https://github.com/mike-north/ember-pro

./EmberPro.pdf

ember PRO

Sept 5, 2017

Mike North

MIKE WORKS

What's this course all about?

- Patterns, not just concepts
- Ecosystem, not just framework
- Tons of ES2018 & some Typescript …
- Focus on:
  - Tools & Debugging
  - Managing Complexity
  - Easy-mode Architecture

Building & Booting

Professional Patterns

Customizing CLI

Amazing Addons

You'll walk away with an in-depth understanding of...

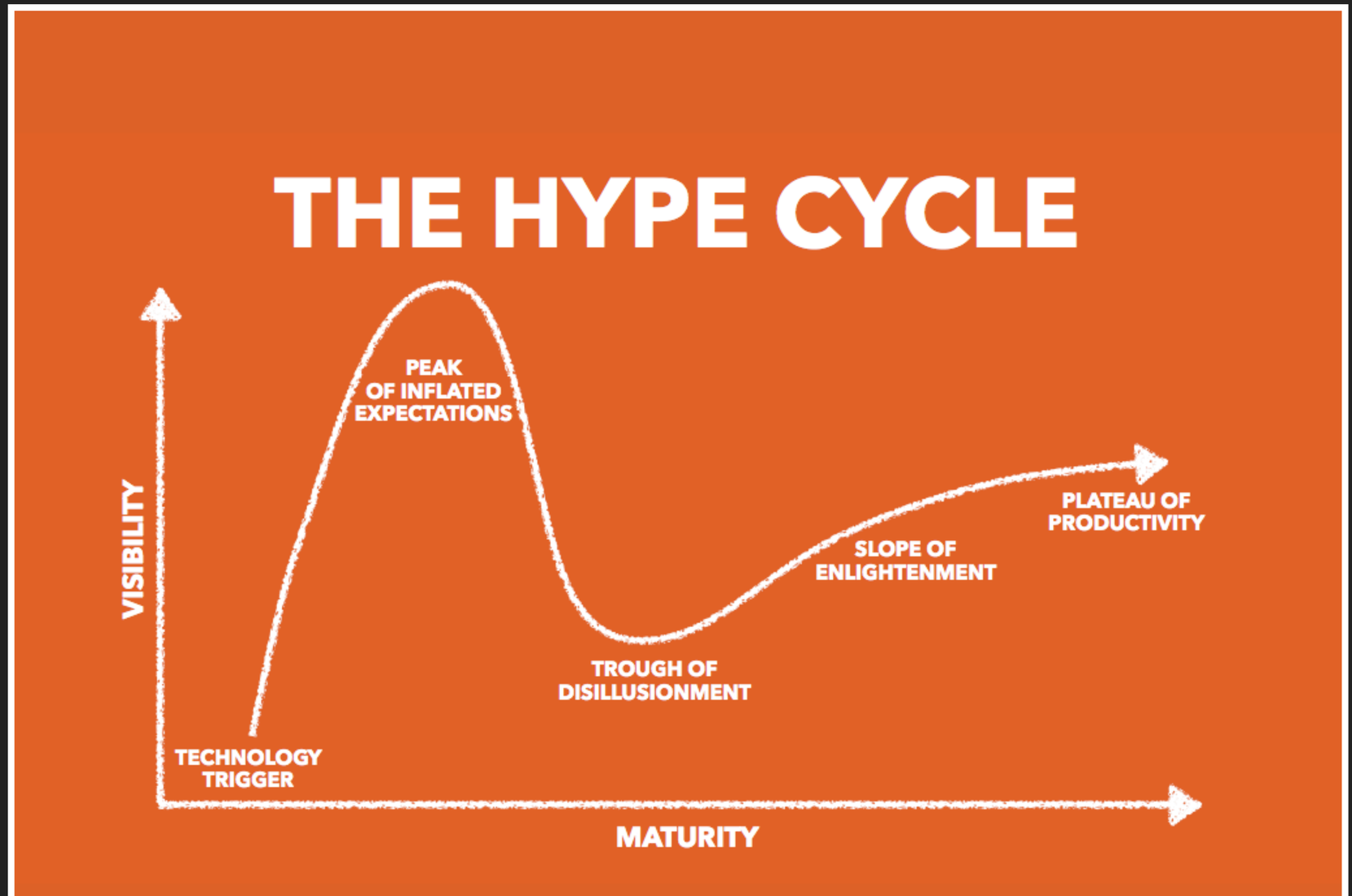
- Broccoli: the asset pipeline
- Ember-cli blueprints
- Build process
- Boot process
- Ember addons
- Debugging like a pro
- CRUD routes & actions
- Useful ES6 in Ember
- Server-side Rendering
- Modular app architecture
- Managing State
- Popular Addons

You'll walk away with a quick look at...

You'll walk away with...

A start on leading a large, complex ember project

Ember's Place among JS Frameworks



Ember ~~MAGIC~~

Automation

Container

A collection of an app's important long-living objects, organized by key

"route:index": Class,
"router:main": Class,

Core Concept

Why do we have a container?

- Keep singletons around
- Refer to important objects by name, keeping code loosely coupled
- Laziness (the awesome kind) & Automation
- App, Engine and Addon-level Encapsulation

Common Container Problems

- Empty or nonexistent component

```
▶ Uncaught Error: Attempting to inject an unknown injection: 'service:auth'(…)
```

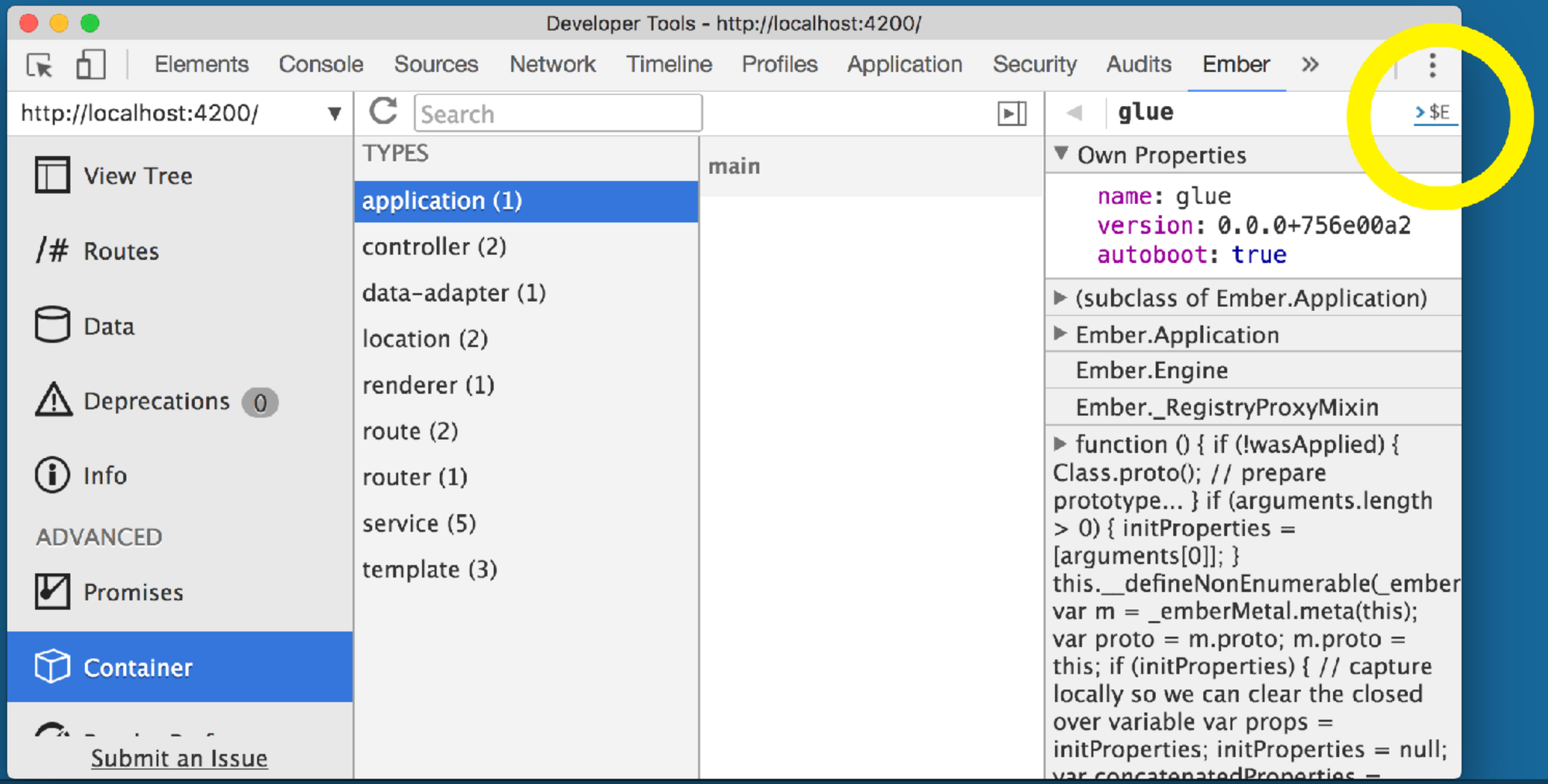
- Custom module behavior is ignored
- Addon modules helpers aren’t available to consuming app

```
▶ Uncaught Error: Compile Error: titlecase is not a helper(…)
```

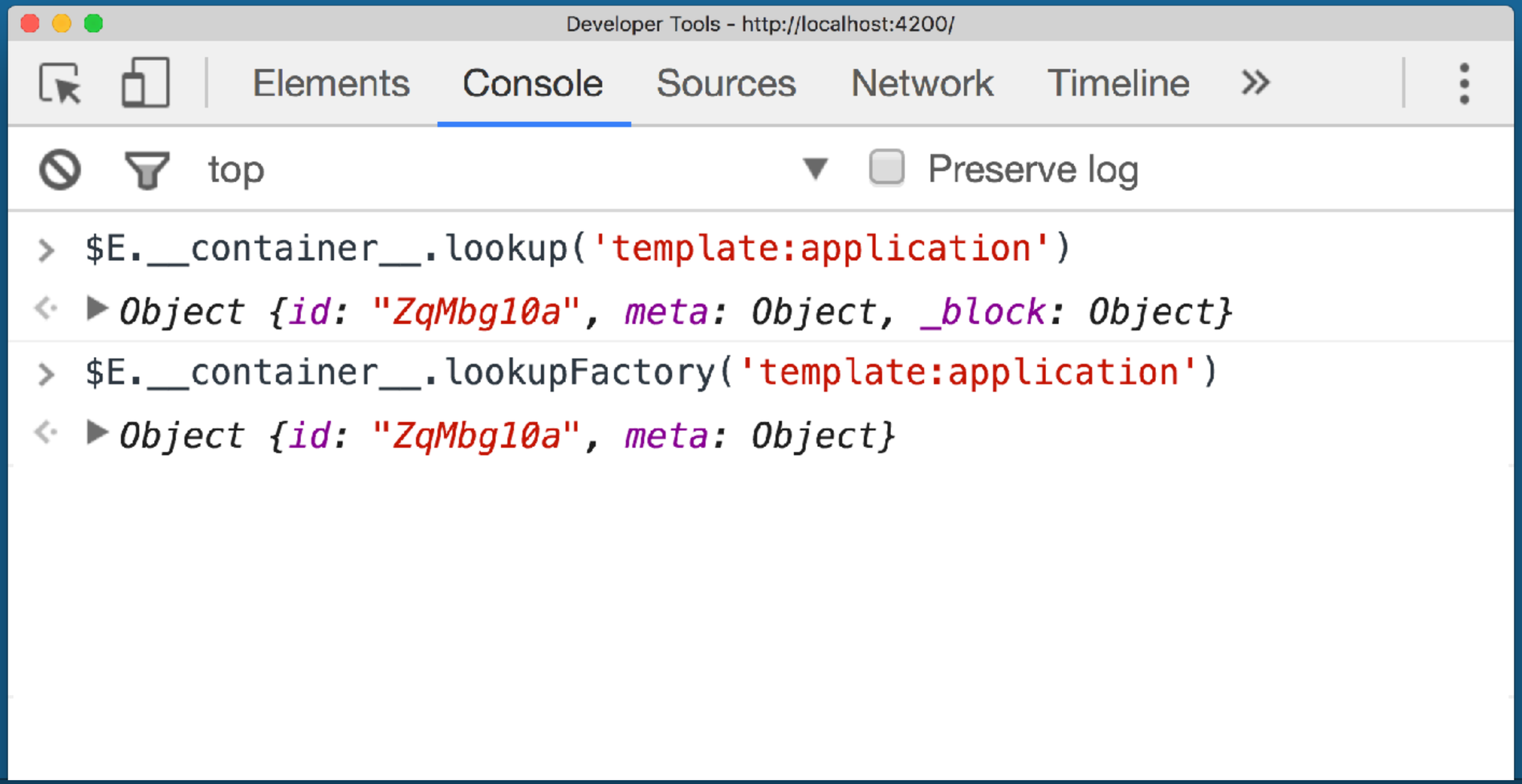
Container Best Practices

- Let Ember manage it. Don't touch it directly
- Little reason to modify it after boot completes
- Always use `Ember.getOwner(this)`
- Great place for "config" objects

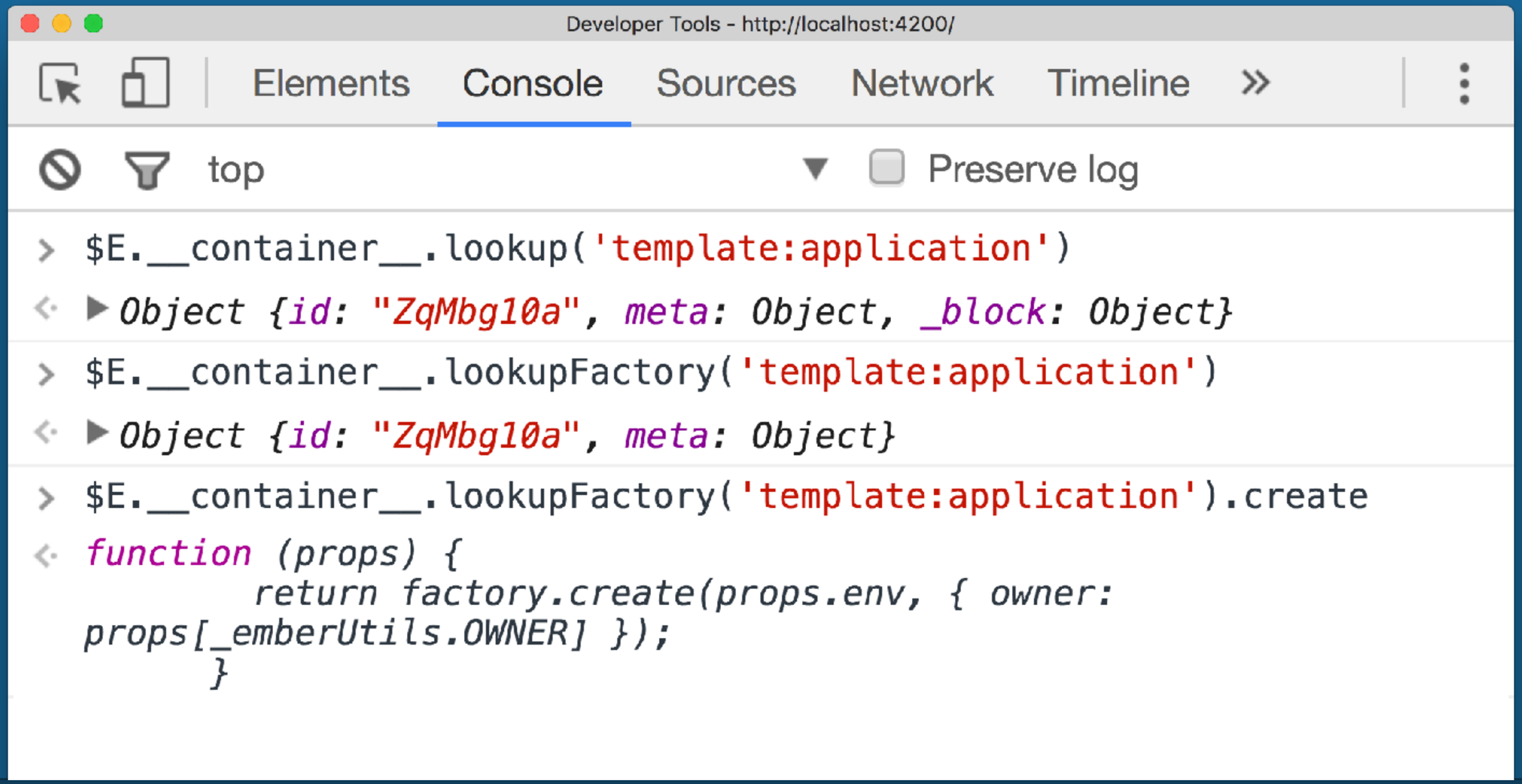
Debugging: Container



Debugging: Container



Debugging: Container



Resolver

Given a container key, Attempts to find the appropriate JS module in your codebase

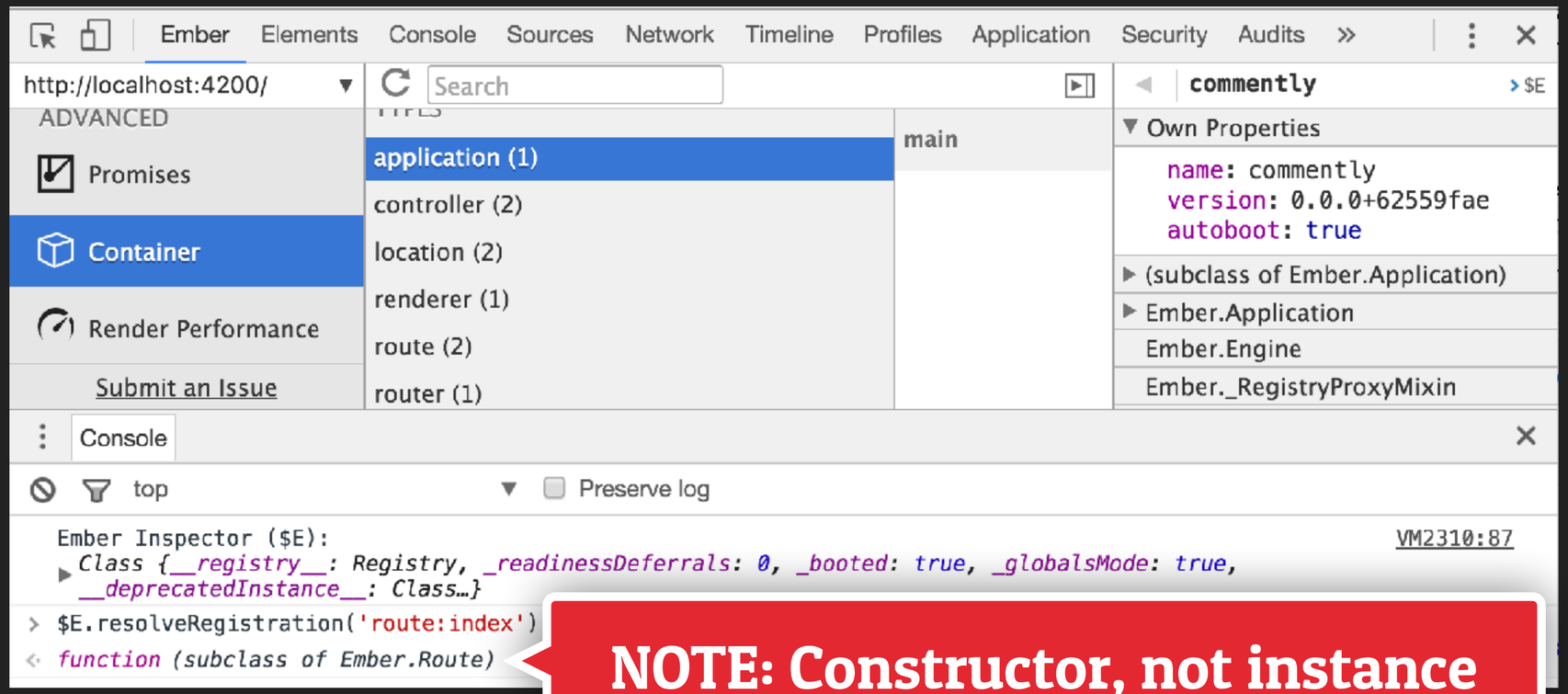
Why do we have a Resolver?

- Many ways for us to organize our code

```
// Globals
Ember.PostsController = ...

// AppKit, ember-cli
controllers/posts.js

// (old) Pods
posts/controller.js
```

How do we play with it?

- Ember.Application instance has `resolveRegistration()`



One step deeper: Defining and loading modules

```
define('mike', ['ember'], function(em) {
  const Ember = em.default;
  console.log(Ember.VERSION);
  return {
    default: {
      hello: "world"
    }
  };
});

const myModule = require('mike');
```

Bringing them into your asset pipeline

ember-cli-build.js

```
app.import('vendor/mike.js', {
  exports: ['mike']
})
```

Peeking Into Loaded Modules

```
> require._stats
◀ ▶ Object {define: 465, require: 15, reify: 139, findDeps: 140, modules: 465…}
> require.entries
◀ ▼ Object i
    ▶ commently/app: Module
    ▶ commently/components/-lf-get-outlet-state: Module
    ▶ commently/components/illiquid-model: Module
    ▶ commently/components/liquid-bind: Module
    ▶ commently/components/liquid-child: Module
    ▶ commently/components/liquid-container: Module
```

Add an ES6-ified module

1

- Your boss wants you to to ES6-ify some Math.* utilities
- For now, you can write code in app.js
- In the end, I should be able to do this:

```
import { default as math, PI } from 'math';

math.sqrt(4); // 2
console.log(PI) // 3.145926 ...
```

Booting Up

```
Ember.Application
```

Booting Up


```
Ember.Application#_bootSync {
}
```

Booting Up

domReady

```
Ember.Application#_bootSync {

    this.runInitializers();

}
```

```
Array[11]
0: "App Version"
1: "container-debug-adapter"
2: "data-adapter"
3: "ember-data"
4: "export-application-global"
5: "injectStore"
6: "startup"
7: "store"
8: "transforms"
9: "domTemplates"
10: "deferReadiness in `testing` mode"
```

Initializer

A function that's invoked while your App boots up.

Initializers

-  Modular boot process

Initializers

```
import Lemon from 'fruits/lemon';

export function initialize(app) {
  app.register('fruits:lemon', Lemon);
}

export default {
  name: 'my-initializer',
  after: 'other-initializer',
  initialize
};
```

Initializers

- 🍱 Modular boot process
- 🕐 Happens **WHILE** Application boots
- ⌛ `deferReadiness`, `advanceReadiness`

Why do we have these things?

- Application-level, pre-boot setup/constructor logic
- Registration API (container stuff)
- Use cases:
  - Add a configuration object to the container
  - Setup for a/b testing
  - Conditionally load code

Booting Up


```
Ember.Application#_bootSync {

    this.runInitializers();
    this.advanceReadiness();

}
```

Booting Up


```
Ember.Application#_bootSync {

    this.runInitializers();

    this.advanceReadiness() {

        this.didBecomeReady();

    }
}
```

Booting Up

```
Ember.Application#_bootSync
  Ember.Application#didBecomeReady {
  }
```

Booting Up

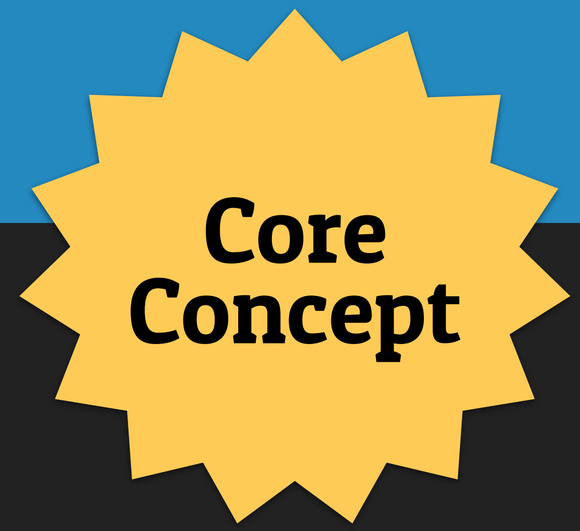
```
Ember.Application#_bootSync
    Ember.Application#didBecomeReady {
        if (this.autoboot) {
            // Instantiate a Ember.ApplicationInstance
            let instance = this.buildInstance();
            instance._bootSync();

            instance.startRouting();
        }

    }
```

Booting Up

```
Ember.Application#_bootSync
    Ember.Application#didBecomeReady
      Ember.ApplicationInstance#_bootSync  {
        this.setupRegistry();
        // define root element
        // define location
        this.runInstanceInitializers();
        if (isInteractive) {
          this.setupEventDispatcher();
        }
      }
```

Instance Initializer

A function that's invoked immediately after your App boots up.



Instance Initializers

- 🍱 Modular boot process
- 🕐 Happens **AFTER** ApplicationInstance boots
- ♻️ Re-runs on Fastboot builds
- 💡Think: Container, not Registry

Instance Initializers

- All the first application-instance initializer is run after the last application initializer
- Access to a fully materialized container, store, etc…
- Cannot defer/advance readiness

Registration API

- In an initializer, you're primarily dealing wth factories, and app.register and app.resolveRegistration
- In an instance-initializer, you're guaranteed a container. app.lookup is now available — this is for instances!

Registration API

```
import Lemon from 'fruits/lemon';

export function initialize(app) {
  app.register('fruits:lemon', Lemon);
}

export default {
  name: 'my-initializer',
  after: 'other-initializer',
  initialize
};
```

Registration API

- An app instance allows registration of anything in the container
- .create() is called on container items when looked up
- Same instance is reused over and over
- An optional third argument allows two options for deviating from that behavior

```
{   singleton: true, // re-use the same instance
  instantiate: true } // call Ember.Object.create when looked up
```

Booting Up

```
Ember.Application#_bootSync
    Ember.Application#didBecomeReady
      Ember.ApplicationInstance#_bootSync  {

        this.setupRegistry();
        // define root element
        // define location
        this.runInstanceInitializers();
        if (isInteractive) {
          this.setupEventDispatcher();
        }
      }
```

Booting Up

Ember.Application#_bootSync

Ember.Application#didBecomeReady {

```
if (this.autoboot) {
  // Instantiate a Ember.ApplicationInstance
  let instance = this.buildInstance();
  instance._bootSync();

  instance.startRouting();
}
```

}

Booting Up

```
Ember.Application#_bootSync
    Ember.Application#didBecomeReady
        Ember.ApplicationInstance#startRouting
            Ember.Router#startRouting {
                let initialURL = location.getURL();
                let initialTransition = this.handleURL(initialURL);
            }
```

Application

vs

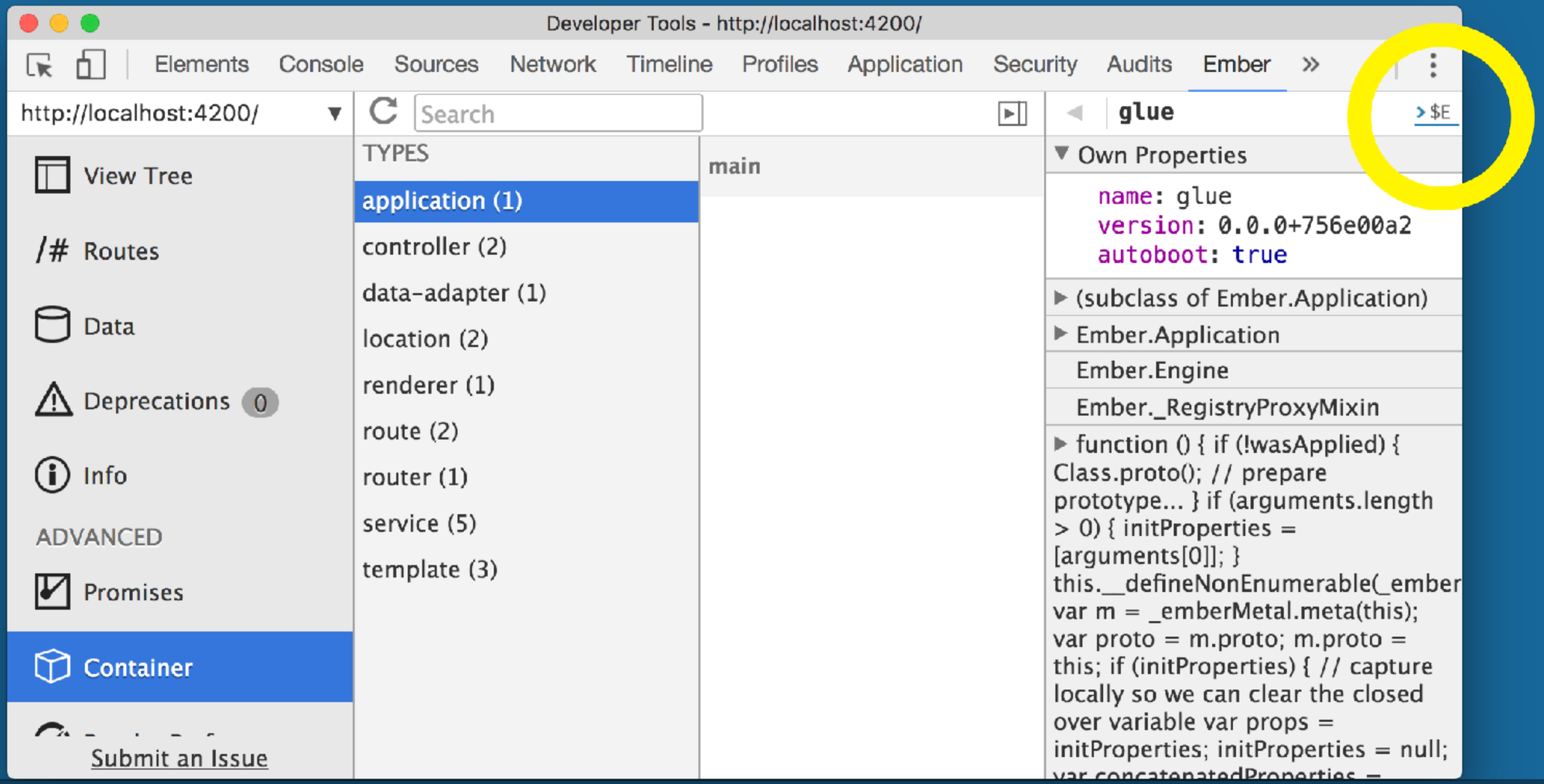
ApplicationInstance

Initializer

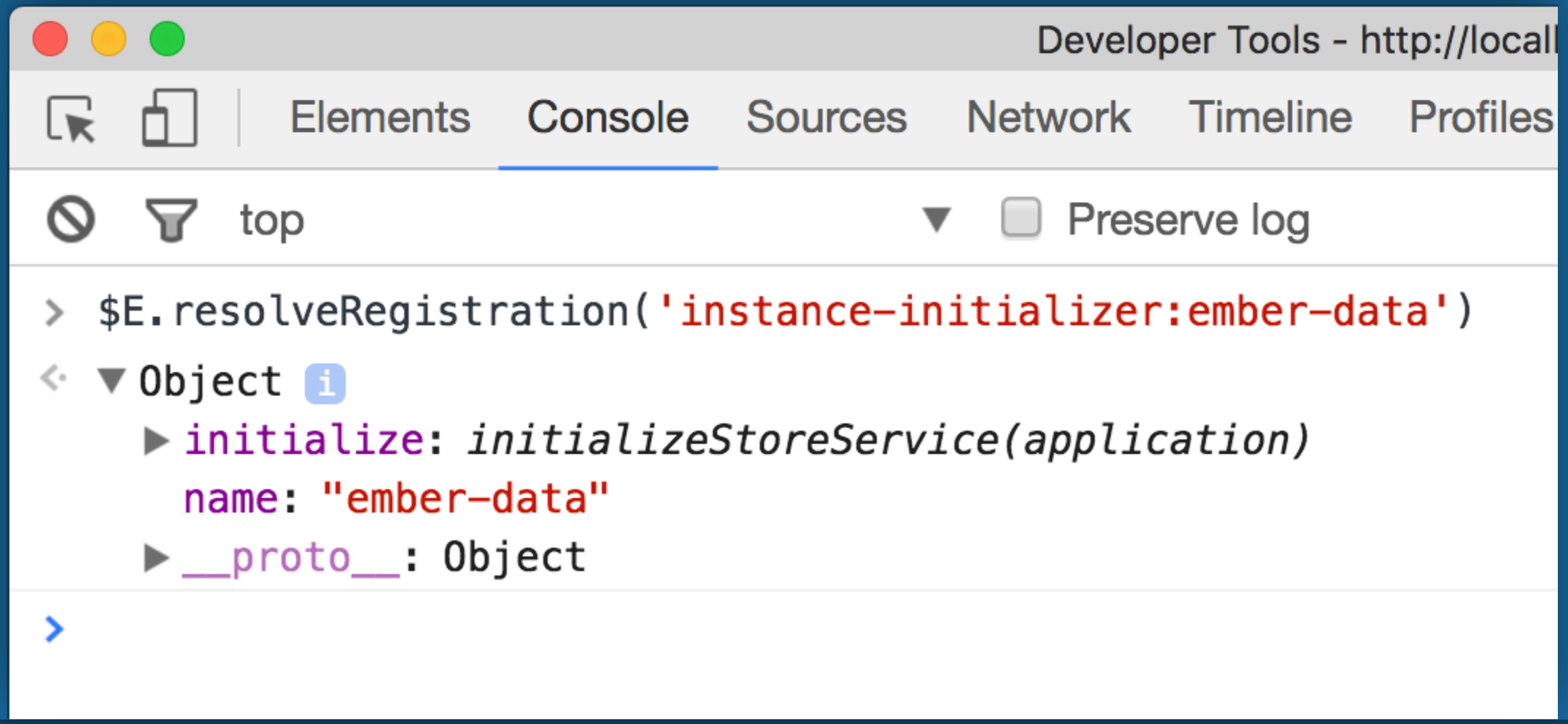
vs

InstanceInitializer

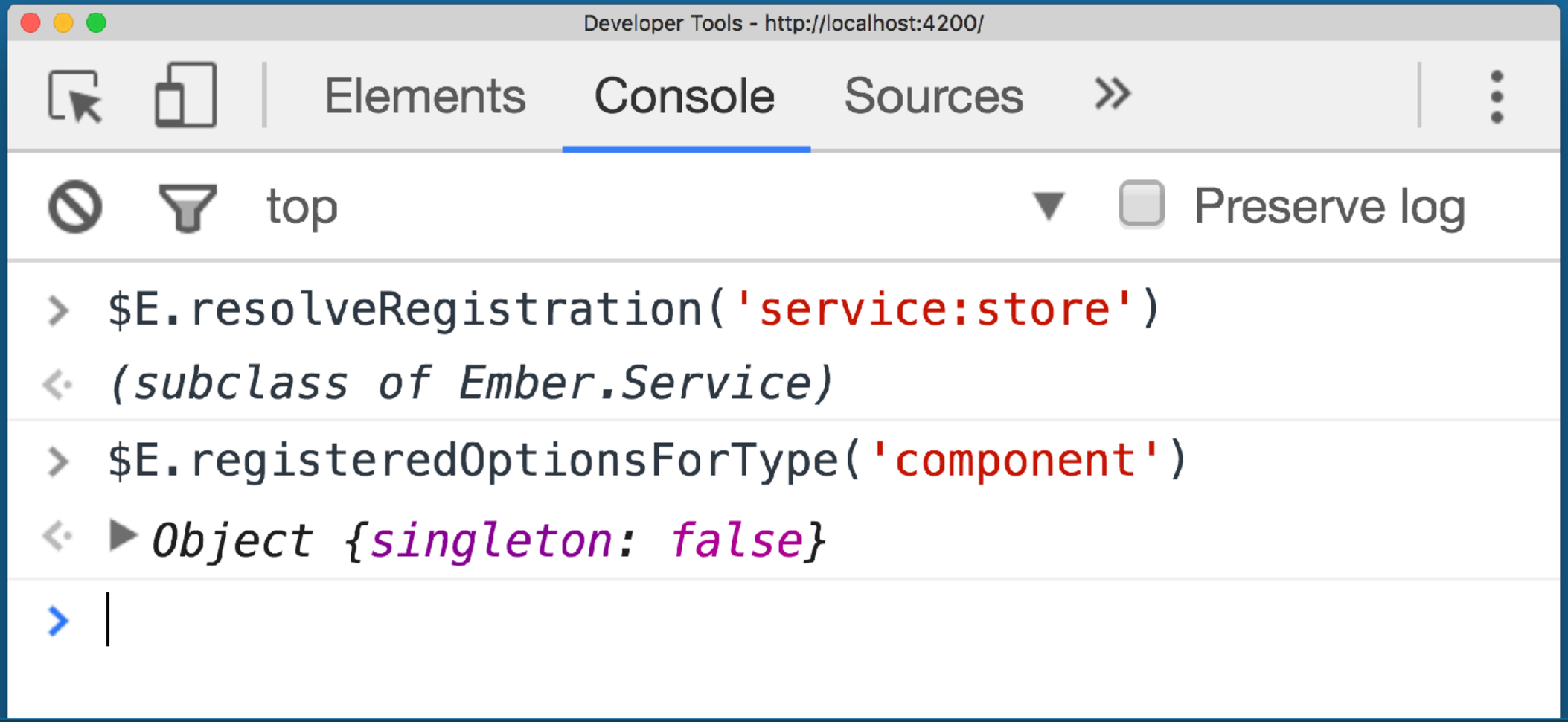
Debugging: Registry



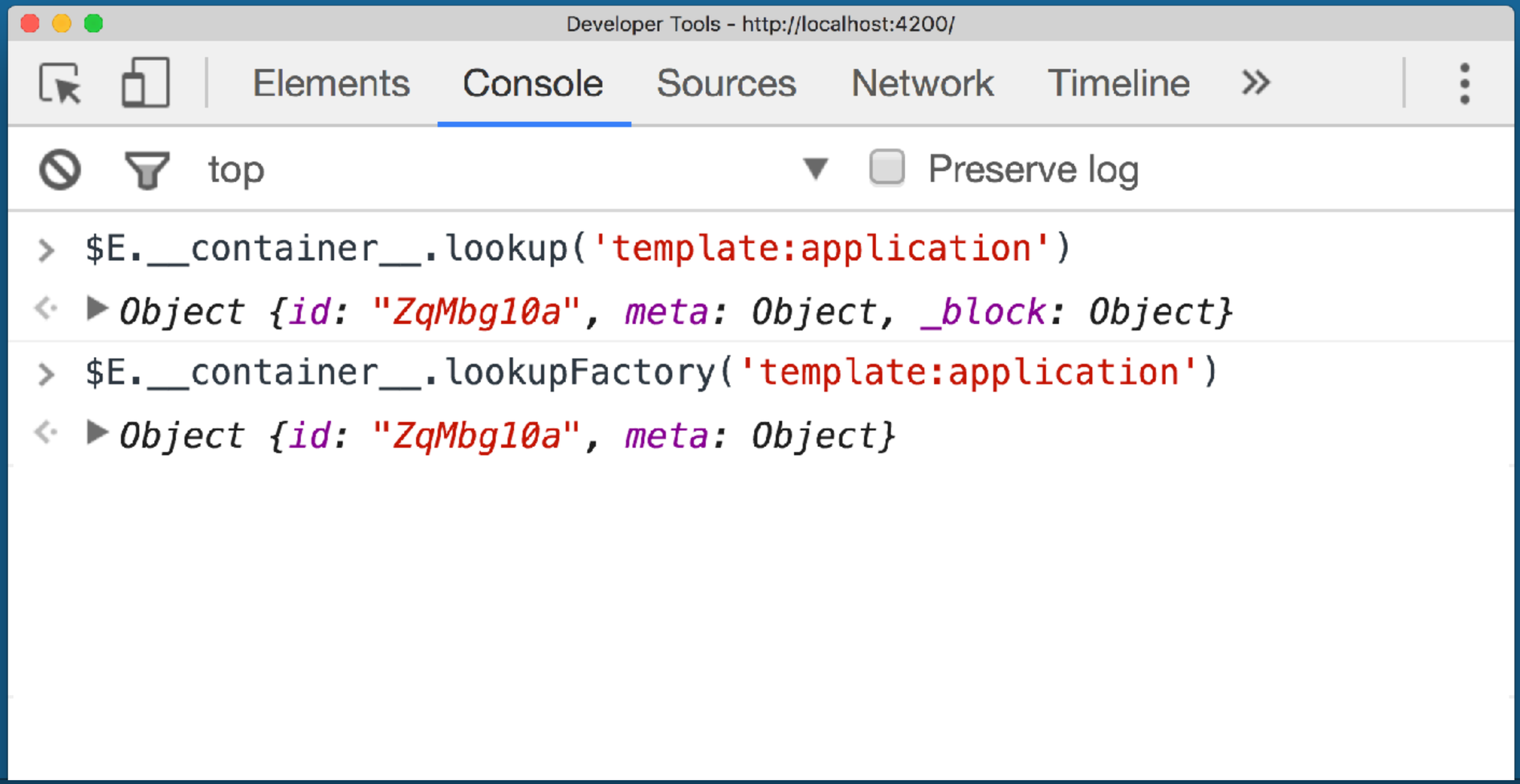
Debugging: Registry



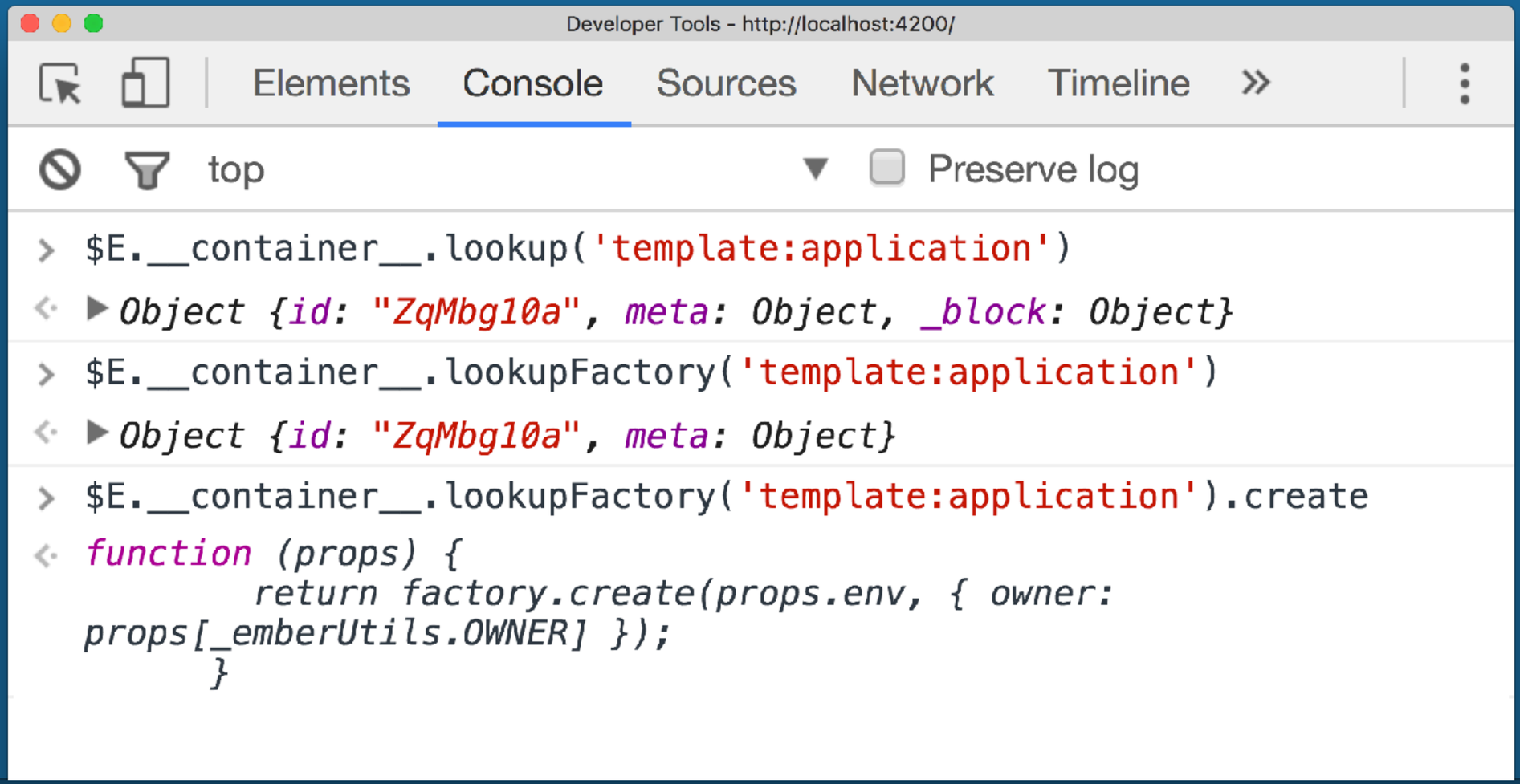
Debugging: Registry



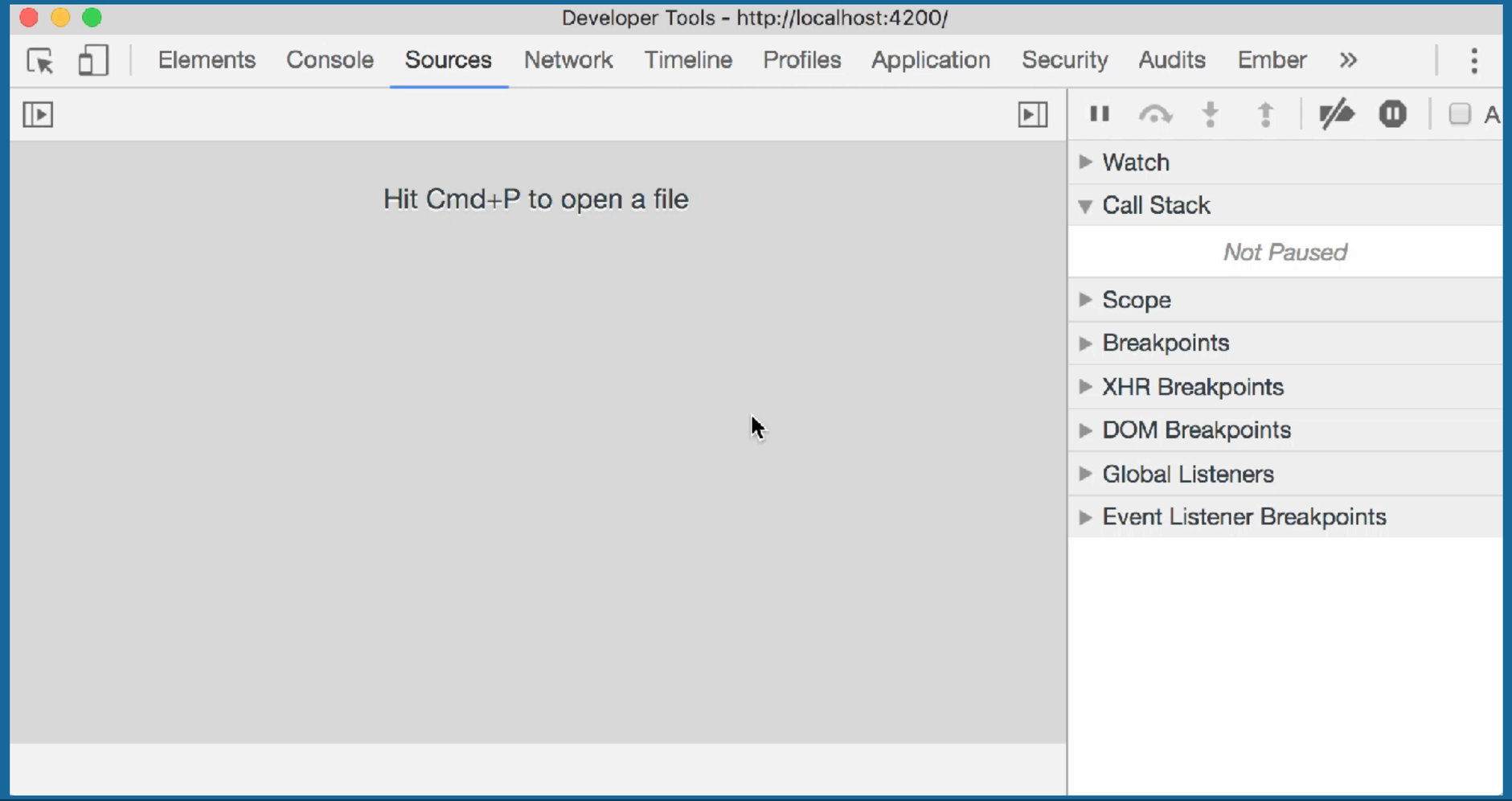
Debugging: Container



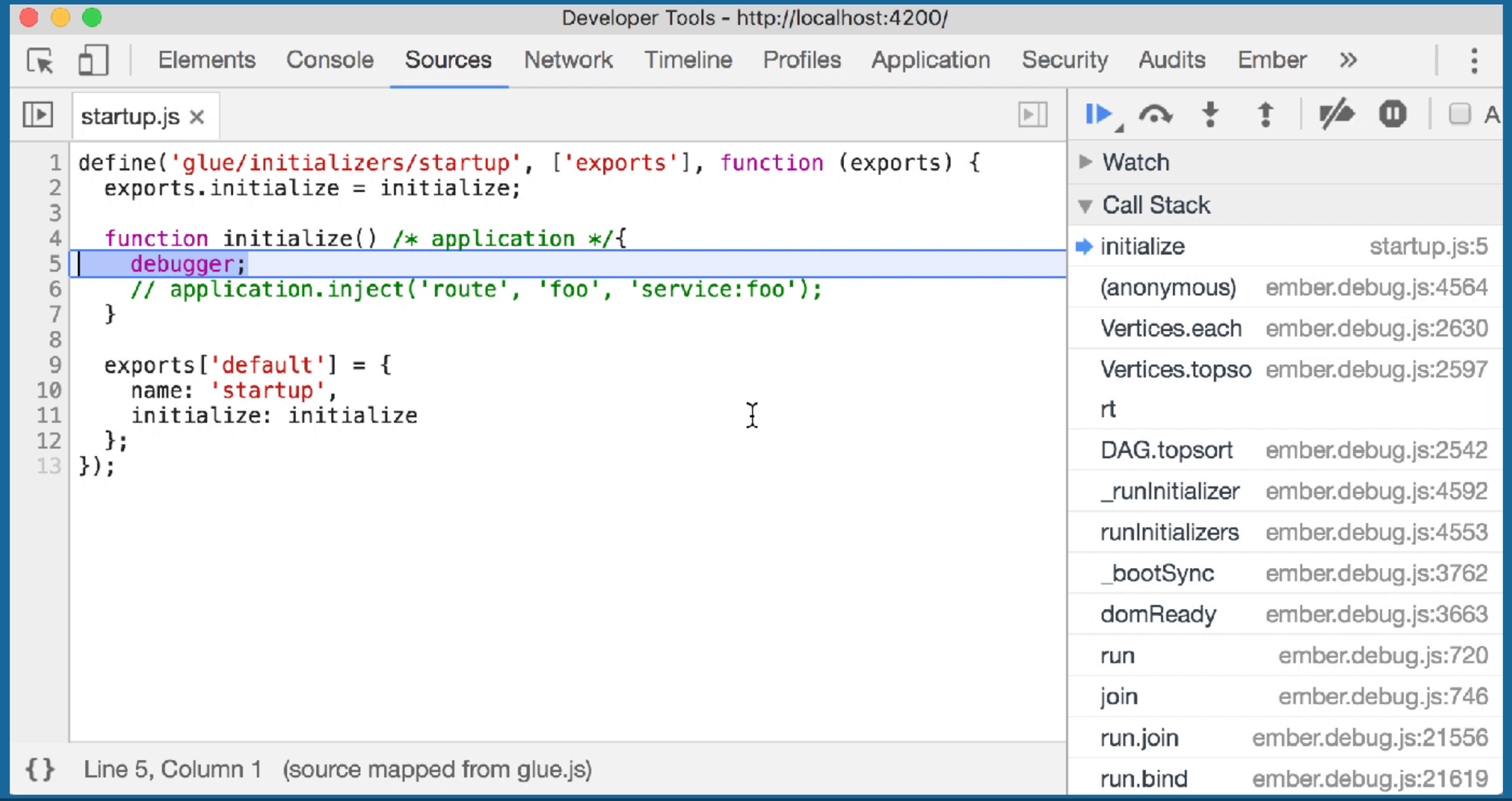
Debugging: Container



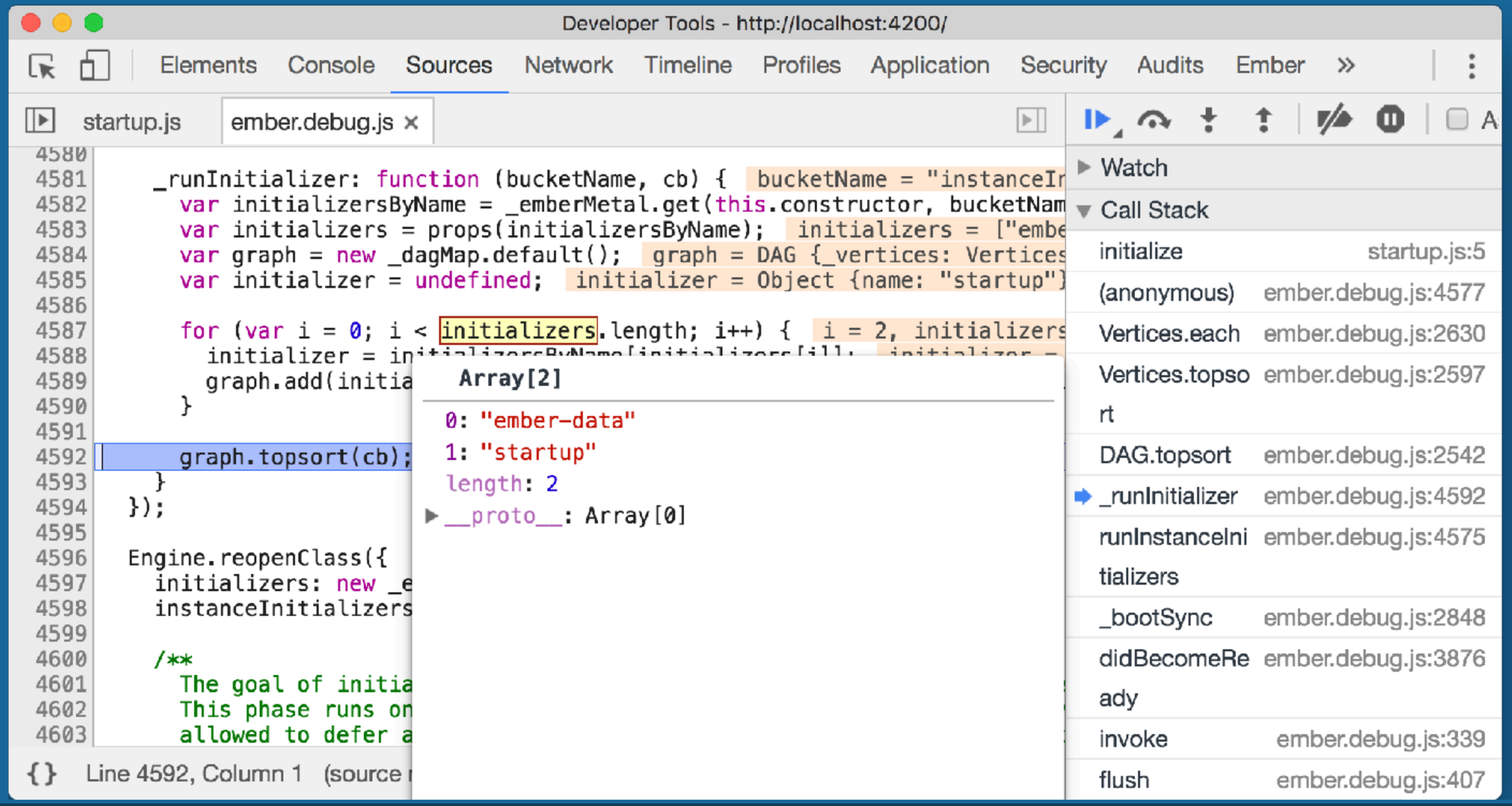
Debugging: Initializers



Debugging: Initializers



Debugging: Instance Initializers



Slow-lo-mo

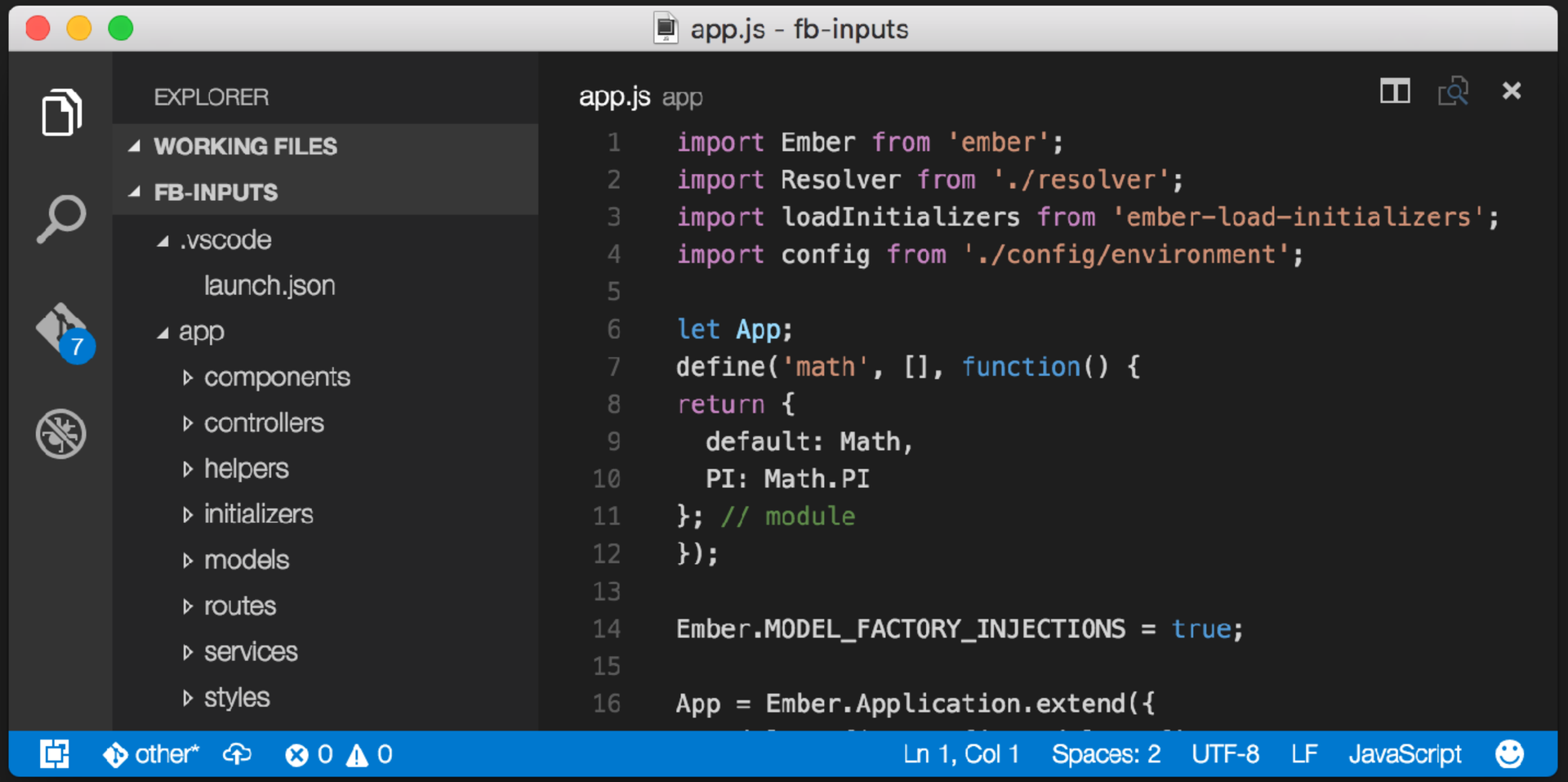
2

- Request access to geolocation data, and ensure your app doesn't start up until you have it
- Store the location data in your container, using the key "data:location" and value {lat: ..., lng: ...}
- In your application route, give the controller access to the location data, so that it can be rendered in the application.hbs template
- ensure that things are working, using ember inspector

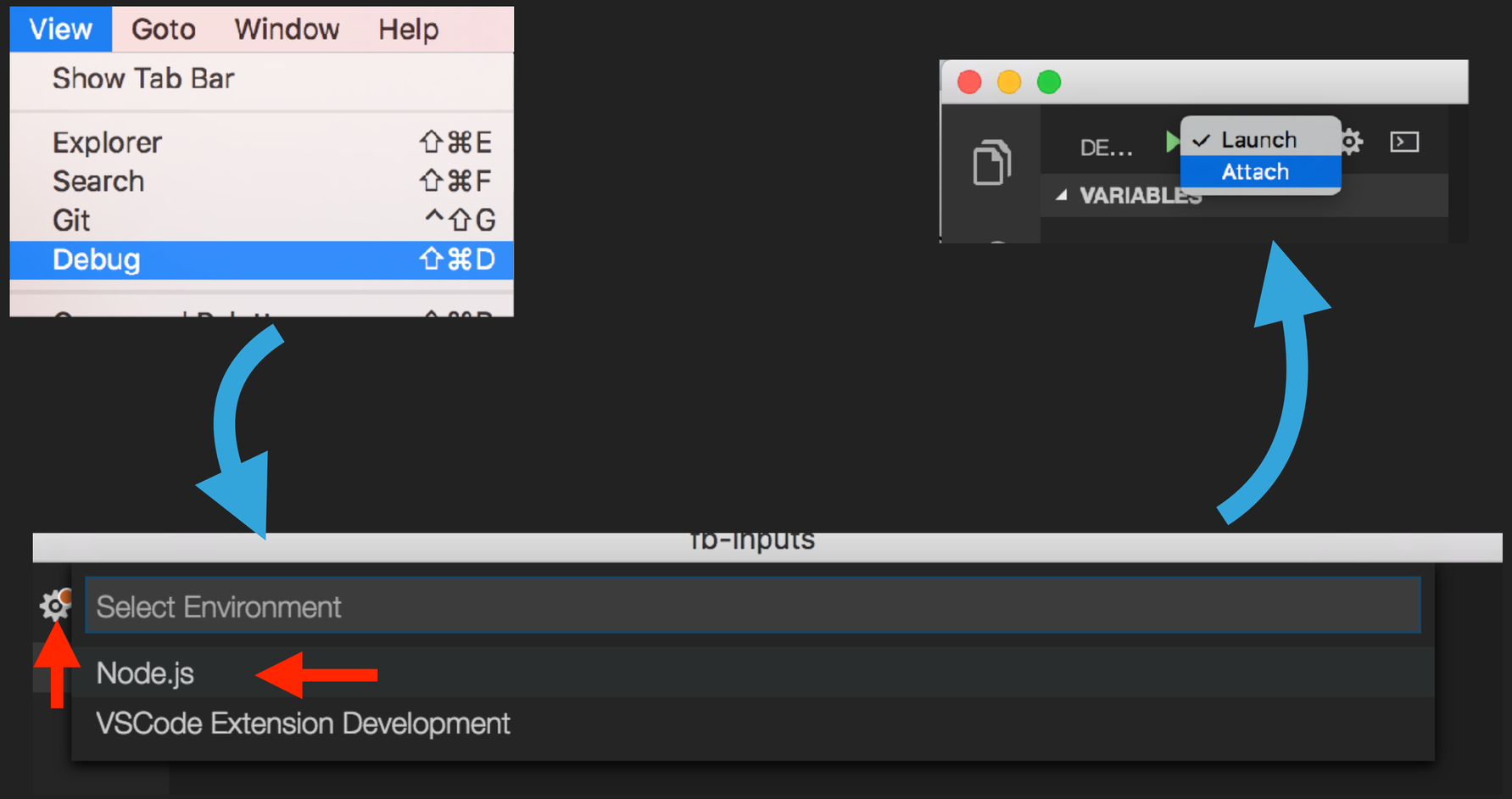
```
const { geolocation } = navigator;
geolocation.getCurrentPosition((pos) ⇒ {
  let {
    coords: { latitude, longitude }
  } = pos;
  do_something(latitude, longitude);
});
```

Building

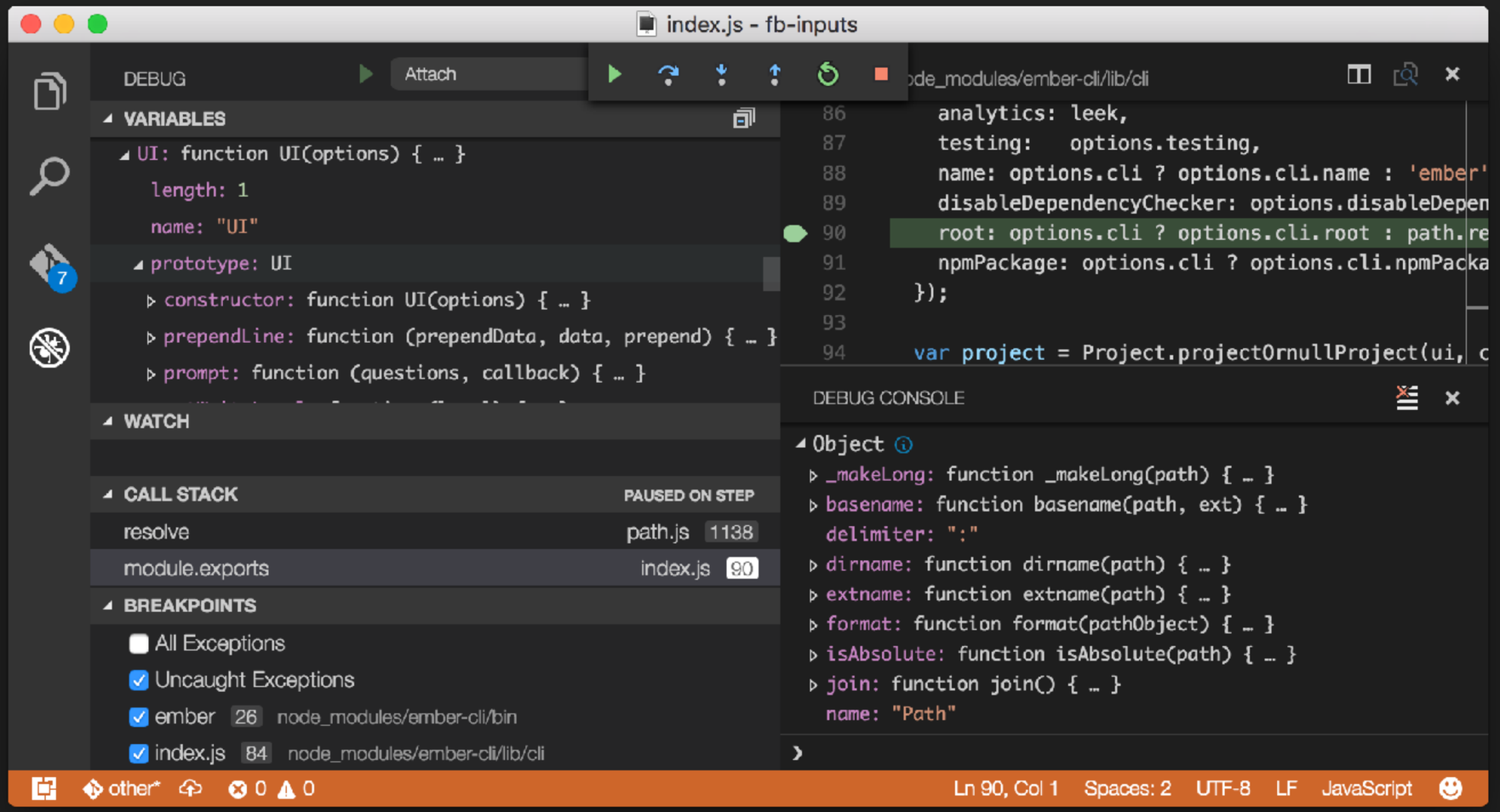
Debugging Node.js With VS Code



Debugging Node.js With VS Code



Debugging Node.js With VS Code



Broccoli - A Fast Asset Pipeline

- Built on node's fs module
- Cache intermediate build results
- Think trees, not files
- Plugins are chainable

Trees & Plugins

- A string is a tree
- Plugins are either N:N or N:1
- TIP: Act like trees are immutable

🚫

```
var appTree = app.toTree();
minify(appTree)
return appTree;
```

✅

```
return new Minify(app.toTree());
```

Example Plugin:

```
function MyFilter(inputNode) {
  Filter.call(this, inputNode);
}

MyFilter.prototype = Object.create(Filter.prototype);

MyFilter.prototype.processString = function(existingString) {
  return 'newString';
};

MyFilter.prototype.extensions = ['coffee'];
MyFilter.prototype.targetExtension = 'js';
```

Debugging Broccoli Chains

You're assembling a chain of functions, evaluated much later.

STEFANPENNER/BROCCOLI-STEW

- `env` - Conditionally runs a callback based upon the current environment.
- `mv` - Moves an input tree to a different location.
- `rename` - Renames files in a tree.
- `find` - Match files in a tree.
- `map` - Maps files, allow for simple content mutation.
- `log` - Logs out files in the passed tree.
- `debug` - Writes the passed tree to disk at the root of the project.
- `rm` - Remove files from a tree.
- `beforeBuild` - Calls a callback function before the tree is read.
- `afterBuild` - Calls a callback function after the tree is read.
- `wrapBuild` - Calls callback functions before and after the tree is read.
- `npm.main` - Create a tree with Node module's main entry

Debugging Broccoli

- npm install –save-dev broccoli-stew

```
var debugTree = require('broccoli-stew').debug;

return new MyPlugin(
  debugTree(
    app.toTree(), {name: 'my-tree'} // Write to this folder
  )
);
```

Cooking Up Some 

3

- I want a copyright notice at the top of each css/js file, listing 🦄🦄🦄🔫🌈🍺🍺 as the author and copyright holder.
- The comment should also include a timestamp so we know when these assets were built.
- Remove ember-cli-sri from package.json

```
/**
 * vendor.js
 *
 * (c) 2016 🦄🦄🦄🔫🌈🍺🍺 All Rights Reserved
 * generated at: 2016-12-06T19:20:23.171Z
 */
```

Building & Deploying - Ember-App Options

- Many broccoli plugins act on options passed to the EmberApp constructor
- If you run ember build –prod, your files will be fingerprinted
- If we want a staging environment, we'll need to customize things a bit.

Ember-App Options

```
var app = new EmberApp({
  fingerprint: {
    exclude: ['fonts/169929'],
    prepend: 'https://sudomain.cloudfront.net/'
  },
  minifyJS: {
    enabled: false
  },
  minifyCSS: {
    enabled: false
  }
});
```

Per-Environment Configuration

- `config/environment.js` is where most per-env switching takes place
- baked into the build, via meta tag
- module is available for client and server-side stuff
- many non-build-related ember addons will be customized in this file

Automated Testing

- When building/testing, it's just like any node app
- Already set up for travis-ci w/ PhantomJS 2.1
- Addons are easily tested against multiple sets of npm/bower dependencies

Travis-CI

Test with Chrome instead of PhantomJS

```
ember g travis-chrome
```

Circle-CI

Test with Phantomjs 2.1

```
ember g circle-ci-phantom
```

Test with headless Chrome

```
ember g circle-ci-chrome
```

Heroku Deployment

```
# create a new app
heroku create
# setup the ember buildpack
heroku buildpacks:set \
    https://codon-buildpacks.s3.amazonaws.com/buildpacks/heroku/emberjs.tgz
# push to the heroku git remote
git push heroku master
```

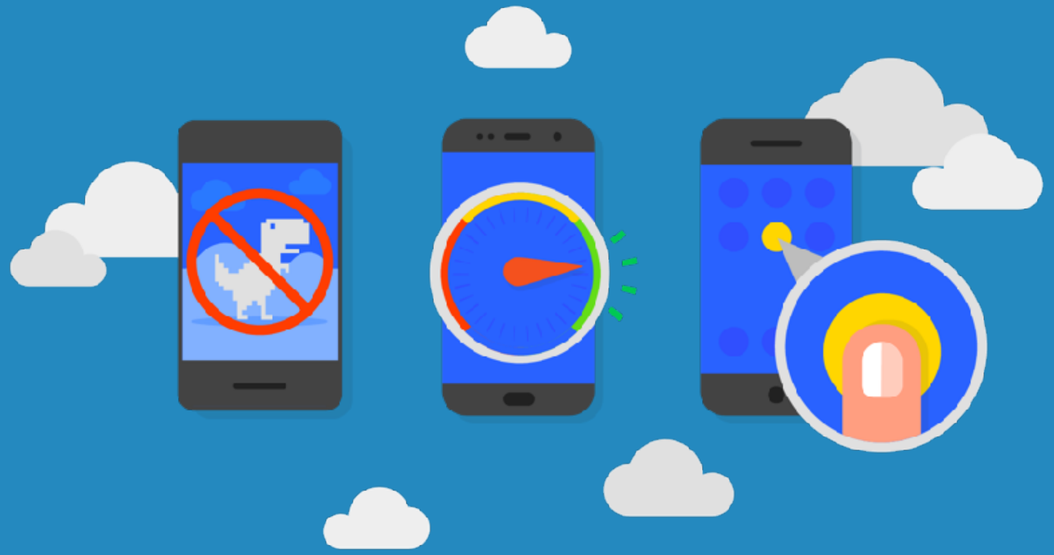
Deploy to Staging

4

- We know how to test with travis-ci and deploy to heroku. Let's put it all together! Push a prod-like build to staging
- HINT: you can install the travis gem and run

```
travis setup heroku
```

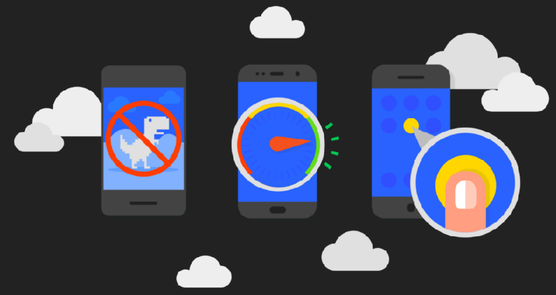
- Make sure anything secret (i.e., your Heroku API keys) are encrypted
- In the end, you should be able to push to GitHub, and test-passing code will deploy w/o human intervention



🏁 Fastboot & PWA

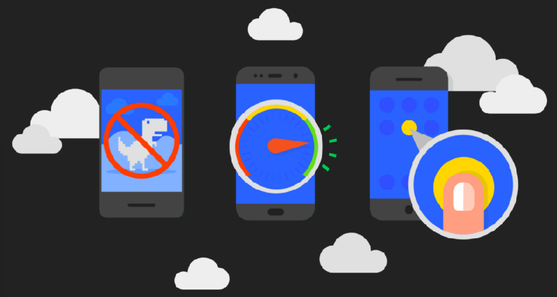
Normal apps, that become superheroes where possible

Progressive Web App Technologies



- Time to first meaningful paint
- Time to first interaction
- Mobile web metadata (theme color, browser UI removal, add to home screen, etc...)
- Resiliency to network quality via service worker
- SEO + Structured Data

Progressive Web App



www.yourwebsite.com

Lighthouse
Version: 1.2.2
BETA

- Progressive Web App
- Best Practices
- Performance Metrics

Results for: https://stage.mike.works/
Generated on: 12/7/2016, 10:39:33 AM CST

97
100

Progressive Web App

These audits validate the aspects of a Progressive Web App.

App can load on offline/flaky connections

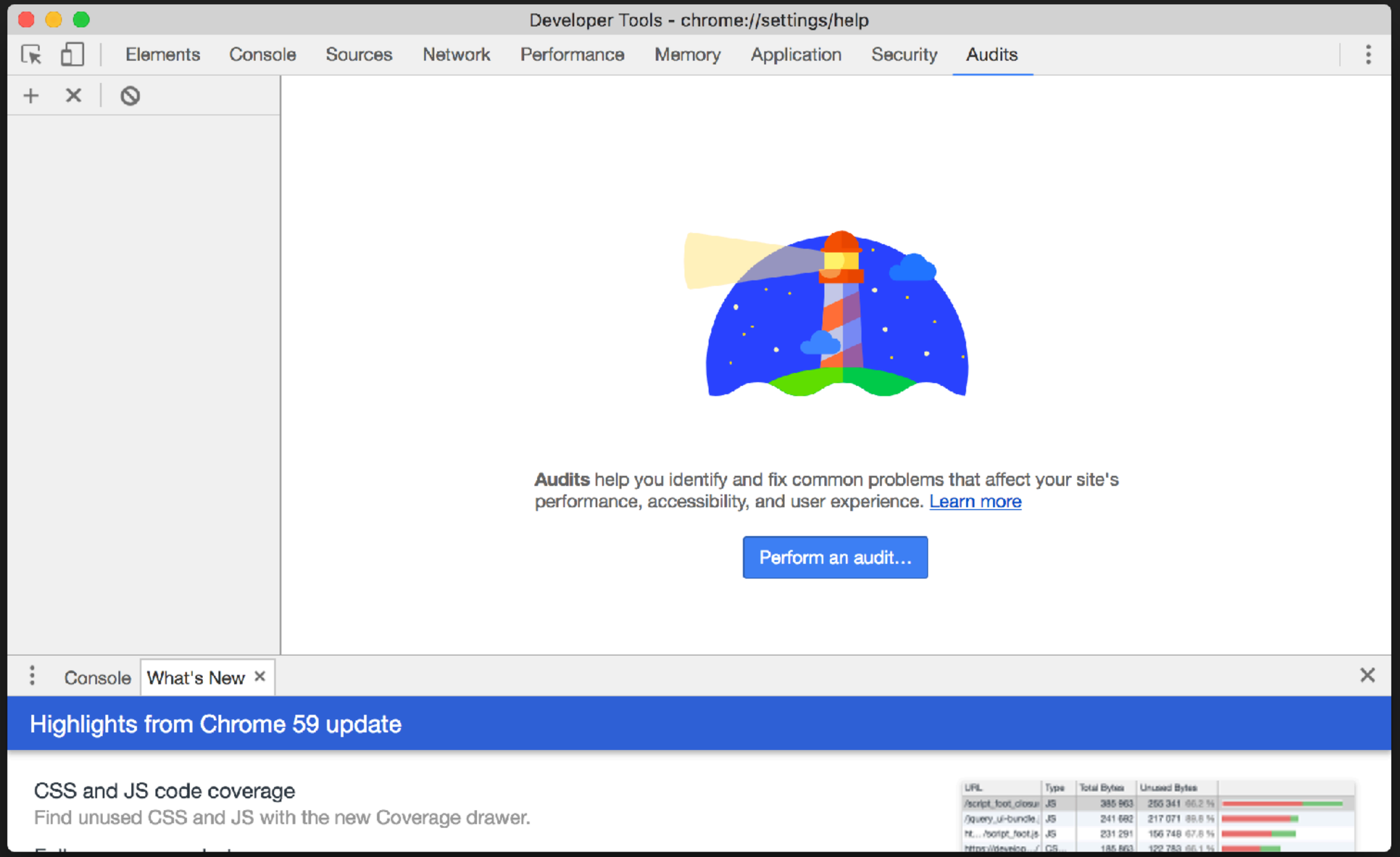
Ensuring your web app can respond when the network connection is unavailable or flaky is critical to providing your users a good experience. This is achieved through use of a Service Worker.

- Has a registered Service Worker
- URL responds with a 200 when offline

Page load performance is fast

Users notice if sites and apps don't perform well. These top-level metrics capture the most important perceived performance concerns.

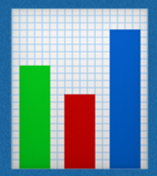
- 59 First meaningful paint: **3558.8ms** (target: 1,600ms)
  - First meaningful paint measures when the primary content of a page is visible. Learn more.
- 85 Perceptual Speed Index: **2622** (target: 1,250)
  - First Visual Change: **2326ms**
  - Last Visual Change: **6781ms**
- 92 Estimated Input Latency: **55ms** (target: 50ms)
- 72 Time To Interactive (alpha): **3564.8ms** (target: 5,000ms)



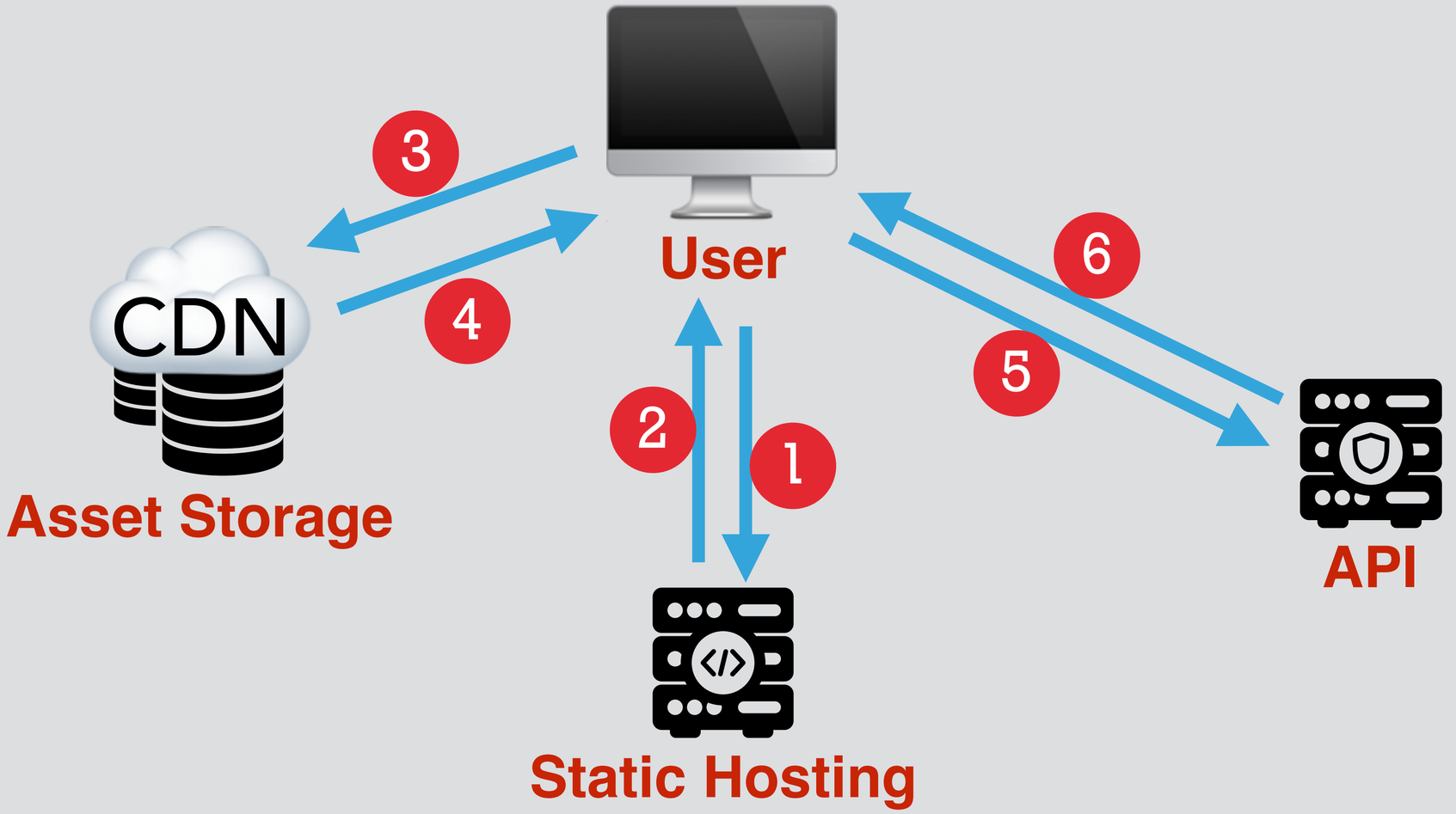
Client Rendered SPA

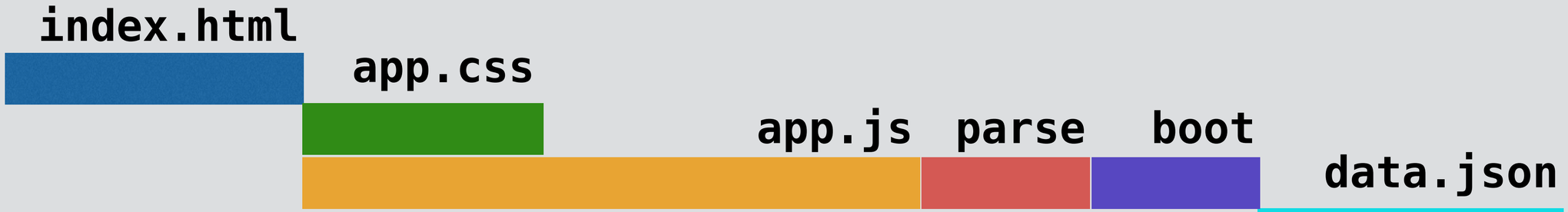
First Paint

Interactive



Data





Major Pieces

App-Like Characteristics

Pruning the Critical Path

Tolerating Network Instability

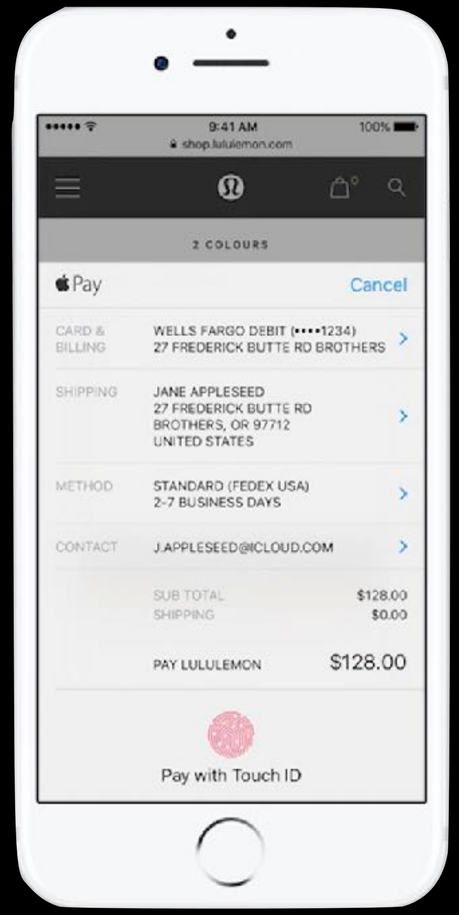
App-Like Characteristics

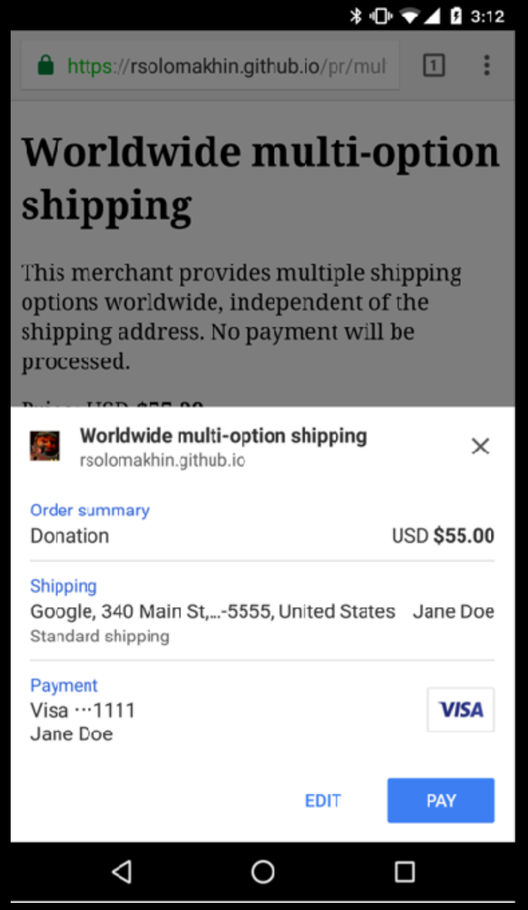
App-Like Characteristics

manifest.json

```
{
  "name": "Mike.Works - World Class Developer Training",
  "short_name": "Mike.Works",
  "start_url": ".",
  "display": "standalone",
  "background_color": "#fff",
  "theme_color": "#fff",
  "description": "World Class Developer Training",
  "orientation": "portrait-primary",
  "lang": "en-US",
  "icons": [ ]
};
```

App-Like Characteristics





In The Future…

- 💬 Push Notifications
- 🔁 Background Sync
- ⚛️ First-Class Framework Support
- 📦 IndexedDB 2.0
- ⚡ HTTP/2 Optimizations

App-Like Characteristics

Get rid of browser UI

```
<meta name="apple-mobile-web-app-capable" content="yes">
```

App Icons

```
<link sizes="180×180" rel="apple-touch-icon" href="icon.png">
```

Bookmark Name

```
<meta name="mobile-web-app-title" content="Mike.Works">
```

Splash Screen

```
<link href="splash.png" rel="apple-touch-startup-image">
```

TOOLS DEMO

https://github.com/mike-north/emberconf-2017-mobile

Looks Like an App


```
ember install ember-cli-head
ember install ember-cli-meta-tags
ember install ember-add-to-homescreen
```

- Open image-assets.zip, and find a bunch of icons
- Ensure the add-to-homescreen behavior works as expected, and presents appropriate icon
- Strip away all browser UI you can for bookmarked launch
- Zooming behavior should match native apps
- Create a manifest.json, and follow all lighthouse advice

Major Pieces

App-Like Characteristics

Pruning the Critical Path

Tolerating Network Instability

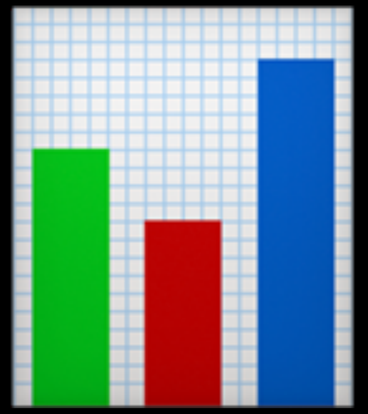
Pruning the Critical Path

Important Metrics

Time to...

First Paint

Interactive

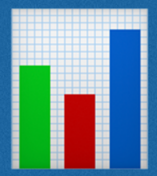


Data

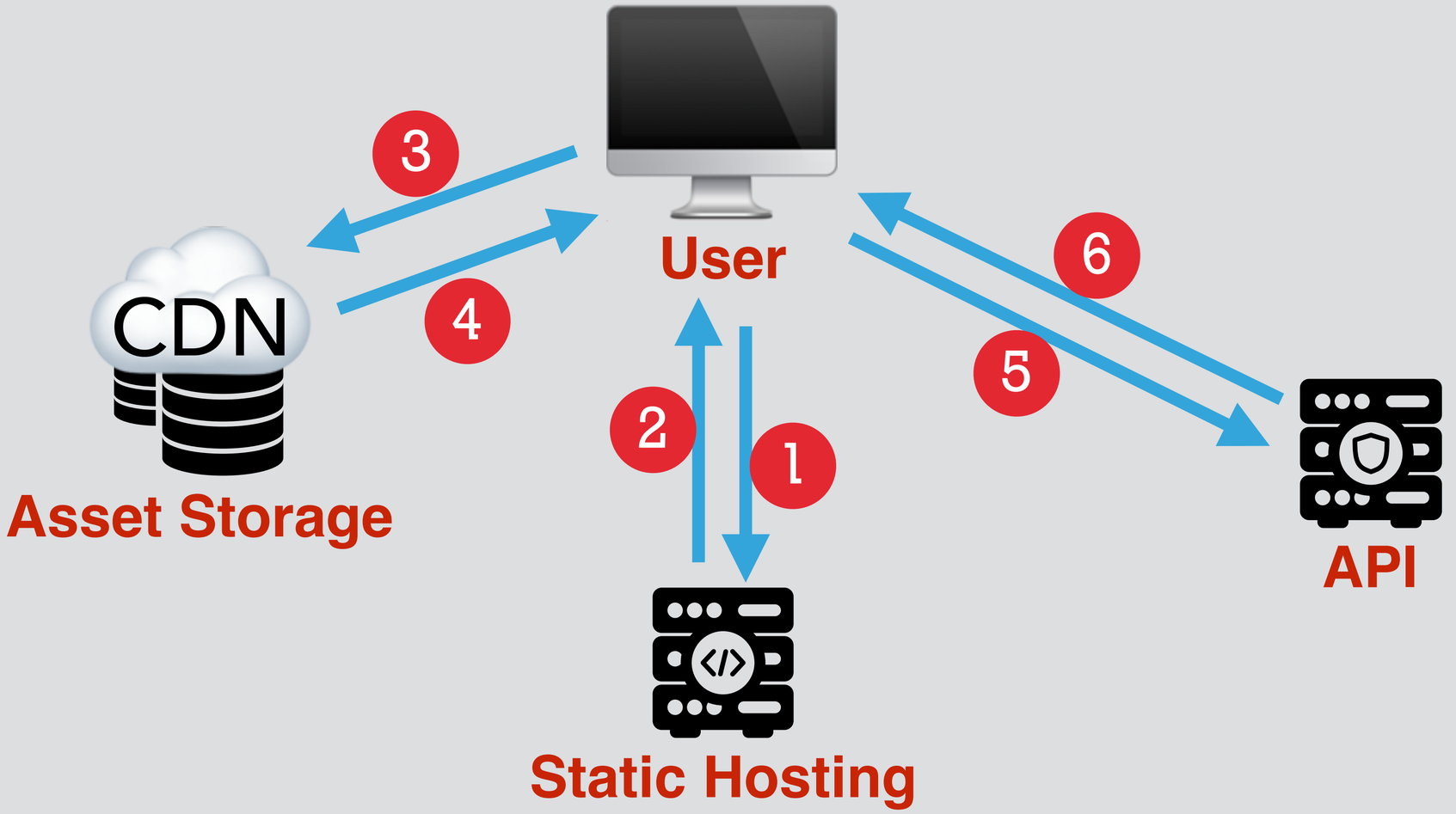
Client Rendered SPA

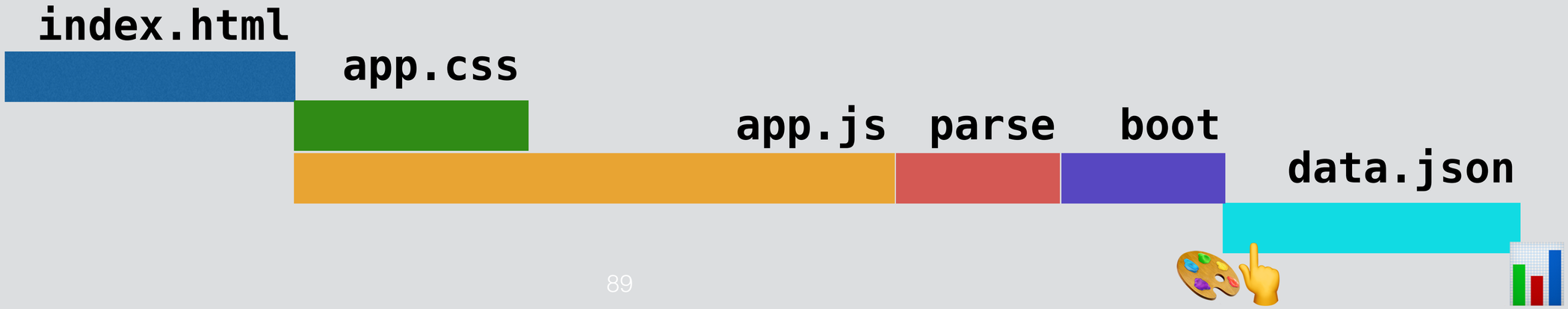
First Paint

Interactive



Data

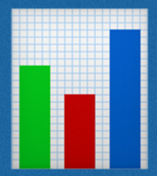




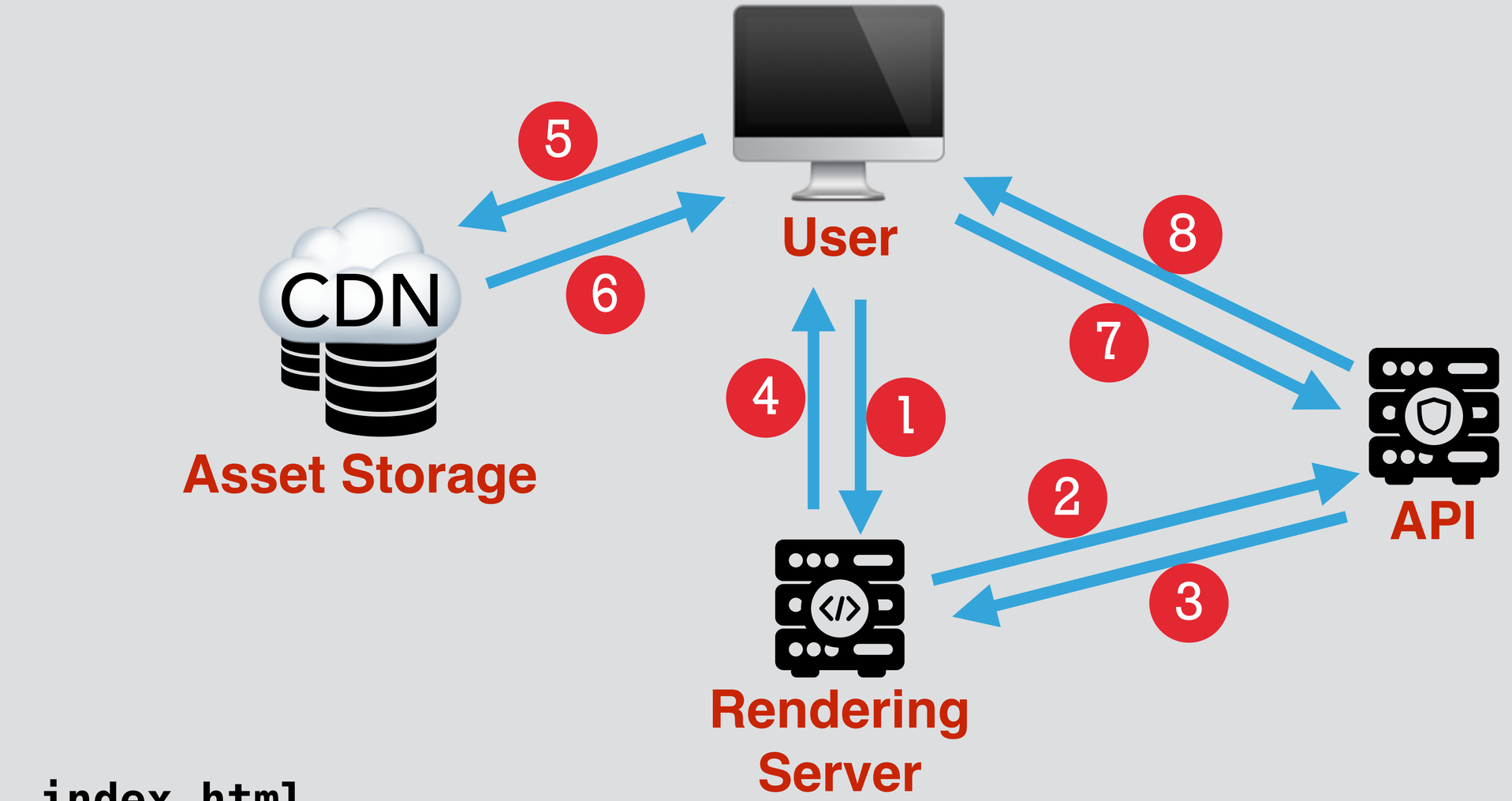
Server Rendered SPA

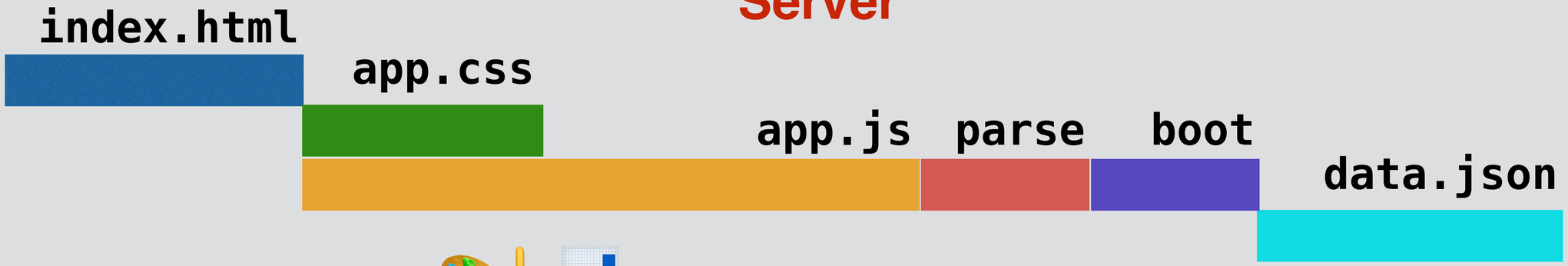
First Paint

Interactive



Data





Server-Side Rendering

- ⏱️ Don’t have to wait for client-side JS
- 🔗 Better for SEO & sharing links
- ⚠️ Node.js and the Browser are different
- 👷 Try not to do the same work twice
- ⚡ HTML is assembled in your data center

Setting up Fastboot

- Consumable as an ember addon

```
ember install ember-cli-fastboot
```

- Runnable on the cli

```
ember serve
```

- In production, using a node app

EMBER-FASTBOOT/FASTBOOT-APP-SERVER

Differences in Capabilities

- Simple-DOM vs browser-DOM
- FastBoot is the only global

Practical Consequences

- Libs that depend on full DOM implementation: 👎
- Common practice: replace a browser lib with a node one
- Use fetch instead of $.ajax

Guards

Running

```
if (typeof FastBoot === 'undefined') {
  👩‍💻 BROWSER CLIENT-SIDE STUFF
} else {
  🛠️ FASTBOOT CLIENT-SIDE STUFF
}
```

Building

```
if (!process.env.EMBER_CLI_FASTBOOT) {
  👩‍💻 BUILD-FOR-BROWSER STUFF
} else {
  🛠️ BUILD-FOR-FASTBOOT STUFF
}
```

Guards

Running

```
if (typeof FastBoot === 'undefined') {
  👩‍💻 BROWSER CLIENT-SIDE STUFF
} else {
  🛠 FASTBOOT CLIENT-SIDE STUFF
}
```

Building

```
treeForFastBoot: function(tree) {
  🛠 BUILD-FOR-FASTBOOT STUFF
}
```

Render with Fastboot

6

- Install ember-cli-fastboot
- Run fastboot locally, serving assets
- If you find anything that's unavoidably browser-specific, see if you can just skip it when rendering on the server side
- Push to heroku (bonus points for pushing to github, and letting travis deploy it for you)

Request Data in Fastboot

- Unlike in the static hosting setup, we have access to a request!

```
export default Ember.Route.extend({
  fastboot: Ember.inject.service(),

  model() {
    let headers = this.get('fastboot.request.headers');
    let xRequestHeader = headers.get('X-Request');
    // ...
  }
});
```

Handing off Data

- You may notice double requests to your API
- Rehydration - not 100% automated yet
- We have a useful tool: shoebox

Shoebox: Concept

- Get data from API using fetch
- Store it in the shoebox (node: memory)
- HTML is rendered, shoebox —> meta tag
- Browser app starts up, and can get shoebox data

```
<script type="fastboot/shoebox" id="shoebox-main-store">
{"data":[
  {"attributes":{"name":"AEC Professionals"},"id":106,"type":"audience"},
  {"attributes":{"name":"Components"},"id":111,"type":"audience"},
]}
</script>
```

```
// Get the shoebox, and the shoebox store we want
let shoebox = this.get('fastboot.shoebox');
let shoeboxStore = shoebox.retrieve('my-store');

if (this.get('fastboot.isFastBoot')) {

}
```

```
// Get the shoebox, and the shoebox store we want
let shoebox = this.get('fastboot.shoebox');
let shoeboxStore = shoebox.retrieve('my-store');

if (this.get('fastboot.isFastBoot')) {
  // If rendering on fastboot server, make the request
  return this.store.findRecord('post', params.post_id)
    .then(post => {



  });
}
```

```
// Get the shoebox, and the shoebox store we want
let shoebox = this.get('fastboot.shoebox');
let shoeboxStore = shoebox.retrieve('my-store');

if (this.get('fastboot.isFastBoot')) {
  // If rendering on fastboot server, make the request
  return this.store.findRecord('post', params.post_id)
    .then(post => {
      if (!shoeboxStore) { // Lazily create the store
        shoeboxStore = {};
        shoebox.put('my-store', shoeboxStore);
      }
      // Put the data in the store
      shoeboxStore[post.id] = post.toJSON();
  });
}
```

Server Data, on Client

7

- Grab the user agent of the incoming request to index.html, and make it available in the container under the container key

`"data:request"`

- Prove that your solution works, by render this data in a template somewhere

```
$E.__container__.lookup('data:request')
▶ Object {user-agent: "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_2) AppleWebKit/537.36 (KHTML"}
```

Shoebox —> Store

- Server data retrieval is cheap
- Shoebox is nice, but very explicit
- ember-data-fastboot: automated serialization & population of the ember-data store

```
ember install ember-data-fastboot
```

Retrieving Data from the store

```
this.store.peekRecord(Post, 7);  // Only cache
this.store.findRecord(Post, 7);  // Cache first, then refresh
```

Retrieving Data from the store

- Usually, findRecord is what you want

```
this.store.peekRecord(Post, 7);  // Only cache
this.store.findRecord(Post, 7);  // Cache first, then refresh

this.store.findRecord(Post, 7, {
  backgroundReload: false        // Equivalent to "peek"
});

this.store.findRecord(Post, 7, {
  reload: true                  // Equivalent to "fetch"
});
```

Ember-Simple-Auth

- **Authenticator:** handles particulars of getting authenticated
- **Authorizer:** uses privileges to do things
- **Session-Store:** serializes/deserializes client-side session
- **Session-Service:** auth state + means of doing stuff w/ it

Shoeboxed Current User

8

/api/users/current

- Immediately after login, and as long as the session appears to be valid, we want a currentUser to be available for use
- We'll need this data in routes, components and elsewhere
- Show user email in the navbar about the current user
- Current authentication scheme: OAuth2 Password Grant
- The application adapter needs some attention to pass token to API

Major Pieces

App-Like Characteristics

Pruning the Critical Path

Tolerating Network Instability

Tolerating Network Instability

Resource Caching Strategies

Immutable Content

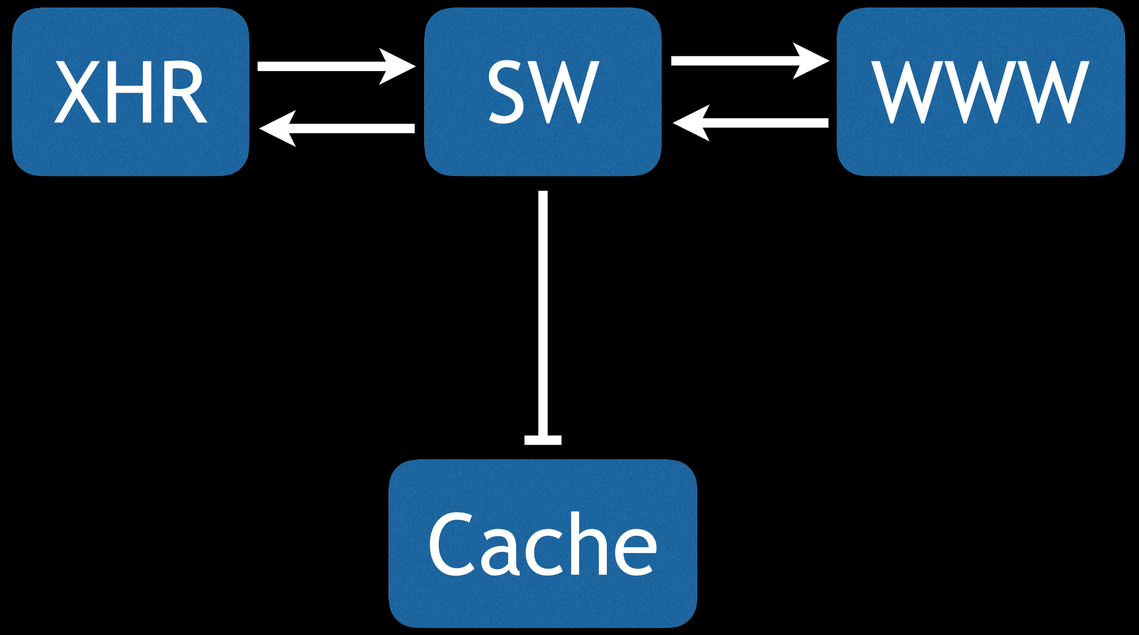
- Content at a URL is never changed
- max-age=year
- index.html is the key
- Offline-friendly

“Check w/ server”

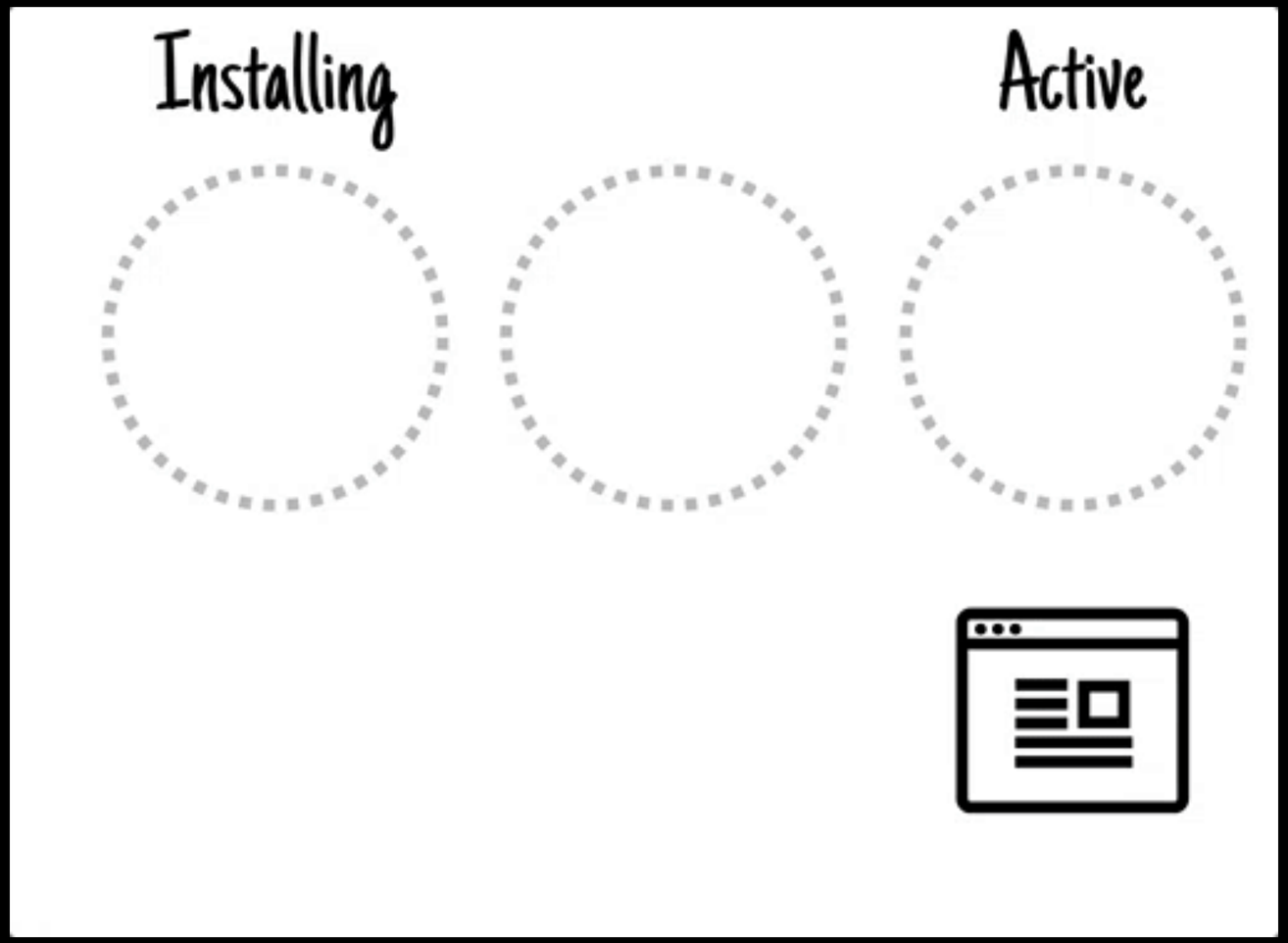
- Content at a URL can change
- Cache-Control: no-cache
- Use Last-Modified or ETag
- Involves extra network requests

Service Workers

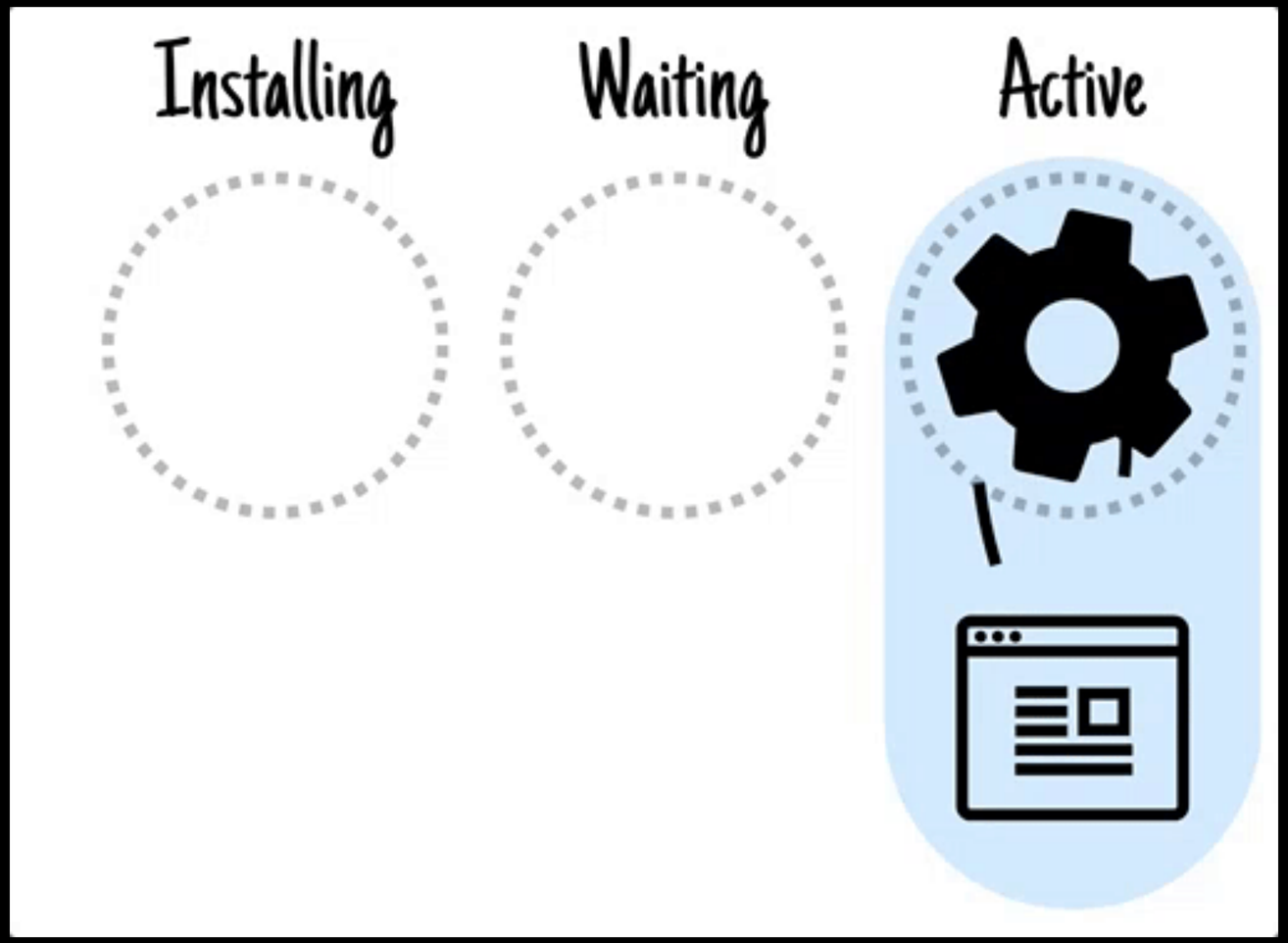
- A programmable network proxy
- A JavaScript worker
- Only works over HTTPS
- Has a predictable lifecycle



Service Workers



Service Workers



Service Workers

Precache

I have a list of important resources. I'll fetch them all and cache them.

Cache

App

I need item.png

Of course! Here it is.

Service Workers

Network or Cache

App

I need item.png

Ok, I'll get it

Cache

Just got it for you!

Service Workers

Network or Cache

App

I need item.png

Ok, I'll get it

Cache

Took too long. Here's a cached version

Service Workers

Network or Cache

App

I need item.png

Ok, I'll get it

Cache

Something went wrong.
Here's a cached version

Service Workers

Cache and Update

App

I need item.png

Here’s a cached version

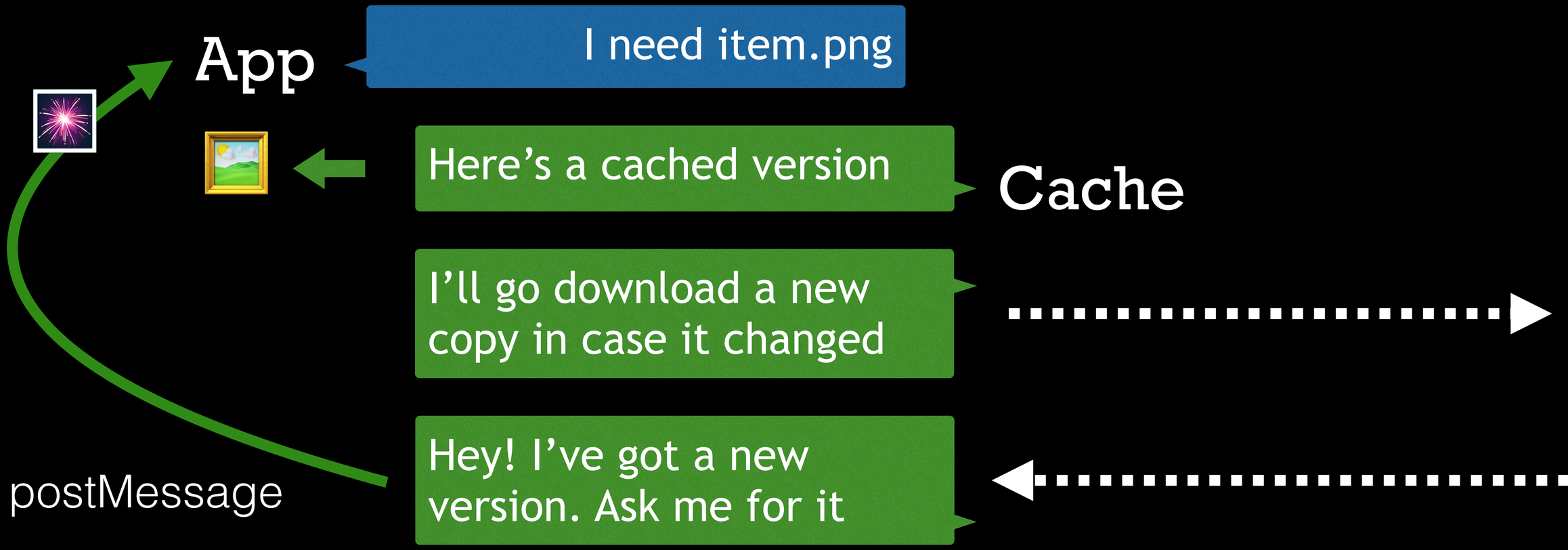
Cache

I’ll go download a new copy in case it changed

Next time you ask, you’ll get this new version

Service Workers

Cache, Update and Refresh



Service Workers

ServiceWorker enthusiasm

The first thing any implementation needs.

Chrome: Shipped.

Firefox: Shipped.

Samsung Internet: Shipped. Based on Chromium 44.2403 with some additions and changes. (See "Service Worker" section.)

Safari: Under consideration, Brief positive signals in five year plan.

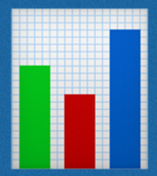
Edge: In development.

Support does not include iOS versions of third-party browsers on that platform (see Safari support).

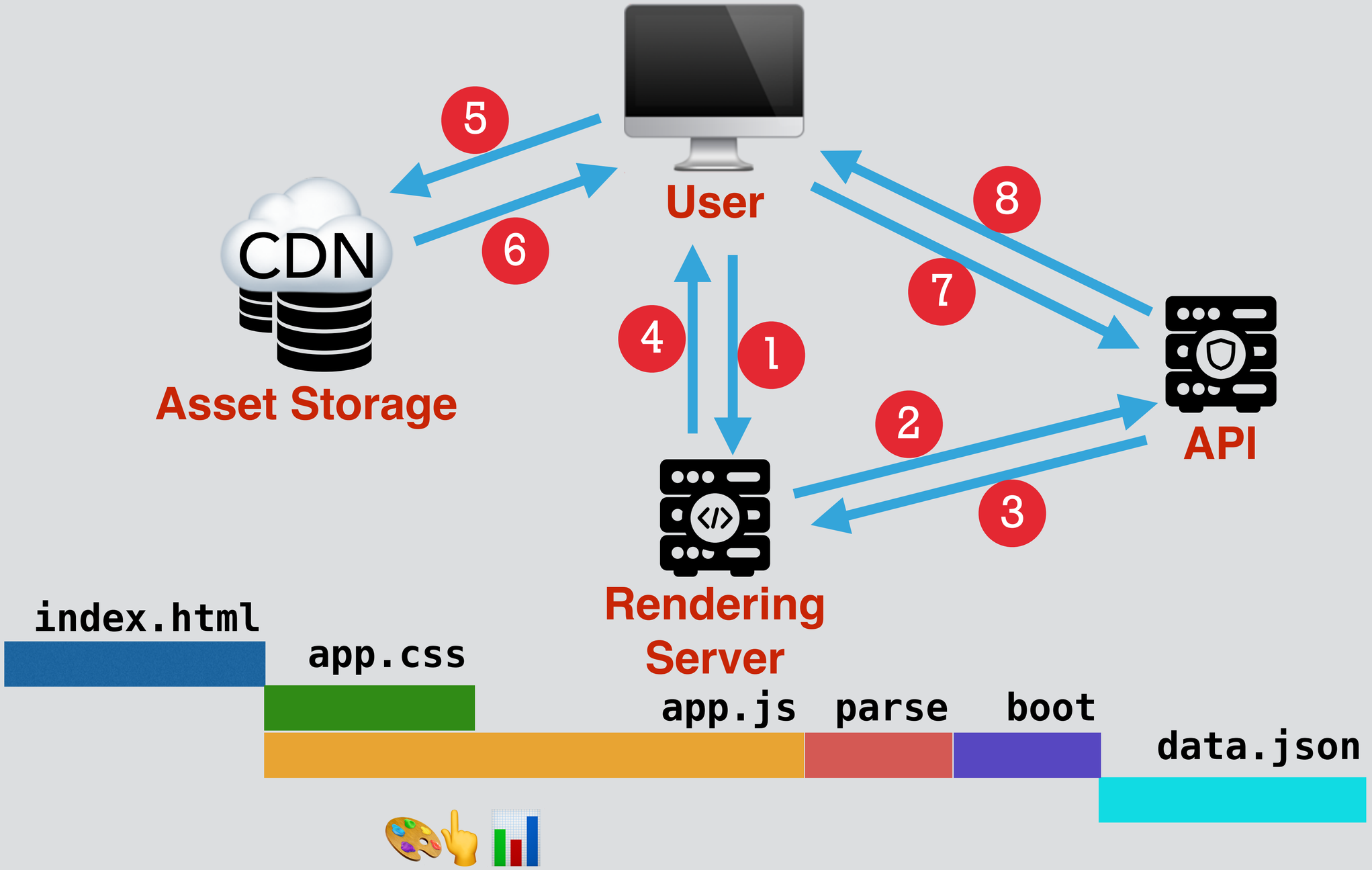
Server Rendered SPA

First Paint

Interactive



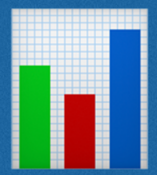
Data



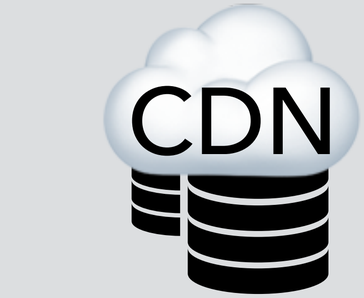
Server Rendered PWA v1

First Paint

Interactive

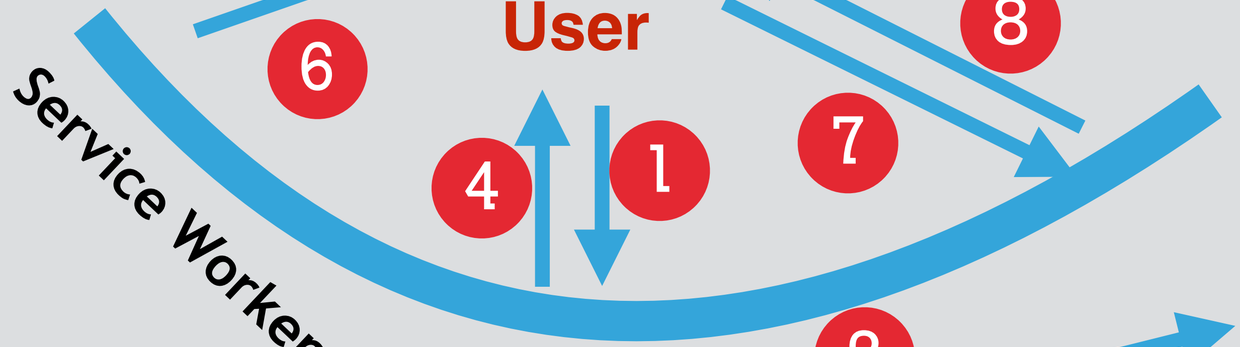


Data



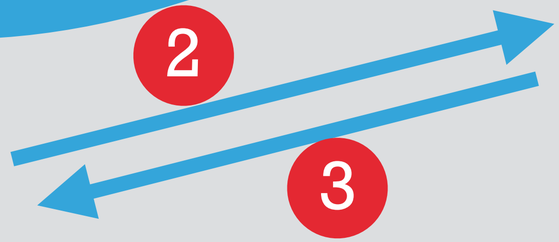
Asset Storage

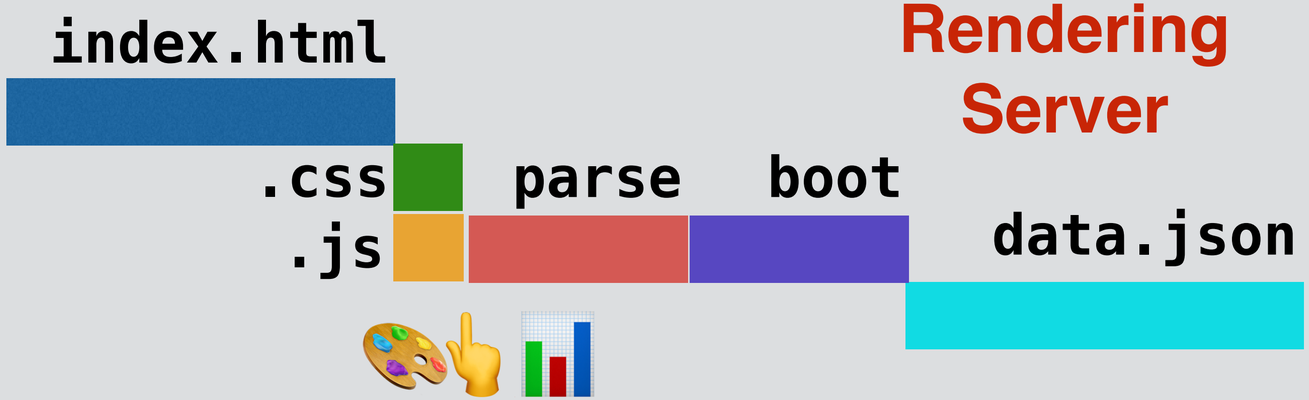
User



API

Rendering Server





Service Workers in Ember Apps

ember-service-worker

- Creates appropriate output files
- A pluggable, event-based approach
- Doesn’t do anything on its own

package.json

```
"keywords": [
  "ember-addon",
  "ember-service-worker-plugin"
],
```

Service Workers in Ember Apps

ember-service-worker-asset-cache

- Creates a list of your static assets at build time
- Instructs the service worker to download these on install
- Once SW activates, they’re ready for instant response
- Versioned — new workers purge old stuff on activate

Service Workers in Ember Apps

ember-service-worker-index

- Caches HTML for initial load
- Currently appears to interfere with fastboot
- Supports app-shell architecture (we'll talk about this later)

Service Workers in Ember Apps

ember-service-worker-cache-first

- Specify regular expressions or patterns for files
- When your app asks for a resource, provide the cached version
- … and then go and refresh the cache with new data

Service Workers in Ember Apps

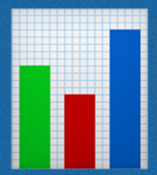
ember-service-worker-cache-fallback

- **When you're online**: Passthrough (cache as requests are made)
- **When you're offline**: Serve responses from the cache

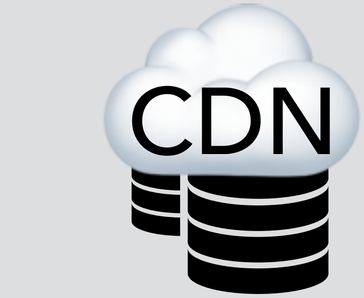
Server Rendered PWA v1

First Paint

Interactive

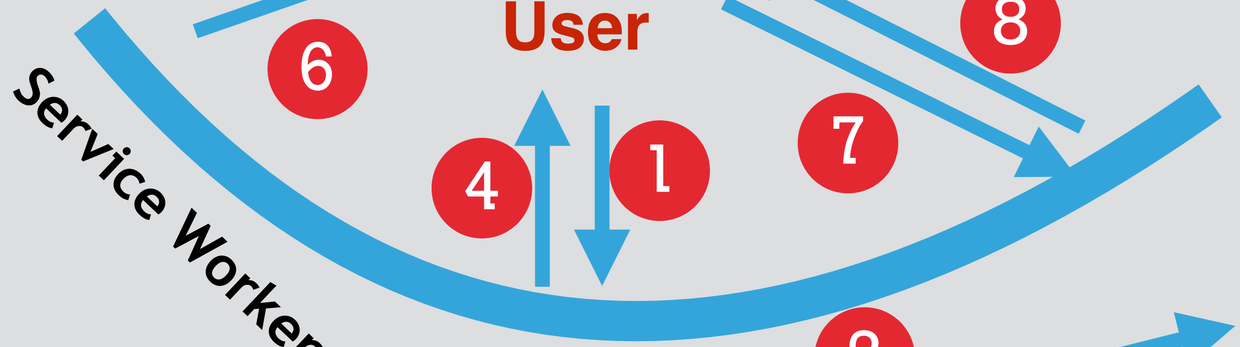


Data



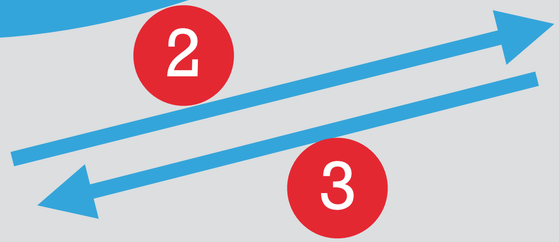
Asset Storage

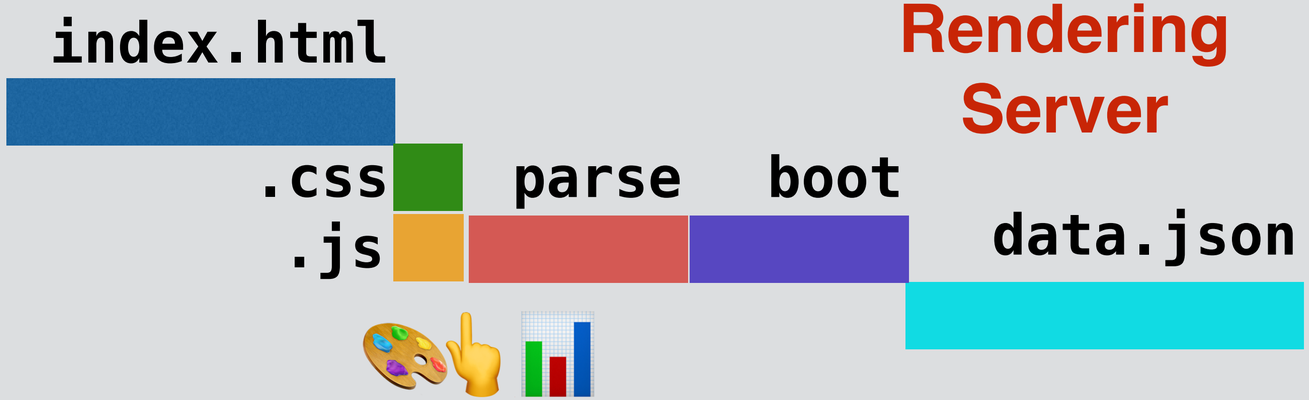
User



API

Rendering Server





https://github.com/mike-north/ember-pro

./EmberPro.pdf

ember PRO

Sept 5, 2017

Mike North

MIKE WORKS

Works Offline

9

```
ember install ember-service-worker
ember install ember-service-worker-asset-cache
ember install ember-service-worker-cache-fallback
ember install ember-service-worker-index
```

- Install these addons, and configure such that
  - JS, CSS, images in the public folder are handled by asset-cache
  - Speaker images and JSON are handled by the cache-fallback
  - HTML is handled by ember-service-worker-index
- Your app should now work even when you're offline

Storing Dynamic Content

- Cookies - Not worker-friendly, just a string
- LocalStorage - See 👆
- What's left???

Storing Dynamic Content

- Cookies - Not worker-friendly, just a string
- LocalStorage - See 👆

IndexedDB

IndexedDB

- 🔢 Versioned
- 🗃️ Indexable
- ⚙️ Worker-Friendly
- 📅 Durable
- 🔣 Supports values of many types
- ⚠️ Not a SQL, or a relational DB

IndexedDB

```
// Open (or create) the database
let open =
  indexedDB.open('MyDatabase', 1);

// Create and/or Migrate the Schema
open.onupgradeneeded = (evt) ⇒ { ... };


// Transactions
open.onsuccess = () ⇒ { ... };
```

IndexedDB

```
// Create and/or Migrate the Schema
open.onupgradeneeded = (evt) ⇒ {

    let db = open.result;
    let store = null;
    switch (evt.oldVersion) {
      case 0: // Upgrade from 0
        //  ...
      case 1: // Upgrade from 1
        //  ...
    }

};
```

IndexedDB

```
// Open (or create) the database
let open =
  indexedDB.open('MyDatabase', 1);


// Create and/or Migrate the Schema
open.onupgradeneeded = (evt) ⇒ { ... };


// Transactions
open.onsuccess = () ⇒ { ... };
```

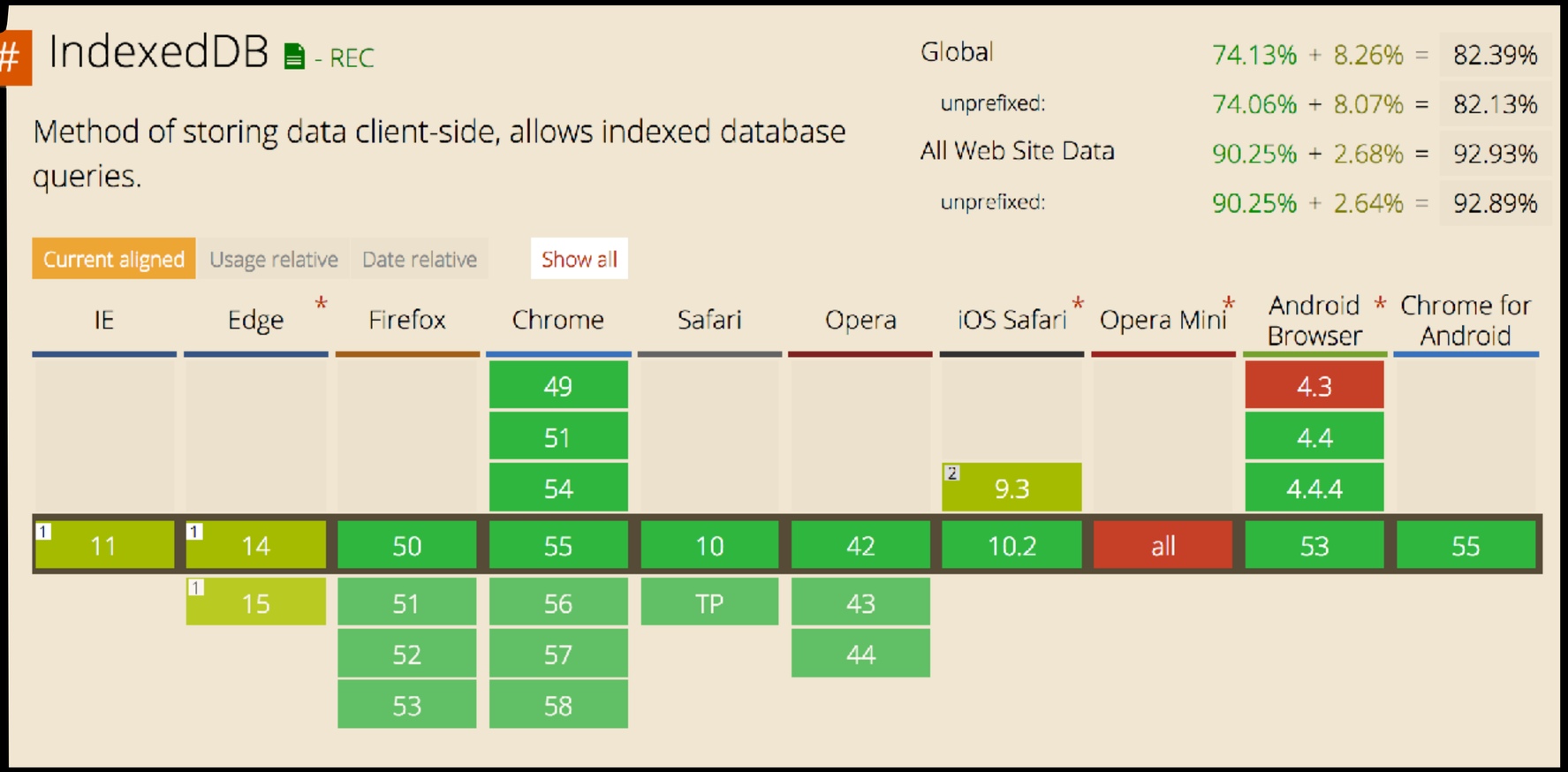
IndexedDB

```
// Transactions
open.onsuccess = () ⇒ {

  // Start a new transaction
   let db = open.result;
   let tx = db.transaction('MyObjectStore', 'readwrite');
   let store = tx.objectStore('MyObjectStore');

   // ⟵ Transaction Particulars ⟶ //

   // Close the db when the transaction is done
   tx.oncomplete = () ⇒ db.close();

};
```

IndexedDB

```
let index = store.index('NameIndex');

 // Add some data
 store.put({
   id: 12345,
   name: { first: 'Mike', last: 'North' },
   age: 33
 });

 // Query the data
 let getJohn = store.get(12345);
 let getBob = index.get(['North', 'Mike']);
```

IndexedDB

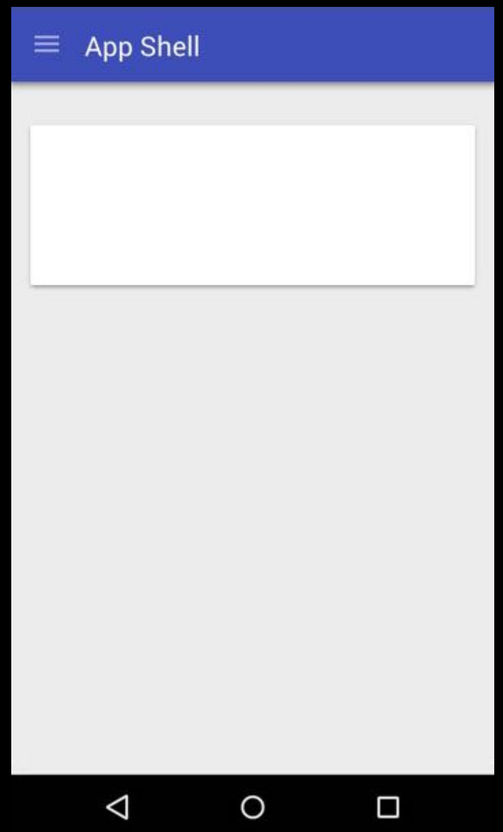

IndexedDB - REC
Method of storing data client-side, allows indexed database queries.
Global 74.13% + 8.26% = 82.39%
unprefixed: 74.06% + 8.07% = 82.13%
All Web Site Data 90.25% + 2.68% = 92.93%
unprefixed: 90.25% + 2.64% = 92.89%
Current aligned
Usage relative
Date relative
Show all
IE
Edge *
Firefox
Chrome
Safari
Opera
iOS Safari *
Opera Mini *
Android Browser *
Chrome for Android
11
14
15
50
51
52
53
49
51
54
55
56
57
58
10
TP
42
43
44
9.3
10.2
all
4.3
4.4
4.4.4
53
55

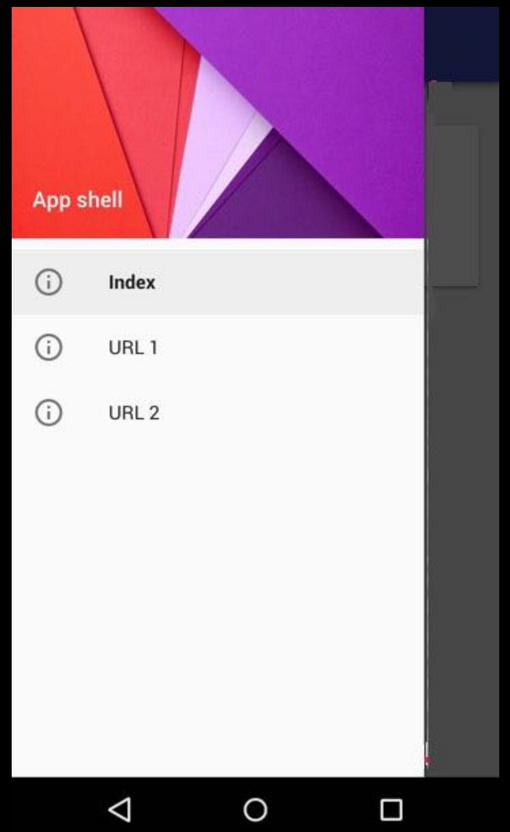
IndexedDB Storage

- Use either localStorage or IndexDB to allow “bingo pieces” to persist across reloads.
- If you use IndexDB, make sure to
  - deferReadiness/advanceReadiness
  - use a promise-aware hook in a route somewhere to ensure “moves” are restored before rendering happens

App Shell Architecture

Shell loads instantly



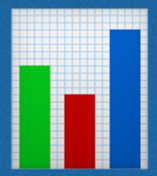


Dynamic content in the shell

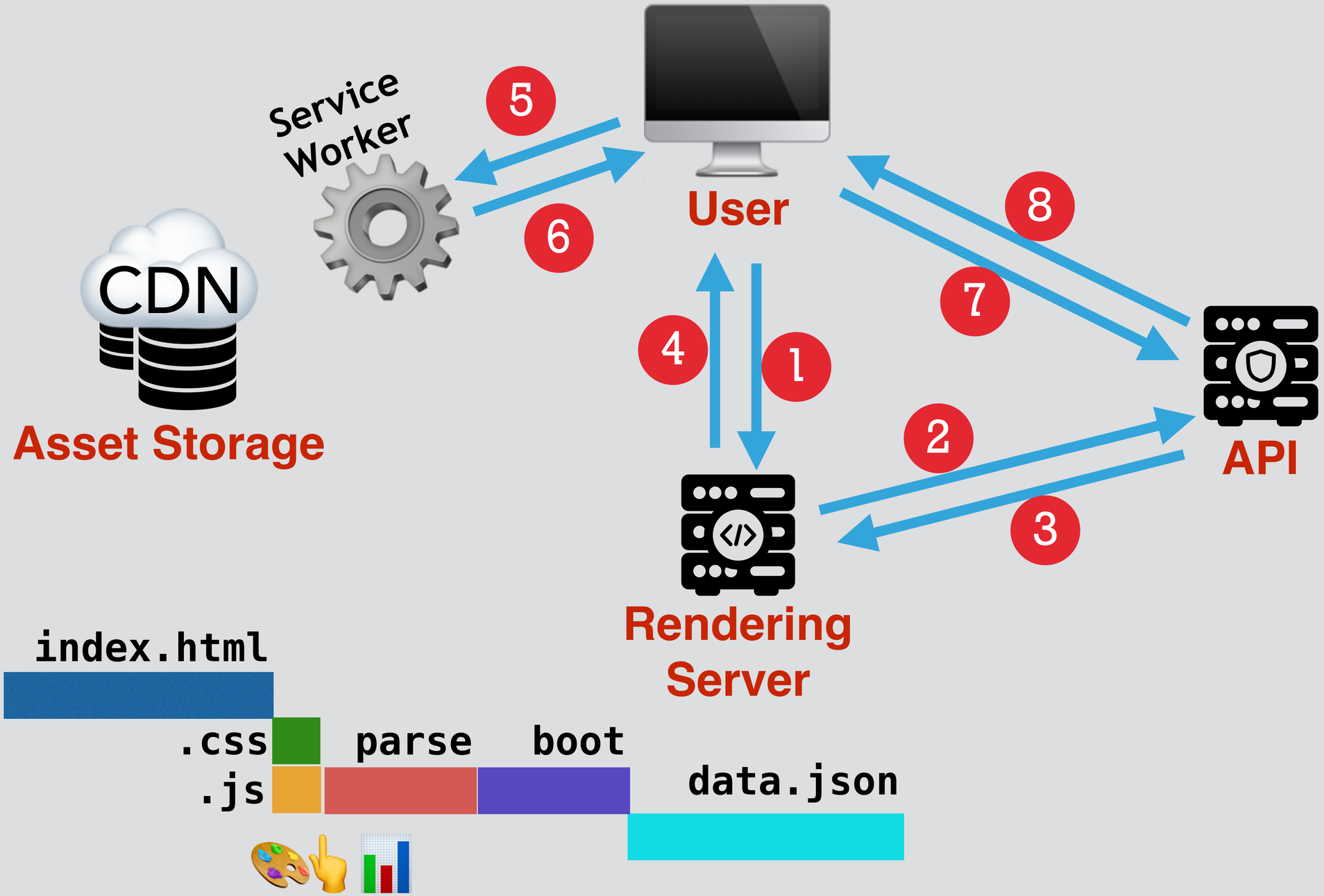
Server Rendered PWA v1

First Paint

Interactive



Data

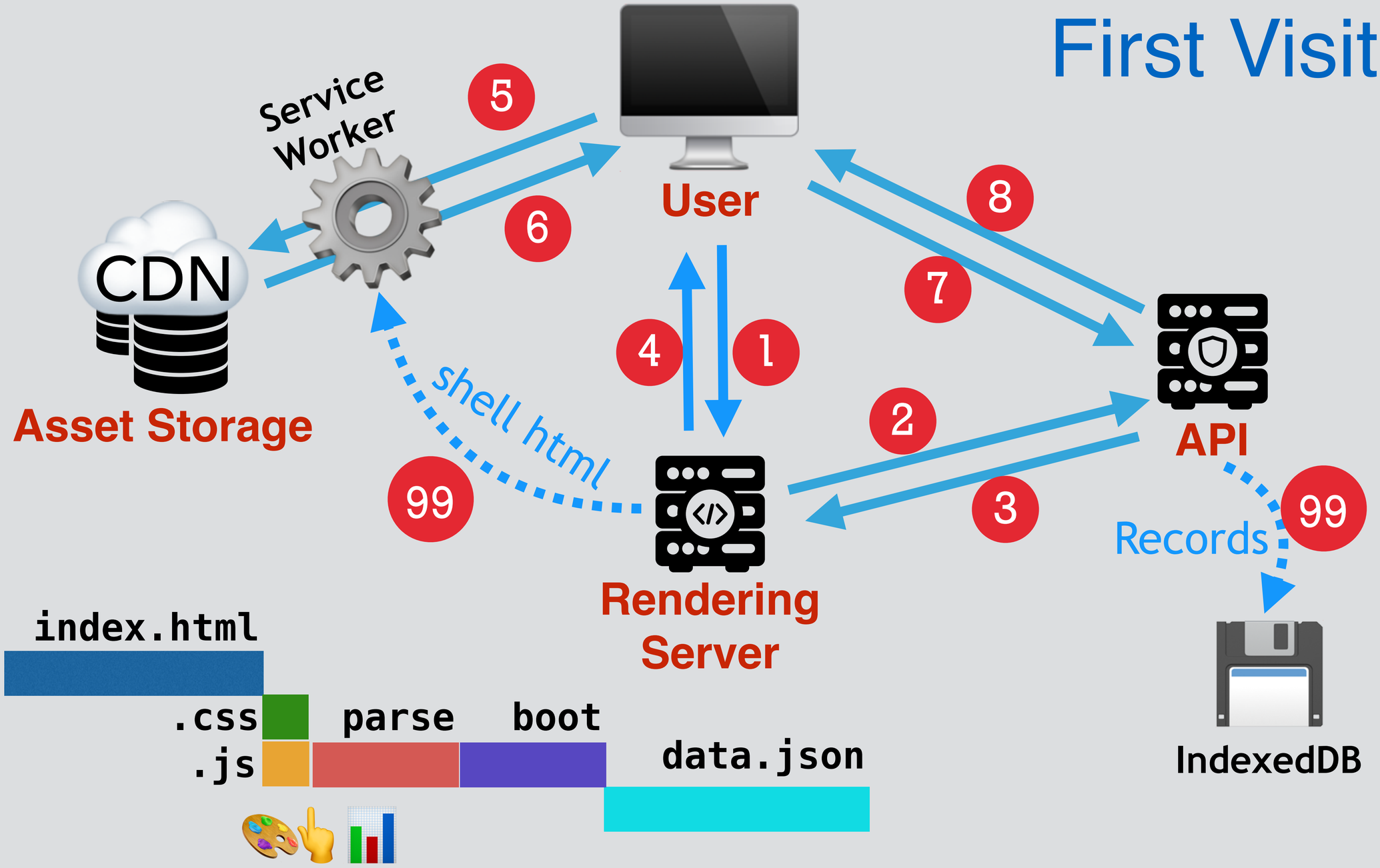


Server Rendered PWA v2

First Paint

Interactive

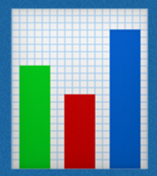
Data



Server Rendered PWA v2

First Paint

Interactive



Data

Local
Remote

Service Worker

CDN

Asset Storage

User

Return Visit

1 2 3 4 5 6 7 8

Rendering Server

API

IndexedDB

shell.html
.css
parse
boot
.js
data.json
IndexedDB

State Management

4 Flavors of state

UI STATE

PERSISTED STATE

ADDRESSABLE STATE

DRAFT STATE

localhost:4200/posts/A96usFSlY4G0W4kwAqsw

Search

Seven Tips From Ernest He… 0

Mike Springer October 2, 2016

To get started, write one true sentence
Sometimes when I was starting a new st...

Where does your state belong 1

Mike North October 10, 2016

For over a decade, most of us built web UIs operating under the assumption that ...

Down the Rabbit Hole 0

Lewis Carroll October 2, 2016

Alice was beginning to get very tired of sitting by her sister on the bank, and ...

Seven Tips From Ernest Hemingway on How to Write Fiction

details...

To get started, write one true sentence

Sometimes when I was starting a new story and I could not get it going, I would sit in front of the fire and squeeze the peel of the little oranges into the edge of the flame and watch the sputter of blue that they made. I would stand and look out over the roofs of Paris and think, “Do not worry. You have always written before and you will write now. All you have to do is write one true sentence. Write the truest sentence that you know.”

So finally I would write one true sentence, and then go on from there. It was easy then because there was always one true sentence that I knew or had seen or had heard someone say. If I started to write elaborately, or like someone introducing or presenting something, I found that I could cut that scrollwork or ornament out and throw it away and start with the first true simple declarative sentence I had written.

Always stop for the day while you still know what will happen next

The best way is always to stop when you are going good and

localhost:4200/posts/1fm1gRRxhS0GAkC2k4G8UG?q=Wher

Where

Where does your state belong 1

Mike North October 10, 2016

For over a decade, most of us built web UIs operating under the assumption that ...

our users make about how apps should work, and we must work a bit harder in order to keep them intact. Take the 'Back' and 'Refresh' buttons, for example: in order for this to work as our users expect, we must keep certain elements of state serialized in the URL in order to avoid breaking this as we simulate a multi-page experience in a SPA.

Add in the concept of server-side rendering, where our asset serving layer sometimes needs browser details (i.e. viewport dimensions) in order to render the correct content, and state decisions become even more consequential and complex.

In this talk, I'll outline four types of state

- Navigation state
- Persisted state
- UI state
- "Will be persisted" state

and provide examples for each. Along the way, we'll start to assemble a framework of questions that you can ask yourself when encountering new pieces of state, to lead you down the right path(s).

This is another comment

21 minutes ago

localhost:4200/posts/1fm1gRRxhS0GAkC2k4G8

Seven Tips From Ernest He… 0

Mike Springer October 2, 2016

To get started, write one true sentence
Sometimes when I was starting a new st...

Where does your state belong 1

Mike North x DRAFT October 10, 2016

For over a decade, most of us built web UIs operating under the assumption that ...

Down the Rabbit Hole 0

Lewis Carroll October 2, 2016

Alice was beginning to get very tired of sitting by her sister on the bank, and ...

'Refresh' buttons, for example: in order for this to work as our users expect, we must keep certain elements of state serialized in the URL in order to avoid breaking this as we simulate a multi-page experience in a SPA.

Add in the concept of server-side rendering, where our asset serving layer sometimes needs browser details (i.e. viewport dimensions) in order to render the correct content, and state decisions become even more consequential and complex.

In this talk, I'll outline four types of state

- Navigation state
- Persisted state
- UI state
- "Will be persisted" state

and provide examples for each. Along the way, we'll start to assemble a framework of questions that you can ask yourself when encountering new pieces of state, to lead you down the right path(s).

This is another comment

21 minutes ago

This talk is going pretty well!

Reply Cancel

localhost:4200/posts/1fm1gRRxhS0GAkC2k4G8

Seven Tips From Ernest He… 0

Mike Springer October 2, 2016

To get started, write one true sentence
Sometimes when I was starting a new st...

Where does your state belong 2

Mike North October 10, 2016

For over a decade, most of us built web UIs operating under the assumption that ...

Down the Rabbit Hole 0

Lewis Carroll October 2, 2016

Alice was beginning to get very tired of sitting by her sister on the bank, and ...

'Refresh' buttons, for example: in order for this to work as our users expect, we must keep certain elements of state serialized in the URL in order to avoid breaking this as we simulate a multi-page experience in a SPA.

Add in the concept of server-side rendering, where our asset serving layer sometimes needs browser details (i.e. viewport dimensions) in order to render the correct content, and state decisions become even more consequential and complex.

In this talk, I'll outline four types of state

- Navigation state
- Persisted state
- UI state
- "Will be persisted" state

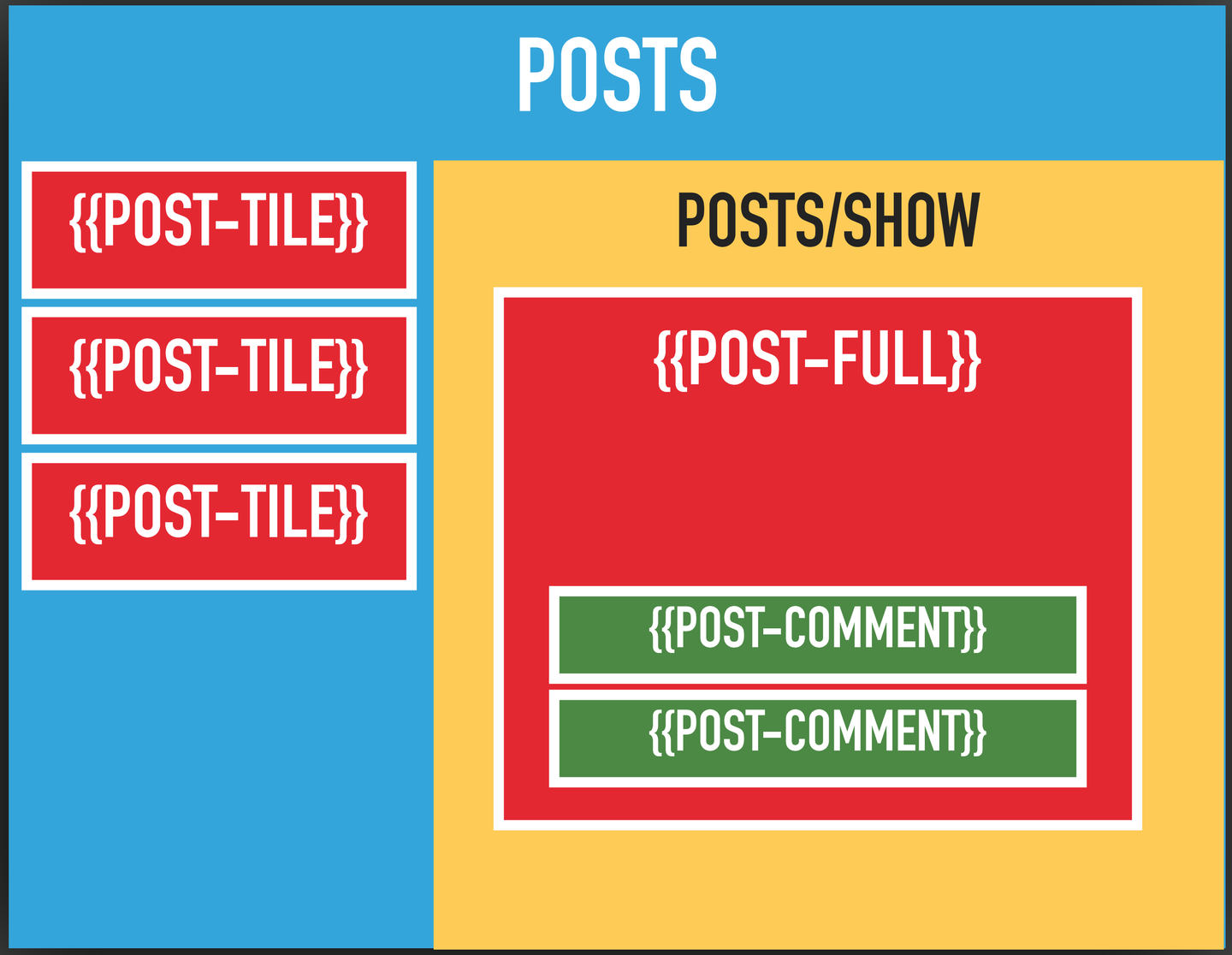
and provide examples for each. Along the way, we'll start to assemble a framework of questions that you can ask yourself when encountering new pieces of state, to lead you down the right path(s).

This is another comment ×

21 minutes ago

This talk is going pretty well! ×

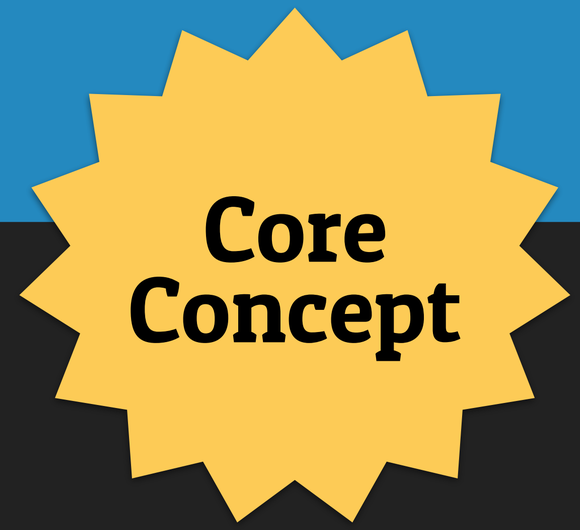
a few seconds ago



Addressable State

Addressable State

Sharable, Historically archived and URL-Driven



Examples

- Sorted/filtered search results
- Tabs/Navigation
- A section of a Github Readme

How to identify Addressable state

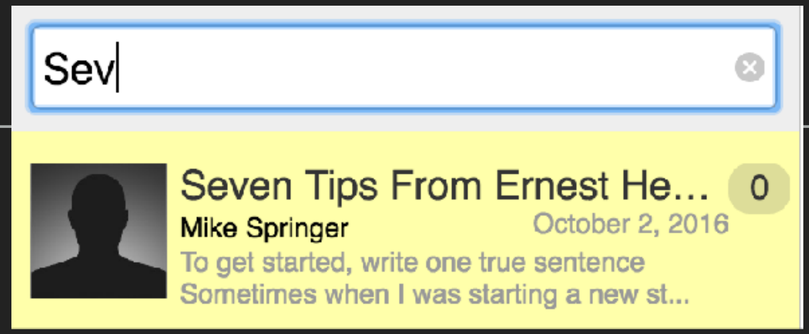
- Affects GET API calls
  - A perspective on a record or collection of records
- Useful when shared
- Mismanaged when - **back** button breaks

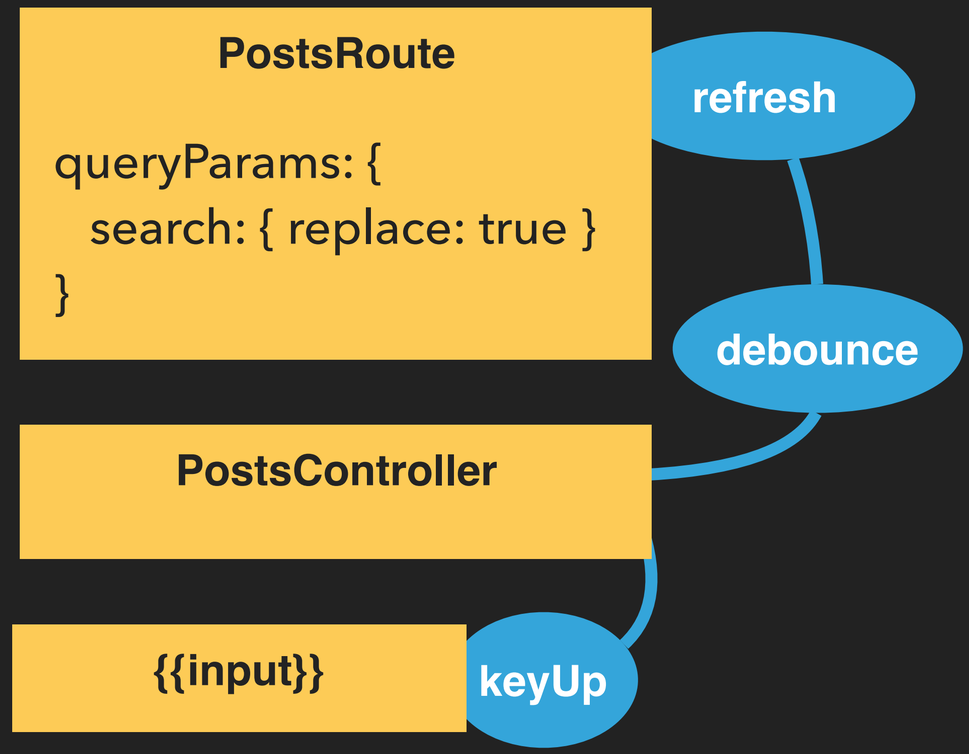
contract: users and the web

- **Refresh**: never destructive by surprise
- **Bookmarks**: get back to where I was
- **Back**: previous major thing
- **Back a lot**: leave the app

Enhancing Our Basic App

- Filter by a name fragment
- Bookmark & Share
- Don’t abuse our API
- **D**ata **D**own, **A**ctions **U**p





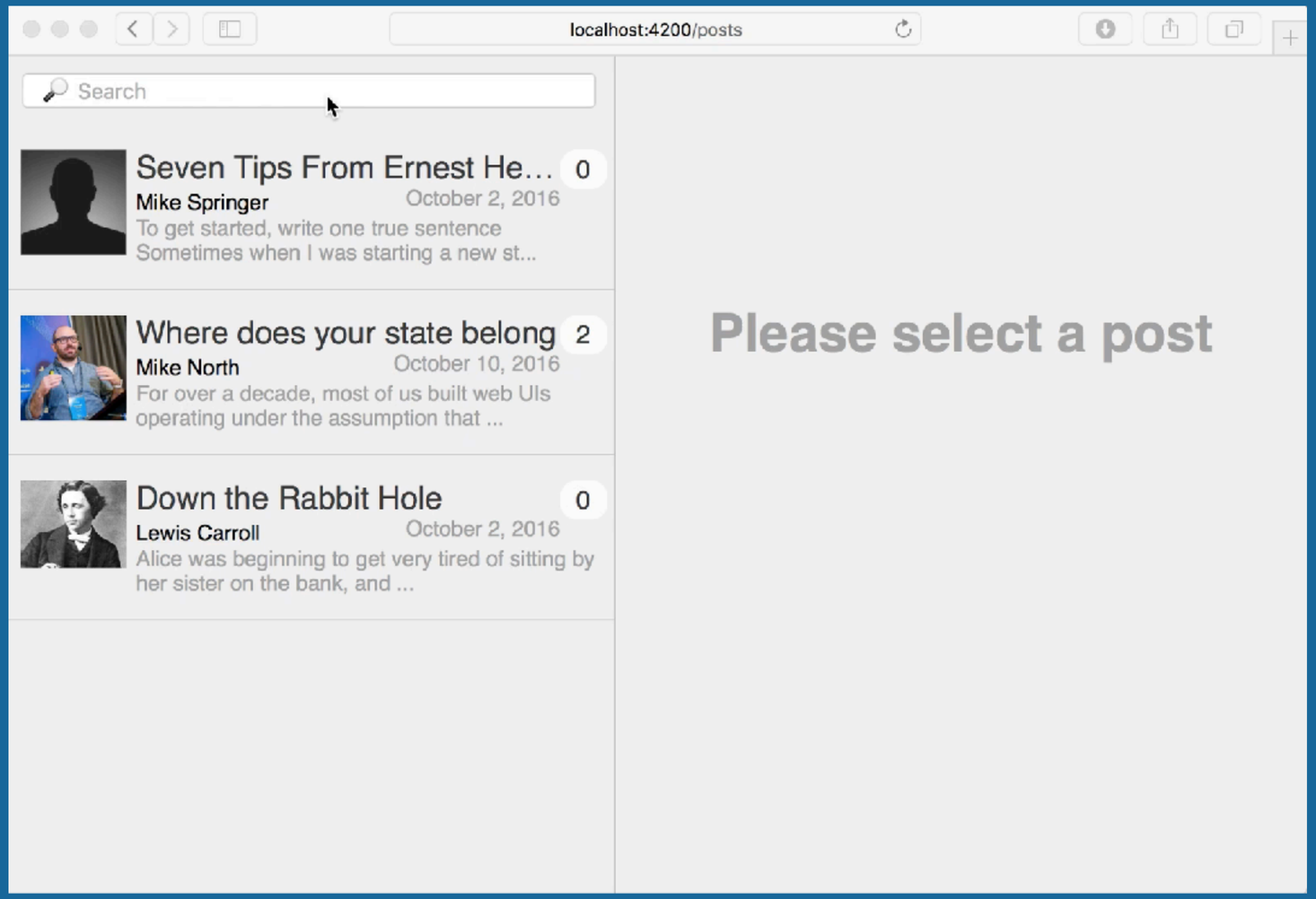
Gotchas and Common Issues

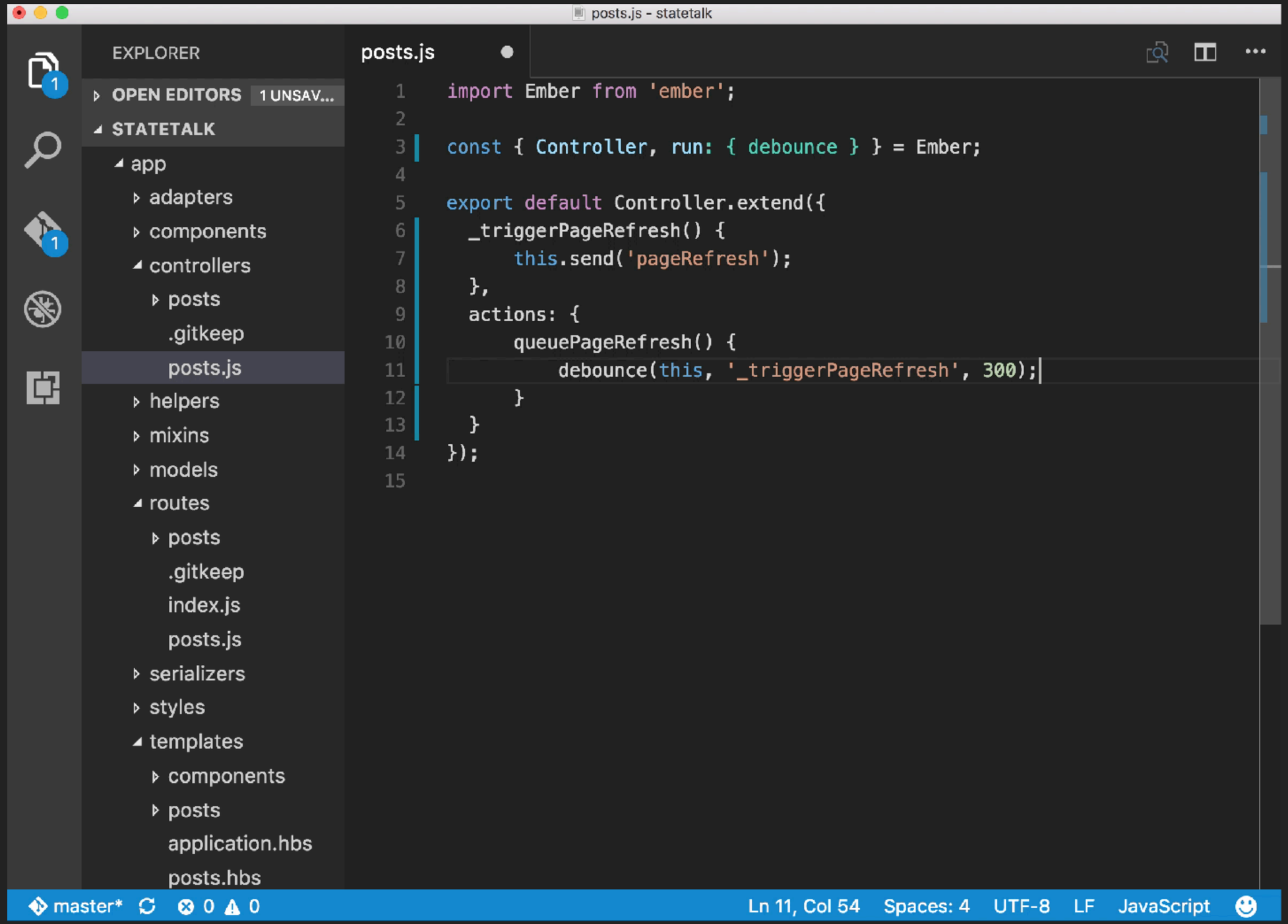
- Trapping users
- Deleting useful history
- Unimportant history states
- Some Route queryParam options are mutually exclusive (right now) (replace vs refreshModel)
- Tricky: Add to Home Screen

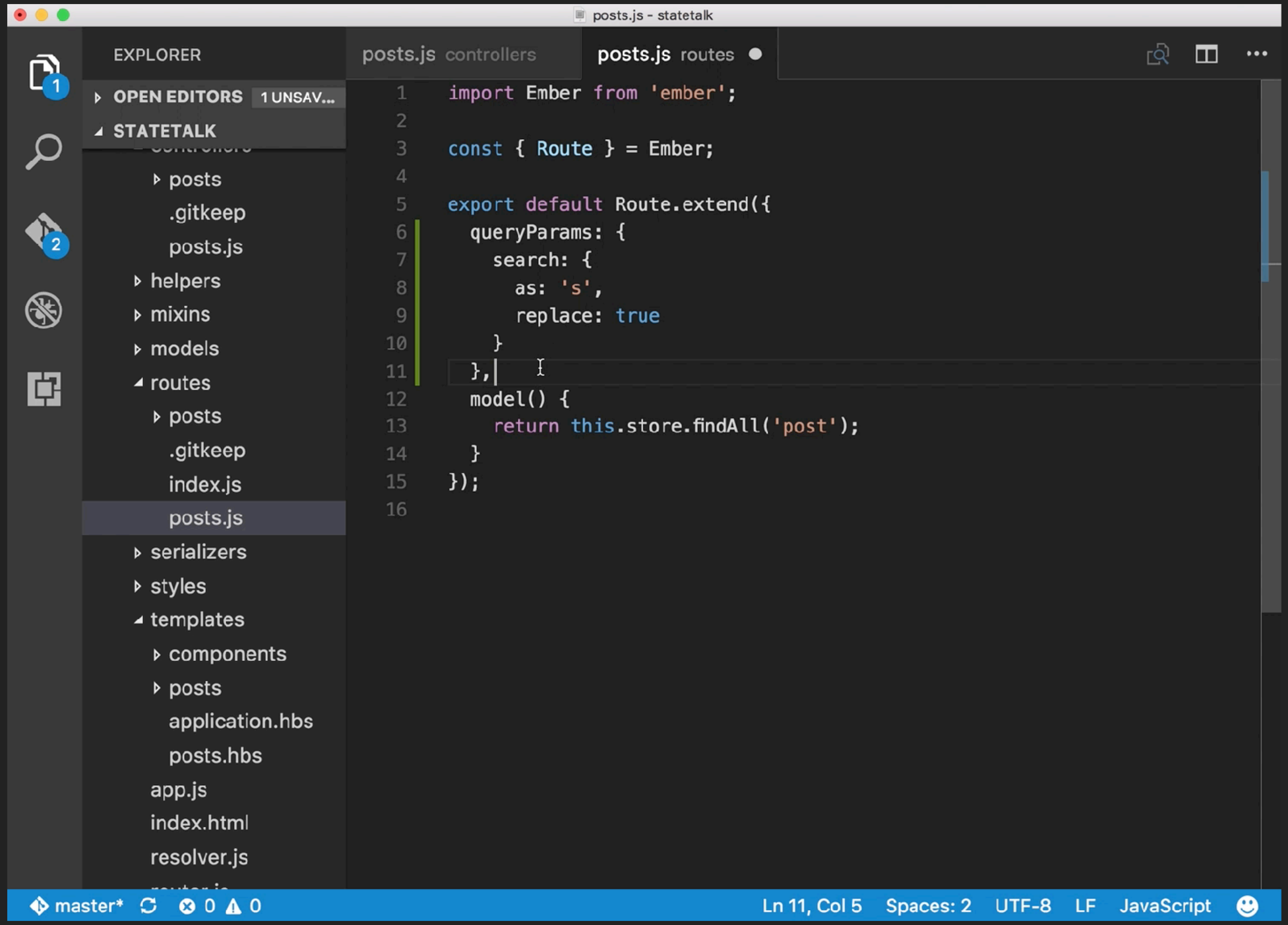
Filter by QueryParam

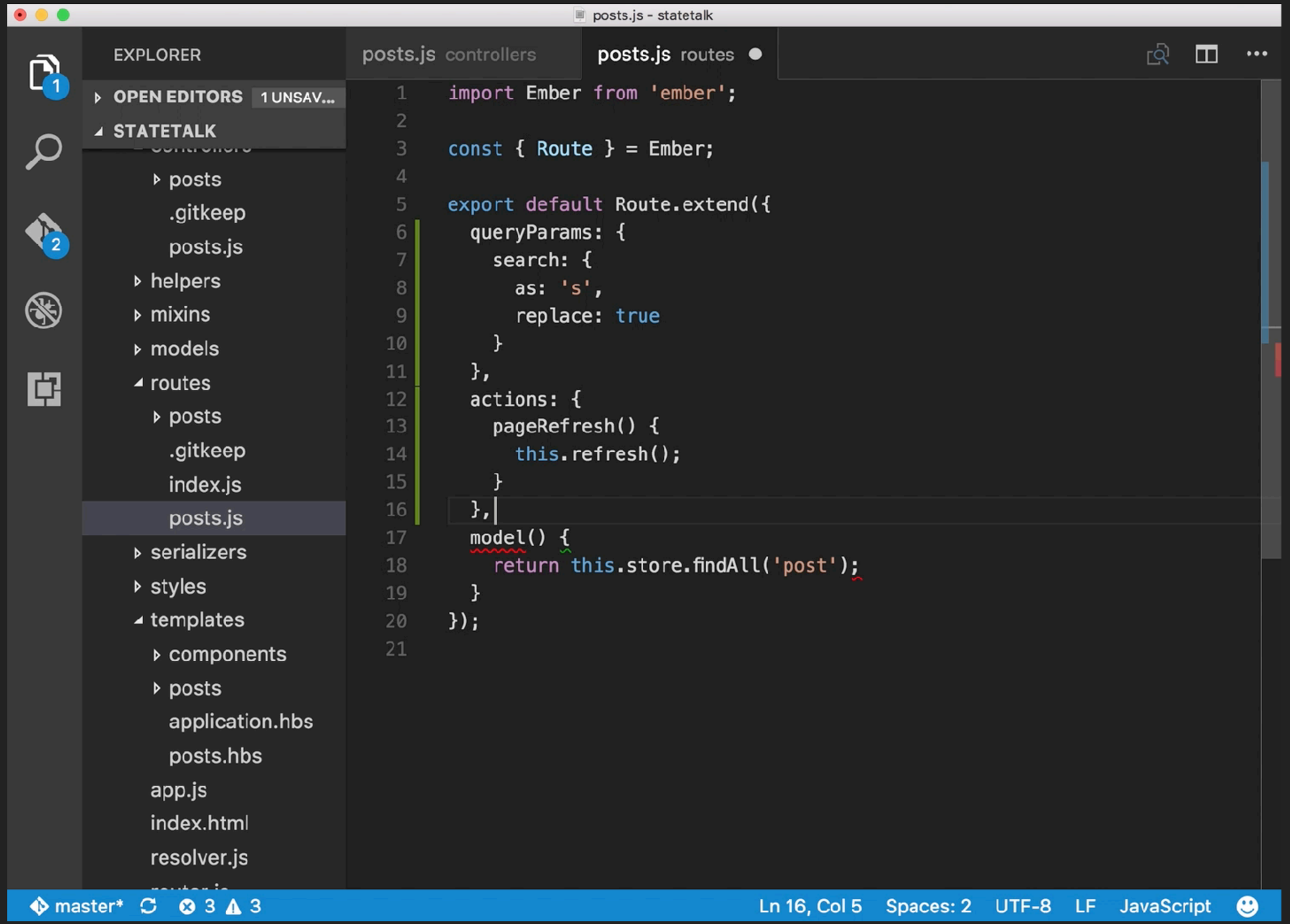
10

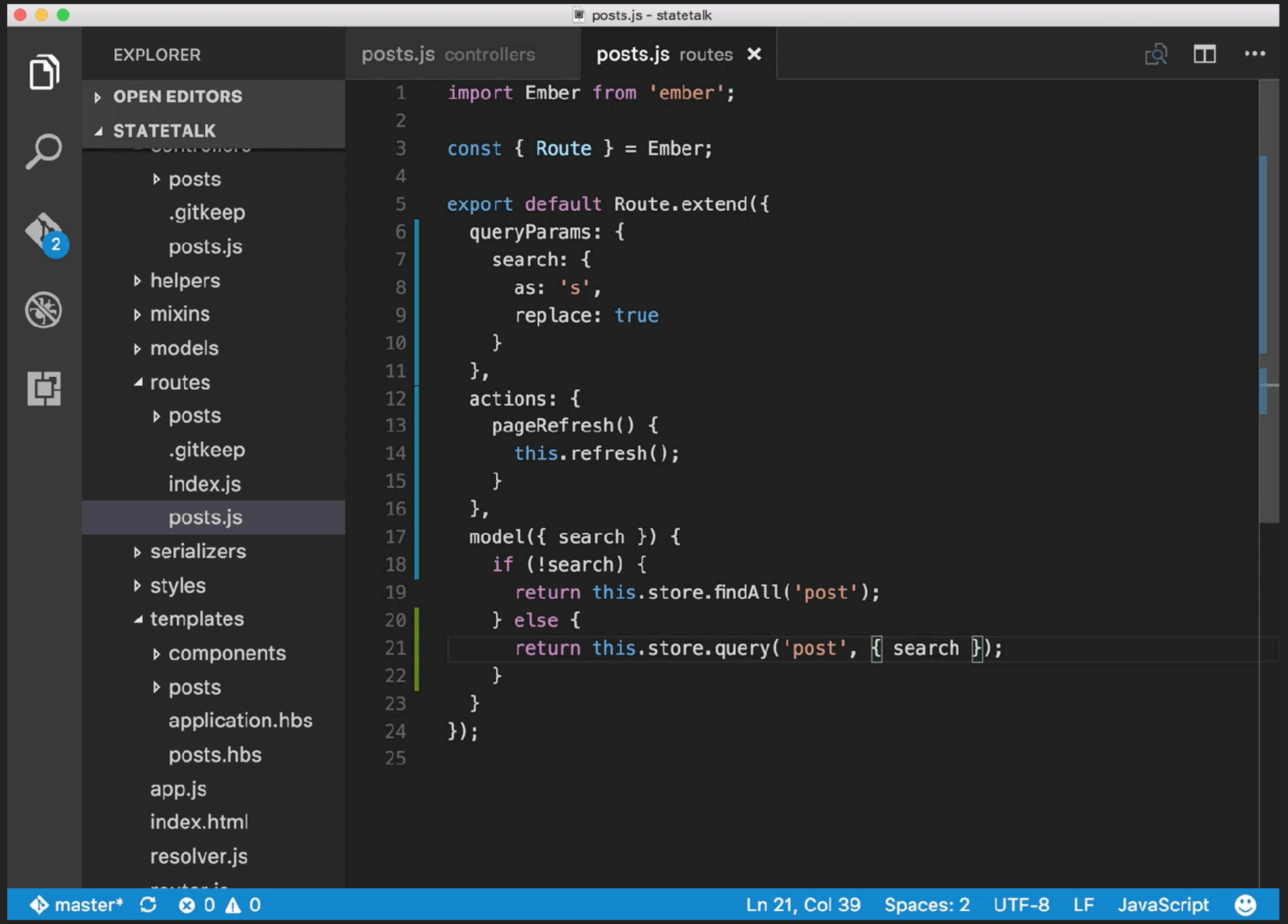
- Set up a queryparam on the posts route
- ▸ Typing 3+ letters in the search field should result in the list
- of posts being filtered
- ▸ Adhere to data-down, actions up
- ▸ Bonus: Debounce if you know what that means




posts.js - statetalk
EXPLORER
OPEN EDITORS 1 UNSAV...
STATETALK
app
adapters
components
controllers
posts
.gitkeep
posts.js
helpers
mixins
models
routes
posts
.gitkeep
index.js
posts.js
serializers
styles
templates
components
posts
application.hbs
posts.hbs
posts.js
1 import Ember from 'ember';
2
3 const { Controller, run: { debounce } } = Ember;
4
5 export default Controller.extend({
6 _triggerPageRefresh() {
7 this.send('pageRefresh');
8 },
9 actions: {
10 queuePageRefresh() {
11 debounce(this, '_triggerPageRefresh', 300);
12 }
13 }
14 });
15
master*
0 0
Ln 11, Col 54
Spaces: 4
UTF-8
LF
JavaScript


posts.js - statetalk
EXPLORER
OPEN EDITORS 1 UNSAV...
STATETALK
posts
.gitkeep
posts.js
helpers
mixins
models
routes
posts
.gitkeep
index.js
posts.js
serializers
styles
templates
components
posts
application.hbs
posts.hbs
app.js
index.html
resolver.js
posts.js controllers
posts.js routes
import Ember from 'ember';
const { Route } = Ember;
export default Route.extend({
queryParams: {
search: {
as: 's',
replace: true
}
},
model() {
return this.store.findAll('post');
}
});
master*
0 0
Ln 11, Col 5
Spaces: 2
UTF-8
LF
JavaScript

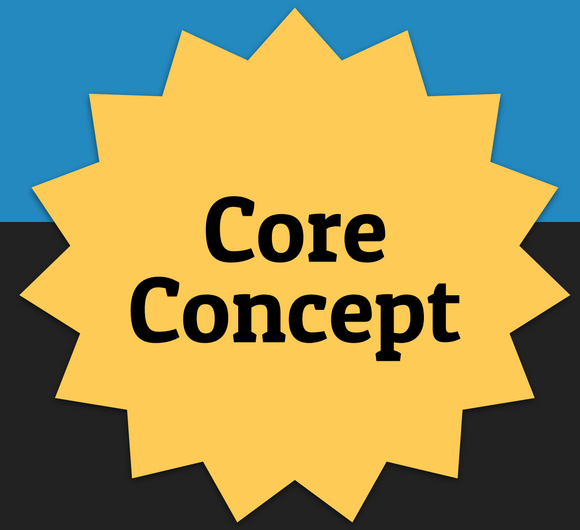

posts.js - statetalk
EXPLORER
OPEN EDITORS 1 UNSAV...
STATETALK
posts
.gitkeep
posts.js
helpers
mixins
models
routes
posts
.gitkeep
index.js
posts.js
serializers
styles
templates
components
posts
application.hbs
posts.hbs
app.js
index.html
resolver.js
posts.js controllers
posts.js routes
import Ember from 'ember';
const { Route } = Ember;
export default Route.extend({
queryParams: {
search: {
as: 's',
replace: true
}
},
actions: {
pageRefresh() {
this.refresh();
}
},
model() {
return this.store.findAll('post');
}
});
master*
3 3
Ln 16, Col 5
Spaces: 2
UTF-8
LF
JavaScript


posts.js - statetalk
EXPLORER
OPEN EDITORS
STATETALK
posts
.gitkeep
posts.js
helpers
mixins
models
routes
posts
.gitkeep
index.js
posts.js
serializers
styles
templates
components
posts
application.hbs
posts.hbs
app.js
index.html
resolver.js
posts.js controllers
posts.js routes
import Ember from 'ember';
const { Route } = Ember;
export default Route.extend({
queryParams: {
search: {
as: 's',
replace: true
}
},
actions: {
pageRefresh() {
this.refresh();
}
},
model({ search }) {
if (!search) {
return this.store.findAll('post');
} else {
return this.store.query('post', { search });
}
}
});
master*
0 0
Ln 21, Col 39
Spaces: 2
UTF-8
LF
JavaScript

Draft State

Draft State

Locally-stored Temporarily, will eventually be persisted



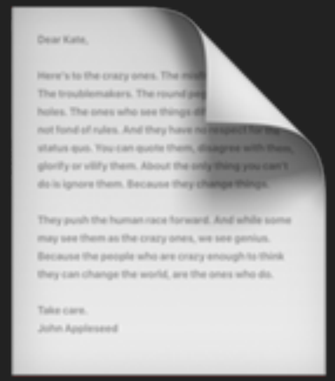
Examples

- A reply to an email message
- The body of a huge GitHub comment
- In-flight records

HOW TO IDENTIFY

- Will turn into persisted state
- High user effort
- Survives **transitions**
- Detrimental to UX if lost

Conceptual Model

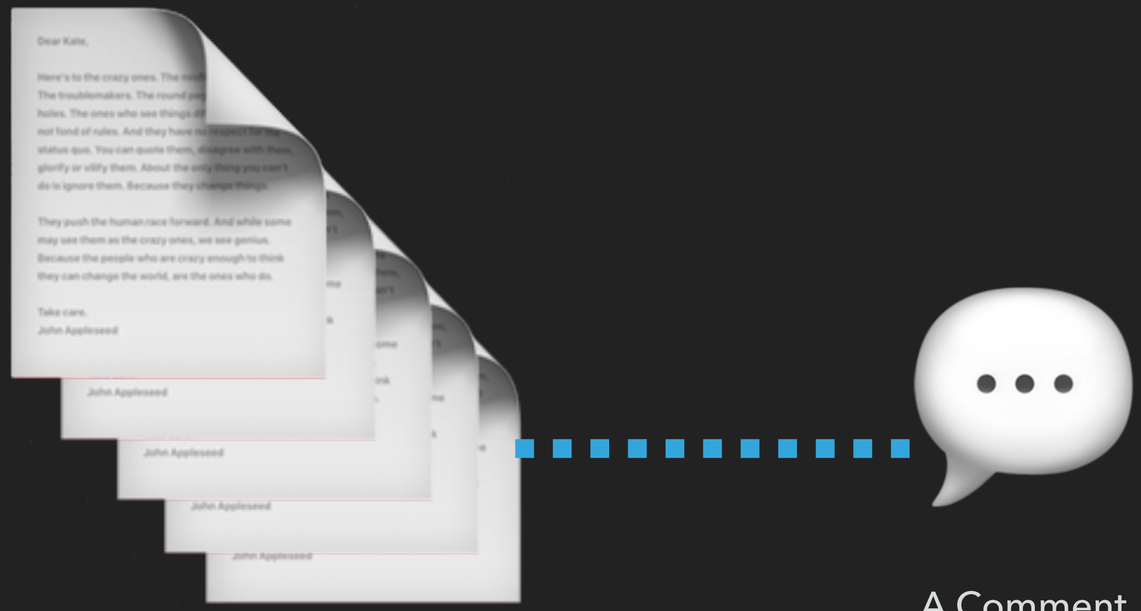


A Post

A Comment

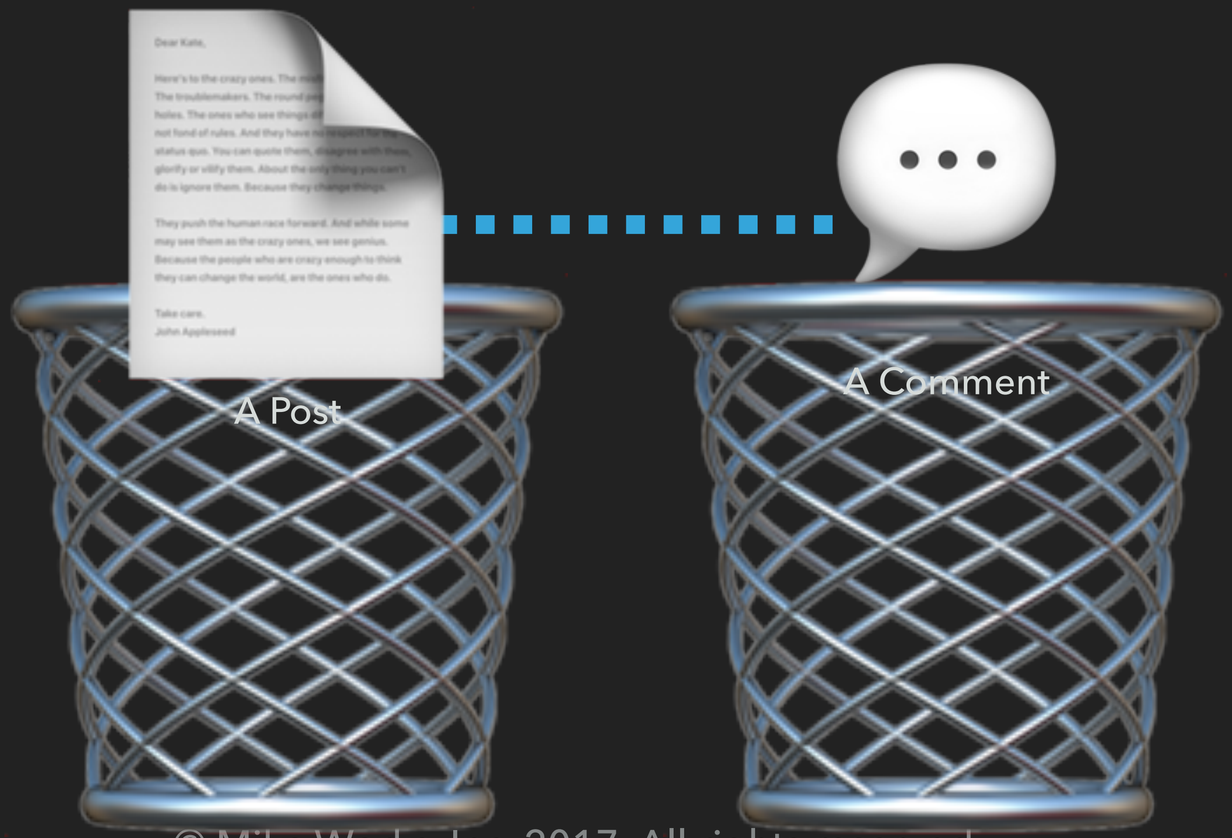
Conceptual Model

Conceptual Model



Conceptual Model

Conceptual Model



Conceptual Model - KEY CONCEPT

WEAK MAP

Conceptual Model - KEY CONCEPT

WEAK MAP

If an object that is being used as the key of a WeakMap key/value pair is only reachable by following a chain of references that start within that WeakMap, that key/value pair is inaccessible and is automatically removed from the WeakMap.

http://www.ecma-international.org/ecma-262/6.0/#sec-weakmap-objects

Conceptual Model - KEY CONCEPT

WEAK MAP

If an object that is being used as the key of a WeakMap key/value pair is only reachable by following a chain of references that start within that WeakMap, that key/value pair is inaccessible and is automatically removed from the WeakMap.

http://www.ecma-international.org/ecma-262/6.0/#sec-weakmap-objects

Conceptual Model - KEY CONCEPT

WEAK MAP

reachable by following a chain of references that start within that WeakMap, that key/value pair is inaccessible and is automatically removed from the WeakMap.

```
// Create
let comm
// Post is
commen
```

http://www.ecma-international.org/ecma-262/6.0/#sec-weakmap-objects

Conceptual Model - KEY CONCEPT

WEAK MAP*

A Post

A Comment

```
// Store the value on the key
post._some_random_property_name = comment;

// Access the value, given the key
WeakMap.prototype.get = function(key) {
  return key._some_random_property_name;
};
```

The solution for Ember Apps

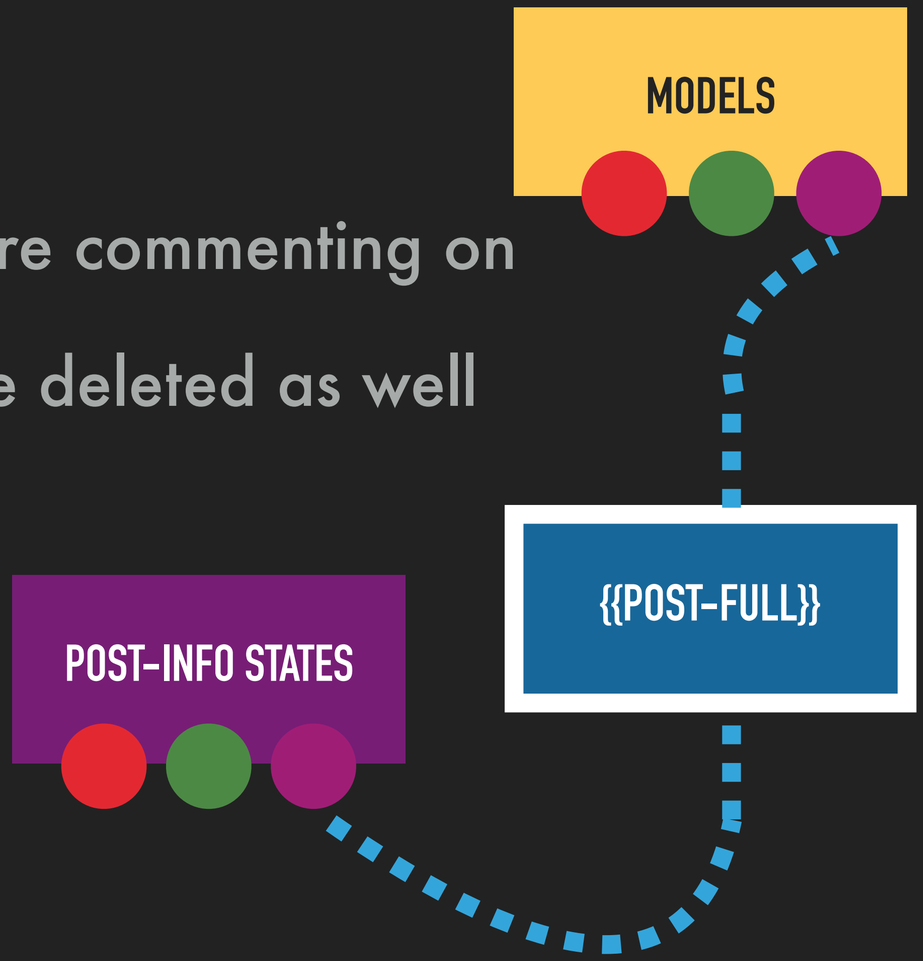
@thoov

@stefanpenner

- **ember-state-services**
- Built on top of WeakMap
- **StateFactory** - creates draft state objects
- **stateFor** - computed property that returns a particular state

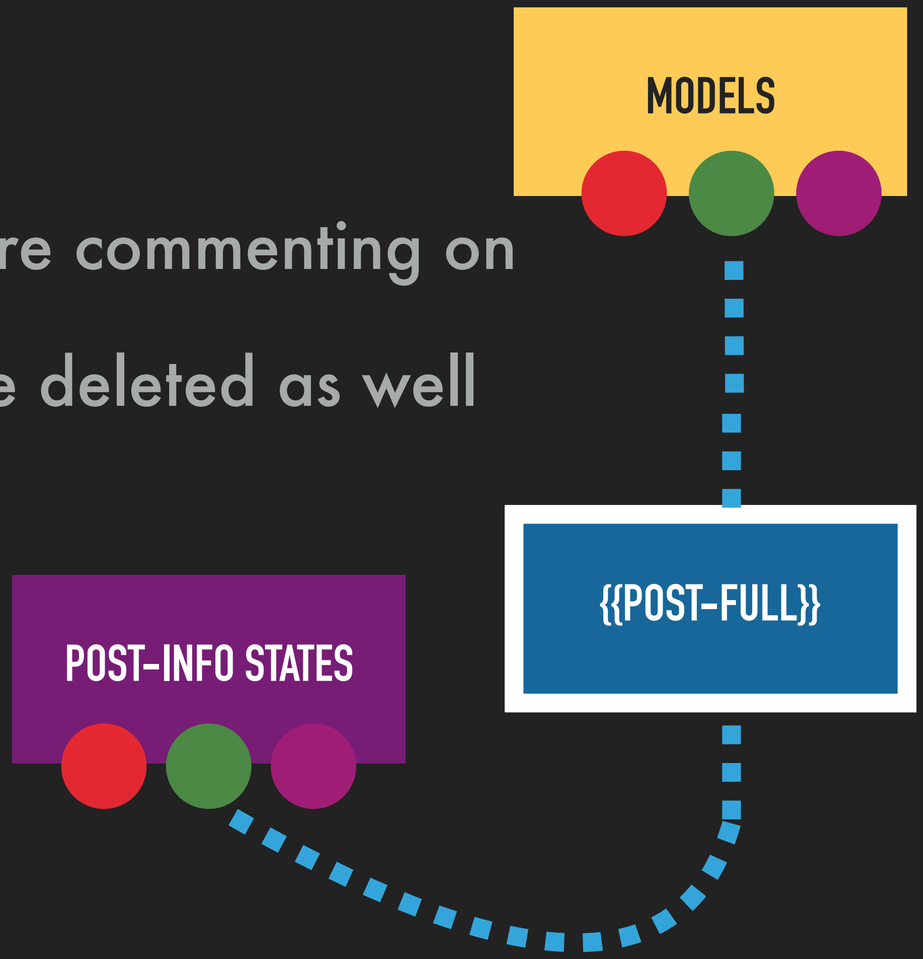
Enhancing our App

- They must “stick” with the post they’re commenting on
- If we delete a post, the draft must be deleted as well
- No leaks



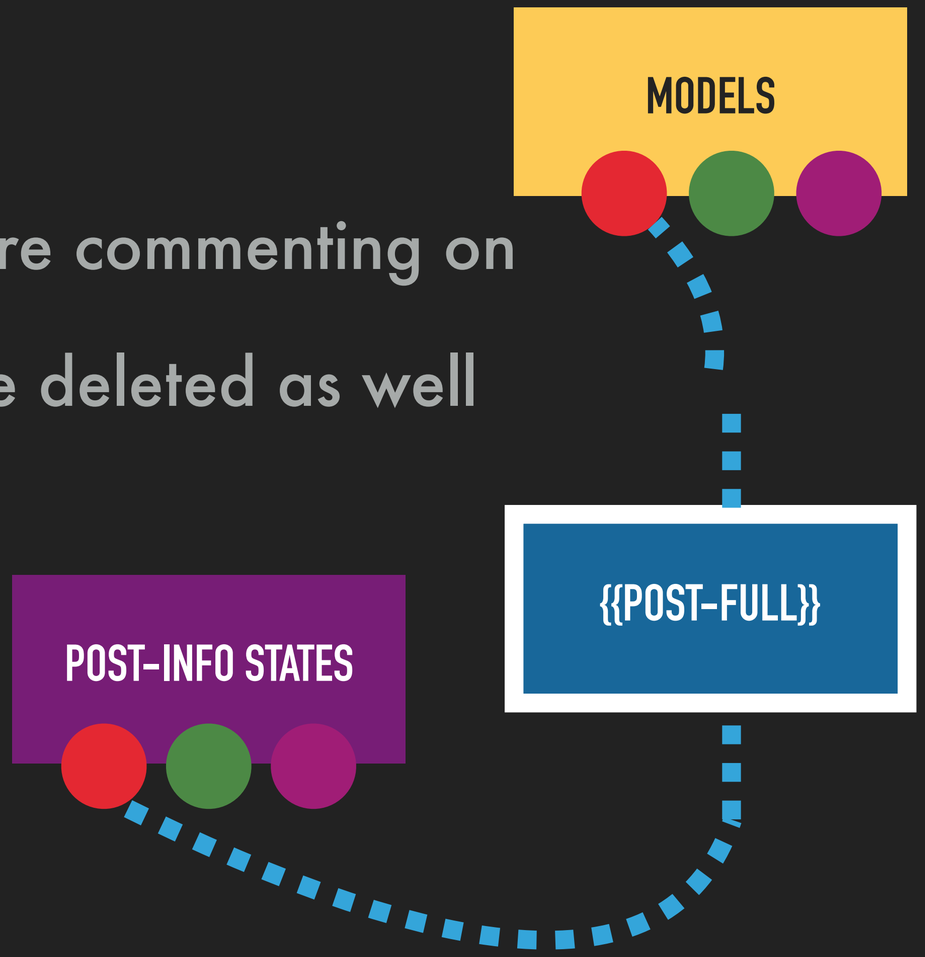
Enhancing our App

- They must “stick” with the post they’re commenting on
- If we delete a post, the draft must be deleted as well
- No leaks



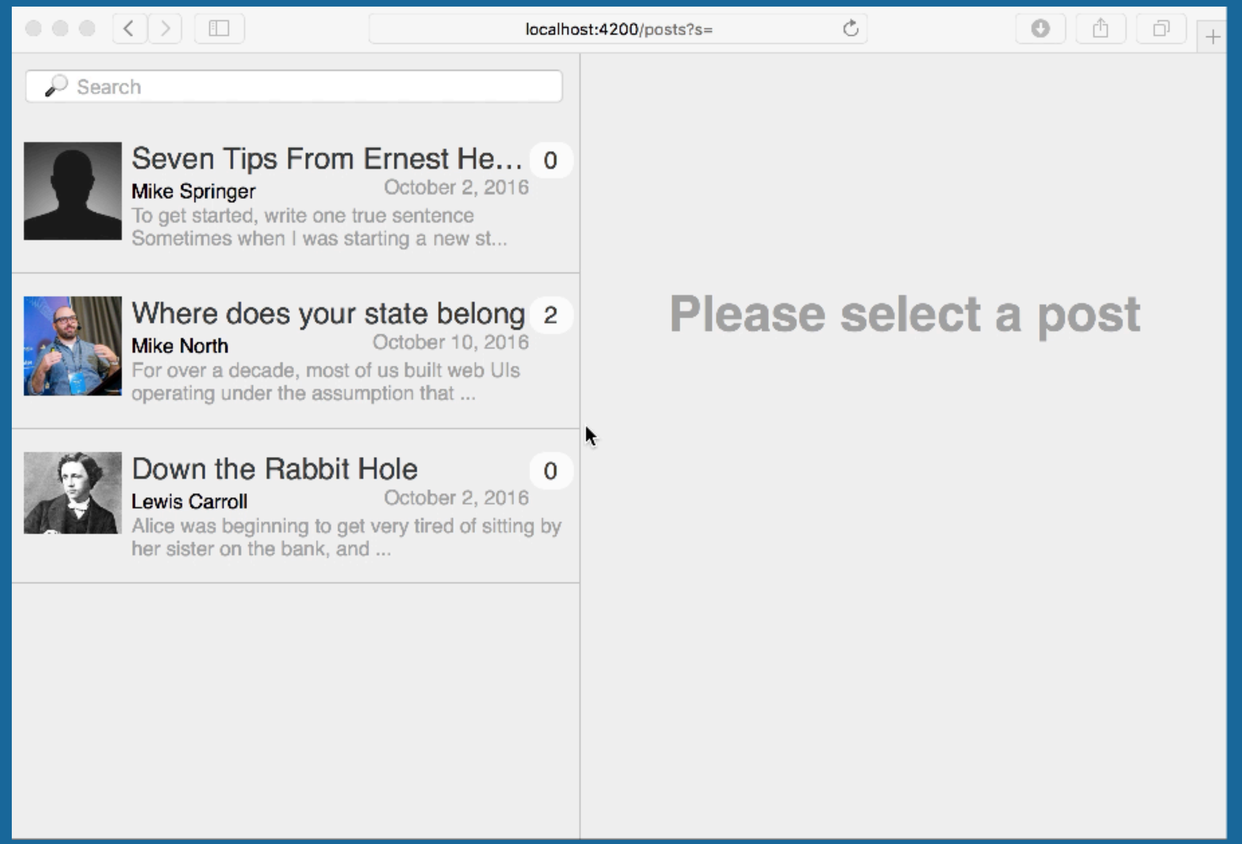
Enhancing our App

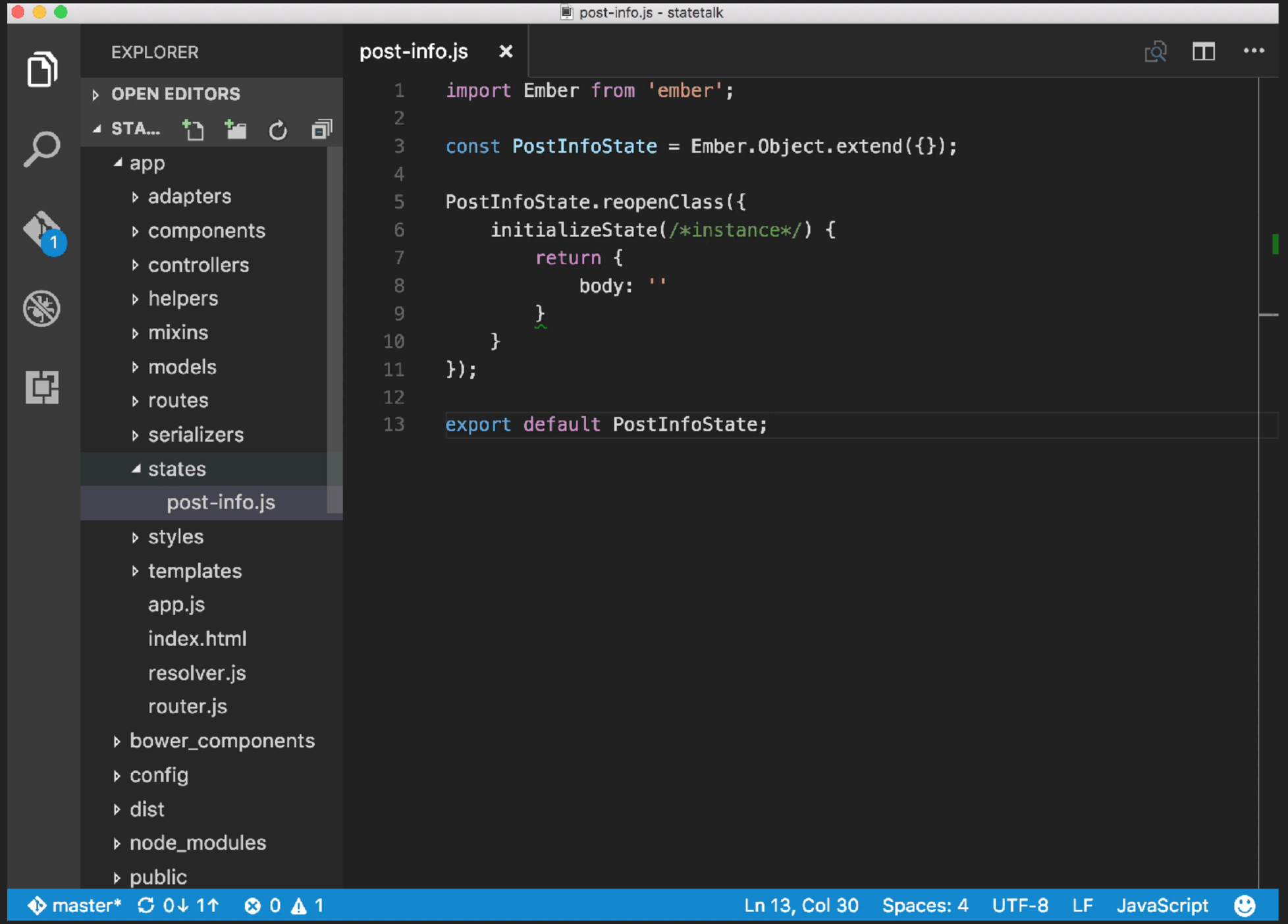
- They must “stick” with the post they’re commenting on
- If we delete a post, the draft must be deleted as well
- No leaks

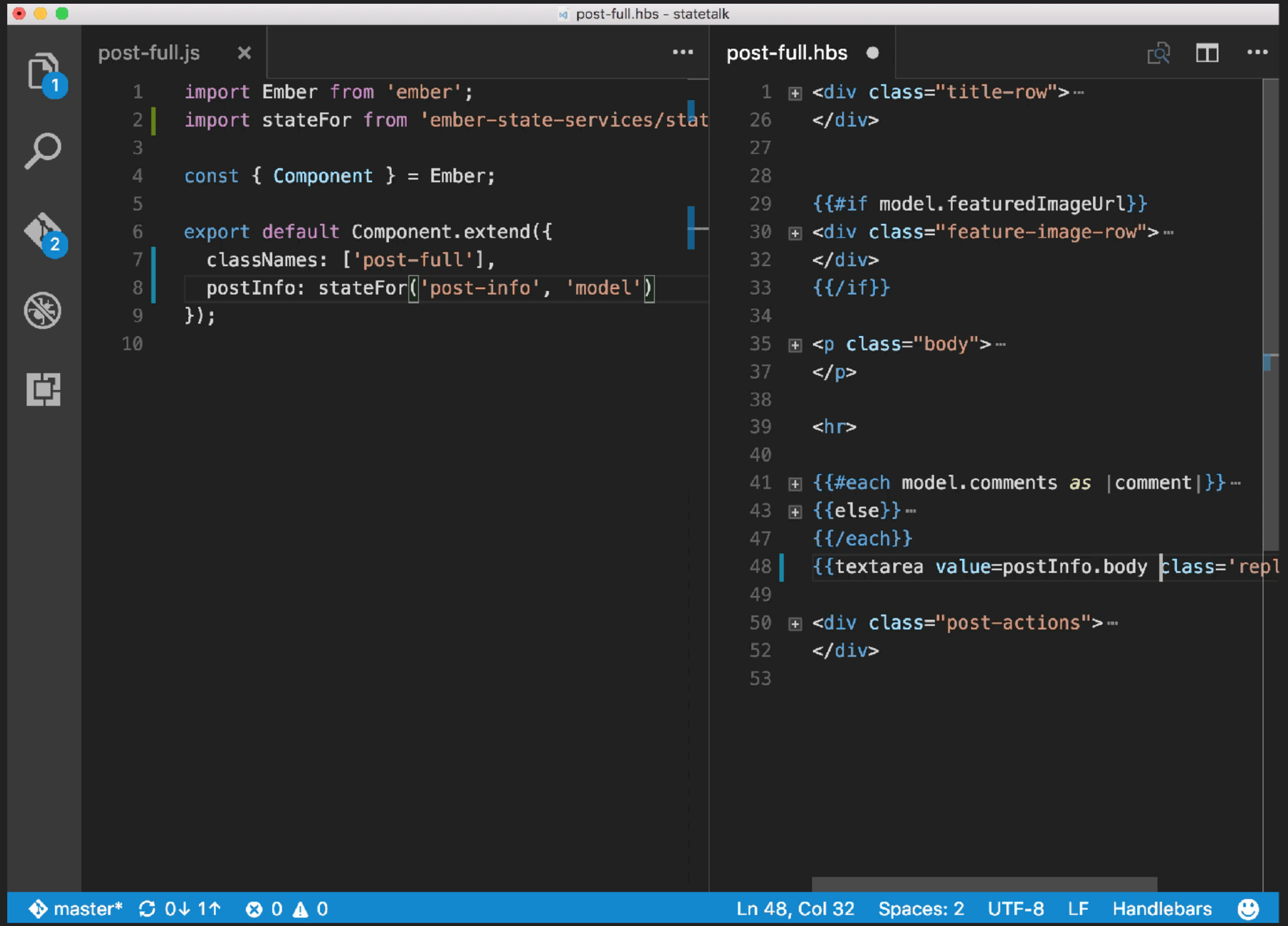


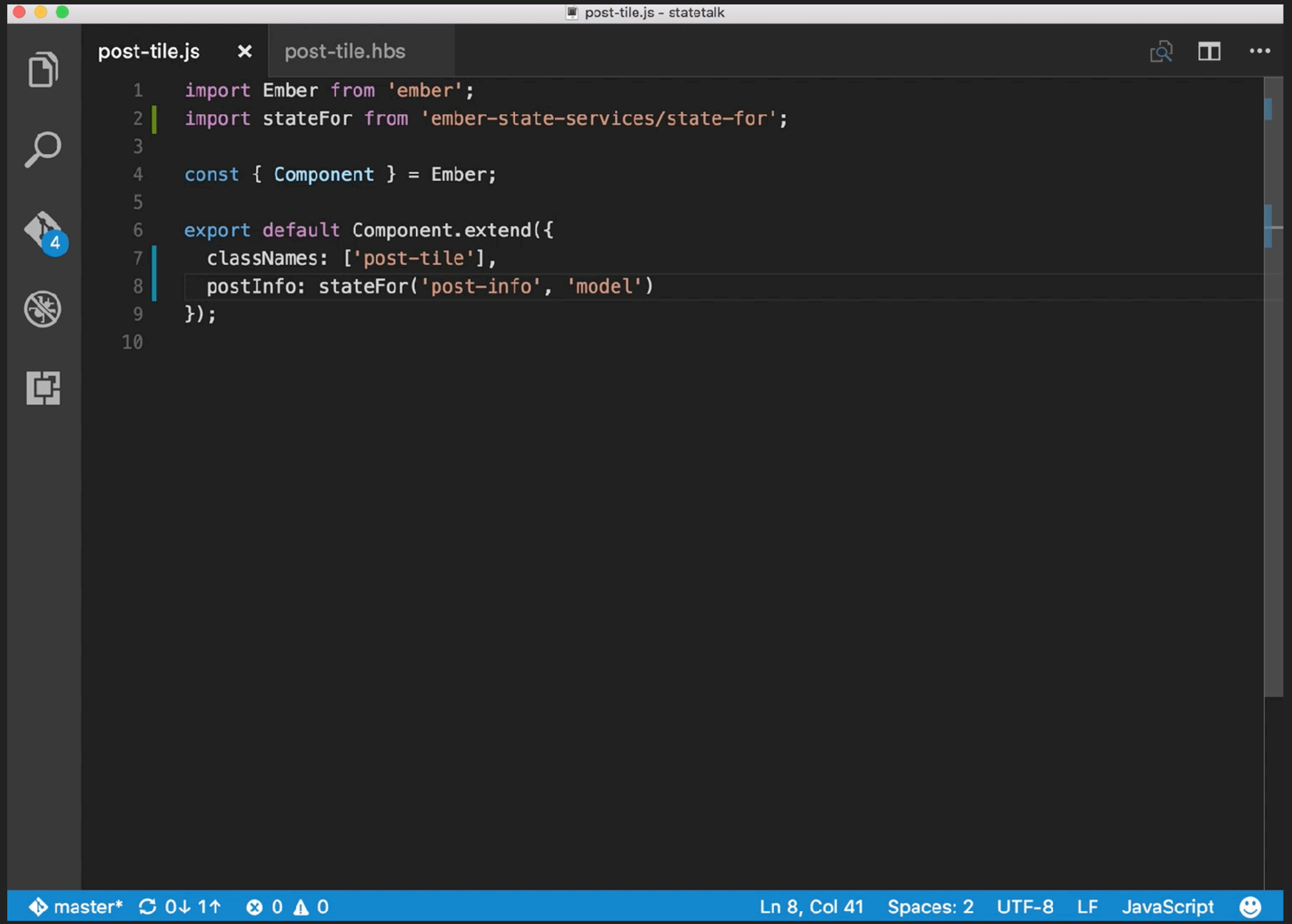
Draft Comments

- Using ember-state-services, allow comments to be “saved” even if we navigate away from (and back to) a post
- Show an indication in the list of posts whether there’s draft data present
- Avoid memory leaks




post-info.js - statetalk
EXPLORER
OPEN EDITORS
STA...
app
adapters
components
controllers
helpers
mixins
models
routes
serializers
states
post-info.js
styles
templates
app.js
index.html
resolver.js
router.js
bower_components
config
dist
node_modules
public
post-info.js
1 import Ember from 'ember';
2
3 const PostInfoState = Ember.Object.extend({});
4
5 PostInfoState.reopenClass({
6 initializeState(/*instance*/) {
7 return {
8 body: ''
9 }
10 }
11 });
12
13 export default PostInfoState;
master* 0↓ 1↑ 0 1
Ln 13, Col 30 Spaces: 4 UTF-8 LF JavaScript


post-full.hbs - statetalk
post-full.js
1 import Ember from 'ember';
2 import stateFor from 'ember-state-services/stat
3
4 const { Component } = Ember;
5
6 export default Component.extend({
7 classNames: ['post-full'],
8 postInfo: stateFor('post-info', 'model')
9 });
10
post-full.hbs
1 …
26
27
28
29 {{#if model.featuredImageUrl}}
30 …
32
33 {{/if}}
34
35 …
37
38
39 <hr>
40
41 {{#each model.comments as |comment|}}…
43 {{else}}…
47 {{/each}}
48 {{textarea value=postInfo.body class='repl
49
50 …
52
53
master* 0↓ 1↑ 0 0 Ln 48, Col 32 Spaces: 2 UTF-8 LF Handlebars


post-tile.js - statetalk
post-tile.js
post-tile.hbs
import Ember from 'ember';
import stateFor from 'ember-state-services/state-for';
const { Component } = Ember;
export default Component.extend({
classNames: ['post-tile'],
postInfo: stateFor('post-info', 'model')
});
master* 0↓ 1↑ 0 0
Ln 8, Col 41 Spaces: 2 UTF-8 LF JavaScript

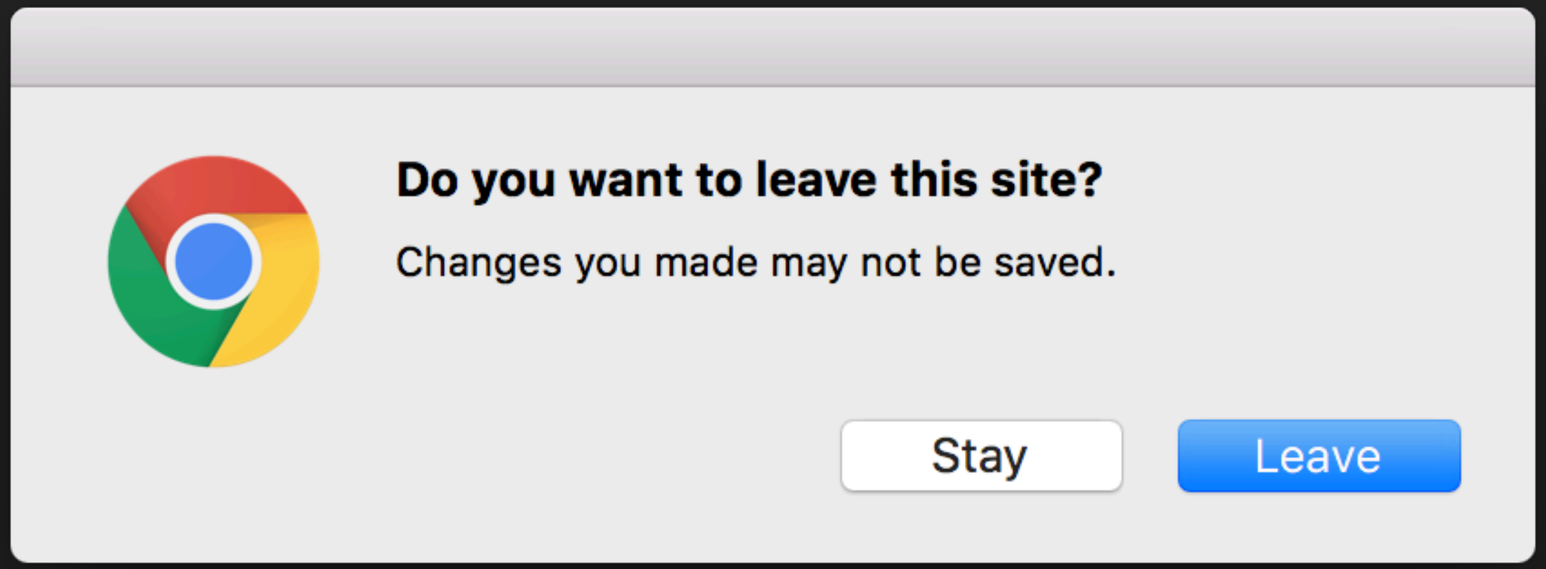
post-tile.hbs - statetalk

post-tile.js | post-tile.hbs

```
{{#link-to "posts.show" model.id}}
  <div class="author-image">
    <img src={{model.authorPhotoUrl}} alt="">
  </div>
  <div class="post-tile-content">
      <span class="comment-count">
        {{model.comments.length}}
      </span>
      <h3>{{model.title}}</h3>
    <span class="author-name">
      {{model.authorName}}
    </span>
    <span class="created-at pull-right">
      {{moment-format model.createdAt "LL"}}
    </span>
    <div class="post-tease">
      {{tease model.body chars=80}}...
    </div>
  </div>
{{/link-to}}
```

master* 0↓ 1↑ 0 0 Ln 1, Col 1 Spaces: 2 UTF-8 LF Handlebars

Making drafts even better

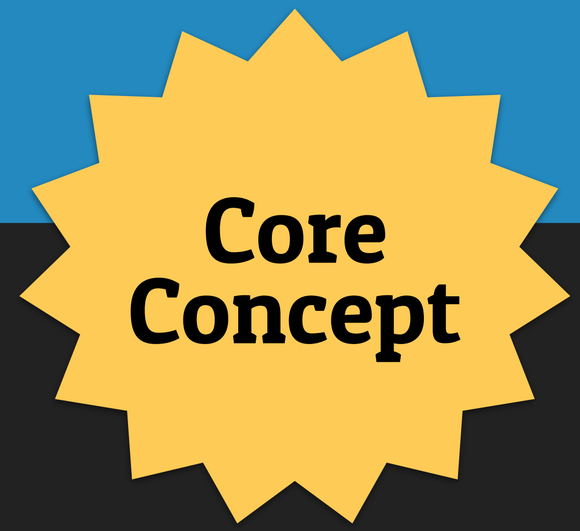


ember-state-services/issues/64

Persisted State

Persisted State

Stored in a long-living medium for a long time



Examples

- Anything stored in/on a…
  - Database/Disk
  - Local storage
  - Third party API

HOW TO IDENTIFY

- Lives longer than a session
- Tracked over time
- Seen by many users

Managing Comments in our App

- **POST** `/api/posts/:post_id/comments` for create
  - Draft cleared if successful
- **DELETE** `/api/posts/:post_id/comments/:comment_id` for delete
- Must work in high-latency environments

Customizing API URLs with Ember-Data

- URLS - an adapter concern
- New hooks for per-operation customization

```
urlForCreateRecord(     modelName, snapshot){ } // One
urlForDeleteRecord(id, modelName, snapshot){ } // One
urlForFindRecord  (id, modelName, snapshot){ } // One
urlForUpdateRecord(id, modelName, snapshot){ } // One
urlForDeleteRecord(id, modelName, snapshot){ } // One
urlForFindAll     (     modelName, snapshot){ } // List
urlForQueryRecord (     modelName, snapshot){ } // One
urlForQuery       (     modelName, snapshot){ } // List
```

Working with Snapshots

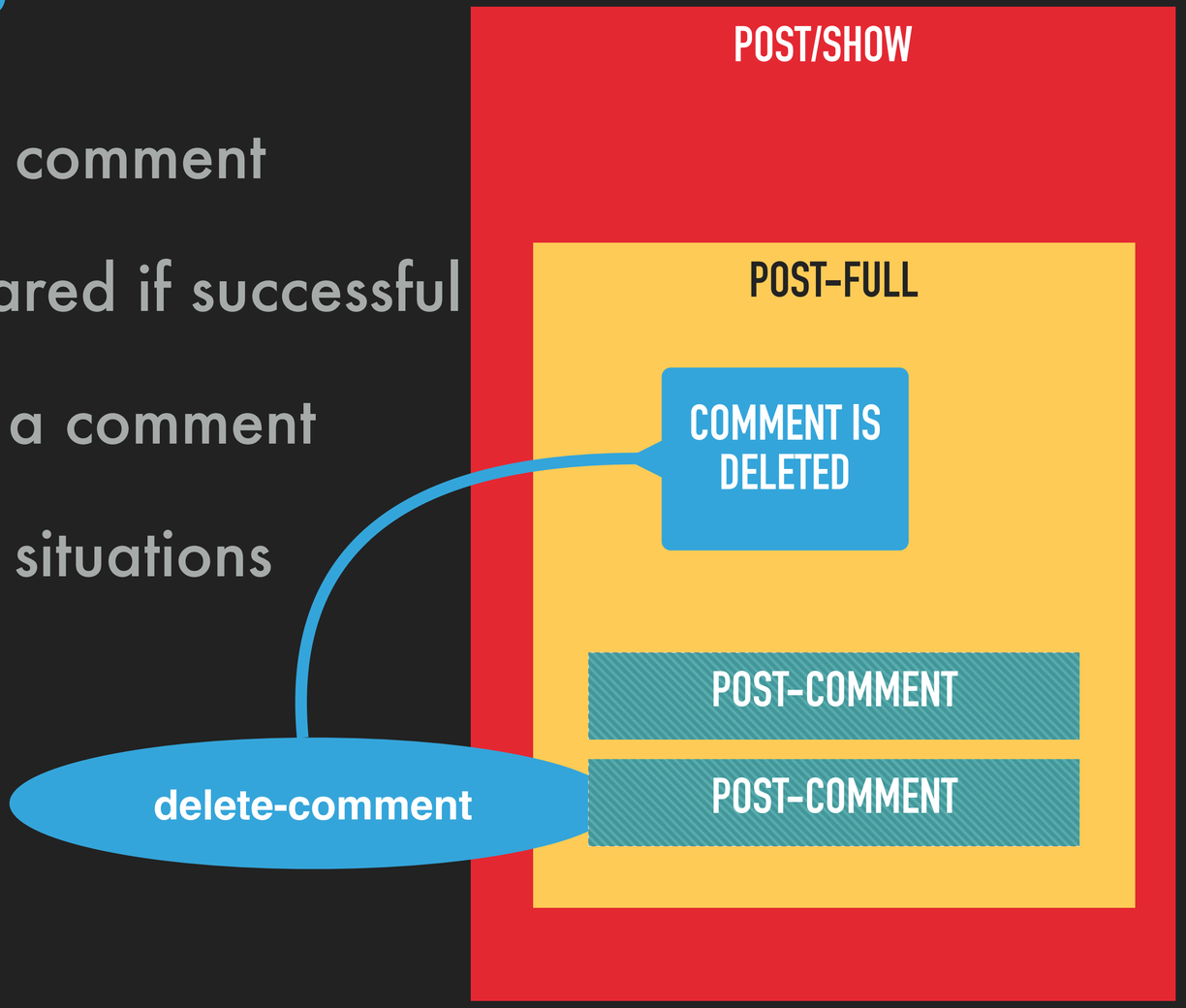
```
snapshot.attr('title'); /// attribute
snapshot.belongsTo('author'); // relationship
snapshot.hasMany('comment');
```

Enhancing our Basic App

- POST to API for create
  - Draft cleared if successful
- DELETE to API for delete
- Must work in high-latency

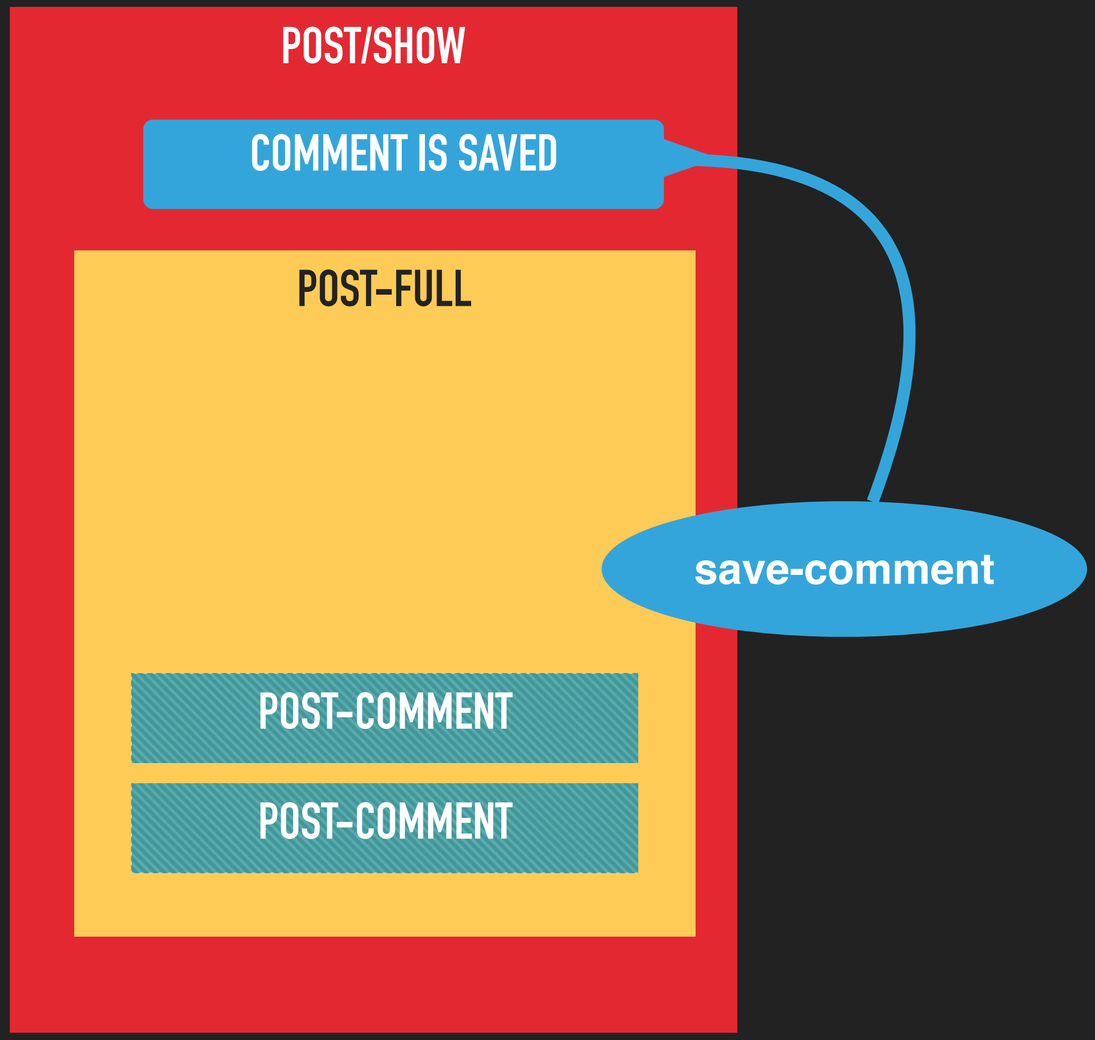
Enhancing our Basic App

- POST to API for creating a comment
  - Draft should only be cleared if successful
- DELETE to API for deleting a comment
- Must work for high-latency situations



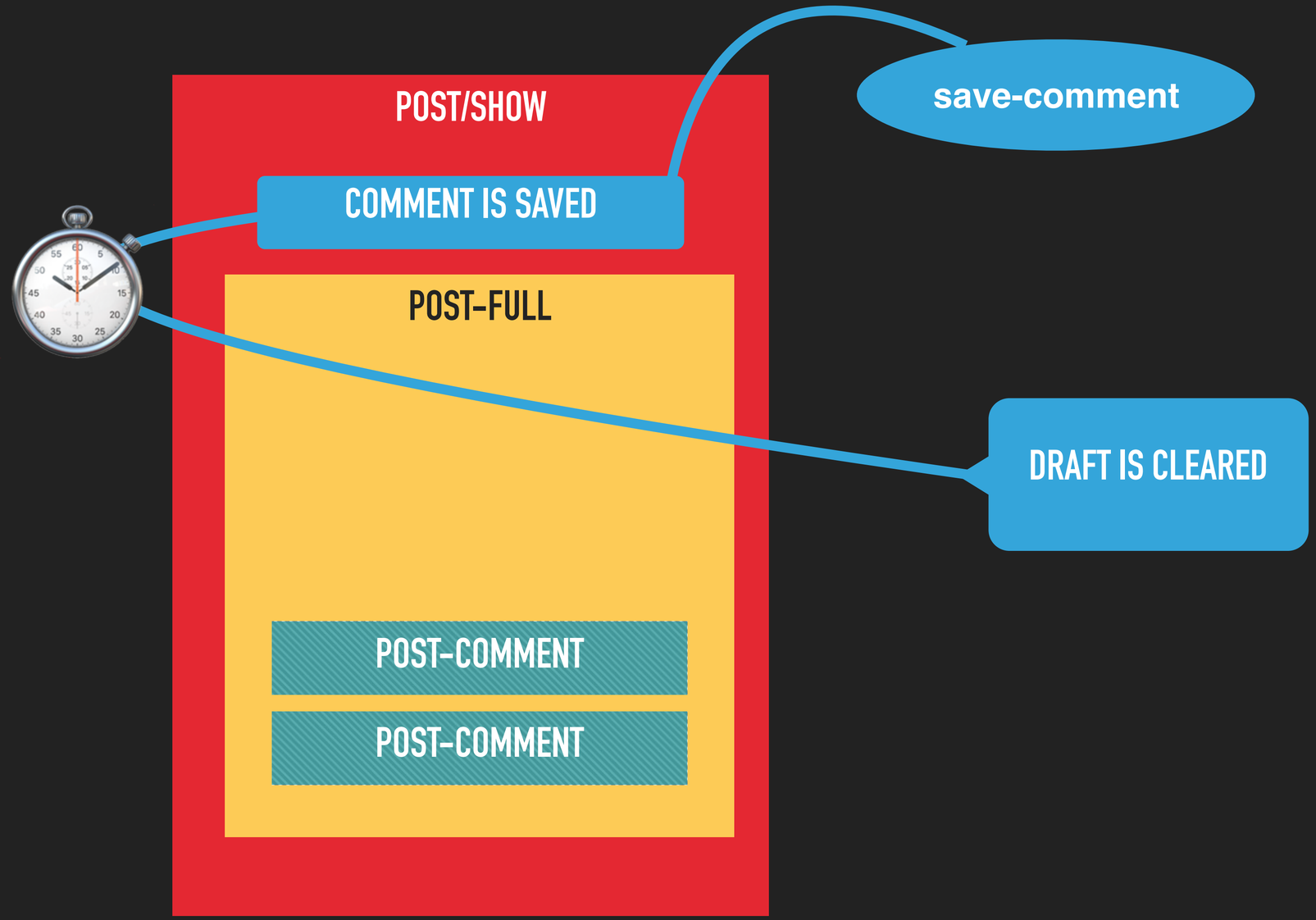
Enhancing our Basic App

- POST to API for creating a comment
  - Draft should only be cleared if successful
- DELETE to API for deleting a comment
- Must work for high-latency situations



Enhancing our Basic App

- POST to API for creating a comment
  - Draft should only be cleared if successful
- DELETE to API for deleting a comment
- Must work for high-latency situations



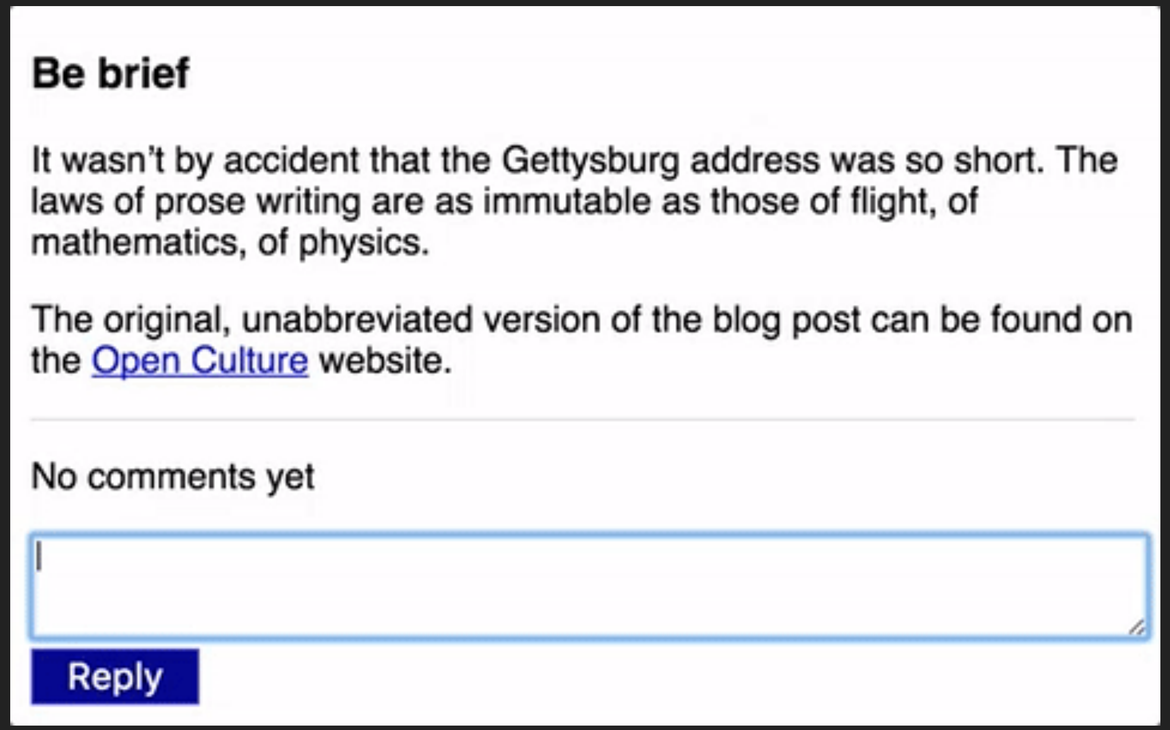
Persisted State - Best Practices

- Consider that things may change during async
- Build simple objects and **then** createRecord
- Consider in-flight records

Persisted State - Best Practices

- Double clicking “delete”
- Double clicking “reply”

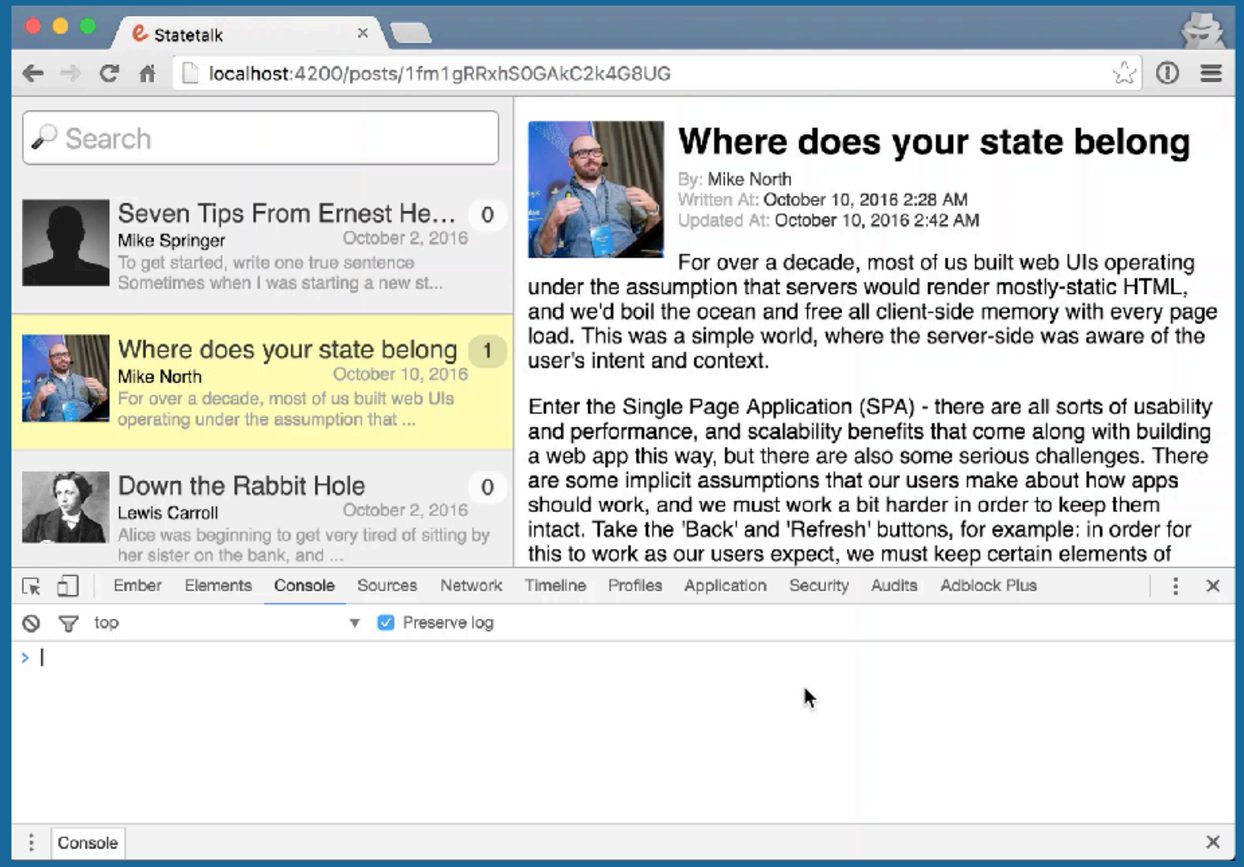
```
Uncaught                                    internal-model.js:502
EmberError {description: undefined, fileName:
undefined, lineNumber: undefined, message: "Attempted
to handle event `deleteRecord` on <state…ber689:35>
while in state root.deleted.inFlight. ", name:
"Error"…}
```



Persisting Comments

12

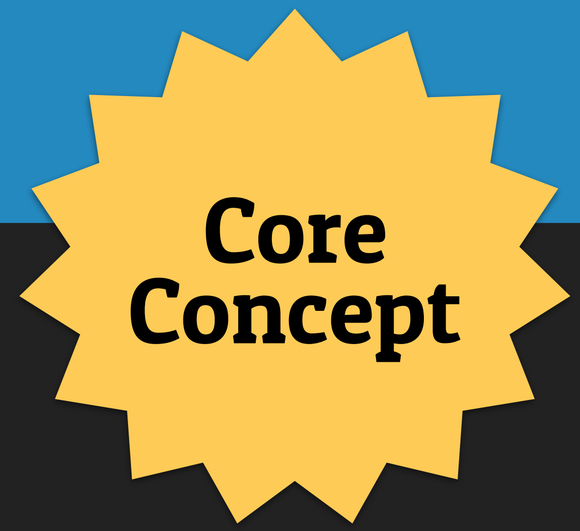
- Persist comments to the API, associating with the current user, and appropriate post
- Make sure the draft gets cleared as appropriate (set to blank is fine)
- This is an async operation. Consider what might happen with very high latency
- Add a new attribute (gravatarUrl) to the comment model & component



UI State

UI State

Short-lived, useful only “in the moment”, And relevant only to the UI

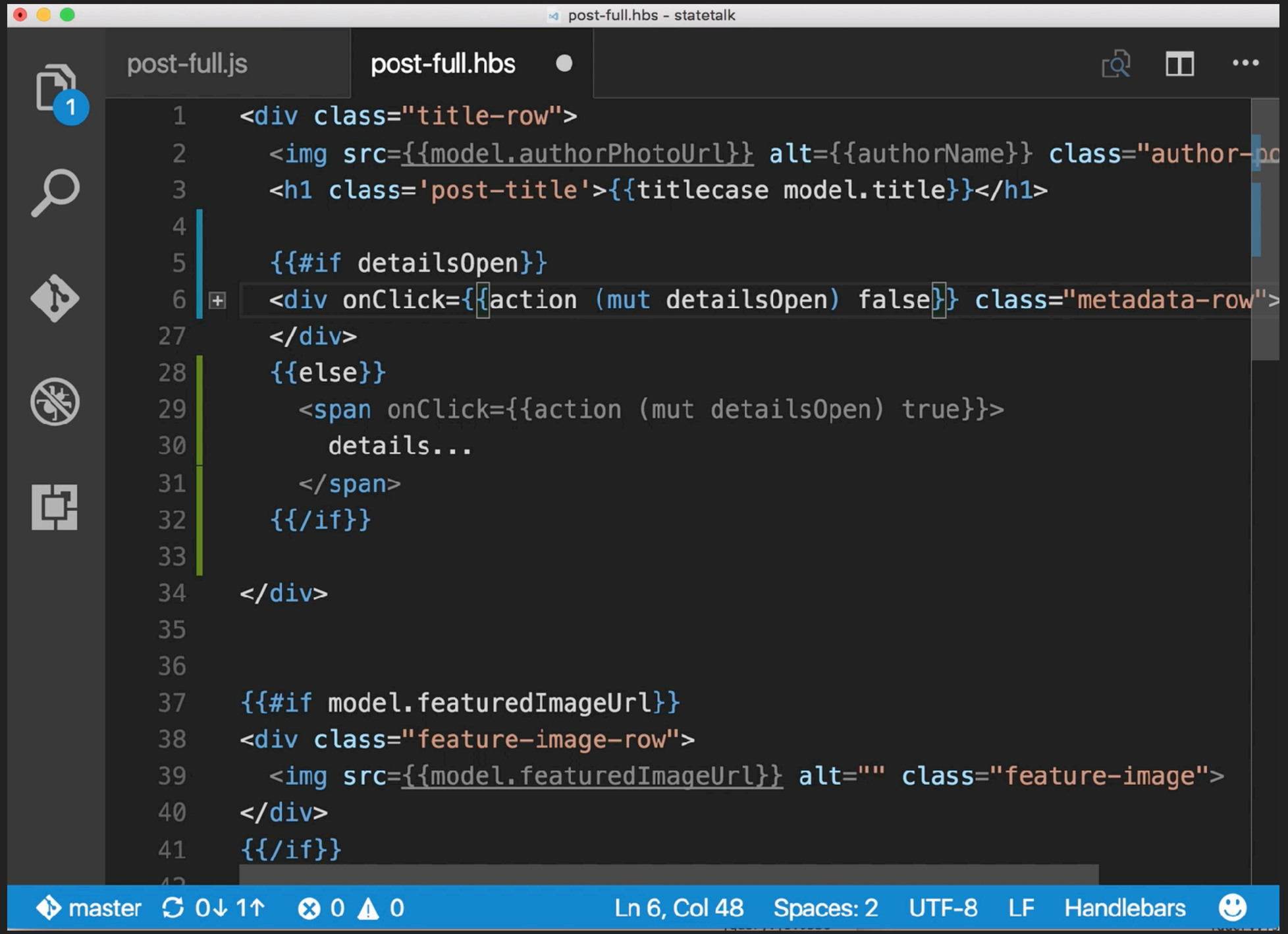


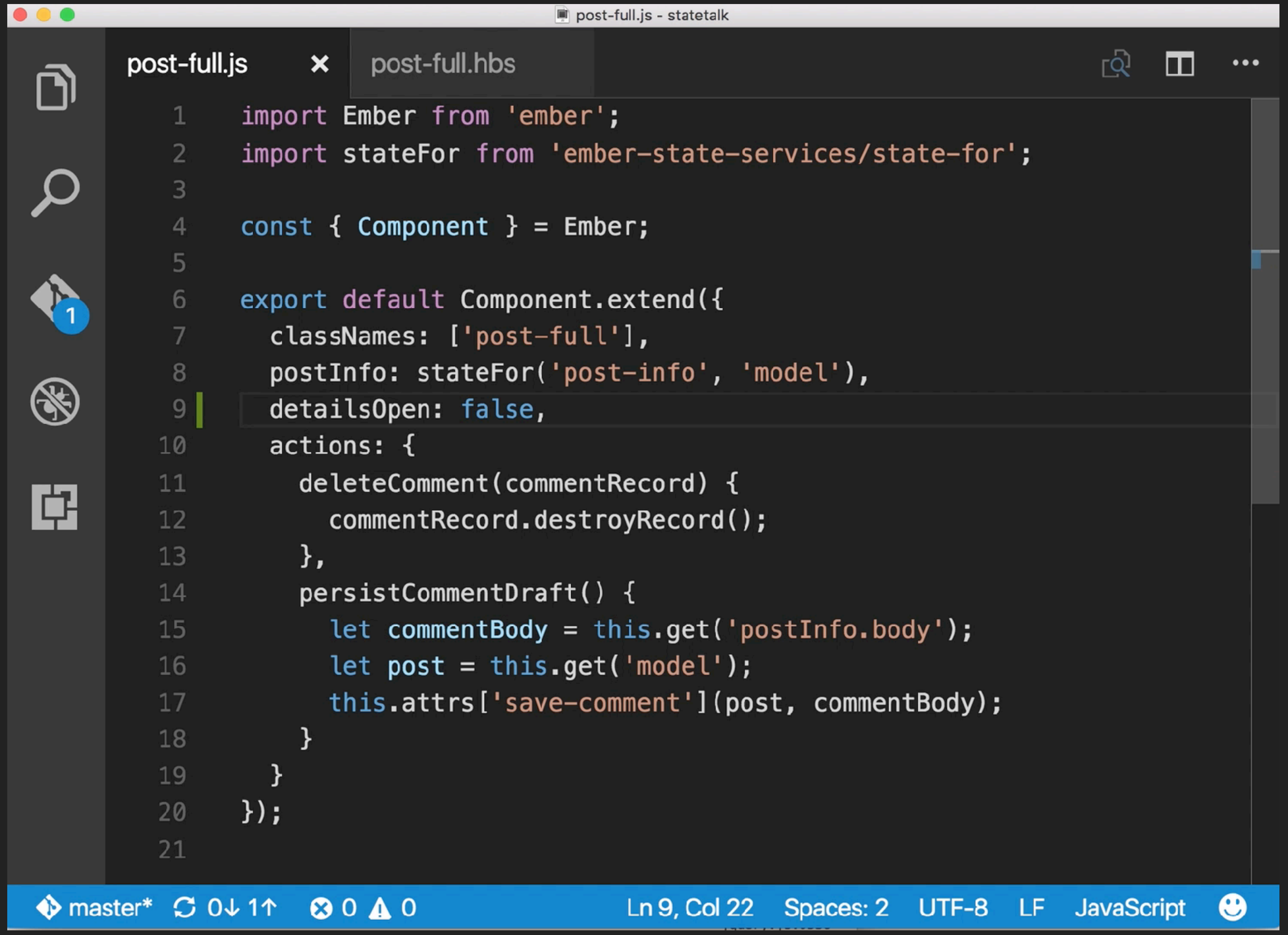
Examples

- Accordion expansion
- Scroll position
- Text selection & Focus
- Matthew Beale is typing…

How To Identify

- Short lived, only meaningful in the moment
- Easiest to re-create
- Potentially harmful if persisted/restored


post-full.hbs - statetalk
post-full.js
post-full.hbs
1
2 <img src={{model.authorPhotoUrl}} alt={{authorName}} class="author-pc
3 <h1 class='post-title'>{{titlecase model.title}}</h1>
4
5 {{#if detailsOpen}}
6 <div onClick={{action (mut detailsOpen) false}} class="metadata-row">
27
28 {{else}}
29 <span onClick={{action (mut detailsOpen) true}}>
30 details...
31
32 {{/if}}
33
34
35
36
37 {{#if model.featuredImageUrl}}
38
39 <img src={{model.featuredImageUrl}} alt="" class="feature-image">
40
41 {{/if}}
master 0↓ 1↑ 0 0 Ln 6, Col 48 Spaces: 2 UTF-8 LF Handlebars


post-full.js - statetalk
post-full.js
post-full.hbs
1 import Ember from 'ember';
2 import stateFor from 'ember-state-services/state-for';
3
4 const { Component } = Ember;
5
6 export default Component.extend({
7 classNames: ['post-full'],
8 postInfo: stateFor('post-info', 'model'),
9 detailsOpen: false,
10 actions: {
11 deleteComment(commentRecord) {
12 commentRecord.destroyRecord();
13 },
14 persistCommentDraft() {
15 let commentBody = this.get('postInfo.body');
16 let post = this.get('model');
17 this.attrs['save-comment'](post, commentBody);
18 }
19 }
20 });
21
master* 0↓ 1↑ 0 0 Ln 9, Col 22 Spaces: 2 UTF-8 LF JavaScript

localhost:4200/posts

Search

Seven Tips From Ernest He… 3

Mike Springer October 2, 2016

To get started, write one true sentence
Sometimes when I was starting a new st...

Where does your state belong 1

Mike North October 10, 2016

For over a decade, most of us built web UIs
operating under the assumption that ...

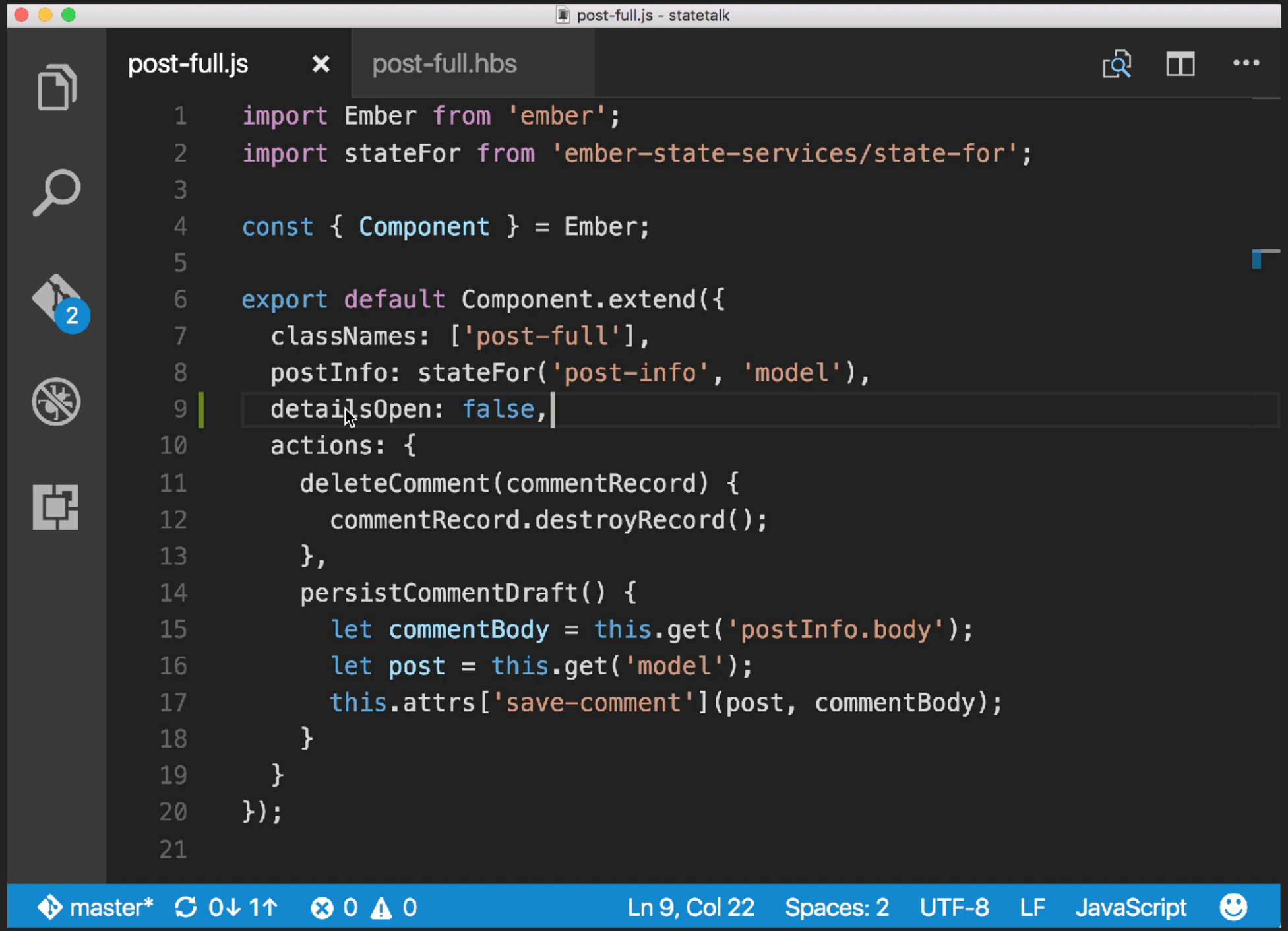
Down the Rabbit Hole 0

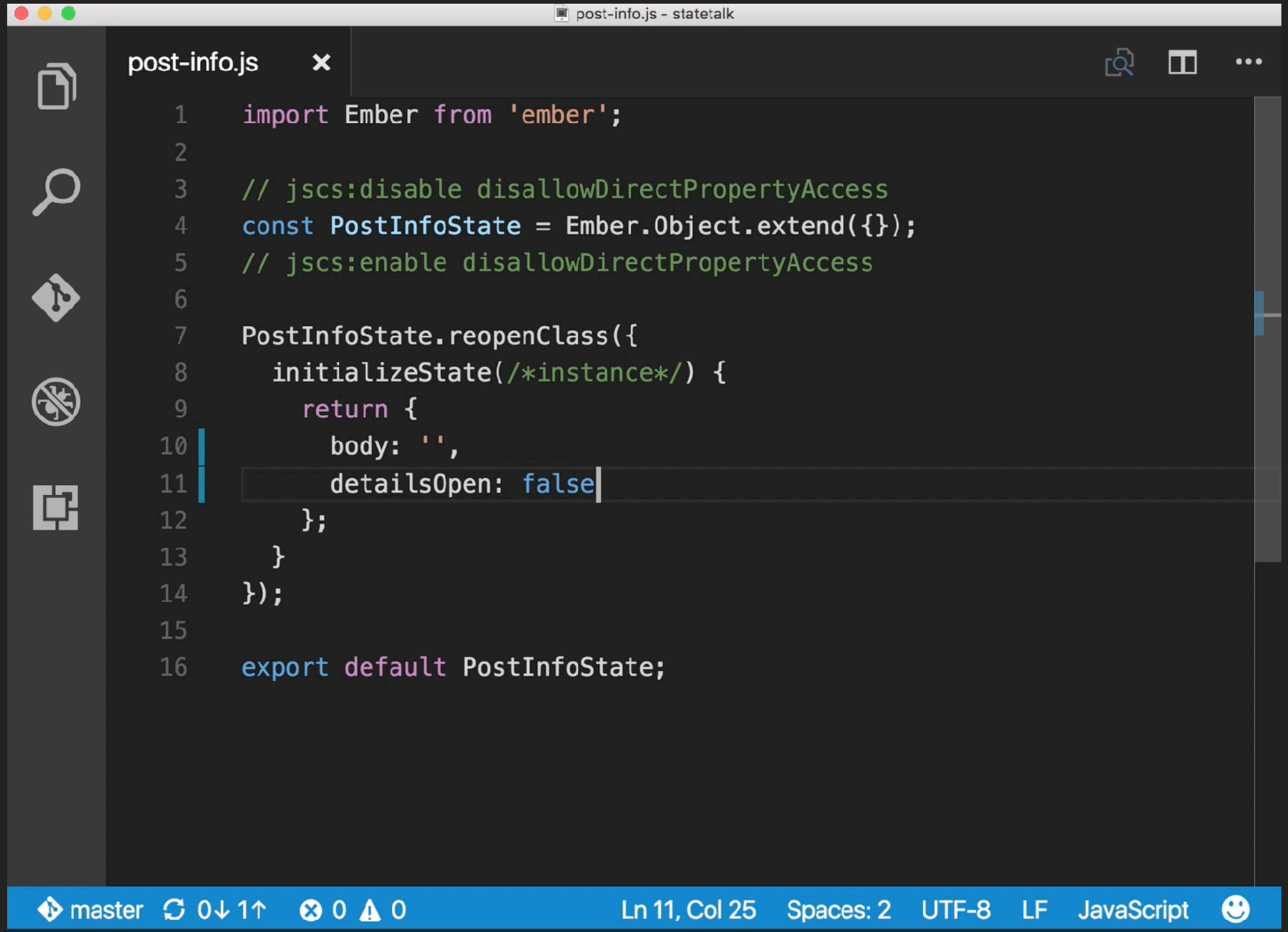
Lewis Carroll October 2, 2016

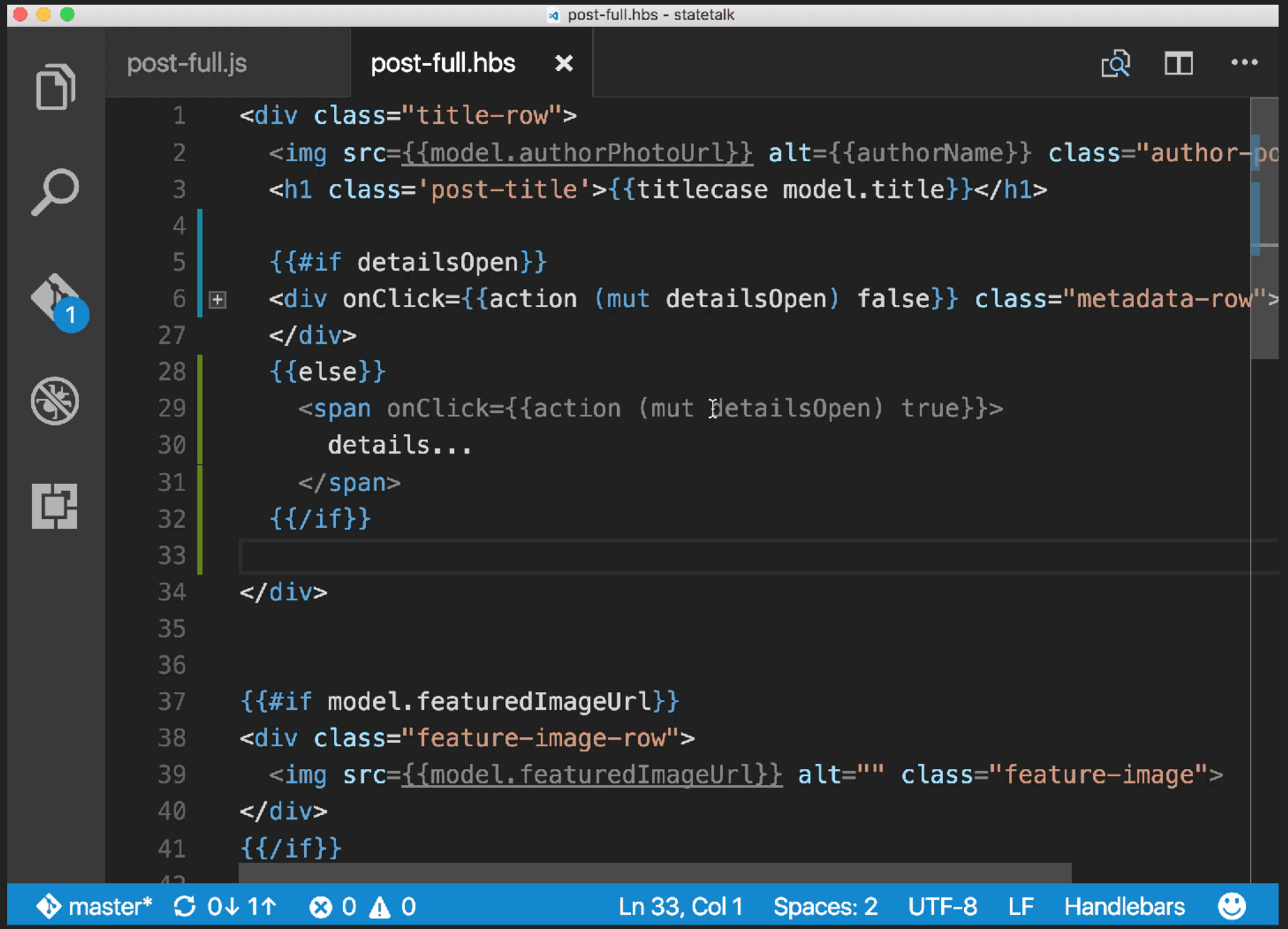
Alice was beginning to get very tired of sitting by
her sister on the bank, and ...

Please select a post

Keep component life-cycle in mind


post-full.js - statetalk
post-full.js
post-full.hbs
import Ember from 'ember';
import stateFor from 'ember-state-services/state-for';
const { Component } = Ember;
export default Component.extend({
classNames: ['post-full'],
postInfo: stateFor('post-info', 'model'),
detailsOpen: false,
actions: {
deleteComment(commentRecord) {
commentRecord.destroyRecord();
},
persistCommentDraft() {
let commentBody = this.get('postInfo.body');
let post = this.get('model');
this.attrs['save-comment'](post, commentBody);
}
}
});
master* 0↓ 1↑ 0 0 Ln 9, Col 22 Spaces: 2 UTF-8 LF JavaScript


post-info.js - statetalk
post-info.js
1 import Ember from 'ember';
2
3 // jscs:disable disallowDirectPropertyAccess
4 const PostInfoState = Ember.Object.extend({});
5 // jscs:enable disallowDirectPropertyAccess
6
7 PostInfoState.reopenClass({
8 initializeState(/*instance*/) {
9 return {
10 body: '',
11 detailsOpen: false
12 };
13 }
14 });
15
16 export default PostInfoState;
master 0↓ 1↑ 0 0 Ln 11, Col 25 Spaces: 2 UTF-8 LF JavaScript


post-full.hbs - statetalk
post-full.js
post-full.hbs
1
2 <img src={{model.authorPhotoUrl}} alt={{authorName}} class="author-pc
3 <h1 class='post-title'>{{titlecase model.title}}</h1>
4
5 {{#if detailsOpen}}
6 <div onClick={{action (mut detailsOpen) false}} class="metadata-row">
27
28 {{else}}
29 <span onClick={{action (mut detailsOpen) true}}>
30 details...
31
32 {{/if}}
33
34
35
36
37 {{#if model.featuredImageUrl}}
38
39 <img src={{model.featuredImageUrl}} alt="" class="feature-image">
40
41 {{/if}}
master* 0↓ 1↑ 0 0 Ln 33, Col 1 Spaces: 2 UTF-8 LF Handlebars

localhost:4200/posts/1fm1gRRxhS0GAkC2k4G8UG

Search

Seven Tips From Ernest He... 3

Mike Springer October 2, 2016

To get started, write one true sentence
Sometimes when I was starting a new st...

Where does your state belong 1

Mike North October 10, 2016

For over a decade, most of us built web UIs operating under the assumption that ...

Down the Rabbit Hole 0

Lewis Carroll October 2, 2016

Alice was beginning to get very tired of sitting by her sister on the bank, and ...

Where does your state belong

details...

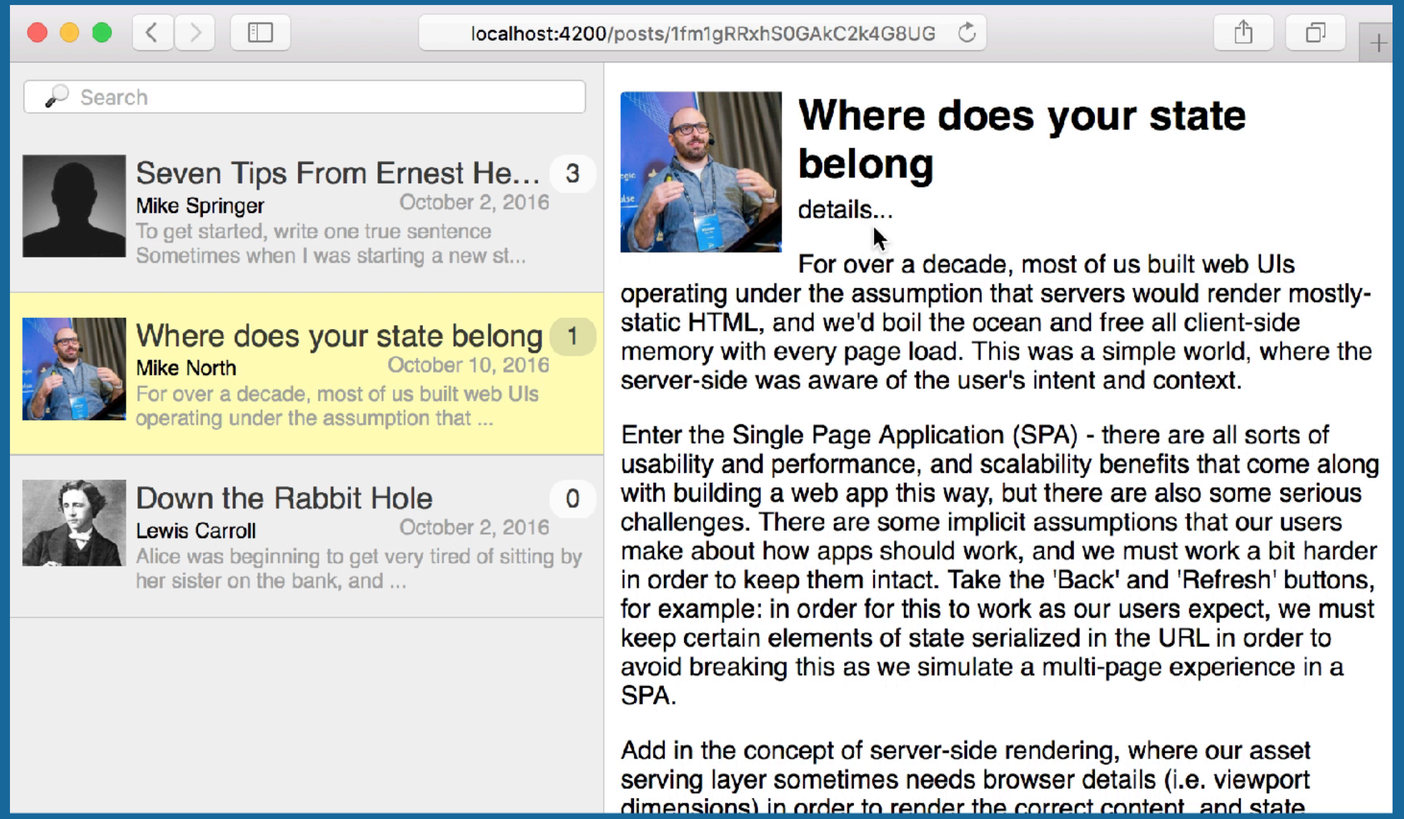
For over a decade, most of us built web UIs operating under the assumption that servers would render mostly-static HTML, and we'd boil the ocean and free all client-side memory with every page load. This was a simple world, where the server-side was aware of the user's intent and context.

Enter the Single Page Application (SPA) - there are all sorts of usability and performance, and scalability benefits that come along with building a web app this way, but there are also some serious challenges. There are some implicit assumptions that our users make about how apps should work, and we must work a bit harder in order to keep them intact. Take the 'Back' and 'Refresh' buttons, for example: in order for this to work as our users expect, we must keep certain elements of state serialized in the URL in order to avoid breaking this as we simulate a multi-page experience in a SPA.

Add in the concept of server-side rendering, where our asset serving layer sometimes needs browser details (i.e. viewport dimensions) in order to render the correct content, and state

Collapsible Metadata

- Make the metadata area at the top of the post-full component hideable/showable
- expanded/collapsed state should “stick” to each record for the duration of our UI session
- Initial state should be: **hidden**

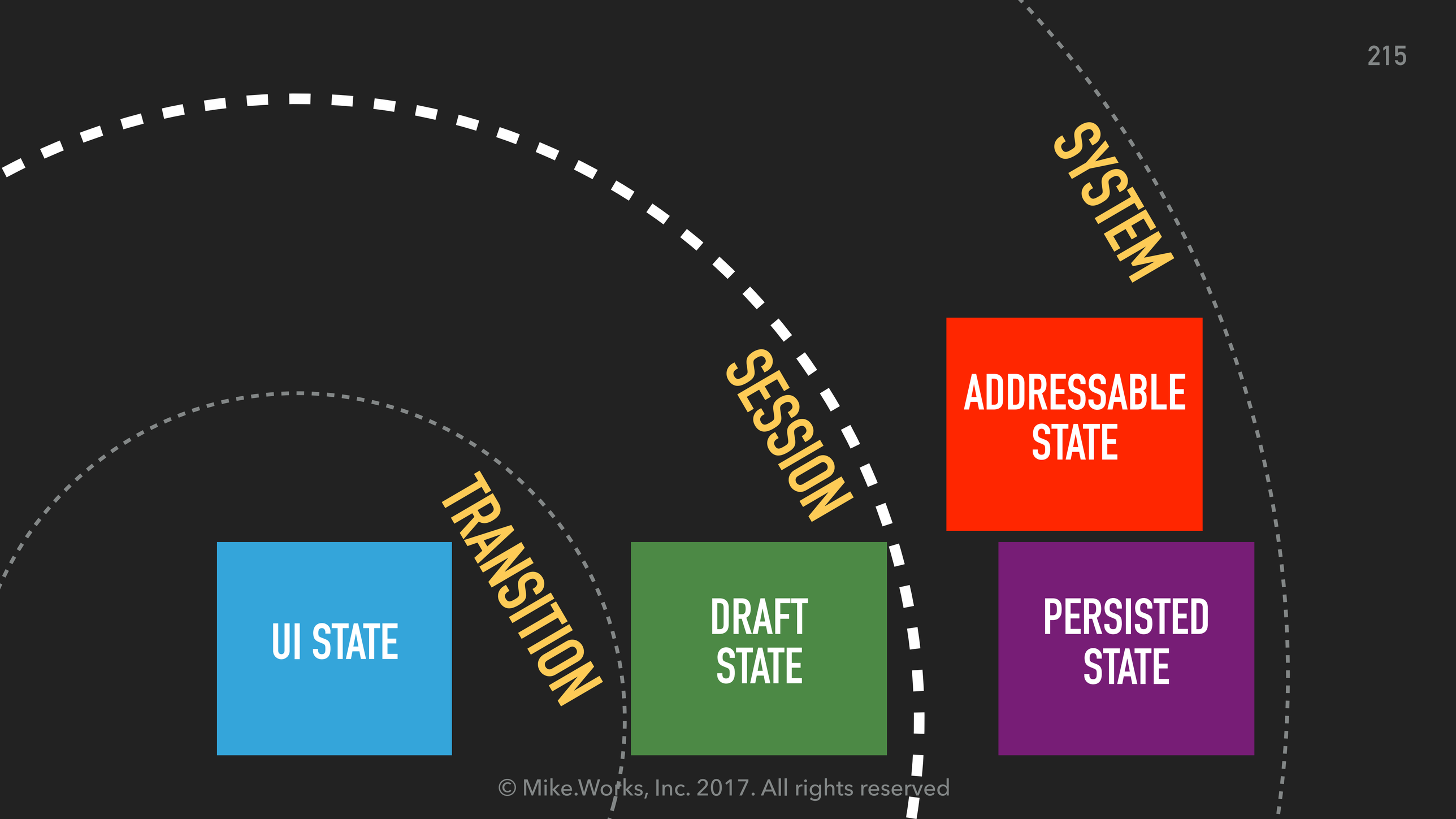


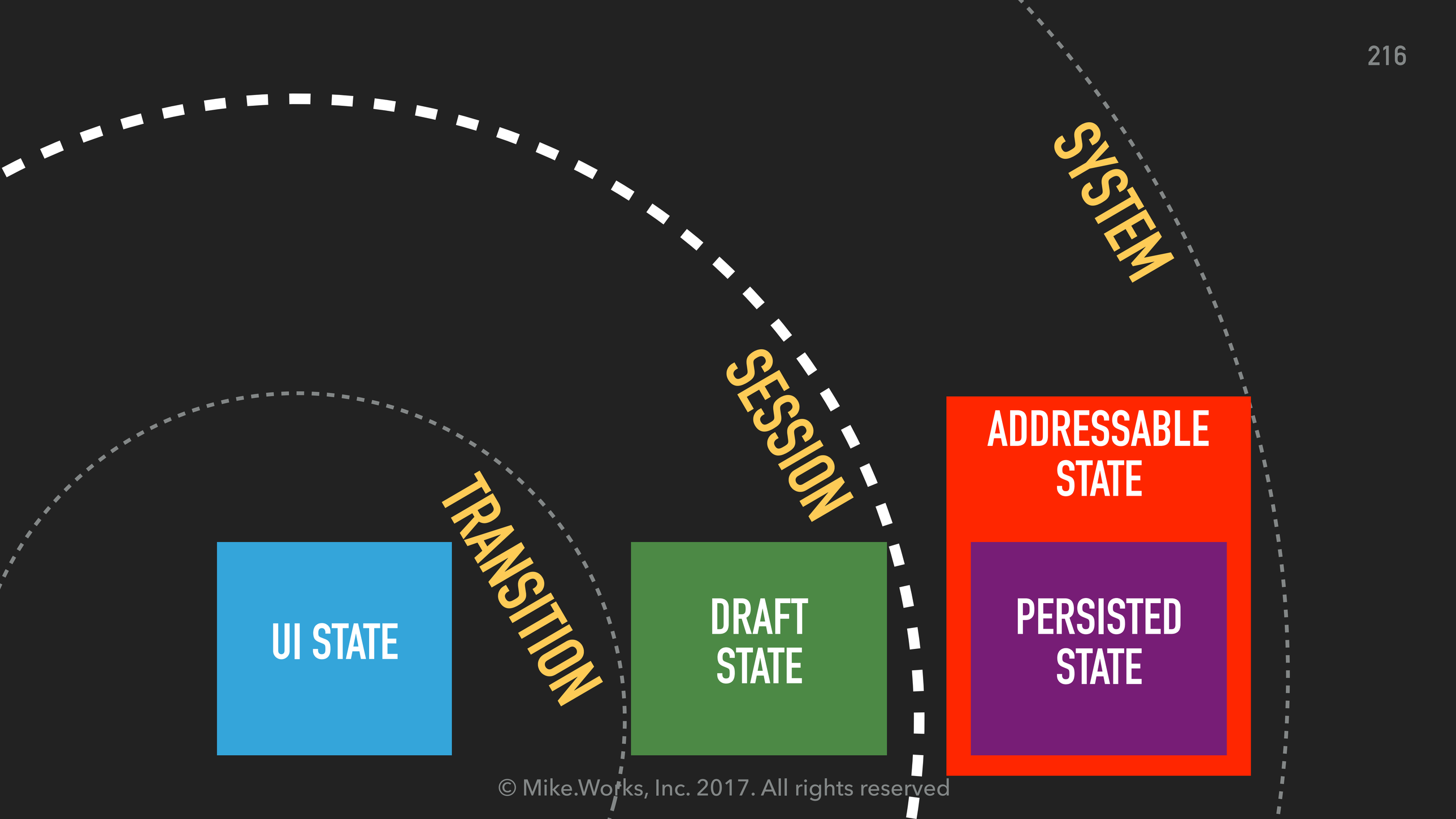
UI STATE

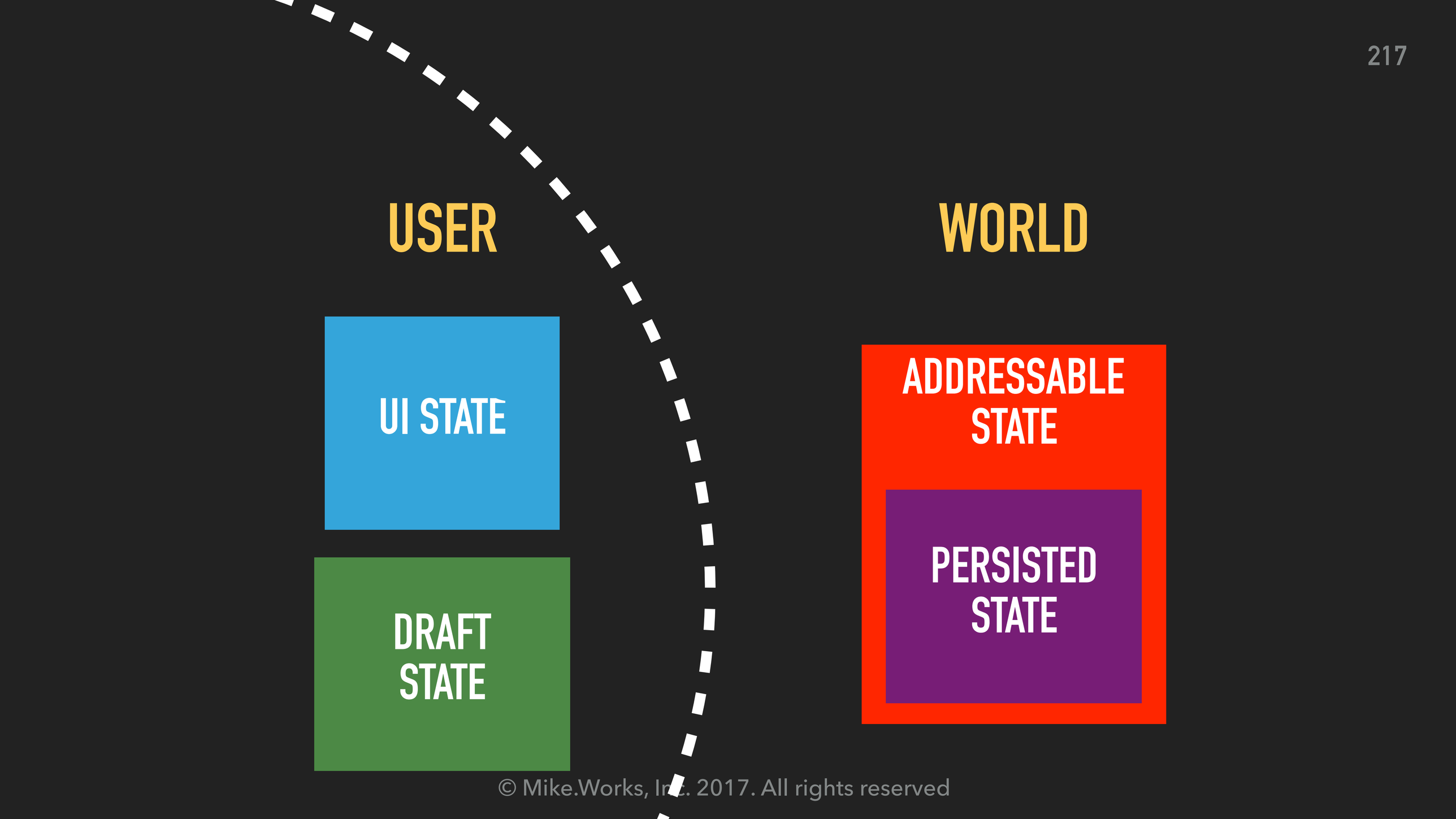
PERSISTED STATE

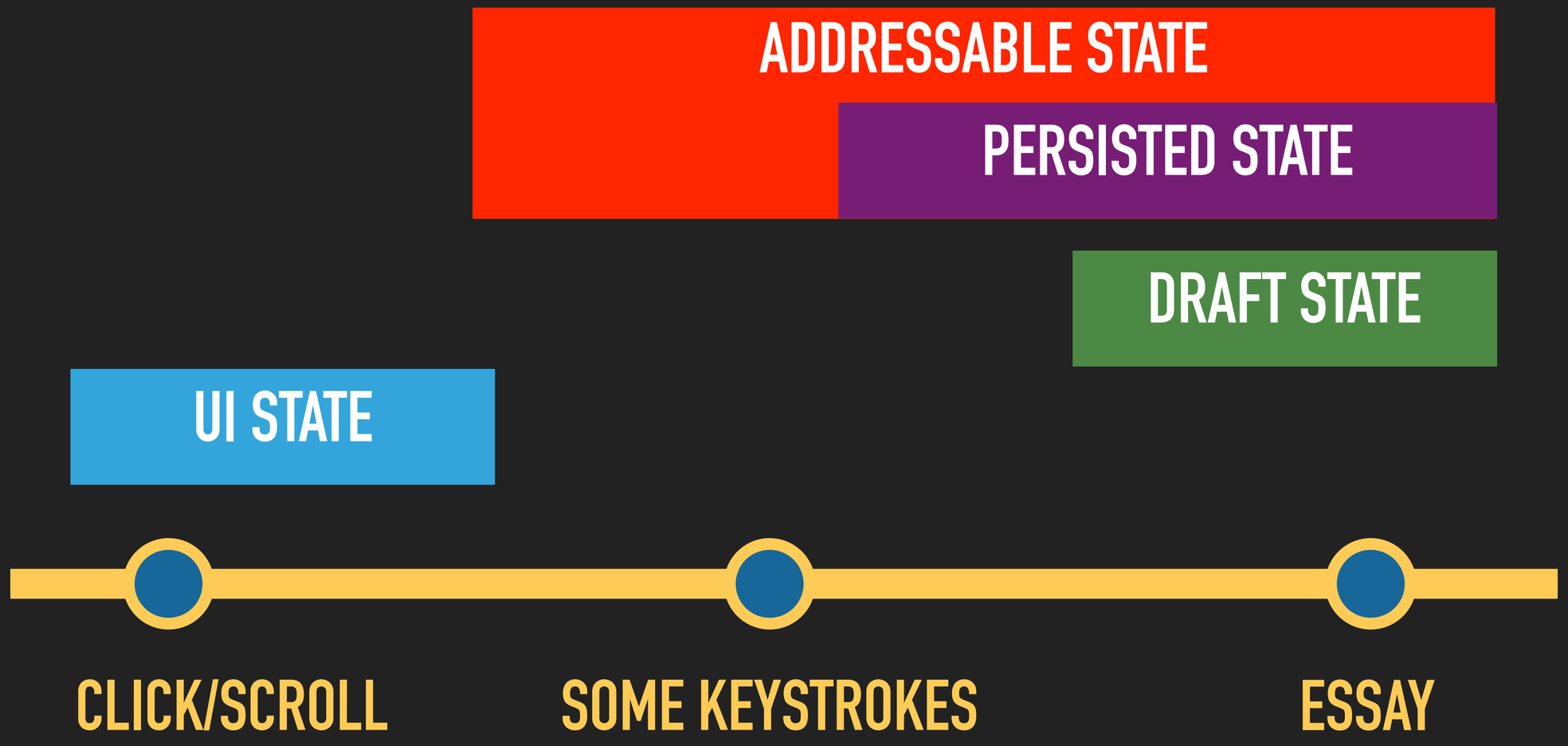
ADDRESSABLE STATE

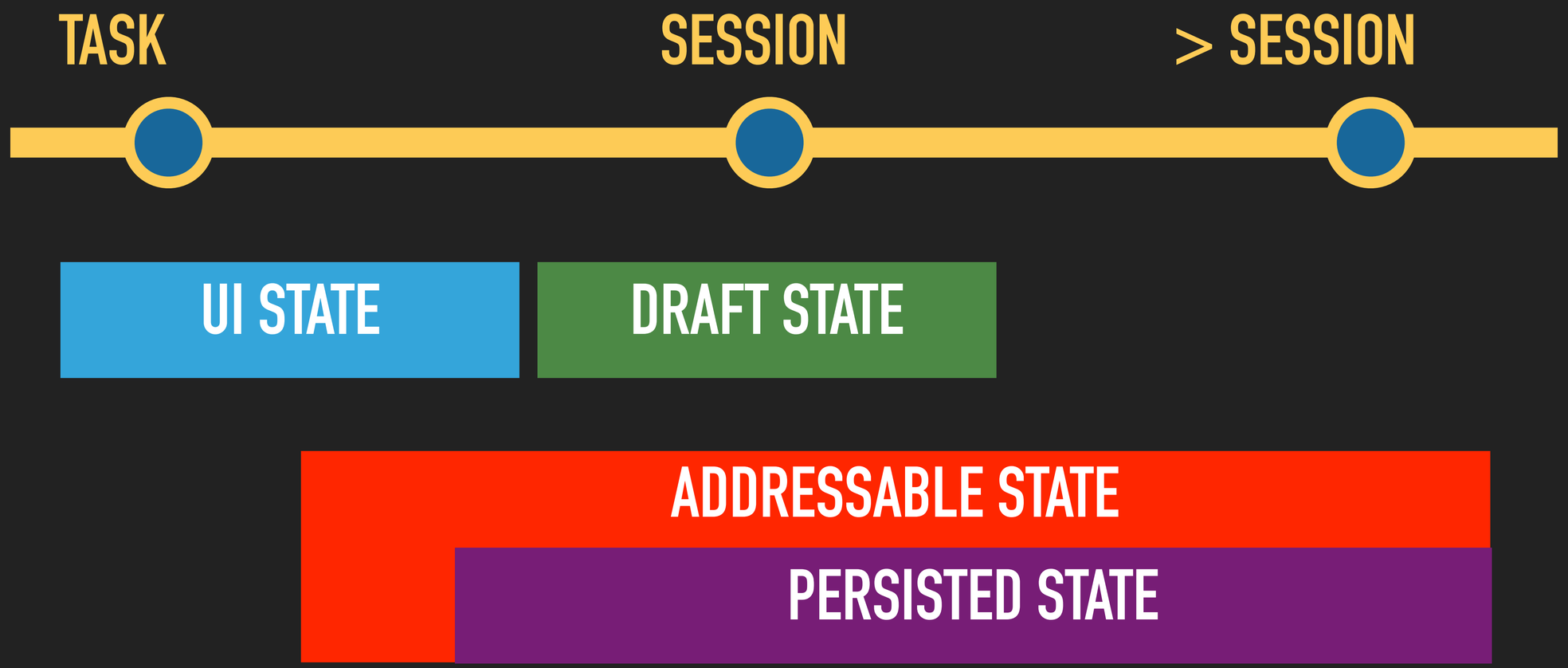
DRAFT STATE











Concurrency

Parallelism vs Concurrency

- **Parallelism** - A lot of things happening at the same time
- **Concurrency** - Keeping organized about many long-running tasks
- Lately, a lot of excitement around Observables & Rx.js

```
Promise.then(() ⇒ {

});
```

Parallelism vs Concurrency

- Cancellation
- Restart
- Limiting concurrency
- Black-holing async stuff

```
Uncaught                                    internal-model.js:502
EmberError {description: undefined, fileName:
undefined, lineNumber: undefined, message: "Attempted
to handle event `deleteRecord` on <state…ber689:35>
while in state root.deleted.inFlight. ", name:
"Error"…}
```

Ember-Concurrency Task

asynchronous, cancelable operations, bound to the lifetime of the object they live on

Defining a Task

```
import Ember from 'ember';
import { task } from 'ember-concurrency';

export default Ember.Component.extend({
  myTask: task(function * () {
    alert("hello!");
  })
});
```

Defining a Task

```
import Ember from 'ember';
import { task } from 'ember-concurrency';

export default Ember.Component.extend({
  myTask: task(function * () {
    alert("hello!");
  })
});
```

ES2015

Iterators & Generators

Iterators

- Iterators allow access one item from a collection at a time, keeping track of current position
- the `next()` method is what's used to get the next item in the sequence.

```
function makeIterator(arr) {
  var i = 0;
  return {
    next: () ⇒ i < arr.length ?
          { value: arr[i++], done: false } :
          { done: true }
  };
}
```

```
let l = ['ready', 'set', 'go'];
let it = makeIterator(l)
for (let p = it.next();
      !p.done;
      p = it.next()) {
  conso
}
```

output

```
ready
set
go
```

Generators

- Define their own iterative algorithm, `yield`ing each item in the sequence
- Use the `function*()` syntax
- Return an iterator
- State of the closure is preserved between `.next()` calls.

```
function* makeIterator(arr) {
  var i = 0;
  while(i < arr.length) {
    yield arr[i++];
  }
}

let l = ['ready', 'set', 'go'];
let it = makeIterator(l)
for (let p = it.next();
      !p.done;
      p = it.next()) {
  console.log(p.value);
}
```

output

```
ready
set
go
```

Iterables

- Support iteration within a `for .. of` loop
- Requires implementation of the `Symbol.iterator` method
- `Array` and `Map` already support this!

```
let arr = ['a', 'b', 'c'];
let it = arr[Symbol.iterator]();
console.log(it.next());
console.log(it.next());
console.log(it.next());
console.log(it.next());
```

output

```
{ value: 'a', done: false }
{ value: 'b', done: false }
{ value: 'c', done: false }
{ value: undefined, done: true }
```

Iterables - Defining our own iterable

- Generator function makes this very simple

```
let mike = {
  [Symbol.iterator]: function*() {
    yield 'M';
    yield 'i';
    yield 'k';
    yield 'e';
  }
}
for (let m of mike) {
  console.log(m);
}
```

Using Iterables - yield*

- In generator functions, the `yield*` keyword will yield each value of an iterable, one by one.

```
let mike = {
  [Symbol.iterator]: function*() {
    yield* 'Mike';
  }
}
for (let m of mike) {
  console.log(m);
}
```

Using Iterables - Destructuring Assignment

- Restructuring assignment works with any iterable, not just arrays!

```
let nums = {
  [Symbol.iterator]: function*() {
    yield* [1, 2, 3];
    yield 4;
    yield* [5, 6];
  }
}

console.log([ ... nums]);
```

output

```
[ 1, 2, 3, 4, 5, 6 ]
```

Defining a Task

- Yielding a promise will pause task execution until it's resolved

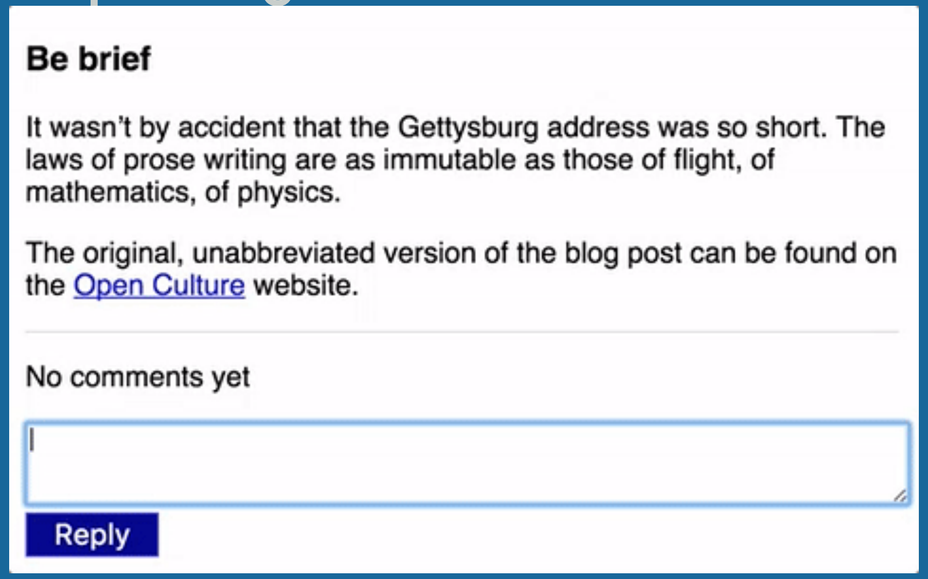
```
waitAFewSeconds: task(function * () {
  this.set('status', "Gimme one second...");
  yield timeout(1000);
  this.set('status', "Gimme one more second...");
  yield timeout(1000);
  this.set('status', "OK, I'm done.");
})
```

Editing Comments

- The “update” API call for comments takes a long time

```
PUT /api/posts/:post_id/comments/:id
```

- We want to avoid double-posting



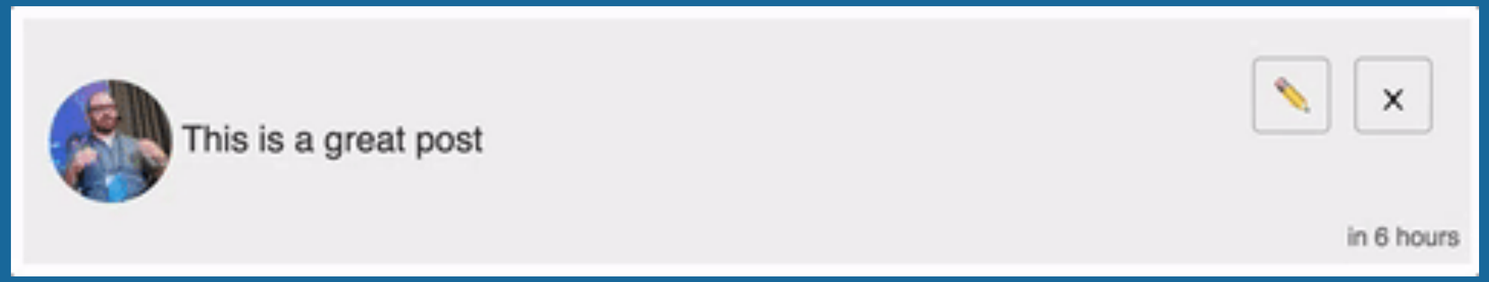
Editing Comments

14

- The “update” API call for comments takes a long time

```
PUT /api/posts/:post_id/comments/:id
```

- We want to avoid double-posting
- Here’s the desired UX. Extra credit for “cancel” edit button



- Update app.css from GitHub latest master commit

Animation

Liquid-Fire

- Ember's official animation library
- Modifies transition logic between views
- Built on velocity.js
- Extensible - you can write your own animations

Liquid-Fire: route transitions

app/transitions.js

```
export default function() {
  this.transition(
    this.fromRoute('index'),
    this.toRoute('posts'),
    this.use('toLeft'),
    this.reverse('toRight')
  );
};
```

app/templates/*

```
{{liquid-outlet}}
```

Liquid-Fire: Bind Transitions

app/transitions.js

```
export default function() {
  this.transition(
    this.childOf('.my-thing'),
    this.use('toUp')
  );
};
```

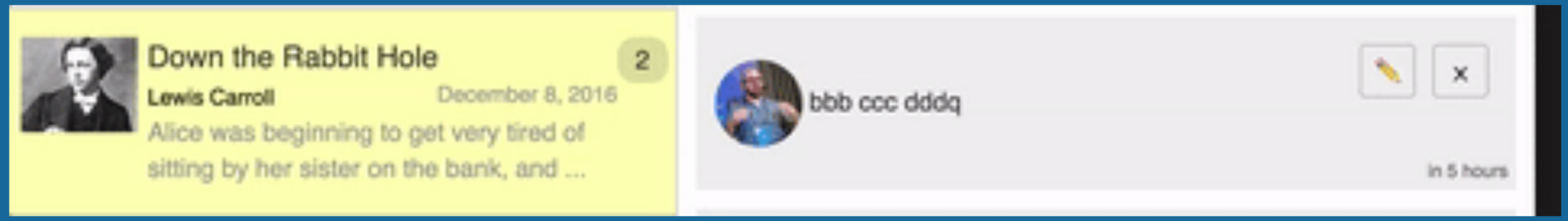
app/templates/*

```
{{liquid-bind x}}
```

Animated Comment Count

15

- in the {{post-tile}} component, animate comment count changes with {{liquid-bind}}



Blueprints

Code Generation in Ember-cli

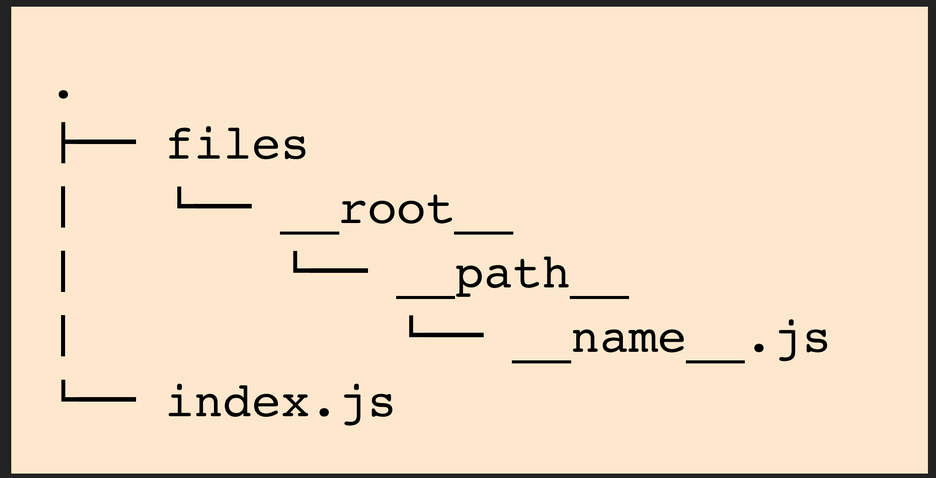
- Generating code with ember-cli leaves you with stubs for test coverage
- Generators are generally reversible
- They can also touch other files and do other things
- Arguments!!!

```
ember g <thing> <name>
```

```
ember g route home --path="/homepage"
```

Code Generation in Ember-CLI

- A blueprint is a folder
- Inside is `index.js` — this is the entry point of the blueprint
- You can customize lots of things here



```
module.exports = {
  locals: function (options) {
    // Return custom template variables here.
    return {};
  },

  normalizeEntityName: function (entityName) {
    // Normalize and validate entity name here.
    return entityName;
  },

  fileMapTokens: function (options) {
    // Return custom tokens to be replaced in your files
    return {
      __token__: function (options) {
        // logic to determine value goes here
        return 'value';
      }
    }
  },
  beforeInstall: function (options) { },
  afterInstall: function (options) { },
  beforeUninstall: function (options) { },
  afterUninstall: function (options) { }
};
```

```
locals: function(options) {
  // Return custom template variables here.
  return {};
},

normalizeEntityName: function(entityName) {
  // Normalize and validate entity name here.
  return entityName;
},

fileMapTokens: function(options) {
  // Return custom tokens to be replaced in your files
  return {
    __token__: function(options) {
      // logic to determine value goes here
      return 'value';
    }
  }
}
```

--path

Posts/Index —› posts/index

__root__ = “app”

__path__ = “posts”

__name__ = “index”

Placeholder-img blueprint

16

- Build a blueprint that makes a helper, for a placeholder image url
- Many of these services accept urls like `<base>/w/h`

http://placekitten.com/400/420
http://placehold.it/400/400

```
<img src={{cat-url w="400" h="400"}} />
```

Modular Architecture

Ember addons

- A uniquely awesome benefit of using ember
- NPM modules with extras
- Can either extend ember, extend ember-cli or both!
- Similar conventions to ember apps
- emberaddons.com
- in-repo addons, in your lib folder!

```
ember g in-repo-addon <addon name>
```

1

A bundle of client-side modules

Two Module Namespaces: App & Addon

- JS Modules in the addon folder will be in the addon's namespace

```
ember-dropdown/utils/calculate-width.js
```

- Certain types of modules need to be in the app namespace
  - Components
  - Helpers
  - Models
  - Adapters
  - Serializers
  - Templates

Components via addon

- create a new in-repo addon called core-ui
- move the x-input and x-textarea components and templates into the appropriate folders of the addon
- keep the interesting stuff in the addon folder, and re-export in to the app tree.

```
import Ember from 'ember';
import layout from 'my-addon/templates/components/x-foo';

export default Ember.Component.extend({
  layout
});
```

2

Modify asset pipeline

Broccoli Plugins for ember apps

- Most of the work is done in the addon's index.js file
- included hook lets us register broccoli plugins for filetypes

```
module.exports = {
  name: 'addon-name',
  included: function(app, parentAddon) {
    var app = (parentAddon || app);
    app.registry.add('hbs', new MyPlugin());
  }
};
```

Pipeline modification

- Move your code commenting broccoli plugin into an in-repo addon
- Host app should still end up with code comments at the top of JS files

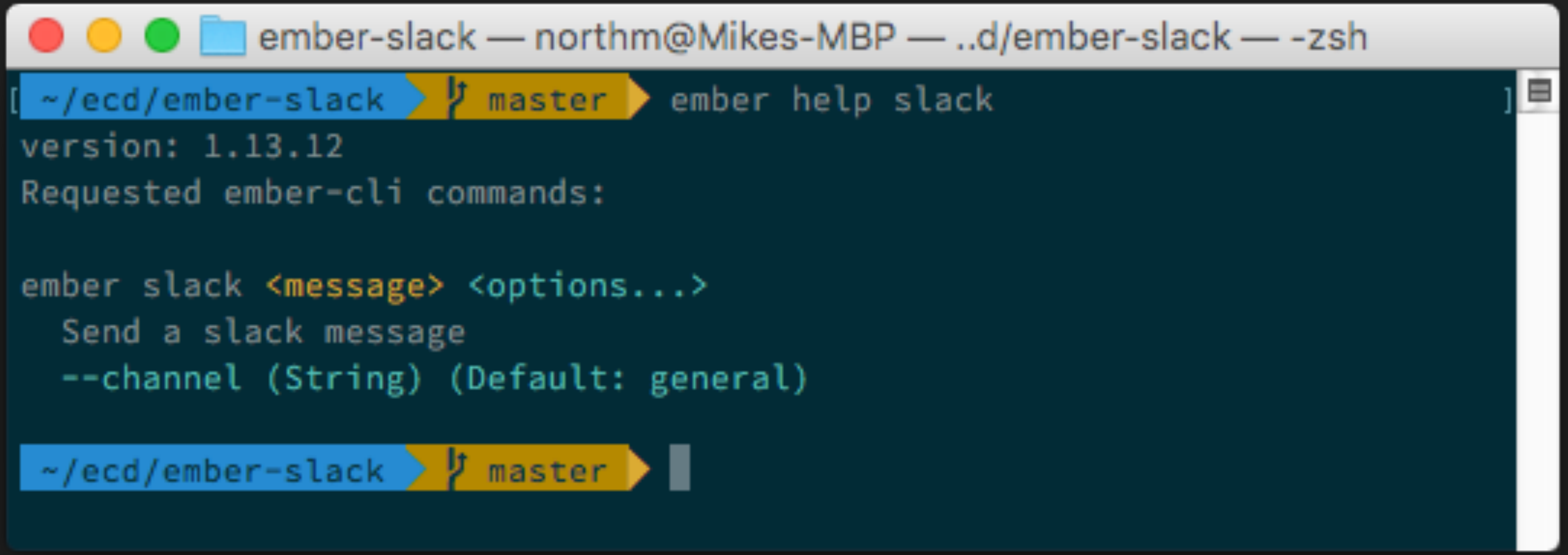
Add a command to ember-cli

```
module.exports = {
  name: 'my-addon',
  includedCommands() {
    return {
      fem: {
        name: 'fem',
        description: 'Prints something to the screen',
        works: 'insideProject',
        anonymousOptions: [],
        availableOptions: [],
        run(commandOptions, rawArgs) {
          console.log('Welcome to Front End Masters')
        }
      }
    }
  }
};
```

named and anonymous options

```
// Un-named options
anonymousOptions: [
  '<message>'
],
// Named options
availableOptions: [{
    name: 'channel',
    type: String,
    default:
'general'
}],
```

Named and Anonymous Options



Slack Notifications

- Let's build the first part of a slack notification ember-cli-command. I want to be able to type

ember slack "🚀 Deployed" --channel announcements

- and we should see the following printed to the console

```
---------------------------
#Announcements: 🚀 Deployed
```