

# RESTFUL SPA OVERVIEW

Putting the pieces together

Server Side:

- `express` for service endpoints and static files
- Separation of concerns in service side
- using a uuid module for uids

Client Side:

- webpack and babel for ES6 imports on client side
- Separation of concerns in client side
- requiring a login (no passwords)

# A TODO LIST

- The site requires the user to login
- Each user has a todo list
- Displays a list of todo items
- Each item can be toggled as done
- Each item can be deleted
- New items can be added

# CONFIGURING THE SERVER SIDE

- create a new npm package (`npm init`)
- installing `express`, `cookie-parser`, `uuid`
- create `public/` dir
- create static HTML
- create static CSS
- create temp static JS
- create `server.js`
- create `todo.js` to hold non-web logic and state

# INSTALLING WEBPACK AND BABEL

```
# babel
npm install --save-dev @babel/core
npm install --save-dev @babel/preset-env
# webpack
npm install --save-dev webpack
npm install --save-dev webpack-cli
# connect the two
npm install --save-dev babel-loader
```

or

```
npm install --save-dev babel-loader @babel/core @babel/preset-env
webpack webpack-cli
```

# CREATE A WEBPACK.CONFIG.JS

```
const path = require('path');
module.exports = {
  mode: 'development',
  entry: './src/todo.js',
  devtool: 'source-map',
  output: {
    filename: 'todo.js',
    path: path.resolve(__dirname, 'public'),
  },
  // ...
}
```

# CREATE A WEBPACK.CONFIG.JS (CONTINUED)

```
// ...
},
module: {
  rules: [
    {
      test: /\.js$/,
      exclude: /node_modules/,
      use: {
        loader: 'babel-loader',
        options: { presets: ['@babel/preset-env'] },
      }
    }
  ]
},
};
```

# CONNECT THE PIECES

To transpile and bundle the `src/todo.js` and anything it imports into `public/todo.js`:

**Do this anytime the `src/*` files change**

```
npx webpack
```

To run the server:

**Do this anytime the `/*.js` files change**

```
node server.js
```

# USING A NEW MODULE

We have mentioned that you should use a UUID/GUID when you need a unique, unpredictable value

- Why do you want an unpredictable value?
- Why not just a random value?

See **<https://github.com/tc39/proposal-uuid#faq>**

Some of the UUIDs ARE just (pseudo) random numbers, but they are good ones.

Other UUID algorithms try for uniqueness

**<https://github.com/uuidjs/uuid>**



# POLLING

The web request/response cycle:

- means the client has to ASK for an update
- ...even if there isn't one yet

This can feel (and be) inefficient

- But is also very common
- We'll do basic polling because it's simple
- ...not because it is better

# POLLING METHODS

- Polling
  - periodic web requests
- "Long Polling"
  - Server keeps res open, trickling empty data
  - Server finishes res once there is an update
  - Client immediately opens new request
- Websockets
  - Not HTTP
  - A different protocol started *from* HTTP
  - Allows server "push" actions