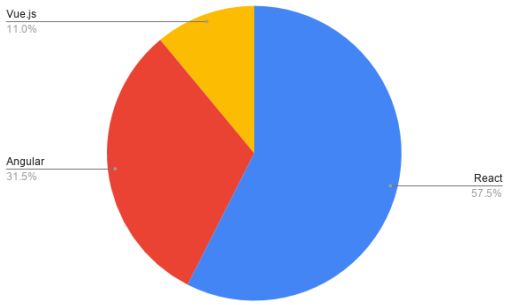
Topic 6 - Angular

Introduction to Client-side Framework

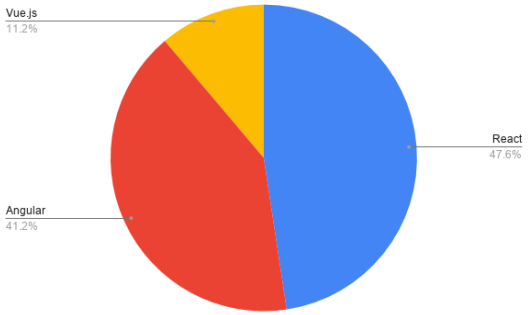
- What is Angular
- Architecture
- Components
- Directives
- Pipes
- Services
- HTTP
- Routing

Popular JavaScript Frameworks

Stack overflow searches 2021

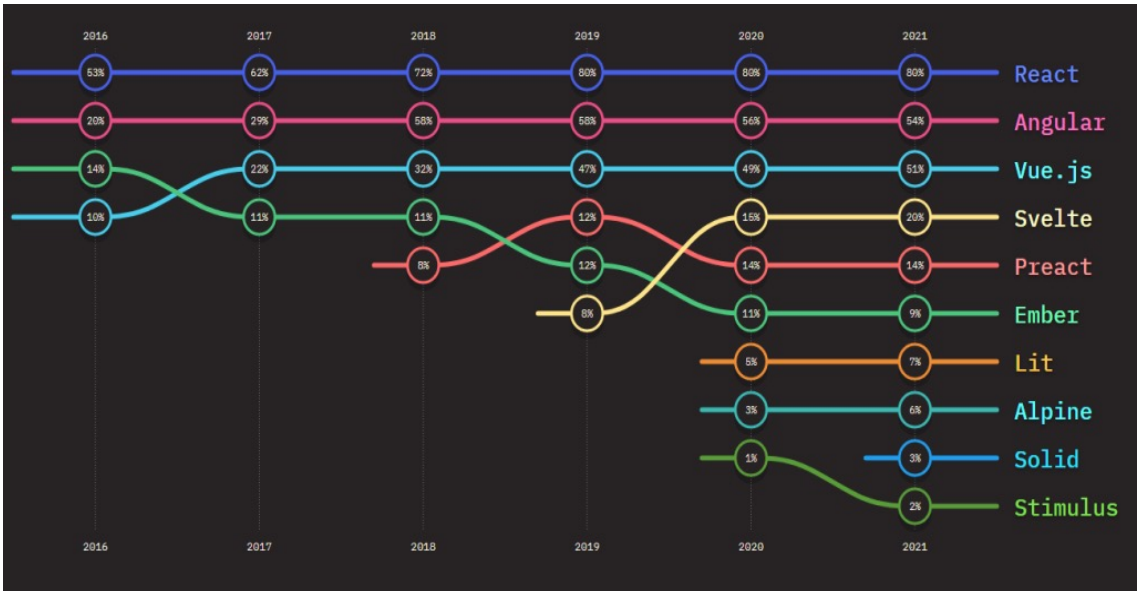


Job postings 2021



Source: medium.com

Popular JavaScript Frameworks



Source: stackdiary.com

JavaScript Frameworks

Other JavaScript Front-End Frameworks

- **Ember** forces developers to adopt a known and well-regarded approach to structuring and implementing a web application. It uses a variant of the MVC pattern.

- **Vue** has many similarities to Ember and Angular. It is a fast and light-weight web framework intended for smaller scale projects

- **React** is a library developed by Facebook. Unlike Ember and Angular, React is not a complete MVC-like framework; instead, it focuses on the view.

What is Angular?

Overview and History

What is Angular

- JavaScript-based **open-source front-end web application framework.**
- Developed and Maintained by Google
- Components organized into Modules
  - Components contain views
  - Components use services
- Two main versions
  - AngularJS – version 1
  - Angular – currently version 14

Model and View

MVC

- The high-level architecture of Angular is based on models and views which is a structural design pattern that separates objects into three distinct groups:
  - **Models** hold application data. They're usually basic interfaces or simple classes.
  - **Views** display visual elements and controls on the screen. It refers to anything that is rendered on the screen (i.e. browser)
  - **Everything else** transforms model information and processes them into values that can be displayed on a view
    - Usually a class.
- Separate the responsibility of different classes

Features

- Data binding – Data in views and models stay in perfect synchronization.
- Templates
- CLI – command line interface
  - provides a CLI or command line interface.
  - Small application that you install and load with your terminal that makes it easier to create the connections between components by automatically inserting code.

Architecture

Angular Components

- Angular works on components
- Loads a root component (known as the bootstrap call)
  - Looks inside the root component to see if any nested components
  - This process repeats until the whole hierachary of components are rendered
  - The result is similar to the DOM
- A component in Angular contains a portion of HTML code and provides functionality to that portion.
  - Uses a component *class* where we can define application logic for the component.

Directives and Pipes

- Component = Directive + Template
- Directives – allows you programming constructs in HTML (extends HTML)
  - Structural – `ngFor`, `ngIf`
  - Attribute – `ngClass`
- Template (View) – is an HTML code fragment that tells Angular how to render the component
- Angular Pipes
  - Changes from data to another data (transforming)
  - `date`, `uppercase`, `lowercase`

Data Binding

- Data binding refers to automatic updates (synch) of data between variables and the view. This is implemented in Angular by:
  - Insertion of items using double curly braces `{{ }}`

    ```
    {{ person.name }}
    ```

  - Click events linked to DOM elements
  - Expressions and statements
  - Expression operators
  - Form Modules (data binding)

Services

- An Angular service is a JavaScript object and provides some very useful functions
- When we write components in Angular, it is good practice to set it up in a way that the class logic only consists of relaying data to and from the view and adding functionality to the view. For example, we can write a service for:

  getPeople, deletePeople, etc.
- We can put this code into a service and tell the component to link to it

Persistent Data

- Data persistence refers to keeping the data available even after you turn off your app.
- There are two ways this can be done
  - Using objects created in your Angular app
  - Locally – via localstorage
- On a server – via Http protocol API calls (more on this later)
  - Angular HTTP
    - XHR / JSONP to make an HTTP call using JSON object and listen for responses from the server

Routing

- A route typically refers to a feature in an application. For example, the ability to View, Edit, and create new People
- Server Routing solution
  - Create a new URL for each feature:

    ```
    https://some_domain.com/create
    https://some_domain.com/view/1
    https://some_domain.com/edit/10
    ```

- Client-side Routing solution in Angular
  - Configure route path to specific components
  - Persist variables in an application to pass them from component to component
  - sending to a server only when needed

Angular CLI

And the project folder structure

Command Line Interface

- Command line interface (CLI) – allows you to do common tasks in terminal
- Scaffolding
  - biased – will not set up the project exactly the way you like it but will be a base for an application
- Pre-requisite – install node, NPM and git

Angular CLI

- To install the Angular CLI, go to terminal and install globally using NPM

```
npm install –g @angular/cli
```

- Now we can use the angular CLI commands to create / alter our project

Angular CLI

- The CLI has several commands that you can use to create, serve, build, generate components, and much more
- `ng new <NAME>` – creates a new angular project along with the required basic files to get started.
- `ng serve` – runs in development mode – uses *webpack* which processes your code and runs a temporary live server that will listen for changes in your code
- `ng build` – processes your project and generates files that you can pass onto a server
- `ng g <TYPE> <NAME>` or `ng generate <TYPE> <NAME>` – generates Angular components, directives, etc. and adds them to your project.

Angular CLI

Useful CLI commands

Command	Purpose
`ng new`	Creates new Angular application
`ng serve`	Builds and runs application using webpack server
`ng eject`	Make webpack config files available to be edited
`ng generate component <name>`	Creates new component
`ng generate directive <name>`	Creates new directive
`ng generate module <name>`	Creates new module
`ng generate pipe <name>`	Creates new pipe
`ng generate service <name>`	Creates new service
`ng generate enum <name>`	Creates new enumeration
`ng generate guard <name>`	Creates new guard
`ng generate interface <name>`	Creates new interface

Understanding Angular folder structure

/ Folder

- Among others, a basic Angular project has the following useful files/folders under the root folder:
    - e2e
    - node_modules – folder containing dependencies from npm
    - src – folder where all your source code lives
    - dist (optional or can be changed) – destination folder of build
    - package.json
    - .gitigore
    - .editorconfig – configures editor (i.e. number of spaces for tabs, etc)
    - .tsconfig.json
    - angular.json – all metadata for your application

Understanding Angular folder structure

src Folder

- The src folder contains the files needed for your application, it contains
    - app – contains our code
    - assets – static files such as pictures, videos, and other resources.
    - environemnts
    - main.ts – this is the startup file for application
    - style.css – global style … styles that applies to all components
    - polyfills.ts
    - tests.ts – initializes the testing framework and the types of tests that we are going to run

Understanding Angular folder structure

app Folder

- app.component.html – this is the default template for your app
- app.component.ts – contains metadata for this component. For example:
  - Selector name ‘app-root’
  - Template info
  - Style info
  - Initializes data (export class `AppComponent` and makes it available to the rest of the application and the template)