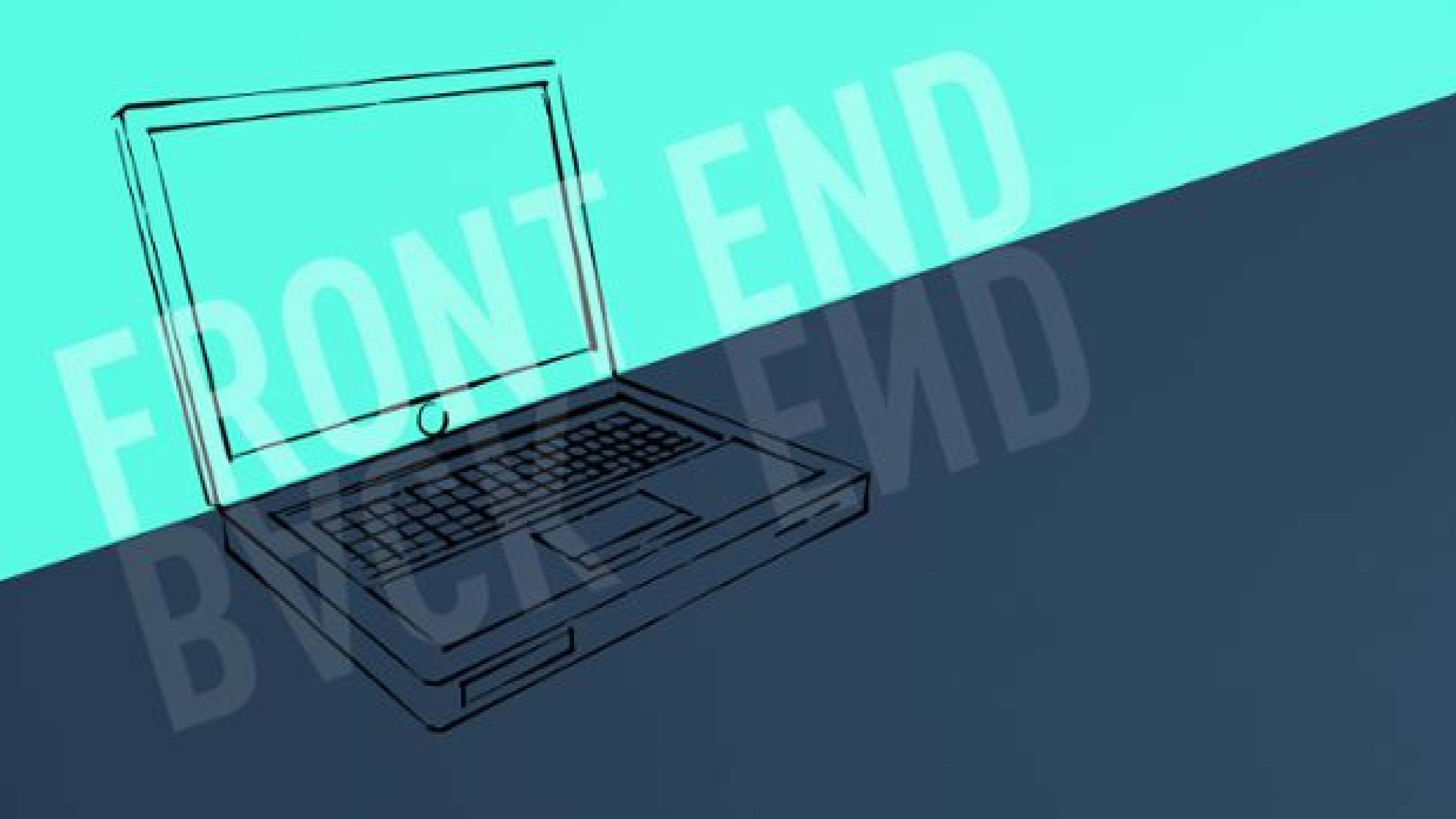


FRONT END
BACK END

Lesson Objectives

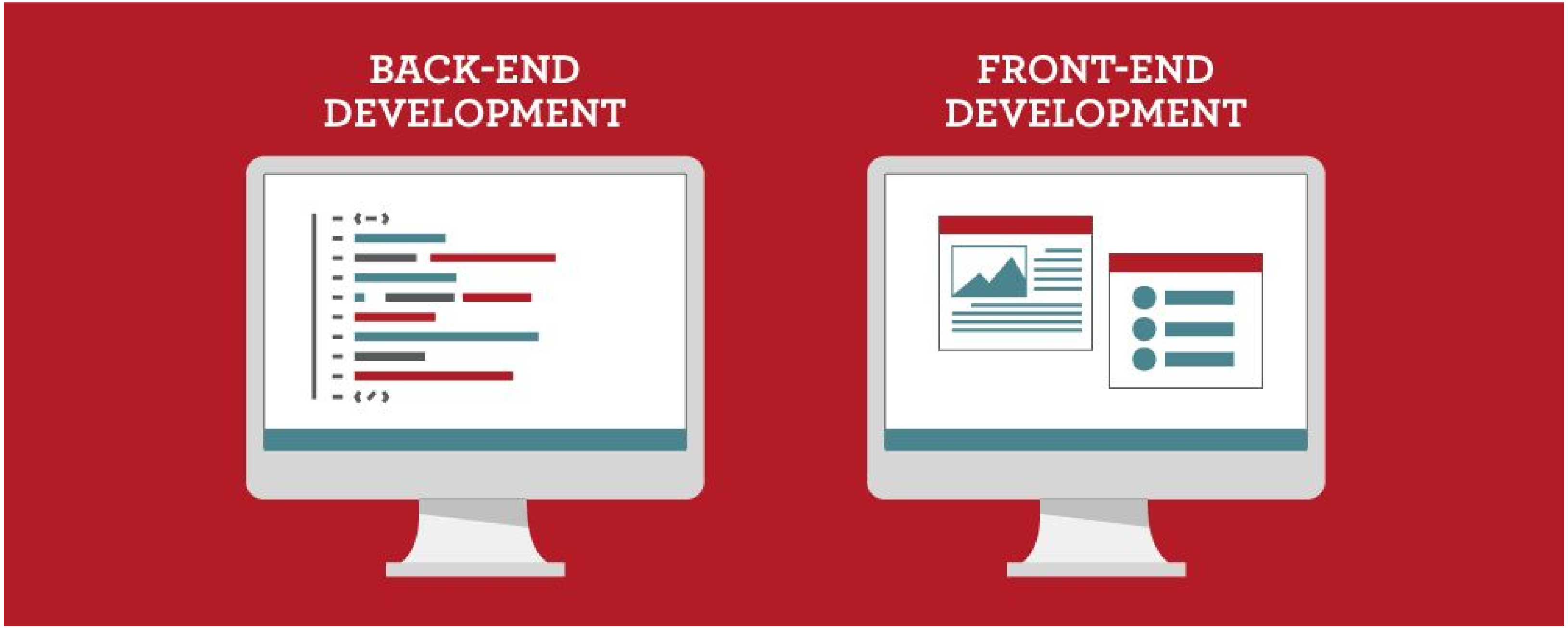
1. Students will be able to differentiate from the Client Side "**Front End**" and Server Side "**Back End**".
2. Students Students will be able to differentiate from a **Server** and **REST API.**
3. Students will be able to differentiate between **GET** and a **POST** requests.
4. Students will apply what they have learned by integrating **NodeRed**, **Little Bits, and IFTTT**.

Cybersecurity First Principles

1. **Data Hiding**: Data hiding is the technique that does not allow certain aspects of an object to be observed or accessed. Data and information hiding keeps the programmer from having complete access to data structures. It allows access to only what is necessary.
2. **Modularization**: Separating unrelated components decreases the chance that modifying one component will adversely impact the other. It will also make it easier to reason about the individual components because there is less to look at when separated.

VOCABULARY!

Front End versus Back End



What is an API?

API stands for **Application Programming Interface**

API’s enable software to interact with other software through exposed functionality, otherwise known as …

REst API

Stands for REpresentational State Transfer

Allows for one piece of software to talk to another

REst API is a “trendy” way to host a web app. It’s a type of server that stores information.

Uses less bandwidth than a typical server, making it more suitable for IoT devices.

GET vs POST

GET - Request data from a specified resource.

GET - Basically used for just getting (retrieving) some data from the server .

POST - Submits data to be processed to a specified resource.

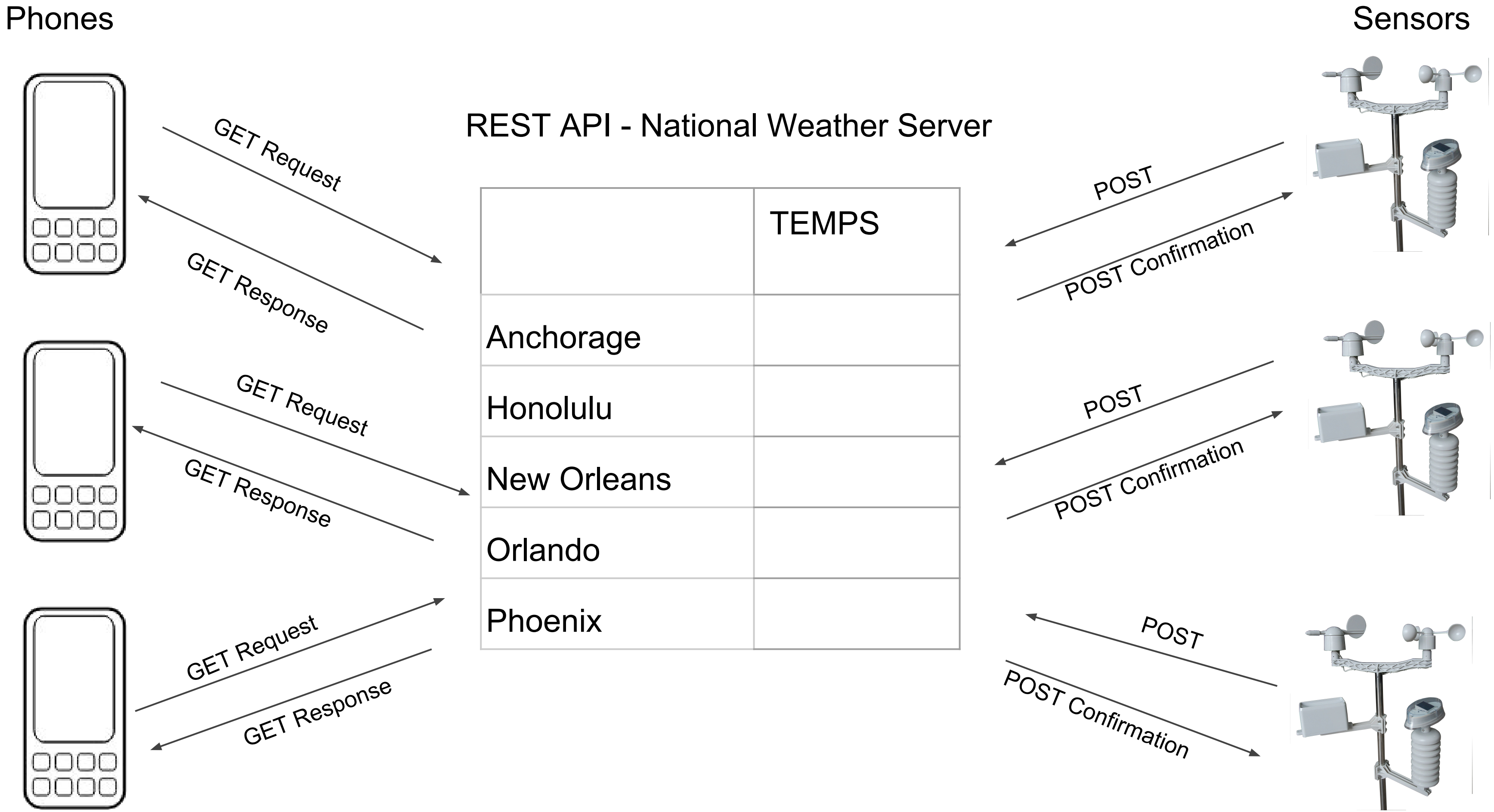
POST - Basically used for putting (sending) some data to the sever.

Endpoint

The URI (**U**nique **R**esource **I**dentifier)that goes after the base URL and points towards the requested API functionality

ACTIVITY

REST API - National Weather Server



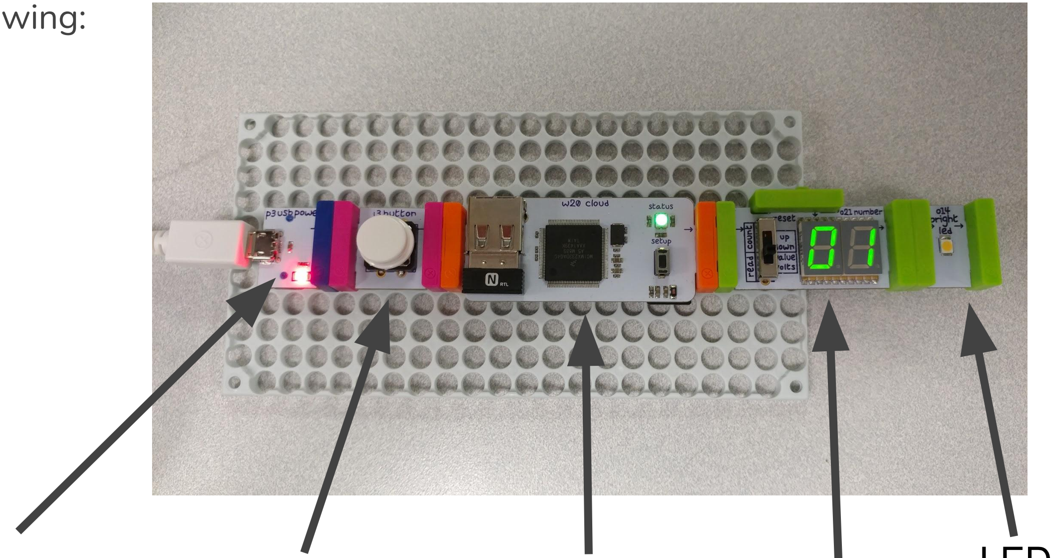
	TEMPS
Anchorage	
Honolulu	
New Orleans	
Orlando	
Phoenix	

ACTIVITY : Removing The Abstraction

Let’s investigate sending our own GET and POST requests to LittleBits using NodeRed!

First Assemble The Little Bits

Please Set Up The Following:



USB Power

Button

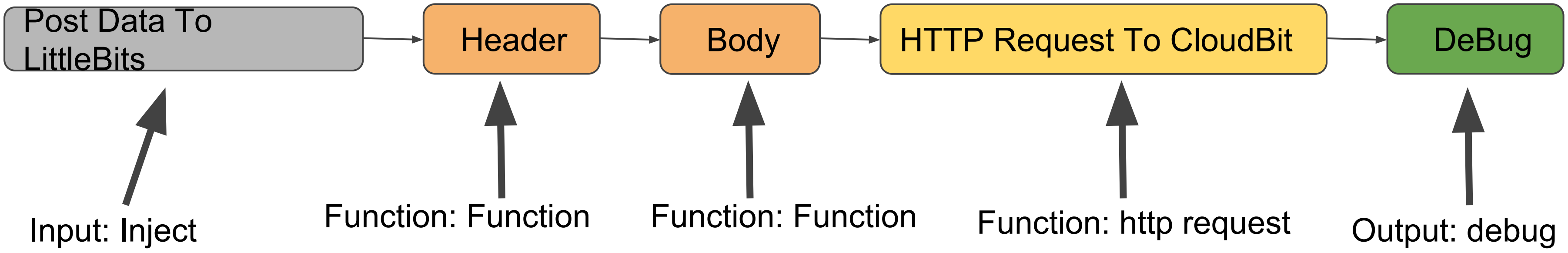
CloudBit

Number Display

LED

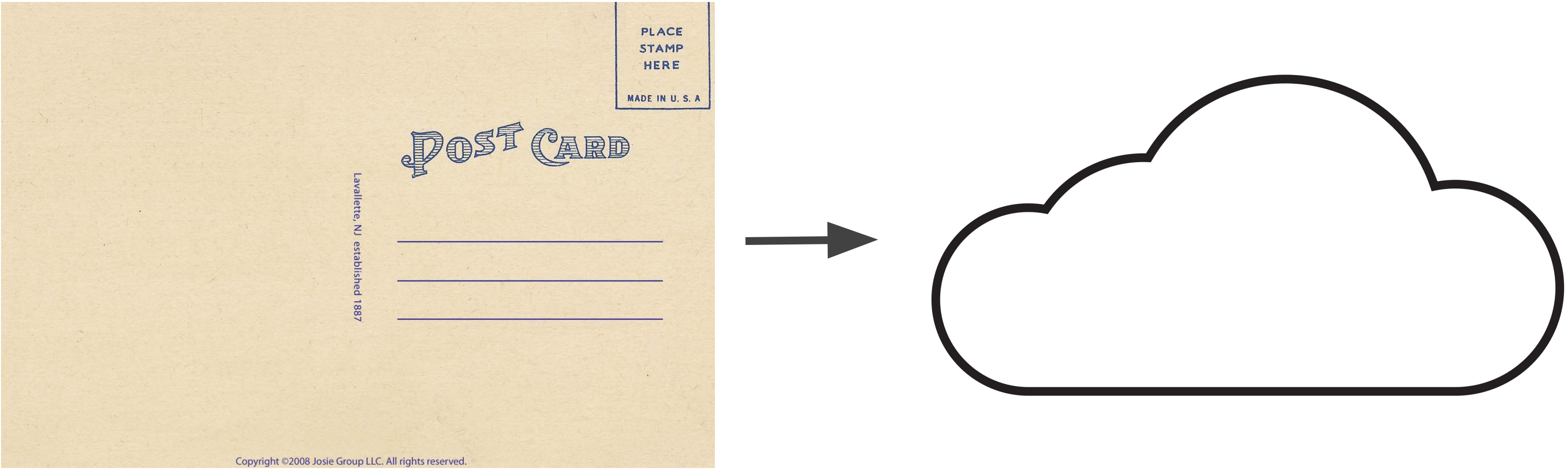
Now, the fun part!

We will be creating this basic flow in Node Red on the Raspberry Pi

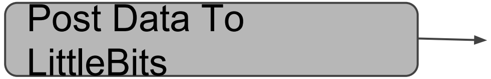


Why?

We want to send our own **Post** Data to the CloudBit


PLACE
STAMP
HERE
MADE IN U. S. A
POST CARD
Lavallette, NJ established 1887
Copyright ©2008 Josie Group LLC. All rights reserved.

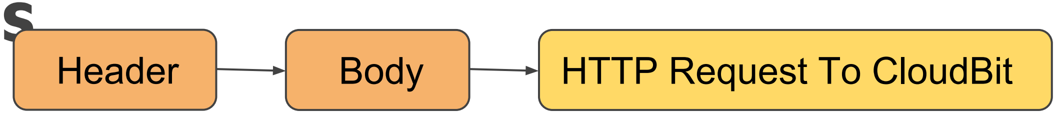
Node 1 - The Button



Post Data To LittleBits

1) Access Node-RED
2) Go To Input: Inject
3) Double click on the node (should be named timestamp)
4) Rename the Event Name to: **Post Data To LittleBits**

The Header, Body, and HTTP Request Nodes

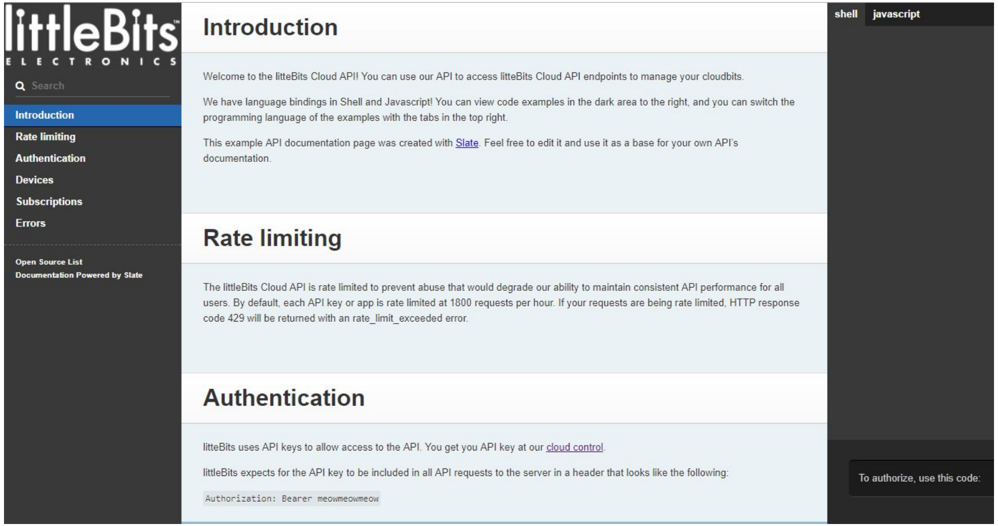


Header

Body

HTTP Request To CloudBit

Please Visit: **http://developers.littlebitscloud.cc/**



Introduction

Welcome to the litteBits Cloud API! You can use our API to access litteBits Cloud API endpoints to manage your cloudbits.

We have language bindings in Shell and Javascript! You can view code examples in the dark area to the right, and you can switch the programming language of the examples with the tabs in the top right.

This example API documentation page was created with Slate. Feel free to edit it and use it as a base for your own API's documentation.

Rate limiting

The littleBits Cloud API is rate limited to prevent abuse that would degrade our ability to maintain consistent API performance for all users. By default, each API key or app is rate limited at 1800 requests per hour. If your requests are being rate limited, HTTP response code 429 will be returned with an rate_limit_exceeded error.

Authentication

litteBits uses API keys to allow access to the API. You get you API key at our cloud control.

littleBits expects for the API key to be included in all API requests to the server in a header that looks like the following:

```
Authorization: Bearer meowmeowmeow
```

To authorize, use this code:

The API Documentation

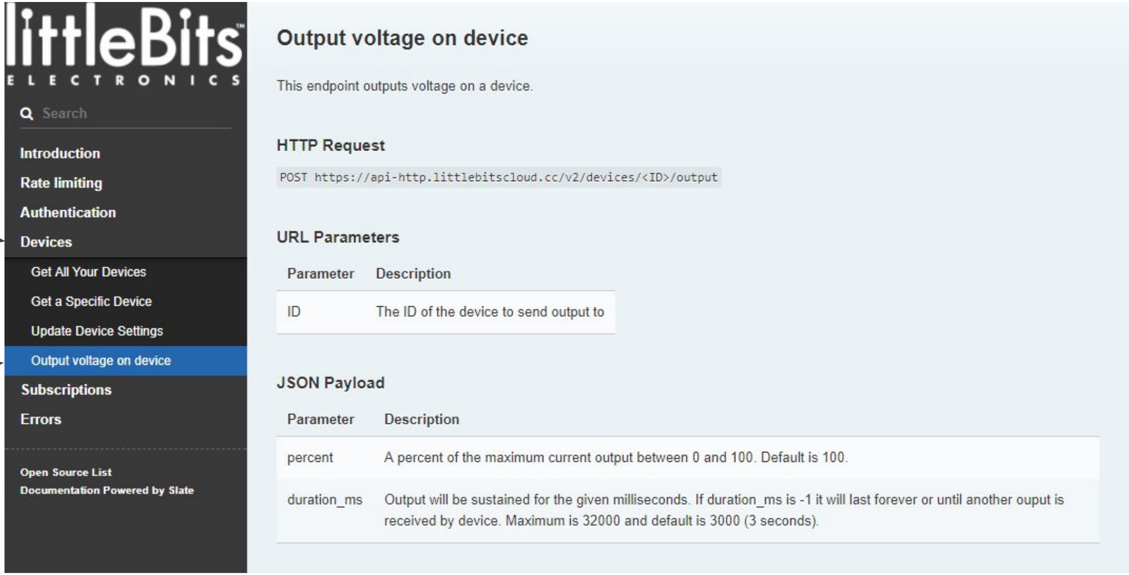
We are going to sing the song we want to sing and become developers!

Go To...

Devices

then

Output voltage on device



Output voltage on device

This endpoint outputs voltage on a device.

HTTP Request

```
POST https://api-http.littlebitscloud.cc/v2/devices/<ID>/output
```

URL Parameters

Parameter	Description
ID	The ID of the device to send output to

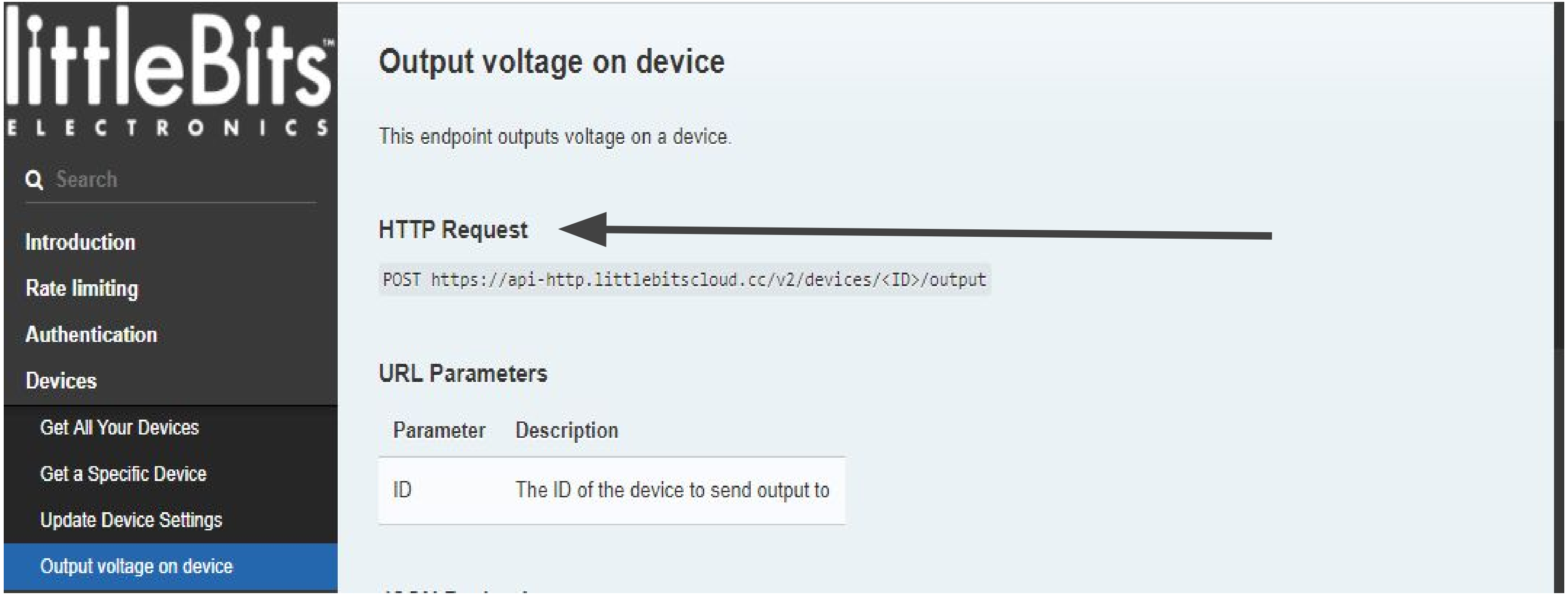
JSON Payload

Parameter	Description
percent	A percent of the maximum current output between 0 and 100. Default is 100.
duration_ms	Output will be sustained for the given milliseconds. If duration_ms is -1 it will last forever or until another ouput is received by device. Maximum is 32000 and default is 3000 (3 seconds).

Why?

Remember the end goal

HTTP Request To CloudBit



Output voltage on device

This endpoint outputs voltage on a device.

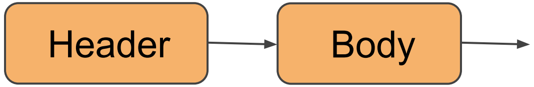
HTTP Request

```
POST https://api-http.littlebitscloud.cc/v2/devices/<ID>/output
```

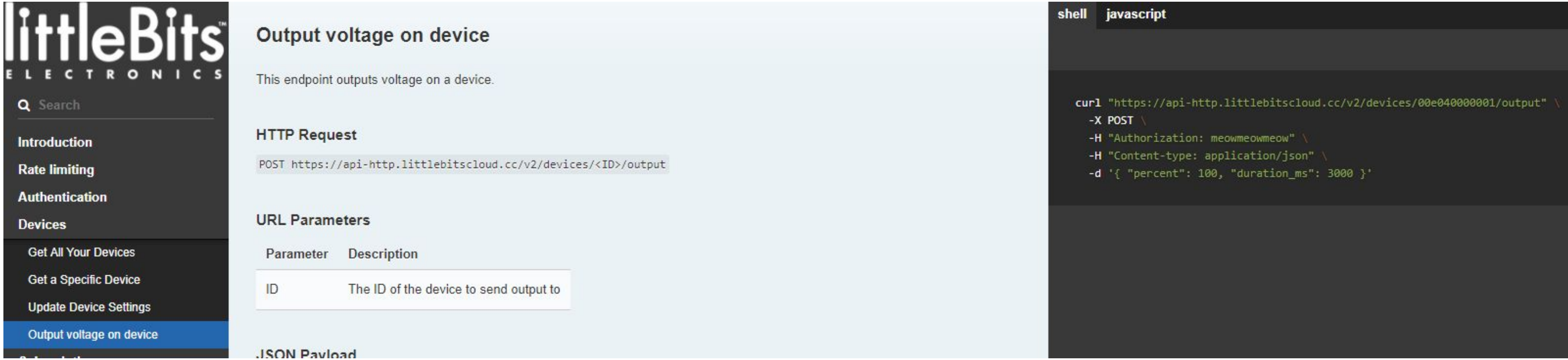
URL Parameters

Parameter	Description
ID	The ID of the device to send output to

Setup Header and Body



Pay Attention to this!



Output voltage on device

This endpoint outputs voltage on a device.

HTTP Request

```
POST https://api-http.littlebitscloud.cc/v2/devices/<ID>/output
```

URL Parameters

Parameter	Description
ID	The ID of the device to send output to

JSON Payload

```
curl "https://api-http.littlebitscloud.cc/v2/devices/00e040000001/output" \
  -X POST \
  -H "Authorization: meowmeowmeow" \
  -H "Content-type: application/json" \
  -d '{ "percent": 100, "duration_ms": 3000 }'
```

Node 2 - The Header

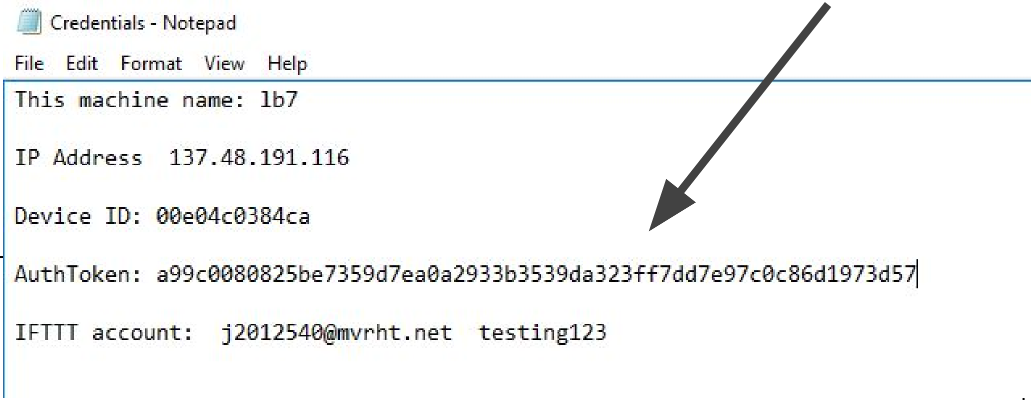
1) Double click on the node
2) Rename the Function Name to: **Header**
3) Type the following 4 lines in the Function Box

Function

```
msg.headers = { };

msg.headers['Authorization'] = "Bearer <AuthToken>";

msg.headers['Content-Type'] = 'application/json';

return msg;
```

Node 2 - The Header

Replace the “<AuthToken>” with the AuthToken from the Credentials File



```
This machine name: lb7

IP Address  137.48.191.116

Device ID: 00e04c0384ca

AuthToken: a99c0080825be7359d7ea0a2933b3539da323ff7dd7e97c0c86d1973d57

IFTTT account:  j2012540@mvrht.net  testing123
```

Function

```
msg.headers = { };

msg.headers['Authorization'] = "Bearer a99c0080825be7359d7ea0a2933b3539da323ff7dd7e97c0c86d1";

msg.headers['Content-Type'] = 'application/json';

return msg;
```

Node 3 - The Body

1) Double click on the node
2) Rename the Function Name to: **Body**
3) Type the following 5 lines in the Function Box

Function

```
msg.payload = {
                    "percent": 25,
                    "duration_ms": 5000
                };

return msg;
```

Node 3 - The Body

The Number We Are POSTING To The CloudBit

Function

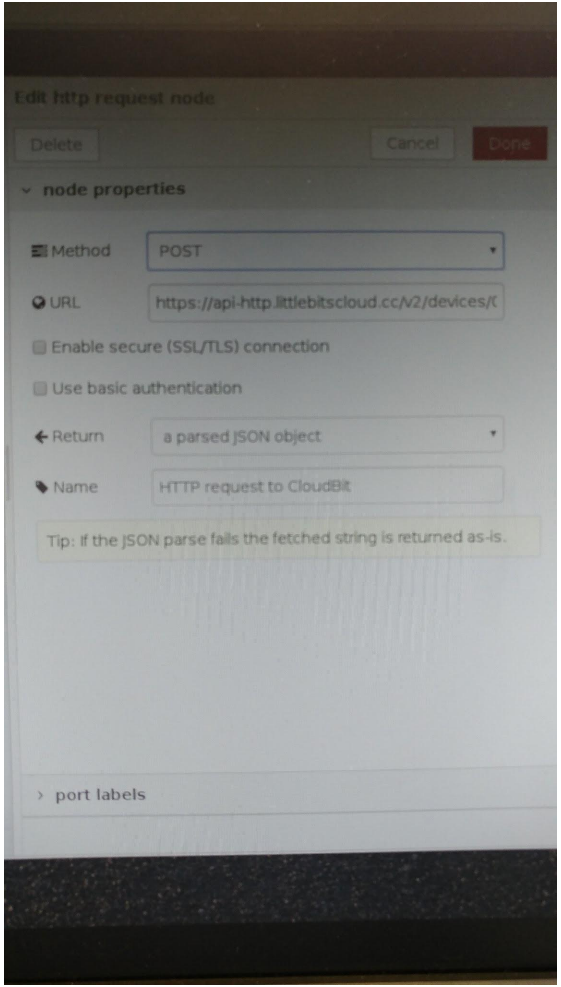
```
msg.payload = {
                            "percent": 25,
                            "duration_ms": 5000
                       };

return msg;
```

The Duration The Number Will Be Displayed (in millis)

Node 4 - The HTTP Response

HTTP Request To CloudBit

1) Double click on the node
2) Method: POST
3) URL:
4) Return: a parsed jSON object
5) Name: HTTP Request To CloudBit



Method: POST

URL: https://api-http.littlebitscloud.cc/v2/devices/

□ Enable secure (SSL/TLS) connection

□ Use basic authentication

Return: a parsed JSON object

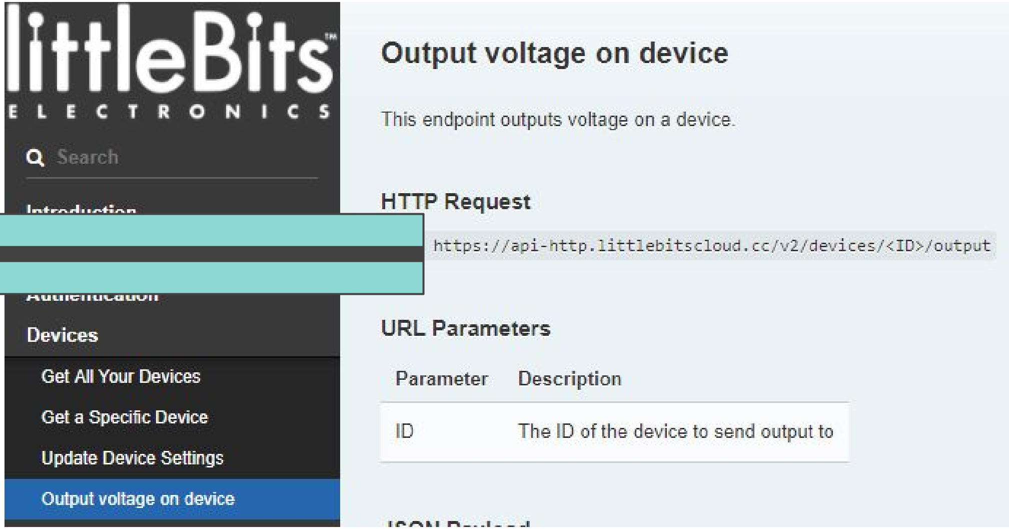
Name: HTTP request to CloudBit

Tip: If the JSON parse fails the fetched string is returned as-is.

Node 4 - The HTTP Response

HTTP Request To CloudBit

1) Double click on the node
2) Method: POST
3) URL:
4) Return: a parsed jSON object
5) Name: HTTP Request To CloudBit



Output voltage on device

This endpoint outputs voltage on a device.

HTTP Request

```
https://api-http.littlebitscloud.cc/v2/devices/<ID>/output
```

URL Parameters

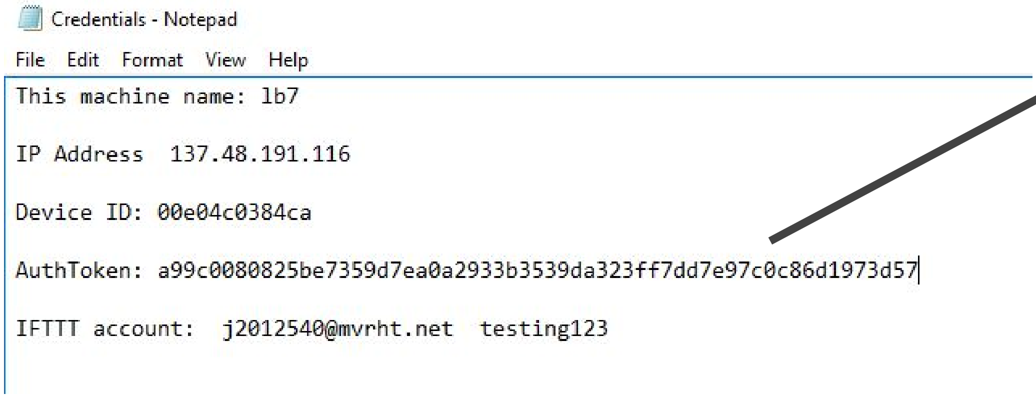
Parameter	Description
ID	The ID of the device to send output to

JSON Payload

Node 4 - The HTTP Response

HTTP Request To CloudBit

3) URL: `https://api-http.littlebitscloud.cc/v2/devices/<ID>/output`



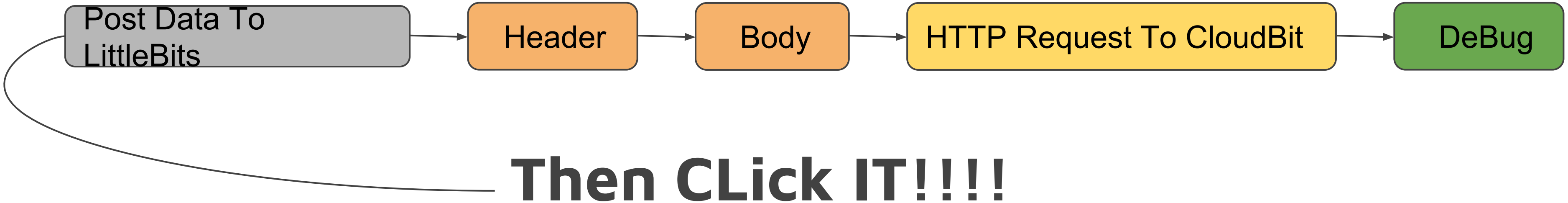
```
This machine name: lb7

IP Address  137.48.191.116

Device ID: 00e04c0384ca

AuthToken: a99c0080825be7359d7ea0a2933b3539da323ff7dd7e97c0c86d1973d57

IFTTT account:  j2012540@mvrht.net  testing123
```

Node 5 - Debug

DeBug

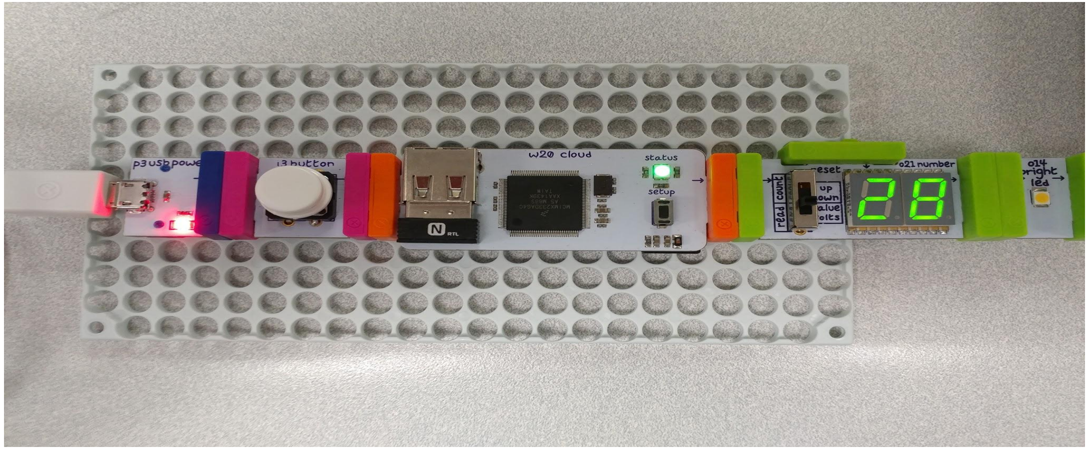
Add a debug.... just incase it all goes wrong

Does the FLOW look like this???



Post Data To LittleBits → Header → Body → HTTP Request To CloudBit → DeBug

Then CLick IT!!!!



What Do You See And Why??

What is Happening?

Is a GET Request involved?

Is there a POST Request involved?

How is this similar to the Classroom Activity? What is different?

How would we make this exactly the same as the Classroom Activity?

Going The Distance

Open The Flow Created For You In Node RED

Rearrange and a connect the nodes in the correct order to create the “real life” classroom situation.

Cybersecurity First Principle Reflections

In this lesson, we saw web services, such as IFTTT, can abstract away details about devices and instead focus on recipes or design patterns to describe how things work. We also saw that by keeping functionality modular, IFTTT can combine Littlebits with many other services.

Web services use resource encapsulation to ensure that all functions related to the execution of an app or service are neatly within the scope of the service itself. IFTTT relies on services to be encapsulated so that they can provide external services with the ability to use them without worrying about connecting to multiple other related services.

Data hiding is also important to prevent internal data in the service from being released outside of the service invocation. Local data remains hidden, while interfaces expose only what the service wants to release (for instance to IFTTT). This also relates to minimization because services can turn ports and other access off except for the specific interfaces it wants to leave open for other services to use.

Acknowledgements
Special thanks to Dr. Matthew L. Hale and Dr. Robin Gandhi for reviewing this lesson.