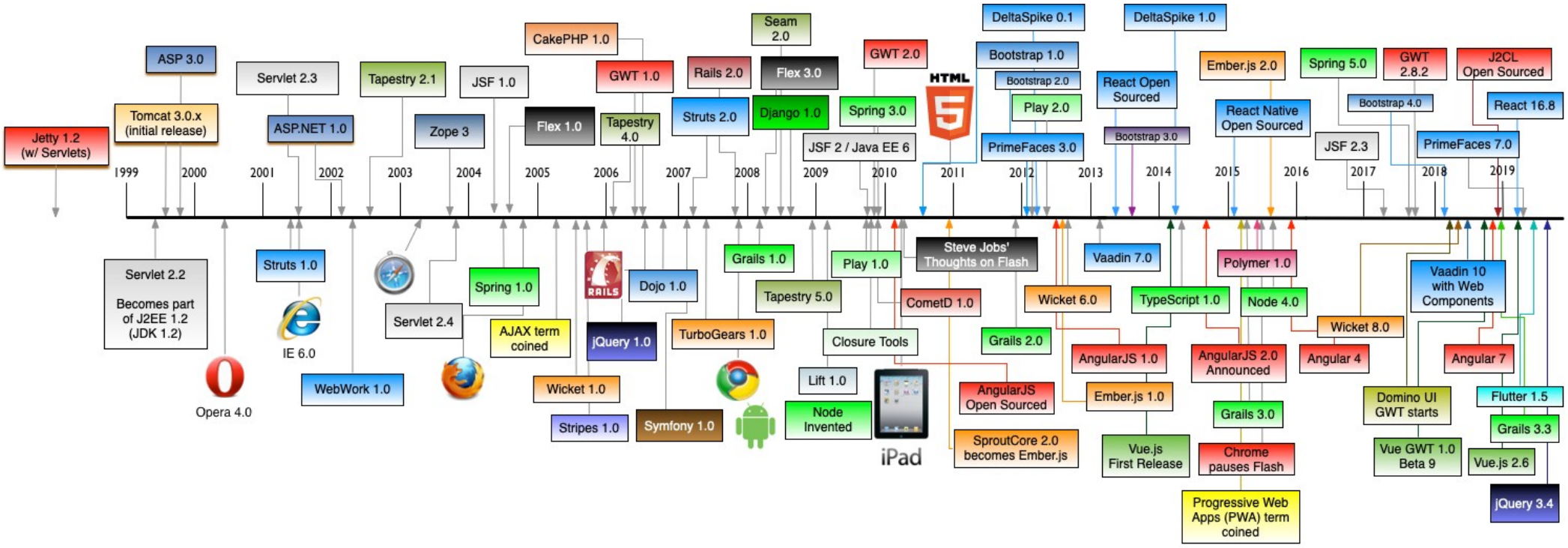
Faculty of Science,
Technology
and Medicine

Web Programming

Volker Müller
University of Luxembourg

Frameworks in Web Development



Source: http://bit.ly/HistoryWebFrameworks

JS Frameworks

Many many JS frameworks exist for different tasks

Overview: https://2019.stateofjs.com

Ember.js has dropped a lot in this ranking

But some concepts, common to many JS frameworks, are nicely explained in Ember.js

→ I will give you short introduction to Ember.js now, later highlight differences to other JS frameworks

JS Framework Ember.js

Open-source TypeScript-based web framework, based on the Model–View–ViewModel (MVVM) pattern

Possible to build desktop and mobile applications in Ember

Focus on ambitious web applications – many Ember specific modules exist

More productive out of the box (lots of auto-generated code)

Ember follows Convention over Configuration

Model–View–ViewModel (MVVM)

Software architectural pattern - variant of MVC

Model: represents state content

View: structure, layout and appearance

ViewModel: abstraction of view, automates binding between view and model

→ Every update on view (by an input) automatically affects bound variables in model and vice versa

Excursus: Reactive Programming

Programming paradigm concerned with data streams / propagation of change

Changes in the model are directly reflected (possibly transitively) in the view

Example reactive operation: Assume assignment a = b + c. If value of b changes, then automatically also value of a changes

Typically only applied for "observers" of "observables"

Getting started with Ember ...

npm install ember-cli

ember new newproject

Out of the box, application will include:

- Development server
- Template compilation
- JavaScript and CSS minification

→ Default template: app/templates/application.hbs

Creating a Route

./ember generate route students outputs

```
installing route
  create app/routes/students.js
  create app/templates/students.hbs
updating router
  add route students
installing route-test
  create tests/unit/routes/students-test.js
```

.hbs: HandleBars.js Templates used

Adding a Static Model (in Route File)

```
import Route from '@ember/routing/route';


export default Route.extend({
    model() {
    return ['VM', 'FL', 'SR'];
    }
});
```

View "app/templates/students.hbs"

```
<h2>List of Students</h2>

<ul>

  {{#each model as |student|}}

    <li>{{student}}</li>

  {{/each}}

</ul>
```

HandleBars.js Template Language has similar ops like templates in Symfony

Creating Components

```
ember generate component teacher-list

installing component
  create app/components/teacher-list.js
  create app/templates/components/teacher-list.hbs
```

We can edit the template as before, then we use it with a "custom tag" as

```
<teacher-list title = "List of Teachers" teachers =
model />
```

Actions – Custom Code for DOM events

Use <li {{action "showPerson" person on="click"}}> …

Used JS code defined in resp. component app/components/teacher-list.js:

```
import Component from '@ember/component';

export default Component.extend({

actions: {  showPerson(person) {  alert(person);  }

…..  }  });
```

More Aspects of Templates

Standard loop exist

Object properties can be accessed with dot notation:

```
{{#each people as |person|}}
    <li>Hello {{person.name}} </li>
```

Links must use <LinkTo @route="…">... together with symbolic route name

Variables can be bound to model {{name}} or dynamically to comp. usage {{@name}}

Model

Model is the most complicated part since we are on the client side → no direct DB connection possible

Model links through an Adapter with various backends: REST, JSON, Web Services, NoSQL, GraphQL,….

EmberData = set of tools to fetch these data, create models, and keep a local data store

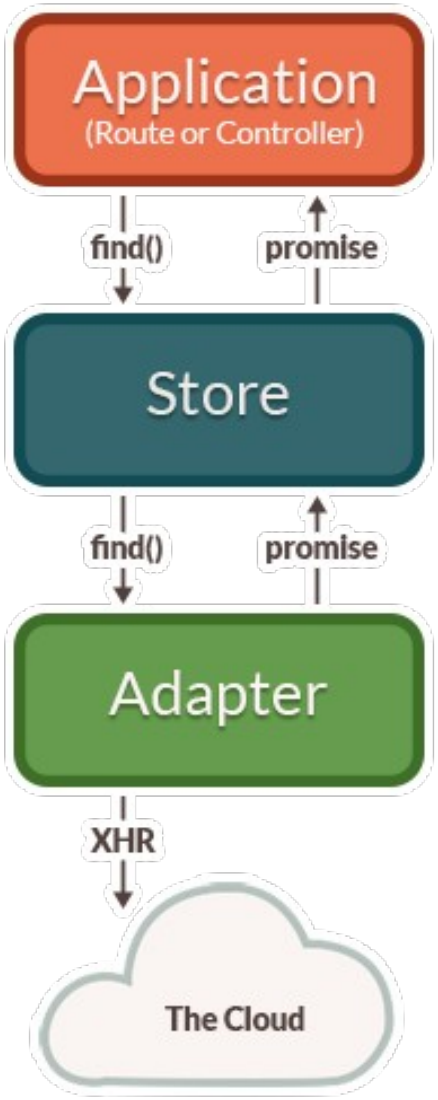
Example Model Definition (app/models/person.js)

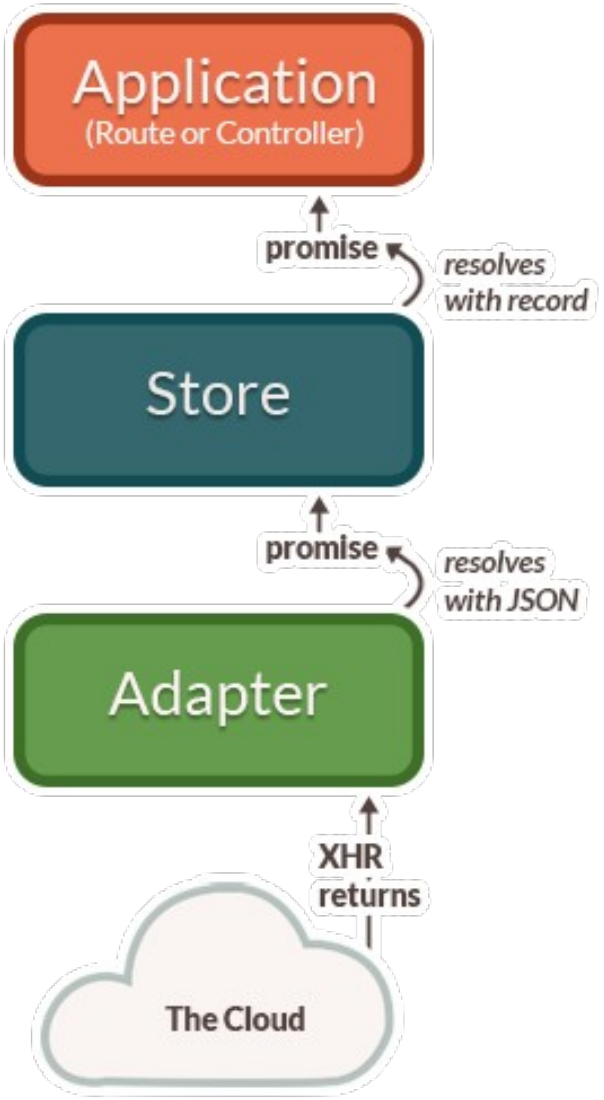
Model "person"

```
import DS from 'ember-data';


export default DS.Model.extend({
  firstName: DS.attr('string'),
  birthday:  DS.attr('date')
});
```

Structure of Data Retrieval





Source: emberjs.com

Data Adapter

Adapter and Model share the same name (but use different pre-defined directories)

Can also be auto-generated

Adapter can be based on predefined adapters: REST, JSON, local storage, …

Meta-data for specific adapter (e.g. URL, paths) are defined inside the adapter class

More details on Ember.js website

Key Points to Remember about Ember

Convention over configuration: file names define controller, model, view, … names – functionalities defined by directory name

Data can not be directly retrieved, but via an adapter from some web service → typical behavior in many JS client side frameworks

Web Components (webcomponents.org)

Web Components allow the creation of reusable HTML custom tags with JS, HTML, CSS

Based on existing standards (which are adapted to also cover web components)

Different types:

- Custom elements ("custom tags")
- Shadow DOM (elements rendered, but not added to DOM)
- HTML template

Web Component Example

```
class AppDrawer extends HTMLElement {...}

window.customElements.define('app-drawer',
AppDrawer, {extends: 'img'  });
```

Usage: `<app-drawer></app-drawer>`

HTMLElement is a quite large JS interface representing any HTML element

Often also sub-interfaces used in "extends"

More details: developer.mozilla.org

Web Component Libraries

Several libraries with web components exist:

- Slim.js (slimjs.com)
- Polymer (polymer-project.org) – free & open source, started by Google

Components typically installed with npm, JS code imported "as usual" into HTML page with <script>

Angular 2+

Current version: 11.0.1 (18.11.2020)

TypeScript-based open source web app framework, running on Node.js, developed at Google (angular.io)

Complete rewrite of AngularJS

AngularJS is first version of Angular – frontend of MEAN stack (MongoDB, Express.js, Angular.js, Node.js)

Improvements in Angular 2+

All improvements of TypeScript vs JS

More modularity (functionality moved into modules)

Asynchronous template compilation

Improved template type checking, payload size

Faster rebuild time

Features of Angular 2

Allows progressive web apps (look and feel like a mobile app)

Build native mobile apps or desktop apps

Lot's of code generation and code splitting into components

Uses simple and powerful template syntax

Includes testing during development

Getting Started ...

Use npm to install package @angular/cli

Script ng used to generate project files and parts of components (TS definition, module, route, template)

ng serve runs a local Node.js development server

Angular Components & Templates

Component templates can be defined either inside component.ts file or in separate file

Components define properties, which can be used in template (assume heroes is an array):

```
<ul> <li *ngFor="let h of heroes"> {{h}} </li> </ul>
```

Templates can use a specific expression language with "structural directives": *ngFor, *ngIf, *ngSwitch, ...

Angular Elements & Forms

Components can be packaged as "Angular elements" – also denoted "Web Components" outside Angular

Equivalent of "custom tags" in JSP – new tags that can be used in HTML

Angular supports standard HTML forms and "Reactive forms" – forms that change state over time

Angular HttpClient

No explicit model build-in, but properties of components have state and can store information

Module HttpClient provides easy way to connect to RESTful WS to retrieve / send data

Normally used with asynchronous communication and JSON encoding

More Information

You see with this first initial steps that there are some similarities with Ember.js, but there are also differences

Please see the tutorial if you want to dig deeper into Angular

React (reactjs.org)

Declarative, efficient, flexible JS library to build user interfaces – not a full MVC framework

Maintained by Facebook + community

Basis for development of single-page or mobile applications

Uses JSX = JS extension to directly embed HTML into JS

Provides components in JS, binding, stateful components

Single Page Applications (SPA)

Web application interacting with users by dynamically rewriting current page rather than loading new pages from server

More similar to Desktop applications

Avoid interruption of user experience (caused by page loading)

Often uses intensively dynamic interaction with web server in background (using Ajax, Web sockets)

SPA must be stateful to record the current "situation"

Example: React Component

```
class HelloMessage extends React.Component {
  render() {  return (<div>Hello {this.props.name}</div>);  }
}

ReactDOM.render(
  <HelloMessage name="Taylor" />,
  mountNode );
```

Usage of JSX

Many more ex. on React website

"mountNode" = id of HTML container where data are mounted

Flux

React provides VIEW only, does not include controller or model

Flux = controller / model architecture used by FB for React

Flux flow: actions → dispatcher → data store, changes to store are propagated back to view

Properties in React should not be changed directly, but via callbacks which trigger actions

Next Week

Some Remarks on Vue.js

Web Assembly