Faculty of Science,
Technology
and Medicine

Web Programming

Volker Müller
University of Luxembourg

Node.js

Open-source, cross-platform JavaScript run-time environment for executing JavaScript code server-side

Uses event-driven, non-blocking I/O model that makes it lightweight and efficient

Often used in combination with nginx as proxy (or other web server) and a NoSQL database

Website: https://nodejs.org/

Web Services

Software system designed to support interoperable machine-to-machine interaction over a network

Often working over HTTP (port 80) or HTTPS (443)

Clients and servers exchange messages in XML format (SOAP) or some other text-based format (JSON)

Now: RESTful WS with Node.js

Example: RESTful WS with Node.js

RESTful WS can be easily implemented with Express.js

Simple Docker-based example provided on Moodle which reads data from a file (instead of a DB), also does no response type negotiation, only returns JSON data

Excursus: REST Service with PHP

In principle, plain PHP code can realize RESTful server

Object for JSON encoding and decoding exists

Main obstacle: URL must be analyzed "by hand"

Less convenient, some way of "routing" would be nice

Use PHP based framework for REST (many exist)

SLIM – a PHP Framework for REST

Micro framework that helps to quickly develop REST server apps in PHP (http://slimframework.com)

Dispatcher that receives HTTP request, invokes appropriate callback based on routing and returns HTTP response

Provides easy definition of "routes"

Installable with composer

SLIM Example

```
$app = new \Slim\App;

$app->get('/hello/{name}', function (Request $req,
Response $res, array $args) {
    $name = $args['name'];
    $data = array("msg" => "Hello, $name");
    $res->getBody()->withJson($data);
    return $res;  });

$app->run();
```

Swagger – Improved REST WS Management

REST WS can implement an API, but there is no dedicated API documentation

OpenAPI = specification for machine-readable interface files for describing, producing, consuming, and visualizing RESTful web services

Swagger (https://swagger.io/) provides several tools for defining / building APIs

Swagger Tools

Swagger Editor: Design, describe, and document your API using OpenAPI

Swagger Codegen: Simplifies build process by generating server stubs and client SDKs for any API defined with OpenAPI

Swagger UI: Visualize / interact with API's resources without any of the implementation logic in place

All exist also in free open-source version

GraphQL

Common problems for REST APIs:

- Many different accessible objects → many different URLs
- WS returns full object, accessing (statically defined) subset of properties only possible with additional URLs

GraphQL = Query language for an API

Detailed information on graphql.org

GraphQL Request

Users sends POST request to server defining requested fields ("GraphQL query"):

```
{
  user (id: 1) {
      name
      height (unit: METER)
  }
}
```

If query “small”, also GET + encoding as URL parameter "query" possible

GraphQL Response

Server validates whether request can be fulfilled, then returns data (normally) in JSON encoding:

```
{
  "data": {
    "user": {
      "name": "Volker Müller",
      "height": 1.70
      }  }
}
```

Building a GraphQL Service

Libraries for many different server-side languages available (see GraphQL website)

Server code has to define type system (defining objects, fields, possible arguments) and queries

Using this type system, server can predetermine whether request valid (or not), then query executed

Each field related with resolver function

GraphQL JS Service Example

Basic example for GraphQL service with JS running on Node.JS available on Moodle

Detailed tutorials:

graphql.org/learn/

www.tutorialspoint.com/graphql

Apollo Server

Apollo server is a open-source GraphQL server based on Node.js and express-graphql

Bundles libraries commonly needed to build GraphQL service

Documentation available at www.apollographql.com

Another Useful Tool: Webpack

Webpack compiles several JS files into a single "bundle" of optimized JS code and ensures that only used functions / modules are included

Usable also for other assets: CSS, images, fonts

Runs within Node.js

Tutorial: webpack.js.org/guides

Example provided on Moodle

Last example: Node.js Cluster + Redis

Node.js also often used with NoSQL DB

On Moodle: Docker application with

- Redis as NoSQL DB
- 3 instances of Node.js servers (cluster)
- NGINX as proxy → forwards requests to Node.js instances with Round Robin

Conclusion on Node.js

Node.js is a hot topic with many demands on the market

Very efficient in certain scenarios, but not always the best choice

Often a combination of a traditional web server (nginx) acting as proxy and Node.js used, where "specific requests" are proxied to Node.js

Very often used in combination with JS frameworks

Excursus: PHP Swoole (www.swoole.co.uk)

PHP Swoole is asynchronous programming framework for PHP

Allows developers to write asynchronous code for PHP

Event-driven, asynchronous, non-blocking IO → easy scalable

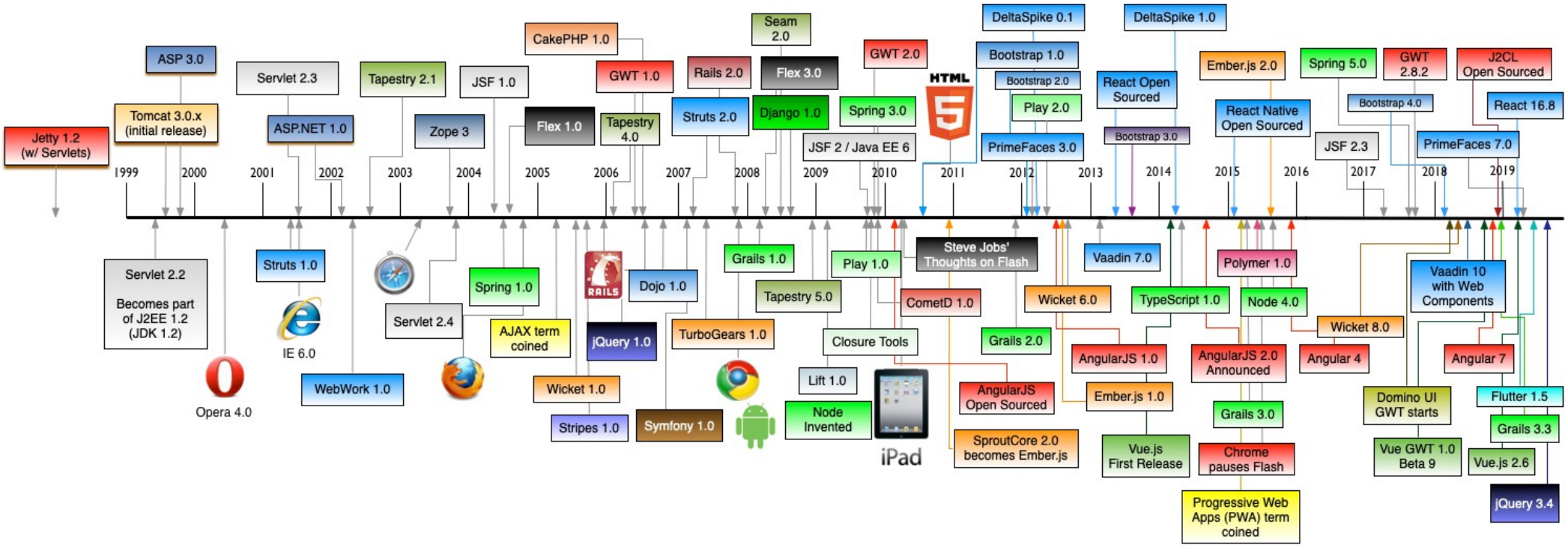
Implemented as PHP extension, but requires support from web server software

→ Brings some ideas of Node.js to PHP

PHP Swoole Example

```
<?php
$http = new swoole_http_server("127.0.0.1", 9501);
$http->on("start", function ($server)
{   echo "Http server started \n";   });
$http->on("request", function ($request, $response) {
    $response->header("Content-Type", "text/plain");
    $response->end("Hello World\n");
});
$http->start(); ?>
```

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 JavaScript web framework, based on the Model–View–ViewModel (MVVM) pattern

Also 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 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 {{link-to "…"}} together with route name

HTML attributes can be bound to model:

```
<img src = "{{logo}}" />
```

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

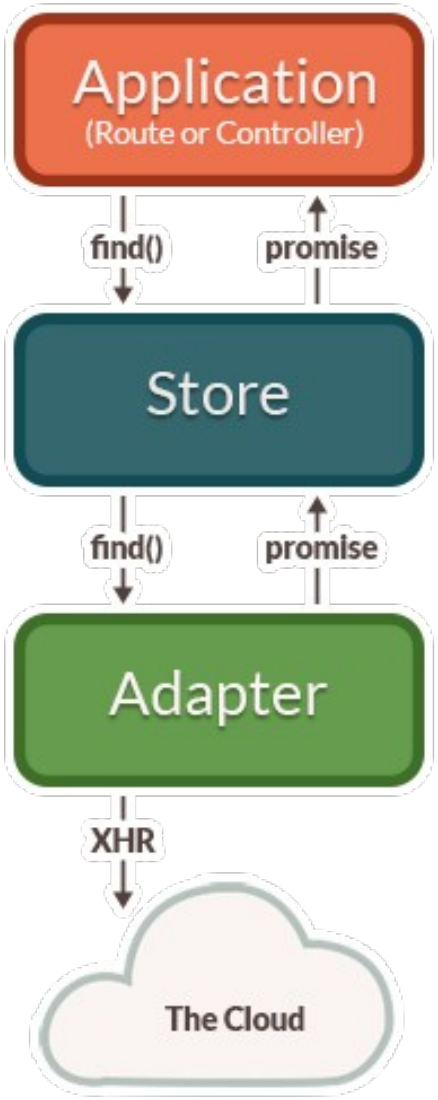
Example Model Definition (app/models/person.js)

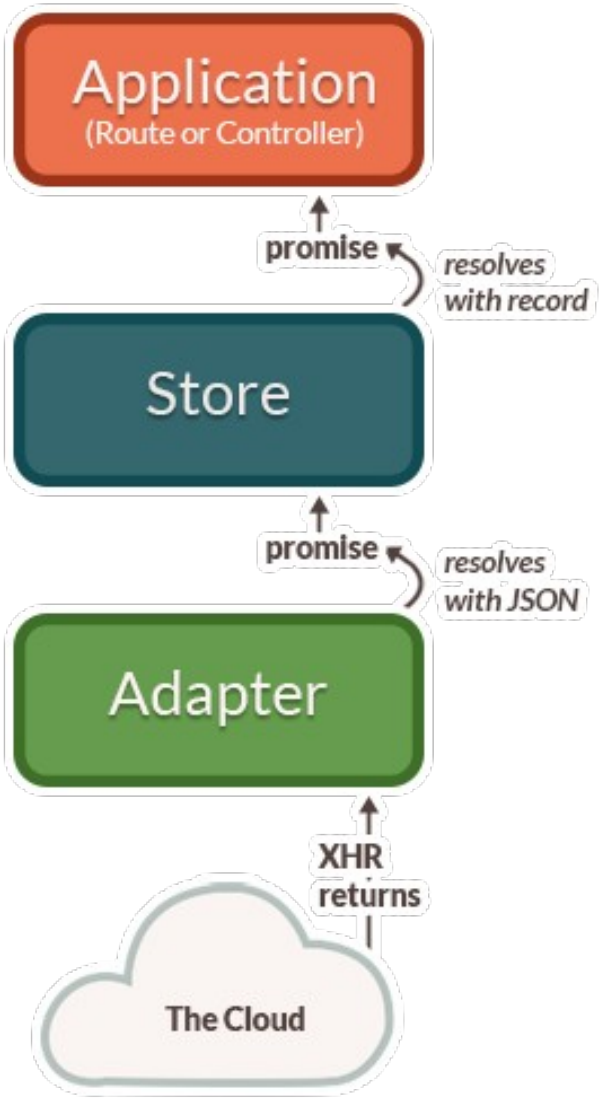
Model "person"

```
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

Next Week

Short Introduction to Angular.js, Vue.js and React.js

New Trends in Web Application Development