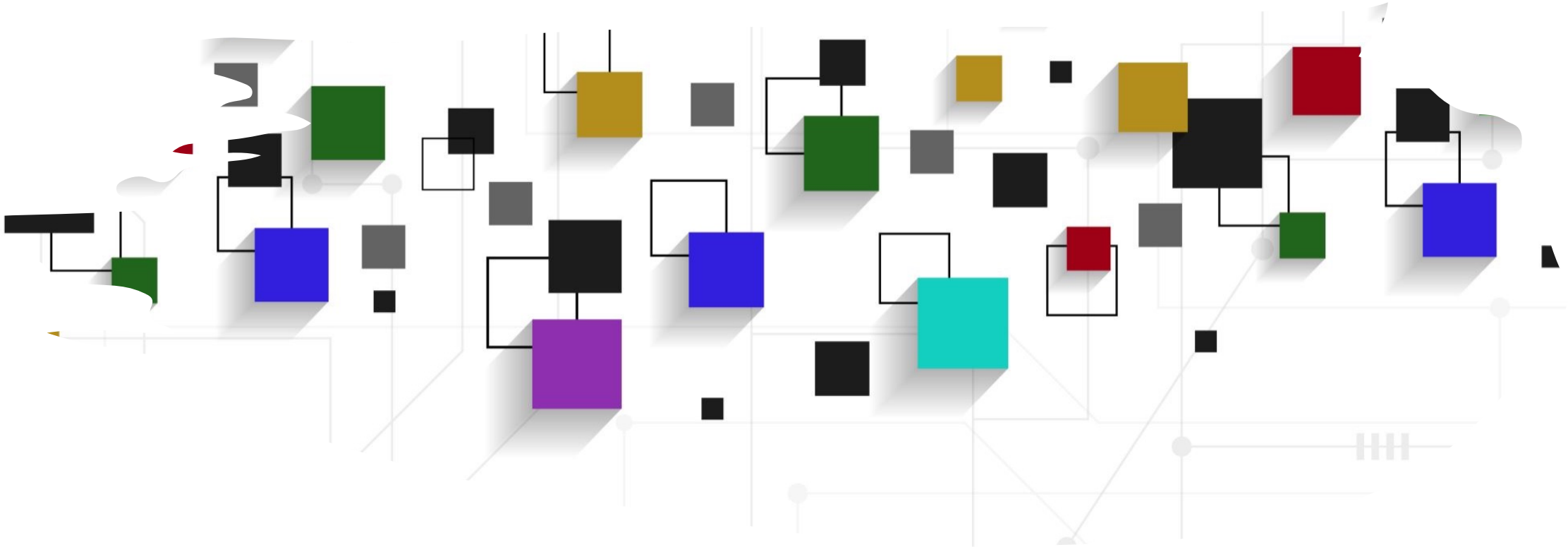


CogLab: jsPsych 101

WEEK 4

logistics: course schedule

Week	Date	Weekly Module
1	Thursday, August 31, 2023	W1: Getting started
2	Tuesday, September 5, 2023	W2: Replication & Design
2	Thursday, September 7, 2023	W2 continued…
2	Sunday, September, 10, 2023	**Project Milestone #1 (Team Plan + Review) Due**
3	Tuesday, September 12, 2023	W3: Experiment Anatomy
3	Thursday, September 14, 2023	W3 continued…
3	Sunday, September, 17, 2023	**Project Milestone #2 (QALMRI) Due**
4	Tuesday, September 19, 2023	W4: jsPsych 101
4	Thursday, September 21, 2023	W4 continued…
4	Sunday, September, 24, 2023	**Project Milestone #3 (Project Proposal) Due**
5	Tuesday, September 26, 2023	W5: Experiment Timeline
5	Thursday, September 28, 2023	W5 continued…
5	Sunday, Oct 1, 2023	**Project Milestone #4 (Design Draft) Due**
6	Tuesday, October 3, 2023	W6: Recording Data
6	Thursday, October 5, 2023	W6 continued…
7	Tuesday, October 10, 2023	**Fall Break!! NO CLASS**

logistics: project milestone #3

- a 5-page project proposal
- a coherent narrative that weaves together the articles you've read so far and asks a novel question
- assign tasks/writing but then come back and re-read: a team effort
- talk about how to make sure everyone is contributing equally

https://www.reddit.com/r/sciencememes/comments/k2oowa/research_proposals_be_like/

recap: Sep 14, 2023

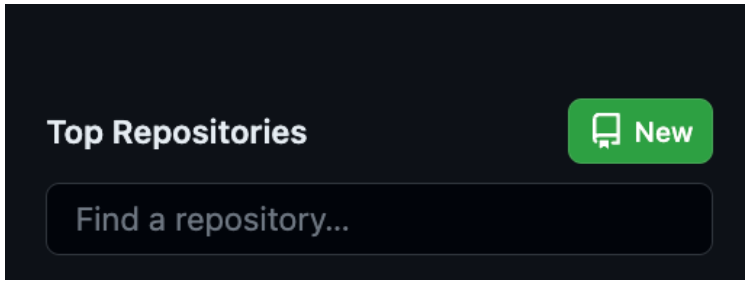
- what we covered:
  - setting up a workflow via github (clone/pull/add/commit/push)
  - building your first webpage! (HTML / CSS)
- your to-dos were:
  - *prep:* de Leeuw (2015)
  - *prep*: A non-technical introduction to basic coding concepts
  - *try*: Week 3 Quiz
  - *apply*: Project Milestone 2 (QALMRI + SPARK)
  - *apply*: Optional Meme

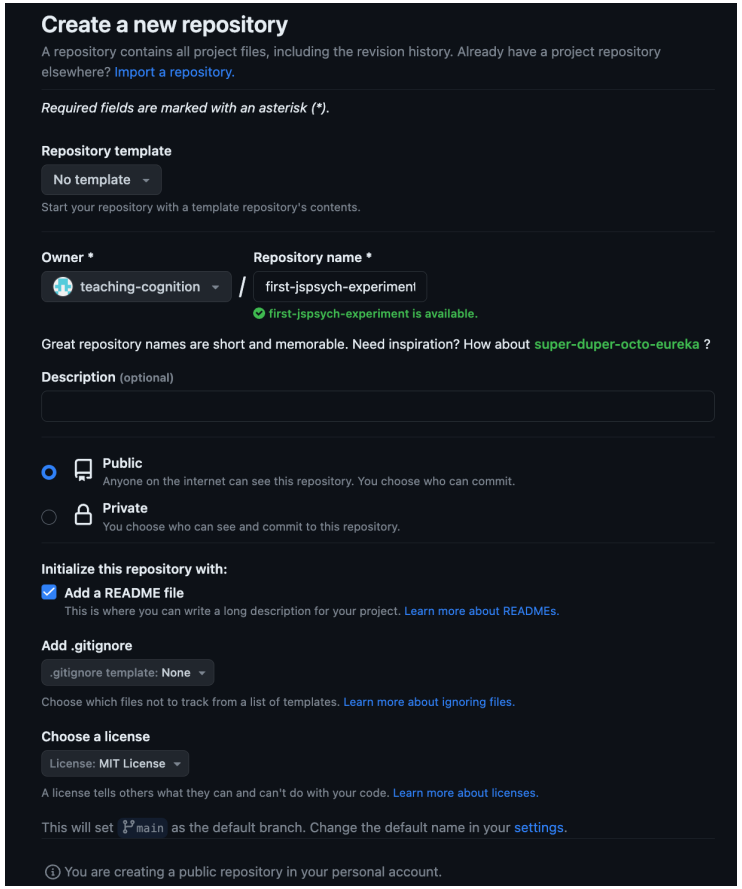
today's agenda

- review creating a HTML page
- scripts in HTML: jsPsych 101
- your first experiment!

where do we start?

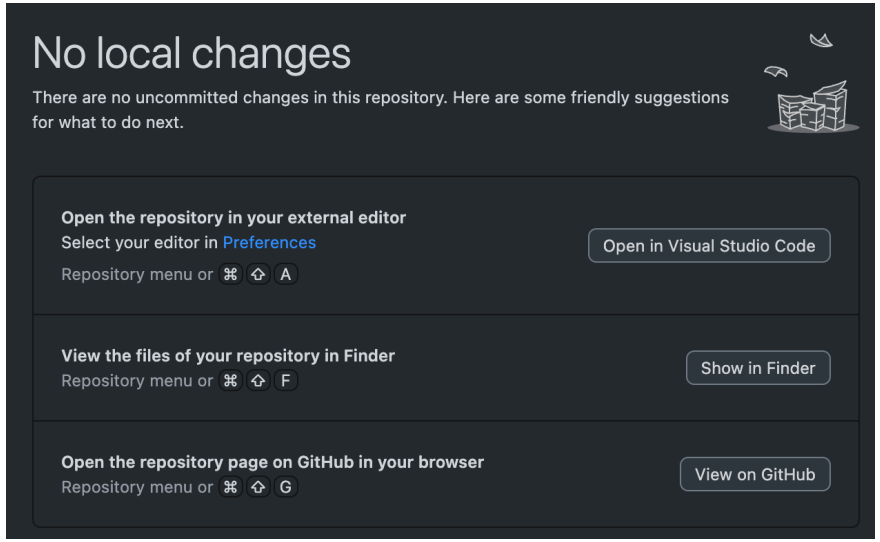
- on github.com, login and create a new repository
  - first-jspsych-experiment
  - public, add a README, MIT license

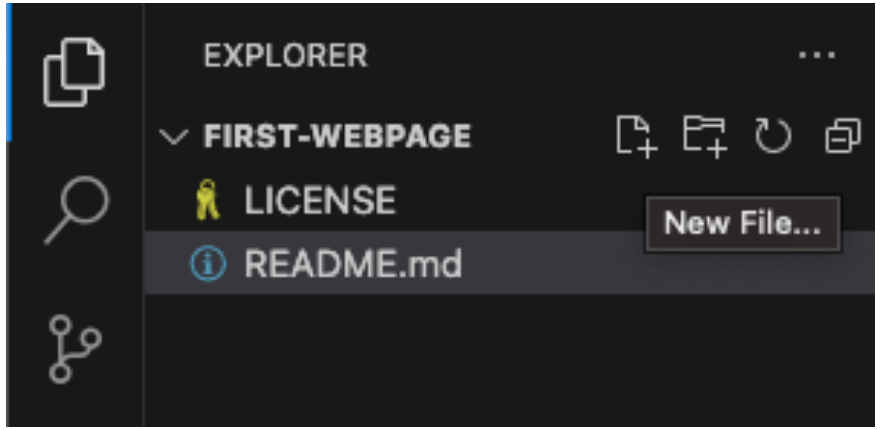




step 1: index.html

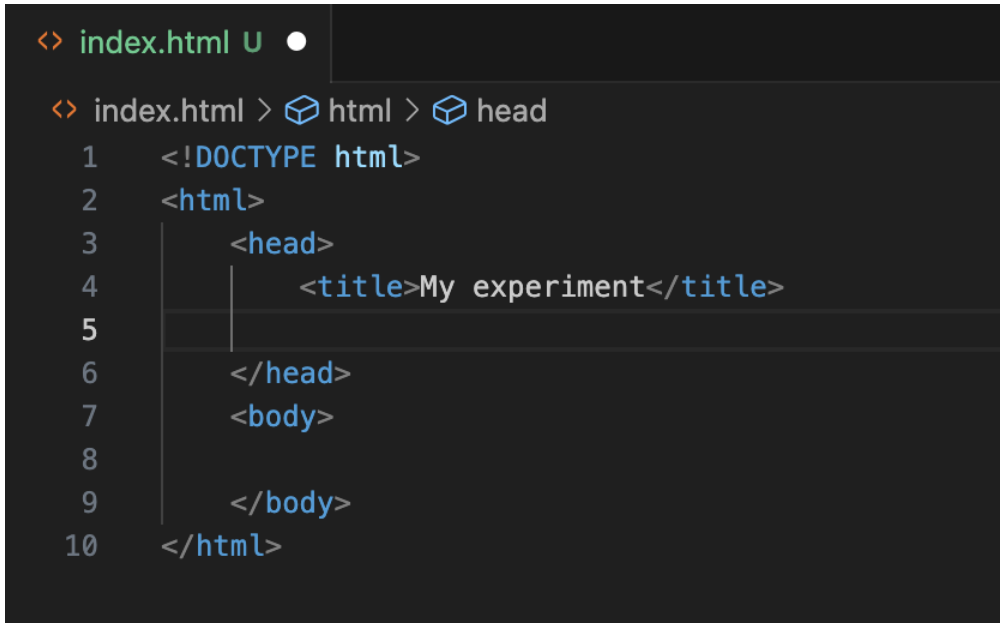
- open Github Desktop & clone this new repo to a local folder inside PSYC 2740
- open this folder in Visual Studio Code
- create a new file called index.html





step 1: index.html

- inside the code editor
  - set up basic HTML syntax
  - doctype, head & title, body
- save and open page in browser



experiment recap

training

association

x 3

sentence

space

novel word?

<response>

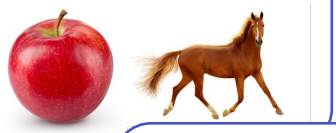
word

<response>

x 3

priming

+



prime

target

A / L

using jsPsych as your experiment builder

- jsPsych is an open-source framework to create web-based experiments, created by Josh de Leeuw (Vassar)
- jsPsych is built using HTML, CSS, and Javascript
  - HTML: the backbone of all webpages (content)
  - CSS: the styling guide
  - Javascript: a language for building interactive webpages
    - 98% of webpages use Javascript!

why jsPsych?

- open-source! great online support
- simplifies technical aspects of coding
- everything happens inside ONE html page (almost)

Measuring sequences of keystrokes with jsPsych: Reliability of response times and interkeystroke intervals

S. Pinet, C. Zielinski, S. Mathôt, S. Dufau, F.-X. Alario & M. Longcamp

Behavior Research Methods **49**, 1163–1176 (2017) | Cite this article

Data quality of platforms and panels for online behavioral research

Eyal Peer, David Rothschild, Andrew Gordon, Zak Evernden & Ekaterina Damer

Behavior Research Methods **54**, 1643–1662 (2022) | Cite this article

(approval ratings). We found considerable differences between the sites, especially in comprehension, attention, and dishonesty. In Study 1 (without filters), we found that only Prolific provided high data quality on all measures. In Study 2 (with filters), we found high data quality among CloudResearch and Prolific. MTurk showed alarmingly low data quality even with data quality filters. We also found that while reputation (approval rating) did not predict data quality, frequency and purpose of usage did, especially on MTurk: the lowest data quality came from MTurk participants who report using the site as their main source of income but spend few hours on it per week. We provide a framework for future investigation

Realistic precision and accuracy of online experiment platforms, web browsers, and devices

Alexander Anwyl-Irvine, Edwin S. Dalmaijer, Nick Hodges & Jo K. Evershed

Behavior Research Methods **53**, 1407–1425 (2021) | Cite this article

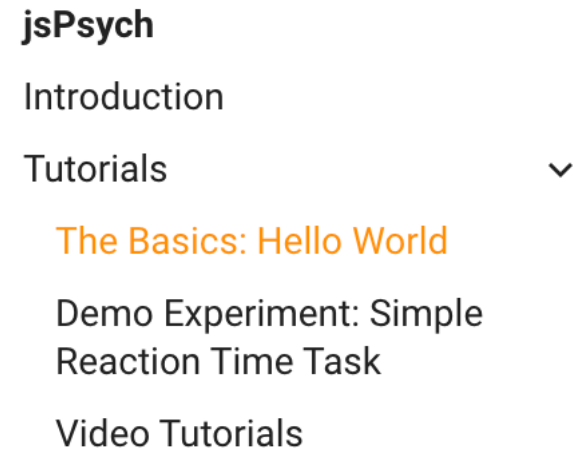
software to provide context to our findings. We found that modern web platforms provide reasonable accuracy and precision for display duration and manual response time, and that no single platform stands out as the best in all features and conditions. In addition, our online

how does jsPsych work?

- jsPsych has pre-designed "plugins" that help us create "screens" for different phases of our experiment
- these plugins vary in several ways:
  - what is displayed on the screen (text / image / checkbox etc.)
  - what can be recorded (keypress, audio, clicks, etc.)
  - how long to display/record
  - appearance
  - etc…

setting up jsPsych

- to use a plugin inside our HTML file, we need to "load" that plugin's information into our index.html file
- go to the jsPsych website
- click on Tutorials > The Basics
- choose the simplest possible setup



Choose your own (setup) adventure

Starting with the release of version 7.0 of jsPsych there are three different ways that you can add jsPsych to your project. Which approach you choose will depend on what your goals are.

- **I want the simplest possible setup**. This approach involves using scripts that are hosted on a CDN. You do not need to download or install anything to start using jsPsych. The limitation is that you cannot customize the library. For most experiments, this approach will be sufficient.
- **I want to be able to do some customization, but have a simple setup.**. This approach involves downloading a bundle of scripts that make up jsPsych. *If you used jsPsych prior to version 7.0, this will feel like the most familiar approach*. Having your own copy of the scripts means that you can make modifications to the library such as tweaking plugin behavior.
- **I want to use modern JavaScript tooling, like `npm` and `import` statements.** You can install jsPsych, plugins for jsPsych, and extensions for jsPsych from NPM. This approach allows you to integrate jsPsych into your favorite JavaScript frameworks and get the benefits of TypeScript, bundlers, and more.

step 2: setting up jsPsych

- notice that you have already completed step 1!
- go to step 2: loading the library
- copy the <script> and <link> lines and add them to your index.html page inside the <head> tag
- what do you think we're doing?

```
<!DOCTYPE html>
<html>
  <head>
    <title>My experiment</title>
    <script src="https://unpkg.com/jspsych@7.3.3"></script>
    <link href="https://unpkg.com/jspsych@7.3.3/css/jspsych.css" rel="stylesheet"
  </head>
  <body></body>
</html>
```

index.html > html

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>My experiment</title>
5          <script src="https://unpkg.com/jspsych@7.3.3"></script>
6          <link href="https://unpkg.com/jspsych@7.3.3/css/jspsych.css" rel="stylesheet" type="text/css" />
7      </head>
8      <body>
9
10     </body>
11 </html>
```

step 3: initialize jsPsych

- add <script> tags after your body tags
- within the <script> tags, create a new variable called jsPsych that stores the output of the initJsPsych() function
  - const : indicates that you are defining a **constant**, it will never change value
  - jsPsych : this is the **name** of the constant
  - initJsPsych : this is a **function** that initializes JSPsych when your page is loaded

```
<!DOCTYPE html>
<html>
  <head>
    <title>My experiment</title>
    <script src="https://unpkg.com/jspsych@7.3.3"></script>
    <link href="https://unpkg.com/jspsych@7.3.3/css/jspsych.css" rel="stylesheet"
  </head>
  <body></body>
  <script>
  </script>
</html>
```

```
<!DOCTYPE html>
<html>
  <head>
    <title>My experiment</title>
    <script src="https://unpkg.com/jspsych@7.3.3"></script>
    <link href="https://unpkg.com/jspsych@7.3.3/css/jspsych.css" rel="stylesheet"
  </head>
  <body></body>
  <script>
    const jsPsych = initJsPsych();
  </script>
</html>
```

basic coding concepts

- datatypes
  - *string*: always in quotes
  - *integer/float*: some type of number
  - *boolean/logical*: can only be TRUE or FALSE
- variables
  - x = 4 (scalar) vs. y = ['ABC', 'DEF'] (vector/array)
  - objects/dictionaries containing keys and values, e.g., Alice = {major: 'math', year: 'junior'}
- functions
  - z = add(x,y): add is the function, x & y are arguments, z is a variable that stores the value returned by the function
- operators
  - assignment (=), conditional (<, ==), arithmetic (+, -), logical (&, | |)

step 4: your first plugin

- we want to display some words on the screen
- we can use the html-keyboard-response plugin for this
- first we load the plugin into HTML using the <script> tags inside <head>

```
<!DOCTYPE html>
<html>
  <head>
    <title>My experiment</title>
    <script src="https://unpkg.com/jspsych@7.3.3"></script>
    <script src="https://unpkg.com/@jspsych/plugin-html-keyboard-response@1.1.2"><
    <link href="https://unpkg.com/jspsych@7.3.3/css/jspsych.css" rel="stylesheet"
  </head>
  <body></body>
  <script>
    const jsPsych = initJsPsych();
  </script>
</html>
```

step 4: your first plugin

- next, we define a trial called hello_trial that will display words on the screen
  - const : indicates that you are defining a **constant**, it will never change value
  - type: what kind of "screen" do you want to create?
    - a screen that prints a message and asks for a keyboard response
  - stimulus: the specific words to be displayed
  - type and stimulus are **parameters/keys** of hello_trial

```
<!DOCTYPE html>
<html>
  <head>
    <title>My experiment</title>
    <script src="https://unpkg.com/jspsych@7.3.3"></script>
    <script src="https://unpkg.com/@jspsych/plugin-html-keyboard-response@1.1.2"><
    <link href="https://unpkg.com/jspsych@7.3.3/css/jspsych.css" rel="stylesheet"
  </head>
  <body></body>
  <script>
    const jsPsych = initJsPsych();

    const hello_trial = {
      type: jsPsychHtmlKeyboardResponse,
      stimulus: 'Hello world!'
    }
  </script>
</html>
```

step 5: run the experiment

- so far, we have defined our trial but haven't actually "run" the experiment
- jsPsych.run() is a **function** that tells HTML to run the trial called hello_trial
- save your index.html file
- open it in the browser

```
<!DOCTYPE html>
<html>
  <head>
    <title>My experiment</title>
    <script src="https://unpkg.com/jspsych@7.3.3"></script>
    <script src="https://unpkg.com/@jspsych/plugin-html-keyboard-response@1.1.2"><
    <link href="https://unpkg.com/jspsych@7.3.3/css/jspsych.css" rel="stylesheet"
  </head>
  <body></body>
  <script>
    const jsPsych = initJsPsych();

    const hello_trial = {
      type: jsPsychHtmlKeyboardResponse,
      stimulus: 'Hello world!'
    }

    jsPsych.run([hello_trial]);
  </script>
</html>
```

HTML vs. jsPsych?

- you already knew how to print a message or any text: how?
- jSPsych allows you to interface with a user
  - you can print a message and ask someone to respond
- in your browser, open/refresh index.html and try pressing a key
  - this will "end" the experiment and make your message disappear
  - this is not something we could do earlier!
  - this allows us to define event sequences
  - we can build a **timeline** of events for our experiment
  - we can even control what is displayed and when
  - we can also control what input is accepted or rejected!

restricting input via html-keyboard-response

- this plugin allows you to get a keyboard response from a user or participant and has some parameters you can control from the experimenter end
  - **stimulus**: what the participant sees and responds to
  - **choices**: what the participant can type
  - **prompt**: extra space for more stimulus content
  - **stimulus_duration**: how long do we want our stimulus to be displayed
  - **trial_duration**: how long do we want this trial to last
  - **response_ends_trial**: whether we want the trial to end when the participant responds or not

```
<!DOCTYPE html>
<html>
  <head>
    <title>My experiment</title>
    <script src="https://unpkg.com/jspsych@7.3.3"></script>
    <script src="https://unpkg.com/@jspsych/plugin-html-keyboard-response@1.1.2"><
    <link href="https://unpkg.com/jspsych@7.3.3/css/jspsych.css" rel="stylesheet"
  </head>
  <body></body>
  <script>
    const jsPsych = initJsPsych();

    const hello_trial = {
      type: jsPsychHtmlKeyboardResponse,
      stimulus: 'Hello world!'
    }

    jsPsych.run([hello_trial]);
  </script>
</html>
```

modifying our original plugin definition

- try restricting the choices to only certain keys such as Y or N
  - make sure Y and N are strings
- save your file
- reload in browser
- the message should disappear (i.e., experiment ends) only when Y or N are pressed

```
<!DOCTYPE html>
<html>
  <head>
    <title>My experiment</title>
    <script src="https://unpkg.com/jspsych@7.3.3"></script>
    <script src="https://unpkg.com/@jspsych/plugin-html-keyboard-response@1.1.2"></script>
    <link href="https://unpkg.com/jspsych@7.3.3/css/jspsych.css" rel="stylesheet" type="text/css" />
  </head>
  <body></body>
  <script>
    const jsPsych = initJsPsych();

    const hello_trial = {
      type: jsPsychHtmlKeyboardResponse,
      stimulus: 'Hello world!',
      choices: ['Y', 'N']
    }

    jsPsych.run([hello_trial]);
  </script>
</html>

```

it's a wonderful jsPsych world!

- SO many plugins!
  - button clicks, keyboard responses, image responses, etc.
- now we can start to build our experiment timeline
  - one plugin at a time!
  - going beyond a single screen to a *sequence* of events

experiment recap

training

association

x 3

sentence

space

novel word?

<response>

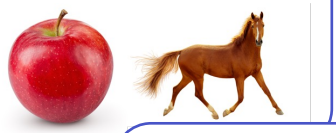
word

<response>

x 3

priming

+



prime

target

A / L

basic coding concepts

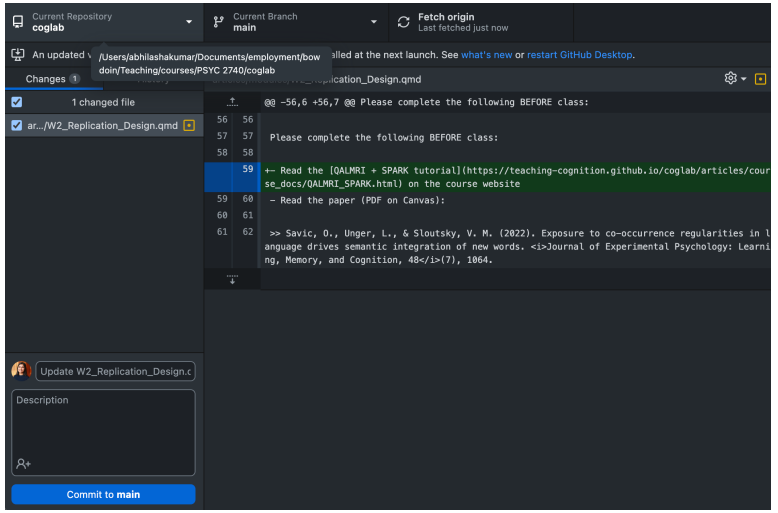
- datatypes
  - *string*: always in quotes
  - *integer/float*: some type of number
  - *boolean/logical*: can only be TRUE or FALSE
- variables
  - x = 4 (scalar) vs. y = ['ABC', 'DEF'] (vector/array)
  - objects/dictionaries containing keys and values, e.g., Alice = {major: 'math', year: 'junior'}
- functions
  - z = add(x,y): add is the function, x & y are arguments, z is a variable that stores the value returned by the function
- operators
  - assignment (=), conditional (<, ==), arithmetic (+, -), logical (&, | |)

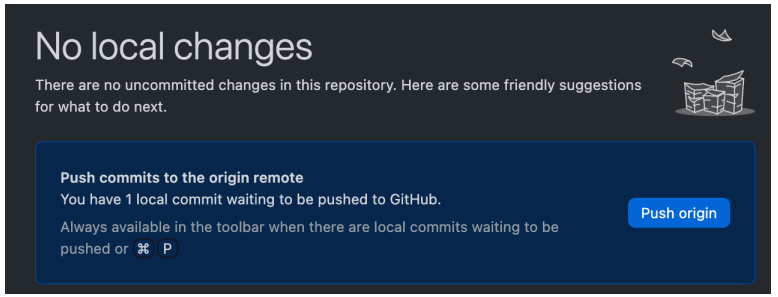
identify the coding concepts

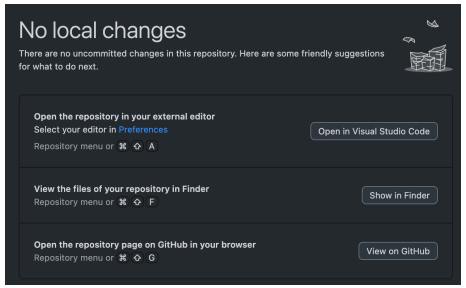
```
<!DOCTYPE html>
<html>
  <head>
    <title>My experiment</title>
    <script src="https://unpkg.com/jspsych@7.3.3"></script>
    <script src="https://unpkg.com/@jspsych/plugin-html-keyboard-response@1.1.2"></script>
    <link href="https://unpkg.com/jspsych@7.3.3/css/jspsych.css" rel="stylesheet" type="text/css" />
  </head>
  <body></body>
  <script>
    const jsPsych = initJsPsych();

    const hello_trial = {
      type: jsPsychHtmlKeyboardResponse,
      stimulus: 'Hello world!',
      choices: ['Y', 'N']
    }

    jsPsych.run([hello_trial]);
  </script>
</html>

```

saving your progress so far...

- save your index.html file
- open GitHub Desktop
- review changes, commit, and push
- check if changes have reflected online!







next class

- **before** class
  - *read*: jsPsych webpage on plugins
- **during** class
  - building an experiment timeline