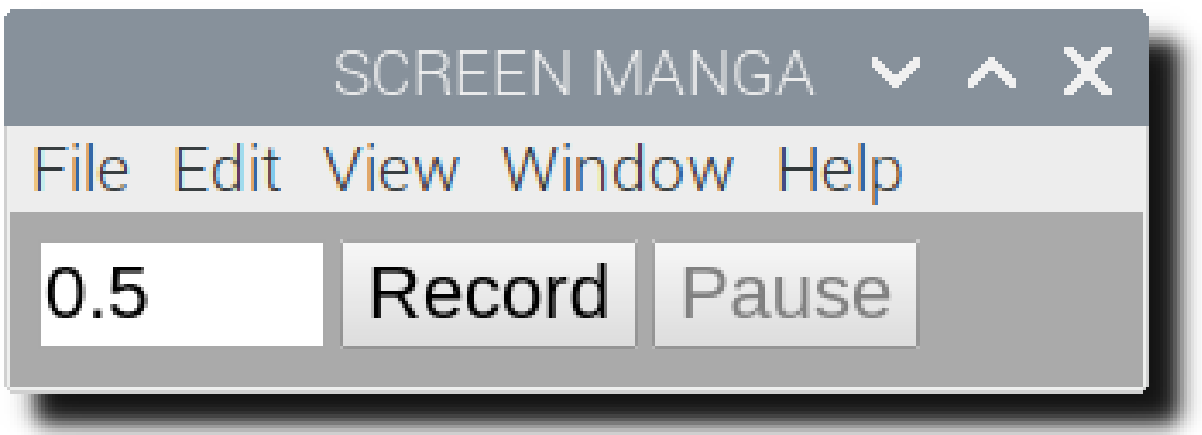
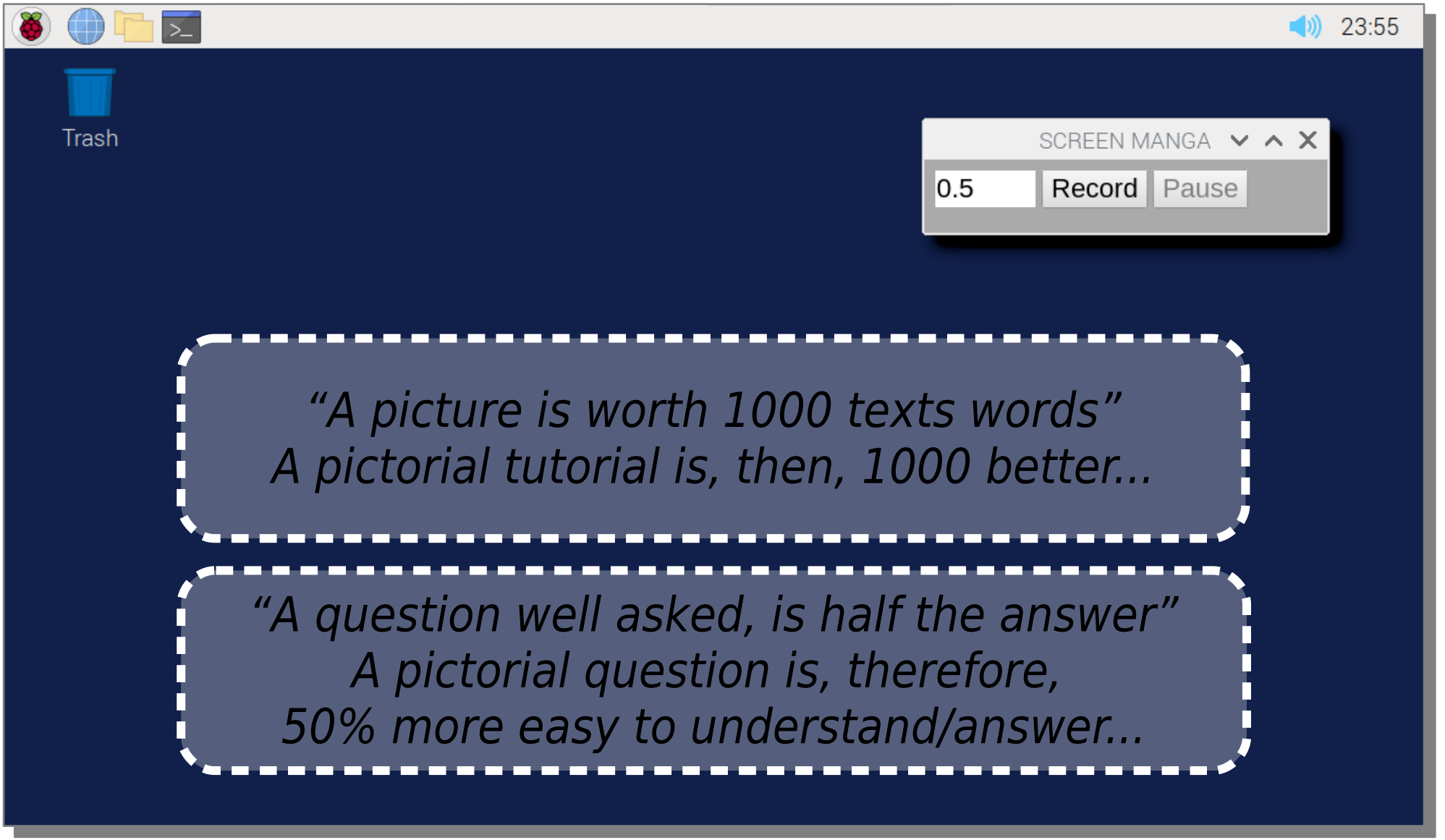
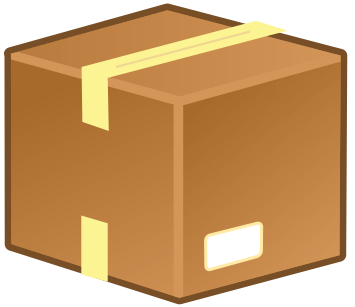
ScreenManga GUI (Graphical Interface)



HTML + JavaScript + NodeJS (JavaScript for OS) + Graphical User Interface


23:55
Trash
SCREEN MANGA
0.5
Record
Pause
“A picture is worth 1000 texts words”
A pictorial tutorial is, then, 1000 better...
“A question well asked, is half the answer”
A pictorial question is, therefore,
50% more easy to understand/answer...



electron-v8.0.3-linux-armv7l.zip

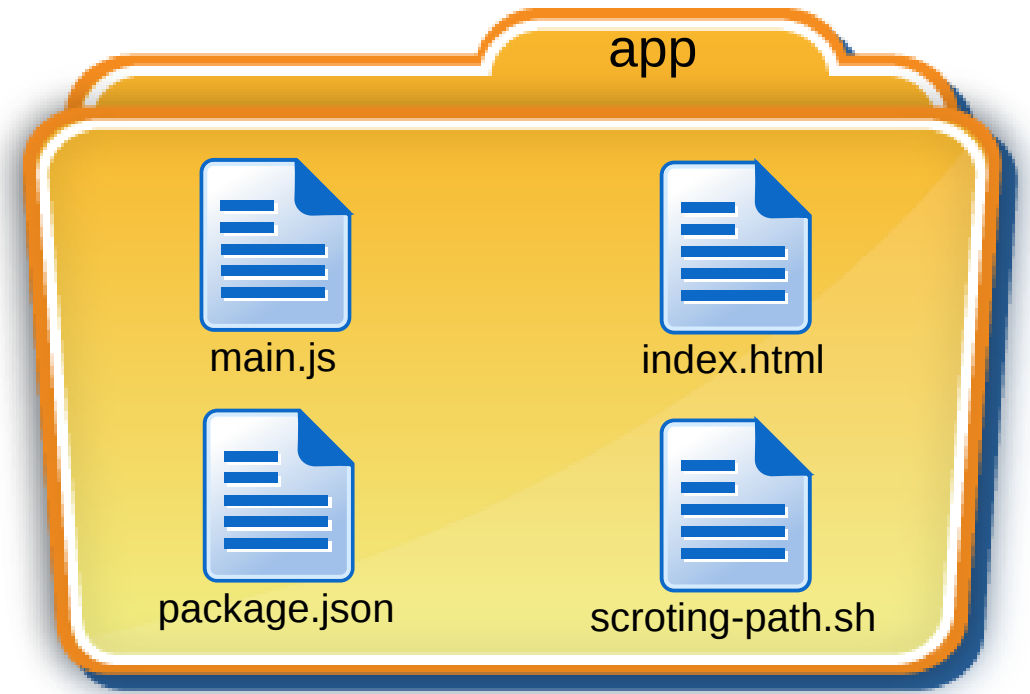
```
#  Download the electron binary package…

#  Following commands can also be done by mouse, if you want...

#  Create a folder to extract the package
mkdir   screenManga

#  Move (mv) or copy (cp) to inside the electron-folder
mv   electron-v8.0.3-linux-armv7l.zip   screenManga/

#  Enter the folder...
cd   screenManga

#  Extract the package...
unzip   electron-v8.0.3-linux-armv7l.zip
```

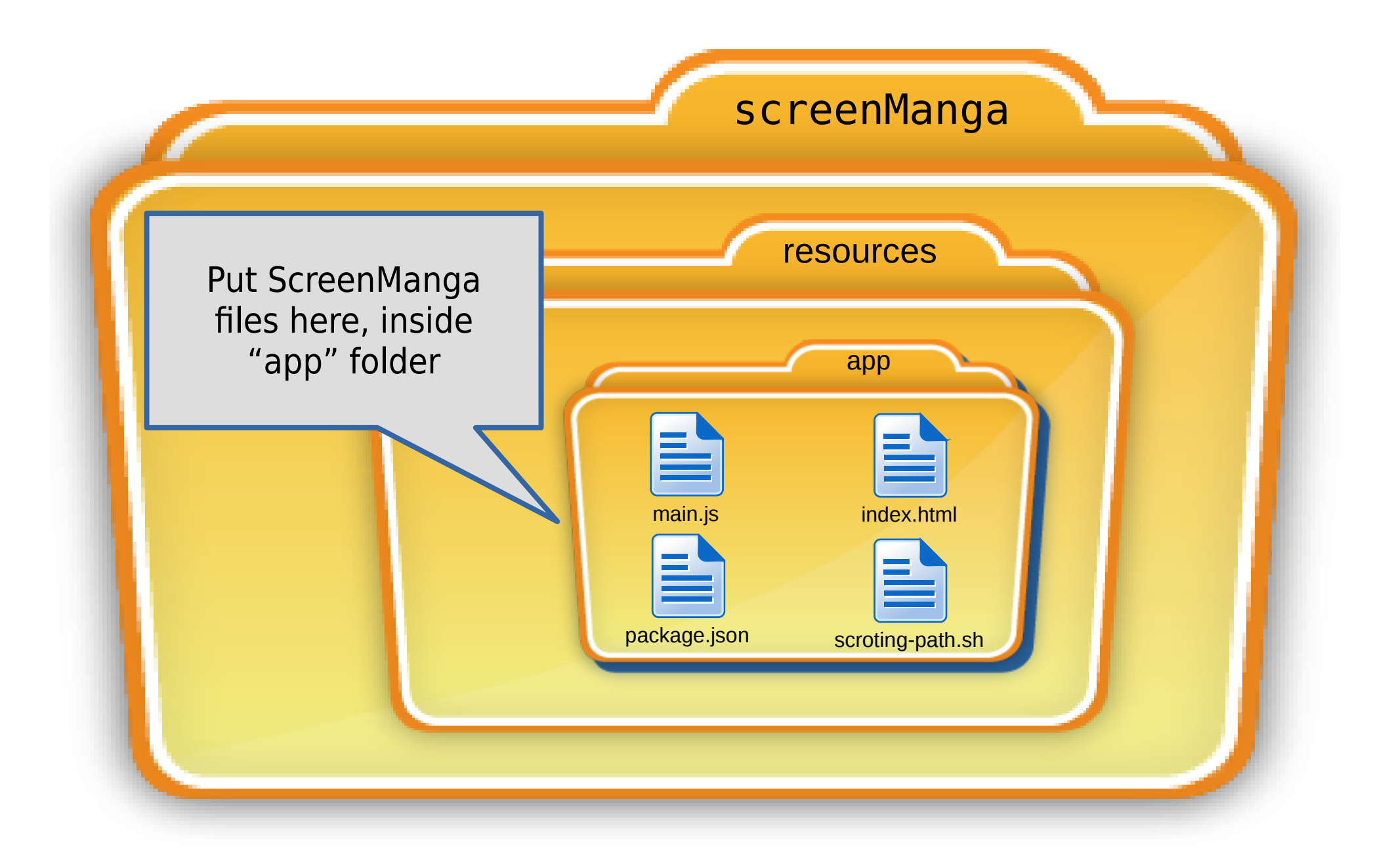
screenManga

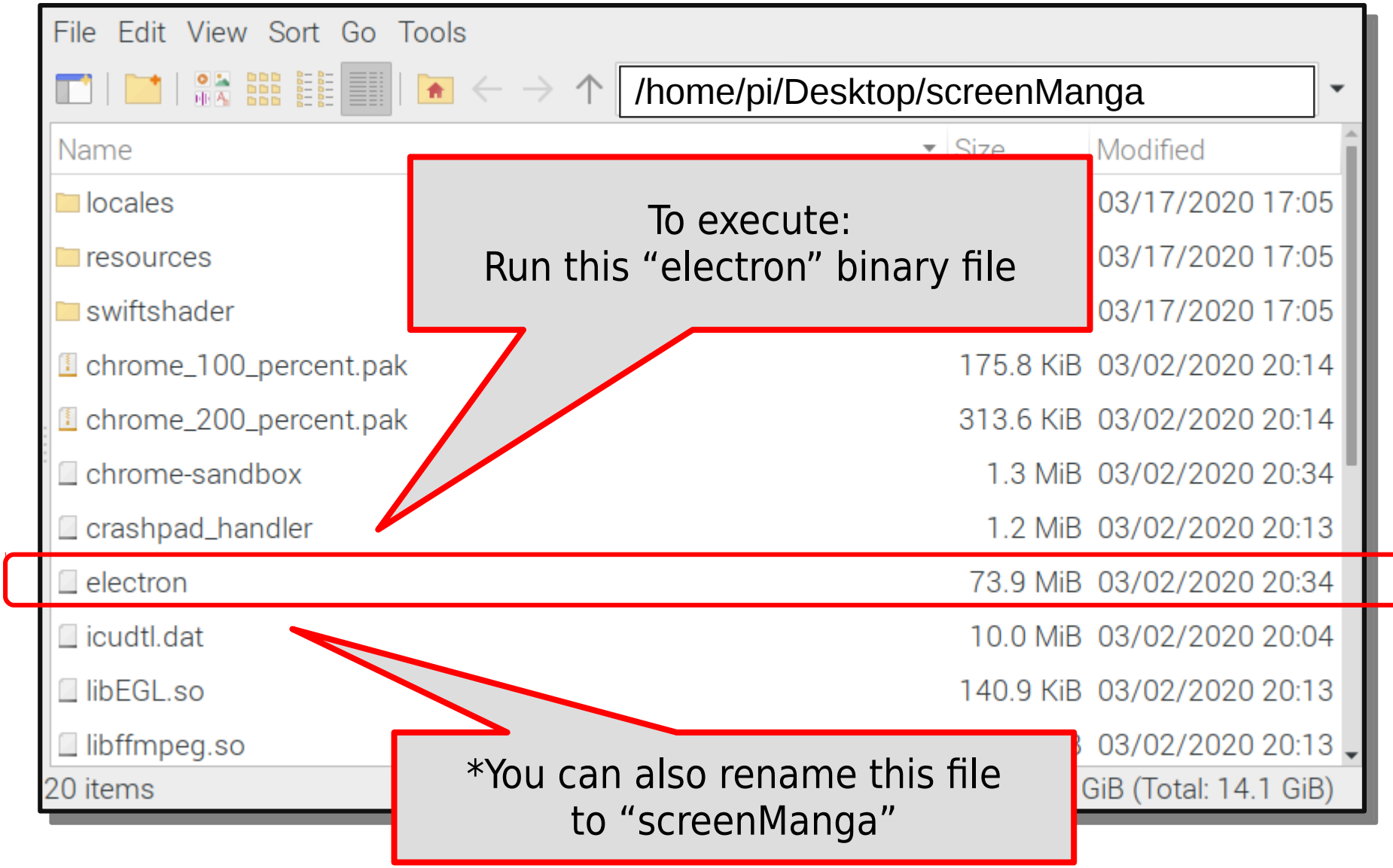
Extract the package here, inside the folder...

electron-v8.0.3-linux-armv7l.zip

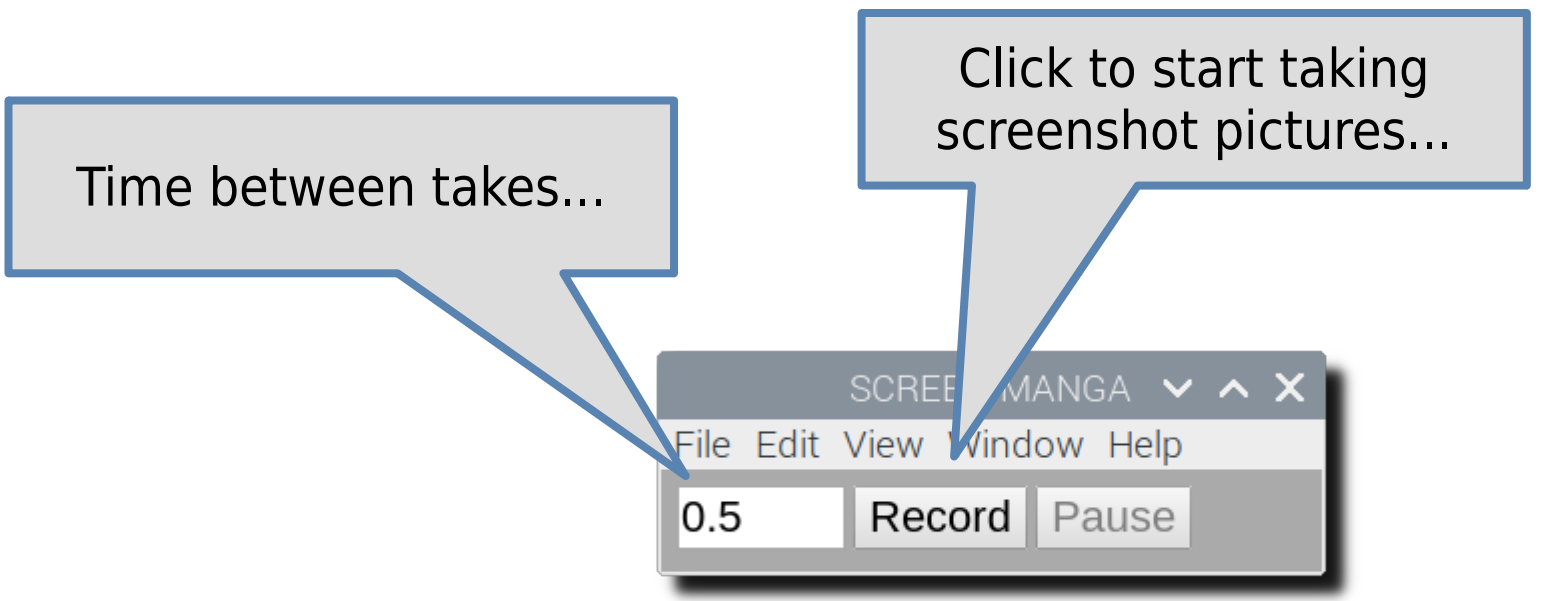
Download the **ScreenManga GUI** script files from Github repository




screenManga
resources
app
Put ScreenManga files here, inside “app” folder
main.js
index.html
package.json
scroting-path.sh


File Edit View Sort Go Tools
/home/pi/Desktop/screenManga
Name
Size
Modified
locales
03/17/2020 17:05
resources
03/17/2020 17:05
swiftshader
03/17/2020 17:05
chrome_100_percent.pak
175.8 KiB 03/02/2020 20:14
chrome_200_percent.pak
313.6 KiB 03/02/2020 20:14
chrome-sandbox
1.3 MiB 03/02/2020 20:34
crashpad_handler
1.2 MiB 03/02/2020 20:13
electron
73.9 MiB 03/02/2020 20:34
icudtl.dat
10.0 MiB 03/02/2020 20:04
libEGL.so
140.9 KiB 03/02/2020 20:13
libffmpeg.so
03/02/2020 20:13
20 items
GiB (Total: 14.1 GiB)
To execute:
Run this “electron” binary file
*You can also rename this file
to “screenManga”

```
#  From inside the screenManga...

./electron

#  Or, if you renamed it...

./screenManga
```



The pictures sequence will be inside the /home/pi/Pictures folder

index.html

```
<html>
<script>
	var screenTaker;
	const { exec } = require('child_process');
	const path = require('path')

	function goOneScroting () {
		exec(path.join(__dirname, '/scroting-path.sh'), (err, stdout, stderr) => {
		  if (err) {
			console.log("Couldn't execute...");
		    return;
		  }
		  console.log(`stdout: ${stdout}`);
		  console.log(`stderr: ${stderr}`);
		});
	}

	function record() {
		var periodOfTakes = document.getElementById("period").value * 1000;
		screenTaker = setInterval(goOneScroting, periodOfTakes);
		document.getElementById("record").disabled = true;
		document.getElementById("pause").disabled = false;
	}

	function pause() {
		clearInterval(screenTaker);
		document.getElementById("record").disabled = false;
		document.getElementById("pause").disabled = true;
	}

</script>

<body style="background-color: #AAAAAA;">
	<input id="period" type="text" value="0.5" size="4" style="font-family: arial; font-size: 22px;" >
	<button id="record" onclick="record();" style="font-family: arial; font-size: 22px;">Record</button>
	<button id="pause" disabled onclick="pause();" style="font-family: arial; font-size: 22px;">Pause</button>
</body>
</html>
```

scroting-path.sh

```
scrot -z -q 60  /home/pi/Pictures/$(date +%M%S-%N).jpg
```

The screenshot images will be save to
/home/pi/Pictures

About Screenshot Filename

You can control the format of the filename,
by changing the string that generates it...

Inside the file "**scroting.sh**" (CLI version) or
inside the file "**scroting-path.sh**" (GUI version)

```
#  For Day, Hour, Minute, Second
$(date +%d%H%M%S-%N).jpg
18024300-445515032.jpg

#  For Year, Month, Day, Hour, Minute, Second
$(date +%Y%m%d%H%M%S-%N).jpg
20200318024210-942038853.jpg
```

*The original format only uses MINUTES + SECONDS, so, pictures generated in different days at same hours will end up “mixing” together in filename order… to prevent that you can use more prefixes like the ones suggested here...