FRESNO STATE
Discovery. Diversity. Distinction.

Web Programming (CSci 130)

Department of Computer Science
College of Science and Mathematics
California State University Fresno
H. Cecotti

Learning outcomes

- Javascript frameworks
  - What are the current main frameworks?
  - Focus on AngularJS
    - Basic syntax!

Rationale

- Why to use a framework?
  - **Cost**
    - Open source and free frameworks
    - To help programmers to create custom solutions faster
      - → to lower the cost of the website
      - /!\ to reduce to the amount of work for the Web developer → Less jobs → More difficult/Abstract
  - **Efficiency**
    - To reduce the time to create a project (to not reinvent the wheel)
      - Limited amount of code
  - **Safety**
    - Best frameworks include security arrangements
      - Supported by large communities (and big companies)
        - Members = Testers (= free Beta tester labor)

Introduction

- With JS/PHP, with the project, you may realize
  - Lots of elements in the code are the same = boilerplate code
- Necessary to have an in depth knowledge of Javascript
  - Core knowledge
  - JS Syntax
- Frameworks
  - New versions, new frameworks… it changes all the time
    - AngularJS/Angular1 (2009)
    - Angular2 (2016)
    - Angular4 (2017)
    - Augular8 (2018 ?)
- Tradeoff
  - Use something old that is used in companies vs. Use something new that can be thrown away in 6 months
  - High flexibility of:
    - The developer
    - The produced code

AngularJS

- **Target**: What HTML would be if it was created for dynamic web app
- Frontend/Client side JavaScript application framework
  - Google
- Powerful single page applications (SPA)
  - **Not** to load a new page after each action
  - Fluid experience for the user
- Features
  - Quick code production
  - Easy testing of any app part and two-way data binding
    - Changes in the backend: **immediately** reflected on the UI
  - What it is **Not**
    - A server side framework
    - A library like jQuery

AngularJS

- Free from
  - **Registering callbacks**
    - Removing common boilerplate code like callbacks
    - → reduces the amount of JS coding to do → easier to see what your application does.
  - **Manipulating HTML DOM programmatically**
    - Manipulating HTML DOM = top feature AJAX applications BUT difficult and error-prone
    - Describing how the UI should change as your application state changes → free from low-level DOM manipulation tasks ! ☺ Most applications with AngularJS → **no DOM manipulation**
  - **Marshaling data to and from the UI**
    - **Create/Read/Update/Delete** operations = majority of AJAX operations
    - The flow of marshaling data:
      - server → an internal object → an HTML form → users to modify the form → validating the form → displaying validation errors, returning to an internal model (and back to the server ☺
  - **Writing tons of initialization code just to get started**
    - "Hello World" with AJAX … not so easy
    - AngularJS: bootstrap your app easily using services
      - auto-injected into your application
      - start developing features quickly

AngularJS

- **Advantages**
  - Built by Google
    - Developed and maintained by Google engineers → a huge community for you to learn from + engineers that can help
    - → clients get what they want.
  - Great MVC
    - Most frameworks require programmers to splitting the app into multiple **Model View Controller** components
    - The programmer has to write a code to put them together again.
    - AngularJS strings it together **automatically** → save time
  - Intuitive
    - It makes use of HTML as a declarative language
  - Comprehensive
    - A comprehensive solution for rapid front-end development
    - No need any other plugins or frameworks
    - Arange of other features : Restful actions, data building, dependency injection, enterprise-level testing, …
  - Unit Testing Ready

AngularJS

- **Disadvantages**
  - Confusion
    - There are many ways to do the same thing
  - Lagging UI
    - More than 2000 watchers → UI to lag = complexity of Angular Forms is limited. (big data grids and lists)
  - Name Clashes
    - no ability to compose many NG-apps on the same page → name clashes.

AngularJS

- **Directives**
  - AngularJS teaches the browser new syntax through a construct:
    - ***directives***
  - New attributes to extend HTML
    - Data binding
    - DOM control structures for repeating, showing and hiding DOM fragments.
    - Support for forms and form validation.
    - Attaching new behavior to DOM elements, such as **DOM event handling**.
    - **Grouping** of HTML into reusable components.

AngularJS

- Directives
  - ng-"something"
    - Examples
      - ng-app: initializes an AngularJS application.
      - ng-init: initializes application data.
      - ng-model: binds the value of HTML controls (input, select,…) to application data
      - ng-bind: use the value linked to ng-model
      - ng-repeat: for all the elements in the array
- Expressions
  - AngularJS expressions are written inside double braces: {{ expression }}
  - {{ expression }}

AngularJS

- Example:

```
<!DOCTYPE html>
<html>
<head>
<title>CSci 130</title>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<head>
<body>
    <div ng-app="">
        <p>Input box:</p>
        <p>Name: <input type="text" ng-model="x"></p>
        <p>{{x}}</p>
    </div>
</body>
</html>
```

Input box:

Name: smurf

smurf

AngularJS

- Basic expressions

```
<!DOCTYPE html>
<html>
<head>
<title>CSci 130</title>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
</head>
<body ng-app="">
    <!-- Simple expresions: numbers and strings-->
    <div ng-init="x=1;y=5;firstName='Tim';lastName='O-tea'">
        <p>Expression 1: {{ 4 * 2 +1 }}</p>
        <p>Expression 2: {{ x * y +1 }}</p>
        <p>Expression 3: <span ng-bind="x * y"></span></p>
        <p>Expression 4: Dear Sir {{ firstName + " " + lastName }}</p>
    </div>
    <!-- Object expresions -->
    <div ng-init="person={firstName:'Ben',lastName:'Dover'}">
        <p>Expression 5: {{ person.lastName }}</p>
        <p>Expression 6: <span ng-bind="person.lastName"></span></p>
    </div
    <!-- Array expressions -->
    <div ng-init="v=[3,5,9,7,4]">
        <p>Expression 7: {{ v[3] }}</p>
        <p>Expression 8: <span ng-bind="v[3]"></span></p>
    </div>
</body>
</html>
```

Expression 1: 9

Expression 2: 6

Expression 3: 5

Expression 4: Dear Sir Tim O-tea

Expression 5: Dover

Expression 6: Dover

Expression 7: 7

Expression 8: 7

AngularJS

- Examples
  - Form
    - example01.html
  - Basic form
    - example02.html
  - JSON and variables
    - example03.html
  - A menu
    - example04.html

AngularJS

- What you need:
  - Node.js and npm
    - https://nodejs.org/en/download/ (get the installer)
  - Bower
    - manage components that contain HTML, CSS, JavaScript, fonts or even image files
    - https://bower.io/ (you need npm to get bower)
      - npm install -g bower
- Install Angular
  - npm install angular@1.6.6

User interfaces

- Vue.js
  - https://vuejs.org/v2/guide/
- React
  - Maintained by Facebook, Instagram
  - to provide speed, simplicity, and scalability
  - Issues
    - No proper W3C standards compliance
    - Lack of supports
    - Mixing JS and HTML
    - Code verbosity

Server side

- Node.js
  - It runs scripts server-side to produce dynamic web page content **before** the page is sent to the user's web browser
  - Javascript can be used everywhere (client+server)
  - Javascript + Modules
    - Modules for: file system I/O, networking, data streams…
  - For the creation of web servers

- Node.js vs. PHP
  - PHP functions **block until completion** (e.g. download)
  - Node.js = possibility to execute commands **concurrently**
  - **Event-driven** programming instead of threading

Conclusion

- You cannot learn and master all the frameworks
  - Your base must be strong (JS, PHP) and you must adapt to the situation
  - Change of versions, frameworks → code goes to the bin ☹
- **Do not program from scratch**
  - From a pedagogical view to a pragmatic view
  - "Literature review" before coding
    - What are the current best available tools?
  - Pride of doing **My** project, managing everything vs. Time issues, Cost, …
    - Use a framework → stuck with it ?
- **AJAX, DOM**, …
  - **Core knowledge**, to not throw to the bin !
  - **POST/GET/PUT/DELETE**
- Employers
  - Time + $
  - Awareness of the last frameworks