Basics of React

The mandatory to-do App (variant)

Day 1 - Fundamentals of Frontend Development

Demo

Happy New Year 2023 🎊

Search for an Item

Name:

Max Price: 0

Type:

Brand:

Search

Items

Id	Name	Price	Type	Brand	Delete
2	potato	3	food	fairprice	Delete
3	garlic	3	seasoning	fairprice	Delete
4	banana	3	fruit	fairprice	Delete

Add an Item

Name:

Price: 0

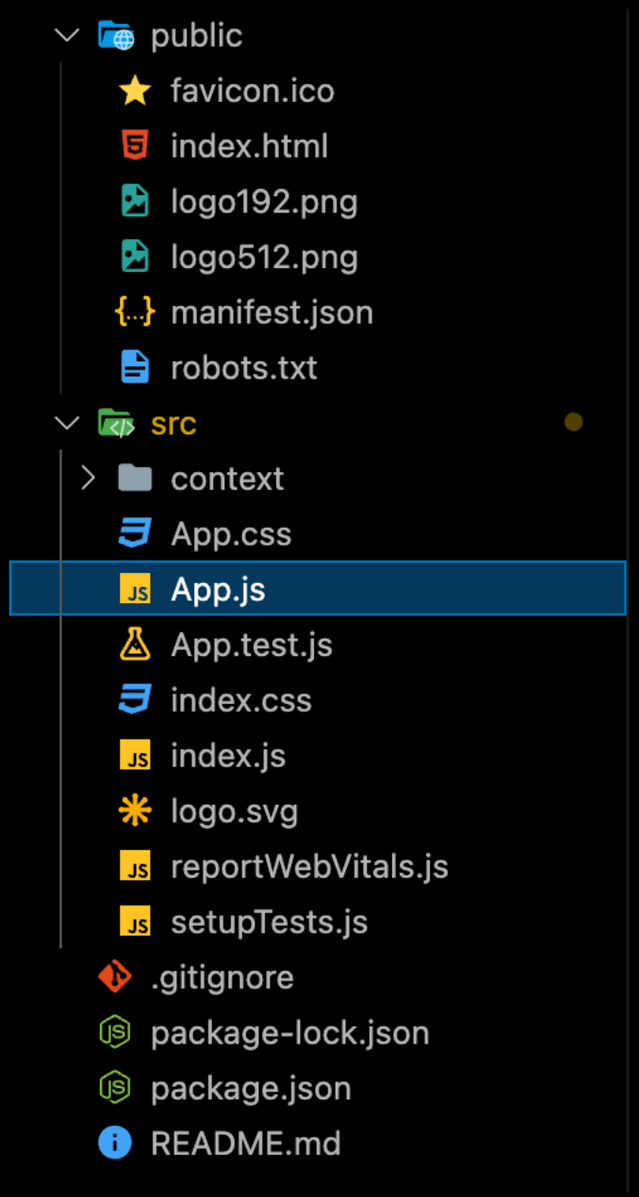
Type:

Brand:

Add Item

create-react-app

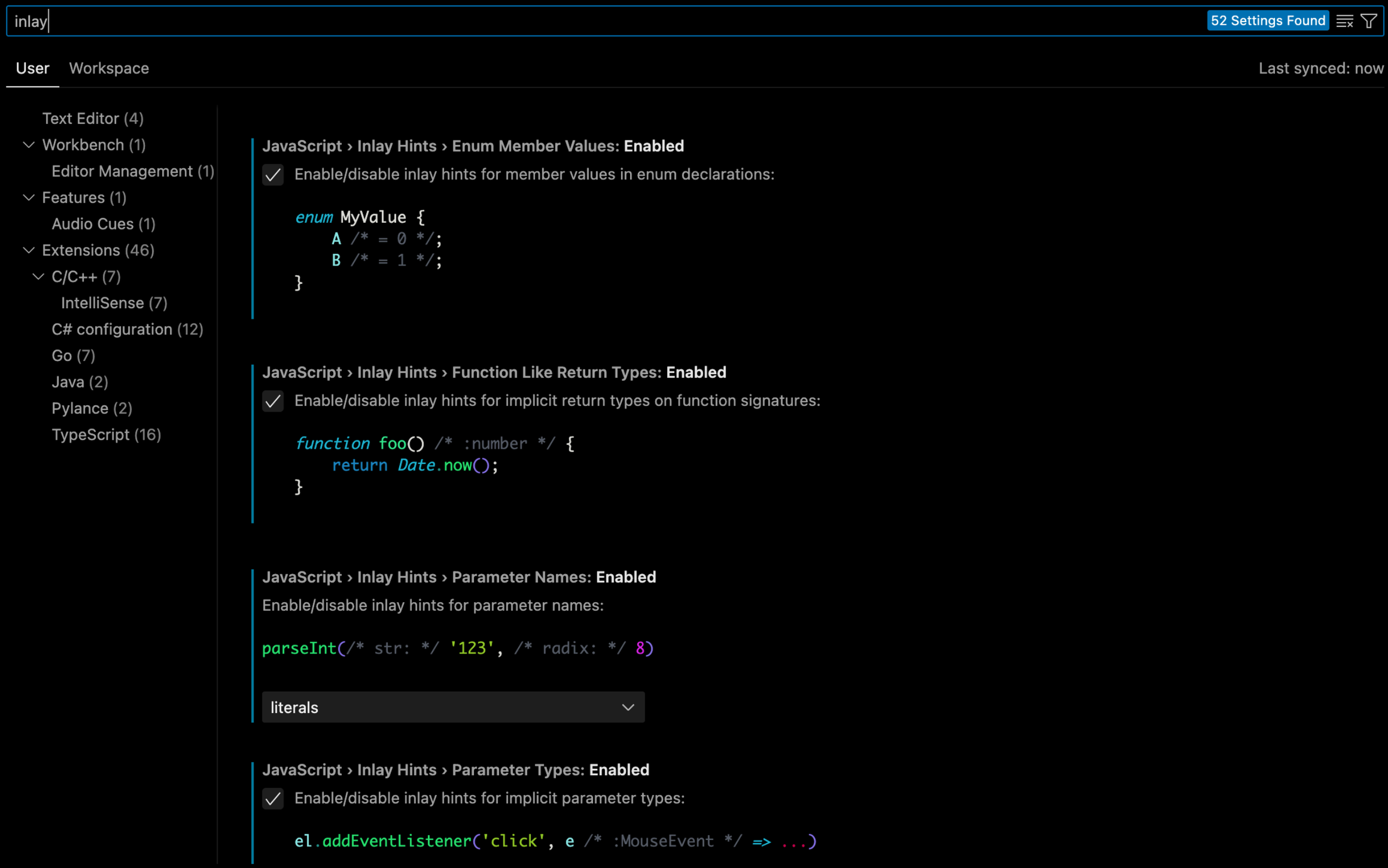
```
npx create-react-app react-inventory
cd react-inventory
npm start
```



Edit `src/App.js` and save to reload.

Learn React

Useful Setting



Useful Extensions

Abracadabra, refactor this! v6.18.2

Nicolas Carlo | 39,911 | ★★★★☆ (19) | ♥ Sponsor

Automated refactorings for VS Code, in JavaScript and TypeScript.

Disable | Uninstall

This extension is enabled globally.

Details | Feature Contributions | Changelog | Runtime Status

VS Code ships with a few basic refactorings. Abracadabra supercharges your editor with:

- 🎁 Much, much more refactorings
- ⚡ Shortcuts to trigger the most useful ones in no-time
- 💡 Quick Fixes to suggest refactorings when appropriate
- 🛠️ Options to customize the UX to your needs
- 💬 Refactorings that work with `.js`, `.jsx`, `.ts`, `.tsx`, `.vue` and `.svelte` files

Refactor Legacy Code in a snap! 👌

playground.js

playground.js > Shop > updateQuality

```
export class Shop {
  constructor(items = []) {
    this.items = items;
  }
  updateQuality() {
    for (var i = 0; i < this.items.length; i++) {
      const extracted = this.items[i];
      if (
        extracted.name ≠ "Aged Brie" &&
        item                          <stage passes to a TAFKAL80ETC concert"
      ) {
        if (extracted.quality > 0) {
          if (extracted.name ≠ "Sulfuras, Hand of Ragnaros") {
            extracted.quality = extracted.quality - 1;
```

Categories

Formatters | Programming Languages | Other

Extension Resources

Marketplace
Repository
License
Nicolas Carlo

More Info

Published	11/07/2019, 09:14:59
Last released	05/01/2023, 20:39:26
Last updated	06/01/2023, 05:34:22
Identifier	nicoespeon.abracadabra

Useful Extensions

Arrow Function Snippets v3.1.2

dein Software | 9,596 | ★★★★★(7) | Sponsor

VS Code Arrow function snippets for JS and TS

Disable | Uninstall

This extension is enabled globally.

Details | Feature Contributions | Runtime Status

Arrow Function

Trigger	Description	Result JS/TS
af→	implicit return without arg(s)	() => ■
afa→	implicit return with arg(s)	(arg) => ■
afad→	implicit return with arg destructuring	({prop, prop}) => ■
afo→	implicit return object	() => ({prop: value■})
afoa→	implicit return object with arg(s)	(arg) => ({prop: value■})

afe→	explicit return	() => {
		  return ■
		}

afea→	explicit return with arg(s)	(arg) => {
		  return ■
		}

afead→	explicit return with arg destructuring	({prop, prop}) => {
		  return ■
		}

afee→	explicit empty	() => {
		  ■
		}

Categories

Snippets

Extension Resources

Marketplace
Repository
License
dein Software

More Info

Published	20/10/2021, 11:45:54
Last released	15/12/2022, 04:43:31
Last updated	15/12/2022, 12:19:11
Identifier	deinsoftware.arrow-function-snippets

Useful Extensions

Const & Props Snippets v1.6.3

dein Software | 376 | ☆☆☆☆☆ | Sponsor

VS Code Const & Props snippets for JS and TS

Disable | Uninstall

This extension is enabled globally.

Details | Feature Contributions | Runtime Status

Below is a list of all available snippets and the triggers of each one. The → means the TAB key and ■ the final cursor position.

Variables

Trigger	Description	Result JS	Result TS
cv→	const variable	`const name = ■`	`const name = ■`
cvt→	const variable type		`const name: type = ■`
cvm→	const variable multiple type		`const name: (type | type) = ■`
cs→	const string	`const name = ''■`	`const name: string = ''■`
cn→	const number	`const name = 0■`	`const name: number = 0■`
cb→	const boolean	`const name = true■`	`const name: boolean = true■`
co→	const object	`const name = {}■`	`const name = {}■`
coi→	const object interface		`const name: Interface = {}■`
ca→	const array	`const name = []■`	`const name = []■`
cat→	const array type		`const name: type = []■`
cam→	const array multiple		`const name: (type | type) = []■`

Categories

Snippets

Extension Resources

Marketplace
Repository
License
dein Software

More Info

Published	21/04/2022, 04:39:17
Last released	15/12/2022, 04:47:57
Last updated	15/12/2022, 12:19:11
Identifier	deinsoftware.const-props-snippets

Useful Extensions

Better Comments v3.0.2

Aaron Bond | 4,118,647 | ★★★★★(155) | Sponsor

Improve your code commenting by annotating with alert, informational, TODOs, and more!

Disable | Uninstall

This extension is enabled globally.

Details | Feature Contributions | Changelog | Runtime Status

- Commented out code can also be styled to make it clear the code shouldn't be there
- Any other comment styles you'd like can be specified in the settings

```typescript
export class MyClass {
    /**
     * MyMethod
     * * Important information is highlighted
     * ! Deprecated method, do not use
     * ? Should this method be exposed in the public API?
     * TODO: refactor this method so that it conforms to the API
     * @param myParam The parameter for this method
     */
    public MyMethod(myParam: any): void {
        let myVar: number = 123;

        //* This is highlighted
        if (myVar > 0) {
            throw new TypeError(); //! This is an alert
        }

        //? This is a query
        let x = 1;

        //// this.lineOfCode() == commentedOut;

        //TODO: Create some test cases
    }
}
```

Categories

Formatters

Extension Resources

Marketplace
Repository
License
Aaron Bond

More Info

Published	10/06/2017, 04:28:12
Last released	30/07/2022, 09:30:35
Last updated	23/09/2022, 14:35:01
Identifier	aaron-bond.better-comments

Configuration

Useful Extensions

Bracket Pair Colorization Toggler v0.0.2

Dzhavat Ushev | 364,731 | ★★★½☆ (3)

Quickly toggle 'Bracket Pair Colorization' setting with a simple command

Disable | Uninstall

This extension is enabled globally.

Details | Feature Contributions | Changelog | Runtime Status

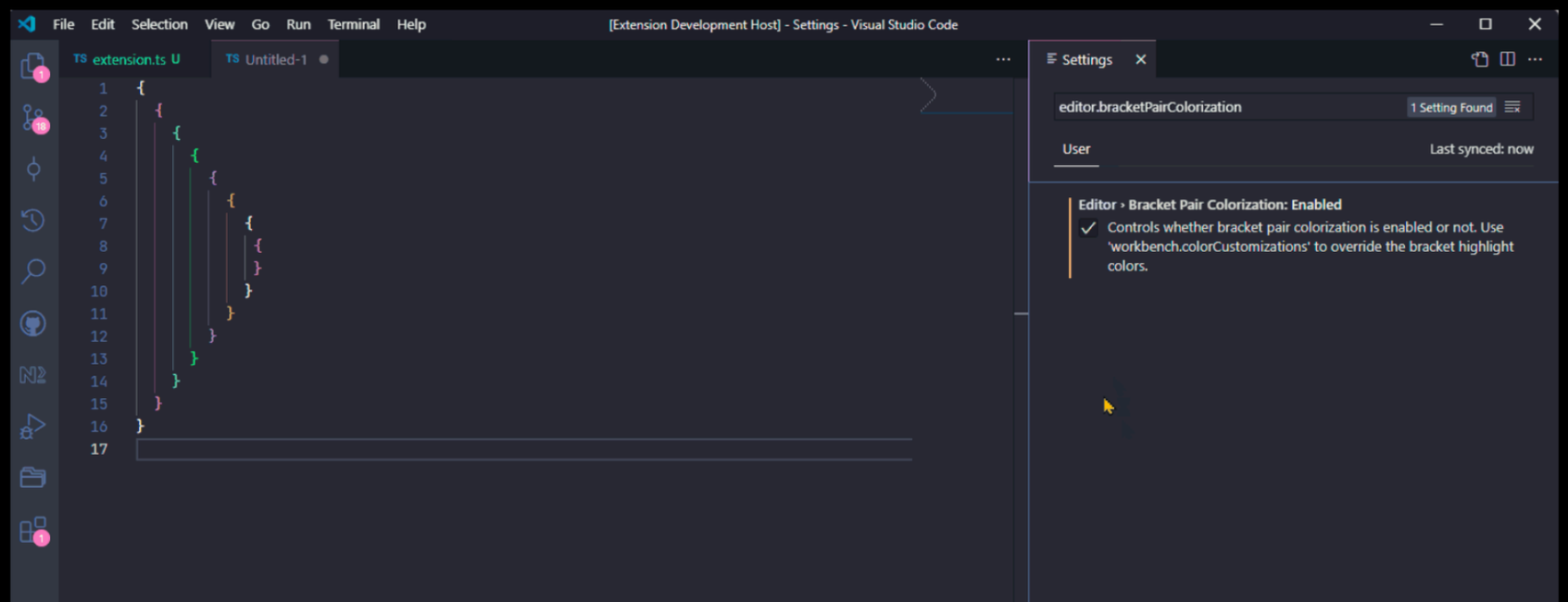
How to run

- Press `Ctrl + Shift + P` (Win, Linux) / `Cmd + Shift + P` (Mac) and search for the `Toggle 'Bracket Pair Colorization'` command.

Important

The extension only toggles the **global** "Bracket Pair Colorization" setting. If you have set the setting in your workspace, the extension will not work.

Demo



Categories

Other

Extension Resources

Marketplace
Repository
License
Dzhavat Ushev

More Info

Published	20/11/2021, 09:59:30
Last released	25/11/2021, 04:50:04
Last updated	27/10/2022, 13:21:02
Identifier	dzhavat.bracket-pair-toggler

Useful Extensions

CSS Peek v4.2.0

Pranay Prakash | 3,784,671 | ★★★½☆ (81)

Allow peeking to css ID and class strings as definitions from html files to respective CSS. Allows peek and goto definition.

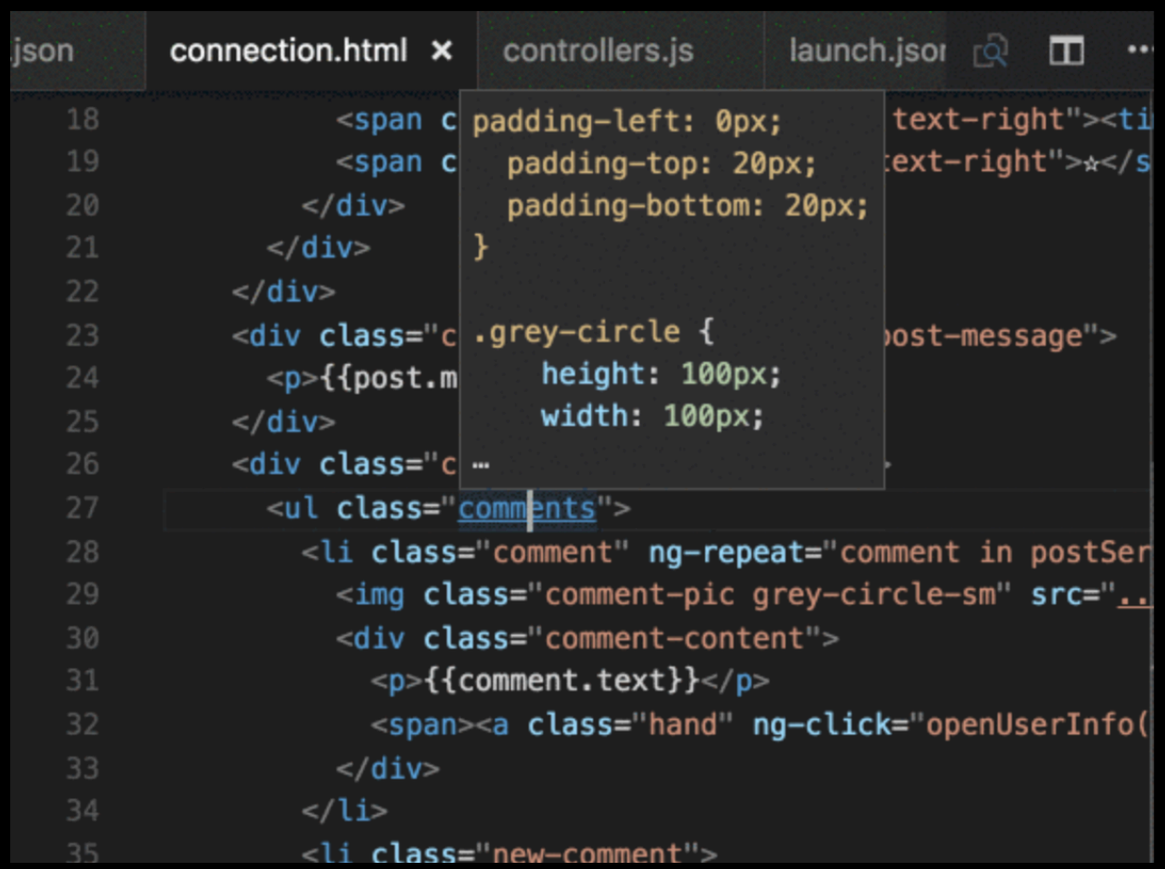
Disable | Uninstall

This extension is enabled globally.

Details | Feature Contributions | Runtime Status

This extension extends HTML and ejs code editing with `Go To Definition` and `Go To Symbol in Workspace` support for css/scss/less (classes and IDs) found in strings within the source code.

This was heavily inspired by a similar feature in Brackets called CSS Inline Editors.



Extension Resources

Marketplace
Repository
License
Pranay Prakash

More Info

Published	02/12/2016, 15:42:19
Last released	05/05/2021, 02:02:52
Last updated	10/10/2022, 12:06:45
Identifier	pranaygp.vscode-css-peek

Useful Extensions

CSS Peek v4.2.0

Pranay Prakash | 3,784,671 | ★★★½☆ (81)

Allow peeking to css ID and class strings as definitions from html files to respective CSS. Allows peek and goto definition.

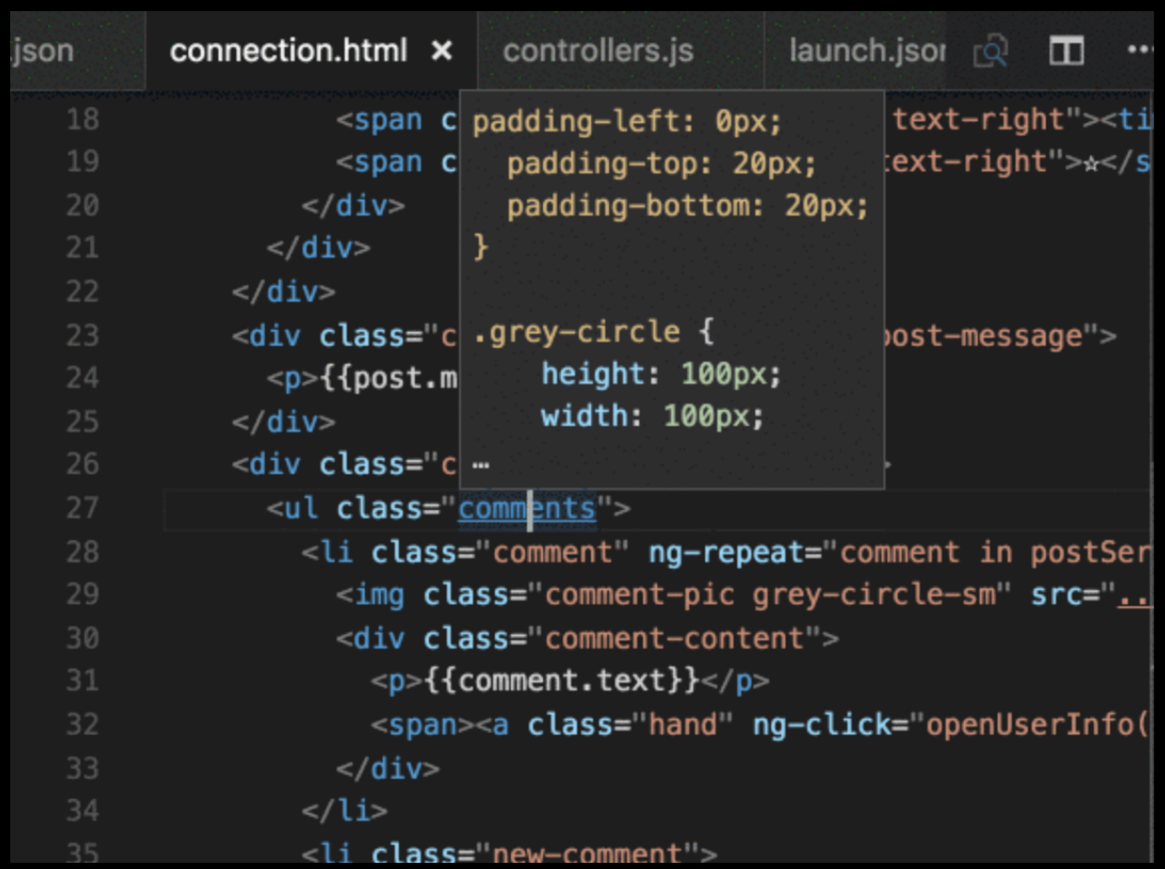
Disable | Uninstall

This extension is enabled globally.

Details | Feature Contributions | Runtime Status

This extension extends HTML and ejs code editing with `Go To Definition` and `Go To Symbol in Workspace` support for css/scss/less (classes and IDs) found in strings within the source code.

This was heavily inspired by a similar feature in Brackets called CSS Inline Editors.



Extension Resources

Marketplace
Repository
License
Pranay Prakash

More Info

Published	02/12/2016, 15:42:19
Last released	05/05/2021, 02:02:52
Last updated	10/10/2022, 12:06:45
Identifier	pranaygp.vscode-css-peek

Useful Extensions

Highlight Matching Tag v0.10.1

vincaslt | 1,532,117 | ★★★★½☆ (98)

Highlights matching closing and opening tags

Disable | Uninstall

This extension is enabled globally.

Details | Feature Contributions | Changelog | Runtime Status

of strings, any files, while also providing extensive styling options to customize how tags are highlighted.

Officially supported markup: **HTML** and **JSX**. Other flavors (XML, Vue, Angular, PHP) should work, but there are no guarantees. Feel free to report the issues on them anyway.

Features

```
const Component = () => (
  <div style="margin: 10px;">
    <FormWithButton
      btn={<button type="submit">Done</button>}
    />
    <a onClick={() => redirect()}>
      <>
        Text
      </>
    </a>
  </div>
)
```

Categories

Other

Extension Resources

Marketplace
Repository
License
vincaslt

More Info

Published	14/04/2017, 04:18:01
Last released	26/06/2021, 23:25:20
Last updated	29/11/2021, 12:30:13
Identifier	vincaslt.highlight-matching-tag

Useful Extensions

JSX HTML <tags/> v1.0.1

Angelo Rafael | 65,740 | ★★★★★ (7)

Use HTML Tags into JSX

Disable | Uninstall

This extension is enabled globally.

Details | Feature Contributions | Changelog | Runtime Status

Snippets

Content sectioning

Command	Result
address	<address></address>
article	<article></article>
aside	<aside></aside>
footer	<footer></footer>
header	<header></header>
h1	
h2	
h3	
h4	
h5	
h6	
main	<main></main>

Categories

Snippets

Extension Resources

Marketplace
Repository
Angelo Rafael

More Info

Published	14/06/2020, 09:05:09
Last released	23/10/2022, 23:44:25
Last updated	24/10/2022, 12:48:26
Identifier	angelorafael.jsx-html-tags

Useful Extensions

Path Intellisense v2.8.4

Christian Kohler | 8,804,883 | ★★★★★ (111)

Visual Studio Code plugin that autocompletes filenames

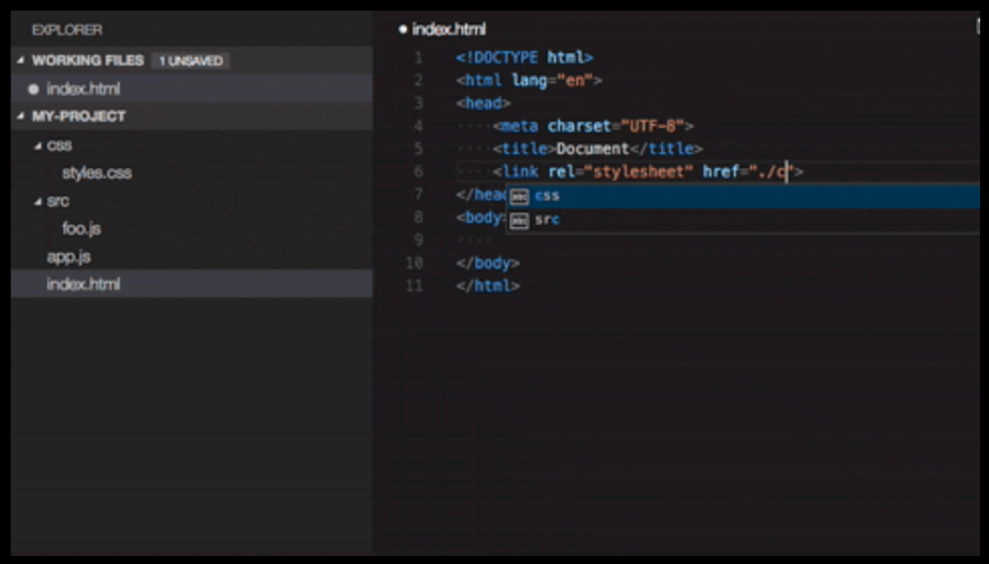
Disable | Uninstall

This extension is enabled globally.

Details | Feature Contributions | Changelog | Runtime Status

```
{ "typescript.suggest.paths": false }
{ "javascript.suggest.paths": false }
```

Usage



Categories

Other

Extension Resources

Marketplace
Repository
License
Christian Kohler

More Info

Published	17/04/2016, 14:06:10
Last released	20/12/2022, 06:20:32
Last updated	20/12/2022, 09:07:23
Identifier	christian-kohler.path-intellisense

Useful Extensions

Turbo Console Log v2.6.2

ChakrounAnas | 661,323 | ★★★★½ (59)

Automating the process of writing meaningful log messages.

Disable | Uninstall

This extension is enabled globally.

Details | Feature Contributions | Changelog | Runtime Status

```
JS index.js ●
1  const a = 1;
2  console.log('TCL: a', a);
3
4  function myFunction(firstParam) {
5      console.log('TCL: myFunction -> firstParam', firstParam);
6      const a = 1;
7      console.log('TCL: myFunction -> a', a);
8  }
9
10 class FisrtClass {
11     firstFunction() {
12         function secondFunction(firstParam) {
13             console.log('TCL: FisrtClass -> secondFunction -> firstParam', firstParam);
14             const a = 1;
15             console.log('TCL: FisrtClass -> secondFunction -> a', a);
16         }
17         const a = 1;
18     }
19 }
20
21 class SecondClass {
22     firstFunction() {
23         function secondFunction(firstParam) {
24             const a = 1;
25         }
26         const a = 1;
27     }
28 }
```

Categories

Other

Extension Resources

Marketplace
Repository
ChakrounAnas

More Info

Published	28/02/2018, 07:58:44
Last released	30/11/2022, 02:08:21
Last updated	01/12/2022, 09:34:34
Identifier	chakrounanas.turbo-console-log

Useful Extensions

SonarLint v3.13.0

SonarSource | 1,538,770 | ★★★★½ (77)

SonarLint is an IDE extension that helps you detect and fix quality issues as you write code in C, C++, Java, JavaScript, PHP, Python, HTML and Typ...

Disable | Uninstall

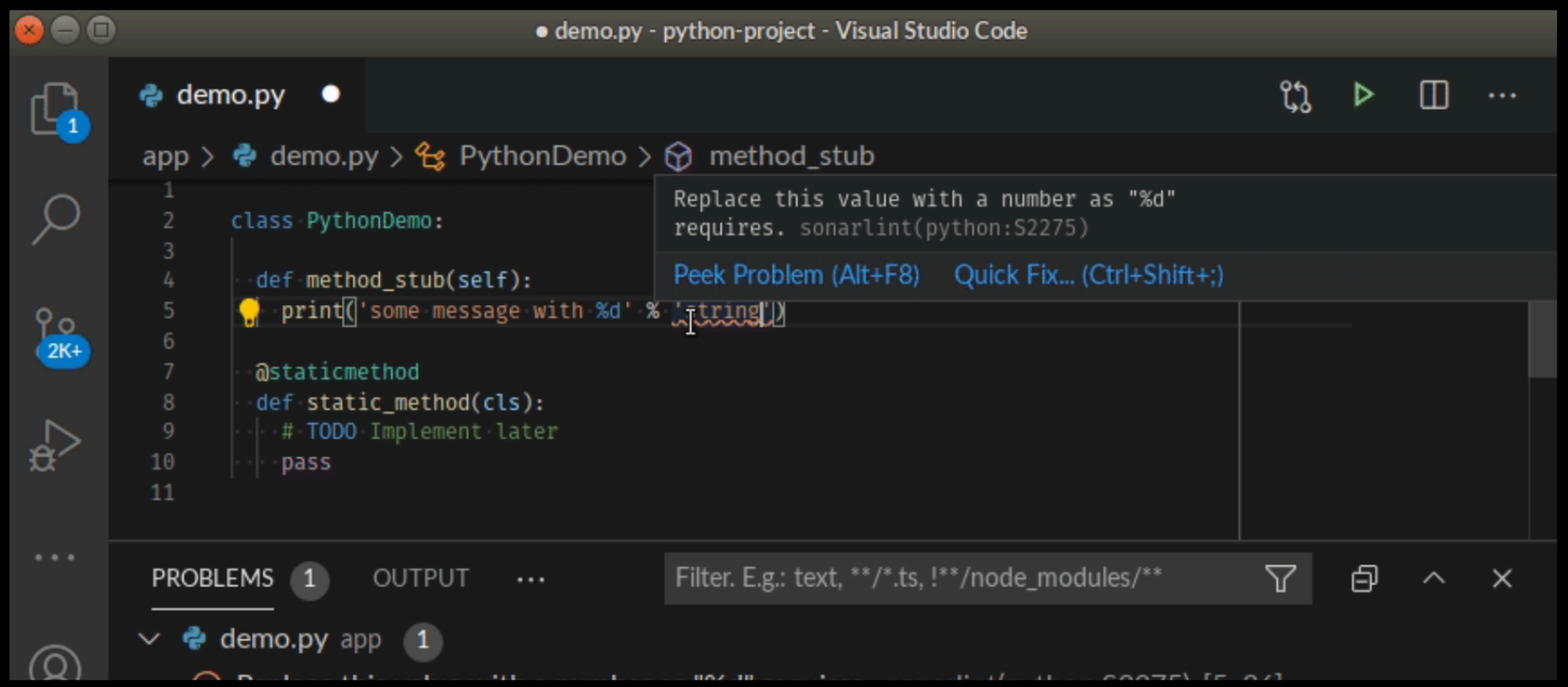
This extension is enabled globally.

Details | Feature Contributions | Changelog | Runtime Status

SonarLint is a free IDE extension that lets you fix coding issues before they exist! Like a spell checker, SonarLint highlights Bugs and Security Vulnerabilities as you write code, with clear remediation guidance so you can fix them before the code is even committed. SonarLint in VS Code supports analysis of C, C++, HTML, Java, JavaScript, PHP, Python and TypeScript, and you can install it directly from the VS Code Marketplace!

How it works

Simply open any source file, start coding, and you will start seeing issues reported by SonarLint. Issues are highlighted in your code, and also listed in the 'Problems' panel.



Categories

Linters

Extension Resources

Marketplace
Repository
License
SonarSource

More Info

Published	31/05/2017, 22:01:03
Last released	20/12/2022, 20:03:03
Last updated	21/12/2022, 14:30:14
Identifier	sonarsource.sonarlint-vscode

Useful Extensions

Vim v1.24.3

vscodevim | 4,462,691 | ★★★★☆ (315)

Vim emulation for Visual Studio Code

Uninstall

Details | Feature Contributions | Changelog | Runtime Status

We also recommend increasing Key Repeat and Delay Until Repeat settings in *System Preferences -> Keyboard*.

Windows

Like real vim, VSCodeVim will take over your control keys. This behavior can be adjusted with the `useCtrlKeys` and `handleKeys` settings.

⚙️ Settings

The settings documented here are a subset of the supported settings; the full list is described in the `Contributions` tab of VSCodeVim's extension details page, which can be found in the extensions view of VS Code.

Quick Example

Below is an example of a settings.json file with settings relevant to VSCodeVim:

```
{
  "vim.easymotion": true,
  "vim.incsearch": true,
  "vim.useSystemClipboard": true,
  "vim.useCtrlKeys": true,
  "vim.hlsearch": true,
  "vim.insertModeKeyBindings": [
    {
      "before": ["j", "j"],
      "after": ["<Esc>"]
    }
  ],
  "vim.normalModeKeyBindingsNonRecursive": [
    {
      "before": ["<leader>", "d"],
      "after": ["d", "d"]
    },
```

Categories

Keymaps

Extension Resources

Marketplace
Repository
License
vscodevim

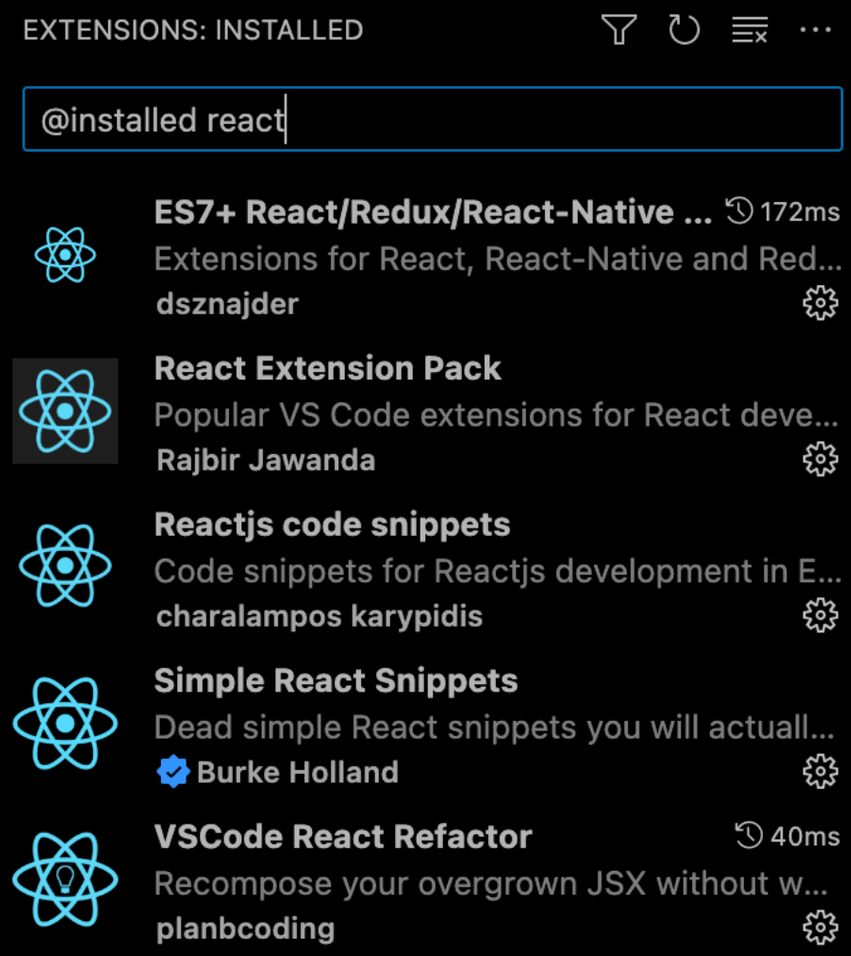
More Info

Published	29/11/2015, 18:38:55
Last released	07/11/2022, 10:59:49
Identifier	vscodevim.vim

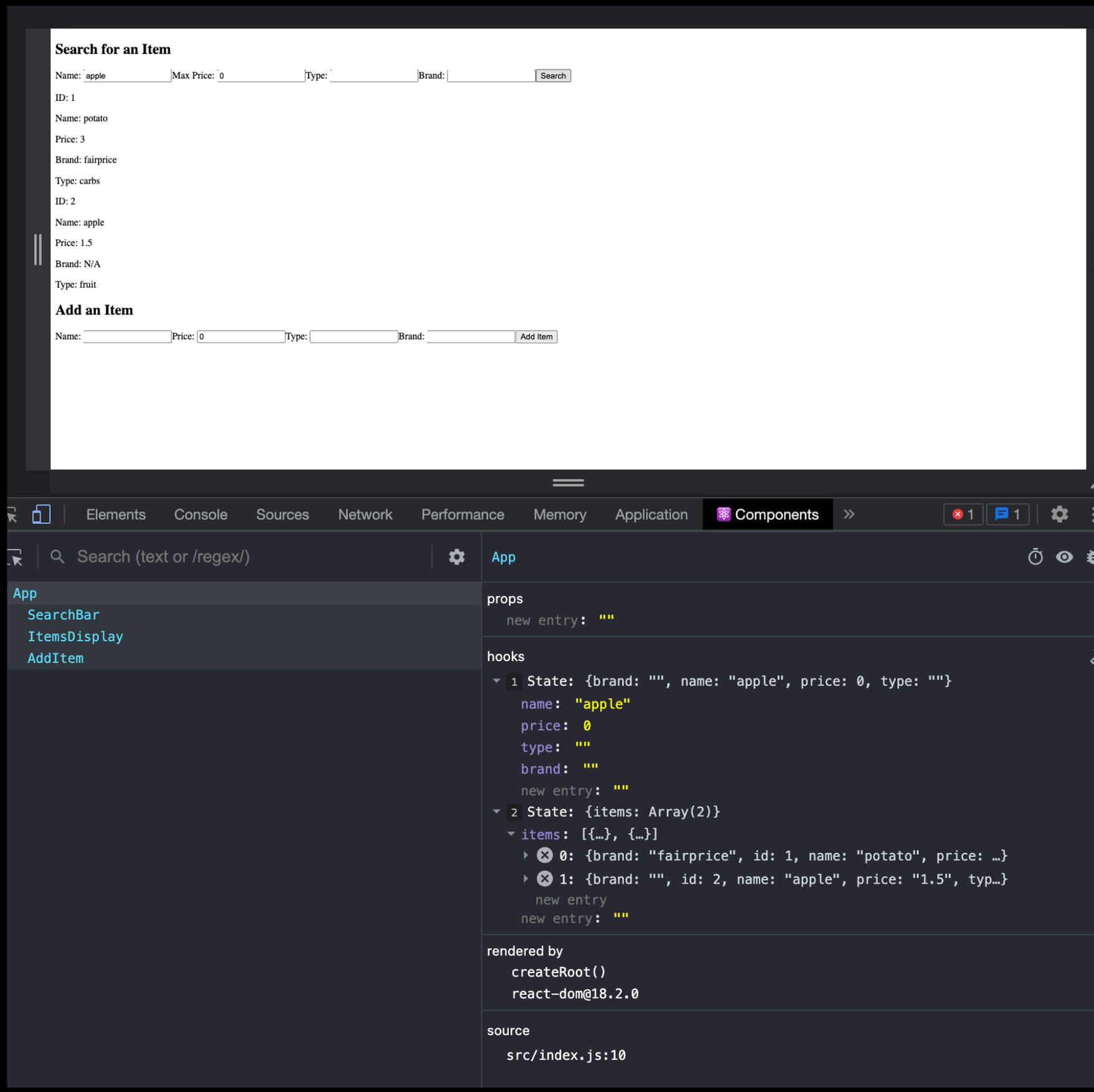
More Extensions

Snippets

- JavaScript (ES6) Code Snippets
- ES7 React/Redux/React-Native Snippets
- JS/JSX Snippets
- React Extension Pack
- Reactjs Code Snippets
- Simple React Snippets



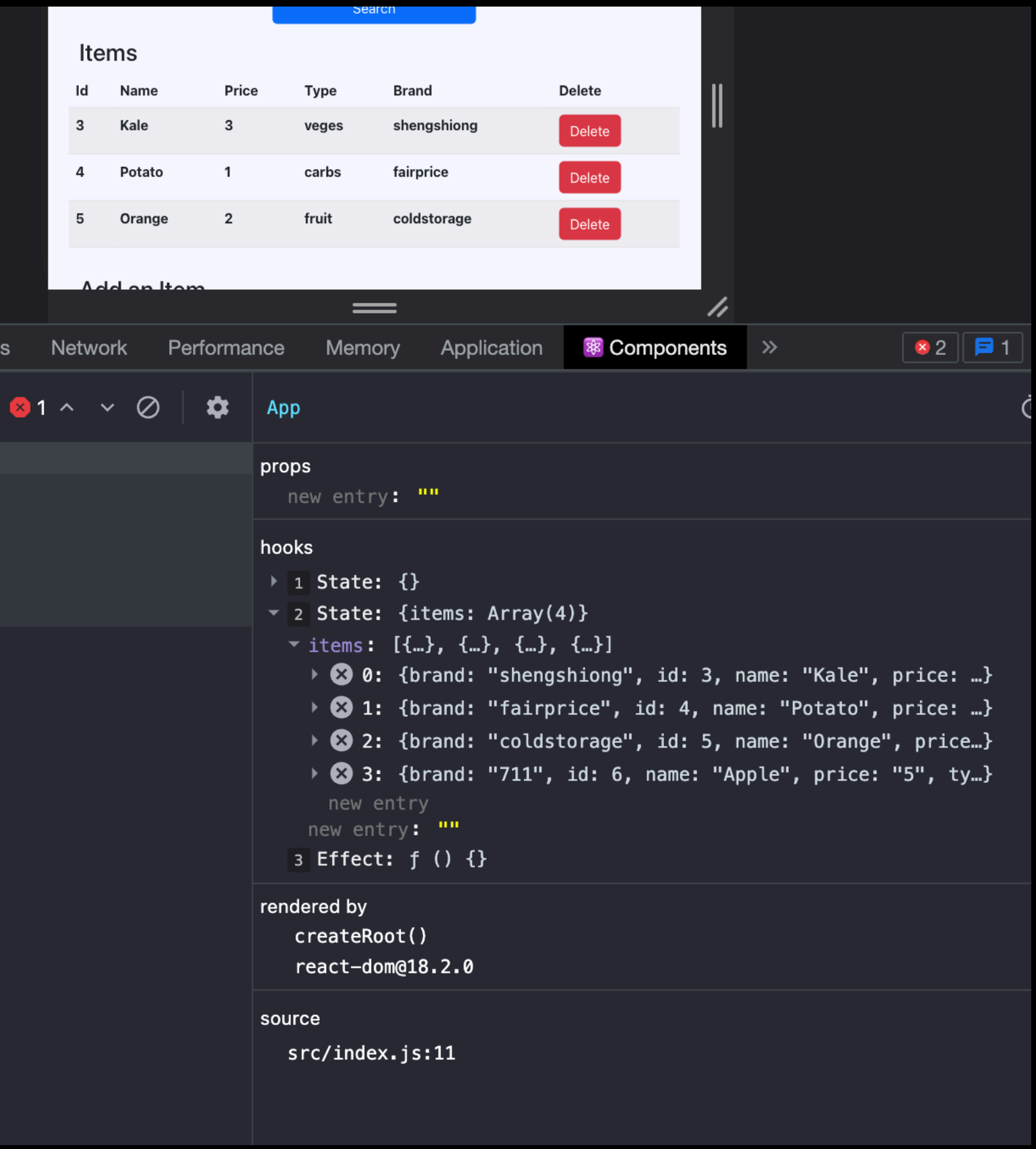
props
functional components
useState hook



useState

updating Arrays in State

```
function App() : Element {
  const [filters, setFilters] = useState({});
  const [data, setData] = useState({ items: [] });

const addItemToData : (item: any) => void = (item : any) : void => {
  let itemsValue : any[] = data["items"]; // get the content of key items
  item.id = itemsValue.length + 1;
  itemsValue.push(item); // add the list of elements inside
  // ! should not do this
  // ! data.items.push(responseData);
  // ! setData(data);
  setData({ items: itemsValue });
};
```



"Apple" **exists in** items

But, it doesn't trigger refresh

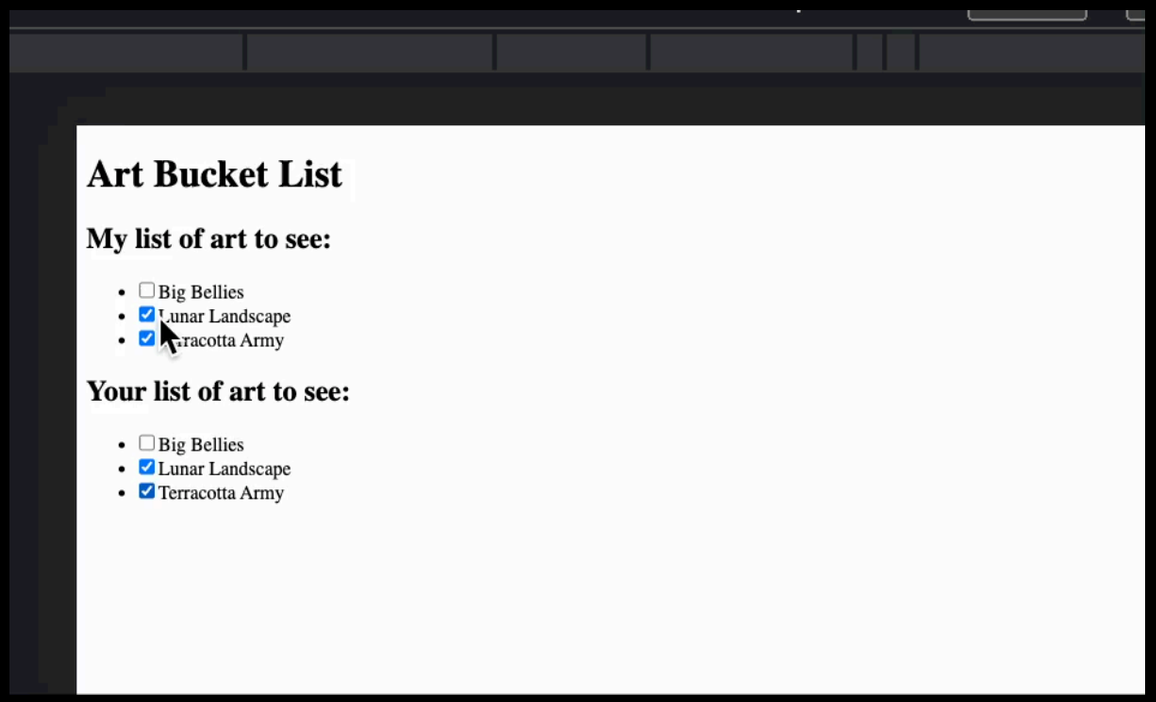
useState

updating Arrays in State

	avoid (mutates the array)	prefer (returns a new array)
adding	`push`, `unshift`	`concat`, `[...arr]` spread syntax (example)
removing	`pop`, `shift`, `splice`	`filter`, `slice` (example)
replacing	`splice`, `arr[i] = ...` assignment	`map` (example)
sorting	`reverse`, `sort`	copy the array first (example)

Buggy Mutation

accidentally sharing a state



" *You should only mutate objects that you have just created. If you were inserting a new object, you could mutate it, but if you're dealing with something that's already in state, you need to make a (deep) copy.* "

useState

updating Objects in State

```
const [position, setPosition] = useState({
  x: 0,
  y: 0
});
```

`state` is read-only

use `setState` to modify the `state`

```
onPointerMove={e => {
  position.x = e.clientX;
  position.y = e.clientY;
}}
```

❌

```
onPointerMove={e => {
  setPosition({
    x: e.clientX,
    y: e.clientY
  });
}}
```

✅

useState

updating nested Objects in State

```jsx
const [person, setPerson] = useState({
  name: "Niki de Saint Phalle",
  artwork: {
    title: "Blue Nana",
    city: "Hamburg",
    image: "https://i.imgur.com/Sd1AgUOm.jpg",
  },
});
```

Actually, they're 2 different objects, **not** exactly *nested*

```jsx
person.name = "Jane Doe"
```

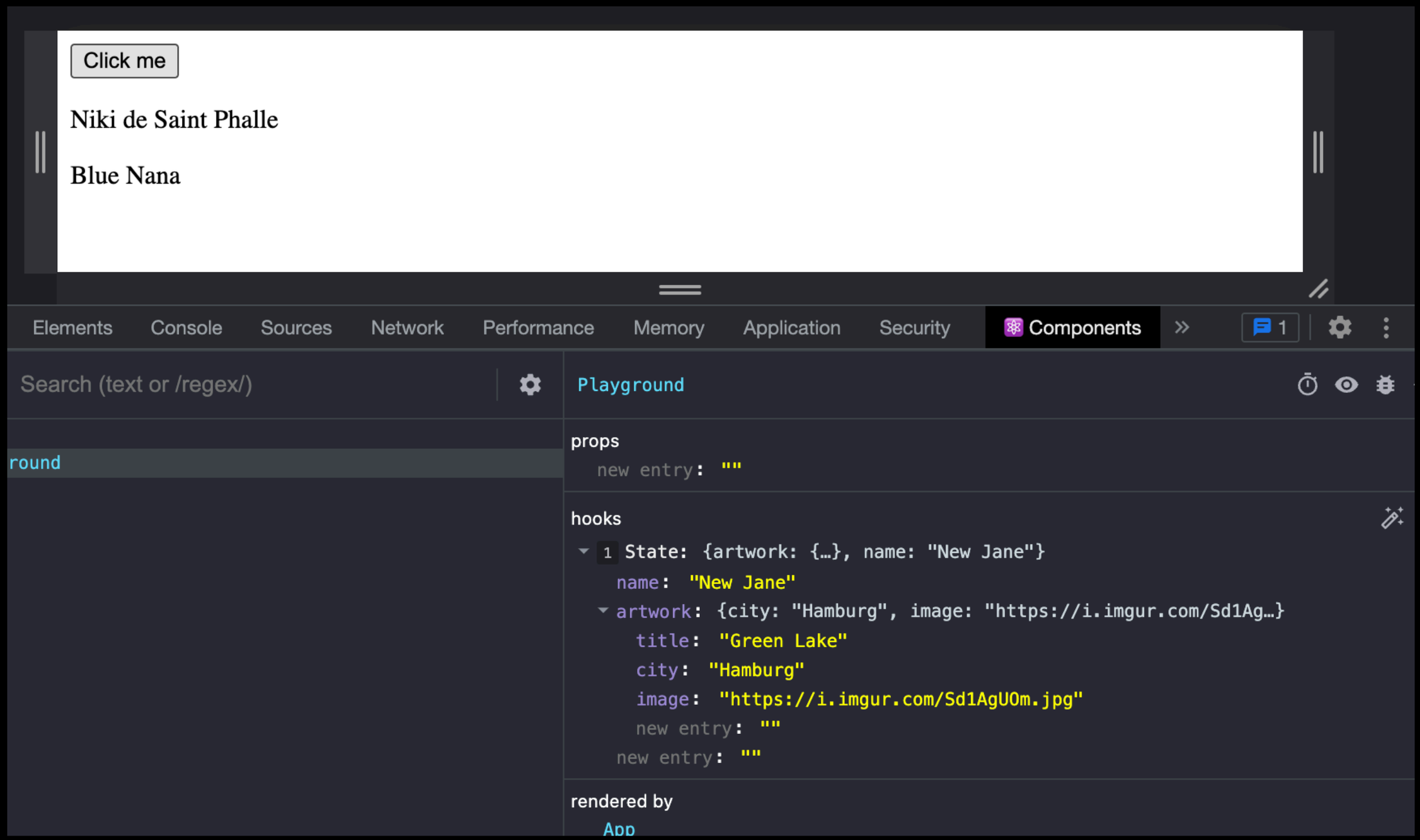
❌

```jsx
setPerson({
  ...person, // Copy other fields
  artwork: { // but replace the artwork
    ...person.artwork, // with the same one
    city: 'New Delhi' // but in New Delhi!
  }
});
```

✅

use the ...spread operator

Chrome React Developer Tools



useState render

```jsx
import { useState } from "react";

export default function Playground() {
  const [age, setAge] = useState(42);
  const [name, setName] = useState("Taylor");
  // ...

  function handleClick() {
    setName("Robin");
    console.log(name); // ? what is the output?
  }
}
```

useState set

```
const [age, setage] = useState(50);

function handleClick() {
  setage(age + 1);
  setage(age + 1);
}
```

useState set with updater

```
const [age, setAge] = useState(50);

function handleClick() {
  setAge((prev) => prev + 1);
  setAge((prev) => prev + 1);
}
```

useState initializer

```
function createInitialTodos() {
  const initialTodos = [];
  for (let i = 0; i < 50; i++) {
    initialTodos.push({
      id: i,
      text: 'Item ' + (i + 1)
    });
  }
  return initialTodos;
}

const [todos, setTodos] = useState(createInitialTodos());

const [todos, setTodos] = useState(createInitialTodos);
```

❌

✅

Too many re-renders

```
  // 🚩 Wrong: calls the handler during render
return <button onClick={handleClick()}>Click me</button>

// ✅ Correct: passes down the event handler
return <button onClick={handleClick}>Click me</button>

// ✅ Correct: passes down an inline function
return <button onClick={(e)
```

useState init

```
function createInitialTodos() {
  const initialTodos = [];
  for (let i = 0; i < 50; i++) {
    initialTodos.push({
      id: i,
      text: 'Item ' + (i + 1)
    });
  }
  return initialTodos;
}

const [todos, setTodos] = useState(createInitialTodos());

const [todos, setTodos] = useState(createInitialTodos);
```

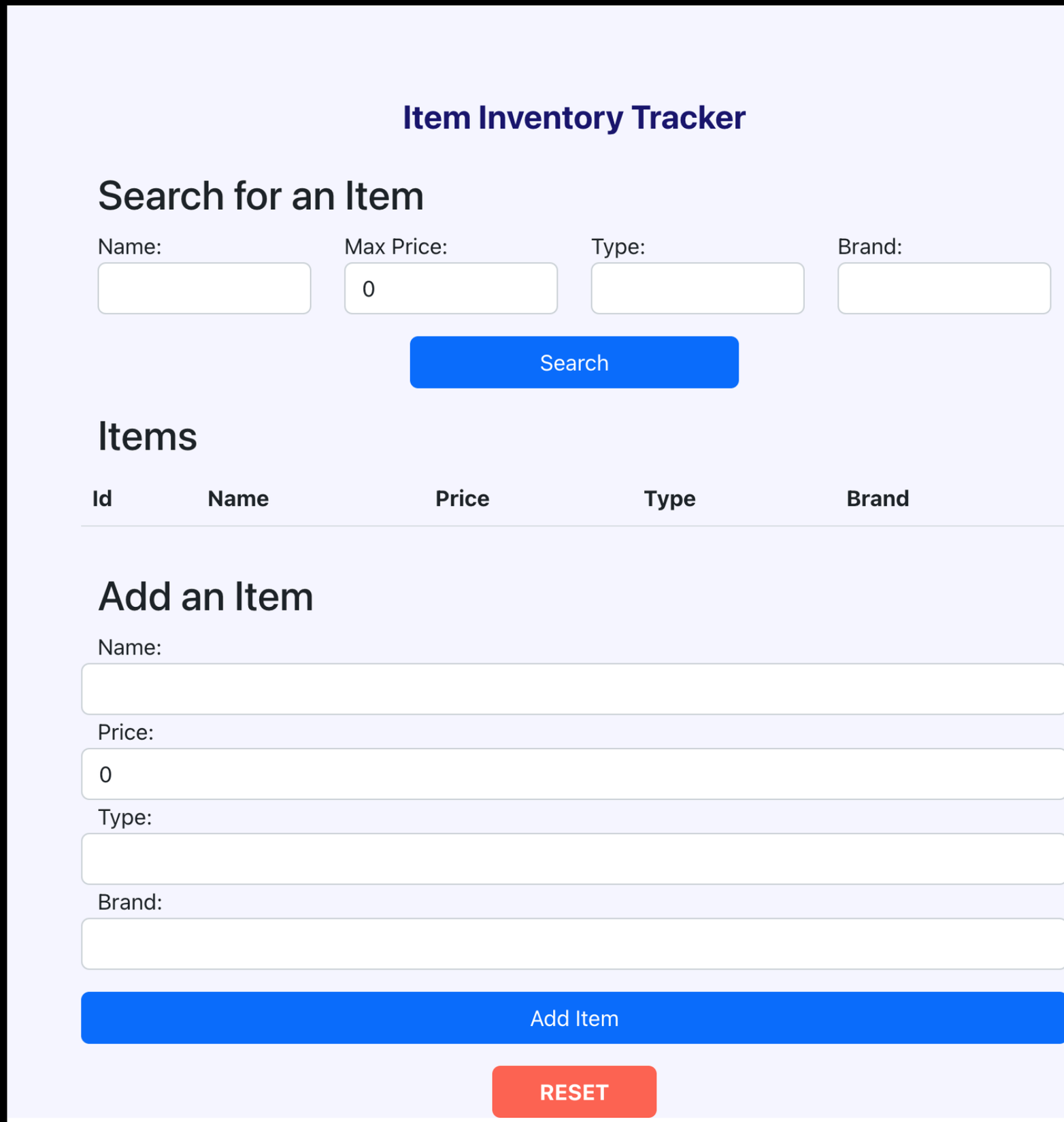
❌

✅

Styling

Bootstrap
Styled components

```
npm i bootstrap@5.2.2
npm i styled-components@5.3.6
```



CSS Overview

Sources | Network | Performance | Memory | Application | Security | CSS Overview

Overview summary

Elements 82	External stylesheets 0	Inline style elements 4	Style rules 1,043
Media queries 18	Type selectors 51	ID selectors 0	Class selectors 973
Universal selectors 1	Attribute selectors 11	Non-simple selectors 47	

Colors

Background colors: 4

#FFFFFF #F8F8FF #FF0000B3 #0D6EFD

Text colors: 4

#FFFFFF #191970 #212529 #000000

Contrast issues: 2

Text	Text
AA ✓	AA ⊘
AAA ⊘	AAA ⊘

Fill colors: 0

Forms

<input>, <select>

```
<div className="col">
  <label htmlFor="price-search-field">Max Price: </label>
  <input
    id="price-search-field"
    type="number" // there's checkbox and radio too
    value={price}
    className="form-control"
    onChange={(e) => {
      setPrice(e.target.value);
    }}
  />
</div>
```

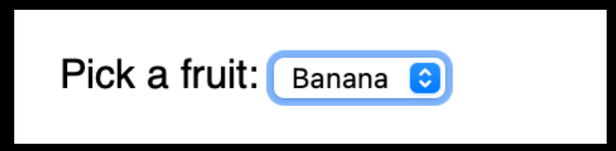
Forms

<input>, <select>

```jsx
// ✅ Good: uncontrolled input with an initial value
<input defaultValue={something} />

// ✅ Good: controlled input with onChange
<input value={something} onChange={e =>
setSomething(e.target.value)} />

// ✅ Good: readonly controlled input without on change
<input value={something} readOnly={true} />

// 🔴 Bug: controlled text input with no onChange handler
<input value={something} />
```

Forms

<input>, <select>




```jsx
export default function FruitPicker() {
  return (
    <label>
      Pick a fruit:
      <select name="selectedFruit">
        <option value="apple">Apple</option>
        <option value="banana">Banana</option>
        <option value="orange">Orange</option>
      </select>
    </label>
  );
}
```

Forms

reset a controlled form

```
const resetButtonPressed = () => {
  setName("");
  setPrice(0);
  setType("");
  setBrand("");
};
```

```
...
<form onReset={resetButtonPressed}>
...

        <div className="row mt-3">
          {/* fake col-4 stuff */}
          <div className="col col-5"></div>
          <button type="reset" className={"col-2 btn " + styles.error}>
            RESET
          </button>
        </div>
```

Forms

submit a controlled form

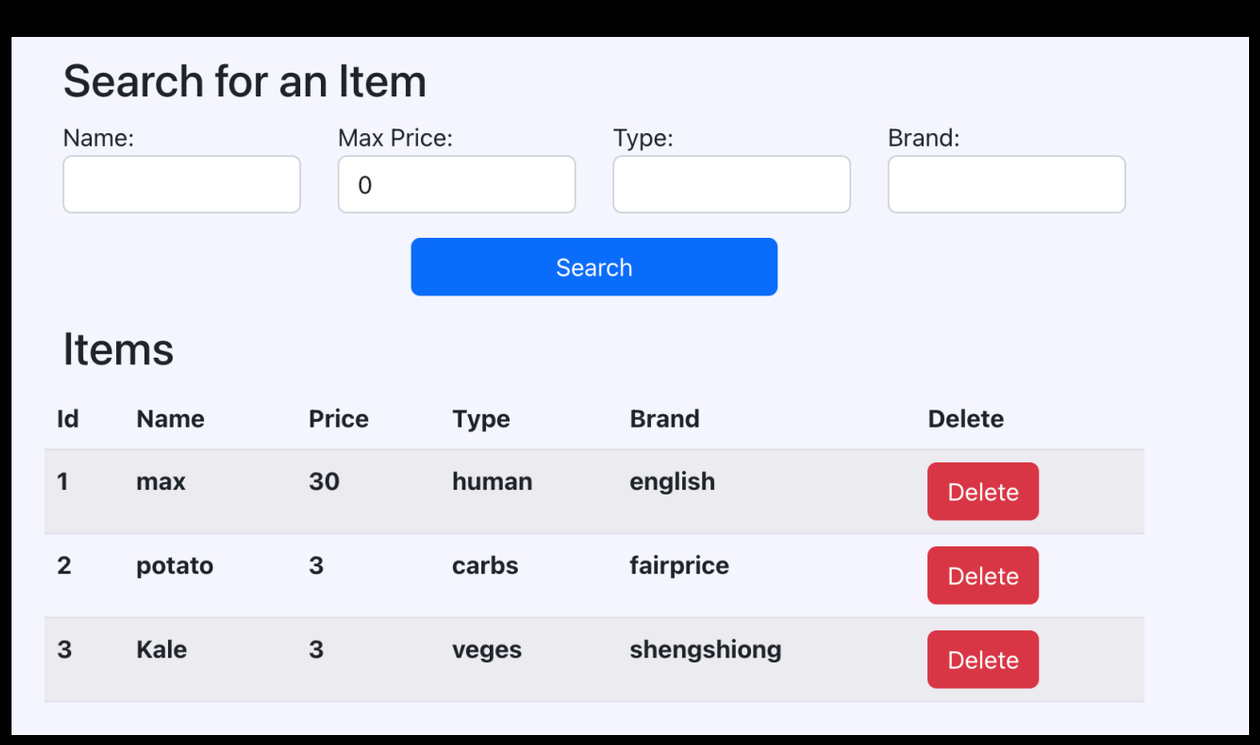
```
...
<form onReset={resetButtonPressed}>
...


          <div className="row mt-3">
           {/* fake col-4 stuff */}
           <div className="col col-4"></div>
            <button
              type="button"
              className="col-4 btn btn-primary"
              onClick={searchButtonPressed}
            >
              Search
            </button>
            {/* // does this submit the form?  */}
            <button>Button A</button>
            {/* // does this submit the form?  */}
            <button type="button">Button B</button>
          </div>
```

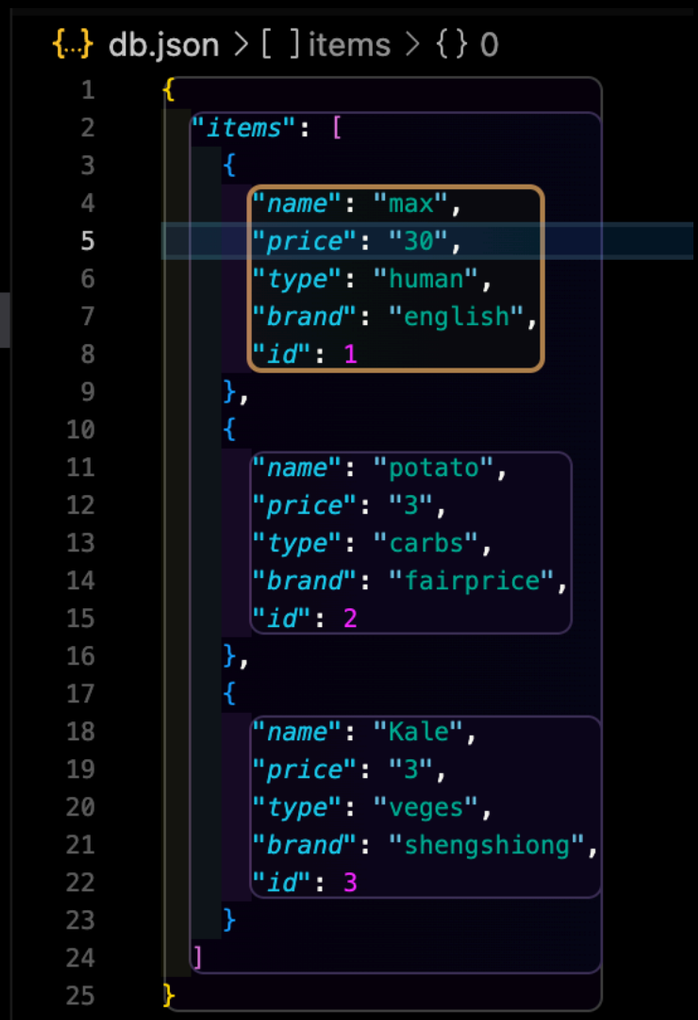
Add backend

```
npm i json-server
```

```
"server": "json-server db.json --port 9000"
```

```
npm run server
```





CR*D

GET, POST, DELETE

```
// fetch data from db
useEffect(() => {
  console.log("Component mount");
  fetch("http://localhost:9000/items")
    .then((response) => response.json())
    .then((data) => {
      setData({ items: data });
    });
}, []);
```

useEffect

setup, cleanup, dependencies

```
  const [serverUrl, setServerUrl] = useState('https://
localhost:1234');

  useEffect(() => {
    const connection = createConnection(serverUrl, roomId);
    connection.connect();
    return () => {
      connection.disconnect();
    };
  }, [serverUrl, roomId]);
```

useEffect

update state based on previous state from an Effect

```
export default function Counter() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    const intervalId = setInterval(() => {
      setCount(count + 1); // increment the counter every second...
    }, 1000)
    return () => {
      clearInterval(intervalId);
      console.log("cleared")
      }
  }, [count]);

  return <h1>{count}</h1>;
}
```

useEffect

update state based on previous state from an Effect

✅

```jsx
export default function Counter() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    const intervalId = setInterval(() => {
      setCount(c => c + 1); // ✅ Pass a state updater
    }, 1000);
    return () => clearInterval(intervalId);
  }, []); // ✅ Now count is not a dependency

  return <h1>{count}</h1>;
}
```

useEffect

dependency array

```
const [serviceList, setServiceList] = useState([]);
const [bookButton, setBookButton] = useState(false);

useEffect(() => {
  fetch("/viewall").then((response) => {
    setServiceList(response.data);
    serviceList.filter(function (serv) {
      if (serv.userId === userId) {
        setBookButton(false);
      } else {
        setBookButton(true);
      }
    });

  });
}, [serviceList, bookButton]);
```

✕

Infinite Renders

useEffect

dependency array: primitive vs reference type

```jsx
const useEffectTest = ({ count }) => {
  useEffect(() => {
    console.log('Run when primitive value is changed.');
  }, [count]);
};

const App = () => {
  const [count, setCount] = useState(0);
  const test = useEffectTest({ count });
  return (
    <>
      <h1>{count}</h1>
      <button onClick={() => setCount(count + 1)}>Click Me</button>
    </>
  )
}
```

- boolean
- null
- undefined
- number
- bitInt
- string
- symbol

useEffect

dependency array: primitive vs reference type

```
const useEffectTest = (user) => {
  useEffect(() => {
    console.log('Run when reference value is changed.');
  }, [user]);
};
const App = () => {
  const [count, setCount] = useState(0);
  const test = useEffectTest({ name: 'Amy', age: 27 });
  return (
    <>
      <h1>{count}</h1>
      <button onClick={() => setCount(count + 1)}>Click Me</button>
    </>
  )
```

- Array
- Function
- Other objects

useEffect

dependency array: primitive vs reference type

```
const useEffectTest = (user) => {
  useEffect(() => {
    console.log('Run when reference value is changed.');
  }, [user.name]);
};
```

Object.property

useEffect

dependency array: primitive vs reference type

```
const useEffectTest = (user) => {
  useEffect(() => {
    console.log('Run when reference value is changed.');
  }, [JSON.stringify(user)]);
};
```

JSON.stringify()

useEffect

dependency array: primitive vs reference type

```
const useEffectTest = (user) => {
  useEffect(() => {
    console.log('Run when reference value is changed.');
  }, [JSON.stringify(user)]);
};
```

JSON.stringify()

useEffect

dependency array: referential equality

```
> {} === {}
< false
> [] === []
< false
```

Arrays

```
> function test() { return () => "hello" }
< undefined
> test1 = test()
< () => "hello"
> test2 = test()
< () => "hello"
> test1 === test2
< false
```

functions

useEffect

update state based on functions

```
function createOptions() {
  return {
    serverUrl: serverUrl,
    roomId: roomId
  };
}
```

```
useEffect(() => {
  const options = createOptions();
  const connection = createConnection();
  connection.connect();
  return () => connection.disconnect();
}, [createOptions]); //
```

useEffect

update state based on functions

✅

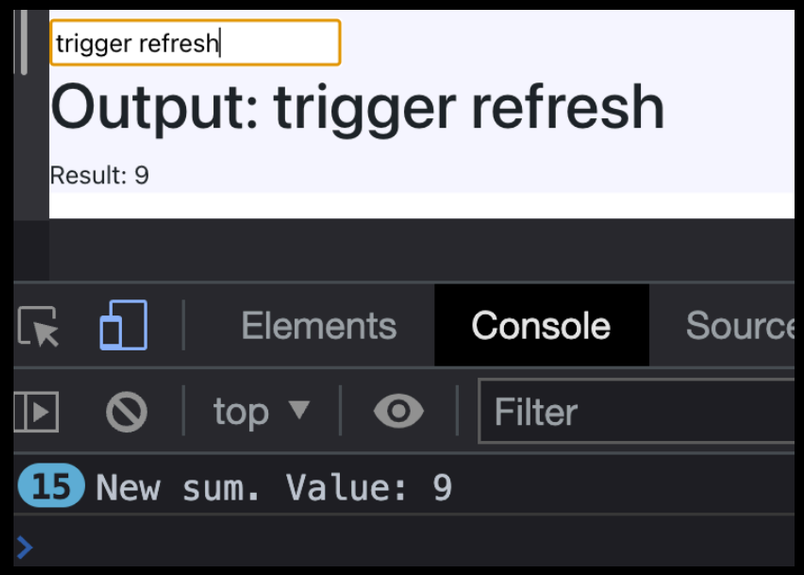
```
useEffect(() => {
  function createOptions() {
    return {
      serverUrl: serverUrl,
      roomId: roomId
    };
  }

  const options = createOptions();
  const connection = createConnection(options);
  connection.connect();
  return () => connection.disconnect();
}, [roomId]);
```

useEffect

should infinite render?

❌

```
const [result, setResult] = useState(0);
const [num1] = useState(4);
const [num2] = useState(5);

const sum = () => {
  return num1 + num2;
};

useEffect(() => {
  console.log(`New sum. Value: ${sum()}`);
  setResult(sum());
}, [sum]);
```




```
New sum. Value: 9                                   TestEffectAndCallback.jsx:16
Component mount                                     App.jsx:28
New sum. Value: 9                                   TestEffectAndCallback.jsx:16
Component mount                                     App.jsx:28
New sum. Value: 9                                   TestEffectAndCallback.jsx:16
```

useEffect

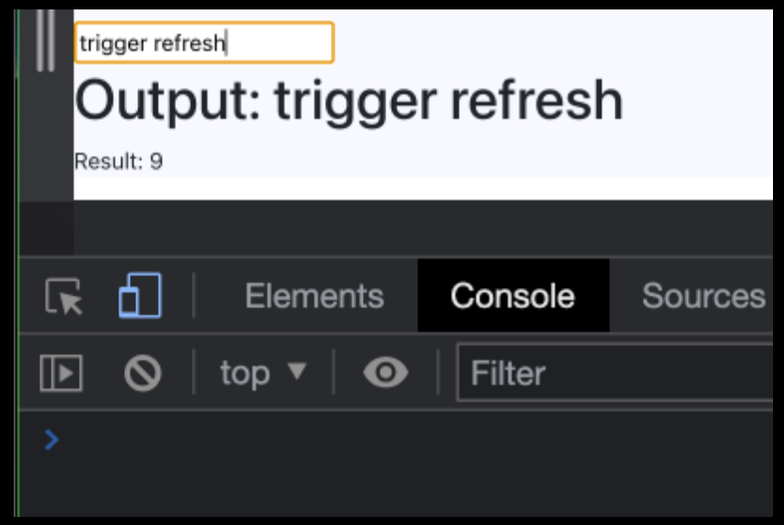
should infinite render?

```jsx
const [result, setResult] = useState(0);
const [num1] = useState(4);
const [num2] = useState(5);

const buildArray = () => [num1, num2];

useEffect(() => {
  console.log(`New sum. Value: ${sum()}`);
  setResult(buildArray());
}, [buildArray]);
```

```
▸Warning: Maximum update depth exceeded. This can happen when a component calls setState inside useEffect, but useEffect either doesn't have a dependency array, or one of the dependencies changes on every render.                react_devtools_backend.js:4012
    at TestEffectAndCallback (http://localhost:3000/main.f219505….hot-update.js:27:84)
    at section
    at O (http://localhost:3000/static/js/bundle.js:40427:8)
    at App (http://localhost:3000/static/js/bundle.js:53:80)
52 New sum. Value: 9                                                          TestEffectAndCallback.jsx:19
▸Warning: Maximum update depth exceeded. This can happen when a component calls setState inside useEffect, but useEffect either doesn't have a dependency array, or one of the dependencies changes on every render.                react_devtools_backend.js:4012
    at TestEffectAndCallback (http://localhost:3000/main.f219505….hot-update.js:27:84)
    at section
    at O (http://localhost:3000/static/js/bundle.js:40427:8)
    at App (http://localhost:3000/static/js/bundle.js:53:80)
52 New sum. Value: 9                                                          TestEffectAndCallback.jsx:19
```

useEffect

+ useCallback, referential equality




```jsx
const [result, setResult] = useState(0);
const [num1] = useState(4);
const [num2] = useState(5);

const sumWithCallback = useCallback(() => num1 + num2, [num1, num2]);
```

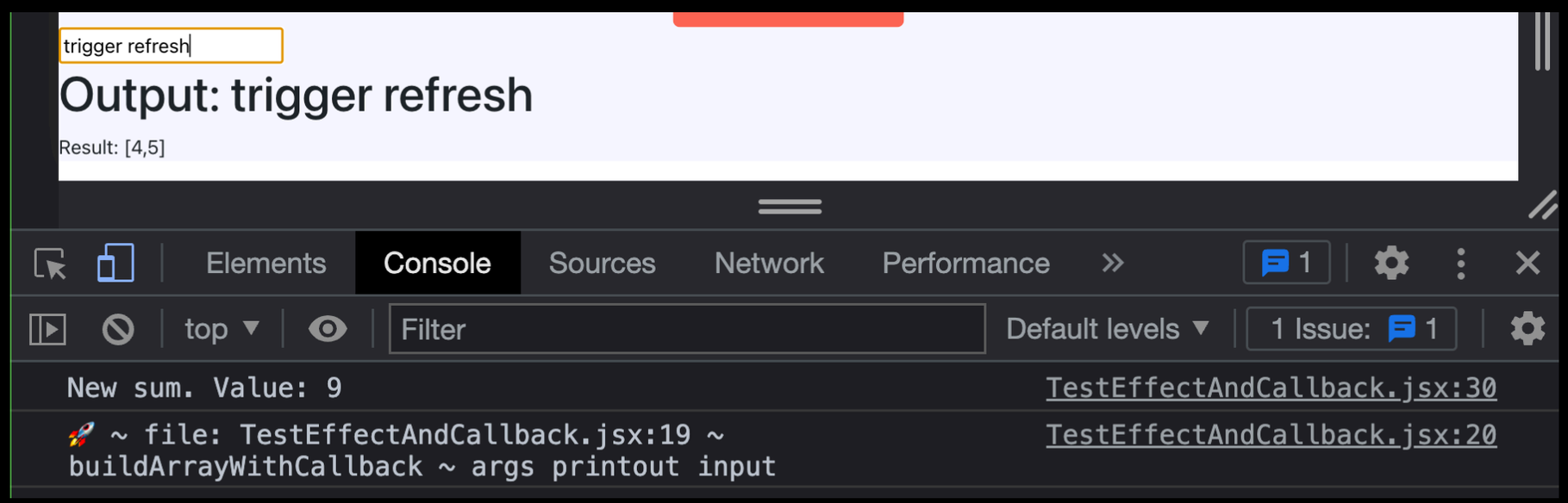
```jsx
useEffect(() => {
  console.log(`New sum. Value: ${sum()}`);
  setResult(sumWithCallback());
}, [sumWithCallback]);
```

```
New sum. Value: 9                                   TestEffectAndCallback.jsx:17
Component mount                                     App.jsx:28
New sum. Value: 9                                   TestEffectAndCallback.jsx:17
Component mount                                     App.jsx:28
```

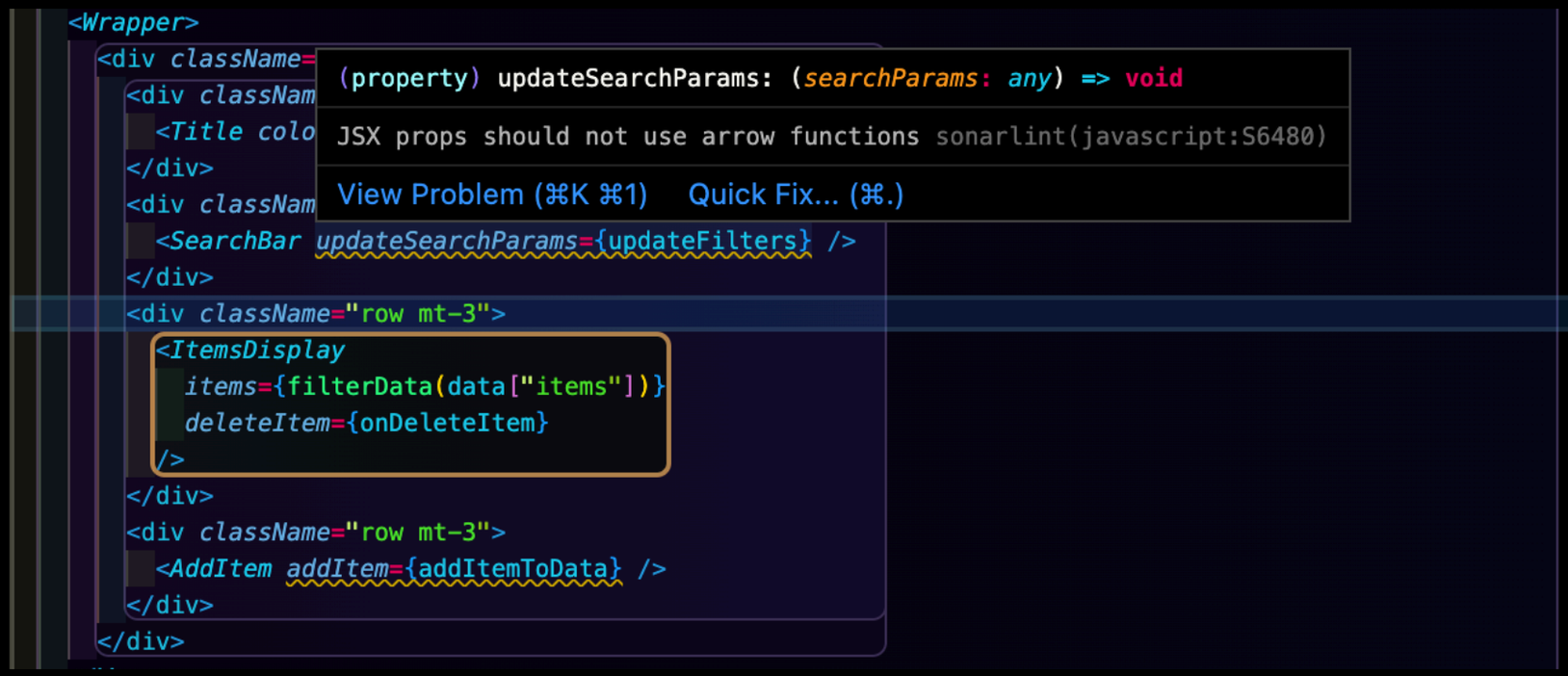
useEffect

+ useCallback, referential equality

```
const buildArrayWithCallback = useCallback(
  (args) => {
    console.log(
      "🚀 ~ file: TestEffectAndCallback.jsx:19 ~ buildArrayWithCallback ~ args printout",
      args
    );
    return [num1, num2];
  },
  [num1, num2]
);

useEffect(() => {
  console.log(`New sum. Value: ${sum()}`);
  setResult(buildArrayWithCallback("input"));
}, [buildArrayWithCallback]);
```



useCallback



Is this a good use case?

useCallback

Good use case

```
import useSearch from './fetch-items';
function MyBigList({ searchTerm, onItemClick }) {
  const items = useSearch(searchTerm);
  const map = item => <div onClick={onItemClick}>{item}</div>;
  return <div>{items.map(map)}</div>;
}
export default React.memo(MyBigList);
```

Child

```
import { useCallback } from 'react';
export function MyParent({ searchTerm }) {

  const onItemClick = useCallback(event => {
    console.log('You clicked ', event.currentTarget);
  }, [searchTerm]);

  return (
    <MyBigList
      searchTerm={searchTerm}
      onItemClick={onItemClick}
    />
  );
}
```

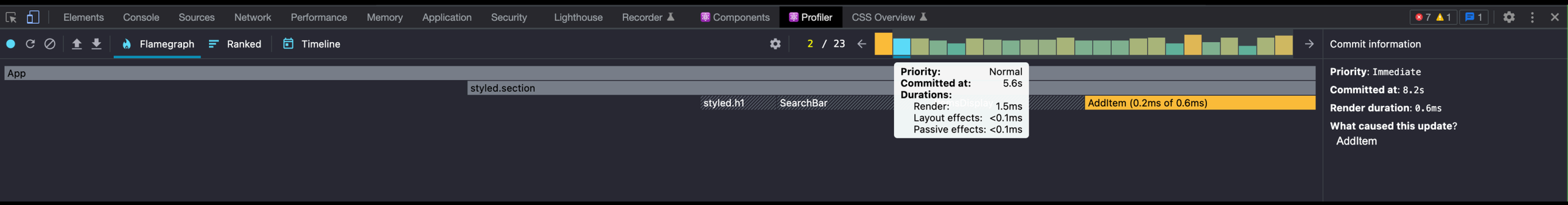
Parent

useCallback

Bad use case

```
import { useCallback } from 'react';


function ParentComponent() {
  // Contrived use of `useCallback()`
  const handleClick = useCallback(() => {
    // handle the click event
  }, []);
  return <ChildComponent onClick={handleClick} />;
}


function ChildComponent ({ onClick }) {
  return <button onClick={onClick}>I am a child</button>;
}
```

Profile before optimizing



Quantify the increased performance