Introduction to Web Application Development Technologies

1

Technology stack for web application development

- To develop a web application, you need to select the server, database, programming language, framework, and frontend tools. These web development technologies build on top of each other and are, in fact, collectively called a stack.
- Let's take a look at the following terminologies and concepts,
  - Frameworks
  - Databases
  - Client and server
  - Front-end and back-end
  - Client-side programming
  - Server-side programming
  - Web Application Architecture

2

- Frameworks
- Databases
- Client and server
- Front-end and back-end
- Client-side programming
- Server-side programming
- Web Application Architecture

Frameworks

- A **framework** is a big library or group of libraries that provides many services (rather than perhaps only one focused ability as most libraries do).
- Frameworks typically take all the difficult, repetitive tasks in setting up a new web application and either does them for you or make them very easy for you to do.
- Examples:
  - Django - a full-stack framework built using python
  - Ruby on Rails - a full-stack framework built using ruby
  - Node.js - a server-side javascript framework to produce dynamic web page content before the page is sent to the user's web browser.
  - AngularJS - a front-end javascript framework maintained by Google
  - Bootstrap - a UI (user interface) framework for building with HTML/CSS/Javascript
  - Laravel - a free, open-source PHP web framework, following the model–view–controller (MVC) architectural pattern.
  - ASP.NET - an open-source server-side web application framework developed by Microsoft.

3

- Frameworks
- Databases
- Client and server
- Front-end and back-end
- Client-side programming
- Server-side programming
- Web Application Architecture

Common web framework functionality

- Frameworks provide functionality in their code or through extensions to perform common operations required to run web applications. These common operations include:
  - URL routing
  - HTML, XML, JSON, and other output format templating
  - Database manipulation
  - Security against Cross-site request forgery (CSRF) and other attacks
  - Session storage and retrieval
- Not all web frameworks include code for all of the above functionality.
- For example, the Django web application framework includes an Object-Relational Mapping (ORM) layer that abstracts relational database read, write, query, and delete operations. However, Django's ORM cannot work without significant modification on non-relational databases such as MongoDB.
- Some other web frameworks such as Flask and Pyramid are easier to use with non-relational databases by incorporating external Python libraries. There is a spectrum between minimal functionality with easy extensibility on one end and including everything in the framework with tight integration on the other end.

4

- Frameworks
- Databases
- Client and server
- Front-end and back-end
- Client-side programming
- Server-side programming
- Web Application Architecture

Do I have to use a web framework?

- Whether or not you use a web framework depends on your experience with web development and what you're trying to accomplish.
- If you are a beginner programmer and just want to work on a web application as a learning project then a framework can help you understand the concepts listed above, such as URL routing, data manipulation and authentication that are common to the majority of web applications.
- On the other hand, if you're an experienced programmer with significant web development experience you may feel like the existing frameworks do not match your project's requirements. In that case, you can mix and match open source libraries with your own code to create your own framework.
- In short, whether or not you need to use a web framework to build a web application depends on your experience and what you're trying to accomplish. Using a web framework to build a web application certainly isn't required, but it'll make most developers' lives easier in many cases.

5

- Frameworks
- Databases
- Client and server
- Front-end and back-end
- Client-side programming
- Server-side programming
- Web Application Architecture

Databases

- Your web application needs a place to store its data, and that’s what a **database** is used for.
- There are two main types of databases: relational and non-relational databases
- Relational databases provides more structure which helps with making sure all the data is correct and validated.
- Non-relational databases (NoSQL) provides a lot of flexibility for building and maintaining applications.
- Examples:
  - MongoDB - an open-sourced NoSQL database.
  - PostgreSQL - a popular open-sourced SQL database.
  - MySQL - another popular open-sourced SQL database.
  - Oracle - an enterprise SQL database.
  - SQL Server - an SQL server manager created by Microsoft.

6

Client (or Client-side)

- This party *requests* pages from the **Server**, and displays them to the user. In most cases, the client is a **web browser**.
  - The **User** - The user *uses* the **Client** in order to surf the web, fill in forms, watch videos online, etc.
  - It's you and me when we visit http://google.com.
- Client can be desktop computers, tablets, or mobile devices.
- There are typically multiple clients interacting with the same application stored on a server.

Server (or Server-side)

- This party is responsible for **serving** pages.
- A server is where the application code is typically stored.
- Requests are made to the server from clients, and the server will gather the appropriate information and respond to those requests.

Typical Client-Server Model

1. The **User** opens his web browser (the **Client**).
2. The **User** browses to http://google.com.
3. The **Client** (on the behalf of the **User**), sends a request to http://google.com (the **Server**), for their home page.
4. The **Server** then acknowledges the request, and replies the client with some meta-data (called *headers*), followed by the page's source.
5. The **Client** then receives the page's source, and *renders* it into a human viewable website.
6. The **User** types something into the search bar, and presses Enter.
7. The **Client** submits that data to the **Server**.
8. The **Server** processes that data, and replies with a page matching the search results.
9. The **Client**, once again, renders that page for the **User** to view.

Front-end & Back-end

- The front-end is comprised of HTML, CSS, and Javascript. This is how and where the website is shown to users.
- The back-end is comprised of your server and database. It’s the place where functions, methods, and data manipulation happens that you don’t want the clients to see.

Client-side programming

- Client-side programming is writing code that will run on the client, and is done in languages that can be executed by the browser, such as JavaScript.
- Client-side (i.e. frontend) web development involves everything users see on their screens. Here are the major frontend technology stack components:
  - Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS). HTML tells a browser how to display the content of web pages, while CSS styles that content. Bootstrap is a helpful framework for managing HTML and CSS.
  - JavaScript (JS). JS makes web pages interactive. There are many JavaScript libraries (such as jQuery, React.js - maintained by Facebook) and frameworks (such as Angular, Backbone) for faster and easier web development.

Typical uses of client-side programming

- **Client side** programming has mostly to do with the user interface, with which the user interacts. In web development it's the browser, in the user's machine, that runs the code, and it's mainly done in **javascript** etc. This code must run in a variety of browsers.
- **Its main tasks are:**
  - Validating input (Validation must be done in the server. A redundant validation in the client could be used to avoid server calls when speed is very critical.)
  - Animation
  - Manipulating UI elements
  - Applying styles
  - Some calculations are done when you don't want the page to refresh so often

Server-Side Programming

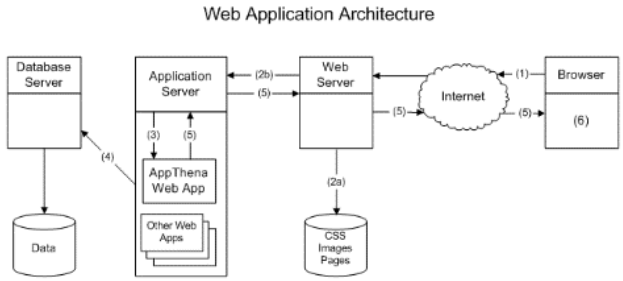
- The server side isn't visible to users, but it powers the client side, just as a power station generates electricity for your house.
- Server-side programming is writing code that runs on the server, using languages supported by the server.
- For server-side programming, they are used to create the logic of websites and applications. Frameworks for programming languages offer lots of tools for simpler and faster coding. Some of the popular programming languages and their major frameworks (in parentheses):
  - Ruby (Ruby on Rails)
  - Python (Django, Flask, Pylons)
  - PHP (Laravel)
  - C# (ASP.NET)

Typical uses of server-side programming

- Process user input.
- Display pages.
- Structure web applications.
- Interact with permanent storage (SQL, files).
  - Querying the database
  - Insert and update information onto the database

Web Application Architecture

- A web application is a client-server application, where there's a browser (the client) and a web server.
- The logic of a web application is distributed among the server and the client, there's a channel for information exchange, and the data is stored mainly on the server.
- Further details depend on the architecture: different ones place and distribute the logic in different ways.



15

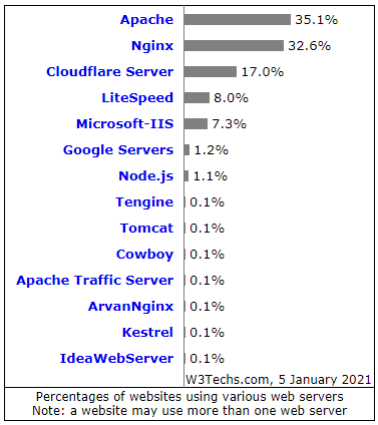
Web Server

- A web server is a system that delivers content or services to end users over the internet. A web server consists of a physical server, server operating system (OS) and software used to facilitate HTTP communication.
- **Web servers** are computers that deliver (serves up) Web pages. Every **Web server** has an IP address and possibly a domain name.
- The primary function of a web server is to store, process and deliver web pages to clients.

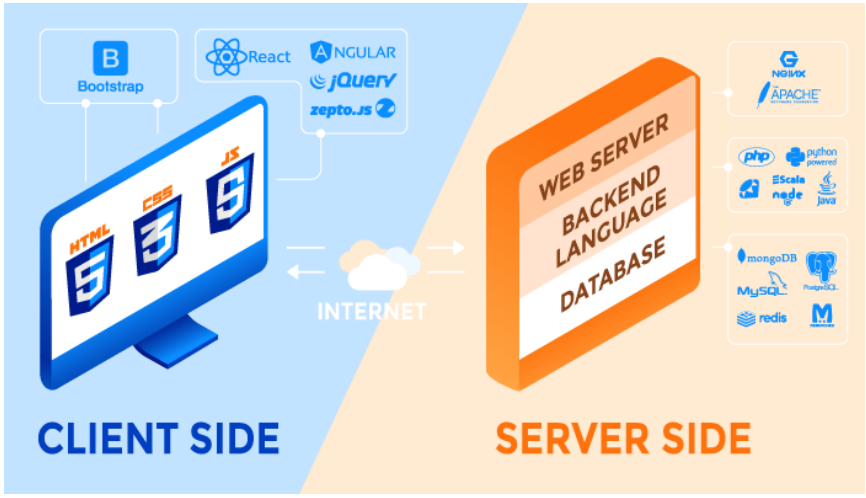
16

Web Server - market share

- For reference, below is the latest statistics of the *market share of all sites* of the top web servers on the Internet by W3Techs, as of January 2021.



17



https://www.upwork.com/hiring/for-clients/how-to-choose-a-technology-stack-for-web-application-development/

18