Leveraging

Code Splitting in React Apps

@glnnrys

Witaj Warszawa!

Glenn Reyes

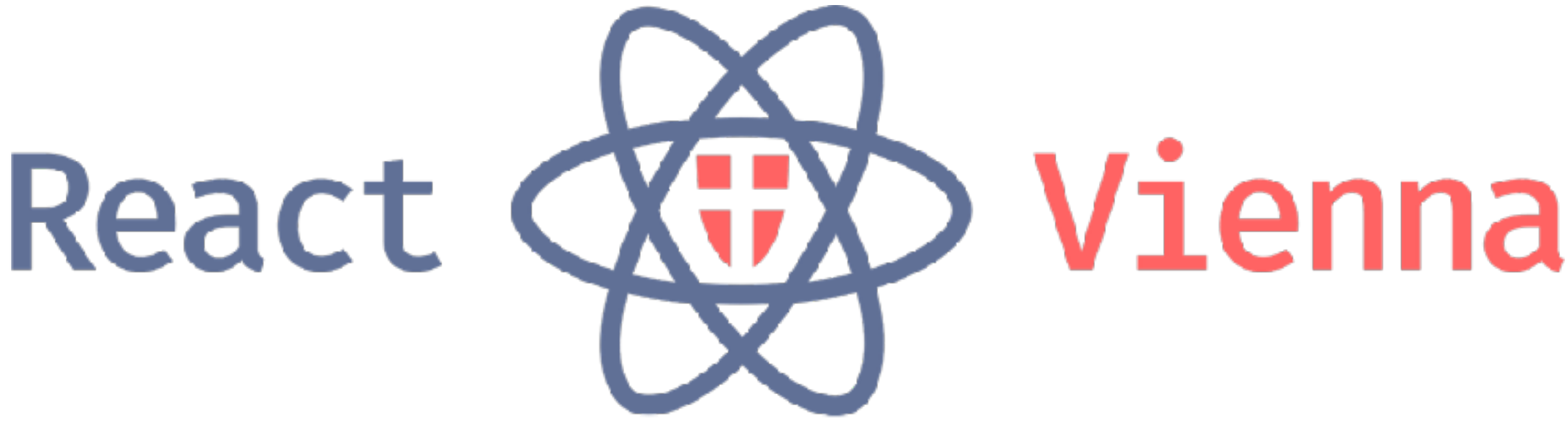
@glnnrys

Glenn Reyes

@glnnrys

👨‍💻 💍 🎸 🌎

Source: Stefan Steinbauer at unsplash.com

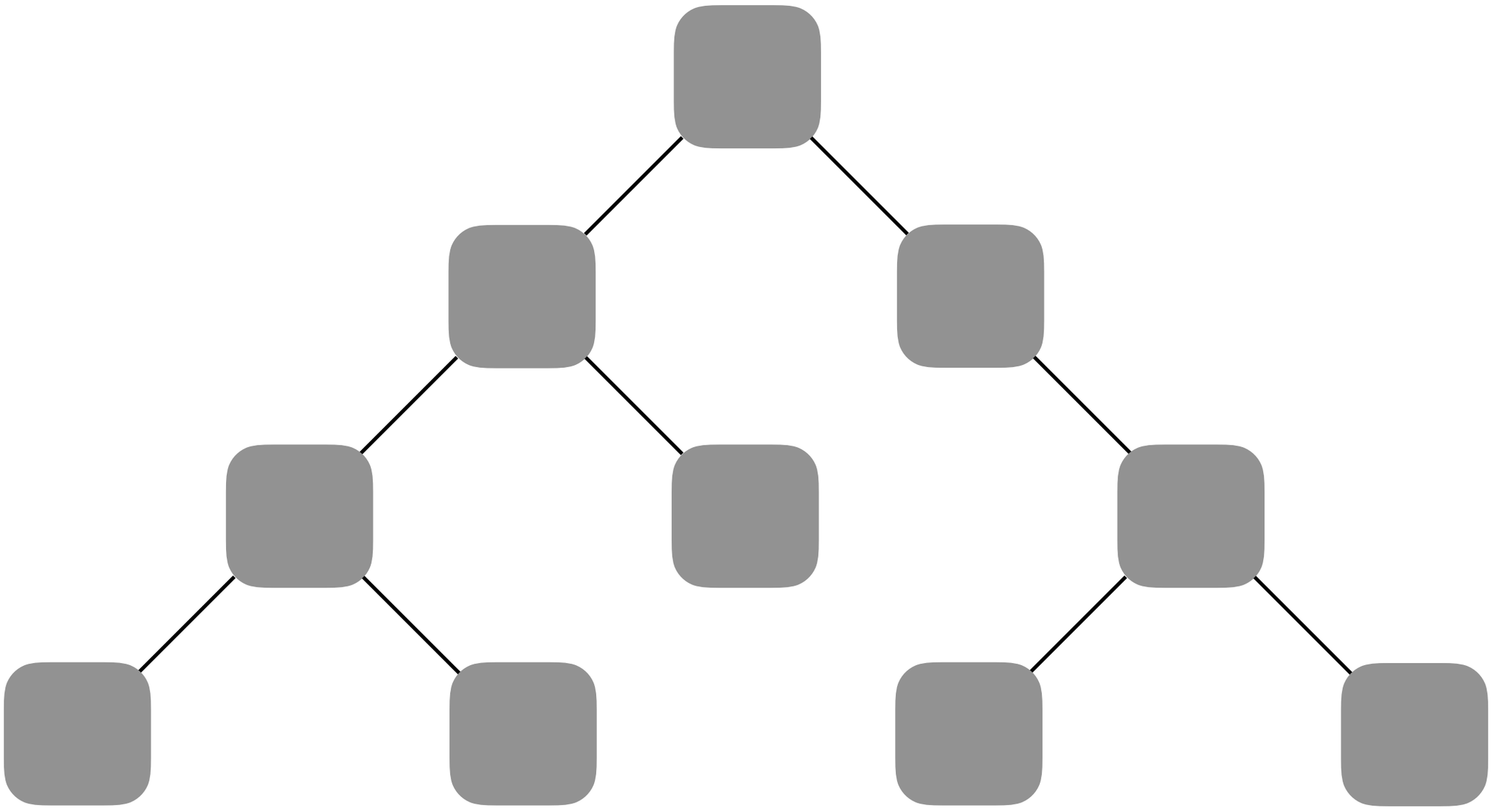

React
Vienna

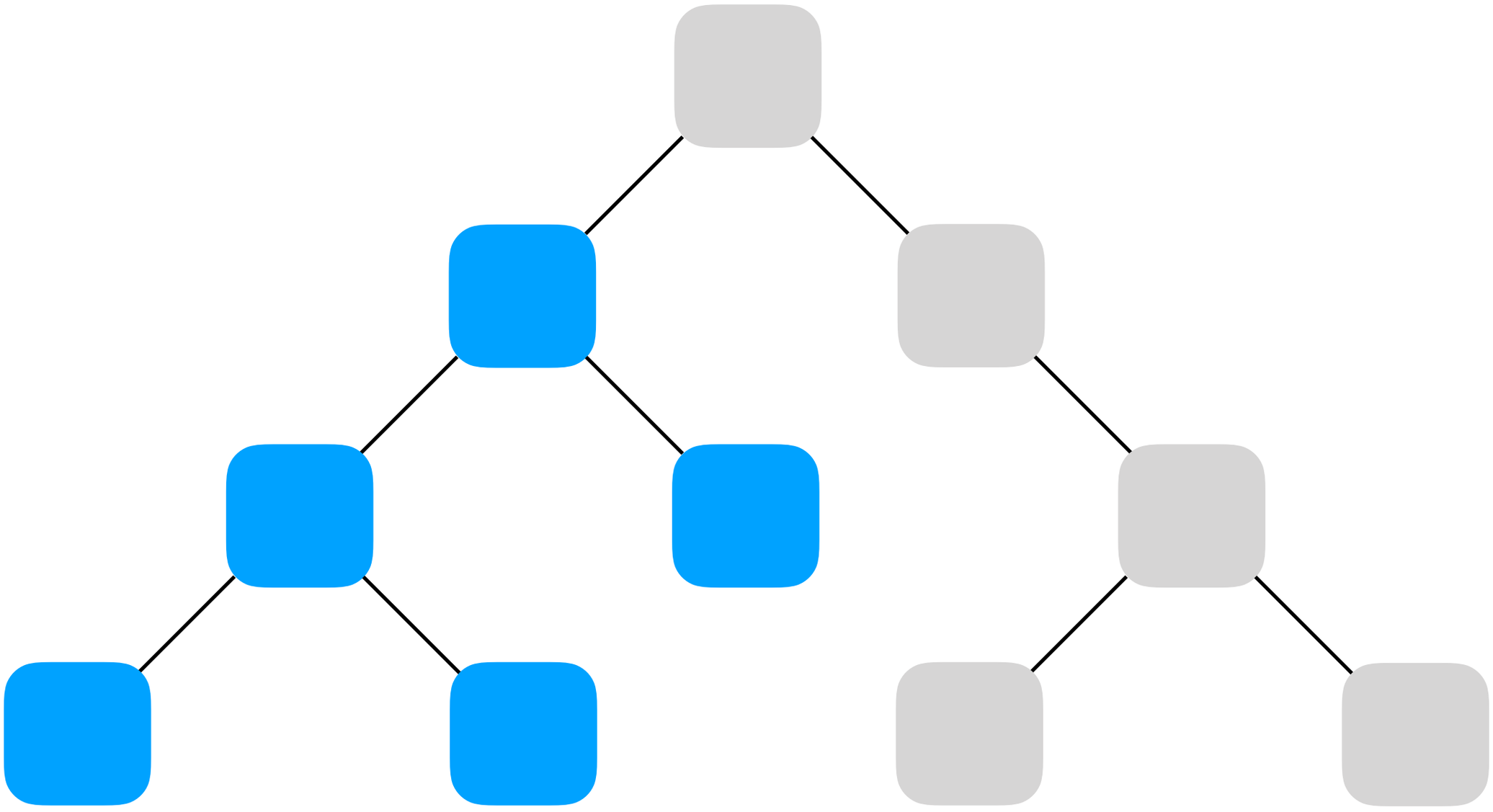
Code Splitting

What is Code Splitting?

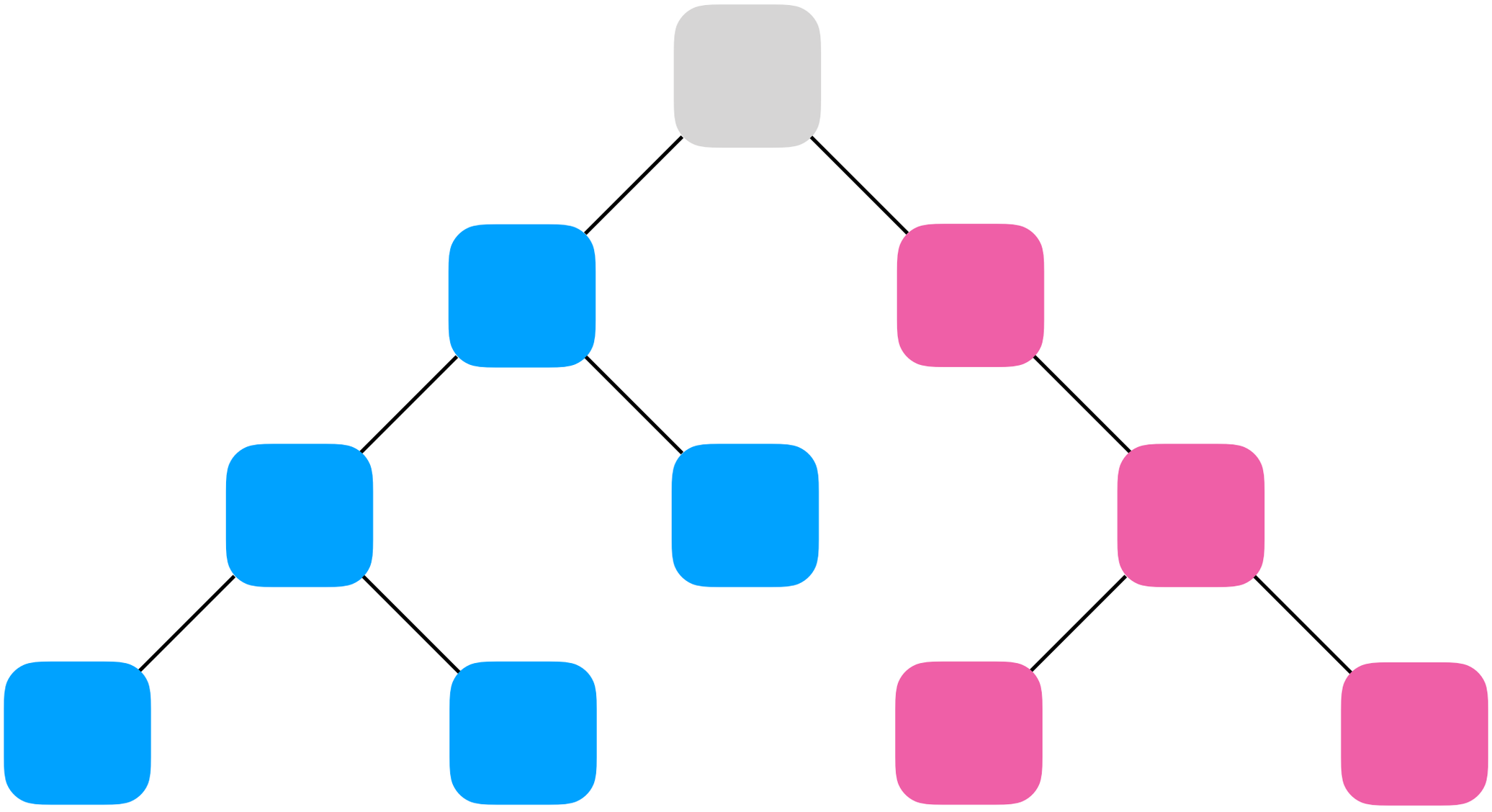
“Splits your code into various bundles which can then be loaded on demand.”

– webpack.js.org





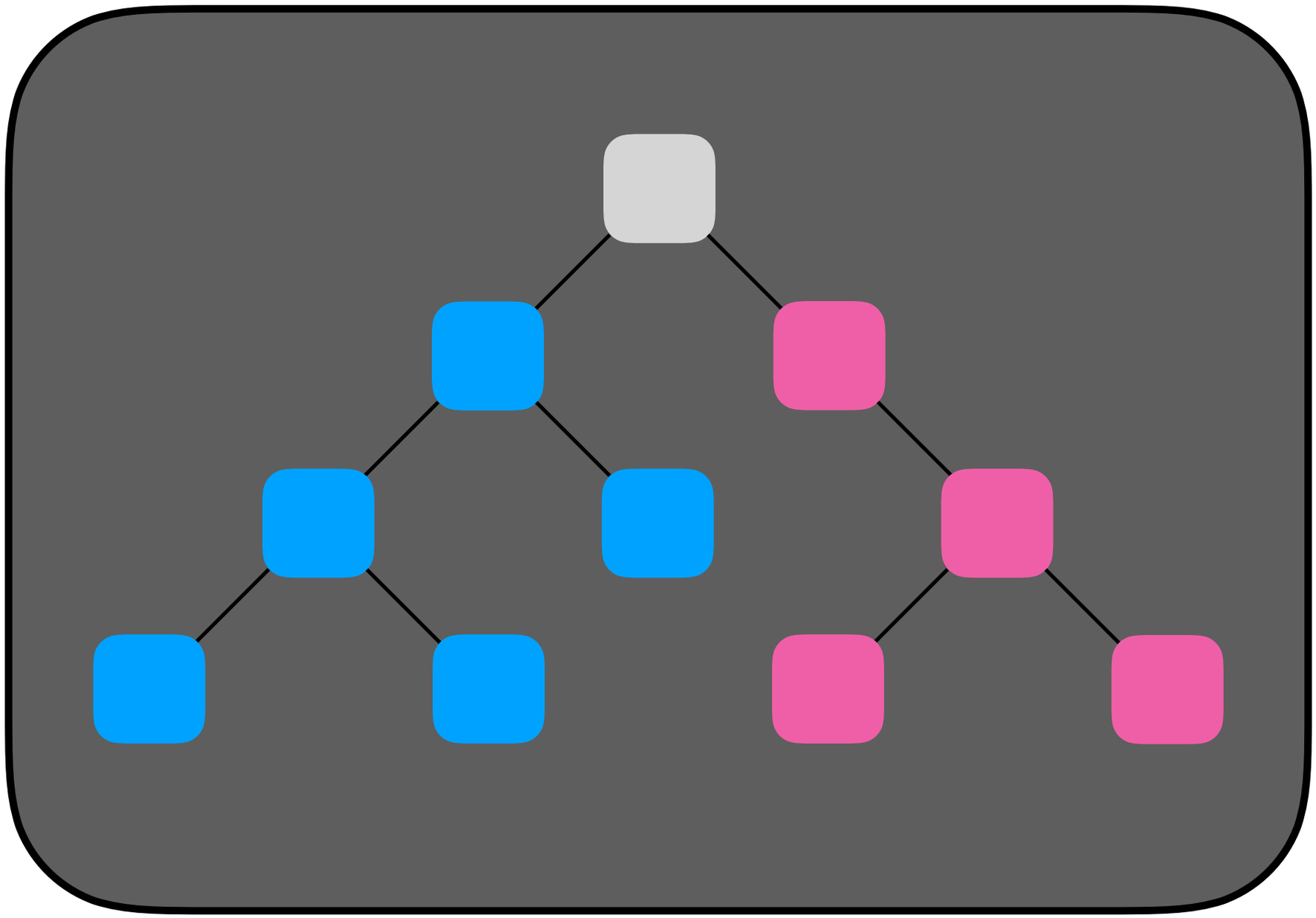
/home

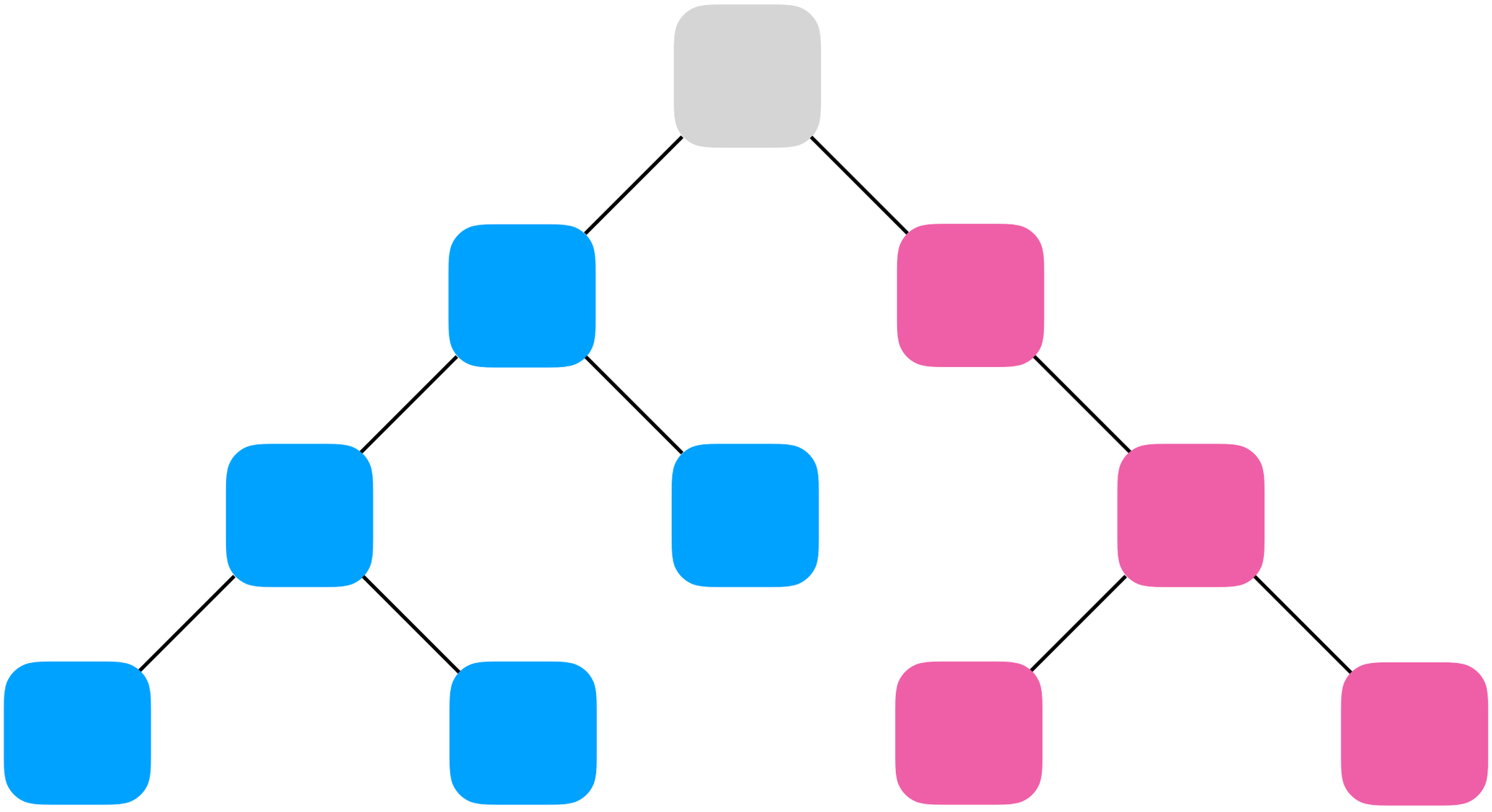


/home

/profile

bundle.js



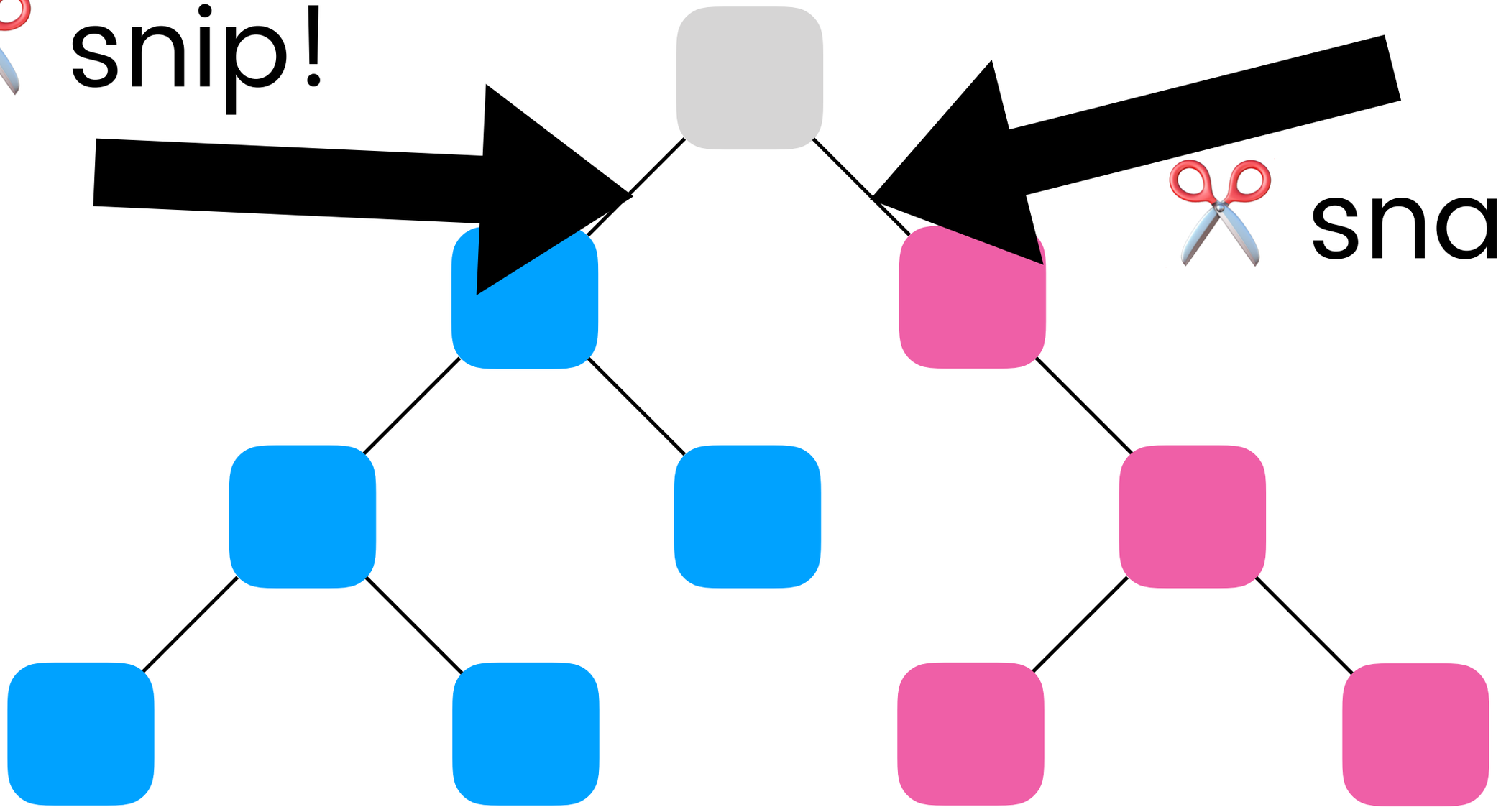


/home

/profile

✂️ snip!

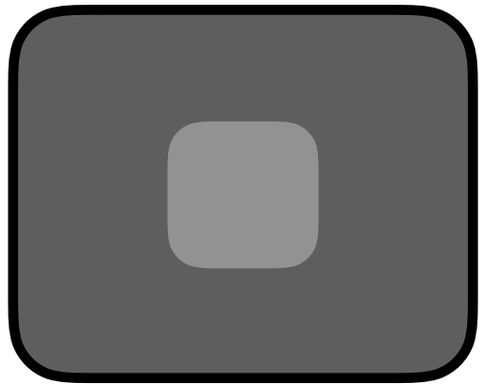
✂️ snap!



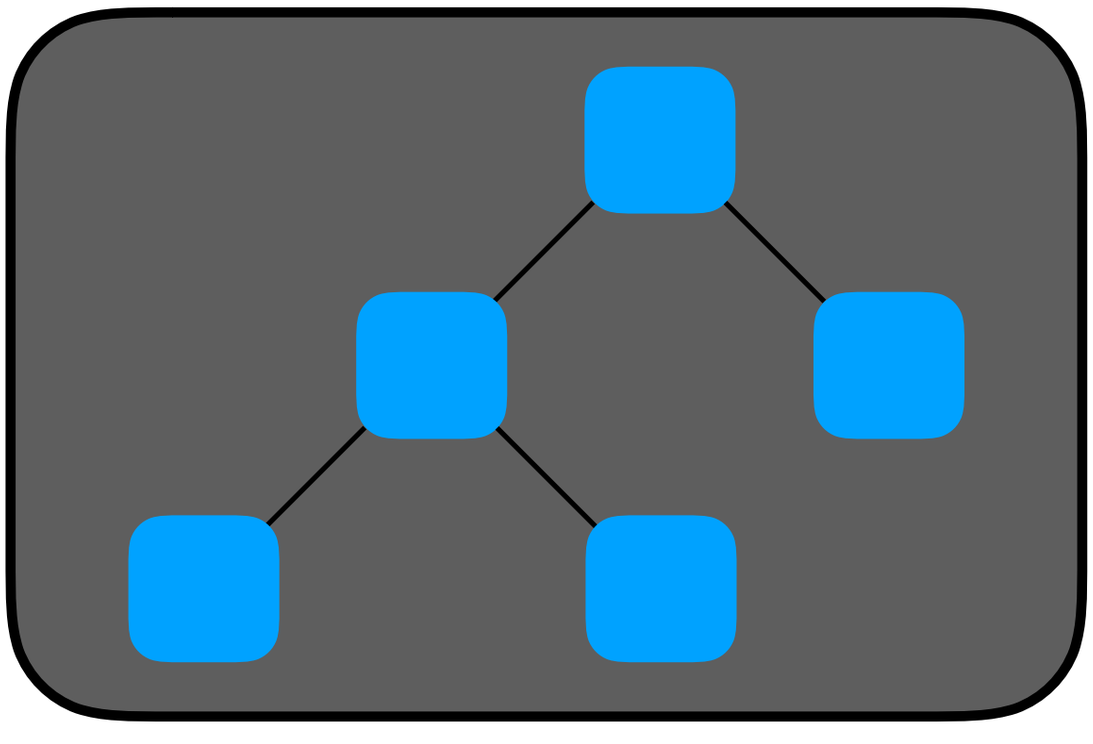
/home

/profile

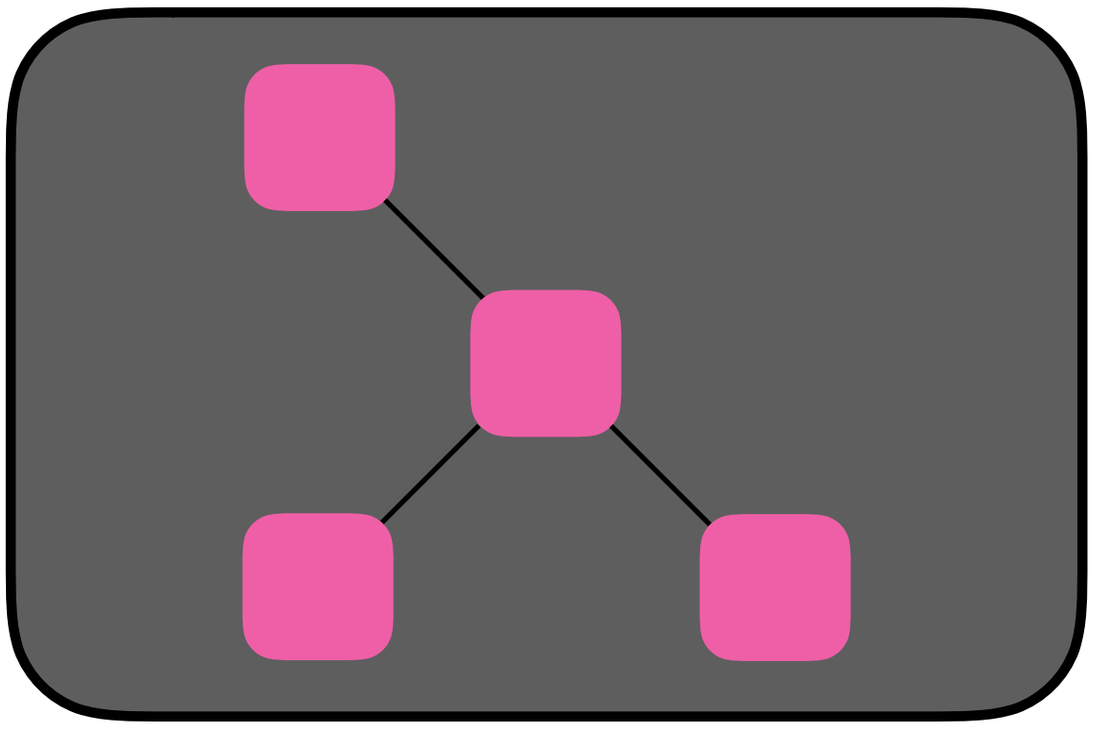
bundle.js

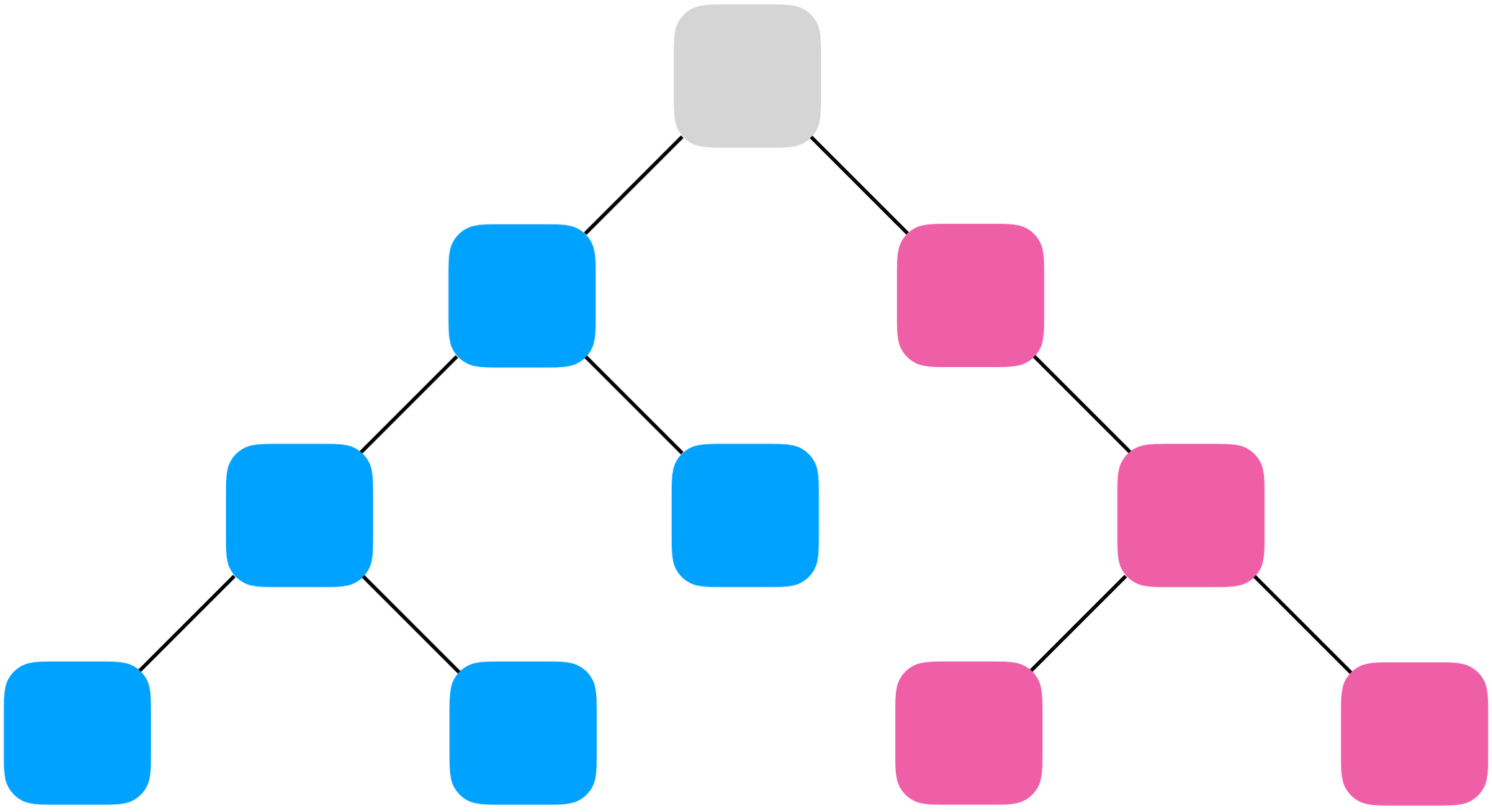


home.js



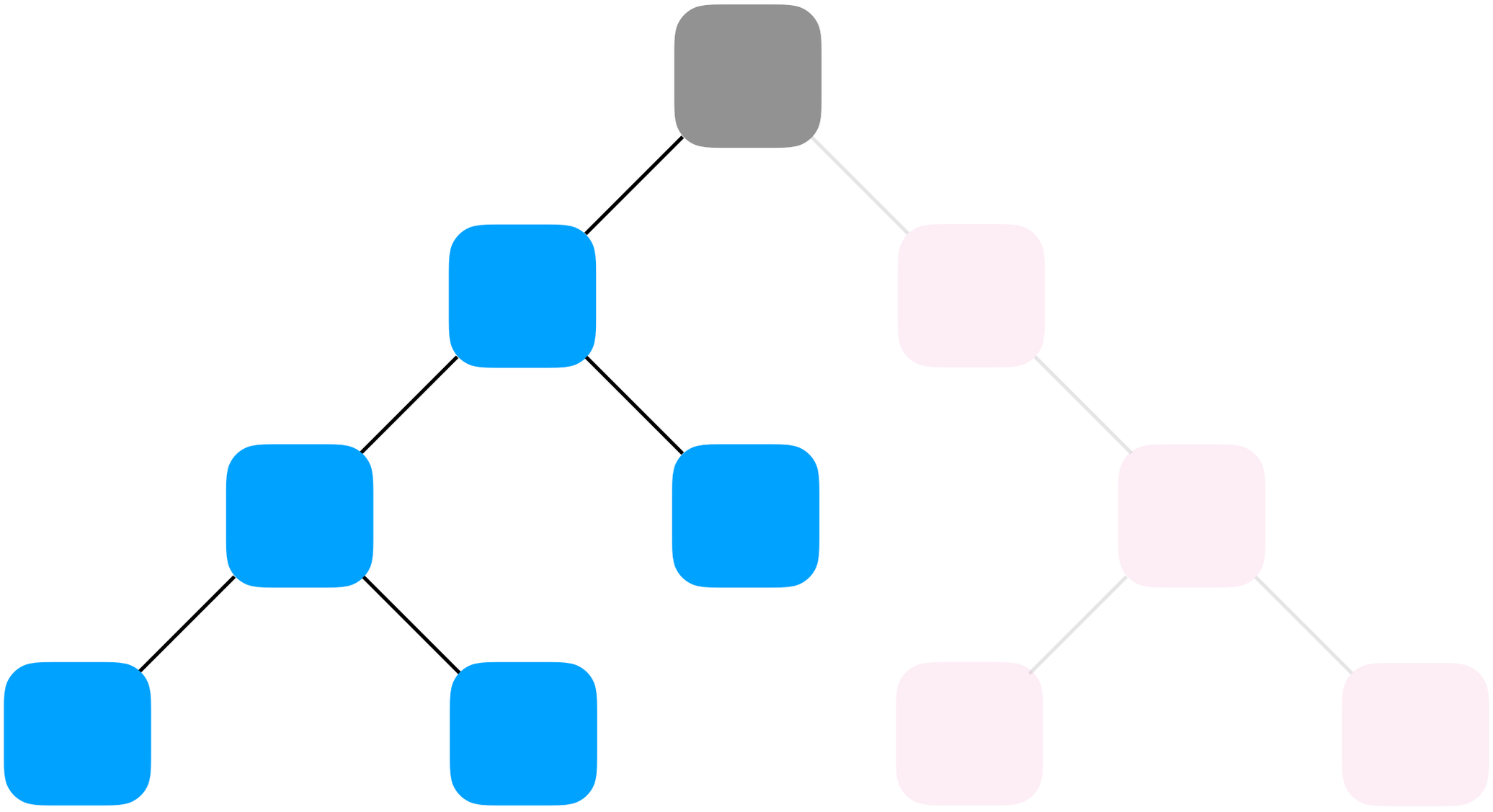
profile.js





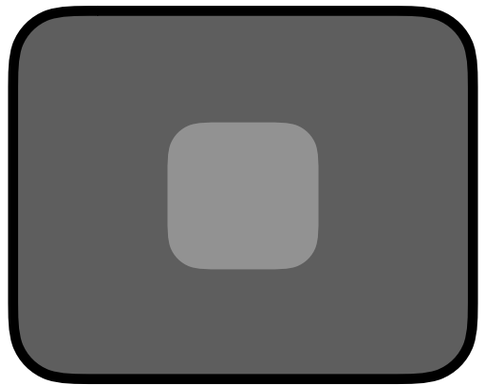
/home

/profile

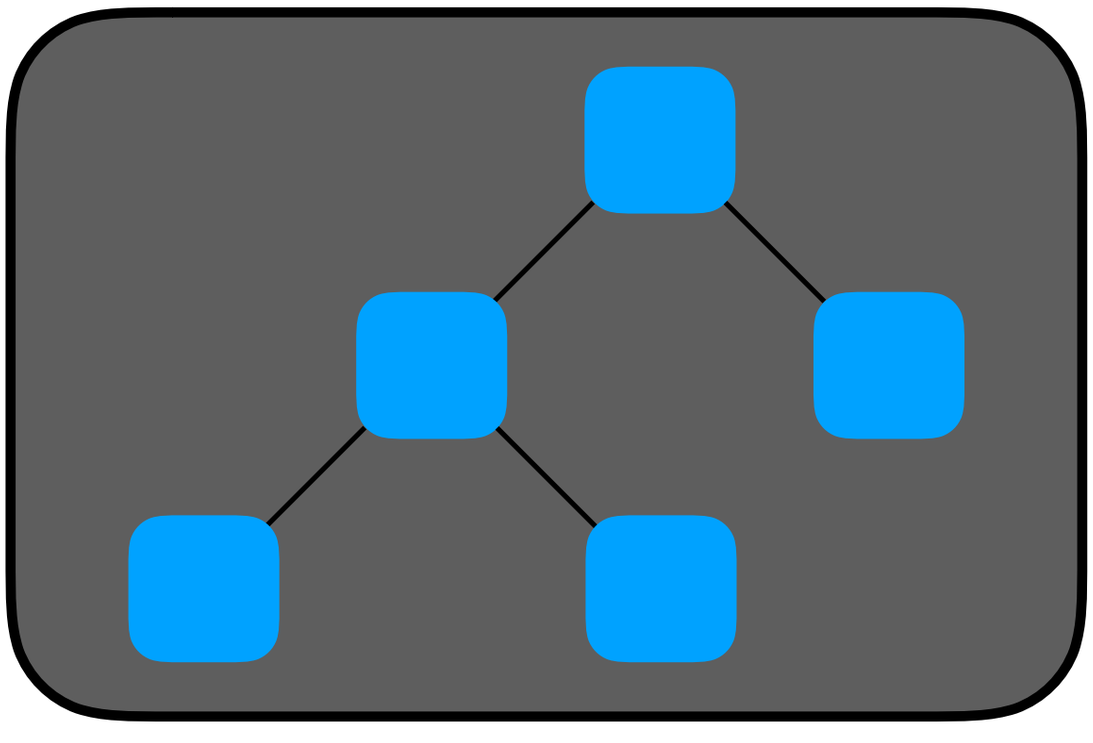


/home

bundle.js



home.js



profile.js

Why should I care?

Internet Speed Test | Fast.com

Glenn

Secure | https://fast.com

FAST

100 Mbps

Compare on SPEEDTEST.NET

?

POWERED BY NETFLIX


prendiville
EAT NEWS
ancona

Why should I care?

it’s simple

How does Code Splitting increase performance?

Caching

Load code

asynchronously

Load code

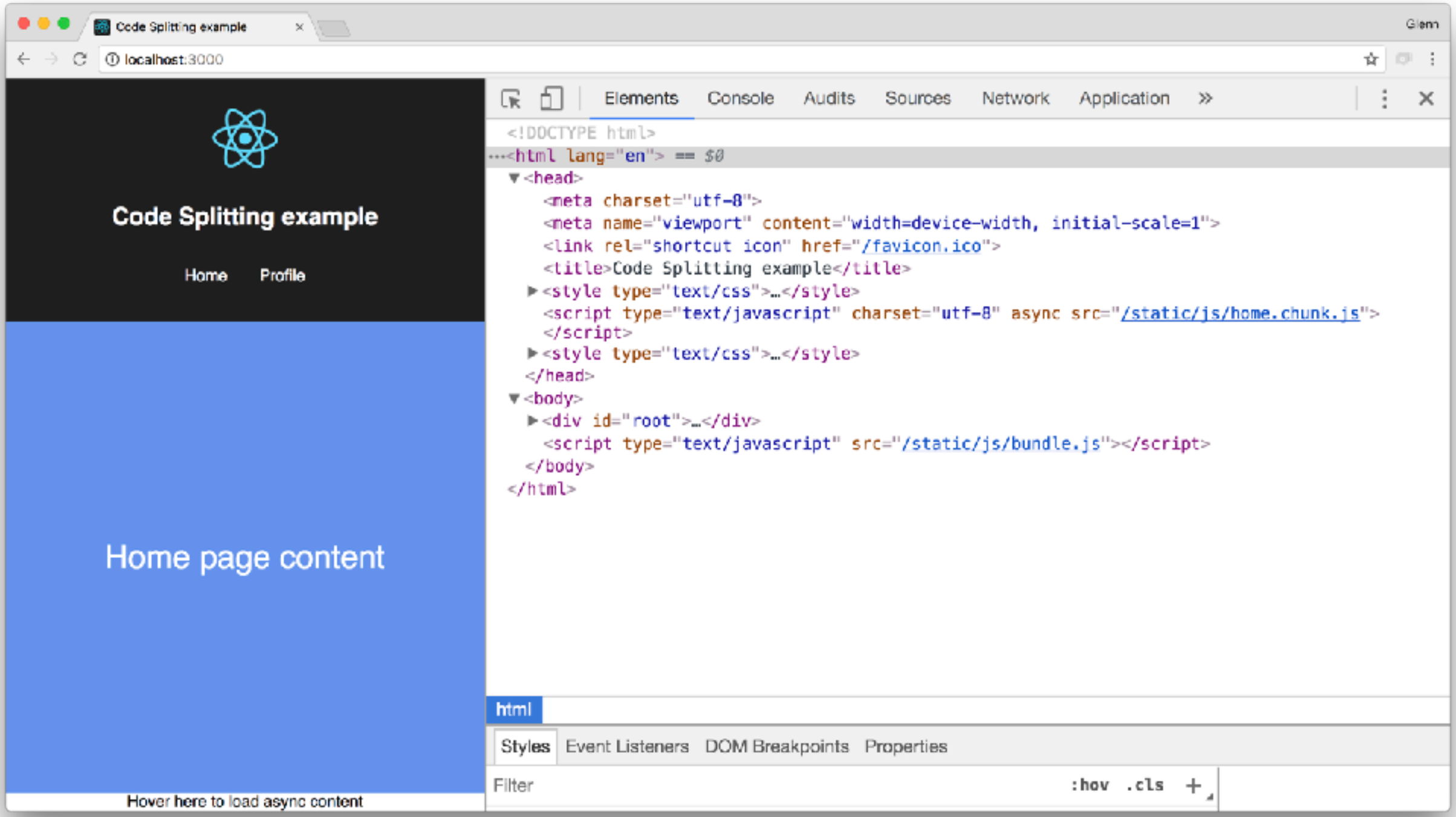
on demand

Load code

as needed

How does it work?

glennreyes/code-splitting-example



```
<html lang="en">
▼<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="shortcut icon" href="/favicon.ico">
    <title>Code Splitting example</title>
  ▶<style type="text/css">…</style>
    <script type="text/javascript" charset="utf-8" async src="/static/js/home.chunk.js">
    </script>
  ▶<style type="text/css">…</style>
  </head>
▼<body>
  ▶<div id="root">…</div>
    <script type="text/javascript" src="/static/js/bundle.js"></script>
  </body>
</html>
```

bundle + home

```
<html lang="en">
▼<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="shortcut icon" href="/favicon.ico">
    <title>Code Splitting example</title>
  ▶<style type="text/css">…</style>
    <script type="text/javascript" charset="utf-8" async src="/static/js/home.chunk.js">
    </script>
  ▶<style type="text/css">…</style>
  </head>
▼<body>
  ▶<div id="root">…</div>
    <script type="text/javascript" src="/static/js/bundle.js"></script>
  </body>
</html>
```

Code Splitting example

localhost:3000

Code Splitting example

Home Profile

Home page content

Hover here to load async content

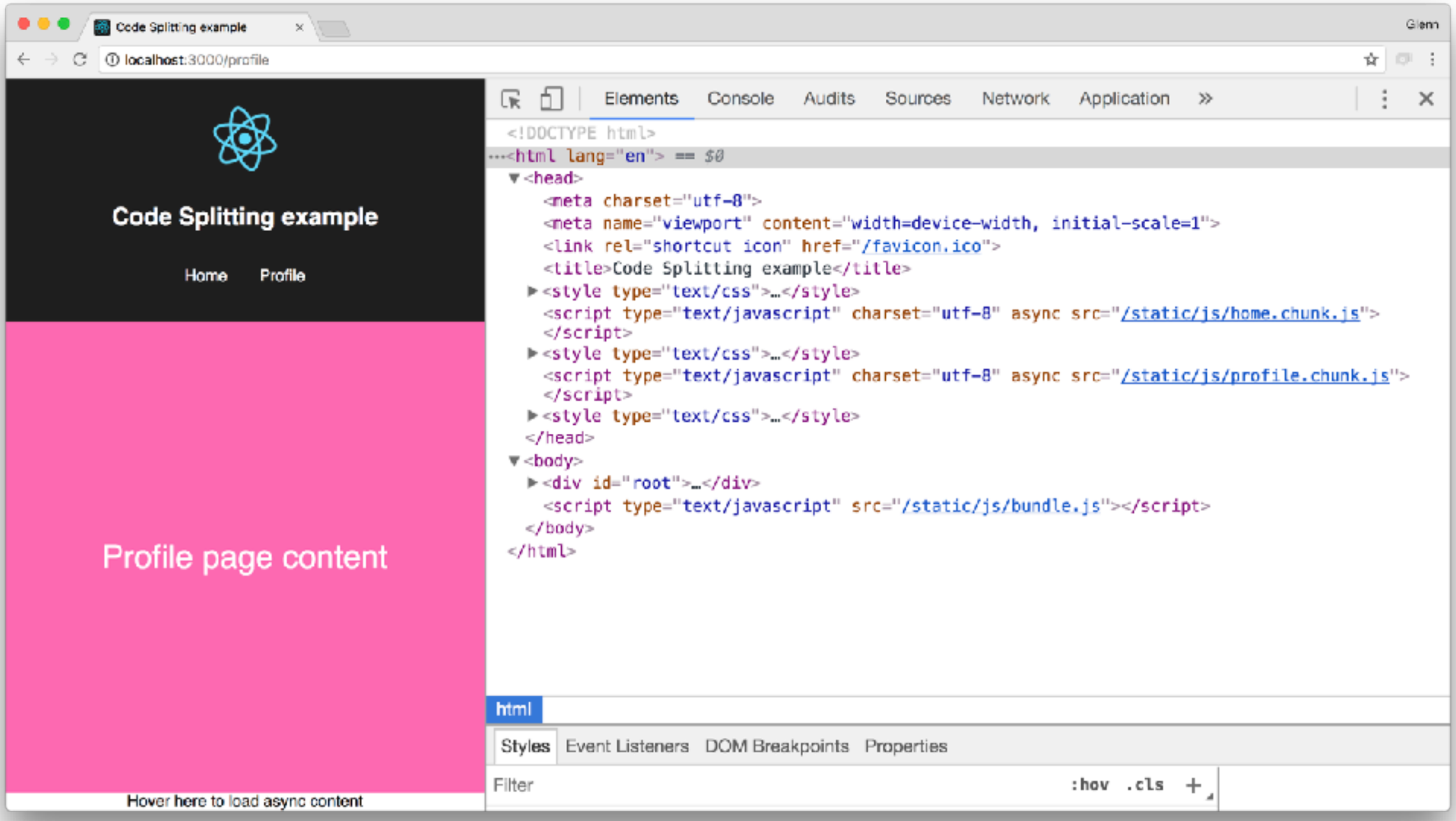
Elements Console Audits Sources Network Application

```
<!DOCTYPE html>
<html lang="en"> == $0
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="shortcut icon" href="/favicon.ico">
    <title>Code Splitting example</title>
  ▶<style type="text/css">…</style>
    <script type="text/javascript" charset="utf-8" async src="/static/js/home.chunk.js">
    </script>
  ▶<style type="text/css">…</style>
  </head>
  <body>
  ▶<div id="root">…</div>
    <script type="text/javascript" src="/static/js/bundle.js"></script>
  </body>
</html>
```

html

Styles Event Listeners DOM Breakpoints Properties

Filter :hov .cls +


Code Splitting example
Glenn
localhost:3000/profile
Code Splitting example
Home
Profile
Profile page content
Hover here to load async content
Elements
Console
Audits
Sources
Network
Application
<!DOCTYPE html>
<html lang="en"> == $0
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="shortcut icon" href="/favicon.ico">
<title>Code Splitting example</title>
<style type="text/css">…</style>
<script type="text/javascript" charset="utf-8" async src="/static/js/home.chunk.js">
</script>
<style type="text/css">…</style>
<script type="text/javascript" charset="utf-8" async src="/static/js/profile.chunk.js">
</script>
<style type="text/css">…</style>
</head>
<body>
…
<script type="text/javascript" src="/static/js/bundle.js"></script>
</body>
</html>
html
Styles
Event Listeners
DOM Breakpoints
Properties
Filter
:hov
.cls

bundle + home + profile

```
<html lang="en">
▼<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="shortcut icon" href="/favicon.ico">
    <title>Code Splitting example</title>
  ▶<style type="text/css">…</style>
    <script type="text/javascript" charset="utf-8" async src="/static/js/home.chunk.js">
    </script>
  ▶<style type="text/css">…</style>
    <script type="text/javascript" charset="utf-8" async src="/static/js/profile.chunk.js">
    </script>
  ▶<style type="text/css">…</style>
  </head>
▼<body>
  ▶<div id="root">…</div>
    <script type="text/javascript" src="/static/js/bundle.js"></script>
  </body>
</html>
```

Before

```
SystemJS.import('./profile.js');

System.import('./profile.js');

require('./profile.js');

require.ensure(['./profile.js'], cb);

MyCustomLoader('./profile.js');
```

A Beginner's Guide to Code Splitting Your React App
https://youtu.be/bb6RCrDaxhw?t=5m51s

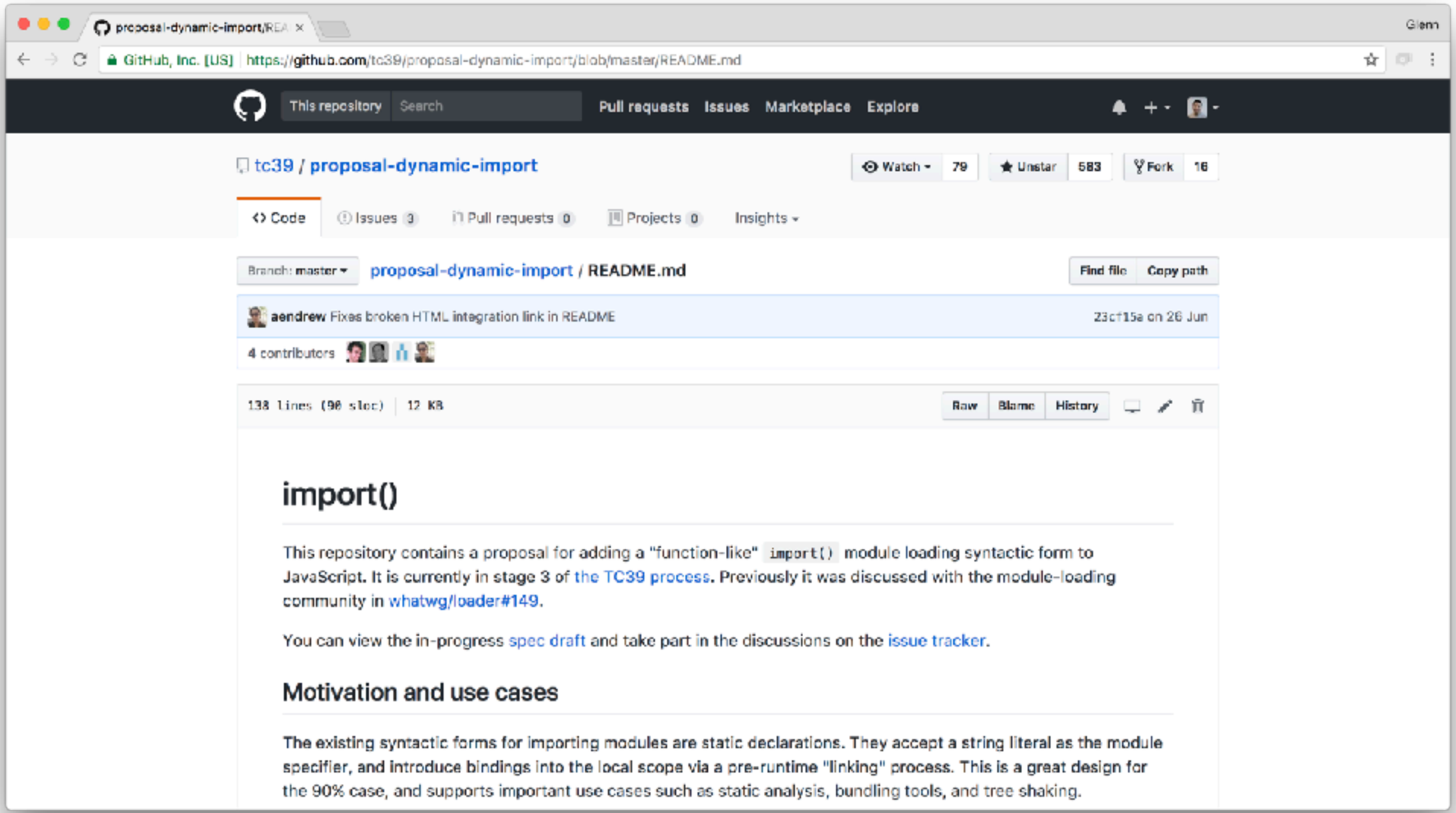
Dynamic import

```
import();
```

Function-like syntactic form

```
import('./Chart');
```

tc39/proposal-dynamic-import



syntax-dynamic-plugin

https://webpack.js.org/guides/code-splitting

https://survivejs.com/webpack/building/code-splitting

Sync vs async imports

Synchronous import

```
import Chart from './Chart';

console.log(Chart);
```


ME
ES6 IMPORTS
COMMONJS IMPORTS
imgflip.com

Asynchronous import

```
import('./Chart').then(module => {
  console.log(module.default);
});
```

Async functions and import();

```
async () => {
  const module = await import('./Chart');
  console.log(module.default);
}
```


ME
DYNAMIC IMPORTS
ES6 IMPORTS
imgflip.com

How with React?

App.js

```
1  import React, { Component } from 'react';
2
3  class App extends Component {
4    state = {
5      AsyncComponent: () => <div>Nothing loaded</div>,
6    };
7
8    async componentDidMount() {
9      const module = await import('./AsyncComponent');
10     const AsyncComponent = module.default;
11
12     this.setState({ AsyncComponent });
13   }
14
15   render() {
16     const { AsyncComponent } = this.state;
17
18     return <AsyncComponent />;
19   }
20 }
21
22 export default App;
23
```

App.js

```
import React, { Component } from 'react';

class App extends Component {
  state = {
    AsyncComponent: () => <div>Nothing loaded</div>,
  };

  async componentDidMount() {
    const module = await import('./AsyncComponent');
    const AsyncComponent = module.default;

    this.setState({ AsyncComponent });
  }

  render() {
    const { AsyncComponent } = this.state;

    return <AsyncComponent />;
  }
}

export default App;
```

src/App.js ⓘ 0 ⚠ 0 ⓘ 0 23:1 522 bytes Prettier ✓

App.js

```
1  import React, { Component } from 'react';
2
3  class App extends Component {
4    state = {
5      AsyncComponent: () => <div>Nothing loaded</div>,
6    };
7
8    async componentDidMount() {
9      const module = await import('./AsyncComponent');
10     const AsyncComponent = module.default;
11
12     this.setState({ AsyncComponent });
13   }
14
15   render() {
16     const { AsyncComponent } = this.state;
17
18     return <AsyncComponent />;
19   }
20 }
21
22 export default App;
23
```

src/App.js ⓘ 0 ⚠ 0 ⓘ 0 23:1 522 bytes Prettier ✓

App.js

```
import React, { Component } from 'react';

class App extends Component {
  state = {
    AsyncComponent: () => <div>Nothing loaded</div>,
  };

  async componentDidMount() {
    const module = await import('./AsyncComponent');
    const AsyncComponent = module.default;

    this.setState({ AsyncComponent });
  }

  render() {
    const { AsyncComponent } = this.state;

    return <AsyncComponent />;
  }
}

export default App;
```

App.js

```
3   class App extends Component {
4     state = {
5       components: [],
6     };
7
8     async componentDidMount() {
9       const components = await Promise.all([
10        import('./Home').then(m => m.default),
11        import('./Profile').then(m => m.default),
12      ]);
13
14      this.setState({ components });
15    }
16
17    render() {
18      return (
19        <div>
20          {this.state.components.map((AsyncComponent, index) => (
21            <AsyncComponent key={index} />
22          ))}
23        </div>
24      );
25    }
```


ReactionGIFs

react-loadable

https://github.com/thejameskyle/react-loadable

sync

```
<Route exact path="/" component={Home} />
```

async

```
<Route exact path="/" component={Loadable({
  loader: () => import('./Home'),
  loading: Loading,
})} />
```

Only 3 more lines of code

create-react-app

https://github.com/facebookincubator/create-react-app

What about SSR?

Next.js

https://github.com/zeit/next.js

Guillermo Rauch
@rauchg

Following

It works for zeit.co with thousands of components and hundreds of code-splitting entry points.

5:07 PM - 8 Aug 2017 from San Francisco, CA

2 Retweets **12** Likes

1 2 12

Where to code split?

#1

Split app and vendor

Andrey Okonetchnikov
Freelance UI-Developer, co-creator of @colorsnapper, @ReactVienna co-organizer, co-owner of @...
Jul 11, 2015 · 5 min read

Long-term caching of static assets with Webpack

Webpack is a great way to package all your static resources such as JavaScript, CSS or even images, but to effectively use generated assets in production one should leverage long-term caching. Documentation on this topic is scattered across different resources and it is not quite easy to get it right. The aim of this article is to guide front-end developers through the complete setup.

Kent C. Dodd

tl;dr

To enable long-term caching of static resources produced by webpack:

1. Use *[chunkhash]* to add a content-dependent cache-buster to each file.

Tim Sebastian Follow
Something, something javascript at Atlassian // twitter @schnibl // github.com/timse // bitbucket.org...
May 20 · 6 min read

Predictable long term caching with Webpack

Getting long-term caching right with Webpack is a problem that never really got a final answer.

There is, however, this open issue on Github:

- it has 162 comments
- is soon to have its 2 birthday
- a lot of suggestions that often make things worse
- and probably quite some google hits.

Arguably there is an easy answer, you could use the `RecordsPlugin` (given the documentation on this is a little sparse. But that requires you to keep track of

#2

Split at route level

webpack chunk names

```
import(/* webpackChunkName: "home" */ './Home'),

import(/* webpackChunkName: "profile" */ './Profile'),
```

/home.360c0ce7.chunk.js

/profile.8b26492e.chunk.js

#3

Split at component level

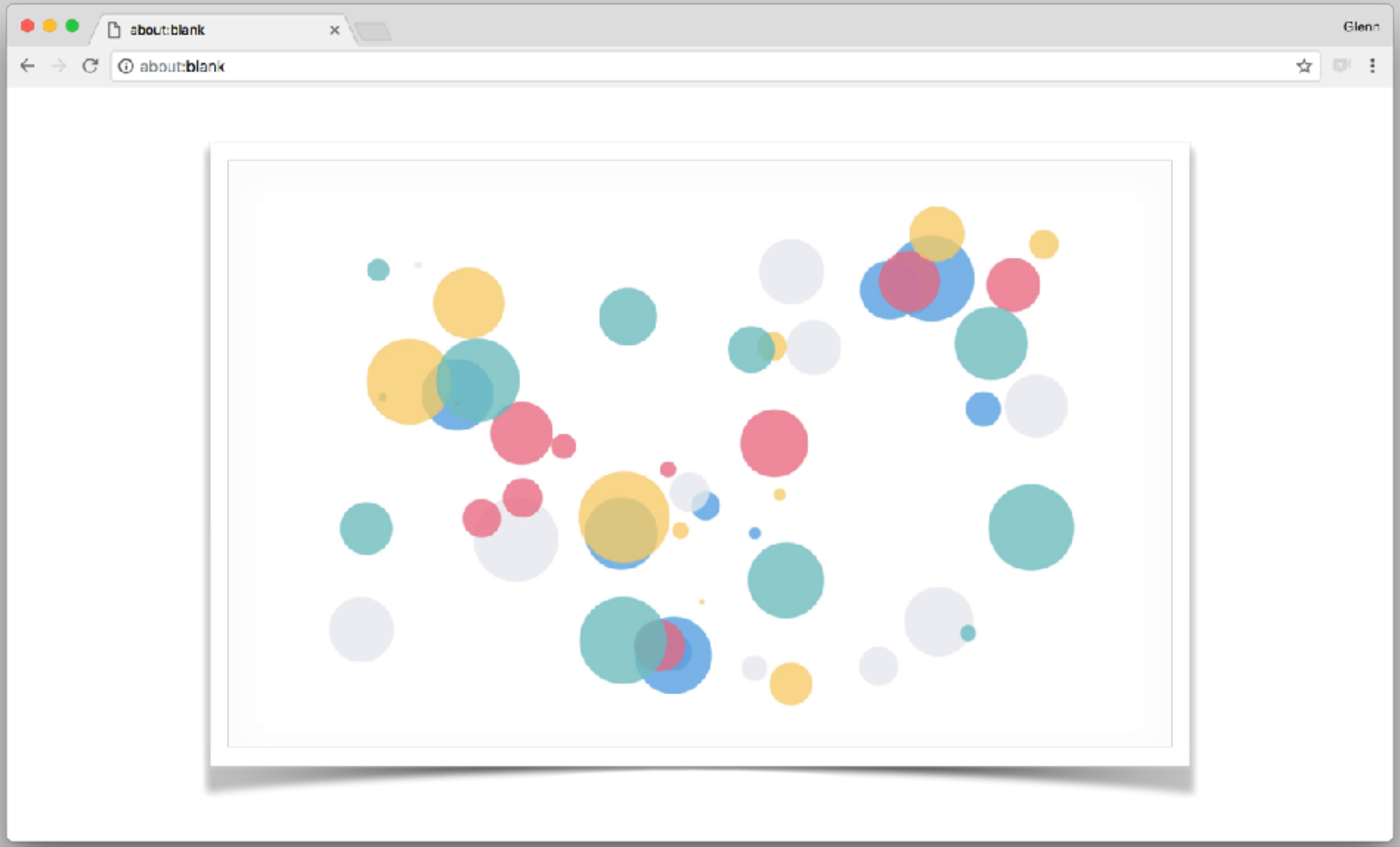


Chart.js

Predict user intent and prefetch

Where to code split

- ✂️ **App** and **vendor**
- ✂️ **Routes**
- ✂️ **Components**

Good reads

- **Straightforward code splitting with React and Webpack**
  https://hackernoon.com/straightforward-code-splitting-with-react-and-webpack

- **ES proposal: import() – dynamically importing ES** modules
  http://2ality.com/2017/01/import-operator.html

- **Tree Shaking vs Code Splitting: A Real-World Benchmark**
  https://medium.com/outreach-engineering/tree-shaking-vs-code-splitting-a-real-world-benchmark-bbbf36245db3

- **Progressive Web Apps with React.js: Part 2 — Page Load Performance**
  https://medium.com/@addyosmani/progressive-web-apps-with-react-js-part-2-page-load-performance-33b932d97cf2

- **The PRPL Pattern**
  https://developers.google.com/web/fundamentals/performance/prpl-pattern/


ARENA

Recap

✨ import();

✨ Load code as needed

✨ Up to 5x faster page load

✨ Split vendor, at route & component

✨ create-react-app & Next.js

 glennreyes

 @glnnrys