Instructor's Manual for Mindmapping go.js and p5.js videos

Brought to you by: The Non-Linear Learning Team

Introduction:

Our mindmapping program incorporates the go.js library for the mindmap software, as well as p5.js library for the interactive videos. The purpose of this document is to instruct the user on how to create their own mindmaps and interactive videos using the program package we provided.

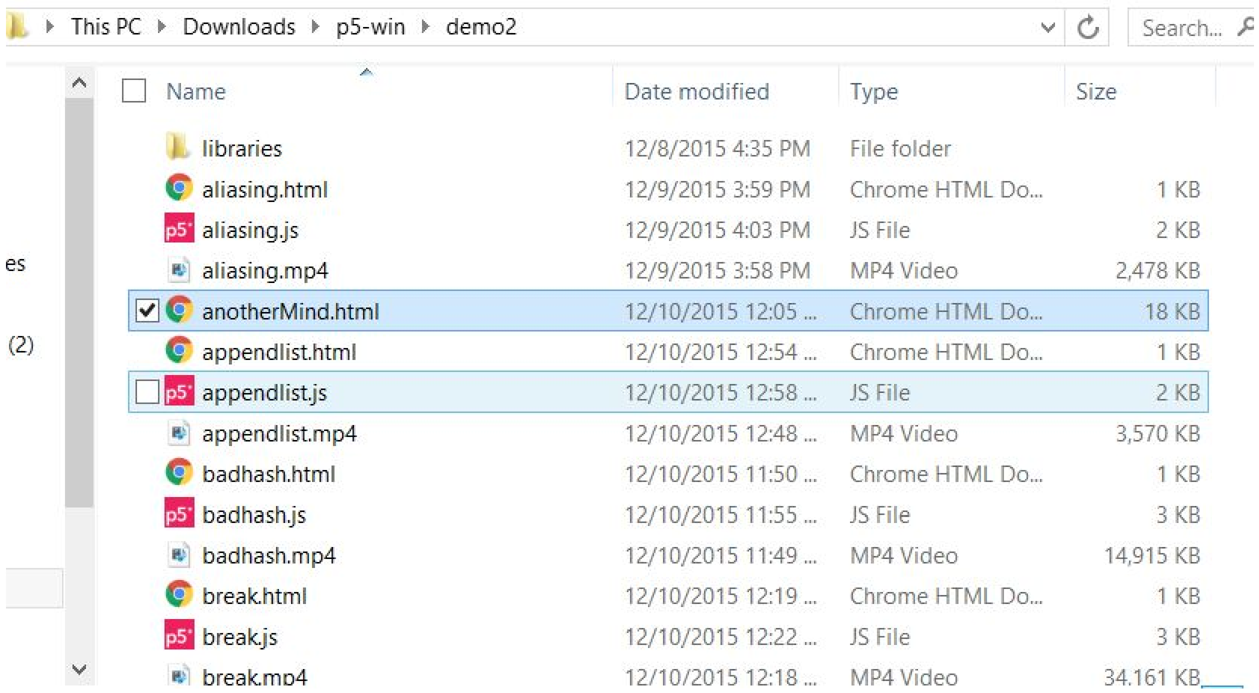
Libraries:

go.js: **http://gojs.net/latest/index.html**

p5.js: **http://p5js.org/**

Creating your first mindmap:

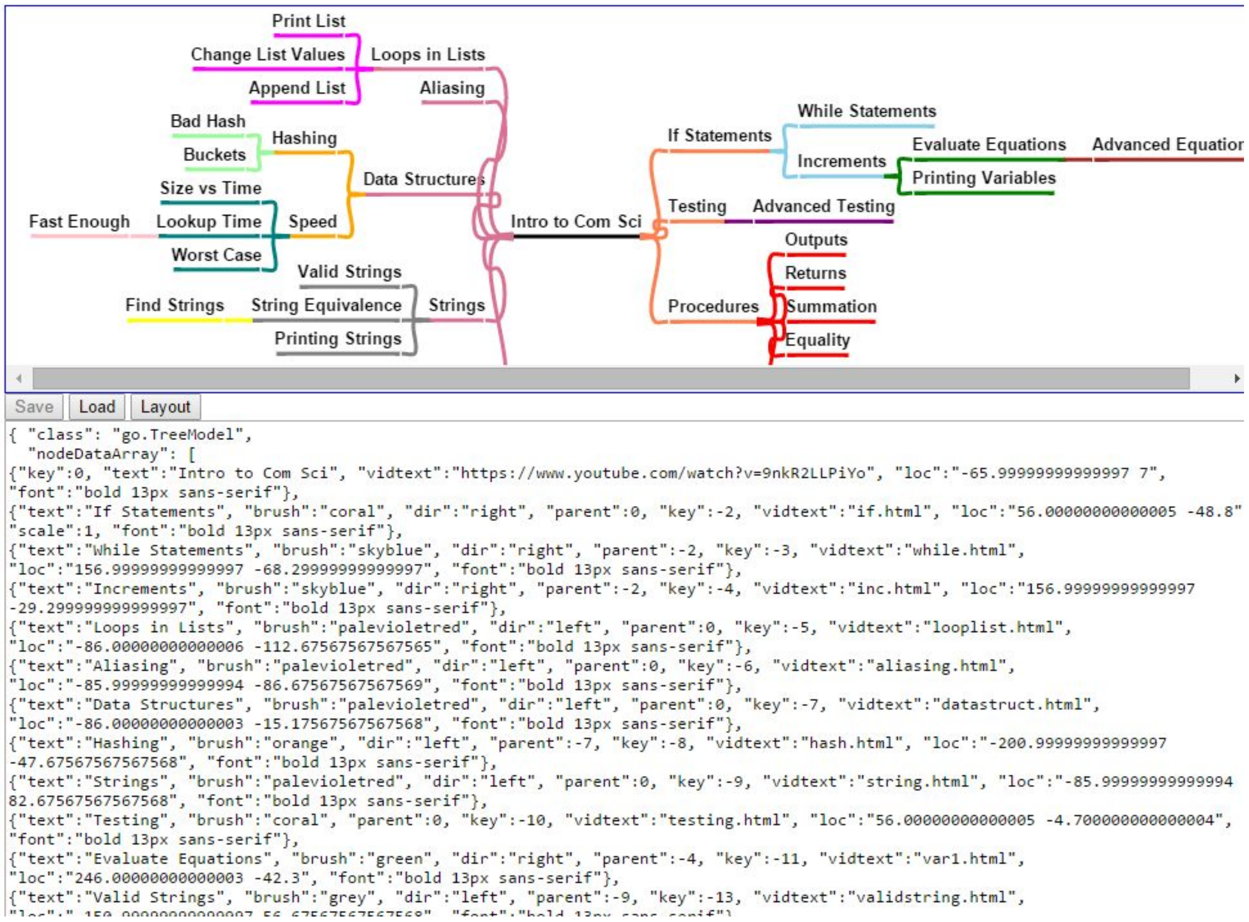
1. First, using a text-editor such as notepad++, open up ***anotherMind.html*** under the demo folder.



This PC ▸ Downloads ▸ p5-win ▸ demo2

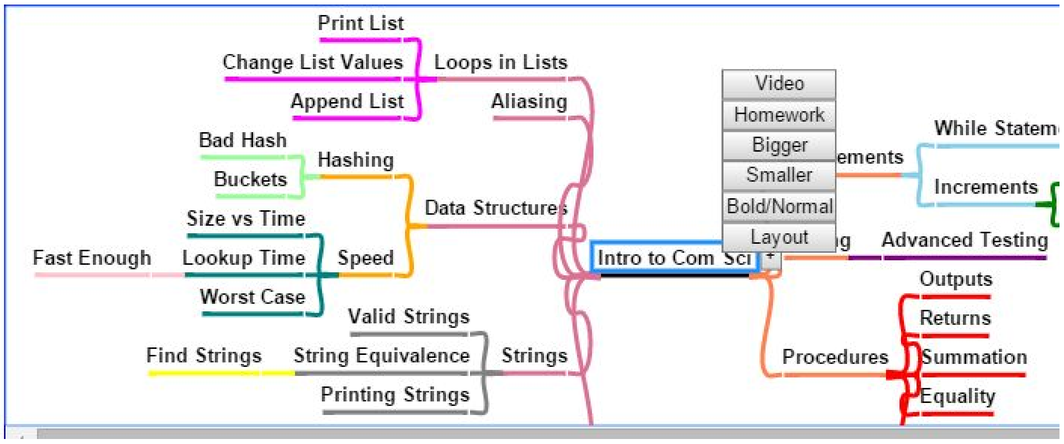
Name	Date modified	Type	Size
libraries	12/8/2015 4:35 PM	File folder	
aliasing.html	12/9/2015 3:59 PM	Chrome HTML Do...	1 KB
aliasing.js	12/9/2015 4:03 PM	JS File	2 KB
aliasing.mp4	12/9/2015 3:58 PM	MP4 Video	2,478 KB
✓ anotherMind.html	12/10/2015 12:05 ...	Chrome HTML Do...	18 KB
appendlist.html	12/10/2015 12:54 ...	Chrome HTML Do...	1 KB
appendlist.js	12/10/2015 12:58 ...	JS File	2 KB
appendlist.mp4	12/10/2015 12:48 ...	MP4 Video	3,570 KB
badhash.html	12/10/2015 11:50 ...	Chrome HTML Do...	1 KB
badhash.js	12/10/2015 11:55 ...	JS File	3 KB
badhash.mp4	12/10/2015 11:49 ...	MP4 Video	14,915 KB
break.html	12/10/2015 12:19 ...	Chrome HTML Do...	1 KB
break.js	12/10/2015 12:22 ...	JS File	3 KB
break.mp4	12/10/2015 12:18 ...	MP4 Video	34,161 KB

2. When you open up the html file, the mindmap should appear in the browser of your choice. For example, in our “Intro to Computer Science demo”, the mindmap page should look like this:

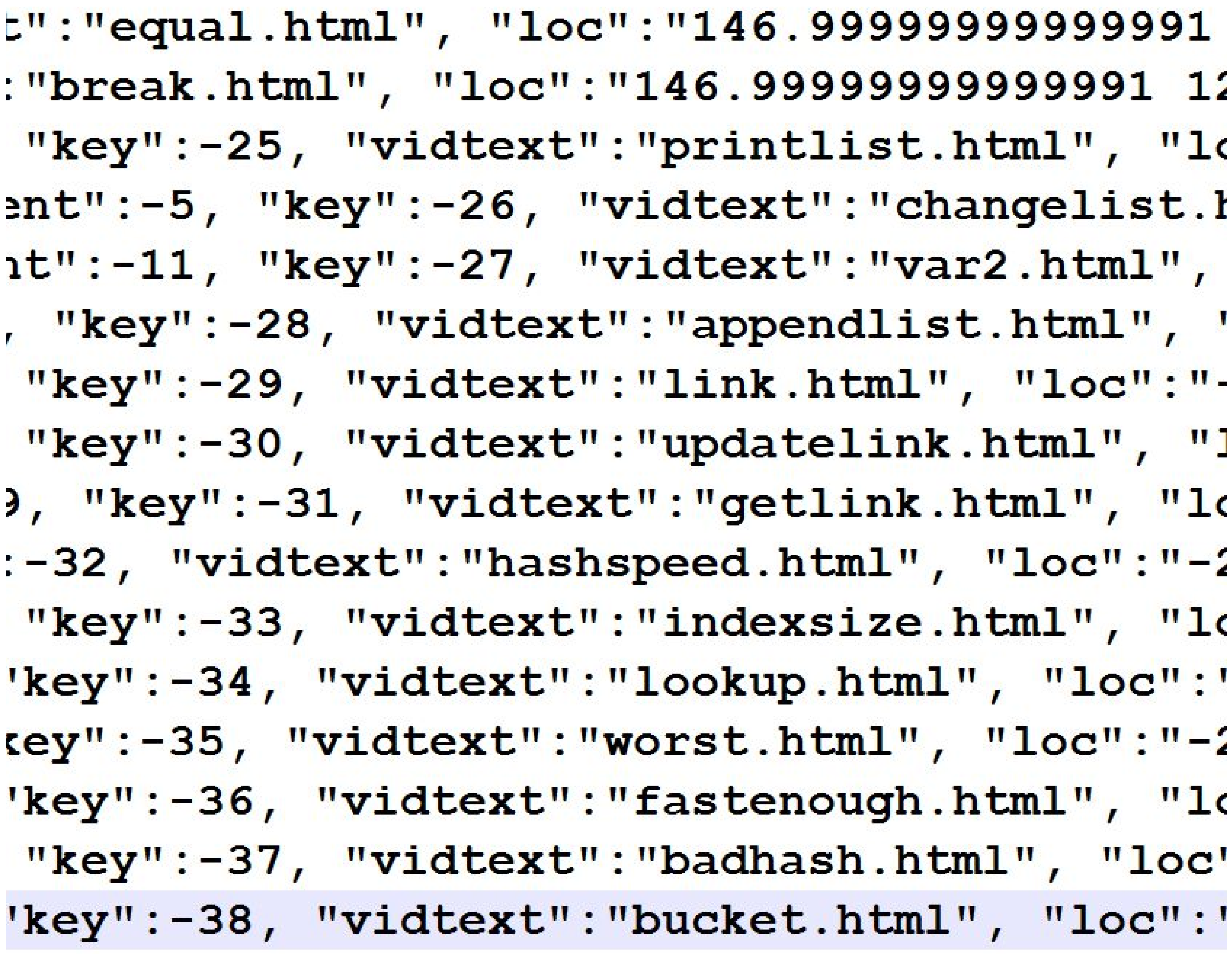


```
{ "class": "go.TreeModel",
  "nodeDataArray": [
{"key":0, "text":"Intro to Com Sci", "vidtext":"https://www.youtube.com/watch?v=9nkR2LLPiYo", "loc":"-65.99999999999997 7",
"font":"bold 13px sans-serif"},
{"text":"If Statements", "brush":"coral", "dir":"right", "parent":0, "key":-2, "vidtext":"if.html", "loc":"56.00000000000005 -48.8"
"scale":1, "font":"bold 13px sans-serif"},
{"text":"While Statements", "brush":"skyblue", "dir":"right", "parent":-2, "key":-3, "vidtext":"while.html",
"loc":"156.99999999999997 -68.29999999999997", "font":"bold 13px sans-serif"},
{"text":"Increments", "brush":"skyblue", "dir":"right", "parent":-2, "key":-4, "vidtext":"inc.html", "loc":"156.99999999999997
-29.299999999999997", "font":"bold 13px sans-serif"},
{"text":"Loops in Lists", "brush":"palevioletred", "dir":"left", "parent":0, "key":-5, "vidtext":"looplist.html",
"loc":"-86.00000000000006 -112.67567567567565", "font":"bold 13px sans-serif"},
{"text":"Aliasing", "brush":"palevioletred", "dir":"left", "parent":0, "key":-6, "vidtext":"aliasing.html",
"loc":"-85.99999999999994 -86.67567567567569", "font":"bold 13px sans-serif"},
{"text":"Data Structures", "brush":"palevioletred", "dir":"left", "parent":0, "key":-7, "vidtext":"datastruct.html",
"loc":"-86.00000000000003 -15.17567567567568", "font":"bold 13px sans-serif"},
{"text":"Hashing", "brush":"orange", "dir":"left", "parent":-7, "key":-8, "vidtext":"hash.html", "loc":"-200.99999999999997
-47.67567567567568", "font":"bold 13px sans-serif"},
{"text":"Strings", "brush":"palevioletred", "dir":"left", "parent":0, "key":-9, "vidtext":"string.html", "loc":"-85.99999999999994
82.67567567567568", "font":"bold 13px sans-serif"},
{"text":"Testing", "brush":"coral", "parent":0, "key":-10, "vidtext":"testing.html", "loc":"56.00000000000005 -4.700000000000004",
"font":"bold 13px sans-serif"},
{"text":"Evaluate Equations", "brush":"green", "dir":"right", "parent":-4, "key":-11, "vidtext":"var1.html",
"loc":"246.00000000000003 -42.3", "font":"bold 13px sans-serif"},
{"text":"Valid Strings", "brush":"grey", "dir":"left", "parent":-9, "key":-13, "vidtext":"validstring.html",
"loc":"150.99999999999997 56.67567567567568", "font":"bold 13px sans-serif"}
```

3. Each mindmap begins with a central node, which you can edit by double clicking the node. Right clicking the node will open up other options, such as bolding the node text, or changing the size of the texts.

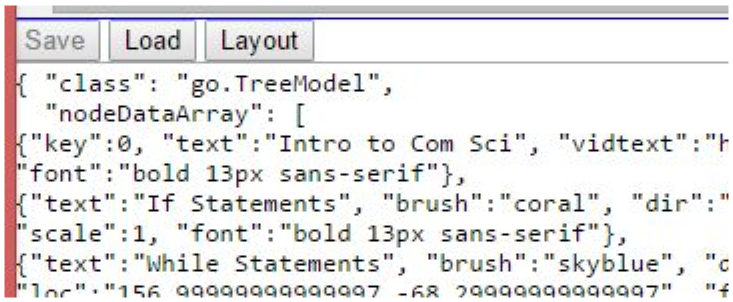


4. For students, there are two options called “Video” and “Homework” relating to the course material. Clicking “Video” will navigate towards the topic’s associating video, while the “Homework” will direct towards the topic’s homework, if there is any. To associate these nodes with video and homework links, you must edit the ***anotherMind.html*** and edit the piece of code shown below (under the nodeDataArray array). Under ***“vidtext”:”INSERT_URL_HERE”,*** you can place the html of the video, and under ***“hwtext:”INSERT_URL_HERE”,*** you can place the html of the homeworks.



```
t":"equal.html", "loc":"146.99999999999991
:"break.html", "loc":"146.99999999999991 12
 "key":-25, "vidtext":"printlist.html", "lc
ent":-5, "key":-26, "vidtext":"changelist.h
nt":-11, "key":-27, "vidtext":"var2.html",
, "key":-28, "vidtext":"appendlist.html", '
 "key":-29, "vidtext":"link.html", "loc":"-
 "key":-30, "vidtext":"updatelink.html", "l
9, "key":-31, "vidtext":"getlink.html", "lc
:-32, "vidtext":"hashspeed.html", "loc":"-2
 "key":-33, "vidtext":"indexsize.html", "lc
'key":-34, "vidtext":"lookup.html", "loc":'
key":-35, "vidtext":"worst.html", "loc":"-2
'key":-36, "vidtext":"fastenough.html", "lc
 "key":-37, "vidtext":"badhash.html", "loc'
'key":-38, "vidtext":"bucket.html", "loc":'
```

5. Everytime you edit the mindmap, you can click the “save” button under the ***anotherMind.html*** page, which should save the new nodeDataArray array under the text box below the mindmap. Copy and paste this code over the nodeDataArray part of the code in ***anotherMind.html*** using a text editor to update this code the next time you refresh the page.



```
{ "class": "go.TreeModel",
  "nodeDataArray": [
{"key":0, "text":"Intro to Com Sci", "vidtext":"h
"font":"bold 13px sans-serif"},
{"text":"If Statements", "brush":"coral", "dir":"
"scale":1, "font":"bold 13px sans-serif"},
{"text":"While Statements", "brush":"skyblue", "c
"loc":"156.99999999999997 -68.29999999999997", "f
```

6. There are different elements of the nodeDataArray object. These are:

brush : color of the node link

text: title of the node

parent: node parent

key: node’s unique key

vidtext: URL of the video for the node

hwtext: URL of the homework for the node

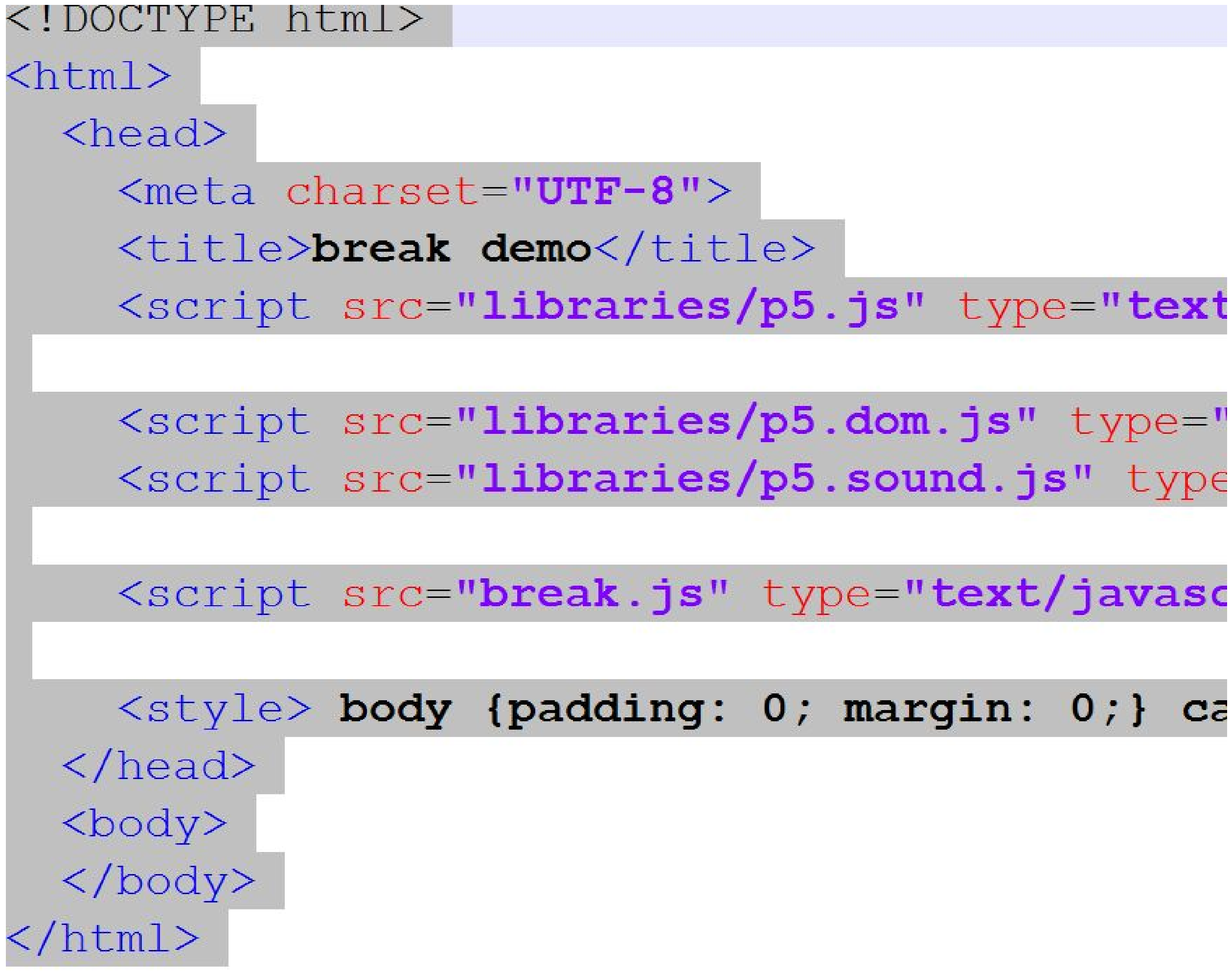
loc: location of the node on the browser

font: font size, color, bold, type

Using p5.js for your instructional videos

You can use the p5.js library to make your instructional videos more “interactive”. For example, if you want to quiz your students on subjects while they are watching your lectures, you can prompt them input boxes or questions on the screen for them to answer and gain feedback. Here are just some examples of what you can do with p5.js.

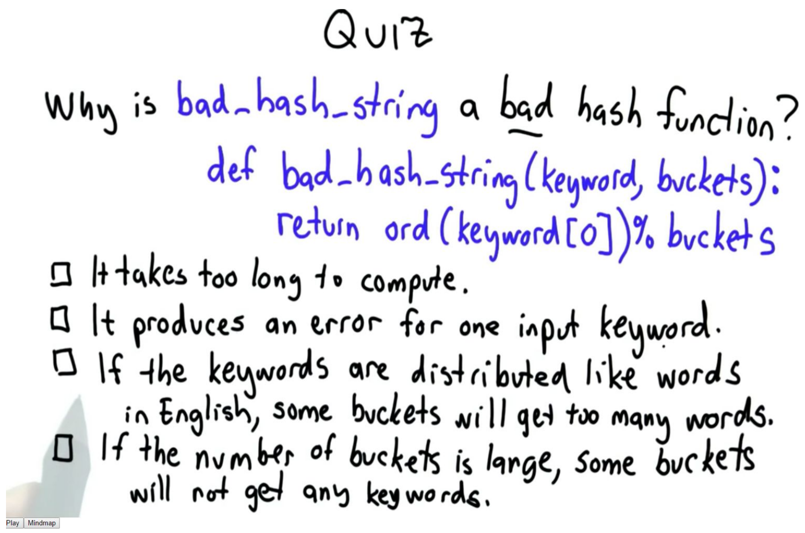
1. Each video requires a html and js file. The html should follow this general format, shown below. The only items you need to change for each html is the <title> section and <script src=”JAVASCRIPT_FILE” section. You should name the associating javascript file and the html file the same, with different extensions for convenience purposes.



```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>break demo</title>
    <script src="libraries/p5.js" type="text

    <script src="libraries/p5.dom.js" type="
    <script src="libraries/p5.sound.js" type

    <script src="break.js" type="text/javasc

    <style> body {padding: 0; margin: 0;} ca
  </head>
  <body>
  </body>
</html>
```

2. Many of the videos share similar attributes, so we created a separate library containing the functions for the setupVideo function found in each js file. This will set up the video to the appropriate size, as well as adding a slider to change the video time. There are several ways you can present questions to your students during lectures. These methods require different pieces of code to function.

Radio buttons/Clicking right answers:

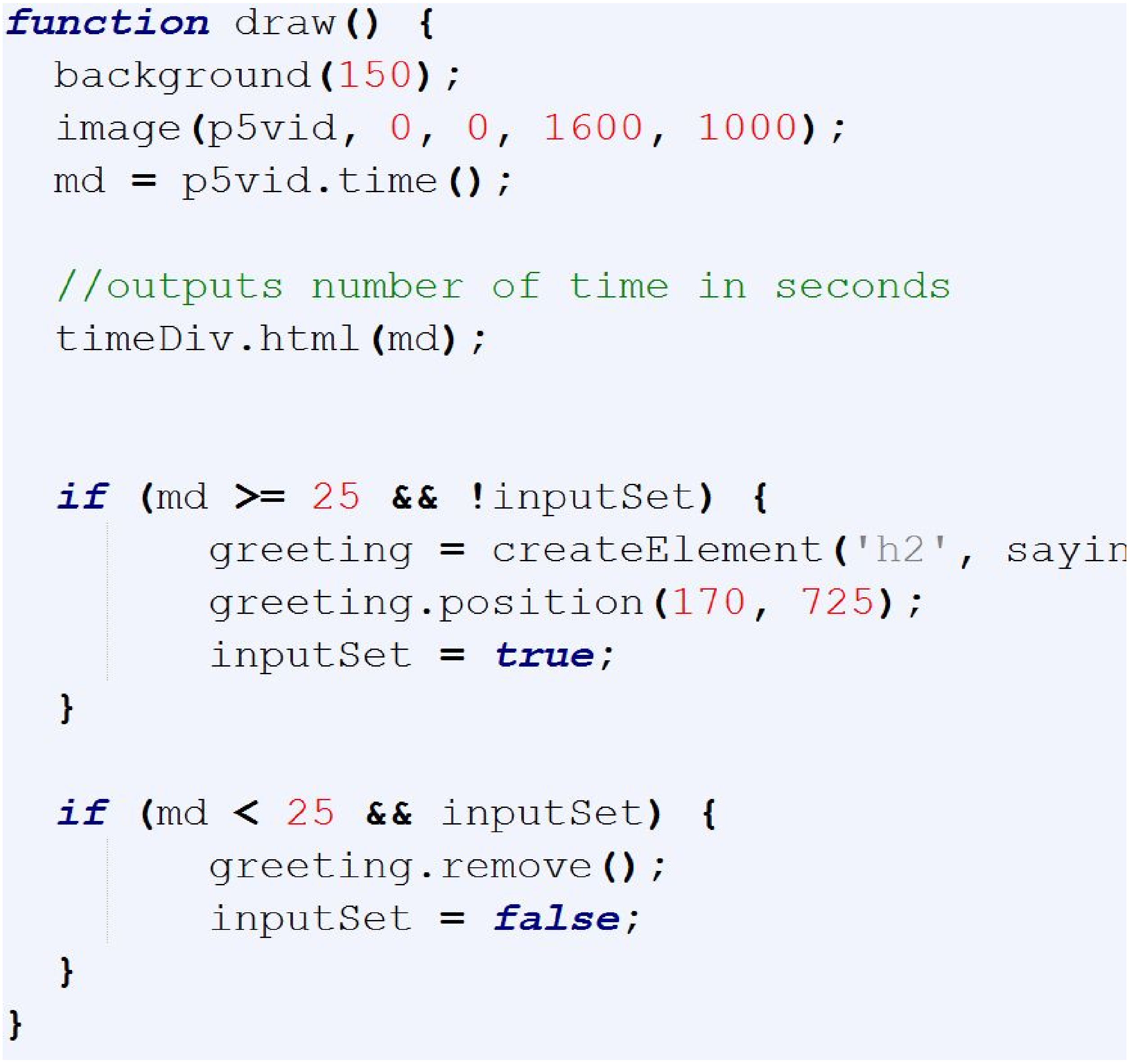

Quiz

Why is bad_hash_string a bad hash function?

```
def bad_hash_string(keyword, buckets):
    return ord(keyword[0])% buckets
```

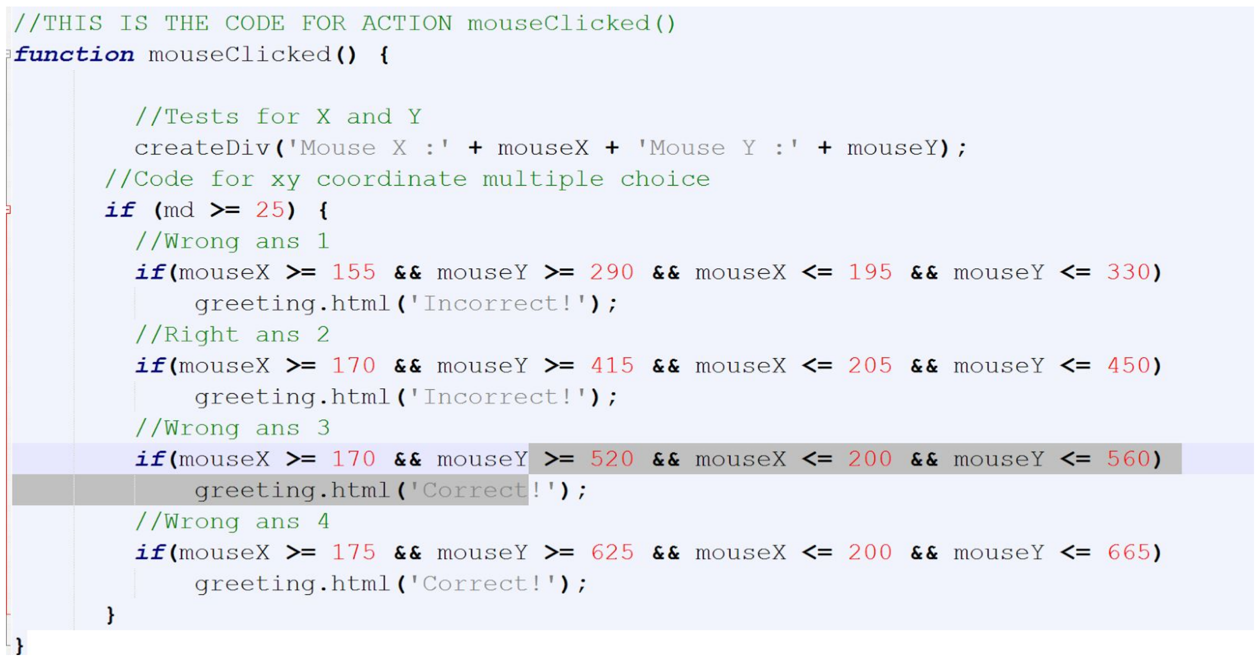
- ☐ It takes too long to compute.
- ☐ It produces an error for one input keyword.
- ☐ If the keywords are distributed like words in English, some buckets will get too many words.
- ☐ If the number of buckets is large, some buckets will not get any keywords.

This figure shows an example of the radio button. Click a certain answer will output “Correct” or “incorrect” on the screen. To accomplish this, you must change the mouseClicked() function of the js file. The variable “md” represents the time of the video, so you can set up the video so input boxes/radio buttons only appear at a certain point of time.



```
function draw() {
  background(150);
  image(p5vid, 0, 0, 1600, 1000);
  md = p5vid.time();

  //outputs number of time in seconds
  timeDiv.html(md);


  if (md >= 25 && !inputSet) {
        greeting = createElement('h2', sayin
        greeting.position(170, 725);
        inputSet = true;
  }

  if (md < 25 && inputSet) {
        greeting.remove();
        inputSet = false;
  }
}
```

The image above shows how md is used to have the prompts show up at a certain point of time. Follow this general template to have the button and greeting objects to appear at a certain time of the video.

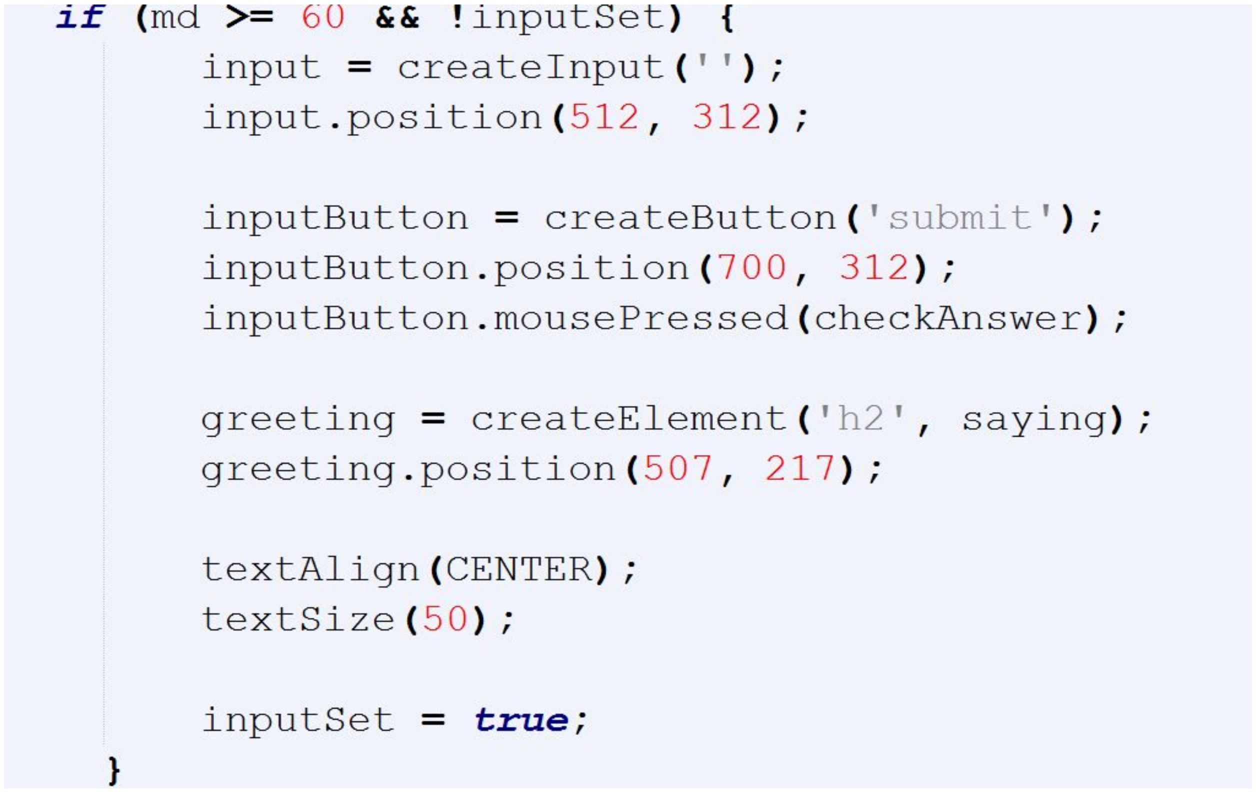


```
//THIS IS THE CODE FOR ACTION mouseClicked()
function mouseClicked() {

        //Tests for X and Y
        createDiv('Mouse X :' + mouseX + 'Mouse Y :' + mouseY);
      //Code for xy coordinate multiple choice
      if (md >= 25) {
        //Wrong ans 1
        if(mouseX >= 155 && mouseY >= 290 && mouseX <= 195 && mouseY <= 330)
            greeting.html('Incorrect!');
        //Right ans 2
        if(mouseX >= 170 && mouseY >= 415 && mouseX <= 205 && mouseY <= 450)
            greeting.html('Incorrect!');
        //Wrong ans 3
        if(mouseX >= 170 && mouseY >= 520 && mouseX <= 200 && mouseY <= 560)
            greeting.html('Correct!');
        //Wrong ans 4
        if(mouseX >= 175 && mouseY >= 625 && mouseX <= 200 && mouseY <= 665)
            greeting.html('Correct!');
      }
}
```

This image shows how you can set which answers are correct or incorrect. mouseX and mouseY represents the position of the mouse. It is recommended you have square shaped input boxes on the video so you can coordinate these positions. The “//Tests fo X and Y” code represents a testing feature for outputting the position of your mouse. To add more options, simply add another if() statement, and another greeting.html statement.

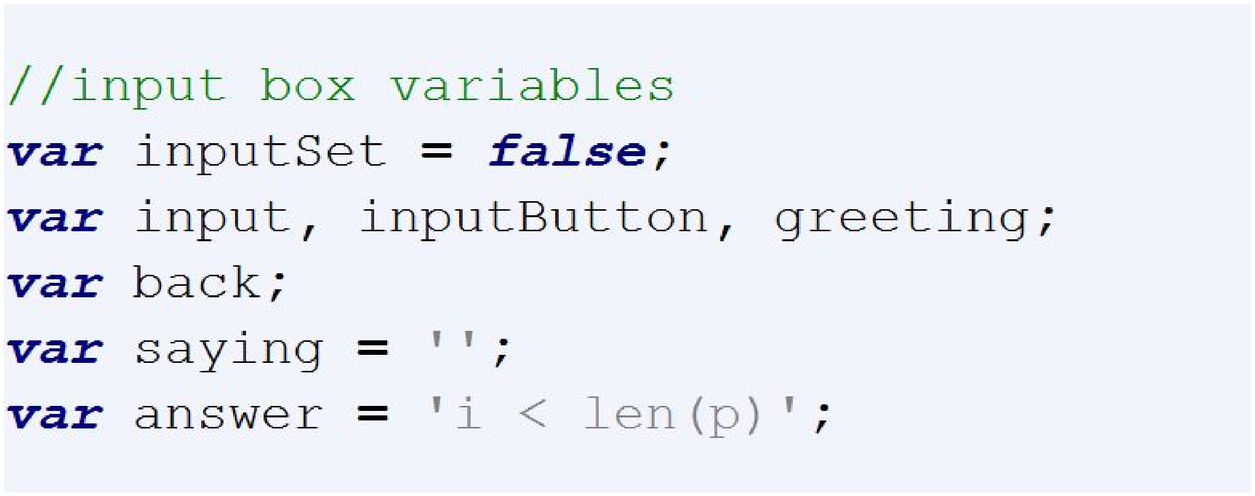
Input Boxes:

Input boxes have a similar setup with the radio buttons. The differences are:



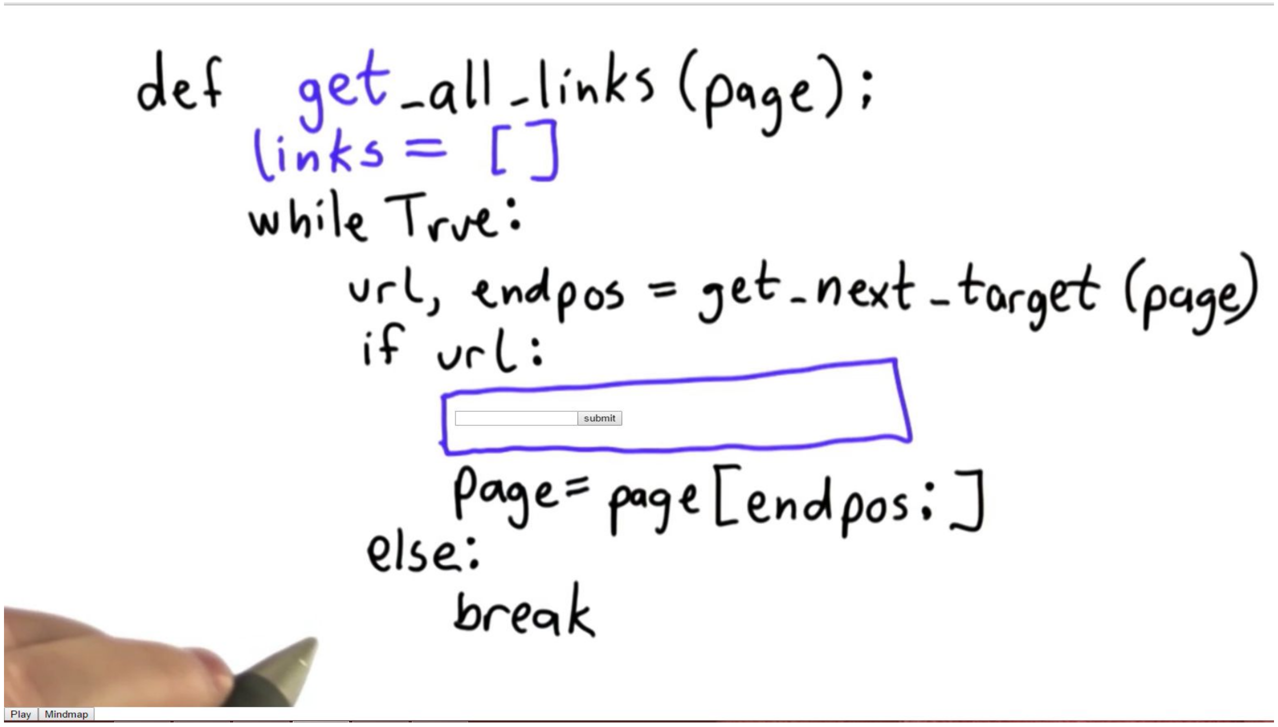
```
  if (md >= 60 && !inputSet) {
        input = createInput('');
        input.position(512, 312);

        inputButton = createButton('submit');
        inputButton.position(700, 312);
        inputButton.mousePressed(checkAnswer);

        greeting = createElement('h2', saying);
        greeting.position(507, 217);

        textAlign(CENTER);
        textSize(50);

        inputSet = true;
    }
```

This part of the code creates an input object. This will make a blank input box on the screen, along with a submit button. Whenever the user presses submit, it will check the answer, comparing it with this piece of code in the beginning:



```
//input box variables
var inputSet = false;
var input, inputButton, greeting;
var back;
var saying = '';
var answer = 'i < len(p)';
```

By changing the variable answer, you can change the answer to the question. The rest of the code should be similar to the templates. Input boxes are the easiest to incorporate, since you only have to change the position of the boxes, the variable md for video time, and the variable answer.



```
def get_all_links(page):
    links = []
    while True:
        url, endpos = get_next_target(page)
        if url:

            page= page[endpos:]
        else:
            break
```

Users can also press the “mindmap” button to the mindmap page.

There are many other features of p5.js you can use, like being able to interact with shapes on the canvas. The use of the p5.js API is recommended, and this tutorial serves to highlight the main uses of p5.js in an intellectual environment.