


# Introduction to Lookit

[Lookit](#) [Experimenter](#) [Studies](#) [FAQ](#) [The Scientists](#) [Resources](#) [My Account](#) [Logout](#)




## Lookit

the online child lab


A project of the MIT Early Childhood Cognition Lab

[Participate in a Study](#)




### Bringing science home

Here at MIT's Early Childhood Cognition Lab, we're trying a new approach in developmental psychology: bringing the experiments to you.



### Help us understand how your child thinks

Your family can contribute to research about how children learn by doing fun activities together, right in your web browser.



### Participate whenever and wherever

Log in or create an account at the top right to get started! You can participate with your child from any computer with a webcam.

### News

June 29, 2020	We're now accepting new studies from research groups around the world!
May 4, 2020	For even more ways to participate in science with your kids at home, check out the newly launched site <a href="#">Children Helping Science</a> .
March 27, 2020	We're gearing up to open up Lookit to more research labs! Our first study for newborns, "Baby See, Baby Do?" is live, and "Baby Euclid" is back for 7-month-olds. And we're still recruiting for Flurps and Zazzes, Your Baby the Physicist, and Baby Laughter - please help us spread the word!
September 18, 2019	Data collection is complete for the Lookit studies "Baby Euclid," "Mind and Manners", "Flurps and Zazzes," and "Words and Objects" (but stay tuned for follow-ups!). We're still looking families with babies to participate in "Look and Listen," "Your Baby the Physicist," and "Baby Laughter Games."

Jenna Croteau  
Smith College  
Dec 18 2020

# How to Use This Document

This document is intended to introduce the reader to Lookit by giving a quick 20 minute overview of how Lookit works. In addition to giving an overview, this document also links to various resources that will be beneficial to researchers who chose to use Lookit. If researchers are interested in implementing a Lookit study after finishing this document, they should see the Lookit tutorial and Lookit documentation available [here](#).

## Sections

What is Lookit?	3
Security, Privacy, and Data Collection on the Internet	4
Javascript, JSON, HTML, and CSS	6
Experiment Runner	7
Web Repositories	10
Experiment Frames	11
Protocol Configuration	17
Protocol Generator	19
Study Approval Process: Peer, IRB, and Lookit Approval	20

# What is Lookit?

Infants and young children possess unique, and sometimes fleeting, strategies for making sense of the world around them. These strategies provide developmental scientists an important window into the underpinnings of human cognition; however, developmental research is often constrained by the unique needs of young children and their families. Some parents may find it difficult to schedule a lab visit and emails are often lost in the flurry of day-to-day caregiving. Young children and babies frequently have their own internal schedule that does not necessarily align with their caregiver's or the researcher's. These hurdles constrict the set of questions developmental scientists can ask and result in suboptimal data collection. Lookit is a collaborative platform designed to fix these problems.

Lookit lets developmental science researchers conduct asynchronous recorded experiments online. Families interested in participating in online experiments sign up for an account on the Lookit interface. Once a family has an account, they can view all the published experiments currently in the data collection phase. Families can also opt-in for email notifications when a new study in their child's eligibility range is published or if their child ages into the eligibility requirement for a new study.

Researchers design experiments either on the experimenter interface using pre-formatted templates for common developmental science paradigms (ie: VOE, preferential looking, etc) or by developing their own Javascript/JSON code. Researchers submit their studies to peer review and their own IRBs for review. Once a research group has received IRB and peer review approval, they can publish to the "Studies" page to start data collection.

Participants are in control of when and where they want to complete experiments. At the end of each experiment, participants also get the decision to withdraw their data. If participants choose to submit their data, the data is sent to the research group who designed the study and the Lookit admin team.

# Security, Privacy, and Data Collection on the Internet

Collecting personal and private data from children on the internet should be taken seriously.

There are a few best practices to keep in mind while using Lookit or any other online experiment software:

- 1. Do not share your username and password with another person, including other researchers in your lab.**

Please don't do this. It might seem like not a big deal if you both have access to the same data but it's not a good idea. Everyone who wants to access the data should use their own personal account that way if there is a data breach, it is easier to track where the breach occurred and whose accounts were compromised.

- 2. Enable Dual-Authentication for any user accounts with access to personal data.**

Dual-authentication is an extra layer of protection against security breaches. For example, if your password was compromised somehow then dual-authentication would prevent another person from using your password to log into your account. You can enable dual-authentication on your email and on Lookit. In fact, you are required to use dual-authentication on Lookit. Google has a good dual authenticator app called "Google Authenticator."

- 3. Password protect your computer.**

Your computer needs to be password protected with a strong password containing capital letters, lowercase letters, numbers and symbols.

- 4. Set your computer to screen lock after a period of time.**

It's easy to walk away from your computer and forget that your computer is still unlocked. You can prevent that by setting your computer to lock after a certain amount of inactivity. Ideally you'll set it to lock after 30 minutes of inactivity or less.

- 5. Use full-disk encryption on your computer.**

Full-disk encryption makes it nearly impossible for someone to access data on your locked computer by encrypting every bit. Enabling full-disk encryption means that even if someone stole your hard-drive, it would be nearly impossible to decipher the data. MacOS users can turn on FileVault. Windows users can turn on BitLocker.

- 6. Always use your university's VPN when downloading personal information off of Lookit.**

There's always a small chance that personal information can be intercepted when it's sent on the internet. Using a VPN reduces this risk. If you are sending or receiving sensitive personal data, it's best practice to turn your VPN on.

- 7. Don't send sensitive personal information about participants over email.**

Standard email messaging is not encrypted and therefore can be intercepted. Also, unless you regularly delete your emails, the sensitive personal information you send will end up being stored on your email. Private personal information about participants shouldn't be stored anywhere except on a fully encrypted computer or firewall protected server.

**8. Store data securely.**

It's best to consult with your university's technology department on how to store your experimental data. Most universities can provide you access to an encrypted server to store your data. If you collect data on Lookit, leaving the data on Lookit is also a storage option!

**9. Only collect/download/store what you need!**

If you don't need to know your participant's name, don't download it! If you don't need to know your participant's address, don't ask for it! Only ask for what you need. The best way to protect data is to not collect it.

# Javascript, JSON, HTML, and CSS

Lookit requires that experimenter's specify their study protocols in either JSON (for the protocol configuration) or in Javascript (for the protocol generator). Also some frames can be supplemented by CSS and HTML code. While knowledge of Javascript, CSS, and HTML is helpful, JSON is the only programming language that you need to understand in order to understand how Lookit works.

W3Schools has excellent tutorials on Javascript, JSON, HTML, and CSS. If you are new to coding, or have never done web development before, then you might find it beneficial to spend some time on these tutorials. Tutorials on Javascript, JSON, HTML, and CSS can all be found at [w3schools.com](http://w3schools.com).

The screenshot shows the W3Schools website, which is described as 'THE WORLD'S LARGEST WEB DEVELOPER SITE'. The navigation bar includes links for TUTORIALS, REFERENCES, EXAMPLES, and EXERCISES, along with a search icon and a LOG IN button. The main content area is divided into two columns. The left column contains a list of tutorials under the heading 'HTML and CSS', including 'Learn HTML', 'Learn CSS', 'Learn Bootstrap', 'Learn W3.CSS', 'Learn Colors', 'Learn Icons', 'Learn Graphics', 'Learn How To', and 'Learn Sass'. Below this is a section for 'JavaScript' with links to 'Learn JavaScript', 'Learn jQuery', 'Learn React', 'Learn AngularJS', 'Learn JSON', 'Learn AJAX', 'Learn HTML-DOM', 'Learn AppML', and 'Learn W3.JS'. Further down is a 'Server Side' section with links to 'Learn SQL', 'Learn PHP', 'Learn ASP', 'Learn Node.js', and 'Learn Raspberry Pi'. The bottom section is 'Programming' with a link to 'Learn Python'. The right column features a large 'HTML' heading with the subtitle 'The language for building web pages' and two buttons: 'LEARN HTML' and 'HTML REFERENCE'. Below this is a 'CSS Example' section showing a code snippet for styling a body, h1, and p. To the right of the CSS example is a 'CSS' heading with the subtitle 'The language for styling web pages' and two buttons: 'LEARN CSS' and 'CSS REFERENCE'. Above the CSS heading is an 'HTML Example' section showing a code snippet for a basic HTML document structure. A 'Try it Yourself »' button is located below the HTML example code.

**w3schools.com** THE WORLD'S LARGEST WEB DEVELOPER SITE [LOG IN](#)

TUTORIALS ▾ REFERENCES ▾ EXAMPLES ▾ EXERCISES ▾ ⓘ 🌐 🔍

**HTML and CSS**

- Learn HTML
- Learn CSS
- Learn Bootstrap
- Learn W3.CSS
- Learn Colors
- Learn Icons
- Learn Graphics
- Learn How To
- Learn Sass

**JavaScript**

- Learn JavaScript
- Learn jQuery
- Learn React
- Learn AngularJS
- Learn JSON
- Learn AJAX
- Learn HTML-DOM
- Learn AppML
- Learn W3.JS

**Server Side**

- Learn SQL
- Learn PHP
- Learn ASP
- Learn Node.js
- Learn Raspberry Pi

**Programming**

- Learn Python

# HTML

The language for building web pages

[LEARN HTML](#) [HTML REFERENCE](#)

**HTML Example:**

```
<!DOCTYPE html>
<html>
<title>HTML Tutorial</title>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

[Try it Yourself »](#)

# CSS

The language for styling web pages

[LEARN CSS](#) [CSS REFERENCE](#)

**CSS Example:**

```
body {
  background-color: lightblue;
}

h1 {
  color: white;
  text-align: center;
}

p {
  font-family: verdana;
  font-size: 20px;
}
```

# Experiment Runner

The experiment runner is software that interprets your study protocol into an interactive web app. It takes the "sequence" and "frames" from either your protocol configuration or protocol generator to make your experiment. In other words, you can think about the experiment runner as the thing that builds your study. You tell the runner how to build your study by giving it a list of materials (i.e.: "frames") and an order (i.e.: "sequence").

The experiment runner periodically undergoes updates, similar to how your computer undergoes a software update. See an excerpt below from the Lookit docs on experiment runner updates.

## Updating the experiment runner

In the future, there may be changes in the Lookit experiment runner that you want your study to use - for instance, a bug fix for an issue your participants are encountering or a new frame you want to use. (By default, your study keeps chugging along using exactly the same code, so that updates can't change how your study works without your knowledge.)

## Checking what's changed

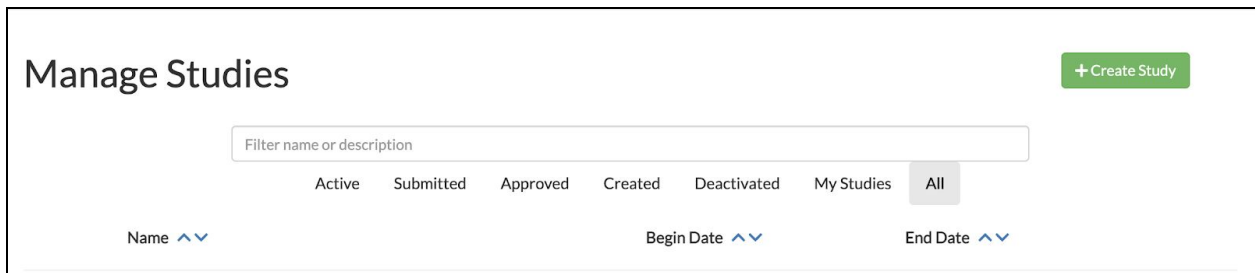
The most straightforward way to view changes to the Lookit code is to review the list of [releases](#). If you're planning to update to the latest version, you should read through the release notes for each version between the one you're using and the new one.

The releases are numbered v<MAJOR>.<MINOR>.<PATCH> - e.g. v1.5.2. We adhere to semantic versioning, so the MAJOR version changes when there are backwards-incompatible changes - e.g., you need to change the name of a frame you're using for it to keep working. The MINOR version changes when features are added but are backwards-compatible. The PATCH version changes when there are backwards-compatible bug fixes.

These updates are optional but recommended. Since the experiment runner is dynamic, you must build a copy of the experiment runner software to run your study protocol on. This ensures that your protocol will be unaffected by an incompatible software update.

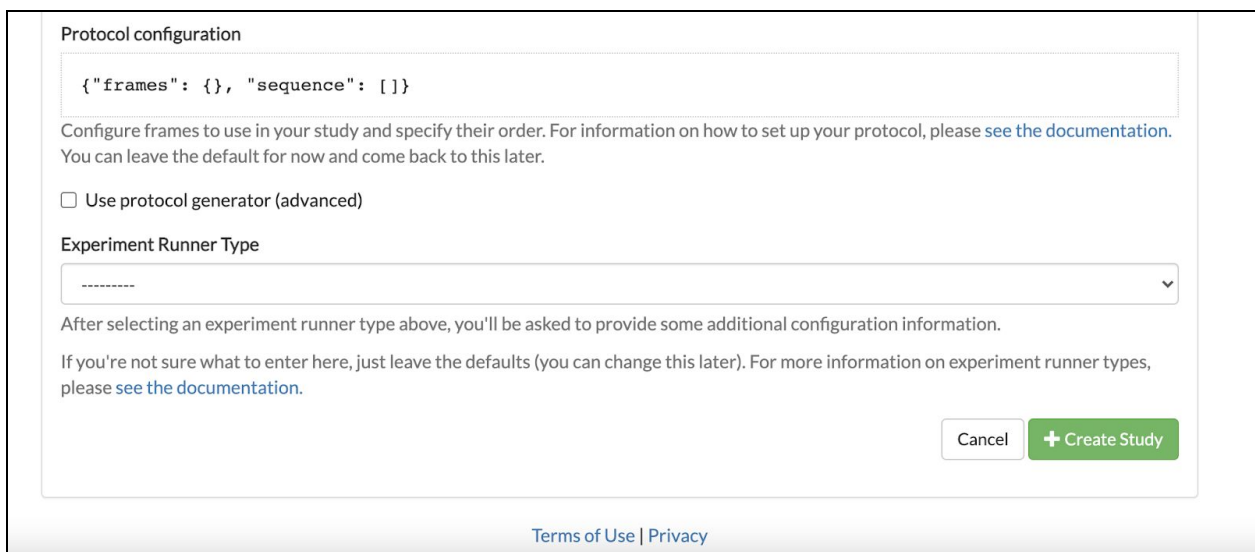
## Building Your Experiment Runner for the First Time

First you need to create a new experiment! Go to "Manage Studies" on your Lookit account and click **+ Create Study**:



The 'Manage Studies' interface features a header with the title 'Manage Studies' and a green '+ Create Study' button. Below the header is a search bar labeled 'Filter name or description'. A horizontal menu contains tabs for 'Active', 'Submitted', 'Approved', 'Created', 'Deactivated', 'My Studies', and 'All', with 'All' currently selected. At the bottom, there are three columns with headers: 'Name' with a sort icon, 'Begin Date' with a sort icon, and 'End Date' with a sort icon.

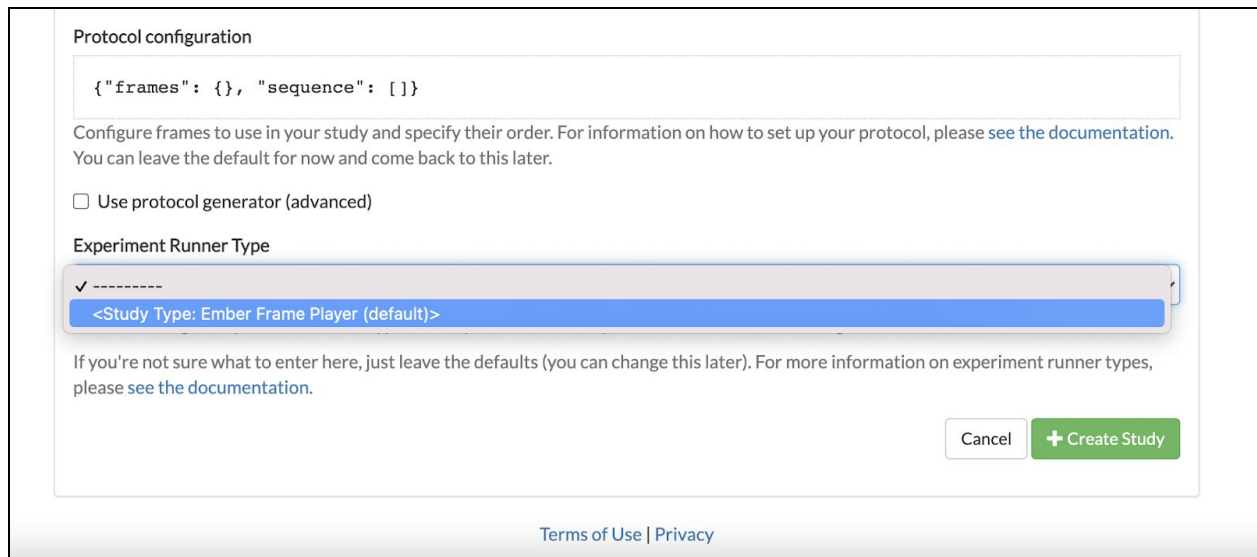
After you click **+ Create Study** you will need to fill out some information about your study in order to create it. This information can be changed later so if you don't have a good response for any of the prompted fields then you can enter some placeholder like, "FILL ME IN LATER." If you leave a field empty Lookit won't let you create a new study. At the bottom of the form you will see the following:



The 'Protocol configuration' form includes a text area for JSON configuration with the default value `{"frames": {}, "sequence": []}`. Below this is explanatory text and a link to documentation. A checkbox for 'Use protocol generator (advanced)' is present and unchecked. The 'Experiment Runner Type' is shown as a dropdown menu with a placeholder '-----'. Further instructions and a link to documentation are provided. At the bottom right are 'Cancel' and '+ Create Study' buttons. A footer contains links for 'Terms of Use' and 'Privacy'.

Click the dropdown "Experiment Runner Type" and select "Ember Frame Player":





Protocol configuration

`{"frames": {}, "sequence": []}`

Configure frames to use in your study and specify their order. For information on how to set up your protocol, please [see the documentation](#). You can leave the default for now and come back to this later.

☐ Use protocol generator (advanced)

Experiment Runner Type

✓ -----

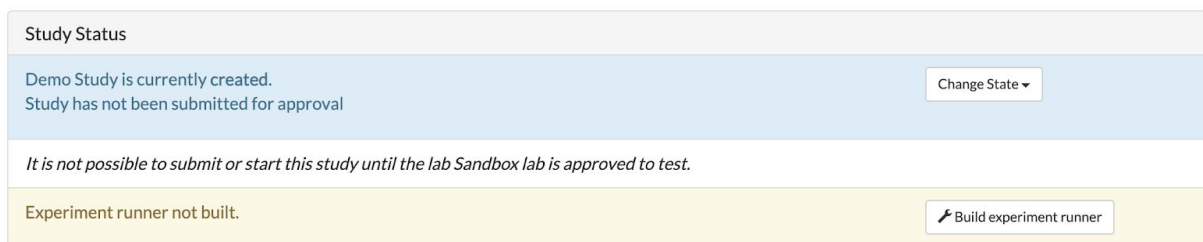
<Study Type: Ember Frame Player (default)>

If you're not sure what to enter here, just leave the defaults (you can change this later). For more information on experiment runner types, please [see the documentation](#).

Cancel [+ Create Study](#)

[Terms of Use](#) | [Privacy](#)

Once you've filled out all the other fields, click "+ Create Study." This will take you to your study page. Here you will see a highlighted orange box. Inside the highlighted orange box you'll want to click on "Build experimenter runner":



Study Status

Demo Study is currently created.  
Study has not been submitted for approval

Change State ▾

*It is not possible to submit or start this study until the lab Sandbox lab is approved to test.*

Experiment runner not built.

[Build experiment runner](#)

You will receive an email notification once the experiment runner is built.

## Updating the Experiment Runner

It's good to keep your experiment runner up to date. Software updates are released for good reason; it could be a minor bug patch, like fixing a glitch where the display is shifted in fullscreen mode, or it could be a major update, like adding a pause button to studies. The Lookit docs does an excellent job of explaining how to update the experiment runner. Click [here](#) for a link to instructions on updating the experiment runner.

# Web Repositories

At some point you will need to incorporate your stimuli into your program. Your stimuli might be audio, image, or video files. Lookit is designed to retrieve stimuli files from secure static web repositories, or web addresses that start with “https://”. You can host your stimuli on a third-party data storage website like GitHub or you can use the secure static web hosts made available by the college. While there is nothing wrong with using a third-party data storage site like GitHub, using your college’s is preferred because you will never have to pay for the storage.

Once you’ve gotten a secure web address setup, you will need to structure it so that the experiment runner can parse your stimuli files. Your base directory should branch into five folders named: img, mp4, webm, mp3, and oog. These folders refer to the five file types compatible with the Experiment Runner. Audio files should be provided with both mp3 and oog file extensions and stored in the mp3 and oog folders. Video files should be provided with mp4 and webm file extensions and stored in the mp4 and webm folders. Finally, image files can either have a png or jpg file extension and should be stored in the img folder.

There isn’t one correct way to upload your files and folders to your web repository. If you use a repository hosted at your college then you will need to download some SSH/SFTP file transfer software. When setting up your web host you can ask your college’s technology department what file transfer software they would prefer you use.

You can check whether you’ve set up your directory properly and put stimuli in the correct folders by typing in the full address of your stimuli into a web browser.

For example, if your web address was: <https://www.science.smith.edu/~jmargarites>

You could check whether your OverShoulder.jpg image was uploaded correctly by typing this into a web browser: <https://www.science.smith.edu/~jmargarites/img/OverShoulder.jpg>

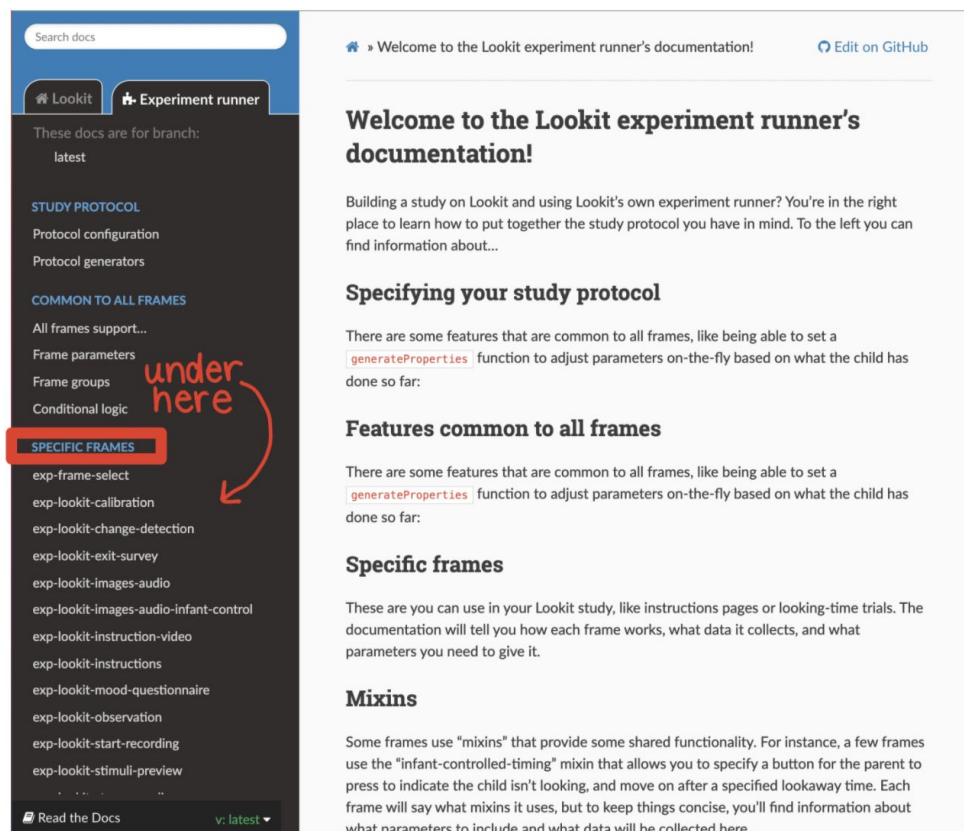
# Experiment Frames

Experiment frames are designed to help you create your experiment's protocol with minimal coding. If the experiment runner is the builder, then the experiment frames are the materials. Each frame represents a different building block that you can use to create your study.

Frames are written in JSON and contain in-built behaviors like turning on/off the participant's webcam or recording a button click. Each frame also has parameters specific to the frame's functionality. For example, some frames have a `baseDir` parameter which tells the Experiment Runner where to look for your media files (ie: images, audio, videos).

Most of the common developmental science paradigms have a corresponding frame or set of frames that can be implemented. For example, if you wanted to conduct a preferential looking study you could use the `exp-lookit-calibration` and `exp-lookit-images-audio` frames. The `exp-lookit-calibration` frame would allow you to record the child looking at different parts of the screen while the `exp-lookit-images-audio` frame would allow you to record the child looking at your stimuli.

All the experiment frames can be found in the Lookit documentation under "Specific Frames":



## Frame Parameters

Frame parameters are optional behaviors or attributes a frame can have. Frame parameters can be found when you click on a specific frame and scroll down to the “Parameters” section. Each frame parameter has a corresponding set of values it can be set to. For example, most frames have a `doRecording` parameter. The `doRecording` parameter can be set to the values `true` or `false`. If you want a frame to record the participant then you would set `doRecording` to `true`, otherwise you would set `doRecording` to `false`. Values can be text, numbers, true/false, media files, and objects (like arrays, templates, and lists). If you want to include a parameter you should add it to your experiment frame with whatever value will implement your desired behavior or attribute.

Some frame parameters have default value settings. This means the parameter is automatically set to some value regardless of whether you include the parameter in your protocol configuration. If you want to disable the default setting for a particular parameter, you must include the parameter in your protocol configuration with a different value. Let’s look at the `exp-lookit-start-recording` parameters. You can find the `exp-lookit-start-recording` parameters in the Lookit docs by clicking on the `exp-lookit-start-recording` frame and scrolling down to the “Parameters” section:

### Parameters

**displayFullscreen** [Boolean | `true`]

Whether to display this frame in full-screen mode

**backgroundColor** [String | `'white'`]

Color of background. See [CSS specs](#) for acceptable syntax: can use color names (`'blue'`, `'red'`, `'green'`, etc.), or rgb hex values (e.g. `'#800080'` - include the `#`)

**video** [String or Array]

Video to play (looping) while waiting. You can optionally supply either a video or image, not both.

This can be either an array of `{src: 'url', type: 'MIMETYPE'}` objects or just a string like `attentiongrabber` to rely on the `baseDir` and `videoTypes` to generate full paths.

**image** [String]

Image to display while waiting. You can optionally supply either a video or image, not both.

This can be either a full URL or just the filename (e.g. `star.png`) to use the full path based on `baseDir` (e.g. `baseDir/img/star.png`).

**imageAnimation** [String | `'spin'`]

Which animation to use for the image. Options are `'bounce'`, `'spin'`, or `''` (empty to not animate).

The frame parameters are highlighted in yellow and the value types are highlighted in blue. The default values for the parameters are in white boxes with red text. Underneath each of the parameters is a short description of what the parameter is intended to implement.

The value types let you know what kind of values would be acceptable. For example, look at the `displayFullscreen` parameter. The `displayFullscreen` parameter has a Boolean value type which means it can be set to the values `true` or `false`. Now let's look at the `video` parameter. The `video` parameter has a String or Array value type. That means the `video` parameter can take either value types. By reading the description underneath the `video` parameter we know that:

- If we assign a String value to the `video` parameter then the Experiment Runner expects that String to either be the video name or the video URL.
- If we assign an Array value to the `video` parameter then the Experiment Runner expect that Array to be in the format:

```
{src: "INSERT_FULL_URL",  
  type: ".mp4"  
}
```

where `src` refers to the video URL and `type` refers to the file extension.

## Data Collected and Events Recorded

If you click on a specific frame, you will find the “Data Collected” and “Events Recorded” sections after the “Parameters” section. These sections are extremely important because we want to make sure we collect the right data for our experiment. Data collected refers to the data that is collected on this particular frame (ex: the name of the image selected, the name of the audio file played). Events recorded refers to the different page interactions that occurred on the frame as well as their time stamp (ex: audio started playing, participant clicked “X”).

All frames collect the following data:

- `generatedProperties`: Any properties generated via a custom `generateProperties` function provided to this frame (e.g., a score you computed to decide on feedback). In general will be null.
- `frameDuration`: Duration between frame being inserted and call to next frame
- `frameType`: Type of frame
- `eventTimings`: Ordered list of events captured during this frame (oldest to newest).

and all frames record the following events:

- `nextFrame`: When moving to next frame
- `previousFrame`: When moving to previous frame

Some frames might not record any additional data or events other than those listed above.

Let's look at the data collected by `exp-lookit-video` frame. You can find the data collected by `exp-lookit-video` frame in the Lookit docs by clicking on the `exp-lookit-video` frame and scrolling down to the “Data collected” section:

## Data collected

The fields added specifically for this frame type are:

**videoShown [String]**

Source of video shown during this trial. Just stores first URL if multiple formats are offered.

**audioPlayed [String]**

Source of audio played during this trial. Just stores first URL if multiple formats are offered.

**hasBeenPaused [Boolean]**

Whether the video was paused at any point during the trial

The fields, or names of the data collected, are highlighted in purple. The type of value recorded for the field is highlighted in yellow and the description of the data collected is highlighted in blue.

If you download a .csv of the data you collect for your experiment, you will see that the frame specific fields appear under “key” and that the values recorded for that key appear next to the field under “value”:

response_uid	child_hashed_id	frame_id	event_number	key	value
		20-test		videoShown	<a href="https://www.science.smith.edu/~jmargarites/webm/attentiongrabber.webm">https://www.science.smith.edu/~jmargarites/webm/attentiongrabber.webm</a>
		20-test		audioPlayed	<a href="https://www.science.smith.edu/~jmargarites/ogg/chimes.ogg">https://www.science.smith.edu/~jmargarites/ogg/chimes.ogg</a>
		20-test	0	eventType	exp-lookit-video:videoStarted
		20-test	0	timestamp	2020-11-13T19:30:19.744Z
		20-test	0	streamTime	
		20-test	1	eventType	exp-lookit-video:audioStarted
		20-test	1	timestamp	2020-11-13T19:30:19.746Z
		20-test	1	streamTime	
		20-test	2	eventType	exp-lookit-video:audioStopped
		20-test	2	timestamp	2020-11-13T19:30:21.634Z
		20-test	2	streamTime	
		20-test	3	eventType	exp-lookit-video:trialCompleted
		20-test	3	timestamp	2020-11-13T19:30:24.744Z
		20-test	3	streamTime	
		20-test	4	eventType	exp-lookit-video:nextFrame
		20-test	4	timestamp	2020-11-13T19:30:24.745Z
		20-test	4	streamTime	
		20-test		frameDuration	

5.356

Now let's scroll down and look at the “Events recorded” section for the `exp-lookit-video` frame:

## Events recorded

The events recorded specifically by this frame are:

<b>videoStarted:</b>	When video begins playing (recorded each time video starts if played through more than once)
<b>videoStopped:</b>	When video completes playback (recorded each time if played more than once)
<b>audioStarted:</b>	When audio begins playing (recorded each time video starts if played through more than once)
<b>audioStopped:</b>	When audio completes playback (recorded each time if played more than once)
<b>trialCompleted:</b>	When trial is complete and begins cleanup (may still then wait for video upload)
<b>pauseTrial:</b>	When trial is paused
<b>unpauseTrial:</b>	When trial is unpaused (actually proceeding to beginning or next frame, after unpauseAudio)
<b>nextButtonEnabled:</b>	When all requirements for this frame are completed and next button is enabled (only recorded if autoProceed is false)

The names of the events recorded are highlighted in orange and the description of the events recorded is highlighted in blue.

If you download a .csv of the data you collect for your experiment, you will see names of the event appear under the “value” column when “key” has the value of “eventType”. Underneath the name of the event recorded is the timestamp for that particular event. The event recordings are a little more complicated than the data recorded but see an example below where the event name is highlighted in orange and the timestamps are highlighted in blue:

response_uuid	child_hashed_id	frame_id	event_number	key	value
		20-test		videoShown	https://www.science.smith.edu/~jmargarites/webm/attentiongrabber.webm
		20-test		audioPlayed	https://www.science.smith.edu/~jmargarites/ogg/chimes.ogg
		20-test	0	eventType	exp-lookit-video:videoStarted
		20-test	0	timestamp	2020-11-13T19:30:19.744Z
		20-test	0	streamTime	
		20-test	1	eventType	exp-lookit-video:audioStarted
		20-test	1	timestamp	2020-11-13T19:30:19.746Z
		20-test	1	streamTime	
		20-test	2	eventType	exp-lookit-video:audioStopped
		20-test	2	timestamp	2020-11-13T19:30:21.634Z
		20-test	2	streamTime	
		20-test	3	eventType	exp-lookit-video:trialCompleted
		20-test	3	timestamp	2020-11-13T19:30:24.744Z
		20-test	3	streamTime	
		20-test	4	eventType	exp-lookit-video:nextFrame
		20-test	4	timestamp	2020-11-13T19:30:24.745Z
		20-test	4	streamTime	
		20-test		frameDuration	

5.356



## Deprecated Frames

Experiment frames periodically undergo updates and sometimes frames are deprecated during an update. When a frame is deprecated you can still use it in your experiment protocol. This is not recommended because typically there is a good reason for retiring a frame; however if you've already written a functioning protocol with the deprecated frame, you might not want to write a new one.

When you use a deprecated frame in your protocol, it is important that your experiment runner version supports the deprecated frame. This means doing nothing if you've already built your runner with the deprecated frame. More specifically, this means do not update your experimenter runner or rebuild it with a newer version. If you want to use a deprecated frame but haven't built an experiment runner, don't do that--frames are deprecated for a reason and you don't want to find out the reason the hard way!

All deprecated frames can be found in the docs by scrolling to the bottom of the docs page listings and looking under "Deprecated Frames"

The screenshot shows the Lookit experiment runner's documentation page. On the left, a dark sidebar contains a list of frames. The 'DEPRECATED FRAMES' section is highlighted with a red box and includes the following frames: `exp-lookit-composite-video-trial`, `exp-lookit-dialogue-page`, `exp-lookit-geometry-alternation`, `exp-lookit-geometry-alternation-open`, `exp-lookit-preferential-looking`, and `exp-lookit-story-page`. The main content area is titled 'Welcome to the Lookit experiment runner's documentation!' and includes a 'Welcome' message, a section for 'Specifying your study protocol', and sections for 'Features common to all frames', 'Specific frames', and 'Mixins'. The 'Specific frames' section mentions that the documentation will tell you how each frame works, what data it collects, and what parameters you need to give it. The 'Mixins' section mentions that some frames use 'mixins' that provide some shared functionality.



# Protocol Configuration

The protocol configuration is where you specify to the experiment runner what your "sequence" and "frames" are. The protocol configuration is written in JSON. Using the protocol configuration is pretty straightforward. You enter a list of frames under "frames" and then put those frames into a particular sequence under "sequence". That means all protocol configurations look something like this:

```
1 {  
2   "frames": {}  
3   ,  
4   "sequence": []  
5 }
```

Now let's fill in the "frames" and "sequence". Pretend you want to create a study where you are interested in how long babies will look at a cartoon picture of a baby. With that goal in mind, we need to look through the frame list in the Lookit docs and find a frame that can display an image and record the participant's webcam. The `exp-lookit-images-audio` frame can do both those things so let's add that one to our "frames" and call it "test-trial":

```
1 {  
2   "frames": {  
3     "test-trial": {  
4       "kind": "exp-lookit-images-audio",  
5       "images": [  
6         {  
7           "id": "Baby",  
8           "src": "OverShoulder",  
9           "position": "fill"  
10        }  
11      ],  
12      "baseDir": "https://www.science.smith.edu/~jmargarites"  
13    }  
14  }  
15  ,  
16  "sequence": []  
17 }
```

Since all studies at Lookit need to start with a configuration frame and consent frame and end with an exit survey frame, we'll need to include the `exp-lookit-config` frame, the `exp-lookit-video-consent` frame, and `exp-lookit-video-consent` frame in our protocol configuration as well. Let's add those too:

```

1 {
2   "frames": {
3     "test-trial": {
4       "kind": "exp-lookit-images-audio",
5       "images": [
6         {
7           "id": "Baby",
8           "src": "OverShoulder",
9           "position": "fill"
10        }
11      ],
12      "baseDir": "https://www.science.smith.edu/~jmargarites"
13    },
14    "consent": {
15      "kind": "exp-lookit-video-consent",
16      "template": "consent_004"
17    },
18    "config": {
19      "kind": "exp-video-config"
20    },
21    "exit-survey": {
22      "kind": "exp-lookit-exit-survey"
23    }
24  },
25  ,
26  "sequence": []
27 }

```

Now that we added the frames we need, we need to tell the experimenter runner what sequence it should display them in:

```

1 {
2   "frames": {
3     "test-trial": {
4       "kind": "exp-lookit-images-audio",
5       "images": [
6         {
7           "id": "Baby",
8           "src": "OverShoulder",
9           "position": "fill"
10        }
11      ],
12      "baseDir": "https://www.science.smith.edu/~jmargarites"
13    },
14    "consent": {
15      "kind": "exp-lookit-video-consent",
16      "template": "consent_004"
17    },
18    "config": {
19      "kind": "exp-video-config"
20    },
21    "exit-survey": {
22      "kind": "exp-lookit-exit-survey"
23    }
24  },
25  ,
26  "sequence": ["config", "consent", "test-trial", "exit-survey"]
27 }

```

And that's it! The hardest part about using the protocol configuration is figuring out what frames to use and which parameters to include for each frame. For more details on how frames work see the **Experiment Frames** section.

# Protocol Generator

The protocol generator provides another way to specify your study protocol; however, the protocol generator requires the experimenter to code in Javascript. More precisely, the protocol generator is defined by the researcher such that it returns a list of frames and a sequence in the same format as the protocol configuration. Here's the template protocol generator function provided by Lookit:

```
1 function generateProtocol(child, pastSessions) {
2   /*
3    * Generate the protocol for this study.
4    *
5    * @param {Object} child
6    *   The child currently participating in this study. Includes fields:
7    *   givenName (string)
8    *   birthday (Date)
9    *   gender (string, 'm' / 'f' / 'o')
10   *   ageAtBirth (string, e.g. '25 weeks'. One of '40 or more weeks',
11   *   '39 weeks' through '24 weeks', 'Under 24 weeks', or
12   *   'Not sure or prefer not to answer')
13   *   additionalInformation (string)
14   *   languageList (string) space-separated list of languages child is
15   *   exposed to (2-letter codes)
16   *   conditionList (string) space-separated list of conditions/characteristics
17   *   of child from registration form, as used in criteria expression
18   *   - e.g. "autism_spectrum_disorder deaf multiple_birth"
19   *
20   *   Use child.get to access these fields: e.g., child.get('givenName') returns
21   *   the child's given name.
22   *
23   * @param {Array<Object>} pastSessions
24   *   List of past sessions for this child and this study, in reverse time order:
25   *   pastSessions[0] is THIS session, pastSessions[1] the previous session,
26   *   back to pastSessions[pastSessions.length - 1] which has the very first
27   *   session.
28   *
29   *   Each session has the following fields, corresponding to values available
30   *   in Lookit:
31   *
32   *   createdOn (Date)
33   *   conditions
34   *   expData
35   *   sequence
36   *   completed
37   *   globalEventTimings
38   *   completedConsentFrame (note - this list will include even "responses")
39   *   where the user did not complete the consent form!
40   *   demographicSnapshot
41   *   isPreview
42   *
43   * @return {Object} Protocol specification for Lookit study; object with 'frames'
44   *   and 'sequence' keys.
45   */
46   var protocol = {
47     frames: {},
48     sequence: []
49   };
50   return protocol;
51 }
```

Working in the generator provides the experimenter with more freedom to randomize and counterbalance stimuli. The protocol generator can access information about the child such as the child's name, age, and gender as well as information about past sessions. This information can be used inside the protocol generator function. It should also be noted that anything that runs in the protocol configuration can also run in the protocol generator.

# Study Approval Process: Peer, IRB, and Lookit Approval

The study approval process consists of 5 steps:

\* These steps only need to be completed the first time a P.I. posts a study.

## 1. Institutional Agreement\*

The Institutional Agreement is an agreement between the P.I., host institution, and MIT that absolves MIT of any liability for your experiment. It must be signed by you (or the P.I. of the study) and an authorized representative on behalf of Smith College. In the past, the Associate Provost has served as an authorized representative. Once the Institutional Agreement is signed it should be sent to [lookit@mit.edu](mailto:lookit@mit.edu).

## 2. Terms of Use\*

Researchers should review the Lookit Terms of Use (see them [here](#)). After reviewing the Terms of Use, interested researchers need to complete a short quiz to demonstrate their comprehension. The quiz is available [here](#).

## 3. Self Review

Once you've finished coded your study it's time to do some self review! The self-review checklist can be found [here](#). You should go through each point on the checklist, paying particular attention to the "Purpose" and "General things to think about" sections.

## 4. Peer Review

Once you've finished self review it's time to submit for peer review. Researchers must join the Lookit Slack channel to access peer review if they haven't done so already. Under the "Researchers" section, you should submit your preview link and the standardized peer review Google form to receive feedback. The standardized peer review form can be found under the "Researchers" section of the Lookit Slack channel. Once you've submitted your study for peer review, you are expected to review at least 2-3 other researcher's studies.

Peer review is finished once you receive feedback from 1-3 researchers outside your lab group. Typically peer review takes ~1 week. Sometimes receiving peer review takes longer than anticipated. Once you've received feedback from at least one researcher outside your lab group, you should try to make as many of the suggested edits as

possible. When you submit for Lookit approval you will need to detail what edits you made based on peer review.

## 5. IRB

Just like any other experiment, a Lookit experiment needs to receive IRB approval before data collection can begin. The IRB will want to see your consent frame (and assent frame if applicable) as well as a few sample frames from your experiment design. You should get as much of the coding as possible done before submitting to the IRB.

In addition to consent/assent and a few sample frames, you should also provide the IRB with the security protocols your lab will be using to ensure participant data is kept confidential. See **Security, Privacy, and Data Collection on the Internet** for best practices you can implement in your security protocol.

## 6. Lookit Review

The final step is Lookit review. You must be approved by your IRB and have gone through peer review to submit for study approval. If you have completed both of those steps, you are ready to submit for approval. First go to the blue highlighted section called “Study Status”. Click “Change State” and select “Submit (submit study for Lookit admin review)” like below:

Lookit Experimenters Manage Studies Manage Labs View Participants Help Jenna Croteau

Manage Studies / Demo Study

### Demo Study

**short description**

**Purpose:** purpose

**Duration:** duration **Exit URL:** <https://lookit.mit.edu/studies/history/>

**Participant eligibility:** eligibility **Compensation:** compensation

**Minimum age cutoff:** 0 years 0 months 0 days **Maximum age cutoff:** 0 years 0 months 0 days

**UUID:** c390a0a8-b228-432a-b8d0-684a2f93af06

**Preview sharing:** Other researchers cannot access your study preview.

**Discoverability:** Private. Your study is not currently active, and is not public. When it is active, participants will be able to access it at your study link, but it will not be listed in [the study listing page](#).

**Study link:** <https://lookit.mit.edu/studies/c390a0a8-b228-432a-b8d0-684a2f93af06>

**Study Status**

Demo Study is currently created.  
Study has not been submitted for approval

**Experiment runner built.**

**Actions:**

- Preview Study
- Edit Study
- View Study Responses
- Message Participants
- Review Consent
- Clone Study

**Change State**

- Submit (submit study for Lookit admin review)
- Delete (delete study and any data)
- Build experiment runner

When you select the “Submit” option you will be prompted to detail all the changes you made based on peer review. After you finish detailing the changes you made, you will submit.

After ~1 week, Lookit admin will either reject or approve your study. Typically all studies get rejected on their first submission. If your study is rejected, the admin who reviewed your study will request some changes. These changes should be implemented for your study. Once you make those changes, you can resubmit your study for Lookit approval.