Where community & Technology meet

Senior Frontend-Fullstack Engineer

A leap into a lead frontend engineer position

Abstract

There is so much knowledge out there, and it's not easy to stay on top of the game. In our new academy, Take a Leap, we put all our knowledge together for you, in small digestible bites, so that you can take your career forward, and be the best.

The “Senior Frontend Engineer” course is designed for software engineers with a passion for web frontend architecture and development. We designed this course so that it will cover all major aspects of frontend development and then some. A successful graduate will be able to tackle any challenge and lead the development of a front end project based on React to successful deployment in a production environment.

Requirements

If you have at least ***2-3 years as a software developer***, you have a passion for front end development, and you want to confidently perform a significant career leap, then this course is for you.

FullStack Developers Israel
Powered by: Tikal Knowledge

Syllabus

1. Introduction to WebApps

Abstract

> WebApps are probably the most common and accessible interactive applications. In order to apply good and solid architecture, we will set the foundations by diving into the main concepts and technical aspects of Web Apps

Goals

1. Understand the history and evolution of web apps
2. Understand the basic mechanisms and architecture of a web app
3. Understand protocols and communication patterns implemented in web-apps
4. Realize the Technical difficulties involved in creating a web app

Topics

1. Server-Client concepts
2. Server Rendered Pages and SSR
3. Single Page Applications
4. REST
5. WebSockets
6. Typescript

2. Building web apps with React

Abstract

> React has become the most popular library for building modern web-apps. The advantages of using React as a foundation for web-apps are many and include a low learning curve, flexibility, ability to choose different supporting tools to create a tailor-fitted stack for almost any web-app

Goals

1. Learn how to set up a new React based project
2. Understand the concepts of reactive based architecture
3. Create high-quality component / container compositions

Topics

1. Components
2. Containers
3. Using Hooks
4. Styled Components
5. Routing
6. State management
7. High Order Components (HoC)
8. Context API
9. IoC (“Inversify.js”)

3. Testing and Automation

Abstract

> In this section, we will learn how to write good and effective tests. Writing tests correctly reduces development time, increases code quality, reduces bugs and gives developers greater confidence in their code.

Goals

1. Understand concepts and terminology related to testing and automation
2. Be able to design and write high-quality automated tests

Topics

1. Configuration and executing Jest and Enzyme
2. Using mocks
3. Snapshot testing
4. Integration and behavior testing
5. e2e tests

4. Advanced styling with CSS

Abstract

CSS is a technology that is often misused and overlooked. It is very simple to get started but if you truly master it it will make your life so much easier when you need to give your web-app that professional look and feel.

Goals

1. Understand how to design and implement the UI
2. Learn how to use app layouts correctly
3. Learn how to give any app a professional look and feel

Topics

1. Responsive design
2. Layouts with Flexbox and Grid
3. Transitions and animations
4. Working with SVG
5. Post-processing and SASS

5. Web App State Management using Redux

Abstract

Managing a complex web-app with many moving parts can be complicated and if managed incorrectly can lead to spaghetti code and poor performance. Redux is the most popular library that uses the “flux unidirectional data flow” pattern to implement event-driven state management for web-apps (and servers).

Goals

1. Understand the importance of good state management for web-apps
2. Understand where and how to store and mutate the state’s data
3. Be able to implement a high performing, easily maintainable state mechanism with React

Topics:

1. Redux Action and Reducers
2. Working with asynchronous actions
3. Using Middleware
4. Error handling and troubleshooting

6. Web App State Management using Mobx

Abstract

> Although Redux is an extremely popular library, many prefer Mobx. While building around the same principles as Redux, Mobx requires less boilerplate and to some may seem more intuitive and natural.

Goals

1. Understand the difference between Redux and Mobx
2. Understand how to use Mobx for state management
3. Be able to implement a high performing, easily maintainable state mechanism with Mobx

Topics:

1. Observers
2. Computed Values
3. Actions
4. Async Actions, and Flows
5. Utility functions

7. Web App Authentication

Abstract

> Most web-apps are not publically accessible and require some sort of authentication to prevent abuse. There are several ways to authenticate and authorize users. We will focus on token-based access and authentication which is the most common and provides a high degree of security.

Goals

1. Understand the importance of authentication and authorization
2. Be able to apply an SSO-based authentication with several authorization roles to a web-app

Topics

1. Working with SSO and JWT
2. Public and private routes
3. Working with Metadata and Roles

8. Build and Bundling

Abstract

A critical part of the development process that is often overlooked by front end developers is the build process and the continuous integration of the product. We are often so pleased that the app is working locally that we overlook the other environments that the product is meant to run on. We will focus on the building of the app as part of the CI process using Webpack, Rollup, and Parcel.

Goals

1. Understand the differences between the different bundling tools
2. Successfully build production-grade bundles for any web-app

Topics:

1. Concepts: Entry-points, Tree-shaking, modules, code-splitting
2. Webpack
3. Rollup
4. Parcel

9. Progressive Web Apps (PWA)

Abstract

As more and more of our users use mobile devices as a primary and necessary tool for interacting with our system, the traditional approach of mobile-friendly web apps sometimes just doesn't cut it. Progressive Web Apps is a paradigm and technology that optimises web applications for mobile and offline consumption.

Goals

1. Understand the key concepts of PWA
2. Implement effective and efficient web app enhancements to support mobile and offline operations

Topics

1. What and why?
2. Architectural concepts - The App Shell Model
3. Service Workers and content cache
4. Distribution (including TWA)
5. Push notifications

10. Building a server

Abstract:

> A good Front End engineer knows his way in the browser. A great one also supplements that knowledge with Server-side development to close an end-to-end (full-stack) development capabilities. Developing servers with JAvascript has never been more fun and appealing to client-side developers.
> We will learn the basics of server-side development with NodeJS. This server can be used just as a server-for-frontend or as a backend for a complex system.

Goals

1. Understand how the backend works in conjunction with the front-end
2. Be able to develop a server to support fetching and storing information
3. Know how to Increase security by applying authentication and authorization on the server-side

Topics:

1. HTTP Protocol overview
2. Introduction to NodeJS
3. Building a RESTful web-server with ExpressJS
4. Express middleware
5. Session management
6. Socket.io
7. Authentication
8. Authorization

11. MongoDB

Abstract

> One of the key responsibilities of a server is to store and organize data. For the past 10 years, NoSQL servers in general and MongoDB have become an industry standard to store and index high volumes of unstructured or loosely structured data. We will get to know how to use MongoDB as a primary data store for our application.

Goals

1. Understand how a NoSQL, document database should be used, where it shines and where it lacks
2. Understand how to structure data with a document-based datastore

3. Learn to quickly and efficiently access data stored in the database

Topics:

1. Introduction to MongoDB
2. Setting up a local and hosted DB server
3. Working with the MongoDB client
4. CRUD Operations
5. Indexing data
6. Aggregation Framework
7. Map-Reduce

12. GraphQL

Abstract

GraphQL is an exciting new technology for exposing server information for increased flexibility, efficiency and stability. We will learn how to take advantage of this technology in order to support complex fetching and mutation operations between the client and the server

Goals

1. Learn how to set up a GraphQL server for exposing information to the client
2. Understand how to build a type-safe schema for organizing and communicating information

Topics:

1. The GraphQL Schema
2. Writing query endpoints
3. Writing mutations
4. GraphQL client
5. Cache optimization
6. Graphql Subscriptions

13. Deployment with Docker

Abstract

We will learn how using containers in general and Docker specifically can ensure successful, reproducible and stable deployments of applications.

Goals

1. Understand the Dockerfile configuration
2. Successfully dockerize an application or server
3. Successfully deploy a dockerized application to a hosting service
4. Working with production environments

Topics:

1. Introduction to Docker - Containers, Images, Networking
2. Docker file configuration
3. Docker Compose

READ MORE ON OUR WEBSITE

FullStack Developers Israel
Powered by: Tikal Knowledge