

CogLab: going online!

WEEK 5

going back to our experiment

- open Visual Studio Xcode and open the jsPsych experiment you created last week
- also open the index.html file in your browser to remind yourself of what we did!

jarring image/prime/target

```
var image = {
  type: jsPsychImageKeyboardResponse,
  stimulus: jsPsych.timelineVariable('image_path'),
  choices: "NO_KEYS",
  trial_duration: 10,
  stimulus_width: 500,
  maintain_aspect_ratio: true,
  prompt: "<span style = 'font-size:200%'><br><br></span>",
  data: {
    | | | | |
    | | | | | typeoftrial: 'image',
    | | | | | },
};
```

```
var prime = {
  type: jsPsychImageKeyboardResponse,
  stimulus: jsPsych.timelineVariable('image_path'),
  trial_duration: 10,
  choices: "NO_KEYS",
  stimulus_width: 500,
  maintain_aspect_ratio: true,
  prompt: function(){
    | return "<span style= 'font-size:200%'><br>" + String(jsPsych.timelineVariable('prime_word')) + "<br></span>";
  },
  data: {
    | | | | |
    | | | | | typeoftrial: 'prime',
    | | | | | },
};
```

jarring image/prime/target

```
var target = {
  type: jsPsychImageKeyboardResponse,
  stimulus: jsPsych.timelineVariable('image_path'),
  choices: ['A', 'L'],
  stimulus_width: 500,
  maintain_aspect_ratio: true,
  trial_duration: 10.
  prompt: function(){
    return "<span style= 'font-size:200%'><br>" + String(jsPsych.timelineVariable('target_word')) + "<br></span>";
  },
  data: {
    typeoftrial: 'target',
    target: jsPsych.timelineVariable('target_word'),
    prime: jsPsych.timelineVariable('prime_word'),
    type: jsPsych.timelineVariable('type'),
    relatedness: jsPsych.timelineVariable('relatedness'),
    correct_key: jsPsych.timelineVariable('correct_key'),
    block_number: jsPsych.timelineVariable('block_number')
  },
  on_finish: function(data){
    data.correct = jsPsych.pluginAPI.compareKeys(data.response, data.correct_key);
  }
}
```

evaluating attention responses: 2

- we use `jsPsych.pluginAPI.compareKeys()` to compare the participant response to each novel word
- note the use of the **OR (||)** operator: if ANY of the novel words are mentioned, it is recorded as correct

```
var attention = {
  type: jsPsychSurveyText,
  questions: [{prompt: "Type any ONE novel word from the previous sentence:"}],
  data: {
    typeoftrial: 'attention',
  },

  on_finish: function(data){
    var last_trial_data = jsPsych.data.get().filter({typeoftrial: 'sentence'}).last(1).values()[0];
    console.log("last_trial_data= ", last_trial_data);

    data.novel1 = last_trial_data.novel1;
    data.novel2 = last_trial_data.novel2;
    data.novel3 = last_trial_data.novel3;
    data.response = data.response.Q0;

    if(
      jsPsych.pluginAPI.compareKeys(data.response, data.novel1) ||
      jsPsych.pluginAPI.compareKeys(data.response, data.novel2) ||
      jsPsych.pluginAPI.compareKeys(data.response, data.novel3)
    ){
      data.correct = 1;
      console.log("correct= " + data.correct);
    } else {
      data.correct = 0;
      console.log("correct= " + data.correct);
    }
  },
};
```

evaluating attention responses: 2

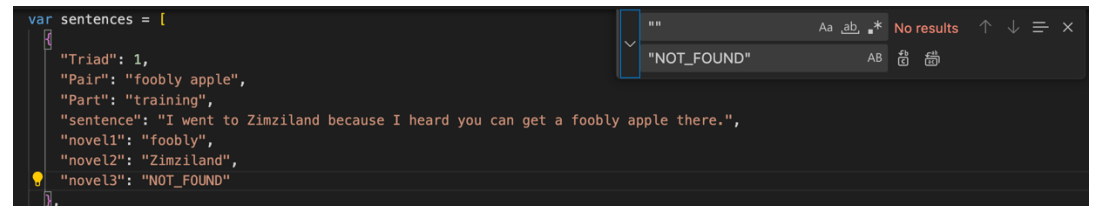
- save and re-run the `training_procedure`
- check that the attention trial now has a key storing whether the response typed in is correct or not

```
{  
  "rt": 775,  
  "response": "dsadas",  
  "typeoftrial": "attention",  
  "trial_type": "survey-text",  
  "trial_index": 15,  
  "time_elapsed": 4301,  
  "internal_node_id": "0.0-2.0-0.0-1.12-0.12",  
  "ID": 64255361,  
  "novel1": "dodish",  
  "novel2": "geck",  
  "novel3": "",  
  "correct": 0  
},
```

evaluating attention responses

- pressing return/enter is being coded as a correct response as novel2/novel3 contain ""
- go to sentences.js and **Command + F** for "" and replace with **NOT_FOUND**
- save and re-run

```
{  
  "rt": 775,  
  "response": "dsadas",  
  "typeoftrial": "attention",  
  "trial_type": "survey-text",  
  "trial_index": 15,  
  "time_elapsed": 4301,  
  "internal_node_id": "0.0-2.0-0.0-1.12-0.12",  
  "ID": 64255361,  
  "novel1": "dodish",  
  "novel2": "geck",  
  "novel3": "",  
  "correct": 0  
},
```



The screenshot shows a code editor with a search bar at the top right. The search bar contains two empty strings "" and shows "No results". Below the search bar, a dropdown menu displays "NOT_FOUND". In the code editor, a variable named 'sentences' is defined as an array of JSON objects. The third object in the array has a 'novel3' property set to an empty string, which has been replaced with 'NOT_FOUND'.

```
var sentences = [  
  {  
    "Triad": 1,  
    "Pair": "foobly apple",  
    "Part": "training",  
    "sentence": "I went to Zimziland because I heard you can get a foobly apple there.",  
    "novel1": "foobly",  
    "novel2": "Zimziland",  
    "novel3": "NOT_FOUND"  
  },
```

other data?

- adding a subject ID to the data
- each time the code is run, generate a random number and store it as the ID
- print this id using console.log
- save and reload, open your inspector

```
const jsPsych = initJsPsych(  
  {  
    on_finish: function(data) {  
      jsPsych.data.displayData();  
    }  
  }  
);  
  
// generate random ID  
  
var id = Math.floor(Math.random() * 1000000000);  
console.log("id =", id)
```


adding subject ID to trials

- we also want to attach this ID to all our trials
- two options:
 - manually by using the data parameter for all plugins
 - jsPsych also has a shortcut for this using `addProperties`

```
// generate random ID

var id = Math.floor(Math.random() * 1000000000);
console.log("id =", id)

// tag all trials with ID

jsPsych.data.addProperties({
  ID: id
});
```

checking ID is being recorded

- look at the data being generated by the experiment
- ALL trials should have an ID associated with them

```
[
  {
    "rt": null,
    "stimulus": "Sometimes I wish it were easier to get a foobly mipp.",
    "response": null,
    "typeoftrial": "sentence",
    "sentence": "Sometimes I wish it were easier to get a foobly mipp.",
    "novel1": "foobly",
    "novel2": "mipp",
    "novel3": "",
    "trial_type": "html-keyboard-response",
    "trial_index": 0,
    "time_elapsed": 107,
    "internal_node_id": "0.0-0.0-0.0",
    "ID": 88255443
  },
  {
    "rt": null,
    "stimulus": "I would love to see a dodish horse.",
    "response": null,
    "typeoftrial": "sentence",
    "sentence": "I would love to see a dodish horse.",
    "novel1": "dodish",
    "novel2": "",
    "novel3": "",
    "trial_type": "html-keyboard-response",
    "trial_index": 1,
    "time_elapsed": 209,
    "internal_node_id": "0.0-0.0-0.1",
    "ID": 88255443
  }
]
```

other nuts and bolts?

- adding a thank you screen
- adding a practice session
- preloading images
- going online!
- logistics

adding a thank you screen

- define & run a `thank_you` screen at the end
- save and reload

```
var thank_you = {
  type: jsPsychHtmlKeyboardResponse,
  stimulus: 'Thank you! You can press any key to end the experiment.',
  data: {
    typeoftrial: 'thank_you'
  },
}

jsPsych.run([initial_instructions, training_plus_association, priming_instructions, priming_proc, thank_you]);
```

practice session for priming

- what is the stimuli that we are using to run the `priming_proc`?
- so far, we've been using the `practice_stimuli`!
- where did we define `practice_stimuli`?

```
var priming_proc = {  
  timeline: [fixation, image, prime, target, priming_feedback],  
  timeline_variables: practice_stimuli,  
  randomize_order: true  
};
```

```
var practice_stimuli = [  
  {  
    "block_number": "practice",  
    "triad": 1,  
    "target_image_pair": "apple-horse",  
    "part": "priming",  
    "prime_word": "boff",  
    "target_word": "apple",  
    "type": "novel",  
    "relatedness": "novel",  
    "correct_response": 1,  
    "image_path": "applehorse.png",  
    "correct_key": "A"  
  },  
];
```

separating practice from test session

- locate `test_stimuli`
- replace `timeline_variables` inside `priming_proc` to `test_stimuli`
- create a copy of the `priming_proc` (call it `practice_procedure`) and replace the `timeline_variables` to `practice_stimuli`
- how do we make sure both practice and actual procedures are run??

```
var priming_proc = {  
  timeline: [fixation, image, prime, target, priming_feedback],  
  timeline_variables: test_stimuli,  
  randomize_order: true,  
  repetitions: 2  
};
```

```
var practice_proc = {  
  timeline: [fixation, image, prime, target, priming_feedback],  
  timeline_variables: practice_stimuli,  
  randomize_order: true  
};
```

```
jsPsych.run([initial_instructions, training_plus_association,  
            priming_instructions, practice_proc, priming_proc, thank_you]);
```

testing hack

- reduce all trial durations for priming to speed through
- **problem**: how can we make sure that we are able to separate practice trials from test trials?

```
{  
  "rt": null,  
  "stimulus": "applehorse.png",  
  "response": null,  
  "typeoftrial": "target",  
  "target": "horse",  
  "prime": "boff",  
  "type": "novel",  
  "relatedness": "novel",  
  "correct_key": "L",  
  "trial_type": "image-keyboard-response",  
  "trial_index": 62,  
  "time_elapsed": 15585,  
  "internal_node_id": "0.0-3.0-3.0",  
  "ID": 591591074,  
  "correct": false  
},
```

practice vs. test trials

- can we use the information inside `priming.js` to help us out?

```
    {  
      "block_number": "practice",  
      "triad": 1,  
      "target_image_pair": "horse-apple",  
      "part": "priming",  
      "prime_word": "nuppical",  
      "target_word": "apple",  
      "type": "novel",  
      "relatedness": "novel",  
      "correct_response": 3,  
      "image_path": "horseapple.png",  
      "correct_key": "L"  
    }  
  ]  
}
```

```
var test_stimuli = [  
  {  
    "block_number": 1,  
    "triad": 1,  
    "target_image_pair": "apple-horse",  
    "part": "priming",  
    "prime_word": "foobly",  
    "target_word": "apple",  
    "type": "direct",  
    "relatedness": "related",  
    "correct_response": 1,  
    "image_path": "applehorse.png",  
    "correct_key": "A"  
  },  
]
```


tagging practice vs. test trials

- add **block_number** to the data parameter of the target trial

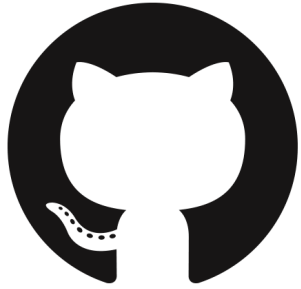
```
var target = {
  type: jsPsychImageKeyboardResponse,
  stimulus: jsPsych.timelineVariable('image_path'),
  choices: ['A', 'L'],
  stimulus_width: 500,
  maintain_aspect_ratio: true,
  trial_duration: 10,
  prompt: function(){
    return "<span style= 'font-size:200%'><br>" + String(jsPsych.timelineVariable('target_word')) + "<br></span>";
  },
  data: {
    typeoftrial: 'target',
    target: jsPsych.timelineVariable('target_word'),
    prime: jsPsych.timelineVariable('prime_word'),
    type: jsPsych.timelineVariable('type'),
    relatedness: jsPsych.timelineVariable('relatedness'),
    correct_key: jsPsych.timelineVariable('correct_key'),
    block_number: jsPsych.timelineVariable('block_number')
  },
  on_finish: function(data){
    data.correct = jsPsych.pluginAPI.compareKeys(data.response, data.correct_key);
  }
}
```

preloading images

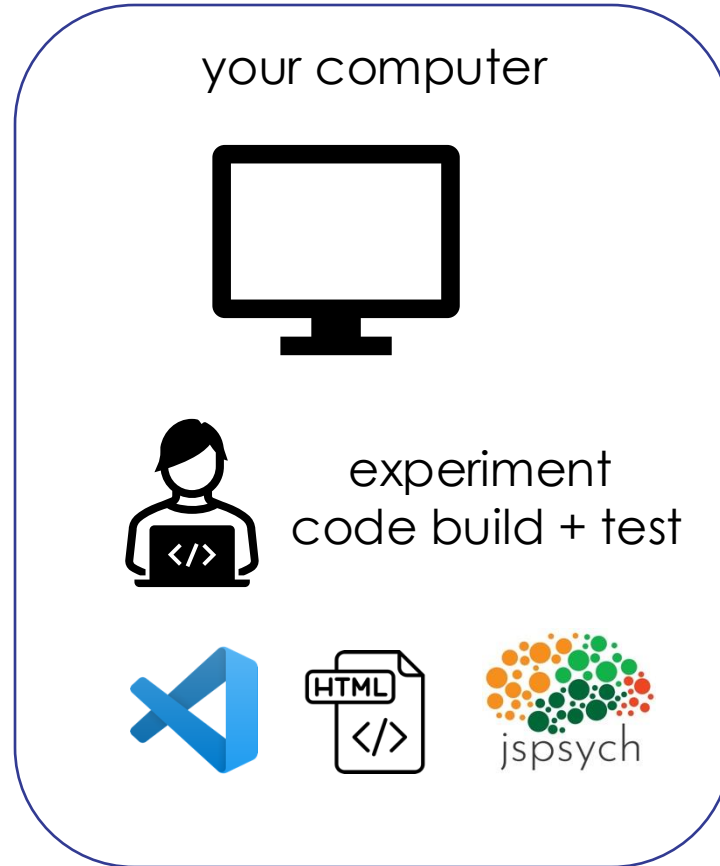
- reaction time tasks are sensitive to small fluctuations or delays
- to prevent delays in loading images, we can **preload** any images we will use in the experiment before the experiment starts
- which plugin?
- load inside <head> and define the preload trial
- add at the beginning of the run sequence

```
var preload = {  
  type: jsPsychPreload,  
  auto_preload: true,  
  images: ['horseapple.png', 'applehorse.png']  
}
```

```
jsPsych.run([preload, initial_instructions, training_plus_association,  
            priming_instructions, practice_proc, priming_proc, thank_you]);
```



github
keeping
track of
changes

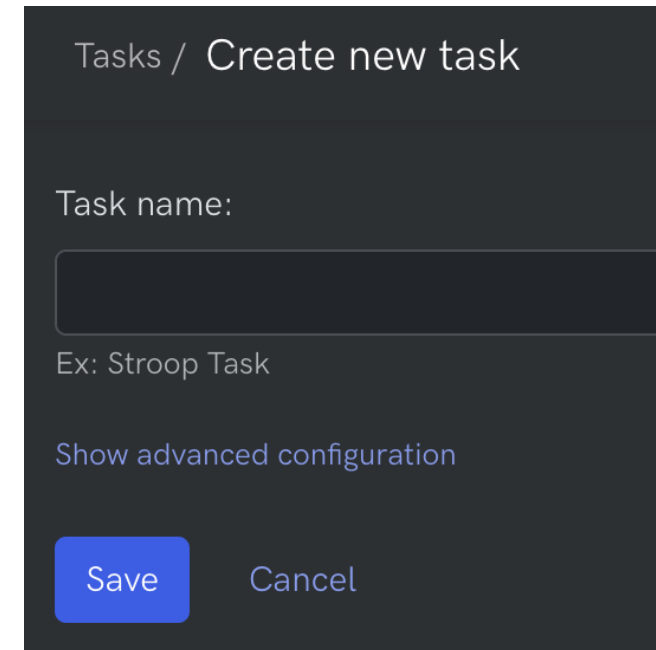
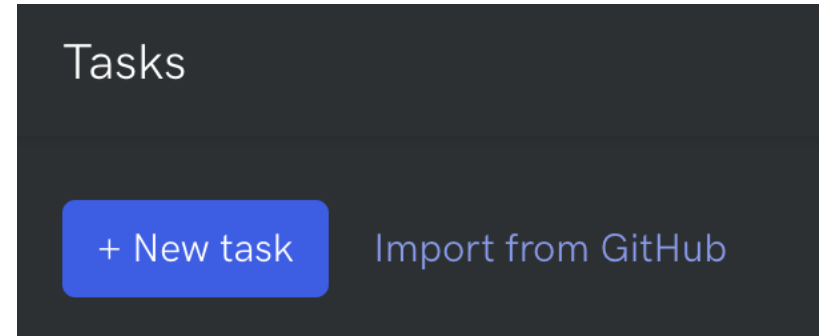


Cognition.

cognition.run
going
online

offline to online

- once we have tested our experiment locally, we can upload it to a site that can host our experiment on their servers
- go cognition.run, create an account and a new task
- give your experiment a name
- click on advanced configuration



advanced configuration

- if our design was using **multiple lists/conditions**, we could change that to our number of conditions
- disable/uncheck IP address tracking

Inter experiment conditions:

1

Used in between-group (or between-groups) study design. Auto participants to conditions to keep them all with equal N. The as

Task language:

English

Select the language of the task. This is meant to assist browsers

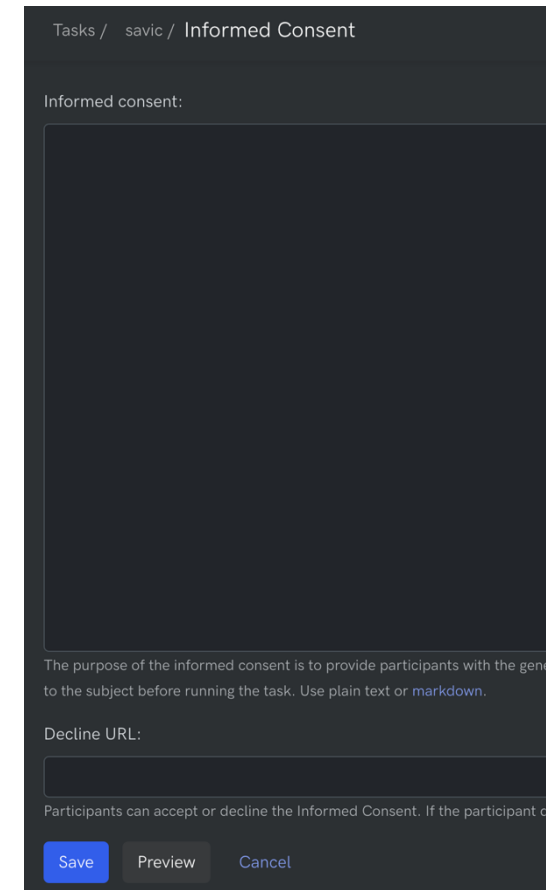
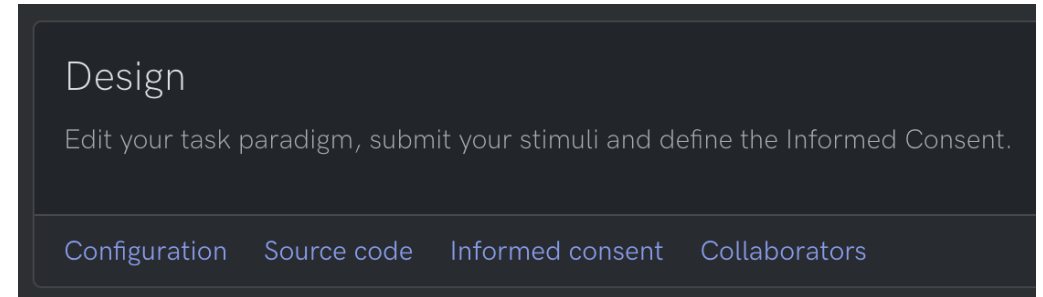
Email notifications:
Get email notifications for new responses.

Store participants IP Address:
Save or not save the IP Address of the participant.


Save Cancel


informed consent

- we can add a **consent form** to our task using the Informed consent option
- copy the text [here](#) and paste it inside the text box
- you could also style/format the text using Markdown
 - [cheatsheet](#)
- preview & save




source code

Tasks / demo / Edit Account 

jsPsych version 


jsPsych library version:

External JS/CSS 


[Choose Files](#) No file chosen
Upload files

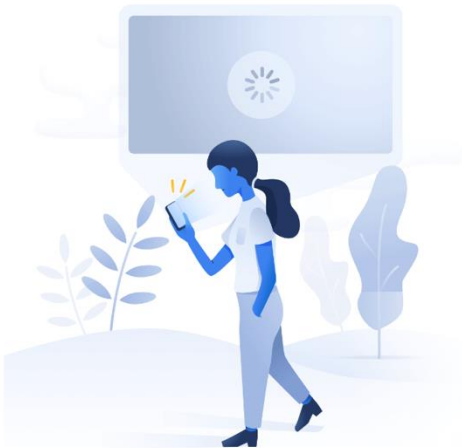
Stimuli

[Choose Files](#) No file chosen
Upload files

Task Code 

1 | Place your jsPsych Javascript code here


Task Preview 




The task is not ready yet.

Come back later.

[Disable preview](#) [Refresh](#) [Add url params](#) [Set condition](#)

Recorded data 

disable preview

Tasks / demo / Edit Account 

jsPsych version ? Task Code ? Task Preview ?

jsPsych library version:

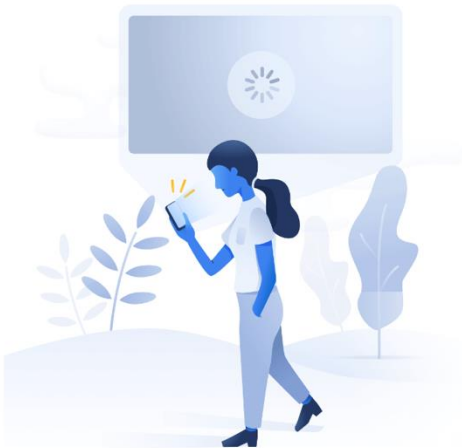
External JS/CSS ?

No file chosen
Upload files

Stimuli

No file chosen
Upload files

1 | Place your jsPsych Javascript code here



The task is not ready yet.

Come back later.

?

editing source/task code

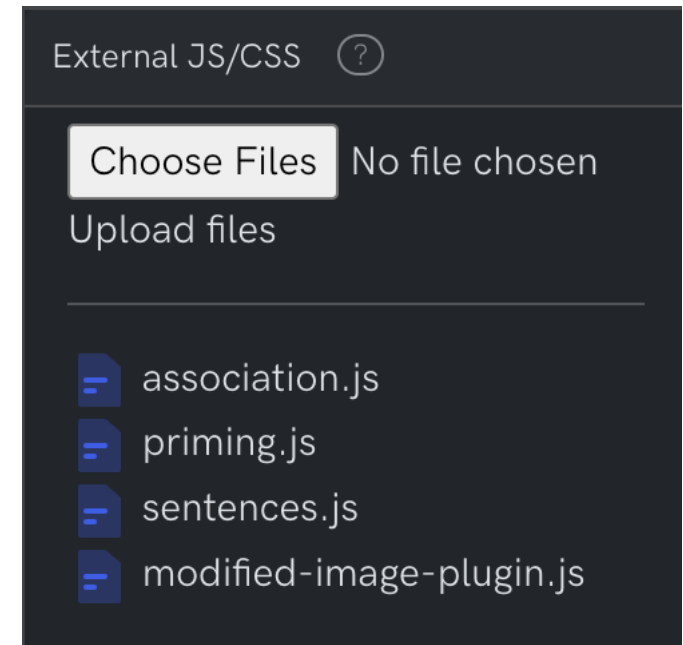
- copy all the code **inside the <script> tags** after <body> from your index.html file into the “task code” pane
- (from `const jsPsych` to `jsPsych.run`)

```
Task Code (?)
1  const jsPsych = initJsPsych(
2      {
3          on_finish: function(data) {
4              jsPsych.data.displayData();
5          }
6      }
7  );
8
9
```

```
jsPsych.run([preload, initial_instructions, training_plus_association,
             priming_instructions, practice_proc, priming_proc, thank_you]);
```

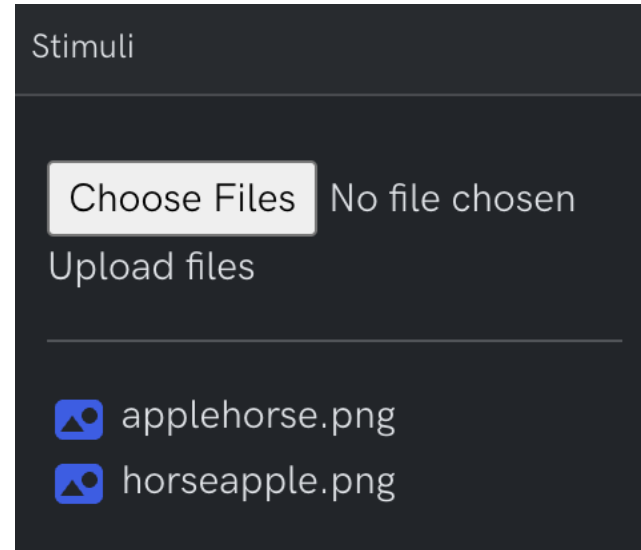
adding external JS/CSS

- upload all the .js files we have used/created:
 - modified_image_plugin.js
 - sentences.js
 - association.js
 - priming.js



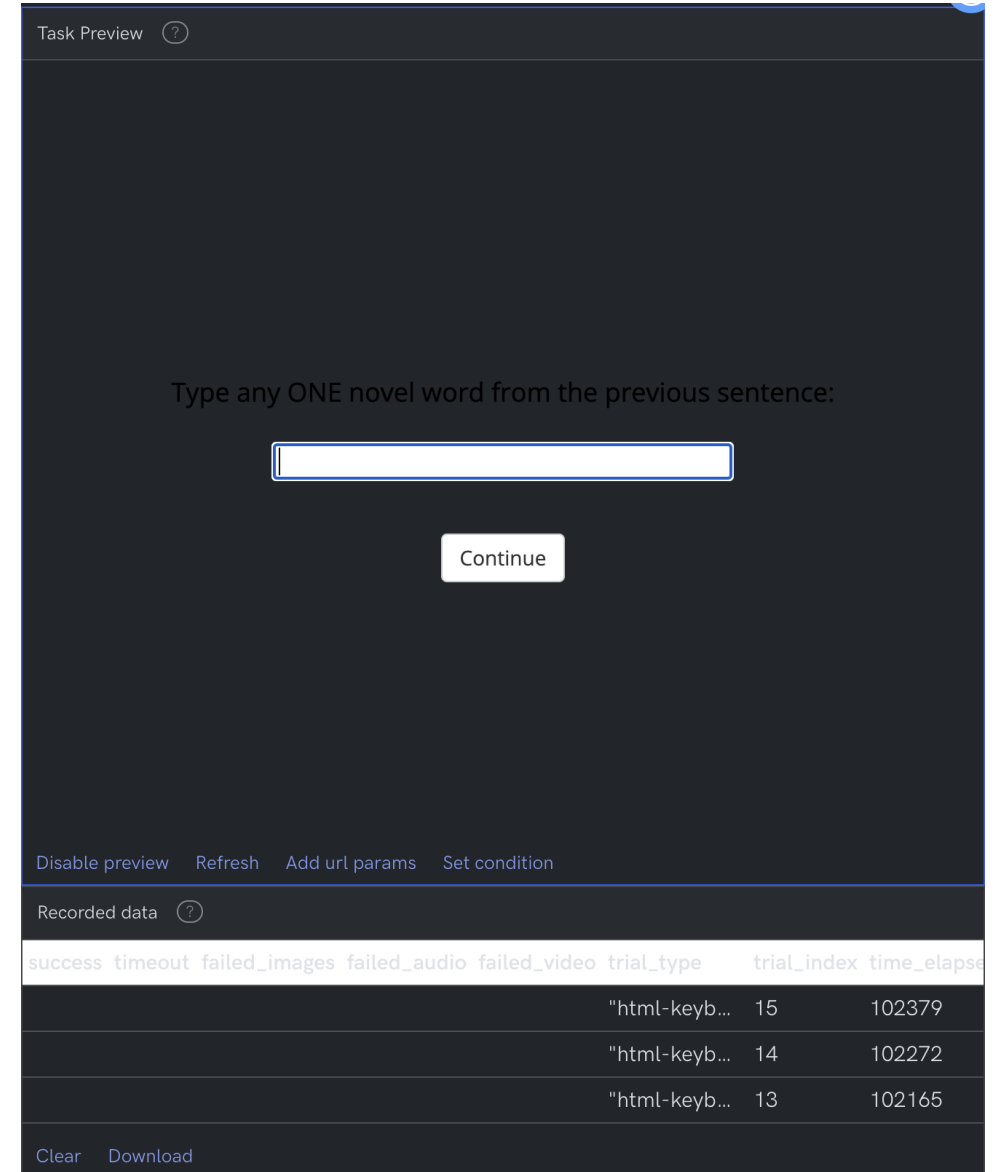
adding stimuli

- upload all images



enable preview

- your preview pane should be running the experiment
- you should also be able to **view the data** being generated from each trial



Task Preview ?

Type any ONE novel word from the previous sentence:

Continue

Disable preview Refresh Add url params Set condition

Recorded data ?

success	timeout	failed_images	failed_audio	failed_video	trial_type	trial_index	time_elapse
					"html-keyb...	15	102379
					"html-keyb...	14	102272
					"html-keyb...	13	102165

Clear Download

download the data

- run through some trials from the experiment
- use the [download](#) button to download and inspect the data from the task

Recorded data ?

success	timeout	failed_images	failed_audio	failed_video	trial_type	trial_index	time_elapsed
					"html-keyb...	15	102379
					"html-keyb...	14	102272
					"html-keyb...	13	102165

[Clear](#) [Download](#)

success	timeout	failed_image	failed_audio	failed_video	trial_type	trial_index	time_elapsed	internal_nod	run_id	condition
TRUE	FALSE	{}	{}	{}	preload	0	1074	0.0-0.0		1
					instructions	1	2642	0.0-1.0		1
					html-keyboa	2	2746	0.0-2.0-0.0-0		1
					html-keyboa	3	2850	0.0-2.0-0.0-0		1
					html-keyboa	4	2953	0.0-2.0-0.0-0		1
					html-keyboa	5	3060	0.0-2.0-0.0-0		1
					html-keyboa	6	3162	0.0-2.0-0.0-0		1
					html-keyboa	7	3265	0.0-2.0-0.0-0		1
					html-keyboa	8	3373	0.0-2.0-0.0-0		1
					html-keyboa	9	3476	0.0-2.0-0.0-0		1
					html-keyboa	10	3581	0.0-2.0-0.0-0		1
					html-keyboa	11	3684	0.0-2.0-0.0-0		1
					html-keyboa	12	3791	0.0-2.0-0.0-0		1
					html-keyboa	13	3898	0.0-2.0-0.0-0		1
				

homework 1: pilot

- make the experiment “[participant ready](#)”
- comment the `displayData` line from `initJsPsych()` using `//`
- fix all the trial durations
- provide real instructions (Savic et al. instructions [here](#))
- pilot the [whole task](#) yourself

homework 1: sanity checks

- is the attention check response being recorded?
- is the free association response being recorded?
- can you differentiate between training / attention / association / prime / target?
- can you differentiate between prime and target trials?
- can you differentiate practice and test trials?
- is subject ID being recorded?
- is RT being recorded?

homework 2: collect + inspect data

- go back to the task home page on cognition.run
- ask **5 friends/family** to take part in your experiment via the link
- you will be able to see their data appear
- download and inspect their data after they complete the task: download **a single CSV file**
- perform all sanity checks!

Tasks / demo / Edit

Link

Share this link with your participants.

<https://sw8vvsfswa.cognition.run>

Data collection

Manage data collected by runs.

There are no records to display. Once a participant visits the task's link, this is where you'll be able to see and download the data.

homework 3: demographics

- use the different plugins to add a demographic survey at the end of the experiment
- review questions to include
- you will need to :
 - decide which questions can go on the same screen vs. different screens
 - think about how to record the data

other nuts and bolts? / today's agenda

- adding a thank you screen
- adding a practice session
- preloading images
- going online!
- logistics

logistics: upcoming schedule

5	Tuesday, October 1, 2024	<u>W5: Recording Data</u>
5	Thursday, October 3, 2024	W5 continued...
6	Tuesday, October 8, 2024	Fall Break!! NO CLASS
6	Thursday, October 10, 2024	<u>W6: Experiment Workflow</u>
6	Sunday, October 13, 2024	Formative Assignment (jsPsych) Due
7	Tuesday, October 15, 2024	<u>W7: Visualize Data</u>
7	Thursday, October 17, 2024	W7 continued...
7	Sunday, Oct 20, 2024	Project Milestone #4 (Full Experiment) Due

resources on class website

Additional resources

Additional resources for PSYC 2740

Class-specific resources

1. [CogLab CheatSheet](#)
2. [jsPsych website](#)
3. [jsPsych online help community](#)
4. [jsPsych practice questions: Answers here](#)
5. [jsPsych debugging checklist](#)

logistics: formative assignment #1

- coding a **new experiment** from start to finish
- due October 13, but start early
- open-resource, but **no collaboration**
- goal is to push you to **code independently**
 - full credit for a reasonable first attempt on all questions (3%)
 - second attempt (after feedback) will be worth 10%

extra credit policies

Extra credit (5 points) [↗](#)

There will be some opportunities to earn extra credit during the semester. These opportunities are described below:

1. Complete class surveys (2 points): There will be 3 surveys (beginning, mid-semester, end of semester) to gather your reflections and suggestions to improve the course. With the exception of the pre-class survey (which is mandatory), all other surveys will be anonymous, and you will be able to earn 1 point for each survey you complete.
2. Win Star Coder (2 points): You will submit 3 formative coding assignments during the semester. The student who scores the combined highest score on the FIRST attempt for these assignments will earn 1 extra credit.
3. Win Team Player (1 point): Throughout the course, I will also evaluate who stood out as a team player, by observing how you participate in groups and contribute to group work. The student who stands out in this respect will earn 1 extra credit point.

logistics: project

- your next milestone is the **full experiment code** (Oct 20)
- feedback provided on **design draft**
- stop by office hours for more input / help
- class after fall break is devoted to project work + jsPsych questions

where are we going?



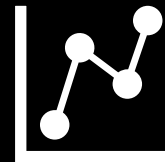
- literature review
- asking questions
- experiment creation
[HTML/jsPsych]

design



- R & Rstudio
- describe data
- infer from data

analyze



- pre-registration
- poster
- short report

communicate



next class

- **before** class
 - *prep*: formative milestone #1
 - *apply*: HW1, HW2, HW3
- **during** class
 - project discussion
 - jsPsych + formative milestone questions
 - conceptualizing an analysis workflow