RYERSON UNIVERSITY

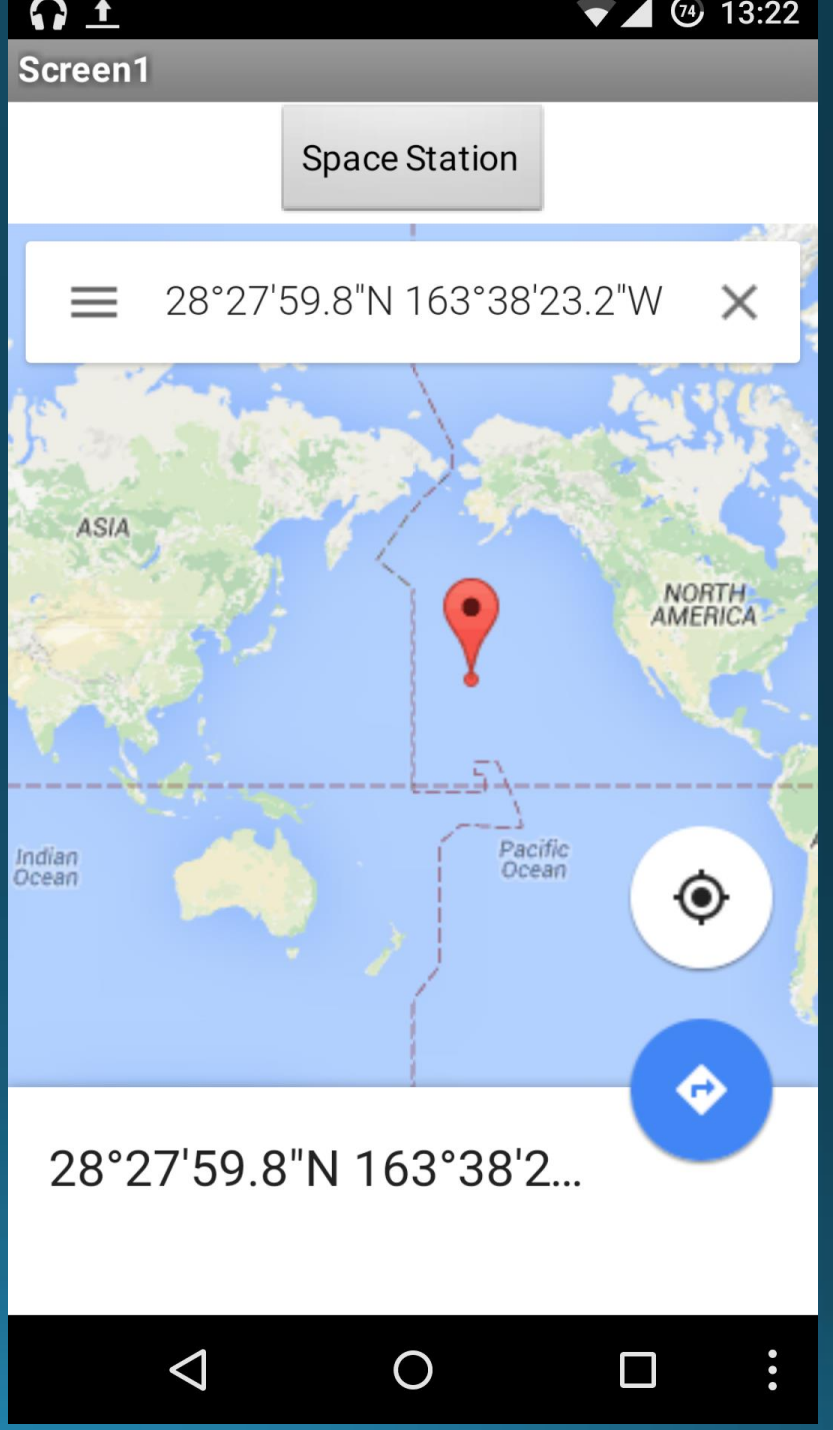
# Maps & Web APIs

# Today's Plan: WWW

- For our last lecture, we'll be powering our apps with the internet.
- We'll see how to use web pages in your app, and how to pass information to websites
- Then we'll learn how websites communicate.

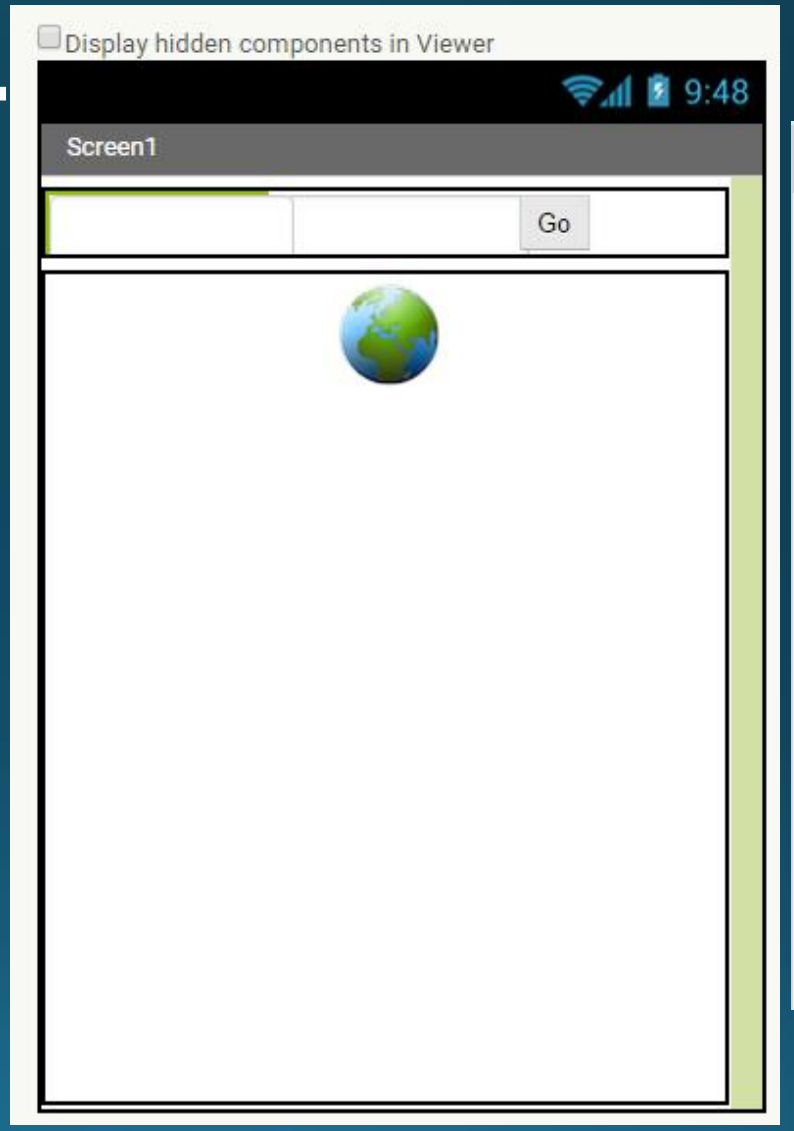
# Today's app: Orbiter

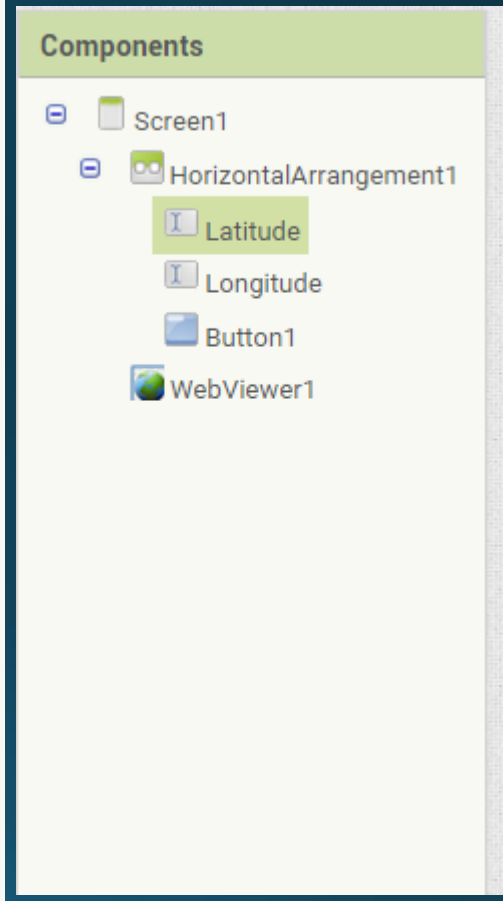
- Make a new project.
- Later on, we're going to track things in **SPACE** (!).
- For now, let's keep track of ourselves



# Let's jump right in.

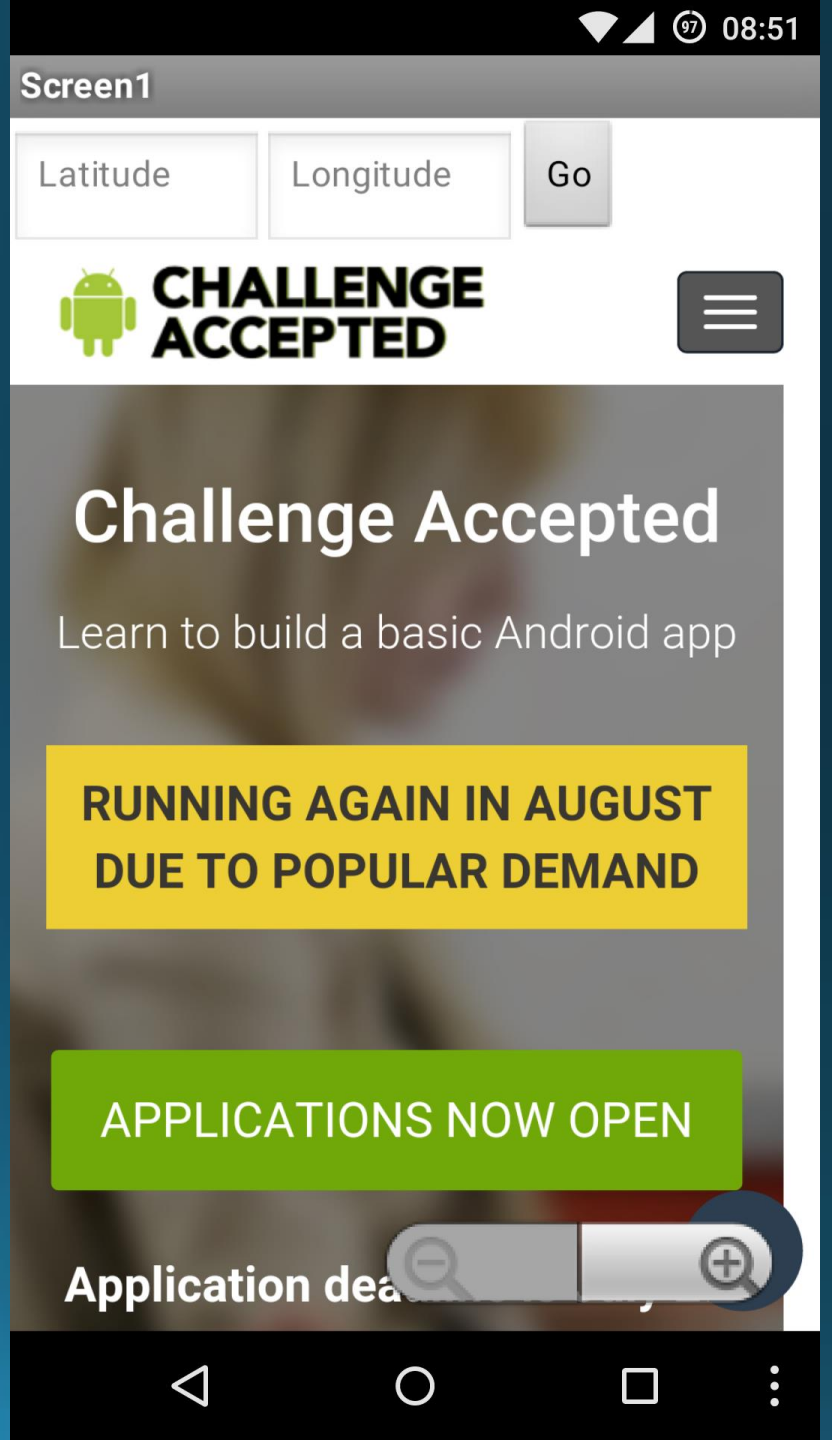
- Horizontal Layout
  - TextBox → Latitude
    - Set Hint property to Latitude
  - TextBox → Longitude
    - Set Hint property to Longitude
  - Button
    - Set Text property to Go
- WebViewer (under User Interface)
  - Set HomeURL property to: http://mydmz.ryerson.ca/learntocode/
  - If you're on an emulator, set it to
  - http://www.google.com/



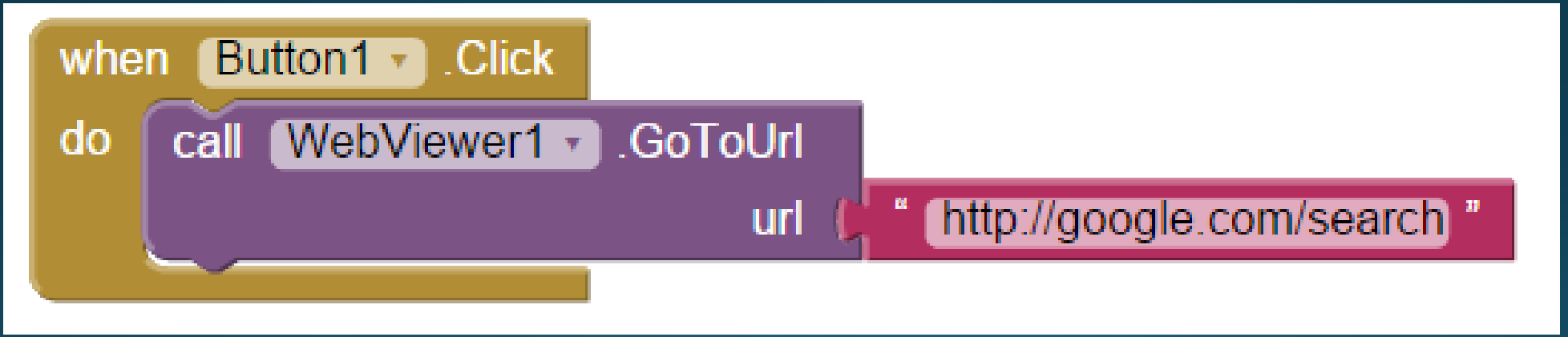
# What is the Webviewer?

- Test your app!
  - (May take some time on emulators)
- The WebViewer Component allows us to embed a web page into our app
- Today we will be using it to make a coordinate navigator



# Let's try some blocks

- Let's add the following blocks to our app:

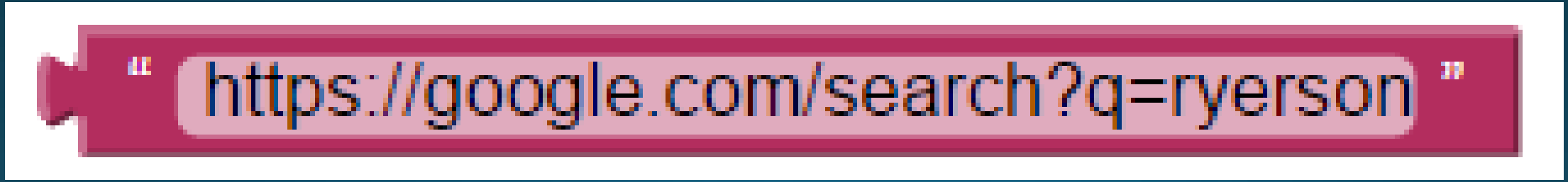



- Test your App, What happens?

# Passing information to websites:

- Try replacing the link inside the textbox from:


To



# Web parameters

- When loading a web page, we can pass information in the url, they get passed as key/value pairs.

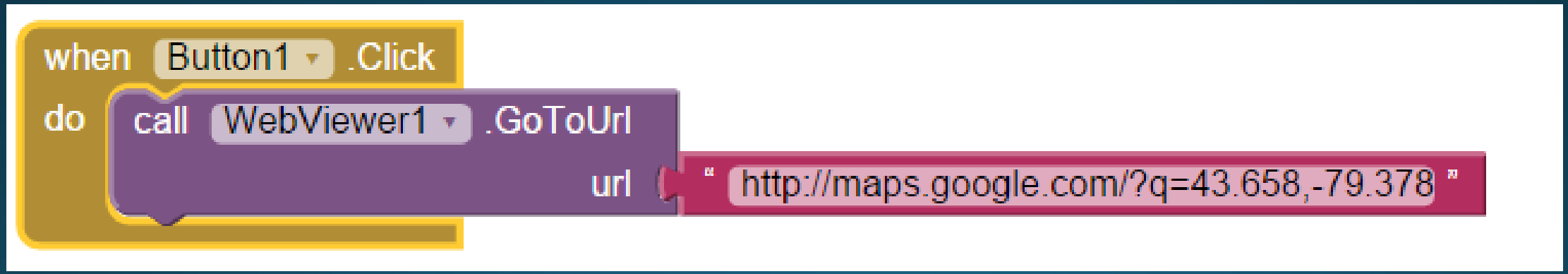
- The standard format is:
  - http://webaddress.domain/page?key=value&other_key=other_value

- Open your browser (on your computer) and do an Image search on Google.
  - What parameters do you see on your address bar?
    - Google keeps track of a lot of stuff!
  - What do you think tbm=isch does?
    - Try adding it to your block (you'll need to add an &)

**tbm** indicates the content type searched for
**tbm=isch** limits the search to **i**mage **s**ear**ch.**

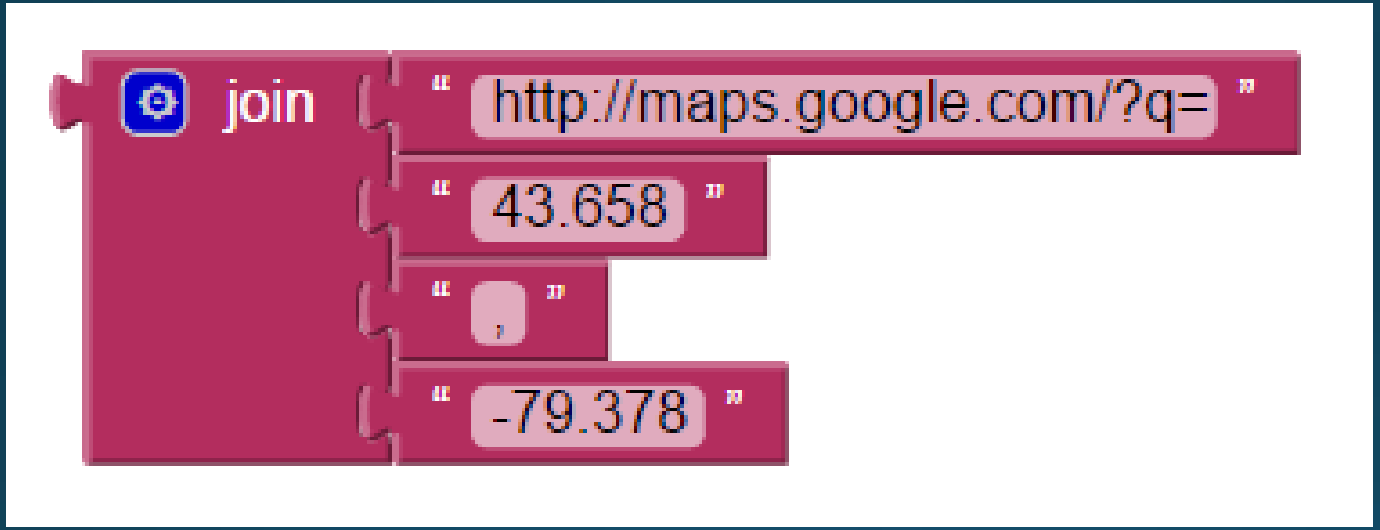
# If you're lost, use a map!

- Let's get back to our app:
- Let's set the URL in the blocks to
- http://maps.google.com/?q=43.658,-79.378



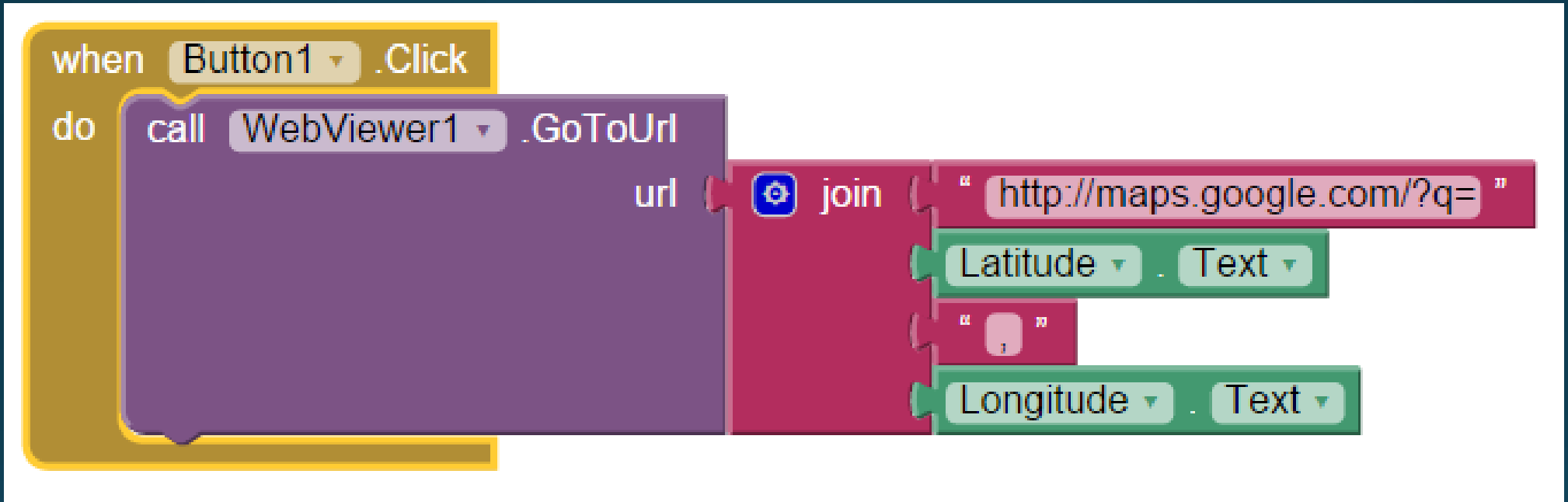

Where does your app take you?

# Ok, so let's split this up:




- What does this code do?
- Why do you think we're doing this?
- Can you guess what the next slide will ask us to do?

# Using user input…




Test the app!

Here are some nice coordinates:

19.422468, -99.129193

35.728629, 51.386211

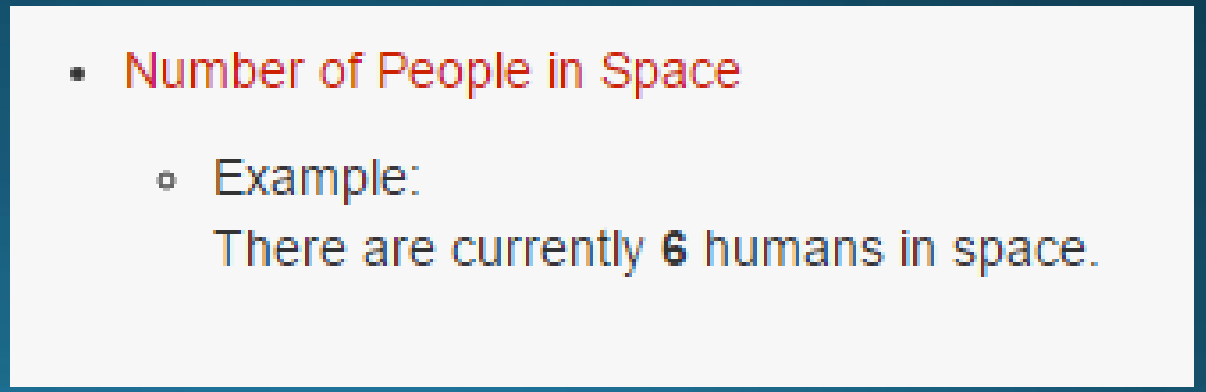
44.046395, -79.459318

Can you think of other uses of the WebViewer?

# End of Locator App

# And now for something completely different

- Let's space out for a bit:
- Have you ever wondered how many people there are in space?
- Open this link on the browser: http://open-notify.org/
  - Check this out:

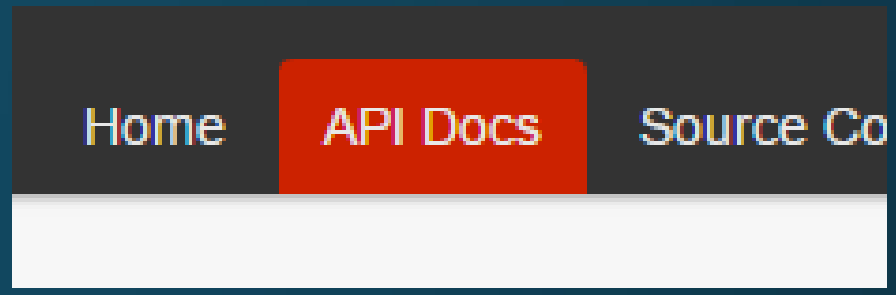


# Our info may be out of date!

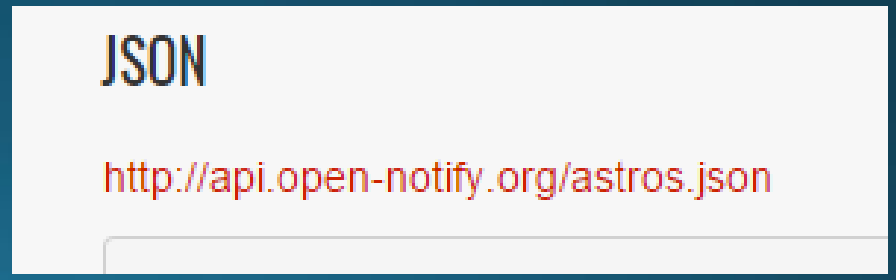
- On the previous slide, the image said that there were 6 people in space
- But these slides could have been made a year ago…
- Let's have our app communicate with NASA and always give us fresh data on the amount of people in space.

# Let's look at the data

- If you're still on the Open Notify website:
  - Click API Docs on the top right
  - Click People In Space on the left menu
  - Click on the first JSON link
- Otherwise, go to:
  - http://api.open-notify.org/astros.json







# Hello, my name is JSON

- Information sent from WebAPIs is usually in this format, called JSON (JavaScript Object Notation)
- JSON data is stored as a list of Key/Value pairs (like a database!)
- The special thing is: the value in each key could itself be a JSON object (another KeyValue list)

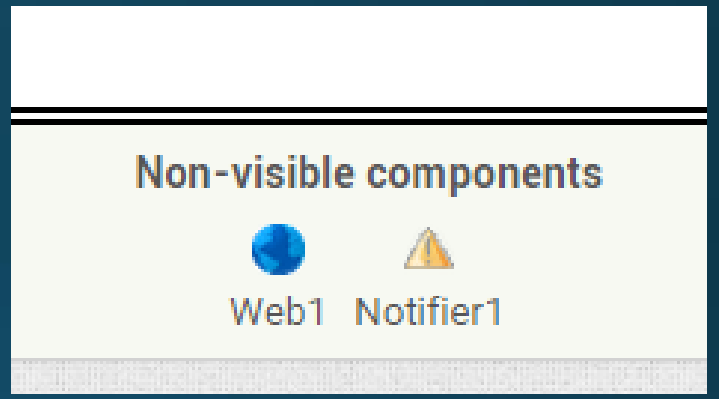
```
{
  "message": "success",
  "number": 6,
  "people": [
    {
      "craft": "ISS",
      "name": "Gennady Padalka"
    },
    {
      "craft": "ISS",
      "name": "Mikhail Kornienko"
    },
    {
      "craft": "ISS",
```

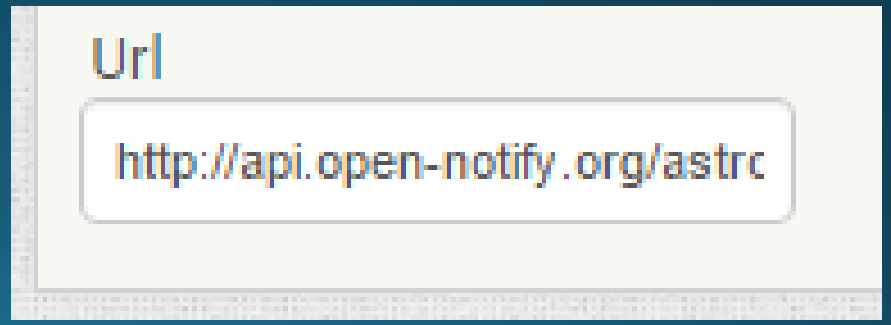
# Can you think of cool things to do with this data?

Maybe using the search webviewer?

# For now, let's just tell the user how many people there are in space

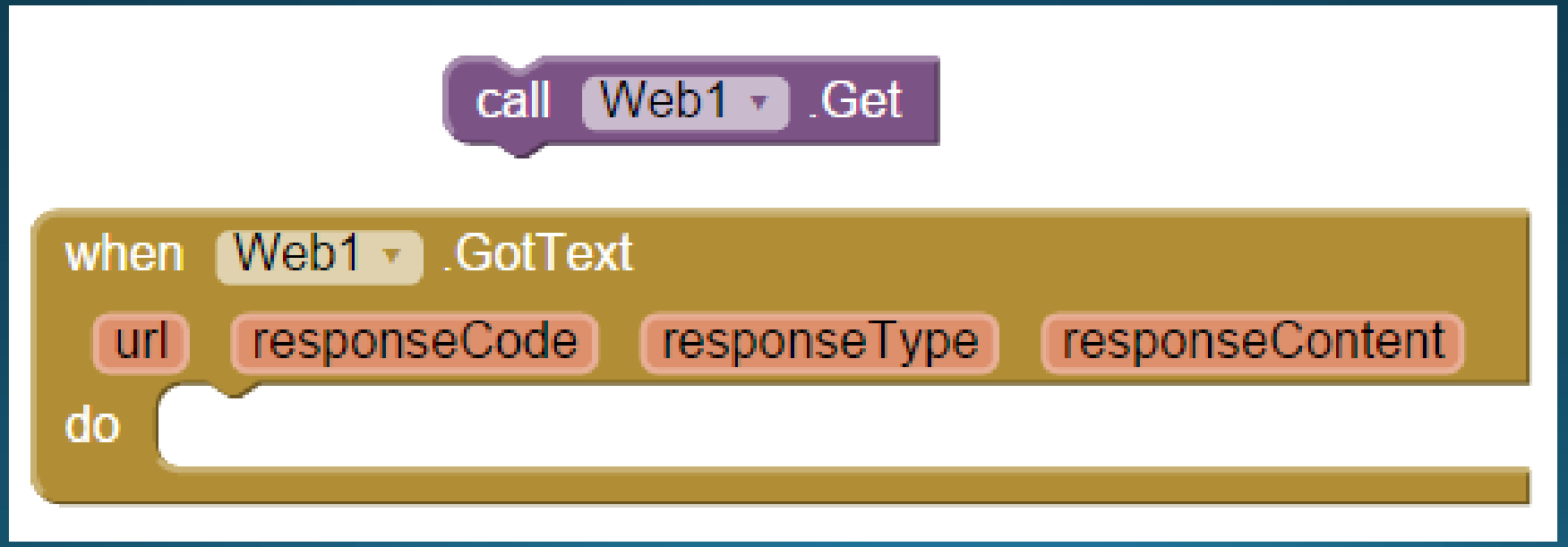
- Drag in a notifier component
  - We've used these before!
- Drag in a Web component from the Connectivity section of the palette
- Set the URL property to the API we saw earlier:
  - http://api.open-notify.org/astros.json



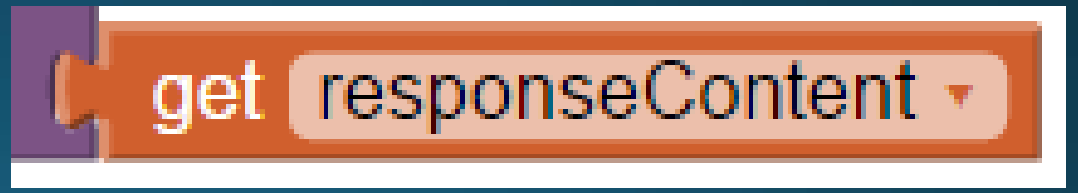
# Callbacks!

- Getting astronaut data from NASA is a bit like ordering a pizza…
- You call to ask for it, and then, when it arrives, you eat it.
- This is why the Get block doesn't have a tab on the left, it doesn't have any output: it's simply a call.

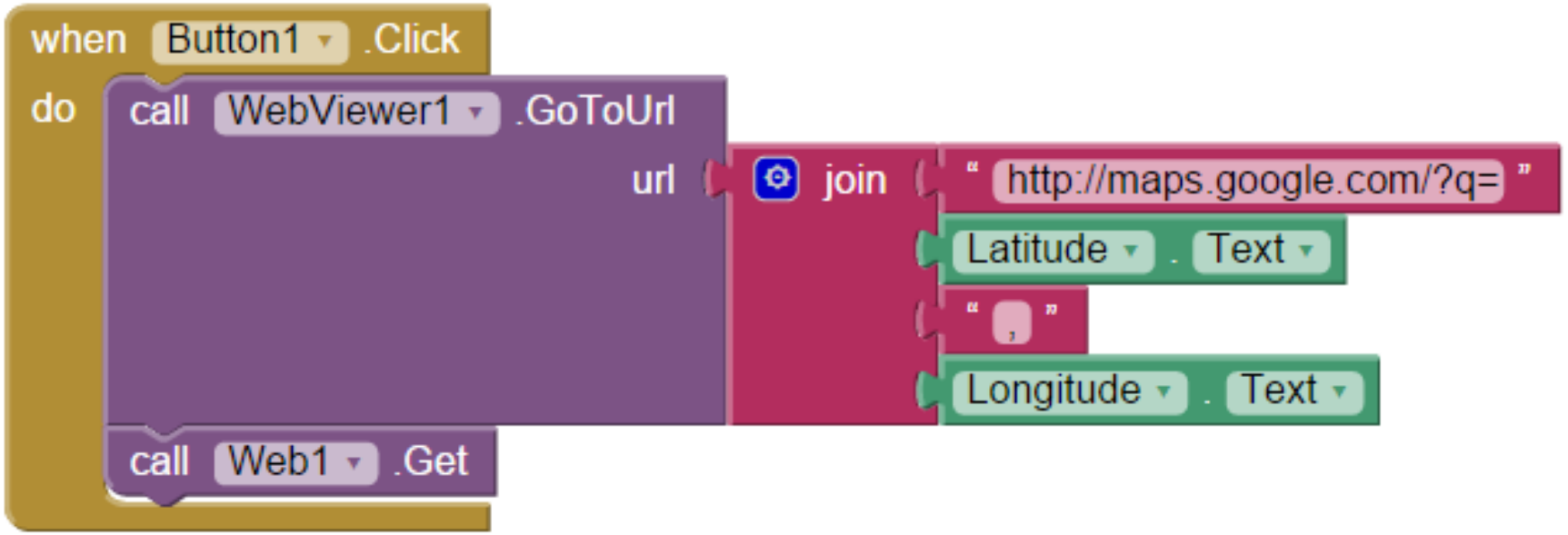


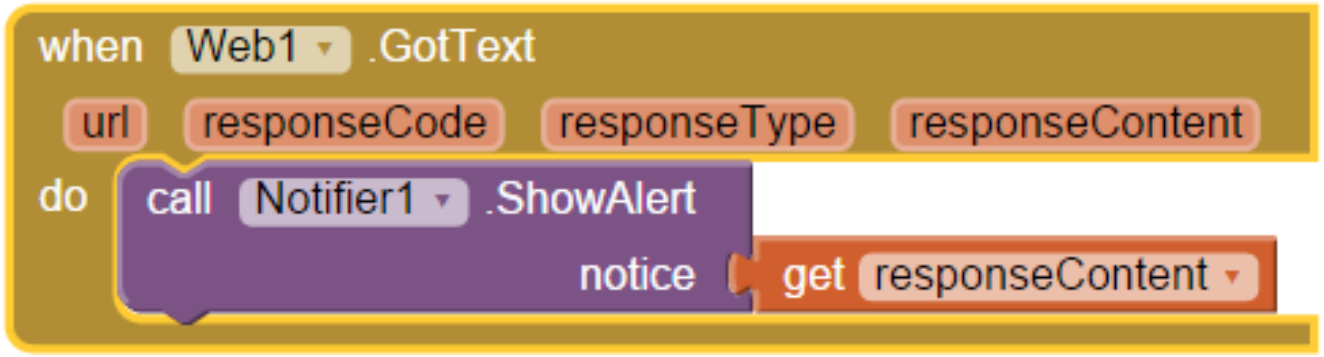
First, we ask for the data


Then, (when it arrives) we use the data

# Add the space data to your code:




Test your app:

What happens?

# Picking out data:

- Just as how we have to slice a pizza, we have to select only the data that is important to us.

- Let's take a short break

- When we come back, we'll pick out the number of astronauts to show to the user.

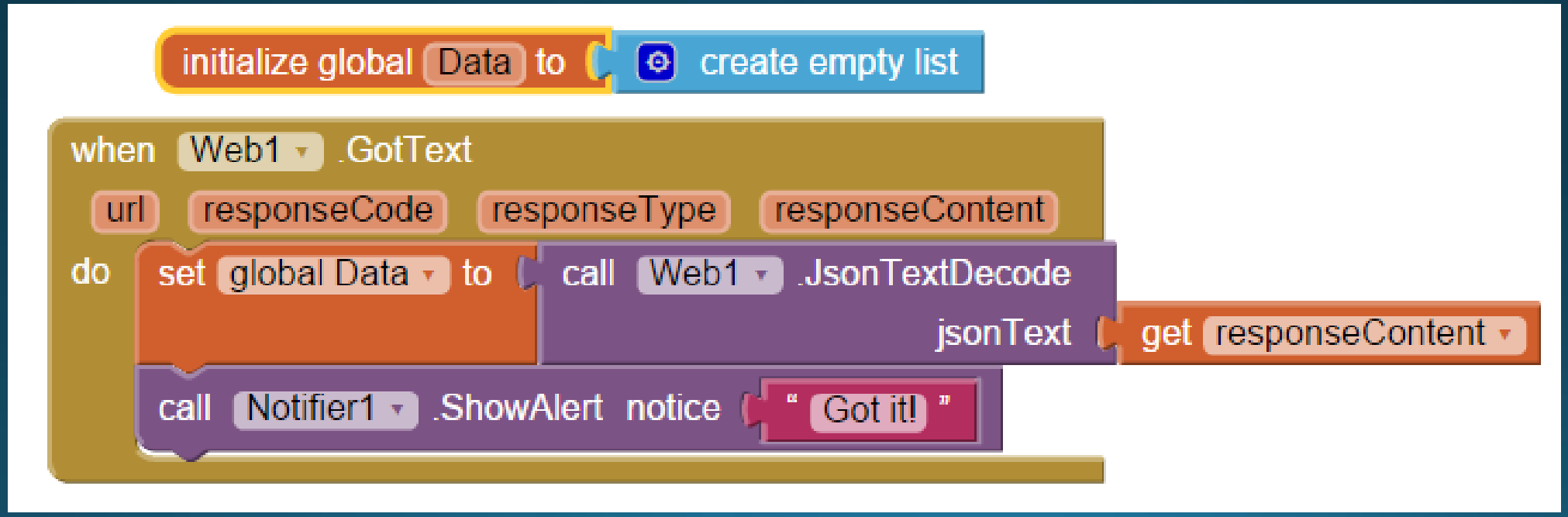
# Translating JSON

- JSON is a data language, which means it has a lot of weird symbols that we don't really want to show to the users.

- Also, to the computer, it's still just a bunch of text, let's convert it into a useful list.

# Blocks:

- Use the Web1.JsonTextDecode block to convert the JSON into a nice set of lists
  - Store the result in a variable called data



# Another look at the data

- Go to http://api.open-notify.org/astros.json on your browser again:
  - How does that data compare to this table:

| Key | Value | | |
|---|---|---|---|
| message | success | | |
| number | 6 | | |
| people | -> | craft | ISS |
| | | name | Gennady Padalka |
| | -> | craft | ISS |
| | | name | Mikhail Kornienko |
| | ... | ... | ... |

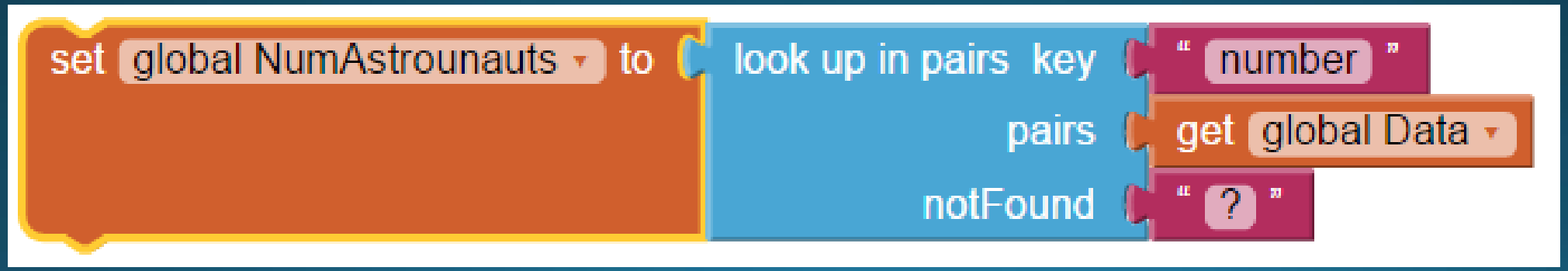
# Notes:

- The value for "people" is a list of smaller json objects
- You can tell it's a list because of the square brackets.
- Notice how each small json object has curly braces
  - Notice how the whole data has curly braces
- What key do we have to use to get the value "6"?

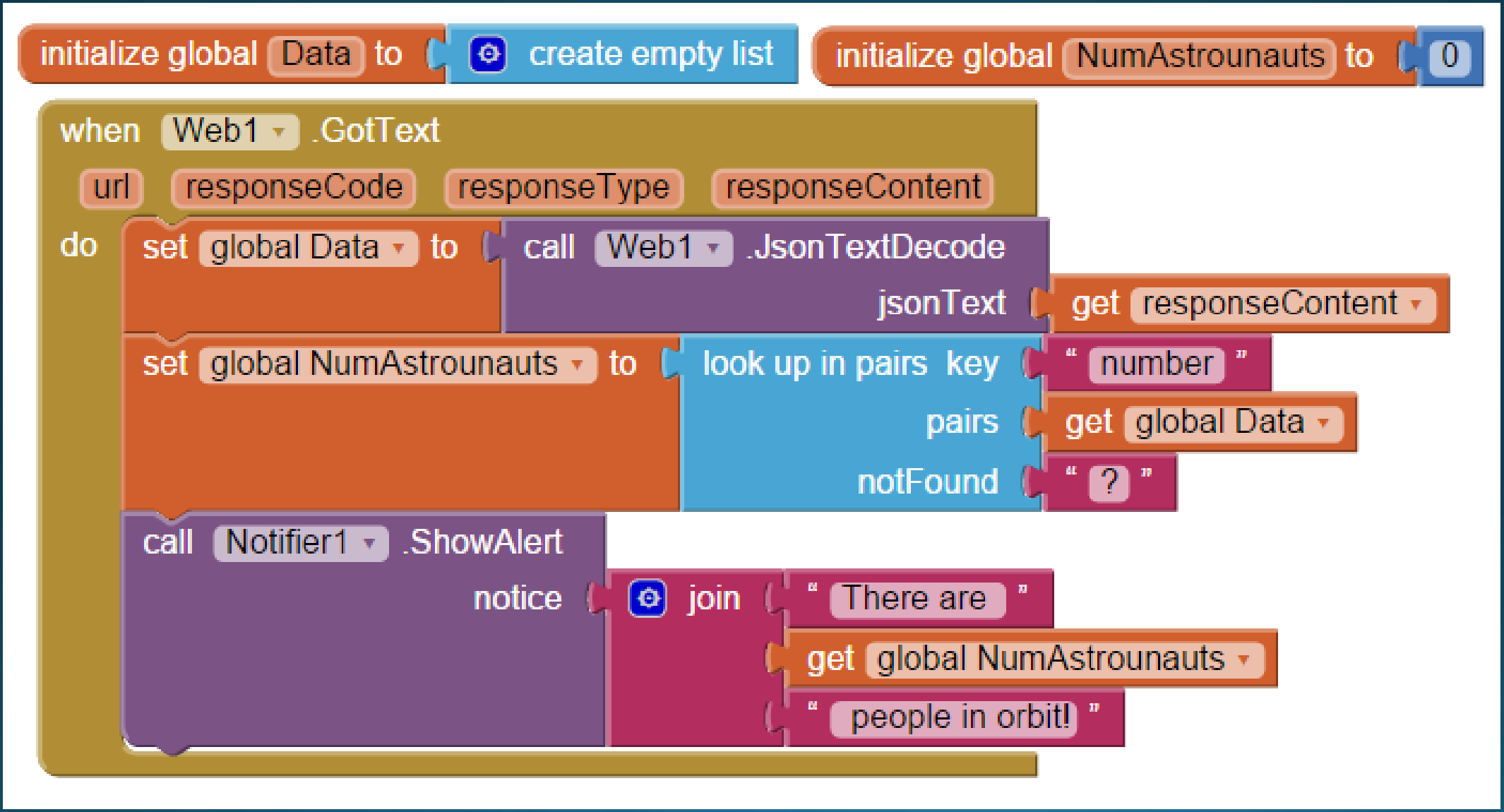
# How to get a value?

- The lists menu has a block that is made specifically for this:


- Use the lookup block to tell the user how many astronauts are in space



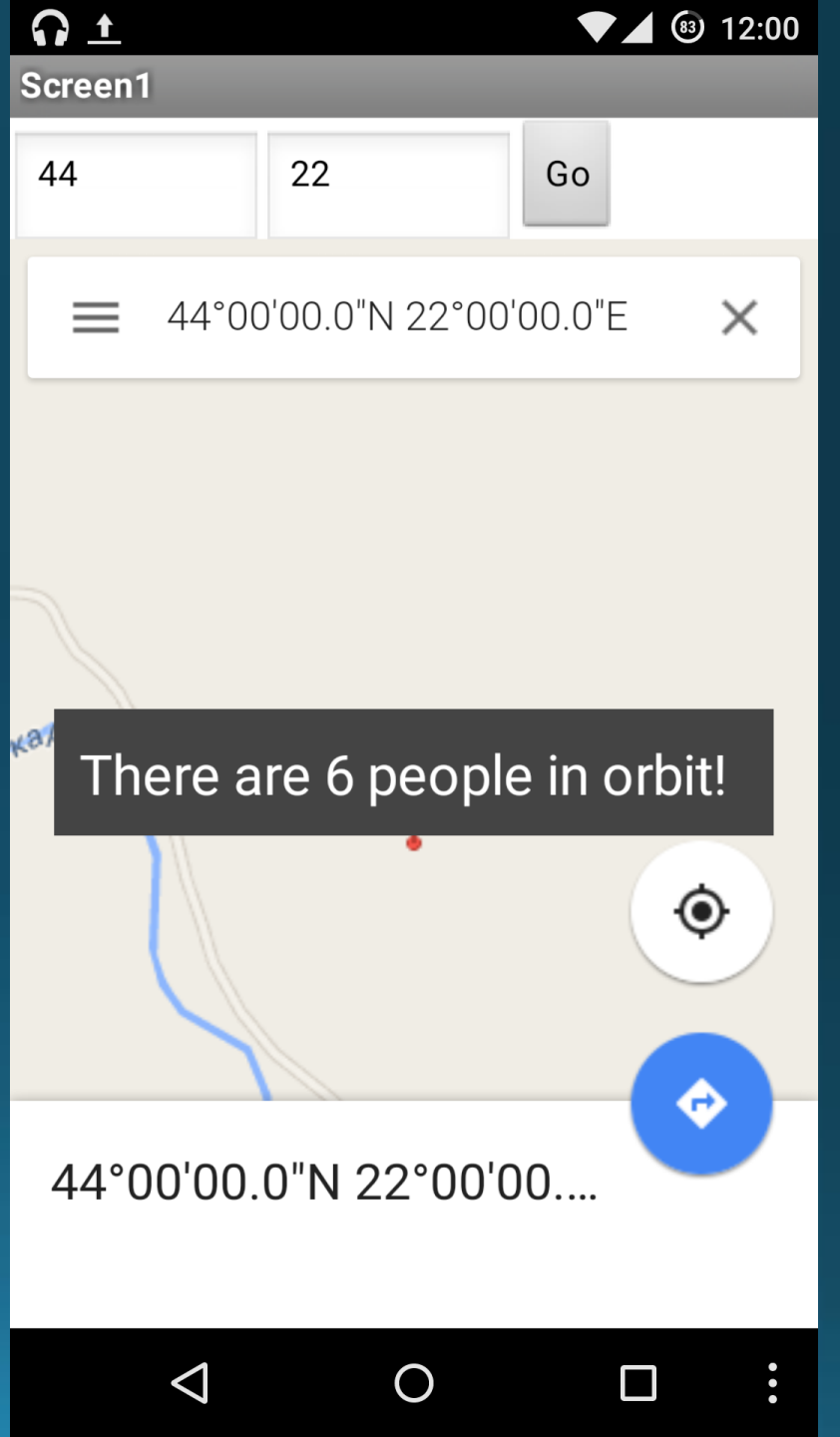
# Web1.GotText Event:



# Test Your App!

- So far, we're using information that any space enthusiast knows (Number of orbiting people)

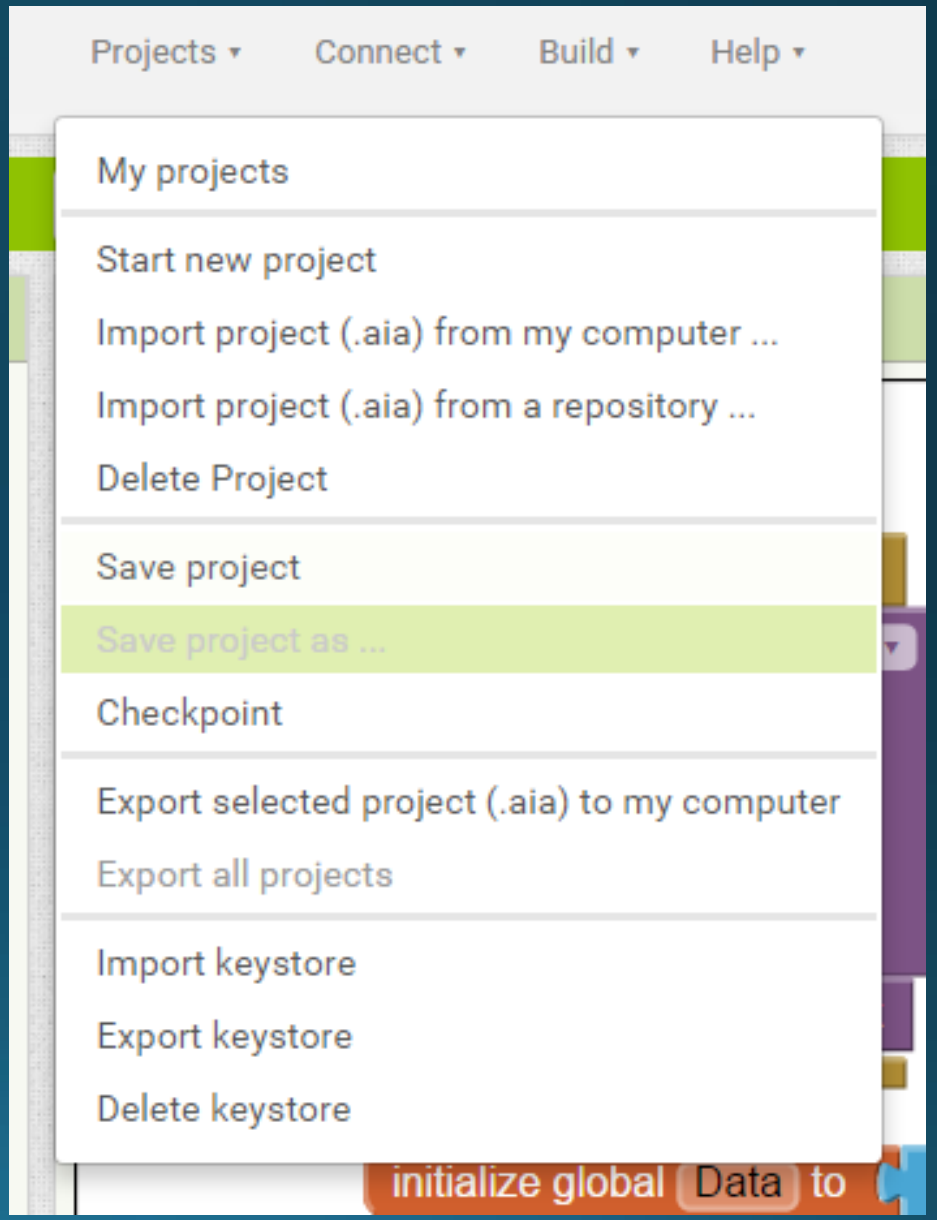
- Let's try putting everything together to communicate with space in real time!



# Overhaul

- We're going to modify the app, so save a copy of the project if like it the way it is now

- To save a copy:
  - Projects menu
  - Save Project As….

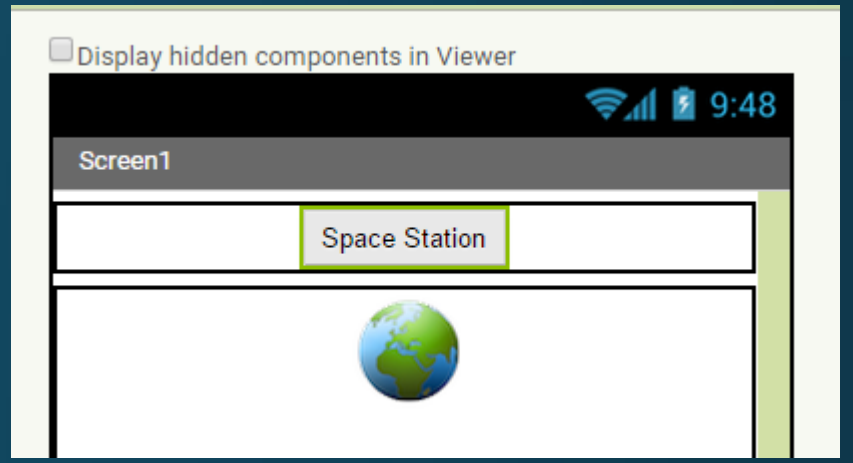


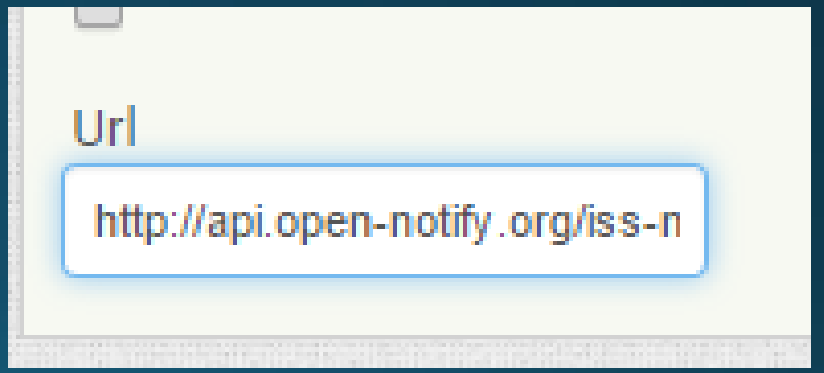
# Put it together

- Let's make an app that keeps track of the location of the international space station
- We'll use an API that accesses live data from the ISS with regards to its location above the earth
- We'll show the location for the user on our map!

# Out with the old…

- In the designer:
  - Remove the textboxes
  - Center the Button and change its Text to "Space Station"
  - Remove the notifier



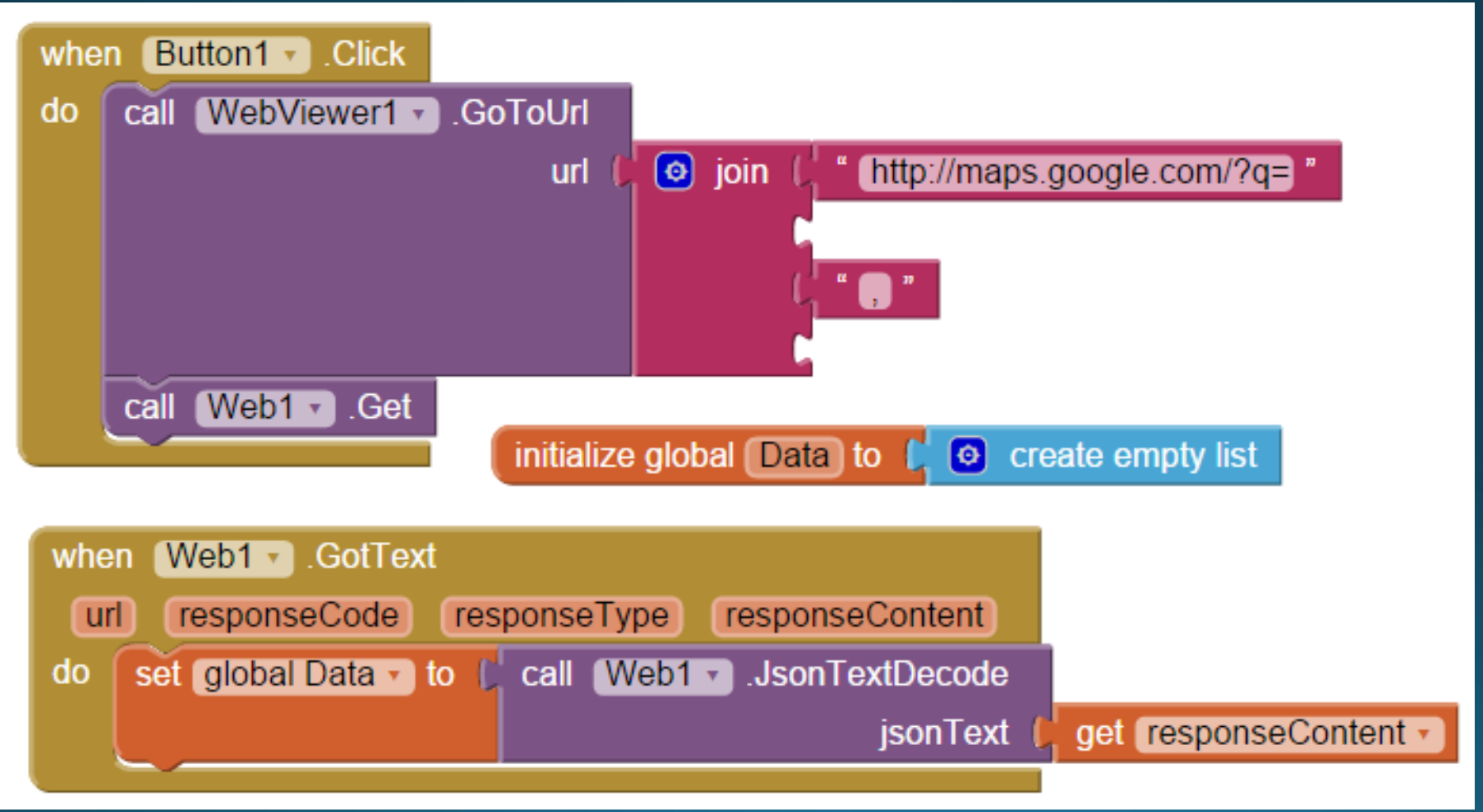

- Change the URL in the Web1 Component to:

```
http://api.open-notify.org/iss-now.json
```

# Blocks:

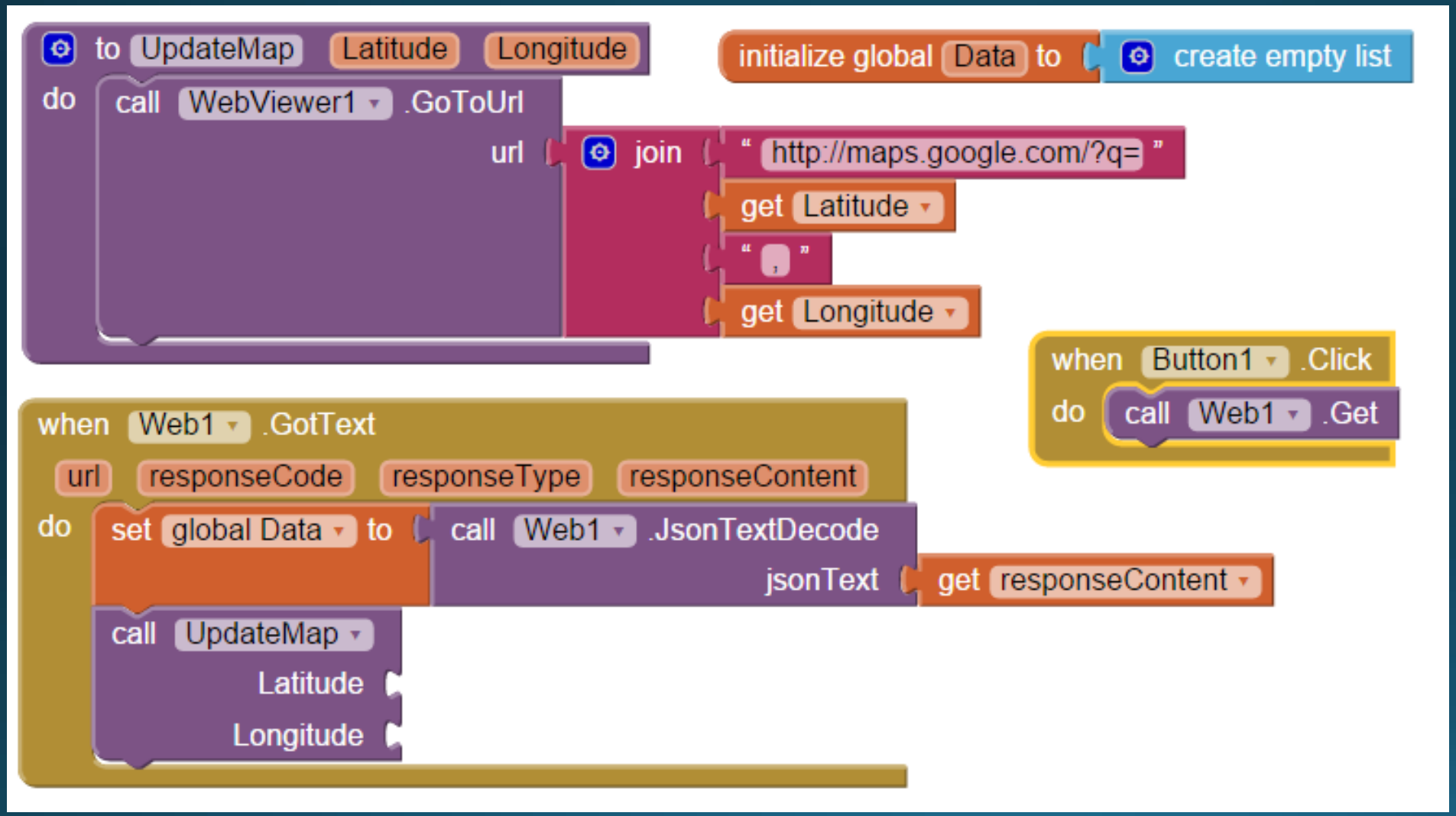
- If you look at our blocks, a few of them have disappeared
  - They were using the textboxes or the notifier
- Since we're no longer going to count the number of astronauts, remove the blocks that use NumAstronauts
- Remove:

# What are we left with?



# Let's make the map update into a function

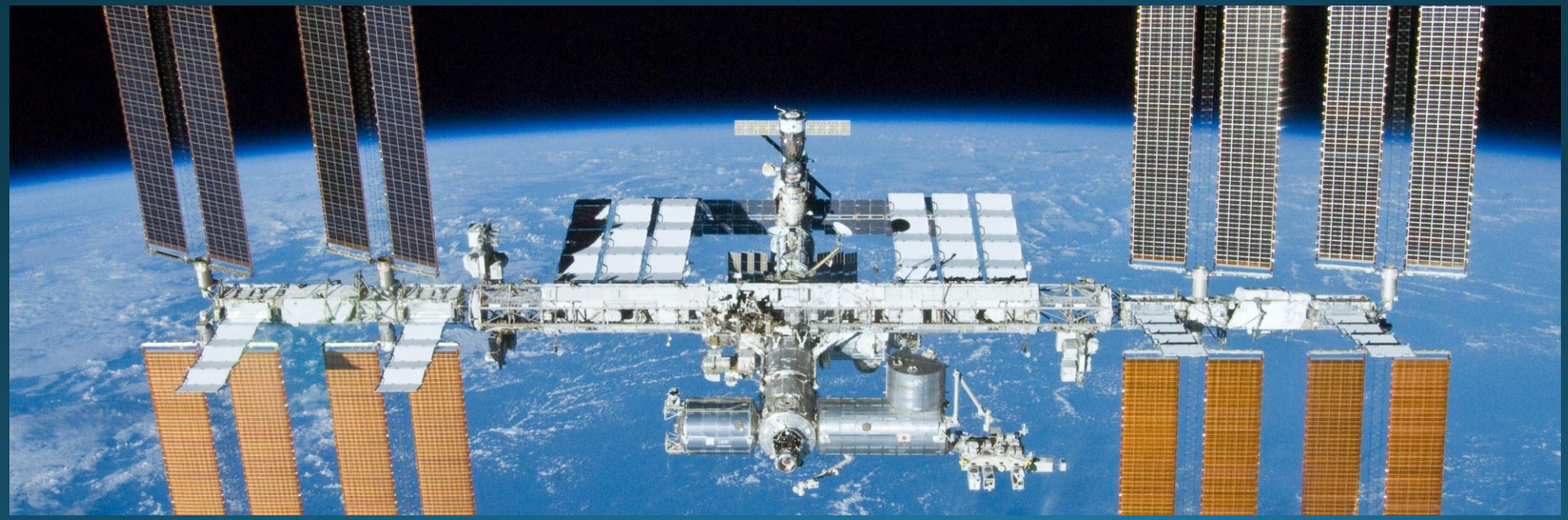
- Abstract the WebViewer1.GoToUrl call into a function (procedure) called "UpdateMap"
- Give the function two parameters:
  - Latitude
  - Lonitude
- Plug in the parameters into the GoToUrl call
- We can't update the map in when the button is clicked anymore since we only know where we're going once the data comes back.




Can you see where there are missing blocks?

# What's next?

- Look at the data in the new API we're using (open in your browser):
  - `http://api.open-notify.org/iss-now.json`
- Do you see anything that might be useful for our missing blocks?
  - Is this the same as the last example?



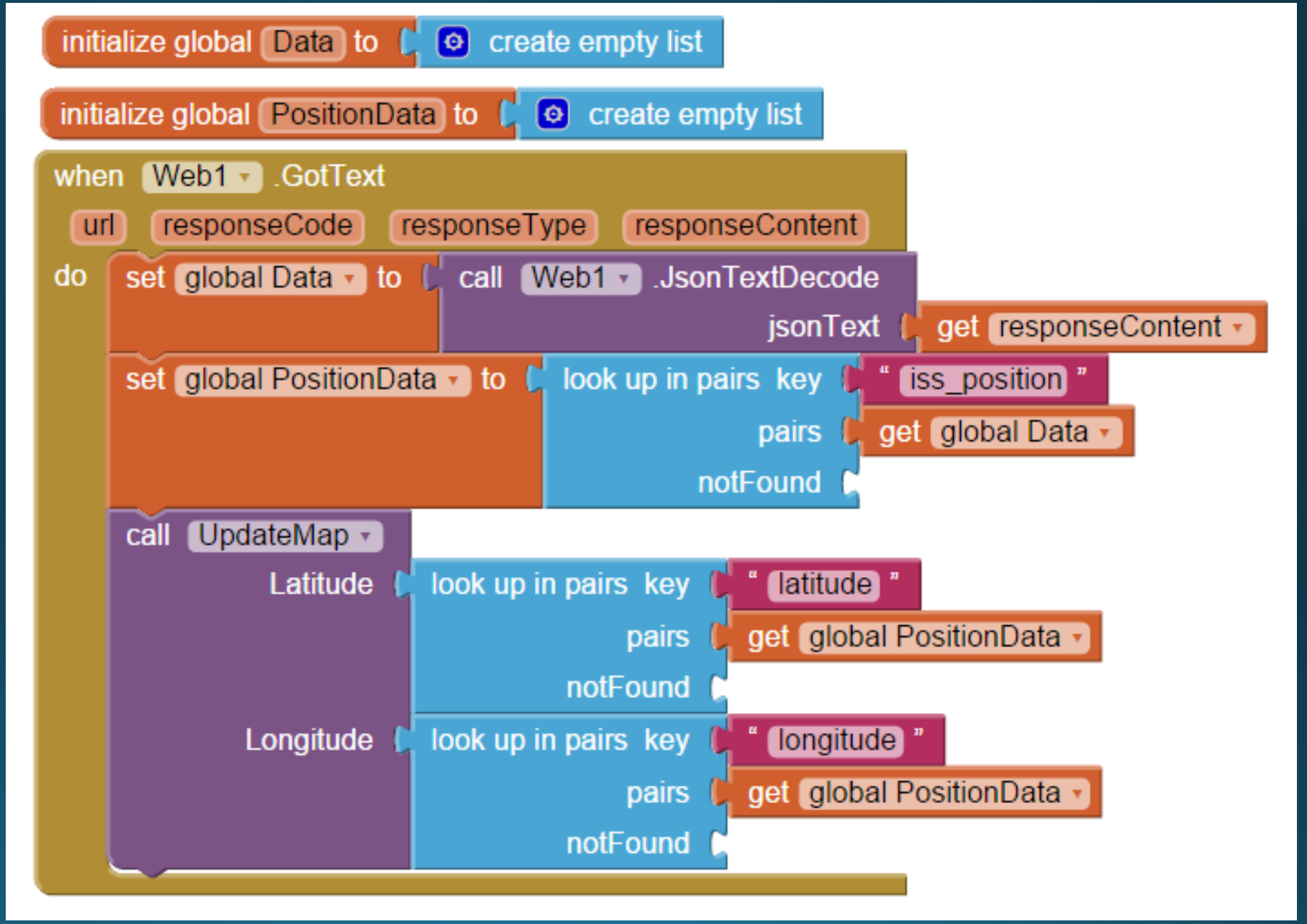
# Lists of lists

- Problem: When we look up the value of iss_position from the list of key/value pairs, we get another list of key/values!
- Solution: do another look up!
- Note: This is a simple example. In most popular APIs, there are lists of lists of lists…..

| Key | Value |
|---|---|
| iss_position | -> |
| message | success |
| timestamp | 1438013469 |

| | |
|---|---|
| latitude | -33.6702774700651 |
| longitude | 22.0711415858709 05 |

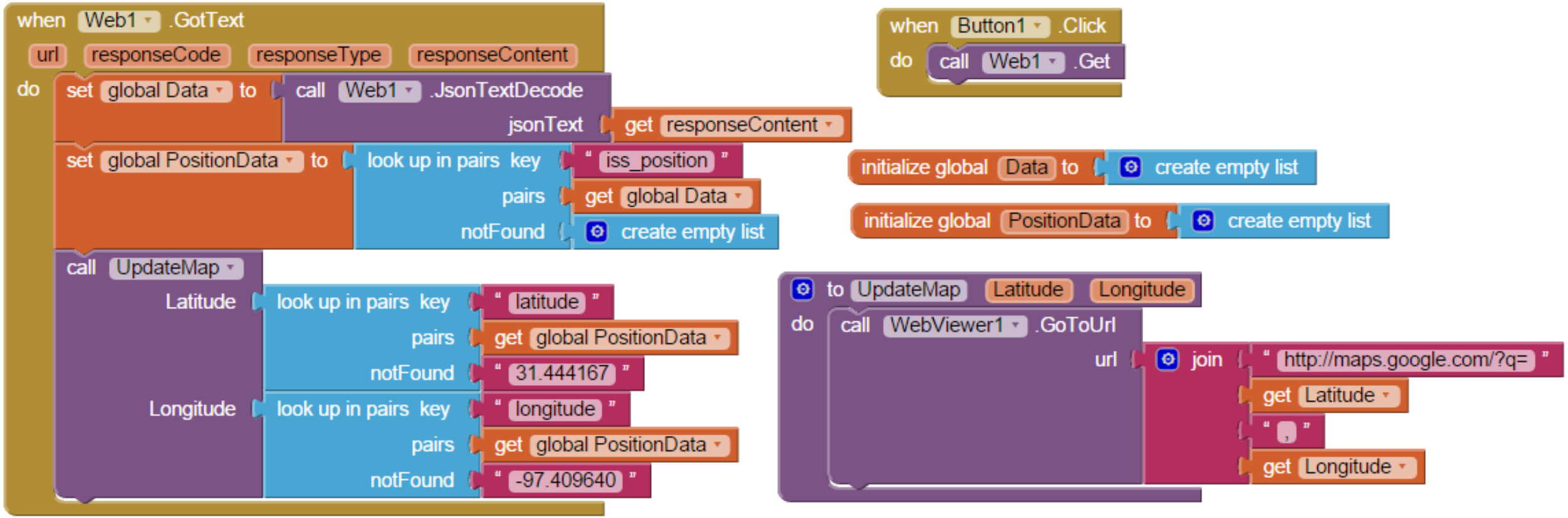
# The code:




Optional:
Can you think of some good default values (notFound) to show if the API doesn't work?

Maybe NASA HQ?

# The whole thing:



# Test your App!

If you see a lot of blue, you might be in the middle of the ocean. Zoom out!

# Thank you very much for making these apps.

- The staff of Challenge Accepted have put a lot of effort into these apps and their accompanying slides. We're sure you've put as much effort into following along.
- Hopefully you've enjoyed the result of these workshops as much as we enjoyed preparing them
- Please let us know in about the apps that you enjoyed making the most and those that you didn't
- We look forward to seeing what you'll make in the next session and there after