

# **Workshop on Experimental Linguistics/Syntax**

## **Experimental Syntax: a walkthrough**

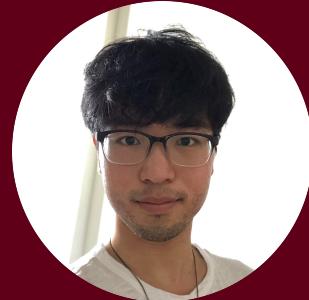
**January 22, 2021**

**Nayoun Kim**

Arts & Science Postdoctoral fellow, Department of Linguistics  
University of Toronto

# Guest

(Will be answering questions in the chat!)



## June Choe

**Ph.D. Student in Linguistics**

University of Pennsylvania  
Philadelphia, PA, USA

---

Research: psycholinguistics, computational linguistics

Contact: [yjchoe@sas.upenn.edu](mailto:yjchoe@sas.upenn.edu) | [yjunechoe.github.io](https://yjunechoe.github.io)

# Presentation Outline



## Introduction to IBEX

- Basic ideas
- Navigating the platform



## Scripting an experiment

- Overview of critical components
- Code walkthrough



## Analysis of the results

- Understanding the output format
- Importing into Excel and R

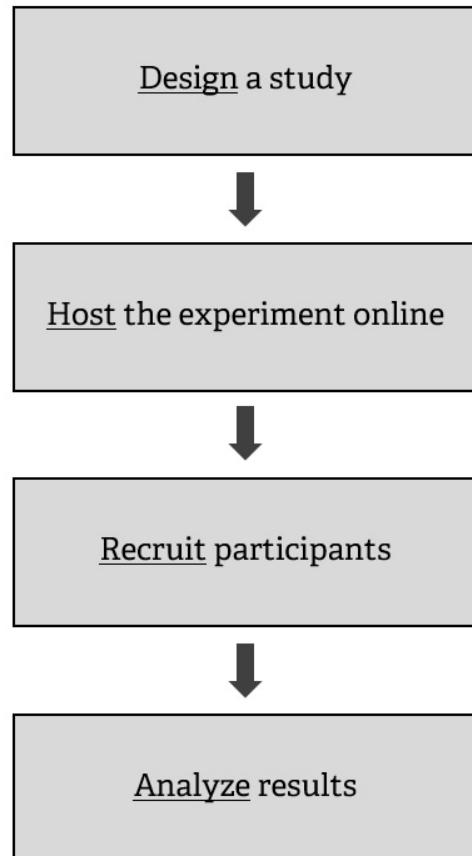
**Link to the slides:** <https://ibex-workshop-slides.netlify.app>

# 1. Introduction to IBEX

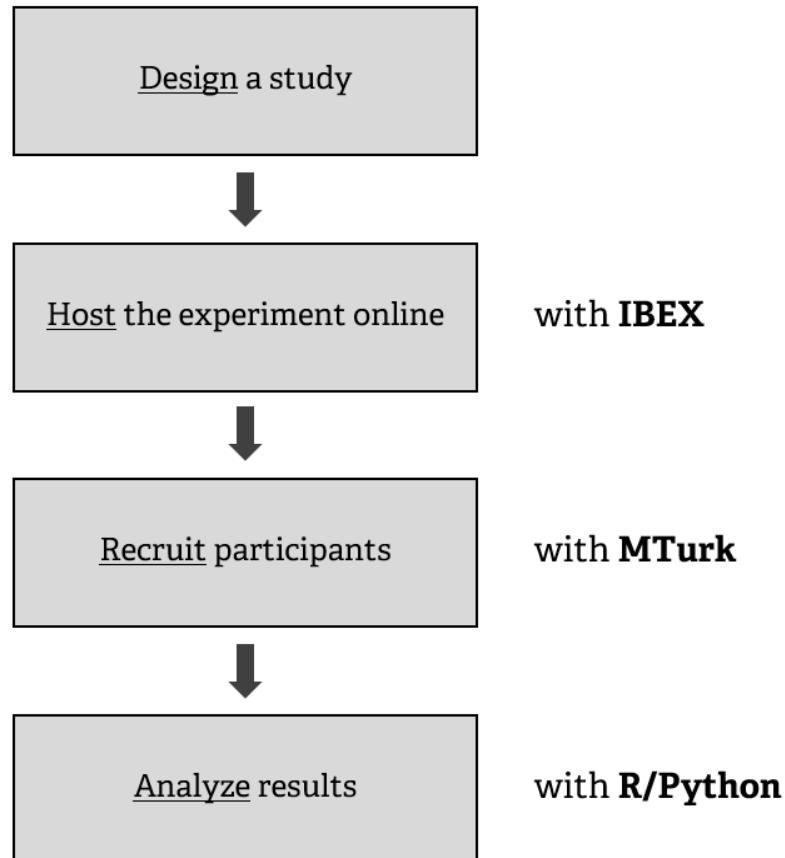
# What is IBEX?

- Stands for (**I**)nternet-(**B**)ased (**EX**)periments
- **DOES:**
  - Host experiments on a webpage
  - Log user interactions with the experiment
  - Store data on a secure server
- **DOES NOT:**
  - Recruit participants (*see* Amazon Mechanical Turk)
  - Provide an analysis of the results (*see* R, Python)

# Where does IBEX fit?



# Where does IBEX fit?



# Navigating IBEX

- Go to <https://spellout.net/ibexfarm>
- Click **create an account** (or **log in**)
- Click **manage my experiments**
- Click **Create a new experiment**
- Give it a unique name like "*workshop\_example*"
- Done!

## Experiments

- **workshop\_example** (ibex 0.3.9) ([delete](#) | [rename](#))

[» Create a new experiment](#)

# Navigating Ibex

**Ibex Farm**

[home](#) | [login](#) | [create an account](#) | [ibex docs](#)

**Welcome to the Ibex farm**



This site provides free hosting for **ibex** experiments.

Upload data files in your browser, then send your participants a link to the experiment.

[View an example experiment](#)

Currently hosting **20564** experiments.

Have you found Ibex Farm useful? Buy me a coffee!  
(Running costs are now covered for the foreseeable future. Thank you all.)

Contact [a.d.drummond@gmail.com](mailto:a.d.drummond@gmail.com) if you have any issues, or try the [google group](#).  
The code for this site is BSD-licensed and available on [github](#).

▶ 0:00 / 0:25

Speaker icon

More options icon

# Experiment file structure

## chunk\_includes

- Stand-alone files go here

## css\_includes

- Style specifications go here

## data\_includes

- Experiment scripts go here

## js\_includes

- Modules ("controllers") go here

## results & server\_state

- Automatically generated/updated

### Experiment 'workshop\_example' (ibex 0.3.9)

Update from git repo» [\(help\)](#)

#### chunk\_includes (upload a file to this directory | refresh)

example\_intro.html ([delete](#) | [rename](#) | [upload new version](#) | [edit](#))  
test1.mp3 ([delete](#) | [rename](#) | [upload new version](#) | [edit](#))  
test2.mp3 ([delete](#) | [rename](#) | [upload new version](#) | [edit](#))

#### css\_includes (upload a file to this directory | refresh)

DashedSentence.css  
FlashSentence.css  
Form.css  
global\_main.css  
Message.css  
Question.css  
Scale.css  
Separator.css

#### data\_includes (upload a file to this directory | refresh)

example\_data.js ([delete](#) | [rename](#) | [upload new version](#) | [edit](#))

#### js\_includes (upload a file to this directory | refresh)

AcceptabilityJudgment.js  
DashedAcceptabilityJudgment.js  
DashedSentence.js  
FlashSentence.js  
Form.js  
Message.js  
Question.js  
Scale.js  
Separator.js  
VBox.js

#### results (upload a file to this directory | refresh)

*This directory is not present*

#### server\_state (upload a file to this directory | refresh)

*This directory is not present*

# Experiment file structure

## chunk\_includes

- Stand-alone files go here

## css\_includes

- Style specifications go here

## data\_includes

- Experiment scripts go here

## js\_includes

- Modules ("controllers") go here

## results & server\_state

- Automatically generated/updated

### Experiment 'workshop\_example' (ibex 0.3.9)

Update from git repo» [\(help\)](#)

#### chunk\_includes ([upload a file to this directory](#) | [refresh](#))

example\_intro.html ([delete](#) | [rename](#) | [upload new version](#) | [edit](#))  
test1.mp3 ([delete](#) | [rename](#) | [upload new version](#) | [edit](#))  
test2.mp3 ([delete](#) | [rename](#) | [upload new version](#) | [edit](#))

#### css\_includes ([upload a file to this directory](#) | [refresh](#))

DashedSentence.css  
FlashSentence.css  
Form.css  
global\_main.css  
Message.css  
Question.css  
Scale.css  
Separator.css

#### data\_includes ([upload a file to this directory](#) | [refresh](#))

example\_data.js ([delete](#) | [rename](#) | [upload new version](#) | [edit](#))

#### js\_includes ([upload a file to this directory](#) | [refresh](#))

AcceptabilityJudgment.js  
DashedAcceptabilityJudgment.js  
DashedSentence.js  
FlashSentence.js  
Form.js  
Message.js  
Question.js  
Scale.js  
Separator.js  
VBox.js

#### results ([upload a file to this directory](#) | [refresh](#))

*This directory is not present*

#### server\_state ([upload a file to this directory](#) | [refresh](#))

*This directory is not present*

## 2. Scripting an experiment

# The script

Only need to modify **one file**: *example\_data.js* (can also be renamed later)

At creation, the default file looks like this:

```
var shuffleSequence = seq("intro", sepWith("sep", seq("practice", rshuffle("s1", "s2"))),  
sepWith("sep", rshuffle("q1", "q2")));  
var practiceItemTypes = ["practice"];  
  
var defaults = [  
    "Separator", {  
        transfer: 1000,  
        normalMessage: "Please wait for the next sentence.",  
        errorMessage: "Wrong. Please wait for the next sentence."  
    },  
    "DashedSentence", {  
        mode: "self-paced reading"  
    },  
    "AcceptabilityJudgment", {  
        as: ["1", "2", "3", "4", "5", "6", "7"],  
        presentAsScale: true,  
        instructions: "Use number keys or click boxes to answer.",  
        leftComment: "(Bad)", rightComment: "(Good)"  
    },  
];
```

The above script creates this self-paced reading experiment.

# Writing your own script

The `example_data.js` script works, but is not very friendly.

We'll use **our own** - template file - to demonstrate how the script works.

```
// Defaults and other settings //

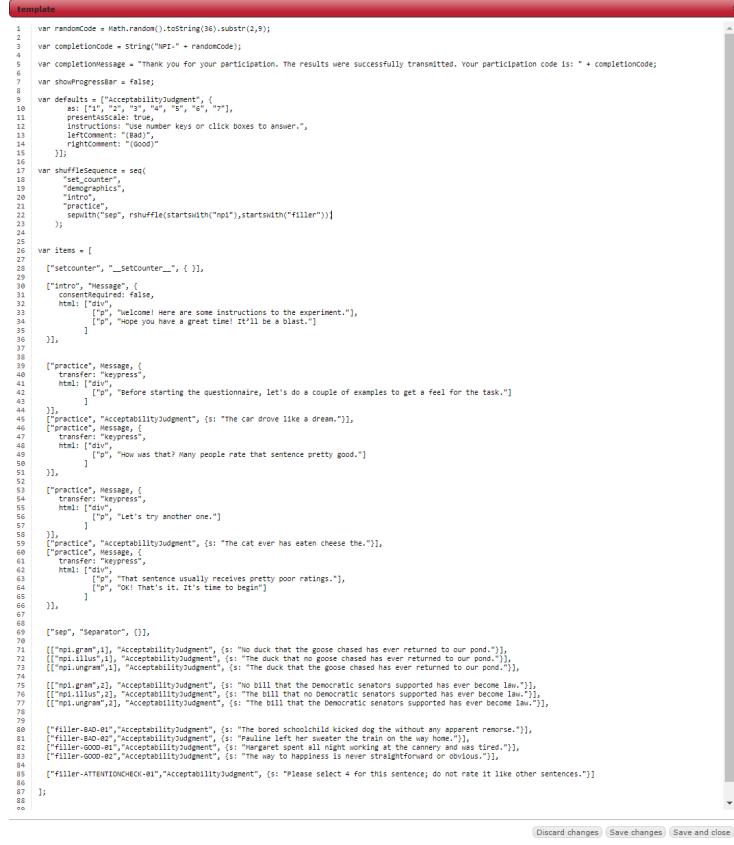
//// Generates random number assigned to participants (Participant ID)
var randomCode = Math.random().toString(36).substr(2,9)
//// A message to show to participants at completion (useful for confirmation, raffle entry, etc.)
var completionMessage = "Thank you for your participation. Your participation code is: " +
randomCode

//// Show a progress bar at the top? (true/false)
var showProgressBar = false

//// Override default settings for controllers (parameters go inside the curly braces { })
var defaults = [
  "AcceptabilityJudgment", {
    as: ["1", "2", "3", "4", "5", "6", "7"],
    presentAsScale: true,
    instructions: "Use number keys or click boxes to answer.",
    leftComment: "(Bad)",
    rightComment: "(Good)"
  }
]
```

The above script creates this acceptability rating experiment.

# A simple layout



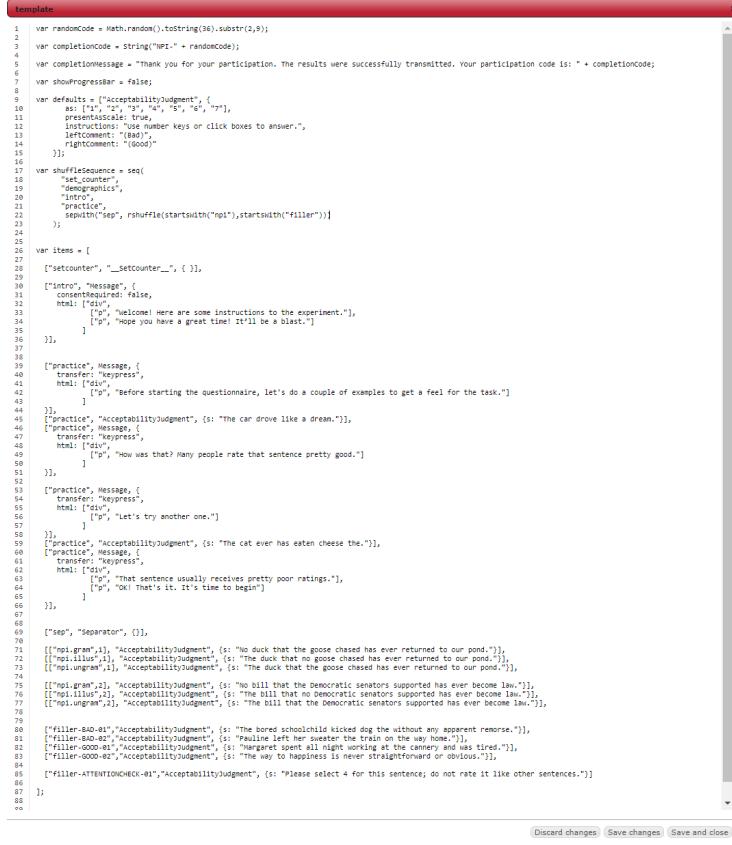
The screenshot shows a text editor window with a red header bar containing the word "template". The main area contains a large amount of JavaScript code. At the bottom of the editor, there are three buttons: "Discard changes", "Save changes", and "Save and close".

```
template
1 var randomCode = Math.random().toString(36).substr(2,9);
2 var completionCode = String("Hpt-") + randomCode;
3 var completionMessage = "thank you for your participation. The results were successfully transmitted. Your participation code is: " + completionCode;
4 var showProgressBar = false;
5 var defaults = {"AcceptabilityJudgment": {
6   as: ["1", "2", "3", "4", "5", "6", "7"],
7   presentationScale: true,
8   instructions: "Please enter number keys or click boxes to answer.",
9   leftComment: "(bad)",
10  rightComment: "(good)"
11 }};
12
13 var shuffleSequence = seq(
14   "set_counter",
15   "demographics",
16   "intro",
17   "practice",
18   sepWith("sep", rshuffle(startsWith("np1"), startsWith("file1")))
19 );
20
21 var items = [
22   {"setcounter": "__setcounter__", {}},
23
24   ["setcounter", {"_setcounter": true}, {}],
25
26   ["intro", "Message", {
27     consentRequired: false,
28     html: ["div", {
29       "p", "Welcome! Here are some instructions to the experiment.", {
30         "p", "I hope you have a great time! It'll be a blast."
31       }
32     }],
33
34   },
35
36   [{"practice", "Message", {
37     transfer: "keypress",
38     html: ["div", {
39       "p", "After starting the questionnaire, let's do a couple of examples to get a feel for the task.", {
40         "p", "1"
41       }
42     }],
43   },
44   {"practice", "AcceptabilityJudgment", {s: "The car drove like a dream."}},
45   {"practice", "Message", {
46     transfer: "keypress",
47     html: ["div", {
48       "p", "How was that? Many people rate that sentence pretty good.", {
49         "p", "2"
50       }
51     }],
52   },
53   {"practice", "Message", {
54     transfer: "keypress",
55     html: ["div", {
56       "p", "Let's try another one.", {
57         "p", "3"
58       }
59     }],
60   },
61   {"practice", "AcceptabilityJudgment", {s: "The cat has eaten cheese the."}},
62   {"practice", "Message", {
63     transfer: "keypress",
64     html: ["div", {
65       "p", "That sentence usually receives pretty poor ratings.", {
66         "p", "OK! That's it. It's time to begin."
67       }
68     }],
69   },
70
71   ["sep", "Separator", {}],
72
73   [{"np1", "Message", "AcceptabilityJudgment", {s: "No duck that the goose chased has ever returned to our pond."}},
74   {"np1", "Message", "AcceptabilityJudgment", {s: "The duck that no goose chased has ever returned to our pond."}},
75   {"np1", "Message", "AcceptabilityJudgment", {s: "No bill that the Democratic senators supported has ever become law."}},
76   {"np1", "Message", "AcceptabilityJudgment", {s: "The bill that no Democratic senators supported has ever become law."}},
77   {"np1", "Message", "AcceptabilityJudgment", {s: "The bill that the Democratic senators supported has ever become law."}},
78
79   {"filler-BAD-Bill", "AcceptabilityJudgment", {s: "The bored schoolchild killed dog the without any apparent remorse."}},
80   {"filler-BAD-Bill", "AcceptabilityJudgment", {s: "Pauline left her sweater the tying on the way home."}},
81   {"filler-GOOD-Bill", "AcceptabilityJudgment", {s: "Margaret spent all night working at the cannery and was tired."}},
82   {"filler-GOOD-Bill", "AcceptabilityJudgment", {s: "The way to happiness is never straightforward or obvious."}},
83
84   [{"filler-ATTENTIONCHECK-B1", "AcceptabilityJudgment", {s: "please select 4 for this sentence; do not rate it like other sentences."}}]
85
86   ],
87 ];
88
89
90
91
92
93
94
95
96
97
98
99
99 ];
```

## Editing the script

When you open the `.js` file in the `data_includes` section of your experiment on Ibex, it will open up a text editor.

# A simple layout



```
template
1 var randomCode = Math.random().toString(36).substr(2,9);
2 var completionCode = String("Hpt-") + randomCode;
3 var completionMessage = "Thank you for your participation. The results were successfully transmitted. Your participation code is: " + completionCode;
4 var showProgressBar = false;
5 var defaults = {
6   "AcceptabilityJudgment": {
7     "as": ["1", "2", "3", "4", "5", "6", "7"],
8     "presentScale: true,
9     "instructions": "Please enter the number keys or click boxes to answer.",
10    "leftComment: "(bad)",
11    "rightComment: "(good)"
12  }
13 };
14
15 var shuffleSequence = seq(
16   "set_counter",
17   "demographics",
18   "intro",
19   "practice",
20   "rshuffle(startswith("npt"), startswith("file"))"
21 );
22
23
24
25 var items = [
26   {"setcounter": "__setcounter__", {}},
27
28   {"setcounter", "Message", {
29     consentRequired: false,
30     html: ["div", "p", "Welcome! Here are some instructions to the experiment."],
31     ["p", "Hope you have a great time! It'll be a blast."]
32   }},
33
34
35
36
37
38   {"practice", "Message", {
39     transfer: "keypress",
40     html: ["div", "p", "Before starting the questionnaire, let's do a couple of examples to get a feel for the task."]
41     ["/p"]
42   }},
43
44   {"practice", "AcceptabilityJudgment", {s: "The car drove like a dream."}},
45   {"practice", "Message", {
46     transfer: "keypress",
47     html: ["div", "p", "How was that? Many people rate that sentence pretty good."]
48     ["/p"]
49   }},
50
51
52   {"practice", "Message", {
53     transfer: "keypress",
54     html: ["div", "p", "Let's try another one."]
55     ["/p"]
56   }},
57
58   {"practice", "AcceptabilityJudgment", {s: "The cat ever has eaten cheese the."}},
59   {"practice", "Message", {
60     transfer: "keypress",
61     html: ["div", "p", "That sentence usually receives pretty poor ratings."]
62     ["/p"]
63   }},
64
65
66
67
68
69   {"sep", "Separator", {}},
70
71   [{"npl:gram1}], "AcceptabilityJudgment", {s: "No duck that the goose chased has ever returned to our pond."}),
72   [{"npl:illuv1}], "AcceptabilityJudgment", {s: "The duck that no good chased has ever returned to our pond."}),
73   [{"npl:illuv2}], "AcceptabilityJudgment", {s: "No duck that the goose chased has ever returned to our pond."}),
74
75   [{"npl:gram2}], "AcceptabilityJudgment", {s: "No bill that the Democratic senators supported has ever become law."}),
76   [{"npl:illuv3}], "AcceptabilityJudgment", {s: "The bill that no Democratic senators supported has ever become law."}),
77   [{"npl:illuv4}], "AcceptabilityJudgment", {s: "The bill that the Democratic senators supported has ever become law."}),
78
79
80   [{"filler-BAD-B1"}, "AcceptabilityJudgment", {s: "The bored schoolchild killed dog the without any apparent remorse."}],
81   [{"filler-BAD-B2"}, "AcceptabilityJudgment", {s: "Pauline left her sweater the tying on the way home."}],
82   [{"filler-GOOD-W1"}, "AcceptabilityJudgment", {s: "Margaret spent all night working at the cannery and was tired."}],
83   [{"filler-GOOD-W2"}, "AcceptabilityJudgment", {s: "The way to happiness is never straightforward or obvious."}],
84
85
86   [{"filler-ATTENTIONCHECK-B1"}, "AcceptabilityJudgment", {s: "Please select 4 for this sentence; do not rate it like other sentences."}]
87
88
89
90
91
92
93
94
95
96
97
98
99
99 ];
```

## Editing the script

When you open the `.js` file in the `data_includes` section of your experiment on Ibex, it will open up a text editor.

Options for editing the file:

- Make changes directly on IBEX
- Download the file and open with an **editor** that supports **JavaScript** syntax checking (e.g., Atom)

# A simple layout

# Parts of the Script

## **Settings:**

- Sets various options for the experiment

**Sequence (shuffleSequence):**

- Specifies the ordering of the different parts of the experiment

## Body (*items*):

- Includes the actual material that will be shown to the participants

# A simple layout

The screenshot shows a software interface for creating experimental stimuli. A large block of JSON-like template code is displayed, organized into several sections:

- Settings:** Contains configuration for a sequence of trials.
- Sequence:** Contains trial definitions.
- Body:** Contains general body text.
- Forms:** Contains form-related code.
- Practice Trials:** Contains practice examples.
- Critical Trials:** Contains critical trials.
- Filler Trials:** Contains filler trials.

The code includes various trial types such as "AcceptabilityJudgment", "Message", and "Filler". It also includes logic for randomizing sequences and generating completion codes.

```
template
1 var randomCode = Math.random().toString(36).substr(2,9);
2 var completionCode = string("http://" + randomCode);
3 var completionMessage = "thank you for your participation. The results were successfully transmitted. Your participation code is: " + completionCode;
4 var showProgressBar = false;
5 var defaults = {"AcceptabilityJudgment": {
6   es: ["1", "2", "3", "4", "5", "6", "7"],
7   presentedScale: true,
8   instructions: "Please enter number keys or click boxes to answer.",
9   leftComment: "(Bad)",
10  rightComment: "(Good)"
11 }};
12
13 var shuffleSequence = seq(
14   "set_counter",
15   "demographics",
16   "intro",
17   "practice",
18   sepWith("sep", rshuffle(startsWith("np1"), startsWith("file1")));
19 );
20
21 var items = [
22   {"setcounter": "_SetCounter__", ()},
23   {"intro": "Message", {
24     consentRequired: false,
25     html: ["div", {
26       ["p", "Welcome! Here are some instructions to the experiment."],
27       ["p", "I hope you have a great time! It'll be a blast!"]
28     }, ()]
29   }},
30   {"practice": "Message", {
31     transfer: "express",
32     html: ["div", {
33       ["p", "After starting the questionnaire, let's do a couple of examples to get a feel for the task."]
34     }, ()]
35   }},
36   {"practice": "AcceptabilityJudgment", {s: "The car drove like a dream."}},
37   {"practice": "Message", {
38     transfer: "express",
39     html: ["div", {
40       ["p", "How was that? Many people rate that sentence pretty good."]
41     }, ()]
42   }},
43   {"practice": "Message", {
44     transfer: "express",
45     html: ["div", {
46       ["p", "Let's try another one."]
47     }, ()]
48   }},
49   {"practice": "AcceptabilityJudgment", {s: "The cat ever has eaten cheese the."}},
50   {"practice": "Message", {
51     transfer: "express",
52     html: ["div", {
53       ["p", "That sentence usually receives pretty poor ratings."]
54     }, ()]
55   }},
56   {"step": "separator", ()},
57   [{"np1.gran1", "AcceptabilityJudgment", {s: "No duck that the goose chased has ever returned to our pond."}},
58   {"np1.llow1", "AcceptabilityJudgment", {s: "The duck that no goose chased has ever returned to our pond."}},
59   {"np1.llow2", "AcceptabilityJudgment", {s: "No duck that the goose chased has ever returned to the pond."}},
60   {"np1.gran2", "AcceptabilityJudgment", {s: "No bill that the Democratic senators supported has ever become law."}},
61   {"np1.llow3", "AcceptabilityJudgment", {s: "The bill that no Democratic senators supported has ever become law."}},
62   {"np1.llow4", "AcceptabilityJudgment", {s: "The bill that the Democratic senators supported has ever become law."}},
63   {"filler-BAD-01", "AcceptabilityJudgment", {s: "The seven schoolchild kicked dog the without any apparent remorse."}},
64   {"filler-BAD-02", "AcceptabilityJudgment", {s: "Pauline left home alone in the dark night and has been lost."}},
65   {"filler-GOOD-01", "AcceptabilityJudgment", {s: "Margaret spent all night working at the cattery and was tired."}},
66   {"filler-GOOD-02", "AcceptabilityJudgment", {s: "The way to happiness is never straightforward or obvious."}},
67   {"filler-ATTENTIONCHECK-01", "AcceptabilityJudgment", {s: "Please select 4 for this sentence; do not rate it like other sentences."}}
68 ],
69 ]
70 ]
```

**Buttons at the bottom:** Discard changes | Save changes | Save and close

## Body

## Forms

- Introduction page, consent form, directions, language background, demographic information, etc.

## Trials

- All stimuli for the experiment
- *Practice, Critical, Filler*
- Can be accompanied by messages, questions, etc.

# Walkthrough of the components

The diagram illustrates the structure of a template file, likely for an experiment or survey. The code is organized into several sections:

- Settings:** Contains variables like randomCode, completionCode, and defaults.
- Sequence:** Contains a shuffleSequence array.
- Body:** Contains items and their corresponding HTML and transfer types.
- Forms:** Contains practice examples and acceptability judgments.
- Practice Trials:** Contains practice examples and acceptability judgments.
- Critical Trials:** Contains critical trials and acceptability judgments.
- Filler Trials:** Contains filler trials and acceptability judgments.

```
template
1 var randomCode = Math.random().toString(36).substr(2,9);
2 var completionCode = String("NPI-" + randomCode);
3
4 var completionMessage = "Thank you for your participation. The results were successfully transmitted. Your participation code is: " + completionCode;
5
6 var showProgressBar = false;
7
8 var defaults = ["AcceptabilityJudgment", {
9   as: ["1", "2", "3", "4", "5", "6", "7"],
10  presentation: "Use number keys or click boxes to answer.",
11  instruction: "Use number keys or click boxes to answer.",
12  leftComment: "(Bad)",
13  rightComment: "(Good)"
14 }];
15
16 var shuffleSequence = seq(
17   "set_counter",
18   "demographics",
19   "intro",
20   "practice",
21   "sep",
22   sepWith("sep", rshuffle(startsWith("npi"),startsWith("filler")));
23 );
24
25 var items = [
26
27   ["setcounter", "..._setcounter...", {}],
28
29   ["intro", "Message", {
30     consentRequired: false,
31     html: ["div",
32       ["p", "Welcome! Here are some instructions to the experiment."],
33       ["p", "Hope you have a great time! It'll be a blast."]
34     ],
35   }],
36
37
38   ["practice", Message, {
39     transfer: "keypress",
40     html: ["div",
41       ["p", "Before starting the questionnaire, let's do a couple of examples to get a feel for the task."]
42     ]
43   }],
44
45   ["practice", "AcceptabilityJudgment", {s: "The car drove like a dream."}],
46   ["practice", Message, {
47     transfer: "keypress",
48     html: ["div",
49       ["p", "How was that? Many people rate that sentence pretty good."]
50     ],
51   }],
52
53   ["practice", Message, {
54     transfer: "keypress",
55     html: ["div",
56       ["p", "Let's try another one."]
57     ],
58   }],
59   ["practice", "AcceptabilityJudgment", {s: "The cat ever has eaten cheese the."}],
60   ["practice", Message, {
61     transfer: "keypress",
62     html: ["div",
63       ["p", "That sentence usually receives pretty poor ratings."],
64       ["p", "OK! That's it. It's time to begin"]
65     ],
66   }],
67
68
69   ["sep", "Separator", {}],
70
71   [{"npi": "gram", 1}, "AcceptabilityJudgment", {s: "No duck that the goose chased has ever returned to our pond."}],
72   [{"npi": "illus", 1}, "AcceptabilityJudgment", {s: "The duck that no goose chased has ever returned to our pond."}],
73   [{"npi": "ungram", 1}, "AcceptabilityJudgment", {s: "The duck that the goose chased has ever returned to our pond."}],
74
75   [{"npi": "gram", 2}, "AcceptabilityJudgment", {s: "No bill that the Democratic senators supported has ever become law."}],
76   [{"npi": "illus", 2}, "AcceptabilityJudgment", {s: "The bill that no Democratic senators supported has ever become law."}],
77   [{"npi": "ungram", 2}, "AcceptabilityJudgment", {s: "The bill that the Democratic senators supported has ever become law."}],
78
79
80   [{"filler": "BAD-01"}, "AcceptabilityJudgment", {s: "The bored schoolchild kicked the dog the without any apparent remorse."}],
81   [{"filler": "BAD-02"}, "AcceptabilityJudgment", {s: "Pauline left her sweater the train on the way home."}],
82   [{"filler": "GOOD-01"}, "AcceptabilityJudgment", {s: "Margaret spent all night working at the cannery and was tired."}],
83   [{"filler": "GOOD-02"}, "AcceptabilityJudgment", {s: "The way to happiness is never straightforward or obvious."}],
84
85   [{"filler": "ATTENTIONCHECK-01"}, "AcceptabilityJudgment", {s: "Please select 4 for this sentence; do not rate it like other sentences."}]
86
87 ];
88
89 ];
```

Discard changes Save changes Save and close

# Our first experiment

**Study:** We are interested in how people recover from **garden-path sentences**.

"While Anna dressed the kitten paid attention."

... \*[<sub>VP</sub> dressed the kitten], ...

... √[<sub>VP</sub> dressed], the kitten ...

**Hypothesis:** Verbs that are frequently **transitive** make recovery more difficult, compared to verbs that are frequently **intransitive**.

(transitive-biased) - "While Anna **trained** the kitten paid attention."

(intransitive-biased) - "While Anna **dressed** the kitten paid attention."

**Prediction:** Lower **acceptability ratings** in the transitive-biased condition (**gp.trans**) than in the intransitive-biased condition (**gp.intrans**).

**Note:** this is a *within-participant* design!

# Trial Syntax

For each trial in our acceptability judgment experiment, we write this code:

```
[["Trial name", Set #], "Trial Type", {s: "Sentence"}]
```

This is a list (array) of three elements (clearer with spacing):

```
[  
  ["Trial name", Set # ],  
  "Trial Type",  
  {s: "Sentence" }  
]
```

```
[ ["Trial name", Set # ],  
  "Trial Type",  
  {s: "Sentence" } ]
```

We have one big bracket which contains **three elements**:

1. Another list, consisting of the *name of the trial* and *set number*
2. A string specifying the *type of the trial* (called **Controllers**)
3. A curly bracket (*braces*), which has "s" and the *sentence* separated by a colon

# Translating the design to code

## Example 1:

For the gp.trans condition, the first stimuli in our acceptability judgment experiment is the sentence "While Anna trained the kitten paid attention".

```
[  
  [ "gp.trans", 1 ],  
  "AcceptabilityJudgment",  
  {s: "While Anna trained the kitten paid attention."}  
]
```

# Translating the design to code

## Example 1:

For the **gp.trans condition**, the first stimuli in our acceptability judgment experiment is the sentence "While Anna trained the kitten paid attention".

```
[  
  [ "gp.trans", 1 ],  
  "AcceptabilityJudgment",  
  {s: "While Anna trained the kitten paid attention."}  
]
```

# Translating the design to code

## Example 1:

For the **gp.trans condition**, the **first stimuli** in our acceptability judgment experiment is the sentence "While Anna trained the kitten paid attention".

```
[  
  [ "gp.trans", 1 ],  
  "AcceptabilityJudgment",  
  {s: "While Anna trained the kitten paid attention."}  
]
```

# Translating the design to code

## Example 1:

For the **gp.trans condition**, the **first stimuli** in our **acceptability judgment experiment** is the sentence "While Anna trained the kitten paid attention".

```
[  
  [ "gp.trans", 1 ],  
  "AcceptabilityJudgment",  
  {s: "While Anna trained the kitten paid attention."}  
]
```

# Translating the design to code

## Example 1:

For the **gp.trans condition**, the **first stimuli** in our **acceptability judgment experiment** is the **sentence** "While Anna trained the kitten paid attention".

```
[  
  [ "gp.trans", 1 ],  
  "AcceptabilityJudgment",  
  {s: "While Anna trained the kitten paid attention."}  
]
```

# Translating the design to code

## Example 1:

For the **gp.trans condition**, the **first stimuli** in our **acceptability judgment experiment** is the **sentence "While Anna trained the kitten paid attention."**.

```
[  
  [ "gp.trans", 1 ],  
  "AcceptabilityJudgment",  
  {s: "While Anna trained the kitten paid attention."}  
]
```

This can be put into a single line:

```
[["gp.trans",1], "AcceptabilityJudgment", {s: "While Anna trained the k-
```

# Translating the design to code

## Example 2:

For the **gp.intrans** condition, the **first stimuli** in our **acceptability judgment experiment** is the **sentence "While Anna dressed the kitten paid attention"**.

```
[  
  [ "gp.intrans", 1 ],  
  "AcceptabilityJudgment",  
  {s: "While Anna dressed the kitten paid attention."}  
]
```

This can be put into a single line:

```
[["gp.intrans",1], "AcceptabilityJudgment", {s: "While Anna dressed the
```

# Practice

Item set #2:

(transitive) - "Since Dave improved the department was satisfied."

(intransitive) - "Since Dave worried the counselor devised a plan."

```
<code here>
```

Hover for answer

# Putting together the stimuli

Wrap in `var items = []` and separate by a **comma**:

```
var items = [  
  
    ///////////////////////////////////////////////////////////////////  
    // Set #1  
    [{"gp.trans",1}, "AcceptabilityJudgment", {s: "While Anna trained the kitten paid attention."}],  
    [{"gp.intrans",1}, "AcceptabilityJudgment", {s: "While Anna dressed the kitten paid attention."}],  
  
    ///////////////////////////////////////////////////////////////////  
    // Set #2  
    [{"gp.trans",2}, "AcceptabilityJudgment", {s: "Since Dave improved the department was satisfied."}],  
    [{"gp.intrans",2}, "AcceptabilityJudgment", {s: "Since Dave worried the counselor devised a plan."}]  
  
]
```

Good scripting habits:

- Grouping item sets together and separating sets with new line
- Adding comments (starts with two or more slashes `//`)
- Saving often! (editing in IBEX not recommended)

# Practice and Fillers

## Practice Trials:

- Presented at the beginning, accompanied by instructions and feedback

## Filler Trials:

- Mixed in with the critical trials

**Both types of trials are invariant across conditions**

This means that the first element of the big bracket can just be the *Trial Name*, without assigning it a *Set #*:

---

Practice      Filler-good      Filler-bad      Filler-catch

---

Practice #1: "The car drove like a dream"

```
["practice-1",
"AcceptabilityJudgment",
{s: "The car drove like a dream."}]
```

# Practice and Fillers

## Practice Trials:

- Presented at the beginning, accompanied by instructions and feedback

## Filler Trials:

- Mixed in with the critical trials

**Both types of trials are invariant across conditions**

This means that the first element of the big bracket can just be the *Trial Name*, without assigning it a *Set #*:

Practice    Filler-good    Filler-bad    Filler-catch

Good Filler #1: "When Harry fell, the audience was shocked."

```
["filler-good-01",
"AcceptabilityJudgment",
{s: "When Harry fell, the audience was shocked."}]
```

# Practice and Fillers

## Practice Trials:

- Presented at the beginning, accompanied by instructions and feedback

## Filler Trials:

- Mixed in with the critical trials

**Both types of trials are invariant across conditions**

This means that the first element of the big bracket can just be the *Trial Name*, without assigning it a *Set #*:

Practice      Filler-good      Filler-bad      Filler-catch

Bad Filler #1: "When Tyler sneezed the driver, he passed a tissue."

```
["filler-bad-01",
"AcceptabilityJudgment",
{s: "When Tyler sneezed the driver, he passed a tissue."}]
```

# Practice and Fillers

## Practice Trials:

- Presented at the beginning, accompanied by instructions and feedback

## Filler Trials:

- Mixed in with the critical trials

**Both types of trials are invariant across conditions**

This means that the first element of the big bracket can just be the *Trial Name*, without assigning it a *Set #*:

---

Practice      Filler-good      Filler-bad      Filler-catch

---

Catch Filler #1: "Please select 4 for this sentence."

```
["filler-catch-01",
"AcceptabilityJudgment",
{s: "Please select 4 for this sentence."}]
```

# Putting together the Body

```
var items = [  
  
    ///////////////////////////////////////////////////////////////////  
    // Practice  
    ["practice-1", "AcceptabilityJudgment", {s: "The car drove like a dream."}],  
  
    // Critical Trials //  
  
    ///////////////////////////////////////////////////////////////////  
    // Set #1  
    [{"gp.trans",1}, "AcceptabilityJudgment", {s: "While Anna trained the kitten paid attention."}],  
    [{"gp.intrans",1}, "AcceptabilityJudgment", {s: "While Anna dressed the kitten paid attention."}],  
  
    ///////////////////////////////////////////////////////////////////  
    // Set #2  
    [{"gp.trans",2}, "AcceptabilityJudgment", {s: "Since Dave improved the department was satisfied."}],  
    [{"gp.intrans",2}, "AcceptabilityJudgment", {s: "Since Dave worried the counselor devised a plan."}],  
  
    ///////////////////////////////////////////////////////////////////  
    // Fillers (Good)  
    ["filler-good-01","AcceptabilityJudgment", {s: "When Harry fell, the audience was shocked."}],  
  
    ///////////////////////////////////////////////////////////////////  
    // Fillers (Bad)  
    ["filler-bad-01","AcceptabilityJudgment", {s: "When Tyler sneezed the driver, he passed a tissue."}],  
  
    ///////////////////////////////////////////////////////////////////  
    // Fillers (Catch)  
    ["filler-catch-01", "AcceptabilityJudgment", {s: "Please select 4 for this sentence."}]
```

**Important:** The ordering of the trials here is just for human readability. We haven't yet told the program what order to present them in!

# Defining the Sequence

```
var items = [  
  
    // Practice  
    ["practice-1", "AcceptabilityJudgment",  
     {s: "The car drove like a dream."}],  
  
    // Critical Trials //  
  
    // Set #1  
    [[{"gp.trans":1},  
     "AcceptabilityJudgment", {s: "While Anna  
     trained the kitten paid attention."}],  
     [{"gp.intrans":1},  
     "AcceptabilityJudgment", {s: "While Anna  
     dressed the kitten paid attention."}],  
  
    // Set #2  
    [{"gp.trans":2},  
     "AcceptabilityJudgment", {s: "Since Dave  
     improved the department was satisfied."}],  
     [{"gp.intrans":2},  
     "AcceptabilityJudgment", {s: "Since Dave  
     worried the counselor devised a plan."}],  
  
    // Fillers (Good)  
    [{"filler-good-  
     01","AcceptabilityJudgment", {s: "When  
     Harry fell, the audience was shocked."}}],
```

**List the *names* of each trial in order:**

```
"practice-1",  
"gp.trans",  
"gp.intrans",  
"gp.trans",  
"gp.intrans",  
"filler-good-01",  
"filler-bad-01",  
"filler-catch-01"
```

# Defining the Sequence

```
var items = [  
  
    // Practice  
    ["practice-1", "AcceptabilityJudgment",  
     {s: "The car drove like a dream."}],  
  
    // Critical Trials //  
  
    // Set #1  
    [[{"gp.trans":1},  
     "AcceptabilityJudgment", {s: "While Anna  
     trained the kitten paid attention."}],  
     [{"gp.intrans":1},  
     "AcceptabilityJudgment", {s: "While Anna  
     dressed the kitten paid attention."}],  
  
    // Set #2  
    [{"gp.trans":2},  
     "AcceptabilityJudgment", {s: "Since Dave  
     improved the department was satisfied."}],  
     [{"gp.intrans":2},  
     "AcceptabilityJudgment", {s: "Since Dave  
     worried the counselor devised a plan."}],  
  
    // Fillers (Good)  
    [{"filler-good-  
     01","AcceptabilityJudgment", {s: "When  
     Harry fell, the audience was shocked."}}],
```

**Wrap them in `seq()`:**

```
seq(  
  "practice-1",  
  "gp.trans",  
  "gp.intrans",  
  "gp.trans",  
  "gp.intrans",  
  "filler-good-01",  
  "filler-bad-01",  
  "filler-catch-01"  
)
```

# Defining the Sequence

```
var items = [  
  
    // Practice  
    ["practice-1", "AcceptabilityJudgment",  
     {s: "The car drove like a dream."}],  
  
    // Critical Trials //  
  
    // Set #1  
    [{"gp.trans",1},  
     "AcceptabilityJudgment", {s: "While Anna  
trained the kitten paid attention."}],  
    [{"gp.intrans",1},  
     "AcceptabilityJudgment", {s: "While Anna  
dressed the kitten paid attention."}],  
  
    // Set #2  
    [{"gp.trans",2},  
     "AcceptabilityJudgment", {s: "Since Dave  
improved the department was satisfied."}],  
    [{"gp.intrans",2},  
     "AcceptabilityJudgment", {s: "Since Dave  
worried the counselor devised a plan."}],  
  
    // Fillers (Good)  
    ["filler-good-  
     01","AcceptabilityJudgment", {s: "When  
Harry fell, the audience was shocked."}],
```

## Assign to **shuffleSequence**:

```
var shuffleSequence = seq(  
    "practice-1",  
    "gp.trans",  
    "gp.intrans",  
    "gp.trans",  
    "gp.intrans",  
    "filler-good-01",  
    "filler-bad-01",  
    "filler-catch-01"  
)
```

# Defining the Sequence

```
var items = [  
  
    // Practice  
    ["practice-1", "AcceptabilityJudgment",  
     {s: "The car drove like a dream."}],  
  
    // Critical Trials //  
  
    // Set #1  
    [{"gp.trans",1},  
     "AcceptabilityJudgment", {s: "While Anna  
trained the kitten paid attention."}],  
    [{"gp.intrans",1},  
     "AcceptabilityJudgment", {s: "While Anna  
dressed the kitten paid attention."}],  
  
    // Set #2  
    [{"gp.trans",2},  
     "AcceptabilityJudgment", {s: "Since Dave  
improved the department was satisfied."}],  
    [{"gp.intrans",2},  
     "AcceptabilityJudgment", {s: "Since Dave  
worried the counselor devised a plan."}],  
  
    // Fillers (Good)  
    ["filler-good-  
     01","AcceptabilityJudgment", {s: "When  
Harry fell, the audience was shocked."}],
```

## Assign to **shuffleSequence**:

```
var shuffleSequence = seq(  
    "practice-1",  
    "gp.trans",  
    "gp.intrans",  
    "gp.trans",  
    "gp.intrans",  
    "filler-good-01",  
    "filler-bad-01",  
    "filler-catch-01"  
)
```

The **shuffleSequence** variable handles the *order of presentation* of the **experiment materials** that are stored inside the **items** variable.

# Defining the Sequence

```
var items = [  
  
    //// Practice  
    ["practice-1", "AcceptabilityJudgment",  
     {s: "The car drove like a dream."}],  
  
    // Critical Trials //  
  
    //// Set #1  
    [{"gp.trans",1},  
     "AcceptabilityJudgment", {s: "While Anna  
trained the kitten paid attention."}],  
    [{"gp.intrans",1},  
     "AcceptabilityJudgment", {s: "While Anna  
dressed the kitten paid attention."}],  
  
    //// Set #2  
    [{"gp.trans",2},  
     "AcceptabilityJudgment", {s: "Since Dave  
improved the department was satisfied."}],  
    [{"gp.intrans",2},  
     "AcceptabilityJudgment", {s: "Since Dave  
worried the counselor devised a plan."}],  
  
    //// Fillers (Good)  
    ["filler-good-  
01","AcceptabilityJudgment", {s: "When  
Harry fell, the audience was shocked."}],
```

## Problems

```
var shuffleSequence = seq(  
    "practice-1",  
    "gp.trans",  
    "gp.intrans",  
    "gp.trans",  
    "gp.intrans",  
    "filler-good-01",  
    "filler-bad-01",  
    "filler-catch-01"  
)
```

1. A lot of typing ("-02", "-03", ...)

# Defining the Sequence

```
var items = [  
  
    // Practice  
    ["practice-1", "AcceptabilityJudgment",  
     {s: "The car drove like a dream."}],  
  
    // Critical Trials //  
  
    // Set #1  
    [{"gp.trans",1},  
     "AcceptabilityJudgment", {s: "While Anna  
trained the kitten paid attention."}],  
    [{"gp.intrans",1},  
     "AcceptabilityJudgment", {s: "While Anna  
dressed the kitten paid attention."}],  
  
    // Set #2  
    [{"gp.trans",2},  
     "AcceptabilityJudgment", {s: "Since Dave  
improved the department was satisfied."}],  
    [{"gp.intrans",2},  
     "AcceptabilityJudgment", {s: "Since Dave  
worried the counselor devised a plan."}],  
  
    // Fillers (Good)  
    ["filler-good-  
01","AcceptabilityJudgment", {s: "When  
Harry fell, the audience was shocked."}],
```

## Problems

```
var shuffleSequence = seq(  
  "practice-1",  
  "gp.trans",  
  "gp.intrans",  
  "gp.trans",  
  "gp.intrans",  
  "filler-good-01",  
  "filler-bad-01",  
  "filler-catch-01"  
)
```

1. A lot of typing ("-02", "-03", ...)
2. Presentation order of some trials  
should be random

# Defining the Sequence

```
var items = [  
  
    //// Practice  
    ["practice-1", "AcceptabilityJudgment",  
     {s: "The car drove like a dream."}],  
  
    // Critical Trials //  
  
    //// Set #1  
    [{"gp.trans",1},  
     "AcceptabilityJudgment", {s: "While Anna  
trained the kitten paid attention."}],  
    [{"gp.intrans",1},  
     "AcceptabilityJudgment", {s: "While Anna  
dressed the kitten paid attention."}],  
  
    //// Set #2  
    [{"gp.trans",2},  
     "AcceptabilityJudgment", {s: "Since Dave  
improved the department was satisfied."}],  
    [{"gp.intrans",2},  
     "AcceptabilityJudgment", {s: "Since Dave  
worried the counselor devised a plan."}],  
  
    //// Fillers (Good)  
    ["filler-good-  
     01","AcceptabilityJudgment", {s: "When  
Harry fell, the audience was shocked."}],
```

## Problems

```
var shuffleSequence = seq(  
  "practice-1",  
  "gp.trans",  
  "gp.intrans",  
  "gp.trans",  
  "gp.intrans",  
  "filler-good-01",  
  "filler-bad-01",  
  "filler-catch-01"  
)
```

1. A lot of typing ("-02", "-03", ...)
2. Presentation order of some trials should be random
3. How do we counterbalance critical trials?

# 1. Sequence: multiple selection

To save us from writing repetitive code, we use `startsWith()`

```
var shuffleSequence = seq(  
  "practice-1",  
  "gp.trans",  
  "gp.intrans",  
  "gp.trans",  
  "gp.intrans",  
  "filler-good-01",  
  "filler-bad-01",  
  "filler-catch-01"  
)
```

```
var shuffleSequence = seq(  
  startsWith("practice"),  
  startsWith("gp")  
  startsWith("filler"))
```

The function `startsWith()` matches all names that starts with the given string.

# 2. Sequence: randomization

To mix critical and filler trials in random order, we use `rshuffle()`

```
var shuffleSequence = seq(  
  startsWith("practice"),  
  startsWith("gp")  
  startsWith("filler")  
)
```

```
var shuffleSequence = seq(  
  startsWith("practice"),  
  rshuffle(  
    startsWith("gp"),  
    startsWith("filler")  
)
```

By wrapping both the critical trials (*gp...*) and the filler trials (*filler...*) in `rshuffle()`, they are mixed together and presented in random order.

We can also write this out more compactly:

```
var shuffleSequence = seq(  
  startsWith("practice"),  
  rshuffle(startsWith("gp"), startsWith("filler"))  
)
```

# 3. Sequence: counterbalancing

The **set number** in our critical trials automatically handles **counterbalancing**:

```
[["gp.trans", 1], ...], [["gp.intrans", 1], ...], [["gp.trans", 2], ...], [["gp.intrans", 2], ...]
```



["gp.trans", 1]

["gp.intrans", 2]



["gp.intrans", 1]

["gp.trans", 2]



["gp.trans", 1]

["gp.intrans", 2]



["gp.intrans", 1]

["gp.trans", 2]

We just need to add a **counter** inside **items** to track group assignment:

```
["setcounter", "__SetCounter__", { }]
```

# 3. Sequence: counterbalancing

To use the counter, you need to modify both `items` and `shuffleSequence`:

```
var items = [
  ...
  ["setcounter", "__SetCounter__", {}], // The counter is defined
  ...
]

var shuffleSequence = seq(
  "setcounter", // The counter is incremented at the start
  ...
)
```

**Note:** Sometimes you want to place "setcounter" in the *middle* of the experiment

```
var shuffleSequence = seq(
  "intro",
  "consent",
  "setcounter",
  ...
)
```

# Body and Sequence together

```
// Presentation Order //
var shuffleSequence = seq(
  "setcounter",
  startsWith("practice"),
  rshuffle(startsWith("gp"), startsWith("filler"))
)

// Experiment Materials //
var items = [
  /////
  // Counter
  ["setcounter", "__SetCounter__", {}],
  /////
  // Practice
  ["practice-1", "AcceptabilityJudgment", {s: "The car drove like a dream."}],
  // Critical Trials //

  /////
  // Set #1
  [[{"gp.trans",1}, "AcceptabilityJudgment", {s: "While Anna trained the kitten paid attention."}],
   [{"gp.intrans",1}, "AcceptabilityJudgment", {s: "While Anna dressed the kitten paid attention."}],
  /////
  // Set #2
  [{"gp.trans",2}, "AcceptabilityJudgment", {s: "Since Dave improved the department was satisfied."}],
  [{"gp.intrans",2}, "AcceptabilityJudgment", {s: "Since Dave worried the counselor devised a
```

Just need one more step: **Settings**

# Settings (Basic)

Consist of miscellaneous options that we can put at the top of the script.

At the very least, we want to do two things:

1. Generate unique **participant IDs**
2. Specify the parameters for the **method design**

# Settings (Basic)

Consist of miscellaneous options that we can put at the top of the script.

At the very least, we want to do two things:

1. Generate unique **participant IDs** (`randomCode`)
2. Specify the parameters for the **method design** (`defaults`)

```
// Defaults and other settings //

//// Generates random number assigned to participants (Participant ID)
var randomCode = Math.random().toString(36).substr(2,9)

//// Override default settings for controllers (parameters go inside the curly braces { })
var defaults = [
    "AcceptabilityJudgment", {
        as: ["1", "2", "3", "4", "5", "6", "7"],
        presentAsScale: true,
        instructions: "Use number keys or click boxes to answer.",
        leftComment: "(Bad)",
        rightComment: "(Good)"
    }
]
```

More details in the **AcceptabilityJudgment** section of the [documentation](#).

# The `defaults` variable

The specifications in `defaults` set the design of the trials:

```
var defaults = [  
  "AcceptabilityJudgment", {  
    as: ["1", "2", "3", "4", "5", "6", "7"],  
    presentAsScale: true,  
    instructions: "Use number keys or click boxes to answer.",  
    leftComment: "(Bad)",  
    rightComment: "(Good)"  
  }]
```

While Anna trained the kitten paid attention.

(Bad)        (Good)

*Use number keys or click boxes to answer.*

# Another defaults example

Suppose you'd like to show participants *multiple sentences* but have them only rate the acceptability of **one** of the sentences.

Code      Output

```
var defaults = [
  "AcceptabilityJudgment", {
    ...
    instructions: "Rate how natural Speaker B's response sounds.",
    ...
  }]
]
```

```
[["ConditionA", 1],
 "AcceptabilityJudgment",
 {s: ["div",
       ["p", "Speaker A: Who left this sandwich on the table?"]
       ["p", "Speaker B: Fred did."]
     ]}]
]
```

**Note:** You can show multiple lines of text using **html** tags like "p" and "div" - more on that later

# Another defaults example

Suppose you'd like to show participants *multiple sentences* but have them only rate the acceptability of **one** of the sentences.

Code

Output

Speaker A: Who left this sandwich on the table?

Speaker B: Fred did.

(Bad)

 1 2 3 4 5 6 7

(Good)

*Rate how natural Speaker B's response sounds.*

# Settings (Miscellaneous)

You can also change other options, such as showing a message at the end:

```
var completionMessage
```

```
//// A message to show to participants at completion (useful for confirmation)
var completionMessage = "Thank you for your participation. Your participation is important to us."
```

And whether to show a progress bar:

```
var showProgressBar
```

```
//// Show a progress bar at the top? (true/false)
var showProgressBar = false
```

You can learn more about these various elements in the **Miscellaneous options** section of the [documentation](#).

# A minimal experiment

We now have a minimally working experiment!

```
// Defaults and other settings //

//// Generates random number assigned to participants (Participant ID)
var randomCode = Math.random().toString(36).substr(2,9)
//// A message to show to participants at completion (useful for confirmation, raffle entry, etc.)
var completionMessage = "Thank you for your participation. Your participation code is: " +
randomCode

//// Show a progress bar at the top? (true/false)
var showProgressBar = false

//// Override default settings for controllers (parameters go inside the curly braces { })
var defaults = [
  "AcceptabilityJudgment", {
    as: ["1", "2", "3", "4", "5", "6", "7"],
    presentAsScale: true,
    instructions: "Use number keys or click boxes to answer.",
    leftComment: "(Bad)",
    rightComment: "(Good)"
  }
]

// Presentation Order //

var shuffleSequence = seq(
  "setcounter",
```

# Interim Summary #1

## What we've covered:

- We use a special syntax to create stimuli
- We store all the materials for our experiment inside `items`
- We specify the order of presentation inside `shuffleSequence`
- We set various options at the top of the script, such as defaults for the experiment *method* and the assignment of *participant ID*

## A few more things to know:

- How can we add *plain text*?
  - Introduction page, consent form, directions, etc.
- How can we extend this workflow for *other experimental designs*?
  - Self-paced reading, comprehension tasks, etc.

# The "Message" controller

The "**Message**" controller shows text on a new page.

It is a list of 3 elements, similar to the "AcceptabilityJudgement" items:

**[*Trial name*", "Message", {html: *text*}]**

- ***Trial name*** is used to reference the trial in sequencing, as seen earlier
- "**Message**" tells IBEX that this trial just shows text on a new screen
- Inside of the curly braces **{}** we can add text in the **html** parameter:

```
["intro", "Message", {html: ["p", "Welcome to the experiment!"]}]
```

Notes on **html**:

The code **["p", "<your text here>"]** prints a single paragraph of text.

The "p" is called an **HTML tag** and there are many others (most common: "div", "strong", "em"), but usually these don't get very complicated.

# Message examples

---

1-paragraph

n-paragraphs

consent

keypress

separator

A message composed of a single paragraph:

```
["intro", "Message", {html: ["p", "Welcome to the experiment!"]}]
```

Welcome to the experiment!

→ [Click here to continue.](#)

# Message examples

1-paragraph

n-paragraphs

consent

keypress

separator

A message composed of multiple paragraphs must be combined inside a "div":

```
["intro", "Message", {html:  
    ["div",  
        ["p", "Welcome to the experiment!"],  
        ["p", "Here's another paragraph."],  
        ["p", "These paragraphs are all wrapped inside \"div\"."]  
    ]  
}]
```

Welcome to the experiment!

Here's another paragraph.

These paragraphs are all wrapped inside "div".

→ [Click here to continue.](#)

**Note:** You can **escape** special characters like quotes **"** with a backslash \

# Message examples

1-paragraph

n-paragraphs

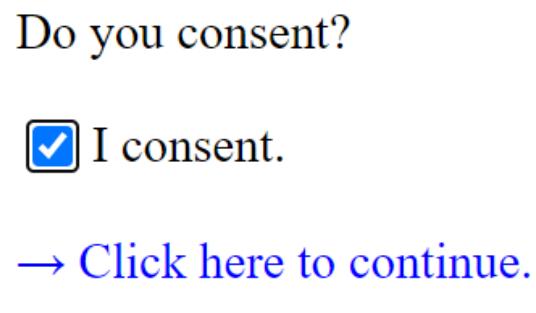
consent

keypress

separator

Ask for consent with a set of `consent...` parameters

```
["consent", "Message", {  
  html: ["p", "Do you consent?"],  
  consentRequired: true,  
  consentMessage: "I consent."  
}]
```



# Message examples

1-paragraph

n-paragraphs

consent

keypress

separator

Use the **transfer** argument to specify how the participant can move on.

Setting **transfer** to "keypress" removes the default "**Click here to continue.**" message and allows participants to proceed with the press of any key.

```
["move_on", "Message", {  
  html: ["div",  
    ["p", "The option for \"transfer\" is \"keypress\""],  
    ["em", "Press any key to continue."]  
],  
  transfer: "keypress"  
}]
```

The option for "transfer" is "keypress"

*Press any key to continue.*

# Message examples

---

1-paragraph

n-paragraphs

consent

keypress

separator

You might want to insert a page that separates the trials:

```
["sep", "Message", {  
  html: ["em", "Press any key to continue."],  
  transfer: "keypress"  
}]
```

*Press any key to continue.*

You can do so using `sepWith()` in `shuffleSequence`:

```
var shuffleSequence = seq(  
  "practice",  
  sepWith("sep", rshuffle(startsWith("gp"), startsWith("filler")))  
)
```

# Putting everything together

Here's a complete experiment with several **Message** controllers added.

```
// Defaults and other settings //

//// Generates random number assigned to participants (Participant ID)
var randomCode = Math.random().toString(36).substr(2,9)
//// A message to show to participants at completion (useful for confirmation, raffle entry, etc.)
var completionMessage = "Thank you for your participation. Your participation code is: " +
randomCode

//// Show a progress bar at the top? (true/false)
var showProgressBar = false

//// Override default settings for controllers (parameters go inside the curly braces { })
var defaults = [
  "AcceptabilityJudgment", {
    as: ["1", "2", "3", "4", "5", "6", "7"],
    presentAsScale: true,
    instructions: "Use number keys or click boxes to answer.",
    leftComment: "(Bad)",
    rightComment: "(Good)"
  }
]

// Presentation Order //

var shuffleSequence = seq(
  "intro",
  "consent",
```

# Getting the experiment up

So we have a new script, but how do we host the experiment?

1. Go to the IBEX website and log in.
2. Click on your experiment (or create one if you haven't already).
3. Click **edit** next to the *example\_data.js* file in the **data\_includes** section.
4. Delete its contents and copy paste your new code.
5. Click on the link at the top of the page.

/ibexexprs/skku\_ibex/workshop\_acceptabilityJudgment/experiment.html

The URL shown in your browser is the link to your experiment!



[https://spellout.net/ibexexprs/skku\\_ibex/workshop\\_minimal/experiment.html](https://spellout.net/ibexexprs/skku_ibex/workshop_minimal/experiment.html)

# A different experiment

Suppose that after a *pilot experiment*, we find acceptability judgments to be inappropriate for answering our research question.

We want a *finer-grained measure* of recovery difficulty, so we'd like to change the experiment to **self-paced reading** and look at differences in *reading time*.

Given our existing template, we take the following steps:

1. Go to the documentation and find a Controller for self-paced reading.
2. Specify the design of that controller in the **defaults** variable.
3. Change our trials in **items** from "**acceptabilityJudgment**" to that Controller.
4. Make changes to the text of the "**Messages**" items (e.g., directions).

Let's go look at the documentation!

# "DashedSentence" Controller

The "**DashedSentence**" Controller creates self-paced reading trials.

We first re-write **defaults** to specify appropriate settings for "**DashedSentence**".

```
var defaults = [
  "DashedSentence", {
    mode: "self-paced reading",
    display: "dashed"
  }
]
```

Then, we replace "**acceptabilityJudgment**" with "**DashedSentence**" in **items**.

```
var items = [
  ...
  [["gp.trans",1], "DashedSentence", {s: "While Anna trained the kitten
  [["gp.intrans",1], "DashedSentence", {s: "While Anna dressed the kitte
  [["gp.trans",2], "DashedSentence", {s: "Since Dave improved the depart
  [["gp.intrans",2], "DashedSentence", {s: "Since Dave worried the coun
  ...
]
```

# Our second experiment

Finally, after re-writing some of the "**Message**" items, we have a [new experiment!](#)

```
// Defaults and other settings //

//// Generates random number assigned to participants (Participant ID)
var randomCode = Math.random().toString(36).substr(2,9)
//// A message to show to participants at completion (useful for confirmation, raffle entry, etc.)
var completionMessage = "Thank you for your participation. Your participation code is: " +
randomCode

//// Show a progress bar at the top? (true/false)
var showProgressBar = false

//// Override default settings for controllers (parameters go inside the curly braces { })
var defaults = [
  "DashedSentence", {
    mode: "self-paced reading",
    display: "dashed"
  }
]

// Presentation Order //

var shuffleSequence = seq(
  "intro",
  "consent",
  "directions",
  startsWith("practice"),
  "setcounter",
```

# Question and Form Controllers

---

Yes/No

Forced-Choice-1

Forced-Choice-2

Free-Response

You can use the **Question** controller to ask **Yes/No** comprehension questions:

```
[["YesNo_example",1],  
 "Question",  
 {q: "Was the kitten trained?", as: ["Yes", "No"]}]
```

Was the kitten trained?

1. Yes

2. No

# Question and Form Controllers

Yes/No

Forced-Choice-1

Forced-Choice-2

Free-Response

The **as** parameter sets the **answer choices** presented to participants.

```
[["FC1_example", 1], "Question", {  
    instructions: "Choose the more natural sentence.",  
    as: ["Who did you see that ate bread?",  
        "What did you see the girl who ate?"]}]
```

1. Who did you see that ate bread?
2. What did you see the girl who ate?

*Choose the more natural sentence.*

# Question and Form Controllers

Yes/No

Forced-Choice-1

Forced-Choice-2

Free-Response

You can ask for **continuations** by modifying the **q** parameter.

```
[["FC2_example",1], "Question", {  
    q: "While Anna dressed the baby _____",  
    instructions: "Choose the more natural continuation.",  
    as: ["started to cry.", "he started to cry."}]]
```

While Anna dressed the baby \_\_\_\_\_

1. started to cry.
2. he started to cry.

*Choose the more natural continuation for this sentence.*

# Question and Form Controllers

Yes/No

Forced-Choice-1

Forced-Choice-2

Free-Response

Use **Form** controller with the "**textarea**" HTML tag to collect a **free response**:

```
[["FR_example",1],  
 "Form", {html: ["div",  
   ["em", "Fill out a continuation for the sentence fragment:"],  
   ["p", "While Anna dressed the baby _____"],  
   ["textarea"]]  
 }]
```

*Fill out a continuation for the following sentence fragment:*

While Anna dressed the baby \_\_\_\_\_

→ Click here to continue

### 3. Analysis of the results

# Where to find the data

The data collected in the experiment can be found on the experiment page, under the **results** section.

**results** ([upload a file to this directory](#) | [refresh](#))

raw\_results ([delete](#) | [rename](#) | [upload new version](#) | [edit](#))

results ([delete](#) | [rename](#) | [upload new version](#) | [edit](#))

The section comprises of two files, of which *results* is the better formatted one.

# The *results* file

`results` is a **csv** file (**comma separated values**), meaning that each line contains a set of values that are separated by a comma.

The lines that start with a pound symbol "#" are **comments** that include *metadata*.

They tell us what **variables** each comma-separated values correspond to.

# The variables

All IBEX experiments return these **7** variables:

1. **Time**
2. **Participant ID**
3. **Controller Name**
4. **Item number**
5. **Element number**
6. **Type**
7. **Group**

# The variables

Generally, we only care about **4** of these:

1. Time
2. **Participant ID**: A unique ID for each participant.
3. **Controller Name**: The controller that the values are from.
4. Item number
5. Element number
6. **Type**: The name of the trial.
7. **Group**: The item group number.

The last two variables uniquely identify each trial created in **items**:

```
[["Type", Group], "Trial Type", {s: "Sentence"}]
```

```
[["gp.trans", 1], "DashedSentence", {s: "..."}]
```

# The variables

Depending on the experiment, IBEX returns other variables specific to the design.

For self-paced reading with "**DashedSentence**", we get **5** more variables:

1. **Word number** - The index of the word in the sentence.
2. **Word** - The text of that word.
3. **Reading time** - Reading time for that word.
4. **Newline?** - 0 or 1 indicating whether there was a line break.
5. **Sentence** - The text of the sentence.

These are also outlined in the documentation for "**DashedSentence**", so you know what variables to expect beforehand.

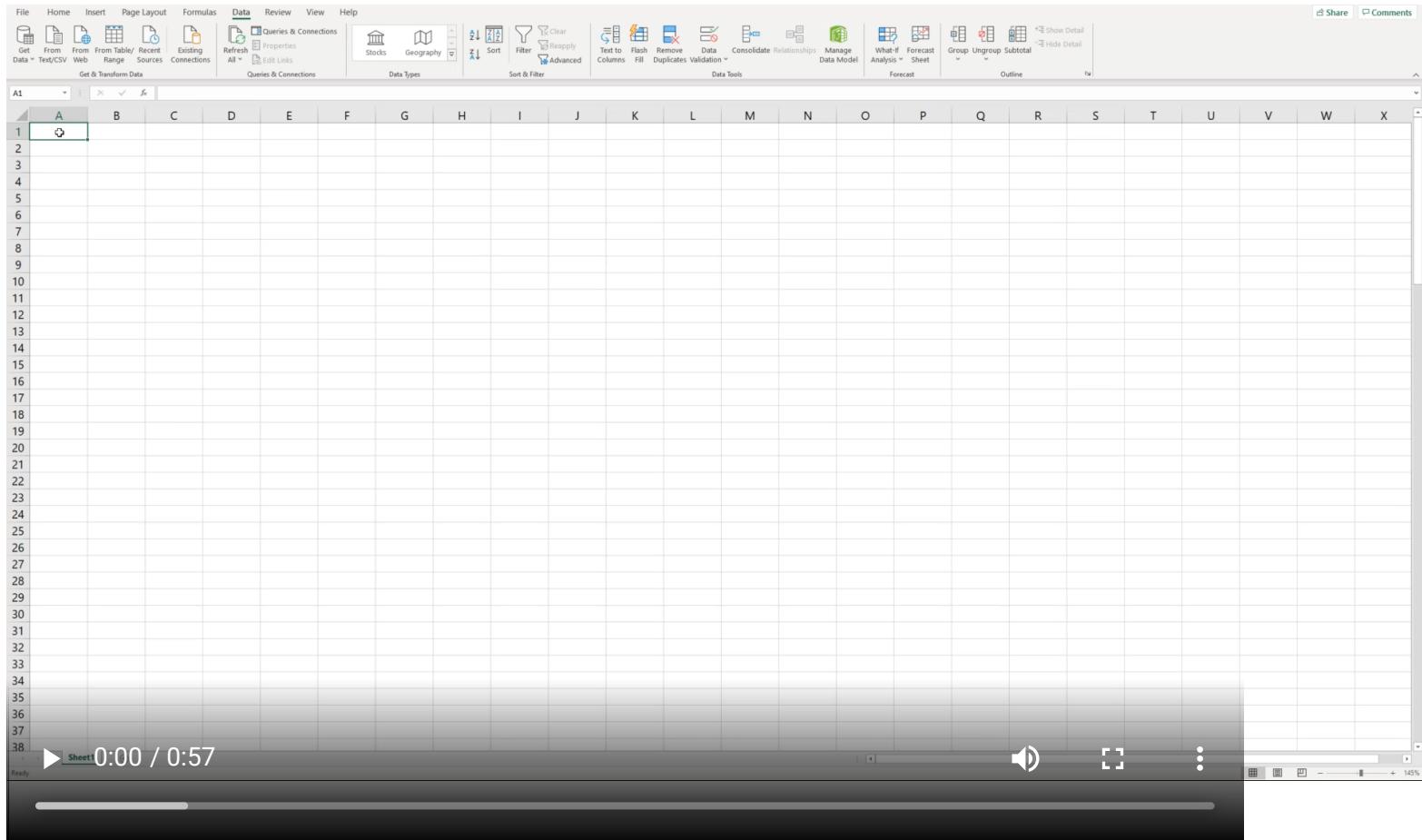
For an actual analysis of the results, we need the results to be imported somewhere as a **data frame** (where every value is a cell and each variable forms a column).

# Importing - Excel

Steps for importing the data into Excel:

1. Copy the text of the *results* file from IBEX
2. Paste into the first column of an Excel spreadsheet
3. Highlight that column and click **Data** tab -> **Filter**
4. Click on the dropdown arrow that appears at the top cell
5. Click **Text Filters** -> **Begins With...** and type in "#"
6. Go to **Home** tab -> **Find & Select**, check "Visible cells only", and click OK
7. Right click on any part of the sheet and select **Delete Row**
8. Click the first column again and go to **Data** tab -> **Text to Columns**
9. Check "Delimited", "Comma", then "General" for each prompt
10. Add an empty row at the top and manually type in the column names from the comments of the original *results* file

# Importing - Excel



# Complex results

Some Controllers return more than 1 line of results, making things complicated.

For example, each **AcceptabilityJudgment** trial adds two lines to *results*:

```
results
-
6 # The lines below this comment are in groups of 2.
7 # The formats of the lines in each of these groups are as follows:
8 #
9 # Line 1:
10 #   Col. 1: Time results were received.
11 #   Col. 2: MD5 hash of participant's IP address.
12 #   Col. 3: Controller name.
13 #   Col. 4: Item number.
14 #   Col. 5: Element number.
15 #   Col. 6: Type.
16 #   Col. 7: Group.
17 #   Col. 8: Sentence (or sentence MD5).
18 # Line 2:
19 #   Col. 1: Time results were received.
20 #   Col. 2: MD5 hash of participant's IP address.
21 #   Col. 3: Controller name.
22 #   Col. 4: Item number.
23 #   Col. 5: Element number.
24 #   Col. 6: Type.
25 #   Col. 7: Group.
26 #   Col. 8: Question (NULL if none).
27 #   Col. 9: Answer.
28 #   Col. 10: Whether or not answer was correct (NULL if N/A).
29 #   Col. 11: Time taken to answer.
30 1609648766,a4dal1f176adfea0b761238da725fd6e8,AcceptabilityJudgment,4,0,practice-1,NULL,The car drove like a dream.
31 1609648766,a4dal1f176adfea0b761238da725fd6e8,AcceptabilityJudgment,4,0,practice-1,NULL,NULL,5,NULL,1920
32 1609648766,a4dal1f176adfea0b761238da725fd6e8,AcceptabilityJudgment,6,0,practice-2,NULL,The cat ever has eaten cheese the.
33 1609648766,a4dal1f176adfea0b761238da725fd6e8,AcceptabilityJudgment,6,0,practice-2,NULL,NULL,4,NULL,513
34 1609648766,a4dal1f176adfea0b761238da725fd6e8,AcceptabilityJudgment,14,0,filler-good-01,NULL,When Harry fell%2C the audience was shocked.
35 1609648766,a4dal1f176adfea0b761238da725fd6e8,AcceptabilityJudgment,14,0,filler-good-01,NULL,NULL,4,NULL,252
36 1609648766,a4dal1f176adfea0b761238da725fd6e8,AcceptabilityJudgment,10,0,gp.trans,1,While Anna trained the kitten paid attention.
37 1609648766,a4dal1f176adfea0b761238da725fd6e8,AcceptabilityJudgment,10,0,gp.trans,1,NULL,4,NULL,272
38 1609648766,a4dal1f176adfea0b761238da725fd6e8,AcceptabilityJudgment,15,0,filler-bad-01,NULL,When Tyler sneezed the driver%2C he passed a tissue.
39 1609648766,a4dal1f176adfea0b761238da725fd6e8,AcceptabilityJudgment,15,0,filler-bad-01,NULL,NULL,4,NULL,284
40 1609648766,a4dal1f176adfea0b761238da725fd6e8,AcceptabilityJudgment,13,0,gp.intrans,2,Since Davi worried the counselor devised a plan.
41 1609648766,a4dal1f176adfea0b761238da725fd6e8,AcceptabilityJudgment,13,0,gp.intrans,2,NULL,4,NULL,306
42 1609648766,a4dal1f176adfea0b761238da725fd6e8,AcceptabilityJudgment,16,0,filler-catch-01,NULL,Please select 4 for this sentence.
43 1609648766,a4dal1f176adfea0b761238da725fd6e8,AcceptabilityJudgment,16,0,filler-catch-01,NULL,NULL,4,NULL,326
```

In cases like these, using a **processing script (R/Python)** is recommended

# Importing - R

Steps for importing into R:

1. Download the *results* file from IBEX
2. Open it with the `read_ibex()` function from this [read\\_ibex.R](#) script

---

DashedSentence      AcceptabilityJudgment

---

```
source("read_ibex.R")
results <- read_ibex("SPR_results.txt")

##      Controller name          Type Group Word number    Word Reading time
## 92  DashedSentence   gp.intrans     2           8 plan.       261
## 75  DashedSentence   gp.trans      1           5 kitten      277
## 95  DashedSentence filler-bad-01  NULL         3 sneezed      339
##                                     Sentence (or sentence MD5)
## 92      Since Dave worried the counselor devised a plan.
## 75      While Anna trained the kitten paid attention.
## 95 When Tyler sneezed the driver%2C he passed a tissue.
```

# Importing - R

Steps for importing into R:

1. Download the *results* file from IBEX
2. Open it with the `read_ibex()` function from this [read\\_ibex.R](#) script

---

DashedSentence      AcceptabilityJudgment

---

```
source("read_ibex.R")
results <- read_ibex("Acceptability_results.txt")

##           Controller name          Type Group
## 16 AcceptabilityJudgment      gp.intrans    2
##  5 AcceptabilityJudgment filler-catch-01  NULL
##  2 AcceptabilityJudgment      gp.intrans    2
##                                     Sentence (or sentence MD5) Answer
## 16 Since Dave worried the counselor devised a plan.      7
##  5             Please select 4 for this sentence.      4
##  2 Since Dave worried the counselor devised a plan.      4
```

# Break

# Exercise



# Task

The experiment script incomplete.js has some missing pieces.

```
///////////
//// Exercise ////
// There are a total of EIGHT tasks for you to complete in this script.
// The instructions for each task are written out as comments.
// Some parts of the task are completed for you and your job is to fill in the blanks _____.
// Make sure to consult the IBEX manual! -
https://github.com/addrummond/ibex/blob/master/docs/manual.md
// 
///////////



// Task #1: Send a completion message that says "Thank you for your participation!"
var _____ = "_____"

var showProgressBar = false

var defaults = [
    "AcceptabilityJudgment", {
        as: _____, // Task #2: Make the acceptability judgment scale from 1-5
        presentAsScale: true,
        instructions: "Use number keys or click boxes to answer.",
        leftComment: "(Bad)",
        rightComment: "(Good)"
    }
]
```

# Task

## Directions:

1. Download [incomplete.js](#)
2. Create a new experiment on your ibex account called "WorkshopExercise" and replace the contents of *example\_data.js* in the **data\_includes** section with *incomplete.js*.
3. Click **edit** and follow the directions in the script to fill in the blanks.
4. Complete as many of tasks as possible (**8 total**).

If you finish, save the edits and click on your experiment link at the top of the page. Check to see that your experiment looks similar to this [complete version](#).

**Make sure to use the IBEX documentation!**

<https://github.com/addrummond/ibex/blob/master/docs/manual.md>

# Solutions

**Live demo on Atom**

*This slide will be updated with solutions*

# Resources

All materials from this presentation can be found on [github](#).

More resources from the community:

- The official [IBEX documentation](#).
- The official [IBEX google group](#) to ask questions.

## UPDATE (01/01/2021)

There is a **new version of IBEX** at a different link: <https://ibex.spellout.net>



A screenshot of a GitHub post from Alex Drummond (@addrummond) dated Jan 1. The post contains a message about a new version of IBEX and includes a link to the new site.

Alex Drummond  
to ibexexperiments

Happy new year!

I'm announcing a complete rewrite of the Ibex Farm deployed at

<https://ibex.spellout.net>

The user-facing changes are conservative and incremental, but there are some significant improvements and new features. You can find an exhaustive list of these at <https://gist.github.com/addrummond/b345d3d4f23436838e52afbe880ebbd9>. The first section should tell you

# Resources

All materials from this presentation can be found on [github](#).

More resources from the community:

- The official [IBEX documentation](#).
- The official [google group](#) to ask questions.

## UPDATE (01/01/2021)

**New feature:** results can be downloaded as a spreadsheet ([full list of changes](#)).

 **workshop\_DashedSentence**

PUBLIC [make private](#)

Ibex version: 0.3.9 (imported from original Ibex Farm)  
The counter is set to **2**.  
**1 set of results.**

**Results for workshop\_DashedSentence**

This experiment has 1 set of results.

[Results as CSV spreadsheet](#)  
[Results as Excel .xlsx spreadsheet](#)  
[Results in original Ibex format \(with no column name comments\)](#)  
[Results in original Ibex format \(with per-line column name comments\)](#)  
[Raw JSON results](#)

# Resources

All materials from this presentation can be found on [github](#).

More resources from the community:

- The official [IBEX documentation](#).
- The official [google group](#) to ask questions.

## Other platforms

- PCIbex (Penn Controller for IBEX) [Link](#)
- PsychoPy [Link](#)
- \_magpie (minimal architecture for the generation of portable interactive experiments) [Link](#)

# Resources

All materials from this presentation can be found on [github](#).

More resources from the community:

- The official [IBEX documentation](#).
- The official [google group](#) to ask questions.

## **Platforms to recruit participants for your IBEX experiment**

- MTurk (Amazon Mechanical Turk) [Link](#)
- Prolific [Link](#)