Server Side Rendering from the trenches

William Durand, jsDay 2018

I am going to talk about server side rendering applied to JavaScript applications and that are usually rendered in web browsers, a.k.a. **universal/isomorphic** apps.

A bit of history

Before 2010

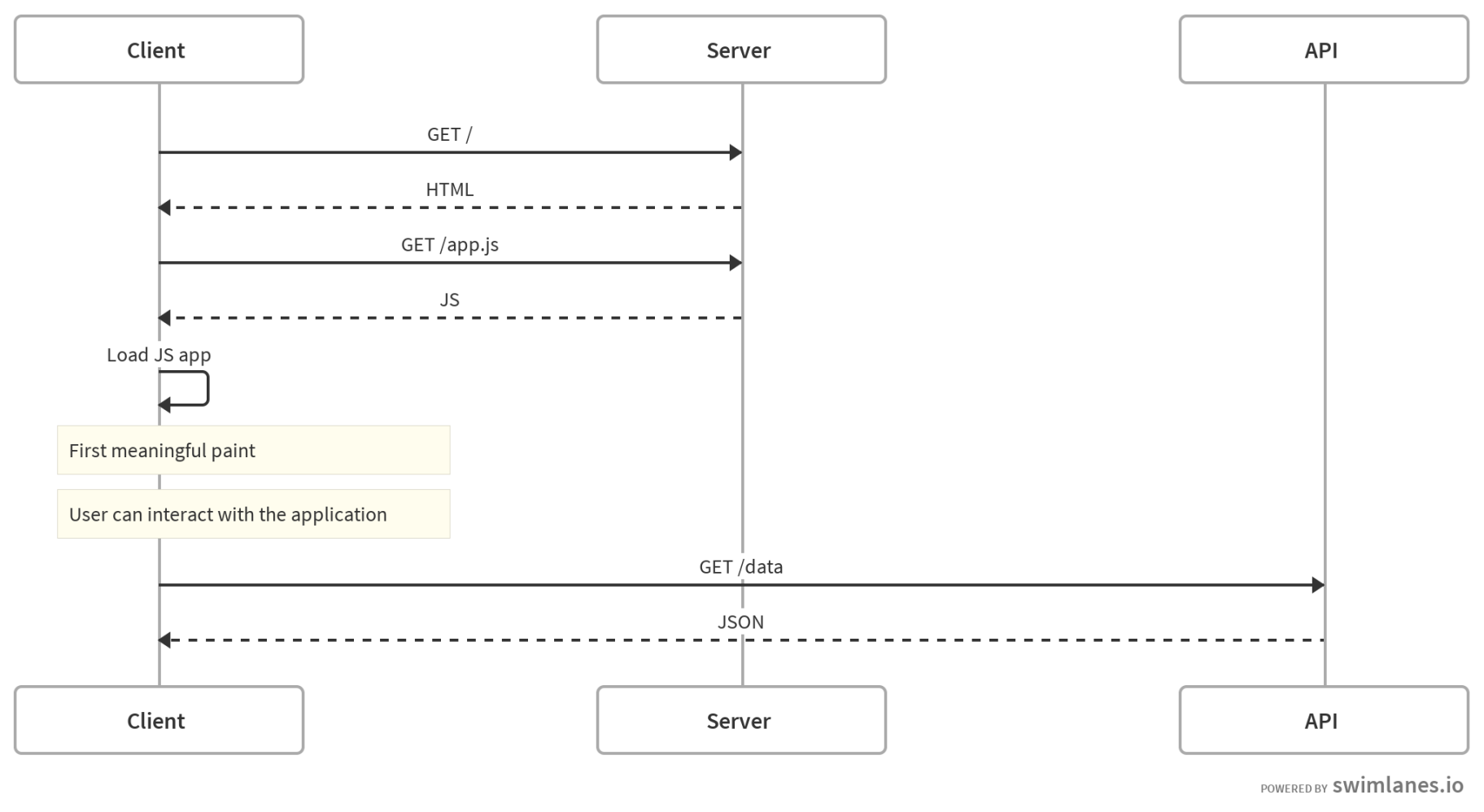
- Server side frameworks
- Template engines
- jQuery, Yahoo UI
- `script.aculo.us` ❤️
- `XMLHttpRequest`

2011

- Backbone.js is 3 months old
- Node.js is 2 years old
- People read Fielding's dissertation (REST)

Let's write client side applications in JavaScript! 😎

Client Side Rendering



2013

- React initial release

Interesting, but what is this Flux architecture again? 🤔

2015

- Vue.js is 1 year old
- Redux initial release

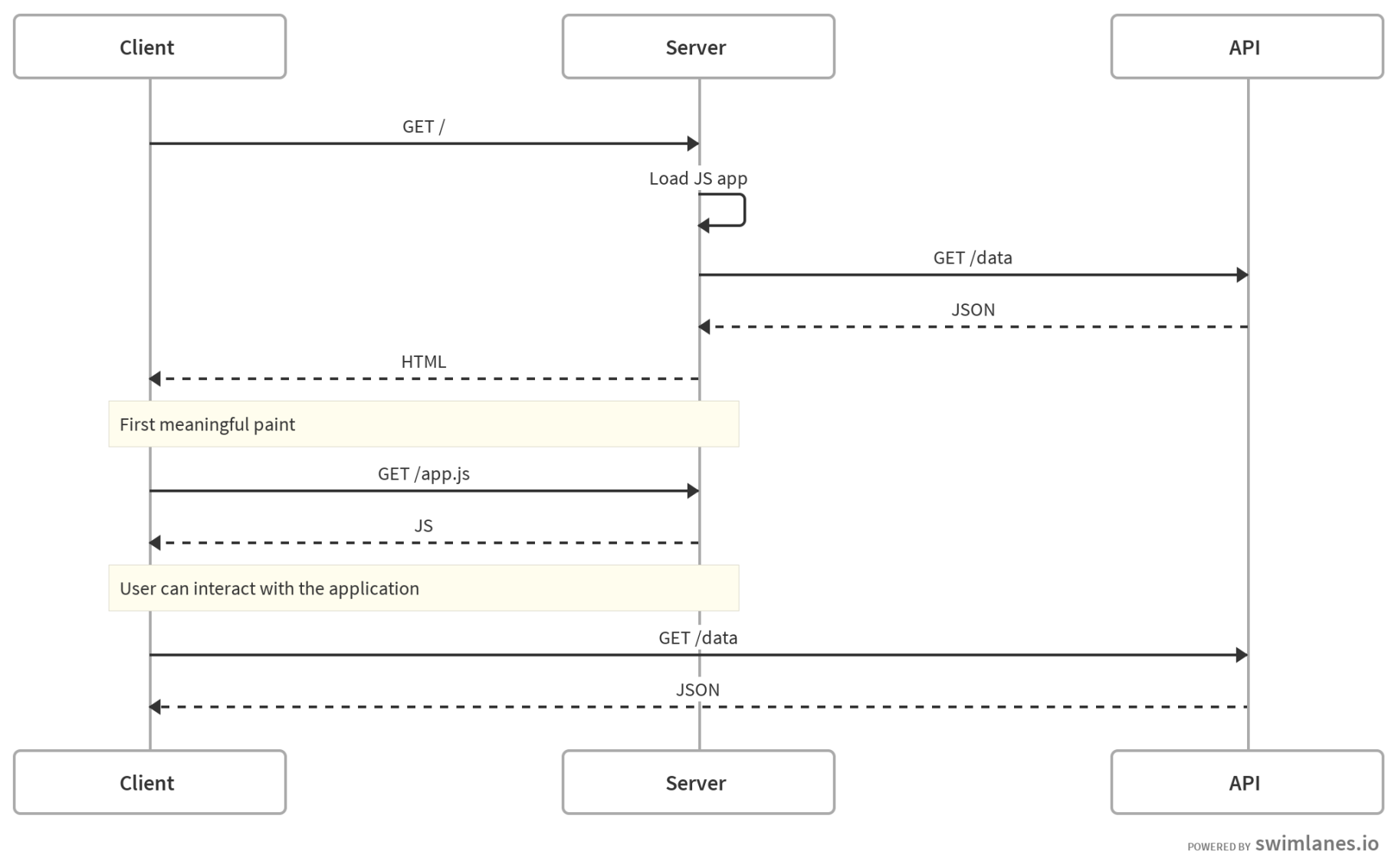
Problem(s) solved 🎉

Since then...

“Can we render this JavaScript app on the server?”

Server Side Rendering

The big picture



How it works

On every incoming request, the server:

1. creates the store/initial app state
2. matches the URL to find the right component
3. loads the component and gets the HTML
4. sends the HTML back to the client

Then, the client loads the JavaScript app.

Benefits?

- Accessibility (limited)
- Better performances (first meaningful paint)
- Better user experience (JS disabled)
- Search Engine Optimization/Social sharing

Googlebot

- Quite good at browsing JS apps
- Give up after ~10 seconds
- Some issues with `react-router`

Source: Testing a React-driven website's SEO using "Fetch as Google", Nov. 2016.

Drawbacks?

- **Makes everything very complicated**
- Time To First Byte (TTFB) usually slower (but Cloudfare says it's fine in [1])
- React `renderToString()` holds the event loop [2], probably also the case for other frameworks

---

[1]: Stop worrying about Time To First Byte (TTFB)
[2]: The Benefits of Server Side Rendering Over Client Side Rendering

Why is it so complicated?

1. Two different environments, one codebase
2. Cookies, redirects/errors, HTTP statuses
3. Data fetching **before** rendering

1. Two different environments, one codebase

+ **isomorphic libraries** and polyfills

2. Cookies, redirects/errors, HTTP statuses

You have to find ~~hacks~~ nice tricks 👻

3. Data fetching **before** rendering

There used to be two approaches...

- static `async` method to fetch data and `Promise.all()` on the server
- double render 💔

Update on Async Rendering - R

https://reactjs.org/blog/2018/03/27/update-on-async-rendering.html

React Docs Tutorial Community Blog Search docs v16.3.2 GitHub

Update on Async Rendering

March 27, 2018 by Brian Vaughn

For over a year, the React team has been working to implement asynchronous rendering. Last month during his talk at JSConf Iceland, Dan unveiled some of the exciting new possibilities async rendering unlocks. Now we'd like to share with you some of the lessons we've learned while working on these features, and some recipes to help prepare your components for async rendering when it launches.

One of the biggest lessons we've learned is that some of our legacy component lifecycles tend to encourage unsafe coding practices. They are:

RECENT POSTS

- React v16.3.0: New lifecycles and context API
- **Update on Async Rendering**
- Sneak Peek: Beyond React 16
- Behind the Scenes: Improving the Repository Infrastructure
- Introducing the React RFC Process
- React v16.2.0: Improved Support for Fragments
- React v16.0
- React v15.6.2
- DOM Attributes in React 16
- Error Handling in React 16
- All posts ...

Some examples

React

- `ReactDOMServer.renderToString()`
- That's all folks! 🤷‍♀️

Naive example (1/4)

```
// server.js (express app)

app.use((req, res) => {
  const context = {};
  // 1. Create the store.
  const store = configureStore();
  // 2. Render the application using a `StaticRouter`.
  const markup = renderToString(
    <Provider store={store}>
      <StaticRouter location={req.url} context={context}>
        <App />
      </StaticRouter>
    </Provider>
  );
  // see next slide...
```

Load the application

Naive example (2/4)

```
  if (context.url) {
    redirect(301, context.url); // A `<Redirect>` was rendered.
  } else {
    // 3. Replace placeholders by generated state and HTML.
    const preloadedState = store.getState();
    const html = INDEX_HTML
      .replace('__SSR__', markup)
      .replace('__PRELOADED_STATE__ = {}', [
        `__PRELOADED_STATE__ =`,
        JSON.stringify(preloadedState).replace(/</g, '\\u003c'),
      ].join(' '));
    }
    res.send(html); // 4. Send the HTML to the client.
  });
});
```

Send the full HTML to the client

Naive example (3/4)

```
     <!-- index.html [...] -->

     <noscript>
       You need to enable JavaScript to run this app.
     </noscript>
-    <div id="root">></div>
+    <div id="root">__SSR__</div>
+    <script>
+      window.__PRELOADED_STATE__ = {};
+    </script>
   </body>
```

Add placeholders in the `index.html`

Naive example (4/4)

```
// src/index.js

-const store = configureStore();
+const preloadedState = window.__PRELOADED_STATE__ || {};
+// Allow the passed state to be garbage-collected
+delete window.__PRELOADED_STATE__;
+
+const store = configureStore(preloadedState);
```

Use the state generated on the server, if any

But...

- No data fetching
- No error handling

Next.js

Powerful React-based framework, SSR-ready.

```
class UsersList extends React.Component {

  static async getInitialProps({ store, isServer, ...props }) {
    const users = await getUsers();

    return { users };
  }

  render() {
    const { users } = this.props;
    // ...
  }
}
```

Vue.js

- SSR is officially supported
- ssr.vuejs.org is pure gold

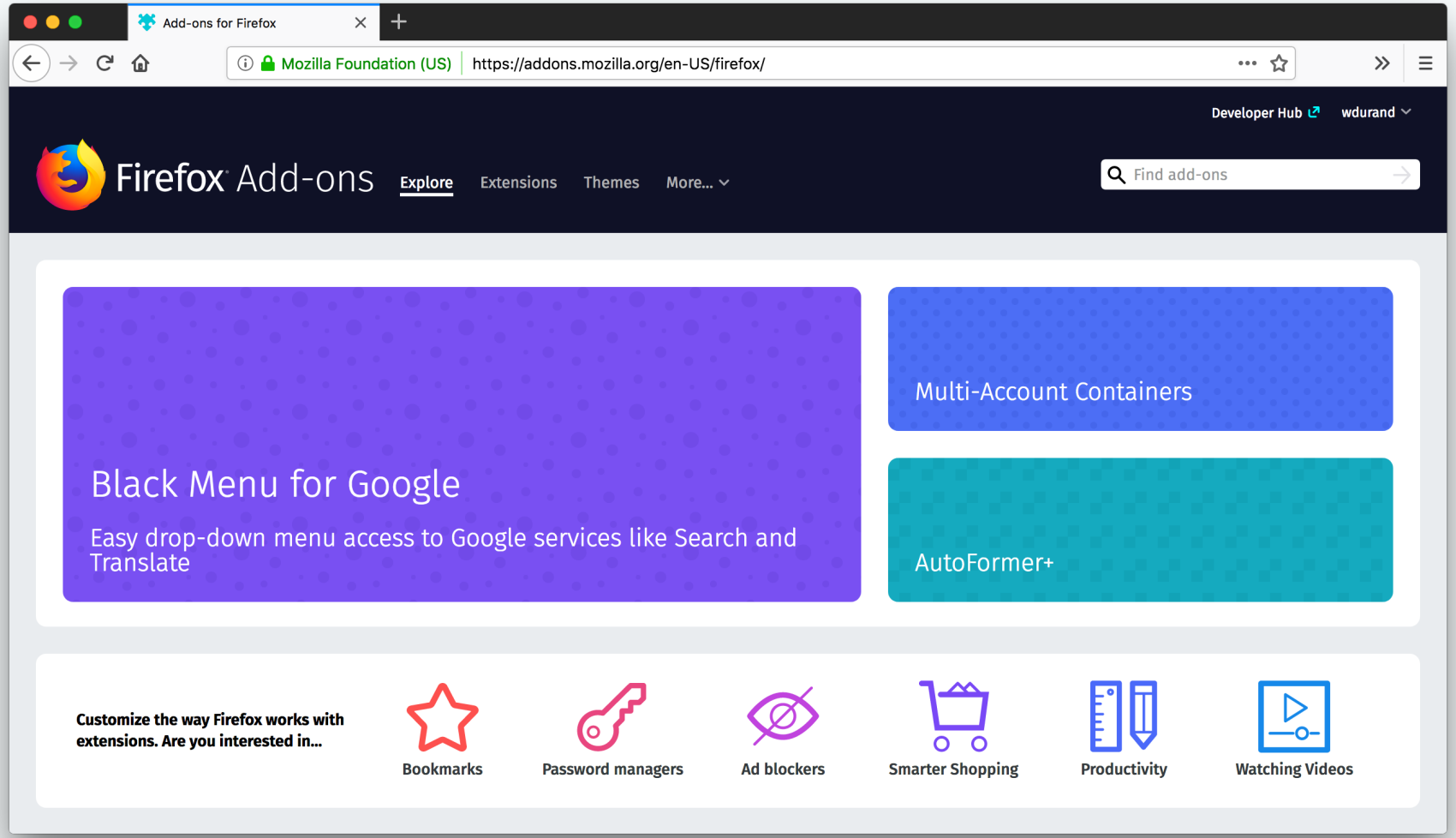
```
<!-- UsersList.vue -->
<template></template>

<script>
export default {
  asyncData ({ store }) {
    return store.dispatch('getUsers');
  }
}
</script>
```

Nuxt.js

- Vue-based framework, inspired by Next.js
- Implement what is described in ssr.vuejs.org

Some lessons learnt


Add-ons for Firefox
Mozilla Foundation (US)
https://addons.mozilla.org/en-US/firefox/
Developer Hub
wdurand
Firefox Add-ons
Explore
Extensions
Themes
More...
Find add-ons
Black Menu for Google
Easy drop-down menu access to Google services like Search and Translate
Multi-Account Containers
AutoFormer+
Customize the way Firefox works with extensions. Are you interested in...
Bookmarks
Password managers
Ad blockers
Smarter Shopping
Productivity
Watching Videos

addons.mozilla.org

- Universal React/Redux app
- i18n/l10n, CSP 👌
- Open Source: mozilla/addons-frontend

Double render is a fragile hack, do not use it 😅

Always be careful

Undefined reference on the server == Error 500.
#GameOver

Error handling is tough

- Accurate HTTP status codes
- Correct error pages
- On both server and client

Debugging made ~~easy~~ complex

- Isomorphic logging layer but no dev tools
- `disableSSR` config option to the rescue!
  - but some issues are hidden

Server logs 😱

```
[...]
INFO: proxy: 302 ~> http://127.0.0.1:3333/service-worker.js (app=
INFO: proxy: 200 ~> https://addons-dev.allizom.org/api/v3/account
INFO: proxy: 200 ~> https://addons-dev.allizom.org/api/v3/account
INFO: proxy: 200 ~> https://addons-dev.allizom.org/api/v3/account
WARN: server: restrictSearchResultsToAppVersion config set; not s
WARN: server: restrictSearchResultsToAppVersion config set; not s
WARN: server: restrictSearchResultsToAppVersion config set; not s
INFO: proxy: 200 ~> https://addons-dev.allizom.org/api/v3/addons/
INFO: proxy: 200 ~> https://addons-dev.allizom.org/api/v3/addons/
INFO: proxy: 200 ~> https://addons-dev.allizom.org/api/v3/addons/
INFO: server: Second component render after sagas have finished (
INFO: proxy: 200 ~> http://127.0.0.1:3333/en-US/firefox/ (app=amo
WARN: server: CSP has been disabled from the config (app=amo)
INFO: server: Prepending lang to URL: en-US (app=amo)
INFO: server: Prepending application to URL: firefox (app=amo)
```

New fun bugs 🤷‍♂️

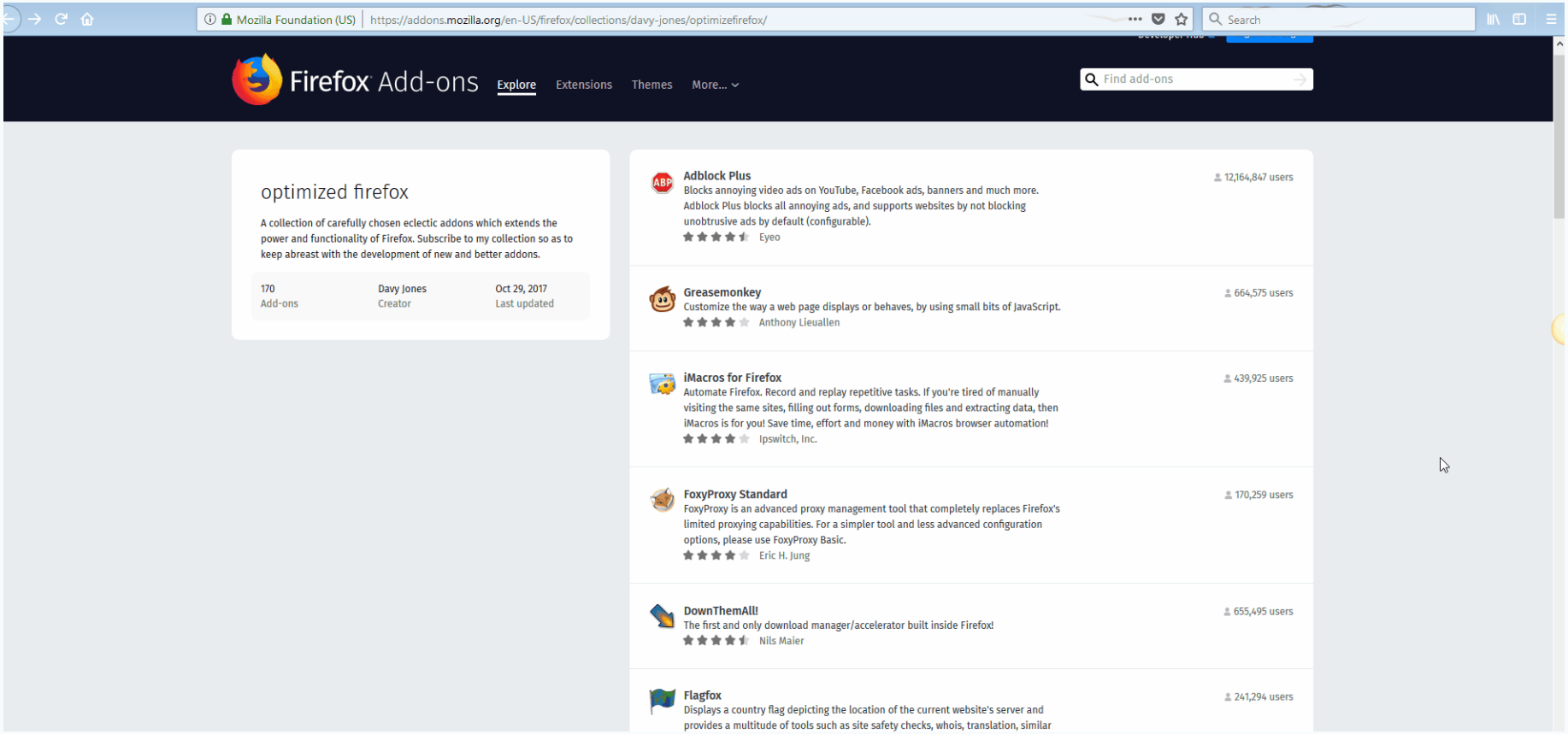
Cannot load an add-on on the client side but it works on the server #3138

Closed **willdurand** opened this issue on Sep 14, 2017 · 3 comments

Server error received when navigating with browser 'back' button from page 2 to page 1 in collection details #4933

Open **AlexandraMoga** opened this issue a day ago · 0 comments

Example


Mozilla Foundation (US)
https://addons.mozilla.org/en-US/firefox/collections/davy-jones/optimizefirefox/
Search
Firefox Add-ons
Explore
Extensions
Themes
More...
Find add-ons
optimized firefox
A collection of carefully chosen eclectic addons which extends the power and functionality of Firefox. Subscribe to my collection so as to keep abreast with the development of new and better addons.
170
Add-ons
Davy Jones
Creator
Oct 29, 2017
Last updated
Adblock Plus
Blocks annoying video ads on YouTube, Facebook ads, banners and much more. Adblock Plus blocks all annoying ads, and supports websites by not blocking unobtrusive ads by default (configurable).
Eyeo
12,164,847 users
Greasemonkey
Customize the way a web page displays or behaves, by using small bits of JavaScript.
Anthony Lieuallen
664,575 users
iMacros for Firefox
Automate Firefox. Record and replay repetitive tasks. If you're tired of manually visiting the same sites, filling out forms, downloading files and extracting data, then iMacros is for you! Save time, effort and money with iMacros browser automation!
Ipswitch, Inc.
439,925 users
FoxyProxy Standard
FoxyProxy is an advanced proxy management tool that completely replaces Firefox's limited proxying capabilities. For a simpler tool and less advanced configuration options, please use FoxyProxy Basic.
Eric H. Jung
170,259 users
DownThemAll!
The first and only download manager/accelerator built inside Firefox!
Nils Maier
655,495 users
Flagfox
Displays a country flag depicting the location of the current website's server and provides a multitude of tools such as site safety checks, whois, translation, similar
241,294 users

No random allowed

Pages stuck in "loading" mode when an error is thrown #3313

Closed **willdurand** opened this issue on Oct 2, 2017 · 28 comments

---

There is a React RFC for introducing isomorphic IDs.

You must have a fresh, isolated server context

The active locale leaks between server responses #3538

Closed kumar303 opened this issue on Oct 17, 2017 · 1 comment

Example

```
$ repeat 10 curl https://addons.mozilla.org/en-US/firefox/addon/a
  | grep --color -oE 'updated</dt><dd data-reactid=\"\d+\">.+?</d

updated</dt><dd data-reactid="194">il y a 2 jours (6 nov. 2017)</
updated</dt><dd data-reactid="194">2 days ago (Nov 6, 2017)</dd>
updated</dt><dd data-reactid="194">2 days ago (Nov 6, 2017)</dd>
updated</dt><dd data-reactid="194">2 hari yang lalu (6 Nov 2017)<
updated</dt><dd data-reactid="194">2 days ago (Nov 6, 2017)</dd>
updated</dt><dd data-reactid="194">2 days ago (6 Nov 2017)</dd>
updated</dt><dd data-reactid="194">2 dias atrás (6 de nov de 2017
updated</dt><dd data-reactid="194">2 days ago (Nov 6, 2017)</dd>
updated</dt><dd data-reactid="194">2 天前 (2017年11月6日)</dd>
updated</dt><dd data-reactid="194">2 days ago (Nov 6, 2017)</dd>
```

You need more servers

Investigate "JavaScript heap out of memory" error #3916

Closed **bqbn** opened this issue on Nov 15, 2017 · 6 comments

In order to reduce the # of 500 caused by the issue, we had to bump up the # of instances in the cluster from 8 to 40. And that eventually stopped the "heap oom" from continuously happening.

Security considerations

- State serialization when transferring the Redux state from the server to the client [1]
- Sensitive data on the server, *e.g.*, env vars

---

[1]: Redux Server Rendering

React has useful dev warnings

```
▶ Uncaught Invariant Violation: You're trying to render a component to the document using server rendering but the checksum was invalid. This usually means you rendered a different      bundle.js:858
  component type or props on the client from the one on the server, or your render() methods are impure. React cannot handle this case due to cross-browser quirks by rendering at the document root. You
  should look for environment dependent code in your components and ensure the props are the same client and server side:
   (client) v data-reactid="7">hello</div></div><scr
   (server) v data-reactid="7">hi</div></div><script
>
```

TL;DR: React on the client does not generate the same HTML sent by the server: there is a bug. [1]

---

[1]: What's New With Server-Side Rendering in React 16

We can test (pretty much) everything 🚀

So what?

You may not need SSR.

If you need it, use a framework that is SSR-ready.

Other ideas

- Prerender.io
- Headless Chrome: an answer to server-side rendering JS sites
- Progressive Web Apps/Service workers?

Thank you to my awesome team!

Thank You.

Questions?

joind.in/talk/b263d

williamdurand.fr

github.com/willdurand

twitter.com/couac