



Hi.

My name is Ehren Coker.

I work at World Wide Technology.

Internet is a cool job.

Event Driven Architecture

(sounds real nerdy)

Bald Eagle Collection 2018

MADE IN AMERICA!

SHOP NOW

PRODUCTS



American Freedom Eagle
\$18.64



Very Cold Eagle
\$35.31



Staring Contest Eagle
\$75.00



Admiral Eagle
\$25.50



WHY?

Well...

**I built a cardboard rocket
for a 4-year-old.**

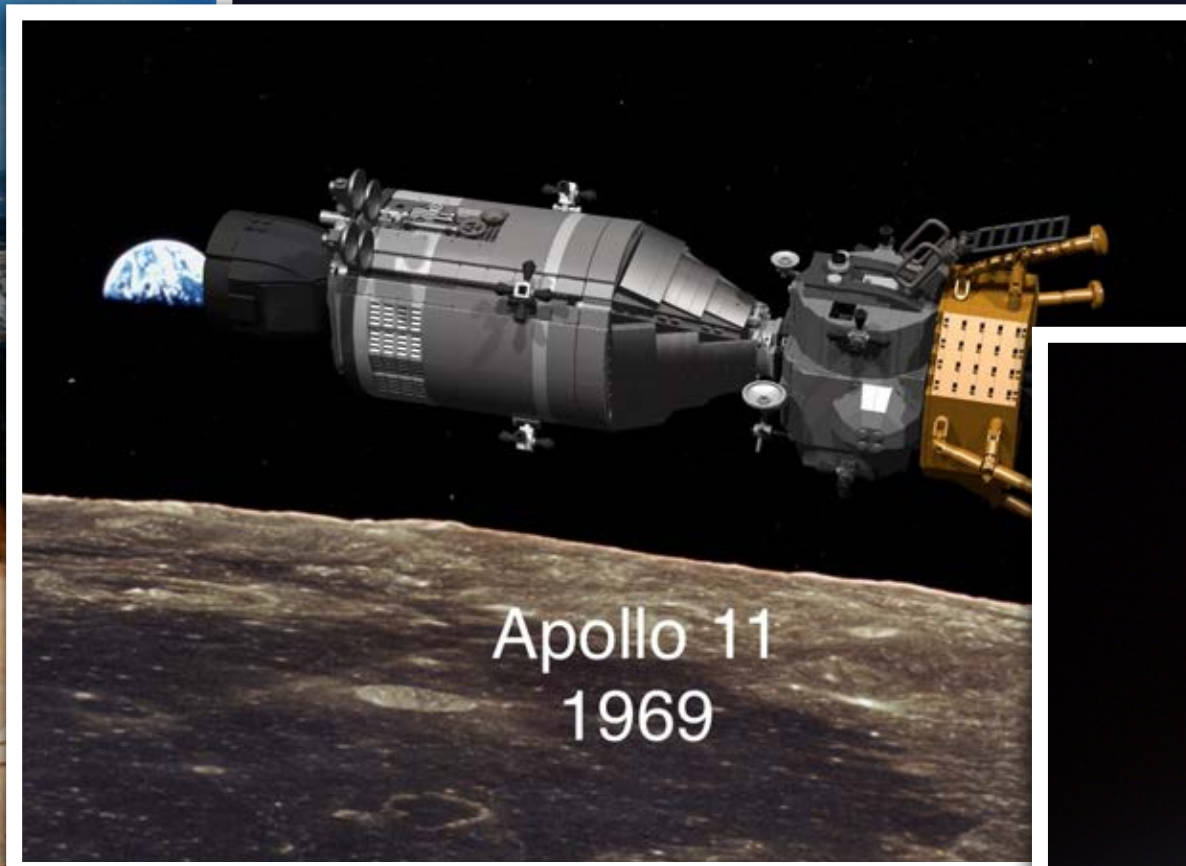
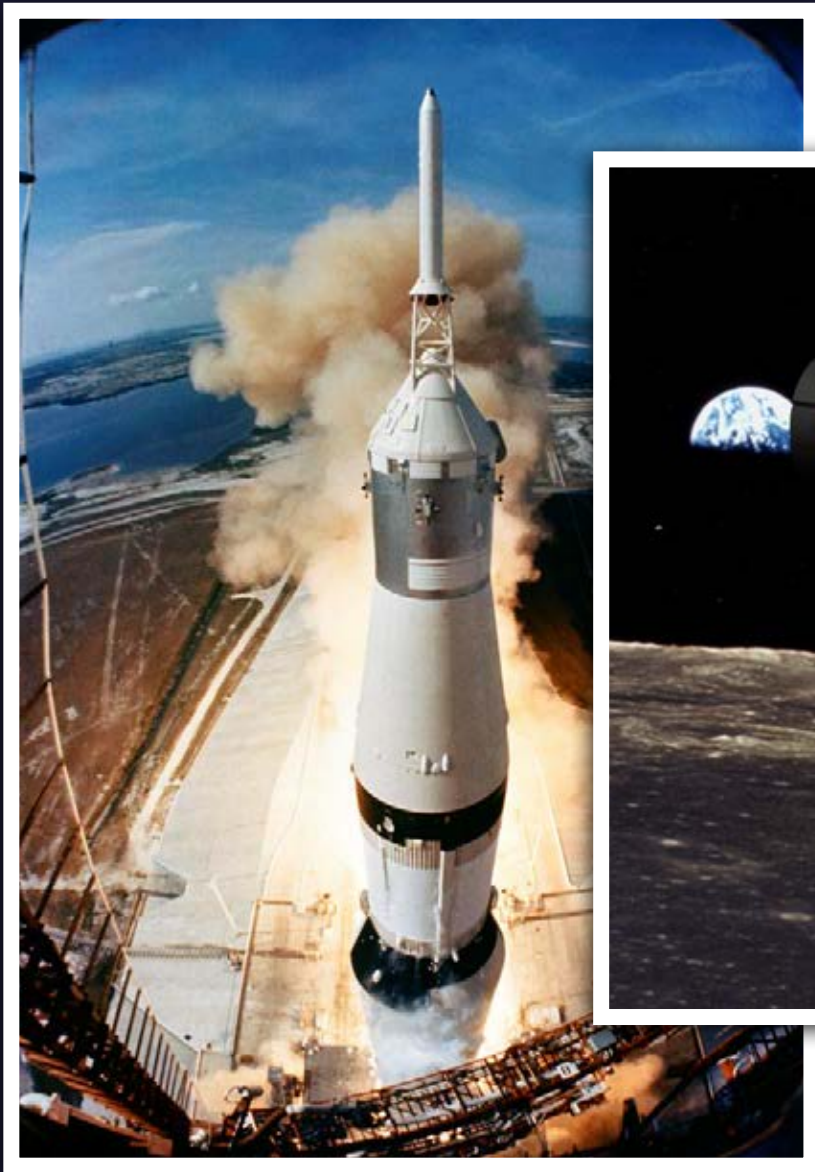


Then I sent some
children into space.



The Mission

Three Stages of Flight



Apollo 11
1969



Equipment:

- Hue Bulbs
- TP-Link Smart Plug
- Apollo 11 Audio
- Sonos Speakers
- Christmas Lights

[Readme](#)[5 Dependencies](#)[68 Dependents](#)[36 Versions](#)

Node Hue API

npm

v2.4.4

An API library for Node.js that interacts with the Philips Hue Bridge to control Philips Hue Light Bulbs and Philips Living Color Lamps.

This library abstracts away the actual Philips Hue Bridge REST API and provides all of the features of the Phillips API and a number of useful functions to control the lights and bridge remotely.

The library supports both function `callbacks` and `Q promises` for all the functions of the API. So for each function in the API, if a callback is provided, then a callback will be used to return any results or notification of success, in a true Node.js fashion. If the callback is omitted then a promise will be returned for use in chaining or in most cases simpler handling of the results.

When using `Q promises`, it is necessary to call `done()` on any promises that are returned, otherwise errors can be swallowed silently.

Table of Contents

- [Change Log](#)
- [Work In Progress](#)
- [Breaking Changes in 2.0.x](#)
- [Philips Hue Resources](#)
- [Installation](#)

[Examples](#)

install

```
> npm i node-hue-api
```

↓ weekly downloads



version

2.4.4

license

Apache-2.0

open issues

9

pull requests

0

homepage

github.com

repository


 github

last publish

4 months ago

collaborators



 Test with RunKit

Report a vulnerability

tplink-cloud-api

0.3.8 • Public • Published 2 months ago

Readme

2 Dependencies

1 Dependents

17 Versions

build passing

Introduction

The `tplink-cloud-api` NPM module allows you to remotely control your TP-Link smartplugs (HS100, HS110), smart switches (HS200), and smartbulbs (LB100, LB110, LB120, LB130) using the TP-Link cloud web service, from anywhere, without the need to be on the same wifi/lan.

It's especially useful in scenarios where you want to control your devices from public web services, like IFTTT, Thinger.io, Webtasks.io, Glitch.com...

It's based on my investigation work on the TP-Link API protocol, which I have been sharing in my blog <http://itnerd.com>.

Installation

You can install this module with `npm` :

```
npm install --save tplink-cloud-api
```

Usage

Authenticate

First instantiate a TP-Link object. TermID (UUIDv4) is generated if not specified:

install

```
> npm i tplink-cloud-api
```

± weekly downloads

32



version

0.3.8

license

GPL-3.0

open issues

4

pull requests

0

homepage

itnerd.space

repository


 github

last publish

2 months ago

collaborators



 Test with RunKit

Report a vulnerability

JS hueController.js x

```
1  const hue = require("node-hue-api");
2  const { lightState, HueApi } = hue;
3
4  const delay = require('../services/delayService');
5  const settings = require('../constants/settings');
6  const api = new HueApi(settings.hue.HOST, settings.hue.USERNAME);
7
8  /* -- Route Controllers -- */
9  exports.getBridges = async () => {
10   |   return hue.nupnpSearch();
11   | }
12
13  exports.getLights = async () => {
14   |   return api.getLights();
15   | }
16
17  exports.getState = async () => {
18   |   return api.getFullState();
19   | };
20
21  exports.getUsers = async () => {
22   |   return api.registeredUsers();
23   | }
24
25  exports.brightness = async (request) => {
26   |   const { id, brightness } = request.params;
27   |   let state = lightState.create().on().brightness(brightness);
28   |   return api.setLightState(id, state);
29   | }
30
31  /* -- Consumed Controllers -- */
32  exports.setBrightness = async (lightId, brightness, transition = 0) => {
33   |   let state = lightState.create().on().brightness(brightness).transitionTime(transition);
34   |   return api.setLightState(lightId, state);
35   | }
36
37  exports.setBrightnessWait = async (lightId, brightness, wait) => {
38   |   let state = lightState.create().on().brightness(brightness).transitionTime(Math.round(wait / 100));
39   |   await api.setLightState(lightId, state);
40   |   return delay(wait);
41   | }
42
```


JS tpLinkController.js x

```
1  const { login } = require("tplink-cloud-api");
2
3  let tplink = null;
4
5  (async () => {
6    |   tplink = await login(credentials.email, credentials.password);
7  })();
8
9  exports.list = async () => {
10   |   return await tplink.getDeviceList();
11 }
12
13 exports.toggle = async () => {
14   |   return await tplink.getHS100("Stars").toggle();
15 }
16
17 exports.off = async () => {
18   |   return await tplink.getHS100("Stars").powerOff();
19 }
20
21 exports.on = async () => {
22   |   return await tplink.getHS100("Stars").powerOn();
23 }
```

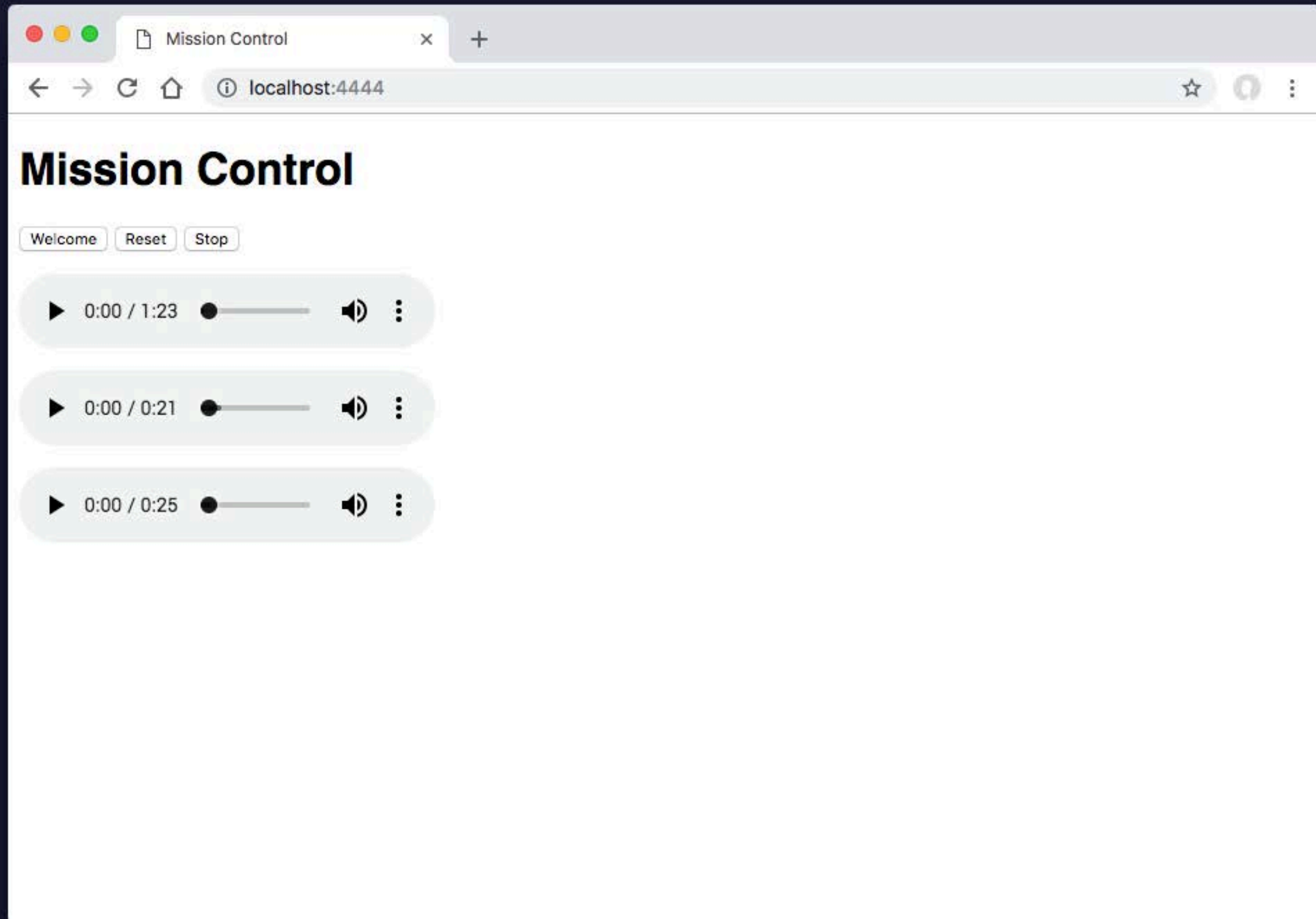
JS scenesService.js x

```
31 module.exports = {
32   reset: resetLights,
33
34   welcome: () => new Promise(async (resolve) => {
35     await resetLights();
36     welcomeInterval = createFlickerInterval(2500, 25, 250, cabinLight);
37     resolve(true);
38   }),
39
40   blastoff: () => new Promise(async (resolve) => {
41     await resetLights();
42     blastoffInterval = createFlickerInterval(200, 50, 1255, emergencyLight);
43     resolve(true);
44   }),
45
46   landing: () => new Promise(async (resolve) => {
47     await resetLights();
48     await hueController.setBrightness(exteriorLight, 15);
49     resolve(true);
50   }),
51
52   countdown: () => new Promise(async (resolve) => {
53     await resetLights();
54     countdownInterval = createFlickerInterval(800, 60, 150, cabinLight);
55     resolve(true);
56   }),
57
58   stop: () => new Promise(async (resolve) => {
59     await resetLights();
60     resolve(true);
61   })
62 }
```

Now I needed an...

AMAZING UI

that I could build in 30 minutes



```
<div class="controls">
  <button @click.prevent="welcome">Welcome</button>
  <button @click.prevent="reset">Reset</button>
  <button @click.prevent="stop">Stop</button>
</div>
```

```
<audio
  @play="countdown"
  @timeupdate="checkLiftoffTime"
  @ended="startMidflight"
  controls
  ref="liftoff"
>
  <source src="/1-liftoff.mp3" type="audio/mp3" />
  Your browser does not support the audio tag.
</audio>
```

```
<audio
  @play="midflight"
  @ended="startLanding"
  controls
  ref="midflight"
>
  <source src="/2-midflight.mp3" type="audio/mp3" />
  Your browser does not support the audio tag.
</audio>
```

```
<audio
  @play="landing"
  @timeupdate="checkLandingTime"
  controls
  ref="landing"
>
  <source src="/3-landing.mp3" type="audio/mp3" />
  Your browser does not support the audio tag.
</audio>
```

Events

```
<audio
  @play="countdown"
  @timeupdate="checkLiftoffTime"
  @ended="startMidflight"
  controls
  ref="liftoff"
>
  <source src="/1-liftoff.mp3" type="audio/mp3" />
  Your browser does not support the audio tag.
</audio>
```

Actions

```
checkLiftoffTime(event) {
  if (
    event.target.currentTime >= 20
    && !this.runningLiftoff
  ) {
    this.liftoff()
  }
},
async countdown() {
  this.reset()
  await axios.get('/tplink/off')
  await axios.get('/director/countdown')
},
async liftoff() {
  this.runningLiftoff = true
  await axios.get('/director/blastoff')
},
```


What Did It Do?

Using `<audio>` native events, we triggered a series of “scenes” to create our flight.

We flashed lights and turned on the “stars” for our lunar landing.

It was pretty dope.

We all know...

Browser Events Drive Experiences

How does this translate to the server?

Events are state changes and...

Servers have “State”

Bald Eagle Collection 2018

MADE IN AMERICA!

SHOP NOW

PRODUCTS



American Freedom Eagle
\$18.64



Very Cold Eagle
\$35.31



Staring Contest Eagle
\$75.00



Admiral Eagle
\$25.50



Checkout Code

```
const checkout = async (cart, payment, billingAddress, shippingAddress, customer) => {
```

Checkout Code

```
const checkout = async (cart, payment, billingAddress, shippingAddress, customer) => {  
  await validatePaymentMethod(payment, billingAddress)  
  
}
```

Checkout Code

```
const checkout = async (cart, payment, billingAddress, shippingAddress, customer) => {  
  await validatePaymentMethod(payment, billingAddress)  
  await confirmInventoryAvailability(cart)  
  
}
```


Checkout Code

```
const checkout = async (cart, payment, billingAddress, shippingAddress, customer) => {  
  await validatePaymentMethod(payment, billingAddress)  
  await confirmInventoryAvailability(cart)  
  await alertAuthorities(customer, shippingAddress)  
  
}
```


Checkout Code

```
const checkout = async (cart, payment, billingAddress, shippingAddress, customer) => {  
  await validatePaymentMethod(payment, billingAddress)  
  await confirmInventoryAvailability(cart)  
  await alertAuthorities(customer, shippingAddress)  
  await alertPETA(customer, shippingAddress)  
  
}
```

Checkout Code

```
const checkout = async (cart, payment, billingAddress, shippingAddress, customer) => {  
  await validatePaymentMethod(payment, billingAddress)  
  await confirmInventoryAvailability(cart)  
  await alertAuthorities(customer, shippingAddress)  
  await alertPETA(customer, shippingAddress)  
  await textThePresident(customer)  
  
}
```

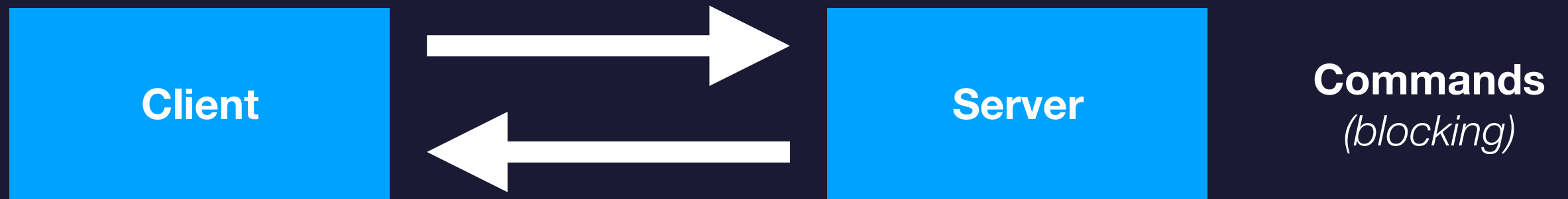
Checkout Code

```
const checkout = async (cart, payment, billingAddress, shippingAddress, customer) => {  
  await validatePaymentMethod(payment, billingAddress)  
  await confirmInventoryAvailability(cart)  
  await alertAuthorities(customer, shippingAddress)  
  await alertPETA(customer, shippingAddress)  
  await textThePresident(customer)  
  await sendConfirmationEmail(customer)  
}
```

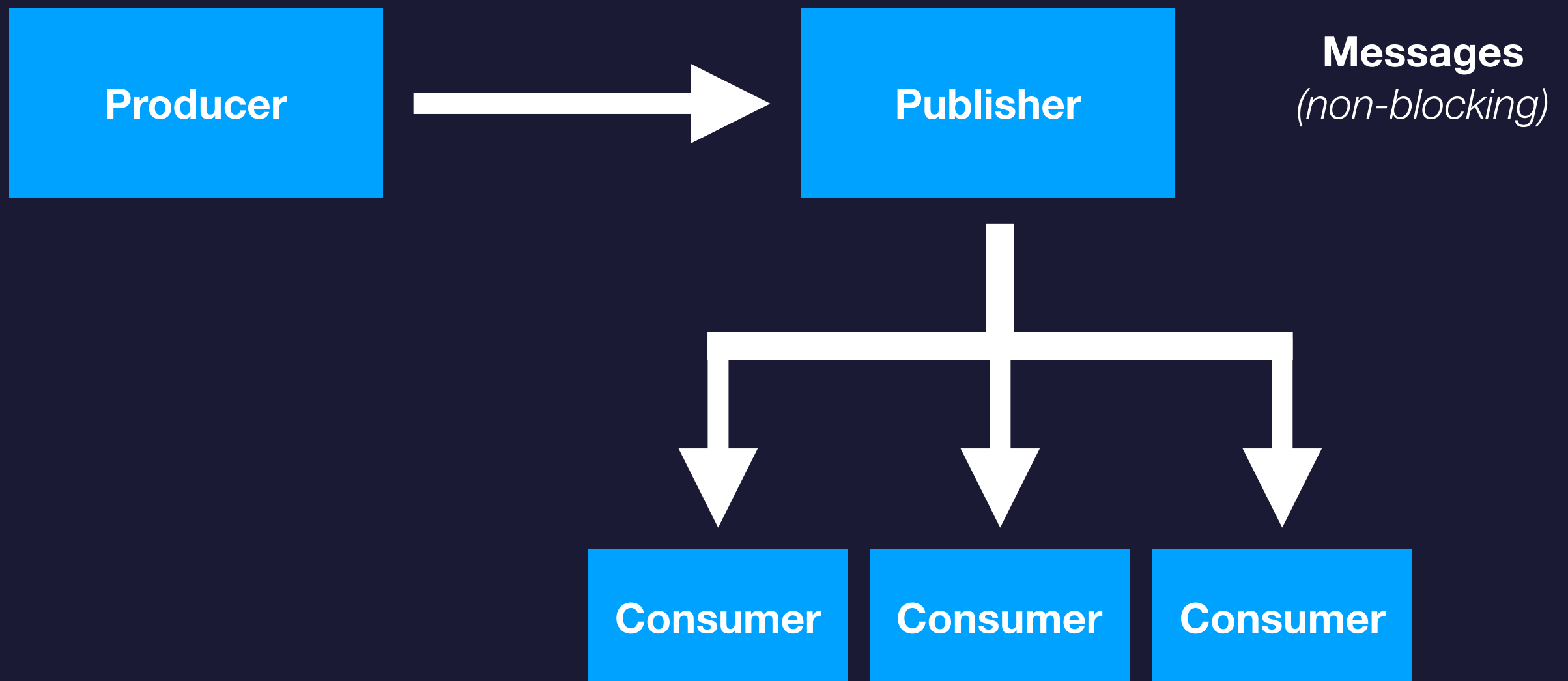
Checkout Code

```
const checkout = async (cart, payment, billingAddress, shippingAddress, customer) => {  
  await validatePaymentMethod(payment, billingAddress)  
  await confirmInventoryAvailability(cart)  
  await alertAuthorities(customer, shippingAddress)  
  await alertPETA(customer, shippingAddress)  
  await textThePresident(customer)  
  await sendConfirmationEmail(customer)  
  return completeCheckout(cart, payment, billingAddress, shippingAddress, customer)  
}
```

REST



Events



Good news, everyone!

Node is still JavaScript

EventEmitter

```
// Define EventEmitter
```

```
const EventEmitter = require('events')  
class Emitter extends EventEmitter {}  
const eagleDepoEmitter = new Emitter()
```

```
module.exports = eagleDepoEmitter
```

EventEmitter

// Publish Event

```
const checkout = {  
  billingAddress: 'billingAddress',  
  cart: 'cart',  
  customer: 'customer',  
  payment: 'payment',  
  shippingAddress: 'shippingAddress',  
}  
eagleEmitter.emit('checkout', checkout)
```

EventEmitter

// Consume Event

```
eagleEmitter.on('checkout', (checkout) => {  
  const { shippingAddress, customer } = checkout  
  // -- Alert Authorities  
});
```

// All Together Now

```
const EventEmitter = require('events');
class Emitter extends EventEmitter {}
const eagleEmitter = new Emitter();

eagleEmitter.on('checkout', (checkout) => {
  // -- Checkout Actions
});

const checkout = {
  billingAddress: 'billingAddress',
  cart: 'cart',
  customer: 'customer',
  payment: 'payment',
  shippingAddress: 'shippingAddress',
}
eagleEmitter.emit('checkout', checkout)
```



```
// A little more modular...
```

```
// Emitter
```

```
const eagleEmitter = require('./services/eagleEmitter')
```

```
// Consumers
```

```
require('./events/checkout')
```

```
const checkout = {  
  billingAddress: 'billingAddress',  
  cart: 'cart',  
  customer: 'customer',  
  payment: 'payment',  
  shippingAddress: 'shippingAddress',  
}
```

```
eagleEmitter.emit('checkout', checkout)
```

Message Brokers

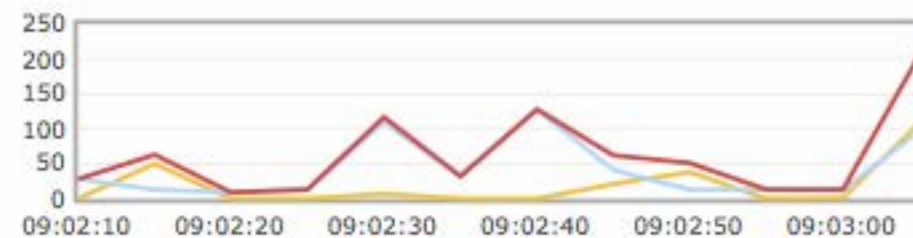
RabbitMQ & Kafka

These tools are used much more frequently

Overview

Totals

Queued messages (chart: last minute) (?)



Ready

0 msg

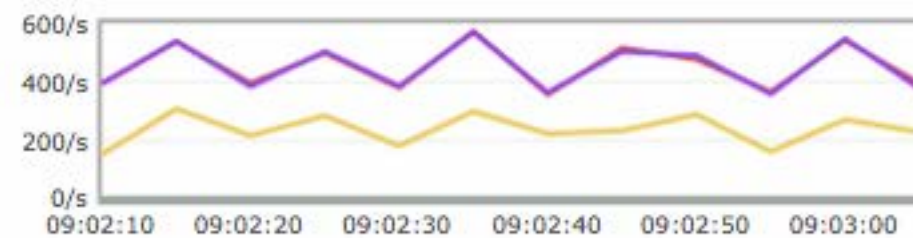
Unacked

12 msg

Total

12 msg

Message rates (chart: last minute) (?)



Publish

230/s

Confirm

0.00/s

Deliver

397/s

Redelivered

0.00/s

Acknowledge

379/s

Get

0.00/s

Get (noack)

0.00/s

Global counts (?)

Connections: 11

Channels: 66

Exchanges: 23

Queues: 14

Consumers: 31

Why Does it Matter?

- Performance
- Removes Tight Coupling of *Non-Essential* Calls
- Easier to Iterate
- They Make you Think Differently
- Events are *EVERYWHERE*

Events represent a state change in...

the

Client

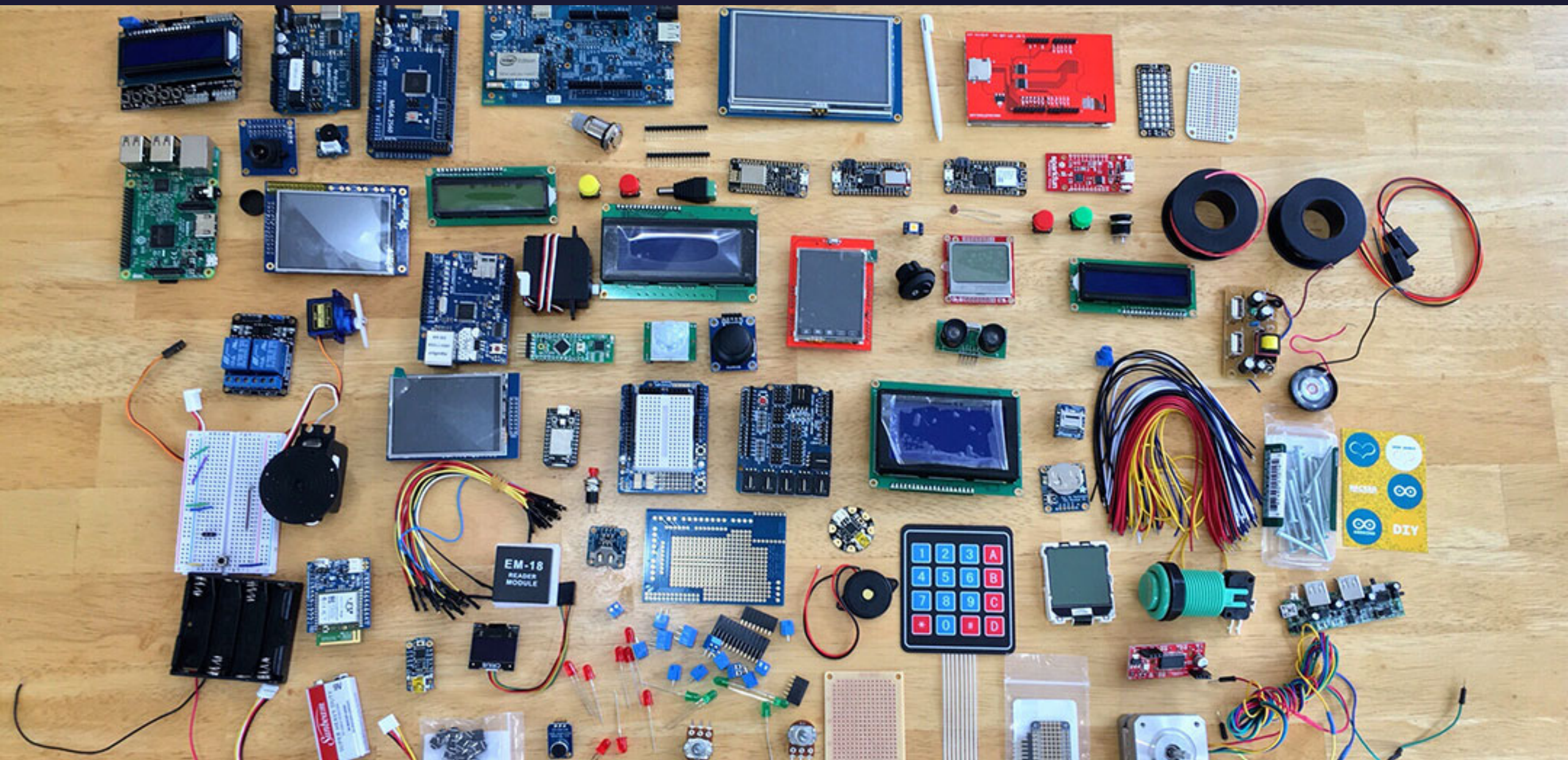
the

Server

your

Home

Using an Events Based Approach,
you can turn this pile of electronics into *magic*.



Thanks!