Downloading a lot of resources

Pavlovia can take a while to load your resources, which can be troublesome if you need to load a lot of them. Here's a workaround:

1. Define your own image loading function and save this JavaScript file beside your experiment JavaScript file

```
var img;
function load_image(src) {
  img = document.createElement("img");
  img.src = 'resources/'+src;
  return img
}
```

Important side note: similar functions for different types of resources with the critical change being the string inside document.createElement. Audio files as an example:

```
var aud;
function load_sound(src) {
  aud = document.createElement("audio");
  aud.src = 'resources/'+src;
  return aud
}
```

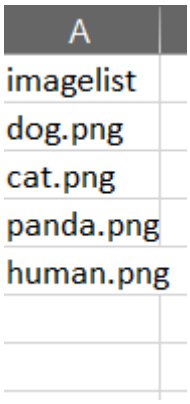
2. On index.html, navigate to the line that calls your experiment JavaScript file

```
<script type='module' src='./experimentname.js'></script>
```

Above this line, insert another line to call the JavaScript file that contains the image loading function

```
<script src = "./functions.js"></script>
```

3. If your resources list is already defined by codes, go straight ahead to step 4. This workaround is comparible with condition lists imported from Excel sheets as well, but it's advised against. What works in our favor is that coding up condition lists is super easy! Here is a simple example of specifying a to-be-randomised condition list with code:



A
imagelist
dog.png
cat.png
panda.png
human.png

Python:

```
imagelist = ['dog.png','cat.png','panda.png','human.png']
```

JavaScript:

```
var imagelist= ['dog.png','cat.png','panda.png','human.png'];
```

4. Use the <load_image> function to load images from <imagelist> with a simple for loop

Python:

```
# make list
imagelist= ['dog.png','cat.png','panda.png','human.png']

# prepare output list
outputimage = []

# loop over imagelist to apply function to each element
for i in range(0,len(imagelist)):
  outputimage.append(imagelist[i])
```

JavaScript:

```
// make list
var imagelist= ['dog.png','cat.png','panda.png','human.png'];

// prepare output list
var outputimage = [];

// loop over imagelist to apply function to each element
for (var i = 0; i < imagelist.length; i++) {
  outputimage [i] = load_image(imagelist[i])
}
```

5. Set the <image> attribute of the image component

Python:

```
# shuffling
shuffle(outputimage)

# indexing
this_image = outputimage[TrialCounter]

# set image
Image.setImage(this_image)
```

JavaScript:

```
// shuffling
outputimage = util.shuffle(outputimage); // util.shuffle comes from the psychoJS
 library

// indexing
var this_image = outputimage[TrialCounter];

// set image
Image.setImage(this_image);
```

Notes:

- The variable `TrialCounter` starts at 0 and updates every trial by adding 1 to itself, such that <this_image> would be the first element in the list <outputimage> in the first trial, the second element in list in the second trial, and so on. Whether you want to index your condition list with a trial

counter, and how to set up a trial counter is beyond the scope of this document.

- `Image` is a PsychoPy ImageStim component
- Usuallly, shuffling would be done once per block or once per experiment; If you intend to index and set the images **per trial,** they must be done under **Begin Routine** in your trial routine. Still, the exact positions to put these lines are specific to the design and beyond the scope of this document.

6. Stop Pavlovia from automatically loading your images

On GitLab, navigate to this part of the code:

```
psychoJS.start({
  expName: expName,
  expInfo: expInfo,
  resources: [] // it might not be an empty list or it might be absent altogethe
r, but it doesn't matter here
  });
```

When <resources> is not defined or left as an empty list, all files in the html/resources folder would be automatically downloaded. As we have already bypassed that process, we don't want that. To stop automatic downloading, simply make it selectively download a dummy file (but make sure the dummy file is actually in there!)

```
psychoJS.start({
  expName: expName,
  expInfo: expInfo,
  resources: [{name: 'dummy.xlsx', path: 'resources/dummy.xlsx'}]
  });
```

7. Testing it out

During the debuggint phase, it might be worth it to know exactly when each image file is being loaded and monitor if that causes unwanted lags to your experiment. To do that, we can add a line to log on the console whenever an image is loaded:

```
var img;
function load_image(src) {
  img = document.createElement("img");
  img.src = 'resources/'+src;
  return img
  console.log('loaded',scr);
}
```