React to Python

Creating React Front-End Web Applications with Python

Inland Empire Python Users Group
February 16, 2021

John Sheehan

Full-Stack Python?

Front-end Web Development

- Back-end code can use any language
- JavaScript is the way of the web
- I’m “less than enamored” with JavaScript
- I really like Python
- I want to develop using just one language
- So what’s a Python developer to do?

Full-Stack Python?

Python in the browser

- **Transcrypt** - *Precompiled to JavaScript*
- **Brython** - *Compiled to JavaScript on page load*
- **Skulpt** - *Compiled to JavaScript after page load*
- **Batavia** - *Precompiled to bytecode with JavaScript interpreter*
- **PyPy.js** - *JavaScript Python interpreter*
- **Pyodide** - *WebAssembly Python interpreter*

Full-Stack Python

What it might look like...

- **Python** - *Because Python, of course*
- **Transcrypt** - *For transpiling Python to JavaScript*
- **React** - *For building functional reactive front-end web applications*
- **Material-UI** - *For theming and stylized React components*
- **npm** - *For managing JavaScript libraries*
- **Parcel** - *For build automation and bundling*
- **Flask / Gunicorn / Nginx** - *For serving up the app in production*

Transcrypt

Features

- PIP installable
- Python code is transpiled to JavaScript before being deployed
- Very small JavaScript runtime (~40K)
- Generates sourcemaps for troubleshooting Python in the browser
- Generated JavaScript is human readable
- Generated JavaScript can be minified
- Performance is comparable to native JavaScript

Transcrypt

Features

- Maps Python data types and language constructs to JavaScript
- Acts as a bridge between the Python and JavaScript worlds
- Supports almost all Python built-ins and language constructs
- Limited support for the Python standard library
- Your Python code can “directly” call JavaScript functions
- Native JavaScript can call your Python functions
- Only supports 3rd party Python libraries that are pure Python

Transcrypt - Hello World

```
$ python -m venv venv
$ source venv/bin/activate
(venv) $ pip install transcrypt
```

hello.py

```
def say_hello():
    document.getElementById('destination').innerHTML = "Hello World!"

def clear_it():
    document.getElementById('destination').innerHTML = ""
```

Transcrypt - Hello World

hello.html

```html
<!DOCTYPE html>
<html lang="en">
    <body>
        <script type="module">
            import {say_hello, clear_it} from "./__target__/hello.js";
            document.getElementById("sayBtn").onclick = say_hello;
            document.getElementById("clearBtn").onclick = clear_it;
        </script>
        <button type="button" id="sayBtn">Click Me!</button>
        <button type="button" id="clearBtn">Clear</button>
        <div id="destination"></div>
    </body>
</html>
```

```
(venv) $ transcrypt --nomin --map hello
```

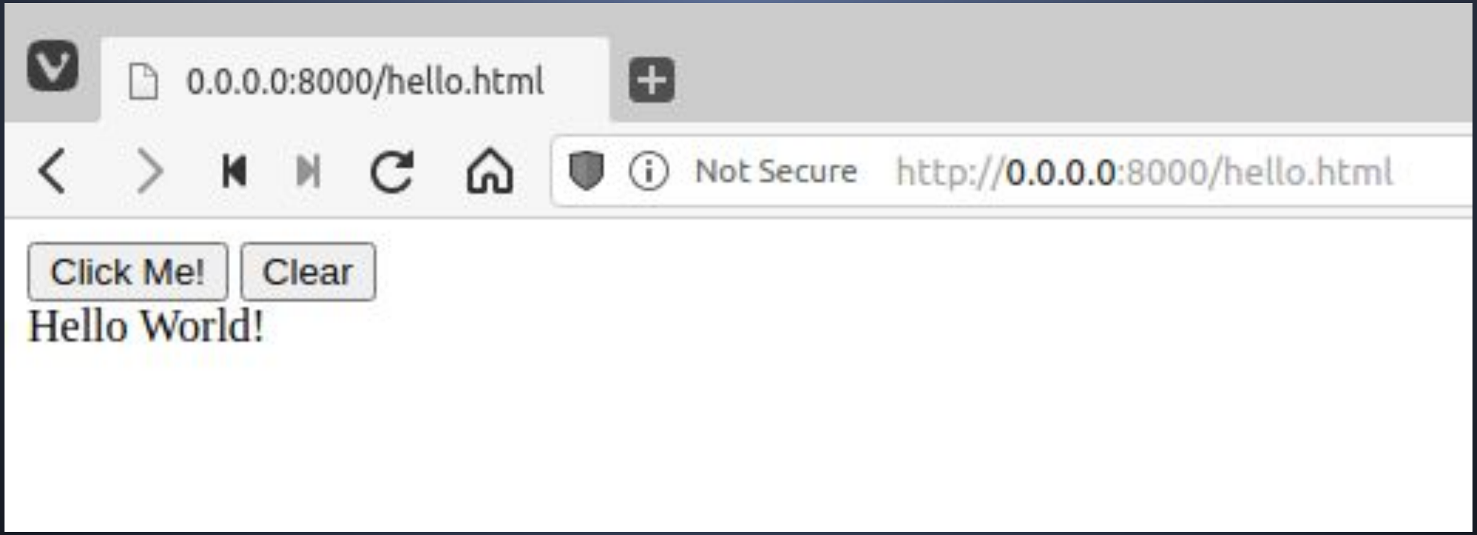
Transcrypt - Hello World

Use the built-in Python HTTP server:

```
(venv) $ python -m http.server
```

This starts up a webserver that serves up files in the current directory at:

http://localhost:8000



Hello World!

Transcrypt - Hello World

Generates readable JavaScript:

__target__/hello.js

```
// Transcrypt'ed from Python
import {AssertionError, ... , zip} from './org.transcrypt.__runtime__.js';
var __name__ = '__main__';
export var say_hello = function () {
   document.getElementById ('destination').innerHTML = 'Hello World!';
};
export var clear_it = function () {
   document.getElementById ('destination').innerHTML = '';
};

//# sourceMappingURL=hello.map
```

Transcrypt - Sourcemaps

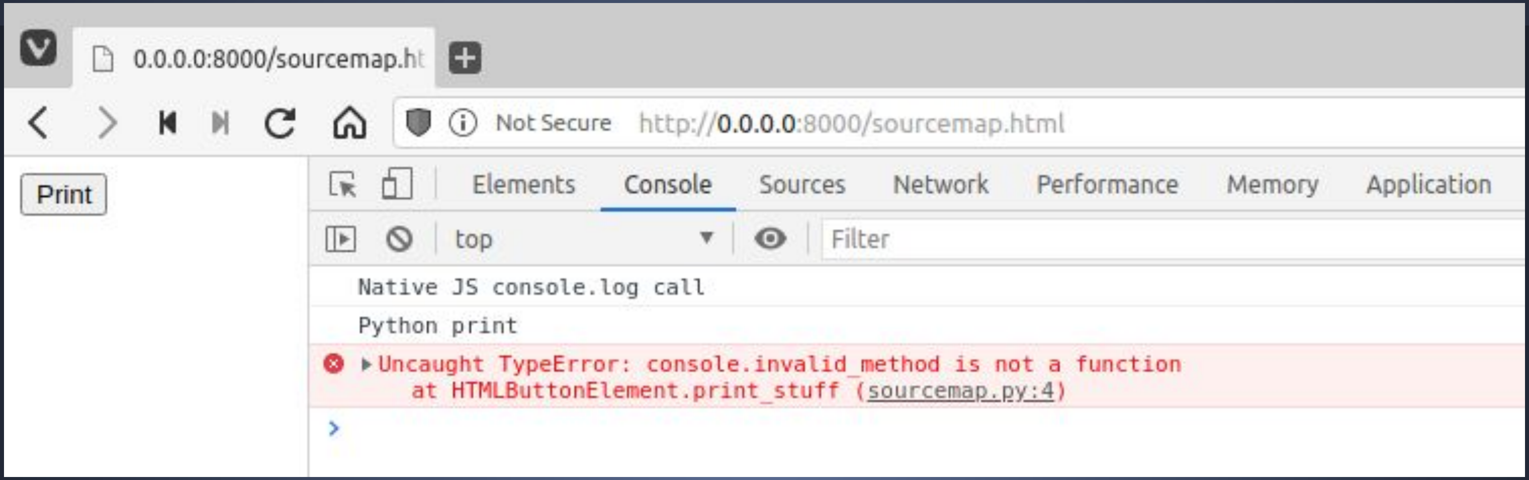
sourcemap.py

```
def print_stuff():
    console.log("Native JS console.log call")
    print("Python print")
    console.invalid_method("This will be an error")
```

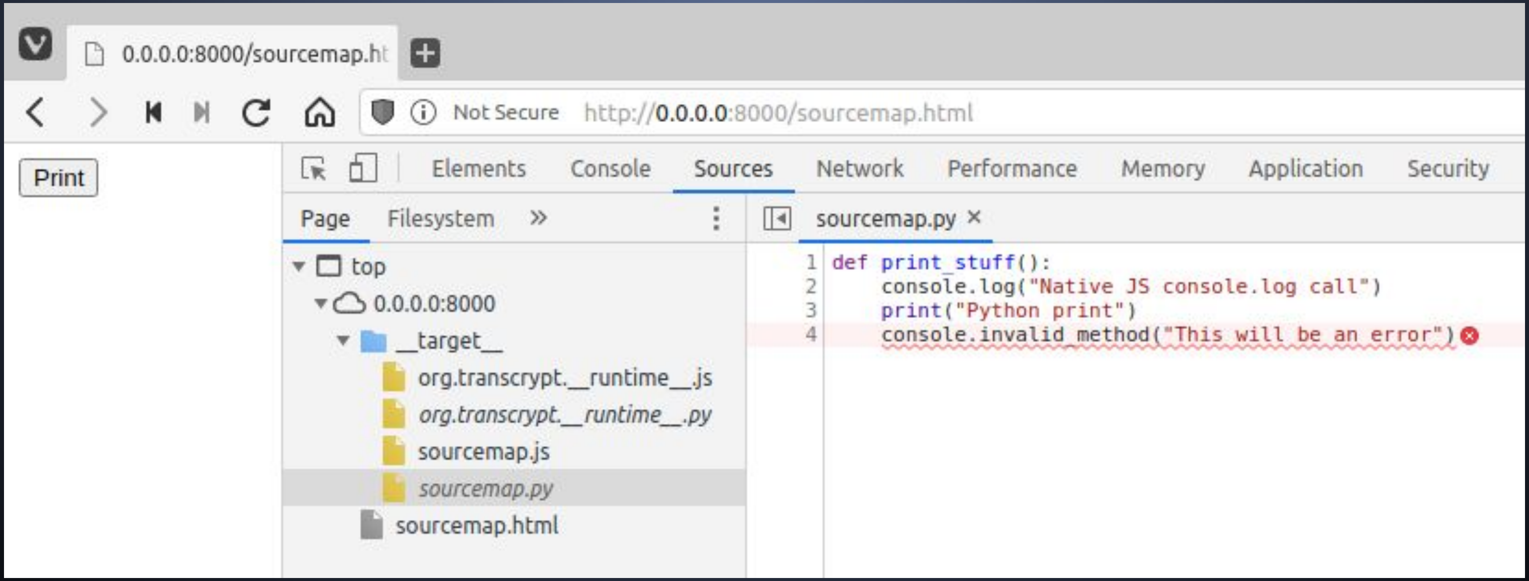
```
<!DOCTYPE html>
<html lang="en">
    <body>
        <script type="module">
            import {print_stuff} from "./__target__/sourcemap.js";
            document.getElementById("printBtn").onclick = print_stuff;
        </script>
        <button type="button" id="printBtn">Print to Console</button>
    </body>
</html>
```

```
(venv) $ transcrypt --nomin --map sourcemap
```

Transcrypt - Sourcemaps


0.0.0.0:8000/sourcemap.ht
Not Secure http://0.0.0.0:8000/sourcemap.html
Print
Elements Console Sources Network Performance Memory Application
top
Filter

```
Native JS console.log call
Python print
Uncaught TypeError: console.invalid_method is not a function
    at HTMLButtonElement.print_stuff (sourcemap.py:4)
```


0.0.0.0:8000/sourcemap.ht
Not Secure http://0.0.0.0:8000/sourcemap.html
Print
Elements Console Sources Network Performance Memory Application Security
Page Filesystem
top
0.0.0.0:8000
__target__
org.transcrypt.__runtime__.js
org.transcrypt.__runtime__.py
sourcemap.js
sourcemap.py
sourcemap.html
sourcemap.py

```python
def print_stuff():
    console.log("Native JS console.log call")
    print("Python print")
    console.invalid_method("This will be an error")
```

React - Installation & Setup

```
$ npm init -y
$ npm install parcel-bundler --save-dev
$ npm install parcel-plugin-transcrypt --save-dev
$ npm install react@16 react-dom@16
```

There is a file in the current version of the plug-in requires a patch:

./node_modules/parcel-plugin-transcrypt/asset.js

In that file, the path for loading the Parcel Logger module need to be changed from this:

```
const logger = require('parcel-bundler/src/Logger');
```

to this:

```
const logger = require('@parcel/logger/src/Logger');
```

React - Hello World

pyreact.py

```
# Load React and ReactDOM JavaScript libraries into local namespace
React = require('react')
ReactDOM = require('react-dom')

# Map React javaScript objects to Python identifiers
createElement = React.createElement
useState = React.useState

def render(root_component, props, container):
    """Loads main react component into DOM"""

    def main():
        ReactDOM.render(
            React.createElement(root_component, props),
            document.getElementById(container)
        )

    document.addEventListener("DOMContentLoaded", main)
```

React - Hello World

hello_react.py

```
from pyreact import useState, render, createElement as el

def App():
    val, setVal = useState("")

    def say_hello():
        setVal("Hello React!")

    def clear_it():
        setVal("")

    return [
        el('button', {'onClick': say_hello}, "Click Me!"),
        el('button', {'onClick': clear_it}, "Clear"),
        el('div', None, val)
    ]

render(App, None, 'root')
```

React - Hello World

hello_react.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <script src="hello_react.py"></script>
    </head>
    <body>
        <div id="root"></div>
    </body>
</html>
```

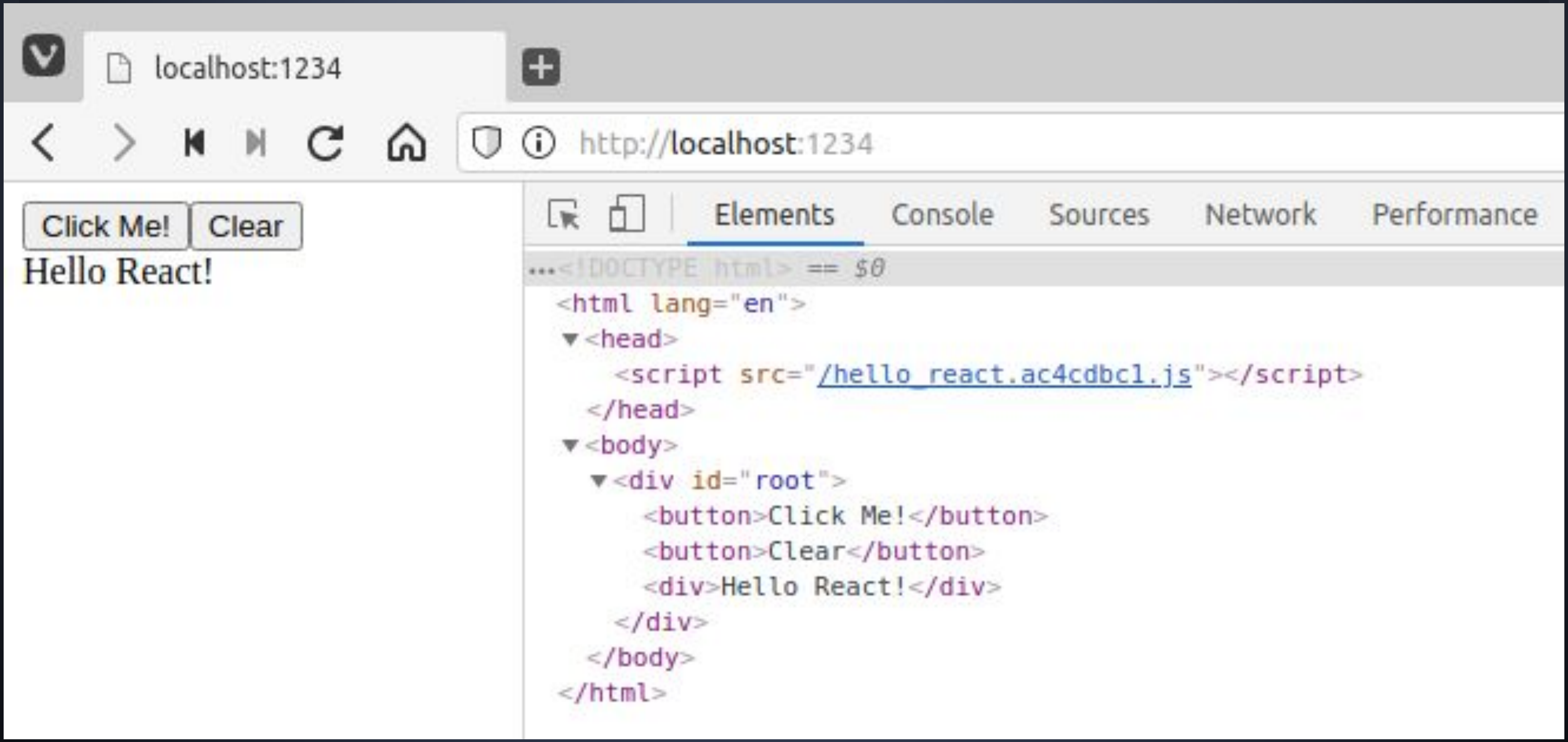
```
(venv) $ npx parcel --log-level 4 --no-cache hello_react.html
```

Parcel will automatically start up a development web server at:

    http://localhost:1234

React - Hello World

React generated content gets added into the DOM at the root node:



```
...<!DOCTYPE html> == $0
<html lang="en">
▼<head>
    <script src="/hello_react.ac4cdbcl.js"></script>
  </head>
▼<body>
  ▼<div id="root">
      <button>Click Me!</button>
      <button>Clear</button>
      <div>Hello React!</div>
    </div>
  </body>
</html>
```

React - App

Add build scripts to package.json:

```
"start": "NODE_ENV=development parcel --log-level 4  index.html --out-dir 
dist/dev",
"build": "NODE_ENV=production parcel --log-level 4 build  index.html 
--no-source-maps --out-dir dist/prod",
```

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>React to Python</title>
        <script src="app.py"></script>
    </head>
    <body>
        <div id="root"></div>
    </body>
</html>
```

React - App

app.py

```
from pyreact import useState, render, createElement as el

def ListItems(props):
    items = props['items']
    return [el('li', {'key': item}, item) for item in items]

def App():
    newItem, setNewItem = useState("")
    items, setItems = useState([])

    def handleSubmit(event):
        event.preventDefault()
        # setItems(items.__add__(newItem))
        setItems(items + [newItem])  # __:opov
        setNewItem("")

    def handleChange(event):
        target = event['target']
        setNewItem(target['value'])
```

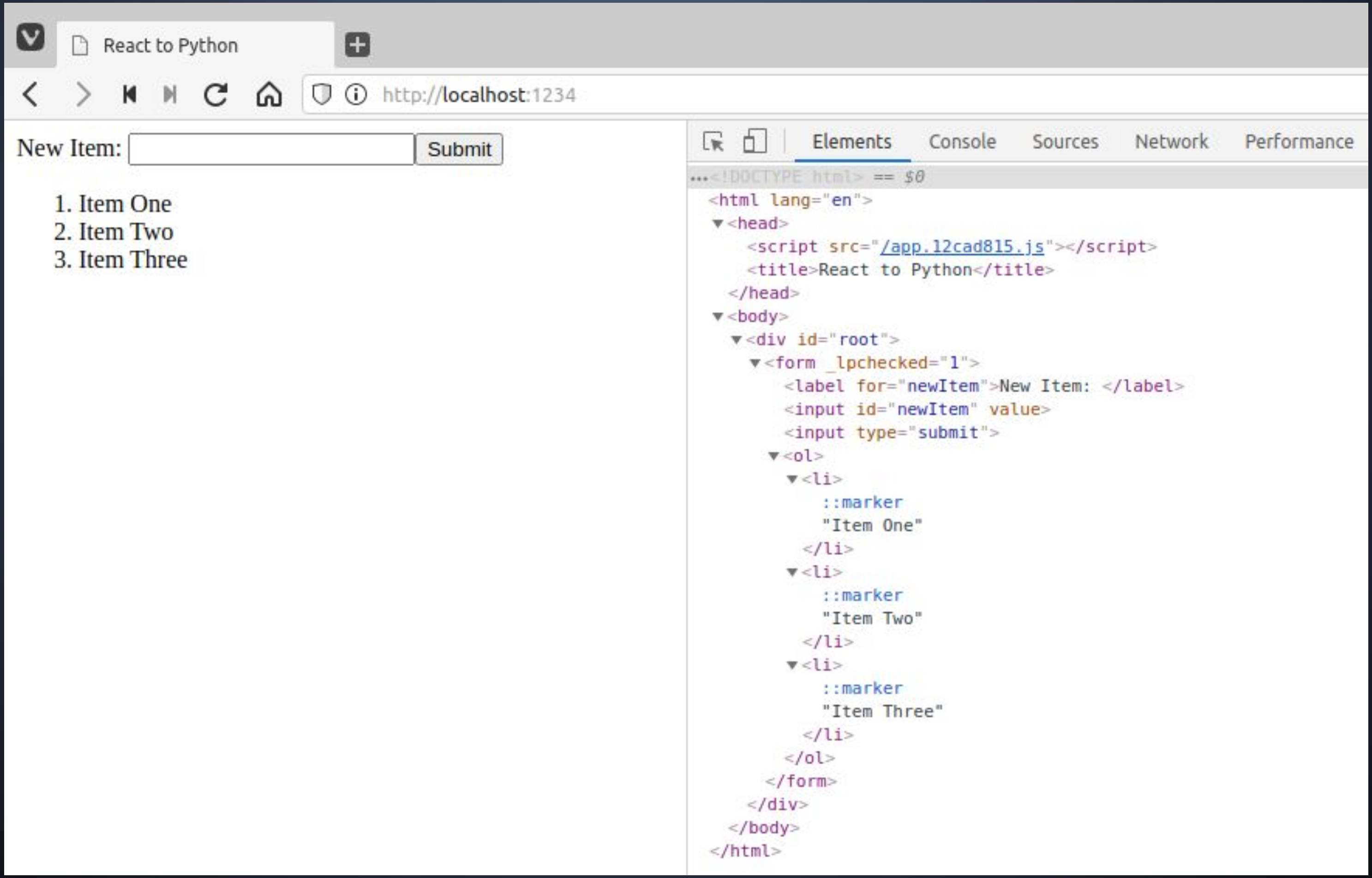
React - App (continued)

```
    return el('form',  {'onSubmit': handleSubmit},
              el('label', {'htmlFor': 'newItem'}, "New Item: "),
              el('input', {'id': 'newItem',
                           'onChange': handleChange,
                           'value': newItem
                          }
                ),
              el('input',  {'type':  'submit'}),
              el('ol', None,
                  el(ListItems, {'items': items})
                )
             )


render(App, None, 'root')
```

```
(venv) $ npm start
```

React - App

React dynamically inserts HTML elements into the DOM:



New Item: [ ] Submit

1. Item One
2. Item Two
3. Item Three

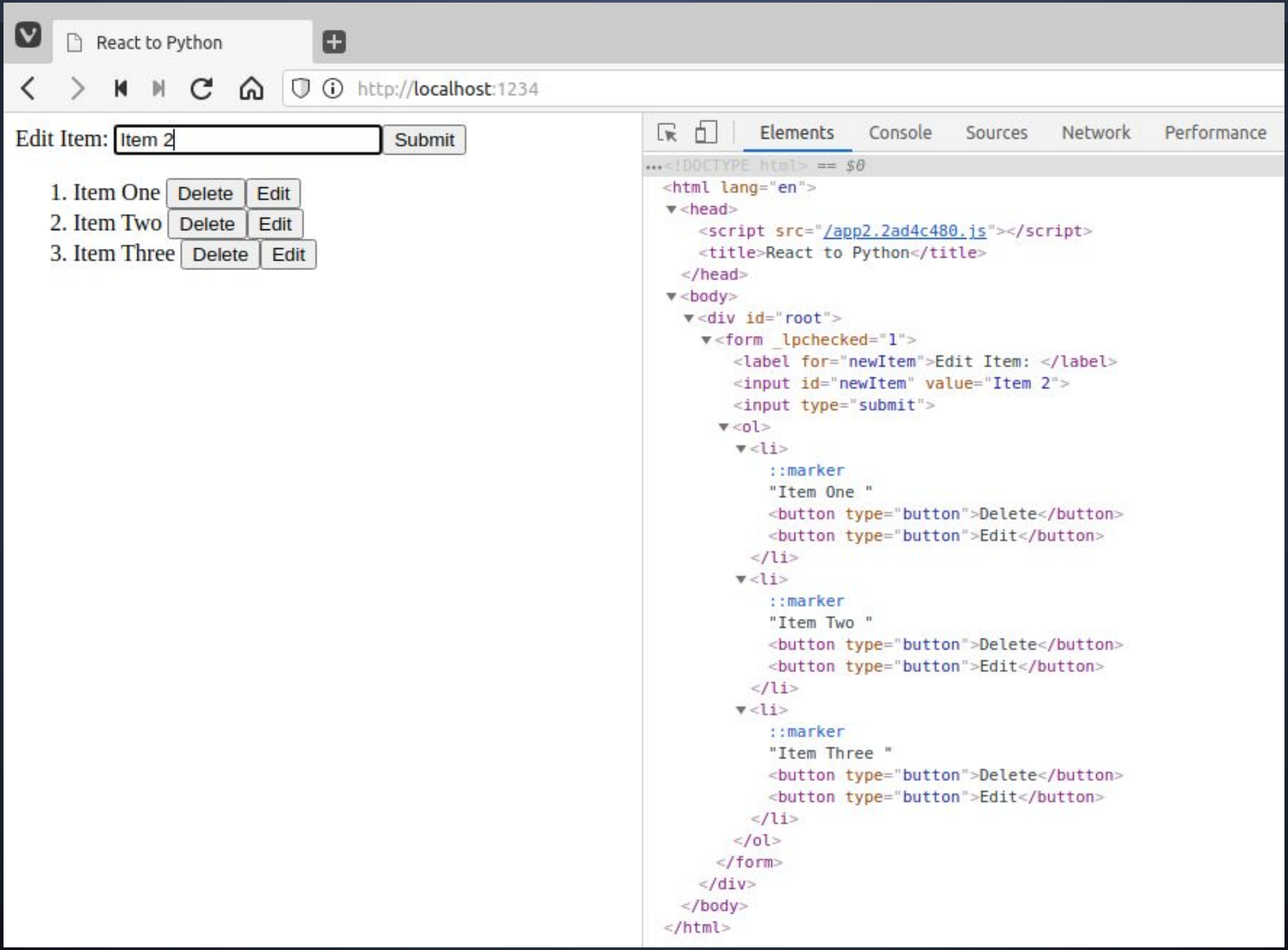
```
...<!DOCTYPE html> == $0
<html lang="en">
 ▼<head>
    <script src="/app.12cad815.js"></script>
    <title>React to Python</title>
  </head>
 ▼<body>
  ▼<div id="root">
   ▼<form _lpchecked="1">
      <label for="newItem">New Item: </label>
      <input id="newItem" value>
      <input type="submit">
     ▼<ol>
      ▼<li>
         ::marker
         "Item One"
       </li>
      ▼<li>
         ::marker
         "Item Two"
       </li>
      ▼<li>
         ::marker
         "Item Three"
       </li>
      </ol>
    </form>
   </div>
  </body>
</html>
```

React - App2

```
from pyreact import useState, render, createElement as el

def App():
    newItem, setNewItem = useState("")
    editItem, setEditItem = useState("")
    items, setItems = useState([])

    def handleSubmit(event):
        event.preventDefault()
        if editItem:  # In edit mode
            new_list = list(items)  # Make a copy
            new_list[new_list.index(editItem)] = newItem
            setItems(new_list)  # Update our state
        else:  # In add mode
            # setItems(items.__add__(newItem))
            setItems(items + [newItem])  # __:opov
        setEditItem("")
        setNewItem("")

    def handleDelete(item):
        new_list = list(items)  # Make a copy
        new_list.remove(item)  # Remove the specified item
        setItems(new_list)  # Update our state

    def handleEdit(item):
        setNewItem(item)  # Set the new item value
        setEditItem(item)  # Set the edit item value

    def handleChange(event):
        target = event['target']
        setNewItem(target['value'])
```

```
    def ListItem(props):
        item = props['item']

        return el('li', None,
                  props['item'] + " ",
                  el('button', {'type': 'button',
                                'onClick': lambda: handleDelete(item)
                                }, "Delete"
                     ),
                  el('button', {'type': 'button',
                                'onClick': lambda: handleEdit(item)
                                }, "Edit"
                     ),
                  )

    def ListItems():
        return [el(ListItem, {'key': item, 'item': item}) for item in items]

    return el('form',  {'onSubmit': handleSubmit},
              el('label', {'htmlFor': 'newItem'},
                 "Edit Item: " if editItem else "Add Item: "
                 ),
              el('input', {'id': 'newItem',
                           'onChange': handleChange,
                           'value': newItem
                           }
                 ),
              el('input',  {'type':  'submit'}),
              el('ol', None,
                 el(ListItems, {'items': items})
                 )
              )

render(App, None, 'root')
```

React - App2



Edit Item: Item 2 Submit

1. Item One Delete Edit
2. Item Two Delete Edit
3. Item Three Delete Edit

```
...<!DOCTYPE html> == $0
<html lang="en">
 ▼<head>
    <script src="/app2.2ad4c480.js"></script>
    <title>React to Python</title>
  </head>
 ▼<body>
  ▼<div id="root">
   ▼<form _lpchecked="1">
      <label for="newItem">Edit Item: </label>
      <input id="newItem" value="Item 2">
      <input type="submit">
     ▼<ol>
      ▼<li>
         ::marker
         "Item One "
         <button type="button">Delete</button>
         <button type="button">Edit</button>
       </li>
      ▼<li>
         ::marker
         "Item Two "
         <button type="button">Delete</button>
         <button type="button">Edit</button>
       </li>
      ▼<li>
         ::marker
         "Item Three "
         <button type="button">Delete</button>
         <button type="button">Edit</button>
       </li>
      </ol>
    </form>
   </div>
  </body>
</html>
```

Resources - Book

React to Python

- Setting up the required developer environment tools
- Creating CRUD Forms
- Asynchronous requests with Flask REST service
- Basics of using Material-UI
- Single Page Applications
- User session management
- View Routing
- Incorporating Google Analytics into your application
- Demo project (https://rtp.jennasys.com/)

Resources - Links

- Transcrypt Site:
  https://www.transcrypt.org
- Transcrypt GitHub:
  https://github.com/qquick/Transcrypt
- React to Python Book:
  https://pyreact.com
- Presentation Source Code:
  https://github.com/JennaSys/rtp_demo

Discussion

jsheehan@jennasys.com

https://github.com/JennaSys

Twitter: @JennaSys