Let me quickly remind you of what a software library is,

a software library is a collection of implementations of

behaviors that you repeatedly use within your applications.

So these behaviors can be implemented by using functions with well-defined

interfaces so

that making use of them in implementing your application becomes straightforward.

If you have done programming in any language, you would often

see that many of the standard functionality that is supported

in these languages are implemented through libraries that you make use of.

For example, the input output libraries and perhaps even

the GUI libraries that many standard programming languages support.

This is where the concept of reuse of behavior is very,

very effective because we are able to implement it once in a library and

then reuse that behavior wherever we need within our software application.

This also brings in modularity to the way our application is implemented.

A good example of a software library that is used in the web world is jQuery.

let me also differentiate a software library from a software framework.

A software framework is an abstraction

in which the software provides such generic functionality and

then provides you with the ability of customizing that functionality for

implementing the solutions for your specific application.

So you will augment the software framework with your own code

in order to accomplish whatever that you are trying to accomplish.

So this is a different way of thinking about how you

attempt to solve the problem as opposed to using a software library.

Now, this software framework provides a universal, reusable environment

with certain functionality that is part of a larger software platform.

We'll look at Angular as an example of how this is accomplished in more detail.

There are several other such software frameworks that are used in the web world

including Ember, Backbone, and many others.

Now here, you might obviously think of the question.

How do you distinguish a library from a framework?

What are the salient characteristics of each and

why would you classify one as a library and another one as a framework?

Where do you draw the distinction between the two?

Now this I should say,

there is a fine line separating a library from a framework.

Depending on how you approach this, you may view one or

the other as either a library, or a framework, and

depending on who you ask, some people may classify a particular

approach as a library approach as opposed to a framework approach.

Again, let's not split our hairs about trying to exactly

classify things into a library or a framework.

From our perspective, if something helps us accomplish our task,

then we are satisfied with that approach.

Again, having said that, let's distinguish between a library and

a framework in a bit more detail.

In order to understand how you approach

that entire way of writing your application.

When you use a library as opposed to using a framework.

So in this context the following definition that I have borrowed

from AngularJS documentation sort of, I feel

gives a bit more clearer distinction between a library and a framework.

So, this definition says that **a library is a collection of functions**

**which are useful for writing web applications, jQuery being an example.**

**So this is a set of functions that are available to you.**

**You make use of this functions when you implement your software**

**solution to your problem.**

When you use this approach, it is you the programmer that is always in control

of how you leverage the functions that are provided

by the library to accomplish whatever that you are trying to do.

A **framework** on the other hand is a specific implementation or

particular implementation of a web application,

where the framework itself gives a broad generic structure for

your application and then you are asked to fill in the details.

The details that are necessary to customize that application

to solve the specific problem that you're trying to address.

So here, the framework takes complete control over how this applications run and

then it'll call in to your user supplied code that helps it to get the work done.

The app specific code is something that you will write and

supply it and that'll becomes part of the entire application,

together with the framework, and it helps to accomplish your task.

Again, this may not completely, clearly enable you to

understand the distinction between a framework and a library.

Let's again, as I said,

let's not spread our hairs trying to figure out all these details.

What we need to understand is Angular is an example of a framework and

when you write applications in Angular, the framework takes control and

you basically supply the missing pieces of the code that are required

to customize the framework to implement your specific web application.

**What is angular**

Angular is a component-based approach for implementing web applications.

So, component has become the front and center of Angular now.

Although, the component-based approach has been back-ported to

AngularJS also from AngularJS version 1.5.

So, if you have been component-based approach with AngularJS,

you will be in a somewhat familiar territory if you come into Angular.

Angular has been designed from the beginning

with mobile support so you can easily address mobile platforms,

and also provides server-side rendering to speed up

the rendering of your web application on the browsers.

Also, Angular provides powerful templating engine and powerful templating support.

We will see some of these in more detail as we go along in this course.

To summarize, what exactly is Angular?

Angular is a structural framework for designing dynamic web applications.

The HTML based approach that we have seen,

for example, with Bootstrap results in primarily static documents, of course,

with jQuery and AJAX,

you can do a little bit of dynamic behavior in your websites using Bootstrap earlier,

but Angular fills in the gap to support

fully dynamic applications, data-driven dynamic applications.

This uses a declarative approach.

We have talked about the declarative programming approach in the previous lecture,

so that's what Angular adopts as its way of solving the problems.