lab.js Experiment Builder Guide

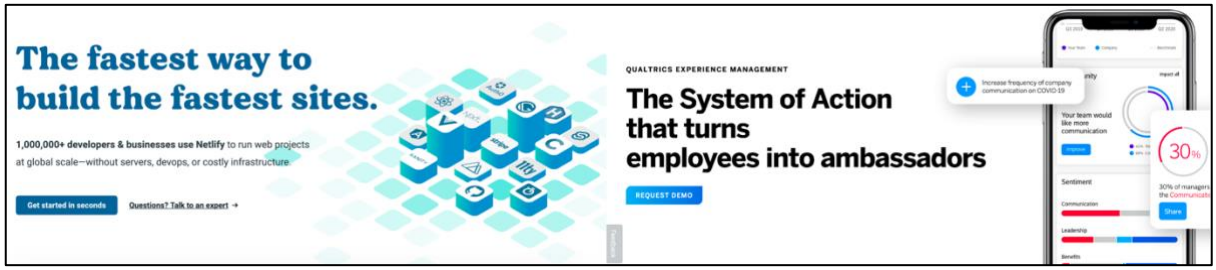
Shannon Rafferty & Glenn Williams

2020-10-21 (last updated: 2020-10-27)

lab.js is a free, open, online experiment builder that allows you to create studies that work in the browser. Experiments built in lab.js can work both online and offline.

Experiments can be built using the visual interface, with code (i.e. using a mix of html, CSS, and JavaScript), or a mix of the two.

This guide focuses on building a study using the visual interface, which requires no programming, and getting your study up and running online through Qualtrics and Netlify.



Experiment Background

We will build a study based on Glenberg, A. M. & Kaschak, M. P. (2002). Grounding language in action. *Psychonomic Bulletin & Review, 9*(3), 558-565.

This study is an influential one in the area of embodied cognition. Some approaches to embodied cognition assume that language comprehension is not an amodal and abstract task, but instead we add meaning to language by simulating the events being described. For example, when we read the sentence “hang the coat on the hanger” we understand this sentence and judge it as sensible by simulating the actions being described. However, when reading “hang the coat on the teacup” comprehension is incomplete or the sentence is judged as being nonsensical as we cannot properly simulate an impossible action. More extremely, when we read action verbs such as “lick” and “kick” we see activity in the sensorimotor regions associated with these actions (i.e. activity in the brain for the tongue for “lick” and the foot for “kick”). It is thought that this activation adds meaning to language which improves comprehension. In short, language processing may be embodied in the sense that it relies on our prior and simulated physical experiences (Hauk, Johnsrude, & Pulvermüller, 2004).

This particular study shows evidence for the action-sentence compatibility effect (ACE). Briefly, it is thought that language developed to coordinate action. The ACE proposes that language processing primes action, such that any sentence that implies an action should facilitate similar movements. In this study, Glenberg and Kaschak asked participants to judge whether sentences make sense. Participants indicated that sentences made sense by pressing a button away from their body, and that sentences did not make sense by pressing a button towards their body. They showed that when reading a sentence such as “close the drawer” – which implies movement away from the body – participants were faster to make the sensibility judgment than with “open the drawer” (which implies movement towards the body).

However, recent articles call this landmark finding into question, with a host of replications showing no evidence for this action-sentence compatibility effect (Papesh, 2015).

Your Task

Build a study in the vein of Glenberg and Kaschak (2002) and introduce a new manipulation. This manipulation can be anything you like as long as it is justified by the literature. Ideally, this should result in at least a 2 × 2 design that will allow for a two-way analysis of variance.

The original study explored main effects of response direction (yes is near vs. yes is far) and sentence direction (movement towards vs. movement away), along with an effect of sentence type (e.g. imperative, concrete transfer, and abstract transfer). However, the Papesh (2015) replication looked at

The experiment will display a fixation screen prior to individual trials which will then present participants with a sentence on screen which is either (a) sensible, or (b) nonsense, and which in case (a) implies either a movement (i) towards the participant or (ii) away from the participant. Participants will indicate whether sentences make sense by pressing a button towards or away from themselves. We will design this study so that it has two phases whereby participants indicate a sensible sentence using each direction of movement respectively. We will also counterbalance the items using various lists such that confounding variables are controlled.

We will provide you with the initial items and a guide to the basic study, but any additional manipulations you introduce will be made by yourself/in your groups.

Minor Details

The version of the study we produce will show participants:

- A screen displaying a fixation dot to attract participants' attention to the centre of the screen before seeing a sentence. To progress to the next screen, participants must press the Space bar on the keyboard.
- A screen showing a short sentence. To progress to the next screen, participants must make a choice as to whether the sentence is sensible or not using the w or s key on the keyboard.
- Before these screens, participants will see instructions on how to respond to sensible or nonsense sentences using the **w and s keys on the keyboard** and how to progress from a fixation dot by pressing the Space bar.

Participants will take part in a short practice phase of four fixed items, two of which are sensible and two of which are nonsense sentences.

There are four lists of experimental items made up of sensible and nonsense sentences. These four lists counterbalance the direction of transfer of the sentence (i.e. towards or away from the participant) and whether or not a correct response requires a w or s key-press. An example of the four versions of item 1 is presented below:

I. You sold the land to Hakeema. (sensible response is w)
II. Hakeema sold the land to you. (sensible response is w)
III. You sold the land to Hakeema. (sensible response is s)
IV. Hakeema sold the land to you. (sensible response is s)

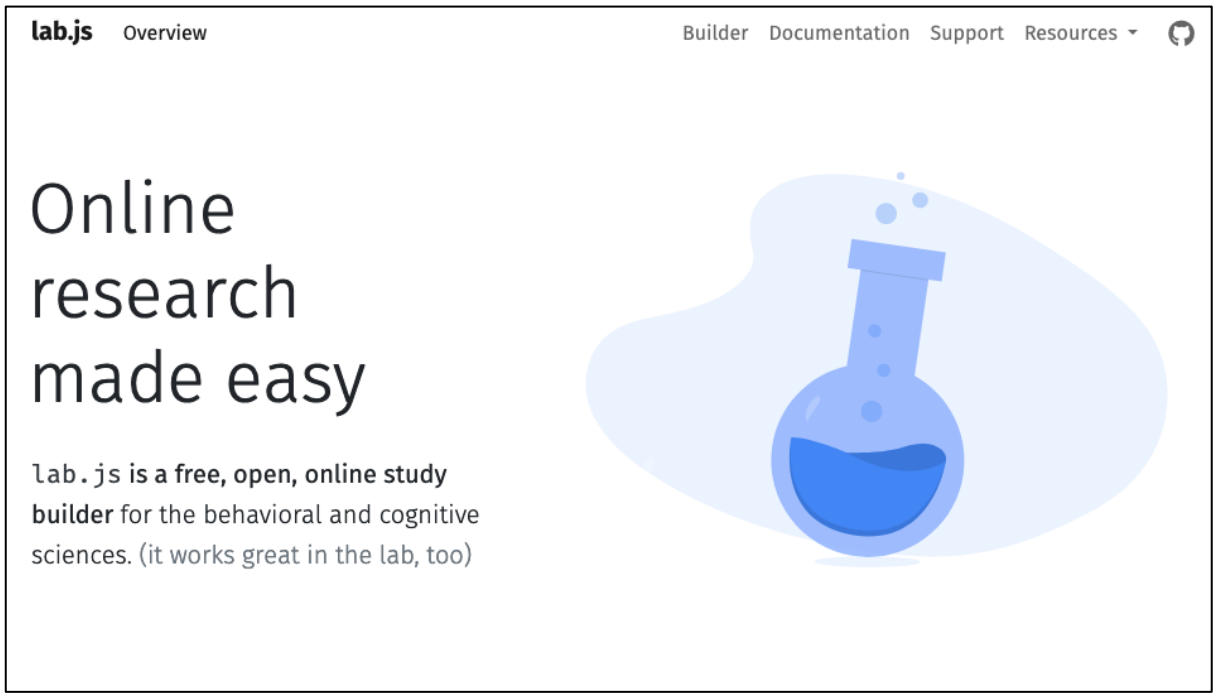
Participants will thus see one version of each item. The items will be split in two blocks and presented to participants separately. In one block, a sensible sentence will require a w key-press,

and in the other block a sensible sentence will require an s key-press. Across items, participants will see both sensible and nonsense sentences. Thus, participants will see all conditions (i.e. this is a within-subjects design), but they will not see all conditions for all items (i.e. this is a between-items design).

Building the Experiment

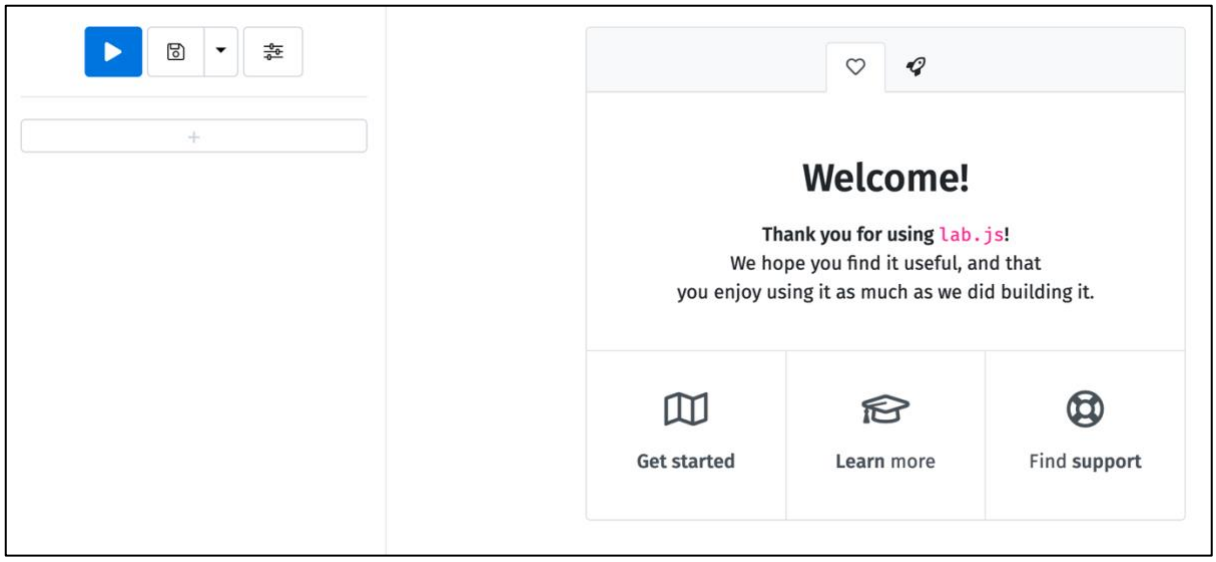
Getting Started with lab.js

Go to https://lab.js.org/.



This landing page has a number of links. **Builder** takes you to the visual interface for the builder, Documentation takes you to documentation about the program (i.e. help and examples), Support links you to the lab.js Slack channel (for posting questions and answers with other users), and Resources outlines details such as publications outlining the timing and accuracy of lab.js, amongst other information.

Click **Builder**. This will open a page that looks like this:



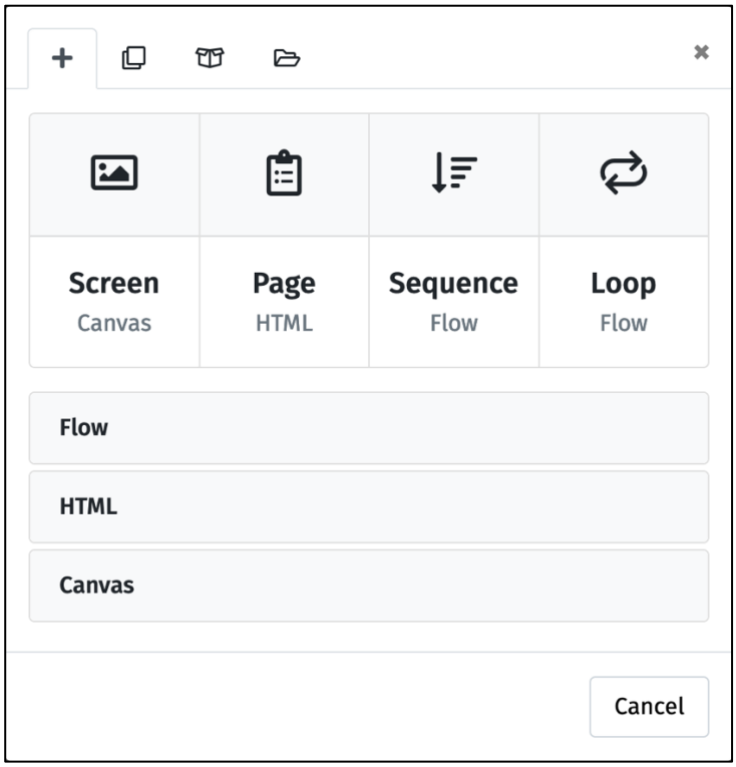
Here, you can find guides on getting started in lab.js and learning more about how it works.

In **Get Started**, there is a tutorial you can work through with videos. But we'll follow this guide which helps you to build an experiment closely tied to your practical assessment for PSY260. In **Learn More**, you can browse some example studies, or go straight to the **rocket tab** at the top of this page on the right-hand side. This allows you to see how certain tasks (e.g. Stroop tasks, Visual Search tasks etc.) can be built in lab.js. It may be a good idea to check these out later on to see how you can implement different features of tasks.

What do the symbols in the left-hand pane do?

- The blue, **play** button (far left) allows you to preview your website. This is useless right now, but it's worth using this intermittently as you develop your study to check it is building properly.
- The **save icon** (middle) allows you to save you work. Because you don't have an account to build the site, it's important to save during building and before you close the browser. This is especially important if you intend to work across multiple computers. This saves your work as a **.json file for the builder**, which can be loaded back up for further editing by using the **down arrow next to the save icon**.
- The **down arrow** allows you to open an experiment from a **saved .json file**, start a new study, and **export your study to an external format** for offline data collection (i.e. in the lab) or various formats for online data collection. We'll export ours for **Generic Survey Tools** when we're done so we can embed the study in Qualtrics.
- The **settings icon** (far right) allows you to define some additional settings for the study, split into **study information**, any **static files** you want to upload (e.g. images/sound/data), and any additional **custom HTML and CSS files** which will change the look and behaviour of the study. At the very least, adding a **title and description** to the study information will help you to keep track of your work.

To build our experiment, we'll use the tabs on the left-hand side of the screen for now. Use the **plus sign** to add a new component to our study. You should see a window that looks like this:



Again, this is split up into tabs as follows:

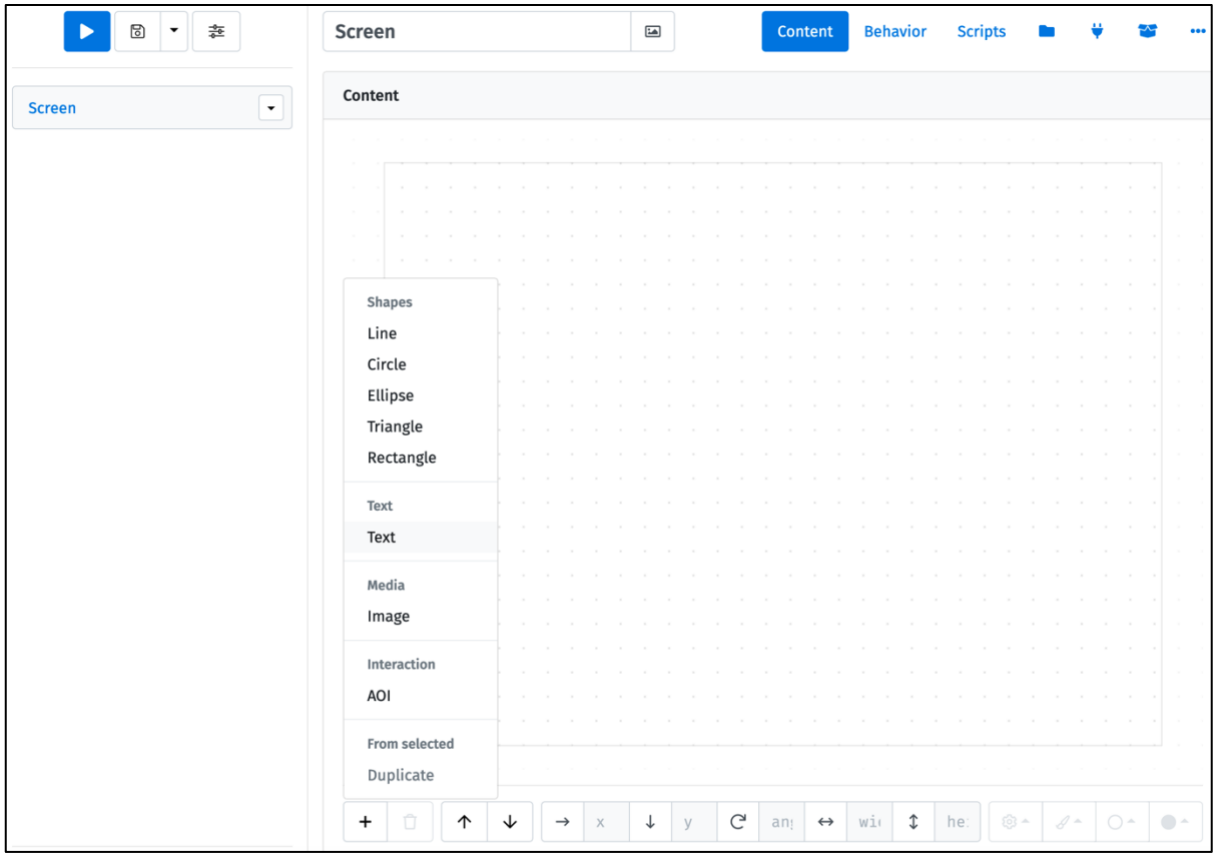
- **Add** (top left): Create a component from scratch. This can be a screen, page, sequence, or loop. More on these later.
- **Copy**: Copy an existing component and add it to the study.
- **Templates**: Add a component to the study from a template. This is useful for saving time on creating everything from scratch. At present, these include demographics forms, fixation dots, and headphone screening amongst other useful templates.
- **Component from file**: If you have a component saved to file on your computer, you can load it up here.

Most of the common components you can add are outlined with large icons in this window. These include:

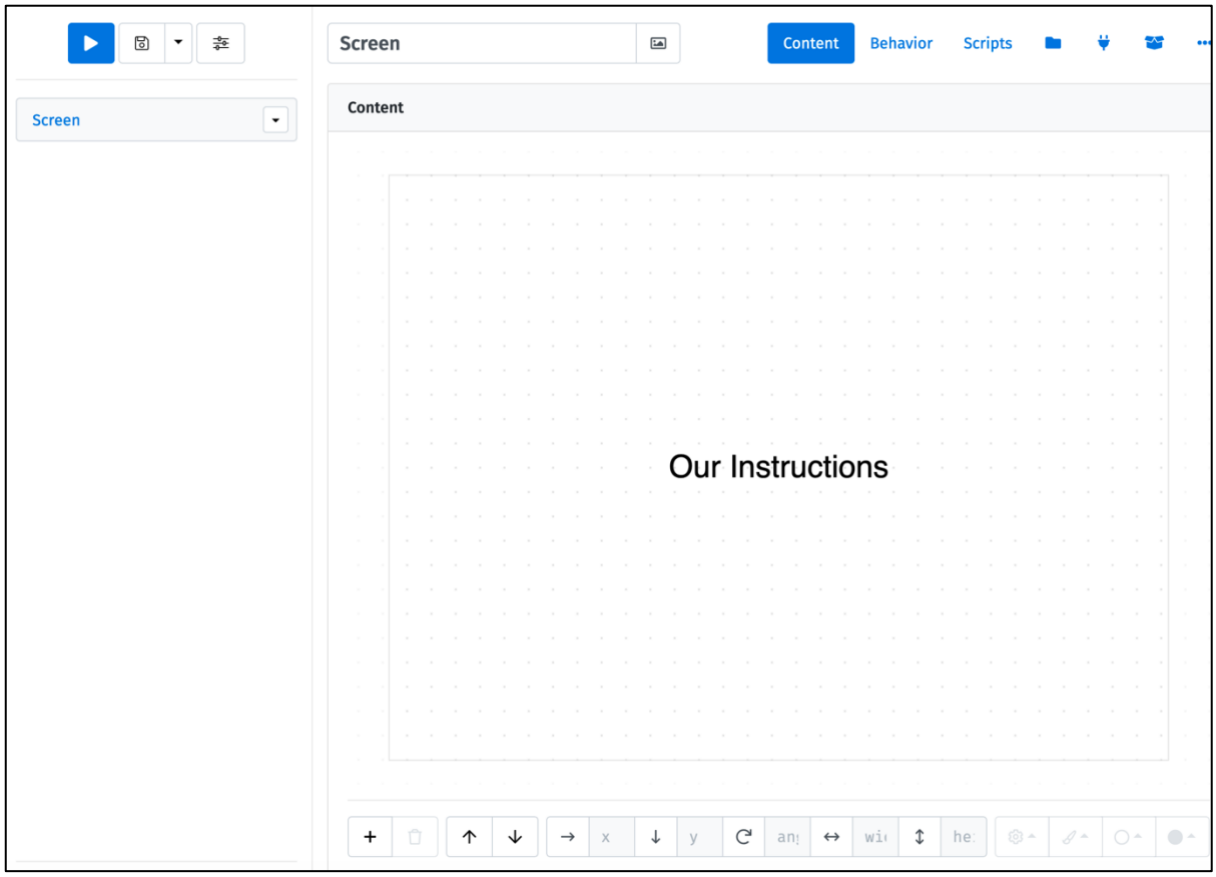
- **Screen**: Most study materials are presented best using a screen. This relies on the HTML canvas to present items to the participant. This is one of the fastest methods of updating stimuli, so has good timing accuracy which is often important for reaction time studies.
- **Page**: Other elements for data capture, such as a form for completing a question, are best presented as an HTML page. Additionally, if you want to copy some existing HTML code to present a custom page, this is your best option.
- **Sequence**: This determines how multiple screens are grouped together. This allows you to control the flow of the study as participants progress through the experiment.
- **Loop**: This determines how screens should change for a given trial. Imagine we have 60 trials in a study which simply displays a different word on screen to the participant. We could insert 60 pages. But to save time (and to allow randomisation of trials) we can instead define the words that should be inserted on a page and tell the study to **loop through these**, displaying them sequentially in the same way for every trial.

Building the Experiment

1. **Click the plus sign** (+) in the builder interface. Choose a **Screen (Canvas)** element. This will allow us to add stimuli to our study.
2. To add content to our study, use the **+ symbol** in the bottom left corner of the canvas. This will bring up some options. For now, we're going to add some instructions to the study. To do this, select the **Text** option in the menu. You can see this in the image below:



3. Some text will appear in the centre of the canvas. Click this to edit the text. Add any instructions for now, but feel free to add to or change these as you develop a better idea for the study. For now, we can just add some placeholder text (e.g. "Our Instructions").



By default the x and y location (i.e. horizontal and vertical location) for this text is set to 0 and 0. This is the centre of the screen. Try changing these values to change the location of the text if you please. You also have the option of rotating the text and changing the formatting of it (e.g. underlining, italics, colour etc.).

Notice that that our new component has a name in the left-hand side of the screen. This controls the flow of the study. Any added components will get a name here. **Components at the beginning of the list come first in the study**.

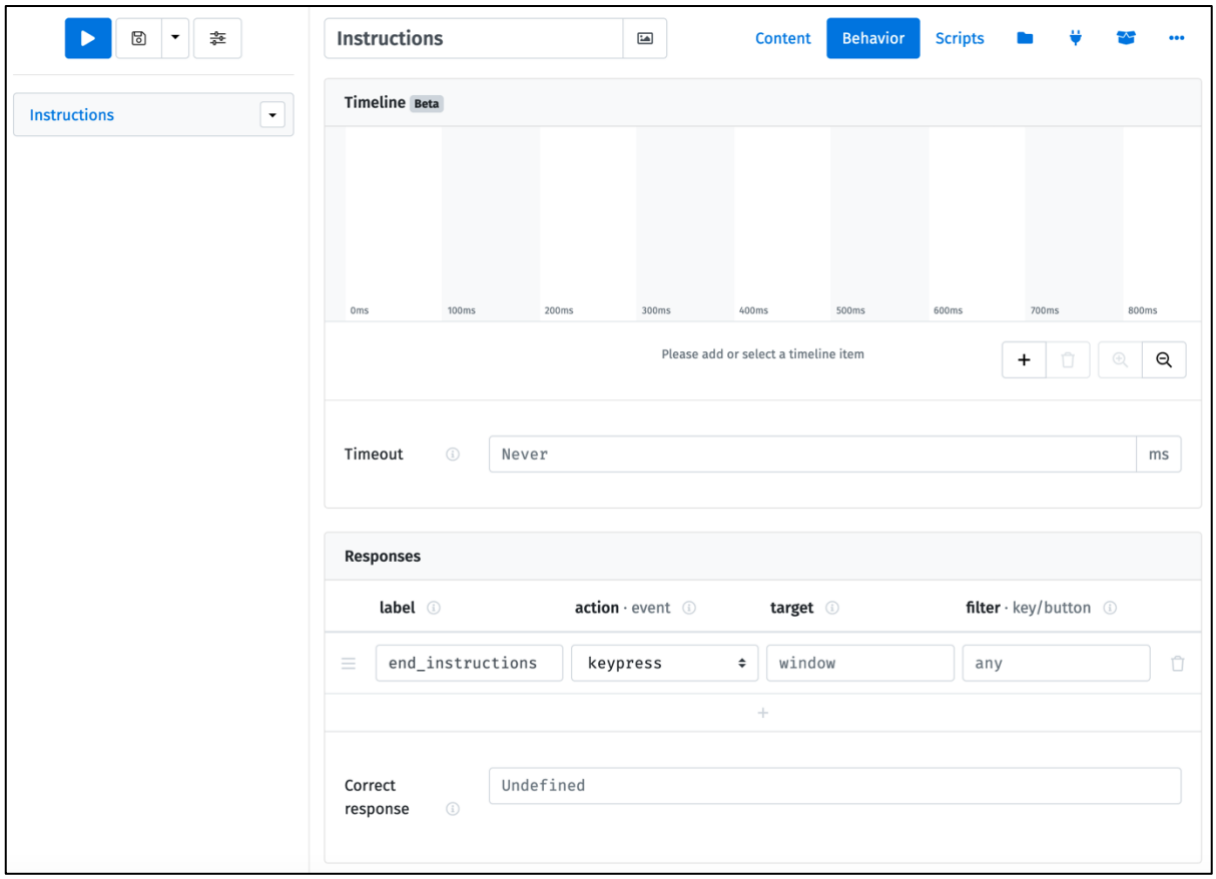
4. Click the name of the screen just above the Content screen and change this to **Instructions**. This allows us to keep track of which screen corresponds to which part of the study.

There are also three tabs at the top of the study now:

- The first, which we are now controls the **content** of a screen, e.g. its appearance.
- The next tab controls its **behaviour**, e.g. when the trial should end or whether any responses can be captured.
- The final tab controls any whether any **additional scripts** can be executed on the screen. This allows us to use our own custom JavaScript code to perform any operations we'd like on the screen that aren't in the visual editor of lab.js. (For example, allowing us to draw on canvases etc.)

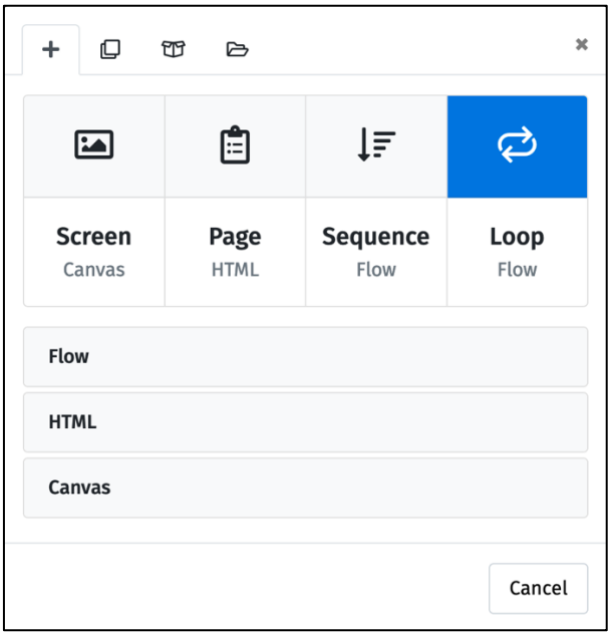
5. Right now, the instructions screen will be displayed forever for participants, with no way to move on in the study. This is because we haven't set any behaviour for when to end the screen. Go to the behaviour tab above the canvas and change go to the **Responses** table. Click on Action cell which currently displays **none (inactive)** and change it to **keypress**. Leave

filter as any, or change this to a specific key you'd like participants to press to end the screen (e.g. **Space** for the space bar). Finally, change the label to **end_instructions**. This allows us to keep track of what each action does in our study. Your screen should now look like this:

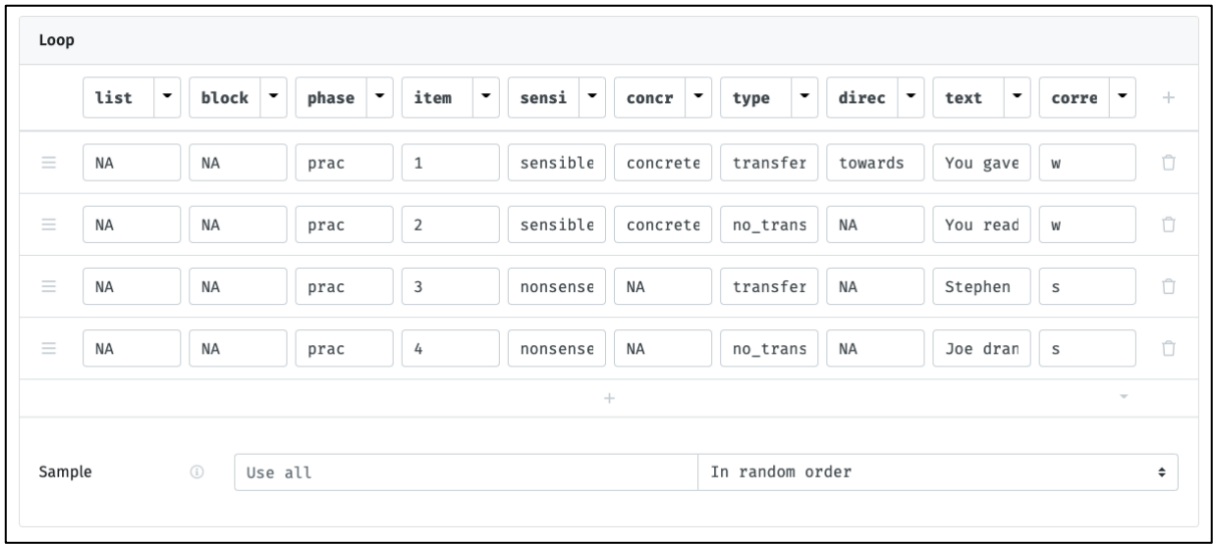


Next, we will make some practice trials for our participants. This allows participants to get a feel for the study first before we collect any data. We usually add the option of asking the experimenter questions after the practice (for lab-based studies) or potentially the option to see the instructions again before progressing to the full, experimental block of trials. For now, we'll just add a practice phase.

6. Hover your mouse below the instructions component in the left-hand bar. This should bring up the plus symbol again. Click this and add a loop component. This will allow us to define our practice trials and later allow us to loop through these trials in a Canvas element (e.g. changing the text in the centre of the page without adding unique Canvases per trial).



7. We will now create some trials using the **table builder** in lab.js. For the content of the loop, change the titles of the initial headings in the table. Then use + next to the headings to add additional headings. Use the + below to add additional rows to the table. Complete the table as follows (please see text below for details):



This table should be populated with the following information:

- **list**: Which randomly generated list we will use. This is NA in this practice section as we will only have separate lists for the main phase of the study.
- **block**: Which block of the study this data belongs to. Again, this is NA as this is only relevant to the main phase of the study.
- **phase**: This is prac (i.e. practice) for every trial, indicating that these trials belong to the practice phase of the study. This will be different for the main phase of the study.
- **item**: The number for the item corresponding to the sentence for each trial.
- **sensibility**: Whether the sentence makes sense (sensible) or not (nonsense).

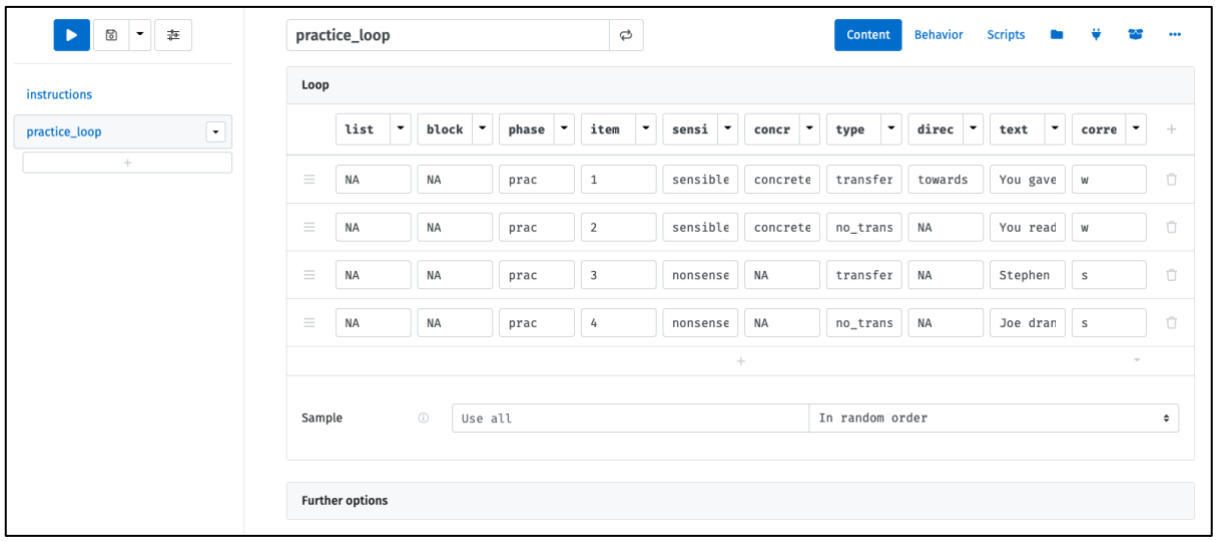
- **concreteness**: Whether the sentence describes a concrete object (concrete) or not (abstract). Examples might be e.g. a book (concrete) or freedom (abstract). Nonsense sentences are NA.
- **type**: Whether the sentence describes transfer of something from one character to another (tranfer) or not (no_transfer).
- **direction**: If a sentence describes transfer, does the described transfer occur from a character to you (towards) or from you to the character (away). If no transfer occurs, this is NA.
- **text**: The sentence to be displayed on screen to participants.
- **correct_response**: The correct key participants should press in deciding if the sentence makes sense or not. Our instructions indicate this is **w** for a sensible sentence and **s** for a nonsense sentence.

In the sentence column, you can add the following sentences in order.

- You gave the news article to Ben.
- You read the email with Ian.
- Stephen sent you the alpha.
- Joe drank the danger with you.

These will be the sentences participants will see on screen across four practice trials.

We will rename this loop **practice_loop**. When done, you should have a screen like this:

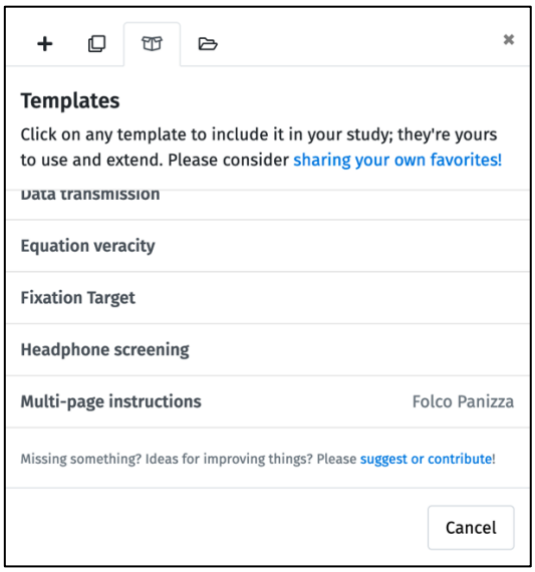


Next we will need to add a **sequence** to the loop. This will allow the experiment to loop through our files and present different screens to participants sequentially. Essentially, we will show the same screens to participants across all practice trials, but only change out the sentences in the middle of the screen (and what the correct button response by then should be).

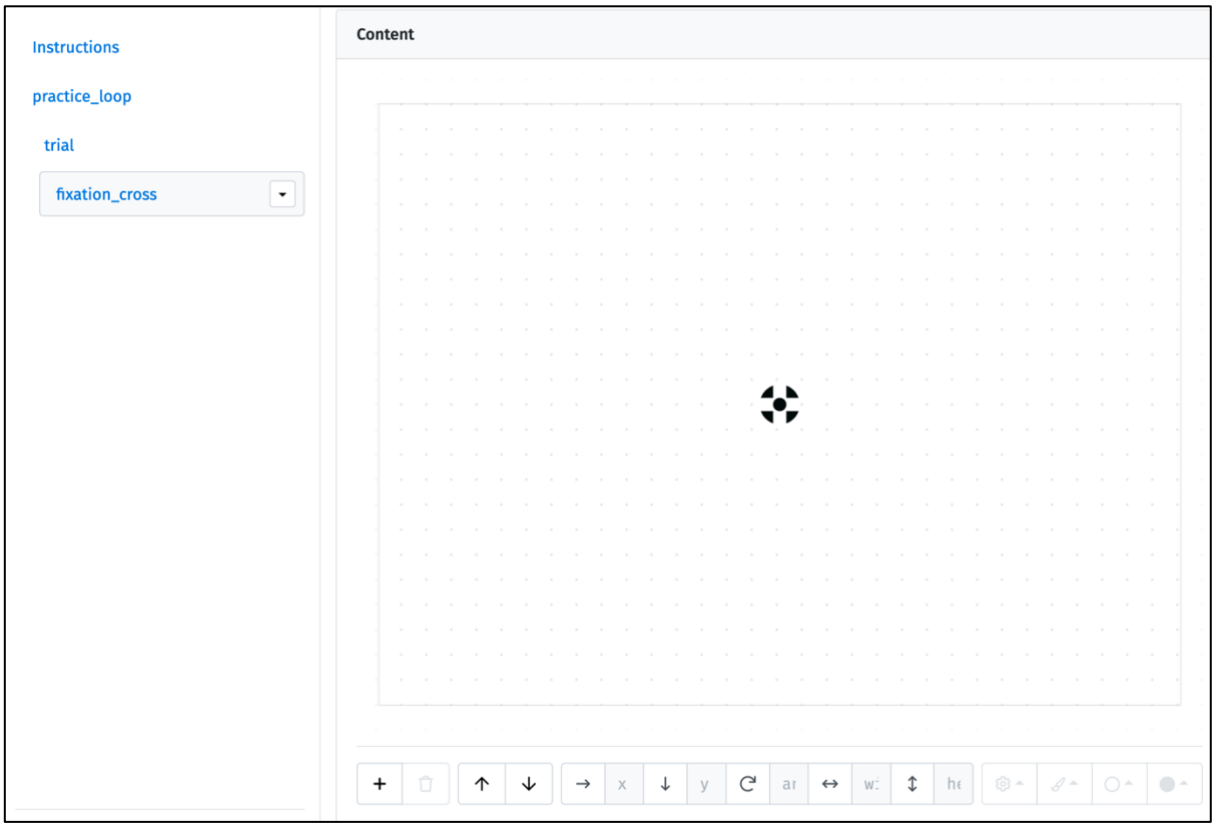
We will first show participants a **fixation cross**, and then the trial which just shows the sentence in the middle of the screen.

8. Click the plus symbol right under the **practice_loop** heading on the left-hand panel and click on **Sequence.** Rename the sequence to **trial**.
9. We can now add screens to the sequence. First, we want to show a fixation cross that ensures participants are focusing on the centre of the screen before the trial begins. Click

the plus below **trial** and choose the **Fixation Target** from the **Templates** (in the box tab; see below). Rename the Fixation cross to **fixation_cross** (for consistency with names).



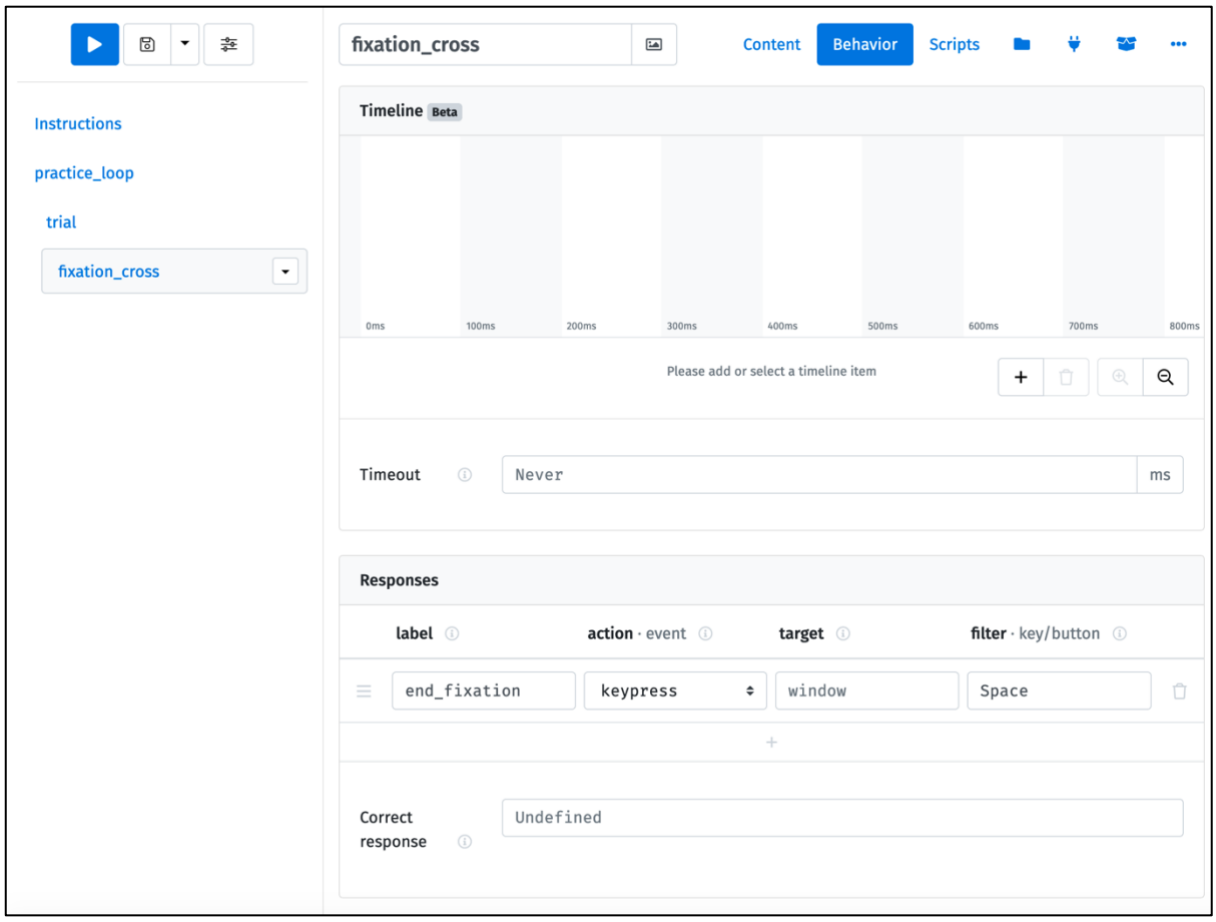
Your screen will look like this.



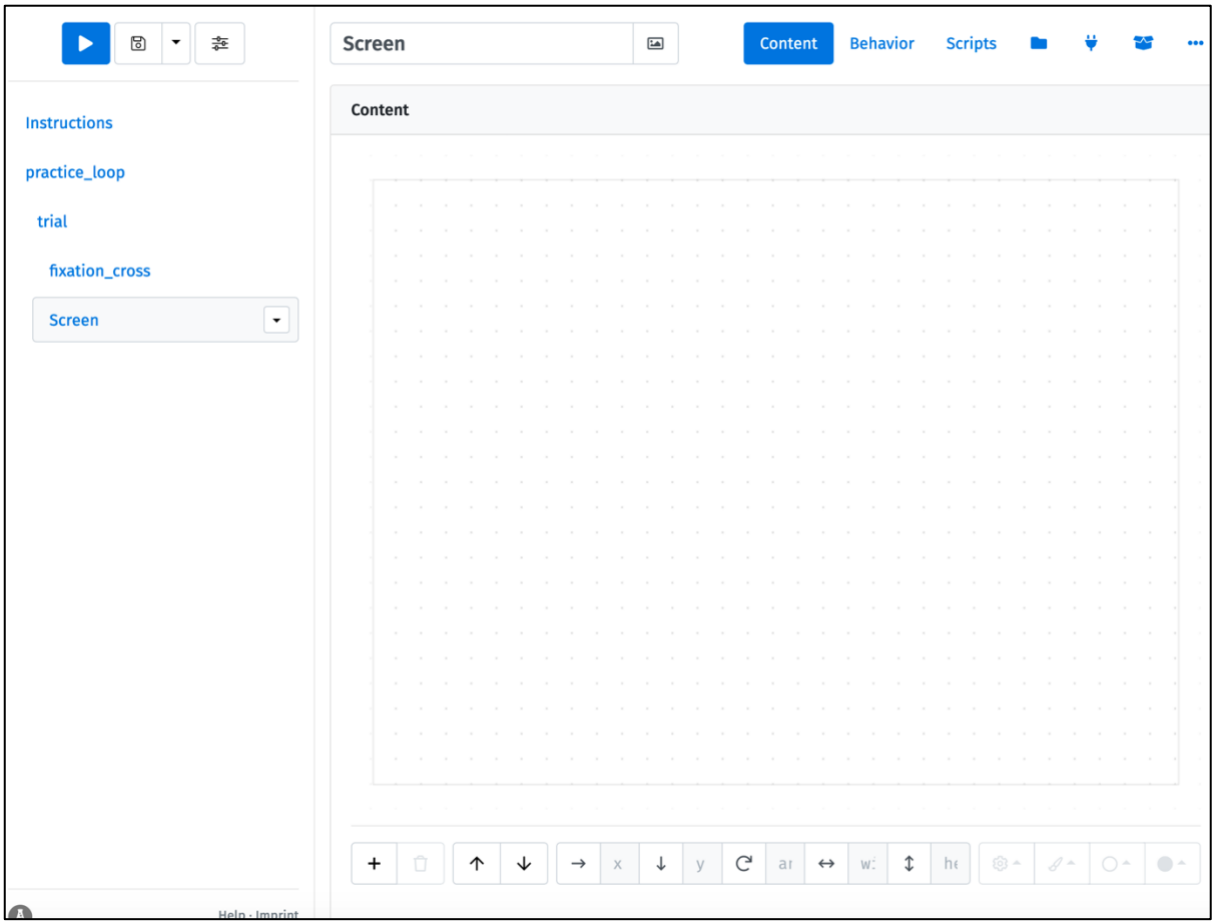
You can change the location and size of the fixation cross by changing the values in the boxes at the bottom, however the defaults we have here are fine.

10. Right now, the fixation cross will appear on the screen forever. So, we can do one of two things: (a) set a **timeout**, so that the cross is on screen for a fixed amount of time and then

the trial begins, or (b) allow a button press so participants only see the trial when they are ready. We'll use the latter method. To do this, go to **Behavior** and edit the Responses box. Here, change the label to **end_fixation**, change the **action** to **keypress**, and the **filter** to **Space**. This will allow people to end the fixation dot screen by pressing the Space bar only.



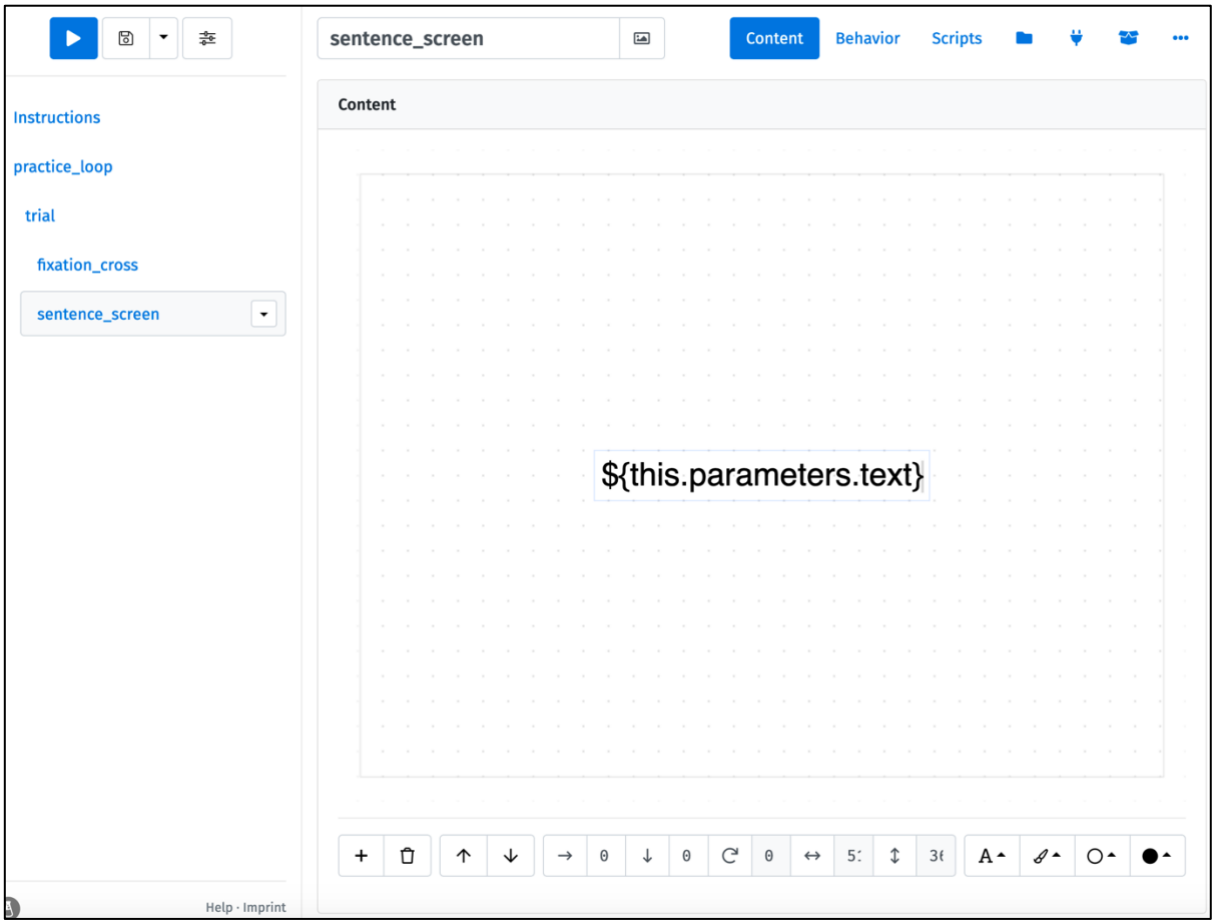
11. Next we will add the sentence screen. To do this, select the plus immediately under the **fixation_cross** heading in the left-hand panel. Please note you can add a screen outside the trial sequence, so be careful you don't do that here. Select the **Screen (Canvas)** element. You should have a screen that looks like this:



12. Rename the screen to **sentence_screen** for consistency. Then, choose the **plus** at the bottom left of the Canvas and choose the **Text** element. This will place some text in the middle of the Canvas. We could change this to a sentence, but right now this will be the same sentence for each iteration of the loop (i.e. showing the same text over the four practice trials). We want the text to change dynamically depending upon the trial. We need some code to achieve this.

13. Select the text box and change the text to **${this.parameters.text}**, the **${}** tells lab.js that we want to run some JavaScript code. Within this, we have written this.parameters.text.

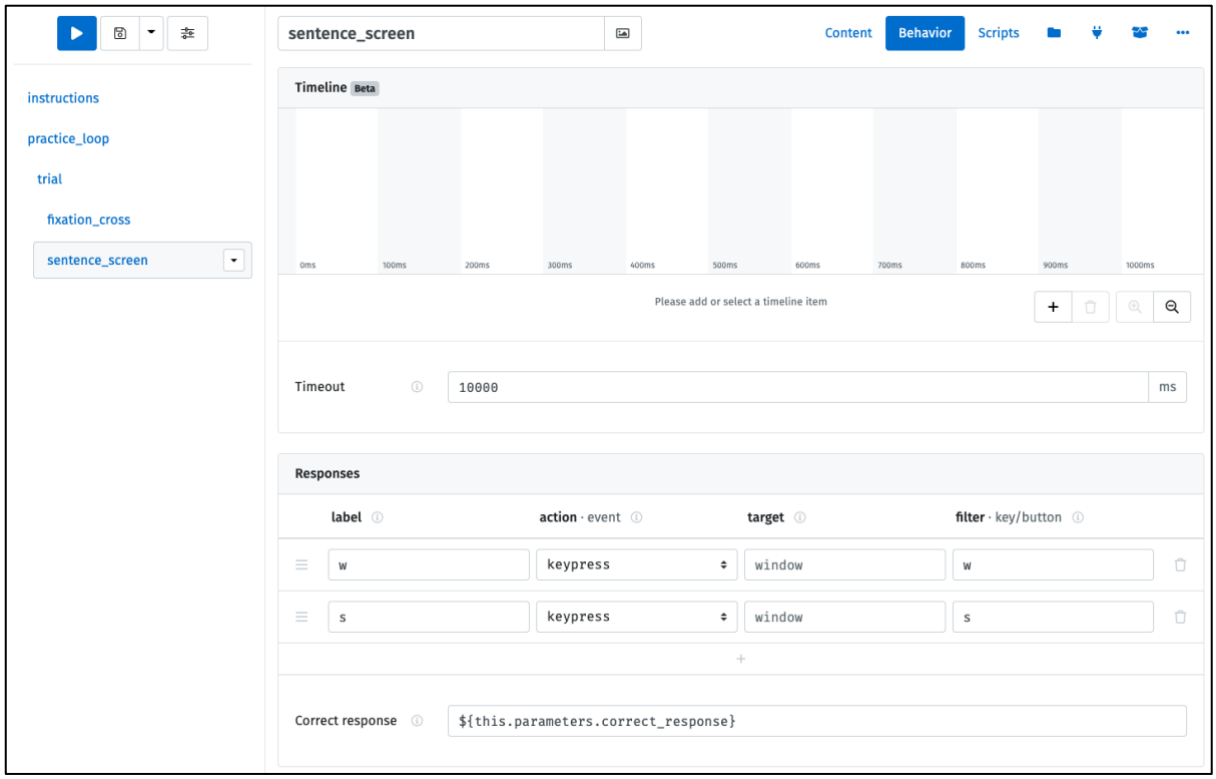
    Remember that we have a column called 'text' in our loop. Using the **this** keyword allows us to access information available to the loop. As we go through the loop, we will be able to access the row number in the table that matches the iteration of the loop. So, if this is the first trial, we will access the first row of the table. If it is the second trial, we will access the second row of the data and so on. We have a few **parameters** available to us in this loop (and hence in the loop table). So we use **dot indexing** (this.parameters) to access these parameters. We specifically want the data from the text column in our table, so the complete command is therefore **this.parameters.text**. Essentially, on the first trial the text in the middle of the screen will be whatever is in the text column in the first row. This then updates with the next row down as we move to the next trial, and keeps progressing in this manner until we run out of trials. Your screen will look like this:



We also don't want the sentences to appear on screen forever and we also want to capture participants' responses. To do this, go to the **Beahavior** tab to change how the page behaves. We will make a couple of changes:

- Change the **timeout** to 10000. This will set a timeout of 10,000ms (i.e. 10 seconds) which will only keep the sentence on screen for a maximum of 10 seconds. After this point, the trial will automatically end.
- Change the **Responses** to capture the two keypresses of w and s. We will label the first **w**, record this action as a keypress, and filter it to only accept the keyboard button w. We will label the second **s**, record this action as a keypress, and filter it to only accept the keyboard button s.
- Finally, we will define the **correct response**. Because this changes according to the trial, we will again pick this value out of the correct row number from our loop table. Change the Correct response entry to **${this.parameters.correct_response}**

You should now have a screen that looks like this:

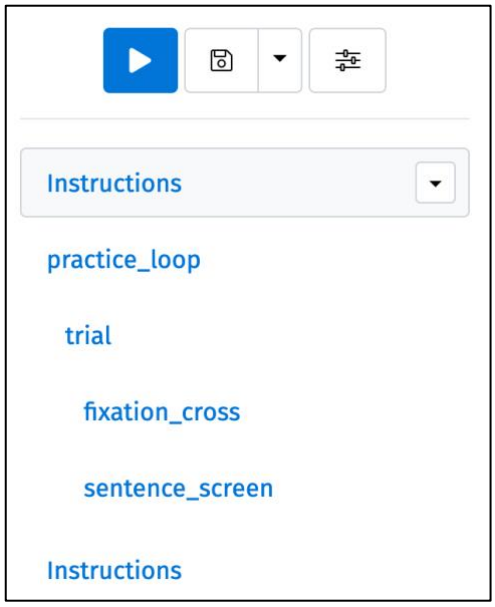


Now our screen will vary according to the trial and end when people press the w or s keys. We will capture their key presses and automatically evaluate whether or not this matches our correct response stored in the loop table for that given trial.

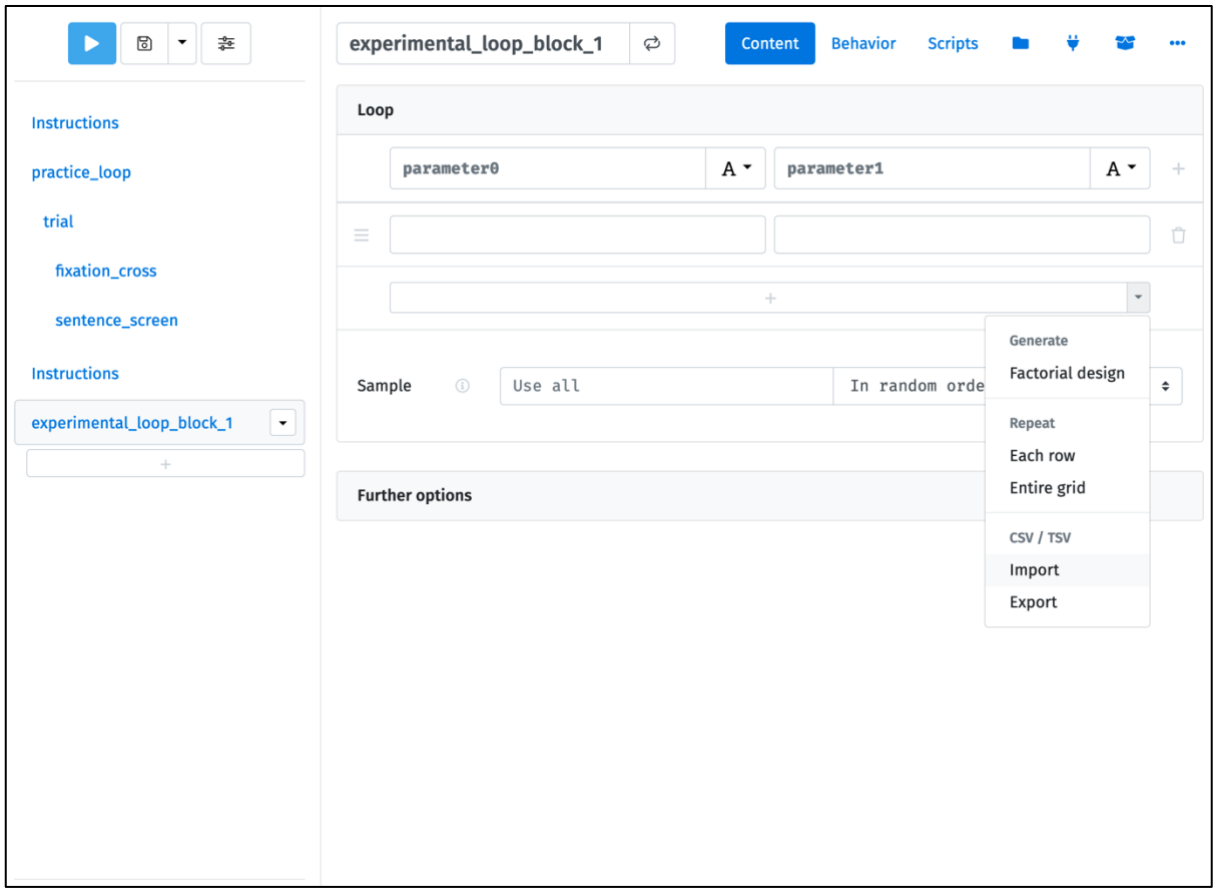
All of the practice trials are now complete. Please **test out your experiment by pressing the play button**. It's also a good idea to **save your experiment as a .json file using the save icon**.

Now we will make the experimental trials for the main phase of the experiment. We will first reiterate the instructions, and then repeat the trials in the same way as in the practice phase, only with more (and unique) trials.

14. Duplicate the instructions screen. Do this by hovering your clicking the instructions heading in the left-hand panel, clicking the drop down arrow, and clicking **Duplicate**. After this, **drag the duplicate instructions below the end of the practice loop**. Please ensure it is outside of the practice loop or people will see these after the end of every trial during the practice phase. Please ensure that the left-hand pane now looks like this:



15. For the experimental phase, we will need another couple of loops. We will present the experiments in **two blocks**, one where sensible items need a response of w and one where they need a response of s (though this varies for a given list of items). This counterbalances the key presses. Add the first block loop by clicking the plus sign below the instructions and clicking **Loop** (Flow). Rename this to **experimental_loop_block_1**. We could input everything for this section of the study by hand, but we have written all the items out for you in two .csv files. You can [find this files on Canvas here]. Click the drop down arrow below the Loop table to add to the table from an external file. Go to the **CSV/TSV** section and click **Import**.



Choose the file downloaded from Canvas. This will populate the table for you with the following headings:

- **list:** There are four versions of each item. These rotate across the two experimental factors of direction (see below), and which button is the correct response which each have 2 levels. This allows our experiment to be within-subjects (i.e. all participants see all conditions) but between-items (only certain items are presented to participants in a given condition). This is a common control with linguistic stimuli to stop for any confounding effects within sentences.
- **phase:** a string indicating that this data belongs to the experimental phase of the study.
- **item:** An ID number for a given item/sentence.
- **block:** An ID number for the block in the experimental phase. In this half of the study, we will switch which button is the one for a correct response (and incorrect response) halfway through the study. This ensures people get to respond to sentences as being sensible or not with both buttons. The items are therefore split into two blocks (correct is w and correct is s). This along with the direction of movement in the sentence, is determined for a given trial by the list number.
- **sensibility:** Does the sentence make sense or not. This has two levels; sensible or nonsense.
- **type:** Whether or not the sentence implies transfer. This has two levels; transfer and no_transfer.
- **direction:** Whether the direction of movement in the sentence is towards or away from the participant. This has three levels; towards and away for transfer sentences, and NA for no_transfer sentences.
- **text:** The text to be displayed on screen for the given trial.
- **correct_response:** Which key participants must press to indicate whether a sentence is sensible or not.

There are a lot of trials here due to having four lists. We want to show the trials to participants in a random order to control for **order effects**, but we don't want to show all lists to each participant. Instead we want to show them only the trials within a list. We can filter our table down to a random list for each participant by using some JavaScript.

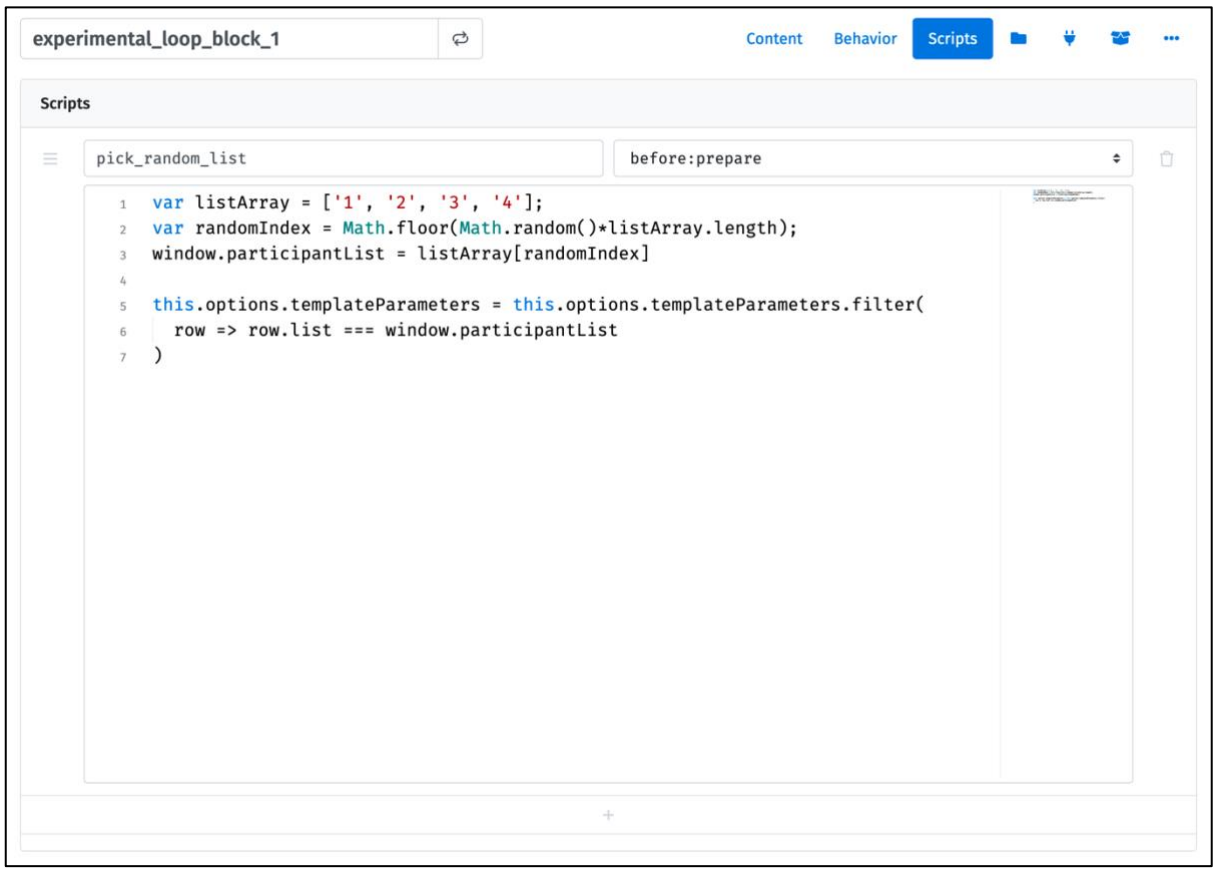
Please note that using the following code isn't necessary for all experiments, but makes this one work a lot easier. You can use lab.js entirely using the builder!

16. Type or copy and paste the following code in the **Scripts** tab. Change the title of the script to **pick_random_list** and ensure that next to this is selected **before:prepare**. (This ensures our code is executed at the start of the experiment and thus trials are properly filtered out.)

```
var listArray = ['1', '2', '3', '4'];

var randomIndex = Math.floor(Math.random()*listArray.length);

window.participantList = listArray[randomIndex]


this.options.templateParameters = this.options.templateParameters.filter(
        row => row.list === window.participantList
)
```

The first block of code here chooses a random list number from between 1 and 4. The second subsets our table of items in the loop to only list items that match our randomly chosen number.

You should have a screen like this now:



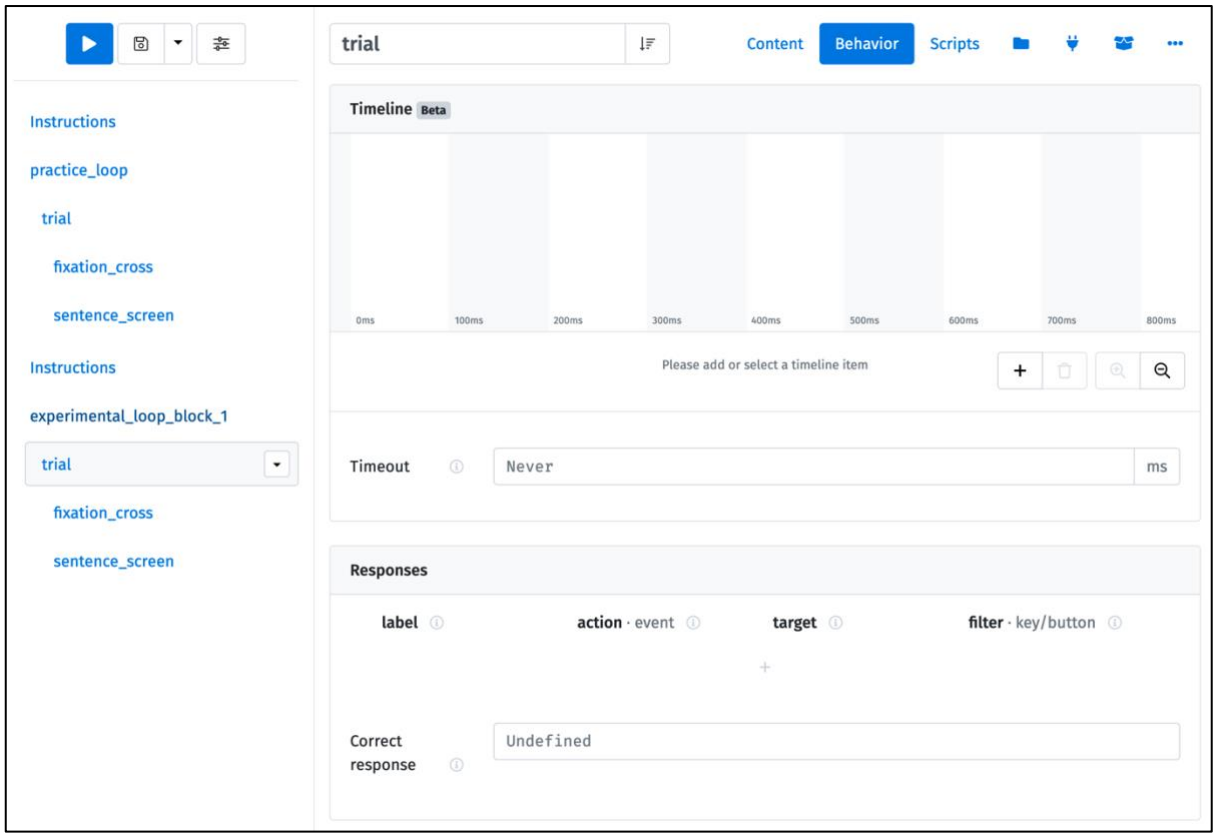
This code works by setting up an array of list numbers. The numbers from the .csv file are read in as a column of string data, so we have to define the list IDs as strings too (hence the quotation marks around them). In JavaScript, we create an array (or object) of strings by defining them in square brackets. This is assigned to the variable listArray. These strings take the indexes of 0, 1, 2, and 3 respectively in the array. We then use some maths to pick a random number between 0 and 3. We will then pick out one of the randomly selected list numbers and assign this to participantList. This is assigned to a global variable using window.participantList because we will need it later for our second block.

Using the randomly selected participant list number, we overwrite our loop table (accessible via this.options.templateParameters) by choosing only rows where the column called list (row.list) is equal to the list number we picked.

If this doesn't make sense, don't worry, but just know that if you want to adapt this code to randomly pick lists or conditions in the future, you only need to update listArray with your lists/conditions.

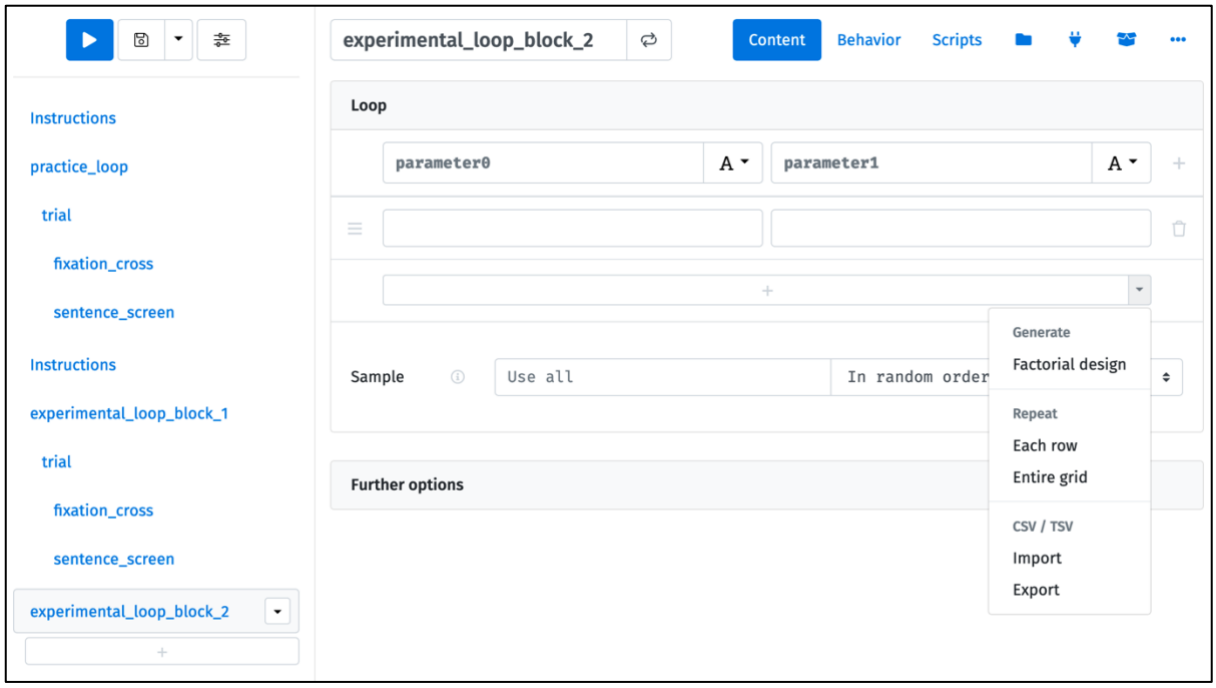
We can next check whether our trials are randomised within the lists by going back to Content, scrolling to the bottom of the table, and then looking at the sampling (in sample). By default, lab.js picks items randomly from our table. Since we already filter it to the trials in a given list, and then lab.js randomly shuffles the items, **we're all set!**

17. Let's next add a **trial sequence** to this loop. Duplicate the whole trial sequence called trial in the same was as you did for the instructions before. Click trial, choose the dropdown list, choose **Duplicate**. Then drag this within the **experimental_loop_block_1** loop.



We will then copy our instructions once more and create an experimental loop for block two.

18. Add a new loop below everything for **experimental_loop_block_1** and name this **experimental_loop_block_2**. Click the drop down menu in the add row section of the loop table and import the CSV file for block 2 as before (CSV, **Import**).



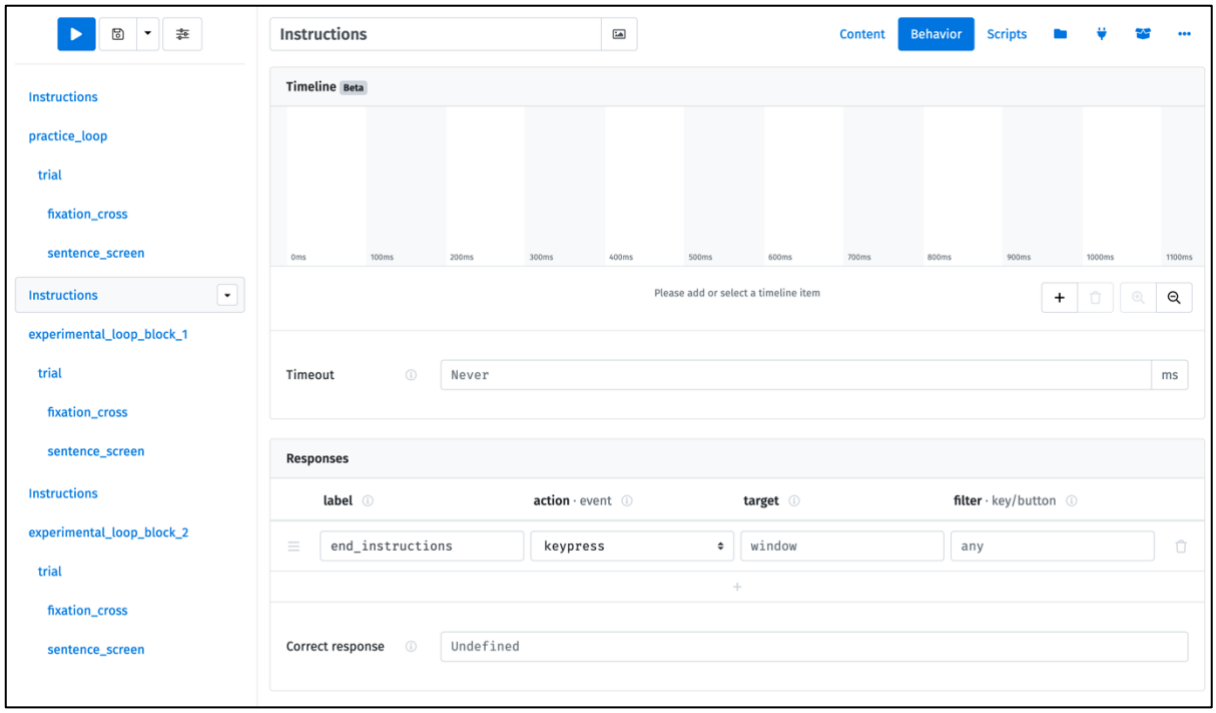
19. Again, we don't want to show every trial to each participant, so copy the following code into the **Scripts** tab.

```
this.options.templateParameters = this.options.templateParameters.filter(
    row => row.list === window.participantList
    )
```

This just repeats the filtering used in **experimental_loop_block_1** but crucially keeps the list number the same so that this time we get the remaining items in the list. (We don't want to randomly choose a list again here because we may give participants two versions of the same item.) Finally, make sure the name of this code is called **pick_random_list_again**, and that it is executed in the before:prepare phase of the experiment.

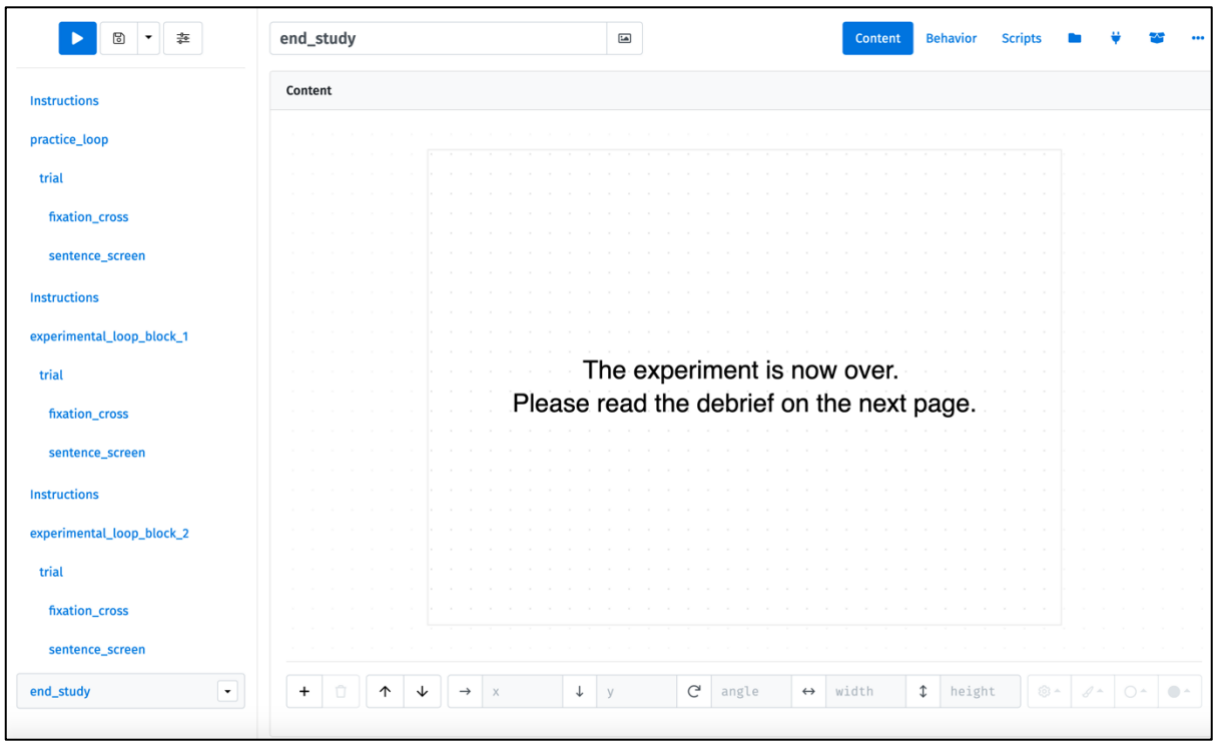
20. We'll now add our trials to this loop. Because we have already created the trial sequence we can just **duplicate** it in the same way we did the instructions and drag it underneath **experimental_loop_block_2**. Again, ensure the trial sequence is within this loop and not outside of it.

21. We will add some instructions just before block 2. Again, **duplicate Instructions**, and insert this just above **experimental_loop_block_2** (but not within experimental_loop_block_1).



22. Finally, we will add an end screen to the experiment. Here you will thank the participant for taking part in the study and let them know that the experiment is over. Add a final **Screen (Canvas)** element to the bottom of the screen that is not indented within the **experimental_loop_block_2** component.

    Add a message saying something like “The experiment is now over. Please read the debrief on the next page.” By selecting the **plus sign** and choosing a **text** element. We will give a full debrief of the experiment within Qualtrics, where the data from this task will be pushed through so we can access it. Rename this component to **end_study**.



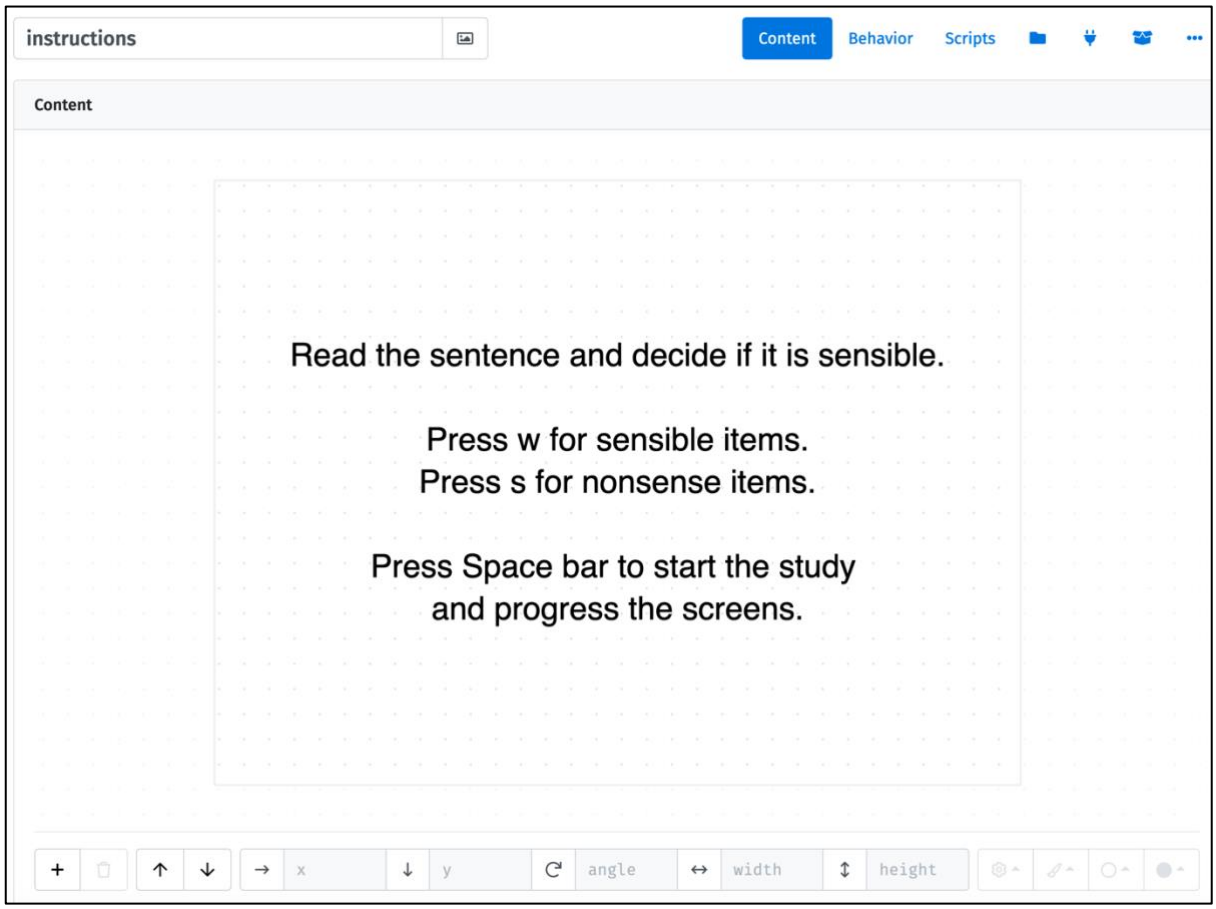
Again, right now this study will be stuck on the **end_study** screen forever right now. Set a timeout on the study to **2000**. This will mean the message is on screen for 2 seconds (AKA 2000ms) and then disappears. After this screen, all data from the study will be sent to Qualtrics before once we set the study up there.

The last thing we will do in lab.js before moving on to Qualtrics is to **change the instructions for each section**.

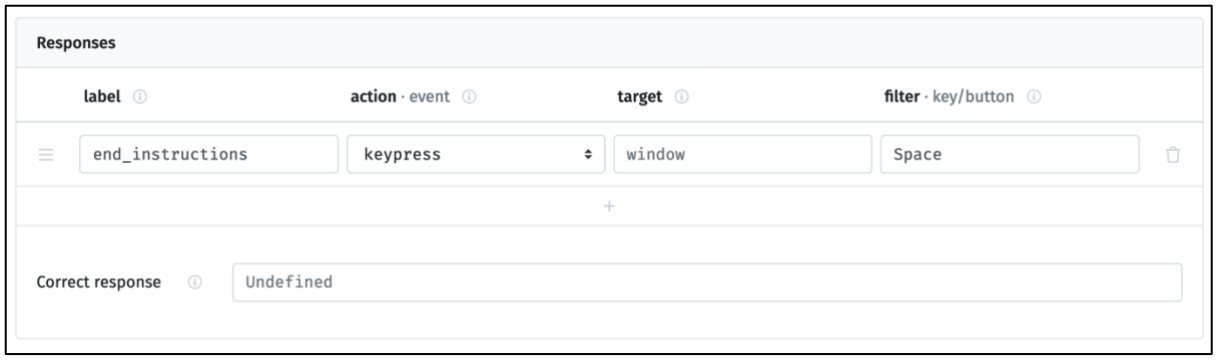
23. For the practice instructions, go to the content tab and change the text reading "Our Instructions" to some instructions you'd like to give participants about what to do in the task.

    Remember, for the practice items the sensible sentences must be responded to with a w, while the nonsense sentences must be responded to with an s. We should tell participants how to progress in the study too. If you tell them to use the Space bar only, it is best to do this and then go to **Behavior** and in **Responses** change the **end_instructions** response such that the **Filter** cell contains the word **Space**.

Your Content screen should look like this:



Your **Responses** section in the **Behavior** tab should look like this (if you choose to only allow the **Space** bar to progress the screens):



Update the instructions for **instructions_block_1**. Remember that these change depending upon the list. This is where our global variable of the random list will come in handy again.

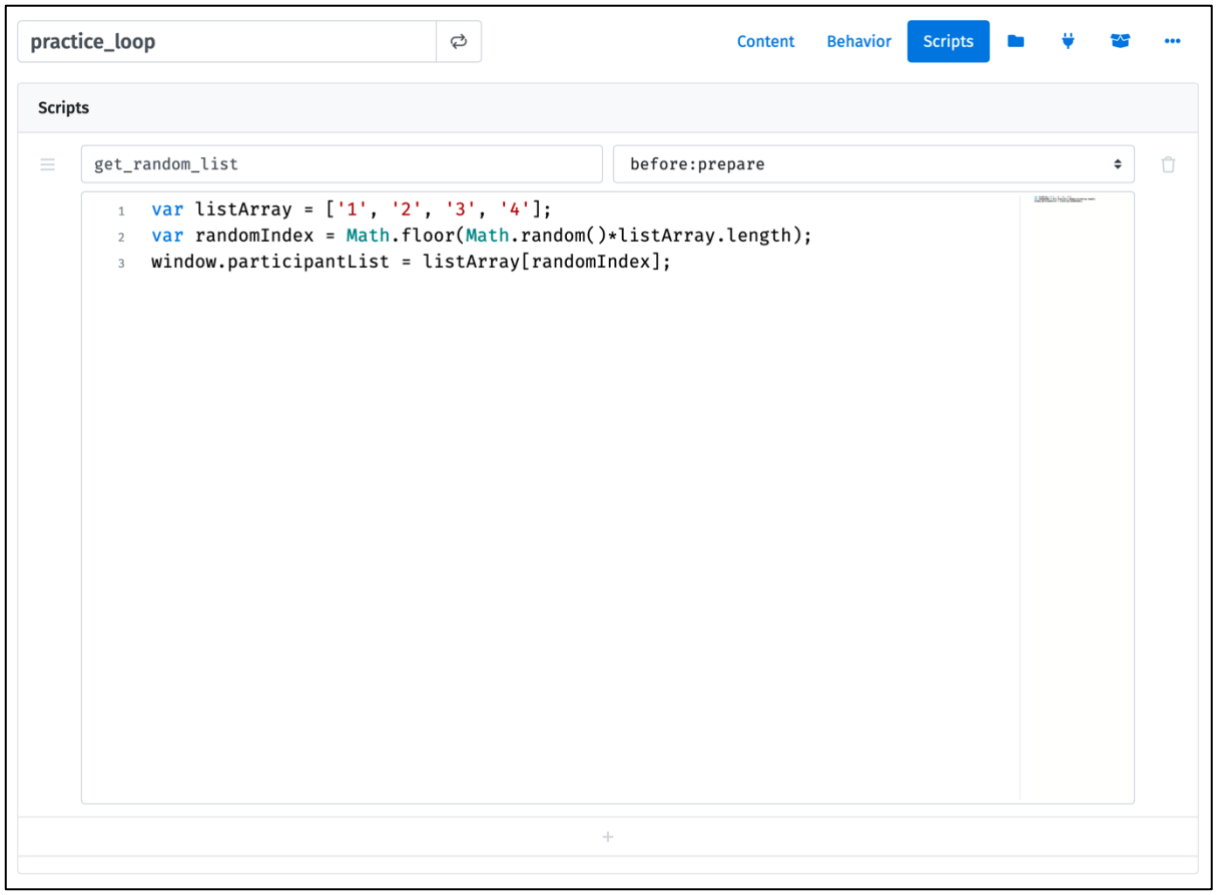
24. **Cut and paste** this section of code from **experimental_loop_block_1** into the **Scripts pane in practice_loop**.

    Please ensure you have cut and not copied this section of code!

```
var listArray = ['1', '2', '3', '4'];
var randomIndex = Math.floor(Math.random()*listArray.length);
window.participantList = listArray[randomIndex]
```

Rename the script to **get_random_list** and put this in the **before:prepare** section.

This will now choose a random list for our participants once they start the practice section of the experiment. Your screen should look like this:

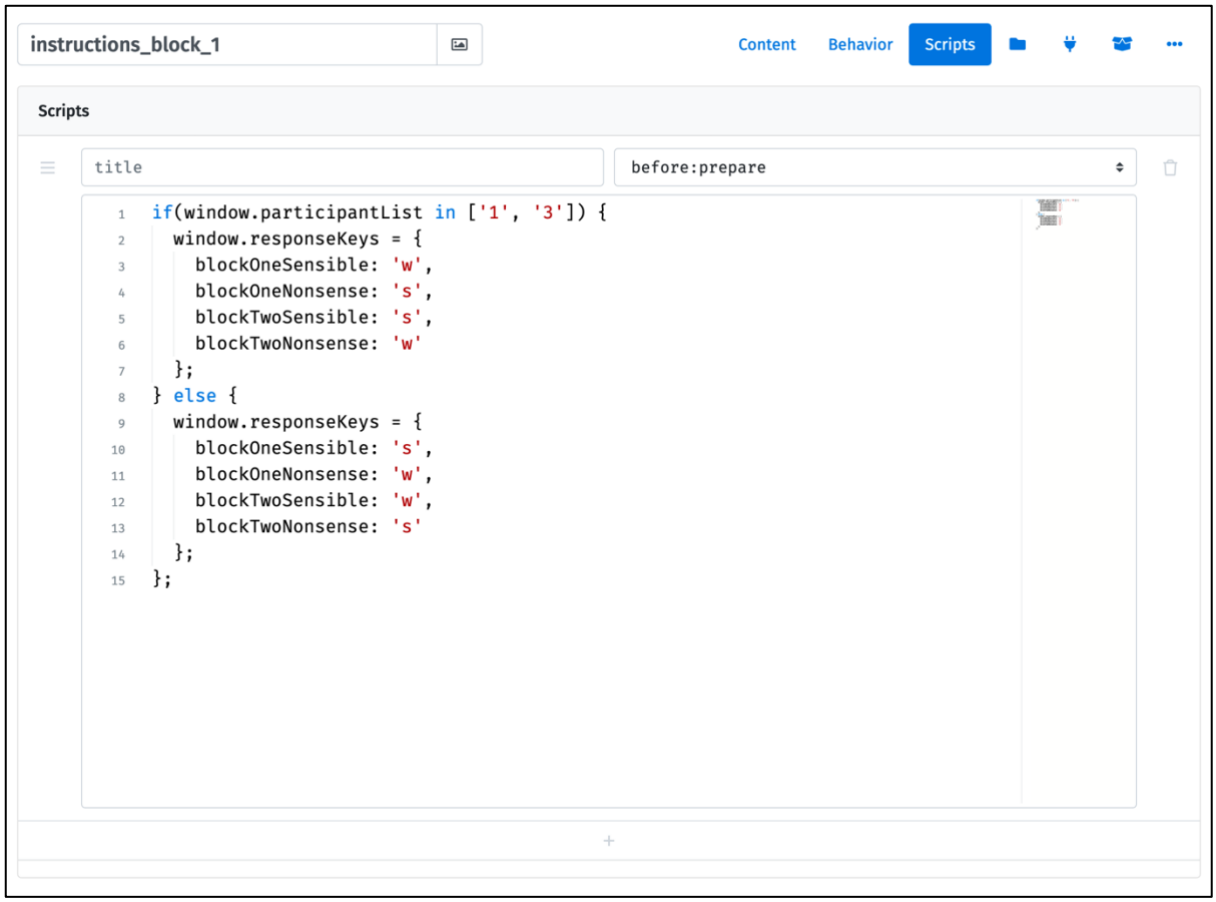


25. In **instructions_block_1** we will paste the following code into the **Scripts**. This looks at the participant's randomly chosen list number from the practice block. It then defines which keys they need to use to respond to sensible and nonsense items in each block depending upon their list. For example, if participants are in list 1 or 3, they have to respond with w for sensible keys and s for nonsense keys in block 1, but this is switched in block 2. For those in list 2 and 4, the order is reversed.

```
if(window.participantList in ['1', '3']) {
  window.responseKeys = {
    blockOneSensible: 'w',
    blockOneNonsense: 's',
    blockTwoSensible: 's',
    blockTwoNonsense: 'w'
  };
} else {
  window.responseKeys = {
    blockOneSensible: 's',
    blockOneNonsense: 'w',
    blockTwoSensible: 'w',
    blockTwoNonsense: 's'
```

```
  };
};
```

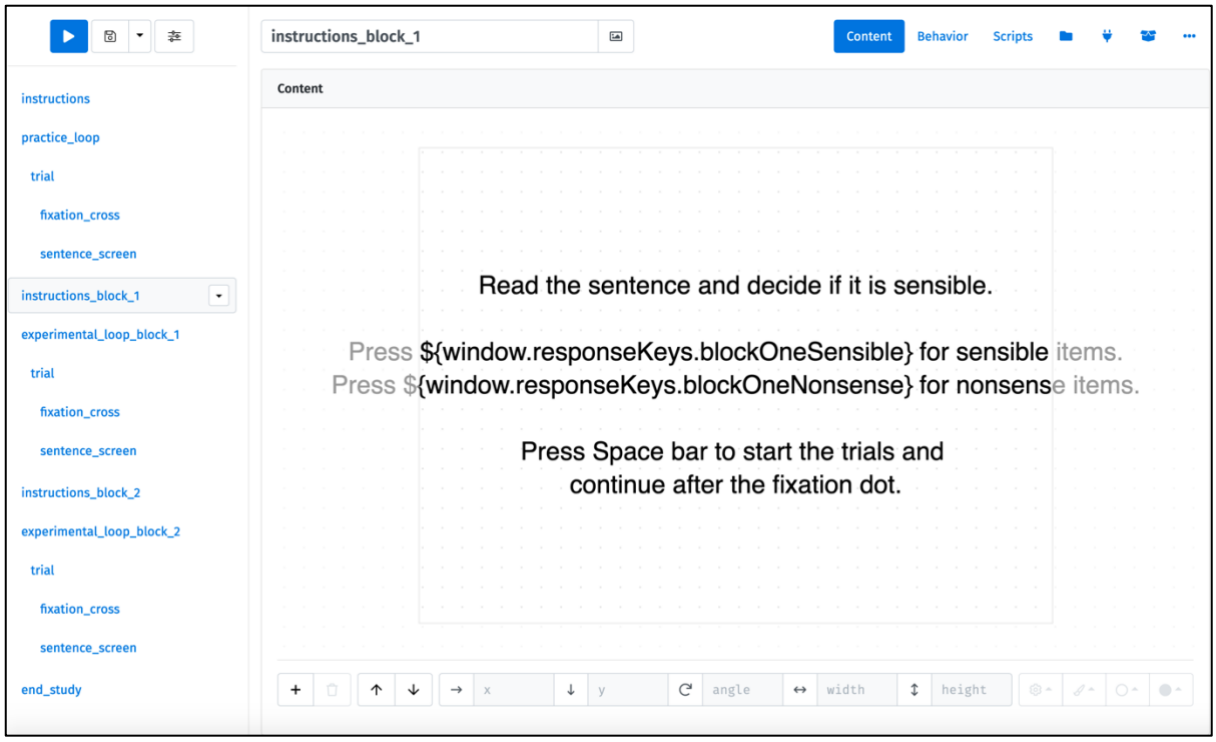
Your screen should look like this:



Next, go to the Content section and type in the following where you have the instructions text:

> Read the sentence and decide if it is sensible.
>
> Press ${window.responseKeys.blockOneSensible} for sensible items.
>
> Press ${window.responseKeys.blockOneNonsense} for nonsense items.
>
> Press Space bar to start the trials and continue after the fixation dot.

Your Content screen should now look like this:



This will change the text in **${}** to the right key depending upon the randomly chosen list the participant was assigned to.

26. Remember that we also filtered the items out in **experimental_loop_block_2** depending on the list that was picked? We also then need to **update the instructions in this block**. Go to **instructions_block_2** and change the text in the Content tab to this:

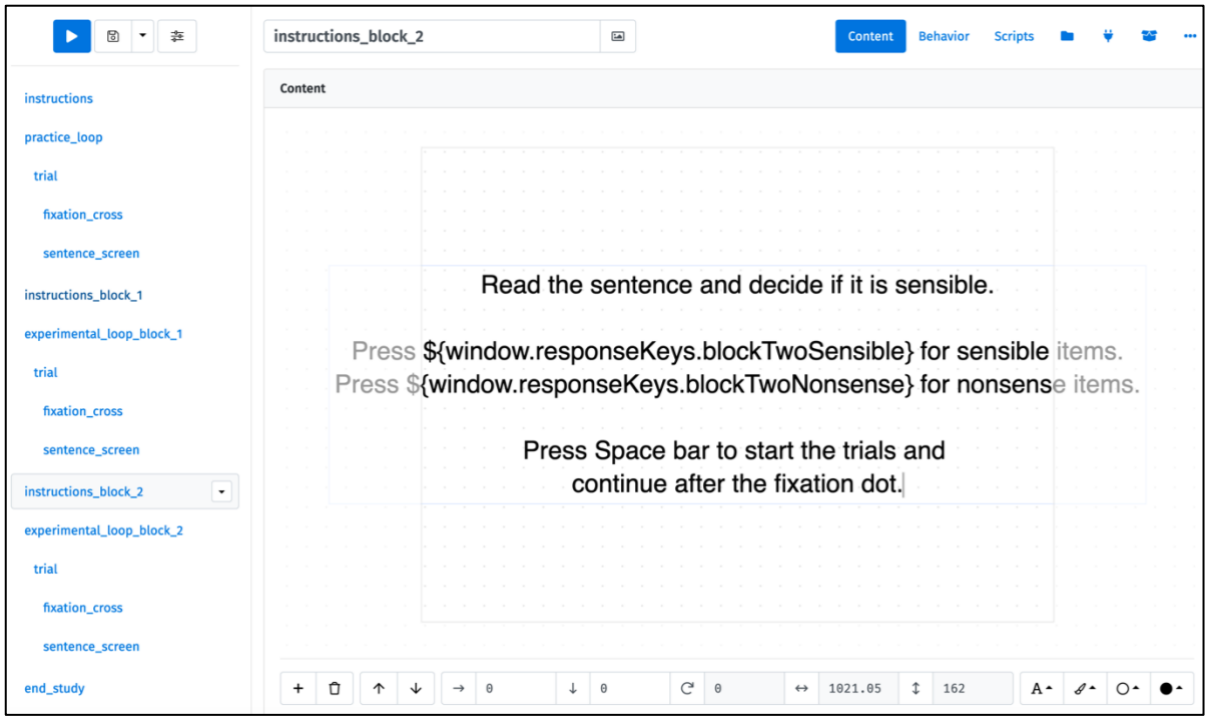
Read the sentence and decide if it is sensible.

Press ${window.responseKeys.blockTwoSensible} for sensible items.

Press ${window.responseKeys.blockTwoNonsense} for nonsense items.

Press Space bar to start the trials and continue after the fixation dot.

Your Content screen should now look like this:

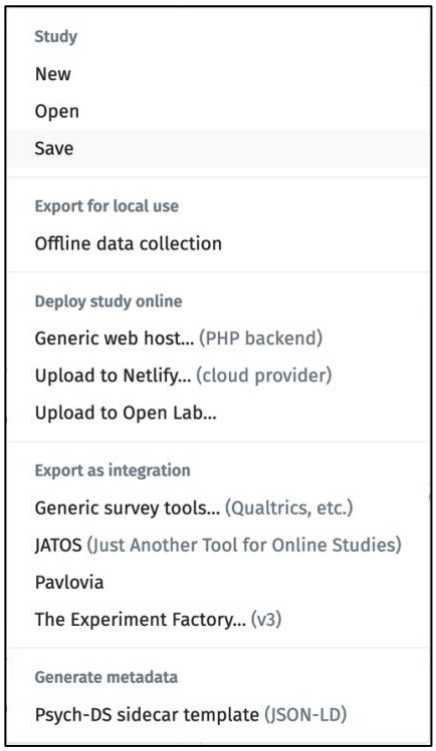


Your experiment is now complete and will work in your browser.

Press the Play button to try it out. Save your work using the Save icon.

Preparing the Experiment for the Online Testing

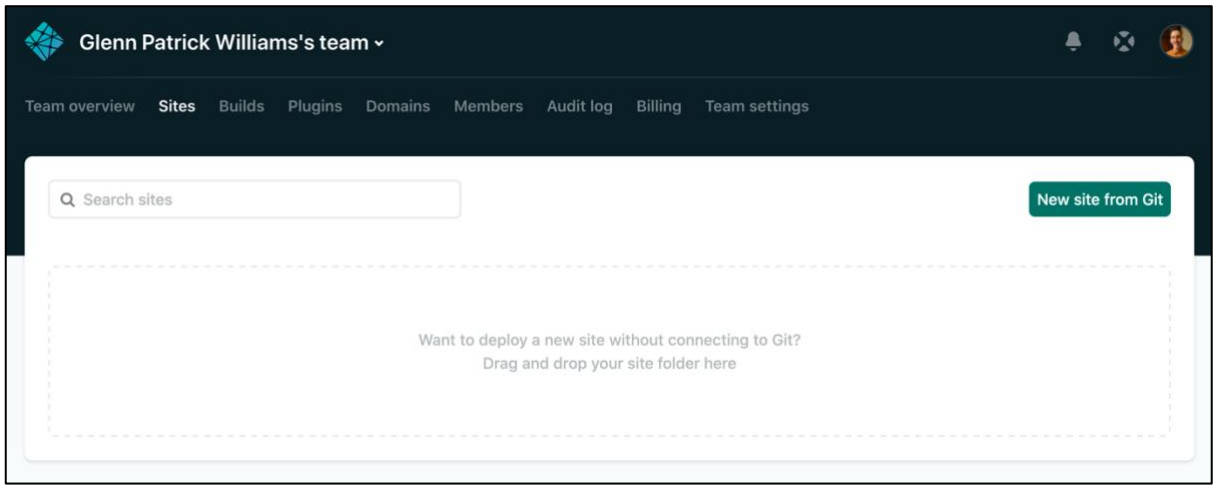
We can save lab.js studies in many formats. If you choose the dropdown menu on the save icon you can see that we can Export the study (or deploy it directly online) for a number of different platforms. You can see these options below:



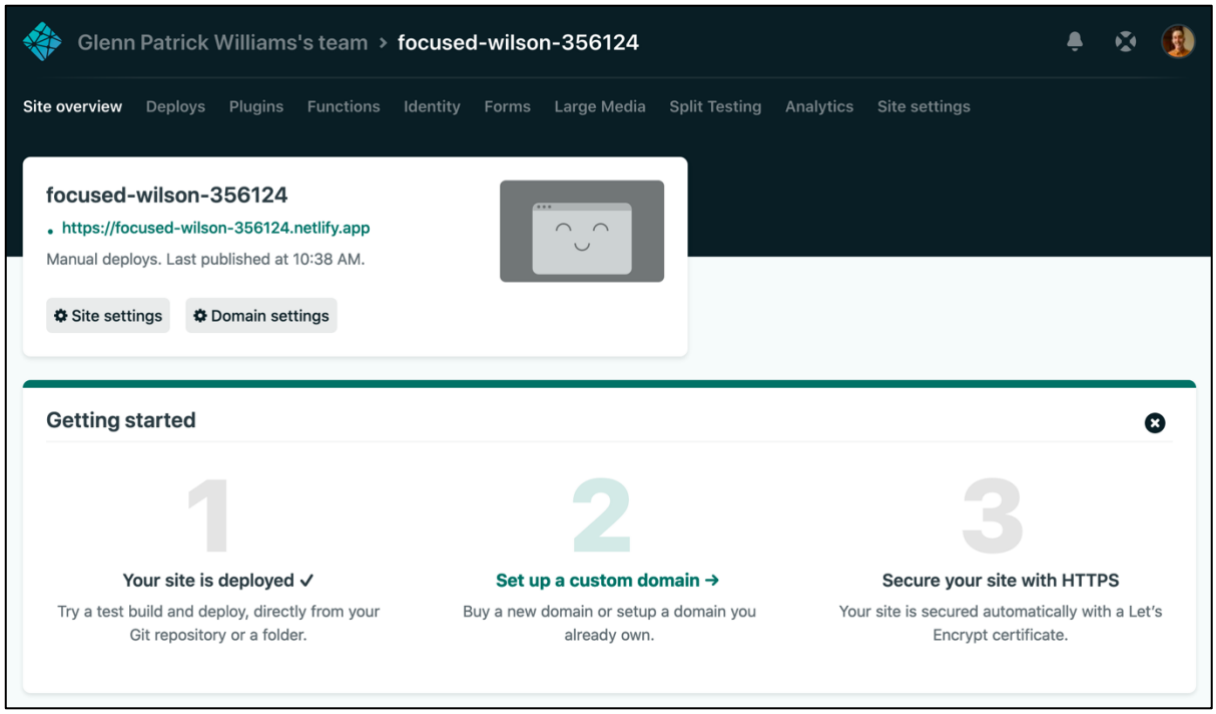
If we want to run our study in the lab, we simply choose **Export for local use (Offline data collection)**. However, we want to deploy our study online. There are a few methods for integrating out study online, but many require us to work with servers and/or paid services. Luckily, the University of Sunderland has a license for Qualtrics. We can **embed our study in Qualtrics** by choosing **Export as integration**, specifically clicking on the **Generic survey tools… (Qualtrics, etc.)** option. Choose this option the click **Continue to download** in the pop-up menu. Save the file as a .zip folder.

Unzip the folder. This folder contains the code and files necessary for running your experiment online in Qualtrics. We need to host these files online somewhere so Qualtrics can build the experiment for us online.

One easy method is to upload your files directly to Netlify. This is a free website for building websites online. Sign up with your email address. Go to the **Sites** tab. You should see a screen like this:



This allows you to drag and drop folders into this location. Drag your unzipped folder into this section. This will take you a screen that looks similar to this:



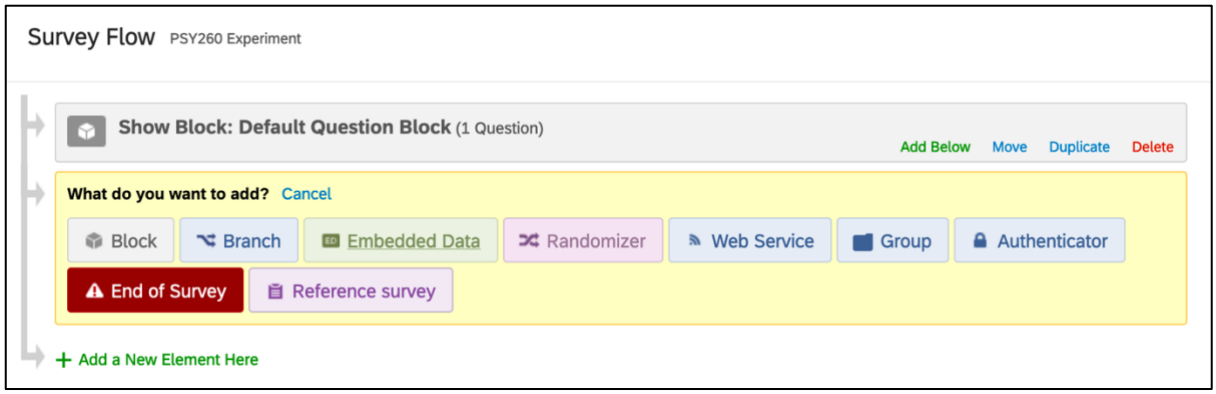
Make a note of the url to your website. In this case, it is https://focused-wilson-356124.netlify.app.

Keep this screen up and go to Qualtrics. Log in and **create a new survey**. You will eventually want to add your own Information Screen, Consent form, Demographic Questions, and Debrief to this. However, this guide will show you how to add your experiment to Qualtrics where you can collect data from the participants.

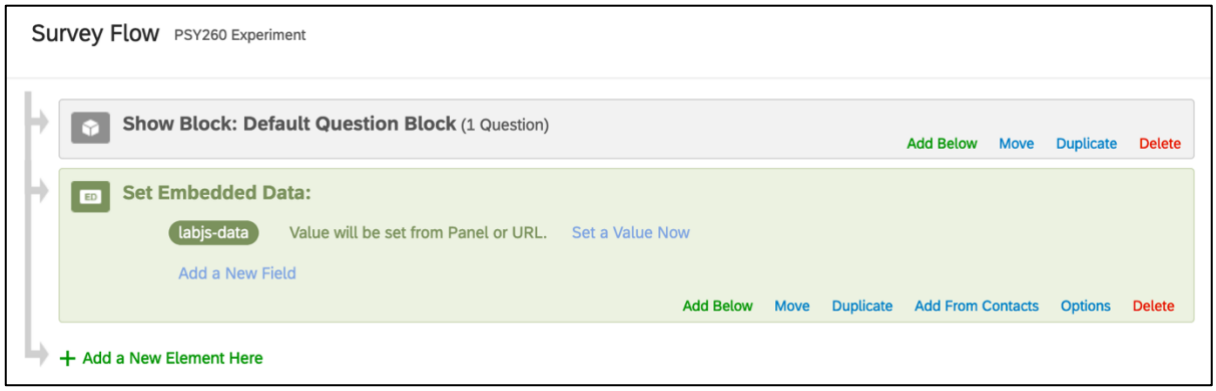
We will assume you are already familiar with Qualtrics from first year. To get started though, follow these instructions.

27. Create a new project by clicking the **Create new project** button.
28. Choose a regular survey then select **Get Started**.
29. Change your project title to **PSY260 Experiment**.
30. First go to survey flow and create a place to save your data from the experiment. Click on **Survey Flow** and then click **Add a New Element Here**.

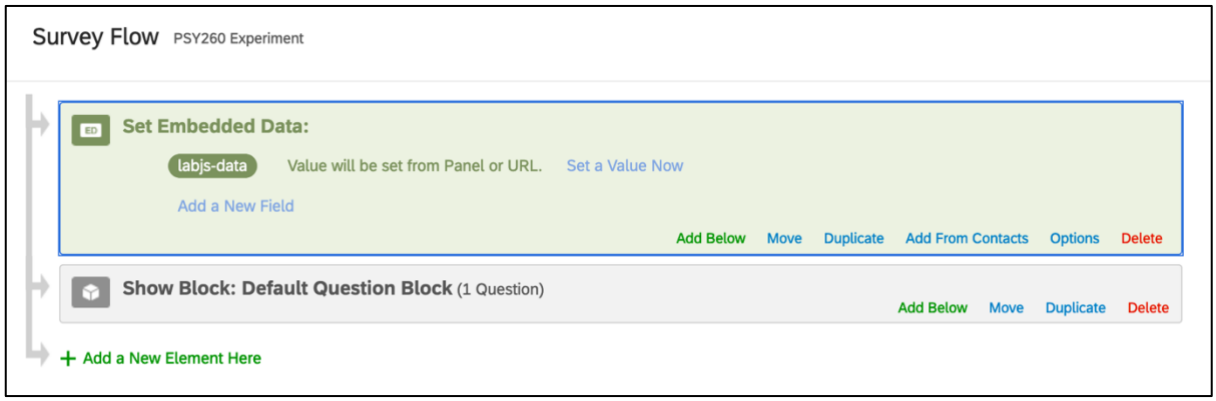
31. Select **Embedded data**.



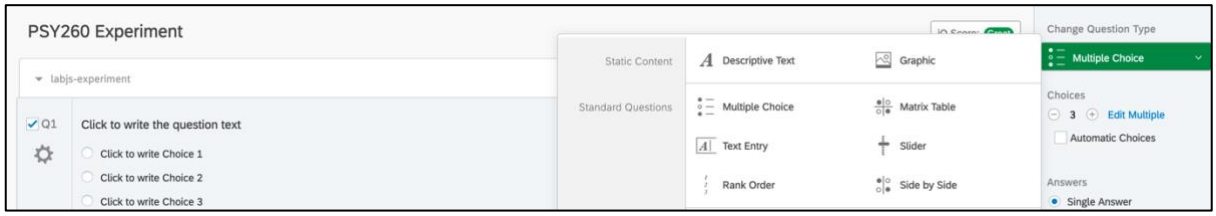
32. Rename this embedded data to **labjs-data**.



33. Click **Move** on this embedded data and drag it to the top of the survey flow.



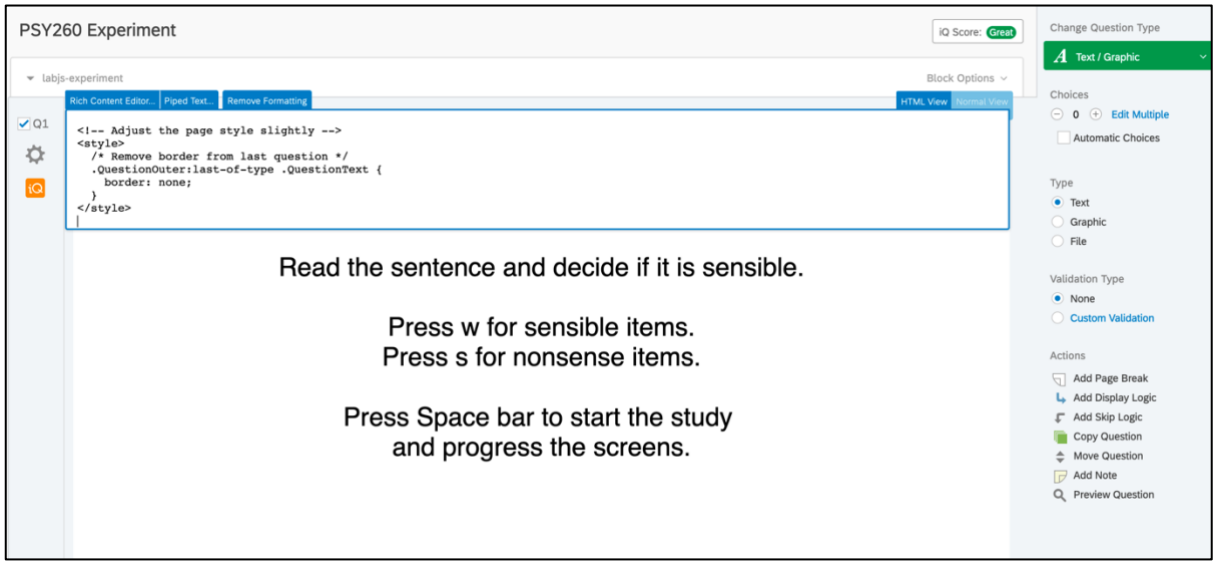
34. Click **Save Flow**. We can now save data from our experiment in Qualtrics.
35. Now we will create a place to show the experiment. **Rename the Default Question Block to labjs-experiment**.
36. Click the cog next to Q1 in this block and select **Change Question type** in the right-hand menu. Choose the dropdown arrow on the green button titled **Create a New Question** and select **Descriptive Text**.



37. Click to write in this text box, and choose **HTML view** at the top right of the box. Copy this html code into your html box:

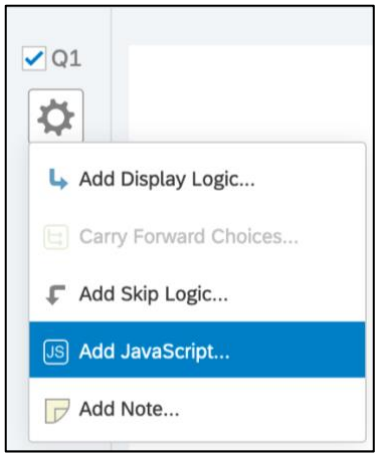
```
<!-- Embed the study -->
<iframe
  src="https://focused-wilson-356124.netlify.app/"
  style="width: 100%; min-height: 600px; border: none;"
  allowfullscreen
></iframe>

<!-- Adjust the page style slightly -->
<style>
  /* Remove border from last question */
  .QuestionOuter:last-of-type .QuestionText {
    border: none;
  }
</style>
```

IMPORTANT: change the text https://focused-wilson-356124.netlify.app/" to your url for your study that you got from Netlify. You should see a screen like this, only with your url in it:



When you click out of the text the screen for your experiment will show up.

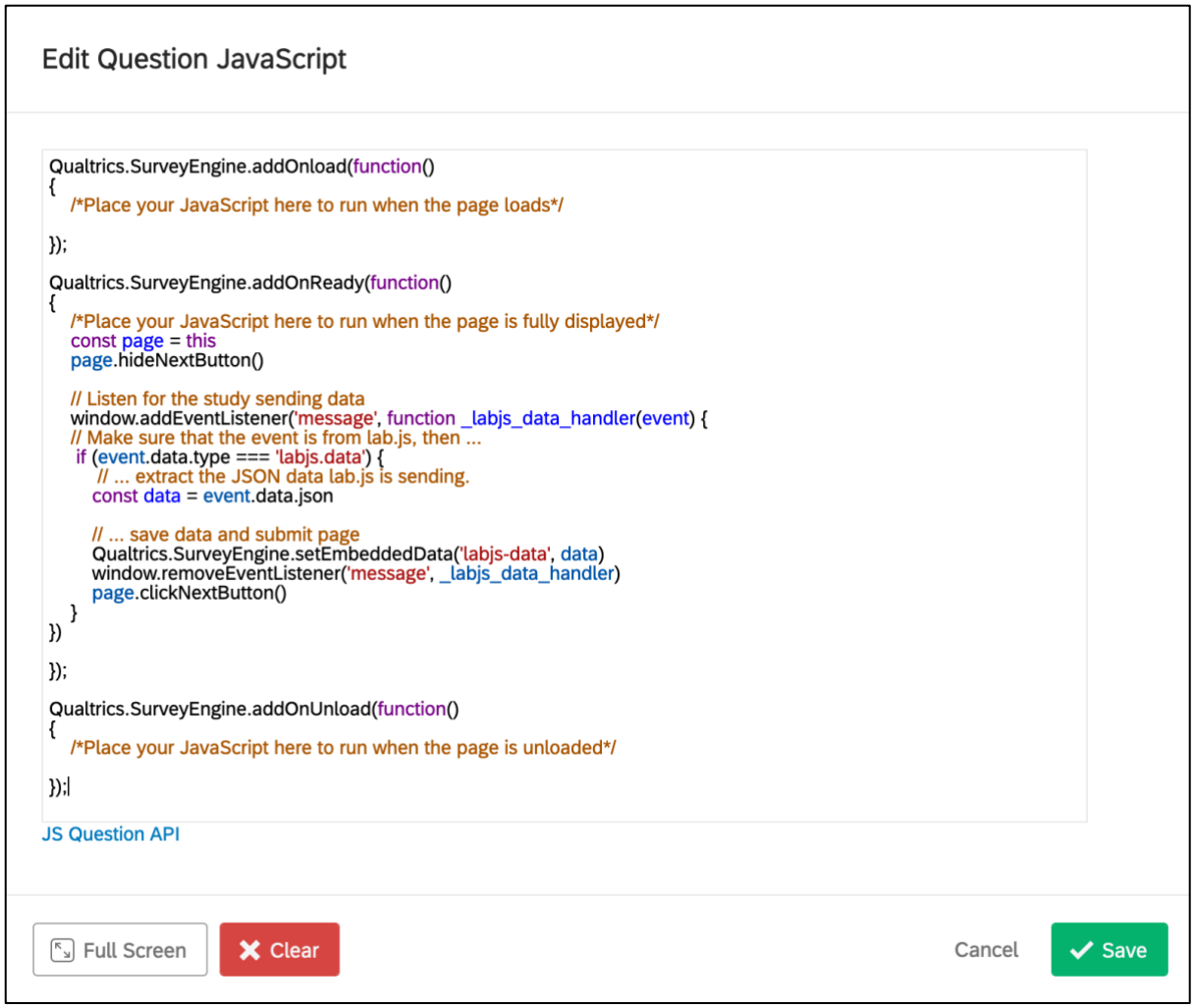
38. Now we need to add some JavaScript to ensure that Qualtrics saves the data from this experiment within our questionnaire. Click on the **settings menu** (the cog) for Q1 within **labjs-experiment** and select **Add JavaScript…**



39. We will change the code here. This will tell Qualtrics how to handle the experiment data. The easiest thing here is to just delete everything in here and replace it with the following code:

```
Qualtrics.SurveyEngine.addOnload(function()
{
  /*Place your JavaScript here to run when the page loads*/
});

Qualtrics.SurveyEngine.addOnReady(function()
{
  /*Place your JavaScript here to run when the page is fully displayed*/
  const page = this
  page.hideNextButton()

  // Listen for the study sending data
  window.addEventListener('message', function _labjs_data_handler(event) {
  // Make sure that the event is from lab.js, then ...
  if (event.data.type === 'labjs.data') {
    // ... extract the JSON data lab.js is sending.
    const data = event.data.json

   // ... save data and submit page
   Qualtrics.SurveyEngine.setEmbeddedData('labjs-data', data)
   window.removeEventListener('message', _labjs_data_handler)
   page.clickNextButton()
  }
})

});

Qualtrics.SurveyEngine.addOnUnload(function()
{
  /*Place your JavaScript here to run when the page is unloaded*/
});
```

Your screen should now look like this:



40. Click **Save**. Your study is now complete and ready to collect data online via Qualtrics.
41. To collect data, choose **Publish** and follow the link to your study.

It's always a good idea to test that your study works properly and is able to collect data accurately.

42. Once published, go through your experiment once or twice.
43. Go to the **Data and Analysis tab** and download your data as a **.csv file**.
44. Open up your .csv file. Is the data in a format you expected? Probably not. It needs to be **unpacked**, and doing this manually is incredibly difficult. To do this, **we will use R**. Please **follow the instructions on Canvas** for how to unpack the data.

Final Notes on the Assessment

You're now able to build experiments and collect data online. **For your assessment** you will adapt this experiment in the lab.js builder, export the new version of the study and update the hosted experiment on Netlify. You will then collect data and unpack it using R before completing your analyses and write up of the experiment.

Important: Make sure that you have saved your experiment for the lab.js builder and an exported version of the study for Qualtrics.

You can reload any saved experiment in lab.js by using the **dropdown menu** and choosing **Open**. This will save you the work of going through the entire step by step guide when it comes to building your own experiment.