Programming MTurk experiments via *JSEXP*

Florian Jaeger with help from Linda Liu, Zach Burchill, Wednesday Bushong, and Xin Xie

February 7, 2022

Contents

1 Overview **1**

2 Setup JSEXP **1**

3 Making your own experiment **2**
3.1 What needs to be in the HTML file? 2

4 Disabling CORS errors when testing locally **3**

5 Calling your own experiment **3**
5.1 Reserved URL parameters 3

6 Example experiments using *JSEXP* **4**

1 Overview

JSEXP is a collection of Javascript programs to facilitate psychological experiments over the web. It works both for MTurk and for Prolific (via Proliferate). JSEXP let's you define blocks of different tasks (e.g., lexical decision, picture naming, transcription, similarity judgments, etc. and records participants keyboard and mouse responses and reaction times. JSEXP was originally written by Dave Kleinschmidt and then extended and modified by Zach Burchill, Wednesday Bushong, Esteban Buz, Florian Jaeger, Linda Liu, Xin Xie, and others.

Make sure you have completed the tutorials on *Setting up Mechanical Turk* and *Conducting Mechanical Turk experiments*. This tutorial walks you through how to program your own experiment using *JSEXP*.

2 Setup JSEXP

Clone JSEXP. We recommend using git submodules to add JSEXP to your project repository. For example, in your git repo for our projects we typically have a folder called experiments/ with subfolders for each series of experiments, e.g., experiments/series-A/. Consider cloning JSEXP into each of those subfolders.

This will allow you to pull/push the most recent updates of this package while continuing to develop your experiment-specific code. When you clone a project that contains a submodule, remember that you need to also clone the submodule (by default the submodule folder will be cloned but not its content). Use:

```
git clone --recurse-submodules YOUR-PROJECT-URL
```

during the cloning to also automatically clone all submodules contained in the project. Once cloned, you can update all submodules in your project by using

```
git submodule update --remote
```

Once updates to a submodule are committed and pushed to your project, you and your collaborators can pull these updates. Make sure to use 'git pull –recurse-submodules' if you want changes to the submodules not only to be fetched but also to be pulled.

3 Making your own experiment

A JSEXP experiment consist of:

- An HTML file that loads all the necessary Javascript (inc. JSEXP)
- Your experiment-specific Javascript that uses JSEXP functionality.
- the JSEXP/ repo
- Optionally:
  - stimulus files
  - list files
  - style (css) files
  - survey files (for post-experiment surveys)

3.1 What needs to be in the HTML file?

- A header with an optional link to a style sheet. In case you plan to run your experiment over Prolific (via Proliferate), the header also needs to load the proliferate javascript.

```
 <head>
   <meta charset="UTF-8" />
   <link rel="stylesheet" type="text/css" href="JSEXP/styles/style.css" />
   <title>Speech perception experiment</title>
   <script src="https://proliferate.alps.science/static/js/proliferate.js"
   ↪  type="text/javascript"></script>
</head>
```

- A form element that posts the results to MTurk when the experiment is done. JSEXP will populate this form with hidden form elements based on a) the URL parameters you provide and b) data collected from the participants.

```
<form id="mturk_form" method="POST"
↪  action="https://www.mturk.com/mturk/externalSubmit">
    <input id="submitButton" type="submit" name="Submit" value="Submit" />
</form>
```

- A link to whatever Javascript code you're using. See the example experiments provides as part of this repo.

```
<!-- general javascript, incl. general definitions of objects
     for experiment, block, and stimulus lists  //-->
<script src="js-adapt/jquery-1.10.1.min.js" type="text/javascript" ></script>
<script src="js-adapt/modernizr.min.js" type="text/javascript"></script>
<script src="js-adapt/stimuli.js" type="text/javascript" ></script>
<script src="js-adapt/labelingBlock.js" type="text/javascript" ></script>
<script src="js-adapt/exposureBlock.js" type="text/javascript" ></script>
<script src="js-adapt/experimentControl2.js" type="text/javascript" ></script>
<script src="js-adapt/soundcheckBlock.js" type="text/javascript"></script>
<script src="js-adapt/mturk_helpers.js" type="text/javascript"></script>
<script src="js-adapt/progressBar.js" type="text/javascript"></script>
<script src="js-adapt/logreg.js"  type="text/javascript"></script>
<script src="js-adapt/utilities.js" type="text/javascript"></script>
<script src="get_stim.js" type="text/javascript"></script>

<!-- Here is where your experiment javascript file is specified -->
<script src="experiment-A.js" type="text/javascript"></script>
```

- Any HTML objects references in the Javascript code. This includes, for example, various ¡div¿s that the Javascript code will fill with content at different points during the experiment.

4 Disabling CORS errors when testing locally

If you Javascript is loading external sources like stimulus files or surveys, local testing will result in a CORS error (unless you're running a web server). You can avoid CORS Errors on localhost (in 2020) by starting Chrome the following way (it's important that the last argument contains the full path to the page you'd like to open). Note, however, that **this will not actually make Javascript *load* those external resources.** It seems to only suppress the error.

```
open -n -a "/Applications/Google Chrome.app/Contents/MacOS/Google Chrome" --args
↪  --user-data-dir="/tmp/chrome_dev_test" --disable-web-security
↪  /Users/tiflo/Documents/GitHub/Project1/experiments/series-A/experiment-A.html
```

5 Calling your own experiment

JSEXP reads in the URL parameters in the *search string* of the URL handed to the browser. This makes it possible to have one HTML file for many different conditions of your experiment. As soon as the URL parameters are read in, JSEXP removes them from the URL search string (except for the experiment_id and participant_id required for Prolific experiments through Proliferate). This avoids that the URL parameters that might, e.g., specify conditions affect participants' behavior.

5.1 Reserved URL parameters

The following is a list of reserved URL parameters that have a fixed interpretation in JSEXP. The values in curly parenthesis list allowed values.

- platform={mturk, prolific, proliferate} Determines the submit method chosen at the end of the experiment. For Prolific and Proliferate, the output is converted into a JSON object that is submitted to Prolific. For Mechanical Turk, the mturk_form in the HTML file is submitted as is. (DEFAULT: mturk)

- debug={T,TRUE}. Enter debug mode. This activates all messages, warnings, and errors that JSEXP would otherwise hide, and prints them to the Javascript console of the browser. The debug mode also suppressed the removal of the URL parameters from the URL search string, so that the experimenter can check whether those parameters look correct. Finally, the debug mode causes the script to pause (through a pop-up alert) right before the results are submitted. This allows the experimenter to read exactly what information would be submitted to the crowdsourcing platform. (DEFAULT: no debugging)

6 Example experiments using *JSEXP*

- The code for the unsupervised and supervised learning experiment in Kleinschmidt et. al (2015) [1] can be cloned from https://bitbucket.org/hlplab/nrtmodule/src/master/. If you'd like to get a sense of a full version of the experiment, you can test it out at https://www.hlp.rochester.edu/mturk/mtadapt/sup-unsup/.
- The code for a web-based version of the priming experiment from Xie & Myers (2017) can be found at https://github.com/xinxie-cogsci/Online-experiments. If you'd like to get a sense of a full version of the experiment, you can test it out at, for example, https://www.hlp.rochester.edu/mturk/xxie/io_perception/exp1/exp1_transcription.html?condition=experimental&speaker=M15&order=2&visual=1&block=2 (for information on the URL parameters, see the readme.md file in the github repository).
- And here is a demo of a whole number of different paradigms available on request: https://www.hlp.rochester.edu/mturk/lliu/demos/.

[1] https://mindmodeling.org/cogsci2015/papers/0200/paper0200.pdf