



Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

Web and Apps

1) Architectures

Emmanuel Benoist
Spring Term 2021

Architectures

- Introduction
- Classical Web Architecture
 - Send information to the server
 - Frameworks
- In Browser Rendering
 - Frameworks
 - Connect to the server
- Mobile Applications
- Secure communication
- Conclusion

Introduction

Architectures

- ▶ **Question: What are the possible architectures for distributed architectures?**
- ▶ **Problems:**
 - ▶ Modern applications run on multiple platforms
 - ▶ Desktops, laptops, tablets, smartphones
 - ▶ Access data from different sources
 - ▶ Database(s), other applications, API's, other servers, ...

Classical Web Architecture

Classical Web Architecture

▶ **Web server**

- ▶ nginx Apache, Internet Information Server (IIS), Google
- ▶ Survey at <https://news.netcraft.com/archives/category/web-server-survey/>

▶ **Desktop Web Browsers**

- ▶ Chrome, Firefox, Safari, Edge, Internet Explorer (MSIE), Opera
- ▶ Survey at https://en.wikipedia.org/wiki/Usage_share_of_web_browsers

▶ **Mobile Web Browsers**

- ▶ Chrome, Safari, Samsung, UC (mainly in Asia), Opera, Firefox
- ▶ Same source

Classical Web Architecture (Cont.)

- ▶ **Pages are produced on Server**
 - ▶ Generates HyperText Markup Language (HTML) document
 - ▶ And links to Cascading Style Sheets (CSS)
 - ▶ and images (GIFs or JPEGs) or animations (mainly JavaScript)
- ▶ **Are transferred to the Browser**
 - ▶ HyperText Transfer Protocol (HTTP)
 - ▶ Files are transferred
- ▶ **Browser prints the content of the files**
 - ▶ HTML and CSS are combined
 - ▶ Browser builds a tree (the Document Object Model DOM) and renders it.
- ▶ **Browser only downloads documents**
 - ▶ All the work is done on the server

HTML Page

- ▶ **Generated on the server**

- ▶ Program generates it
- ▶ Is sent in HTTP to the client

- ▶ **Contains tags**

- ▶ `<h1></h1>` (headings 1), `<h2></h2>` (headings 2), `<p></p>` (paragraph)
- ▶ `` (unordered list), contains `` (list items)

HTML Page

```
<html>
```

```
<body>
```

```
<h1>Hello X2A and X3R</h1>
```

```
<p>
```

Guess Euch mitenand – Bonjour a tous

```
<ul>
```

```
<li></li>
```

```
<li></li>
```

```
<li></li>
```

```
</ul>
```

```
</body>
```

```
</html>
```

Example of HTML Page

- ▶ **Le Temps** (Swiss french journal) <https://www.letemps.ch/>

Send information to the server

Plain HTML interaction with the server

- ▶ **HTML Forms**

- ▶ Contains text-inputs, radio-buttons, select-lists, ...

- ▶ **Request**

- ▶ Content of the fields is sent to the server

- ▶ **Work is done on the server**

- ▶ Server reads data
- ▶ Server stores data in a Database
- ▶ Server reads files
- ▶ ...

- ▶ **Program on the server**

- ▶ Generates an HTML document.
- ▶ Including tags and content.

- ▶ **Server sends a page back to the browser**

- ▶ Browser displays the page.

Example of a page containing a Form

- ▶ **Two different forms:**
 - ▶ GET: Information is in the URL
 - ▶ POST: Information is in the body of the HTTP Request (not in the URL).
- ▶ `https://www.benoist.ch/WebApps/examples/architectures/forms.php`

Frameworks

Frameworks for Working ServerSide

▶ Frameworks?

- ▶ Contain a persistency layer (connection to database -> objects)
- ▶ Model View Controller (MVC) design pattern
- ▶ Templating engine
- ▶ Session, HTTP and connection support

▶ Java

- ▶ Java Server Faces, Struts, Spring

▶ PHP

- ▶ Laravel, CodeIgniter, Symphony, Zend

▶ JavaScript (Server Side)

- ▶ Node.js

In Browser Rendering

In Browser Rendering

- ▶ **HTML page contains a program**
 - ▶ Written in JavaScript
- ▶ **Manipulates the DOM**
 - ▶ Document Object Model (HTML Tree)
 - ▶ Contains the structure of the document
 - ▶ Properties = layout (from CSS)
- ▶ **Server generates a page**
 - ▶ Contains some placeholders
 - ▶ Contains the program
- ▶ **Browser evaluates the JavaScript program**
 - ▶ Fill elements into the placeholders
 - ▶ Generates the DOM
 - ▶ Manipulate the DOM
 - ▶ Interacts with the user
- ▶ => **One page applications**

JavaScript Client-side

- ▶ **Pur JavaScript**
 - ▶ Very rare
- ▶ **JQuery**
 - ▶ For easy to write JavaScript
- ▶ **TypeScript**
 - ▶ For writing a code that is better typed than JS.

Frameworks

JavaScript Client-side Frameworks

For developing **Single Page Application (SPA)**.

- ▶ **Frameworks offer often used functionalities**
 - ▶ Layout functionalities
 - ▶ Markup language to be integrated into HTML
 - ▶ Model View Controller Design pattern
 - ▶ Connection to a server
 - ▶ ...
- ▶ **Angular**
 - ▶ <https://angular.io/>
- ▶ **React**
 - ▶ <https://reactjs.org/>
- ▶ **Vue**
 - ▶ <https://vuejs.org/>
- ▶ **Others**
 - ▶ Ember, Meteor, Polymer, ...

Client Side Rendering

- ▶ **The server generates a HTML page**
 - ▶ Contains a minimal HTML document
 - ▶ The document refers to different JavaScript files
 - ▶ Framework
 - ▶ Library
 - ▶ Program
 - ▶ ...
 - ▶ Document does not contain the things to be drawn
- ▶ **Browser Evaluates the HTML**
 - ▶ Builds the minimal Document Object Model (HTML Tree)
 - ▶ Downloads all the JS files
 - ▶ Evaluates JS and executes the program.

Program runs inside the browser

- ▶ **Instructions are written inside HTML**
 - ▶ directives look like HTML tags
- ▶ **Are evaluated by JavaScript**
 - ▶ All the files are downloaded to the browser
 - ▶ The browser runs the program
- ▶ **Problem**
 - ▶ A hacker can easily take control of this code
 - ▶ Can easily change all the rules.
 - ▶ Often programmers do not realize it!!!!
- ▶ **Other Problem**
 - ▶ How to connect a Database?

Connect to the server

Connect to the server

- ▶ **Impossible to connect to the Database**
 - ▶ The Database server port is closed for the Internet
 - ▶ Access is restricted to only one server
- ▶ **Idea use a “proxy”**
 - ▶ Use a program on the server
 - ▶ The program connects to the Database
 - ▶ JavaScript connect to this program

Connect to the server (Example)

- ▶ **We have the following Angular file**
 - ▶ This file is placed inside the web server, but it is transferred to the client and run inside the browser.

```
<div ng-app="myApp" ng-controller="patientCtrl">
<table>
  <tr ng-repeat="x in names">
    <td>{{ x.Name }}</td>
    <td>{{ x.FirstName }}</td>
  </tr>
</table>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('patientCtrl', function($scope, $http) {
  $http.get("patients_mysql.php")
  .then(function (response) {$scope.names = response.data.records;});
});
</script>
```

Connect to the server (Example II)

- ▶ This file is evaluated inside the server

```
header("Access-Control-Allow-Origin:_*");
header("Content-Type: application/json; charset=UTF-8");
$conn = new mysqli($hostname, $username, $password, $dbname
→);
$result = $conn->query("SELECT name, first_name, MRN
→FROM patient");
$outp = "";
while($rs = $result->fetch_array(MYSQLI_ASSOC)) {
    if ($outp != "") {$outp .= ",";}
    $outp .= '{"Name":"' . $rs["name"] . '";';
    $outp .= '"FirstName":"' . $rs["first_name"] . '";'
→;
    $outp .= '"MRN":"' . $rs["MRN"] . '"}';
}
$outp = '{"records":[' . $outp . ']}';
$conn->close();
```

Connect to the server (Example III)

- ▶ It generates the following JSON file

```
{ "records":  
  [  
    { "Name": "Laurie", "FirstName": "Hugh", "MRN": "24598" },  
    { "Name": "Spencer", "FirstName": "Jesse", "MRN": "24610" },  
    { "Name": "Jacobson", "FirstName": "Peter", "MRN": "25009" },  
    { "Name": "Edelstein", "FirstName": "Lisa", "MRN": "25566" }  
  ]  
}
```

Mobile Applications

Mobile Applications

- ▶ **Are executed in mobile environments**
 - ▶ Smartphones
 - ▶ Tablets
 - ▶ Smart-TV's
 - ▶ ...
- ▶ **Can be started in a browser**
 - ▶ Is a Web Application (see previous sections)
- ▶ **Can be written in native language**
 - ▶ One development for Android
 - ▶ One development for iOS (Apple devices)
- ▶ **Can be written in JavaScript**
 - ▶ Similar to a Web Application
 - ▶ But JavaScript program is stored inside the device.
 - ▶ Example Ionic Framework

Apps connections to DB

- ▶ **App can not open a connection to a central DB**
 - ▶ Huge Security problems
 - ▶ Database port can not be accessible from the Internet
- ▶ **Restrictions are similar to Web Apps**
 - ▶ No direct access to files
 - ▶ No direct access to Database
- ▶ **Solution**
 - ▶ Use the same gateway (API) as Web Apps (Client side rendering)
- ▶ **Advantage**
 - ▶ Develop only once for both targets
- ▶ **Requirement**
 - ▶ Have a protocol that can be used by both Apps and Web Frameworks
 - ▶ De Facto Standard : JavaScript Object Notation JSON.

Secure communication

Secure Communication

- ▶ **HTTPS = Secure version of HTTP**
 - ▶ Corresponds to the Lock in the address bar
- ▶ **HTTPS uses a secure layer : TLS**
 - ▶ TLS bases on the Public Key Cryptography
 - ▶ A server publishes a Certificate.
- ▶ **Certificate**
 - ▶ Certificate contains: a public key and the name of the site (host name, or domain name).
 - ▶ It is signed by a Certificate Authority (Verysign, Let's Encrypt, ...)
- ▶ **Secure communication**
 - ▶ Browser generates a symmetric key.
 - ▶ Sends the symmetric key to the server encrypted using the public key (Hand shaking)
 - ▶ Communication is then encrypted using the symmetric key.

Conclusion

Conclusion: What have we seen?

- ▶ **Web pages are generated by a web server**
 - ▶ A program generates all the data (Server Side Rendering)
 - ▶ Or the server just sends JavaScript Files
- ▶ **Modern Applications are :**
 - ▶ Client-side rendering with a JavaScript framework
 - ▶ Apps (Native or JavaScript) inside mobile devices
- ▶ **Modern Applications need a gateway into the server**
 - ▶ API for accessing Data
 - ▶ Connection to resources on the server
 - ▶ Files, databases, other servers, ...
- ▶ **Rest of the course**
 - ▶ We will focus on the building of such gateways
- ▶ **Part Reto Koenig**
 - ▶ Develop mobile Apps consuming the data.

References

- ▶ **Netcraft Web server survey**
 - ▶ `https://news.netcraft.com/archives/category/web-server-survey/`
- ▶ **Angular Tutorial by W3Schools**
 - ▶ `https://www.w3schools.com/angular/default.asp`
- ▶ **Survey on JavaScript frameworks**
 - ▶ `https://hackr.io/blog/best-javascript-frameworks`