**Shaping up with AngularJS**

The goal of this Video is to get you comfortable with the terminology and the technology and to give you an understanding of how AngularJS works.

**What Is a Browser?**

Before we jump straight into our coverage of Angular, it’s important to know what your browser is doing when it renders a web page.

There are many different web browsers are there. The most commonly used today include Chrome, Safari, Mozilla Firefox, and Internet Explorer. they all basically do the same thing: fetch web pages and display them to the user.

Your browser gets the HTML text of the page, parses it into a structure that is internally meaningful to the browser, lays out the content of the page, and styles the content before displaying it to you. All of this work happens behind the scenes.

Our goal as web developers is to build the structure and content of our web page so that the browser will make it look great for our users.

With Angular, we’re not only building the structure, but we’re constructing the interaction between the user and our app as a web application.

**How Web Pages Get to Your Browser**

Let’s think of the Internet as a post office. When you want to send a letter to your friend, you first write your message on a piece of paper. Then you write your friend’s address on an envelope and place the letter inside of it.

When you drop the letter off at the post office, the mail sorter looks at the postal code and address and tries to find where your friend lives. If she lives in a giant apartment complex, the postal service might deliver the mail to your friend’s front desk and let the building’s employees sort it out by apartments.

The Internet works in a similar way. Instead of a bunch of houses and apartments connected by

Streets, it is a bunch of computers connected by routers and wire. Every computer has a unique

Address that tells the network how to reach it.

When you open your web browser on your computer and type in http://interviewgully.com, your web Browser “asks” the internet (more precisely, it “asks” a DNS server) where interviewgully.com address is.

If the DNS server knows the IP address you/re looking for, it responds with the address. If not,

It passes the request along to other DNS servers until the IP address is found and served to your

Computer. You can see the DNS server response by typing this code into a terminal:

Once the DNS server responds with the IP address of the computer you’re trying to reach (i.e., once It finds interviewgully.com), it also sends a message to the computer located at that IP address asking for the web page you’re requesting.

Now that your computer has the IP address it needs to get http://interviewgully.com, it asks the Interviewgully server for the HTML it needs to display the page.

Once the remote server sends back that HTML, your web browser renders it (i.e., the browser works to make the HTML look the way interviewgully.com is designed to look.

**What You Should Already Know**

Before you study AngularJS, you must have a basic understanding of:

* HTML
* CSS
* JavaScript

**Nice to Know**

* Automated Testing
* BDD – Behavior Driven Development
* TDD – Test Driven Development

**Why AngularJS?**

AngularJS helps you to organize your javascript.

Angular help you to create responsive web stuff.

Angular play very well with JQuery.

Angular is easy to test.

**What Is AngularJS**

It is a framework that is primarily used to build single-page web applications. AngularJS makes it

easy to build interactive, modern web applications by increasing the level of abstraction between the developer and common web app development tasks.

The AngularJS team describes it as a “structural framework for dynamic web apps.”

AngularJS is perfect for SPAs (Single Page Applications).

AngularJS takes care of advanced features

• Separation of application logic, data models, and views (MVC Implement)

• Ajax services

• Dependency injection

• Browser history (makes bookmarking and back/forward buttons work like normal web apps)

• Testing

• Data Binding

• Expression

• Routing

• Directives

• Services

• Filter

**How is it different?**

In other JavaScript frameworks, we are forced to extend from custom JavaScript objects and

Manipulate the DOM from the outside in. For instance, using jQuery, to add a button in the DOM, we’ll have to know where we’re putting the element and insert it in the appropriate place:

var btn = $("<button>Hi</button>");

btn.on('click', function(evt) { console.log("Clicked button") });

$("#checkoutHolder").append(btn);

Although this process is not complex, it requires the developer to have knowledge of the entire DOM and force our complex logic inside JavaScript code to manipulate a foreign DOM.

AngularJS, on the other hand, augments HTML to give it native Model-View-Controller (MVC)

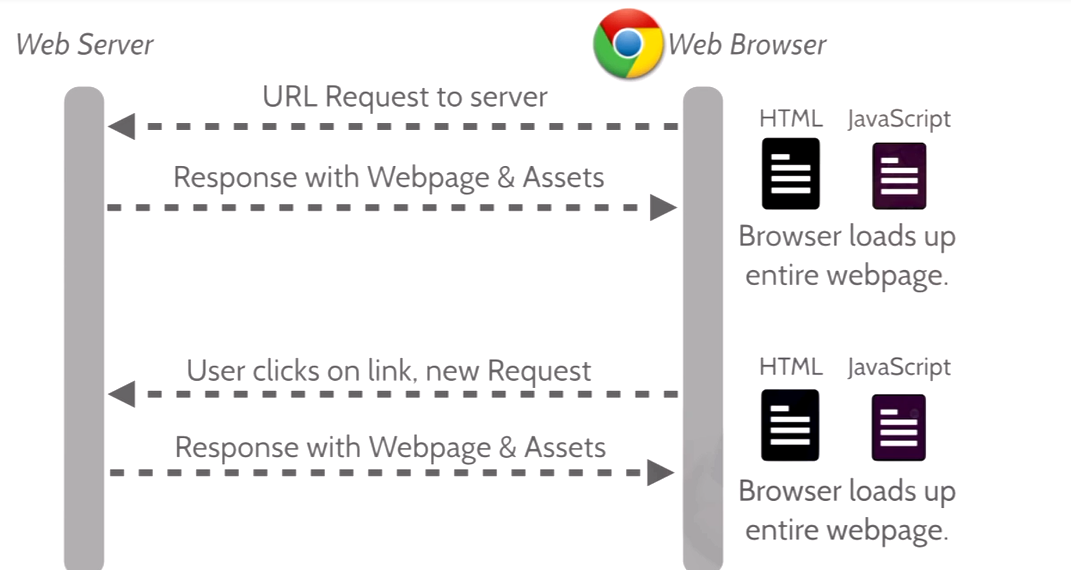
Capabilities. This choice, as it turns out, makes building impressive and expressive client-side

Applications quick and enjoyable.

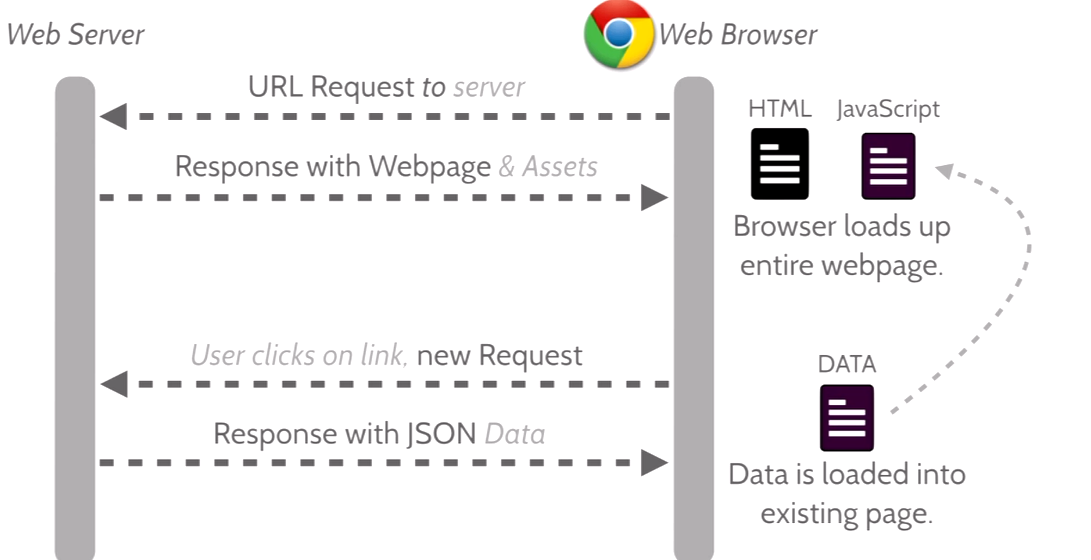
It enables you, the developer, to encapsulate a portion of your entire page as one application, rather than forcing the entire page to be an AngularJS application. This distinction is particularly beneficial if your workflow already includes another framework or if you want to make a portion of the page dynamic while the rest operates as a static page or is controlled by another JavaScript framework.

Additionally, the AngularJS team has made it a point to keep the library small when compressed, such that it does not impose heavy penalties for using it (the compressed, minified version weighs in under 9KB at the time of this writing). This feature makes AngularJS particularly good for prototyping new features.

**Traditional Webpage refresh**



**Responsive website (Using AngularJS)**



**Simple Example**

<!DOCTYPE html>

<html **ng-app**>

<head>

<title> InterviewGully Simple Application Example</title>

<script

src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.6/angular.js">

</script>

</head>

<body>

<input ng-model="name" type="text" placeholder="Your name">

<h1>Hello {{ name }}</h1>

</body>

</html>