Web and Cloud Computing
Week 1 - Overview

2615 - Master in Business Analytics - 23/24 T3

João Costa Seco (joao.seco@fct.unl.pt)

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

NOVA SCHOOL OF BUSINESS & ECONOMICS

Getting to know each other

Who am I?

- **João Costa Seco**
  - Associate Professor (NOVA SST)
  - Researcher at NOVA LINCS (Software Systems)
  - Programming Language Design, Software Verification, Software Security and Automated Programming

Getting to know each other

Who are you?

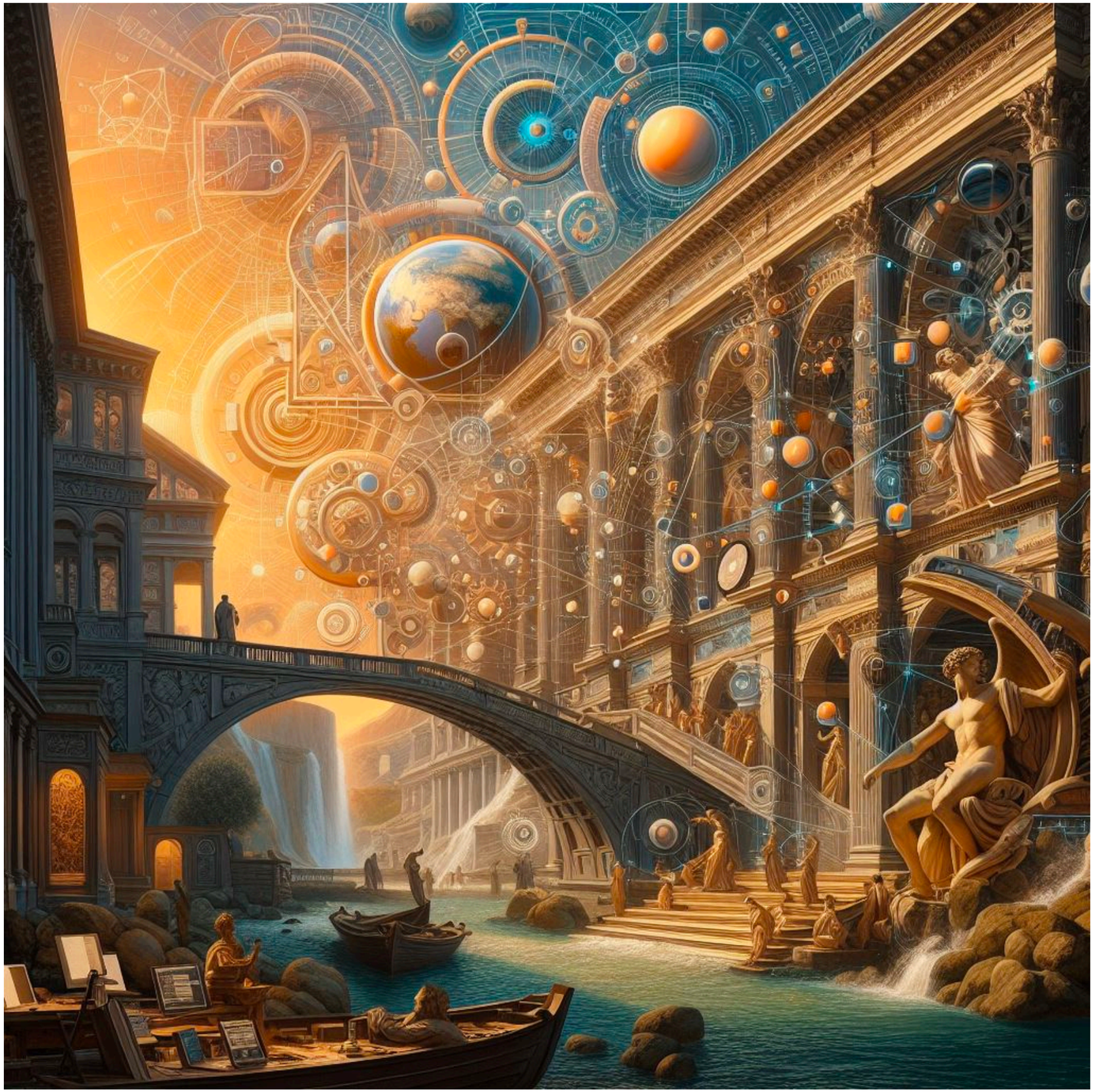
- Where are you from?
- What is your academic background?
- How good a programmer are you? (Or, How fast can I go?)
- What programming languages do you know?

slido.com

#40454

COURSE UNIT AIMS

"This course covers the fundamentals of web applications and the key architectural, functional and technological principles for the development of state-of-the-art web and cloud applications. This course has a special focus on client-side applications in scenarios that leverage data science and big data integration."



COURSE UNIT AIMS

"This course covers the **fundamentals of web applications** and the **key architectural, functional and technological principles** for the development of state-of-the-art **web and cloud applications**. This course has a special focus on **client-side** applications in scenarios that leverage data science and big data integration."

Roadmap

- Week 1: Overview. Logistics & Project Requirements. Web applications.
  First crash course: HTML, CSS, and simple TypeScript/JavaScript.
- Week 2: Software Architecture. Cloud Data, REST, GraphQL, etc.
  Second crash course: Dynamic HTML and Events in JavaScript.
- Week 3: Mock-up project presentations. Project work.
- Week 4: The *de facto* UI tech. HTML/CSS components and layout techniques
- Week 5: Advanced JavaScript. Server-side Programming in Express.
- Week 6: Evaluation. Project presentation and feedback.

Logistics

Lectures: 3h, 2 parts: lecture + break + project work
Schedule: Tue 3:30 PM,

TA: Luís Carvalho

Moodle: Slides and assignments

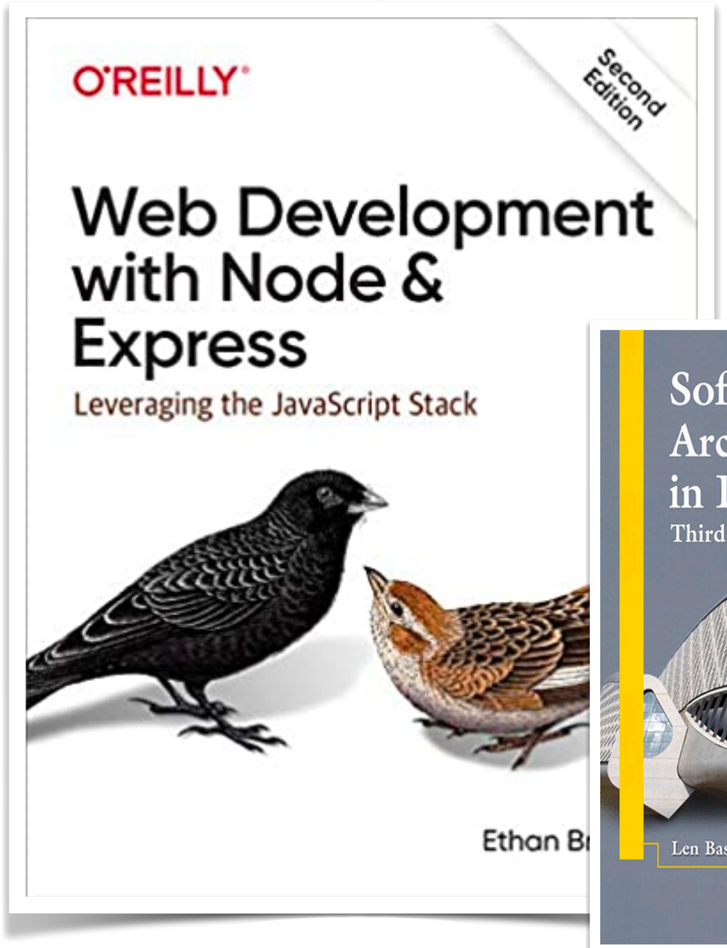
Office hours: on Teams, on Campus if needed, by appointment

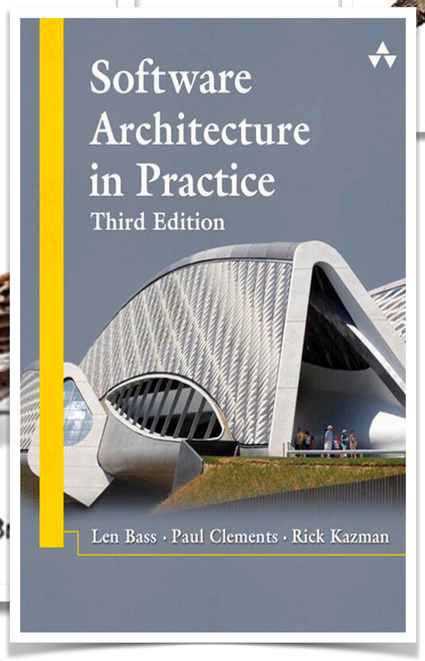
Starter code and helper code: a GitHub repository available

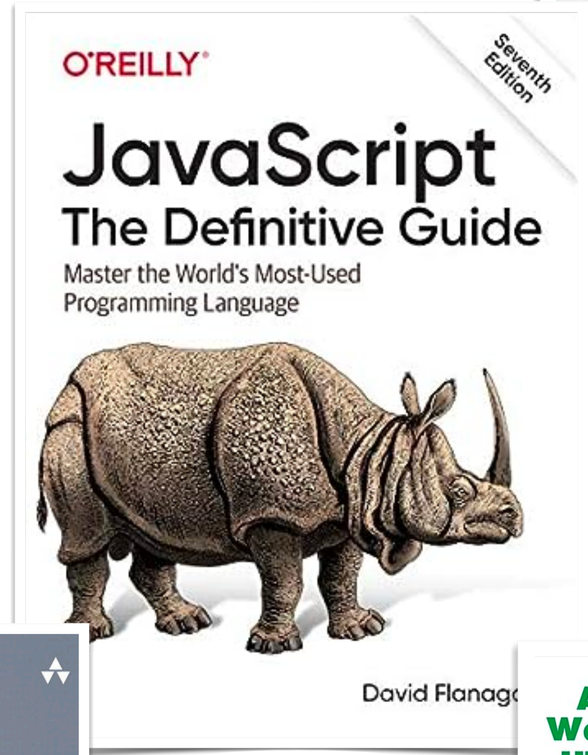
Work submission: GitHub Classroom

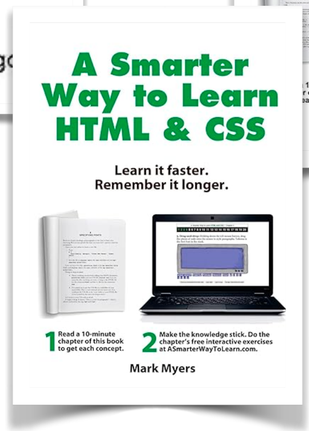
Bibliography

Main reference:









Evaluation

Project/Assignments (50%)

Exam (50%)

About AI Tools: ChatGPT, Bing, Co-Pilot, etc.

You may use it responsibly when developing the project. Not in the exam.

Responsibly = All AI mistakes are on you…

You are in an early stage, you need to learn… don't take shortcuts.

It's possible (likely) that solutions provided by AI Tools are wrong and/or too complex for your needs in this course… try to LEARN as much as you can and to do it yourself!

Use Co-Pilot Plug-in in Visual Studio Code

Use Bing instead of ChatGPT because it gives you references that you can check!

Now about the Project!

Group Formation:

Teams of 3-4 students

Scope of work

To develop a dynamic client Web page to select and show data (list, chart)

Student picked topics

My example uses: https://open-meteo.com/

Pick a free API from:
https://github.com/public-apis/public-apis
https://rapidapi.com/collection/list-of-free-apis

Project Requirements / Evaluation Criteria

- An original topic with significant data available online
- The basic layout of one or more web pages using **basic** HTML & CSS.
- Successfully filter and load data from external APIs
- Dynamic construction of layout and UI components
- The use of a form to submit data or invoke queries to the external API

Lecture 3:

- (mandatory) Presentation by each group (5-10 minutes)
  - The group (please register on Moodle/Google Docs)
  - The choice of topic!
  - The available API and its data
  - The prototype for the project (using PowerPoint)

Web and Cloud Applications

The World Wide Web

CERN DD/OC

Information Management: A Proposal

Tim Berners-Lee, CERN/DD

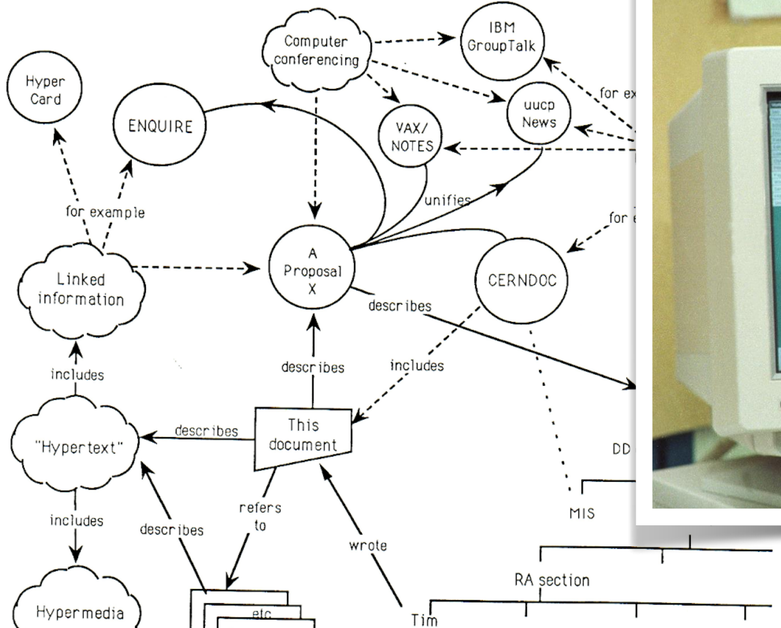
March 1989

Information Management: A Proposal

Abstract

This proposal concerns the management of general information about accelerators and experiments at CERN. It discusses the problems of loss of information about complex evolving systems and derives a solution based on a distributed hypertext sytstem.

Keywords: Hypertext, Computer conferencing, Document retrieval, Information manag
control



The World Wide Web project

World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , olicy , November's W3 news , Frequently Asked Questions .

e? Pointers to the world's online information, subjects , W3 servers, etc.

on the browser you are using

ucts A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)

Details of protocols, formats, program internals etc

Paper documentation on W3 and references.

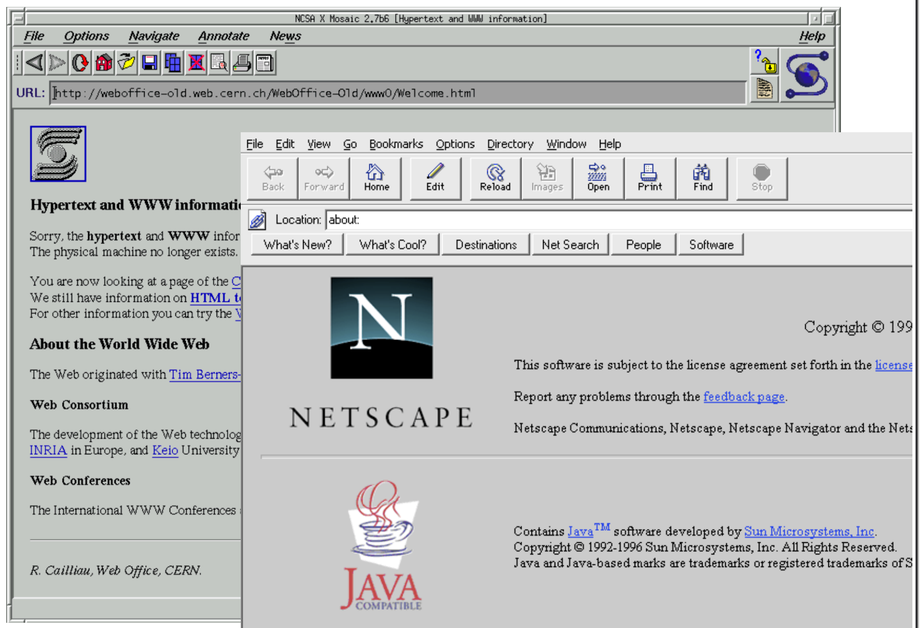
A list of some people involved in the project.

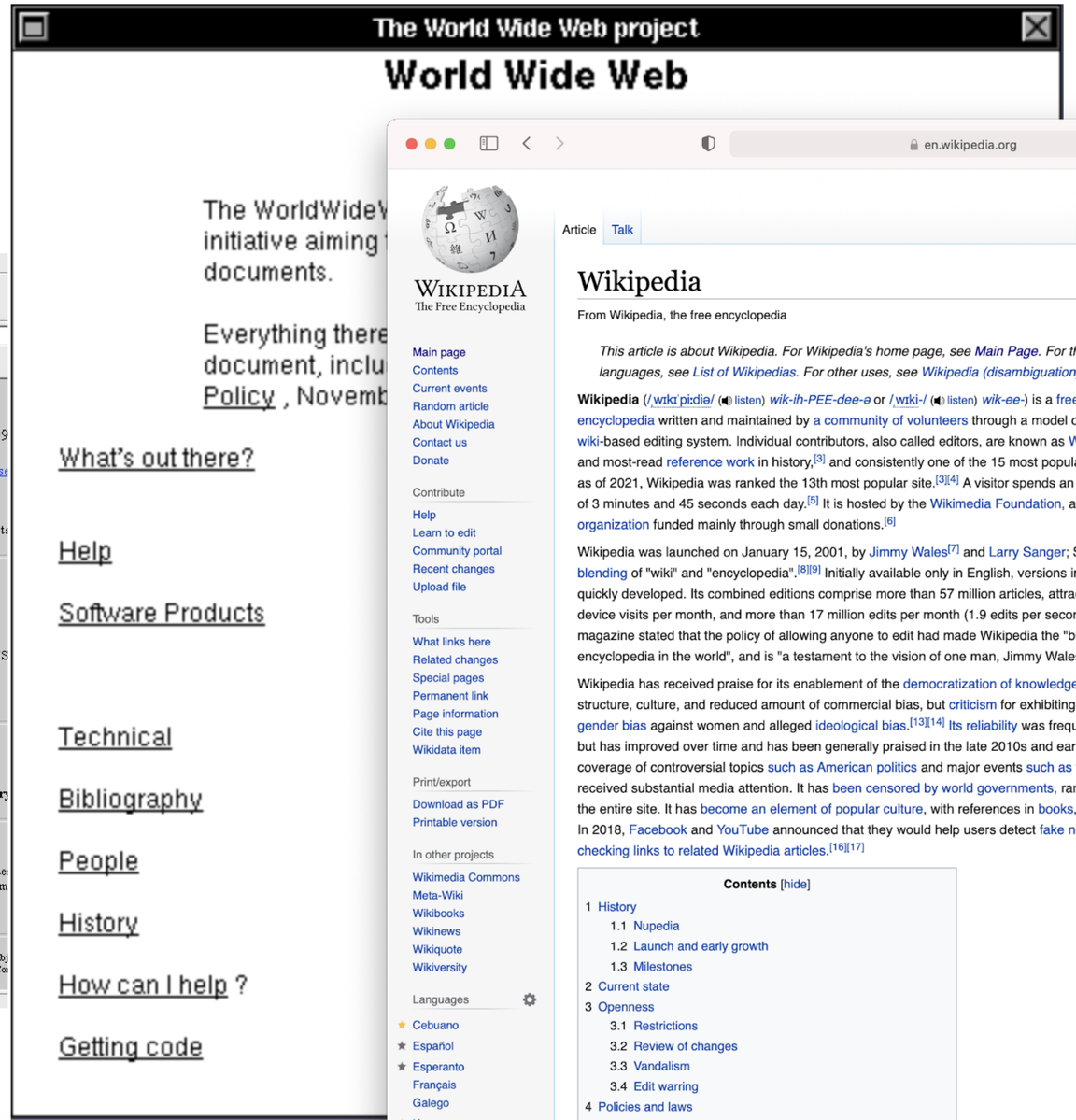
A summary of the history of the project.

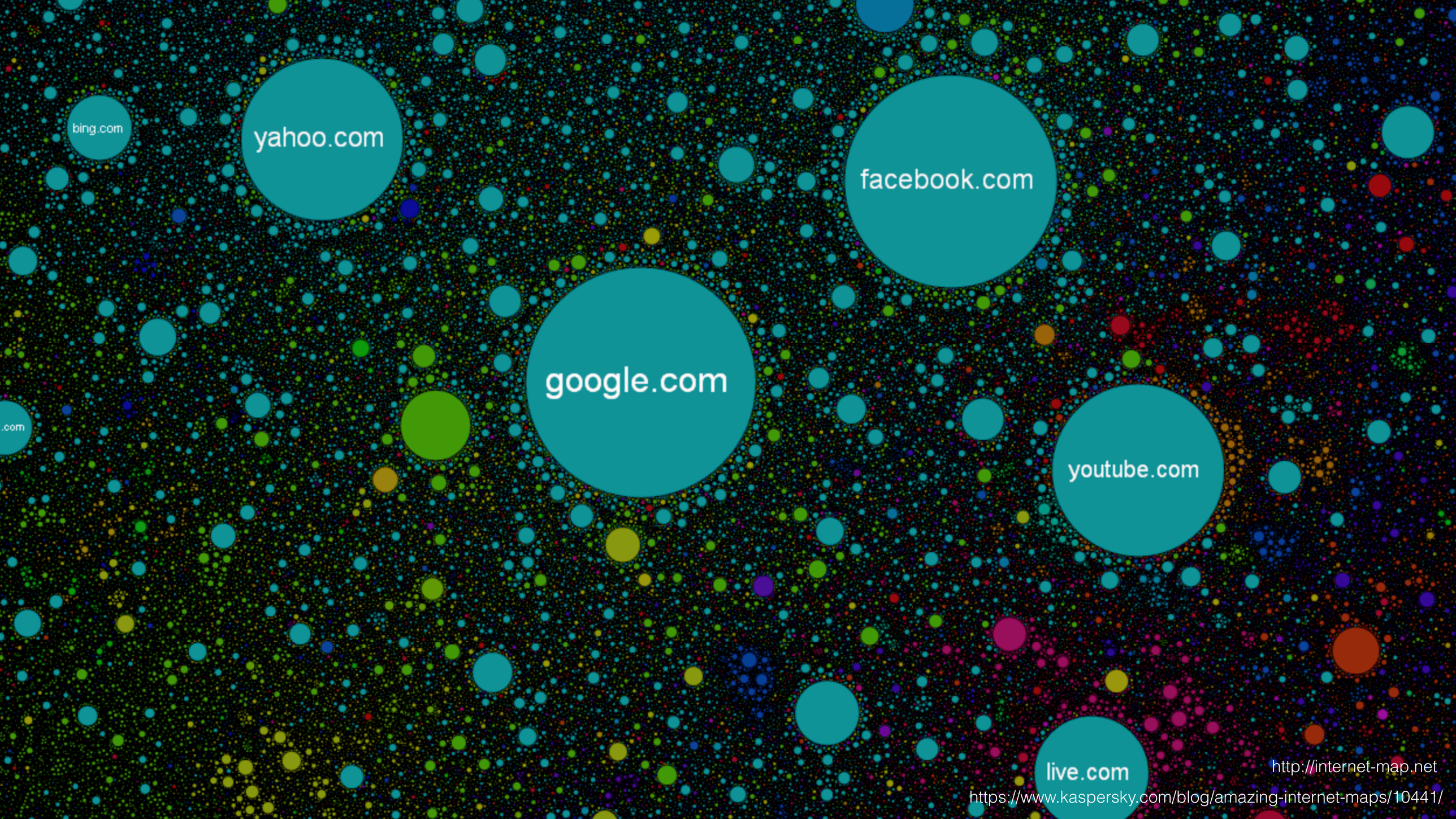
? If you would like to support the web..

Getting code Getting the code by anonymous FTP , etc.

The World Wide Web - The browser






bing.com
yahoo.com
facebook.com
google.com
youtube.com
live.com
http://internet-map.net
https://www.kaspersky.com/blog/amazing-internet-maps/10441/

Web and Cloud (Internet) Applications

Internet application | ˈɪntənɛt aplɪˈkeɪʃ(ə)n |, noun

Any application that uses the internet to consume and/or provide services and data.

Web and Cloud Applications

Web application | wɛb aplɪˈkeɪʃ(ə)n |, noun

An internet application that runs on a browser and obtains HTML pages and data from a web server.

Mobile application | ˈməʊbʌɪl aplɪˈkeɪʃ(ə)n |, noun

An application installed and running in a mobile device, usually dependent on online data sources.

Web service | wɛb ˈsəːvɪs |, noun

useful to support Web applications

A computational procedure available on the Internet to provide services and data to other apps and services. Usually associated to binding technologies like SOAP or REST.

Web and Cloud Applications

Cloud application | klaʊd aplɪˈkeɪʃ(ə)n |, noun

An internet application deployed on an independent hosting service, providing computation and additional resources and features like replication and storage.

Service-based architecture | ˈsəːvɪs beɪs ˈɑːkɪtɛktʃə |, noun

A conceptual structure and logical organisation of web services, usually implemented using orchestration languages and tools.

Data-Centric Applications | Dataˈdeɪtə-ˈsɛntrɪk ˌæplɪˈkeɪʃ(ə)nz |, noun

Applications that are developed primarily based on resource-based rules, making their data available to others through well defined interfaces

Panorama of Web & Cloud Applications

They are the base for any **digital transformation**

End-user applications are the tip of the Iceberg

Web to Mobile, Desktop Computers to Car Navigation Systems

data sources are ubiquitous

User interactions, User activity, Smart devices, …

Endless possibilities to **retrieve and mashup** data

Cloud computing is…

- On-demand remote computational resources, without user intervention (Computation and data storage)
- Remote network access to the assigned resources (heterogeneous client platforms)
- Resources can be allocated to multiple clients/services
- Elasticity (you pay as you use)

Flexible usages with Cloud Infrastructures

Infrastructure as a Service

Virtualisation of computational resources
off premises.
You manage the operating system, e.g. AWS, Azure

Platform as a Service

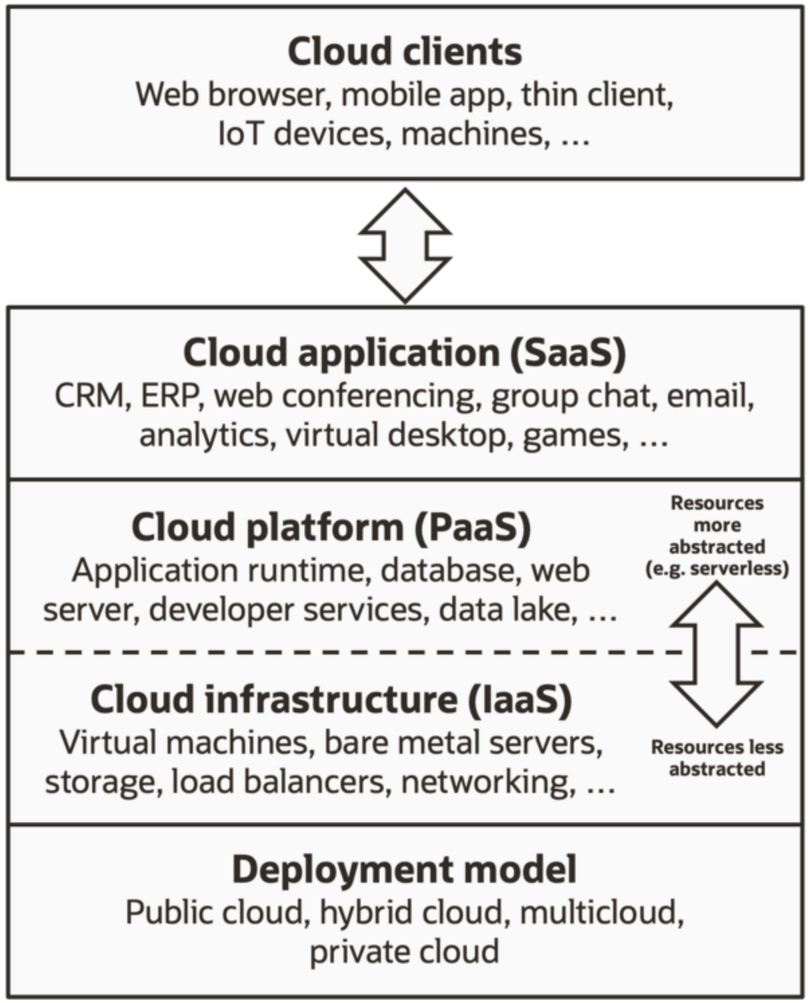
An abstraction layer for resources that allows development in the cloud

e.g. AWS, Azure, Google App Engine, OutSystems

Software as a Service

Contract to customise and use a particular product in the cloud

e.g. Office365, Google GSuite, Wordpress



In https://en.wikipedia.org/wiki/Cloud_computing

Development of Web and Cloud Applications

Opportunities

- Huge user base
- Flexible Computing power
- Data gathering mechanisms
- Data analytics & visualisation
- Short delivery cycles
  (pushed to users)

Development of Web and Cloud Applications

Opportunities

- Huge user base
- Flexible Computing power
- Data gathering mechanisms
- Data analytics & visualisation
- Short delivery cycles (pushed to users)

Challenges

- Systematic development process
- Robust quality assurance methods
- Scalability & performance
- Security (attacks and leaks)
- User experience & user engagement

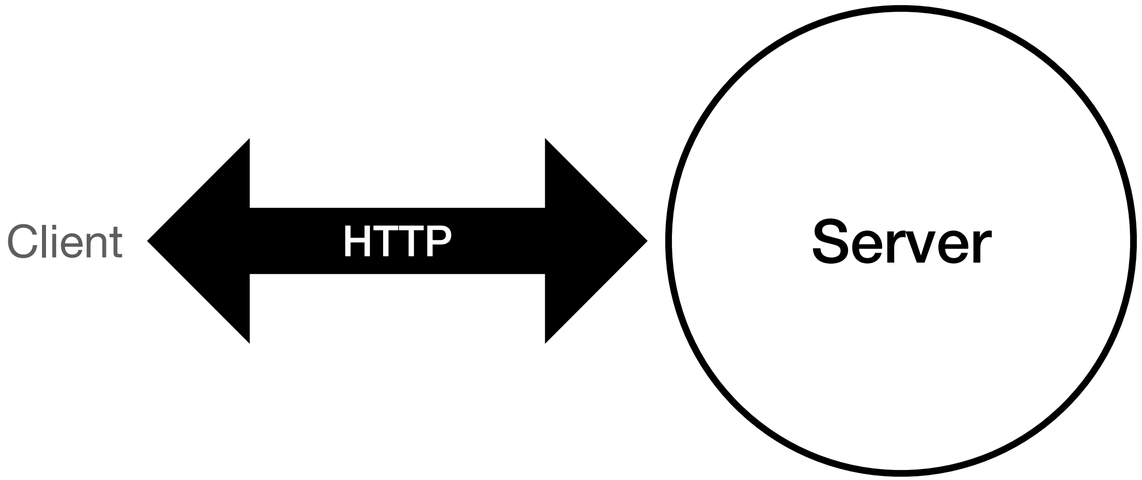
Cloud computing demands

Flexible services

Modularity in the development

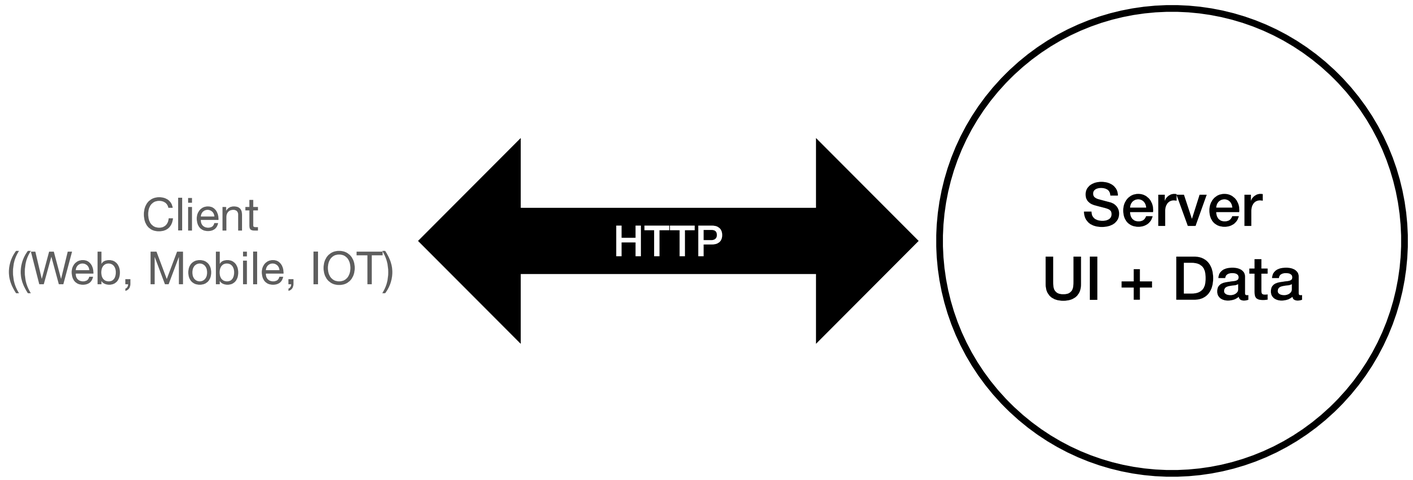
Loosely coupled of components (few dependencies)

Basic Building Blocks



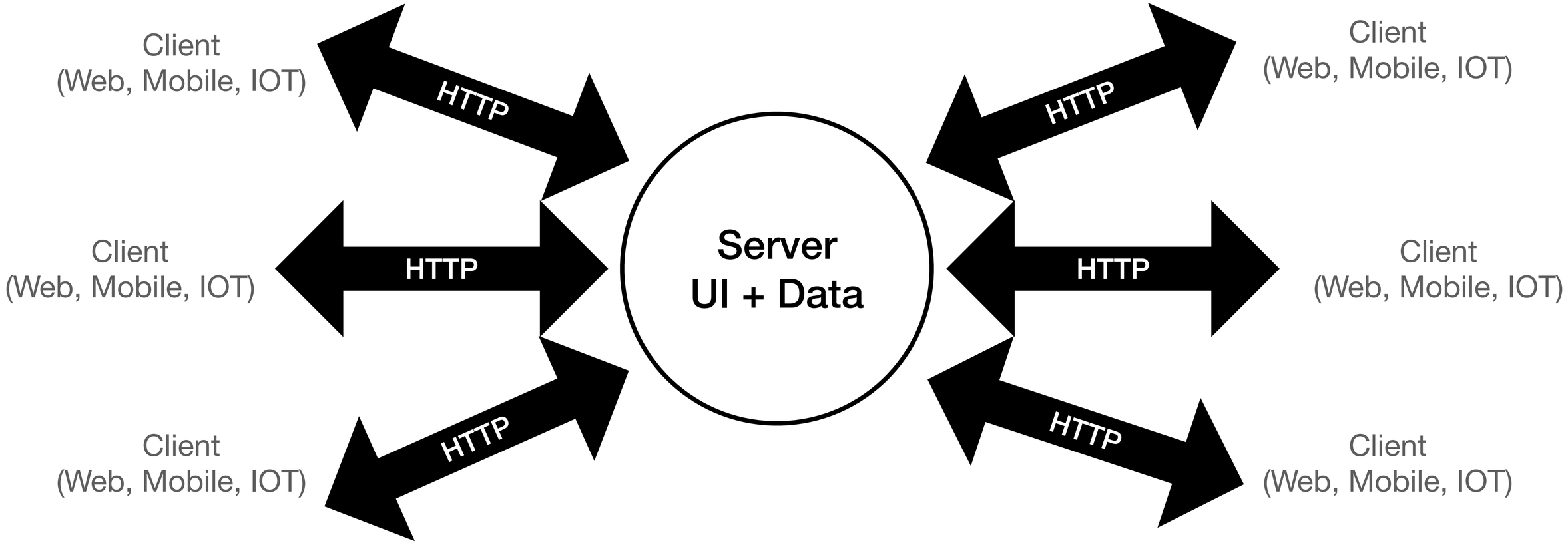
HTTP - Hypertext Transfer Protocol

Basic Building Blocks

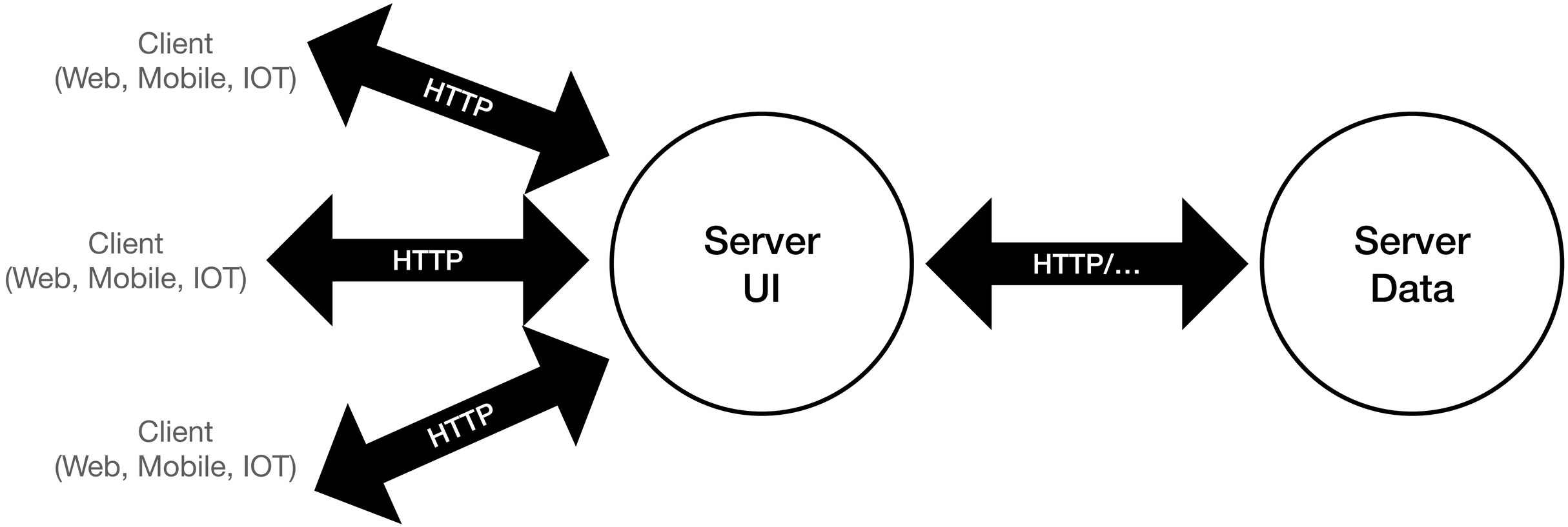


HTTP - Hypertext Transfer Protocol

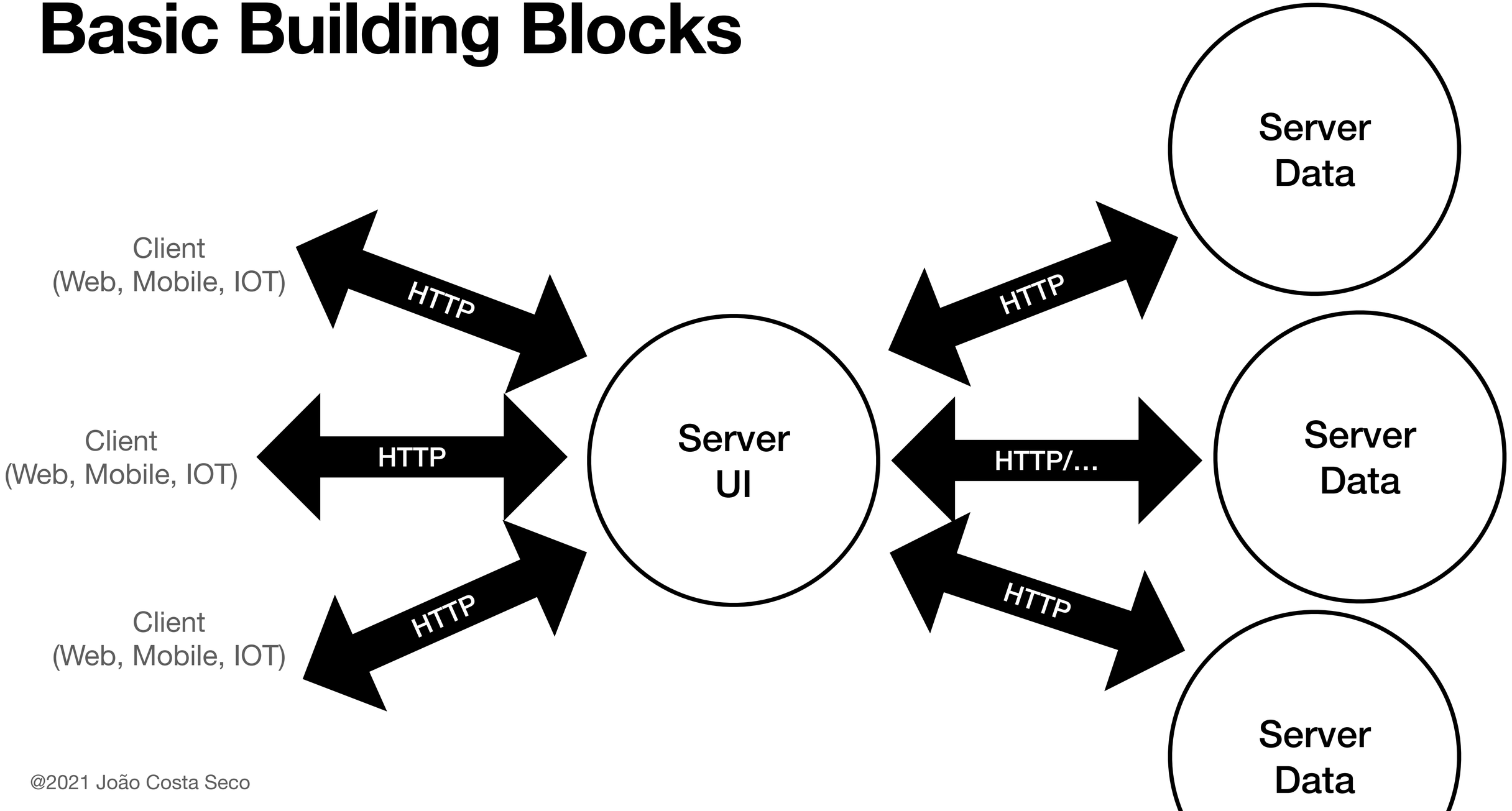
Basic Building Blocks

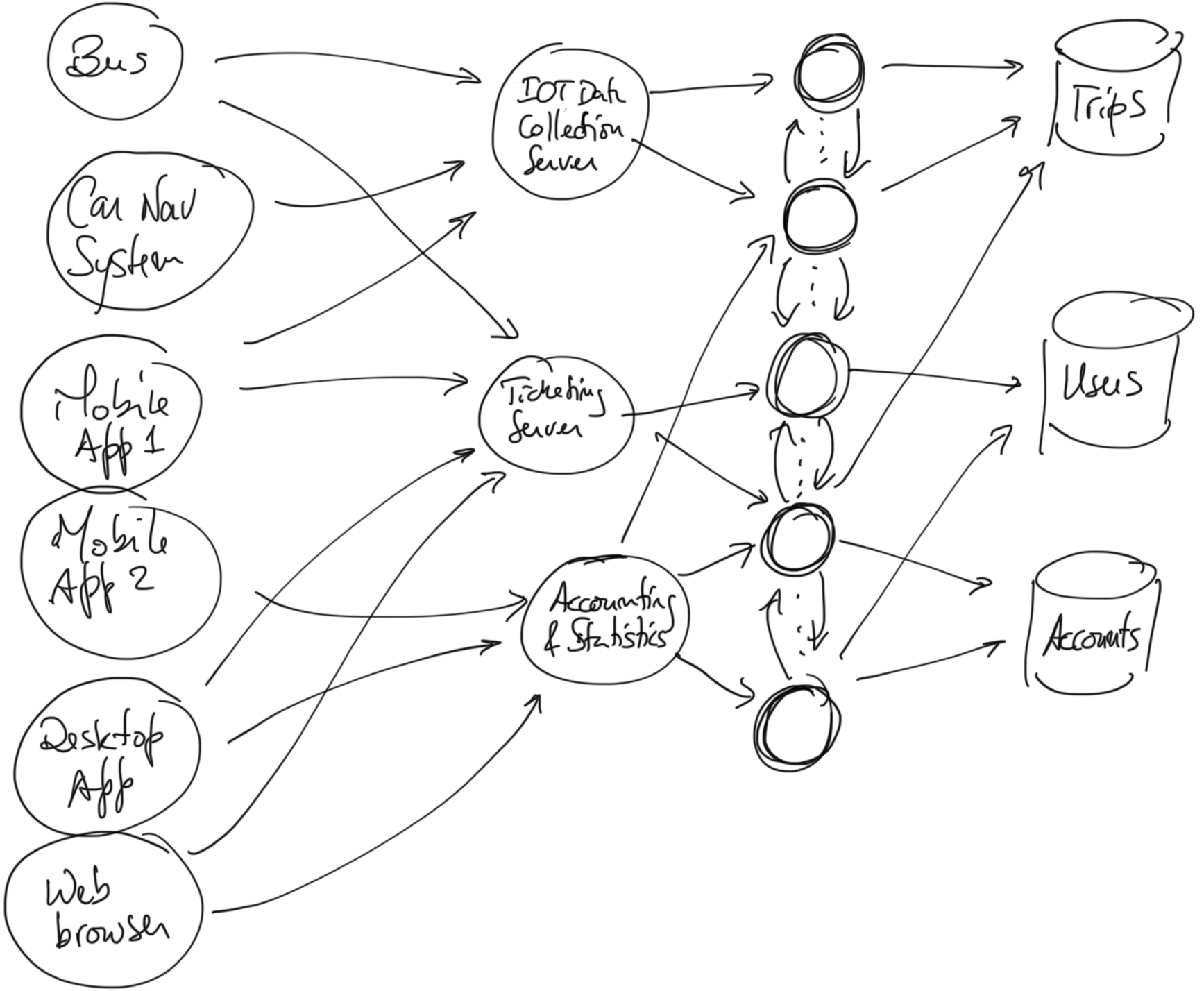


Basic Building Blocks



Basic Building Blocks





Break Down

Partition is related to

Abstraction/Reuse

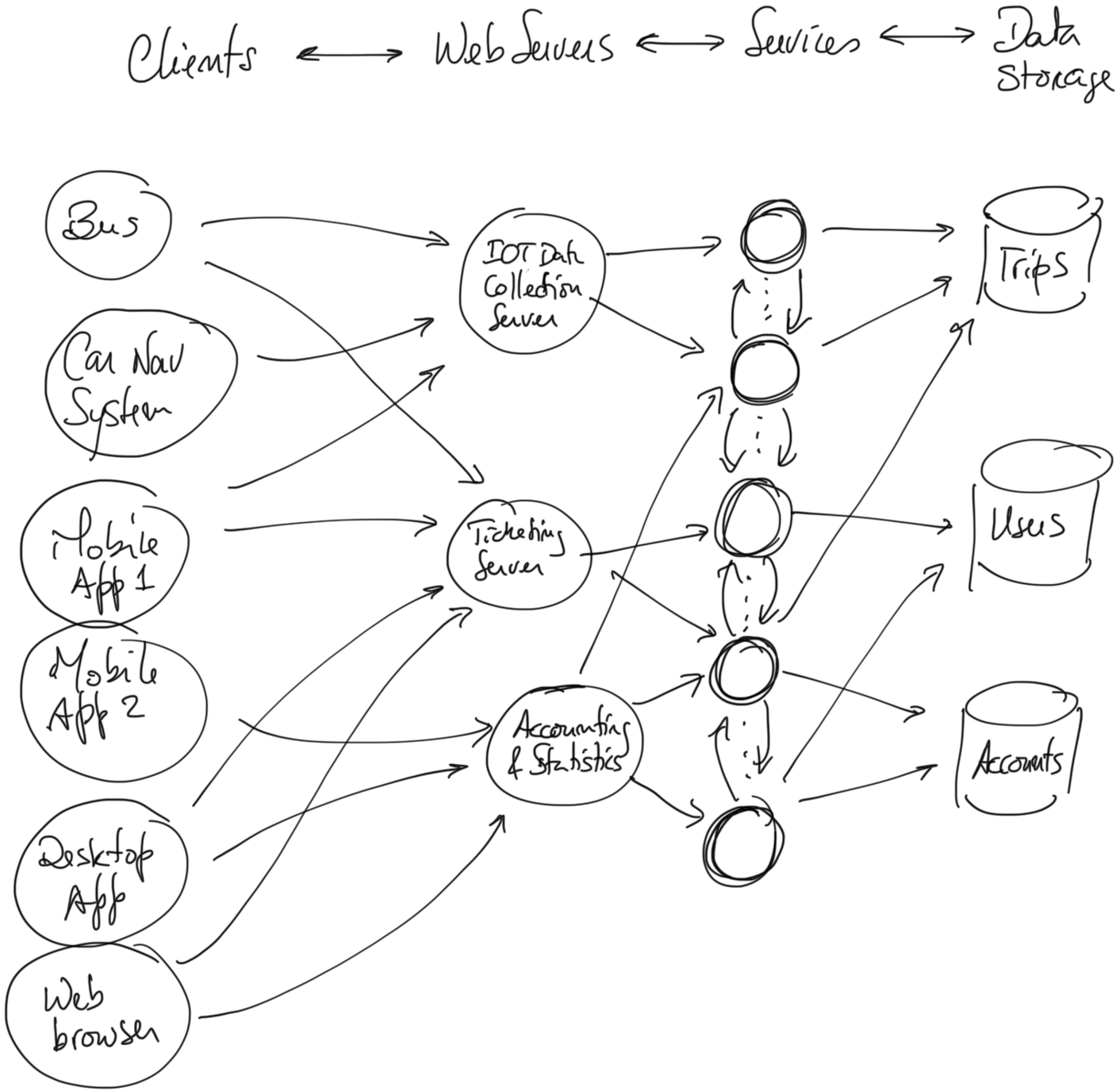
Different Concerns

Complexity

Technology

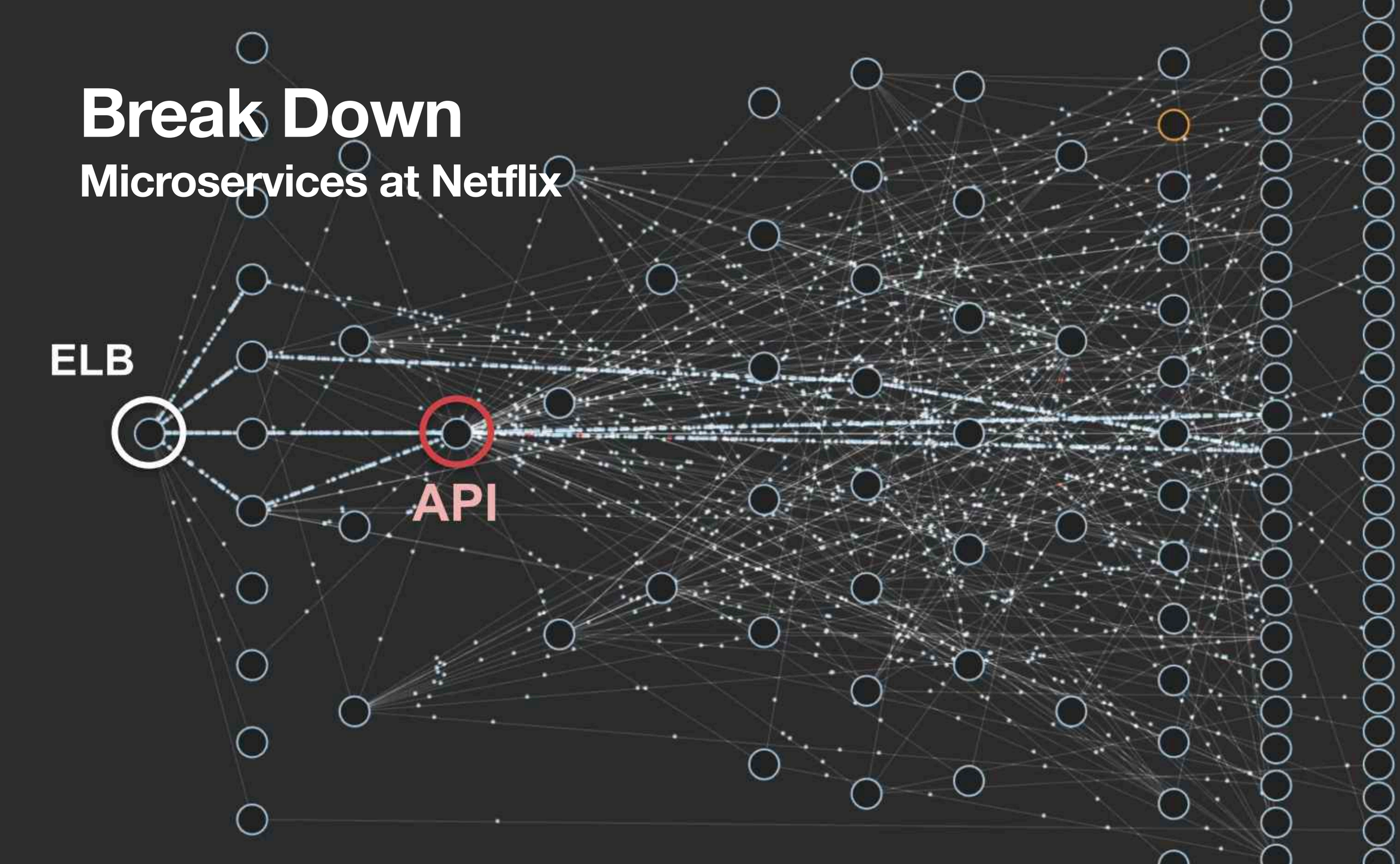
Ownership

Performance



Break Down

Microservices at Netflix



Software Architecture

What is it? Why do we need it?

Good organisation of the internal structure of systems leading to:

- Productive development
- Easy evolution
- Easy maintenance
- Trustworthiness (Correctness, Security)
- High Performance Level

Technology Stack

Three tier architecture

Web client-side applications

HTML, Javascript — React/Vue/Angular, TypeScript

Server-side applications

JavaScript, Express — Java/Spring, Python/Flask

Data storage and manipulation

WebServices/Relational/NoSQL

The spectrum of client-side applications

website / static HTML pages

static data, poor behaviour, efficient response times (w/caching), dynamic websites are sometimes expanded for more efficiency.

website / dynamic server rendered HTML pages

poorer loading times, poorer development features (distance between code and result). Poor modularity, high usage of templating languages.

website / asynchronous communication (AJAX)

better loading times, development closer to the result (no templating needed). Better modularity and testing conditions.

web applications (Single Page Application)

complete application context, service based back-ends, richer behaviour and fluid UI

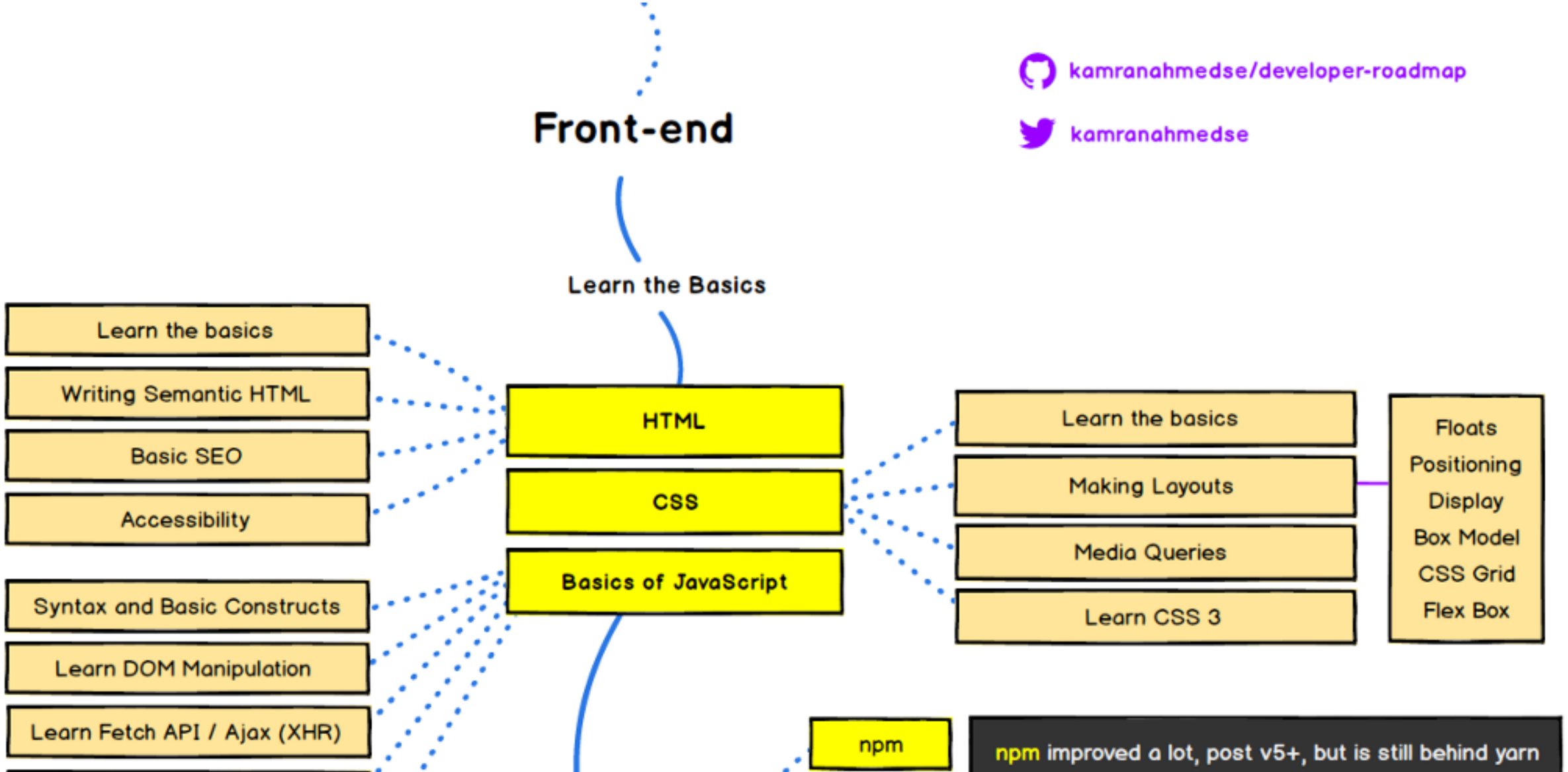
progressive web applications (PWA)

device agnostic, and yet, closer to native client applications (storage, resources, responsive UI, notifications, etc.)

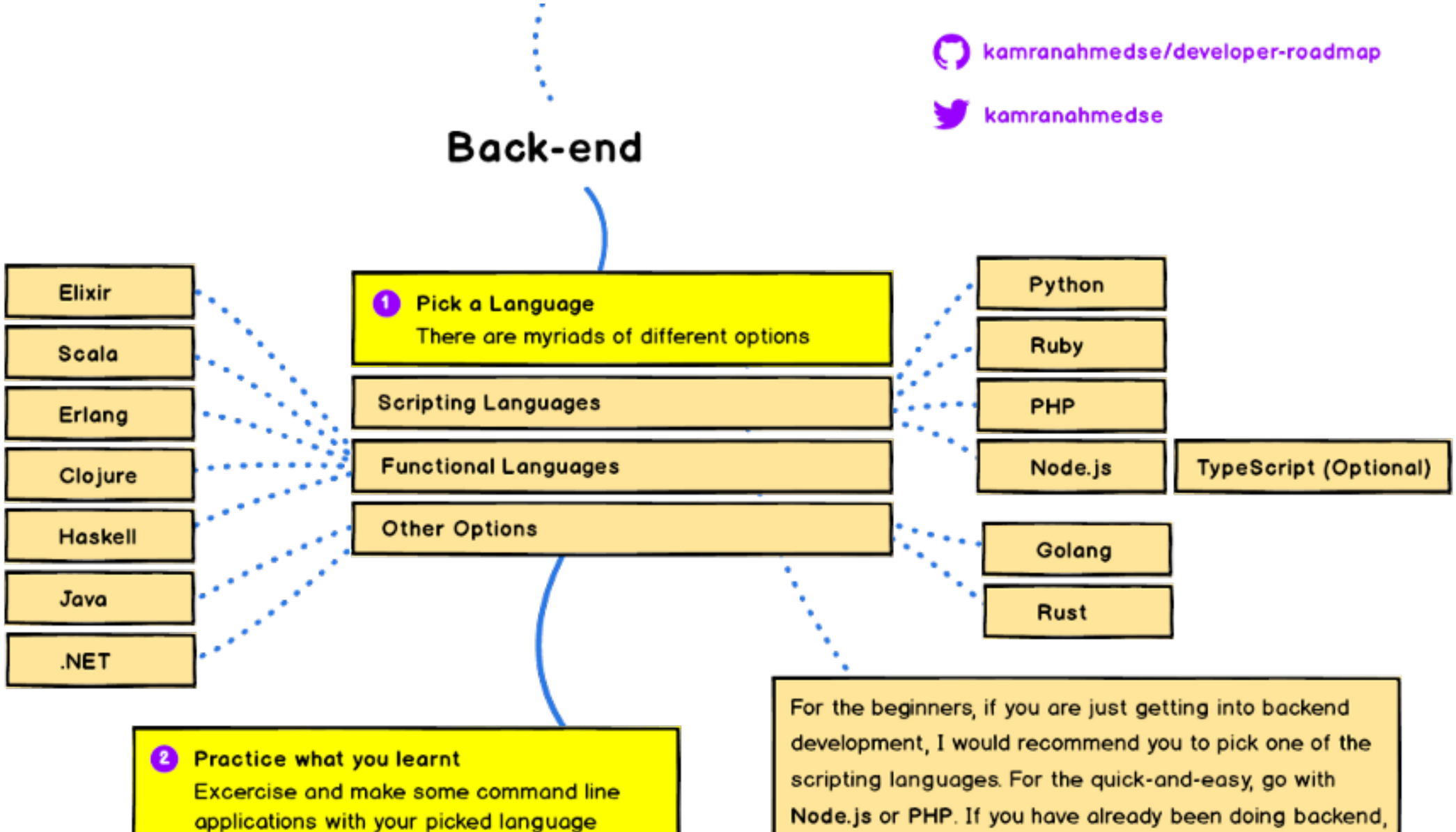
mobile applications

more device resources, many times PWA in a shell that links to the device native features.

The spectrum of client-side technologies



The spectrum of server-side technologies



Extra Readings

W3Schools tutorials

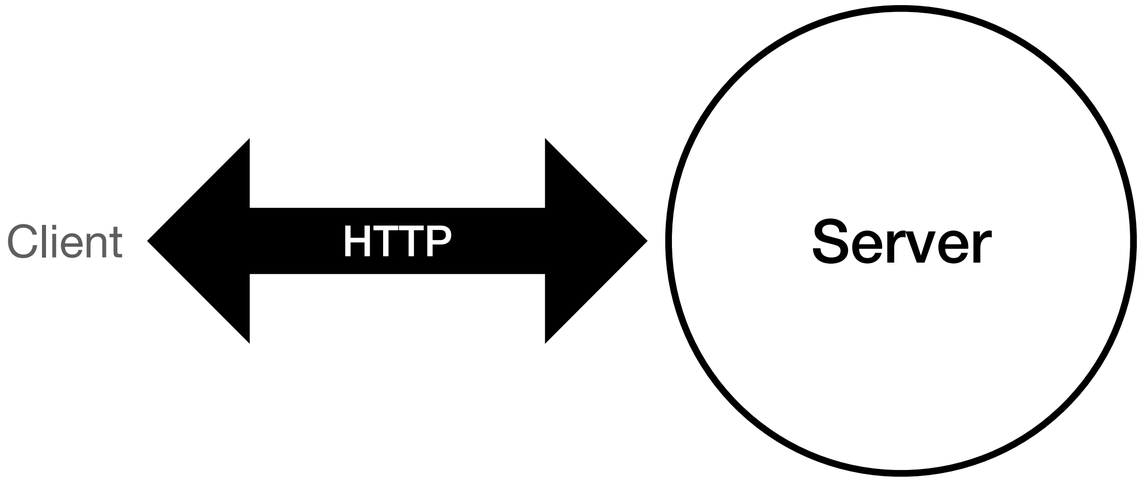
Mozilla Developer Network tutorials

Intro to Cloud Computing (CACM)

Break

Web Technologies

Basic Building Blocks: HTTP



HTTP - Hypertext Transfer Protocol

WWW = HTTP + HTML + Server + Browser

HTTP Request = URL + Method + Body + …

HTTP Response = Code + Body + …

URL / URI

http://serveraddress/pathtoresourse:port#id?querystring

protocol (http, ssh, git, mailto, etc.)

server address (www.novasbe.com) or IP address (e.g. 193.136.122.122)

path to resource (just like the path to a file in a disk, but with parameters)

port (a server may be listening in different ports with different applications)

identifier of the element in an HTML page

querystring includes parameters to the request

The HTTP protocol - URL

Uniform Resource Locator

```
scheme://user@host:port/path?query#fragment
```

```
https://www.google.com/search?q=summer

git://costaseco@bitbucket.org/costaseco/todo.git

http://www.google.com/search?q=summer

http://localhost:3000/todo-list?status=done#last
```

Method = GET, POST, PUT DELETE, …

Body = HTML, JSON, XML, Form-data, “data”

The HTTP protocol

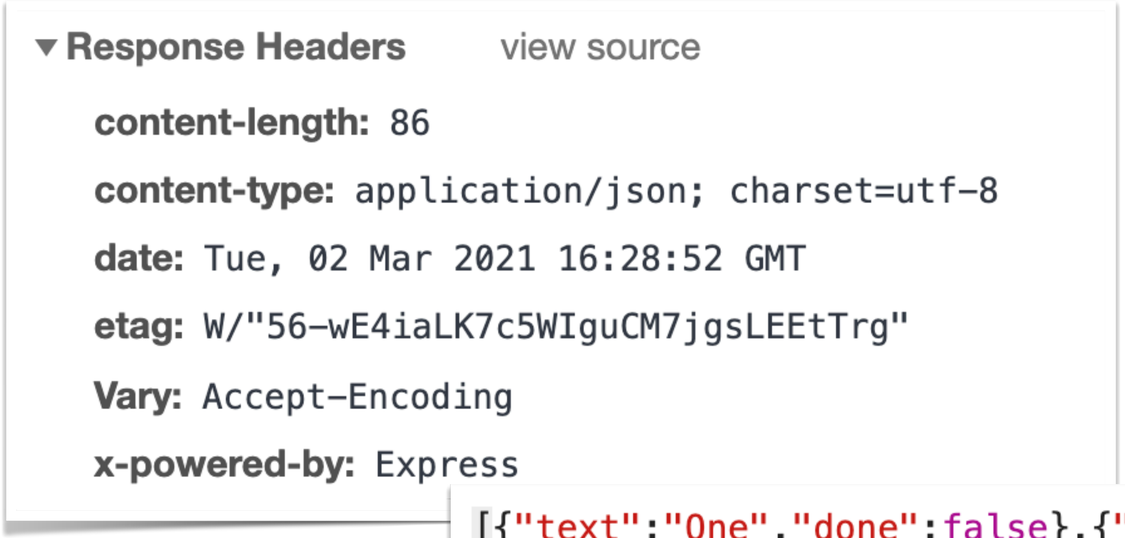
The request

- The target resource, a **URL**
- The HTTP **method**, a verb that describes an action: GET, POST, PUT
- The **body** of the request (optional): one or more files containing a sequence of name/value pairs, a json object, a binary file, etc. (MIME type - Multipurpose Internet Mail Extensions)
- The **kind of content** in the request (application/json, multi-part/form-data, etc.)
- Security information (using **cookies** and **tokens**)
- Other **headers** indicating the client of the request (user-agent) and what kind of answer is expected (text/html, application/json, etc.)

The HTTP protocol

The Response

- The Status code of the response
- The kind of content in the response (text/html, application/json, etc.)
- The body of the response (optional) containing data. One or more resources.




```
[{"text":"One","done":false},{"text":"Two","done":true},{"text":"Three","done":false}]
```

Data exchange formats

- Servers and services use a myriad of technologies and native data representations and need a common data format to communicate.
- The most common data formats are hierarchical and text-based
- JSON: Javascript object notation (Lists, Records & basic types)
- XML: Extensible Markup Language (Tree-like representation of data)

Data exchange formats

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

es and native data
t to communicate.

and text-based

& basic types)

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

https://json.org/example.html

HTTP Status Codes

httpstatuses.com is an easy to reference database of HTTP Status Codes with their definitions and helpful code references all in one place. Visit an individual status code via `httpstatuses.com/code` or browse the list below.

@ Share on Twitter ⊕ Add to Pinboard

1×× Informational

100 Continue
101 Switching Protocols
102 Processing

2×× Success

200 OK
201 Created
202 Accepted
203 Non-authoritative Information
204 No Content
205 Reset Content
206 Partial Content
207 Multi-Status
208 Already Reported
226 IM Used

3×× Redirection

300 Multiple Choices
301 Moved Permanently
302 Found
303 See Other
304 Not Modified
305 Use Proxy
307 Temporary Redirect
308 Permanent Redirect

4×× Client Error

400 Bad Request
401 Unauthorized
402 Payment Required
403 Forbidden
404 Not Found
405 Method Not Allowed
406 Not Acceptable
407 Proxy Authentication Requ
408 Request Timeout
409 Conflict
410 Gone
411 Length Required
412 Precondition Failed
413 Payload Too Large
414 Request-URI Too Long
415 Unsupported Media Type
416 Requested Range Not Satisfiable
417 Expectation Failed
418 I'm a teapot
421 Misdirected Request
422 Unprocessable Entity
423 Locked
424 Failed Dependency
426 Upgrade Required
428 Precondition Required
429 Too Many Requests
431 Request Header Fields Too Large
444 Connection Closed Without Response
451 Unavailable For Legal Reasons
499 Client Closed Request

5×× Server Error

500 Internal Server Error
501 Not Implemented
502 Bad Gateway
503 Service Unavailable
504 Gateway Timeout
505 HTTP Version Not Supported
506 Variant Also Negotiates
507 Insufficient Storage
508 Loop Detected

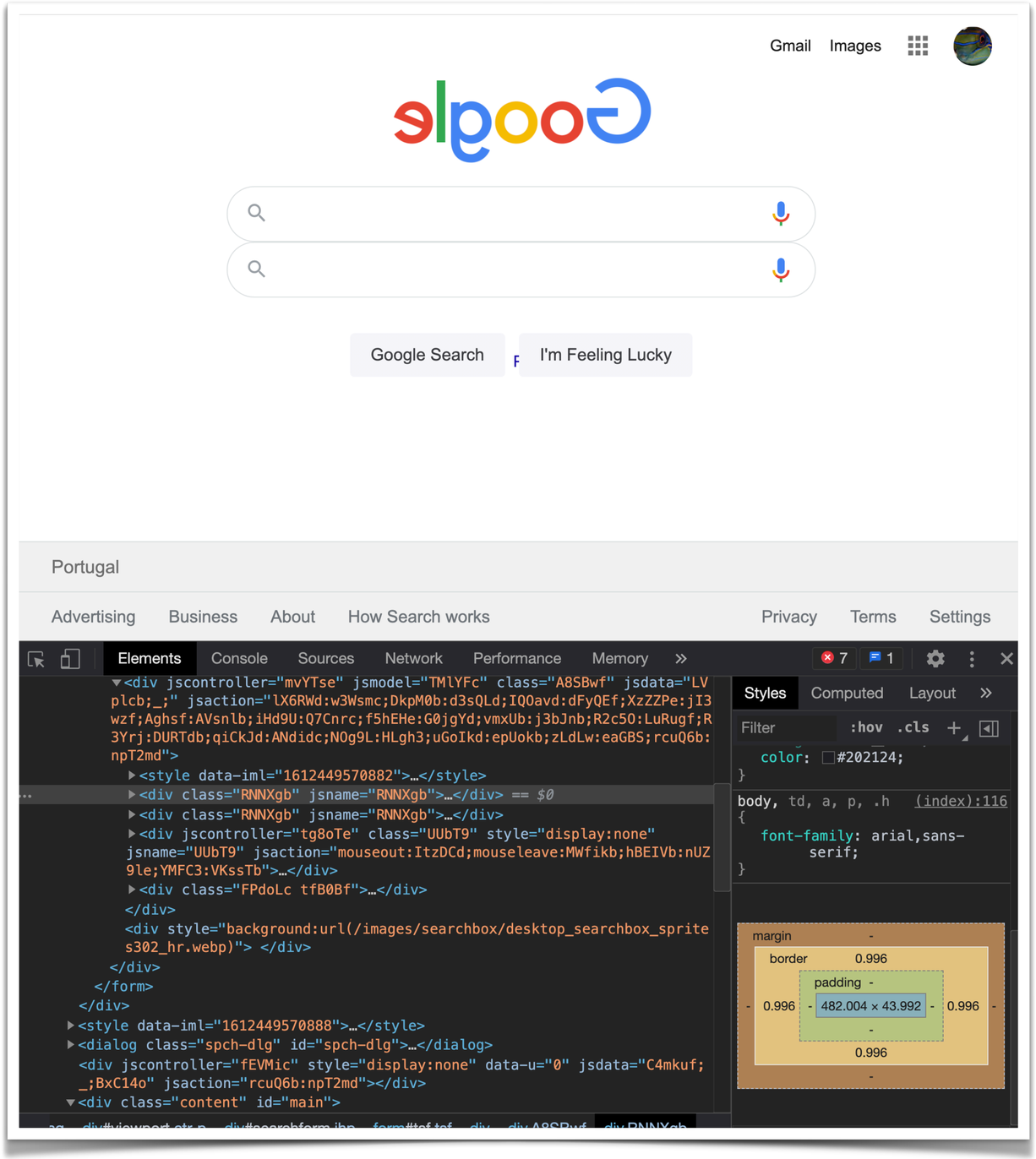
HTML in a 

MDN web docs
moz://a

HTML under the hood

Developer tools

- Learn how to use the browser inspector (press F12)
- Use right-button “inspect” on the page
- Play around and edit something to make your own Google page



Hypertext Markup Language (HTML5)

Not a programming language

W3C recommendation for the WWW

It is a bundle of technologies

HTML Markup Language, Javascript, and CSS

Technology under the hood

webpages and many mobile applications

Low entry point and flexible

As it is, not so good in the mid-range (needs more abstraction layers)

The inventor of the WWW

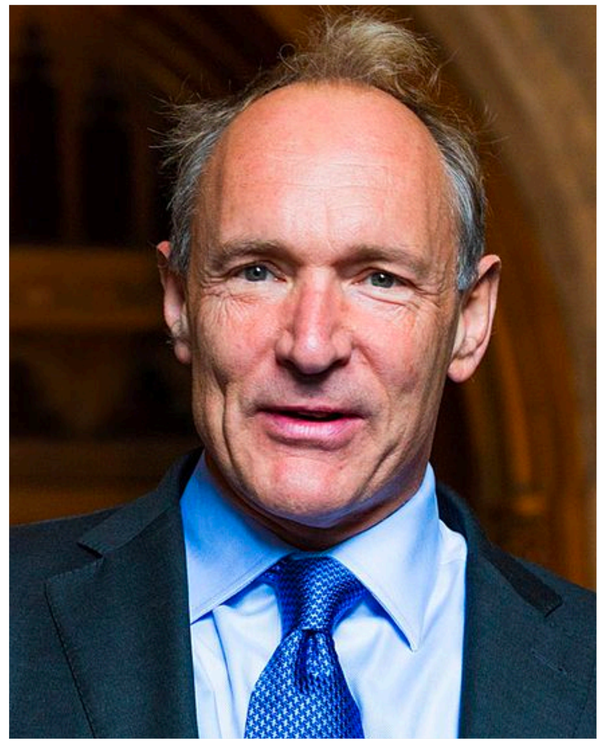
Sir Timothy Berners-Lee

CERN 1989

Director of the W3C (WWW consortium)

The first communication using HTTP protocol

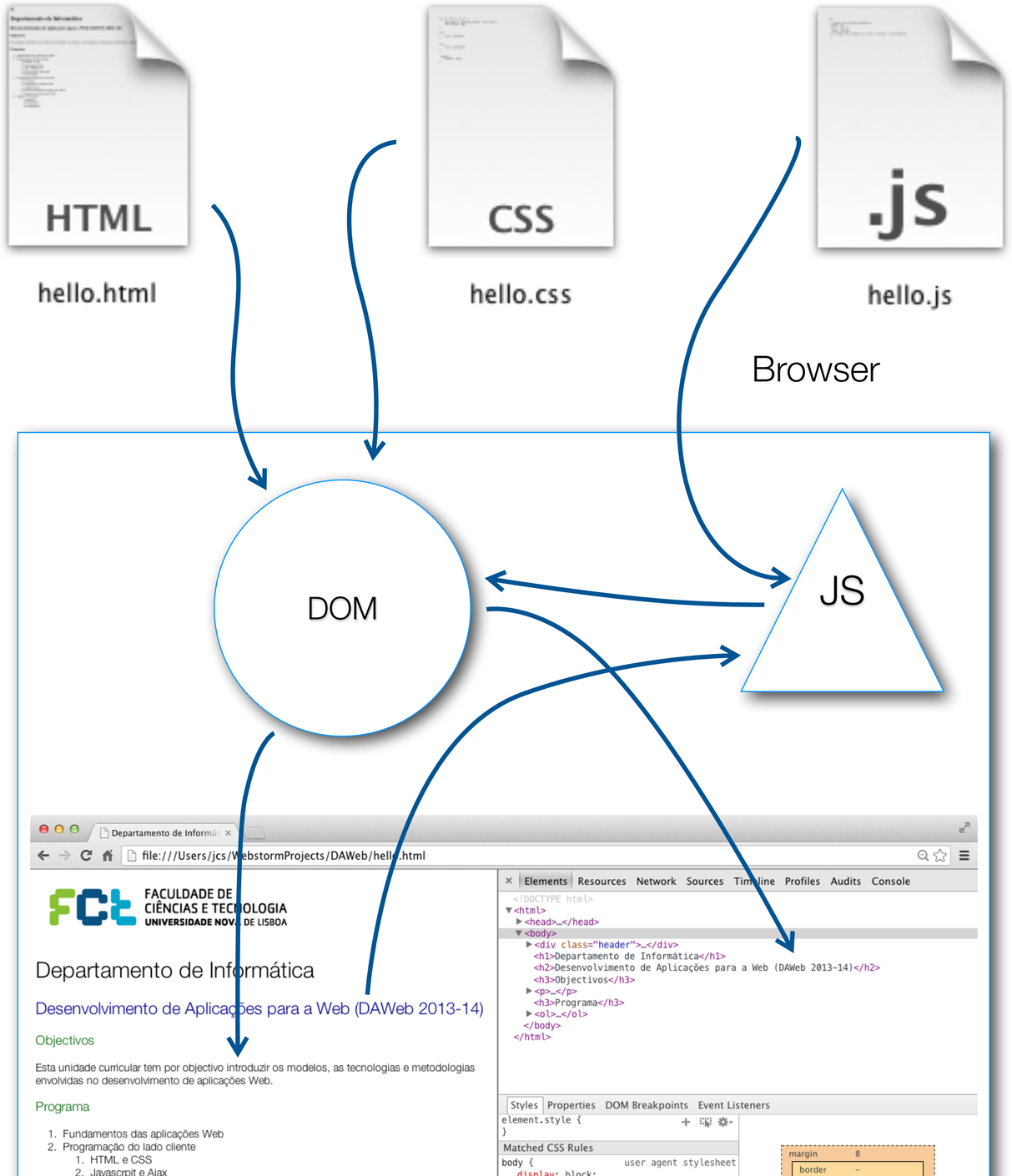
The first browser (called WorldWideWeb)



Web Browser

Logic architecture

- Source files with static content: HTML / Images / CSS / Javascript
- Internal representation (Document Object Model)
  - Abstract Elements
  - Style annotations
  - Event handlers (behaviour)
- User-interface (visible elements)
- Extra behaviour (Javascript)



HTML in a 

- HTML derives from the XML format
- A nested structure of HTML elements
- HTML Elements are formed by content enclosed with tags

  This is a paragraph

- Different tags mean different things in a HTML document
  - **<html> <head> <body>   ** …
- HTML Elements may have attributes
  - Generic: set on any HTML element. (ex: **id**, **class**, **hidden**, …)
  - Particular: img/**src**, option/**value**, etc.
  - Events: dependent on the kind of element body / **onload**, button / **onclick**, etc.

```
<html>
<head>
    <!-- Head defines invisible information about the page -->
    <!-- Meta information, only read by the machine -->
    <meta charset="utf8">
    <meta name="description" content="My personal web page">
    <meta name="keywords" content="HTML, CSS, XML, Javascript"
    <meta name="author" content="João Costa Seco">

    <meta property="og:url" content="http://www.about.me" />
    <meta property="og:type" content="article" />
    <meta property="og:title" content="My favourite cyber spac
    <meta property="og:description" content="This is my person
    <meta property="og:image" content="http://www.about.me/ima

    <link href="/css/main.css" rel="stylesheet">
    <link rel="icon" href="favicon.ico">

    <style type="text/css">
        p { color: #26b72b; }
    </style>

    <script src="/js/main.js"/>
    <script>
        var a = 1;
    </script>

    <title>This is a web page</title>

    <!-- Up to this point, everything is invisible to the end-
</head>
<body>
    <p>Hello World!</p>
</body>
</html>
```

HTML tags in a 

- `<!DOCTYPE html>` defines the kind of document
- `<html></html>` encloses all the elements, it defines the root element
- `<head></head>` defines meta-information, stylesheets and scripts
- `<body></body>` encloses the elements that define the content
- `<p> <h1>… <ul> <div>` define block elements
- `<span> <strong> <i>` define inline elements
- `&lt; &gt; &#233;` escaped characters

MDN web docs
moz://a

HTML in a 

```
    <!-- Meta information, only read by the machine -->
    <meta charset="utf8">
    <meta name="description" content="My personal web page">
    <meta name="keywords" content="HTML, CSS, XML, Javascript">
    <meta name="author" content="João Costa Seco">

    <meta property="og:url" content="http://www.about.me" />
    <meta property="og:type" content="article" />
    <meta property="og:title" content="My favourite cyber space" />
    <meta property="og:description" content="This is my personalized web
page." />
    <meta property="og:image" content="http://www.about.me/images/me.jpg" />

    <link href="/css/main.css" rel="stylesheet">
    <link rel="icon" href="favicon.ico">

    <style type="text/css">
        p { color: #26b72b; }
     * { font-family: sans-serif; }
    </style>

    <script src="/js/main.js"/>
    <script>
        var a = 1;
    </script>

    <title>This is a web page</title>

    <!-- Up to this point, everything is invisible to the end-user -->
</head>
<body>
```

HTML in a 

Monday Times

- News
- Sports
- Weather

News Section

News Article

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque in porta lorem. Morbi condimentum est nibh, et consectetur tortor feugiat at.

News Article

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque in porta lorem. Morbi condimentum est nibh, et consectetur tortor feugiat at.

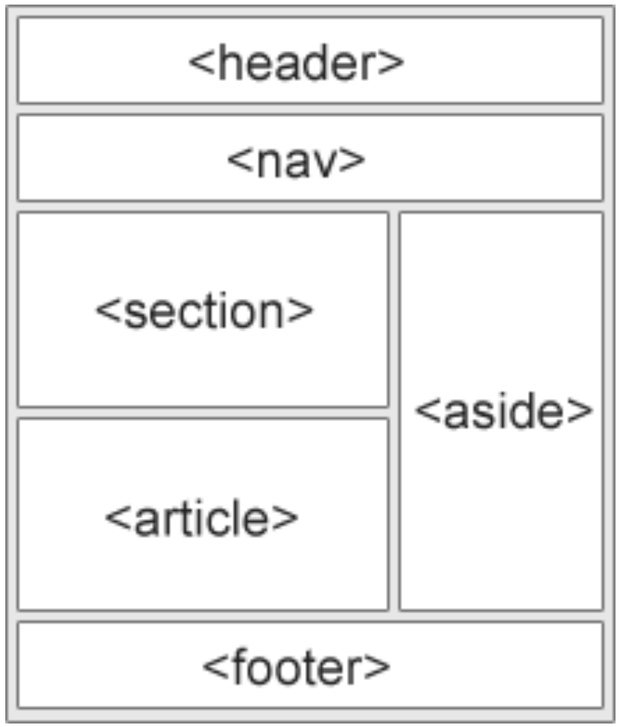
© 2014 Monday Times. All rights reserved.

```
<!DOCTYPE html>
<html>
    <head>…</head>
    <body>
        <header>
            <h1>Monday Times</h1>
        </header>

        <nav>
            <ul>
                <li>News</li>
                <li>Sports</li>
                <li>Weather</li>
            </ul>
        </nav>

        <section>
            <h2>News Section</h2>
            <article>
                <h2>News Article</h2>
                <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
                   Pellentesque in porta lorem. Morbi condimentum est
                   nibh, et consectetur tortor feugiat at. </p>
            </article>
            <article>
                <h2>News Article</h2>
                <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
                   Pellentesque in porta lorem. Morbi condimentum est
                    nibh, et consectetur tortor feugiat at.</p>
            </article>
        </section>

        <footer>
            <p>&copy; 2014 Monday Times. All rights reserved.</p>
        </footer>
    </body>
</html>
```

HTML 5 semantic elements

New tags have a semantic meaning which is important for a good interpretation by all possible readers (not only browsers) and media where a web page is rendered.

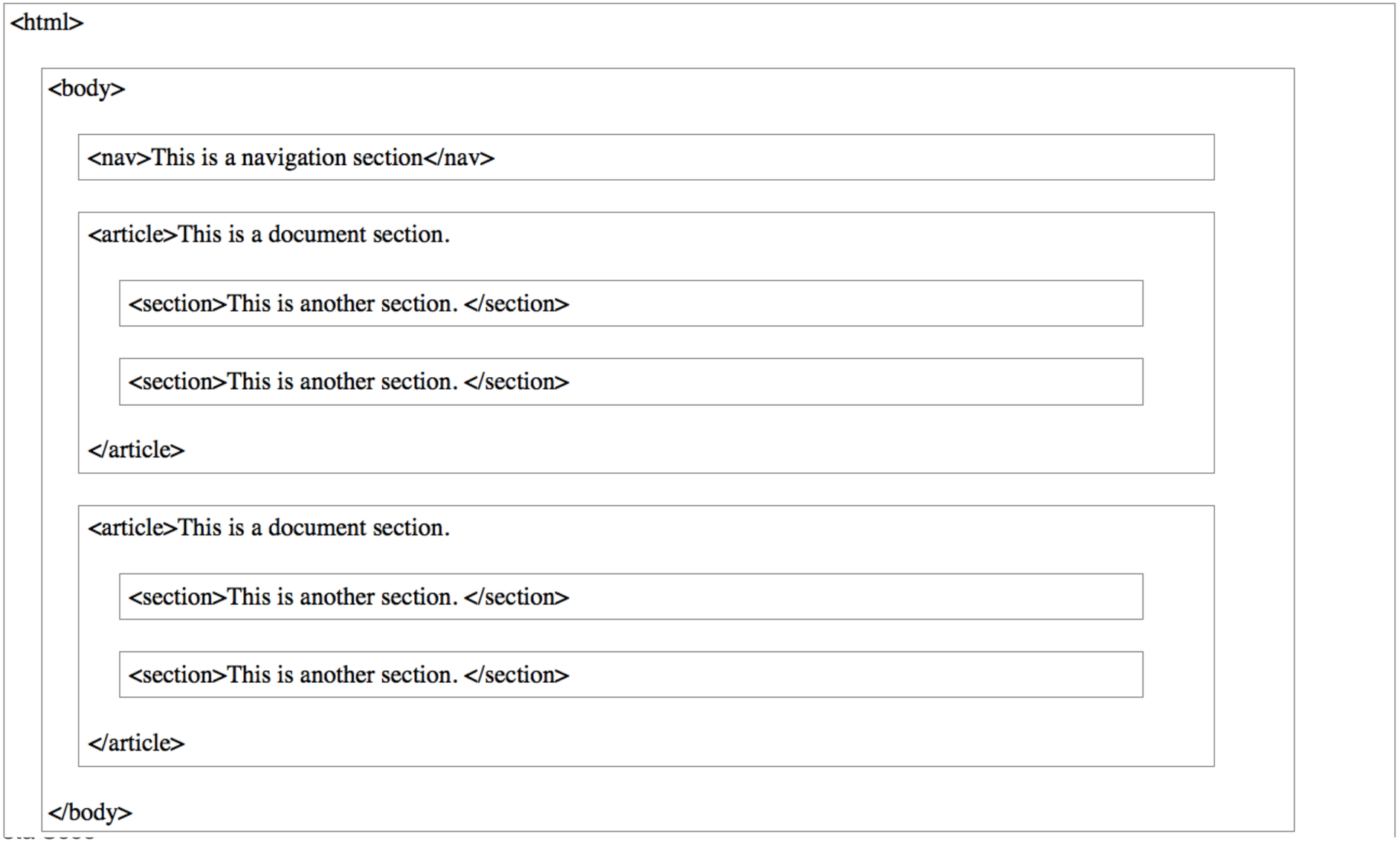
Styling and the use of frameworks and tools become a lot easier and more effective



HTML 5 semantic elements (part of)

Tag	Description
<article>	Defines an article in the document
<figcaption>	Defines a caption for a <figure> element
<figure>	Defines self-contained content, like illustrations, diagrams, photos, code listings, etc.
<footer>	Defines a footer for the document or a section
<header>	Defines a header for the document or a section
<main>	Defines the main content of a document
<menuitem>	Defines a command/menu item that the user can invoke from a popup menu
<nav>	Defines navigation links in the document
<section>	Defines a section in the document

Structural block elements (layout)



Textual block elements

```
<html>
    <body>
        <h1>This is a heading</h1>
        <p>This is a paragraph.</p>
        <p>This is another paragraph.</p>
    </body>
</html>
```

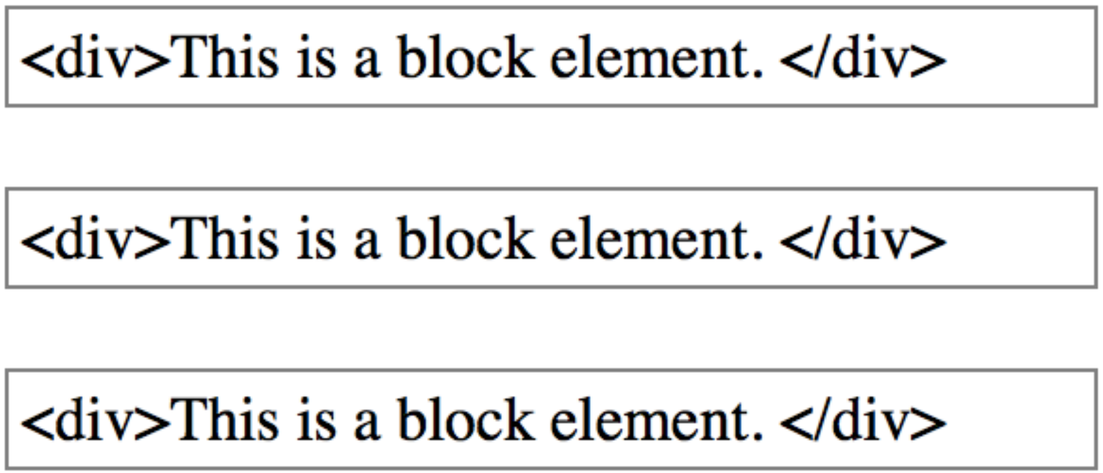
Textual inline elements

```
<html>

    <body>

        <h1>This is a heading</h1>

        <p>This is a paragraph. <span>This is a span</span> <b>This is a strong tag</b>
        <a>This is an anchor</a> </p>

        <p>This is another paragraph.</p>

    </body>

</html>
```

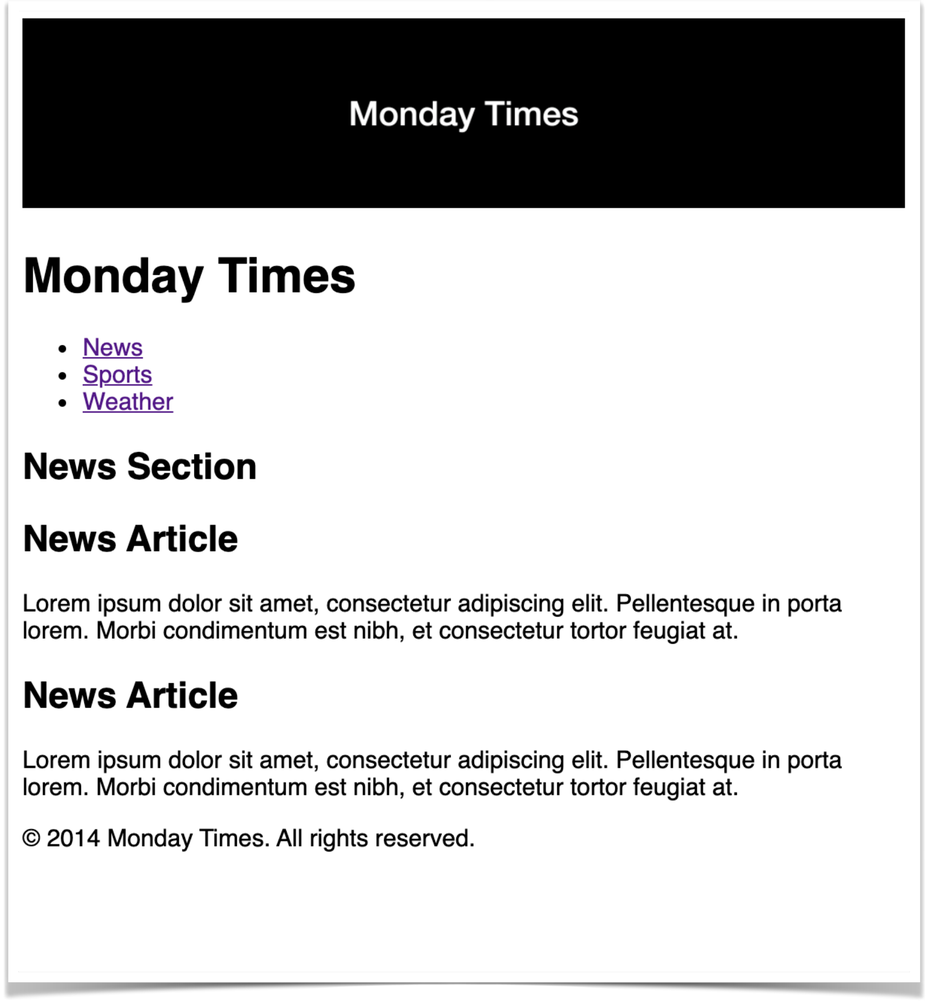
Inline elements

```
<span>This is </span> <span> an inline </span> <span> element.
</span> <span>This is </span> <span> an inline </span> <span>
element. </span> <span>This is </span> <span> an inline </span>
<span> element. </span>
```

Block elements



Attributes




```
<!DOCTYPE html>
<html>
    <head>…</head>
    <body>
        <header>
            <img src="mondaytimes.png" width="100%">
            <h1>Monday Times</h1>
        </header>

        <nav>
            <ul>
                <li><a href="#news">News</a></li>
                <li><a href="#sports">Sports</a></li>
                <li><a href="http://weather.com">Weather</a></li>
            </ul>
        </nav>

        <section>
            <h2 id="news">News Section</h2>
            <article>
                <h2>News Article</h2>
                <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit
                    nibh, et consectetur tortor feugiat at.</p>
            </article>
            <article>
                <h2>News Article</h2>
                <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit
                    nibh, et consectetur tortor feugiat at.</p>
            </article>
        </section>

        <footer>
            <p>&copy; 2014 Monday Times. All rights reserved.</p>
        </footer>
    </body>
</html>
```

Attributes

Important ones

`src` - links an element to a file, img or script

`id` - identifies a single element in a page. Manipulation and styling can be defined using identifiers.

`class` - identifies a group of elements in a page. Manipulation of elements and styling can be defined using class names.

`name` - used in forms to identify data sent to a server.

HTML in a 

Default Server interaction

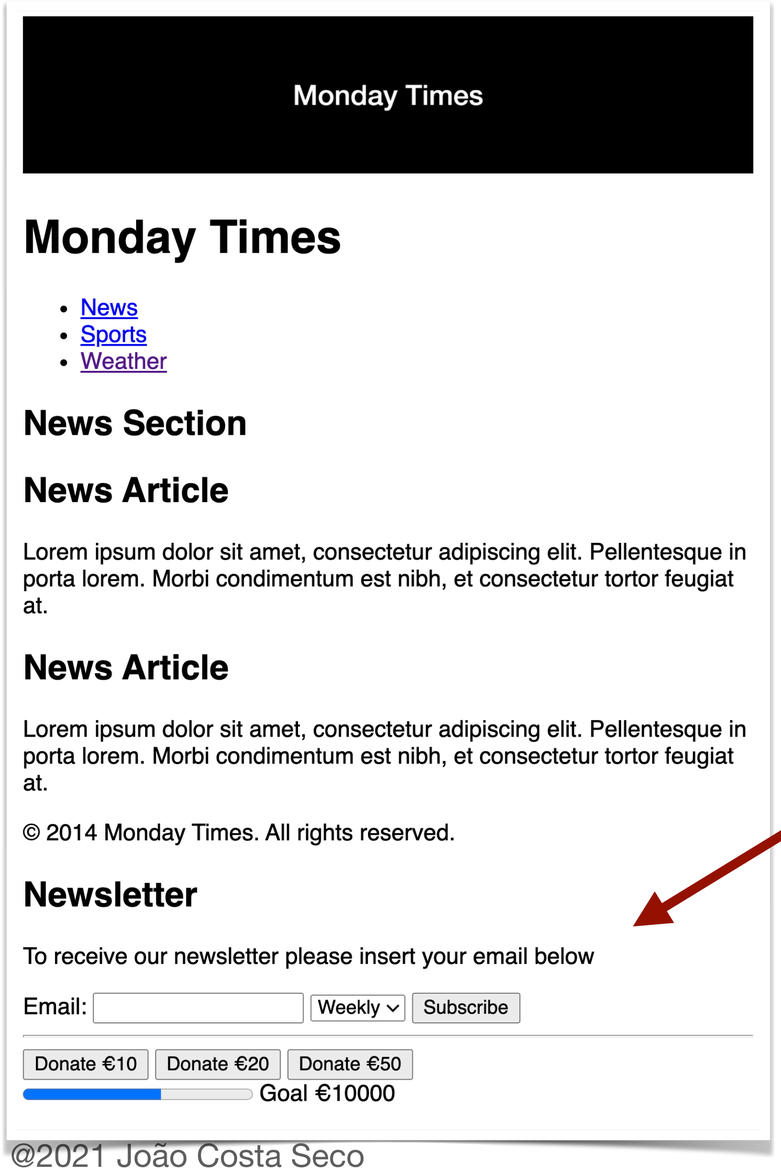
Links

- The browser performs a request (GET) to the given URL and replaces the entire content.

Forms

- Pre-defined controls (input, text areas, dropdown, buttons).
- The browser performs a request and replaces the entire content.
- method attribute defines the request type (GET/POST)
- called URL format depends on the used method (body/query string)

HTML - FORMS




```
<!DOCTYPE html>
<html>
    <head>…</head>

    <body>
        <header>
            <img src="mondaytimes.png" width="100%">
            <h1>Monday Times</h1>
        </header>
        …
        <footer>
            <p>&copy; 2014 Monday Times. All rights reserved.</p>

            <h2>Newsletter</h2>
            <p>To receive our newsletter please insert your email below</p>
            <form action="http://someservice.to" method="POST" _target="">
                <Label for="email">Email: </Label>
                <input type="text" id="email" name="email">
                <select name="period">
                    <option value="daily">Daily</option>
                    <option value="weekly" selected>Weekly</option>
                </select>
                <input type="submit" value="Subscribe">
            </form>

            <hr>
            <form>
            <div>
                <button>Donate €10</button>
                <button>Donate €20</button>
                <button>Donate €50</button>
            </div>
            </form>

            <progress value="6000" max="10000"></progress> Goal €10000
        </footer>
    </body>
</html>
```

w3schools.com

Break

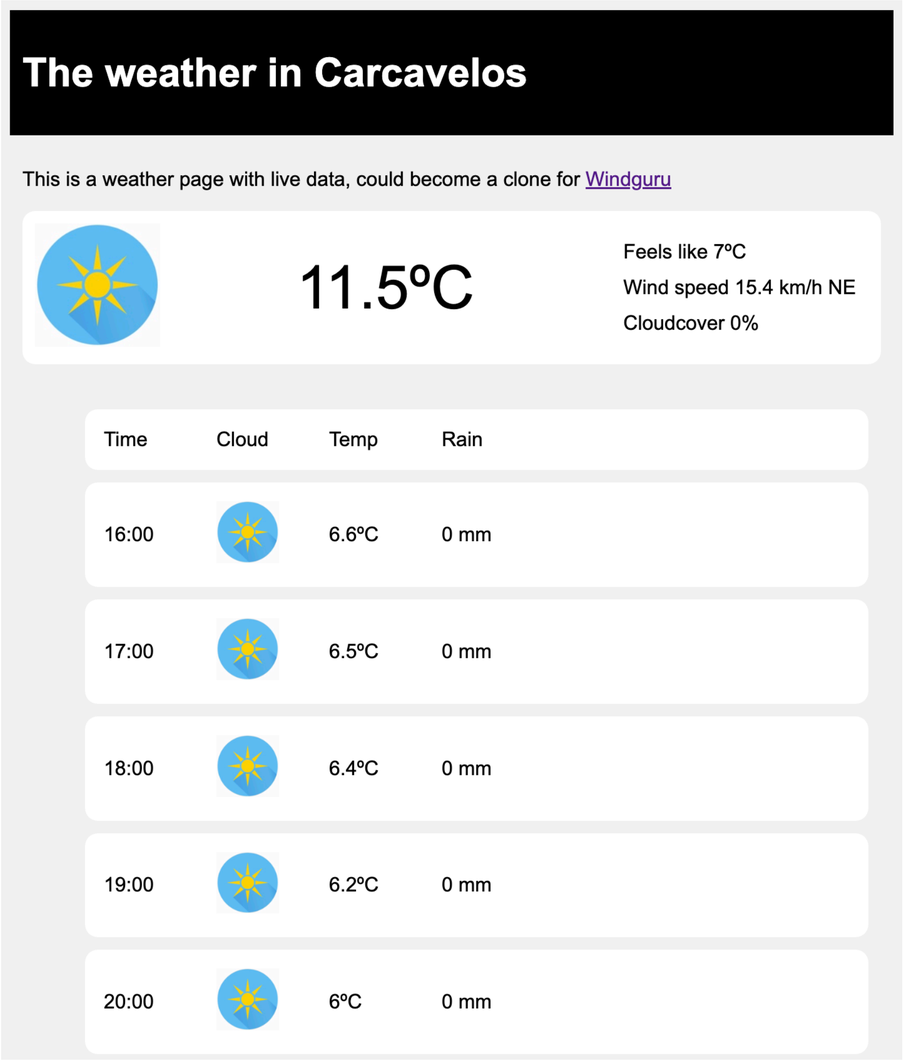
Demo: The First “Hello, World!” web-server

Demo: The First Web-client Application

Writing an HTML page from scratch to report the weather, and …

load it from the API

https://open-meteo.com/



The End