Building a Full Stack Application with React and Express

STRUCTURE OF FULL STACK APPLICATIONS

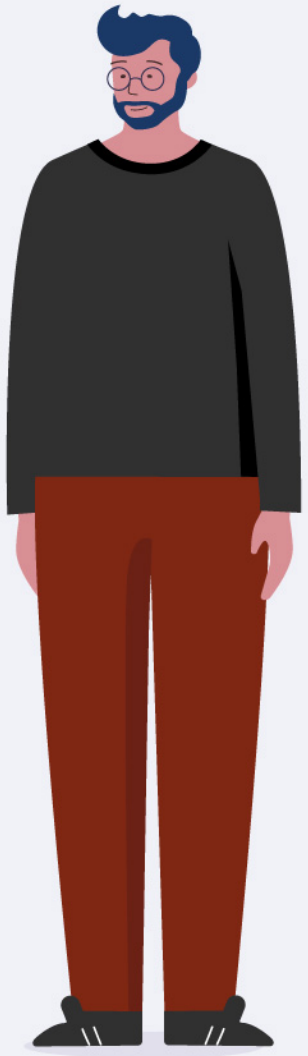
Daniel Stern
CODE WHISPERER

http://github.com/danielstern

Course Roadmap

- Understand what full stack applications are
- Learn about the unique capabilities of the front end and back end of an application
- Create a front end application using React and Redux
- Connect it to a back end application using Express and MongoDB
- Application deployment

Building Full Stack Applications: A Scenario



Dev's boss, Mr. C. Eeyo, asks Dev, a computer programmer, to help him solve a business problem.

The company has a terrific new product that will be sold on a brand-new website.

"Dev," says Mr. Eeyo, "Can you build this website?"

The website needs to have a simple, clear interface that makes it fun and easy for your customers to buy your product.

The website *also* needs to validate credit cards and store confidential user information.

Why Do Businesses Need Full Stack Applications?

Why Businesses Need Full Stack Applications

Users expect a fast, fluid experience (dedicated front end component)

User-created content must still be there next time (data persistence)

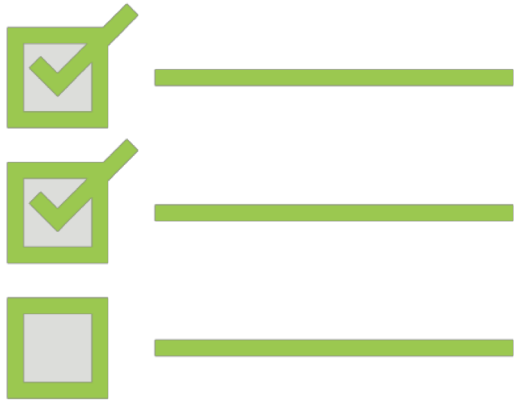
Processing payments and managing user data are critical to generating revenue

What are Full Stack Applications?

Full Stack Application

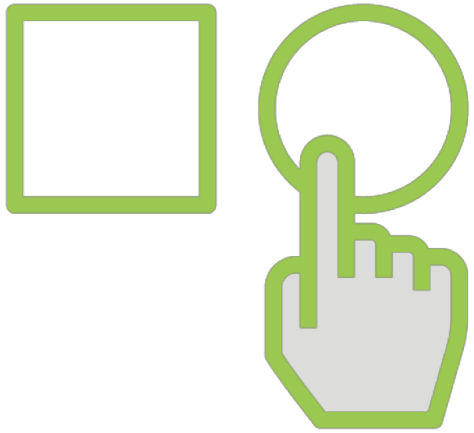
An application that can be viewed in a web browser with an accompanying means of persisting data and working with private information that exists on a server.

Front End
(Also called the “client”)



Comprises pages, buttons and forms for the user to interact with

Concerned with user experience: design, polish, speed, etc.



Can change appearance for different devices (“reactivity”)

JS
HTML
CSS

Consistently made up of JavaScript (JS), HTML and CSS

Client Limitations
(Why Do We Need A Back End?)

Client can't persist data reliably

Not possible to hide secrets on client

No control over end user's hardware (may be too slow to handle necessary calculations)

“Pay no attention to the man behind the curtain.”

The Wizard of Oz

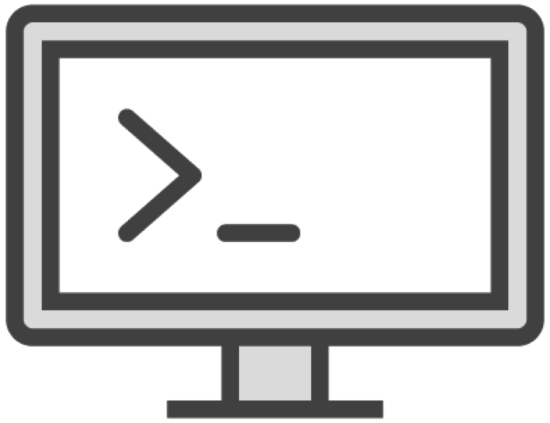
Back End
(Also called the “server”)

Persists user experience by storing data permanently in databases

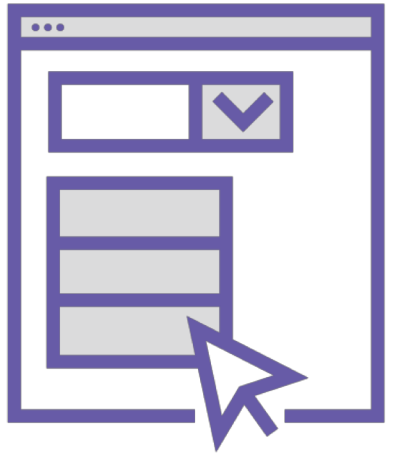
Conceals information (such as secret keys, other users’ data) from end user

Communicates with third-party APIs, i.e. payment processors

Server Limitations
(Why Do We Need a Client?)



Applications without client are difficult to use without technical knowledge (i.e. BASH commands, SQL queries)



Web browsers allow for images, animation and styling, creating a favorable impression of your organization

What Comprises a Back End?

Provides a place for data to go. When databases do their job, they are very boring and predictable

DATABASE

Provides a place to store secret business logic or authorization, and to communicate with the database

SERVER

What Comprises a Full Stack Application?

FRONT END (CLIENT)

Comprises all the things the end user sees or interacts with

Responsible for facilitating a smooth user experience and achieving business goals

Built with HTML, CSS, JavaScript and frameworks such as *React*

BACK END (SERVER)

Almost all back end features are invisible to the end user

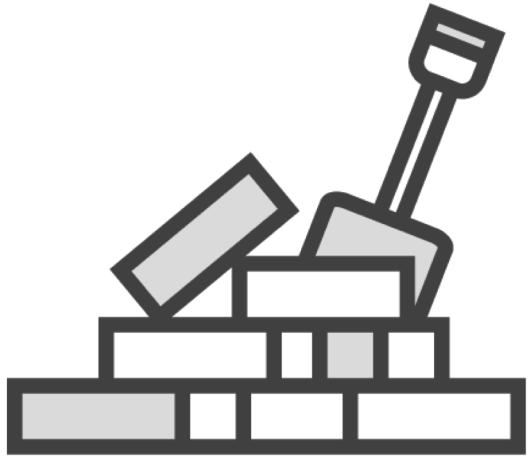
Responsible for very reliable data persistence and very high uptime

Built using JavaScript (with Node) or PHP, C#, Java or many other languages

JavaScript and Full Stack Applications

Advantages of JavaScript-based Back End vs. Other Languages

Developers can be hired flexibly



Constants and formulas may be shared directly between front and back end

Server can more easily pre-render pages or assist with calculations

Limitations of Using a JavaScript Back End

Sluggish processing, greatly limited and slow math capabilities (no integer math, only floating-point)

Some languages have a larger selection of certain libraries (i.e., data science and Python)

Typically more challenging and expensive to deploy than Java, PHP, etc.

Understanding Security Considerations

Some Code Must Not Be Visible to End User

```
const VERY_SECRET_KEY = `tmoltuae-42`;

function TOP_SECRET_BUSINESS_FORMALA(x,s){
  return Math.sqrt (
    x * (8 * s - 16) +
    Math.pow(s - 4, 2)) +
    s - 4 / (2 * s - 4);
}
```

Disclaimer:
I am not a security expert.

Before implementing security procedures to protect your **actual data**, please consult the relevant course on security here at Pluralsight

pluralsight.com/authors/troy-hunt

Understanding the Client and Server

Client-Server Workflow

End user accesses desired page, needs certain data

HTTP Request is sent from client to server

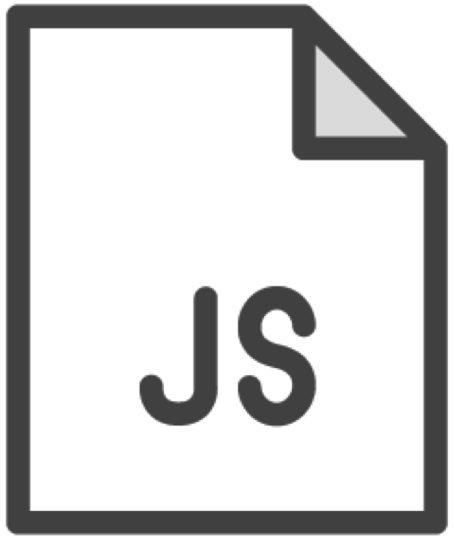
Server authenticates user through opaque process

Server accesses database and gets required data (secure)

Server responds to HTTP request with desired data

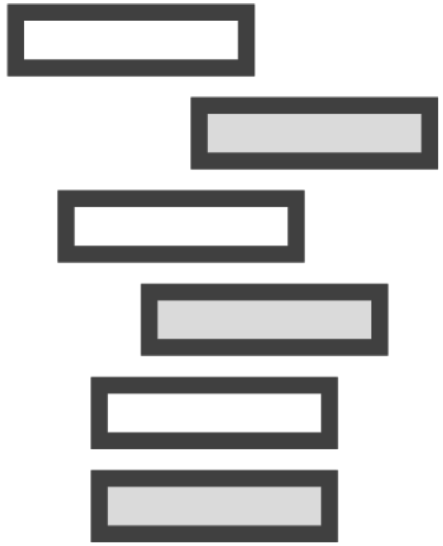
Data is rendered into components on end user's client

Bringing the Client and Server Together with JavaScript



Front End Apps are virtually all written with JS

Possible to write back end using other language, but not optimal

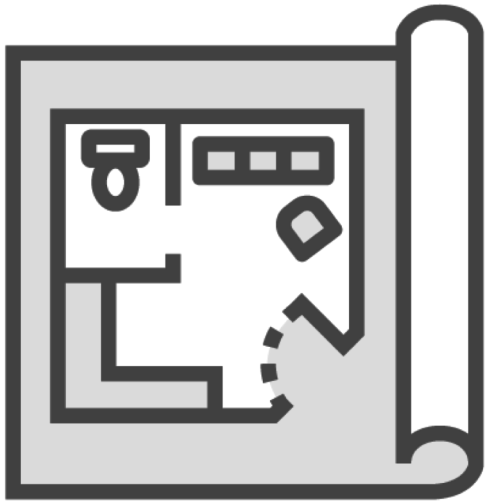


Modern tools (Node) allow front and back end to be written in same language

Node-based full stack app can be approached as a single application

A Look at the Finished Application

Finished Application Preview – Front End



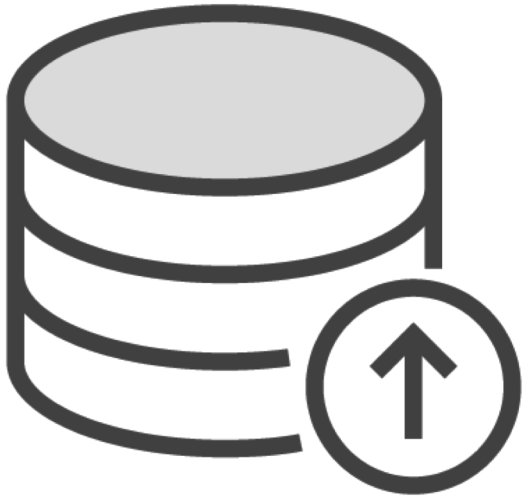
React and Redux used to display components

Consists of interactive forms and lists that reflect user's unique data

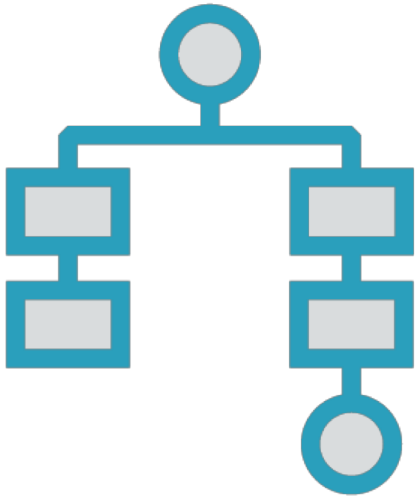
http://

URL determines which components to display (routing)

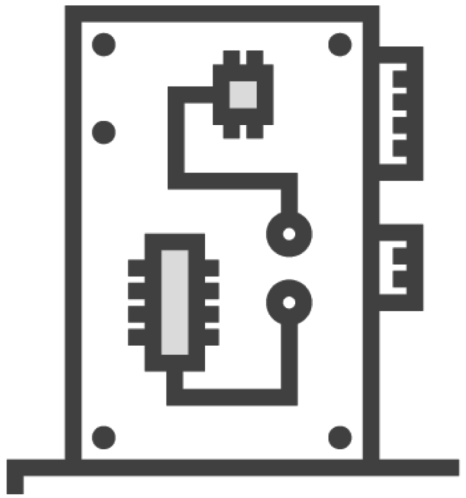
Finished Application Preview – Back End



MongoDB stores data persistently in non-relational database



Express serves a static HTML page with the application



REST API lets us work with the database via HTTP

Coming up in the Next Module...

Set up Webpack and Babel to support our application (ES6 and JSX)

Create a Redux store to update and manage local state

Assemble website out of React components

Add styling with Bootstrap