Tecnologie e Applicazioni Web

Introduction

Filippo Bergamasco (filippo.bergamasco@unive.it)
http://www.dais.unive.it/~bergamasco/
DAIS - Università Ca'Foscari di Venezia
Academic year: 2023/2024

About the teacher

Filippo Bergamasco

http://www.dais.unive.it/~bergamasco

- Associate Professor
- Research group:
  Artificial Intelligence and Image Understanding
  https://www.unive.it/pag/45973/
- Interests:
  - Computer Vision / Machine Learning / Data Science

About the course

https://moodle.unive.it/course/view.php?id=17011

48 Hours - 24 Frontal lessons

- The Moodle page is your primary reference to obtain information about the course, materials, etc.

- Timetable:
  - Monday 12:15 – 13:45
  - Thursday 14:00 – 15:30

Exam

- Exam is based on a **project work**
- The project consists in:
  - a **web application** using technologies and frameworks studied in this course
  - a **report** that will be discussed with an oral examination
- Project work must be submitted at least 1 week before the oral

Exam

- Project goals will be presented at the end of the course.
- The only accepted submission method is via Moodle.
- Please read and **understand** the exam info page: https://moodle.unive.it/mod/page/view.php?id=776703

Course prerequisites

No formal requisites are needed to follow this course

Suggested requirements/skills:

- JavaScript programming language
- Basic understanding of HTML/CSS
- Computer networks

…but also:

- Being curious
- Desire to understand how the things work “under the hood”

Why should I need this course?

This course provides an overview of the technologies involved in the **modern** World Wide Web

Theoretical side:

- Protocols/Patterns/Security/Networking

Practical side:

- We'll learn how to use some modern **technologies** to develop web applications.

Why should I need this course?

In particular, this course is focused on the technologies related to the **JavaScript** ecosystem:

- A unified language for client and server-side
- Friendly for event-driven programming
- Extremely fast and optimized JIT interpreters (ex. Google V8)
- Base language on which many dialects were born (TypeScript, Coffee-Script, etc.)

Why should I need this course?

"I came to appreciate that there is a real art to gluing together applications made from different technologies. It is a skill in itself; just knowing the technologies and what they can do is only part of the challenge."

Simon Holmes - Getting MEAN with Mongo, Express, Angular, and Node

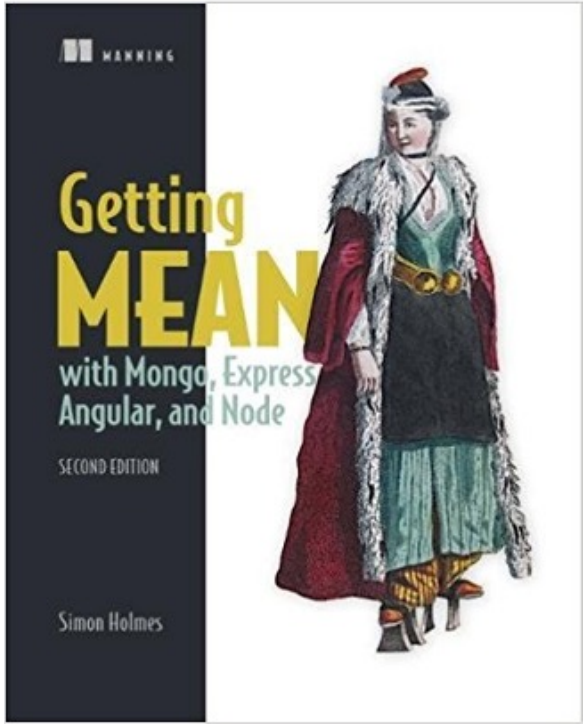
Referral texts

Simon Holmes,

"Getting MEAN with Mongo, Express, Angular, and Node",

Second Edition, Manning Publications, 2019.

ISBN: 9781617294754



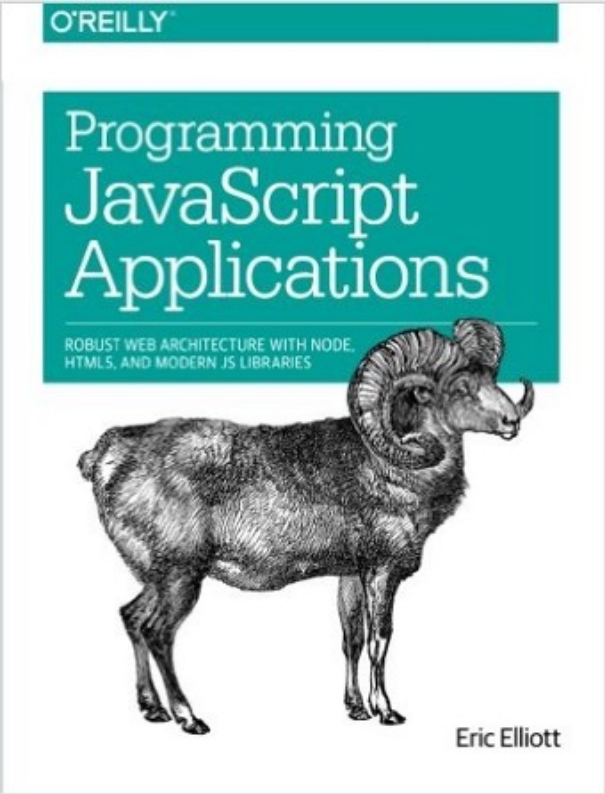
Referral texts

Eric Elliott,

"Programming JavaScript Applications: Robust Web Architecture with Node, HTML5, and Modern JS Libraries",

O'Reilly Media, 2014.

ISBN-10: 1491950293



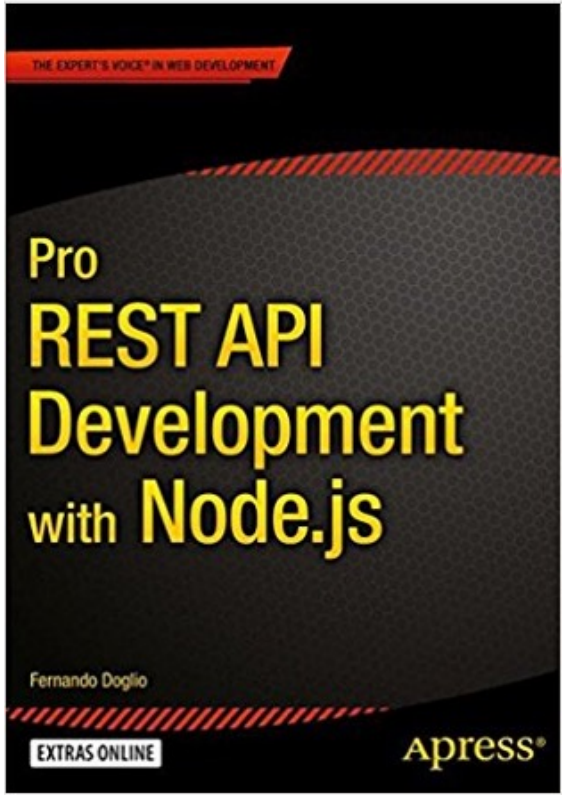
Referral texts

Fernando Doglio,

"Pro REST API Development with Node.js",

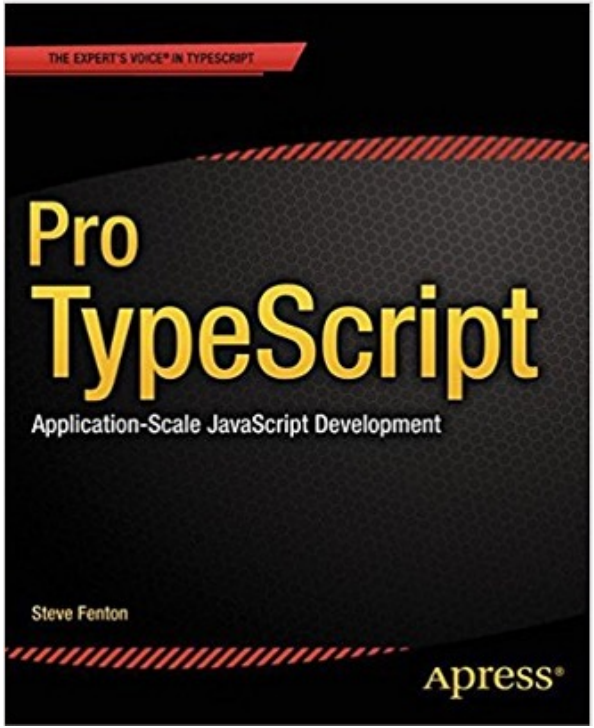
Apress, 2015.

ISBN-10: 1484209184



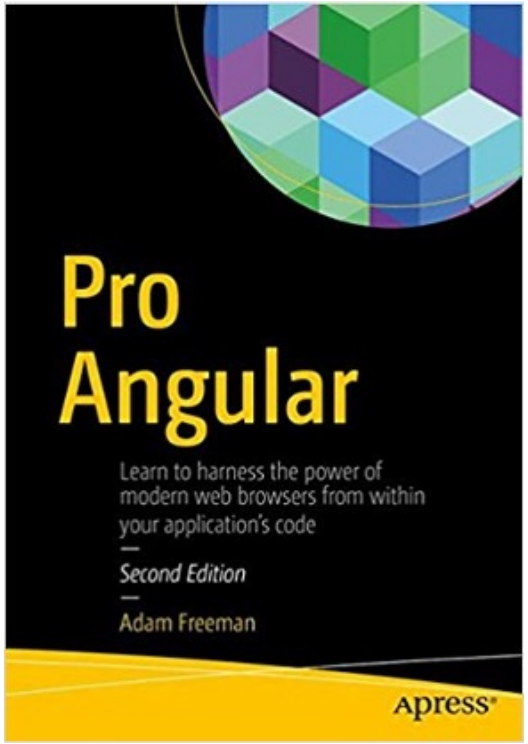
Referral texts

Steve Fenton,
"Pro Typescript",
Apress, 2014.
ISBN 978-1-4302-6790-4



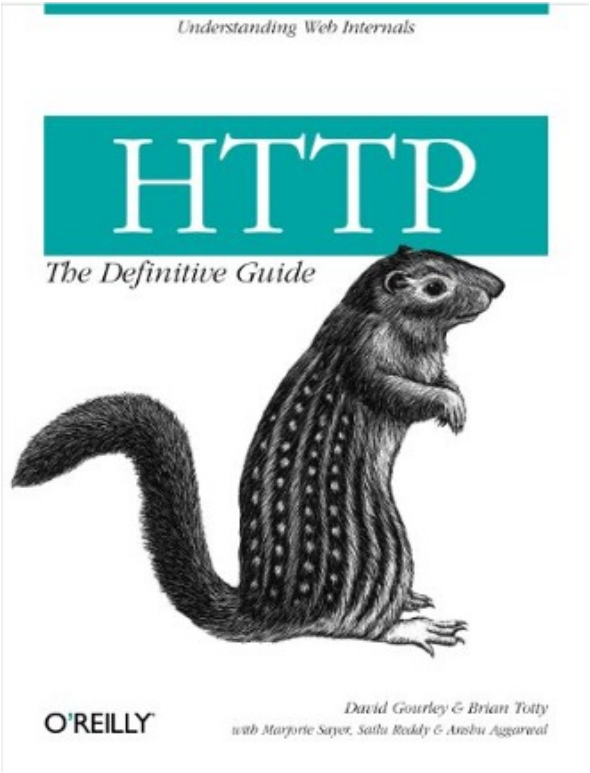
Referral texts

Adam Freeman,
"Pro Angular",
Apress 2017.
ISBN 978-1-4842-2307-9



Suggested readings

David Gourley et al.,
"HTTP: The Definitive Guide",
O'Reilly Media.
ISBN 1565925092



Syllabus: Part 1

- HTTP and HTTPS protocols
- Session
- Cookies
- Authentication & JWTs
- WebSocket
- REST style APIs
- Containerized applications with Docker

Syllabus: Part 2

- JavaScript
- Typescript
- Node.js server-side runtime
- Asynchronous IO
- Event-driven programming

Syllabus: Part 3

NO-SQL storage

- MongoDB
- REDIS in-memory DB/cache/message broker

Syllabus: Part 4

Stack middleware

- Express js

Syllabus: Part 5

Web frontend

- Angular

Syllabus: Part 6 (optional)

Mobile hybrid and progressive apps

- Apache Cordova

Cross-platform web-based native applications

- Electron framework

Before we begin...

Let's clarify some key concepts

What is the Internet?

What is the Internet?

The Internet is the **global** system of interconnected computer **networks** that use the **Internet Protocol Suite** (TCP/IP) to link **devices worldwide**.

What is the Internet?

The Internet is the **global** system of interconnected computer **networks** that use the **Internet Protocol Suite** (TCP/IP) to link **devices worldwide**.

It is global in the sense that it consists of private, public, academic, business, and government networks. All interconnected together

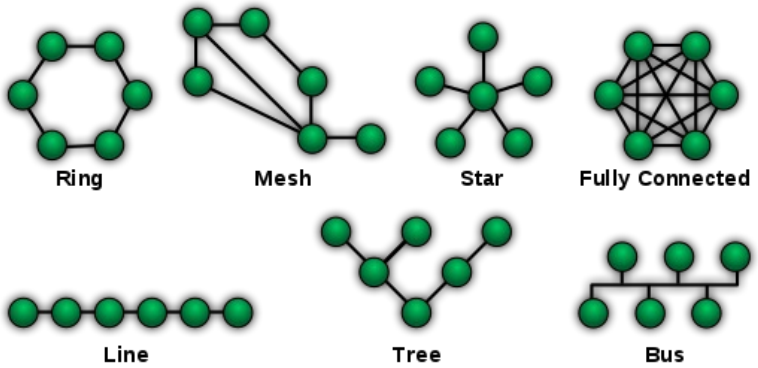
What is the Internet?

The Internet is the **global** system of interconnected computer **networks** that use the **Internet Protocol Suite** (TCP/IP) to link **devices worldwide**.

A computer network allows nodes (computers) to share resources (**exchange data**).

More on computer networks

- Each node is identified by an **address** and is generally referred to as a **host**
- Two hosts may exchange data even if they have no direct connection (different network topologies!)



What is the internet?

The Internet is the **global** system of interconnected computer **networks** that use the **Internet Protocol Suite** (TCP/IP) to link **devices worldwide**.

A communication protocol is a set of rules for exchanging information over a network.

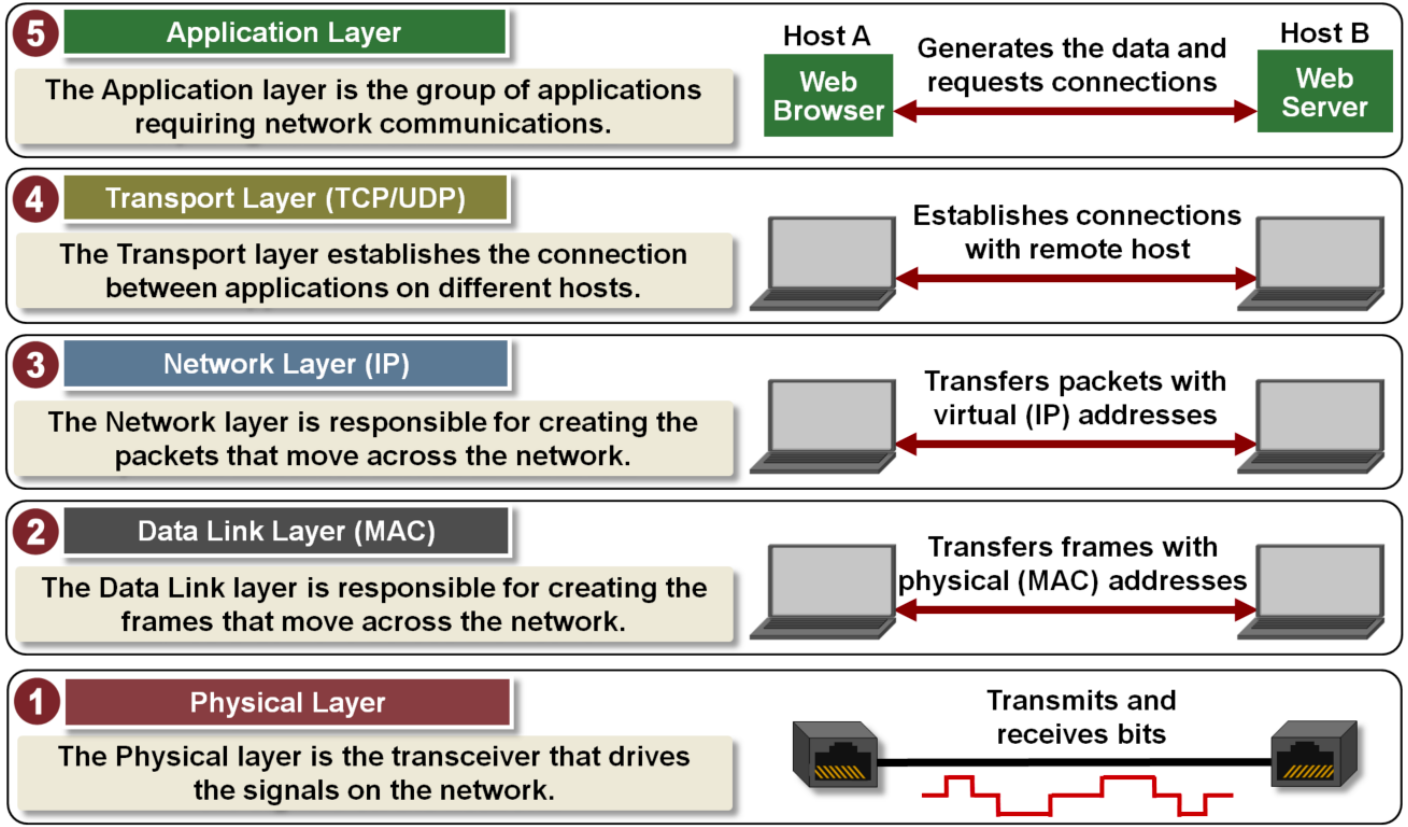
Protocol stack

- Protocols are usually layered in a stack
- Each protocol leverages the services of the protocol layer below it
- The layer at the lowest level controls the hardware, physically sending the information across the media

The internet protocol suite

- Is a conceptual model for all the protocols used on the Internet
- Commonly known as TCP/IP for the two fundamental protocols, TCP and IP
- Specifies how data should be **packetized**, **addressed**, **transmitted**, **routed**, and **received**.

TCP/IP 5-layers reference model



What is the World Wide Web?

What is the World Wide Web?

The WWW is an **information system** in which the items of interest (referred to as **resources**) are identified by Uniform Resource Locators (**URL**).

Resources can be linked by **hypertext** and are accessible over the **Internet**. Resources may be accessed by a software application called **web browser**.

Architectural bases of the WWW

1. **URL**s are used to identify resources
2. Web agents communicate using **standardized protocols** that enable interaction through the exchange of messages that adhere to a defined syntax and semantics
3. Resources have a specific **representation** that can be interpreted (and visualized) by web browsers

Browsing the web

Suppose you want to visit a certain website…

Everything starts by opening your favorite browser and entering a URL

For example:

http://www.example.org/home.html

Browsing the web

http://www.example.org/home.html

The URL is divided in different parts that together specify the **protocol** to use, the **address** of the host machine owning the resource, and the **resource name**.

Browsing the web

http://**www.example.org**/home.html

1. The host's **name** is **resolved** to an Internet Protocol (IP) address using the globally distributed Domain Name System (DNS). After querying the DNS, an IP address like 203.0.113.4 is obtained

Browsing the web

http://www.example.org/home.html

2. The browser will establish a TCP connection with the other host (called **server**) and starts communicating as specified in the HTTP protocol

Browsing the web

http://www.example.org/**home.html**

3. Using the HTTP protocol, the browser asks for a resource named **`home.html`**
4. If the resource exists on the remote server, it is downloaded via the established TCP connection.
   Note: the resource is not necessarily a file. It can be generated on the fly by the server

Browsing the web

http://www.example.org/home.html

5. The resource has an associated representation (called **mime-type**). In this example, the resource is a Hyper Text Markup Language (HTML) document, a standard used to create web pages.
6. The browser interprets the resource and visualizes it graphically

Browsing the web

- The HTML document can contain **hyperlinks** to other web pages or resources.
- Such a collection of useful, related resources, interconnected via hypertext links was called a ***web* of information** by its original inventor: **Tim Berners Lee**

Evolution of the web

1989: Tim Berners Lee

- Researcher at CERN
- Thinks about the idea of having a software platform to ease the collaboration among researchers worldwide

Evolution of the web

1990: HTTP is defined (Tim Berners Lee et al.)

- Hyper-text transfer protocol
- **Request-response** protocol initially designed to exchange hypertext (now used to exchange any kind of resource)
- Fundamental concept of **resource** uniquely identified by a **URL**

Evolution of the web

1990: HTML is invented (Tim Berners Lee et al.)

- HTML is a language interpreted by web browsers to visualize documents containing text, images, sounds, videos, and links to other similar documents
- **Declarative** language: only allows to define the document structure (pages are not interactive)
- Based on another markup language, SGML

Evolution of the web

1993: MOSAIC is released

- Revolutionize the concept of web browser allowing the visualization of text and images (multimedia) on the same page

Evolution of the web

1994-1996: Many browsers were born

Evolution of the web

1995: JavaScript

- Included in Netscape Navigator 2
- Syntax similar to “Java”
- Idea (Marc Andreessen): HTML needed a simple (imperative) language to allow a web page to dynamically modify its own content (static pages became dynamic!)
- JavaScript code was embedded in the page itself

Evolution of the web

1996: CSS

- A language used to define **how to visualize** elements defined by a markup language (like HTML).
- Idea: Separate **content** from **presentation**
  - More flexibility
  - Resources are now accessible
  - Reduce the complexities and repetitions

Evolution of the web

https://oldweb.today/

Browse the Web like in the '90!

For example:

https://oldweb.today/?browser=nm3-mac#19960101/http://www.google.com

Evolution of the web

1996-2000: Companies grow their interest in better presenting their content.

- CSS and JavaScript became fundamental technologies for developing web pages
- **Contents are now dynamically generated**
- Born the role of front-end developer

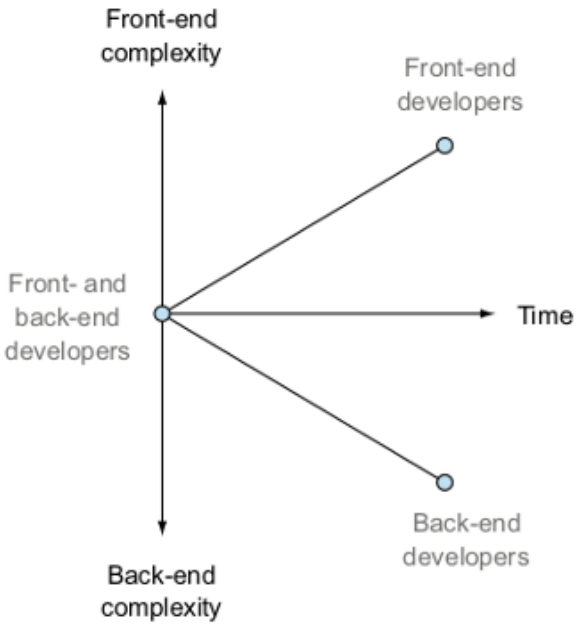
Evolution of the WEB

Back-end developers:

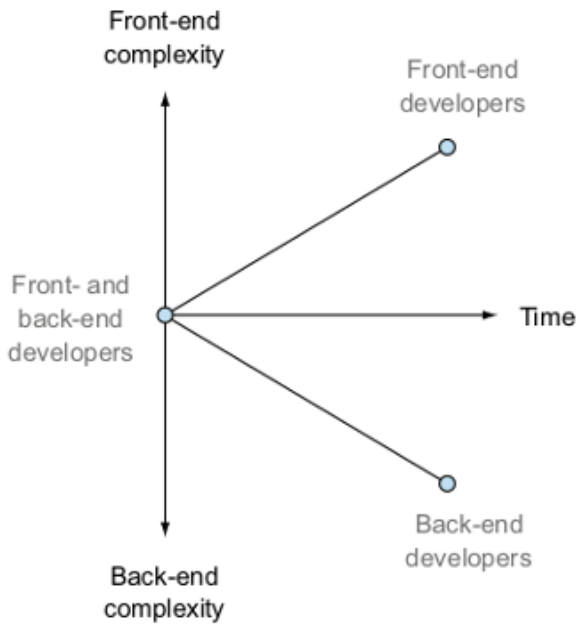
Website “back-stage” (data, content, security, structure, business logic)

Front-end developers:

User experience and content presentation



Evolution of the WEB



- In the past, a web developer had to choose in advance where to specialize
- Front- and back-end usually comprised very different technologies

Evolution of the WEB

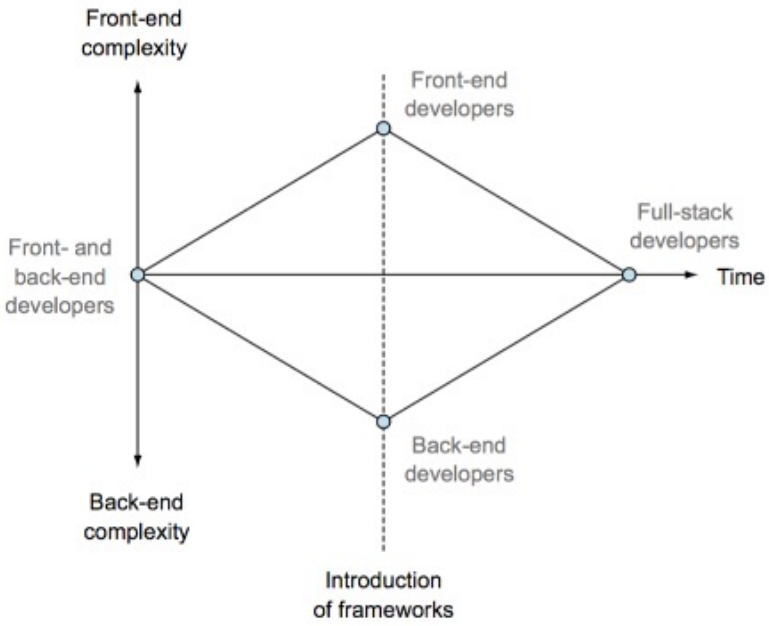
Starting from the early 2000s, several backend and front-end frameworks emerged:

- Backend: PHP, Ruby on Rails, Java/JSP
- Frontend: jQuery, Dojo

A good framework can abstract many of the complexities characterizing modern web applications

- Faster development and less expertise required

Evolution of the WEB



- This trend towards abstraction and simplification gradually led to the concept of «full-stack developer»
- Simultaneous development (often with the same technologies) of front- and back-end

Frameworks full-stack

Developers are not forced anymore to specialize between front-end and back-end!

- One can follow the development of a web app in all its parts
- Increased productivity, freelance work, etc.

Frameworks full-stack

Advantages:

- You have control of all parts of a web application. If the same people develop both aspects (back- and front-end), they can generally better interoperate
- The same technologies are used for multiple platforms: web/mobile/desktop
- Quickly move from idea to implementation

Frameworks full-stack

Which framework to choose?

.... depends, as always, on the context

Nowadays we can use the same language to develop every aspect of a web application: JavaScript

In this course we'll learn to develop with a set of technologies known as «MEAN»

Evolution of the web

2005 - Jesse James Garrett published a paper "Ajax: A New Approach to Web Applications"

- The term Ajax is coined to describe a set of technologies to develop web applications that **communicate asynchronously** with the server without interfering with the display and behavior of the existing **page**
- AJAX = **A**synchronous **J**avaScript **A**nd **X**ML
  (modern implementations use JSON instead of XML)

Evolution of the web

Trend: moving the application code forward in the stack

- AJAX allows to push the application logic from the server to the client (web browser)

Advantage:

- Reduce the server load and costs

Server or client side?

Server-side

A server manages every aspect of a web application:

- Application logic
- User input validation
- Webpage structure generation (HTML, etc.)

Client-side (Single-page app)

The client manages both the presentation and application logic. Interactions with the server are limited to a pure application-data exchange

Single page application

Simple server:

Pure data API or web service

Complex client

Requires sophisticated frameworks to manage the application logic and user interface

Evolution of the web

2008 - Google publishes Chrome

- Highly-compliance to web standards (ACID tests)
- Minimalist approach
- Secure, integrated with Google services

Implements the (open-source) JavaScript engine V8

- JIT compiler, optimized at runtime, inline caching

Web 1.0 vs 2.0

2004 - Tim O'Reilly popularizes the concept of web 2.0

- Is not a standard but defines the way in which web pages are developed and used

Web 1.0

- Static pages
- Content is generated by few "publishers" and viewed by many users

Web 2.0

- Dynamic pages
- Content dynamically generated by the users in virtual communities
- Social media

Web 3.0?

Semantic web

- Content is organized in ontologies to be processed and understood by both humans and machines
- Still not a standard and under active research