escola superior
de tecnologia e gestão
instituto politécnico
de leiria

Web Architecture

AInet 2018/19

Marco Monteiro

WWW or Web

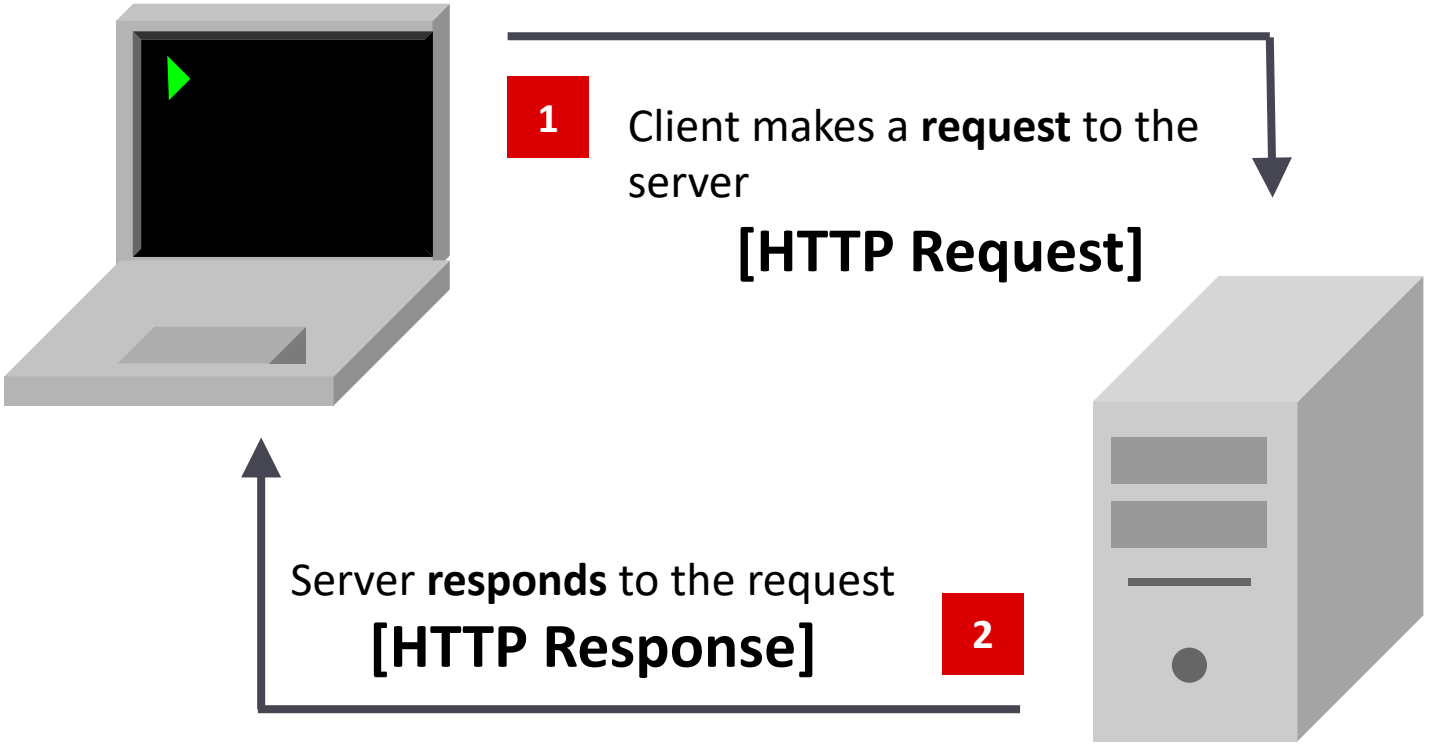
- WWW– World Wide Web, also know as "**Web**"
- Created by Tim Berners-Lee (at CERN)
  - 1989 - first proposal
  - 1990 - first implementation (within CERN only)
  - 1991 - first web servers outside CERN
- Definition
  - Philosophic (Tim Berners-Lee – web creator)
    - *"The World Wide Web is the universe of network-accessible information, an embodiment of human knowledge."*
  - Technical:
    - Set of resources and users of the Internet that use the HyperText Transfer Protocol (HTTP) or HTTPS

Web vs Internet

- Web ≠ Internet
- Web is part (uses) the Internet
- Internet includes the Web and <u>much more</u> (email, ftp, telnet, etc.)
- TCP/IP is the base protocol for the Internet
  - TCP/IP -> transport layer
  - HTTP runs over TCP/IP
  - HTTP -> application layer
  - Examples of other application layer protocols that run over TCP/IP:
    FTP, POP3, SMTP, SNMP, etc.

Web Architecture

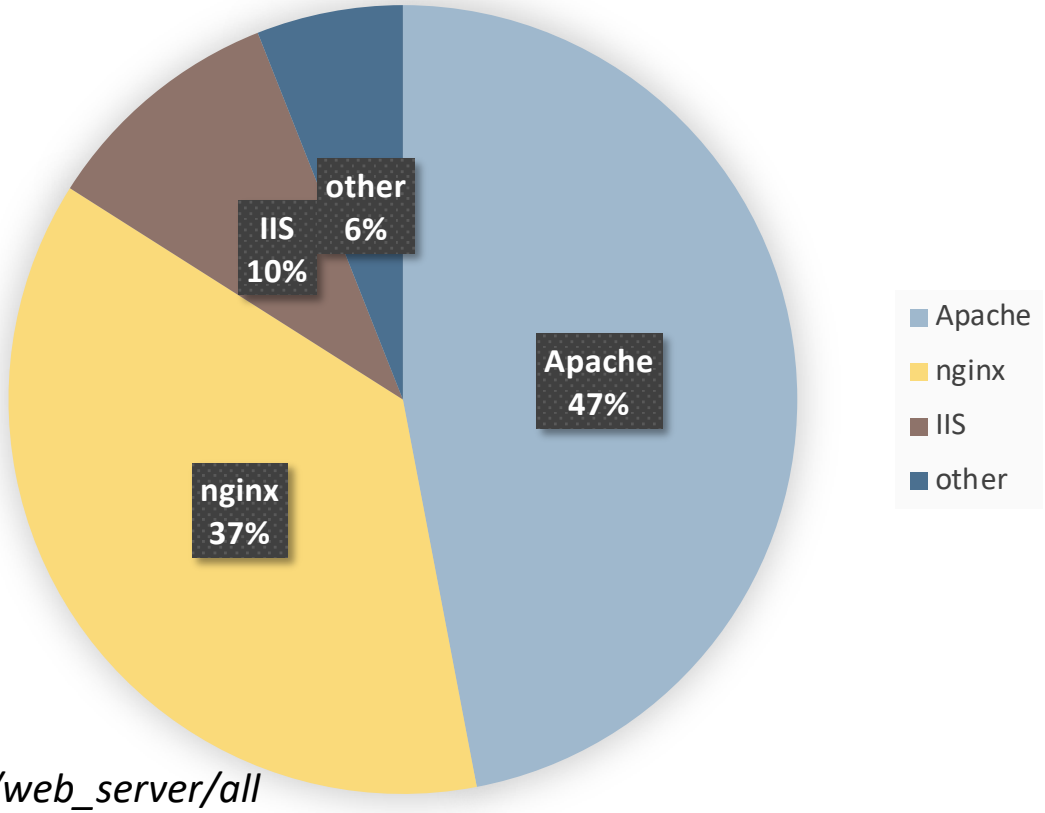
- Web uses a Client-Server architecture



Web Servers

- Example of Web Servers (aka HTTP Servers):
  - Apache; nginx;
    Microsoft Internet Information Services (IIS)

Web Server market share
(top 10 million websites)
Feb 2018

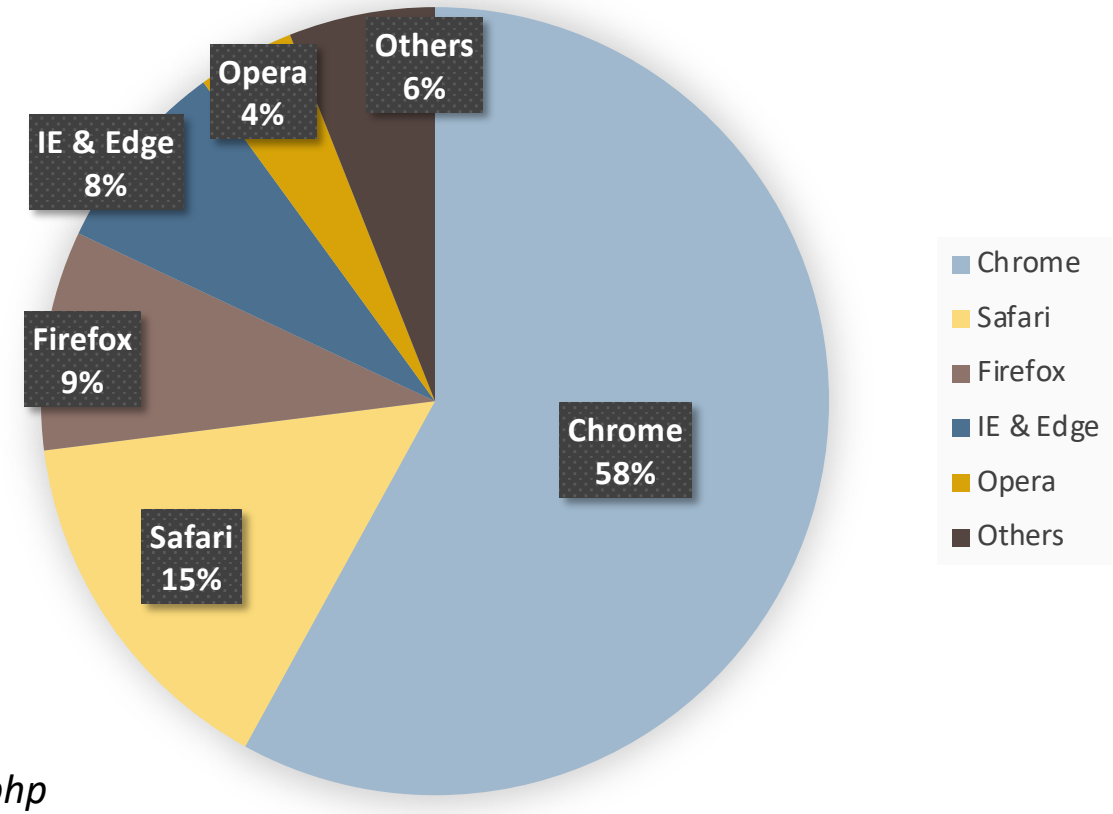


https://w3techs.com/technologies/overview/web_server/all

Web Browsers

- Example of Web Browsers (aka User Agents):
  - Chrome; Safari; Firefox;
    Edge; Internet Explorer; Opera

Web Browsers market share
Jan 2018



https://www.w3counter.com/globalstats.php

Client-Server Architecture

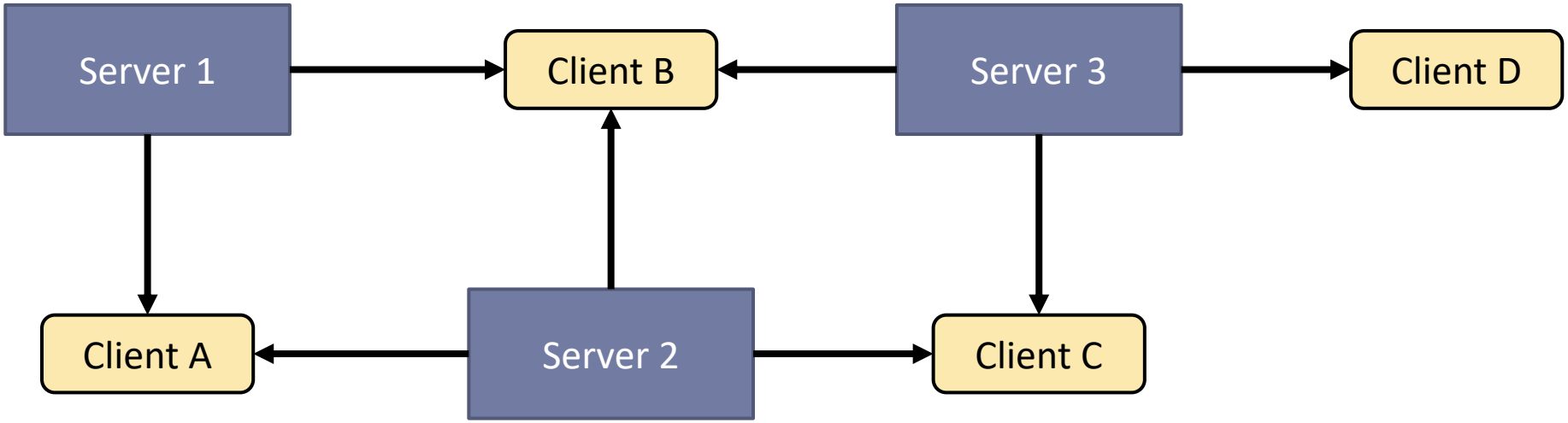
- Servers provides **resources** to clients
  - HTML documents
  - Images
  - CSS files
  - JavaScript files
  - Fonts
  - Data (XML; JSON; …)
  - . . .
- Clients (browsers) consume resources from servers, and display a representation of them (the resources) to the users

Layout Engines

- Web Browsers use **Layout Engines** (aka Rendering Engines) for rendering.
  - The Layout Engine accepts resources with content (HTML, XML, images, etc.) and formatting information (CSS, XSL, etc.) and displays the formatted content (a representation of the resources) on the screen
- Examples of Layout Engines:
  - Blink (Chrome / Opera)
  - Webkit (Safari)
  - Trident (IE)
  - EdgeHTML (Edge)
  - Gecko (Firefox)

N:M Relation

- There is a N:M relation between web clients and servers
- A server provides resources to multiple clients
- A client consumes resources from multiple servers



N:M Relation

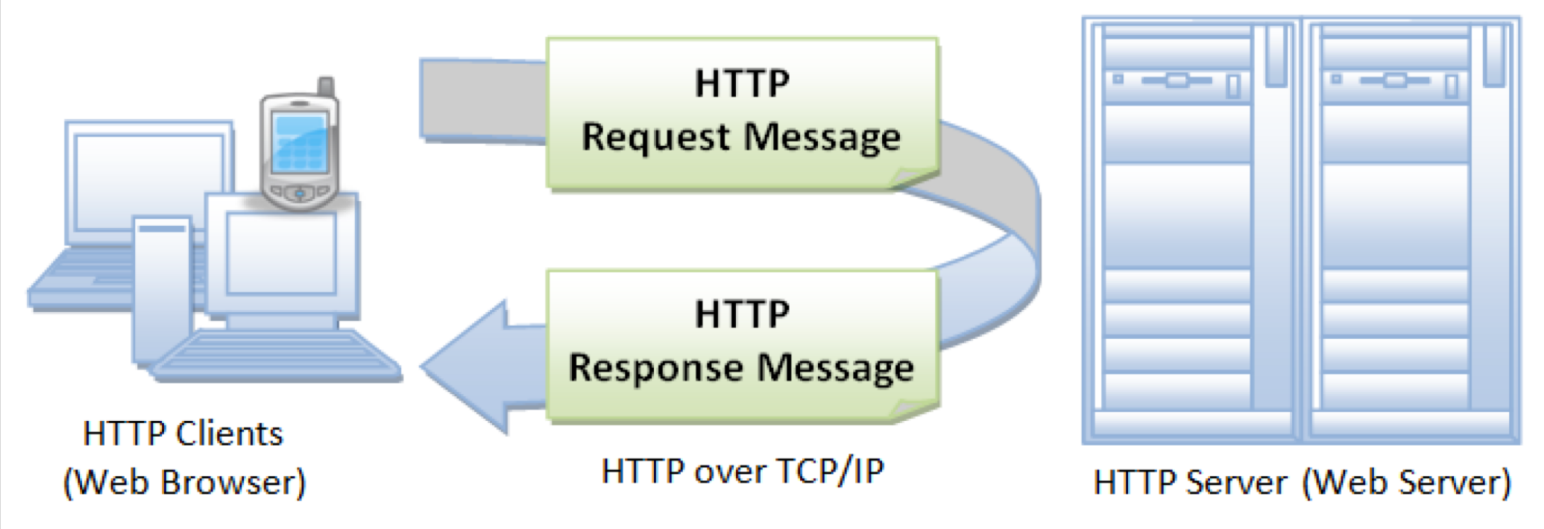
- One Web Page (what is shown to user) can use resources from multiple servers. Example:
  - HTML Document from 1 server
  - Images from multiple servers
  - CSS files from multiple servers
  - JavaScript files from multiple servers
- Web client (browser, or User Agent), load resources from multiple servers, processes them, and renders the content of the Web Page to the user.

Live Demo

- Analyze HTTP Requests and HTTP Responses with Google Chrome & Postman
  - http://<demourl>/simple.html
  - http://<demourl>/oneimage.html
  - http://<demourl>/img1.jpeg
  - http://<demourl>/multipleimages.html
  - http://<demourl>/multipleresources.html
  - http://<demourl>/styles.css
  - http://<demourl>/app.js
  - http://<demourl>/multipleresources2.html
  - http://somepage_from_somewhere

HTTP Protocol

- HTTP – Hypertext Transfer Protocol
- The HTTP, or secure alternative (HTTPS), is the message protocol responsible for all Web interaction
  - Client sends a HTTP request to the server
  - After the server receives the HTTP request, it will send a HTTP response

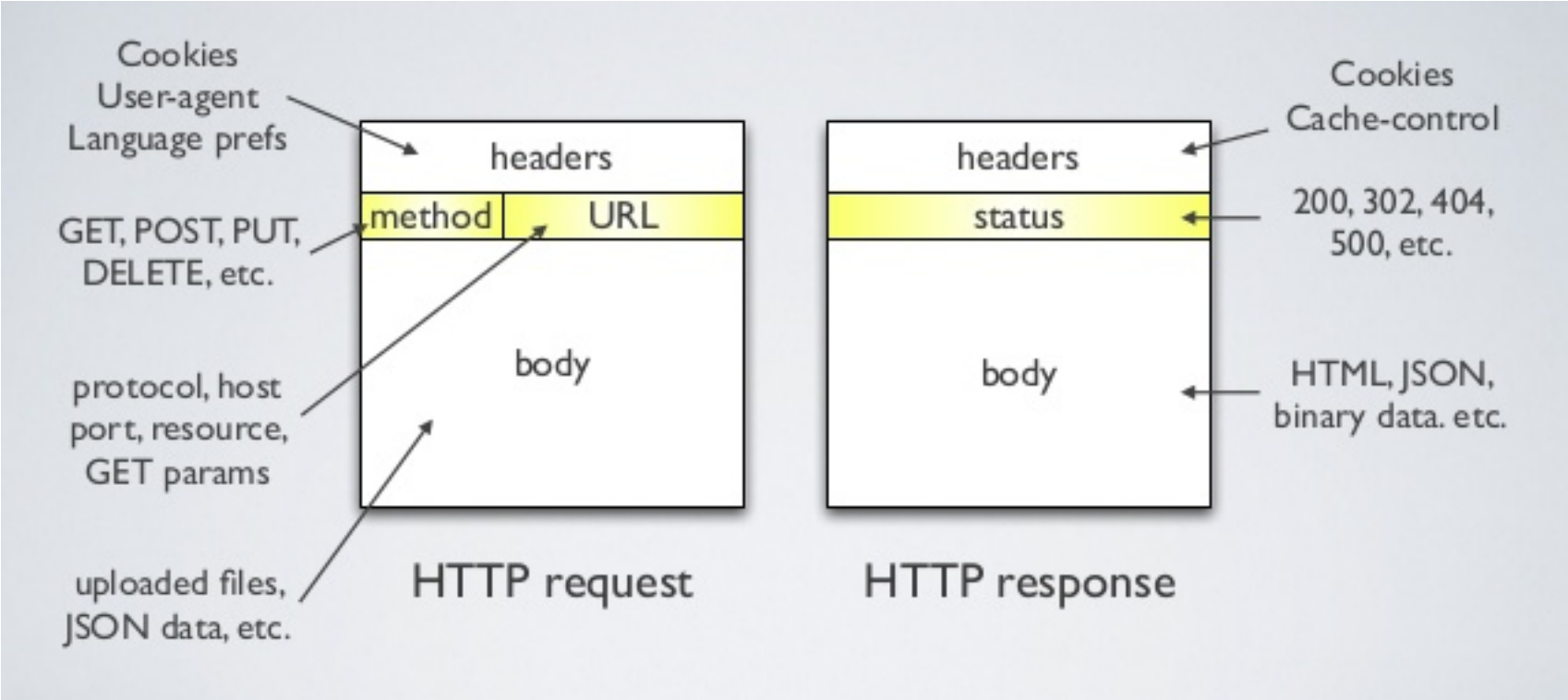


HTTP Protocol

- Works in pairs of Request/Response
- Independent of the resource type
  - Can transport HTML documents, images, CSS, Javascript, etc.
- Text based protocol
- **Stateless**
  - The server does not retain information or state about each user for the duration of multiple requests
  - Each Request/Response pair is independent – after sending a response to the client, the connection between the 2 (server and client) is terminated

HTTP Protocol

- HTTP Request and HTTP Response



HTTP Request

Method of HTTP request = **GET**

```
GET /doc/index.html HTTP/1.1
Host: www.meusite.com
User-Agent: Mozilla/5.0 (Windows;en-GB; rv:1.8.0.11) Gecko/20070312 Firefox/1.5.0.11
Accept: text/xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-gb,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://www.meusite.com/doc/pagina_no_browser_quando_foi_feito_pedido_HTTP.html
```

- Client is requesting the resource /doc/index.html, using GET method
- List of HTTP methods:

GET; POST; HEAD; PUT; DELETE; TRACE; OPTIONS; CONNECT; PATCH;

HTTP Response

- Status code examples:

  200 OK

  401 Unauthorized

  403 Forbidden

  404 Not Found

Status code

Content Type

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 1354

<!DOCTYPE html>
<head>
<title>Título da página</title>
. . .
```

Content

Live Demo

- Analyze HTTP Requests and HTTP Responses with Google Chrome & Postman

  (view Show Postman Console)

  - http://<demourl>/simple.html
  - http://<demourl>/oneimage.html
  - http://<demourl>/img1.jpeg
- Compare “file://” version of a web page, with a hosted version “http:// of the same page
  - file://simple.html
  - http://<demourl>/simple.html

MIME Types

- HTTP can transport multiple types of content
  - Each HTTP message transports only 1 type of content
- Content type is defined by the **MIME Type**

MIME Type

```
HTTP/1.1 200 OK
Content-Type:image/png
. . .
```

- MIME - Multi-purpose Internet Mail Extensions
- MIME type includes the type and subtype
  - image/png (type = image; subtype = png)

MIME Types

- The list of standard MIME Types is maintained by the IANA (**I**nternet **A**ssigned **N**umbers **A**uthority)
- https://www.iana.org/assignments/media-types/media-types.xhtml
- Some examples:

MIME Type	Content type
text/html	HTML document
text/plain	Simple text
text/css	CSS file
application/javascript	Javascript
image/png	PNG image
image/jpeg	JPEG image

MIME Type	Content type
video/mp4	MPEG-4 video
application/zip	ZIP file
application/pdf	PDF document
application/ms-word	Word document
multipart/form-data	For data codification of HTML forms (to upload files)

URI

- Web implements the concept of hypertext (text + links)
- **Uniform Resource Identifier (URI)** is a string used to identify a name or a resource on the web
- URIs can be classified:
  as locators (URLs), as names (URNs), or as both.
- **Uniform Resource Name (URN)** defines an item's identity

  `urn:oasis:names:specification:docbook:dtd:xml:4.1.2`

  `tel:+1-816-555-1212`
- **Uniform Resource Locator (URL)** provides a method for finding it

  `http://www.meusite.com/docs/index.html`
- An URL is an URI, but an URI may not be an URL

URL

- URL components

```
http://www.mysite.com:80/docs/index.html
```

Protocol Host(Server) Port Path_&_File

- **Protocol:**
  - On the Web it usually is http or https
  - Other protocols: ftp; mailto; news; telnet; gopher; ws; wss; file

- **Host (Server):**
  - Web server address
  - Usually a DNS (Domain Name System) domain name
  - May also be an IP address (ex: 149.145.201.034)

URL

- URL components

```
http://www.mysite.com:80/docs/index.html
```

Protocol | Host(Server) | Port | Path_&_File

- **Port:**
  - The port used to establish the connection
  - If the default port is used,
    URL does not require the port specification

Protocol	Default Port
HTTP	80
HTTPS	443

- **Path_&_File:**
  - Path and file name that indicates where the file is located “within” the Web Server
  - Relative to the site root (as defined on the Web Server) and not to the Server file system root

URL

- URL components: Query Strings

```
http://www.mysite.com/a.php?idcat=3&name=John+Doe
```

Query String

- **Query String:**
  - Represents an operation applied to the resource.
  - Usually, it is used to specify filter parameters on a method GET request used to obtain (read) data
  - Query string starts after the **?** character
  - Some special characters:

Query string parameters: idcat=3&name=John+Doe	
Name	Value
Idcat	3
name	John Doe

Character	Serve para:
?	Marks the query string beginning
&	Parameter separator
=	Separates parameter name and value **(name=value)**
+	Represents the space character

URL

- URL components: Fragments

```
http://www.mysite.com:80/docs/index.html#bookmark
```

Fragment Identifier (Bookmark)

- **Fragment** Identifier (bookmark)
  - Identifies a fragment (section) of the web page (aka Bookmark)
  - On the URL, fragment identifier appears after the **#** character
- How to define a fragment (bookmark) on a HTML document?
  - Identify an HTML element through the **id** atribute

```
<h1 id="NomeBookmark">
```

URL

- URL components: Fragments
  - Fragment identifiers are always **at the end** of the URL
  - Example

Query String

`http://www.meusite.com:80/a.php?id=3#bookmark`

Fragment Identifier (Bookmark)

Absolute URL vs Relative URL

- **Absolute** URL `http://www.meusite.com/docs/index.html`
  - Full address
  - Always starts by the protocol
  - Use for external resources (from another site)
- **Relative** URL
  - Address is relative to the current page location
  - Use for internal resources (located within the current site)
  - examples:

`../images/x.jpg`	.. previous page
`images/x.jpg`	images is a sub-folder
`/images/x.jpg`	Images is a folder on the root

Web Application

- What are the fundamental components of a typical Web Application?
- On the Server:
  - Web Server programming language.
    Examples: **PHP**, Java, C#, Python, Ruby
- On the client:
  - Content specification: **HTML**
  - Format and styling: **CSS**
  - Client programming language: **JavaScript**

Web Application

- Other components on the server:
  - Relational **database** server. Examples: MySQL; PostgreSQL, MsSQL, Oracle, MariaDB, etc.
  - Web server **Framework**. Examples:
    - PHP - Laravel; Zend Framework; Symfony; etc.
    - Java – Struts, JSF, Spring MVC | C# - ASP.Net MVC; Python – Django | Ruby – Ruby on Rails . . .
  - Other servers – Examples:
    - Non relational databases (NoSQL). Examples : MongoDB; Redis; CouchDB
    - E-Mail Server (SMTP; POP3)

Web Application

- Other components on the Client:
  - Responsive CSS Frameworks. Examples: **Bootstrap**; Bulma; Materialize; Foundation; Pure; Base, Ink
  - Javascript libraries/Frameworks. Examples:
    - jQuery
    - Client Frameworks: Vue.JS; AngularJS; React; Backbone.js; Ember.js; KnockoutJS; Famo.us
    - Mobile applications: React Native; PhoneGap; Ionic;
    - Others: Meteor (Isomorphic); CoffeScript (language); Construct 2 (games); **etc...**

HTML

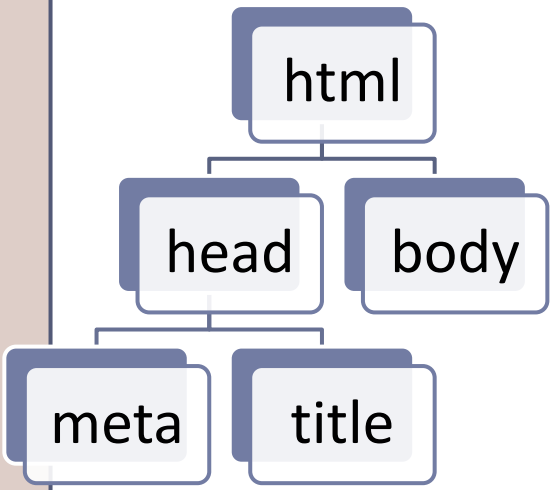
- HTML- Hyper Text Markup Language
  - Markup language that specifies Web Page content
  - Developed by Tim Berners Lee at CERN (Switzerland) in 1990
  - HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets
    - Example: <table>
  - Current version: HTML5
    - W3C specification at: http://www.w3.org/TR/html5

HTML

- Minimum HTML 5 structure

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Título</title>
  </head>
  <body>


  </body>
</html>
```



CSS

- CSS- Cascading Style Sheet
  - Language that specifies web page format (visual appearance and layout)
  - All page format should be done by using CSS. Advantages:
    - Separate the content from the appearance
    - More compact code
    - Better control over visual layout
    - Update the visual appearance of several documents simultaneously

CSS Rules

- Format is specified by a set of **CSS rules**
- Each rule has:
  - **Selector**
    - Selects which elements the rule is applied to
  - **Declaration**
    - Set of properties that define the visual appearance/layout of the selected elements
- Example:

Selector

```
p {
  color: blue;
  display: inline;
}
```

Declaration

CSS Selectors

- Example of basic CSS selectors
  - Element selector
    ```
    p { . . . }
    ```
  - ID selector
    ```
    #idx { . . . }
    ```
  - Class selector
    ```
    .clsx { . . . }
    ```
  - Descendent selector
    ```
    div p { . . . }
    ```
  - Direct descendent selector
    ```
    div > p { . . . }
    ```

This is a very small sample of CSS selectors. Further reading is fundamental to understand the basis of CSS selectors

Bibliography

- MDN Web Docs
  - https://developer.mozilla.org/docs/Web/HTTP
- W3C (Web standards)
  - http://www.w3.org/
- HTTP
  - http://www.tutorialspoint.com/http/index.htm
  - http://www.w3.org/Protocols/
- List of MIME Types
  - http://www.iana.org/assignments/media-types/index.html
- List of HTTP Status Codes
  - http://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml
- URL
  - http://www.w3.org/Addressing/
  - https://url.spec.whatwg.org
  - http://doepud.co.uk/blog/anatomy-of-a-url
  - http://en.wikipedia.org/wiki/Uniform_resource_locator