RECOGNITION MEMORY EXPERIMENT FRAMEWORK

DESIGNERS:
M. RABE MMRABE@UVIC.CA
DR. S. LINDSAY SLINDSAY@UVIC.CA

DEVELOPER:
A. RICHARDSON
RICHARDSON.ASHLIN@GMAIL.COM

INSTITUTION:
UNIVERSITY OF VICTORIA

Contents

Overview 3
0.1. Audience 3
0.2. Description 3
0.3. Requirements for using the software 3
0.3.1. Server-side 3
0.3.2. Client-side 3
1. **The System** 4
1.1. Installation 4
1.2. Project Structure 4
2. **Setup** 5
2.1. File Permissions on Server 5
2.2. Accessing an Existing Survey 5
2.3. Creating A New Survey or Modifying An Existing One 5
3. **The Examples** 6
3.1. experiments/instructions/my-experiment.js 7
3.1.1. Instructions Statements 7
3.1.2. Instructions: Fixed Duration 7
3.1.3. Instructions: Fixed Duration or User Intervention 7
3.1.4. The file my-experiment.js 7
3.2. experiments/delay/my-experiment.js 8
3.2.1. Delay Task 8
3.2.2. Delay Task: Fixed Interval 8
3.2.3. The file my-experiment.js 8
3.3. experiments/feedback/my-experiment.js 9
3.3.1. The file my-experiment.js 9
3.4. experiments/study-phase/my-experiment.js 9
3.4.1. The file my-experiment.js 10

Date: May 23, 2017.

3.5. experiments/test-phase/my-experiment.js 10
3.5.1. The file my-experiment.js 11
3.6. experiments/my-experiment/my-experiment.js 12
3.6.1. Study/Test Phase: Declaring Multiple Stimuli Pools 12
3.6.2. Study/Test Phase: Using Multiple Stimuli Pools to Declare Study and Test Phases 12
3.6.3. Inserting a Delay Task between Study/Test Phases 12
3.6.4. Modified Instructions Slide between Study/Test Phases 12
3.6.5. The file my-experiment.js 12
4. **Sample Response Data** 14
4.1. instructions 14
4.2. delay 14
4.3. study-phase 15
4.4. test-phase 15
4.5. my-experiment 17
5. **Source Code: Client Side** 20
5.1. egg-timer.js 20
5.2. key.js 21
5.3. main.js 24
5.4. memory.js 27
5.5. pool.js 28
5.6. state.js 31
5.7. task.js 36
5.8. text.js 40
5.9. util.js 41
6. **Source Code: Server Side** 43
6.1. xml-receive.py 43
7. **Ideas For Possible Future Improvements** 44

Overview

0.1. **Audience.** The intended audience for this document is a researcher with some programming experience.

0.2. **Description.** The **Recognition Memory Experiment Framework** is an online utility for parametric generation of **Recognition Memory experiments** to support researchers at the University of Victoria. The software is intended to be web based, self contained yet comprehensive, and reasonably flexible.

The software is a foundation upon which to develop and deploy interactive surveys/questionnaires as an essential component of Recognition Memory experiment methodologies, and is comprised of two aspects:

(1) The **researcher facing** portion, which is a simplified API-style programming interface, specifies the sequencing of both:

- online interactive visual elements, and
- requests to the participant for feedback.

(2) The **participant facing** portion consists of

- an experimental survey (based on JavaScript/HTML5) which runs client-side in the participant's web browser, and is specified by the researcher in terms of:
  - the different stimuli or other interactive visual elements revealed to the participant, and
  - the possible responses/feedback requested of the partcipant, and
- a server-side program (written in Python) running on the web server administrated by the researcher, that receives user responses, timing, and other information sent back to the researcher by the JavaScript/HTML5 program.

0.3. **Requirements for using the software.**

0.3.1. *Server-side.*

- Host:
  - An ordinary web server with Python/CGI enabled, is required.
  - Note: the system was tested with server: Apache/2.2.23 (Unix).

0.3.2. *Client-side.*

- For experiment participants:
  - A modern web browser (Firefox, Google Chrome, or Safari) on a desktop computer is required.
    - The system was tested with Chrome v. 57, Safari 10.1, and Firefox 53.0 (64 bit).
- For researchers and experiment administrators:
  - A text editor is required to edit experiment script files.
  - Limited technical knowledge about JavaScript is required to edit or modify experiments.
  - An FTP program is required for uploading experiment scripts (and downloading response data).
    - A free/open implementation is FileZilla, available at https://filezilla-project.org/

1. The System

1.1. **Installation.** The system may be installed on a web server by downloading:

- https://github.com/ashlinrichardson/m3m0ry/archive/master.zip

and extracting the ZIP file to an administrator-determined folder/location on the server. It's important to check the permissions on the files (please see section 2.1). Note: the University of Victoria Unix web server does not require **.htaccess** files. If using the software on another server, an **.htaccess** file may be required. A typical **.htaccess** file may contain the following:

Options +ExecCGI
AddHandler cgi-script .py

and is usually placed in the same folder as the python files that should be run (in this case the only Python file is **xml_receive.py**).

1.2. **Project Structure.** The system has the following directory structure as in Fig. (1.1).

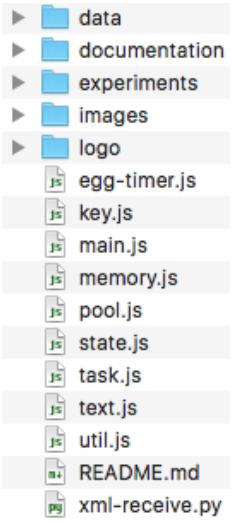

Figure 1.1.

In Fig. (1.1) above:

- **data/** should, once survey(s) have been successfully completed, contain CSV data files representing the user experience:
  - If all goes well, an additional data file should automagically appear in the **data/** folder, for any given survey/experiment that is successfully completed.
  - Upon completion of a survey/experiment, the client-side JavaScript code submits (via util.js::xml_send()) a CSV data file to the web server, which receives the data using CGI/Python (via xml-receive.py).
  - The CSV file is saved with a name reflecting the date/time when the file was recorded, and also a randomly-generated string that is added to prevent file-naming "collisions".
- **documentation/** is where this manual resides.
- **experiments/**
  - Contains a number of sub-folders, one for each of the included examples:
    - **delay/**
    - **feedback/**
    - **instructions/**

    * **study-phase**/
    * **test-phase**/
    * **my-experiment**/

  – Each of the above subfolder contains a file **memory.html**, which always has the contents:

```
6 lines (5 sloc)  70 Bytes
1  <html>
2  <body>
3  <script src="../../memory.js"></script>
4  </body>
5  </html>
```

FIGURE 1.2.

Note: any experiment/survey project developed by the user must also:
  * reside in the **experiments**/ folder (as with the examples provided), and:
  * include a **memory.html** file, which should be the same as in Fig. (1.2).

- **images**/ contains image data used in experiments. To change image data used in experiments, the administrator should:

  – upload new image data into the **images**/ folder, ensuring that the image data is consistently named according to the same numbered format followed by the provided image data: **1.jpg, 2.jpg,** and so on.

2. SETUP

2.1. **File Permissions on Server.** Unix file permissions on the server side should all be set to 755, with the exception of the data folder for which the permissions should be set to 754, to protect submitted data from global visibility.

Unix file permissions may be set recursively using an FTP program such as FileZilla.

N.b., also want to set the permissions on the **experiments**/ folder to 754, to prevent any given survey being accessed by someone who was not given the exact path to the specific survey.

2.2. **Accessing an Existing Survey.** Supposing the project is uploaded to the main HTTP directory of a web server with URL **http://my-web-server.com**/, the survey in the folder **experiments/my-experiment**/ as represented by the JavaScript file **experiments/my-experiment/my-experiment.js** administered by the researcher/administrator, will be accessed by navigating in a web broswer to the following address:

- **http://my-web-server.com/experiments/my-experiment/memory.html**

Of course, for any given experiment, a researcher/administrator is advised to thoroughly test the survey, in a web browser, by accessing a URL like the one above.

2.3. **Creating A New Survey or Modifying An Existing One.** To create your own experiment we recommend, with respect to the **experiments**/ folder on the local computer (which will later be uploaded to the web server e.g., via FTP):

(1) making a copy (with a different name) of
  - the folder corresponding to the given example provided which is
    – most closely representative of the experiment which one would like to create.

      * For example, if we wanted to make simple modifications to the instructions-type example, we could make a copy of the **experiments/instructions**/ folder, and rename that folder **experiment/test-experiment**/

    * If we wanted to develop a more detailed and realistic complete experiment, we might make a copy of the experiment folder **experiment/my-experiment/**, and rename the copied folder to e.g.: **experiment/test-experiment/**

(2) given the folder we created, e..g, **experiment/test-experiment/,** we should

- edit the file **my-experiment.js** within the **my-experiment/** folder, for example:
  - we could change the messages on any instructions slides;
  - adjust timing parameters (e.g. ISI: Inter-Stimulus-Interval) for certain components;
  - add additional word/image stimuli to an experiment (study/test phase components) with respect to one or more "stimulus pools";
  - add further tasks to the experiment (more study/test phases, instructions, delay activities, or extra feedback questions).

(3) **To deploy such an experiment on the web,** please make sure to upload your revised folder on the local computer (e.g., **experiment/test-experiment/**) to the web server.

- Please ensure that the revised folder is uploaded in the correct location (i.e., within the **experiment/** relative to the main project folder, as it appears on the web server), and that the revised folder includes both a **my-experiment.js** file and a **memory.html** file.

3. The Examples

Five example surveys are included:

(1) **instructions/**: involving display of instructions text or other directions/information;

(2) **delay/**: involving presenting the user with free-form or other response fields, to occupy the participant between different experiment/survey components;

(3) **feedback/**: involving presenting the user with various (multiple-choice) opportunities to reply with feedback/information;

(4) **study-phase/**: involving presenting the user with a variety of stimuli (usually accompanied by instructions that indicate to try to remember the stimuli presented);

(5) **test-phase/**: involving requesting feedback from the user, with respect to a sequence of stimuli information;

(6) **my-experiment/**: a more-detailed example, representing a typical possible experiment.

3.1. **experiments/instructions/my-experiment.js.** The file my-experiment.js below exemplifies the required format of a **my-experiment.js** file, which must contain a function called my_experiment, as indicated in line 2:

var my_experiment = function(){

Please note the closing bracket for that function, on line 29. For those not familiar with JavaScript code, the text which appears within the marks: /* */ is a "comment", e.g., /* this is a comment: text within a program that does not represent instructions to be executed */

3.1.1. *Instructions Statements.* Note that, e.g. in line 5:

instructions('welcome to the recognition memory experiment framework\n\n\n\t* please press any key to continue')

this indicates presenting the text within the single quotation-marks, to the user. Note: "/n" and "/t" represent "control characters" which affect the flow of text on the screen: "/n" is the newline-character which causes the rendering of text to proceed on the next available line, and "\t" is the tab-character, which indents text several spaces. The researcher can insert as many such statements as they like. Note some possible modifications to the "vanilla" instructions statement above, as follows:

3.1.2. *Instructions: Fixed Duration.* For displaying instructions for a fixed interval (in milliSeconds) we can insert a code block as follows, as in lines 17-19:

var x = instructions('this information will be displayed for 5 seconds')
x.set_expiry(5000)
x.key_expiry = false

3.1.3. *Instructions: Fixed Duration or User Intervention.* For instructions that are shown until a key is pressed, but are shown for (at most) a given fixed interval (in milliSeconds) we can insert a code block as follows, as in lines 22-24:

var x = instructions('this information will be displayed for 5 seconds')
x.set_expiry(5000)
x.key_expiry = true

3.1.4. *The file my-experiment.js.*

```
1  /* recognition memory experiment set-up */
2  var my_experiment = function(){
3
4    /* instruction slide */
5    instructions('welcome to the recognition memory experiment framework\n\n\n\t* please press any key to
         continue')
6
7    /* instruction slide */
8    instructions('here is what happens when you put in a lot of text— if you put in lots of text, it
         might wrap around the edge\n\n\n\t* please press any key to continue')
9
10   /* instruction slide */
11   instructions('this is an instructions slide\n\n\n\t* please press any key to continue')
12
13   /* instruction slide */
14   instructions('this is an instructions slide with extra line breaks:\nsingle line break:\ndouble line
         break:\n\ntriple line break:\n\n\n\t* please press any key to continue')
15
16   /* instruction slide -- fixed duration */
17   var x = instructions('this instructions slide will display for 5 seconds:\n\n\n\t* if you press a key
         , it will do nothing')
18   x.set_expiry(5000)
19   x.key_expiry = false
20
21   /* instruction slide -- fixed duration or user intervention */
22   var y = instructions('this instructions slide will display for up to 5 seconds:\n\n\n\t* if you press
         a key, the transition will happen before 5 seconds is up')
23   y.set_expiry(5000)
24   y.key_expiry = true
```

```
25
26   /* instruction slide */
27   instructions('this is a normal instructions slide\n\n\n\t* please press any key to continue')
28
29 }
```

3.2. **experiments/delay/my-experiment.js.** Although other manifestations could be possible for the delay task, the current implementation involves a free-form reponse section to involve the participant in an activity other than observing stimuli (an interval-type activity to be deployed between other experimental components, e.g., between a study-phase component and the subsequent test-phase component).

3.2.1. *Delay Task.* The basic syntax for specifying a "delay" task is like that for the "instructions" task:

> **delay('please enter the names of as many countries as you can think of, followed by the escape key\n\n\n\t* please press any key to continue')**

By default, the delay task will continue collecting user input, until the <escape> key is pressed.

3.2.2. *Delay Task: Fixed Interval.* It's also possible to specify a "delay" task that doesn't wait for an <escape> key to be pressed: this version waits for a specified interval (in mS) instead. To operate the delay with a fixed interval, simply add a time parameter (in mS) to the invocation:

> **delay('please enter the names of as many countries as you can think of, in 5 seconds\n\n\n\t* please press any key to continue', 5000)**

where 5000 mS is five seconds.

3.2.3. *The file my-experiment.js.*

```
1  /* recognition memory experiment set-up */
2  var my_experiment = function(){
3
4    instructions('first delay phase coming up (please press any key to continue)')
5
6    delay_task('please write out anything that comes to mind (please press <esc> key when finished)')
7
8    /* instruction slide */
9    instructions('second delay phase (5 seconds): (please press any key to continue)')
10
11   /* set up delay task: 5 seconds */
12   delay_task('please type names of as many countries as you can think of in 5 seconds, separated by
         spaces.. (please press any key to continue)',
13             5000 /* 5000 mS */)
14
15   /* instruction slide */
16   /* instruction slide -- fixed duration */
17   var x = instructions('thank you for completing the delay task: test phase coming up in 5 seconds..')
18   x.set_expiry(5000)
19   x.key_expiry = false
20
21   instructions('third delay phase (6 seconds): (please press any key to continue)')
22
23   /* set up delay task: 6 seconds */
24   delay_task('please type names of as many countries as you can think of in 6 seconds, separated by
         spaces.. (please press any key to continue)',
25             6000 /* 10000 mS */)
26
27   /* instruction slide */
28   instructions('all done.. thank you.. (please press any key to continue)')
29 }
```

3.3. **experiments/feedback/my-experiment.js.** To collect user feedback in a multiple-choice format, we use the "feedback" command, which takes two parameters: 1) the text to be supplied to the user, and 2) an array of key-codes for the permitted responses.

- E.g.,
  - the array [49, 50, 51, 52, 53] represents the numeric keys 1-5
  - [49, 50, 51, 52, 53, 54, 55, 56, 57, 48] represents the numeric keys 0-9
  - [65, 66, 67, 68] represents the letters A,B,C,D

- and of course, other key combinations are possible. It may be necessary to consult a reference table such as
  - https://en.wikipedia.org/wiki/List_of_Unicode_characters#Basic_Latin

Please see the below file **my-experiment.js** for examples.

3.3.1. *The file my-experiment.js.*

```
1  /* recognition memory experiment set-up */
2  var my_experiment = function(){
3
4    /* instructions */
5    instructions('feedback coming up.. (please press any key to continue)')
6
7    /* feedback "task" */
8    feedback('please enter your affinity with the last stimulus on a scale of 1-5',
9            [49, 50, 51, 52, 53])
10
11   /* instructions */
12   instructions('thank you... more feedback coming up.. (please press any key to continue)')
13
14   /* more feedback "task" */
15   feedback('please enter your affinity with the last stimulus on a scale of 0-9',
16            [49, 50, 51, 52, 53, 54, 55, 56, 57, 48])
17
18   /* instructions */
19   instructions('thank you... multiple choice style feedback coming up.. (please press any key to
         continue)')
20
21   /* feedback "task" */
22   feedback('skill testing question: 10*10 is: a) 100 b) 200 c) 1000 d) 10000',
23            [65, 66, 67, 68])
24
25   /* instructions */
26   instructions('thank you.. (please press any key to continue)')
27 }
```

3.4. **experiments/study-phase/my-experiment.js.** The study phase is the part of an experiment/survey where word/image (or other) stimuli are revealed in sequence. To implement a "study-phase", we must:

(1) declare a stimulus pool (e.g., line 9 of the file below)

(2) optionally, add a number of images to the stimulus pool (e.g., line 12 of the file below)

(3) optionally, add a number of words to the stimulus pool (e.g., lines 15-17 of the file below)

(4) make a selection of items from the stimulus pool (e.g., line 20 of the file below):

- note that only one parameter is supplied on line 20 (the parameter "N" from the spec), although it will be necessary for a "M" parameter to be included in subsequent examples, in order to implement the "test-phase". This will be discussed re: the next example.

(5) declare the study phase:

- note that declaring the study phase is as simple as (where "p" is a stimulus pool, previously defined in the **my-experiment.js** file):

  study_phase(p)

  although, in line 23 of the file below, two extra parameters are added:

  - an ISI (which was set to be 111 mS in the example): an (optional) duration of exposure of nothing, between exposure of stimuli, and

  - a SET (stimulus expiry time) which represents an (optional) maximum duration of exposure of the stimulus. Note: if one additional parameter is included in hte "study_phase" statement, this will be interpreted as ISI. In order to declare a "study_phase" with a SET (but not an ISI), one should provide two extra parameters (after "p") where the first parameter (the ISI) is set as 0, e.g.
    study_phase(p, 0, 5000)

    where the parameters above represent a "study_phase" with ISI of 0 mS, and a SET of 5 seconds.

3.4.1. *The file my-experiment.js.*

```
1  /* recognition memory experiment set-up */
2  var my_experiment = function(){
3
4    /* instructions */
5    instructions('study phase coming next: (please press any key to continue)')
6    instructions('please remember each word/image shown\n\n\neach word/image is displayed for up to 5
         seconds:\n\n\nif you are done with a particular word/image in less than 5 seconds, please press
         any key to advance to the next word/image\n\n\n(please press any key to continue)')
7
8    /* set up a stimulus pool */
9    var p = stimulus_pool()
10
11   /* add images to stimulus pool */
12   p.add_image(10)
13
14   /* add words to stimulus pool */
15   p.add('floccinaucinihilipilification')
16   p.add('supercalifragilisticexpialidocious')
17   p.add('umdiddlediddlediddleumdiddlei')
18
19   /* select portion of items from stimulus pool */
20   p.select(5)
21
22   /* set up 'study phase': show selected portions of pool */
23   study_phase(p,  /* stimulus pool */
24               111 /* ISI (optional) */,
25               5000  /* SET (optional) */ )
26 }
```

3.5. **experiments/test-phase/my-experiment.js.** For the "test-phase" example, we don't introduce anything new in lines 1-16.

However, on line 19, note that the second "M" parameter is added: calling the function **p.select()** with two parameters, as in line 19, is required before implementing the "study-phase" and "test-phase" in lines 22 and 28, respectively.

Note that, to declare a "test_phase" as on line 28 below, an optional ISI parameter has been included. The ISI parameter is optional, so the simplest invocation is:

test_phase(p)

Similarly to declaring a "study-phase" in the previous example, a SET parameter may also be added when declaring a "test-phase" (again, if one wishes to declare a "test_phase" without ISI, and with SET > 0, two parameters need to be included: the first being ISI, which should be 0). So, while

test_phase(p)

is the simplest invocation,

test_phase(p, 111)

adds an ISI of 111 mS, and

test_phase(p, 111, 5000)

adds a SET of 5 seconds. Of course,

test_phase(p, 0, 5000)

would add a SET of 5 seconds, without adding ISI.

3.5.1. *The file my-experiment.js.*

```
/* recognition memory experiment set-up */
var my_experiment = function(){

  /* set up some instruction slides */
  instructions('study phase: please remember words/images and,\n\n\n\t* please press any key to advance
       to the next word/image\n\n\n(please press any key to continue)')

  /* set up a stimulus pool */
  var p = stimulus_pool()

  /* add 10 available images to stimulus pool */
  p.add_image(10)

  /* add words to stimulus pool */
  p.add('floccinaucinihilipilification')
  p.add('supercalifragilisticexpialidocious')
  p.add('umdiddlediddlediddleumdiddlei')

  /* selection from stimulus pool: parameters are N, M as per the requirements */
  p.select(5, 5)

  /* set up 'study phase': show selected portions of pool */
  study_phase(p, 111 /* ISI of 111 mS */ )

  /* some instructions before 'test phase' */
  instructions('test phase coming up:\n\n\nwhen you see an image/word, please press m or n:\n\n\n\t*
      please press m if you saw an image/word before\n\n\n\t* please press n if you did not see the
      image/word before\n\n\n(please press any key to continue)')

  /* set up 'test phase' (user input recorded for whole randomized pool) */
  test_phase(p, 333 /* ISI of 333 mS */ )
}
```

3.6. **experiments/my-experiment/my-experiment.js.** This more-substantial example is more representative of an "actual" experiment, than the previous examples.

3.6.1. *Study/Test Phase: Declaring Multiple Stimuli Pools.* In this example, the procedure for adding stimulus to a pool is diversified by the specification of multiple stimulus pools, as in lines 8-16 for a first pool, and lines 19-27 for a second pool.

Note: the required selection from each of the two stimulus pools, happens at lines 30 and 31, respectively.

3.6.2. *Study/Test Phase: Using Multiple Stimuli Pools to Declare Study and Test Phases.* Whereas before when we passed a stimuli pool, when declaring a study_phase or test_phase, e.g. cf the last examples:

```
study_phase(p)
  test_phase(p);
```

when combining multiple stimuli pools together, we combine the stimuli pools together in an array before passing them, e.g.:

```
var pools = [p1, p2]
  study_phase(pools)
  test_phase(pools)
```

cf lines 34, 37 and 60 in the file **my-experiment.js** below.

3.6.3. *Inserting a Delay Task between Study/Test Phases.* An important difference between this example, and the earlier **experiments/test-phase/** example, is: note the delay task inserted between the study and test phases cf line 45 of **my-experiment.js** below.

3.6.4. *Modified Instructions Slide between Study/Test Phases.* Note on lines 49-51 the modified instructions slide (fixed duration) borrowed from the **experiments/instructions/** example.

This message that displays for the fixed interval of five seconds, was added to prevent the user from over-running the testing phase, with key presses (we added this interval during testing when we noticed that frantic data entry during the "delay" task could result in keypresses that would cause hiding the instructions for the testing phase).

3.6.5. *The file my-experiment.js.*

```
1  /* recognition memory experiment set-up: customized/ complex experiment */
2  var my_experiment = function(){
3
4    /* set up some instruction slides */
5    instructions('study phase: please remember words/images and press any key (please press any key to
         continue)')
6
7    /* set up a stimulus pool */
8    var p1 = stimulus_pool()
9
10   /* add images to stimulus pool */
11   p1.add_image(10)
12
13   /* add words to stimulus pool */
14   p1.add('floccinaucinihilipilification')
15   p1.add('supercalifragilisticexpialidocious')
16   p1.add('equanimity')
17
18   /* set up a stimulus pool */
19   var p2 = stimulus_pool()
20
21   /* add images to stimulus pool */
22   p2.add_image(10)
23
24   /* add words to second stimulus pool */
25   p2.add('compassion')
26   p2.add('dogovarivatsya')
27   p2.add('umdiddlediddlediddleumdiddlei')
```

```
28
29  /* selection from stimulus pool (parameters are N, M) */
30  p1.select(5, 5)
31  p2.select(5, 5)
32
33  /* need to bundle the two pools together, into an array */
34  var two_pools = [p1, p2]
35
36  /* set up 'study phase': show selected portions of pool */
37  study_phase(two_pools,
38              111, /* ISI */
39              4500 /* SET */ )
40
41  /* instruction slide */
42  instructions('second delay phase (5 seconds): (please press any key to continue)')
43
44  /* set up delay task: 5 seconds */
45  delay_task('please type names of as many countries as you can think of in 5 seconds, separated by
        spaces.. (please press any key to continue)',
46              10000 /* 5000 mS */)
47
48  /* instruction slide -- fixed duration */
49  var x = instructions('thank you for completing the delay task: test phase coming up in 5 seconds..')
50  x.set_expiry(5000)
51  x.key_expiry = false
52
53  /* some instructions before 'test phase' */
54  instructions('test phase coming up (please press any key to continue)')
55  instructions('when you see an image/word, please press m or n (please press any key to continue)')
56  instructions('please press m if you saw an image/word before (please press any key to continue)')
57  instructions('please press n if you did not see the image/word before (please press any key to
        continue)')
58
59  /* set up 'test phase' (user input recorded for whole randomized pool) */
60  test_phase(two_pools, /* stimulus pools */
61             111, /* ISI */
62             6000, /* SET */
63             6, /* extra feedback (one for every 6 slides, approx.) */
64             "How did you feel about the last stimulus? A=positive, B=negative, C=neutral, D=not sure",
                   /* message for extra feedback */
65             [65, 66, 67, 68] /* accepted keypresses for extra feedback */ )
66 }
```

4. Sample Response Data

4.1. **instructions.** Because the instructions example doesn't request any feedback from the user, the data appearing on the server after a run of the **experiments/instructions/** example isn't highly complex or informative. For this example, note that

- the "task_type" field is "instructions" in each case;
- the duration for which each message was displayed appear in the col.: "duration (mS)";
- the form responsible for generating the data is recorded in the first col.

```
1 url,event_id,task_id,task_type,trial_id,duration(mS),start(yyyy:mm:dd:hh:mn:ss:mls),end(yyyy:mm:dd:hh:
      mn:ss:mls),isi,set,stim_type,stim_id,stim_pool_id,response
2 http://web.uvic.ca/~lindlab/memory/experiments/instructions/memory.html,0,0,instructions
      ,0,1121,2017:50:23:18:51:20:315,2017:50:23:18:51:30:436,,,,,,""
3 http://web.uvic.ca/~lindlab/memory/experiments/instructions/memory.html,1,1,instructions
      ,0,414.6,2017:50:23:18:51:30:436,2017:50:23:18:51:30:851,,,,,,""
4 http://web.uvic.ca/~lindlab/memory/experiments/instructions/memory.html,2,2,instructions
      ,0,430.9,2017:50:23:18:51:30:851,2017:50:23:18:51:40:282,,,,,,""
5 http://web.uvic.ca/~lindlab/memory/experiments/instructions/memory.html,3,3,instructions
      ,0,470.9,2017:50:23:18:51:40:282,2017:50:23:18:51:40:753,,,,,,""
6 http://web.uvic.ca/~lindlab/memory/experiments/instructions/memory.html,4,4,instructions
      ,0,5002.2,2017:50:23:18:51:40:753,2017:50:23:18:51:90:755,,5000,,,,""
7 http://web.uvic.ca/~lindlab/memory/experiments/instructions/memory.html,5,5,instructions
      ,0,1974.8,2017:50:23:18:51:90:755,2017:50:23:18:51:11:729,,5000,,,,""
8 http://web.uvic.ca/~lindlab/memory/experiments/instructions/memory.html,6,6,instructions
      ,0,430.2,2017:50:23:18:51:11:729,2017:50:23:18:51:12:160,,,,,,""
```

4.2. **delay.** Below, an example of data rec'd due to the **experiments/delay/** example. Noteworthy aspects of this example include:

- ISI (please see line 3 in the CSV file below);
- text response data (free form) rec'd as part of the delay task (please see lines 5, 9 and 14 in the file below);
- differing values for SET on lines 9 and 14 below;
- for line 5, SET is absent (i.e., it's zero).

```
1 url,event_id,task_id,task_type,trial_id,duration(mS),start(yyyy:mm:dd:hh:mn:ss:mls),end(yyyy:mm:dd:hh:
      mn:ss:mls),isi,set,stim_type,stim_id,stim_pool_id,response
2 http://web.uvic.ca/~lindlab/memory/experiments/delay/memory.html,0,0,instructions
      ,0,3421.2,2017:50:23:18:51:43:394,2017:50:23:18:51:46:815,,,,,,""
3 http://web.uvic.ca/~lindlab/memory/experiments/delay/memory.html,1,1,isi
      ,0,502.9,2017:50:23:18:51:46:815,2017:50:23:18:51:47:318,500,500,,,,""
4 http://web.uvic.ca/~lindlab/memory/experiments/delay/memory.html,2,2,instructions
      ,0,1610.1,2017:50:23:18:51:47:318,2017:50:23:18:51:48:928,,,,,,""
5 http://web.uvic.ca/~lindlab/memory/experiments/delay/memory.html,3,1,delay
      ,0,16514.8,2017:50:23:18:51:48:928,2017:50:23:18:52:50:442,,,,,,"peru zanzibar zimbabwe india
      canada japan"
6 http://web.uvic.ca/~lindlab/memory/experiments/delay/memory.html,4,3,instructions
      ,0,1001.8,2017:50:23:18:52:50:443,2017:50:23:18:52:60:444,,,,,,""
7 http://web.uvic.ca/~lindlab/memory/experiments/delay/memory.html,5,4,isi
      ,0,505.1,2017:50:23:18:52:60:444,2017:50:23:18:52:60:949,500,500,,,,""
8 http://web.uvic.ca/~lindlab/memory/experiments/delay/memory.html,6,5,instructions
      ,0,1124.1,2017:50:23:18:52:60:949,2017:50:23:18:52:80:730,,,,,,""
9 http://web.uvic.ca/~lindlab/memory/experiments/delay/memory.html,7,4,delay
      ,0,5002.4,2017:50:23:18:52:80:730,2017:50:23:18:52:13:760,,5000,,,,"peru mozam"
```

```
10 http://web.uvic.ca/~lindlab/memory/experiments/delay/memory.html,8,6,instructions
     ,0,5005.6,2017:50:23:18:52:13:760,2017:50:23:18:52:18:810,,5000,,,,""
11 http://web.uvic.ca/~lindlab/memory/experiments/delay/memory.html,9,7,instructions
     ,0,3094,2017:50:23:18:52:18:810,2017:50:23:18:52:21:175,,,,,,""
12 http://web.uvic.ca/~lindlab/memory/experiments/delay/memory.html,10,8,isi
     ,0,501,2017:50:23:18:52:21:175,2017:50:23:18:52:21:676,500,500,,,,""
13 http://web.uvic.ca/~lindlab/memory/experiments/delay/memory.html,11,9,instructions
     ,0,1342.9,2017:50:23:18:52:21:676,2017:50:23:18:52:23:190,,,,,,""
14 http://web.uvic.ca/~lindlab/memory/experiments/delay/memory.html,12,8,delay
     ,0,6000.6,2017:50:23:18:52:23:190,2017:50:23:18:52:29:190,,6000,,,,"canada japan brazil usa "
15 http://web.uvic.ca/~lindlab/memory/experiments/delay/memory.html,13,10,instructions
     ,0,826,2017:50:23:18:52:29:190,2017:50:23:18:52:29:845,,,,,,""
```

4.3. **study-phase.** For the study_phase example response data below please note the following differences:

- ISI on lines 4, 6, 8, and so on;
- the "stim_type" field which takes values of "word" and "image" for various lines (each such line represents a screen where word or image data was shown);
- for lines where the "stim_type" field takes values of "word" and "image": the "stim_id" field takes a value which is either the word stimulus displayed (in the case that "stim_type"=="word") e.g., line 5, or the file-name of the image displayed (in the case the data line represents a screen where an image was displayed) e.g., in line 7;
- the field "stim_pool_id" which is seen to be 1 for each stimulus below (note that, in the last example, this will change).

```
1 url,event_id,task_id,task_type,trial_id,duration(mS),start(yyyy:mm:dd:hh:mn:ss:mls),end(yyyy:mm:dd:hh:
    mn:ss:mls),isi,set,stim_type,stim_id,stim_pool_id,response
2 http://web.uvic.ca/~lindlab/memory/experiments/study-phase/memory.html,0,0,instructions
    ,0,1298.3,2017:50:23:18:52:52:890,2017:50:23:18:52:53:388,,,,,,""
3 http://web.uvic.ca/~lindlab/memory/experiments/study-phase/memory.html,1,1,instructions
    ,0,755.2,2017:50:23:18:52:53:388,2017:50:23:18:52:54:143,,,,,,""
4 http://web.uvic.ca/~lindlab/memory/experiments/study-phase/memory.html,2,2,isi
    ,0,115.4,2017:50:23:18:52:54:143,2017:50:23:18:52:54:258,111,111,,,1,""
5 http://web.uvic.ca/~lindlab/memory/experiments/study-phase/memory.html,3,2,study_phase
    ,0,461.5,2017:50:23:18:52:54:259,2017:50:23:18:52:54:720,,5000,word,floccinaucinihilipilification
    ,1,""
6 http://web.uvic.ca/~lindlab/memory/experiments/study-phase/memory.html,4,2,isi
    ,1,113.1,2017:50:23:18:52:54:720,2017:50:23:18:52:54:833,111,111,,,1,""
7 http://web.uvic.ca/~lindlab/memory/experiments/study-phase/memory.html,5,2,study_phase
    ,1,302.9,2017:50:23:18:52:54:833,2017:50:23:18:52:55:136,,5000,image,../../images/70.jpg,1,""
8 http://web.uvic.ca/~lindlab/memory/experiments/study-phase/memory.html,6,2,isi
    ,2,114.9,2017:50:23:18:52:55:136,2017:50:23:18:52:55:251,111,111,,,1,""
9 http://web.uvic.ca/~lindlab/memory/experiments/study-phase/memory.html,7,2,study_phase
    ,2,310.5,2017:50:23:18:52:55:251,2017:50:23:18:52:55:562,,5000,image,../../images/48.jpg,1,""
10 http://web.uvic.ca/~lindlab/memory/experiments/study-phase/memory.html,8,2,isi
    ,3,115.1,2017:50:23:18:52:55:562,2017:50:23:18:52:55:677,111,111,,,1,""
11 http://web.uvic.ca/~lindlab/memory/experiments/study-phase/memory.html,9,2,study_phase
    ,3,265.2,2017:50:23:18:52:55:677,2017:50:23:18:52:55:942,,5000,image,../../images/194.jpg,1,""
12 http://web.uvic.ca/~lindlab/memory/experiments/study-phase/memory.html,10,2,isi
    ,4,113.5,2017:50:23:18:52:55:942,2017:50:23:18:52:56:550,111,111,,,1,""
13 http://web.uvic.ca/~lindlab/memory/experiments/study-phase/memory.html,11,2,study_phase
    ,4,287.2,2017:50:23:18:52:56:550,2017:50:23:18:52:56:343,,5000,image,../../images/16.jpg,1,""
```

4.4. **test-phase.** For the test_phase example response data below, please note the following news:

- user response to stimuli ("deja-vu"): note that an "M" (have already seen this sample) was recorded on line 15 (the data confirms that this sample was previously shown in the study_phase section, on line 6);
- regular ISI lines (blank screen intervals) throughout.

```
1 url,event_id,task_id,task_type,trial_id,duration(mS),start(yyyy:mm:dd:hh:mn:ss:mls),end(yyyy:mm:dd:hh:
      mn:ss:mls),isi,set,stim_type,stim_id,stim_pool_id,response
2 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,0,0,instructions
      ,0,3588.2,2017:50:23:18:53:22:211,2017:50:23:18:53:25:799,,,,,,""
3 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,1,1,isi
      ,0,112.7,2017:50:23:18:53:25:799,2017:50:23:18:53:25:912,111,111,,,1,""
4 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,2,1,study_phase
      ,0,467.9,2017:50:23:18:53:25:912,2017:50:23:18:53:26:380,,,image,../../images/16.jpg,1,""
5 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,3,1,isi
      ,1,114.6,2017:50:23:18:53:26:380,2017:50:23:18:53:26:495,111,111,,,1,""
6 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,4,1,study_phase
      ,1,376,2017:50:23:18:53:26:495,2017:50:23:18:53:26:871,,,word,floccinaucinihilipilification,1,""
7 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,5,1,isi
      ,2,114.9,2017:50:23:18:53:26:871,2017:50:23:18:53:26:986,111,111,,,1,""
8 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,6,1,study_phase
      ,2,345.9,2017:50:23:18:53:26:986,2017:50:23:18:53:27:332,,,image,../../images/70.jpg,1,""
9 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,7,1,isi
      ,3,115.4,2017:50:23:18:53:27:332,2017:50:23:18:53:27:447,111,111,,,1,""
10 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,8,1,study_phase
      ,3,310.6,2017:50:23:18:53:27:447,2017:50:23:18:53:27:758,,,image,../../images/48.jpg,1,""
11 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,9,1,isi
      ,4,116.2,2017:50:23:18:53:27:758,2017:50:23:18:53:27:874,111,111,,,1,""
12 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,10,1,study_phase
      ,4,324.9,2017:50:23:18:53:27:874,2017:50:23:18:53:28:199,,,image,../../images/194.jpg,1,""
13 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,11,2,instructions
      ,0,937.5,2017:50:23:18:53:28:199,2017:50:23:18:53:29:136,,,,,,""
14 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,12,3,isi
      ,0,337.7,2017:50:23:18:53:29:137,2017:50:23:18:53:29:474,333,333,,,1,""
15 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,13,3,test_phase
      ,0,1763.9,2017:50:23:18:53:29:474,2017:50:23:18:53:31:238,,,word,floccinaucinihilipilification,1,"M
      "
16 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,14,3,isi
      ,1,335.5,2017:50:23:18:53:31:238,2017:50:23:18:53:31:574,333,333,,,1,""
17 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,15,3,test_phase
      ,1,649.9,2017:50:23:18:53:31:574,2017:50:23:18:53:32:224,,,image,../../images/29.jpg,1,"N"
18 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,16,3,isi
      ,2,334.9,2017:50:23:18:53:32:224,2017:50:23:18:53:32:558,333,333,,,1,""
19 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,17,3,test_phase
      ,2,637,2017:50:23:18:53:32:559,2017:50:23:18:53:33:196,,,image,../../images/42.jpg,1,"N"
20 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,18,3,isi
      ,3,338.2,2017:50:23:18:53:33:196,2017:50:23:18:53:33:534,333,333,,,1,""
21 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,19,3,test_phase
      ,3,634.7,2017:50:23:18:53:33:534,2017:50:23:18:53:34:168,,,image,../../images/70.jpg,1,"M"
22 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,20,3,isi
      ,4,334.7,2017:50:23:18:53:34:168,2017:50:23:18:53:34:503,333,333,,,1,""
23 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,21,3,test_phase
      ,4,642.5,2017:50:23:18:53:34:503,2017:50:23:18:53:35:146,,,image,../../images/16.jpg,1,"M"
24 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,22,3,isi
      ,5,338.6,2017:50:23:18:53:35:146,2017:50:23:18:53:35:484,333,333,,,1,""
25 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,23,3,test_phase
      ,5,598.1,2017:50:23:18:53:35:484,2017:50:23:18:53:36:820,,,image,../../images/97.jpg,1,"N"
26 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,24,3,isi
      ,6,334.7,2017:50:23:18:53:36:820,2017:50:23:18:53:36:417,333,333,,,1,""
27 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,25,3,test_phase
      ,6,733.6,2017:50:23:18:53:36:417,2017:50:23:18:53:37:151,,,image,../../images/48.jpg,1,"M"
28 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,26,3,isi
      ,7,336.3,2017:50:23:18:53:37:151,2017:50:23:18:53:37:487,333,333,,,1,""
29 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,27,3,test_phase
      ,7,871.7,2017:50:23:18:53:37:487,2017:50:23:18:53:38:358,,,word,supercalifragilisticexpialidocious
      ,1,"N"
30 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,28,3,isi
      ,8,335.7,2017:50:23:18:53:38:358,2017:50:23:18:53:38:694,333,333,,,1,""
31 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,29,3,test_phase
      ,8,581.4,2017:50:23:18:53:38:694,2017:50:23:18:53:39:276,,,image,../../images/34.jpg,1,"M"
32 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,30,3,isi
      ,9,335.7,2017:50:23:18:53:39:276,2017:50:23:18:53:39:611,333,333,,,1,""
```

```
33 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,31,3,test_phase
     ,9,600.6,2017:50:23:18:53:39:611,2017:50:23:18:53:40:212,,,image,../../images/194.jpg,1,"M"
34 http://web.uvic.ca/~lindlab/memory/experiments/test-phase/memory.html,32,4,instructions
     ,0,847.4,2017:50:23:18:53:40:212,2017:50:23:18:53:41:590,,,,,,""
```

4.5. **my-experiment.** This more complex example combines aspects of the previous examples. The most important news with this example: note that the "stim_pool_id" field varies between samples: e.g., note that "floccinaucinihilipilification" carries the desired stimulus pool label "1", as it should, whereas "dogovarivatsya" carries the stimulus pool label "2", as it should.

The various data lines with types including: instructions, isi, study_phase, delay, and test_phase: these follow the pattern of the previous examples.

```
1 url,event_id,task_id,task_type,trial_id,duration(mS),start(yyyy:mm:dd:hh:mn:ss:mls),end(yyyy:mm:dd:hh:
    mn:ss:mls),isi,set,stim_type,stim_id,stim_pool_id,response
2 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,0,0,instructions
    ,0,554.6,2017:50:23:18:53:58:554,2017:50:23:18:53:59:109,,,,,,""
3 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,1,1,isi
    ,0,115,2017:50:23:18:53:59:109,2017:50:23:18:53:59:224,111,111,,,2,""
4 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,2,1,study_phase
    ,0,610.2,2017:50:23:18:53:59:224,2017:50:23:18:53:59:834,,4500,word,dogovarivatsya,2,""
5 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,3,1,isi
    ,1,114.6,2017:50:23:18:53:59:834,2017:50:23:18:53:59:949,111,111,,,2,""
6 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,4,1,study_phase
    ,1,346.4,2017:50:23:18:53:59:949,2017:50:23:18:54:00:295,,4500,image,../../images/198.jpg,2,""
7 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,5,1,isi
    ,2,113.5,2017:50:23:18:54:00:296,2017:50:23:18:54:00:409,111,111,,,2,""
8 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,6,1,study_phase
    ,2,332.6,2017:50:23:18:54:00:409,2017:50:23:18:54:00:742,,4500,image,../../images/194.jpg,1,""
9 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,7,1,isi
    ,3,113.9,2017:50:23:18:54:00:742,2017:50:23:18:54:00:856,111,111,,,2,""
10 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,8,1,study_phase
    ,3,311.6,2017:50:23:18:54:00:856,2017:50:23:18:54:10:167,,4500,image,../../images/16.jpg,1,""
11 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,9,1,isi
    ,4,113.3,2017:50:23:18:54:10:167,2017:50:23:18:54:10:281,111,111,,,2,""
12 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,10,1,study_phase
    ,4,278,2017:50:23:18:54:10:281,2017:50:23:18:54:10:559,,4500,image,../../images/48.jpg,1,""
13 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,11,1,isi
    ,5,114.8,2017:50:23:18:54:10:559,2017:50:23:18:54:10:673,111,111,,,2,""
14 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,12,1,study_phase
    ,5,230.2,2017:50:23:18:54:10:673,2017:50:23:18:54:10:904,,4500,image,../../images/70.jpg,1,""
15 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,13,1,isi
    ,6,113,2017:50:23:18:54:10:904,2017:50:23:18:54:20:170,111,111,,,2,""
16 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,14,1,study_phase
    ,6,258,2017:50:23:18:54:20:170,2017:50:23:18:54:20:275,,4500,word,floccinaucinihilipilification
    ,1,""
17 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,15,1,isi
    ,7,113.6,2017:50:23:18:54:20:275,2017:50:23:18:54:20:388,111,111,,,2,""
18 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,16,1,study_phase
    ,7,242.3,2017:50:23:18:54:20:388,2017:50:23:18:54:20:631,,4500,image,../../images/73.jpg,2,""
19 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,17,1,isi
    ,8,116.3,2017:50:23:18:54:20:631,2017:50:23:18:54:20:747,111,111,,,2,""
20 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,18,1,study_phase
    ,8,214.6,2017:50:23:18:54:20:747,2017:50:23:18:54:20:961,,4500,image,../../images/186.jpg,2,""
21 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,19,1,isi
    ,9,114.6,2017:50:23:18:54:20:961,2017:50:23:18:54:30:760,111,111,,,2,""
22 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,20,1,study_phase
    ,9,261.5,2017:50:23:18:54:30:760,2017:50:23:18:54:30:337,,4500,image,../../images/170.jpg,2,""
23 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,21,2,instructions
    ,0,340.9,2017:50:23:18:54:30:338,2017:50:23:18:54:30:678,,,,,,""
24 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,22,3,isi
    ,0,501.1,2017:50:23:18:54:30:678,2017:50:23:18:54:40:179,500,500,,,,""
25 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,23,4,instructions
    ,0,941.9,2017:50:23:18:54:40:180,2017:50:23:18:54:50:121,,,,,,""
```

```
26 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,24,3,delay
     ,0,10003.9,2017:50:23:18:54:50:122,2017:50:23:18:54:15:125,,10000,,,,"peru brazil panama canada
     mexico usa e"
27 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,25,5,instructions
     ,0,5003.9,2017:50:23:18:54:15:125,2017:50:23:18:54:20:129,,5000,,,,"N"
28 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,26,6,instructions
     ,0,1127.3,2017:50:23:18:54:20:129,2017:50:23:18:54:21:256,,,,,,""
29 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,27,7,instructions
     ,0,494.2,2017:50:23:18:54:21:256,2017:50:23:18:54:21:750,,,,,,""
30 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,28,8,instructions
     ,0,386.1,2017:50:23:18:54:21:750,2017:50:23:18:54:22:136,,,,,,""
31 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,29,9,instructions
     ,0,371.2,2017:50:23:18:54:22:136,2017:50:23:18:54:22:508,,,,,,""
32 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,30,10,isi
     ,0,111.5,2017:50:23:18:54:22:508,2017:50:23:18:54:22:619,111,111,,,1,""
33 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,31,10,test_phase
     ,0,1277.9,2017:50:23:18:54:22:619,2017:50:23:18:54:23:897,,6000,image,../../images/97.jpg,1,"N"
34 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,32,10,isi
     ,1,115,2017:50:23:18:54:23:897,2017:50:23:18:54:24:120,111,111,,,2,""
35 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,33,10,test_phase
     ,1,1061.4,2017:50:23:18:54:24:120,2017:50:23:18:54:25:740,,6000,image,../../images/186.jpg,2,"N"
36 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,34,10,isi
     ,2,113.4,2017:50:23:18:54:25:740,2017:50:23:18:54:25:187,111,111,,,2,""
37 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,35,10,test_phase
     ,2,819.9,2017:50:23:18:54:25:187,2017:50:23:18:54:26:700,,6000,word,compassion,2,"N"
38 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,36,11,feedback
     ,0,1668.5,2017:50:23:18:54:26:700,2017:50:23:18:54:27:675,,,,,,"A"
39 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,37,10,isi
     ,3,115.6,2017:50:23:18:54:27:675,2017:50:23:18:54:27:791,111,111,,,2,""
40 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,38,10,test_phase
     ,3,726.7,2017:50:23:18:54:27:791,2017:50:23:18:54:28:518,,6000,image,../../images/73.jpg,2,"N"
41 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,39,10,isi
     ,4,115,2017:50:23:18:54:28:518,2017:50:23:18:54:28:633,111,111,,,2,""
42 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,40,10,test_phase
     ,4,627,2017:50:23:18:54:28:633,2017:50:23:18:54:29:260,,6000,image,../../images/9.jpg,2,"N"
43 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,41,12,feedback
     ,0,806.7,2017:50:23:18:54:29:260,2017:50:23:18:54:30:660,,,,,,"B"
44 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,42,10,isi
     ,5,116.2,2017:50:23:18:54:30:660,2017:50:23:18:54:30:183,111,111,,,2,""
45 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,43,10,test_phase
     ,5,980.9,2017:50:23:18:54:30:183,2017:50:23:18:54:31:163,,6000,word,umdiddlediddlediddleumdiddlei
     ,2,"N"
46 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,44,10,isi
     ,6,115.4,2017:50:23:18:54:31:163,2017:50:23:18:54:31:279,111,111,,,2,""
47 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,45,10,test_phase
     ,6,697.8,2017:50:23:18:54:31:279,2017:50:23:18:54:31:977,,6000,image,../../images/170.jpg,2,"M"
48 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,46,10,isi
     ,7,115.8,2017:50:23:18:54:31:977,2017:50:23:18:54:32:920,111,111,,,2,""
49 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,47,10,test_phase
     ,7,595.3,2017:50:23:18:54:32:920,2017:50:23:18:54:32:688,,6000,image,../../images/198.jpg,2,"M"
50 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,48,10,isi
     ,8,112.8,2017:50:23:18:54:32:688,2017:50:23:18:54:32:800,111,111,,,2,""
51 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,49,10,test_phase
     ,8,669,2017:50:23:18:54:32:801,2017:50:23:18:54:33:469,,6000,word,dogovarivatsya,2,"M"
52 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,50,10,isi
     ,9,113.2,2017:50:23:18:54:33:469,2017:50:23:18:54:33:583,111,111,,,2,""
53 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,51,10,test_phase
     ,9,517.9,2017:50:23:18:54:33:583,2017:50:23:18:54:34:101,,6000,image,../../images/78.jpg,2,"N"
54 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,52,10,isi
     ,10,113.8,2017:50:23:18:54:34:101,2017:50:23:18:54:34:214,111,111,,,1,""
55 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,53,10,test_phase
     ,10,617.8,2017:50:23:18:54:34:214,2017:50:23:18:54:34:832,,6000,image,../../images/42.jpg,1,"N"
56 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,54,10,isi
     ,11,116.2,2017:50:23:18:54:34:832,2017:50:23:18:54:34:948,111,111,,,1,""
57 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,55,10,test_phase
     ,11,601.1,2017:50:23:18:54:34:948,2017:50:23:18:54:35:550,,6000,image,../../images/70.jpg,1,"M"
58 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,56,13,feedback
     ,0,1337.7,2017:50:23:18:54:35:550,2017:50:23:18:54:36:887,,,,,,"A"
```

```
59 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,57,10,isi
     ,12,116,2017:50:23:18:54:36:887,2017:50:23:18:54:37:300,111,111,,,1,""
60 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,58,10,test_phase
     ,12,901.1,2017:50:23:18:54:37:300,2017:50:23:18:54:37:904,,6000,image,../../images/194.jpg,1,"M"
61 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,59,14,feedback
     ,0,1704.8,2017:50:23:18:54:37:904,2017:50:23:18:54:39:609,,,,,,"C"
62 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,60,10,isi
     ,13,116.6,2017:50:23:18:54:39:609,2017:50:23:18:54:39:726,111,111,,,1,""
63 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,61,10,test_phase
     ,13,755.7,2017:50:23:18:54:39:726,2017:50:23:18:54:40:481,,6000,image,../../images/16.jpg,1,"M"
64 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,62,10,isi
     ,14,115.2,2017:50:23:18:54:40:481,2017:50:23:18:54:40:597,111,111,,,2,""
65 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,63,10,test_phase
     ,14,761.4,2017:50:23:18:54:40:597,2017:50:23:18:54:41:358,,6000,image,../../images/80.jpg,2,"N"
66 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,64,10,isi
     ,15,111.7,2017:50:23:18:54:41:358,2017:50:23:18:54:41:470,111,111,,,1,""
67 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,65,10,test_phase
     ,15,726.8,2017:50:23:18:54:41:470,2017:50:23:18:54:42:196,,6000,image,../../images/48.jpg,1,"M"
68 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,66,10,isi
     ,16,112.4,2017:50:23:18:54:42:196,2017:50:23:18:54:42:309,111,111,,,1,""
69 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,67,10,test_phase
     ,16,758.4,2017:50:23:18:54:42:309,2017:50:23:18:54:43:670,,6000,word,
     supercalifragilisticexpialidocious,1,"N"
70 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,68,10,isi
     ,17,112.3,2017:50:23:18:54:43:670,2017:50:23:18:54:43:180,111,111,,,1,""
71 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,69,10,test_phase
     ,17,639.6,2017:50:23:18:54:43:180,2017:50:23:18:54:43:819,,6000,image,../../images/34.jpg,1,"N"
72 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,70,10,isi
     ,18,115,2017:50:23:18:54:43:819,2017:50:23:18:54:43:934,111,111,,,1,""
73 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,71,10,test_phase
     ,18,672,2017:50:23:18:54:43:934,2017:50:23:18:54:44:606,,6000,image,../../images/29.jpg,1,"N"
74 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,72,10,isi
     ,19,111.8,2017:50:23:18:54:44:606,2017:50:23:18:54:44:718,111,111,,,1,""
75 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,73,10,test_phase
     ,19,705.3,2017:50:23:18:54:44:718,2017:50:23:18:54:45:423,,6000,word,floccinaucinihilipilification
     ,1,"M"
76 http://web.uvic.ca/~lindlab/memory/experiments/my-experiment/memory.html,74,15,instructions
     ,0,1072,2017:50:23:18:54:45:423,2017:50:23:18:54:46:495,,,,,,""
```

5. Source Code: Client Side

5.1. **egg-timer.js.** This file develops a timer object which is used to control transitions between stimuli or other messages.

```
/* via developer.mozilla.org/en-US/docs/Web/API/WindowOrWorkerGlobalScope/clearTimeout */
var egg_timer = {

  /* callback */
  setup: function(t_ms){

    /* assert parameter is a number */
    if(typeof this.timeoutID === "number"){
      this.cancel()
    }

    /* what to do when the timer expires */
    this.timeoutID = window.setTimeout(
      function(){
        var now = ctx.get_state()
        var id = now.id
        now.ding = true
        if(now.key_expiry == false || now.expiry_ms > 0){
          now.expire()
        }
      }.bind(this), t_ms
    )
  }, cancel: function(){
    window.clearTimeout(this.timeoutID)
    this.timeoutID = undefined
  }
}
```

5.2. **key.js.** This file represents the keyboard event handling which includes filtering of keys, actions determined by various specified keys, and special treatment for the appropriate response for a given key-code, according to the type of experiment component that is currently active.

```
/* convert from unicode to familiar symbol */
function unicode_from_key_event(e){
  return e.charCode ? e.charCode : e.keyCode
}

/* keyboard status array (unicode format) */
var key_unicode = {}

/* keyboard event handler function */
function keyboard_module(){

  /* set up key-down event handler function */
  document.onkeydown = function(e){

    /* unicode vs. character representation */
    var unicode = unicode_from_key_event(e), key = String.fromCharCode(unicode)

    /* inverted question mark */
    if(unicode == 191){
      unicode = 63, key = '?'
    }else if(unicode == 188){
      unicode = 44, key = ','
    }else if(unicode == 190){
      unicode = 46, key = "."
    }else if(unicode == 13){

      /* replace enter with space */
      unicode = 32, key = " "
    }

    if(unicode == 27){

      /* do nothing if we get a key that is code 27, but not an escape key.. */
      if(!(e.key == "Escape" || e.key == "Esc")){
        return;
      }
    }

    if(unicode == 222){
      unicode = 39, key ="'"
    }

    /* console.log("unicode", unicode) */

    key_unicode[unicode] = true

    var ignore = [20, 192, 189, 187, 93, 91, 219, 221, 222, 220, 186, 33, 36, 34, 35, 37, 38, 40]

    /* ignore caps-lock and other special key */
    if(ignore.includes(unicode)){
      return
    }

    var allow = [];
    for(var i=65; i<=90; i++){
      allow.push(i);
    }
    for(var i=48; i<=57; i++){
      allow.push(i);
    }

    /* allow space bar */
    allow.push(32)

```

```
65      /* allow escape key */
66      allow.push(27)
67
68      /* allow comma */
69      allow.push(44)
70
71      /* allow period */
72      allow.push(46)
73
74      /* allow question mark */
75      allow.push(63)
76
77      /* allow backspace */
78      allow.push(8)
79
80      /* allow single right quotation mark */
81      allow.push(39)
82
83      if(!allow.includes(unicode)){
84        return
85      }
86
87      /* when are we? */
88      var now = ctx.get_state()
89
90      /* record key press, if admissible */
91      var admissible_keys = now.get_admissible_keys()
92      if(admissible_keys.includes(unicode) || now.type == 'delay'){
93        now.record_key_stroke(unicode)
94      }
95
96      /* by default, transition from a slide upon key-press */
97      var go = true
98
99      /* special treatment for delay task */
100     if(now.type == 'delay'){
101       if(now.txt == null){
102
103         /* init */
104         now.txt = ''
105       }
106       if(unicode == 8){
107
108         /* backspace */
109         var len = now.txt.length
110         now.txt = now.txt.substring(0, len - 1)
111
112       }else if(admissible_keys.includes(27) && unicode == 27){
113
114         /* break out of free-form text input mode with <esc> key */
115         ctx.clear_tmr()
116         now.expire()
117
118         return key_unicode
119
120       }else{
121
122          /* add character to buffer */
123         if(unicode >= 65 && unicode <= 90){
124           now.txt += key.toLowerCase()
125         }else{
126           now.txt += key
127         }
128       }
129
130       /* redraw */
131       update()
132     }
```

```
133
134     /* check if this state "requires" keyboard input */
135     if(now.require_key() == true){
136
137       /* is the key that was pressed, in the list of "admissible" keys? */
138       if(admissible_keys.includes(unicode)){
139
140         /* if we have a "deja-vu" variable, calculate a score */
141         if(!(now.deja == undefined)){
142           ctx.questions_total += 1
143
144           /* check for N or M keypress */
145           if((now.deja == true && unicode == 77) || (now.deja == false && unicode == 78)){
146             ctx.questions_correct += 1
147           }
148         }
149       }else{
150         /* block if a key was required but the one entered was not admissible */
151         go = false
152       }
153     }
154
155     /* t <-- t + 1 */
156     if(now && now.key_expiry && go){
157
158         /* clear the timer and "go next" */
159         ctx.clear_tmr()
160         now.expire()
161     }
162   }
163   return key_unicode
164 }
```

5.3. **main.js.** This file represents the various housekeeping activities necessary to establish a graphics context in HTML5, set up an experiment according to user specifications, load various scripts and images, and enter an event-driven loop, in order to display a series of stimuli and respond to user interactions.

```
var abs_path = '../../', ctx = canvas.getContext("2d")

/* background color, shape parameter and font size */
document.bgColor = "#FFFFFF", ctx.pad = 20, ctx.font_size = 30

/* canvas dimensions manipulation */
var less = function(x){
  return x - ctx.pad
}

ctx.w = function(){
  return less(window.innerWidth)
}

ctx.h = function(){
  return less(window.innerHeight)
}

/* canvas resize */
function resize(){
  canvas.width = ctx.w(), canvas.height = ctx.h()
}

/* load corporate logo */
ctx.symbol = new Image()
ctx.symbol.fn = abs_path + "logo/uvic_gray.png"

/* algo to draw scaled corporate logo */
ctx.draw_symbol = function(){
  var s_f = 5, pad = this.pad, s = this.symbol
  var ww = window.innerWidth, wh = window.innerHeight
  var w = ww - pad, h = wh - pad, w_s = s.width, h_s = s.height
  var wf = (ww - pad) / (s_f * w_s), lwf = w_s * wf, lhf = h_s * wf
  this.drawImage(s, w - lwf, h - lhf, lwf, lhf)
}

/* access current "state" (a state represents a particular "trial" in an experiment) */
ctx.set_state = function(s){
  last_state = null
  if(ctx.current_state != null){
    last_state = ctx.current_state
  }
  ctx.current_state = s

  /* sanity check */
  if(s != null){
    s.daddy = last_state
  }
  return(s)
}

/* access present "state" */
ctx.get_state = function(){
  return ctx.current_state
}

/* trigger update/plotting from window resize event */
window.onresize = function(event){
  update()
}

/* update the canvas (present the current "trial") */
function update(){
  resize()
```

```
65   var now = ctx.get_state()
66   if(now){
67     now.show(ctx)
68   }
69 }
70
71 /* "in" hook: plot the current trial */
72 window.onload = function(){
73   update()
74 }
75
76 /* set up timer to coordinate transitions between trials */
77 ctx.egg_timer = egg_timer
78
79 ctx.clear_tmr = function(){
80   ctx.egg_timer.cancel()
81 }
82
83 ctx.init_tmr = function(t_ms){
84   ctx.egg_timer.setup(t_ms)
85 }
86
87 /* initialize reference to first and most-recently-initialized trials */
88 ctx.last_new_state = null, ctx.first_new_state = null
89
90 /* count number of questions answered correctly (this is redundant) */
91 ctx.questions_correct = 0, ctx.questions_total = 0
92
93 /* this function sets up the experiment (according to the user function my_experiment)
94 and we trigger this function after all the images have loaded. */
95 function run_before_loading_images(){
96
97   /* set up an experiment according to user specs/code */
98   my_experiment(ctx)
99
100  /* display a goodbye message every time */
101  instructions('survey complete: thank you for your participation')
102
103  ctx.last_state = ctx.last_new_state, ctx.first_state = ctx.first_new_state
104
105  /* start at the very beginning, it's a very good place to start.. */
106  ctx.set_state(ctx.first_state)
107
108  /* respond to keyboard events */
109  key_unicode = keyboard_module()
110
111  /* start "stopwatch" */
112  ctx.t0 = window.performance.now()
113
114 }
115
116
117 /* load some image files: need to change if the image database changes */
118 var n_imgs = 200, n_imgs_to_load = 0, n_imgs_loaded = 0
119
120 var images_to_load = []
121
122 /* scan images to determine which need to be loaded */
123 var idx = new Array()
124 ctx.imgs = new Array()
125 for(var i = 1; i <= n_imgs; i++){
126     idx.push(i)
127 }
128
129 /* randomize the order of the images */
130 shuffle(idx)
131
132 for(var i=1; i<=n_imgs; i++){
```

```
133   var img = new Image()
134   img.fn = abs_path + 'images/' + idx[i-1] + '.jpg'    // load_img(img) //var my_img = load_img(img_fn)
135   ctx.imgs.push(img)
136 }
137
138 var get_image = function(){
139   return ctx.imgs[n_imgs_to_load++]
140 }
141
142 /* load image data */
143 function load_img(i){
144   ctx.imgs[i].onload = function(){
145
146     /* have all images been loaded? */
147     if(++n_imgs_loaded == n_imgs_to_load){
148
149       /* proceed to init the experiment */
150       ctx.get_state().start()
151     }
152   }
153
154   /* load the image */
155   ctx.imgs[i].src = ctx.imgs[i].fn
156   return ctx.imgs[i]
157 }
158
159
160 /* keep track of the "task-index" as the experiment is intialized */
161 var next_task_id = 0
162
163 run_before_loading_images()
164
165
166 /* load the symbol */
167 ++ n_imgs_to_load
168
169 ctx.symbol.onload = function(){
170
171    /* have all images been loaded? */
172   if(++n_imgs_loaded == n_imgs_to_load){
173
174      /* proceed to init the experiment */
175       ctx.get_state().start()
176    }
177 }
178 ctx.symbol.src = ctx.symbol.fn
179
180 /* load the other images.. */
181 for(var i=0; i<ctx.imgs.length; i++){
182   if(ctx.imgs[i].load_me){
183     load_img(i)
184   }
185 }
```

5.4. **memory.js.** This file loads the various dependencies in order. Without this kind of approach, JavaScript loads scripts "asynchronously" i.e., not in order, so an approach like this is required for the system to function properly, as the system does not work if the dependencies load out of order.

```
var js_added = -1, deps = []

/* j4v4scr1pt 4n4l0g 0f 1nclud3 st4t3m3nt */
function add_js(fn){
  var body = document.getElementsByTagName('body')[0], s = document.createElement('script')
  s.async = false, s.src = fn + '.js'

  /* wait until script is loaded before proceeding.. */
  s.onload = function(){
    if(++js_added < deps.length){
      add_js(deps[js_added])
    }
  }
  body.appendChild(s)
}

/* c411 411 th3 ch1ldr3n */
dependencies = ['text', 'key', 'util', 'task', 'pool', 'state', 'egg-timer']
for(var d in dependencies){
  deps.push('../../' + dependencies[d])
}
deps.push('my-experiment')
deps.push('../../main')
add_js(deps[0], '')
```

5.5. **pool.js.** This file develops the notion of "stimulus pool", a list of objects that might either be word-data or image-data. Importantly, this includes pseudo-random sampling and pseudo-random shuffling of word/image stimulus data.

```
var next_pool_id = 0

/* stimulus pool - object that has words or images added to it. Selections drawn randomly for "study phase"
by draw() method. That selection is shuffled back into the deck, for the "test phase" */
function pool(){

  /* keep count */
  ++ next_pool_id

  this.is_pool = true, this.pool_id = next_pool_id, this.ctx = ctx, this.stimuli = new Array()

  /* add a stimulus to the pool */
  this.add = function(stim){
    this.stimuli.push(stim)
    stim.load_me = true
    return stim
  }

  /* add one or more images to the stimulus pool */
  this.add_image = function(n=1){
    for(var i = 0; i < n; i++){
      this.add(get_image())
    }
  }

  /* set number of samples for study phase */
  this.set_n = function(n){
    this.n = n
  }

  /* set number of additional samples to be included for test phase */
  this.set_m = function(m){

    /* subsequently to drawing "n" items from the pool (without replacement),
       a further "m" samples are drawn from the pool. For the test phase, the
      "n" and "m" selections are mixed together and shuffled. */
    this.m = m
  }

  /* get */
  this.get_n = function(){
    return this.n
  }

  /* get */
  this.get_m = function(){
    return this.m
  }

  /* remove any "blank" elements that appeared from drawing elements without
    replacement */
  this.remove_blanks = function(){
    this.stimuli = this.stimuli.filter(function(){return true})
  }

  /* pseudorandom selection of size "n" */
  this.draw_n = function(){

    if(this.selection_n){
      console.log('error: n-selection already made from this pool.')
      return null
    }

    /* check the selection size */
```

```
65      var n = parseInt(this.get_n())
66      if(n > this.stimuli.length){
67        console.log('error: n > this.stimuli.length')
68        return null
69      }
70
71      /* make a pseudorandom selection */
72      this.selection_n = new Array()
73      var rem = this.stimuli.length
74      for(var i = 0; i < n; i++){
75        var qx = rand() * parseFloat(rem --), idx = parseInt(qx)
76        this.selection_n.push(this.stimuli[idx])
77        delete this.stimuli[idx]
78        this.remove_blanks()
79      }
80    }
81
82    /* pseudorandom selection of size "m" */
83    this.draw_m = function(){
84
85      if(this.selection_m){
86        console.log('error: m-selection already made from this pool.')
87        return null
88      }
89
90      /* check the selection size */
91      var m = parseInt(this.get_m())
92      if(m > this.stimuli.length){
93        console.log('error: m > this.stimuli.length')
94        return null
95      }
96
97      /* make a pseudorandom selection */
98      this.selection_m = new Array()
99      var rem = this.stimuli.length
100     for(var i = 0; i < m; i++){
101       var qx = rand() * parseFloat(rem --), idx = parseInt(qx)
102       this.selection_m.push(this.stimuli[idx])
103       delete this.stimuli[idx]
104       this.remove_blanks()
105     }
106   }
107
108   /* for initializing a test phase: mix "N"-selection and "M"-selection together */
109   this.reshuffle = function(){
110
111     /* put the "N"-selection and "M" selection, together in array to_shuffle,
112       which will be shuffled */
113     var to_shuffle = [], i = 0
114
115     /* add the "N"-selection */
116     for(i = 0; i < this.selection_n.length; i++){
117       var dat_i = new Array()
118       dat_i.push(this.selection_n[i])
119       dat_i.push(true)
120       to_shuffle.push(dat_i)
121     }
122
123     /* add the "M"-selection */
124     for(i = 0; i < this.selection_m.length; i++){
125       var dat_i = new Array()
126       dat_i.push(this.selection_m[i])
127       dat_i.push(false)
128       to_shuffle.push(dat_i)
129     }
130
131     /* "shuffle"-- randomize the ordering of the combined array */
132     var shuffled = new Array(), deja_vu = new Array(), rem = to_shuffle.length
```

```
133     while((rem --) > 0){
134       var idx = parseInt(rand() * parseFloat(rem)), dat_i = to_shuffle[idx]
135       shuffled.push(dat_i[0])
136       deja_vu.push(dat_i[1])
137       delete to_shuffle[idx]
138       to_shuffle = to_shuffle.filter(function(){return true})
139     }
140     return [shuffled, deja_vu]
141   }
142
143
144   /* perform all of the above */
145   this.draw = function(){
146     this.draw_n()
147     this.draw_m()
148     this.reshuffle()
149   }
150
151   /* set N, M parameters and make a selection cf the above */
152   this.select = function(n, m=n){
153     this.set_n(n)
154     this.set_m(m)
155     this.draw()
156   }
157
158   /* end of "pool::pool()" */
159   return this
160 }
161
162 /* following the convention to wrap away the new() operator */
163 function stimulus_pool(){
164   return new pool()
165 }
```

5.6. **state.js.** This file develops the notion of "state": a "state" represents an atomic interaction of the interactive survey (much like a "slide" in MS Powerpoint).

The notion of "state" includes logical aspects of determining whether to wait for specified input keys, before proceeding to the next "state": an experiment is represented as a sequential list of "states".

Notably, the "state" concept includes the functions to display word/image stimuli or messages, on the screen. Logic for dumping data to the server (upon reaching the terminal "state") also resides here.

```
/* global counter for states/ AKA frames/ AKA slides */
var last_state_id = -1

/* reference to 2d canvas graphics context */
function get_ctx(){
  return canvas.getContext("2d") //document.getElementsByTagName("canvas")[0].getContext("2d");
}

/* state: generic object representing trial (like a card in "hypercard") */
function state(expiry_ms  =     0,  /* max. presentation time (mS) */
               key_expiry =  true,  /* force expiry by key-press (true <--> on) */
               intvl_ms   =     0,  /* interval btwn stimuli.. (ISI) 'blank slide' */
               img_idx    =    -1,  /* image data (if any) */
               txt        =  null,  /* text data (if any) */
               successor  =  null){
  var ctx = get_ctx()
  this.action = null, this.ding = false, this.id = ++ last_state_id

  /* is a key-press required to transition? */
  this.key_required = false

  /* array to store admissible key-codes for data entry or transition to next "slide":
    default: M, N */
  this.admissible_keys = [77, 78]

  this.get_admissible_keys = function(){
    return this.admissible_keys
  }

  this.clear_admissible_keys = function(){
    this.admissible_keys = new Array()
  }

  this.add_admissible_key = function(k){
    this.admissible_keys.push(k)
  }

  /* this array will record the keystroke data received while residing in this state */
  this.key_strokes = new Array()

  this.record_key_stroke = function(k){
    this.key_strokes.push(k)
  }

  this.set_pool_id = function(pid){
    this.pool_id = pid
  }

  this.get_pool_id = function(){
    return this.pool_id ? this.pool_id : ""
  }

  /* keep a reference to this state, if it's the first one ever.. */
  if(ctx.first_new_state == null){
    ctx.first_new_state = this
  }

  /* only applies if there's a "next" trial, if this is a trial */
```

```
59  this.intvl_ms = intvl_ms
60
61  /* numeric */
62  this.expiry_ms = expiry_ms
63
64  /* boolean */
65  this.key_expiry = key_expiry
66
67  /* global image index (images added as member of ctx) */
68  this.img_idx = img_idx, this.successor = null, this.predecessor = ctx.last_new_state
69
70  this.require_key = function(){
71    return this.key_required
72  }
73
74  var id = (this.predecessor == null) ? -1 : this.predecessor.id
75  ctx.last_new_state = this
76
77  /* sanity check: make sure the predecessor points here */
78  if(this.predecessor){
79    this.predecessor.set_successor(this)
80  }
81
82  /* where are we going? */
83  this.set_successor = function(s){
84    this.successor = s
85  }
86
87  /* plot text or images */
88  this.show = function(){
89
90    /* execute associated action, if we have one */
91    if(this.action){
92      this.action(this)
93    }
94    var ctx = get_ctx()
95    ctx.clearRect(0, 0, ctx.w(), ctx.h())
96
97    /* upper text */
98    if(this.txt){
99      wrap_text(this.txt, ctx, 0)
100   }
101
102   /* middle text */
103   if(this.txt2){
104     wrap_text(this.txt2, ctx, ctx.h() - (2 * ctx.font_size + 20))
105   }
106
107   /* img or middle text (if word stim) */
108   if(this.img_stim){
109     draw_img(this.img_stim, ctx)
110   }
111
112   /* might need the wrap_text back on for long strings.. */
113   if(this.wrd_stim){
114
115     /* no wrap */
116     centre_text(this.wrd_stim)
117   }
118
119   /* logo of no image/ lower text present */
120   if(!this.txt2){
121     ctx.draw_symbol()
122   }
123 }
124
125 /* state expires by timer or key press */
126 this.set_expiry = function(t_ms){
```

```
127
128     /* follow clock or key to keep the show going */
129     this.expiry_ms = t_ms
130
131     /* state expires by key press */
132     if(t_ms <= 0){
133       this.key_expiry = true
134     }
135   }
136
137   /* enter a state (begin) */
138   this.start = function(){
139     var ctx = get_ctx()
140
141     /* start the clock.. */
142     this.t0 = window.performance.now(), this.start_date_time = date_time()
143
144     /* do data dump, if we're at the end */
145     if(this.id >= last_state_id){ //== ctx.last_state){
146
147         /* window.location.href == http://domain/memory/examples/test_phase/memory.html */
148         var href = window.location.href
149
150         /* go through all the states and record (in string format) the info we'd like to appear on the
                server */
151         var state_i = ctx.first_state, state_index = 0, message = "url,event_id,task_id,task_type,
                trial_id,duration(mS),start(yyyy:mm:dd:hh:mn:ss:mls),end(yyyy:mm:dd:hh:mn:ss:mls),isi,set,
                stim_type,stim_id,stim_pool_id,response\n"
152         for(var state_i = ctx.first_state; state_i != ctx.last_state; state_i = state_i.successor){
153
154           var stim_type = null, my_stim  = null, pi = ""
155
156           /* "the right way to check if a variable is undefined or not" */
157           if(typeof state_i.pool_id !== 'undefined'){
158             pi = JSON.parse(JSON.stringify(state_i.pool_id))
159           }
160
161           /* assign "stimulus type" keyword */
162           if(state_i.wrd_stim){
163             stim_type = "word", my_stim = state_i.wrd_stim
164           }
165           if(state_i.img_stim){
166             stim_type = "image", my_stim = state_i.img_stim.fn
167           }
168           if(!stim_type){
169             stim_type = ""
170           }
171           if(!my_stim){
172             my_stim = ""
173           }
174
175           /* for a given "state", record a line of data */
176           message += href + ","
177
178           /* event_id: global index / line number */
179           message += state_index.toString() + ","
180
181           /* task_id */
182           message += state_i.task_id + ","
183
184           /* task_type */
185           message += state_i.type + ","
186
187           /* trial_id */
188           message += state_i.trial_id + ","
189           message += Math.round(10. * (state_i.t1 - state_i.t0)) / 10. + ","
190           message += parse_date_time(state_i.start_date_time).toString() + ","
191           message += parse_date_time(state_i.end_date_time).toString() + ","
```

```
192
193           /* ISI */
194           if(state_i.type == 'isi'){
195             message += state_i.expiry_ms.toString()
196           }
197           message += ","
198
199           if(!state_i.expiry_ms){
200             state_i.expiry_ms = ""
201           }
202
203           /* SET */
204           message += state_i.expiry_ms.toString() + ","
205
206           /* stimulus type */
207           message += stim_type.toString() + ","
208
209           /* stimulus id */
210           message += my_stim.toString() + ","
211
212           /* stimulus-pool id */
213           message += pi.toString() + ","
214
215           /* user response */
216           var response = '"'
217
218           if(state_i.type == 'delay'){
219
220             /* use the response text (not the sequence of characters). When testing with Max,
                    discovered we could see a symbol for each keystroke, in the data stream (incl., e.g.,
                    backspace characters). We want the final result, not the intermediary. */
221             response += state_i.txt
222           }else{
223
224             /* write out the individual response key(s) in terms of the representative characters */
225             for(var k in state_i.key_strokes){
226               response += String.fromCharCode(state_i.key_strokes[k])
227             }
228           }
229           message += response + '"'
230           if(response=='""'){
231             response = ''
232           }
233
234           /* filter the response data for possible newline characters */
235           response.replace('\n', ' ')
236
237           /* add a newline character */
238           message += "\n"
239
240           /* go next */
241           ++ state_index
242         }
243
244         /* remove last three elements from array: take current page and navigate to:
245           ../../xml-receive.py == http://domain/memory/xml-receive.py */
246         var words = href.split('/')
247         var nwords = words.length
248         var target = words.splice(0, nwords-3).join('/') + '/xml-receive.py'
249
250         /* send the message to the server-side script at URL: target */
251         xml_send(message, target)
252       }
253
254       /* clear the timer */
255       ctx.clear_tmr()
256
257       /* plot the current trial */
```

```
258     this.show(ctx)
259
260     /* start the timer? */
261     if(this.expiry_ms > 0){
262       ctx.init_tmr(this.expiry_ms, this.expire)
263     }
264     return null
265   }
266
267   /* pr0c33d t0 th3 n3xt 5+4t3 */
268   this.expire = function(){
269     var ctx = get_ctx()
270
271     /* st0p 4ll th3 cl0ck5 */
272     ctx.clear_tmr()
273
274     /* r3c0rd st0p t1m3 */
275     this.end_date_time = date_time(), this.t1 = window.performance.now()
276     var txt = this.txt, suc_txt = null, suc = this.successor
277
278     if(suc && suc.txt){
279       suc_txt = suc.txt
280     }
281
282     /* enter next state */
283     if(this.successor && (this.successor!=this)){
284       ctx.set_state(this.successor)
285       ctx.get_state().start()
286     }
287   }
288   return this
289 }
```

5.7. **task.js.** This file develops the various types of possible components available from which an experiment is assembled: instructions, delay, feedback, study phase, and test phase; in each case, the component (a "task") is developed in terms a number of the atomic constructions "states" developed in the last file

```
/* Event hierarchy: 1) Experiment (includes multiple tasks) 2) Task (includes multiple trials) 3) Trial
     (each task includes multiple basic events) */

/* instructions task (show a slide with a message on it) */
function instructions(txt){
  var my_task_id = next_task_id++

  /* initialize generic "trial" object */
  var x = new state()

  /* set associated text field */
  x.txt = txt

  /* no timer for the trial */
  x.set_expiry(0)
  x.type = 'instructions', x.task_id = my_task_id, x.trial_id = 0
  return x
}

/* previously known as feedback task */
function feedback(txt, keys){
  var my_task_id = next_task_id ++

  var x = new state()
  x.set_expiry(0)
  x.txt = txt, x.key_required = true
  x.clear_admissible_keys()
  for(var i in keys){
    x.add_admissible_key(keys[i])
  }
  x.type = 'feedback', x.trial_id = 0, x.task_id = my_task_id
}

/* list as many countries as possible during e.g., a 3-minute period (default, 30s)
  20170515: default for delay_time used to be 30000. Today we added the end on <esc>
  key feature
*/
function delay_task(txt, delay_time=0, isi_=500){
  var my_task_id = next_task_id ++, isi = parseInt(isi_)

  /* if ISI was set, prefix with a "blank" slide */
  if(isi > 0){
    var x = new state()
    x.set_expiry(isi)
    x.type = 'isi', x.wrd_stim = "", x.trial_id = 0, x.task_id = my_task_id
    x.clear_admissible_keys()
    x.key_expiry = false
  }

  var y = instructions(txt)

  /* time [mS] */
  var x = new state()
  x.set_expiry(delay_time)
  x.key_expiry = false, x.txt = '', x.type = 'delay', x.trial_id = 0, x.task_id = my_task_id
  if(delay_time <= 0){
    x.clear_admissible_keys()
    x.add_admissible_key(27)
    console.log('admissible_keys', x.admissible_keys)
  }
  return x
}

```

```
63 /* study phase, formerly known as orientation task: multiple 'trials' / events occur here.. random
      selection of inputs... (for the test phase, the random selection is shuffled back into the pool)..
      */
64 function study_phase(my_pool, isi=0, time_limit=0, extra_feedback=false, extra_feedback_message="",
      extra_feedback_keys=[]){
65
66   /* the above constructor (same with test_phase) can accept either a single stimulus pool (pool()),
67     or an array of stimulus pools (pool()) */
68   var my_pools = []
69   if(my_pool.is_pool){
70     my_pools.push(my_pool)
71   }else{
72     my_pools = my_pool
73   }
74
75   var trial_index = -1, my_task_id = next_task_id++
76   this.ctx = ctx, this.p = my_pools, this.pool_ids = new Array()
77
78   /* for study phase, selection is built from combination of all selection_n arrays, from each of the
        supplied pools */
79   var my_selection = new Array()
80   for(var a_pool in my_pools){
81     var my_pool = my_pools[a_pool]
82     this.pool_ids.push(my_pool.pool_id)
83     for(var i  in my_pool.selection_n){
84       var extra_feedback_this_slide = false
85       if(extra_feedback != false){
86         if(0 == i % parseInt(extra_feedback)){
87           extra_feedback_this_slide = true
88         }
89       }
90       my_selection.push([my_pool.selection_n[i], my_pool.pool_id, extra_feedback_this_slide])
91     }
92   }
93
94   /* randomize the order of the array */
95   shuffle(my_selection, true)
96
97   for(var selection_ind in my_selection){
98
99     /* increment the trial-index counter */
100    ++ trial_index
101
102    var a_selection = my_selection[selection_ind]
103
104    /* data (word or image) assigned to "trial" */
105    var data = a_selection[0], p_id = a_selection[1], extra_feedback_this_slide = a_selection[2]
106
107    /* if ISI was set, prefix with a "blank" slide */
108    if(isi > 0){
109      var x = new state()
110      x.set_expiry(isi)
111      x.type = 'isi', x.wrd_stim = "", x.trial_id = trial_index, x.task_id = my_task_id
112      x.set_pool_id(my_pool.pool_id)
113      x.clear_admissible_keys()
114      x.key_expiry = false
115    }
116
117    /* initialize generic "trial" object for each case */
118    var x = new state()
119    if(time_limit <= 0){
120      x.set_expiry(0)
121      x.key_required = false
122    }else{
123      x.set_expiry(time_limit)
124      x.key_required = false
125    }
126
```

```
127     /* discern by image or word, respectively */
128     if( typeof(data) === 'object'){
129       x.img_stim = data
130     }else if(typeof(data) === 'string'){
131       x.wrd_stim = data
132     }
133     x.type = 'study_phase', x.trial_id = trial_index, x.task_id = my_task_id
134     x.set_pool_id(p_id)
135     if(extra_feedback_this_slide){
136       var x_f = feedback(extra_feedback_message, extra_feedback_keys)
137     }
138   }
139   return this
140 }
141
142 /* test phase, formerly known as recognition task - for this phase,
143 the random selection is shuffled back into the pool -- all elements
144 from the pool are shown (feedback is recorded).. */
145 function test_phase(my_pool, isi=0, time_limit=0, extra_feedback=false, extra_feedback_message="",
        extra_feedback_keys=[]){
146   var my_pools = []
147   if(my_pool.is_pool){
148     my_pools.push(my_pool)
149   }else{
150     my_pools = my_pool
151   }
152
153   var trial_index = -1, my_task_id = next_task_id++
154   this.ctx = ctx, this.p = my_pools, this.pool_ids = new Array()
155
156   /* for test phase, selection is built from combination of all selection_m arrays, from each of the
          supplied pools */
157   var my_selection = new Array()
158   for(var a_pool in my_pools){
159     var my_pool = my_pools[a_pool]
160     this.pool_ids.push(my_pool.pool_id)
161     var trial_index = -1, shuffled_data = my_pool.reshuffle(), shuffled = shuffled_data[0], deja_vu =
          shuffled_data[1]
162     for(var i  in shuffled){
163       var extra_feedback_this_slide = false
164       if(extra_feedback != false){
165         if(0 == i % parseInt(extra_feedback)){
166           extra_feedback_this_slide = true
167         }
168       }
169       my_selection.push([shuffled[i], my_pool.pool_id, deja_vu[i], extra_feedback_this_slide])
170     }
171   }
172   shuffle(my_selection, true)
173
174   for(var selection_ind in my_selection){
175     ++ trial_index
176
177     var a_selection = my_selection[selection_ind]
178     var data = a_selection[0], p_id = a_selection[1], deja = a_selection[2], extra_feedback_this_slide
          = a_selection[3]
179
180     /* if ISI was set, prefix with a "blank" slide */
181     if(isi > 0){
182       var x = new state()
183       x.set_expiry(isi)
184       x.type = 'isi', x.wrd_stim = "", x.trial_id = trial_index, x.task_id = my_task_id
185       x.set_pool_id(p_id)
186       x.clear_admissible_keys()
187       x.key_expiry = false
188     }
189
190     var x = new state()
```

```
191     x.key_required = true
192     if(time_limit <= 0){
193       x.set_expiry(0)
194     }else{
195       x.set_expiry(time_limit)
196     }
197
198     /* record within the object: do we have deja-vu? */
199     x.deja = deja
200
201     /* word or image? */
202     if( typeof(data) === 'object'){
203       x.img_stim = data
204     }else if(typeof(data) ==='string'){
205       x.wrd_stim = data
206     }
207     x.type = 'test_phase', x.trial_id = trial_index, x.task_id = my_task_id
208     x.set_pool_id(p_id)
209
210     if(extra_feedback_this_slide){
211       var x_f = feedback(extra_feedback_message, extra_feedback_keys)
212     }
213   }
214   var m = 'Thank you for completing this section. ', end = instructions(m)
215
216   end.action = function(me){
217     var msg = m + 'Your score: ' + ctx.questions_correct.toString() + '/' + ctx.questions_total.
          toString() + ". Please press any key."
218     me.txt = msg
219   }
220   return this
221 }
```

5.8. **text.js.** This file includes a routine for rendering text, including wrapping, to allow rendering that is somewhat adaptive to the shape of the screen.

```
/* wrap text around a window region— via ashblue */
function wrap_text(s, ctx, start_y=0){
  var myX = 10, myY = 50, line = '', lines = [], w = ctx.w(), h = ctx.h(), line_test = '', words0 = s.
      split(' '), font_size = ctx.font_size
  ctx.font = font_size +'px Arial'
  var words = new Array()
  for(var i = 0; i < words0.length; i++ ){
    var w = words0[i]
    ws = w.split('\n')
    words.push(ws[0])
    if(ws.length > 1){
      for(var j = 1; j < ws.length; j++){
        words.push('\n')
        if(ws[j] != ""){
          words.push(ws[j])
        }
      }
    }
  }

  w = ctx.w()

  /* place words one by one */
  for(var j = 0; j < words.length; j++){
      if(words[j] == "\n"){
        myY = lines.length * font_size + font_size
        lines.push({text: line, height: myY})
        line = ''
        continue
      }

      line_test = line + words[j] + ' '

      /* wrap if over the edge */
      if(ctx.measureText(line_test).width > w){
        myY = lines.length * font_size + font_size
        lines.push({text: line, height: myY})
        line = words[j] + ' '
      }else{
        line = line_test
      }

  }

  /* catch last line if something left over */
  if(line.length > 0){
    current_y = lines.length * font_size + font_size
    lines.push({text: line.trim(), height: current_y})
  }

  /* plot text */
  for(var j = 0, len = lines.length; j < len; j++){
    ctx.fillText(lines[j].text, 0, lines[j].height + start_y)
  }
}

/* write centred text */
function centre_text(s){
  var font_size = ctx.font_size, textString = s
  ctx.font = 30 + 'px Arial'
  textWidth = ctx.measureText(textString).width
  ctx.fillText(textString, (canvas.width / 2) - (textWidth / 2), canvas.height / 2)
}
```

5.9. **util.js.** This file includes various utilities for aspects such as pseudo-random number generation, transferring data over the internet, and so on.

```
/* cr34t3 a c4nv4s wh3r3 th3 m4g1c h4pp3ns */
var canvas = document.createElement('canvas')
document.body.appendChild(canvas)

/* get date and time */
function date_time(){
  return new Date()
}

/* seed for rand() below */
var seed = 5

var get_seconds = function(){
  var d = new Date()

  /* return an epoch time (S) */
  return d.getMilliseconds()
}

var mutable_seed = get_seconds()

/*random-number generator http://indiegamr.com/generate-repeatable-random-numbers-in-js/ : initial seed
    .. in order to work 'Math.seed' must NOT be undefined, so in any case, you HAVE to provide a Math.
    seed */
function rand(max, min, mutable=false){
    max = max || 1, min = min || 0
  if(mutable){
    mutable_seed = (mutable_seed * 9301 + 49297) % 233280
    return min + (mutable_seed / 233280) * (max - min)
  }else{
    seed = (seed * 9301 + 49297) % 233280
    return min + (seed / 233280) * (max - min)
  }
}

/* Shuffle array in place, via http://stackoverflow.com/questions/6274339/how-can-i-shuffle-an-array
 * @param {Array} a items The array containing the items.

  setting the parameter "mutable" to true, makes random selections that will change between runs. */
function shuffle(a, mutable=false) {
  var j, x, i
  for(i = a.length; i; i--){

    /* use our seeded random number generator, so we get the same results every time */
    j = Math.floor(rand(null, null, mutable) * (1. * i))  /* j = Math.floor(Math.random() * i) */
    x = a[i - 1]
    a[i - 1] = a[j]
    a[j] = x
  }
}

/* pad to length n (with 0's on the left) */
function pad_n(x, n){
  var s = parseInt(trim(x)).toString(), m = s.length, d = n - m
  if(d > 0){
    s += '0'.repeat(d)
  }
  return s
}

/* via stackoverflow.com/users/4321/jw */
function get_keys(dictionary){

  /* keys recursive */
  var keys = []
```

```
64
65   /* filter for direct ancestors */
66   for(var key in dictionary){
67     if(dictionary.hasOwnProperty(key)){
68       keys.push(key)
69     }
70   }
71   return keys
72 }
73
74 /* draw an image */
75 function draw_img(x, ctx){
76   var cf = 4 * ctx.font_size
77   var h = ctx.h() - cf, w = ctx.w()
78   var lw = x.width, lh = x.height
79   var sf = Math.min(w, h) / Math.max(lw, lh)
80   var a = (w - lw * sf) / 2, b = (h - lh * sf) / 2
81   var c = lw * sf, d = lh * sf, df =  (-20 + cf / 2)
82   ctx.drawImage(x, a, b + df, c, d)
83 }
84
85 /* write the above to a standardized format */
86 function parse_date_time(today){
87
88   /* most significant units first */
89   var bits = [today.getFullYear(),
90               today.getMonth() + 1,
91               today.getDate(),
92               today.getHours(),
93               today.getMinutes(),
94               today.getSeconds(),
95               today.getMilliseconds()]
96
97   /* pad with zeros */
98   for(var i = 0; i < bits.length; i++){
99     var n_pad = 2
100    if(i == 0){
101      n_pad = 4
102    }
103    if(i == 6){
104      n_pad = 3
105    }
106    var bts = bits[i].toString()
107    bits[i] = pad_n(bts, n_pad)
108  }
109  return(bits.join(':'))
110 }
111
112 /* "faster trim" via blog.stevenlevithan.com */
113 function trim(s){
114   return s.toString().replace(/^\s\s*/,'').replace(/\s\s*$/,'')
115 }
116
117 /* send text format data (string s) via XML to receive script at url (string): xml-receive_script_url
      */
118 function xml_send(s, xml_receive_script_url){
119
120   /* xml http request object */
121   var xhr = (window.XMLHttpRequest) ? new XMLHttpRequest() : new activeXObject("Microsoft.XMLHTTP")
122   var data = new FormData()
123   data.append("data", s)
124   xhr.open('post', xml_receive_script_url, true)
125   xhr.send(data)
126 }
```

6. Source Code: Server Side

6.1. **xml-receive.py.** The folder data/ in the directory structure, relative to the installation folder: if it doesn't yet exist, the server-side python code should create it (provided that the appropriate permissions are available to do so).

```
#!/usr/bin/python
''' server-side python-CGI script to receive text data sent over
the internet by the client-side function util.js::xml_send()'''
import os
import cgi
import uuid
import datetime

# create /data folder if it does not yet exist
dat_f = os.getcwd() + '/data/'
if not os.path.exists(dat_f):
    os.mkdir(dat_f)

# retrieve CGI form data
dat = None
try:
    dat = str(cgi.FieldStorage().getvalue('data'))
except:
    pass

# write the data to file in the data/ folder
if dat:
    fn = dat_f + str(datetime.datetime.now().isoformat())
    open(fn + '_' + str(uuid.uuid4().hex) + '.txt', 'wb').write(dat)
```

7. Ideas For Possible Future Improvements

A short point-form list of possible improvements to the software:

- Finish drag-and drop implementation, that
  - prevents programmatic errors;
  - does not allow invalid experiments to be constructed; and
  - removes any technicality from the process of coding an experiment.
- Smarter image loading:
  - Automagically detect available images from folder, and
  - modify program to allow administrator to upload images with file names that aren't required to follow the numbered format.
- Support for mobile devices, or possibly:
  - show a warning message for un-supported devices.
- Accomplish API-level integration with "Mechanical Turk" for detailed/complex systematic/automated deployment of surveys for use in Recognition Memory experiments.