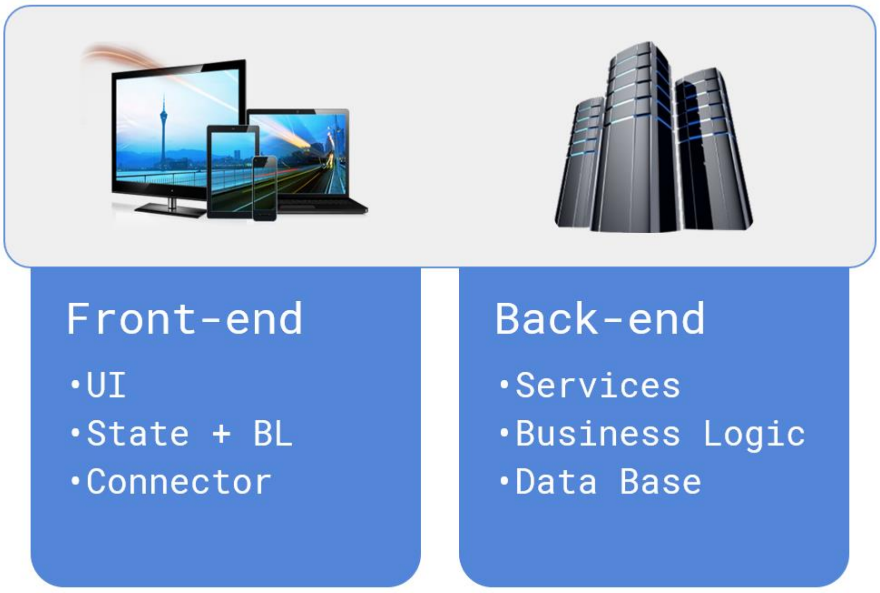
Angular Application Architecture

Angular is a framework for building web applications, so let's start by reviewing some concepts of these applications.

Web application architecture



Front-end
- UI
- State + BL
- Connector

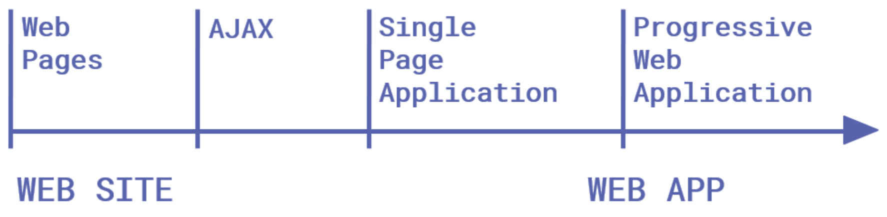
Back-end
- Services
- Business Logic
- Data Base

Web applications have two components: the front-end that runs on the web browser and the back-end that runs on a web server.

The front-end is where the application actually runs and is the part that the user interacts with. Also, it has the user interface, maintains the state of the application and handles the business logic.

The back-end is where the services that serve the front-end are located, the business logic is validated and the data is obtained from the database.

Evolution of web applications



Web applications have evolved since their first implementation, from almost static pages to applications with rich content and interactivity.

To increase interactivity with page controls, AJAX technology emerged that allowed some things to be resolved on the client without the need for the request to travel to the server.

This was followed by the emergence of single-page applications (SPA). One particular feature of these applications is that, in response to user interactions, the page is not reloaded again. Instead, only the necessary part is updated, providing a smoother user experience.

More recently, progressive web applications (PWA) have emerged, which are web applications that provide access to hardware and software resources of the machine as if they were native applications and can be installed on desktop or mobile devices.

What is Angular?

ANGULAR

What is Angular?

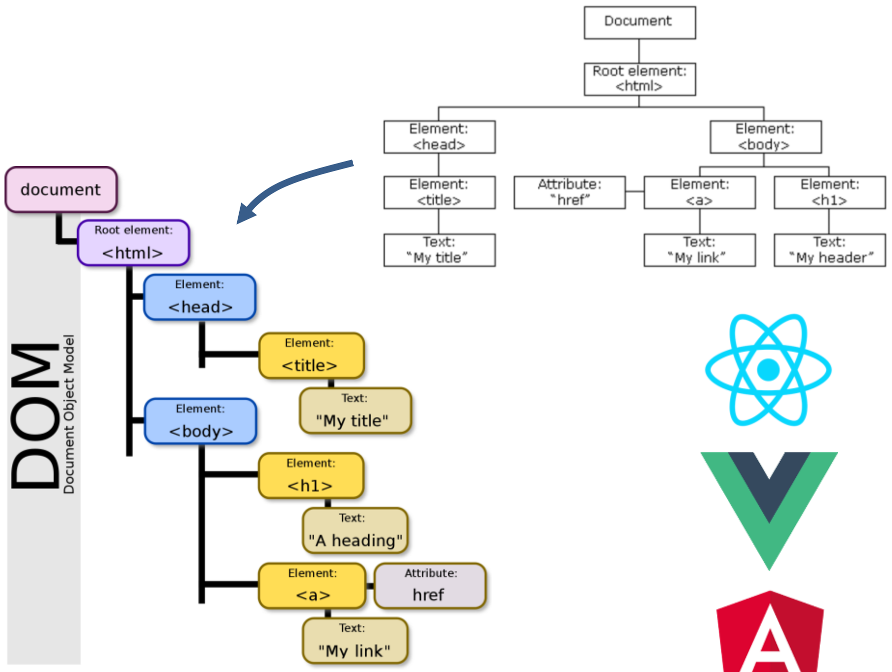
Angular is an open source JavaScript development framework created by Google, which is used to generate front-end web applications, in particular single-page and progressive web applications.

Angular is programmed with TypeScript, which is a superset of JavaScript that adds data types to it, among other things. So, ultimately it is JavaScript that runs in the browser.

This framework specializes in increasing the speed at which the web page is drawn to the browser – that is to say, rendering – which makes it ideal for front-end applications that are end-user oriented – “customer-facing” applications.

DOM: Document Object Model

```html
<html>
    <head>
        <title>"My title"</title>
    </head>
    <body>
        <h1>"A heading"</h1>
        <a href="My link"></a>
    </body>
</html>
```

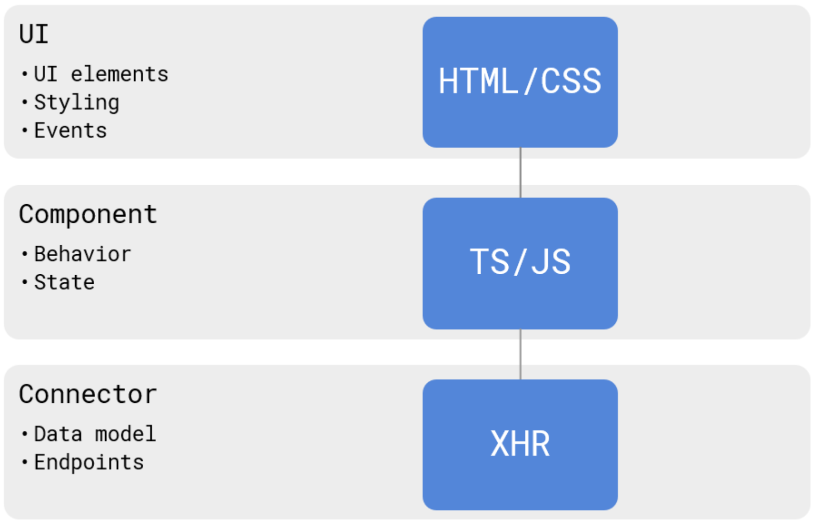


To have the screen drawn quickly, Angular efficiently handles the DOM (Document Object Model) which is the hierarchical structure of objects that is generated by the browser when a document is loaded. It can be altered using JavaScript to dynamically change the contents and appearance of the page.

As single-page applications and PWAs are heavy and increasingly complex, efficient DOM management is very important. This is achieved through DOM virtualization and then updating the actual DOM very quickly. There are other front-end web frameworks such as React or VueJS that also provide efficient DOM management, but they only cover part of the development.

On the other hand, Angular is a complete solution that in addition to the logic for DOM management, includes component management, libraries for navigating the application, management of the state of the controls on the screen and communication mechanisms between the client and the server.

Front-end application structure



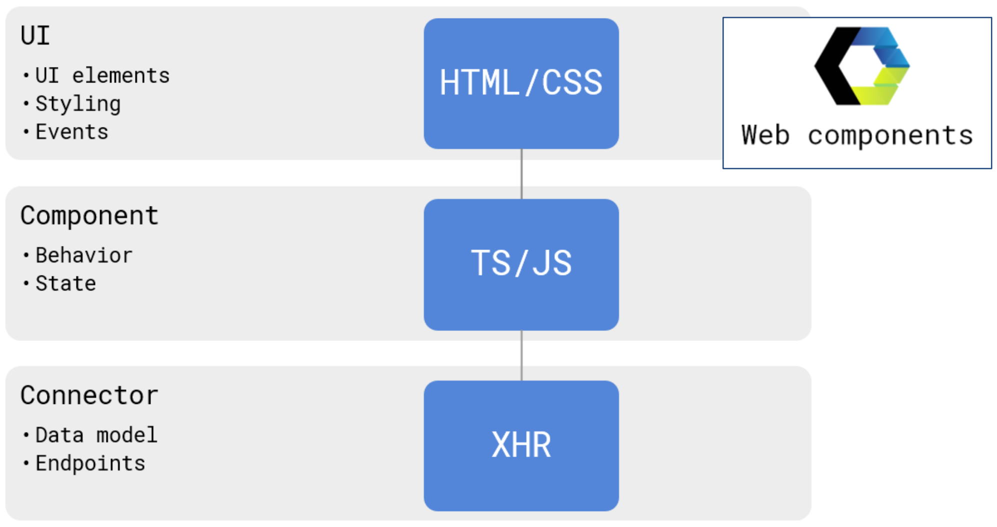
A front-end web application has the following 3 layers or elements.

The User Interface, which is what the user sees. They are the screen elements or controls coded in HTML, with the appearance definitions written in CSS (styling). When the user interacts with this user interface, events are generated which cause changes in the page content.

Component – It is the code that reacts to the events of the User Interface elements; that is, where the behavior of the UI controls is programmed. This layer is also responsible for maintaining the state of each control and its code is programmed in TypeScript or JavaScript.

Connector – It acts as the representative of the server on the client; what it does is to ask the server for data and in some cases it can persist them, to minimize communications with the server.

How the front-end is built with the GeneXus Angular generator



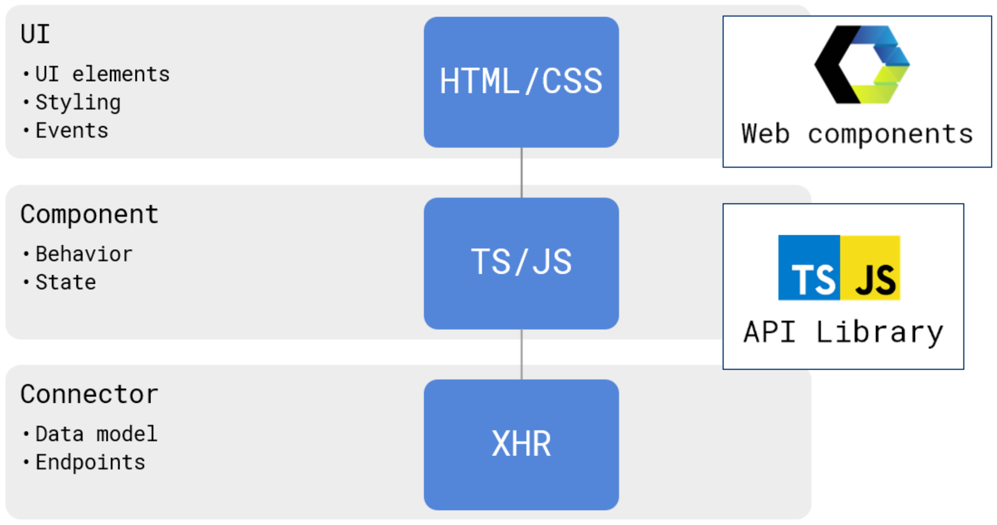
Now let's see how the GeneXus Angular generator builds each of these parts.

Web Components are used to generate the User Interface. Web components are a W3C standard that allows us to encapsulate HTML, JavaScript and CSS in HMTL tags defined by us.

This means that we can extend the HTML and create our own tags based on elements that run natively in all browsers. For example, we can create our own button with its style and behavior, which is encapsulated in a web component and then we can use it in other web projects or distribute it for others to use.

It is a new way to componentize on the web and make components portable.

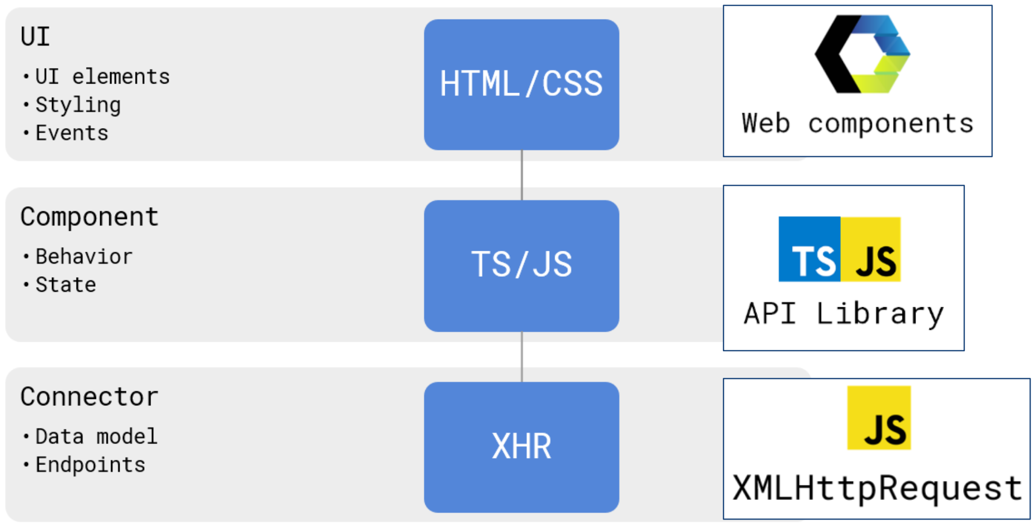
How the front-end is built with the GeneXus Angular generator



The GeneXus API Library is used to implement the behavior layer.

These APIs are general purpose functions such as character handling, date functions, etc., which used to be generated in the specific code of each generator, and are now implemented in TypeScript and JavaScript so that they can be reused in all generators, including Angular.

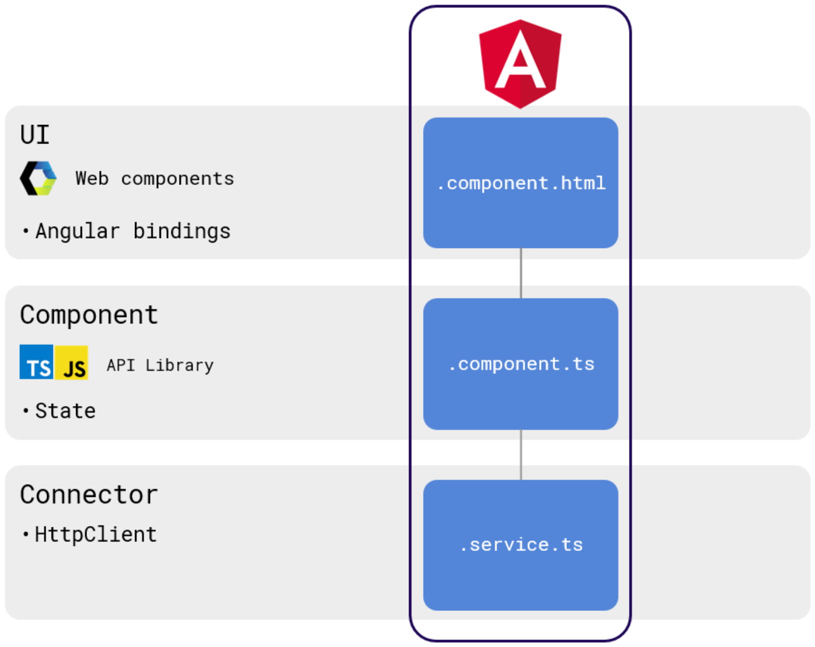
How the front-end is built with the GeneXus Angular generator



For the connector, the XMLHttpRequest object is used, which is a JavaScript object designed by Microsoft and adopted by Mozilla, Apple and Google.
It is currently a W3C standard and allows you to:

- Refresh a web page without reloading it.
- Request data from the server and receive data from the server, after the page has been loaded.
- Send data to the server in the background.

Angular component of the application and generated files



The basic building block of an Angular application is called a Component. Each component consists of 3 parts that map the layers of a front-end application and are represented in 3 files that are created by the Angular generator.

The user interface definitions are stored in a file whose name ends in .component.html. This file is generated by GeneXus and contains the web components that define the HTML template and the User Interface style definitions. The associations of the UI elements with the corresponding data in the database are also included.

The component that defines the behavior and state of the controls is stored in a file ending in .component.ts. The behavior and state of the controls are programmed there in TypeScript, using the GeneXus API Library.

The connector that communicates with the REST services of the web server is generated in a file ending in .service.ts and GeneXus uses the HttpClient to implement these communications, using the XMLHttpRequest.

For each panel object generated, these 3 files will be created with names consisting of the name of the object to be executed, followed by the extensions that are displayed on the screen.

Angular generator use scenarios

Develop a customer-facing front-end web application

When we know we will need a native mobile application

Convert a native mobile application developed in GeneXus to a Web app

Develop back-end in GeneXus for front-end developed with React, VueJs or Angular

One question we can ask ourselves is when we should choose Angular to generate our web application.

One possible scenario is when we need to develop a customer-facing front-end web application. As we saw, Angular is designed to have a good performance in these cases of user interfaces with high interactivity requirements.

Another case in which Angular development is the right choice is when we are sure that after the web front-end, we will need a native mobile front-end. In this case, the panels created for the Angular application can be reused to generate the Android or iOS application, defining the layouts corresponding to the devices we are going to use.

Angular can also be used when we have already developed a native Android or iOS app with GeneXus and we need a web front-end. This case is similar to the previous one, since the same objects can be used, changing the layout design for web and then generating the same application we had in Angular.

Another case in which it is useful to use Angular as a generator is when we want to integrate applications coded manually in React, VueJs, or Angular with services or applications developed in GeneXus. In this way, if there is a team of developers specialized in the use of these frameworks, they can continue to build applications in their tool but the entire back-end can be developed in GeneXus and integrated to the front-end.

In the following videos we will see how to develop web applications with the GeneXus Angular generator.

GeneXus™

training.genexus.com
wiki.genexus.com
training.genexus.com/certifications