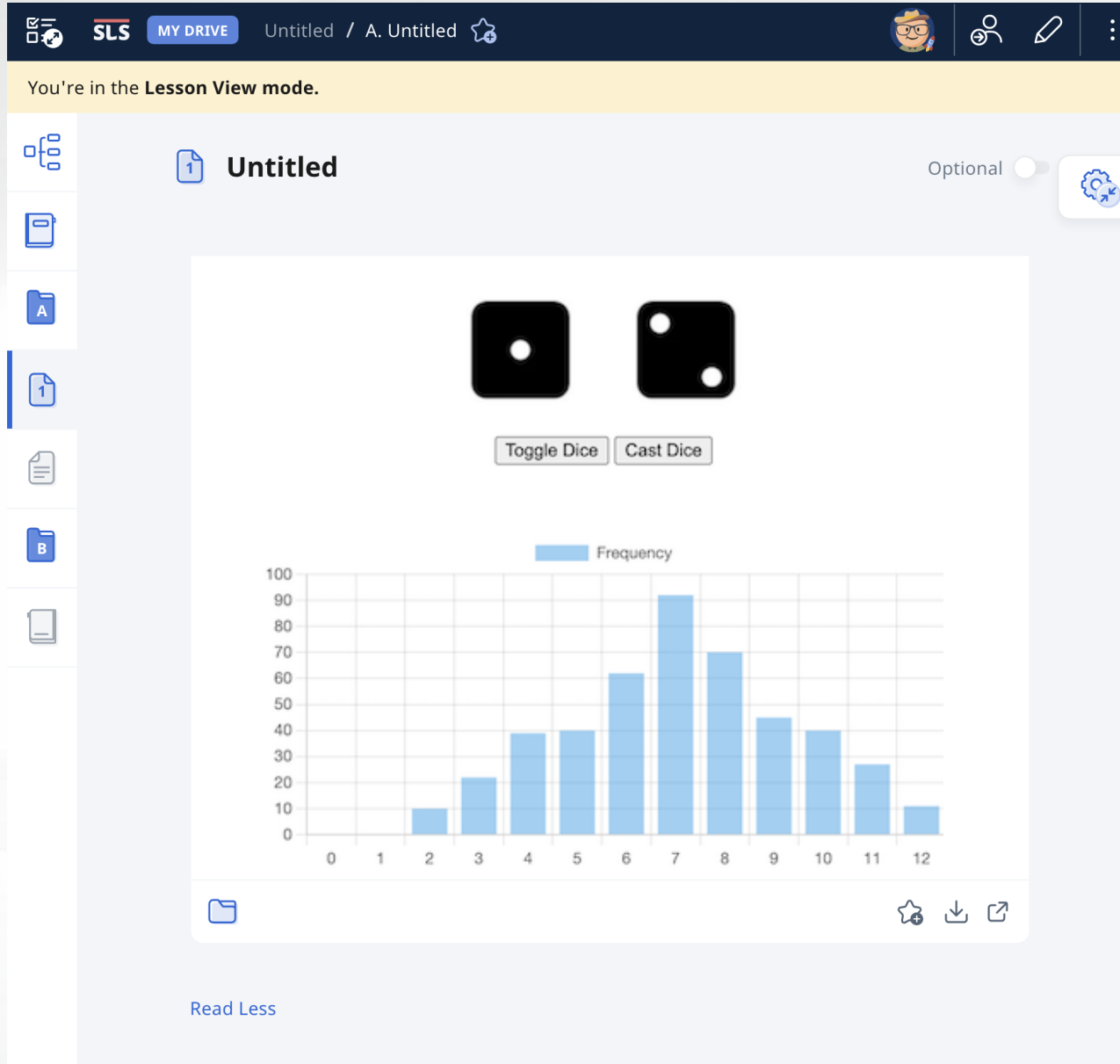
Web-based Code Editor  
produced with ChatGPT!



[for.edu.sg/html](https://for.edu.sg/html)



# Example 8: Dice Simulator with Bar Graph



The screenshot shows a web application interface for a dice simulator. At the top, there is a navigation bar with 'SLS MY DRIVE' and 'Untitled / A. Untitled'. Below this, a yellow banner indicates 'You're in the Lesson View mode.' The main content area features a dice simulator with two dice, a 'Toggle Dice' button, and a 'Cast Dice' button. Below the dice is a bar graph titled 'Frequency' showing the distribution of outcomes from 2 to 12. The y-axis represents frequency, ranging from 0 to 100. The x-axis represents the sum of the dice, ranging from 0 to 12. The bars show a distribution that is roughly bell-shaped, peaking at 7.

| Sum of Dice | Frequency |
|-------------|-----------|
| 2           | 10        |
| 3           | 20        |
| 4           | 30        |
| 5           | 40        |
| 6           | 60        |
| 7           | 90        |
| 8           | 70        |
| 9           | 45        |
| 10          | 40        |
| 11          | 25        |
| 12          | 10        |

- Learning statistics representation
- Inquiry activity to verify expected probability distribution
- Accounting for differences due to randomness

<https://physicstjc.github.io/sls/dice>

# Example 8: Dice Simulator with Bar Graph

The screenshot shows a Google Slides presentation titled 'Untitled'. The slide content features a dice simulator interface. At the top, there are two black dice images. Below them are two buttons: 'Toggle Dice' and 'Cast Dice'. At the bottom of the slide is a bar chart titled 'Frequency'. The x-axis represents the sum of two dice, ranging from 0 to 12. The y-axis represents the frequency, ranging from 0 to 100. The bars show the following frequencies: 2: 10, 3: 20, 4: 30, 5: 40, 6: 50, 7: 60, 8: 70, 9: 80, 10: 90, 11: 80, 12: 70.

Read Less

Provide the code for the following in a single html file:

1. Create a 6-sided dice simulator where the user can click on the image of a dice, the dice image will change randomly and the final result will be shown on the image.
2. The dice image will be represented by the file "dice1.png" for the number 1, "dice2.png" for number 2, "dice3.png" for number 3 and so on. The image size is 80 pixels by 80 pixels.
3. The default setting shows one dice. The initial image shown is "dice1.png".
4. Create a button to cast the dice.
5. Create a bar chart with 6 vertical columns at the bottom that shows the frequency of the numbers obtained by the dice. Adjust the bar chart to show all 6 possible outcomes.
6. Each time the dice is cast and the result is shown, it is added to a list shown below the chart.
7. Create a button for an option to toggle between the use of one dice or two die. When this button is clicked, the list of results and the bar chart is cleared. The initial images shown are both "dice1.png". If two die are used, both die will be shown side by side.
8. Both die will be cast when clicking on either dice. The bar chart now shows the frequency of the total of the numbers obtained by the die. Adjust the bar chart such that it now has 12 vertical columns show all 10 possible outcomes.
9. The total number for each toss will be shown in the list.

# Example 9: Linear Graph Plotter

The screenshot displays a web-based Linear Graph Plotter interface. At the top, there is a navigation bar with 'SLS MY DRIVE' and 'Untitled / A. Untitled'. Below this, a yellow banner indicates 'You're in the Lesson View mode.' The main area shows a graph titled 'Graph Plotter' with a 'Choose file' button set to 'data.csv'. The graph plots 'I/A' on the y-axis (ranging from 0.2 to 1.0) against 'V/V' on the x-axis (ranging from 2.5 to 7.0). Five data points are plotted, and a red best-fit line is drawn through them. The legend indicates 'Data Points' (green circles) and 'Best Fit Line:  $y = 0.17x + -0.20$ '. A 'Take Screenshot' button is located below the graph. At the bottom left, there is a 'Read Less' link.

| V/V | I/A |
|-----|-----|
| 2.5 | 0.2 |
| 3.0 | 0.3 |
| 4.0 | 0.5 |
| 4.8 | 0.6 |
| 5.9 | 0.8 |
| 6.7 | 0.9 |

- For visualizing and analyzing experimental data
  - Converts data from CSV file to graph
  - Displays best-fit line and equation
  - Exports screenshot in image file for upload into SLS

# Ready to give it a shot?



# Prompt Engineering

- In your prompt:
  - Start by asking for the html and javascript codes in a single file.
  - Describe what the app is supposed to show. Do you need a canvas (for drawings) or a graph, or both?
  - Describe the interactions in the app. Use elements such as buttons, sliders or input boxes.
  - (If necessary) Give the names of image/video files to be used and put them in the same directory as the html file.
  - (If optimizing for SLS), max width: 580px, height: 460px

# Prompt Engineering – Minimum Viable Product

- Prompt Sample:

- Using javascript and the plotly library, plot the graph of velocity versus time for a ball dropping from rest and bouncing off the ground three times. The time of contact with the ground is negligible.
- Create an inputbox that allows the user to key in the initial height in metres.
- Create a slider that changes the percentage energy loss after every collision with the ground
- Create a dropdown menu that changes the vertical axis to velocity or displacement.
- Limit the page to a size of max width 580px and max height: 460px

- Copy and Paste into [for.education.sg/html](https://for.education.sg/html) and test it!

# Prompt Engineering - Debugging

If you are not sure, just describe the unexpected behaviour.



**You**

The speed slider is now affecting the number of particles, which should not be the case



**ChatGPT**

I apologize for the oversight. I've corrected the code so that the speed slider only affects the speed of particles, not their number. Here's the updated code:

```
html Copy code  
  
<!DOCTYPE html>  
<html lang="en">
```

If it still does not work after some iterations, ask it to try a new approach.



**You**

Does not seem to work. Is there another way



**ChatGPT**

Certainly! We can use a combination of FileReader, JSZip, and Blob APIs to achieve the desired functionality. This approach will handle the zipped folder processing entirely on the client-side without relying on experimental APIs.

Here's how you can implement it:

```
html Copy code  
  
<!DOCTYPE html>  
<html lang="en">  
<head>
```

# Prompt Engineering - Make Improvements

- Prompt Sample:
  - Add a canvas at the side of the graph to show the bouncing of the ball with time.
  - Use a button to start and stop the simulation.
  - Update the graph as the ball moves.
- Copy and Paste into [for.edu.sg/html](https://for.edu.sg/html) and test it!

# Prompt Engineering - Adding Images / Videos / Audio

- Prompt Sample:

- Replace the ball with an image file called ball.png



- Copy and Paste into `for.edu.sg/html`
- Download the image file
- Place the image file in the same directory

# Prompt Engineering - Using External Code Libraries

- Code libraries: a collection of pre-written code that you can use to perform specific tasks, e.g.
  - plotly.js for continuously changing graphs
  - chart.js for static charts
  - papaparse.js for processing csv data
- To embed into SLS, make a copy of the file: Paste the link on browser, copy the entire page and paste in an editor. Save the file in same directory as index.html

Change

```
<script src="https://cdn.plot.ly/plotly-latest.min.js"></script>  
<style>
```

to

```
<script src="plotly-latest.min.js"></script>  
<style>
```

# Limitations of standalone app in SLS

1. No connection to online databases. However, data can be stored in browser.
2. No transfer of information between SLS and app.

# Students can write code for ALP/Experiments

- (MicroPython) Use of micro:bit sensors for experiment, e.g. accelerometer for egg drop
- (C++) Arduino  
[https://www.youtube.com/watch?v=rHiiYMVU\\_ok](https://www.youtube.com/watch?v=rHiiYMVU_ok)
- (Python) Streamlit apps for data visualization, e.g. apps using external APIs, e.g. wind direction, [carpark availability](#)
- Guiding users to use block programming e.g. Scratch, Tello app

# Contact Us



[tan\\_seng\\_kwang@schools.gov.sg](mailto:tan_seng_kwang@schools.gov.sg)



[physicslens.com](http://physicslens.com)