HACKSTACK.JS

WORKING WITH BROKEN APIs DELIVERED LATE

Yuri Takhteyev
CTO, Rangle.io

RANGLE.IO
REWRITING THE WEB

Working with a backend API

- A common case: you build the client, someone provides the API.
- The API team usually provides the API on time and it works flawlessly.
- The API never changes.

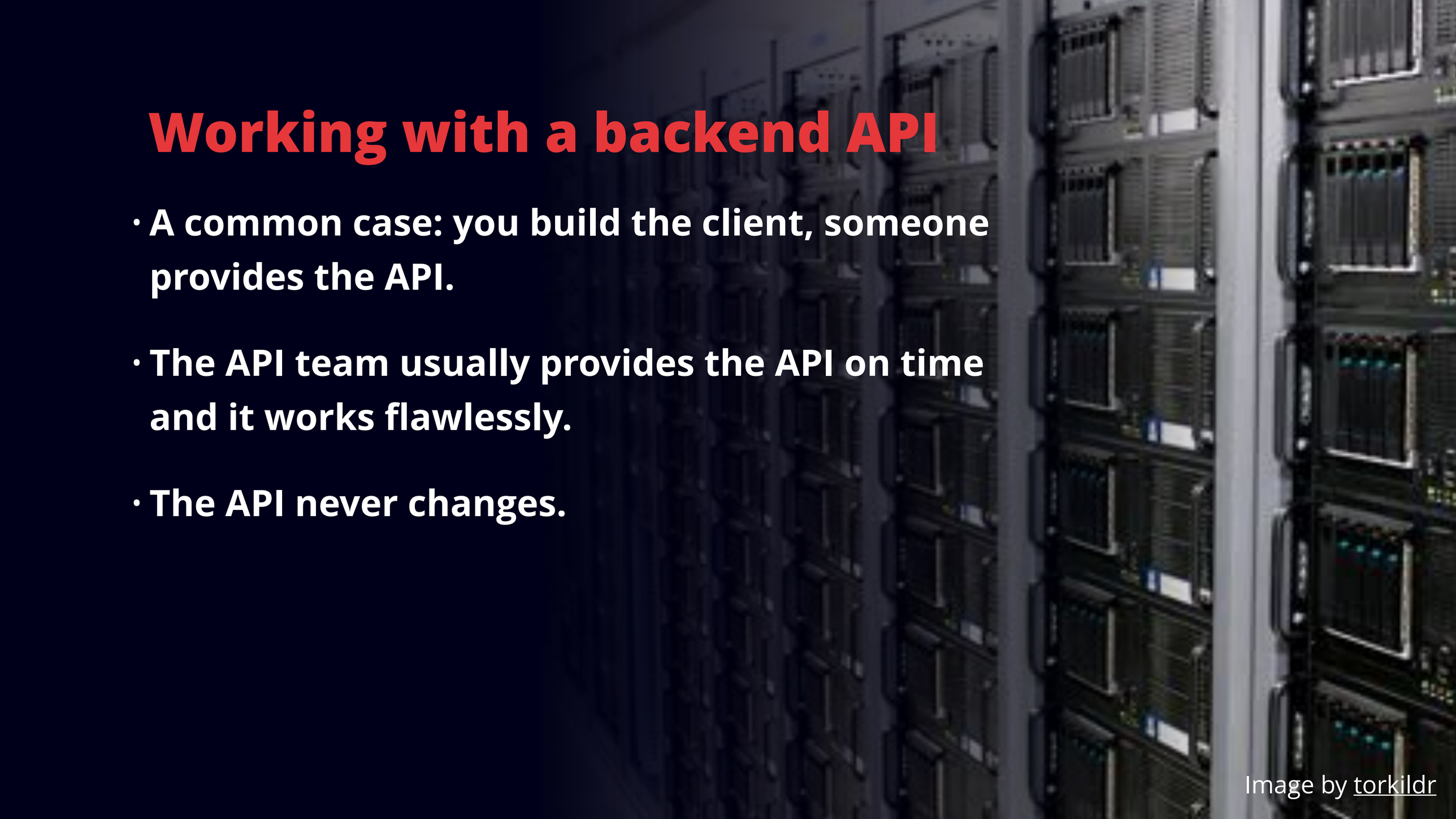

Image by torkildr

The Reality

- Who knows what’s a good API? Hopefully you!
- Who will test the API? Probably you!
- When will it be ready?

Image by torkildr

Translation Sheet

- **"It's working now."** It's working but hasn't been tested and will probably be redesigned as the project unfolds.
- **"We'll have it in a few days."** You'll see the API in a few weeks or months.
- **"We are working on it."** You might have to complete all of your work before the API is ready.

☛ Prepare for the worst: a broken API delivered late in the project.

Image by fuzzyyol

“Hack Stack" to the Resque

- A method for working with a broken API delivered late.
- ~~Not a library or a tool – rather, a set of best practices based on our experience.~~
- Actually, it ***is*** a library now: HackStack.JS.

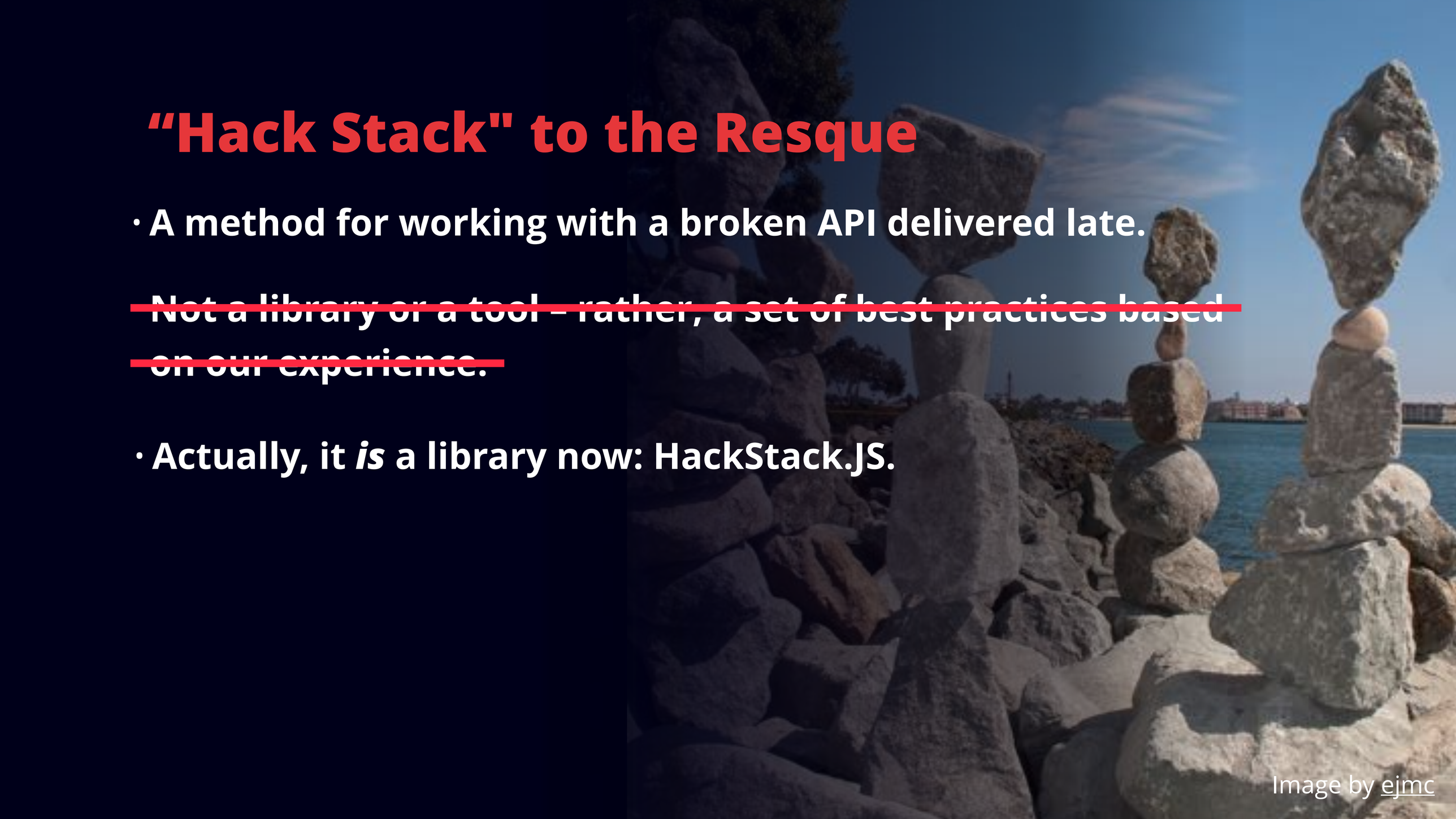

Image by ejmc

Why HackStack?

Mocking the API

- Document the API.
- Use TDD.
- Mocking with online tools: too limiting.
- A mock server: can work well, but expensive.
- Client-side mocking: our preferred solution.
- HackStack.JS provides an implementation.

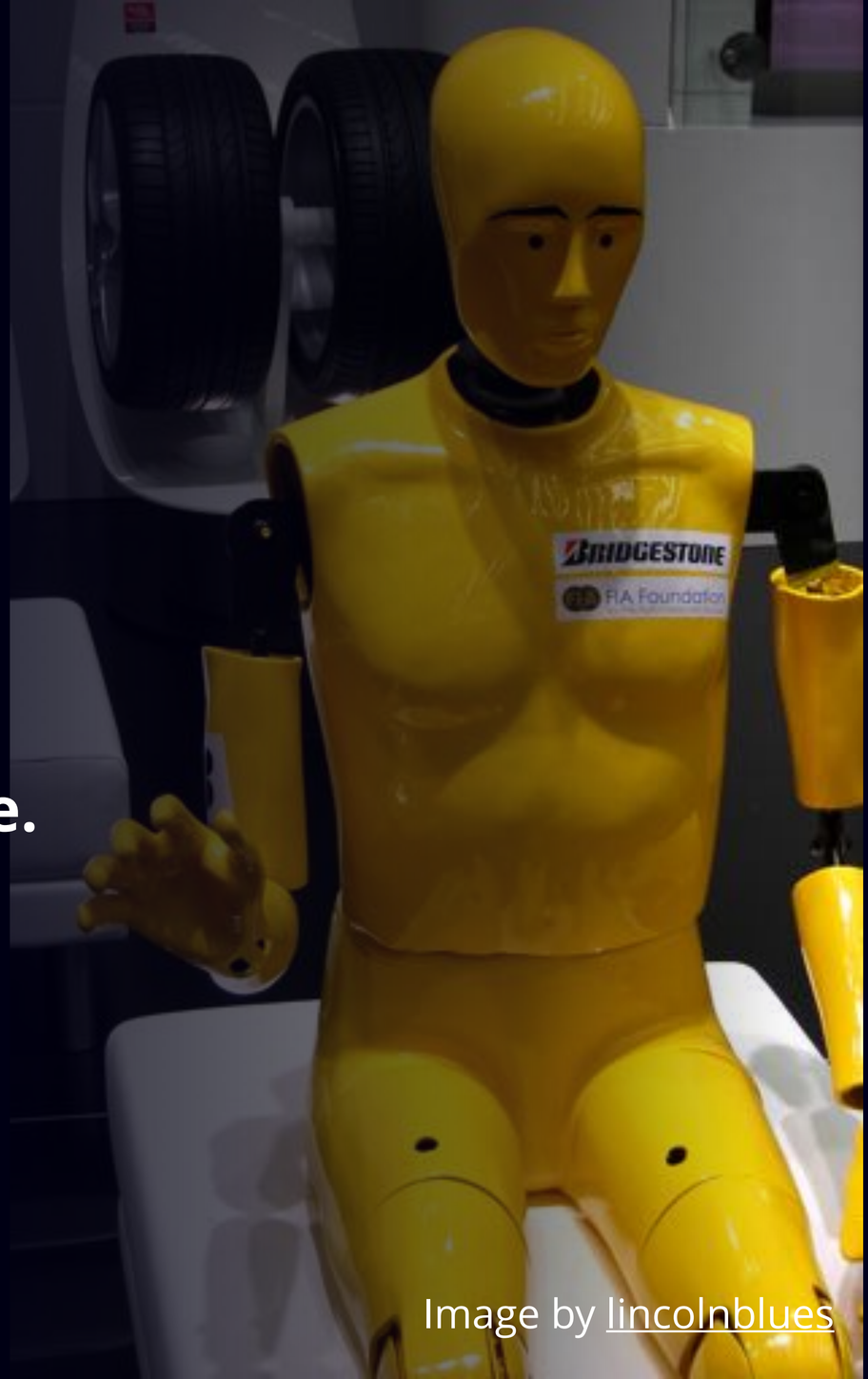

Image by lincolnblues

Getting HackStack

Bower: `bower install angular-hackstack`

Github: https://github.com/rangle/hackstack

Slides: http://yto.io/xngu

☛ Version 0.1.0.

★ Implementation by Brian Olynyk
& Ahmed Al-Sudani

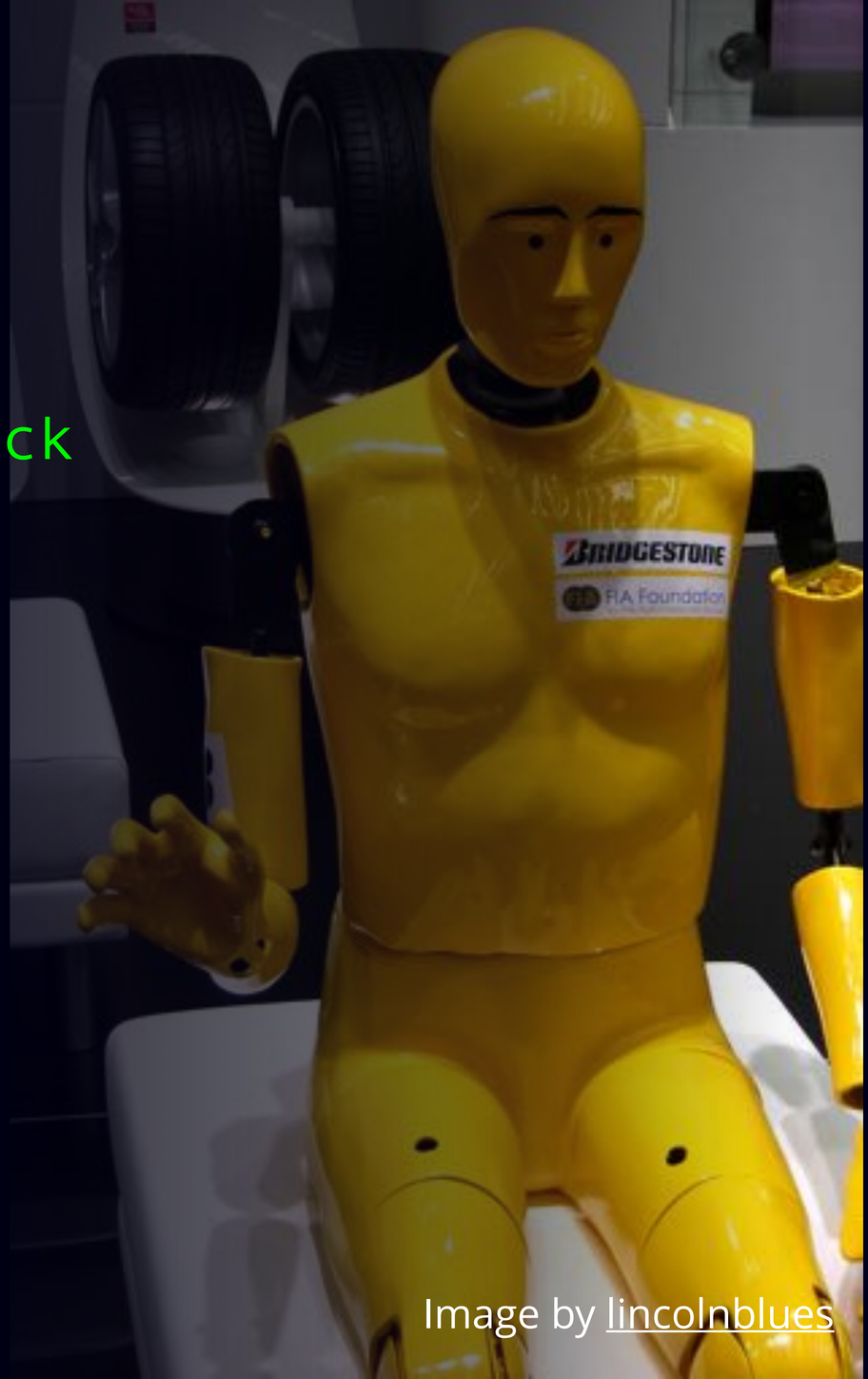

Image by lincolnblues

Scenario: `/birds/`

- We expect to eventually have a `/birds/` endpoint that would give us REST access to a collection of birds.
- We've agreed on what the returned JSON would look like.
- We've also agreed on how problems are going to be handled.
- Now we are waiting for the endpoint...

Image by lincolnblues

Client-Side Mocking

- Leave our “birds” service out of it: have it work via an “api” service, which should handle mocking (and other things).
- Put the actual mock data into a “mockData” service or constant.
- Use HackStack.js for the common functionality.
- Put additional mocking logic in separate services.

Image by lincolnblues

The “Real” Service

```
.service('birds', function (api) {
  this.getAllBirds = function getAllBirds() {
    return api.getAll('birds');
  };

  this.getBird = function getBird(id) {
    return api.get('birds', id);
  };
});
```

The Mock Data: Keep It Separate

```
.constant('mockData', {
  birds: [
    {
      "id": 0,
      "name": "European robin",
      "scientificName": "Erithacus rubecula",
      "age": "55-60 million years",
      "img": “https://.../...jpg”
    }
  ],
  ...
});
```

Generating the Data

- http://www.json-generator.com/
- https://www.uuidgenerator.net/
- https://placekitten.com/

Image by lincolnblues

Implementing the Mockable API

```
.service('api', function(hackstack, mockData, ...) {
  ...
  var mockBirds = hackstack.mock(mockData.birds);
  $window.mockBirds = mockBirds;
```

☛ mockBirds is our mock endpoint.

Implementing the Mockable API, Cont.

```
this.getAll = function getAll(endpoint) {
  return getEndpoint(endpoint).getAll();
};

this.get = function get(endpoint) {
  return getEndpoint(endpoint).get();
};
```

☛ getEndpoint needs to return the right endpoint.

Implementing the `birds` service.

```
.service('birds', function (api) {
  this.getAllBirds = function getAllBirds() {
    return api.getAll('birds');
  };

  this.getBird = function getBird(id) {
    return api.get('birds', id);
  };
});
```

☛ Higher level services shouldn’t know about mocking.

Key Points

- Mock API needs to match the eventual “real” API.
- Which means it must be asynchronous.
- High-level code shouldn’t know about mocking.

Image by lincolnblues

Mock Likely Problems

- Slow connection. ✓
- Server-side errors. ✓
- Dropped connection.
- Loss of authentication.
- Values too short or too long.
- Etc.

Image by lincolnblues

Demo

Image by lincolnblues

When the API Arrives

Testing the API

- Test it with Postman.
- Maybe test it with `supertest`.

Image by torkildr

Dealing with Changes

- Run the API server locally.
- Control your schedule.
- Set up a toggle between client-side mocks and the real API.
- What about an ***incomplete*** API?

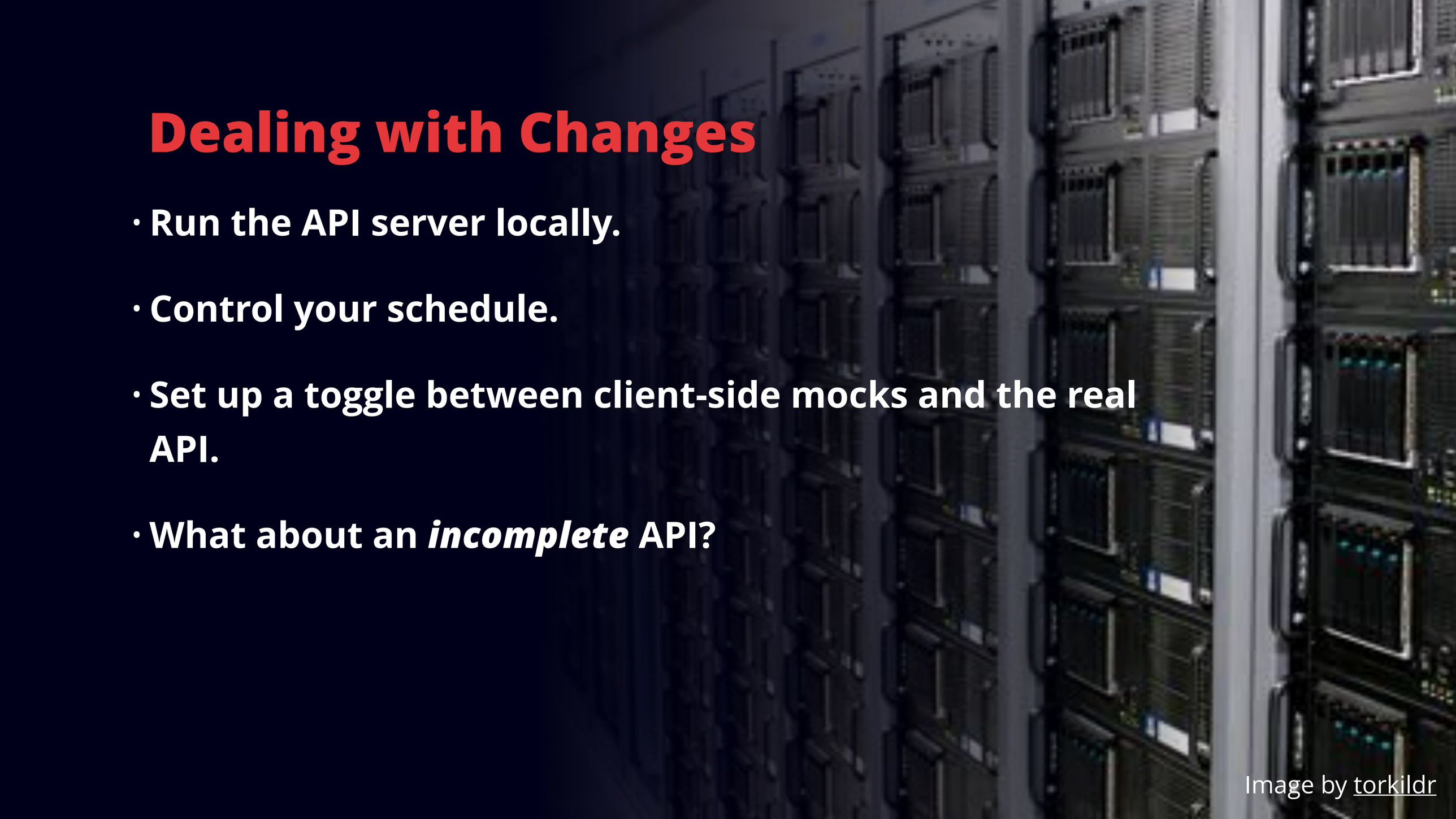

Image by torkildr

Working in the Hybrid Mode

- Mixing data from live API and mocks: `hackstack.wrap`.
- Filtering API data through a mock layer.
- Using a proxy server: e.g. for CORS.

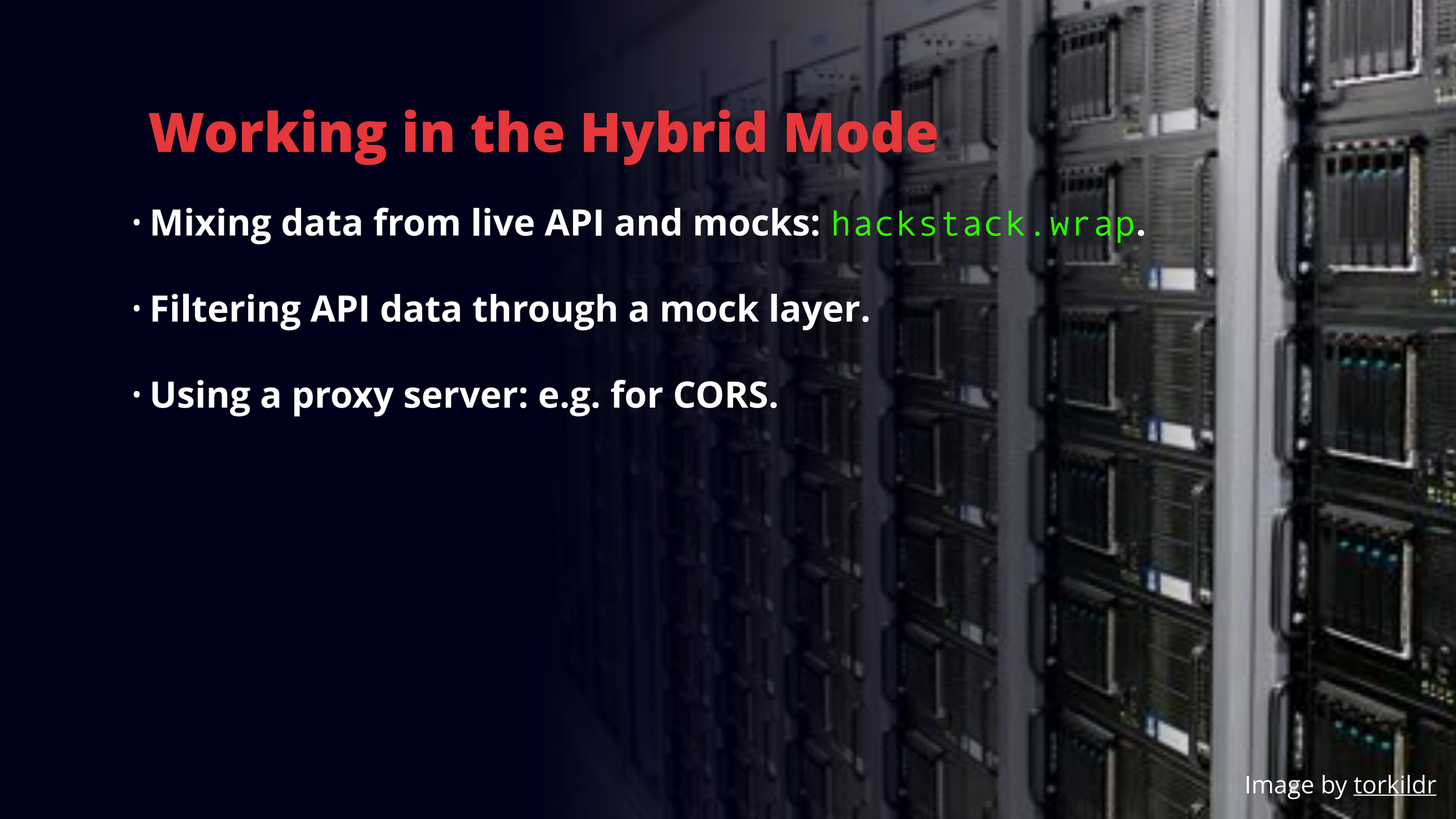

Image by torkildr

Mixing Real and Mock Data

```
var wrappedBirds = hackstack.wrap(birdsUrl,
  mockDataOverrides.birds);
```

Mixing Real and Mock Data

```
.constant('mockDataOverrides', {
  birds: {
    "name": "Lorem ipsum name",
    "scientificName": "Loremus ipsumus",
    "age": "~40 quadrillion years",
    "img": "images/rangle.jpg"
  }
});
```

Getting HackStack

Bower: `bower install angular-hackstack`

Github: https://github.com/rangle/hackstack

THANK YOU!

Yuri Takhteyev
CTO, Rangle.io

@qaramazov, @rangleio

https://github.com/rangle/hackstack

Image Credits

by torkildr

by fuzzyyol

by ejmc

by lincolnblues