**Micro Frontends For Mobile**

**What is Micro Frontends (MFE) ?**

**Understanding The Concept Of Micro-services That Applies To MFEs**

MFEs use the same concept of micro-services but in terms of developing the front-end. For understanding purposes, lets get a basic understanding of what micro-services actually are. Micro-services is an architectural and organizational approach to software development where the software is broken down into smaller independent services that communicate over well structured APIs. For example imagine an online commerce store which sells clothing, naturally this site will offer various different services like user authentication, cart management, processing transaction etc and each one is known as a “service”. In the micro-services context, each of these services work independently and communicate with each other via APIs. Lets get a better understanding of how APIs work before we continue.

**How Do APIs (Application Programming Interface) Work?**

When it comes to APIs, there are a few things that must be present in order for it to be complete and these include a **Client,** a **Server, Endpoints, Requests** and **Responses.**

**Client –** The client refers to the application that initiates the request.

**Server –** Another application that receives the request from the client application and this provides the response to that request.

**Endpoints –** This refers to the specific address that the client can access through the API.

**Requests –** This refers to the specific request that the client application is sending over to the server application.

**Responses –** These refer to the responses or in essence, the answers (data) to the requests that was sent from the client application to the server application.

So when it comes to communication via API this happens via a **request response model** where one application is sending requests to different applications and these applications are sending responses to the application requesting the response. This **communication is also done following a set of definitions and protocols**.

**Example of API usage from** [**amazon aws documentation**](https://aws.amazon.com/what-is/api/#:~:text=APIs%20are%20mechanisms%20that%20enable,weather%20updates%20on%20your%20phone.) **:**

API architecture is usually explained in terms of client and server. The application sending the request is called the client, and the application sending the response is called the server. So in the weather example, the bureau’s weather database is the server, and the mobile app is the client.

There are also various different types of APIs which can be used for different purposes. The following are the different types of APIs that are available as of right now.

**Private API –** Internal only to an enterprise and used for connecting systems and applications used within an organization or within a business.

**Public API –** These APIs are open to the public and can be used by anyone, most of the time these APIs are free to use but in some cases a subscription or payment may be required. A good example of this could be the google maps API which is open and accessible to everyone to use for free.

**Partner API –** These APIs are exclusive within partner companies, for instance if company X has a partnership with company Y then the APIs under company X will be available to be used by company Y as there exists a partnership between them and these APIs will not be available to anyone else.

**Composite API –** A more unique case where two or more API needs to be working together to address a complex request from the client application hence this is called a composite API.

**So How Do Micro-Frontends Work?**

Now that we have a comprehensive understanding of micro-services and how APIs work, lets see how this understanding will help in aiding us understand how micro-frontends work. Initially when frontend development was a thing, most teams adopted a monolithic architecture approach when it came to development which might have been applicable back then but since the emergence of the micro-service architecture, development of front-ends has taken this approach as well. When it comes to micro-frontends, this breaks modern web pages into pages and feature slices and each feature is owned by a different team. For instance, in a homepage of an e-commerce website, there are product displays and product descriptions, team A could own the product displays and work on it and team B could own the product descriptions portion and work on it independently without having to rely on each other. Similar to micro-services, when development of each team is completed, they could come together and be deployed as a single cohesive unit, meaning a proper and complete functional front-end page.

**What Are The Benefits Of Adopting Micro-Frontends?**

**Scalability**

Since multiple different teams can be working on different features of the website/application, this allows for better scalability as **parallel computing** is achievable in this way. **Parallel computing** is essentially executing multiple tasks and processes at the same time instead of working on them individually and this naturally speeds up the time required to work on projects. Other than that, micro-frontends also allows the developers to develop in different tech stacks instead of sticking to one tech stack that needs to be followed and obeyed by everyone. For instance, if team A is prefers coding using React and team B prefers developing using Vue, both of these teams can proceed and develop their portions in those tech stacks as they are developing using micro-frontends.

**Reusability**

For instance team A has developed a really good carousel for product display while working on project X and they are moving onto developing project Y, instead of developing the carousel from scratch again, they can reuse the same code that has already been developed by the team beforehand to achieve the same goal.

**Common Frameworks Used For Micro-Frontends**

Some popular frameworks typically used for micro-frontends include **React.js**, **Vue**, **Angular**, **Single SPA** and more. When developing using micro-frontends, each of them will have their own independent git repositories and their own package.json files and it is built on its own tool configuration.