

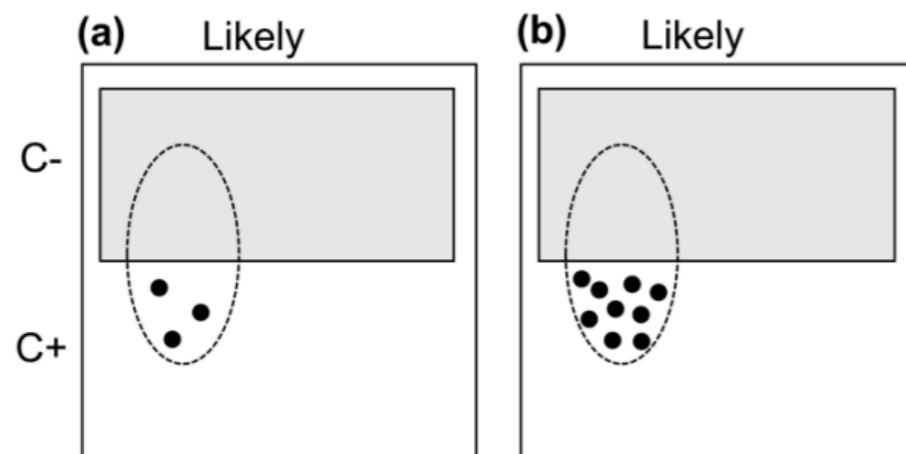
DAY 1: EXPERIMENT

Not just coding it up, but all workflow stuff up to running it

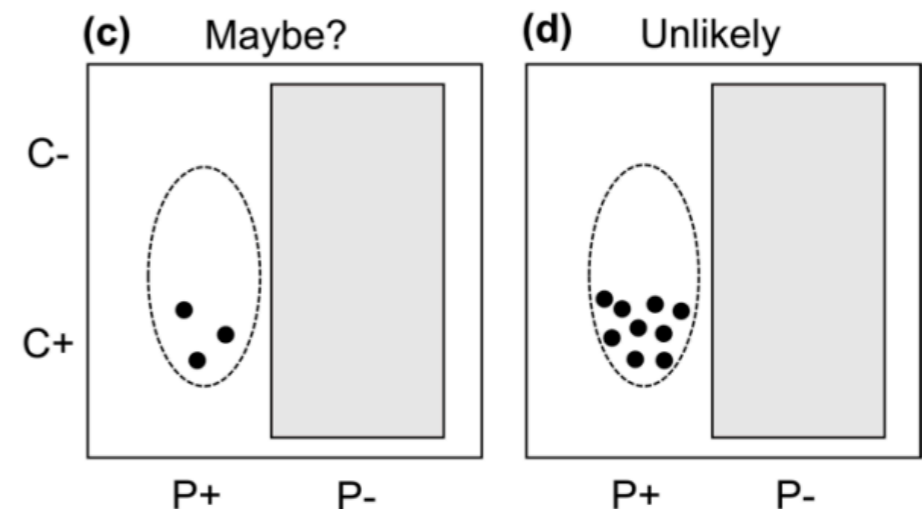
1. Background: replicability and proper procedure
2. Workflow and organisation
3. Experiment design
4. **Coding experiment**
5. Ethics and pre-registration
6. Hosting experiment on a server
7. Downloading data

OUR TASK: DESIGN AN EXPERIMENT TO TEST THIS HYPOTHESIS

Prediction of category sampling with increasing N



Prediction of property sampling with increasing N



What is the probability of C-P+?

- Conditions / manipulation?
- Task?
- Instructions?

EXPERIMENTAL DESIGN

Cover story: You are in charge of a robot probe exploring the planet Sodor, which is covered by spherical rocks. Your job is to determine which rocks contain a valuable substance called **plaxium**.

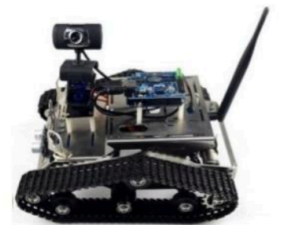
Category sampling

Only small rocks sampled because that is the only size that will fit into the robot's collecting claw.



Property sampling

Only rocks with plaxium sampled because that the robot selects those that set off its plaxium detector.



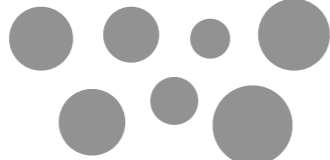
2

Sample 2 rocks (both small)



Test

Are these plaxium?



6

Sample 4 more rocks (all small)



Test

Are these plaxium?

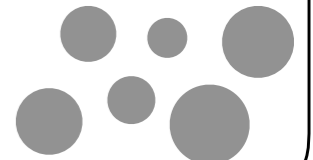


12

Sample 6 more rocks (all small)

Test

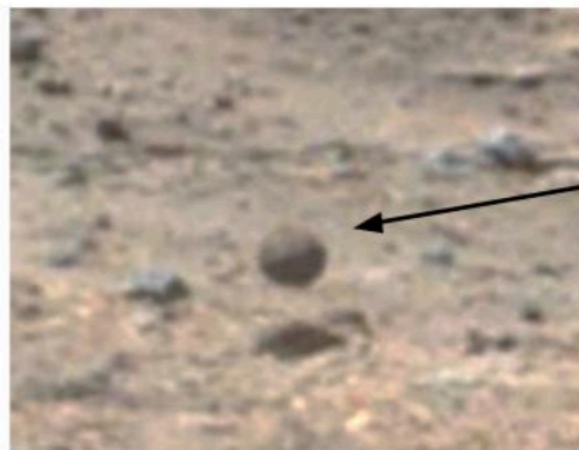
Are these plaxium?



WALK THROUGH THE EXPERIMENT

<https://chdssprojecttest1.appspot.com/>

1. Instructions are simple, not super wordy, click through (with pictures!)



When the probe lands, it discovers that the surface of Sodor is covered with a variety of spherical rock-like objects

- Need engaged participants
- Need them to understand it!!

WALK THROUGH THE EXPERIMENT

<https://chdssprojecttest1.appspot.com/>

2. There are “understanding check questions” after the instructions

Check your knowledge before you begin!

Question 1: What is your goal in this task?

- To find out if the planet Sodor is made of cheese
- To find out which Sodor spheres have plaxium coatings

Question 2: Which is true about the size of Sodor spheres?

- Spheres on Sodor come in a variety of sizes
- Spheres on Sodor are always large
- Spheres on Sodor are always small

Question 3: Does the probe transmit data about any sphere it encounters?

- Yes, it checks every sphere it encounters
- No, it only tests the small spheres

- Make sure the manipulation worked
- Implicit test for English speaking ability

Submit Answers

WALK THROUGH THE EXPERIMENT

<https://chdssprojecttest1.appspot.com/>

3. Reiterate the important instructions in the experiment; don't assume people will remember everything



Transmissions from the probe will be displayed here when they arrive.

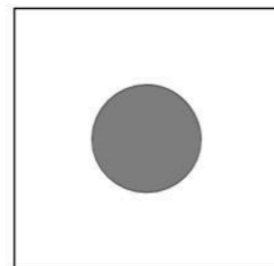
After every few transmissions from the probe, we will pause to ask for your guesses about which spheres have plaxium coatings

The probe has found and tested a small sphere: [Click here to view](#)

WALK THROUGH THE EXPERIMENT

<https://chdssprojecttest1.appspot.com/>

4. Test questions are very clear with clearly labeled axes



In your opinion, how likely is it that a Sodor sphere of this size has a plaxium coating?

1 2 3 4 5 6 7 8 9 10

[1 = Definitely does not]

[10 = Definitely does]

- Depending on the experiment you may designate a few **ahead** of time to yourself (in pre-registration) as filter ones to catch people who aren't paying attention and discard their data
- These should be non-obvious but also clearly justifiable as a filter

WALK THROUGH THE EXPERIMENT

<https://chdssprojecttest1.appspot.com/>

5. Between participants, everything unimportant is randomised as much as possible (e.g., order of test questions, etc)

WALK THROUGH THE EXPERIMENT

<https://chdssprojecttest1.appspot.com/>

6. Clear instructions at the end for what to do

All done!

Your completion code is **216722**. To receive payment for the HIT, return to the Amazon Mechanical Turk page and enter this code. Please contact us if something goes wrong and we'll fix it as quickly as possible.

- Also good to have: debriefing statement explaining what the experiment was about (improves reputation and engagement long-term, builds goodwill)

FIRST STEP: CODING

There are *lots* of ways to do this, and we don't have the scope to teach you Javascript (or whatever) now.

Goal today: Give you the tools you need to get started and teach yourself the rest of it

1. Putting code on your machine
2. Figuring out to run a local version so you can debug it
3. Giving you the basics of how Javascript works and what the code parts are trying to do

PUTTING CODE ON YOUR MACHINE

We're going to work from some example code, which (if you haven't already) you can download here:

<http://chdsommerschool.com/resources.html>



docs



experiment



resources



code



data

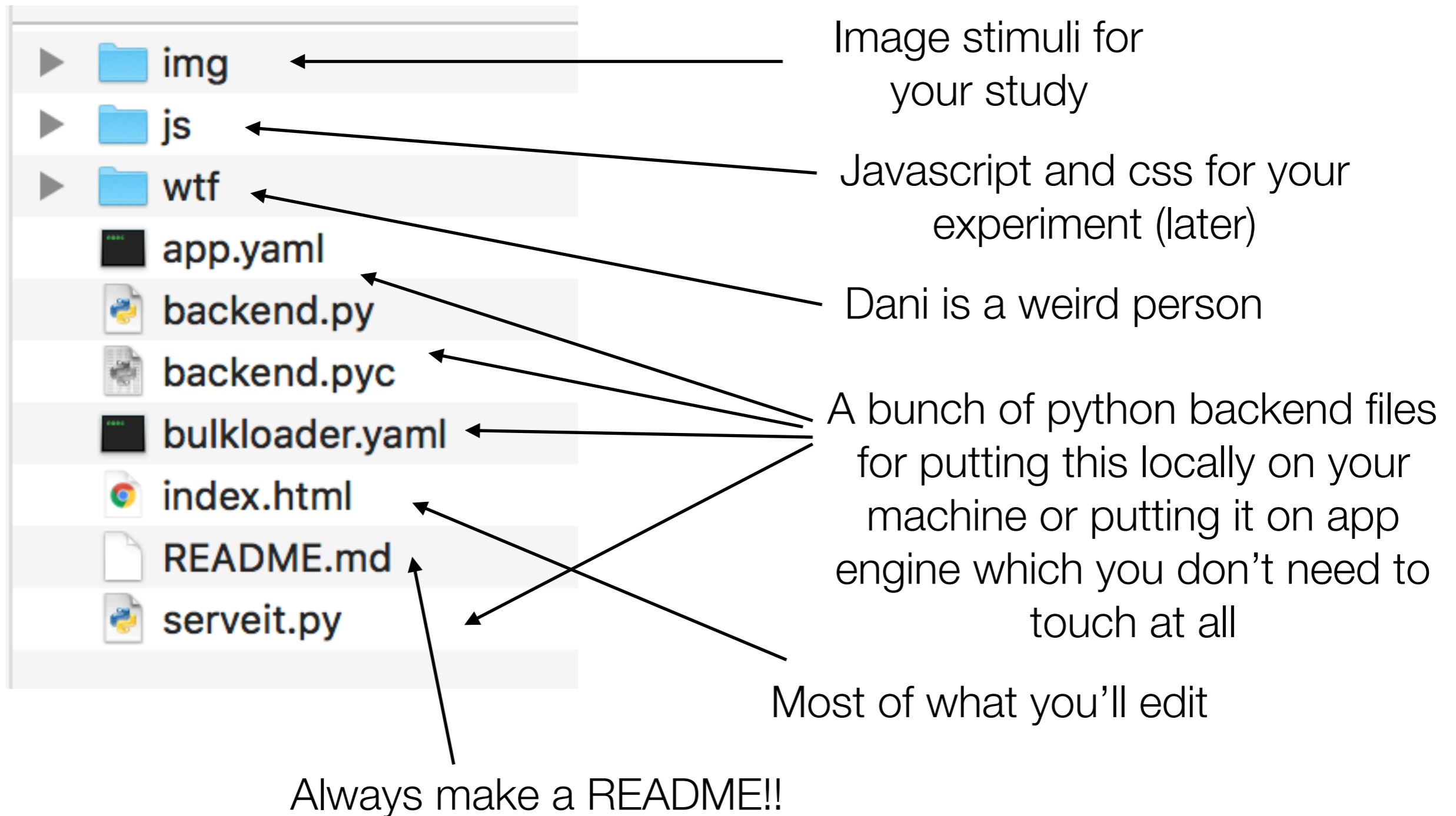


analysis

Unzip the file and put the contents in your experiment directory. Rename it "code"

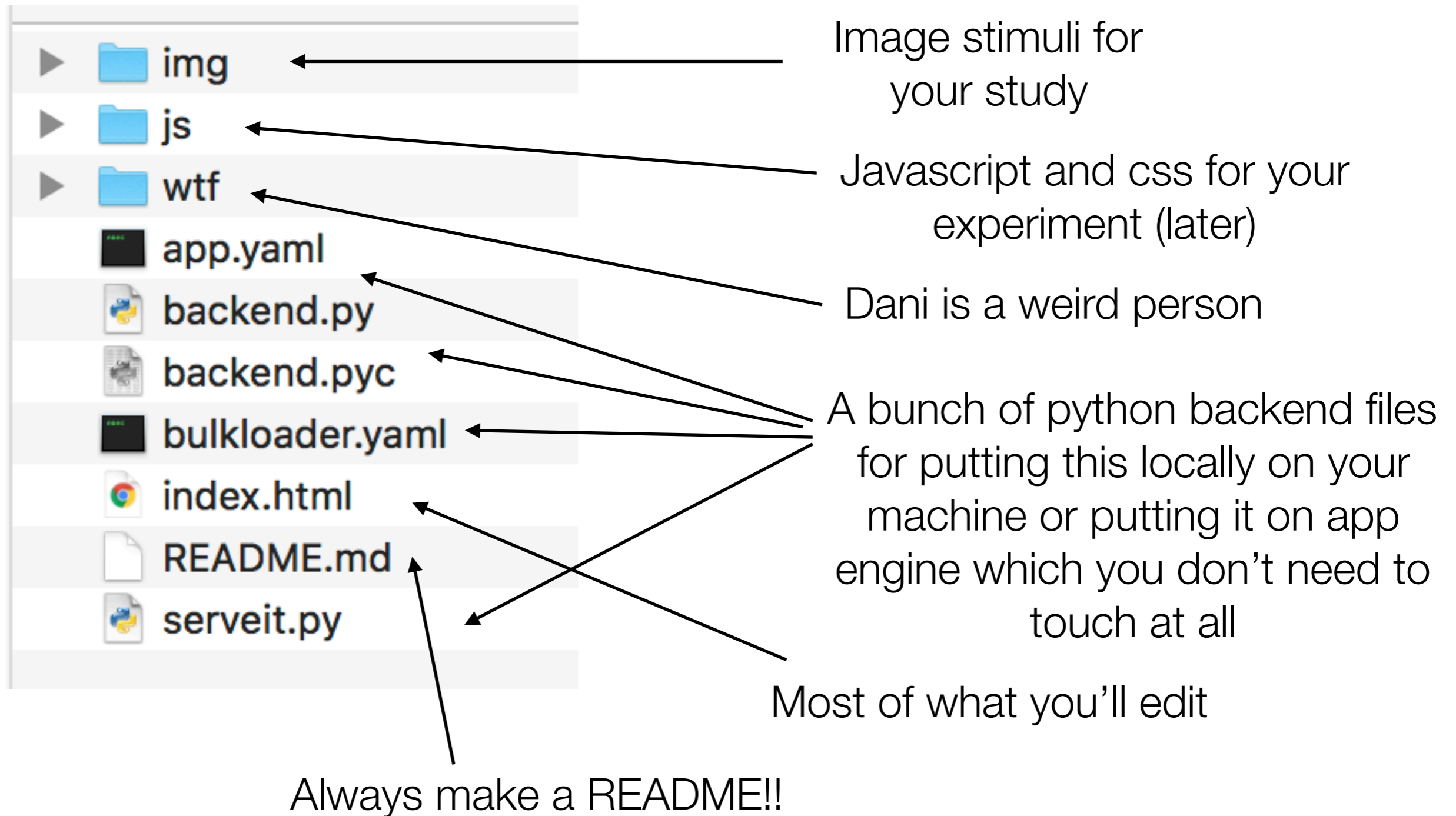
PUTTING CODE ON YOUR MACHINE

It should look something like this:



PUTTING CODE ON YOUR MACHINE

It should look something like this:



README

Contains information you will need later to contextualise your experiment. At a minimum:

- This was the within-subjects sample size experiment
- It is reported as Experiment 2 in the paper

If you have anything unusual about it or want to explain things about the file structure, etc., include that as well.

LOCAL VERSION

Before looking at the specific files, let's see if we can get it to run on your computer. This is super easy if you are using jsPsych and your main file is `index.html` (as we are doing here).

Click on `index.html`.

It will open in your browser!

LOCAL VERSION

More generally if you want to run the python directly, you can use the python script called `serveit.py`

Mac

1. Open terminal (in Applications - Utilities)
2. Go to your folder using `cd` command (`ls` to show contents of directory). This folder needs to be the one with `serveit.py` in it.
`cd Documents/teaching/2018/.../experiment/code/`
3. Type `python serveit.py 8000`. It should say something like:
Serving HTTP on 0.0.0.0 port 8000 ...
4. Go to your browser and type `0.0.0.0:8000`

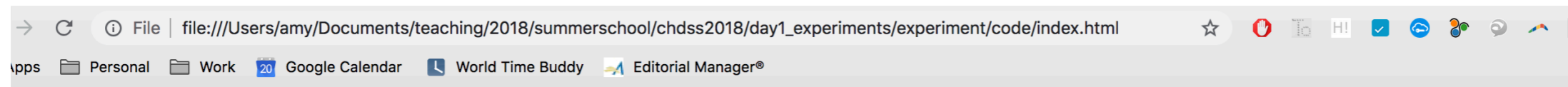
LOCAL VERSION

More generally if you want to run the python directly, you can use the python script called `serveit.py`

Windows

1. Open terminal (Start - then type `cmd` at the Search/Run line)
2. Unlike Mac, Windows must pass the full path of the script to the Python interpreter. If your interpreter is in the `C:\Python27` folder you would type:
`C:\Python27\python.exe C:\Users\Username\Desktop\...\experiment\code\serveit.py 8000`
3. Go to your browser and type `0.0.0.0:8000`

YOU SHOULD SEE THE EXPERIMENT!



UNSW Computational Cognitive Science

Thanks for accepting the HIT. "**The Spheres of Sodor**" is a short psychological study investigating how people make decisions. It involves the following steps:

1. We ask for demographic information (not connected to your Amazon ID)
2. Because this is a University research project, we ask for your informed consent. (The format of the consent form is a standard university document, so it sometimes looks a little weird on MTurk)
3. The study then explains how to do the task in detail. You will need to pass a short test to check that you understand how the study works.
4. Next comes the experiment itself.
5. At the end, we'll give you the completion code you need to get paid for the HIT.

The total time taken should be about 5 minutes. Please don't use the "back" button on your browser or close the window until you reach the end and receive your completion code. This is very likely to break the experiment and may make it difficult for you to get paid. However, if something does go wrong, please contact us! When you're ready to begin, click on the "start" button below.

Start!

NOW... HOW CAN WE MAKE
AND PUT UP SOMETHING LIKE
THIS OURSELVES?

JAVASCRIPT: THE BASIC IDEA

Server side

In this case, this is you!
You (or your server) are serving up an experiment to your participant.



Client side

The client's web browser serves up webpages.

Usually HTML: a markup language for displaying all your content

Javascript is a client-side language that lets you do more complex things

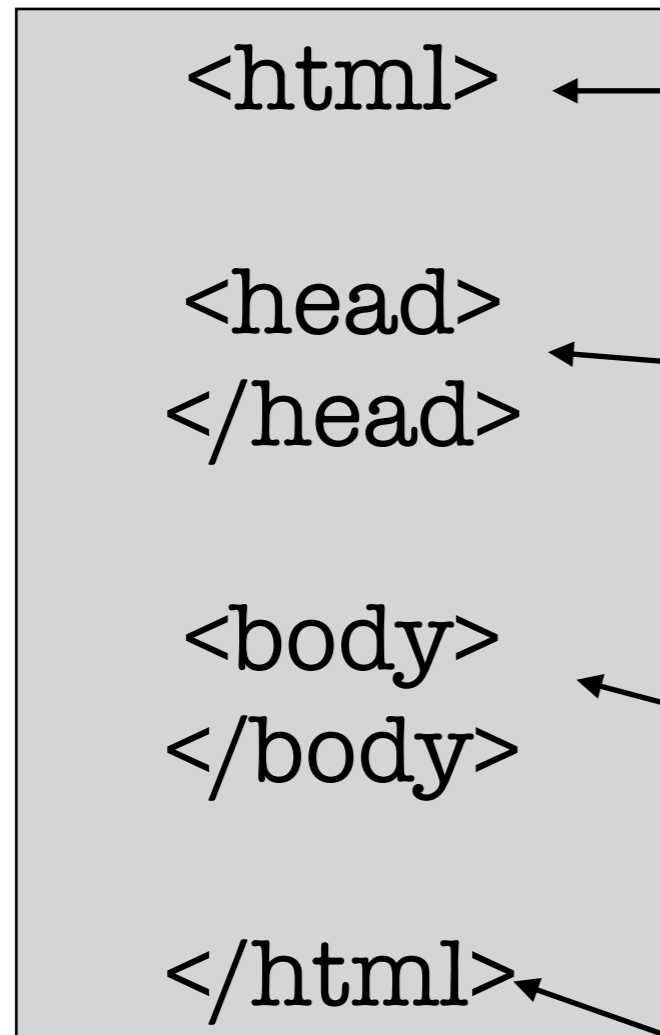
It is thus embedded within html

As programmer, you are writing the html/js so that the browser on the client side knows what to do



HTML

Any webpage has the same basic structure



Tells the browser that an html file is beginning


Tells the browser that here is the header info, and ends it (with /) ... in between is usually lots of stuff

Begins and ends the body ... in between is usually lots of stuff

Tells the browser that an html file is ending

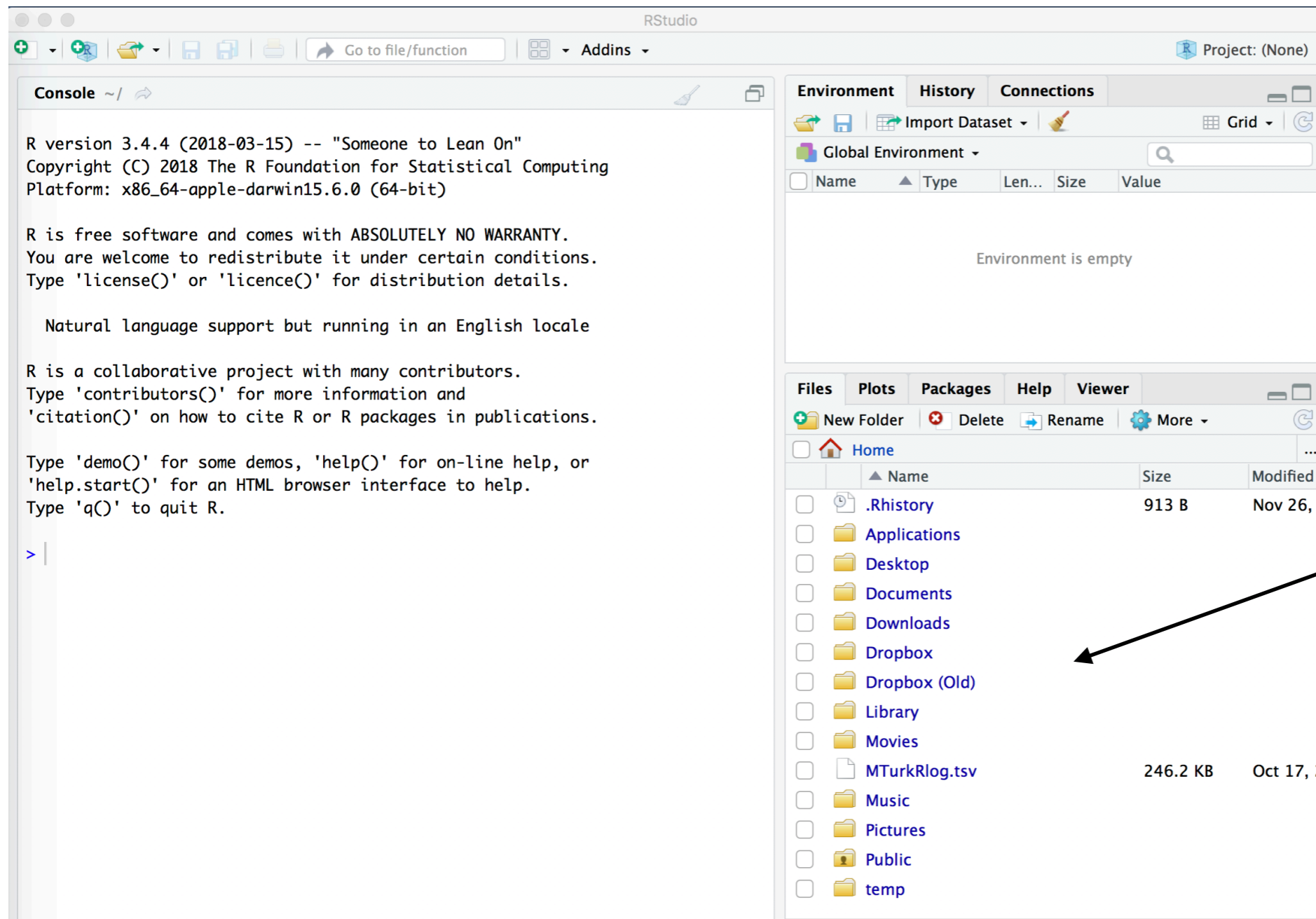
HTML

HTML works by using tags, which are basically commands for the browsers. Many lists of tags can be found online, e.g. here: <https://www.w3schools.com/tags/>

<pre><a></pre> <pre>
</pre> <pre></pre> <pre></pre>	<p>← Hyperlink <code>Link to summer school</code></p> <p>← Line break Hello!<code>
</code>I am so pleased to be here.</p> <p>← Emphasised text Summer school is <code></code>so<code></code> cool.</p> <p>← Defines an image <code></code></p>	<pre>Link to summer school</pre> <pre>Hello!</pre> <pre>I am so pleased to be here.</pre> <pre>Summer school is so cool.</pre> <pre></pre> 
---	--	---

HTML

Let's make a super basic webpage.



Open RStudio,
and navigate to
your experiment/
code folder

HTML

Let's make a super basic webpage.

The screenshot shows the RStudio interface. The console on the left displays the R version 3.4.4 (2018-03-15) and various help messages. The file explorer on the right shows the current working directory: `~/school/chdss2018/day1_experiments/experiment/code`. The file list includes:

Name	Size	Modified
..		
app.yaml	310 B	Oct 29, 2018
backend.py	1.1 KB	Oct 29, 2018
backend.pyc	2.1 KB	Oct 29, 2018
bulkloader.yaml	1.7 KB	Oct 29, 2018
img		
index.html	35.5 KB	Oct 29, 2018
js		
README.md	100 B	Oct 29, 2018
serveit.py	536 B	Mar 29, 2018
wtf		

Open RStudio,
and navigate to
your experiment/
code folder

Set it as your
working directory

HTML

Make a new file (choose 'text file'. R HTML would work but that adds a bunch of stuff we don't need right now)

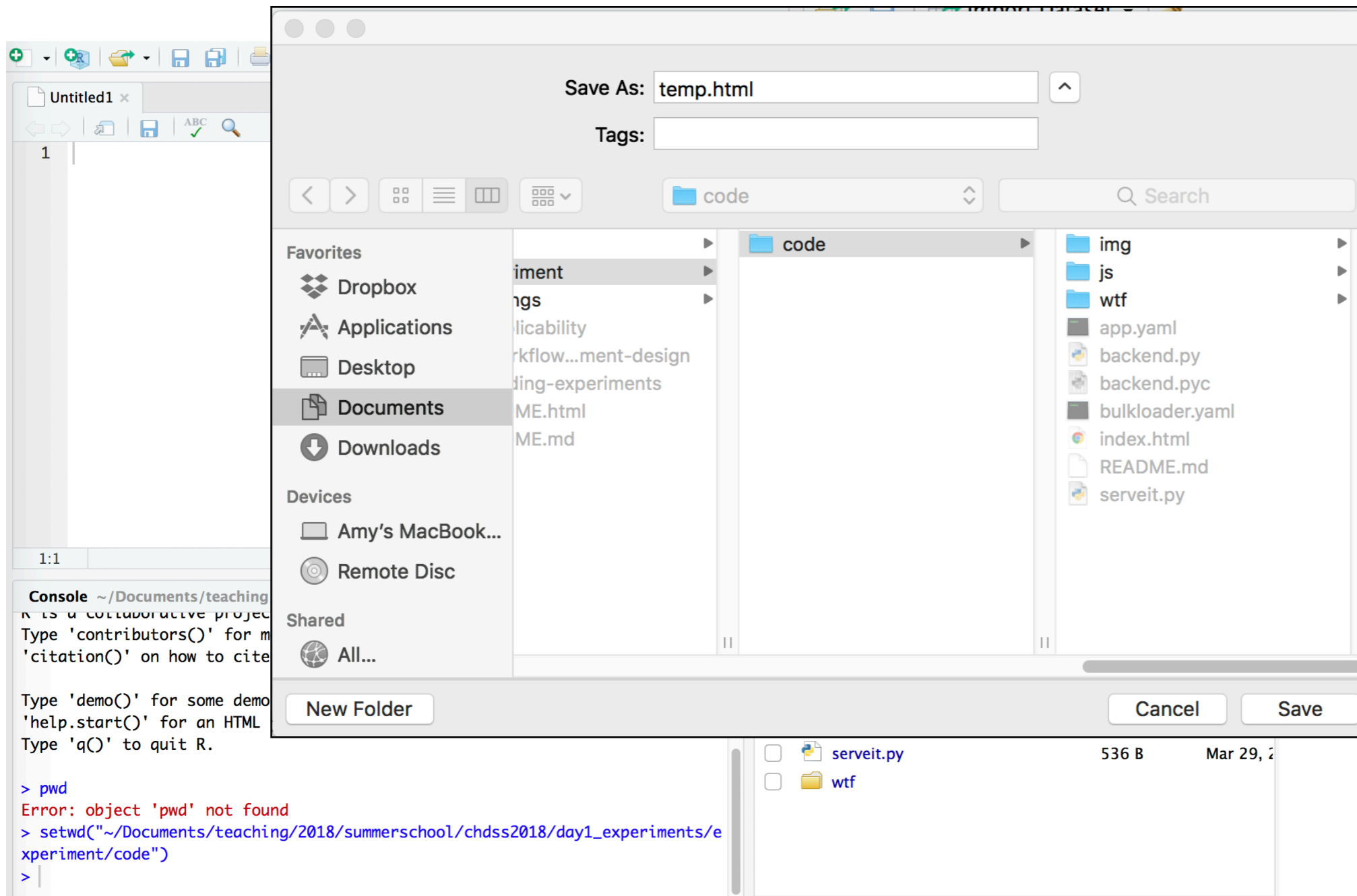
The screenshot shows the RStudio IDE interface. The main editor window is titled 'Untitled1' and contains a single line of text '1'. The environment pane on the right shows 'Global Environment' and is empty. The file explorer pane shows the current directory structure: `~/Documents/teaching/2018/summerschool/chdss2018/day1_experiments/experiment/code`. The console pane at the bottom shows the following output:

```
> pwd
Error: object 'pwd' not found
> setwd("~/Documents/teaching/2018/summerschool/chdss2018/day1_experiments/experiment/code")
>
```

Name	Size	Modified
..		
app.yaml	310 B	Oct 29, 2
backend.py	1.1 KB	Oct 29, 2
backend.pyc	2.1 KB	Oct 29, 2
bulkloader.yaml	1.7 KB	Oct 29, 2
img		
index.html	35.5 KB	Oct 29, 2
js		
README.md	100 B	Oct 29, 2
serveit.py	536 B	Mar 29, 2
wtf		

HTML

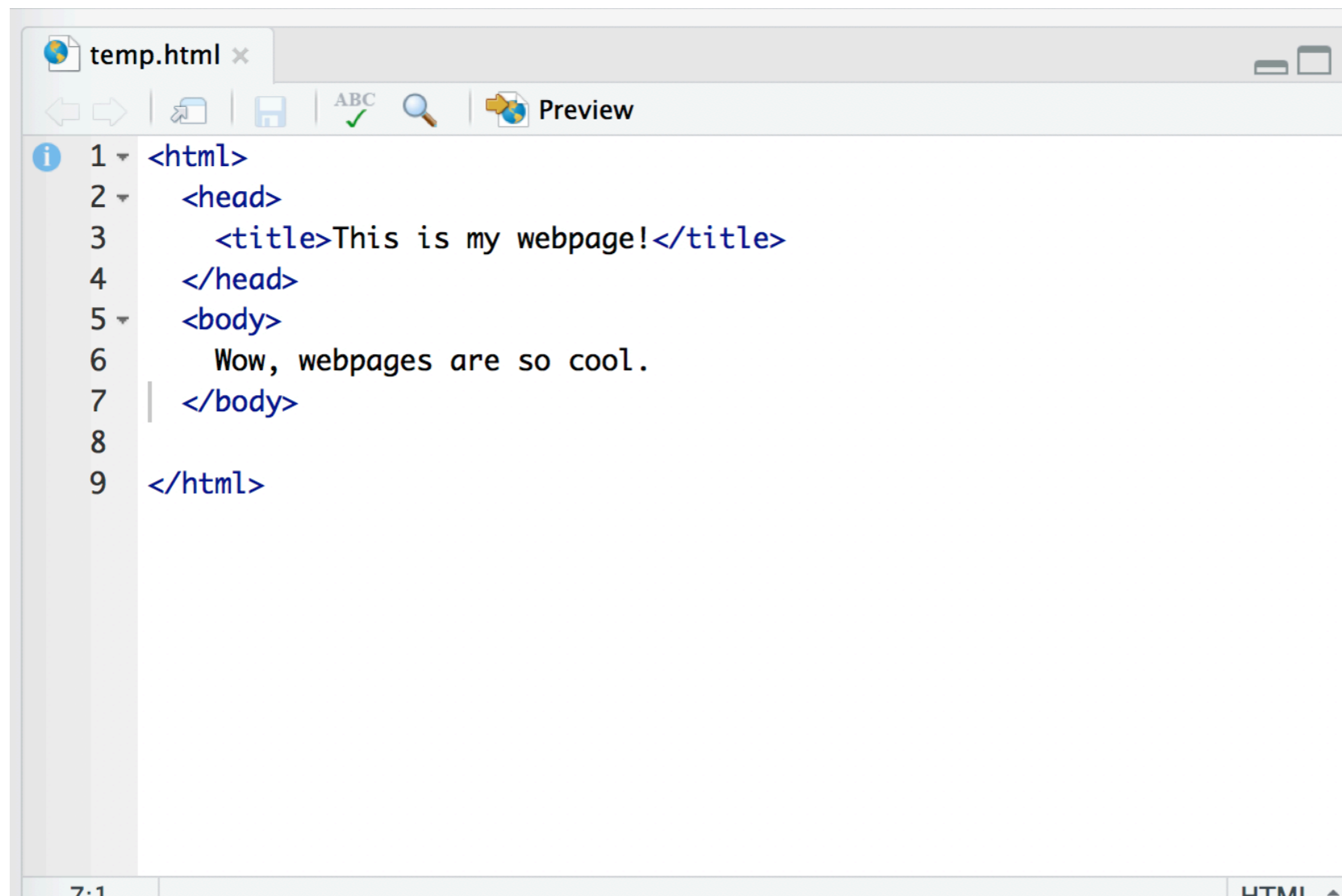
Make a new file (choose 'text file'. R HTML would work but that adds a bunch of stuff we don't need right now)



Because I'm paranoid, I always save first thing

HTML

Now let's create a hello message using our template.

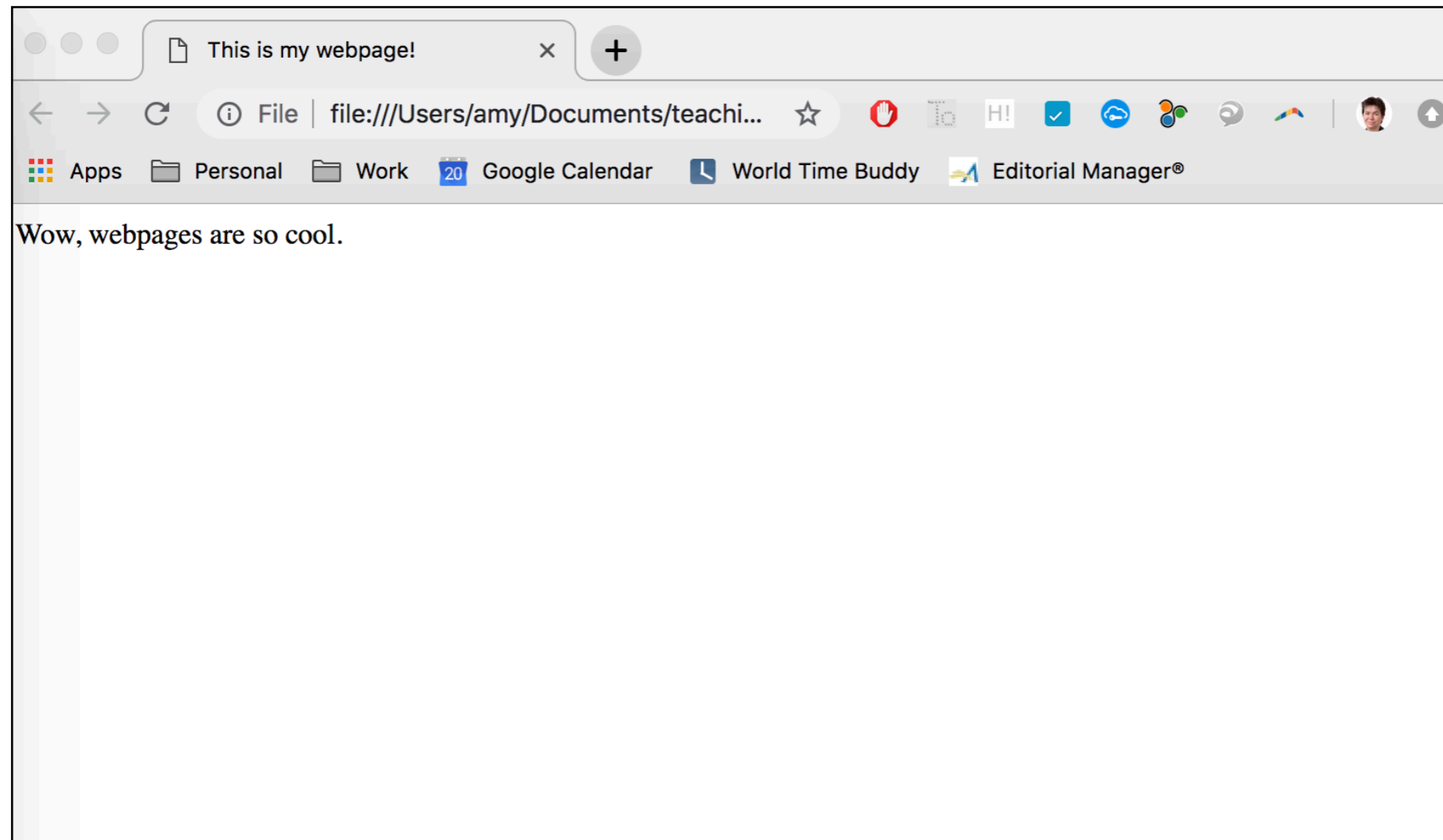


```
temp.html x
← → | 📄 | 💾 | ABC ✓ | 🔍 | 🌐 Preview
1 <html>
2   <head>
3     <title>This is my webpage!</title>
4   </head>
5   <body>
6     Wow, webpages are so cool.
7   </body>
8
9 </html>
```

7:1 | HTML ▲

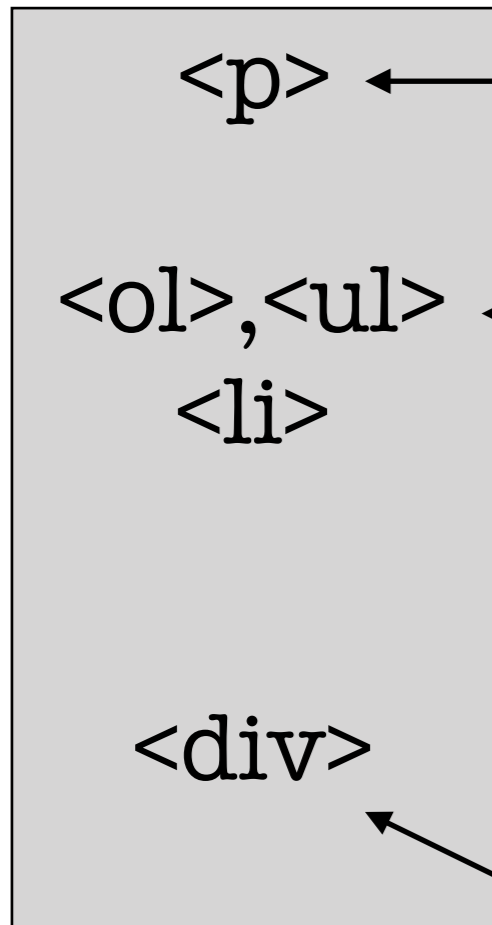
HTML

Save and click on `temp.html` and you should see something like this come up in your browser!



HTML

Let's add a few more useful tags...



Paragraph break

Wow!
`<p>`This is so cool.

Wow!

This is so cool.

``, ``

List making

``

``

`` Pianos ``

`` Lettuce ``

`` Cockatoos ``

``

1. Pianos
2. Lettuce
3. Cockatoos

`<div>`

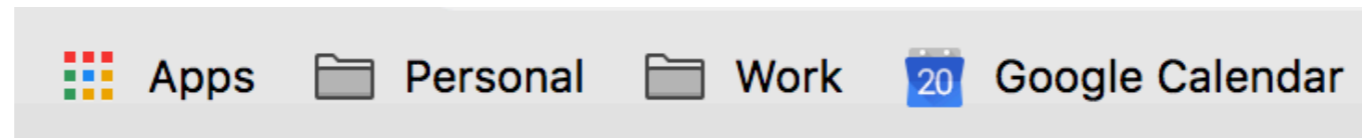
Defines a section ("division")
of your page which you can
refer to later

```
<ol>  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ol>
```

```
<div class="beginningPart" style="width: 600px">
```

EXERCISES

Modify `temp.html` so the website looks like the one below:



Hello world!

My favourite things are:

1. Bunnies.
2. Even more bunnies.
3. Minecraft.

[Here](#) is a link to the summer school website.

HOW DOES JAVASCRIPT COME IN?

Remember that javascript is *code* that can go into a webpage so it can do more complicated things than display information. There are lots of ways to do this, but we'll start with a very simple exercise to illustrate the idea.

HOW DOES JAVASCRIPT COME IN?

First thing we want to do is put the actual functions there; they can go in the head or after the body.

```
</body>
<script language="JavaScript">
  function temperature(form) {
    var c = parseFloat(form.DegC.value, 10);
    var f = 0;
    f = c * (9.0/5.0) + 32;
    form.DegF.value = f;
  }
</script>
```

Tells the browser what language the code is in; this is optional, can just do `<script>`

Name of the function is `temperature` and it takes a `form` element as input

Two variables: `c` is the value entered at the form, `f` is what we are converting to

Once we calculate the temp in F, we assign that to the relevant value on the form

HOW DOES JAVASCRIPT COME IN?

This by itself does nothing visible, because it doesn't affect what is showing on the webpage. For that we need to change the body.

```
<body>
  <form>
    <div class="question" style="width: 1000px">
      <h2>Celsius to Fahrenheit Converter</h2>
      Enter a temperature in degrees C:
      <input name="DegC" value="0" maxlength="15" size=15>
      <br>
    </div>
    <div class="answer" style="width: 1000px">
      Click this button to calculate the temperature in degrees F:
      <input name="calc" value="Calculate" type="button" onClick=temperature(this.form)>
      <br><br>
      Temperature in degrees F is:
      <input name="DegF" readonly size=15>
    </div>
  </form>
</body>
```

Labels this part of the code in case we want to refer to it later

Tells the user what the webpage does (`h2` makes it a headline font)

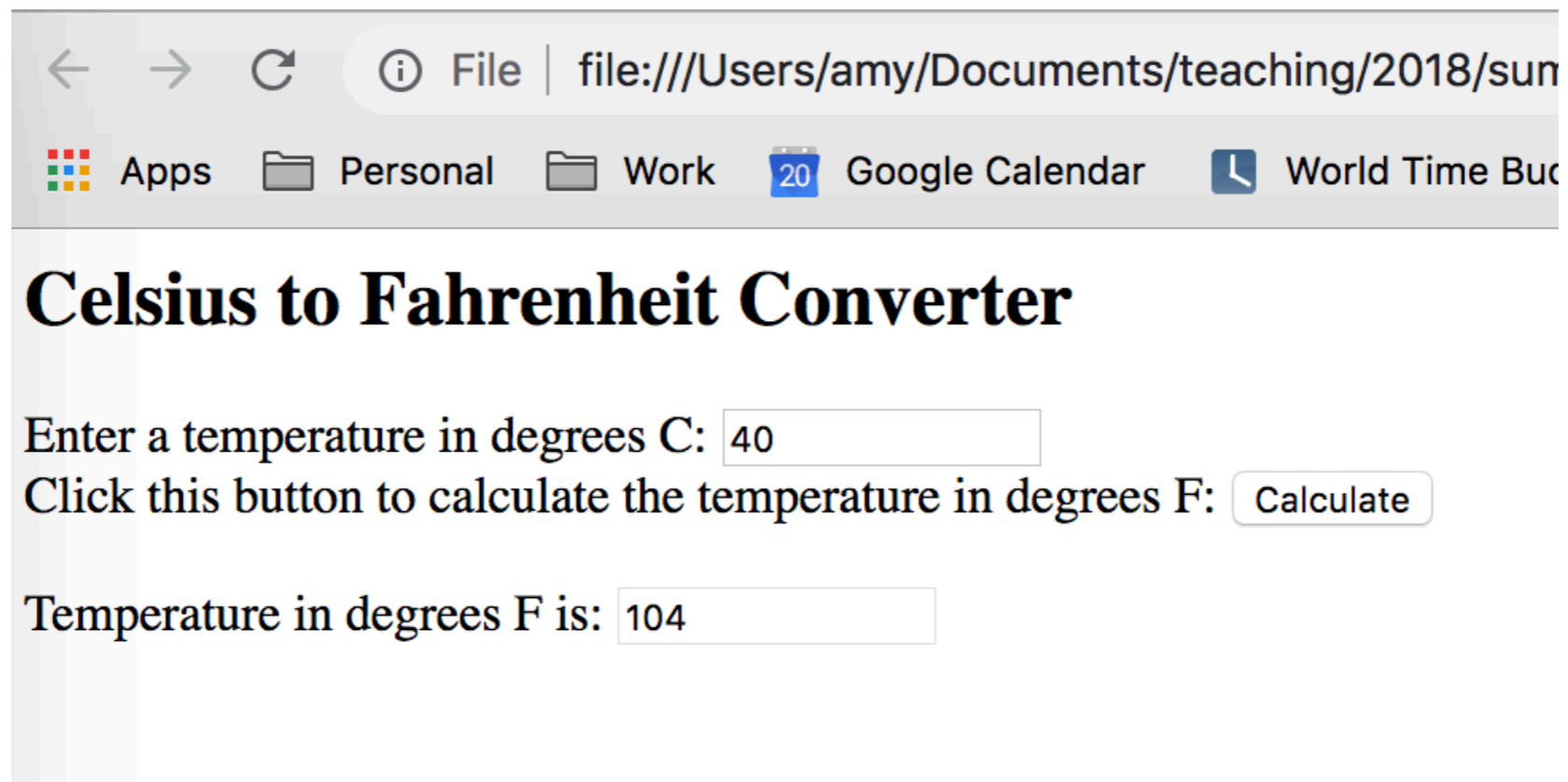
Creates an input box which is initialised at value 0, and calls it `DegC` (for the function to refer to)

Creates a button which when clicked calls the `temperature()` function we just created, and sends it this form element

Once the script calculates `DegF` this is made visible in this readonly element

HOW DOES JAVASCRIPT COME IN?

Give it a try!



The screenshot shows a web browser window with a file:// URL. The browser's address bar contains the text "file:///Users/amy/Documents/teaching/2018/sun". Below the address bar, there are several icons for "Apps", "Personal", "Work", "Google Calendar", and "World Time Buc". The main content of the page is a "Celsius to Fahrenheit Converter". It features a text input field with the value "40" and a "Calculate" button. Below the input field, it says "Temperature in degrees F is:" followed by another text input field containing the value "104".

← → ↻ ⓘ File | file:///Users/amy/Documents/teaching/2018/sun

Apps Personal Work 20 Google Calendar World Time Buc

Celsius to Fahrenheit Converter

Enter a temperature in degrees C:

Click this button to calculate the temperature in degrees F:

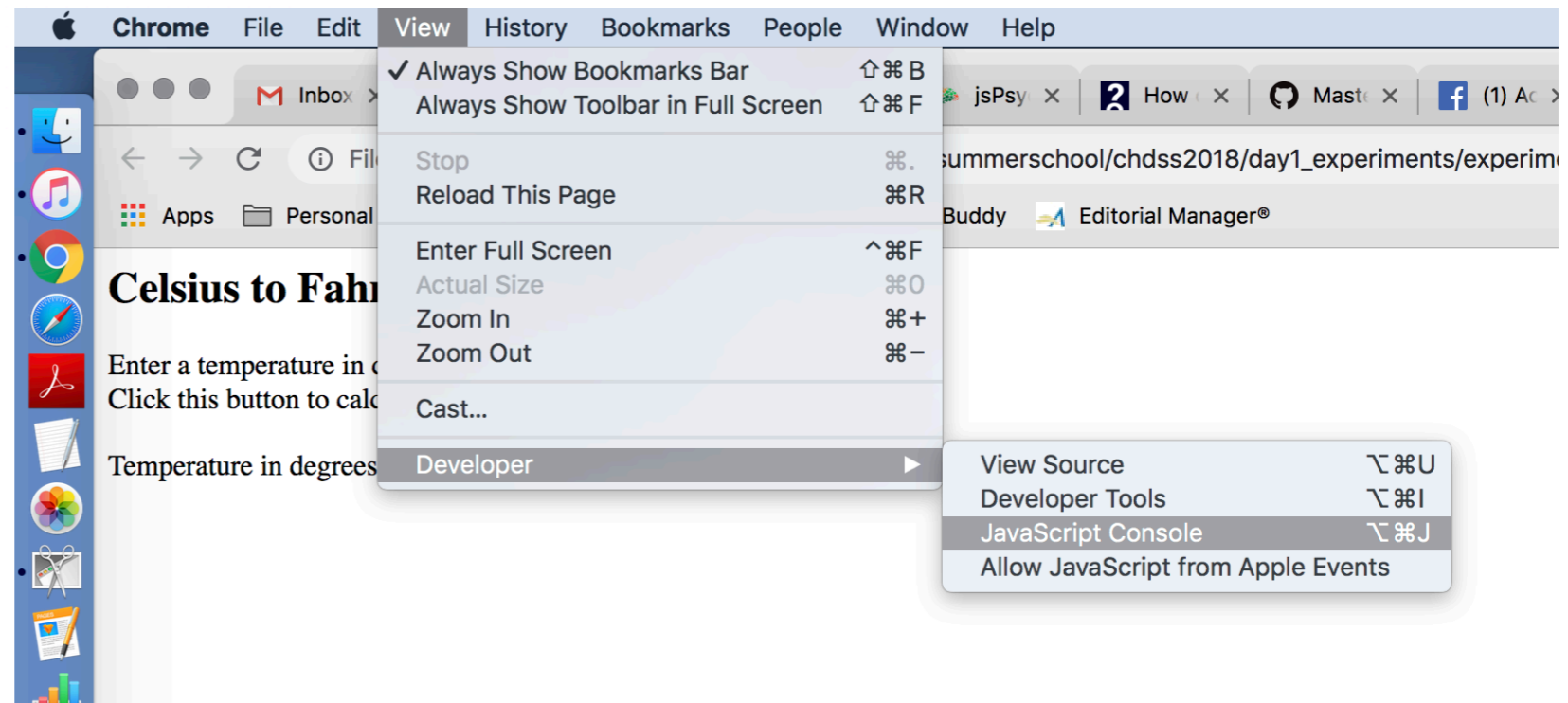
Temperature in degrees F is:

HOW DOES JAVASCRIPT COME IN?

What if you screwed up?

In Chrome, go to Javascript console, which gives you error messages

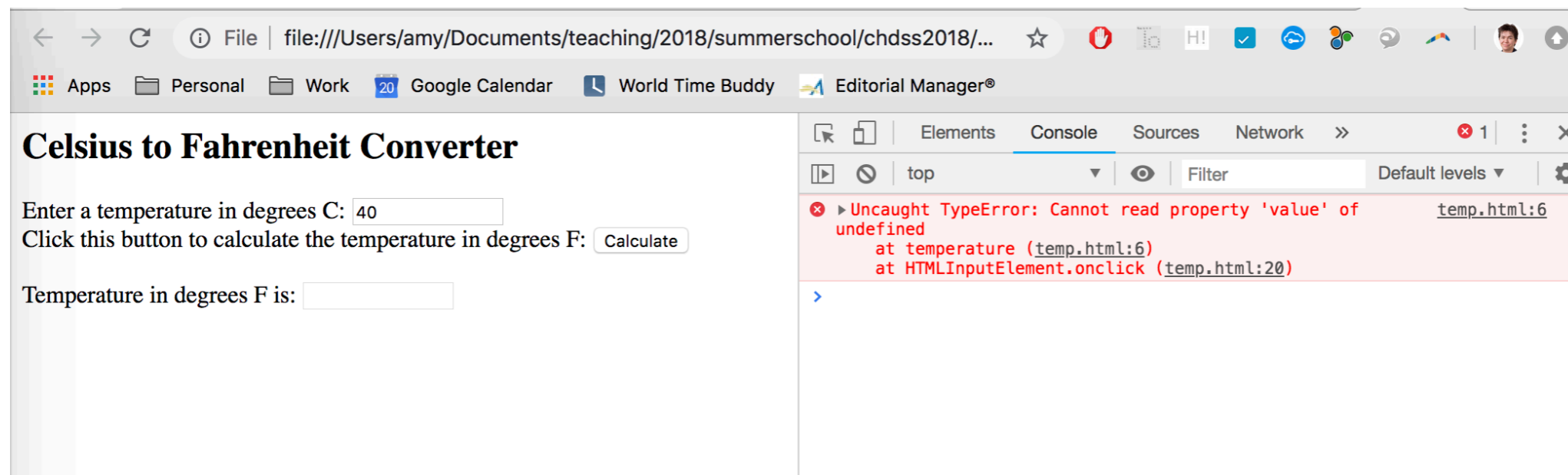
```
<script language="JavaScript">
function temperature(form) {
  var c = parseFloat(form.DeggC.value, 10);
  var f = 0;
  f = c * (9.0/5.0) + 32;
  form.DegF.value = f;
}
</script>
```



HOW DOES JAVASCRIPT COME IN?

What if you screwed up?

In Chrome, go to Javascript console, which gives you error messages



The screenshot shows a web browser window with a file:// URL. The page title is "Celsius to Fahrenheit Converter". The form contains an input field with the value "40", a "Calculate" button, and an output field. The Chrome DevTools console is open, displaying a red error message: "Uncaught TypeError: Cannot read property 'value' of undefined". The stack trace shows the error occurred at line 6 of temp.html, specifically within the HTMLInputElement.onclick function at line 20.

Celsius to Fahrenheit Converter

Enter a temperature in degrees C:

Click this button to calculate the temperature in degrees F:

Temperature in degrees F is:

Uncaught TypeError: Cannot read property 'value' of undefined
at temperature (temp.html:6)
at HTMLInputElement.onclick (temp.html:20)

EXERCISE

Try modifying this function to convert from Fahrenheit to Celcius instead

A SUPER USEFUL TOOL: JSPSYCH

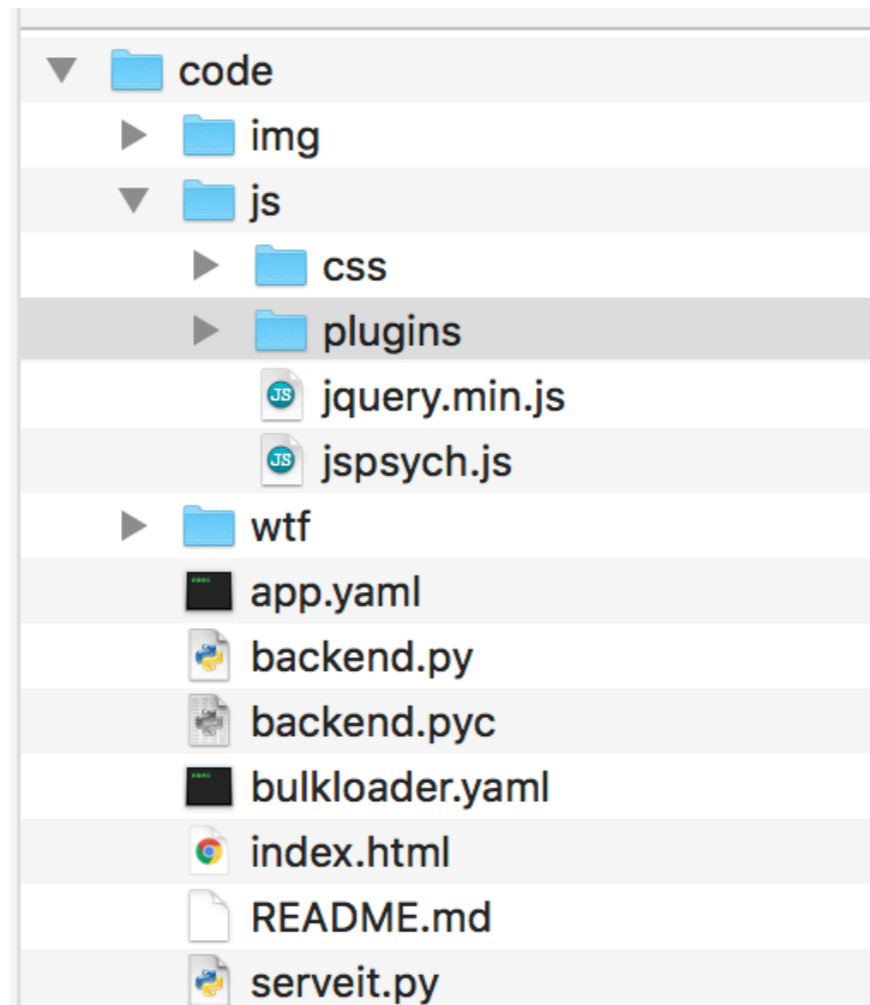
jsPsych is a library of javascript functions designed specifically to administer web experiments

We do not have time to make you experts in this, but they have great tutorials on their webpage:
<https://www.jspsych.org/>

My goal here is to give you sufficient background that, in combination with the tutorial, the template code, and other javascript resources online, you can teach yourself what you need

A SUPER USEFUL TOOL: JSPSYCH

The way jsPsych works is by creating a bunch of plugins that are Javascript code you can call to do some of the complicated stuff in your experiment



You've already
downloaded
jsPsych and the
plugins!

(although maybe
not all of them)

A SUPER USEFUL TOOL: JSPSYCH

To include jsPsych and the plugins in your html file, you can just link to them rather than writing out the whole script!

Let's start a new one called `sampleExpt.html` and try:

Imports the main jsPsych library

```
<html>
  <head>
    <title>Sample Experiment</title>
    <script src="./js/jspsych.js"></script>
    <link href="./js/css/jspsych.css" rel="stylesheet" type="text/css"></link>
  </head>
  <body>
  </body>
</html>
```

Imports a stylesheet, which basically is a set of guidelines making the visual presentation nice

A SUPER USEFUL TOOL: JSPSYCH

Let's start with what we know...

```
<body>
<!-- Starting screen -->
<div class="start" style="width: 1000px">
  <!-- Text box for the splash page -->
  <div class="start" style="text-align:left; border:0px solid; padding:10px; width:800px; float:right; font-size:90%">
    <p><b>"The Spheres of Sodor"</b> is a short psychological study investigating how people make decisions.</p>
  </div>
</div>
</body>
```

So far this doesn't use Javascript, it's just presenting the webpage. (However, the formatting is pretty nice!)

"The Spheres of Sodor" is a short psychological study investigating how people make decisions.

A SUPER USEFUL TOOL: JSPSYCH

Now we can add a button...

```
<body>
<!-- Starting screen -->
<div class="start" style="width: 1000px">
  <!-- Text box for the splash page -->
  <div class="start" style="text-align:left; border:0px solid; padding:10px; width:800px; floa
    <p><b>"The Spheres of Sodor"</b> is a short psychological study investigating how people ma
    <!-- Next button for the splash page -->
    <p align="center">
      <input type="button" id="splashButton" class="start jspsych-btn" value="Start!"
        onclick="splashButtonClick()"> </p>
  </div>
</div>
</body>
```

We haven't yet written the code for the function `splashButtonClick()` so this makes a button but the button doesn't do anything

"The Spheres of Sodor" is a short psychological study investigating how people make decisions.

Start!

A SUPER USEFUL TOOL: JSPSYCH

In the scripts at the end we add:

```
<script>
```

```
// Some basic functions
```

```
function splashButtonClick() {  
    setDisplay('start', 'none');  
    setDisplay('consent', '');  
}
```

```
// Function to change the display property of a set of objects
```

```
function setDisplay(theClass, theValue) {  
    var i, classElements = document.getElementsByClassName(theClass);  
    for (i = 0; i < classElements.length; i = i + 1) {  
        classElements[i].style.display = theValue;  
    }  
}
```

```
</script>
```

Sets the display value for the “start” element (div) to “none” (i.e., gets rid of it) and also calls the “consent” element (which doesn’t exist yet)

Now when you press on the button, the screen clears but nothing replaces it

We need to make a `div` for the consent form!

A SUPER USEFUL TOOL: JSPSYCH

Adding a consent form...

```
<!-- Consent form -->  
<div class="consent" style="display:none; width:1000px">  
  <!-- Text box for the splash page -->  
  <div class="consent" style="text-align:left; border:0px solid; padding:10px; width:800px;  
    font-size:90%; float:right">  
    <p align="center">Consent form page</p>  
  </div>  
  <br><br>  
</div>
```

Now when you press the button it
goes there!

A SUPER USEFUL TOOL: JSPSYCH

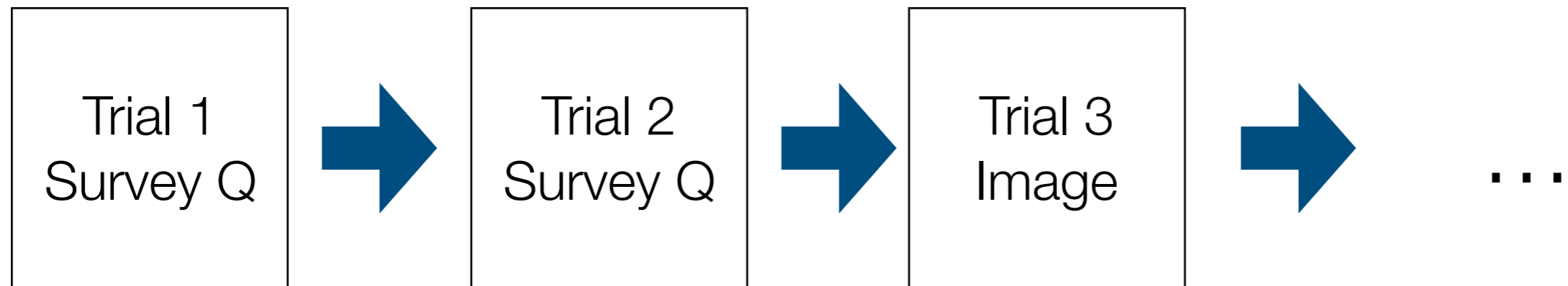
So far we haven't actually used jsPsych much, so let's give it a shot.

First, let's include a bunch more plugins

```
<head>
  <title>Sample Experiment</title>
  <script src="./js/jquery.min.js"></script>
  <script src="./js/jspsych.js"></script>
  <script src="./js/plugins/jspsych-text.js"></script>
  <script src="./js/plugins/jspsych-single-stim.js"></script>
  <script src="./js/plugins/jspsych-survey-multi-choice.js"></script>
  <script src="./js/plugins/jspsych-button-response.js"></script>
  <link href="./js/css/jspsych.css" rel="stylesheet" type="text/css"></link>
</head>
```

A SUPER USEFUL TOOL: JSPSYCH

The way jsPsych works is by building a description of an experiment known as a **timeline**, which is basically a series of variables defining each step (trial).



Timeline

A SUPER USEFUL TOOL: JSPSYCH

We'll start by creating a function that is called when you click "I agree" on the consent form:

```
<!-- Consent form -->
<div class="consent" style="display:none; width:1000px">
  <!-- Text box for the splash page -->
  <div class="consent" style="text-align:left; border:0px solid; padding:10px; width:800px;
    font-size:90%; float:right">
    <p align="center">Consent form page</p>
    <p align="center">
      <input type="button" id="consentButton" class="consent jspsych-btn" value="I agree" onclick="startExperiment()" >
    </div>
    <br><br>
  </div>
</div>
```

```
// start experiment
function startExperiment(){
  setDisplay('consent', 'none');
  jsPsych.init({
    timeline: [instruction_check]
  })
}
```

Down in `<script>...</script>` land

Disappears the consent form page

Initialises the experiment with the timeline, which has one thing in it

A SUPER USEFUL TOOL: JSPSYCH

What is the thing in the timeline?!??!

This is basically a complicated kind of javascript variable which contains multiple values. The plugins describe what values are appropriate for that kind of variable. In this case, our instruction checks are a 1-question survey.

```
<script>
```

```
// initialise variables //  
var timeline=[];  
var instruction_check = {  
  type: "survey-multi-choice",  
  preamble: ["<p align='center'><b>Check your knowledge before you begin!</b></p>"],  
  questions: ["<b>Question 1</b>: Does the probe transmit data about any sphere it encounters?"],  
  options: [["Yes","No"]]  
}
```

Initialises an empty timeline

Tells which function to use

Questions and answers

A SUPER USEFUL TOOL: JSPPSYCH

This now gives you a single question!

Check your knowledge before you begin!

Question 1: Does the probe transmit data about any sphere it encounters?

- Yes
- No

Submit Answers

WHERE FROM HERE?

We do not have the time to go into more details, but the core is already here — the rest is just scaling

- ▶ Can have multiple trials in a timeline, not just one
 - ▶ (or nest them, which lets you create blocks of trials)
- ▶ Can have variables that save the data
- ▶ Plugins naturally let you randomise, or do different things depending on condition
- ▶ Can sample trials / test questions from a pool
- ▶ Loop timeline (e.g, if people need to redo for any reason)

ONE FINAL THING

The code we gave you for index.html inadvertently commented out the saving data part — please uncomment that so it saves data!

```
/* save and finish */  
function endExperiment(dataset, callback) {  
  
    $.post('submit', {"content": dataset});  
    setTimeout(callback, 1000)  
  
}
```

This should look like this and not be
in a comment



RESOURCES

1. The actual code from the Sodor experiment, so you can see how it was done and change things that way
2. *Strongly* recommend the tutorial pages at www.jspsych.org
3. If in doubt, just google! I seriously learned everything I know that way; never took a class myself.

Note: There are *lots* of ways to do things in Javascript, and also lots of ways this code could be improved — don't be afraid to play or to trust yourself