AngularJS Service

- In AngularJS, a service is a function, or object, that is available for, and limited to, your AngularJS application.

- AngularJS has about 30 built-in services. One of them is the $location service.

- The $location service has methods which return information about the location of the current web page.

AngularJS Service

Example

- Use the $location service in a controller:

```
var app = angular.module('myApp', []);
app.controller('customersCtrl', function($scope, $location)
{
  $scope.myUrl = $location.absUrl();
});
```

- Note that the $location service is passed in to the controller as an argument.
- In order to use the service in the controller, it must be defined as a dependency.

AngularJS Service

Why use Services?

- For many services, like the $location service, it seems like you could use objects that are already in the DOM, like the window.location object, and you could, but it would have some limitations, at least for your AngularJS application.

- AngularJS constantly supervises your application, and for it to handle changes and events properly, AngularJS prefers that you use the $location service instead of the window.location object.

The $http Service

- The $http service is one of the most common used services in AngularJS applications.

- The service makes a request to the server, and lets your application handle the response.

Example
Use the $http service to request data from the server:

```
var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope, $http)

{

 $http.get("welcome.htm").then(function (response)

 {

   $scope.myWelcome = response.data;

 });

});
```

The $timeout Service

The $timeout service is AngularJS' version of the window.setTimeout function.

Example
Display a new message after two seconds:

```
var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope, $timeout) {
  $scope.myHeader = "Hello World!";
  $timeout(function () {
    $scope.myHeader = "How are you today?";
  }, 2000);
});
```

The $interval Service

The $interval service is AngularJS' version of the window.setInterval function.

Example
Display the time every second::

```
var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope, $interval) {
  $scope.theTime = new Date().toLocaleTimeString();
  $interval(function () {
    $scope.theTime = new Date().toLocaleTimeString();
  }, 1000);
});
```

Where Do We Use $timeout and $interval?

- The $timeout help to wait for the function execution at the specified time, sometimes you need to call a method after some time, that scenario you will use $timeout method and passed your delayed time in second.

- The $interval help to call a method again and again on a given time interval.

- Like we want to call a method on each 5 seconds interval that time we will use $interval and passed interval time 5000 millisecond

AngularJS Events:

- When creating web-based applications, sooner or later your application will need to handle DOM events like mouse clicks, moves, keyboard presses, change events, etc.

- AngularJS can add functionality which can be used to handle such events.

- For example, if there is a button on the page and you want to process something when the button is clicked, we can use the ng-click event directive.

- ng-disabled
- ng-click
- ng-show
- ng-hide

AngularJS Events:

What is ng-disabled directive?

- The ng-disabled directive binds AngularJS application data to the disabled attribute of HTML elements.

- Example:

```
<div ng-app="" ng-init="mySwitch=true">
<p>
<button ng-disabled="mySwitch">Click Me!</button>
</p>

<p>
<input type="checkbox" ng-model="mySwitch">Button
</p>

<p>
   {{ mySwitch }}
</p>
</div>
```

AngularJS Events:

What is ng-disabled directive?

Code Explanation:

- The ng-disabled directive binds the application data mySwitch to the HTML button's disabled attribute.
- The ng-model directive binds the value of the HTML checkbox element to the value of mySwitch.
- If the value of mySwitch evaluates to true, the button will be disabled:

AngularJS Events:

What is ng-click directive?

- The "ng-click directive" is used to apply custom behavior to when an element in HTML clicked.

- This is normally used for buttons because that is the most common place for adding events which respond to clicks performed by the user

- Example:

```
<button ng-click="count = count + 1" ng-init="count=0">
    Increment
</button>

<div>The Current Count is {{count}}</div>
```

AngularJS Events:

What is ng-click directive?

Code Explanation:

- We are first using the ng-init directive to set the value of a local variable count to 0.
- We are then introducing the ng-click event directive to the button. In this directive, we are writing code to increment the value of the count variable by 1.
- Here we are displaying the value of the count variable to the user.

AngularJS Events:

What is ng-show directive?

- ng-Show directive is used to show or hide a given HTML element based on the expression provided to the ngShow attribute.

- In the background, the element is shown or hidden by removing or adding the .ng-hide CSS class onto the element.

AngularJS Events:

What is ng-show directive?

Example:

```
<input type="button" value="Show Angular" ng-click="ShowHide()"/>

    <br><br><div ng-show = "IsVisible">Angular</div>
</div>

<script type="text/javascript">

    var app = angular.module('DemoApp',[]);

    app.controller('DemoController',function($scope){
        $scope.IsVisible = false;

        $scope.ShowHide = function(){
            $scope.IsVisible = true;
        }
        });
</script>
```

AngularJS Events:

What is ng-show directive?

Code Explanation:

- We are attaching the ng-click event directive to the button element. Over here we are referencing a function called "ShowHide" which is defined in our controller – DemoController.
- We are attaching the ng-show attribute to a div tag which contains the text Angular.
- This is the tag which we are going to show/hide based on the ng-show attribute.
- In the controller, we are attaching the "IsVisible" member variable to the scope object. This attribute will be passed to the ng-show angular attribute (step#2) to control the visibility of the div control. We are initially setting this to false so that when the page is first displayed the div tag will be hidden.

- Note:- When the attributes ng-show is set to true, the subsequent control which in our case is the div tag will be shown to the user. When the ng-show attribute is set to false the control will be hidden from the user.

- We are adding code to the ShowHide function which will set the IsVisible member variable to true so that the div tag can be shown to the user.

AngularJS Events:

What is ng-show directive?

Code Explanation:

- From the output,

- You can initially see that the div tag which has the text "AngularJS" is not shown and this is because the isVisible scope object is initially set to false which is then subsequently passed to the ng-show directive of the div tag.

- When you click on the "Show AngularJS" button, it changes the isVisible member variable to become true and hence the text "Angular" becomes visible to the user. The below output will be shown to the user.

AngularJS Events:

What is ng-hide directive?

- With the ng-hide directive an element will be hidden if the expression is TRUE. If the Expression is FALSE the element will be shown.

- In the background, the element is shown or hidden by removing or adding the .ng-hide CSS class onto the element.

- On the other hand with ng-hide, the element is hidden if the expression is true and it will be shown if it is false.

AngularJS Events:

What is ng-hide directive?

- Example:

```
<input type="button" value="Hide Angular" ng-click="ShowHide()"/>

    <br><br><div ng-hide="IsVisible">Angular</div>
</div>

<script type="text/javascript">

    var app = angular.module('DemoApp',[]);

    app.controller('DemoController',function($scope){
        $scope.IsVisible = false;

        $scope.ShowHide = function(){
            $scope.IsVisible = $scope.IsVisible = true;
        }
        });
</script>
```

AngularJS Events:

What is ng-hide directive?

- Code Explanation:

- We are attaching the ng-click event directive to the button element. Over here we are referencing a function called ShowHide which is defined in our controller – DemoController.

- We are attaching the ng-hide attribute to a div tag which contains the text Angular. This is the tag, which we are going to show/hide based on the ng-show attribute.

- In the controller, we are attaching the isVisible member variable to the scope object. This attribute will be passed to the ng-show angular attribute to control the visibility of the div control. We are initially setting this to false so that when the page is first displayed the div tag will be hidden.

- We are adding code to the ShowHide function which will set the IsVisible member variable to true so that the div tag can be shown to the user.

AngularJS Event Listener Directives

You can add AngularJS event listeners to your HTML elements by using one or more of these directives:

- ng-blur
- ng-change
- ng-click
- ng-copy
- ng-cut
- ng-dblclick
- ng-focus
- ng-keydown
- ng-keypress
- ng-keyup
- ng-mousedown
- ng-mouseenter
- ng-mouseleave
- ng-mousemove
- ng-mouseover
- ng-mouseup
- ng-paste

AngularJS Event Listener Directives

- Events directives are used in Angular to add custom code to respond to events generated by user intervention such as button clicks, keyboard and mouse clicks, etc.

- The most common event directive is the ng-click directive which is used to handle click events. The most common use of this is for button clicks wherein code can be added to respond to a button click.

- HTML elements can also be hidden or shown to the user accordingly by using the ng-show and ng-hide angular attributes.

AngularJS Forms

- Forms in AngularJS provides data-binding and validation of input controls.

- Input controls are the HTML input elements:

- input elements
- select elements
- button elements
- textarea elements

AngularJS Forms

Data-Binding

- Input controls provides data-binding by using the ng-model directive.

- <input type="text" ng-model="firstname">
- The application does now have a property named firstname.

- The ng-model directive binds the input controller to the rest of your application.

- The property firstname, can be referred to in a controller:

Example

```
<script>
var app = angular.module('myApp', []);
app.controller('formCtrl', function($scope) {
  $scope.firstname = “ABCD";
});
</script>
```

AngularJS Forms

Checkbox

A checkbox has the value true or false. Apply the ng-model directive to a checkbox, and use its value in your application.

Example

Show the header if the checkbox is checked:

```
 <form>
  Check to show a header:
  <input type="checkbox" ng-model="myVar">
</form>

<h1 ng-show="myVar">My Header</h1>
```

AngularJS Forms

Radiobuttons
Bind radio buttons to your application with the ng-model directive.

Radio buttons with the same ng-model can have different values, but only the selected one will be used.

The ng-switch directive hides and shows HTML sections depending on the value of the radio buttons.

Example
Display some text, based on the value of the selected radio button:

```
<form>
  Pick a topic:
  <input type="radio" ng-model="myVar" value="dogs">Dogs
  <input type="radio" ng-model="myVar" value="tuts">Tutorials
  <input type="radio" ng-model="myVar" value="cars">Cars
</form>
```

AngularJS Forms

Selectbox
Bind select boxes to your application with the ng-model directive.

The property defined in the ng-model attribute will have the value of the selected option in the selectbox.

Example
Display some text, based on the value of the selected option:

```
<form>
  Select a topic:
  <select ng-model="myVar">
    <option value="">
    <option value="dogs">Dogs
    <option value="tuts">Tutorials
    <option value="cars">Cars
  </select>
</form>
```

AngularJS Form Example

Form.htm

```
<script>
var app = angular.module('myApp', []);
app.controller('formCtrl', function($scope) {
  $scope.master = {firstName: “Amit", lastName: “Jain"};
  $scope.reset = function() {
    $scope.user = angular.copy($scope.master);
  };
  $scope.reset();
});
</script>
```

AngularJS Forms

Example Explained:

- The ng-app directive defines the AngularJS application.
- The ng-controller directive defines the application controller.
- The ng-model directive binds two input elements to the user object in the model.
- The formCtrl controller sets initial values to the master object, and defines the reset() method.
- The reset() method sets the user object equal to the master object.
- The ng-click directive invokes the reset() method, only if the button is clicked.
- The novalidate attribute is not needed for this application, but normally you will use it in AngularJS forms, to override standard HTML5 validation.

AngularJS Forms and Validation

- AngularJS offers client-side form validation.
- AngularJS monitors the state of the form and input fields (input, textarea, select), and lets you notify the user about the current state.
- AngularJS also holds information about whether they have been touched, or modified, or not.
- You can use standard HTML5 attributes to validate input, or you can make your own validation functions.
- Client-side validation cannot alone secure user input. Server side validation is also necessary.

AngularJS Forms and Validation

Required

- Use the HTML5 attribute required to specify that the input field must be filled out:

- Example
  The input field is required:

```
<form name="myForm">
  <input name="myInput" ng-model="myInput" required>
</form>

<p>The input's valid state is:</p>
<h1>{{myForm.myInput.$valid}}</h1>
```

AngularJS Forms and Validation

E-mail

Use the HTML5 type email to specify that the value must be an e-mail:

Example

The input field has to be an e-mail:

```
<form name="myForm">
  <input name="myInput" ng-model="myInput" type="email">
</form>

<p>The input's valid state is:</p>
<h1>{{myForm.myInput.$valid}}</h1>
```

AngularJS Forms and Validation

Form State and Input State:

AngularJS is constantly updating the state of both the form and the input fields.

Input fields have the following states:

$untouched The field has not been touched yet

$touched The field has been touched

$pristine The field has not been modified yet

$dirty The field has been modified

$invalid The field content is not valid

$valid The field content is valid

They are all properties of the input field, and are either true or false.

AngularJS Forms and Validation

CSS Classes

AngularJS adds CSS classes to forms and input fields depending on their states.

The following classes are added to, or removed from, input fields:

ng-untouched The field has not been touched yet

ng-touched The field has been touched

ng-pristine The field has not been modified yet

ng-dirty The field has been modified

ng-valid The field content is valid

ng-invalid The field content is not valid

ng-valid-key One key for each validation.

Example: ng-valid-required, useful when there are more than one thing that must be validated

ng-invalid-key Example: ng-invalid-required

AngularJS Forms and Validation

Accessing AngularJS Properties of Form and Input Fields

We can access properties of form and input fields for validations like as shown following

- Access the form properties like <form name>.<angular property>

- Access the Input field properties like <form name>.<input name>.<angular property>

AngularJS Forms and Validation

Directive	Description
ng-required	Sets required attribute on an input field.
ng-minlength	Sets minlength attribute on an input field.
ng-maxlength	Sets maxlength attribute on an input field. Setting the attribute to a negative or non-numeric value, allows view values of any length.
ng-pattern	Sets pattern validation error key if the ngModel value does not match the specified RegEx expression.

AngularJS Forms and Validation

Property	Description
$error	$error object contains all the validation attributes applied to the specified element.
$pristine	Returns true if the user has not interacted with control yet else returns false.
$valid	Returns true if the model is valid
$invalid	Returns true if the model is invalid
$dirty	Returns true if user changed the value of model at least once
$touched	Returns true if the user has tabbed out from the control.
$untouched	Returns true if the user has not tabbed out from the control.

AngularJS ng-template Directive

- In angularjs, the ng-template directive is used to load the content of a <script> element into $templateCache, and this is done further by using ng-include, ng-view, template, etc.

There is one rule which we need to follow when we use ng-template directive in angularjs application.

- In ng-template the type of the <script> element must be specified as text/ng-template, and a cache name for the template must be assigned through the element's id, which can then be used as a directive's templateUrl.

- In the above example we used a ng-template in script tag to load the HTML. It contains "id" attribute which is used by $routeProvider to map the view with a controller.

AngularJS ng-options Directive to Bind Select / Dropdown List

- In angularjs ng-options directive is used to generate list of options and bind to select or dropdown list.
- In angularjs ng-options directive will use array list to bind dropdown list option values.
- Generally in angularjs most of the cases we use ng-repeat directive to achieve same functionality of ng-options but ng-options will reduce memory usage and increase speed of application by not creating scope for every repeat instance when compared with ng-repeat directive.
- While using ng-options in angularjs optionally we need to define one hard coded <option> element with value as empty string in <select> element.
- This hard coded empty string value will represent starting value of dropdown list.

AngularJS ng-options Directive to Bind Select / Dropdown List

```
<select ng-options="array expression">

<option value="">--Select--</option>

</select>
```

AngularJS Routing

The ngRoute module helps your application to become a Single Page Application.

What is Routing in AngularJS?

- If you want to navigate to different pages in your application, but you also want the application to be a SPA (Single Page Application), with no page reloading, you can use the ngRoute module.

- The ngRoute module routes your application to different pages without reloading the entire application.

AngularJS Routing

```
<body ng-app="myApp">

<p><a href="#/!">Main</a></p>

<a href="#!red">Red</a>
<a href="#!green">Green</a>
<a href="#!blue">Blue</a>

<div ng-view></div>
```

AngularJS Routing

```
<script>
var app = angular.module("myApp", ["ngRoute"]);
app.config(function($routeProvider) {
  $routeProvider
  .when("/", {
    templateUrl : "main.htm"
  })
  .when("/red", {
    templateUrl : "red.htm"
  })
  .when("/green", {
    templateUrl : "green.htm"
  })
  .when("/blue", {
    templateUrl : "blue.htm"
  });
});
</script>
</body>
```

AngularJS Routing

What do I Need?

- To make your applications ready for routing, you must include the AngularJS Route module:

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular-route.js"></script>
```

- Then you must add the ngRoute as a dependency in the application module:

```
var app = angular.module("myApp", ["ngRoute"]);
```

AngularJS Routing

What do I Need?

- Now your application has access to the route module, which provides the $routeProvider to configure different routes in your application:

```
app.config(function($routeProvider) {
  $routeProvider
  .when("/", {
    templateUrl : "main.htm"
  })
  .when("/red", {
    templateUrl : "red.htm"
  })
  .when("/green", {
    templateUrl : "green.htm"
  })
  .when("/blue", {
    templateUrl : "blue.htm"
  });
});
```

AngularJS Routing

Where Does it Go?

- Your application needs a container to put the content provided by the routing.
- This container is the ng-view directive.
- Applications can only have one ng-view directive, and this will be the placeholder for all views provided by the route.

AngularJS Routing

$routeProvider

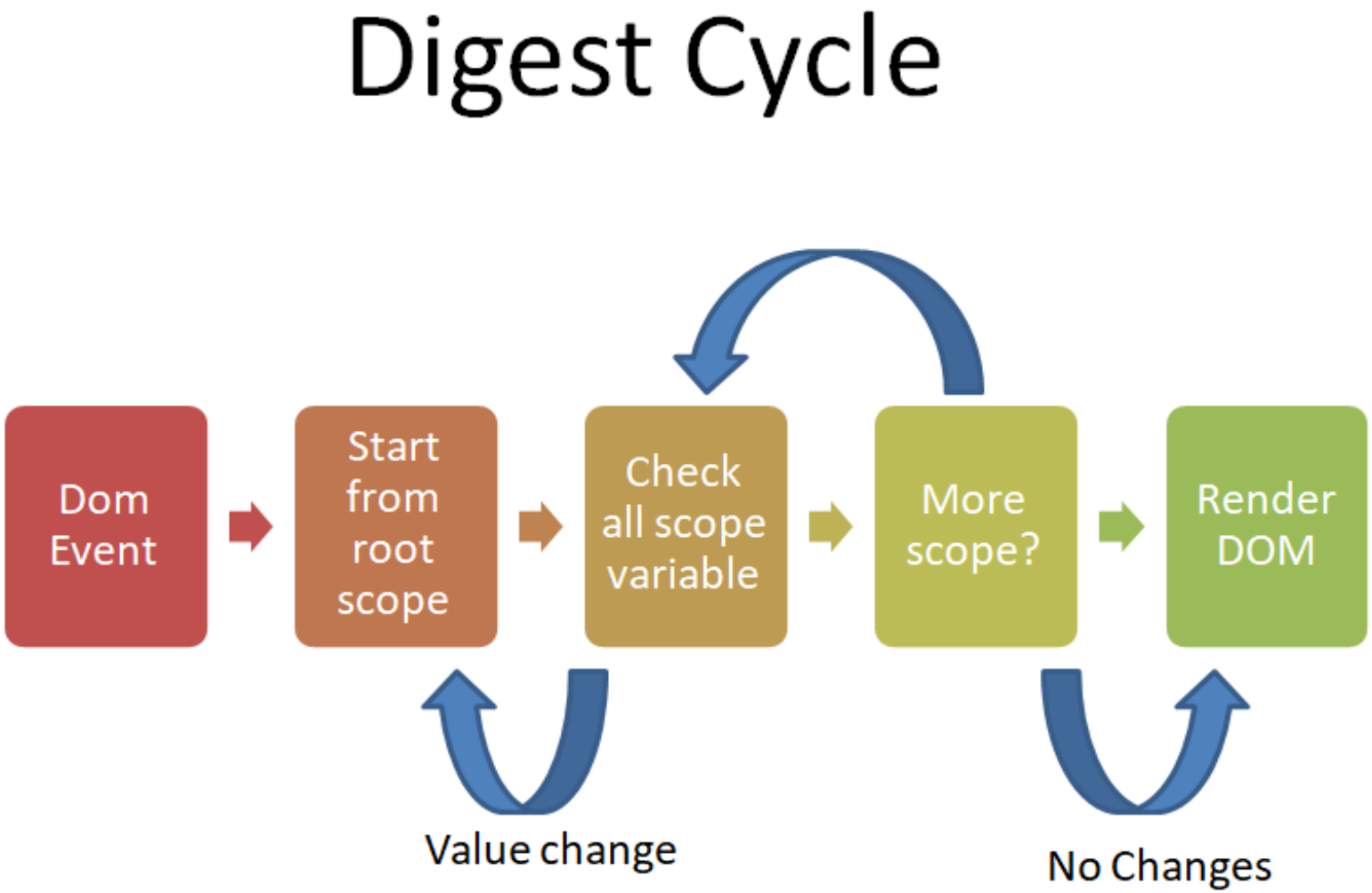
With the $routeProvider you can define what page to display when a user clicks a link.

Example:

Define a $routeProvider:

```
var app = angular.module("myApp", ["ngRoute"]);
app.config(function($routeProvider) {
  $routeProvider
  .when("/", {
    templateUrl : "main.htm"
  })
  .when("/london", {
    templateUrl : "london.htm"
  })
  .when("/paris", {
    templateUrl : "paris.htm"
  });
});
```

What is the digest cycle in AngularJs?



What is the digest cycle in AngularJs?

- In order to work with angularJS, to know how data binding is happening behind the scene, we need a sound understanding of digest cycle.

- Digest process monitors the watchlist to keep track if there any changes in the value of watch variable.

- It compares the present value with its previous value.

- This checking is called “Dirty Checking”.

- If there are any changes it will make a note of it and execute respective watch listener.

- After the execution of watch listener, it will notify the DOM and finally render to the browser. DOM get updated after the digest process.

- Digest process gets executed as a part angular context.

What is the digest cycle in AngularJs?

AngularJs watch:

- Watch variables are keeping track of scope variables along with its value.
- All watches of the same scope are stored in a list called watch list.
- It is a memory variable in angularJS context.
- AngularJS provides us to write custom function while the value of scope variable change.
- This custom function is called watch listener. Every scope has its own watch list.

What is the digest cycle in AngularJs?

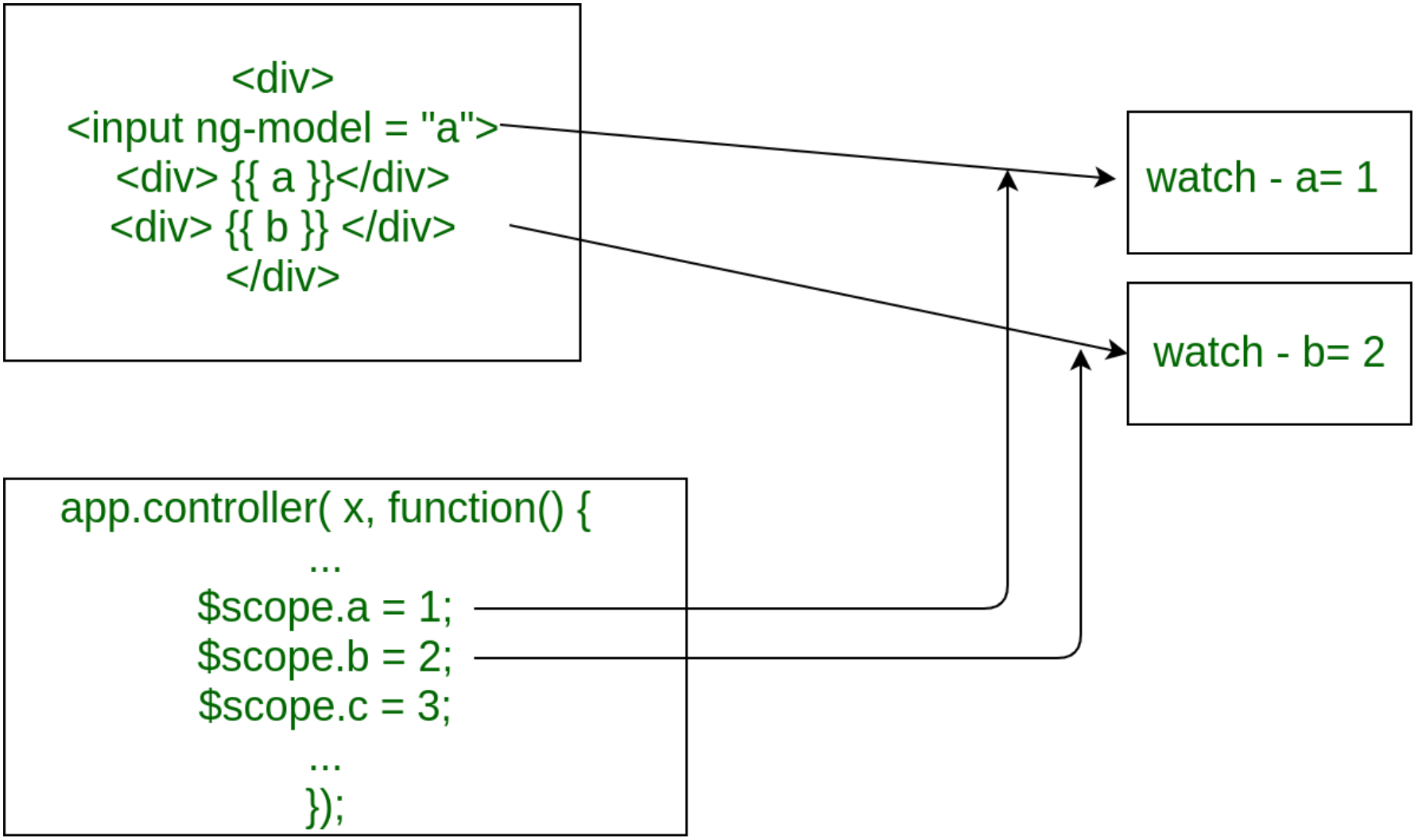
AngularJs watch:

- For example, consider the below code snippet :
- Here scope variable a, b and c are watch variable.

```
var app=angular.Module("myApp",[]);
app.Controller("myCtrl",function($scope){
  $scope.a=5;
  $scope.b=10;
  $scope.c=12;
});
```

What is the digest cycle in AngularJs?

AngularJs watch:



Two watches created for a and b. No watch for c

What is the digest cycle in AngularJs?

Watch counts:

- If watch count is below 2000, performance is better.

- We may use Angular watchers extension to count them. Angular performs watching on data-binded variables but if we want we can perform watch on normal variables as well, using watch function.

- It takes parameter the variable that we explicitly want to watch.

Watch list:

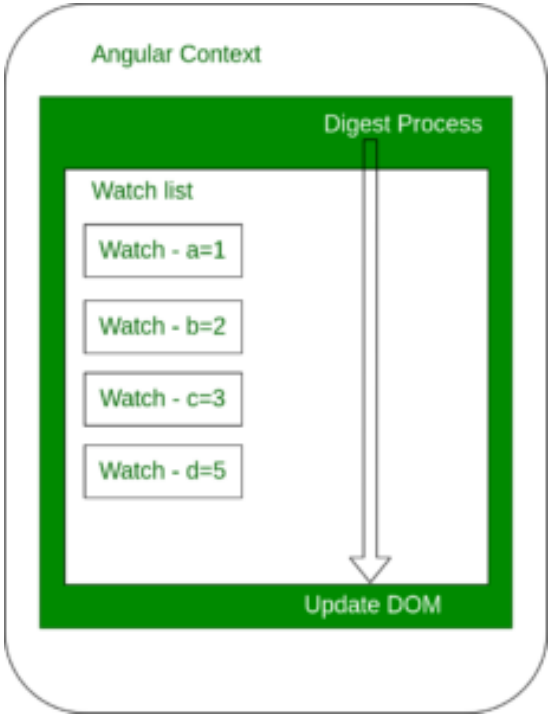
- Maintains a list of all the watches associated with an angular application. i.e all the data bindings being monitored.

- A watch list is maintained for all scopes including root.

What is the digest cycle in AngularJs?

Digest cycle

- Watches keep on updating new values and update DOM thus render the changes.
- This process is responsible for walk-through entire watches for changes. It performs dirty checking over the watches present in the watch-list.
- Dirty checking is to check the current values of variables from their previous values.



What is the digest cycle in AngularJs?

Digest cycle

- Watch listeners are automatically executed whenever the digest process finds any modifications in the variables. It keeps note of changes and then notifies AngularJs Framework to update DOM. Thus at the end of every digest process, DOM is updated.

- Angular Context is a runtime environment of AngularJs Framework.

- First digest process performs a dirty check on watches, and checks if there are any modifications

- It again performs the second cycle of dirty checking on the previous cycle, watches listeners. Because there may have been variables that have got changed by others. Minimum 2 iterations are performed and the maximum 10 can run. Although it is preferred to minimize the digest cycle for better performance. Error is thrown after maximum.

What is the digest cycle in AngularJs?

First level and second level updating

- First level updates: Suppose the variable b was updated by any event, then during the first cycle Digest cycle informs the AngularJs Framework about the changes and goes through the second cycle after that. Since there are no more updates, it thus updates DOM and completes it.

- Second level watch updates: Whenever there is a change encountered in the first cycle for any particular watch say c, digest process executes watch listener for it. Now watch listener further modifies variable a to a new value. After the first cycle, c gets updated. During the second cycle, we encounter changes in a and thus update takes place for a. Now a third cycle takes place and there are no more modifications encountered. DOM gets updated.

What is the digest cycle in AngularJs?

- Considering scope variables a, b, c are data binded and are eligible for the digest process.

- If we inspect the angular application in the browser and open the console. We can track the changes as the print statements will help us.

- Suppose there was a two way binding for c with an input box we could easily track the number of times it gets modified. In fact, we can inspect the digest process too, by applying $watch function on $rootscope.

What is the digest cycle in AngularJs?

- $watch: This function takes three parameters- watch expression, listener and object equality. Except for expression the other two are optional.

- The digest process starts with the root scope and later on identifies the other scopes. If our code uses DOM events (ng-click), ajax with callback, timer with callback, Browser location changes, manual invocations like $apply then it is bound to have Digest process for all of them.

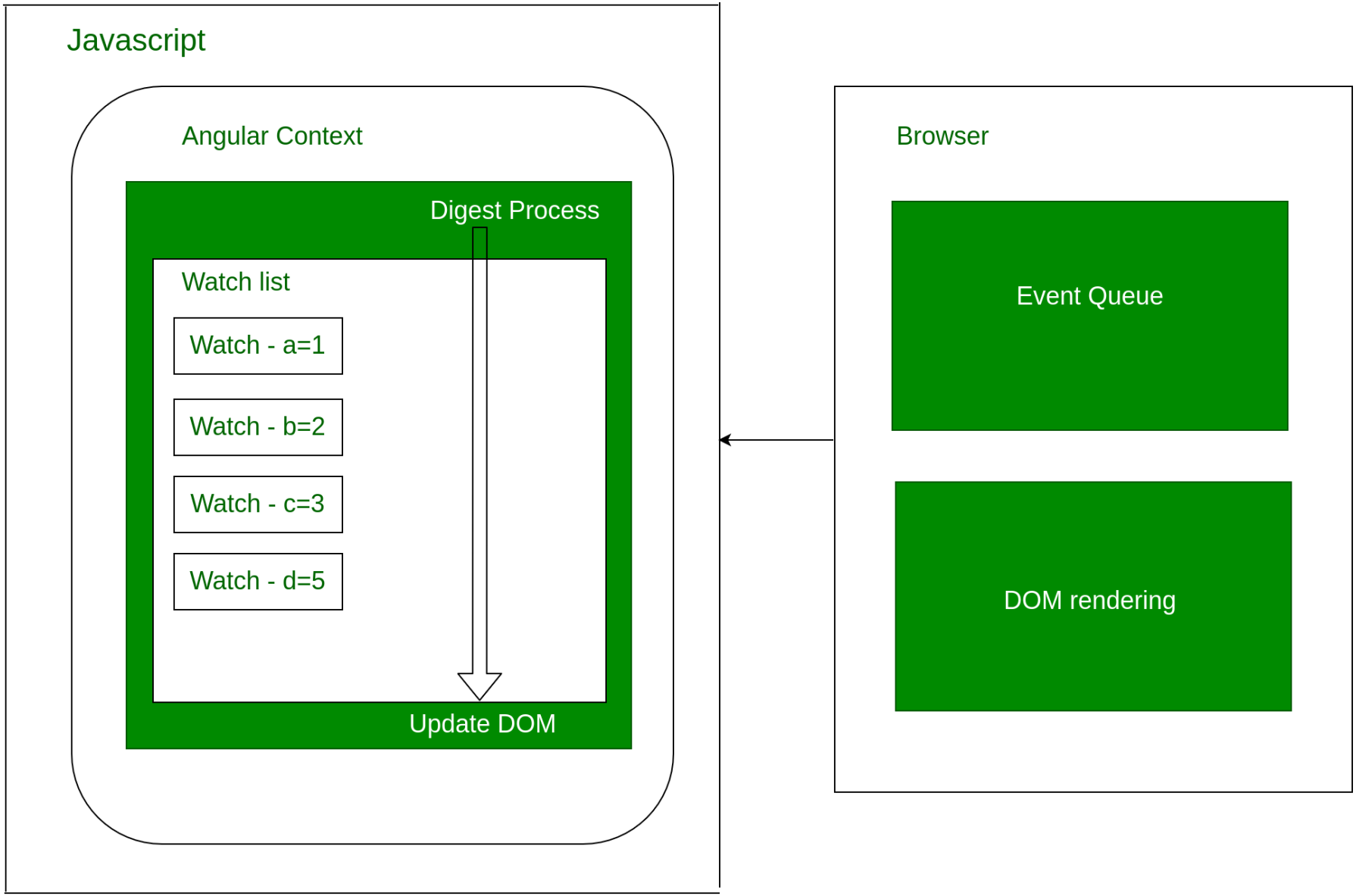
- As we know, the browser is responsible to render the DOM and it may have events like Timer, On-click. The browser maintains a queue for these events called Event Queue. And it sends those to Javascript.

- Consequently, a digest takes place for them. If events are not related to Javascript i.e written in Jquery or other languages, then it is our duty to write $apply function for maintaining digest for them.

```
$scope.$apply(function()
{

});
```

What is the digest cycle in AngularJs?



Reference Links:

- https://www.journaldev.com/6225/angularjs-routing-example-ngroute-routeprovider
- https://docs.angularjs.org/tutorial/step_09
- https://www.tutorialsteacher.com/angularjs/angularjs-service-http
- https://angular.io/guide/router-tutorial
- https://stackblitz.com/angular/nnpxyydpnvv?file=src%2Fapp%2Fhero-list.component.html
- https://www.slideshare.net/sunilos/angular-8