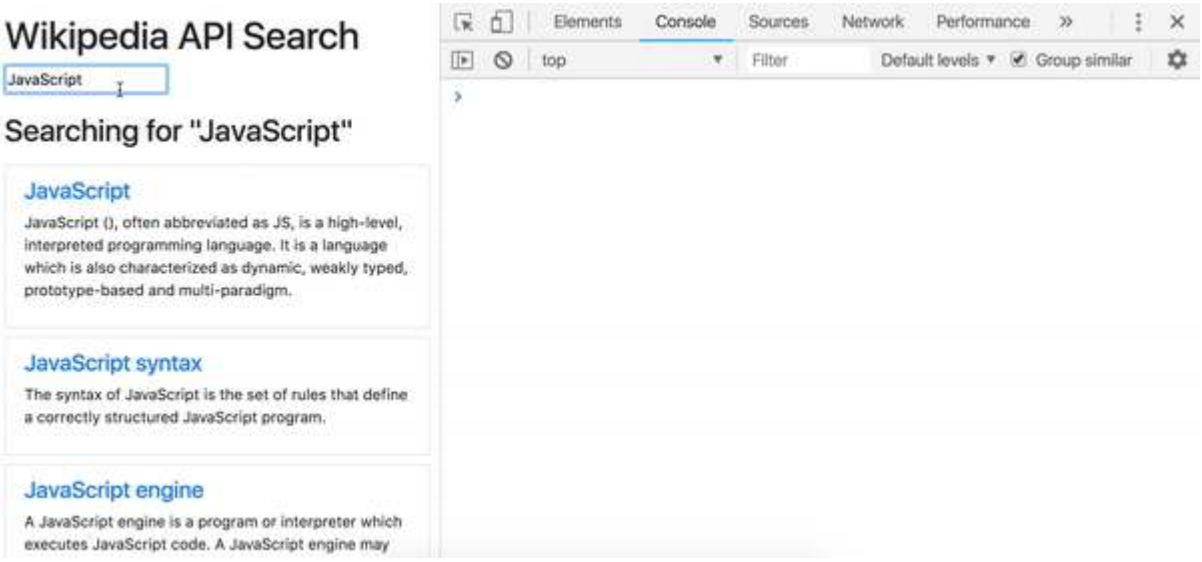
Fixing a Little Bug

The app sends AJAX requests even if the input's empty. We can fix that! (5 min. read)

I found a little bug while building this. Did you notice it?



Emptying the `input` throws this error.

```
▶Uncaught (in promise) SyntaxError: Unexpected token < in JSON at position 0        index.js:18
    at index.js:18
```

That's because we're sending an AJAX request without a search topic. Check out the URL in your Network tab.

https://en.wikipedia.org/w/api.php?origin=*&action=opensearch&search=

That link points to a default HTML page. We didn't get JSON back because we didn't specify a search topic.

To prevent this from happening we can avoid sending the request if the `input`'s empty.

We need a function that **does nothing** if the `input`'s empty, and **does the search** if it's filled.

Let's first create a function called `doNothing`. You can guess what it looks like.

```
import 'bootstrap/dist/css/bootstrap.min.css';
import { pipe, tap } from 'ramda';
import getInputValue from './getInputValue';
import getUrl from './getUrl';
import Results from './Results';

const doNothing = () => {};

const render = markup => {
  const resultsElement = document.getElementById('results');

  resultsElement.innerHTML = markup;
};

const searchAndRenderResults = pipe(
  getInputValue,
  getUrl,
  url =>
    fetch(url)
      .then(res => res.json())
      .then(Results)
      .then(render)
);

const inputElement = document.querySelector('input');

inputElement.addEventListener('keyup', searchAndRenderResults);
```

Just trust me for now :D

Next remove `getInputValue` from your `searchAndRenderResults` function. We need a bit more security before using it there.

```
import 'bootstrap/dist/css/bootstrap.min.css';
import { pipe, tap } from 'ramda';
import getInputValue from './getInputValue';
import getUrl from './getUrl';
import Results from './Results';
```

```
const doNothing = () => {};

const render = markup => {
  const resultsElement = document.getElementById('results');

  resultsElement.innerHTML = markup;
};

const searchAndRenderResults = pipe(
  getUrl,
  url =>
    fetch(url)
      .then(res => res.json())
      .then(Results)
      .then(render)
);

const inputElement = document.querySelector('input');

inputElement.addEventListener('keyup', searchAndRenderResults);
```

And create a new function, `makeSearchRequestIfValid`. This will use `getInputValue`.

```
import 'bootstrap/dist/css/bootstrap.min.css';
import { ifElse, isEmpty, pipe, tap } from 'ramda';
import getInputValue from './getInputValue';
import getUrl from './getUrl';
import Results from './Results';

const doNothing = () => {};

const render = markup => {
  const resultsElement = document.getElementById('results');

  resultsElement.innerHTML = markup;
};

const searchAndRenderResults = pipe(
  getUrl,
  url =>
    fetch(url)
      .then(res => res.json())
      .then(Results)
      .then(render)
);

const makeSearchRequestIfValid = pipe(
  getInputValue,
  ifElse(isEmpty, doNothing, searchAndRenderResults)
);

const inputElement = document.querySelector('input');

inputElement.addEventListener('keyup', searchAndRenderResults);
```

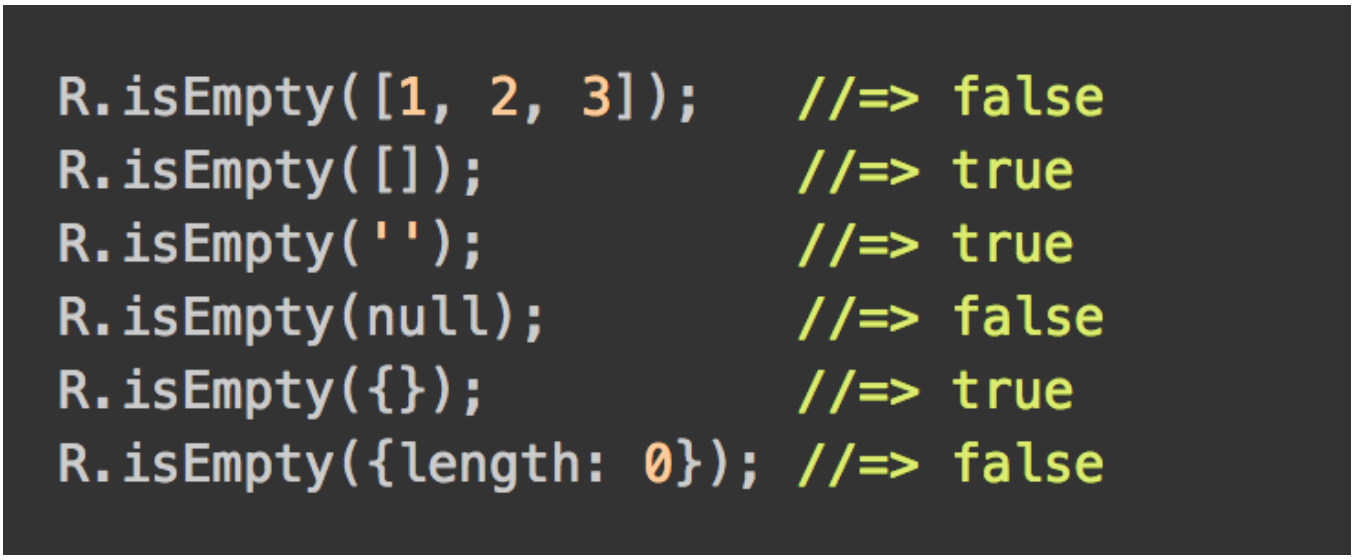
Take a minute to absorb this snippet.

```
const makeSearchRequestIfValid = pipe(
  getInputValue,
  ifElse(isEmpty, doNothing, searchAndRenderResults)
);
```

If the input value's empty, do nothing. Else, search and render the results.

You can gather that information just by reading the function. *That's* expressive.

Ramda's `isEmpty` function works with strings, arrays, objects. Here's a screenshot from their site.



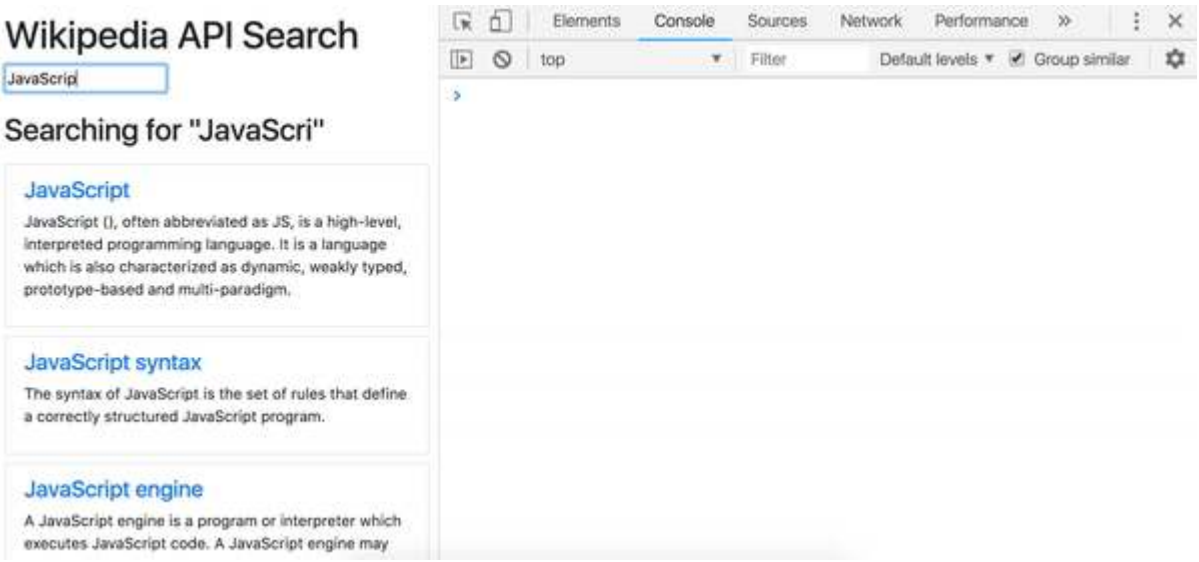
This makes it perfect to test our input value.

`ifElse` fits here nicely because when `isEmpty` returns true, `doNothing` runs. Otherwise `searchAndRenderResults` runs.

Update your event handler

```
inputElement.addEventListener('keyup', makeSearchRequestIfValid);
```

And check the results. No more errors!


Wikipedia API Search
JavaScrip
Searching for "JavaScri"
JavaScript
JavaScript (), often abbreviated as JS, is a high-level, interpreted programming language. It is a language which is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm.
JavaScript syntax
The syntax of JavaScript is the set of rules that define a correctly structured JavaScript program.
JavaScript engine
A JavaScript engine is a program or interpreter which executes JavaScript code. A JavaScript engine may
Elements
Console
Sources
Network
Performance
top
Filter
Default levels
Group similar