МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«__1__» ___02___ 2018 г.

ИЗУЧЕНИЕ ФРЕЙМВОРКА ANGULAR

Методические указания по выполнению лабораторной работы по дисциплине «Веб-программирование» для студентов специальности 10.05.02 и 10.03.01.

Курск 2018

УДК 004.056.55

Составитель А.Л. Марухленко

Рецензент
Кандидат технических наук, доцент *И.В. Калуцкий*

Изучение фреймаорка Angular: методические указания по выполнению лабораторной работы по дисциплине «Веб-программирование» / Юго-Зап. гос. ун-т; сост. А.Л. Марухленко. Курск, 2018. – 18 с.

Рассматриваются основы использования фреймворка Angular. Указывается порядок выполнения лабораторной работы и содержание отчета.

Методические указания соответствуют требованиям программы, утвержденной учебно-методическим объединением по образованию в области информационной безопасности (УМО ИБ).

Предназначены для студентов специальности 10.05.02 и 10.03.01.

Текст печатается в авторской редакции

Подписано в печать 01.02.18. Формат 60x84 1/16.
Усл.печ. л. 1.04. Уч.-изд.л. 0,94. Тираж 30 экз. Заказ. Бесплатно. 241
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

ЦЕЛЬ РАБОТЫ

Целью выполнения лабораторной работы является формирование у студентов навыков и умений по использованию фреймворка Angular.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

AngularJS – JavaScript-фреймврк с открытым исходным кодом. Предназначен для разработки одностраничных приложений. Его цель – расширение браузерных приложений на основе MVC шаблона, а также упрощение тестирования и разработки.

AngularJS является структурной основой для динамических веб-приложений. Он позволяет использовать HTML в качестве языка шаблонов и позволяет расширять синтаксис HTML, чтобы четко и лаконично выражать компоненты приложения. Увязка данных AngularJS и инъекция зависимостей устраняют большую часть кода, который в противном случае вам приходилось писать. И все это происходит в браузере, что делает его идеальным партнером любой серверной технологии.

AngularJS – это то, что было бы HTML, если бы оно предназначалось для приложений. HTML – отличный декларативный язык для статических документов. Он не содержит много способов создания приложений, и в результате создание веб-приложений - это упражнение в том, что мне нужно сделать, чтобы обмануть браузер, делая то, что я хочу?

AngularJS преподает новый синтаксис браузера через конструкцию, которую мы называем директивами.

На директивах держится практически вся декларативная часть AngularJS. Именно они используются для обогащения синтаксиса HTML. В процессе компиляции DOM директивы берутся из HTML и исполняются. Директивы могут добавить какое-то новое поведение и/или модифицировать DOM. В стандартную поставку входит достаточно большое количество директив для построения веб приложений. Но ключевой особенностью является возможность разработки своих директив, за счет чего HTML может быть превращен в DSL.

Директивы именуются с помощью lowerCamelCase, например,

ngBind. При использовании директиву необходимо именовать в нижнем регистре с использованием в качестве разделителя одного из спец символов: :, -, или _. По желанию для получения валидного кода можно использовать префиксы x- или data-. Примеры: ng:bind, ng-bind, ng_bind, x-ng-bind и data-ng-bind.

Директивы могут использоваться как элемент (<my-directive></my-directive>), атрибут (), в классе () или в комментарии ()[4]. Это зависит от того, как конкретная директива была разработана:

ng-app – объявляет элемент корневым для приложения.

ng-bind – автоматически заменяет текст HTML-элемента на значение переданного выражения.

ng-model – то же что и ng-bind, только обеспечивает двустороннее связывание данных. Изменится содержимое элемента, ангулар изменит и значение модели. Изменится значение модели, ангулар изменит текст внутри элемента.

ng-class – определяет классы для динамической загрузки.

ng-controller – определяет JavaScript-контроллер для вычисления HTML-выражений.

ng-repeat – создает экземпляр для каждого элемента из коллекции.

ng-show и ng-hide – показывает или скрывает элемент в зависимости от значения логического выражения.

ng-switch – создает экземпляр шаблона из множества вариантов, в зависимости от значения выражения.

ng-view – базовая директива, отвечает за обработку маршрутов, которые принимают JSON перед отображением шаблонов, управляемых указанными контроллерами.

$s**cope – это объект, имеющий отношение к модели в приложении. Он является контекстом выполнения для выражений. Scope-ы выстраиваются в иерархическую структуру, похожую на DOM. При этом они наследуют свойства от своих родительских **$scope.

**$s**cope являются как бы «клеем» между контроллером и представлением. В процессе выполнения фазы связывания шаблона директивы устанавливают наблюдение ($watch) за выражениями в рамках scope. $watch дает директивам возможность реагировать на изменения для отображения обновленного значения или каких-либо

других действий. И контроллеры, и директивы имеют ссылку на **$s**cope, но не имеют ссылок друг на друга. Так контроллеры изолируются от директив и от DOM-а. За счет этого возрастают возможности по тестированию приложения.

Безопасность приложений.

В любом веб-приложении необходимо предусматривать меры за-щиты конфиденциальной информации от несанкционированного доступа. Единственным, по-настоящему безопасным местом в таких приложениях является сервер. За его пределами необходимо учитывать возможность взлома программного кода и вследствие этого предусматривать проверки в точке, где данные поступают или покидают сервер. Необходимые меры безопасности, которые следует предпринять и на стороне сервера, и на стороне клиента, перечисленные ниже:

– предотвращение на стороне сервера несанкционированного доступа к данным и разметке HTML;

– шифрование трафика с целью исключить перехват пакетов;

– предотвращение атак типа «межсайтовый скриптинг» (Cross-Site scripting, XSS) и «подделка межсайтовых запросов» (Cross-Site Request Forgery, XSRF);

– блокирование попыток внедрения кода JSON.

На стороне сервера меры по обеспечению безопасности должны осуществляться всегда, но это не значит, что они не нужны на стороне клиента – мы обязаны предусматривать подобные меры и в пользовательском интерфейсе приложения, чтобы явно показать, что пользователь может иметь доступ только к определенному кругу функциональных возможностей, соответствующих его привилегиям. Кроме того, мы обязаны реализовать ясную процедуру аутентификации, не мешающую пользователю взаимодействовать с приложением. Важными являются вопросы обеспечения безопасности с применением средств фреймворка AngularJS, в том числе:

– различия в обеспечении безопасности полнофункциональных клиентских приложений и более традиционных приложений, опирающихся на хранение информации на стороне сервера;

– обработка ошибок авторизации на сервере посредством перехвата HTTP-ответов;

– ограничение доступа к разделам путем обеспечения безопас-ности маршрутов;

Предотвращение перехвата cookie («атака через посредника»).

Всякий раз, когда между клиентом и сервером осуществляется передача данных по протоколу HTTP, существует вероятность вмешательства третьей стороны с целью прочитать конфиденциальную информацию или, хуже того, cookies авторизации, чтобы перехватить управление сеансом и получить возможность обращаться к серверу от вашего имени. Нападения такого вида часто называют «атака через посредника» («man-in-the-middle»). Самый простой способ предотвращения таких нападений – использование протокола HTTPS вместо HTTP.

Шифруя соединение посредством протокола HTTPS , мы исключаем возможность чтения конфиденциальных данных посторонними лицами, оказавшимися между сервером и клиентом, а также не позволяем неавторизованным пользователям получать блоки cookies аутентификации, с помощью которых они могли бы перехватить управление сеансом.

Нападения вида «межсайтовый скриптинг» (Cross-Site Scripting, XSS) заключаются во внедрении злонамеренного сценария в веб-страницу при просмотре ее другим пользователем. Наибольший вред от таких нападений может быть нанесен, если внедренный сценарий сможет обращаться к серверу, потому что сервер будет считать эти обращения аутентифицированными и выполнять их.

Существует большое разнообразие нападений XSS. Чаще всего используется разновидность, основанная на отображении получаемого из сети содержимого без надлежащего экранирования, предотвращающего интерпретацию специально подготовленной разметки HTML.

Предотвращение подделки межсайтовых запросов.

В любом приложении, где сервер оказывает доверие зарегистрированному пользователю и позволяет выполнять некоторые операции на сервере, полагаясь на это доверие, существует вероятность, что другие сайты смогут получить доступ к этим операциям и выполнять их от вашего имени. Эта разновидность нападений называется «подделка межсайтовых запросов» (Cross-Site Request Forgery, XSRF). Если пользователь

посетит злонамеренный сайт, когда он уже прошел процедуру аутентификации на защищенном сайте, веб-страница со злонамеренного сайта сможет послать защищенному сайту запрос, поскольку в данный момент вы считаетесь аутентифицированным пользователем.

Такие нападения часто реализуются в виде мошеннического атрибута src в теге <img>, который пользователь может загрузить по неосторожности при переходе на злонамеренную страницу, не закрывая сеанс работы с защищенным сайтом. Когда браузер попытается загрузить изображение, он фактически выполнит запрос к защищенному сайту.

<imgsrc=”http://my.securesite.com/createAdminUser?username=badguy”>;

Чтобы устранить эту проблему, сервер может передавать браузеру секретный ключ, доступный только сценарию на JavaScript, выполняющемуся в браузере, и недоступный в атрибуте src. При обращении к серверу этот ключ должен включаться в заголовки запросов, чтобы подтвердить аутентичность пользователя.

Служба $http уже реализует это решение для защиты от нападений подобного рода. Чтобы его активизировать, необходимо на стороне сервера обеспечить передачу в сеансовом блоке cookie параметра с именем XSRF-TOKEN, в ответ на первый GET-запрос приложения. Значение этого параметра должно быть уникальным для данного сеанса.

На стороне клиента служба $http будет извлекать этот ключ из cookie и добавлять его в каждый HTTP-запрос в виде заголовка X-XSRF-TOKEN. Сервер должен проверять ключ в каждом запросе и блокировать доступ, если он окажется недействительным. Разработчики AngularJS рекомендуют использовать этот ключ в дополнение к cookie аутентификации, чтобы повысить защищенность приложения.

Служба security – это созданный нами компонент, реализующий основной прикладной интерфейс для управления открытием и закрытием сеанса, и для получения информации о текущем пользователе.

Эту службу можно внедрять в контроллеры и директивы, которые в свою очередь могут добавлять перечисленные ниже

свойства и методы в объект контекста, чтобы обеспечить доступ к ним из шаблонов.

currentUser: свойство с информацией о текущем, аутентифицированном пользователе;

getLoginReason(): этот метод возвращает локализованное сообщение, объясняющее необходимость аутентификации, например: «Текущий пользователь не авторизован».

showLogin(): этот метод отображает форму аутентификации.

Вызывается в ответ на щелчок на кнопке Login(Войти), а также при получении HTTP-ответа HTTP 401 Unauthorized;

login(email, password): этот метод отправляет указанные параметры на сервер для аутентификации. Вызывается, когда пользователь отправляет форму аутентификации. В случае успеха форма закрывается и повторяется попытка выполнить запрос, вызвавший появление этой формы.

logout(redirectTo): этот метод закрывает текущий сеанс работы с пользователем и выполняет переход по указанному маршруту. Вызывается в ответ на щелчок на кнопке Log out (Выйти).

cancelLogin(redirectTo): этот метод прерывает попытку открыть сеанс, аннулирует все неавторизованные запросы, приведшие к появлению формы аутентификации, и выполняет переход по указанному маршруту. Вызывается, когда пользователь закрывает форму аутентификации или щелкает на кнопке Cancel(Отмена).

Предотвращение переходов по защищенным маршрутам.

Предотвращение доступа к защищенным маршрутам с использованием кода на стороне клиента не обеспечивает необходимый уровень безопасности. Единственный надежный способ, гарантирующий невозможность попадания неавторизованных пользователей в защищенные разделы приложения, требует перезагрузки страницы, чтобы дать серверу возможность отказать в доступе. Но перезагрузка страницы не является идеальным вариантом, потому что сводит на нет все достоинства полнофункциональных клиентских приложений. На деле, так как мы можем защитить данные, отображаемые перед пользователем, не так важно блокировать доступ к маршрутам с переадресацией на сервер. Вместо этого можно просто запретить переход неавторизованного пользователя к защищенному маршруту

на стороне клиента, в момент изменения маршрута[5].

Практическая часть:

AngularJS можно скачать на официальном сайте angularjs.org на главной странице, нажав на кнопку download. Скачанную библиотеку помещаем в папку libraries. Создадим файл Helloworld.html. Нам необходимо подключить библиотеку angular с помощью элемента Script:

<script src="./js/angular.js"></script>

Далее размещаем свой элемент Script, в котором будет происходить настройка и подготовка нашего кода для работы.

Создадим переменную:

var *model* = "Hello Word";

Ниже создадим модуль. Библиотека angular.js представлена одним глобальным объектом angular. Если мы хотим получить что-то от библиотеки angular.js, нам необходимо обратиться к этому объекту и вызвать на нем специальный метод. AngularJS приложения строятся из строительных блоков, которые называются модулями. Когда мы вызываем метод module, мы должны передать ему несколько параметров: первый – имя модуля, второй – зависимости модуля. Если не указать второй параметр, то это означает, что мы хотим найти в JavaScript приложении модуль с названием, который мы указали, как первый параметр.

В нашем случает мы создали модуль:

var *HelloWorldApp* = angular.*module*(”helloworldApp”, []);

В этот модуль потом помещаются все строительные блоки AngularJS приложения: контроллеры, директивы, фильтры, сервисы и так далее. Для удобства модуль сохранили в переменную. Для того чтобы наше приложение начало работать именно с этого модуля, в элементе html мы добавляем директиву **ng-app:**

<html ng-app="HelloWordApp">.

Это означает, что наше приложение, которое будет работать на это html страничке , оно представлено модулем HelloWordApp.

Теперь нам нужно создать полезную функциональность приложения. Контроллеры создаются следующим образом:

```
HelloWorldApp.controller("HellowWorldCtrl", function($scope){
$scope.message = model;
```

```
});
```

Мы вызываем метод controller на уже существующем модуле. Первым аргументом мы передаем имя контроллера,а вторым аргументом функцию. Контроллер у нас называется HellowWorldCtrl, после запятой — поведение(функция), которое определяет наш контроллер. HellowWorldCtrl будет использоваться в разметке для того, чтобы определить какая часть пользовательского интерфейса контролируется этим контроллером. Далее в элементе body должна присутствовать еще одна директива **ng-controller**, в которой указывается имя ранее созданного контроллера. Благодаря этому имени AngularJS приложение будет понимать, что те действия и те данные, которые будут находится в элементе body, адресовываются HellowWorld контроллеру.

Таким образом мы получили следующий код:

```
<html ng-app="HelloWordApp">
<head>
   <meta charset="utf-8">
   <title>HelloWord</title>

   <script src="./js/angular.js"></script>
   <script>
      //Модель
      var model = "Hello Word";
      var helloworldApp = angular.module("helloworldApp", []);
      //Контроллер
      helloworldApp.controller("HellowWorldCtrl", function($scope) {
         $scope.message = model;
      });
   </script>
</head>

<body ng-controller="HellowWorldCtrl">
    <h1> {{message}}</h1>
</body>
</html>
```

На странице у нас изображается:

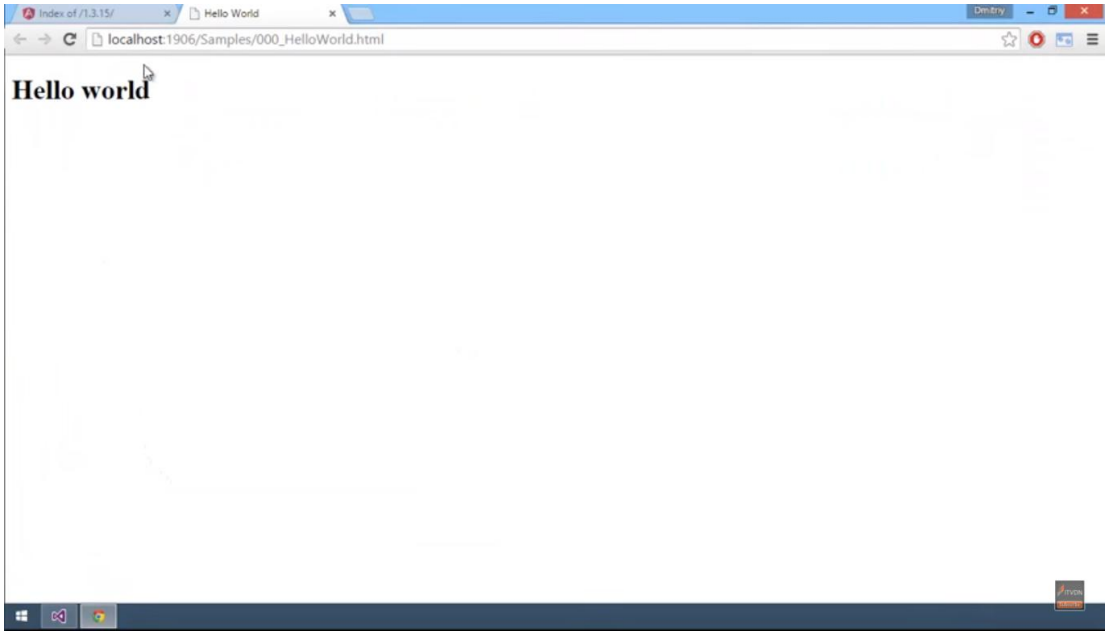

Рисунок 1 – Это пример простейшего angular js приложения.

ЗАДАНИЕ И ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Разработать простейшее приложение при помощи Angular.js, которое будет выполнять следующие действия:

Таблица 6 – Варианты заданий.

№	Дано	Найти
0.	Число	Простое оно или составное
1.	Две стороны треугольника и угол между ними	Длину третьей стороны и площадь этого треугольника
2.	Четырёхзначное число	Произведение всех цифр числа
3.	Катеты прямоугольного треугольника	Периметр и площать треугольника
4.	Два действительных числа	Среднее арифметическое и среднее геометрическое этих чисел
5.	Два действительных числа	Среднее арифметическое этих чисел и среднее геометрическое их модулей
6.	Гипотенуза и катет прямоугольного треугольника	Второй катет и радиус вписанной окружности
7.	Внешний радиус кольца равен числу r (r>20)	Площадь кольца, внутренний радиус которого равен 20
8.	Длина окружности	Площадь круга
9.	Основания и высота равнобедренной трапеции	Её периметр и площадь
10.	Три положительных числа	Дробную часть среднего арифметического трех заданных положительных чисел
11.	Сторона равностороннего треугольника	Площадь и периметр треугольника
12.	Сторона квадрата. В квадрат вписана окружность	Площадь квадрата и окружности, вписанной в этот квадрат
13.	Радиус окружности. В окружность вписан	Найти площади окружности и квадрата

	квадрат	
14.	Пятизначное число	Сумму всех цифр
15.	Четырёхзначное число	Записать цифры числа в обратной последовательности

Требуется создать страницу, которая должна включать в себя возможность проверки вводимого числа на простоту.

Для наглядности возможностей Angular.js сначала выполним это задание без использования этой библиотеки.

Напишем простой код для визуальной оболочки:

Код Index.html:

```
<html>
<head>
   <meta charset="utf-8">
   <title>Проверка числа</title>
   <link href="css/style.css" rel="stylesheet">
   <script src="js/script.js"></script>
   <script src="js/jquery-2.1.4.min.js"></script>
</head>
<body>
   <div class="center">
      <h4>Проверка числа</h4>
      <input id="data" class="center" type="text" maxlength="10"
placeholder="Введите число">  <!-- поле для ввода числа -->
      <button type="submit" class="proverka" onclick="proverka()"
>Проверка</button>          <!-- кнопка проверки -->
      <br>
      <br>
      <input id="result" class="center" type="text" value="" readonly
placeholder="Здесь будет результат"> <!-- поле для вывода резуль-
тата -->
   </div>
</body>
</html>
```

Код, выполняющий проверку числа на простоту:

```
    function proverka() {
var number=$("#data").val();
var root = Math.sqrt(number);
var root_int = Math.round(root);
var leftover;
var is_simple = 1;
   for (var i = 2; i <= root_int; i++) {
      leftover = number % i;
      if (leftover == 0) {
         is_simple = 0;
         break;
      }
   }
   if (is_simple == 1) {
      result = "Число " + number + " простое";
   }
   else {
      result = "Число " + number + " составное";
   }
   $("#result").val(result);
}
```

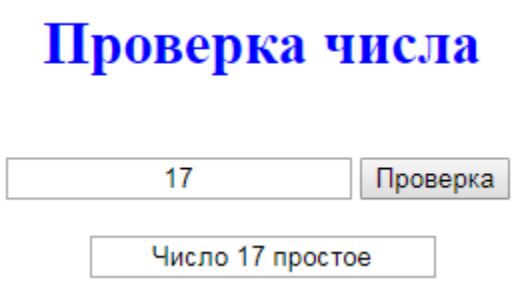

Рисунок 2 – Результат выполнения кода программы.

Используя библиотеку Angular.js, предполагается, что пользователю предлагается вести число, и сразу же появляется информация является ли число простым или составным.

Первом делом, на странице lab.html требуется объявить контроллер lab, который создан в файле app.js Это достигается добавлением к первому использованному тегу директивы ng-controller, которая определяет JavaScript-контроллер для вычисления HTML-выражений, в данном случае контроллер lab.

```
        <body ng-controller="lab">
    <form role = "form">
        <div class = "form-group">
            <label for = "number" id = "number1">Введите число</label>
            <input ng-model="number" type = "number" class = "form-
control" id = "number" placeholder = "Введите число для проверки">
        </div>
    </form>
    <span ng-bind="Func()"></span>
    <div>
        {{result}}
    </div>
</div>
        </body>
```

С помощью фигурных скобок {{}} можно вызывать значение переменных на страницу. Тег <input> является одним из разносторонних элементов формы и позволяет создавать разные элементы интерфейса и обеспечить взаимодействие с пользователем. Главным образом <input> предназначен для создания текстовых полей, различных кнопок, переключателей и флажков. С помощью директивы ng-model и ее значения, заключенных в надстрочные запятые, мы показываем браузеру, что в поле ввода input вводится переменная number, которая принимается на вход контроллера. Атрибут type сообщает браузеру, к какому типу относится элемент формы. Атрибут placeholder своим значением отвечает за то, что будет написано в поле ввода, пока пользователь не внес свои данные. Директива ng-bind - автоматически заменяет текст HTML-элемента на значение переданного выражения, а с помощью {{result}} выводится значение переменной result нашего контроллера:

```
        app.controller("lab", ['$scope',function($scope){
    $scope.title = 'Введите число';
    $scope.number = '';
    $scope.Func=function () {
        var number = $scope.number;
```

```
        var root = Math.sqrt(number);
        var root_int = Math.round(root);
        var leftover;
        var is_simple = 1;
        for (var i = 2; i <= root_int; i++) {
            leftover = number%i;
            if (leftover == 0) {
                is_simple = 0;
                break;
            }
            }
        if (is_simple == 1) {
            $scope.result="Число " + number + " простое";
        }
        else {
            $scope.result="Число " + number + " составное";
        }
    }
}]);
```

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1) AngularJS – фреймворк для динамических веб-приложений от Google [Электронный ресурс] \ Хабрахабр \\Internet – https://habrahabr.ru/post/149060/.

2) Разработка веб-приложений с использованием AngularJS [Текст] \ Козловский П., Бэкон Дарвин П. – М.: Издательство "ДМК Пресс", 2014. – 394 с.

3) Введение в AngularJS [Электронный ресурс] \ Сайт о программировании Internet – https://metanit.com/web/angular/1.1.php.

4) Буч Г. Объектно-ориентированный анализ и проектирование. – М.: Вильямс, 2008.

5) Леоненков А.В. Самоучитель языка UML. – СПб.: БХВ-Петербург, 2004.

6) Розенберг Д., Скотт К. Применение объектного моделирования с использованием UML и анализ прецедентов. – М.: ДМК Пресс, 2002.