

# REACT: HTML STILL MATTERS\*



@jbmoelker

# REACT: HTML STILL MATTERS\*

*\*unless you're developing for React Native*



@jbmoelker

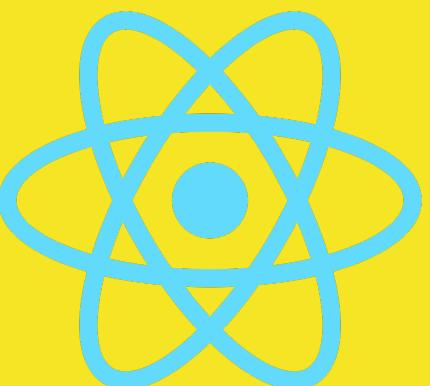
# ABOUT ME



Jasper Moelker  
[@jbmoelker](https://twitter.com/jbmoelker)

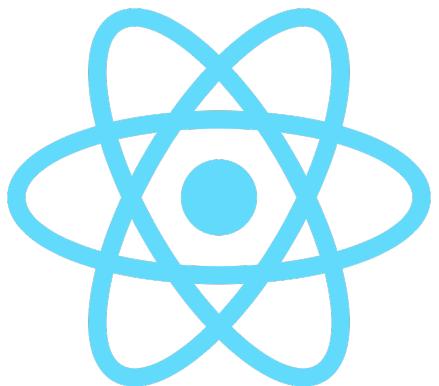


front-end architect  
[@devoorhoede](https://twitter.com/devoorhoede)



React amateur

# DEVELOPERS ❤ CONVENIENCE



## REACT

- does **1 thing**, does it well
- is **module** oriented
- plays well with other **JS**
- has **declarative** JSX syntax
- has great tools, docs, **community**

JSX is for developers  
**HTML is for browsers**



@jbmoelker

# BROWSERS

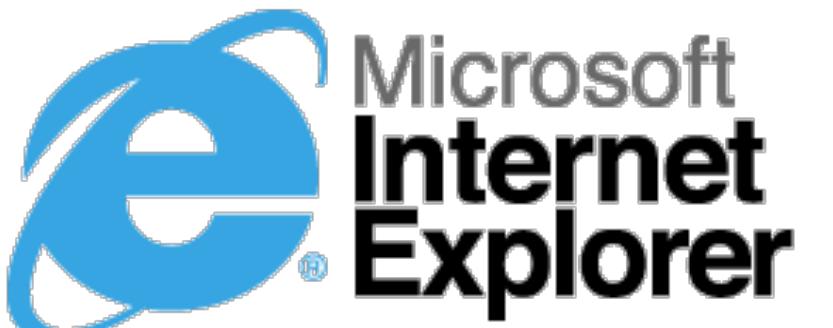


# BROWSERS

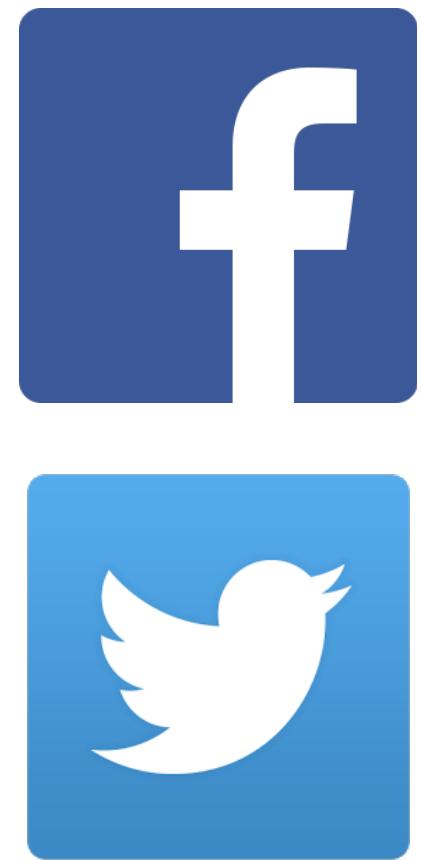
modern



oldies



in-app



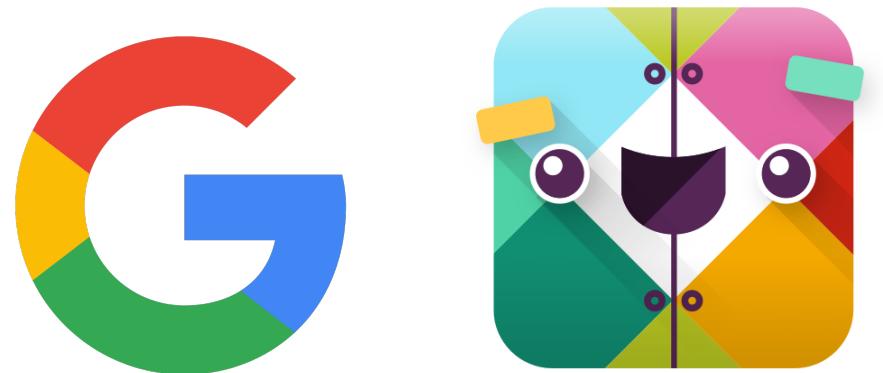
proxy



text only

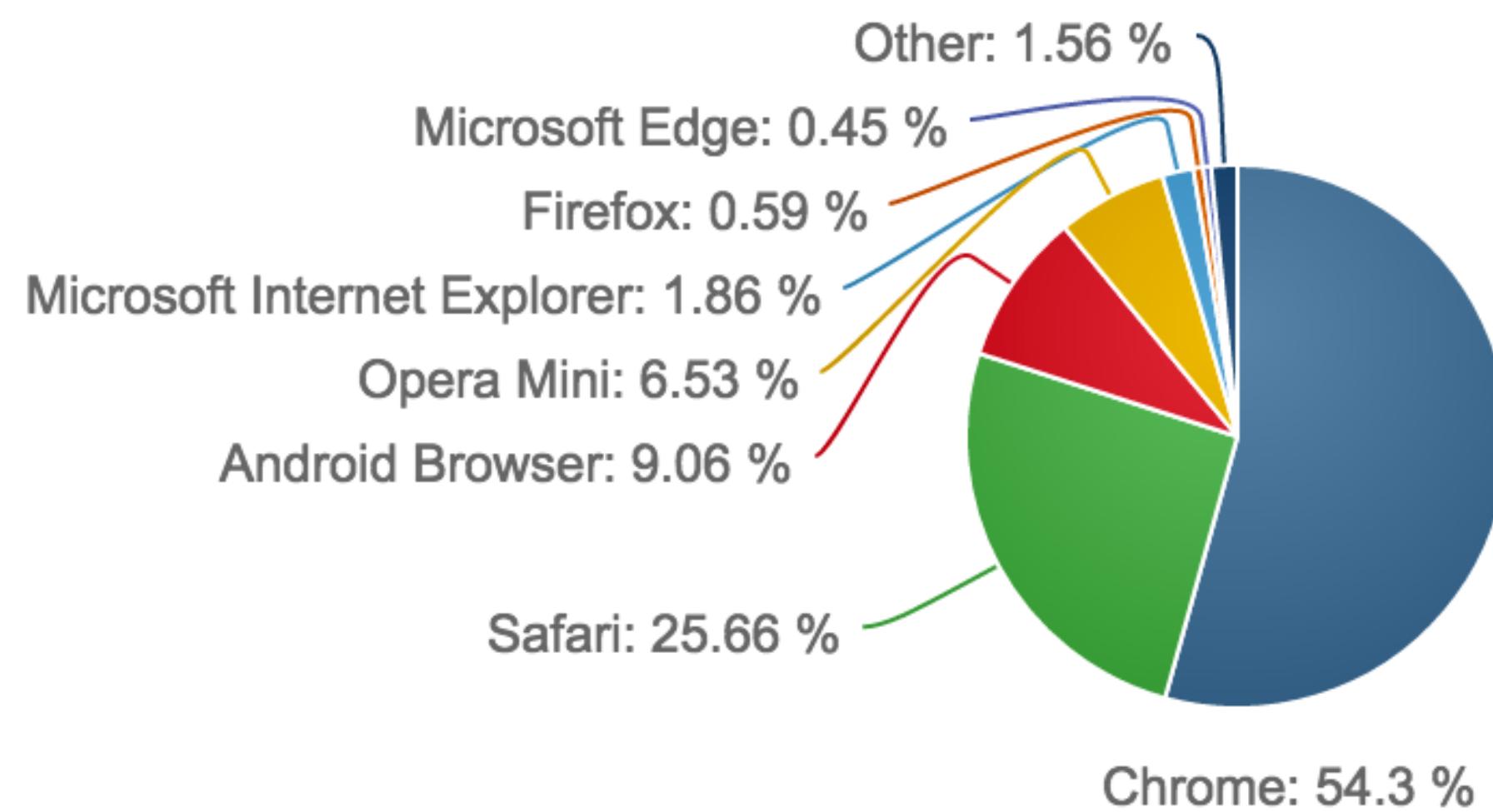


robots

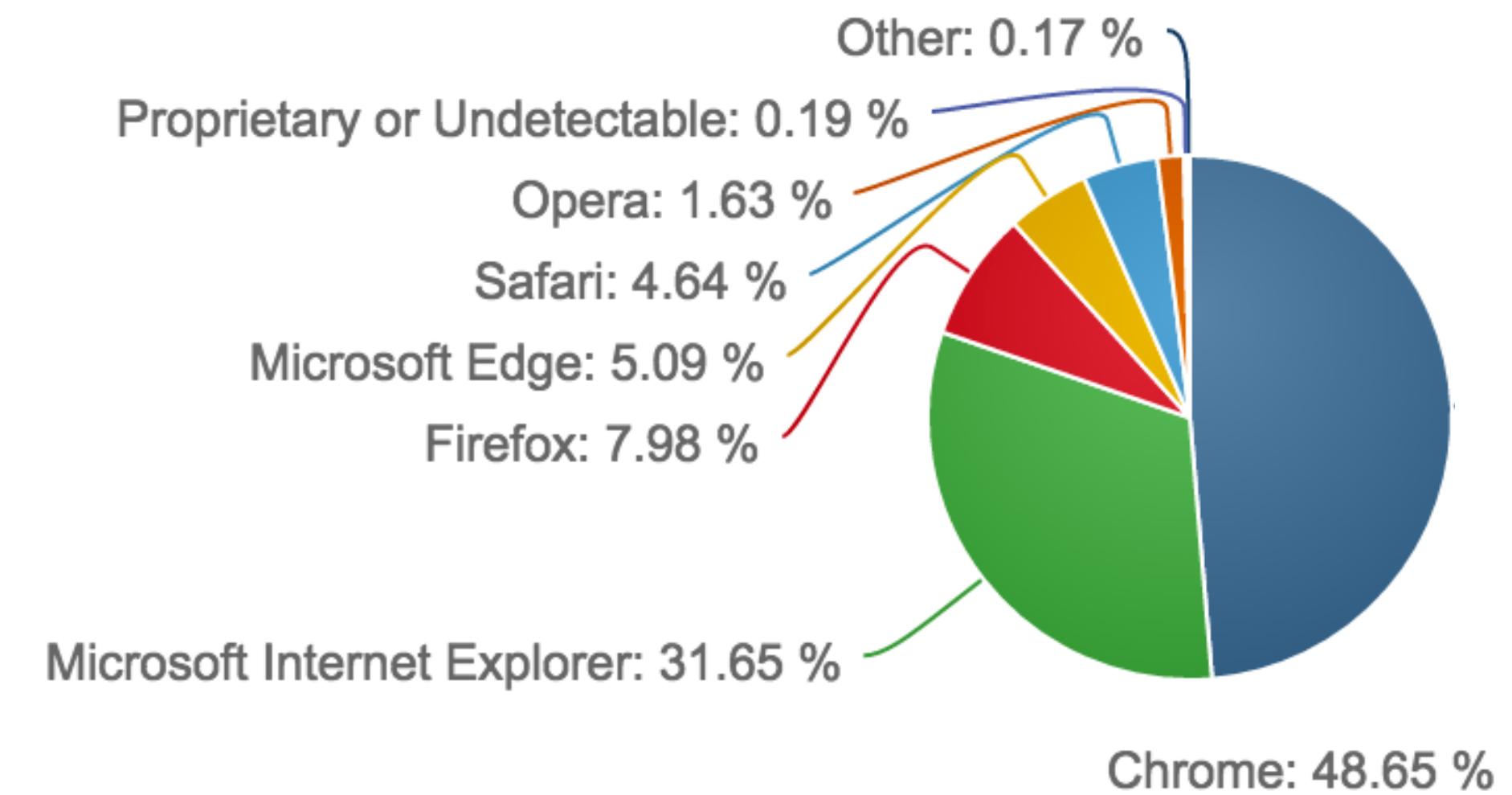


# BROWSER MARKETSHARE

mobile

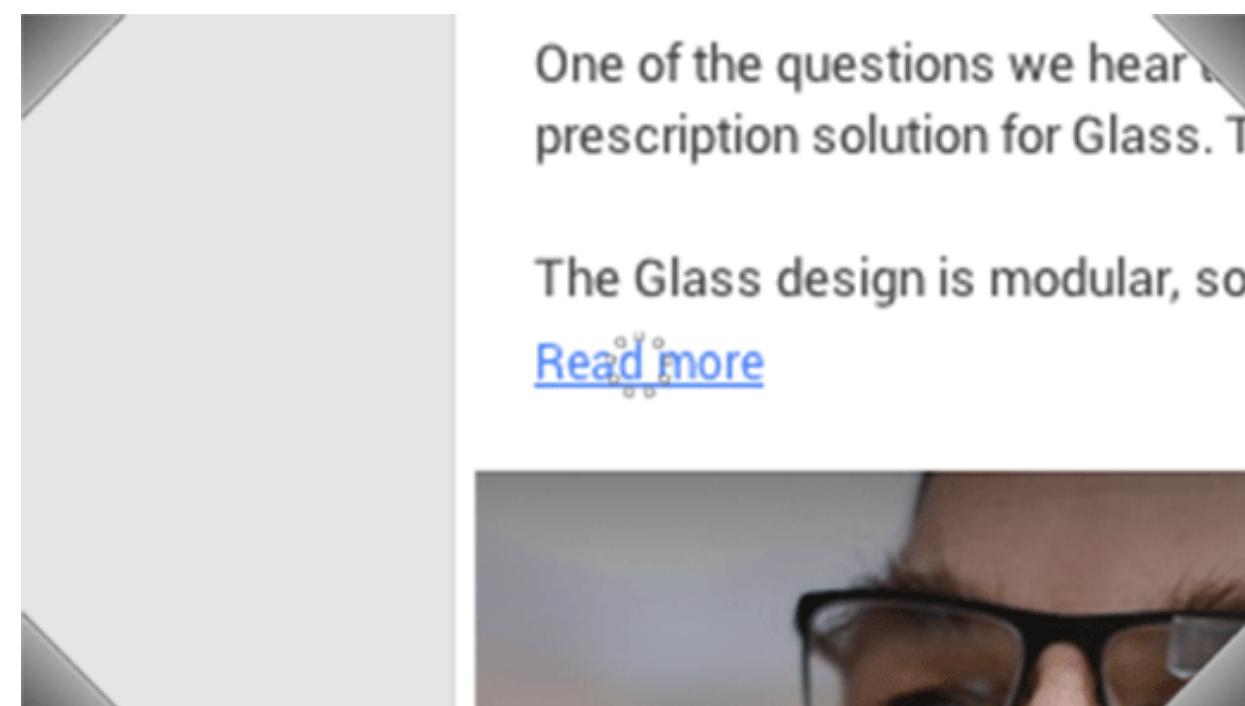
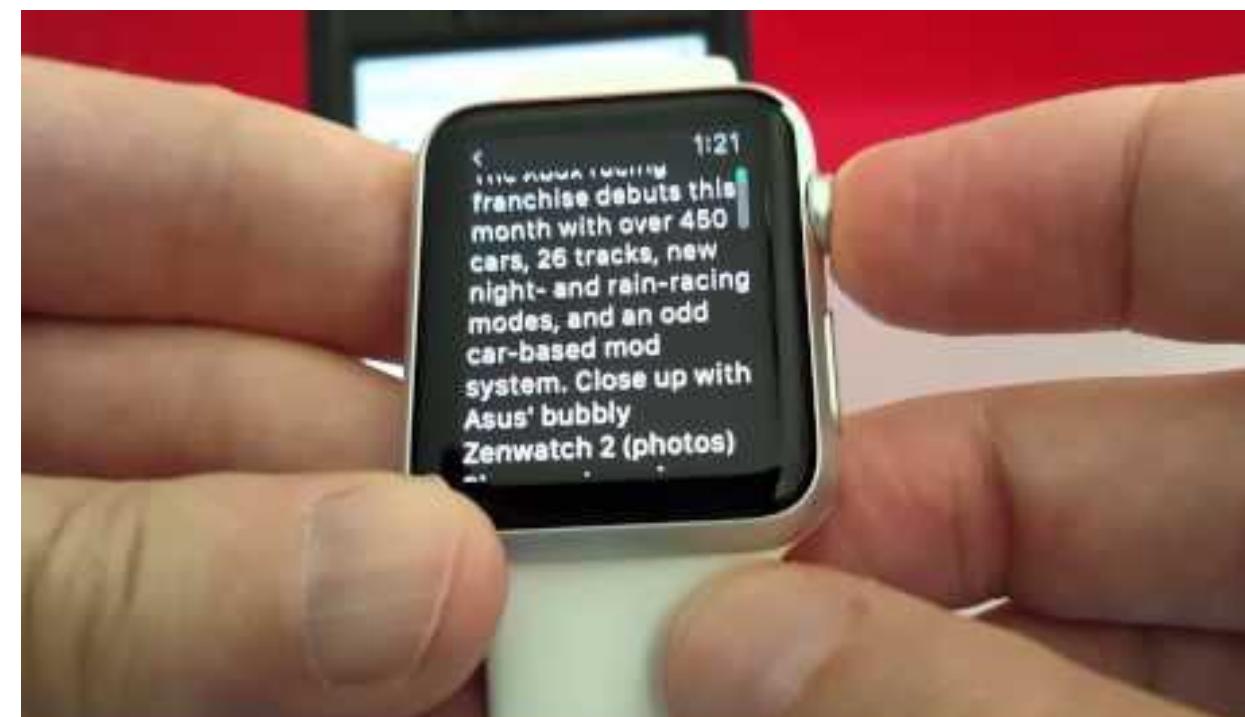


desktop



Opera Mini > 6%  
marketshare mobile!

# MORE BROWSERS



# BROWSER TESTING



@jbmoelker

# React

A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES

[Get Started](#)[Download React v15.3.2](#)

## Declarative

React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

Declarative views make your code more predictable and easier to debug.

## Component-Based

Build encapsulated components that manage their own state, then compose them to make complex UIs.

Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM.

## Learn Once, Write Anywhere

We don't make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.

React can also render on the server using Node and power mobile apps using [React Native](#).

## A Simple Component

React components implement a `render()` method that takes input data and returns what to display. This example uses an XML-like syntax called JSX. Input data that is passed into the component can be accessed by `render()` via `this.props`.

React is flexible and provides hooks that allow you to interface with other libraries and frameworks. This example uses **remarkable**, an external Markdown library, to convert the textarea's value in real time.

Live JSX EditorCompiled JS

```
class MarkdownEditor extends React.Component {
  constructor(props) {
    super(props);
    this.handleChange = this.handleChange.bind(this);
    this.state = {value: 'Type some *markdown* here!'};
  }

  handleChange() {
    this.setState({value: this.refs.textarea.value});
  }

  getRawMarkup() {
    var md = new Remarkable();
    return { __html: md.render(this.state.value) };
  }

  render() {
    return (
      <div className="MarkdownEditor">
        <h3>Input</h3>
        <textarea
          onChange={this.handleChange}
          ref="textarea"
          defaultValue={this.state.value} />
        <h3>Output</h3>
        <div
          className="content"
          dangerouslySetInnerHTML={this.getRawMarkup()}>
        </div>
      );
    }
  }

ReactDOM.render(<MarkdownEditor />, mountNode);
```

**Input**

Type some \*markdown\* here!

**Output**

Type some *markdown* here!

React is flexible and provides hooks that allow you to interface with other libraries and frameworks. This example uses **remarkable**, an external Markdown library, to convert the textarea's value in real time.

Live JSX Editor   Compiled JS

```
class MarkdownEditor extends React.Component {
  constructor(props) {
    super(props);
    this.handleChange = this.handleChange.bind(this);
    this.state = {value: 'Type some *markdown* here!'};
  }

  handleChange() {
    this.setState({value: this.refs.textarea.value});
  }

  getRawMarkup() {
    var md = new Remarkable();
    return { __html: md.render(this.state.value) };
  }

  render() {
    return (
      <div className="MarkdownEditor">
        <h3>Input</h3>
        <textarea
          onChange={this.handleChange}
          ref="textarea"
          defaultValue={this.state.value} />
        <h3>Output</h3>
        <div
          className="content"
          dangerouslySetInnerHTML={this.getRawMarkup()}>
        </div>
      </div>
    );
  }
}

ReactDOM.render(<MarkdownEditor />, mountNode);
```

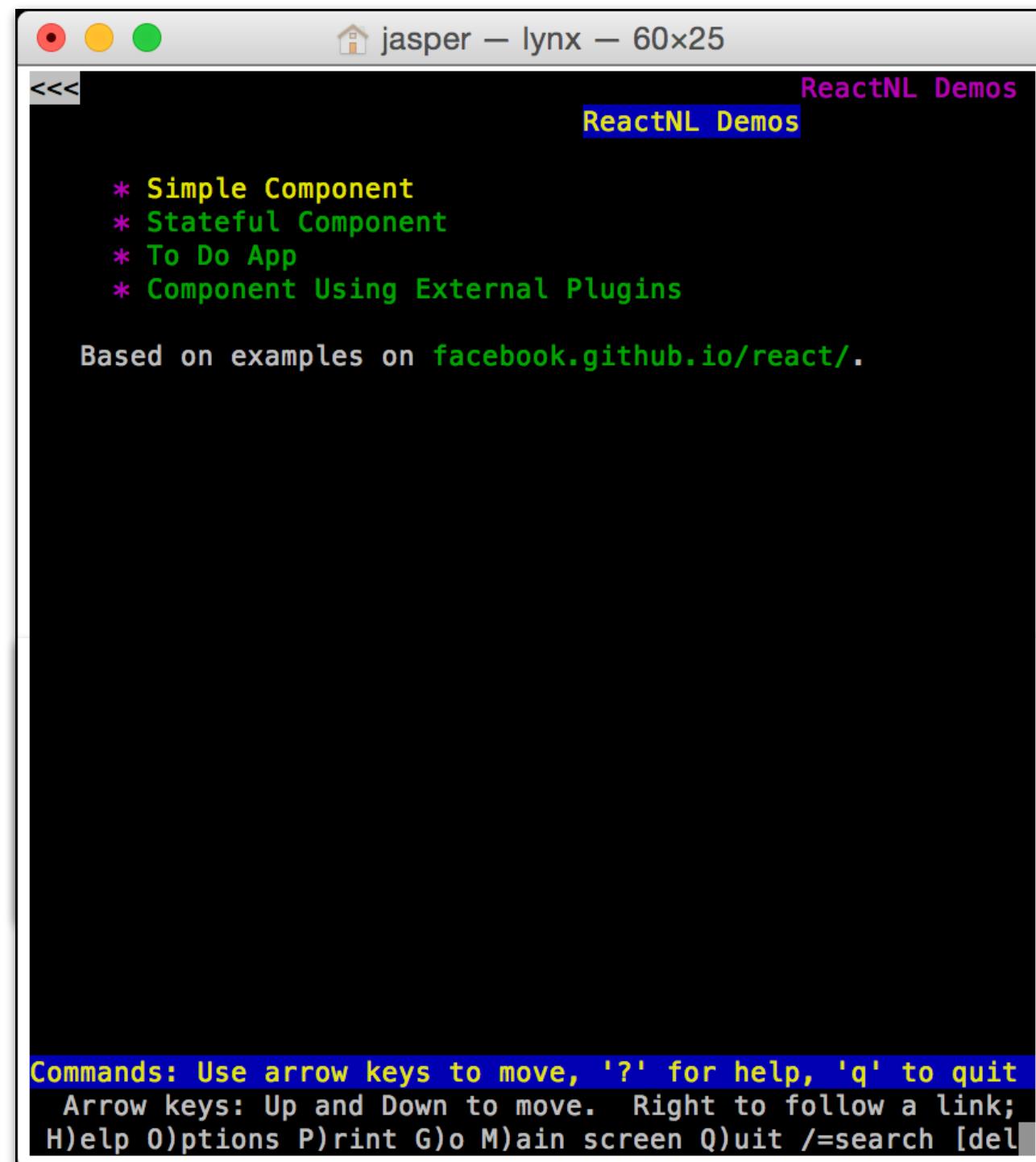
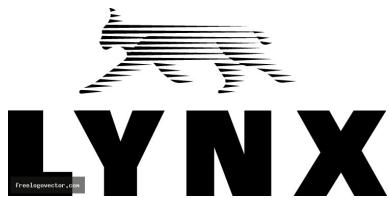
**Input**

Type some \*markdown\* here!

**Output**

Type some *markdown* here!

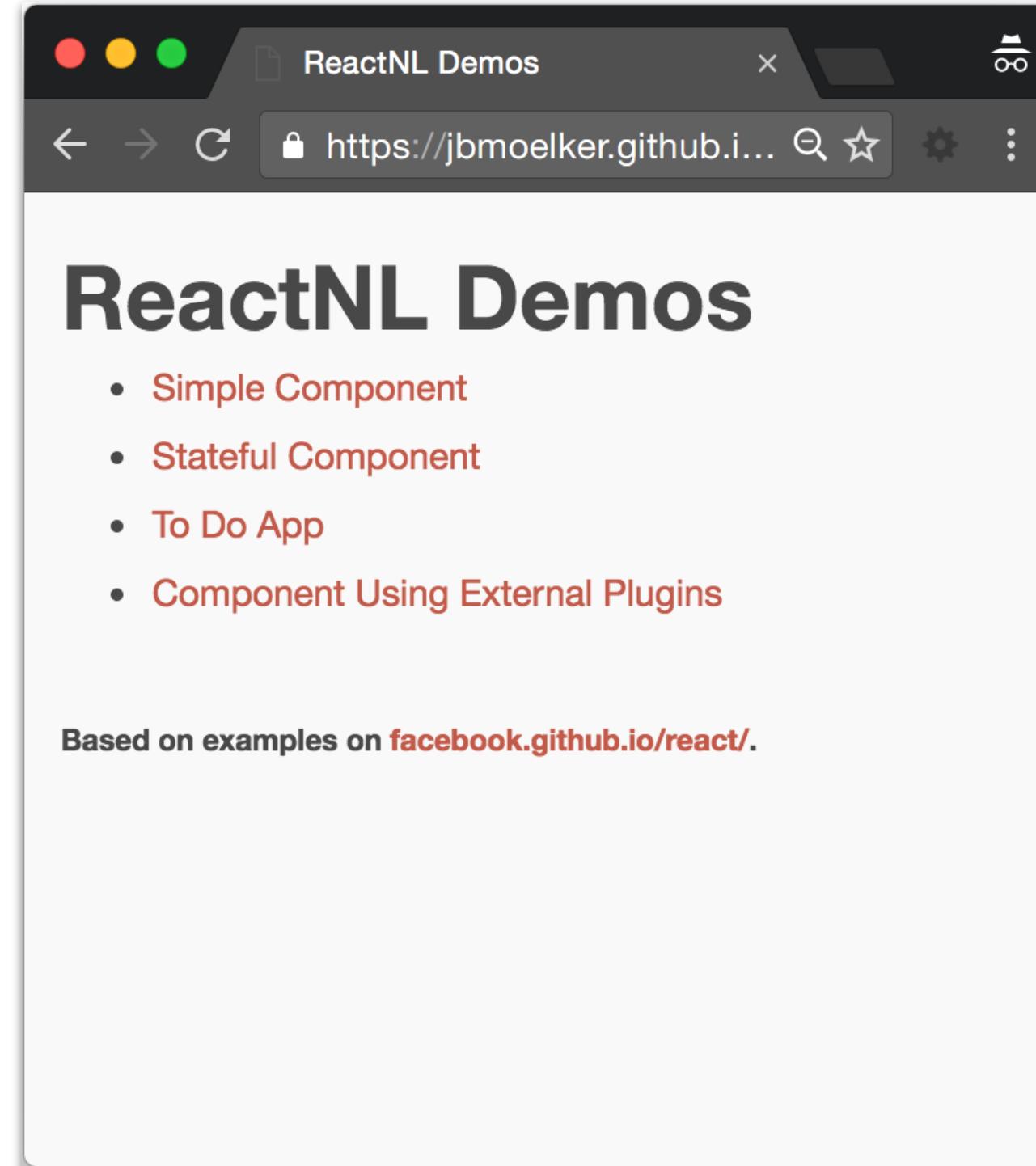
# EXAMPLES FROM REACT DOCS



A screenshot of a Lynx web browser window titled "jasper - lynx - 60x25". The title bar also includes "ReactNL Demos". The main content area displays a list of links:

- \* Simple Component
- \* Stateful Component
- \* To Do App
- \* Component Using External Plugins

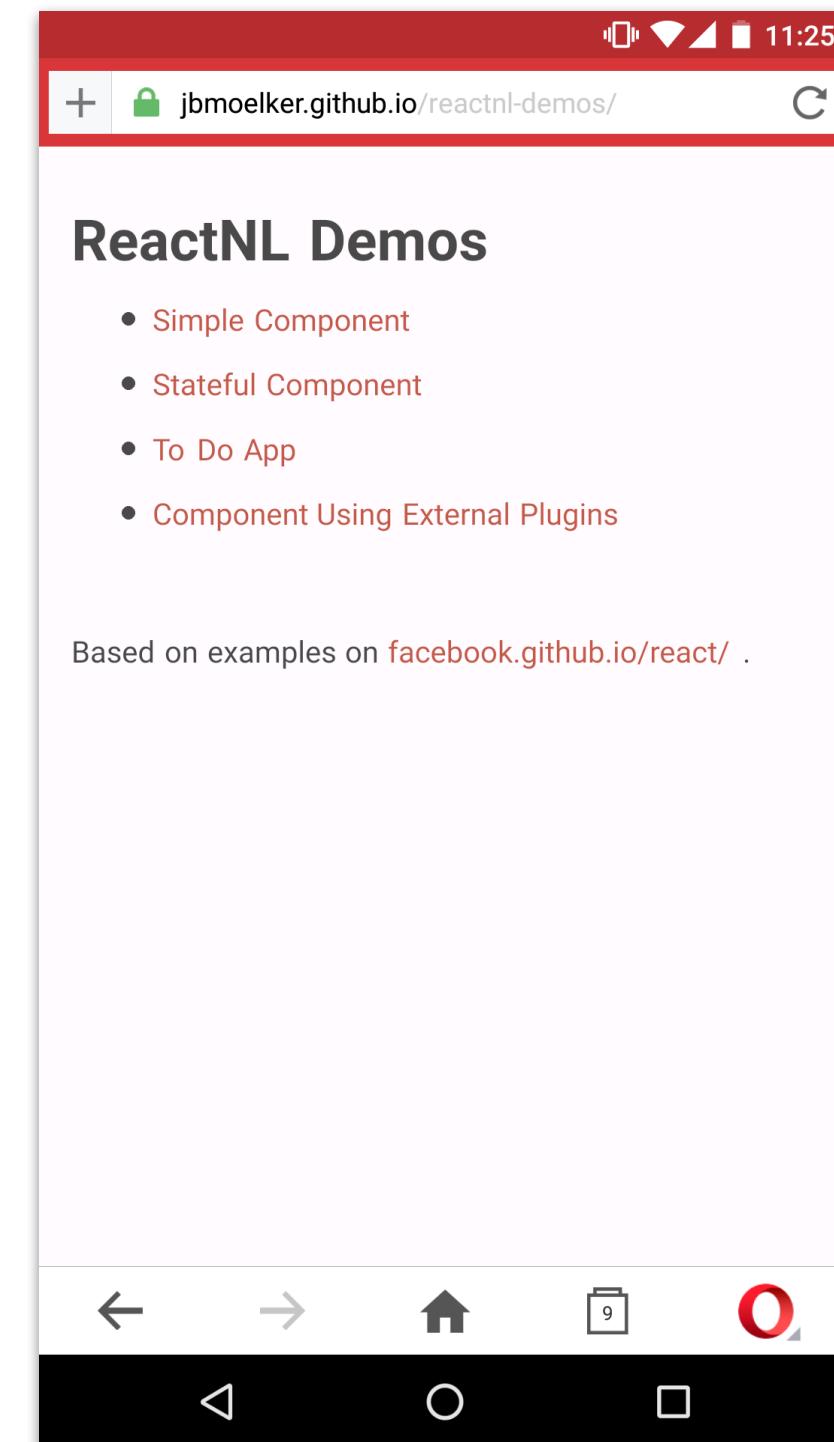
Below the list, a note states: "Based on examples on [facebook.github.io/react/](https://facebook.github.io/react/)". At the bottom of the window, there is a blue status bar with the text: "Commands: Use arrow keys to move, '?' for help, 'q' to quit" and "Arrow keys: Up and Down to move. Right to follow a link; H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [del]".



A screenshot of a Google Chrome browser window titled "ReactNL Demos". The address bar shows the URL "https://jbmoelker.github.i...". The main content area features a large heading "ReactNL Demos" and a bulleted list of links:

- Simple Component
- Stateful Component
- To Do App
- Component Using External Plugins

Below the list, a note states: "Based on examples on [facebook.github.io/react/](https://facebook.github.io/react/)".



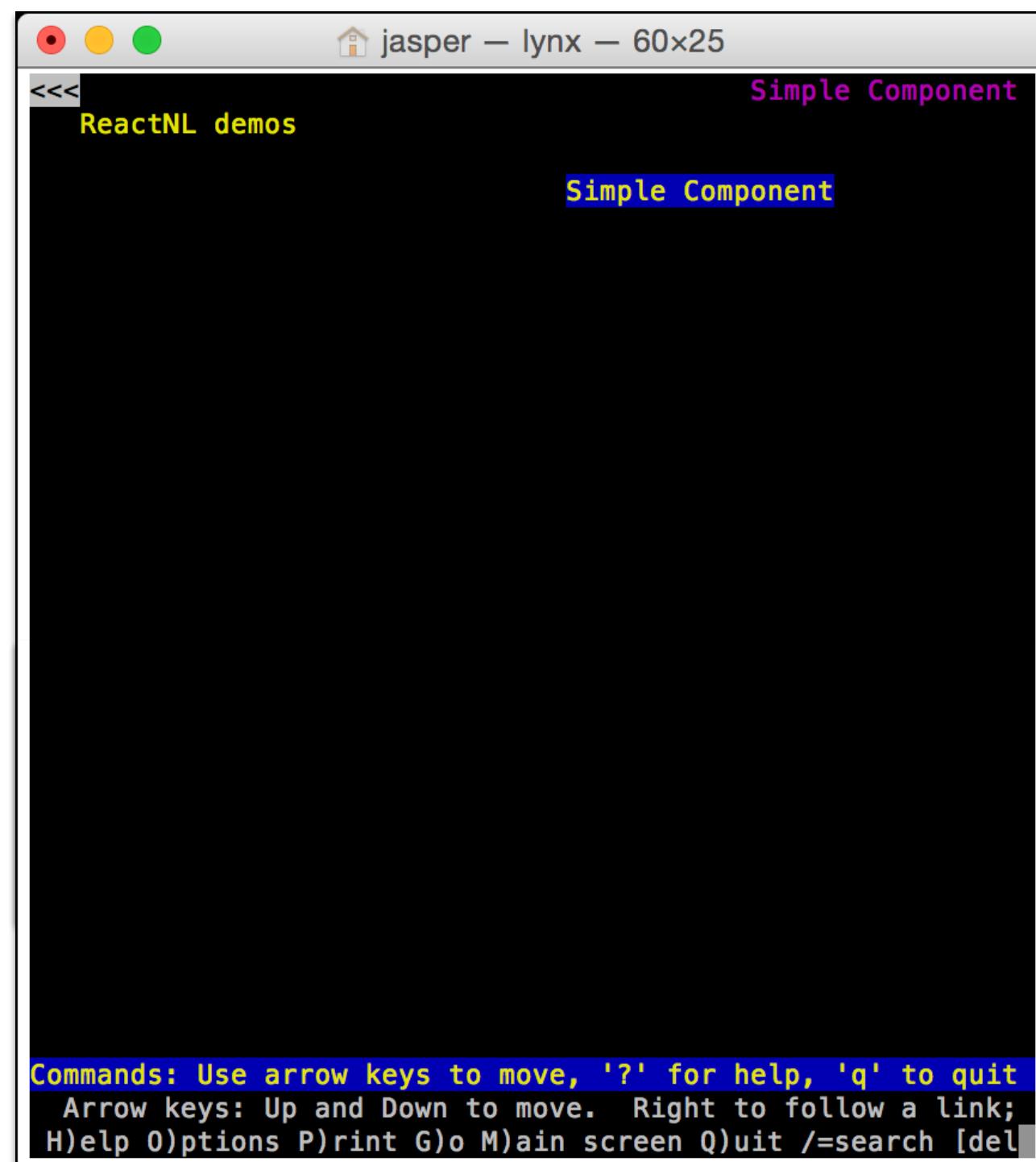
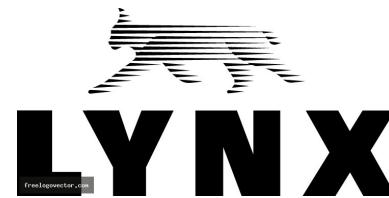
A screenshot of a Mini browser window titled "ReactNL Demos". The address bar shows the URL "jbmoelker.github.io/reactnl-demos/". The main content area features a large heading "ReactNL Demos" and a bulleted list of links:

- Simple Component
- Stateful Component
- To Do App
- Component Using External Plugins

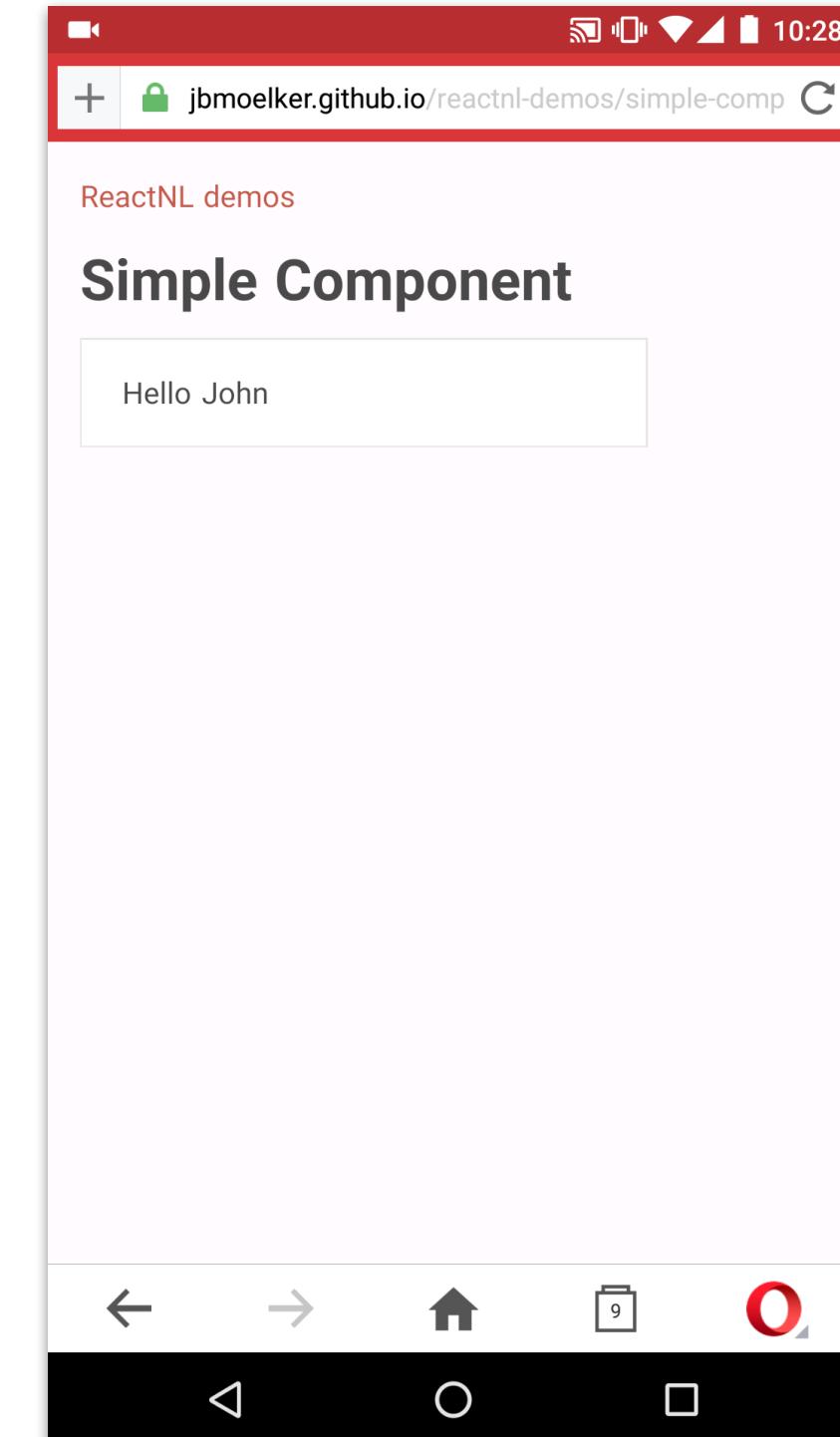
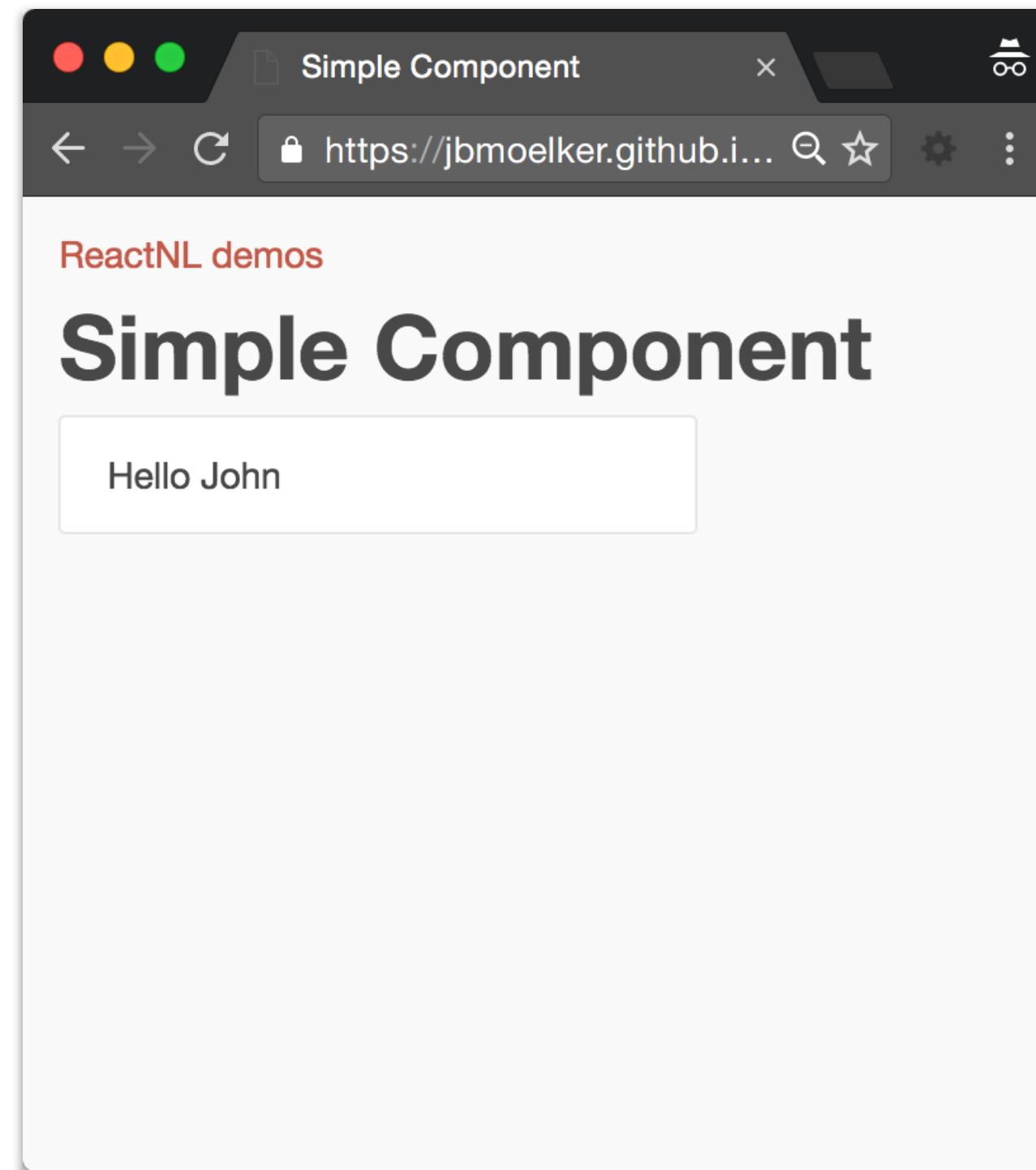
Below the list, a note states: "Based on examples on [facebook.github.io/react/](https://facebook.github.io/react/)".

[jbmoelker.github.io/reactnl-demos](https://jbmoelker.github.io/reactnl-demos)

# EXAMPLE: SIMPLE



A screenshot of the Lynx web browser running in a terminal window. The title bar says "jasper - lynx - 60x25". The main content area displays the text "Simple Component" and "ReactNL demos". At the bottom, there is a blue status bar with the text "Commands: Use arrow keys to move, '?' for help, 'q' to quit" and "Arrow keys: Up and Down to move. Right to follow a link; H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [del]".

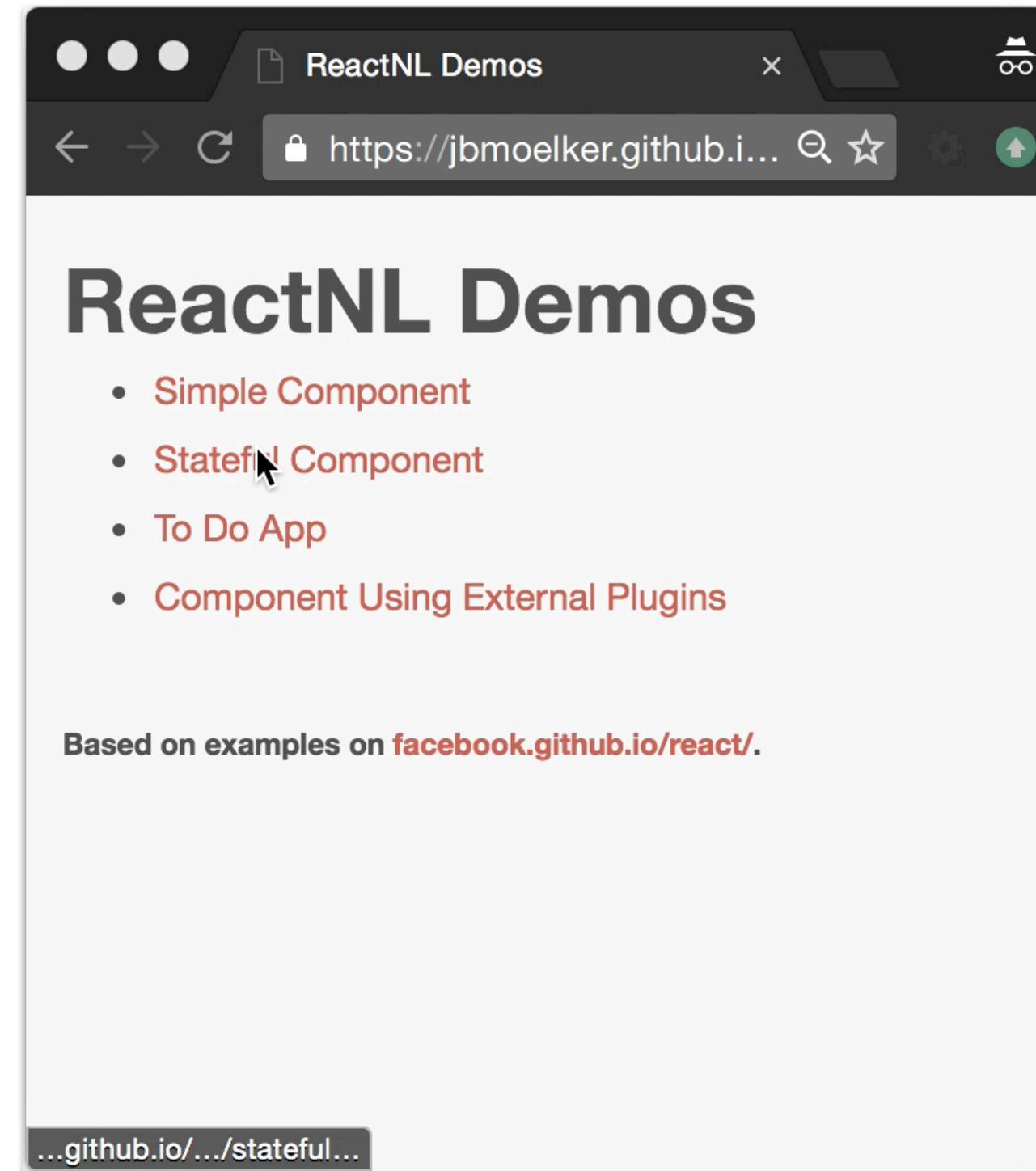


[jbmoelker.github.io/reactnl-demos/simple-component](https://jbmoelker.github.io/reactnl-demos/simple-component)

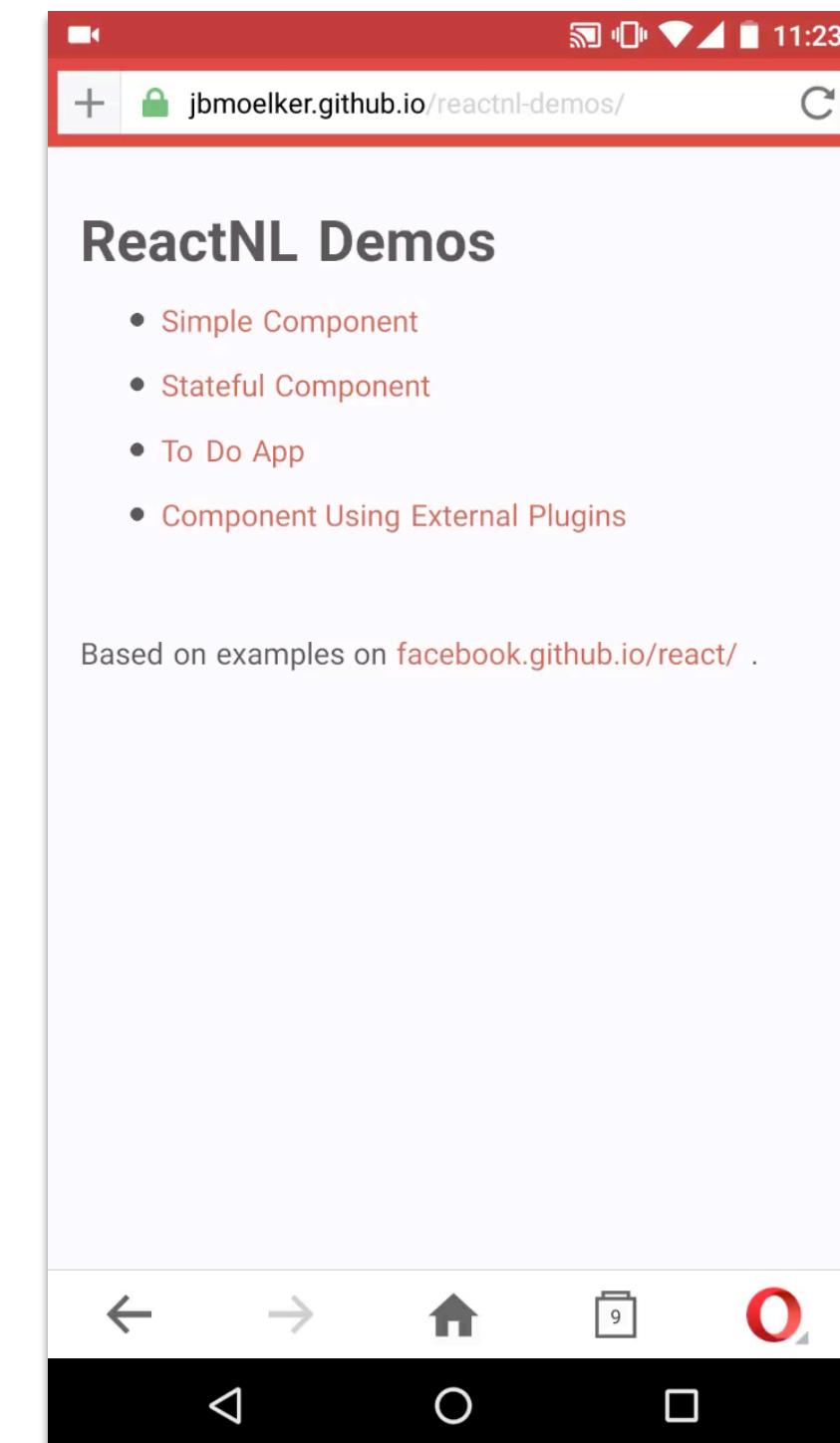
# EXAMPLE: STATEFUL



A screenshot of a Lynx terminal window. The title bar says "jasper - lynx - 60x25". The main area displays a "Stateful Component" page from "ReactNL demos". The page content includes the text "Stateful Component" and a bulleted list of components: "Simple Component", "Stateful Component", "To Do App", and "Component Using External Plugins". At the bottom, there's a help message: "Commands: Use arrow keys to move, '?' for help, 'q' to quit" and "Arrow keys: Up and Down to move. Right to follow a link; H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [del]".



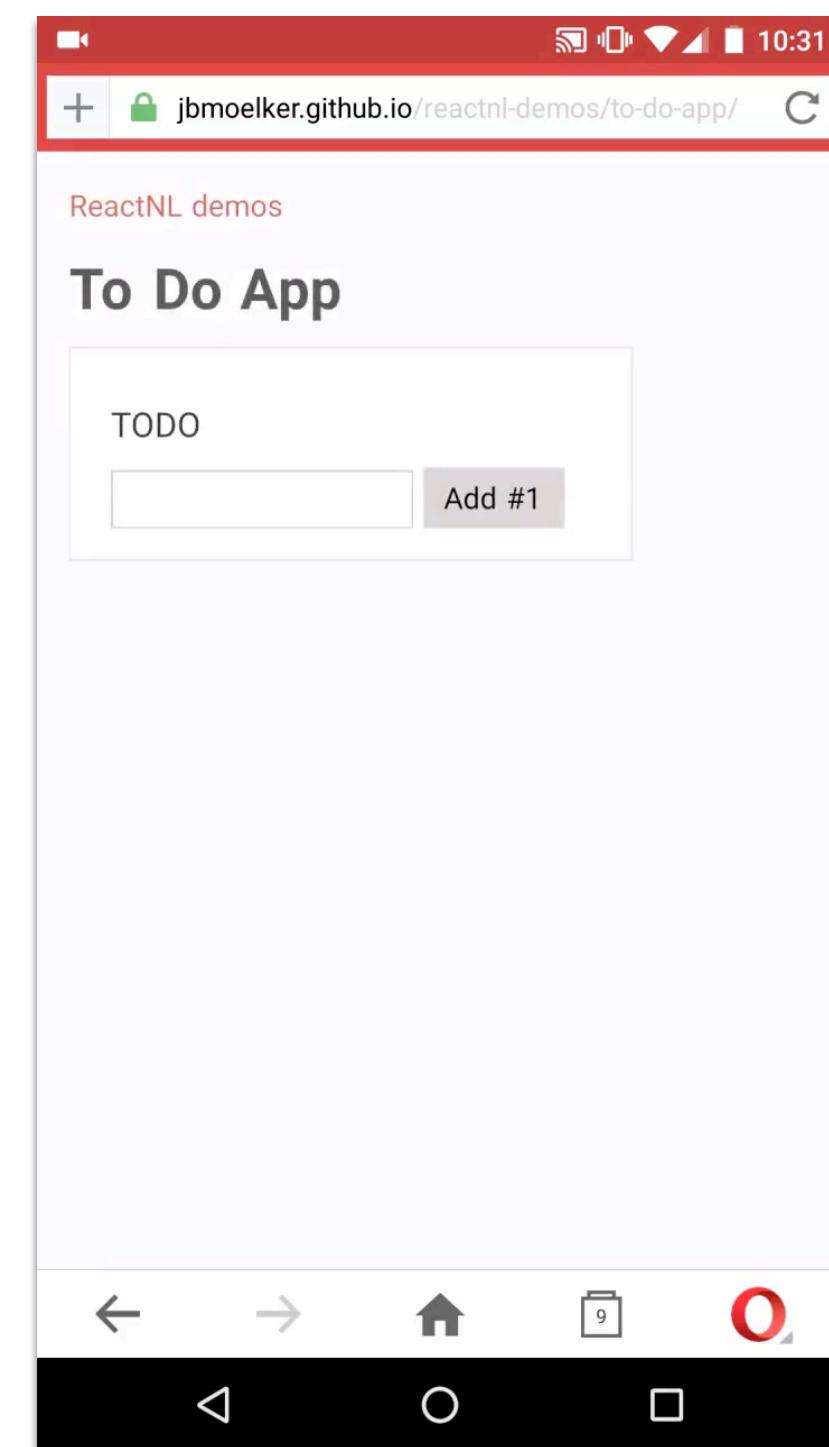
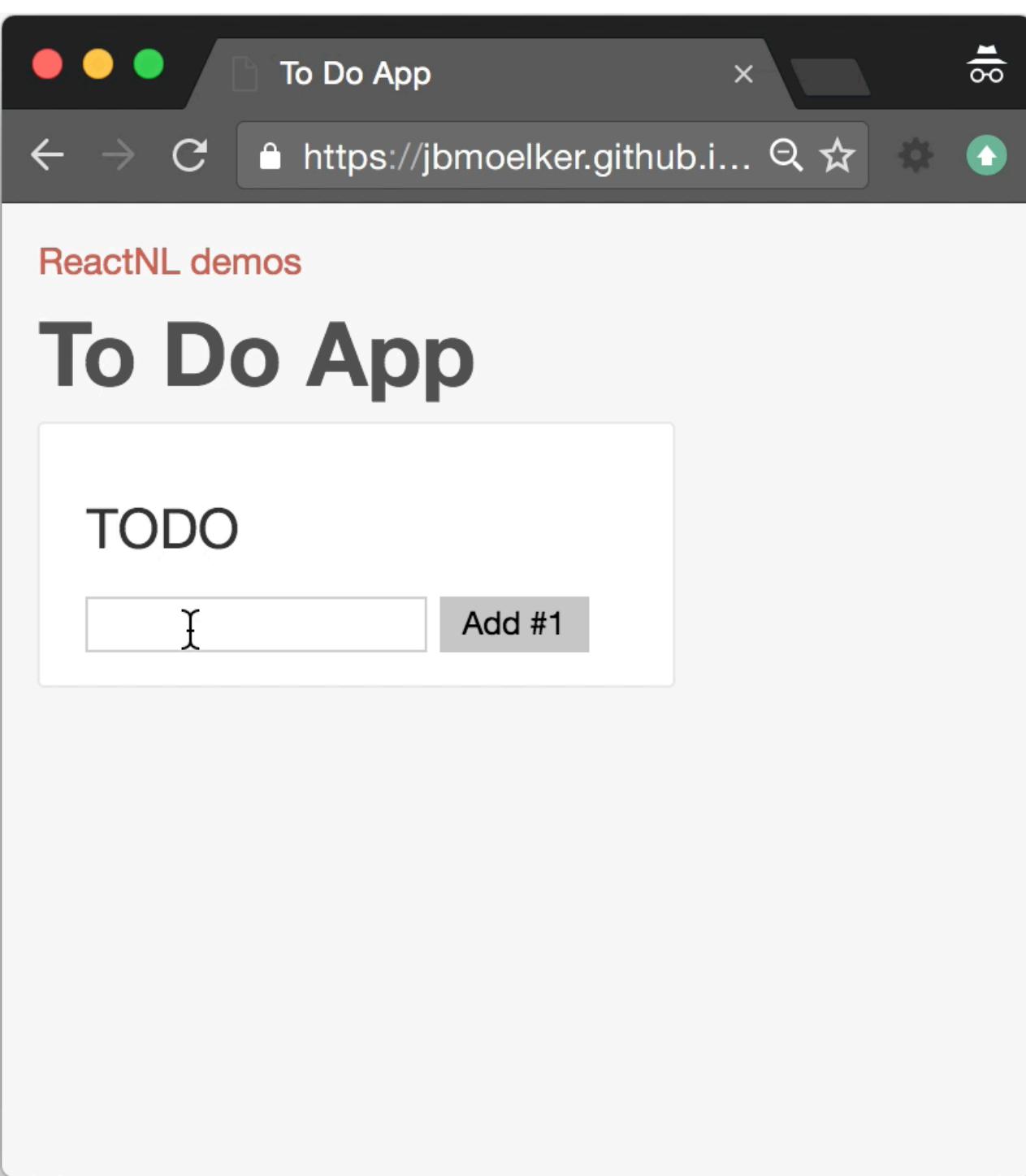
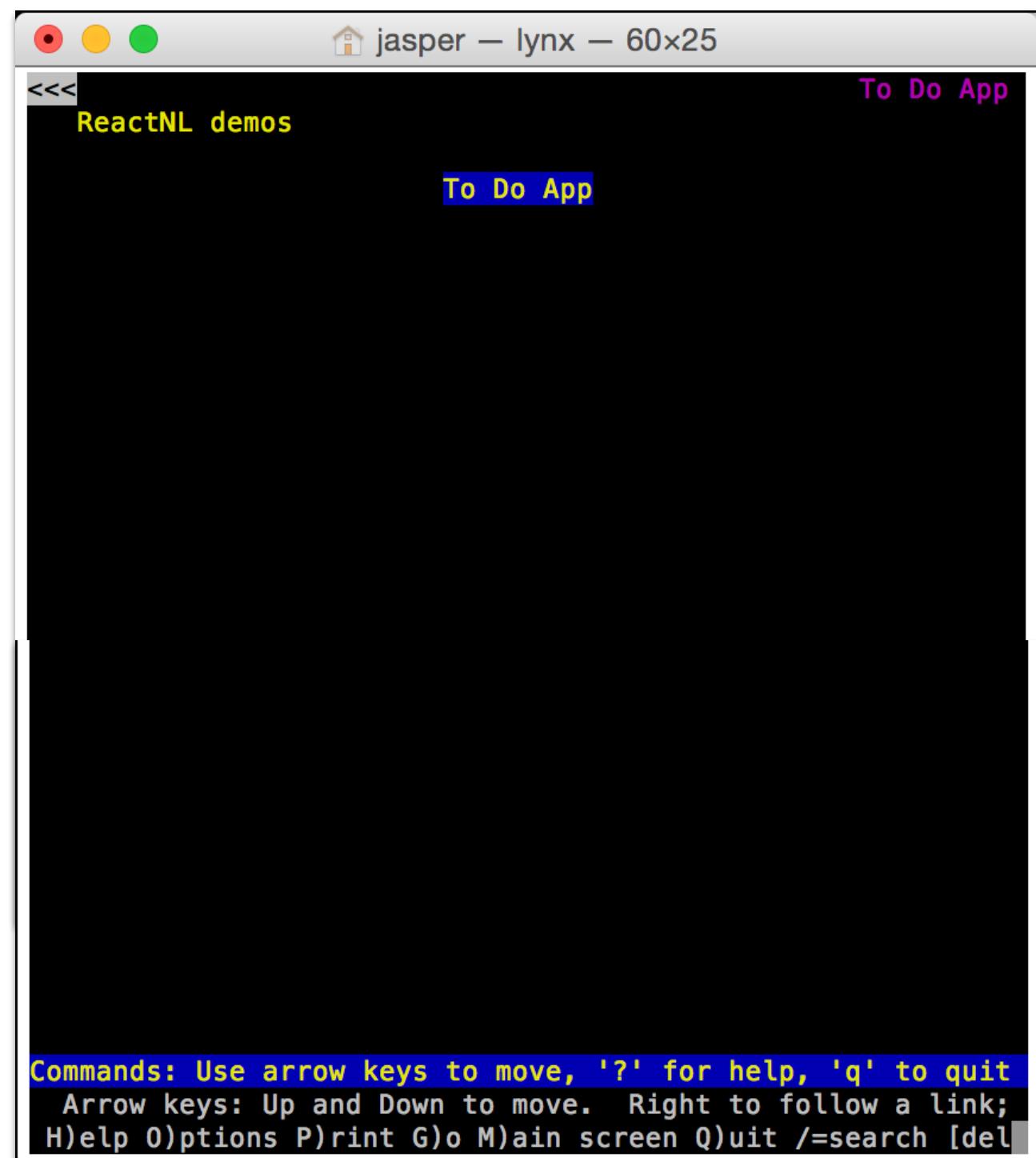
A screenshot of a Google Chrome browser window. The title bar says "ReactNL Demos". The address bar shows the URL "https://jbmoelker.github.io/reactnl-demos/". The main content area displays the "ReactNL Demos" page with the same list of components as the Lynx terminal. A cursor is hovering over the "Stateful Component" link. At the bottom of the page, it says "Based on examples on [facebook.github.io/react/](https://facebook.github.io/react/)".



A screenshot of a QMobile Mini browser window. The title bar says "ReactNL Demos". The address bar shows the URL "jbmoelker.github.io/reactnl-demos/". The main content area displays the "ReactNL Demos" page with the same list of components. A cursor is hovering over the "Stateful Component" link. At the bottom, it says "Based on examples on [facebook.github.io/react/](https://facebook.github.io/react/)".

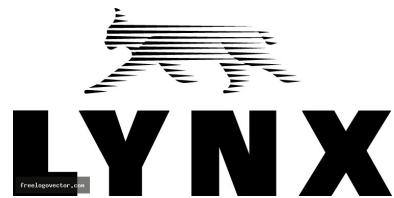
[jbmoelker.github.io/reactnl-demos/stateful-component](https://jbmoelker.github.io/reactnl-demos/stateful-component)

# EXAMPLE: TO DO APP



[jbmoelker.github.io/reactnl-demos/to-do-app](https://jbmoelker.github.io/reactnl-demos/to-do-app/)

# EXAMPLE: USING PLUGINS



A screenshot of the Lynx browser window. The title bar reads "jasper — lynx — 60x25". The main content area shows a terminal-like interface with the text "ReactNL demos" and "Using External Plugins". At the bottom, there is a blue status bar with the text: "Commands: Use arrow keys to move, '?' for help, 'q' to quit", "Arrow keys: Up and Down to move. Right to follow a link; H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [del]".



A screenshot of a Google Chrome browser window. The title bar says "Using External Plugins". The address bar shows the URL "https://jbmoelker.github.io/reactnl-demos/using-plugins". The main content area displays the "Using External Plugins" demo page from ReactNL. It includes two sections: "Input" (with a text input field containing "Type some \*markdown\* here!") and "Output" (with a text area containing "Type some markdown here!").



A screenshot of a Q MINI mobile browser window. The title bar shows the URL "jbmoelker.github.io/reactnl-demos/using-plugins". The main content area displays the "Using External Plugins" demo page, identical to the one in the Chrome window, with "Input" and "Output" sections.

[jbmoelker.github.io/reactnl-demos/using-plugins](https://jbmoelker.github.io/reactnl-demos/using-plugins)

# ENHANCE WITH HTML



@jbmoelker

**HTML is a standard**  
supported by browser vendors,  
assistive technologies etc.



# BUILT-IN SEMANTICS

**<article>** <h1> ... </h1> <p> ...

**<nav>** <ol> <li> ...

**<form>** <fieldset> <legend> ...

**<div>** <span>

# BUILT-IN INTERACTIVITY

**<a href="...">**

**<form action="..." method="...">**

**<input name="..." value="..." required>**

**<button>**

# BUILT-IN STATE

<form> values

:focus

:target

:disabled

:valid

# BUILT-IN RELATIONSHIPS

<button> **submits** <form>

<h1-6> **indexes** <section>

<summary> **toggles** <details>

<label> **describes** <input>

<output> **combines** <input>s

# STRUCTURED DATA

[**itemscope**]

[**itemtype**=“[http://schema.org/...](http://schema.org/)”]

[**itemprop**=“...”]

<meta property=“**og:**...” content=“...”>

# ENHANCED ACCESSIBILITY

[aria-**role**= "alert"]

[aria-**controls**= "..."]

[aria-**expanded**]

[aria-**label**= "..."]

# HTML BEFORE JS



@jbmoelker

React is JavaScript first,  
but most user agents are not



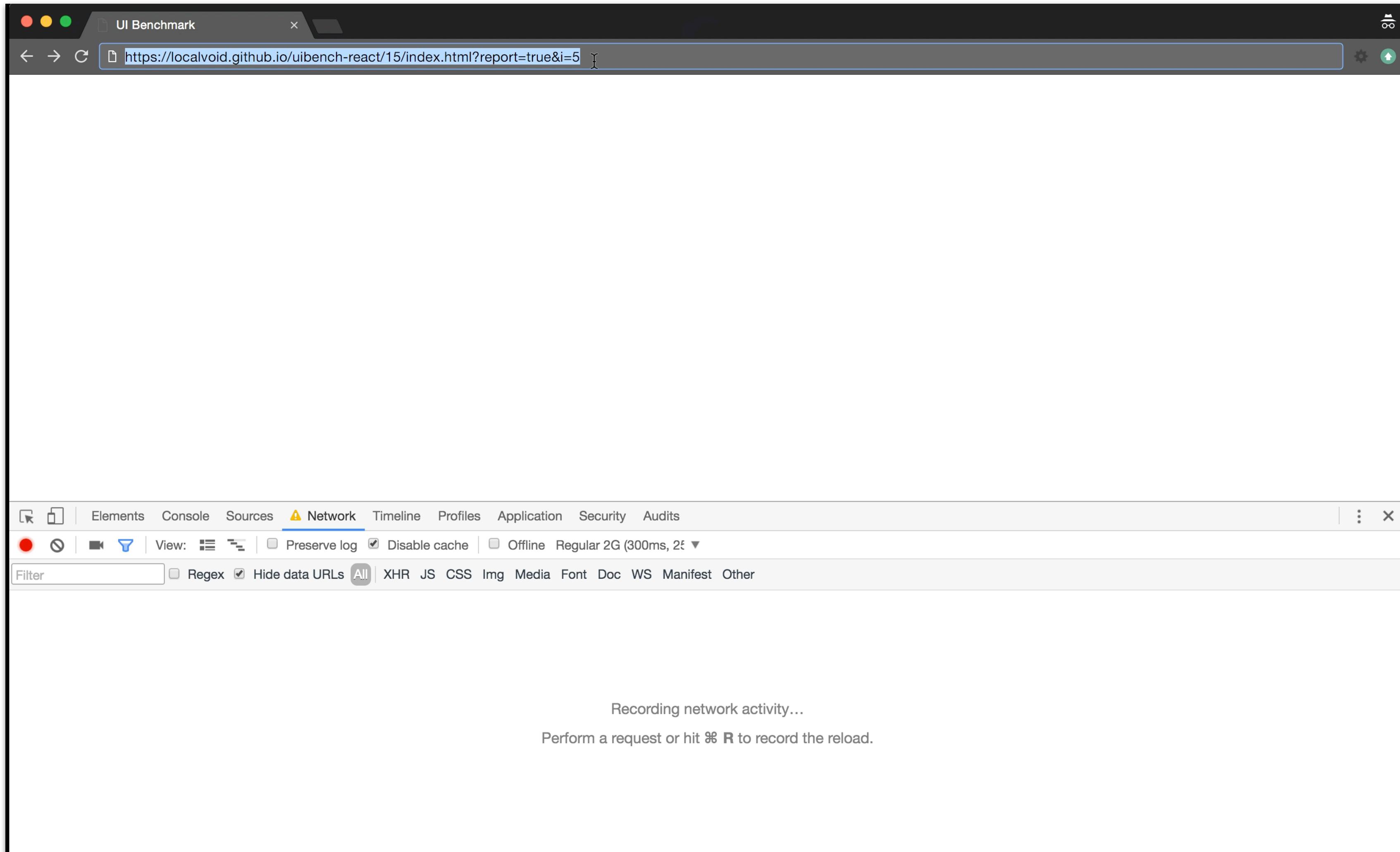
@jbmoelker

# REACT IS LIGHTNING FAST ...

The screenshot shows a web browser window titled "UI Benchmark" at <https://localvoid.github.io/uibench/>. The page contains the following content:

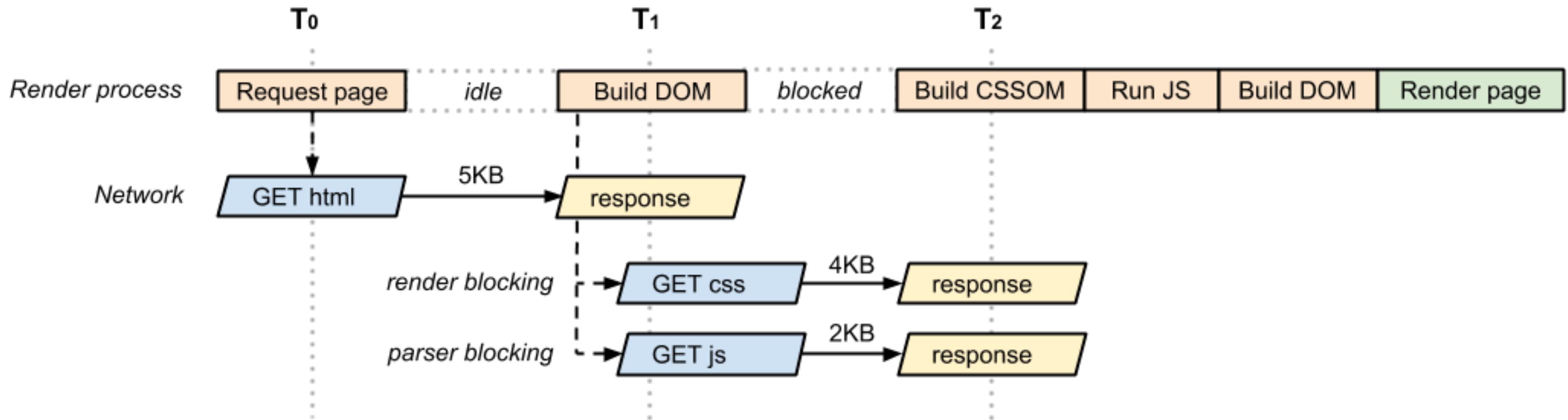
- UI Benchmark** heading.
- Text: "To start benchmarking, click on a button below library name that you want to test, it will open a new window, perform tests and send results back to the main window, results will be displayed at the bottom section 'Results'."
- Text: "In the 'Results' section there will be different test cases, for example test case `table/[100,4]/render` represents update from empty table to table with 100 rows and 4 columns. Test case `table/[100,4]/filter/32` is an update from table with 100 rows and 4 columns to the same table where each 32th item is removed. Details about all test cases can be found inside the `uibench.js` file."
- Buttons: "Star" and "47".
- Form fields:
  - Checkboxes:
    - Enable full render time measurements (recalc style/layout/paint/composition/etc)
    - Disable `shouldComponentUpdate` optimization
    - Mobile mode (reduces number of DOM elements in tests)
  - Text input: "Iterations" with value "5".
  - Text input: "Tests filter" with placeholder "For example: render".
- Section: "React"
  - Text: "Virtual DOM. Compiled with: es2015-loose, transform-react-inline-elements."
  - Buttons: "14" and "15" (with a cursor icon over "15").

# ... AFTER REACT IS LOADED



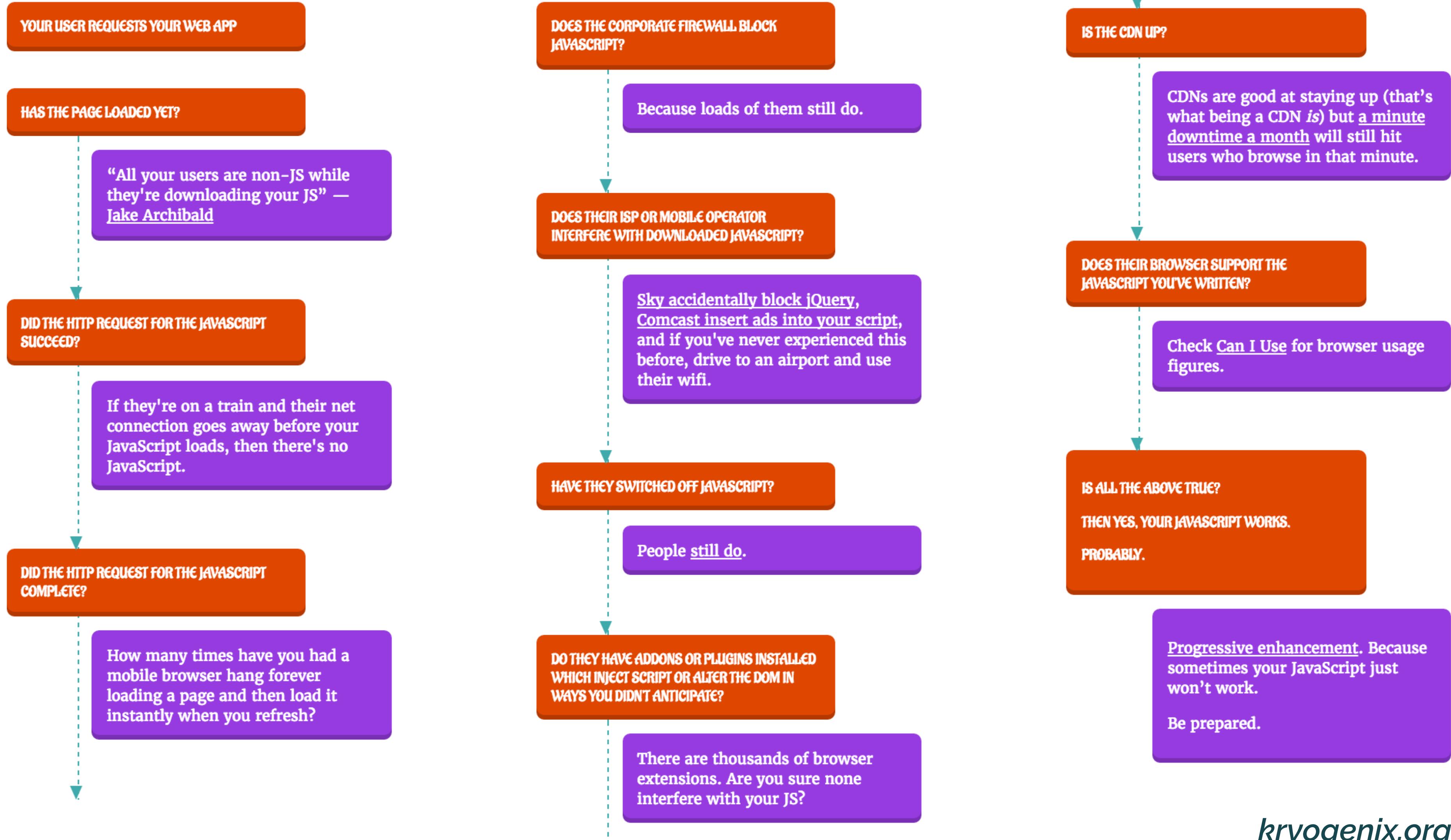
[localvoid.github.io/uibench](https://localvoid.github.io/uibench)

# CRITICAL RENDERING PATH

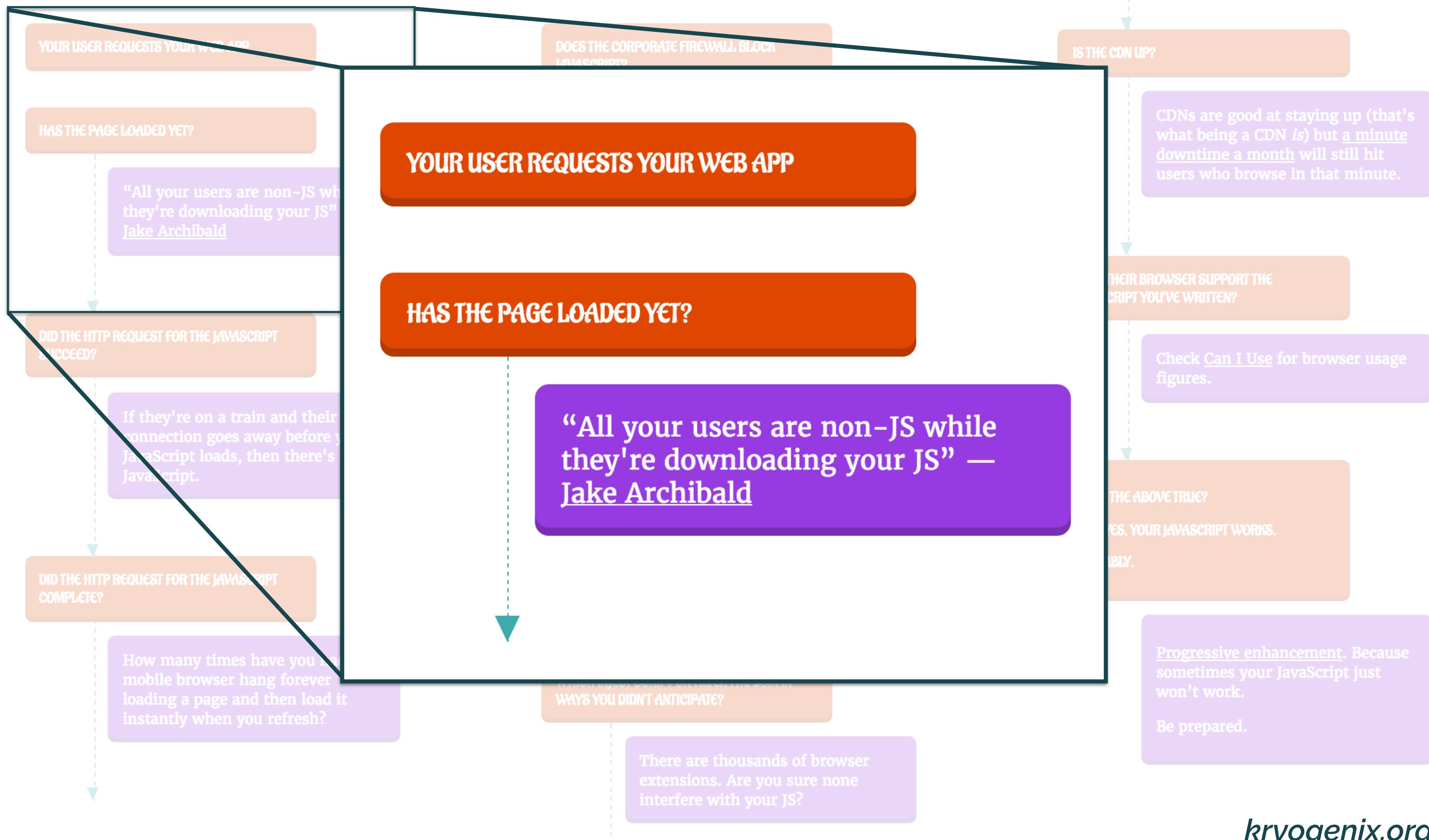


[critical rendering path on developers.google.com](https://developers.google.com/web/fundamentals/performance/critical-rendering-path/)

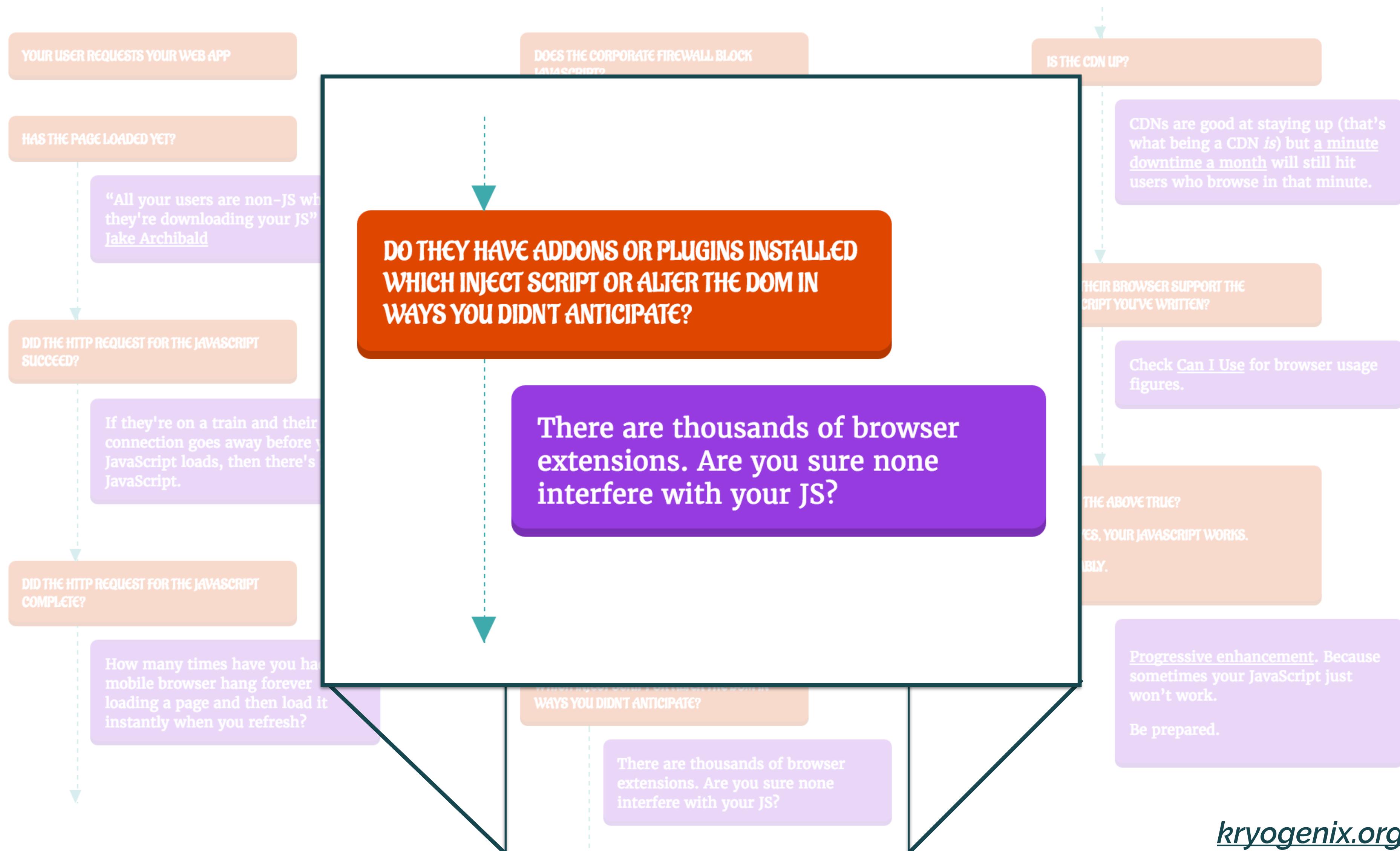
# EVERYONE HAS JS, RIGHT?



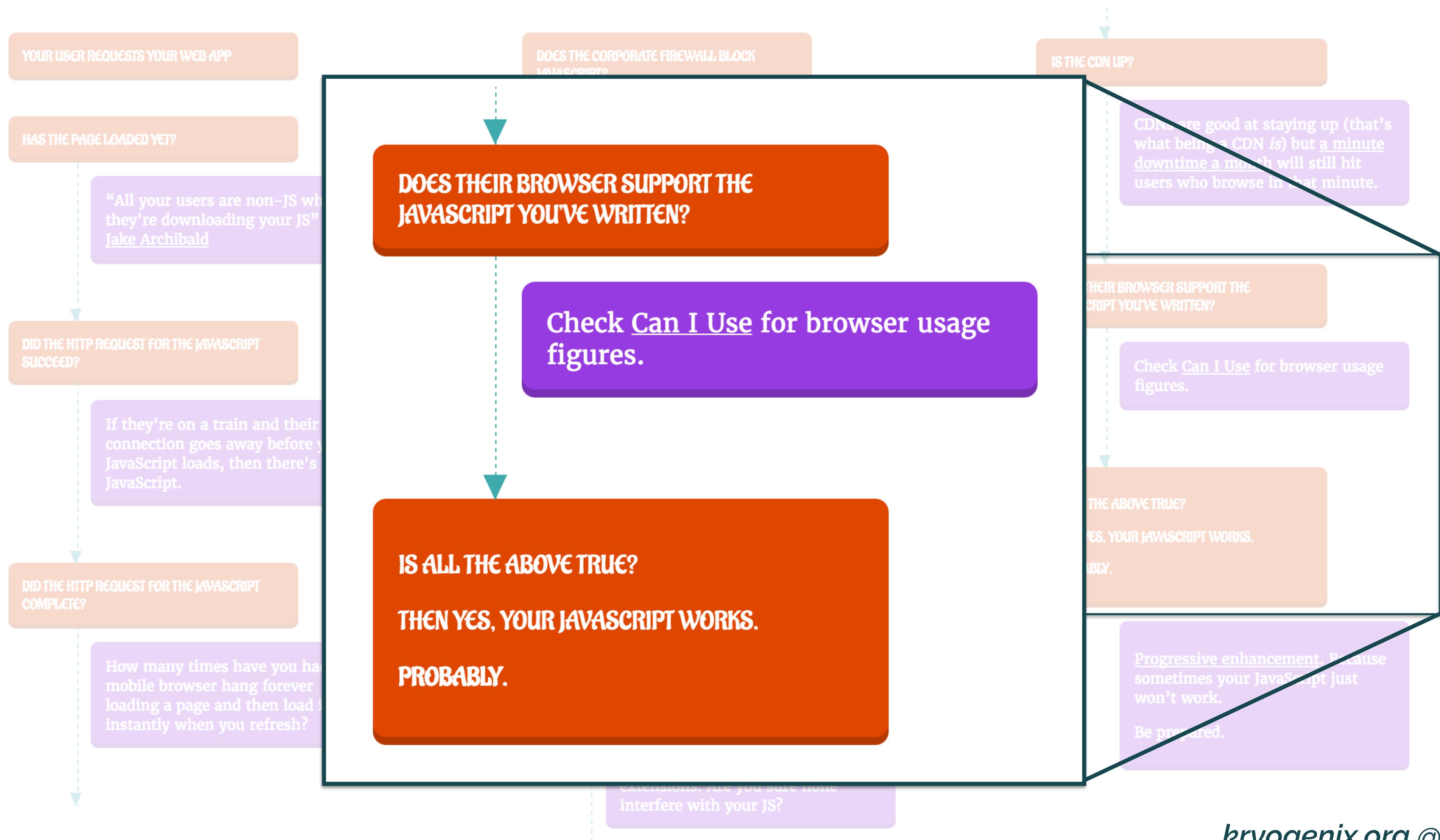
# EVERYONE HAS JS, RIGHT?



# EVERYONE HAS JS, RIGHT?



# EVERYONE HAS JS, RIGHT?



whatever language or  
(JS) framework you use

**start by serving HTML**



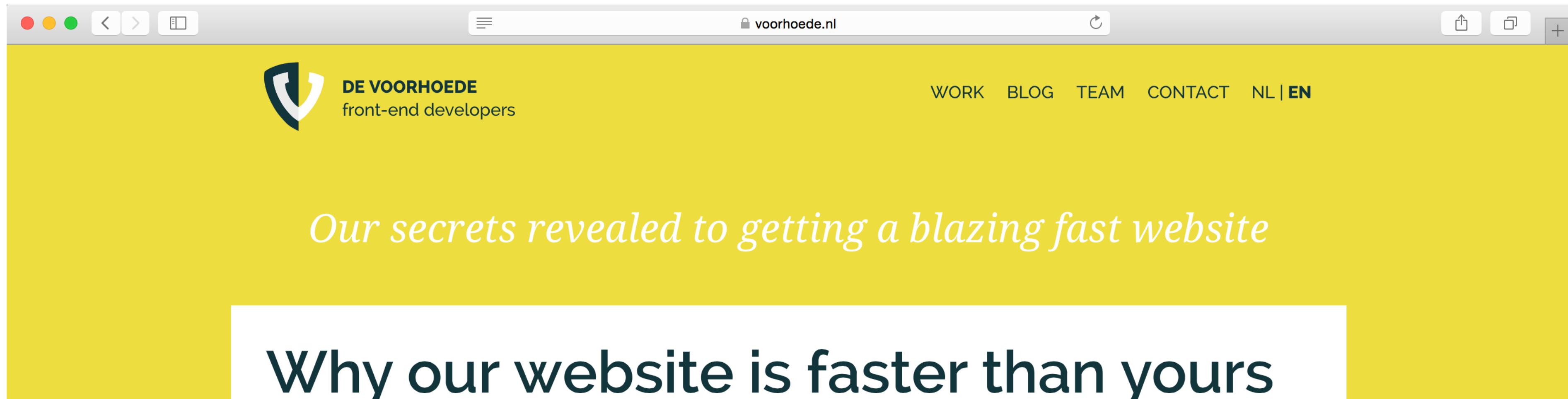
@jbmoelker

# ENHANCING REACT APPS



@jbmoelker

# OPTIMISE FOR CRITICAL PATH



The screenshot shows a Mac OS X browser window displaying the website voorhoede.nl. The page has a yellow header bar with the site's logo and name 'DE VOORHOEDE front-end developers'. Below the header, a large white text area contains the tagline 'Our secrets revealed to getting a blazing fast website'. A prominent white rectangular box in the center contains the title 'Why our website is faster than yours'. To the left of this box is a circular profile picture of a man with a beard, identified as 'by Declan July 26, 2016'. On the right side of the main content area, there is a vertical sidebar containing several optimization tips listed as links.



by Declan  
July 26, 2016

## Table of contents

[Design for performance](#)

[Content first](#)

[Take control](#)

[Image delivery](#)

[SVG animations](#)

[Custom web fonts](#)

We recently updated our site. Yes, it has a complete design overhaul, but as real software developers we focused a lot on the technical bits and pieces as well. Our goal was to take control, focus on performance, be flexible for the future and make it fun to write content for the site. Here's how we made our website faster than yours (Yup, sorry!)

## Design for performance

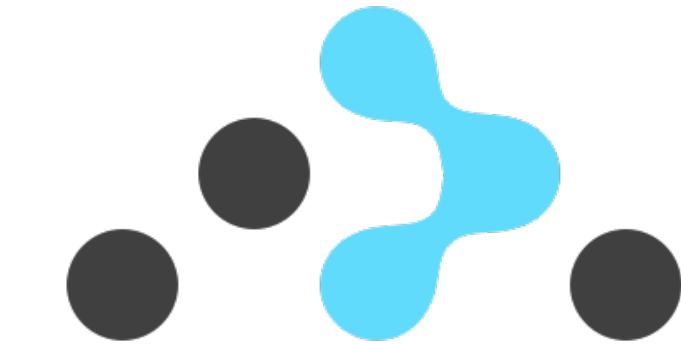
In our projects we have daily discussions with designers and product owners about balancing aesthetics and performance. For our own site, this was easy. Simply said: we believe that a good user experience starts with delivering content as fast as possible. That means **performance > aesthetics**.

Good content, layout, images, and interactivity are essential for engaging your audience, but each of these elements have impact on page load time and the end-user experience. In every step we looked at how we could get a nice user experience and design while having minimum impact on

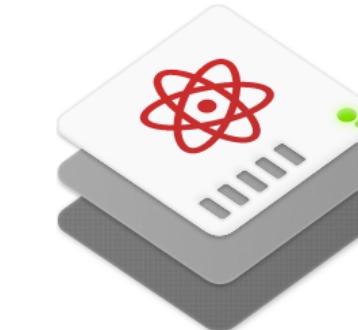
[post on voorhoede.nl](#)

# ISOMORPHIC

"a seamless transition  
from server-rendered HTML  
to client-rendered DOM  
without losing state"



**REACT ROUTER**



**React Server**

React Starter Kit

**React Isomorphic Form**

# SMALLER MOUNT NODES

```
<body>  
<div id="megaApp">  
</div>  
</body>  
  
ReactDOM.render(  
  <App />,  
  getById('megaApp')  
)
```



```
<body>  
<h1>Static title</h1>  
<div id="miniApp1">  
</div>  
static content  
<div id="miniApp2">  
</div>  
more static content  
</body>
```

# USEFUL INITIAL CONTENT

```
<div id="map" data-address="Sieraad, Amsterdam">  
</div>
```

```
const map = document.getElementById('map');  
const address = map.dataset.address;  
ReactDOM.render(  
  <Map address={ address } />,  
  map  
)
```

# USEFUL INITIAL CONTENT

```
<div id="map" data-address="Sieraad, Amsterdam">  
</div>
```



```
<div id="map" data-address="Sieraad, Amsterdam">  
<a href="https://google.com/maps/place/Sieraad,Amsterdam">  
    
</a></div>
```

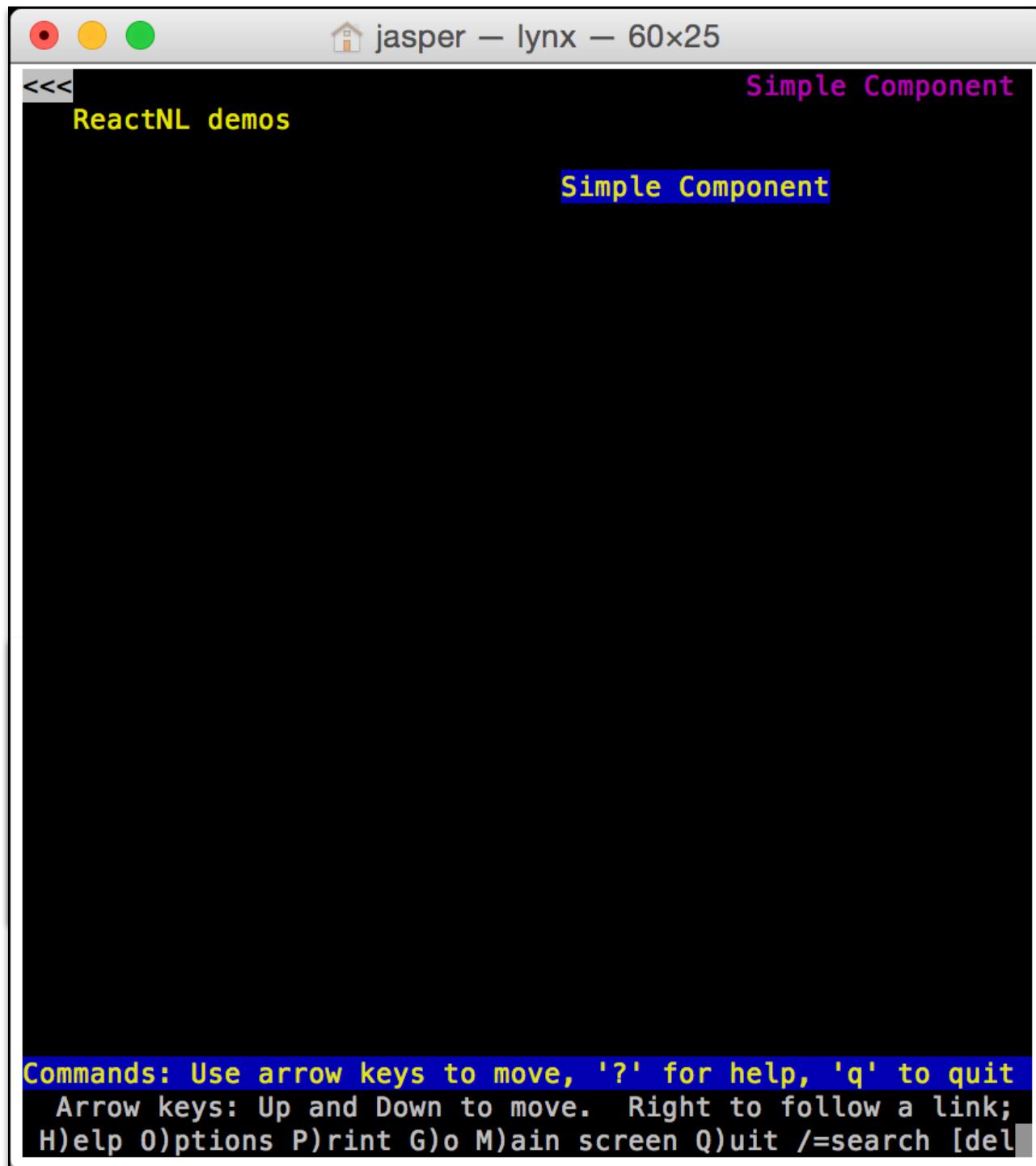
# ENHANCED EXAMPLES



@jbmoelker

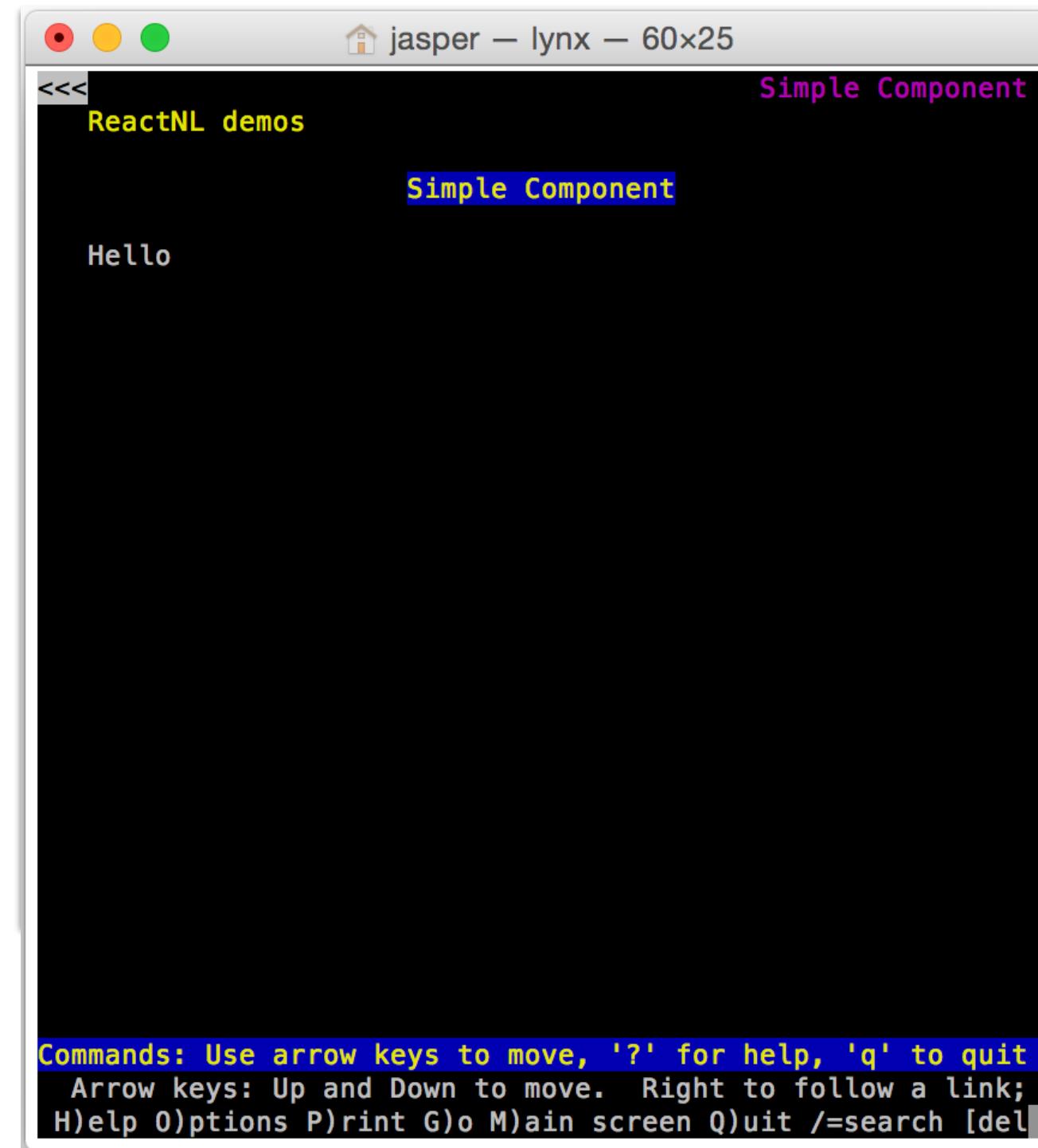
# EXAMPLE: SIMPLE

 **LYNX** original



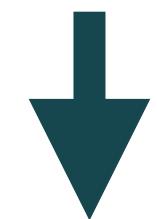
A screenshot of the Lynx web browser window. The title bar says "jasper - lynx - 60x25". The main content area displays a "Simple Component" page with the text "ReactNL demos" and "Simple Component". At the bottom, there is a status bar with the text "Commands: Use arrow keys to move, '?' for help, 'q' to quit" and "Arrow keys: Up and Down to move. Right to follow a link; H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [del]".

 **LYNX** enhanced



A screenshot of the Lynx web browser window. The title bar says "jasper - lynx - 60x25". The main content area displays a "Simple Component" page with the text "ReactNL demos" and "Simple Component". Below that, it shows the word "Hello". At the bottom, there is a status bar with the text "Commands: Use arrow keys to move, '?' for help, 'q' to quit" and "Arrow keys: Up and Down to move. Right to follow a link; H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [del]".

<div id="mount">  
</div>



<div id="mount">  
  <p>Hello</p>  
</div>

# EXAMPLE: SIMPLE



original



enhanced

A screenshot of a Google Chrome browser window titled "Simple Component". The address bar shows "127.0.0.1:7322/simple-com...". The page content area displays "ReactNL demos" and "Simple Component" in large bold letters. Below that is a button with the text "Hello John".

A screenshot of a Google Chrome browser window titled "Simple Component". The address bar shows "127.0.0.1:7322/simple-com...". The page content area displays "ReactNL demos" and "Simple Component\*" in large bold letters. Below that is a button with the text "Hello John". A tooltip "Reload this page" is visible above the title. At the bottom left of the page, there is a small note "\*enhanced".

```
<div id="mount">
```

```
</div>
```

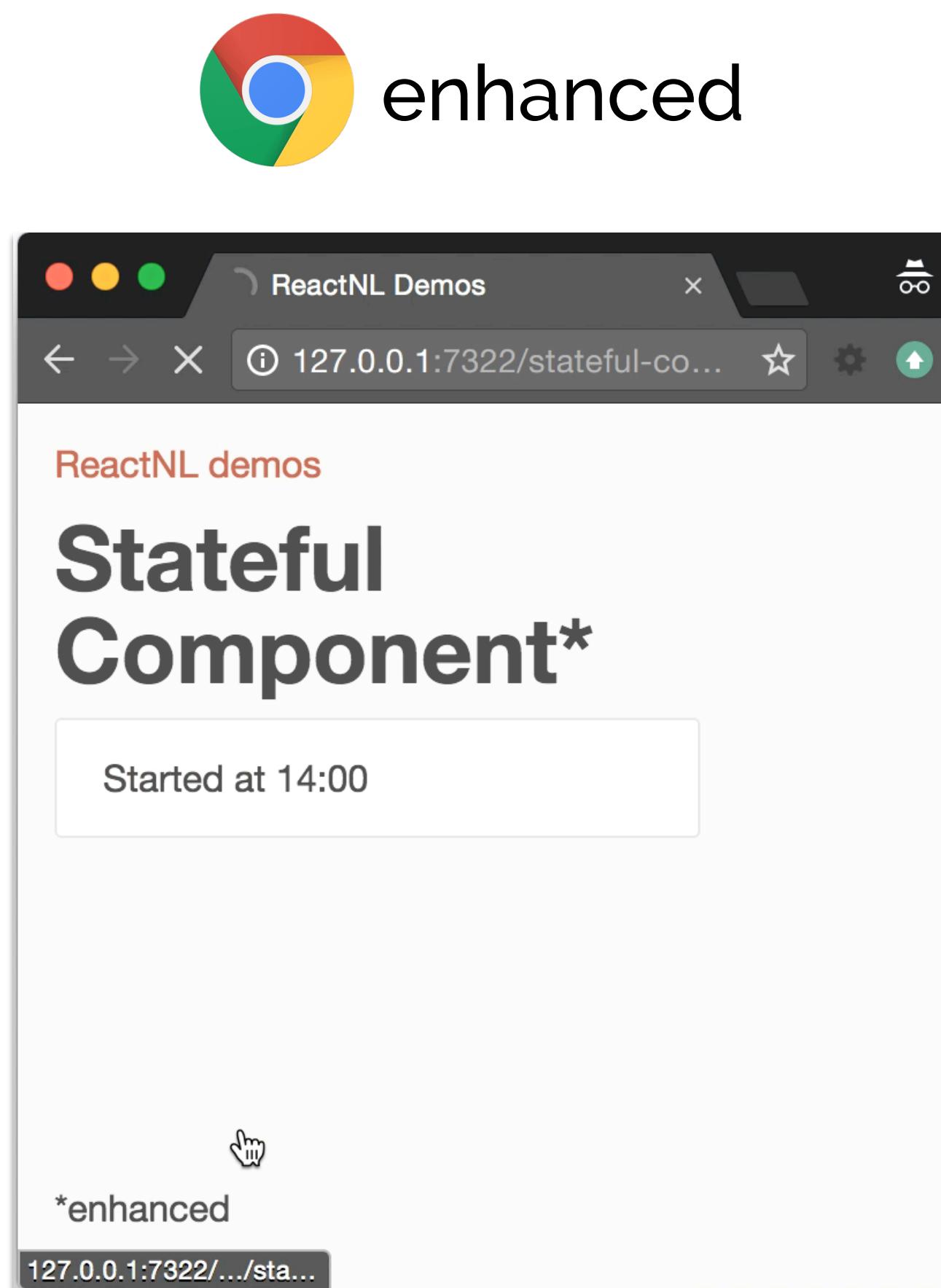
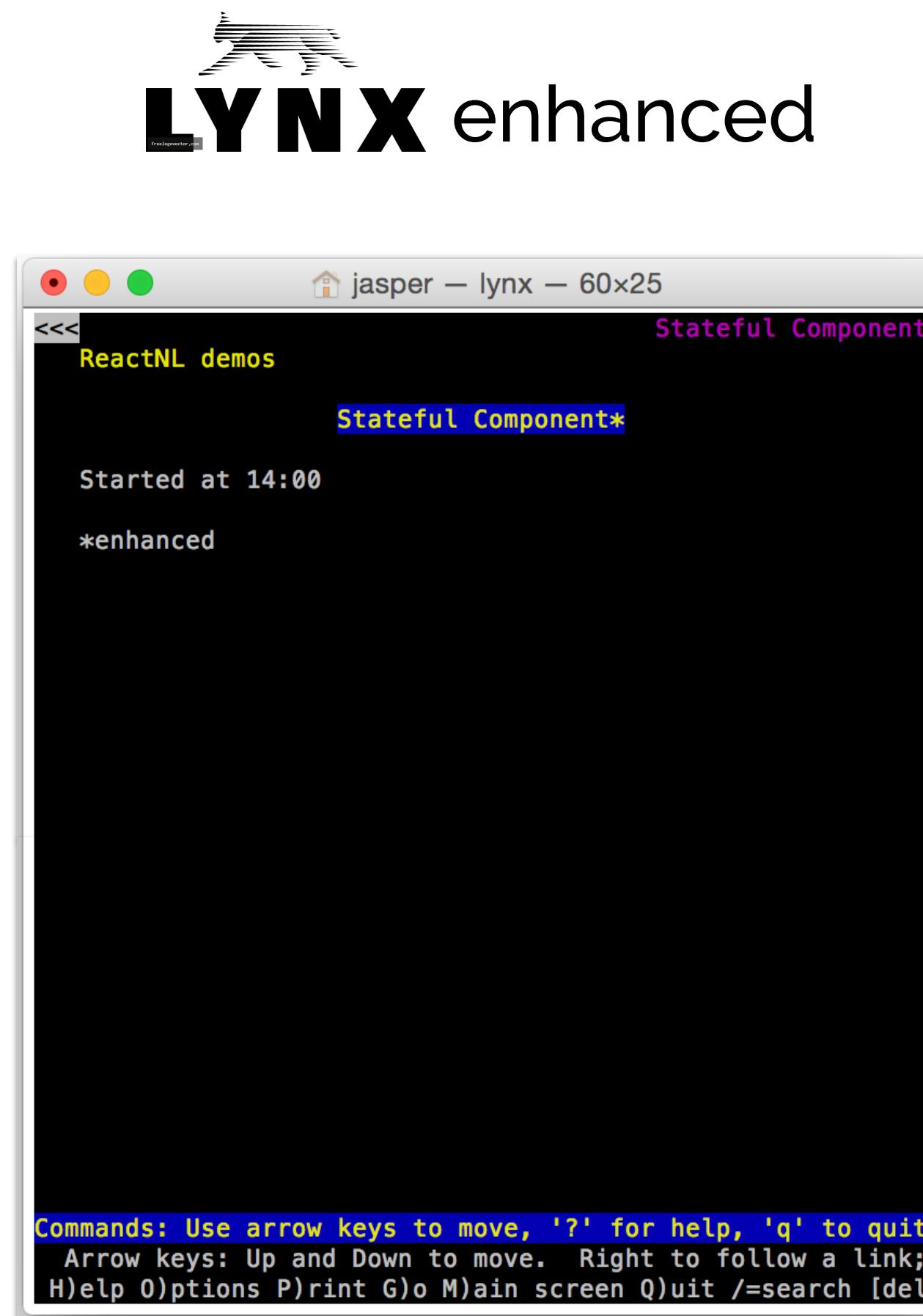


```
<div id="mount">
```

```
  <p>Hello</p>
```

```
</div>
```

# EXAMPLE: STATEFUL



```
<div id="mount">  
</div>  
↓  
<time id="mount" datetime="2016-10-13T14:00">  
Started at 14:00  
</time>
```

# EXAMPLE: TO DO APP



original



enhanced

To Do App

TODO

Add #1

ReactNL demos

Elements

```
<div data-reactroot>
  <h3>TODO</h3>
  <ul></ul>
  <form>
    <input value="" type="text" />
    <button>Add #1</button>
  </form>
```

To Do App\*

TODO

new todo

Add #1

ReactNL demos

Elements

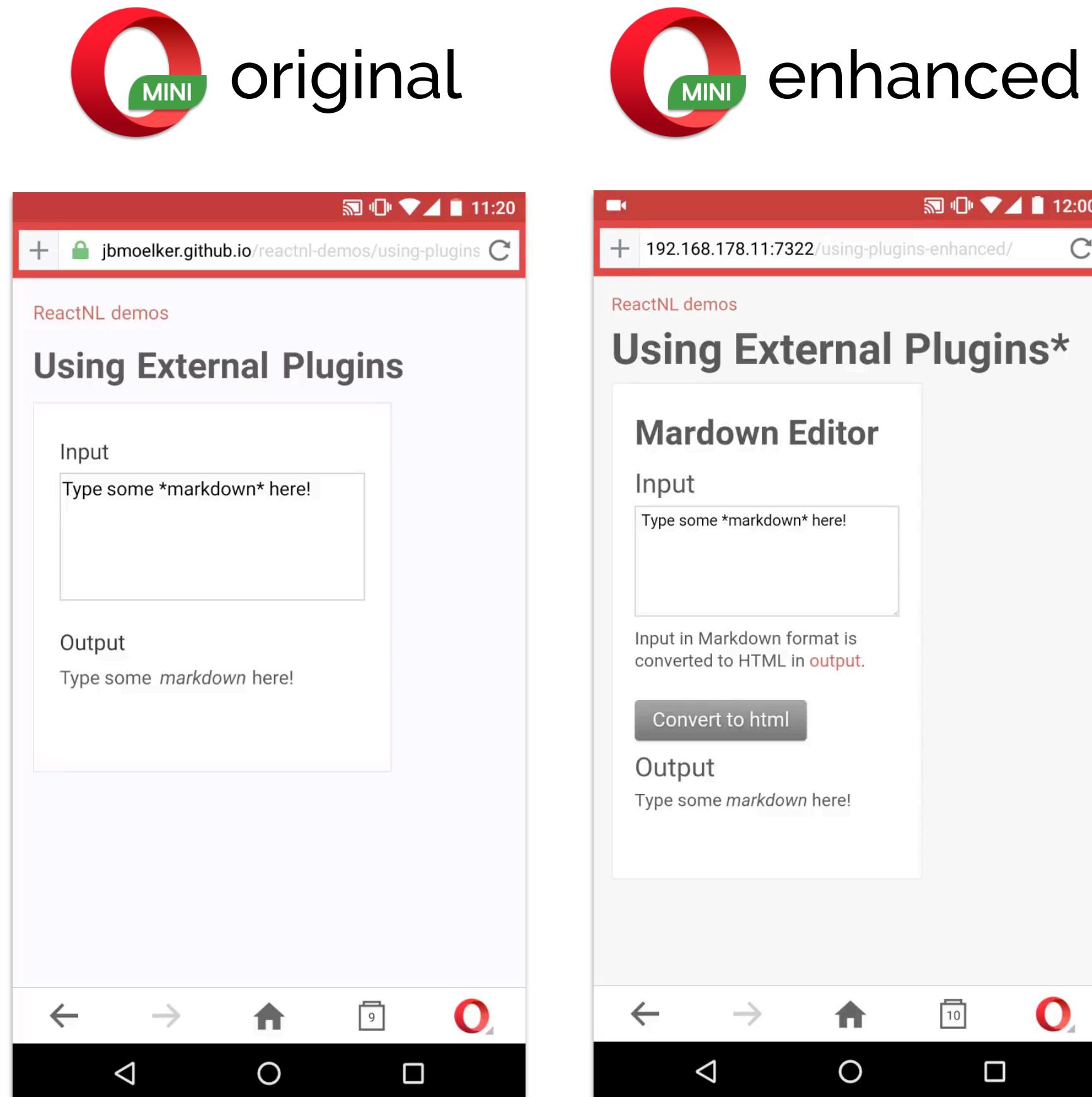
```
<form>
  <label for="todo">new todo</label>
  <input id="todo" value="" type="text" />
  <button>Add #1</button>
</form>
```

`<input id="todo">`



`<label for="todo">`  
new todo  
`</label>`  
`<input id="todo">`

# EXAMPLE: USING PLUGINS



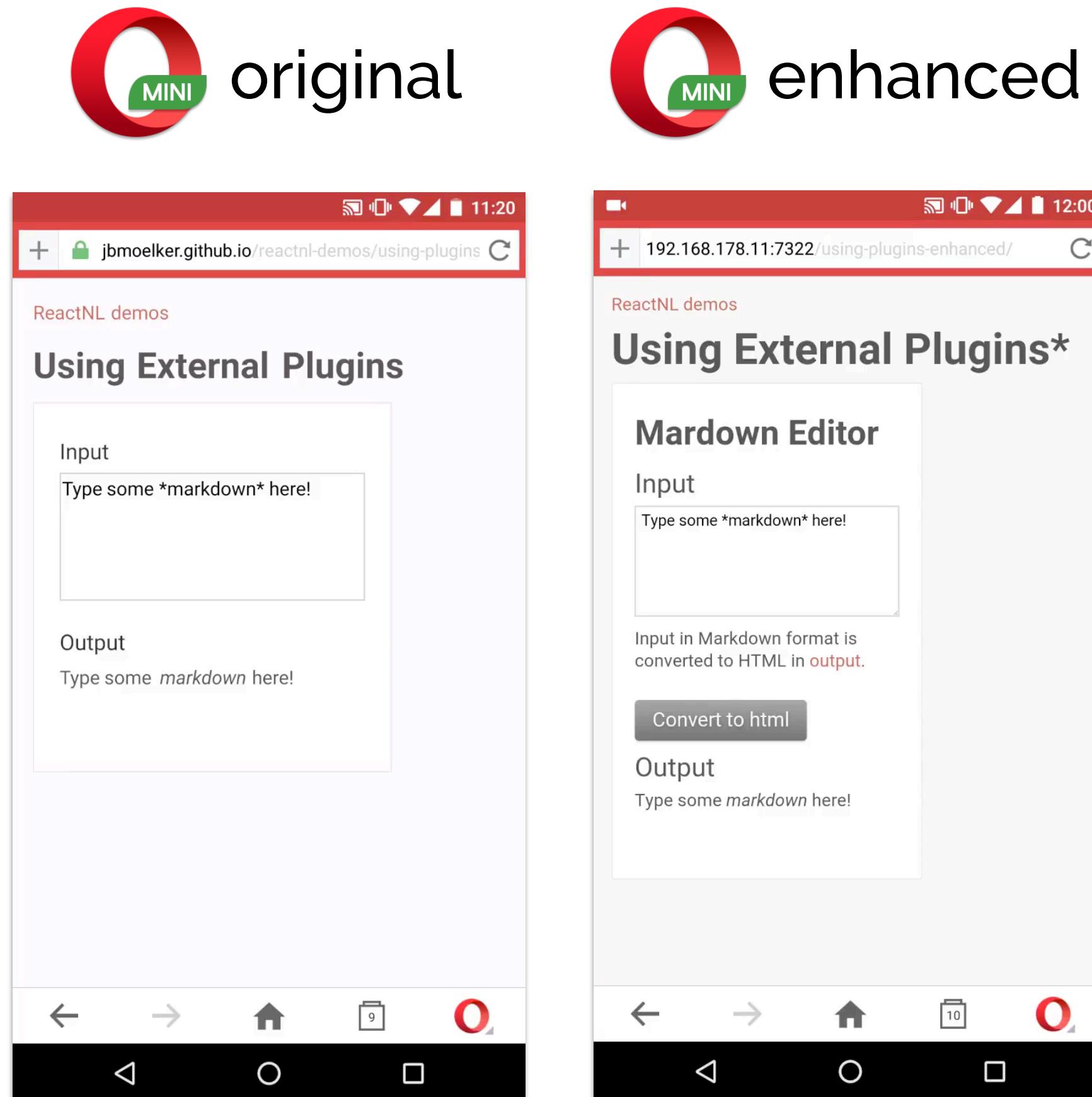
`<div class="MarkdownEditor">`



`<form method="..." action="..."  
class="MarkdownEditor">  
<h2>Markdown Editor</h2>  
...  
<button>Convert</button>`

[jbmoelker.github.io/reactnl-demos/using-plugins-enhanced](https://jbmoelker.github.io/reactnl-demos/using-plugins-enhanced)

# EXAMPLE: USING PLUGINS



<h3>Input</h3>

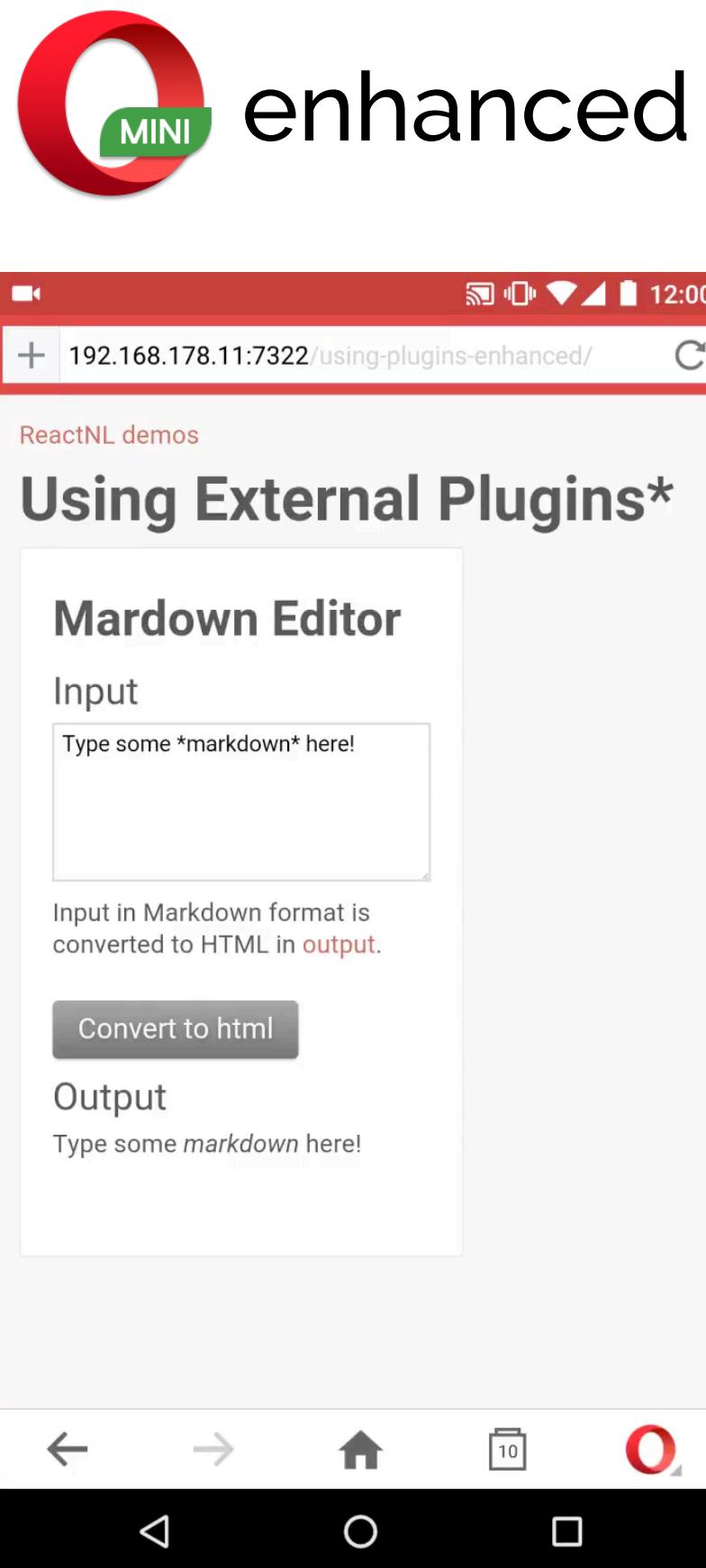
<textarea ...>...</textarea>

↓

<label for="input" class="h3">  
Input</label>

<textarea id="input" ...>  
...</textarea>

# EXAMPLE: USING PLUGINS



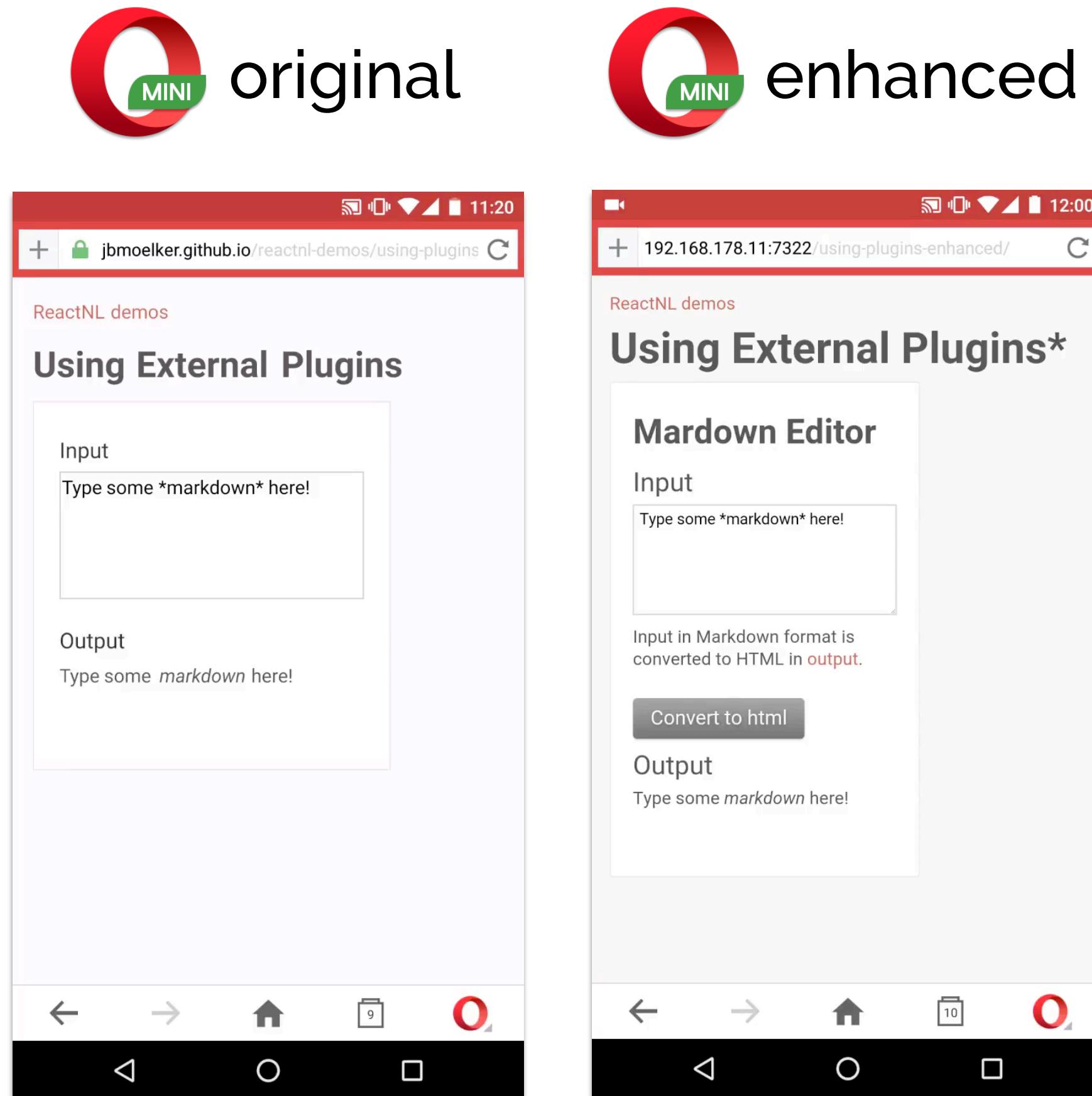
`<h3>Output</h3>`  
`<div> ... </div>`



`<label for="output"`  
`class="h3">Output</label>`  
`<output id="output"`  
`tabindex="-1"`  
`for="input"> ... </output>`

<jbmoelker.github.io/reactnl-demos/using-plugins-enhanced>

# EXAMPLE: USING PLUGINS



```
<textarea id="input"
aria-describedby="tooltip">
</textarea>
<p role="tooltip" id="tooltip">
  Input in Markdown format
  is converted to HTML in
  <a href="#output">output
</a></p>
```

[jbmoelker.github.io/reactnl-demos/using-plugins-enhanced](https://jbmoelker.github.io/reactnl-demos/using-plugins-enhanced)

# IN SUMMARY



@jbmoelker

think about **users & browsers**

serve useful **initial HTML**

use HTML's **built-in features**



# THANK YOU!



@jbmoelker

questions?



@jbmoelker