Introduction To Web Development

Basics of Web Interaction

- **HTML: Defines the structure and content(i.e text, links, images/videos, buttons, page dividors) of the page**
- **CSS: Adds style and flair to the page**
    - Also used to create websites that have cross browser compatibility / responsive design i.e compatible with multiple browsers and multiple devices such as PC, mobile devices, iPads etc.
- **JavaScript: Adds interactivity and dynamic content**
- Example: You use HTML to add a login button to a page, and CSS to style that button. You then use JavaScript to add log-in functionality to that button.
- Content displayed by websites can contain elements that are either
    - **Static :** Previously stored on the server
    - **Dynamic:** Generated each time they are requested by the client
        - Dynamic elements can involve information coming from other systems and applications, such as databases.
- Most websites contain static and dynamic elements to provide the best user experience.

- The environment for building websites is divided into **two** primary areas:
  - **Front-end :** Deals with everything that happens at the **client-side**
    - The front end usually deals with everything the user can see and interact.
  - **Back-end :** Deals with everything that happens on the **server-side** before the code and data are sent to the client.
    - The back-end coding usually handles the logic and functionality that make the website or app work, and the authentication processes that keep data secure.

Cloud Applications

- Cloud Applications are similar to Websites : they request content that a server returns.
- Cloud Apps are built to work seamlessly with a **cloud-based back-end infrastructure** i.e cloud-based data storage / data processing, and other Cloud services, making them very scalable and very resilient.

JavaScript Frameworks

- An application framework that is written in **JavaScript**.
- **Angular.js**
  - An open-source framework maintained by Google.
  - Allows websites to make the HTML pages quickly and efficiently.
  - It has built-in tools for routing and form validation.
- **React.js**
  - An open-source front-end library developed and maintained by Facebook.
  - Builds and provides components for a web page.
  - It is not a complete suite of tools.
    - Routing is not a part of this framework and will need to be added using a third-party tool.
    - Only helps build and drop components into a page.

Development Tools

A cloud application developer's workbench includes:

1. Version control

- When many developers are working on the same project, knowing in what order the changes were made is difficult.
- Version control systems keep track of **what** changes were made **when** and by **whom** and resolve any conflicts between changes.
  - Version control can be useful even when you are the sole contributor on a project.
- Gives you a way to revert to an older version of the code if something goes wrong and gives you some basic information about how the code developed over time.
- **Git and GitHub** are extremely popular for source code storage and management.
  - Git stores files in repositories where you can track changes, split code into different branches for more focused development, and then merge them back into the main body of code.

2. Libraries

- Libraries are **collections of code** to solve a specific problem or add a specific feature set.
    - Supplies the code so that you don't have to spend the time and energy creating one from scratch.
    - Being able to reuse code in this way makes developing your app much quicker and easier.
- Multiple libraries can be integrated into your existing project.
- You determine when to call the required method as needed.
    - The control returns to the program flow once the subroutine is finished.
    - When you use a library, you are in control.
- Here are some examples of code libraries:
    - **jQuery** is a JavaScript library that simplifies Dom manipulation.
    - **Email-validator** is a small library that checks an email address is correctly constructed and valid.
    - **Apache Commons Proper** is a repository of reusable Java components.

3. Frameworks

- A **skeleton** that you **can extend by adding your own code**.
- The framework you intend to use must be determined early and used right from the beginning.
    - New frameworks can’t be incorporated into an existing project.
- Your chosen framework dictates the architecture of your program and controls the program flow.
    - The framework determines which subroutines and methods will be called when.
    - Frameworks are less flexible than libraries, allowing you less control, but they do provide good standardization and can help you create efficient code.
    - Frameworks remove a lot of the decisions you would otherwise have to make about how the code is written, the location of files, and even file names.
- This Inversion of control allows you to create standardized apps, and takes away a lot of the tedious configuration work, so you can focus on the code for your app.
- Frameworks often include their own libraries, which they call when needed.

More Development Tools

- **CI / CD**
  - CI: Continuous Integration
    - Ensures that **all the code components work together** smoothly
  - CD: Continuous Delivery / Deployment
    - **Deploys all code changes** to a testing environment
    - CD begins where CI ends
  - CI/CD:
    - A method for releasing code and integrating it into code that has already been developed in order to prevent the application from breaking throughout the app's lifecycle.
    - Best practice developers use to deliver frequent changes reliably.

- **Build Tools**
  - Transforms source code into the **binaries** needed for installation.
  - A build can be initiated from the command line or from an IDE.
  - They are most important in environments where there are many inter-connected projects, with multiple developers contributing to each project.
    - Build tools can automate a wide variety of tasks that developers do in their day-to-day activities like:
      - Downloading dependencies
      - Compiling source code into binary code
      - Packaging that binary code
      - Running test
      - Deployment to production systems.
  - There are two categories of Build Tools widely in use:
    - Build-automation utilities
      - Generates build artifacts like executables, by compiling and linking source code.
    - Build-automation servers
      - Executes build-automation utilities on a scheduled or triggered basis.

- **Packages**
  - After the code has been developed and tested we need to **collect all the necessary files and bundle them together** into a package for users to smoothly run/install the application.
  - Packages are archive files that contain
    - The app files
    - Instructions for installation
    - Any metadata that you choose
    - They have their own metadata too such as
      - Package description
      - Package version
      - Dependencies such as packages that need to be installed beforehand.

- **Package Manager**
  - After the app has been bundled into a package, use a package manager to distribute it.
  - Package management systems:
    - Coordinate with file archivers to extract package archives.
    - Verify checksums and digital certificates to ensure the integrity and authenticity of the package.
    - Locate, download, install, or update existing software from a software repository.
    - Manage dependencies to ensure a package is installed with all packages it requires.
  - Some commonly used package managers for each of the major platforms are listed here:
    - On Linux - Debian Package Management System (DPKG), Red Hat Package Manager (RPM).
    - On Windows - Chocolatey.
    - On Android - Package Manager.
    - On MacOS - Homebrew and MacPorts.

Developer Roles

Front-end developer

- Creates websites and Cloud Apps that the user interacts with

Back-end developer

- Creates and manages resources needed to respond to client requests
- Enables server infrastructure to process request, supply data and provide other services securely

Both work together closely throughout the life of the website or Cloud App to resolve issues and add functionality

- Front-end interactions such as
  - Request for data like an image
  - Accepting input from a user filling out a form
  - Securing sensitive information like a credit card number
- All of these require different services from the back-end server.
- Back-end developers use **APIs, routes,** and **endpoints** to process incoming requests.
  - API (Application Programming Interface)
    - Code that allows two software programs to communicate with one another usually using JSON or XML.
    - APIs have set rules and structure.
  - Route
    - Allows front-end client to plug into correct socket on the backend.
    - Routes generally take user input and show results based on the input.
  - Endpoint
    - The point at which an API connects with the software program.
    - If the backend has an end point defined for the front end request by using routing, the request will be addressed and replied to. If the end point is missing, the server returns a 404 error.
- **API vs Endpoint**
  - An endpoint is a component of an API.
  - While an API is a set of rules that allow two applications to share resources.
  - Endpoints are the locations of the resources, and the API uses endpoint URLs to retrieve the requested resources.

Frameworks For Frontend & Backend

- **Frontend**
  - JavaScript based
    - React.js
    - Angular.js
- **Backend**
  - JavaScript based
    - Node.js
    - Express.js
  - Python based
    - Flask
    - Django