# Programmazione Web a.a. 2017/2018
## Cenni: Ajax,JQuery,AngularJS,Bootstrap


- PhD Ing.Antonino Raucea
- antonino.raucea@dieei.unict.it

# AJAX Introduction

AJAX is a developer's dream, because you can:

- Read data from a web server - after the page has loaded
- Update a web page without reloading the page
- Send data to a web server - in the background

# Example

```
<!DOCTYPE html>
<html>
<body>
<div id="demo">
<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Change Content</button>
</div>
<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
      this.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
</script>
</body>
</html>c
```

# What is AJAX?

AJAX = Asynchronous JavaScript And XML.

AJAX is not a programming language.

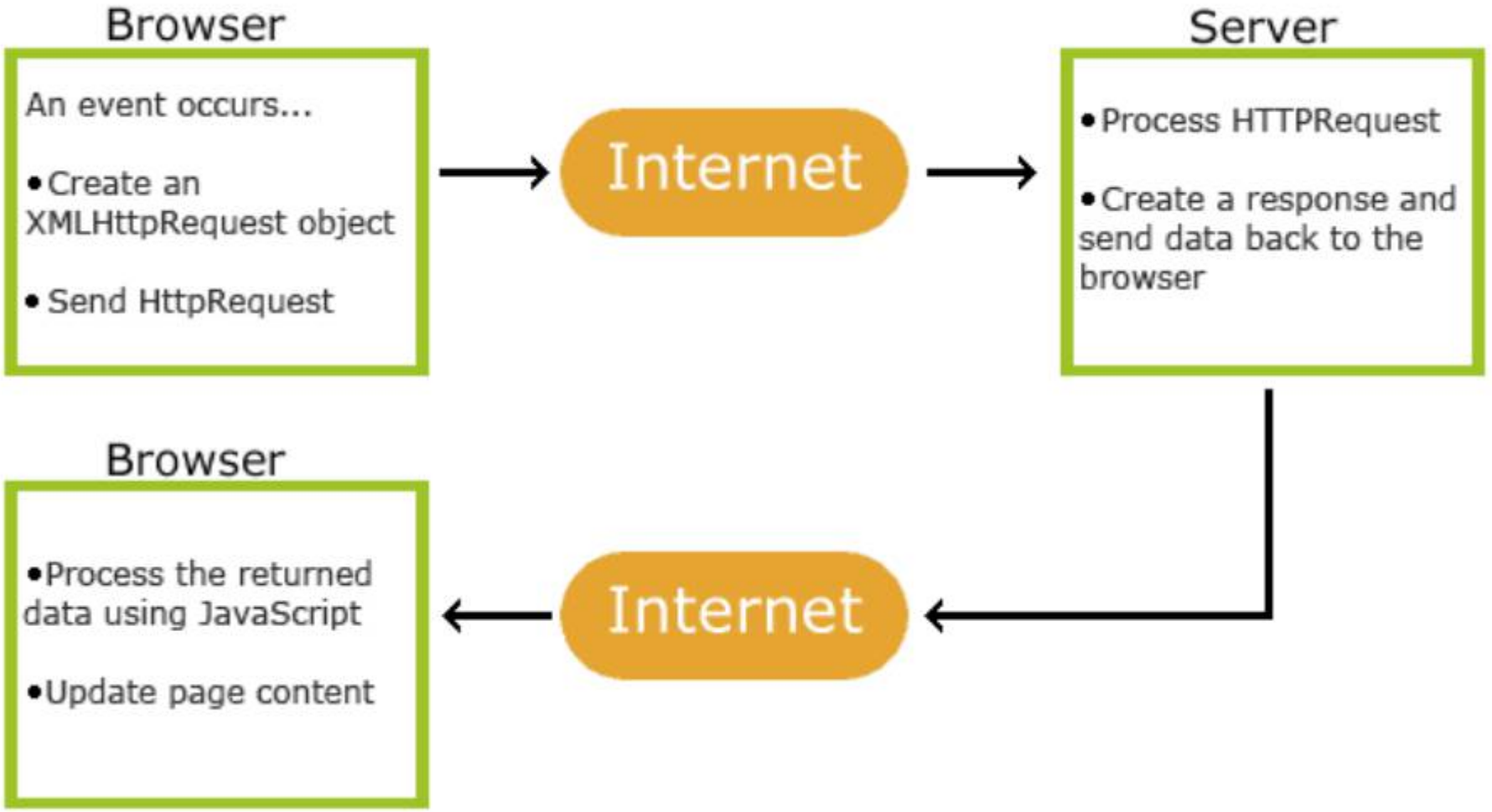
AJAX just uses a combination of:

- A browser built-in XMLHttpRequest object (to request data from a web server)
- JavaScript and HTML DOM (to display or use the data)

AJAX is a misleading name. AJAX applications might use XML to transport data, but it is equally common to transport data as plain text or JSON text.

AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

# How AJAX Works



# How AJAX Works

1. An event occurs in a web page (the page is loaded, a button is clicked)
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read by JavaScript
7. Proper action (like page update) is performed by JavaScript

# AJAX - The XMLHttpRequest Object

The keystone of AJAX is the XMLHttpRequest object.

The XMLHttpRequest Object is supported by all modern browsers.

```
var xhttp = new XMLHttpRequest();
```

# Access Across Domains

For security reasons, modern browsers do not allow access across domains.

This means that both the web page and the XML file it tries to load, must be located on the same server.

The examples on W3Schools all open XML files located on the W3Schools domain.

If you want to use the example above on one of your own web pages, the XML files you load must be located on your own server.

# AJAX - Send a Request To a Server

Send a Request To a Server

To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object:

```
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

# GET or POST?

GET is simpler and faster than POST, and can be used in most cases.

However, always use POST requests when:

- A cached file is not an option (update a file or database on the server).
- Sending a large amount of data to the server (POST has no size limitations).
- Sending user input (which can contain unknown characters), POST is more robust and secure than GET.

```
xhttp.open("GET", "demo_get.asp?t=" + Math.random(), true);
xhttp.send();

xhttp.open("GET", "demo_get2.asp?fname=Henry&lname=Ford", true);
xhttp.send();
```

# POST Requests

To POST data like an HTML form, add an HTTP header with setRequestHeader(). Specify the data you want to send in the send() method:

```
xhttp.open("POST", "ajax_test.asp", true);
xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xhttp.send("fname=Henry&lname=Ford");
```

# Asynchronous - True or False?

Server requests should be sent asynchronously.

The async parameter of the open() method should be set to true:

```
xhttp.open("GET", "ajax_test.asp", true);
```

By sending asynchronously, the JavaScript does not have to wait for the server response, but can instead:

execute other scripts while waiting for server response

deal with the response after the response is ready

# The onreadystatechange Property

With the XMLHttpRequest object you can define a function to be executed when the request receives an answer.

The function is defined in the onreadystatechange property of the XMLHttpResponse object:

```
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    document.getElementById("demo").innerHTML = this.responseText;
  }
};
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

# Synchronous Request

o execute a synchronous request, change the third parameter in the open() method to false:

```
xhttp.open("GET", "ajax_info.txt", false);
```

Sometimes async = false are used for quick testing. You will also find synchronous requests in older JavaScript code.

Since the code will wait for server completion, there is no need for an onreadystatechange function:

```
xhttp.open("GET", "ajax_info.txt", false);
xhttp.send();
document.getElementById("demo").innerHTML = xhttp.responseText;
```

Synchronous XMLHttpRequest (async = false) is not recommended because the JavaScript will stop executing until the server response is ready. If the server is busy or slow, the application will hang or stop.

# AJAX - Server Response

The onreadystatechange Property

The readyState property holds the status of the XMLHttpRequest.

The onreadystatechange property defines a function to be executed when the readyState changes.

The status property and the statusText property holds the status of the XMLHttpRequest object.

# AJAX - Server Response

| Property | Description |
|---|---|
| onreadystatechange | Defines a function to be called when the readyState property changes |
| readyState | Holds the status of the XMLHttpRequest.<br>0: request not initialized<br>1: server connection established<br>2: request received<br>3: processing request<br>4: request finished and response is ready |
| status | 200: "OK"<br>403: "Forbidden"<br>404: "Page not found"<br>For a complete list go to the Http Messages Reference |
| statusText | Returns the status-text (e.g. "OK" or "Not Found") |

The onreadystatechange function is called every time the readyState changes.

When readyState is 4 and status is 200, the response is ready

# jQuery

jQuery is a JavaScript Library.

jQuery greatly simplifies JavaScript programming.

jQuery is easy to learn.

```
$(document).ready(function(){
    $("p").click(function(){
        $(this).hide();
    });
});
```

# What is jQuery?

jQuery is a lightweight, "write less, do more", JavaScript library.

The purpose of jQuery is to make it much easier to use JavaScript on your website.

jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.

jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

The jQuery library contains the following features:

- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations
- AJAX
- Utilities

Tip: In addition, jQuery has plugins for almost any task out there.

# Why jQuery?

There are lots of other JavaScript frameworks out there, but jQuery seems to be the most popular, and also the most extendable.

Many of the biggest companies on the Web use jQuery, such as:

- Google
- Microsoft
- IBM
- Netflix

The jQuery team knows all about cross-browser issues, and they have written this knowledge into the jQuery library. jQuery will run exactly the same in all major browsers, including Internet Explorer 6!

# Downloading jQuery

There are two versions of jQuery available for downloading:

- Production version - this is for your live website because it has been minified and compressed
- Development version - this is for testing and development (uncompressed and readable code)

Both versions can be downloaded from jQuery.com.

The jQuery library is a single JavaScript file, and you reference it with the HTML <script> tag (notice that the <script> tag should be inside the <head> section):

```
<head>
<script src="jquery-3.2.1.min.js"></script>
</head>
```

# jQuery Syntax

The jQuery syntax is tailor-made for selecting HTML elements and performing some action on the element(s).

Basic syntax is: *$(selector).action()*

- A $ sign to define/access jQuery
- A (selector) to "query (or find)" HTML elements
- A jQuery action() to be performed on the element(s)

Examples:
(a)$(this).hide() - hides the current element.
(b)$("p").hide() - hides all <p> elements.
(c)$(".test").hide() - hides all elements with class="test".
(d)$("#test").hide() - hides the element with id="test".

# The Document Ready Event

You might have noticed that all jQuery methods in our examples, are inside a document ready event.

This is to prevent any jQuery code from running before the document is finished loading (is ready).

It is good practice to wait for the document to be fully loaded and ready before working with it. This also allows you to have your JavaScript code before the body of your document, in the head section.

Here are some examples of actions that can fail if methods are run before the document is fully loaded:

Trying to hide an element that is not created yet

Trying to get the size of an image that is not loaded yet

# The Document Ready Event

```
$(document).ready(function(){

   // jQuery methods go here...

});
```

YThe jQuery team has also created an even shorter method for the document ready event:

```
$(function(){

   // jQuery methods go here...

});
```

# jQuery Selectors

jQuery selectors allow you to select and manipulate HTML element(s).

jQuery selectors are used to "find" (or select) HTML elements based on their name, id, classes, types, attributes, values of attributes and much more. It's based on the existing CSS Selectors, and in addition, it has some own custom selectors.

All selectors in jQuery start with the dollar sign and parentheses: $().

# The element Selector

The jQuery element selector selects elements based on the element name.

You can select all <p> elements on a page like this: `$("p")`

```
$(document).ready(function(){
    $("button").click(function(){
        $("p").hide();
    });
});
```

# The #id Selector

The jQuery #id selector uses the id attribute of an HTML tag to find the specific element.

An id should be unique within a page, so you should use the #id selector when you want to find a single, unique element.

To find an element with a specific id, write a hash character, followed by the id of the HTML element: `$("#test")`

```
$(document).ready(function(){
    $("button").click(function(){
        $("#test").hide();
    });
});
```

# The .class Selector

The jQuery class selector finds elements with a specific class.

To find elements with a specific class, write a period character, followed by the name of the class:

```
$(".test")
```

```
$(document).ready(function(){
    $("button").click(function(){
        $(".test").hide();
    });
});
```

# More Examples of jQuery Selectors

| Syntax | Description |
|---|---|
| $("*") | Selects all elements |
| $(this) | Selects the current HTML element |
| $("p.intro") | Selects all <p> elements with class="intro" |
| $("p:first") | Selects the first <p> element |
| $("ul li:first") | Selects the first <li> element of the first <ul> |
| $("ul li:first-child") | Selects the first <li> element of every <ul> |
| $("[href]") | Selects all elements with an href attribute |
| $("a[target='_blank']") | Selects all <a> elements with a target attribute value equal to "_blank" |
| $("a[target!='_blank']") | Selects all <a> elements with a target attribute value NOT equal to "_blank" |
| $(":button") | Selects all <button> elements and <input> elements of type="button" |
| $("tr:even") | Selects all even <tr> elements |
| $("tr:odd") | Selects all odd <tr> elements |

# What are Events?

All the different visitor's actions that a web page can respond to are called events.

An event represents the precise moment when something happens.

Examples:

- moving a mouse over an element
- selecting a radio button
- clicking on an element

The term "fires/fired" is often used with events. Example: "The keypress event is fired, the moment you press a key".

# Events

| Mouse Events | Keyboard Events | Form Events | Document/Window Events |
|---|---|---|---|
| click | keypress | submit | load |
| dblclick | keydown | change | resize |
| mouseenter | keyup | focus | scroll |
| mouseleave | | blur | unload |

# jQuery Syntax For Event Methods

In jQuery, most DOM events have an equivalent jQuery method.

To assign a click event to all paragraphs on a page, you can do this:

```
$("p").click();
```

The next step is to define what should happen when the event fires. You must pass a function to the event:

```
$("p").click(function(){
  // action goes here!!
});
```

# jQuery fadeIn() fadeOut() Method

The jQuery fadeIn() method is used to fade in a hidden element.

Syntax:

```
$(selector).fadeIn(speed,callback);
```

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is a function to be executed after the fading completes.

The following example d od with different parameters:

```
$("button").click(function(){
    $("#div1").fadeIn();
    $("#div2").fadeIn("slow");
    $("#div3").fadeIn(3000);
});
```

# AngularJS Intro

- AngularJS is an open-source web application framework. It was originally developed in 2009 by Misko Hevery, a Google employee, and Adam Abrons
- AngularJS version 1.0 was released in 2012.
- Latest version is 1.2.21
- The idea turned out very well, and the project is now officially supported by Google.
- AngularJS is a JavaScript framework. It can be added to an HTML page with a <script> tag.
- AngularJS extends HTML attributes with Directives, and binds data to HTML with Expressions

# AngularJS

*AngularJS is a structural framework for dynamic web applications. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application components clearly and succinctly. Its data binding and dependency injection eliminate much of the code you currently have to write. And it all happens within the browser, making it an ideal partner with any server technology.*

# General Features

- AngularJS is a efficient framework that can create Rich Internet Applications (RIA).
- AngularJS provides developers an options to write client side applications using JavaScript in a clean Model View Controller (MVC) way.
- Applications written in AngularJS are cross-browser compliant. AngularJS automatically handles JavaScript code suitable for each browser.
- AngularJS is open source, completely free, and used by thousands of developers around the world. It is licensed under the Apache license version 2.0

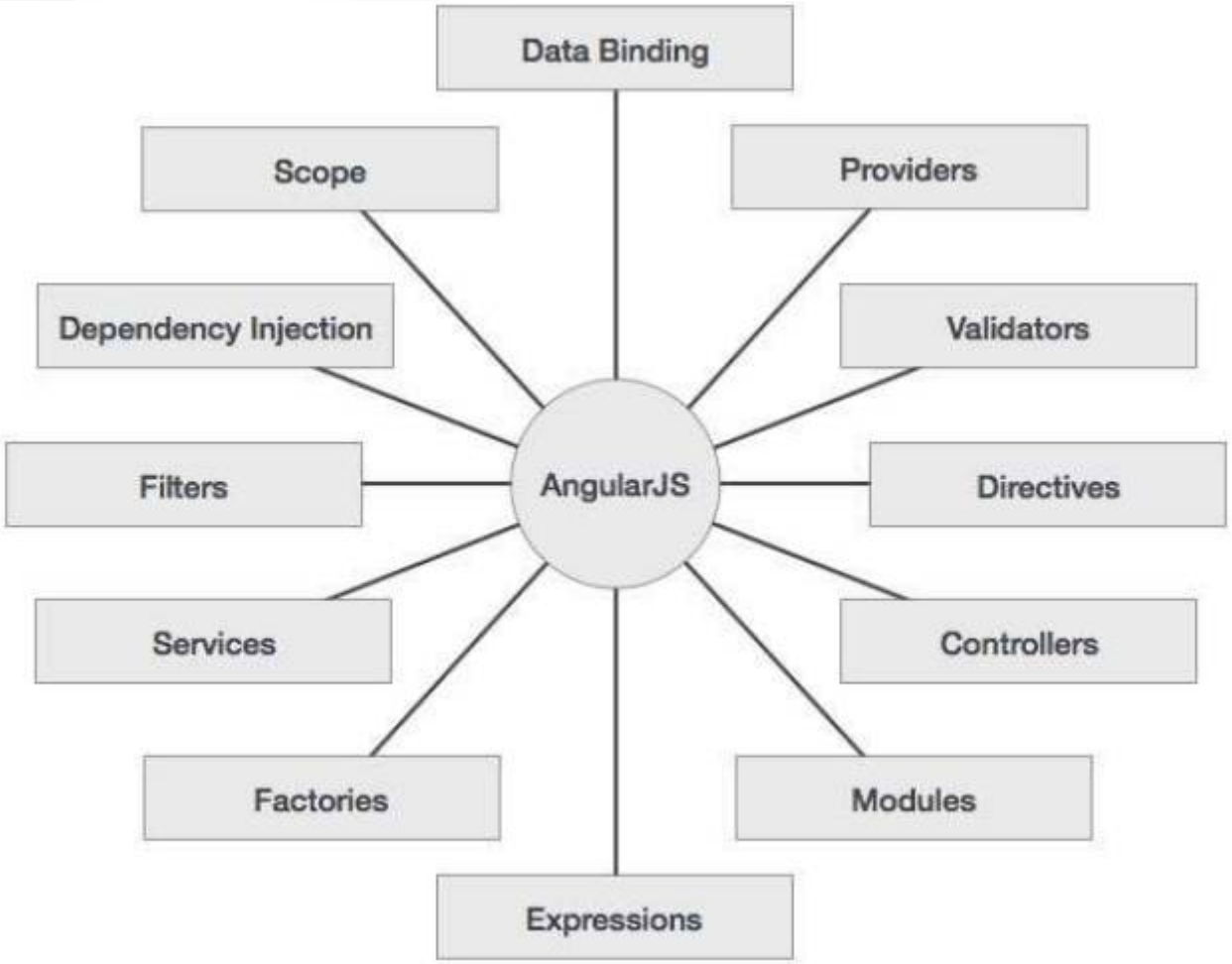
# Core Features

- Data-binding: It is the automatic synchronization of data between model and view components.
- Scope: These are objects that refer to the model. They act as a glue between controller and view.
- Controller: These are JavaScript functions bound to a particular scope.
- Services: AngularJS comes with several built-in services such as $http to make a XMLHttpRequests. These are singleton objects which are instantiated only once in app.
- Filters: These select a subset of items from an array and returns a new array.
- Directives: Directives are markers on DOM elements such as elements, attributes, css, and more. These can be used to create custom HTML tags that serve as new, custom widgets. AngularJS has built-in directives such as ngBind, ngModel, etc.
- Templates: These are the rendered view with information from the controller and model. These can be a single file (such as index.html) or multiple views in one page using partials.

# Core Features

- Routing: It is concept of switching views.
- Model View Whatever: MVW is a design pattern for dividing an application into different parts called Model, View, and Controller, each with distinct responsibilities. AngularJS does not implement MVC in the traditional sense, but rather something closer to MVVM (Model-View-ViewModel). The AngularJS team refers it humorously as Model View Whatever.
- Deep Linking: Deep linking allows to encode the state of application in the URL so that it can be bookmarked. The application can then be restored from the URL to the same state.
- Dependency Injection: AngularJS has a built-in dependency injection subsystem that helps the developer to create, understand, and test the applications easily.

# Concepts


Data Binding
Scope
Providers
Dependency Injection
Validators
Filters
AngularJS
Directives
Services
Controllers
Factories
Modules
Expressions

# Advantages of AngularJS

- It provides the capability to create Single Page Application in a very clean and maintainable way.
- It provides data binding capability to HTML. Thus, it gives user a rich and responsive experience.
- AngularJS code is unit testable.
- AngularJS uses dependency injection and make use of separation of concerns.
- AngularJS provides reusable components.
- With AngularJS, the developers can achieve more functionality with short code.
- In AngularJS, views are pure html pages, and controllers written in JavaScript do the business processing.

# Disadvantages of AngularJS

- **Not secure:** Being JavaScript only framework, application written in AngularJS are not safe. Server side authentication and authorization is must to keep an application secure.
- **Not degradable:** If the user of your application disables JavaScript, then nothing would be visible, except the basic page

# AngularJS is a JavaScript Framework

AngularJS is a JavaScript framework. It is a library written in JavaScript.

AngularJS is distributed as a JavaScript file, and can be added to a web page with a script tag:

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js">
</script>
```

# AngularJS Extends HTML

AngularJS extends HTML with ng-directives and it  can be divided into three major parts:

- The ng-app directive defines an AngularJS application.
- The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.
- The ng-bind directive binds application data to the HTML view.

# AngularJS Example

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js">
</script>
<body>

<div ng-app="">
  <p>Name: <input type="text" ng-model="name"></p>
  <p ng-bind="name"></p>
</div>

</body>
</html>
```

**AngularJS** starts automatically when the web page has loaded.

- The **ng-app** directive tells AngularJS that the  element is the "owner" of an AngularJS application.
- The **ng-model** directive binds the value of the input field to the application variable name.
- The **ng-bind** directive binds the **innerHTML** of the <p> element to the application variable name.

# AngularJS Example

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js">
</script>
<body>

<div ng-app="">
  <p>Name: <input type="text" ng-model="name"></p>
  <p ng-bind="name"></p>
</div>

</body>
</html>
```

**AngularJS** starts automatically when the web page has loaded.

- The **ng-app** directive tells AngularJS that the  element is the "owner" of an AngularJS application.
- The **ng-model** directive binds the value of the input field to the application variable name.
- The **ng-bind** directive binds the **innerHTML** of the <p> element to the application variable name.

# AngularJS Directives

As you have already seen, AngularJS directives are HTML attributes with an ng prefix.

The ng-init directive initializes AngularJS application variables.

```
<div ng-app="" ng-init="firstName='John'">

<p>The name is <span ng-bind="firstName"></span></p>

</div>
```

# AngularJS Expressions

- AngularJS expressions are written inside double braces: {{ expression }}.
- AngularJS will "output" data exactly where the expression is written:

```
<div ng-app="">
  <p>My first expression: {{ 5 + 5 }}</p>
</div>
```

My first expression: 10

# Fine