Client-Side Web Technologies

Databases and Web Applications Laboratory (LBAW)
Bachelor in Informatics Engineering and Computation (L.EIC)

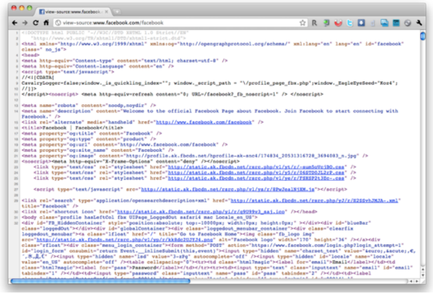
Sérgio Nunes
DEI, FEUP, U.Porto

The Big Picture

➔ Web browsers issue requests to web servers, which produce and return HTML documents for browsers to parse and display.

1. HTTP request

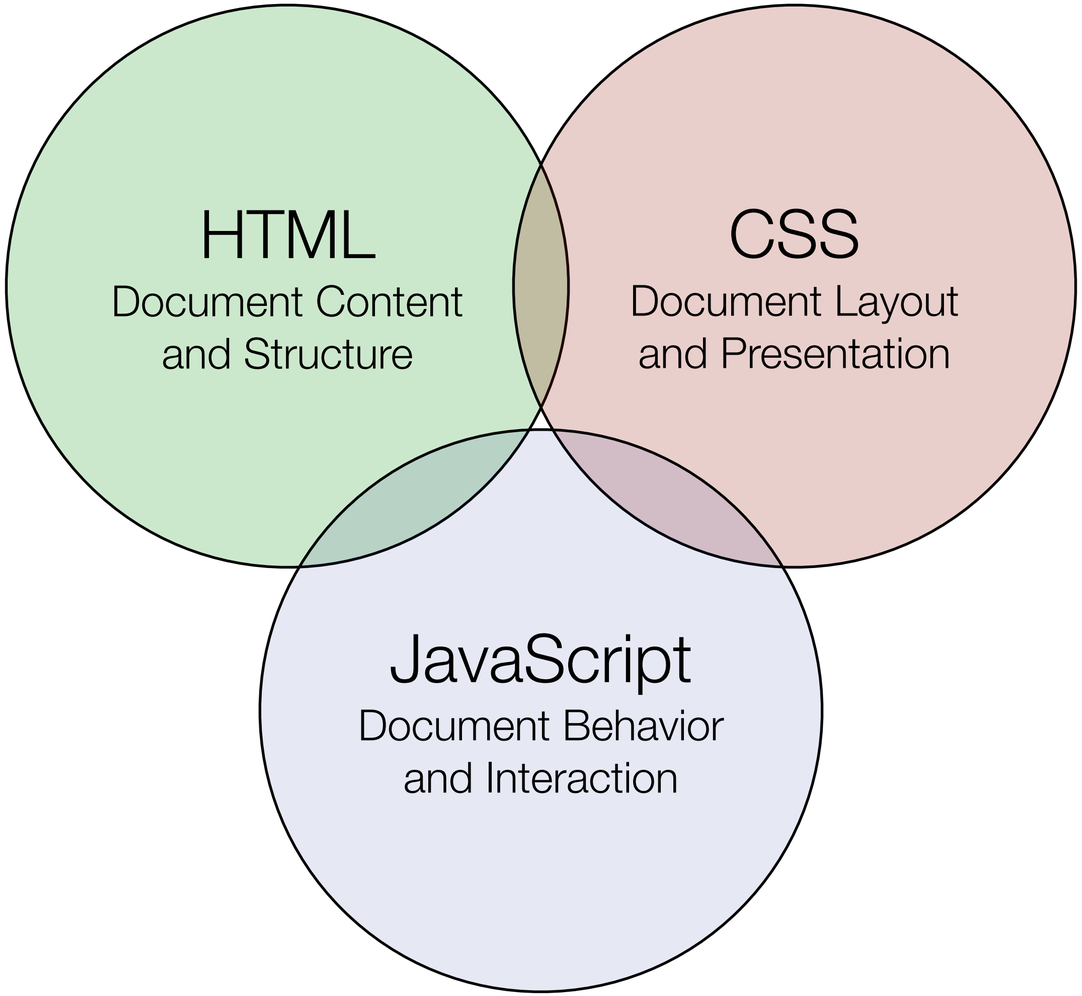
2. HTTP answer + HTML document



Client

Server

Client-side Web Technologies



HTML: HyperText Markup Language

HyperText Markup Language

➔ HTML is an acronym for HyperText Markup Language and is a format for providing linked structured information.

➔ HTML documents are simply text files containing marked-up text using tags.

➔ An HTML document is an hypertext node within an hypertext network.

Hypertext

➔ Concept defined by Ted Nelson in the 1950s.

➔ A way to organize text (and information) in a non-linear fashion.

➔ “Hypertext: Human-readable information linked together in an unconstrained way.”

➔ From the original WorldWideWeb: Proposal for a HyperText Project (1990)

  ➔ “HyperText is a way to link and access information of various kinds as a web of nodes in which the user can browse at will.

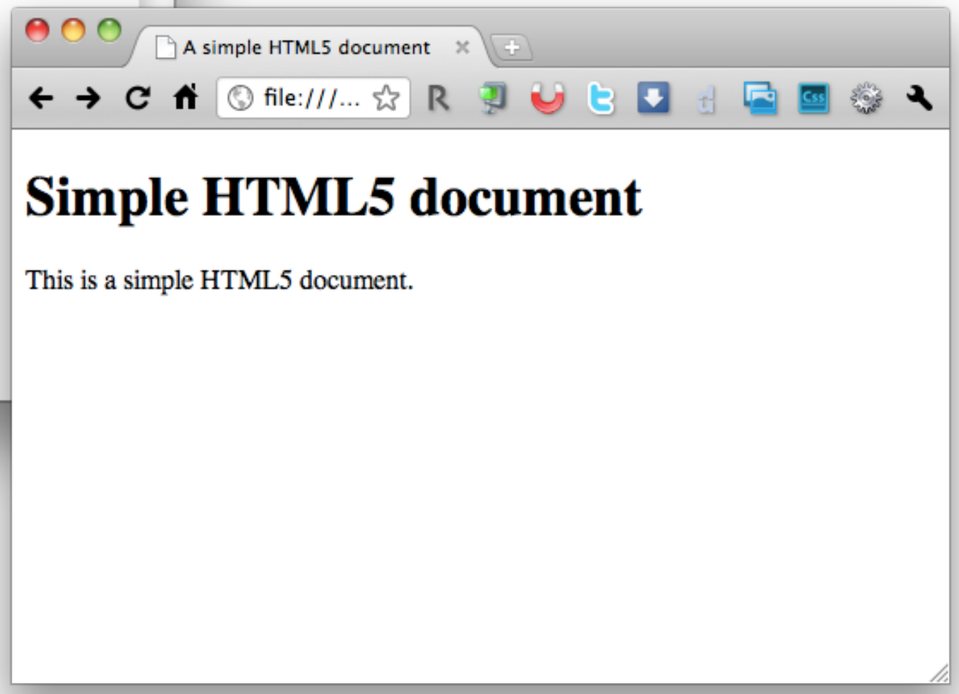
  It provides a single user-interface to large classes of information (reports, notes, data-bases, computer documentation and on-line help).”

Basic HTML Document

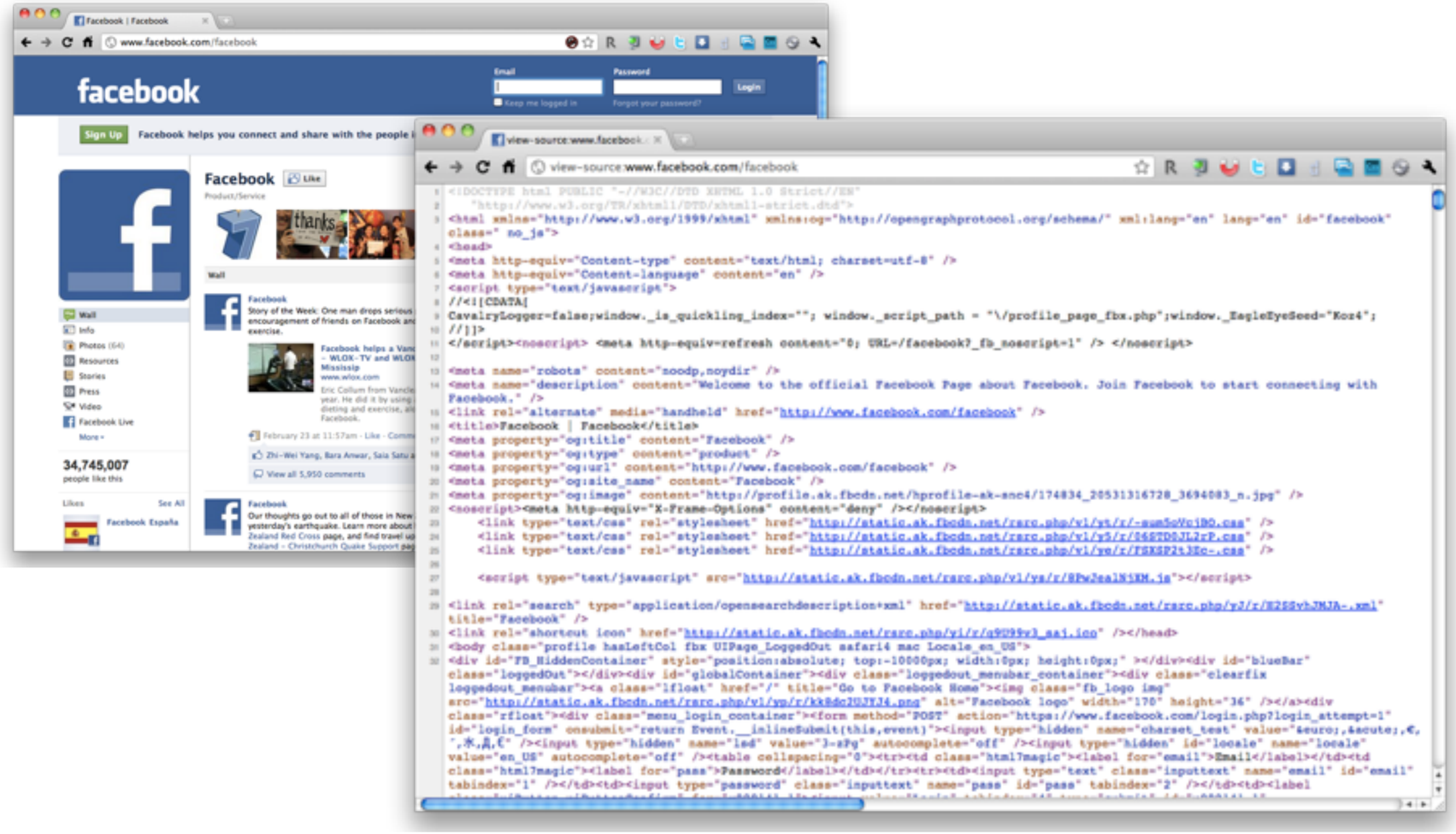
simple.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>A simple HTML5 document</title>
  </head>

  <body>
    <h1>Simple HTML5 document</h1>
    <p>This is a simple HTML5 document.</p>
  </body>
</html>
```



View Source


Facebook | Facebook
www.facebook.com/facebook
facebook
Email
Password
Login
Sign Up
Facebook helps you connect and share with the people
Facebook
Like
Product/Service
Wall
Info
Photos (64)
Resources
Stories
Press
Video
Facebook Live
More
34,745,007
people like this
Likes
See All
Facebook España
view-source:www.facebook
view-source:www.facebook.com/facebook
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:og="http://opengraphprotocol.org/schema/" xml:lang="en" lang="en" id="facebook"
class=" no_js">
<head>
<meta http-equiv="Content-type" content="text/html; charset=utf-8" />
<meta http-equiv="Content-language" content="en" />
<script type="text/javascript">
//<![CDATA[
CavalryLogger=false;window._is_quickling_index=""; window._script_path = "\/profile_page_fbx.php";window._EagleEyeSeed="Kox4";
//]]>
</script><noscript> <meta http-equiv=refresh content="0; URL=/facebook?_fb_noscript=1" /> </noscript>
<meta name="robots" content="noodp,noydir" />
<meta name="description" content="Welcome to the official Facebook Page about Facebook. Join Facebook to start connecting with
Facebook." />
<link rel="alternate" media="handheld" href="http://www.facebook.com/facebook" />
<title>Facebook | Facebook</title>
<meta property="og:title" content="Facebook" />
<meta property="og:type" content="product" />
<meta property="og:url" content="http://www.facebook.com/facebook" />
<meta property="og:site_name" content="Facebook" />
<meta property="og:image" content="http://profile.ak.fbcdn.net/hprofile-ak-snc4/174834_20531316728_3694083_n.jpg" />
<noscript><meta http-equiv="X-Frame-Options" content="deny" /></noscript>
<link type="text/css" rel="stylesheet" href="http://static.ak.fbcdn.net/rsrc.php/v1/y5/r/-wumSoVrjBO.css" />
<link type="text/css" rel="stylesheet" href="http://static.ak.fbcdn.net/rsrc.php/v1/y5/r/S6ETD6JL2rP.css" />
<link type="text/css" rel="stylesheet" href="http://static.ak.fbcdn.net/rsrc.php/v1/ye/r/FSKSP2tJEc-.css" />
<script type="text/javascript" src="http://static.ak.fbcdn.net/rsrc.php/v1/ys/r/8PwJealN5KM.js"></script>
<link rel="search" type="application/opensearchdescription+xml" href="http://static.ak.fbcdn.net/rsrc.php/y2/r/H2SSvhJMJA-.xml"
title="Facebook" />
<link rel="shortcut icon" href="http://static.ak.fbcdn.net/rsrc.php/yi/r/a9U99v3_aaj.ico" /></head>
<body class="profile hasLeftCol fbx UIPage_LoggedOut safari4 mac Locale_en_US">
<div id="blueBar"
class="loggedOut"> <div class="clearfix
loggedout_menubar"><img class="fb_logo img"
src="http://static.ak.fbcdn.net/rsrc.php/v1/yp/r/kk8dc2UJYJ4.png" alt="Facebook logo" width="170" height="36" /><div
class="rfloat"> <form method="POST" action="https://www.facebook.com/login.php?login_attempt=1"
id="login_form" onsubmit="return Event.__inlineSubmit(this,event)"><input type="hidden" name="charset_test" value="€,´,水,Д,Є" /><input type="hidden" name="lsd" value="]-zFg" autocomplete="off" /><input type="hidden" id="locale" name="locale"
value="en_US" autocomplete="off" /><table cellspacing="0"><tr><td class="html7magic"><label for="email">Email</label></td><td
class="html7magic"><label for="pass">Password</label></td></tr><tr><td><input type="text" class="inputtext" name="email" id="email"
tabindex="1" /></td><td><input type="password" class="inputtext" name="pass" id="pass" tabindex="2" /></td><td><label

A Brief History of HTML

Origins of HTML

➔ Created by Tim Berners-Lee and Robert Cailliau at CERN in the late 1980s.

➔ Main goal was to facilitate document sharing between researchers.

➔ CERN released it as royalty free in 1993.

➔ First official version published by IETF in 1993.

➔ World Wide Web Consortium (W3C) was created to define common standards for browsers and developers to adhere to.

HTML Proposal

- **Information Management: A Proposal**
  https://www.w3.org/History/1989/proposal.html
  - “This proposal concerns the management of general information about experiments at CERN.”
  - “It discusses the problems of loss of information about complex evolving systems and derives a solution based on a distributed hypertext system.”
  - Some practical requirements: remote access, heterogeneity, non-centralization, text-based, “live links”.
- Problems being addressed:
  - Information loss - “Often, the information has been recorded, it just cannot be found.”
  - Constantly changing information. Keeping a “book-like” organization of all information at CERN is impractical. Changes are distributed.
  - Tree-like organizations and keyword-based organization are also not feasible. Too strict and inflexible.

HTML Timeline

- ➔ During its first years (1990-1995), HTML revisions and extensions where first hosted at CERN and then IETF.
- ➔ Development was moved to the W3C after its creation in 1994.
- ➔ HTML development stopped in 1998 with the publication of HTML 4.
- ➔ W3C decided to migrate to an XML-based equivalent, named XHTML.
- ➔ XHTML was not widely adopted by web authors.
- ➔ HTML development continued outside W3C, with the WHATWG, whose work is now the basis for HTML5.
  - ➔ WHATWG - Web Hypertext Application Technology Working Group

The Early Days (1989 - 1993)

➔ From proposal (1989) to Mosaic release (1993).

➔ Web users were mostly from academia and research institutions.

➔ Few browsers, most of them text-based.

➔ HTML documents were simple and usually written by hand.

Growth Years (1994 - 2002)

➔ Wide adoption of the web to the dot.com bubble (1995-2000).

➔ Companies dispute the web browser market (aka “browser wars”).

➔ Browser development focused on new features, less on standards support.

➔ Wide differences between rendering engines.
Many web pages “designed for browser version x.x”.

➔ Extensive use of tables and sliced graphics to achieve “pixel perfect” layouts - “print-like design”. Resulted in ugly and complex HTML code.

Modern Era (2003 -)

➔ Wide adoption of modern web browsers.

➔ Separation of content and structure from layout and presentation.

➔ HTML controls content and structure.

➔ CSS controls layout and presentation.

➔ Clean and simple code (again!).

➔ CSS (2003), AJAX (2005), mobile (2007).

➔ A platform for (web) applications.

HTML

XHTML

➔ In 1998, the W3C decided to abandon HTML development and focus on a XML-based equivalent, named XHTML.

➔ XHTML 1.0 was completed in 2000.

➔ W3C then moved to XHTML 2.0, introducing several new features and less backward compatibility.

➔ Real world adoption of XHTML was small.

➔ In 2004, a proposal to refocus on HTML was discarded by the W3C, leading to outside development of HTML.

WHATWG

➔ Members of the W3C formed a new group: the Web Hypertext Application Technology Working Group (WHATWG).

➔ WHATWG didn't follow a consensus-based approach, so it was able to move much faster.

➔ In 2006, the W3C acknowledged that XHTML wasn't being adopted and work on HTML was resumed.

➔ Instead of starting from scratch, the W3C decided to use the work from WHATWG.

➔ Work on XHTML 2.0 ended in 2009.

W3C and WHATWG

➔ WHATWG continues working on HTML as a "living standard" (no versions).
https://html.spec.whatwg.org/

➔ Latest published W3C version of HTML is 5.2.
https://www.w3.org/TR/html52/

➔ Ongoing discussions on how to manage the work and collaboration between WHATWG and W3C, e.g. stop publishing two separate specifications.

➔ More details: https://wiki.whatwg.org/wiki/W3C

HTML5 Technologies

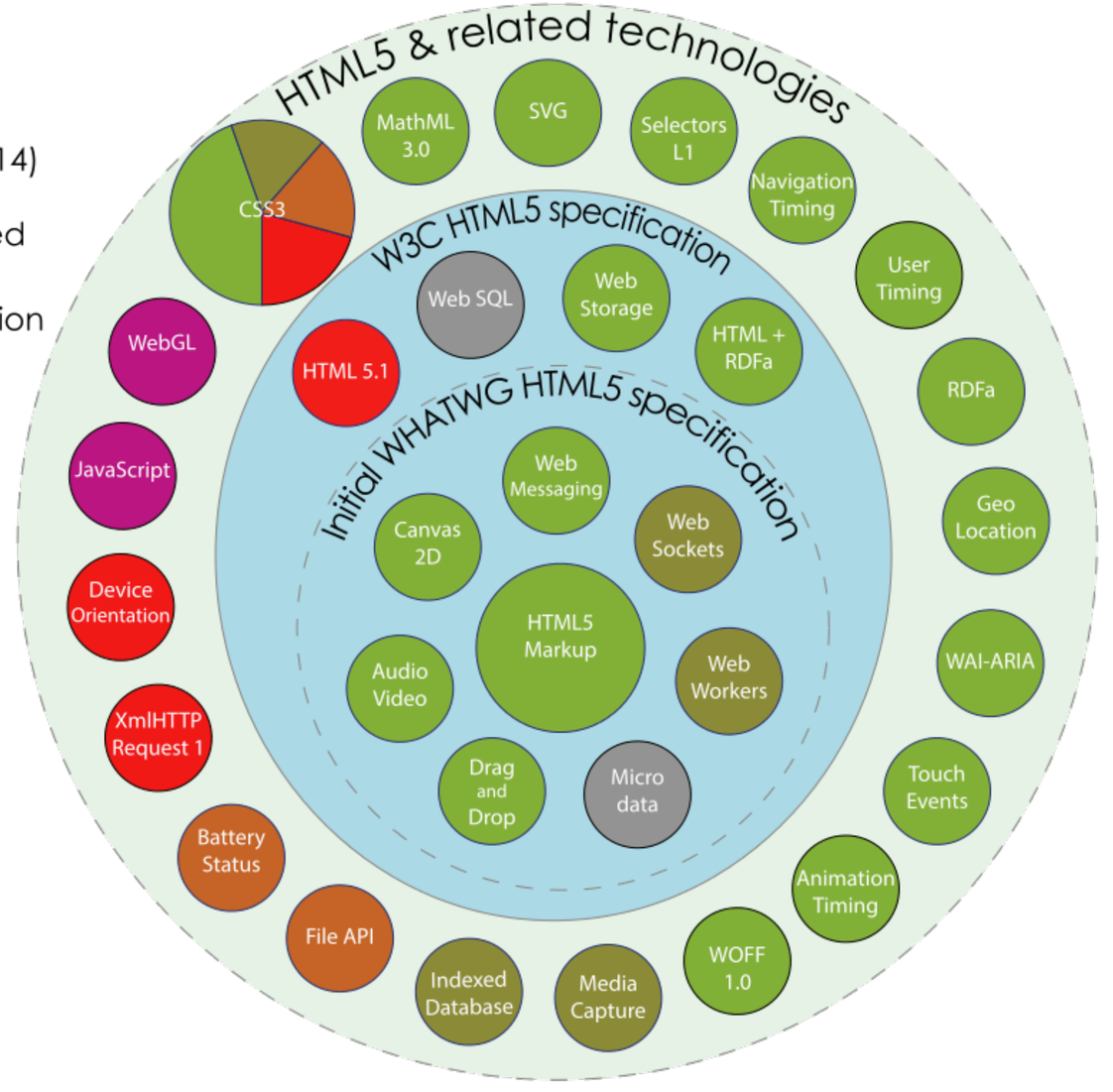
➔ HTML5 is a collection of features and technologies.

- ➔ Language / Markup features
- ➔ Document Model Definition (DOM)
- ➔ APIs for supporting JavaScript interaction with the DOM

HTML5

Taxonomy & Status (October 2014)

- Recommendation/Proposed
- Candidate Recommendation
- Last Call
- Working Draft
- Non-W3C Specifications
- Deprecated or inactive

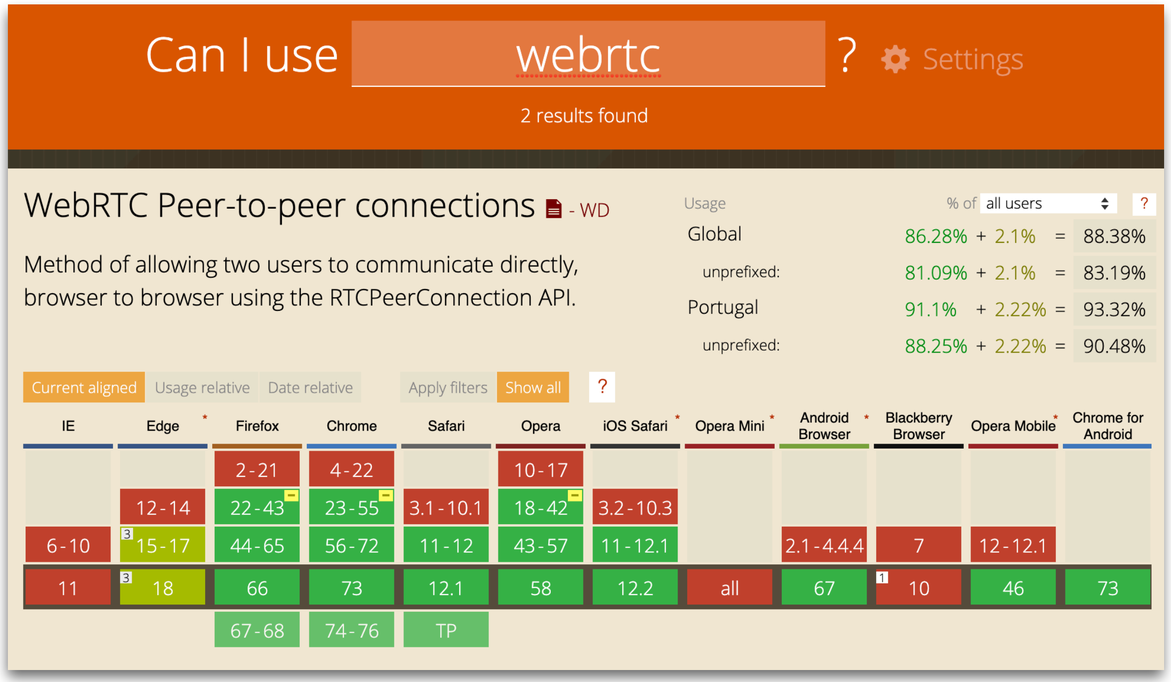


From: http://en.wikipedia.org/wiki/HTML5

Browser Support

➔ Support for these technologies has different levels of support in browsers.

➔ "Can I Use" provides up-to-date information about browser support of front-end technologies. https://caniuse.com



HTML Microdata

HTML Microdata

- ➔ Extension to define new attributes and embed simple machine-readable data in HTML documents.
- ➔ Goal: annotate content with machine-readable labels.
- ➔ Common use case: search engines can better 'understand' and index information that has been annotated using schema.org vocabulary.
- ➔ Microdata provides a mechanism to identify items and define their properties.
  - ➔ The `itemscope` attribute creates an item.
  - ➔ The `itemprop` attribute descends of `itemscope` and defines an item property.
  - ➔ With `itemtype` is possible to associate a vocabulary to an item.
  - ➔ An `itemid` can be used to define a global unique identifier for the item.

Microdata Example

➔ Defines an item with two properties.

```
<div itemscope>
 <p>Flavors in my favorite ice cream:</p>
 <ul>
  <li itemprop="flavor">Lemon sorbet</li>
  <li itemprop="flavor">Apricot sorbet</li>
 </ul>
</div>
```

Schema.org

➔ Vocabularies define concepts and relationships used to describe and represent areas of concern. Can be very simple (one or two concepts) or very complex (thousands of terms).

➔ A shared vocabulary makes it possible to have a common understanding of defined concepts and relationships.

➔ Schema.org is a collaborative, community driven initiative to create, maintain, and promote the use of schemas for structured data on the web. Founded by Google, Microsoft, Yahoo, and Yandex.

➔ Schema.org defines more than 600 types and >900 properties. Such as CreativeWork, Book, Movie, Event, Organization, Person, Place, Restaurant, etc.

Microdata Example using Vocabulary

➔ Example using Schema.org vocabulary.

➔ Defines an item of the type LocalBusiness, as defined by the Schema.org vocabulary, containing three properties, one of which is a item of the type PostalAddress, containing four properties.

```
<div itemscope itemtype="http://schema.org/LocalBusiness">
  <h1 itemprop="name">Beachwalk Beachwear & Giftware</h1>
  <span itemprop="description"> A superb collection [...].</span>
  <div itemprop="address" itemscope itemtype="http://schema.org/PostalAddress">
    <span itemprop="streetAddress">3102 Highway 98</span>
    <span itemprop="addressLocality">Mexico Beach</span>,
    <span itemprop="addressRegion">FL</span>
  </div>
  Phone: <span itemprop="telephone">850-648-4200</span>
</div>
```

HTML Microdata References

➔ W3C Editor’s Draft - Microdata (April 2021)
https://w3c.github.io/microdata/

➔ HTML Standard Microdata Specification
https://html.spec.whatwg.org/#microdata

➔ Schema.org
https://schema.org/

➔ Semantic Web (aka Web of Data)
https://www.w3.org/standards/semanticweb/

Web APIs

Web APIs

- In addition to the language specification, HTML5 introduced several Web APIs that can be used with JavaScript. There is a large number of APIs in different stages of development.
  - Documents manipulation APIs (e.g. DOM, Drag and Drop)
  - Fetch remote data APIs (e.g. Fetch, Web Sockets)
  - Drawing and graphics manipulation APIs (e.g. Canvas, WebGL)
  - Audio and Video APIs (e.g. Web Audio, WebRTC)
  - Device APIs (e.g. Notification, Vibration, Fullscreen)
  - Client-side storage APIs (e.g. Web Storage, IndexedDB)

Geolocation API

Geolocation API

➔ The Geolocation API provides scripted access to geographical location information associated with the device.

➔ Common sources of location information include Global Positioning System (GPS) and location inferred from network signals such as IP address, RFID, WiFi and Bluetooth MAC addresses, and GSM/CDMA cell IDs, as well as user input.

➔ Available both as single-shot request or continuous tracking.

- `navigator.geolocation.getCurrentPosition( callback )`
- `navigator.geolocation.watchPosition( callback )`

➔ Geolocation API Specification
https://www.w3.org/TR/geolocation-API/

Web Storage API

Web Storage API

➔ Local storage is an important feature for web applications.

➔ Cookies can be used for persistent local storage but are limited in size and are included in every HTTP request, slowing down the communication and exposing data.

➔ The Web Storage API specifies a mechanism to persistently store data in web clients, as key-value pairs. Unlike cookies, this data is never shared with the server and can only be accessed by the client.

➔ Data can be kept during page sessions, using `sessionStorage`, or persisted even when the browser is closed, using `localStorage`.

➔ Web Storage API Specification
https://www.w3.org/TR/webstorage/

Web Storage API

➔ Data can be stored and retrieved using keys.

- `localStorage.setItem("key", data)`
- `localStorage.getItem("key")`

➔ It is possible to keep track of changes trapping the `storage` event.

➔ For structured data, the IndexedDB API can be used. This API specified a low-level API for storing and indexing large volumes of data in the client.

➔ Indexed Database API 3.0, W3C Working Draft (March 2021)
https://www.w3.org/TR/IndexedDB/

Web Sockets API

Web Sockets API

➔ Web applications are not restricted to request-response interaction.

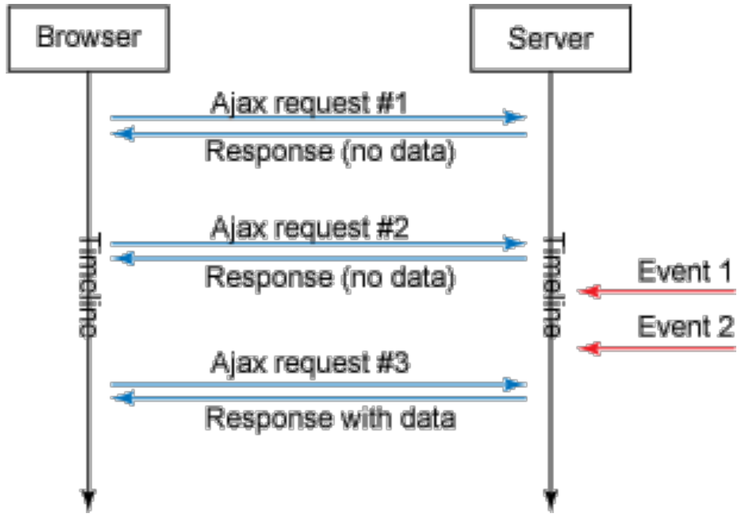
➔ A particularly important use case is the need for server initiated communication (aka "server push").

➔ Common scenarios include notifications on long running tasks, chat systems, multi-user collaboration systems (e.g. live collaborative text editors).

➔ How to push information from the server to the client?

Polling

➔ Make periodic requests to the server to check for new data.



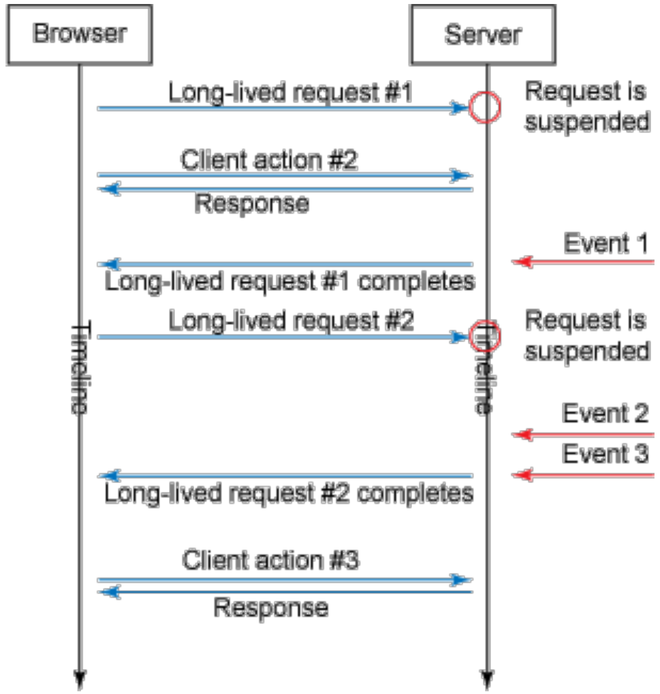
➔ The smaller the interval between request the more up to date the data is.

➔ Drawbacks: resource and bandwidth consumption even when no new data is available. Does not scale well and doesn't guarantees low-latency.

Image source: http://www.ibm.com/developerworks/web/library/wa-reverseajax1/

Comet

➔ Requests are initiated by clients and kept alive for long periods, until a timeout occurs or a response is sent.



➔ On the server, the request is suspended or paused until a response is ready.

Image source: http://www.ibm.com/developerworks/web/library/wa-reverseajax1/

Web Sockets

➔ Web Sockets enables bidirectional communications between the web browser and the web server. No polling is needed to get messages from the server.

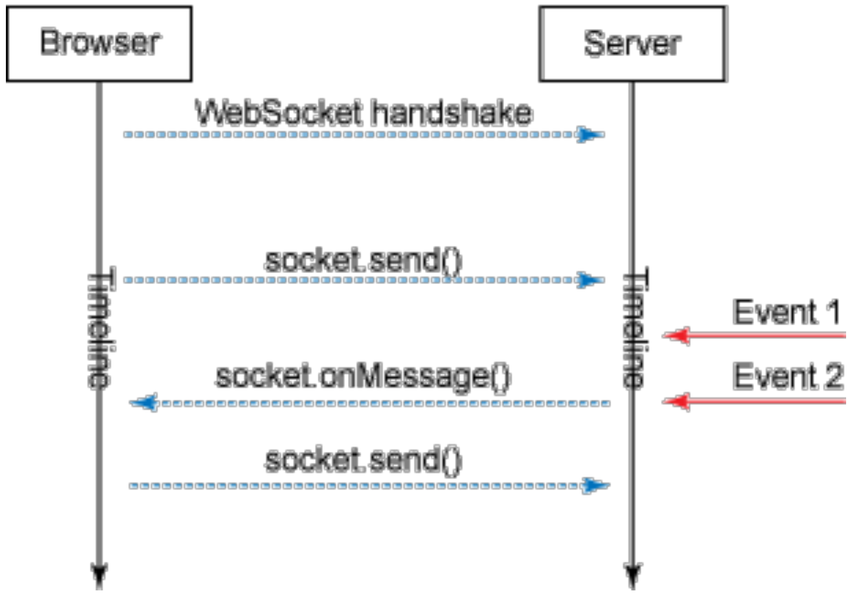


Image source: http://www.ibm.com/developerworks/web/library/wa-reverseajax1/

Web Socket Example

```
// Create WebSocket connection.
const socket = new WebSocket('ws://localhost:8080');

// Connection opened
socket.onopen = function (event) {
    socket.send('Hello Server!');
});

// Listen for messages
socket.onmessage = function (event) {
    console.log('Message from server ', event.data);
};
```

Adapted from: https://developer.mozilla.org/en-US/docs/Web/API/WebSocket

Web Sockets References

- The CometD Reference Book
  https://docs.cometd.org/current/reference/

- The WebSocket API | MDN web docs
  https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API

- The WebSocket API | W3C
  https://www.w3.org/TR/websockets/

WebRTC API

WebRTC API

➔ WebRTC (Web Real-Time Communications) is a technology which enables communication between browsers without requiring an intermediary.

➔ It includes the building blocks for high-quality communications on the web, such as network, audio and video components used in voice and video chat.

➔ Example file sharing P2P web application: https://www.sharedrop.io/

➔ More: https://webrtc.github.io/samples/

➔ WebRTC Home
https://webrtc.org/

➔ WebRTC API Specification
https://www.w3.org/TR/webrtc/

Web Workers API

Web Workers API

- ➔ Web Workers provide support for background execution of scripts.
- ➔ JavaScript execution is single-threaded. Web Workers are designed to bring concurrency to web applications through the execution of scripts in background threads, independently of any user interface scripts.
- ➔ Example use cases:
  - ➔ Perform background computationally expensive task.
  - ➔ Periodically prefetch data.
  - ➔ Share state between multiple clients using a shared worker.
  - ➔ Split computationally expensive tasks between clients.

Web Workers API

➔ Generally, workers are expected to be long-lived, have a high start-up performance cost, and a high per-instance memory cost.

➔ There are two kinds of workers: dedicated workers, which are used by a single script, and shared workers, that can be used by multiple scripts.

➔ Data is shared between the main thread and workers using messages.

➔ HTML Standard — Web workers (April 2021)
https://html.spec.whatwg.org/multipage/workers.html

Web Workers Example

```
<p>The highest prime number discovered so far is: <output id="out"></output></p>
<script>
  var worker = new Worker('worker.js');
  worker.onmessage = function (event) {
    document.getElementById('out').textContent = event.data;
  };
</script>
```

```
var n = 1;
search: while (true) {
  n += 1;
  for (var i = 2; i <= Math.sqrt(n); i += 1)
    if (n % i == 0)
     continue search;
  // found a prime!
  postMessage(n);
}
```

worker.js

Progressive Web Applications

➔ Progressive Web Applications (or PWAs) represent a new type of web applications, that combine multiple technologies and design patterns to improve user experience.

➔ Characteristics of progressive web apps: discoverable, installable, linkable, network independent, progressive, responsive, safe.

➔ Key technology: web workers, which intercept page requests and can use the local storage to provide an answer or make server requests.

➔ Other relevant technologies: web app manifest, web storage, notifications, etc.

➔ Progressive Web Apps
https://developers.google.com/web/progressive-web-apps/

Progressive Web Apps

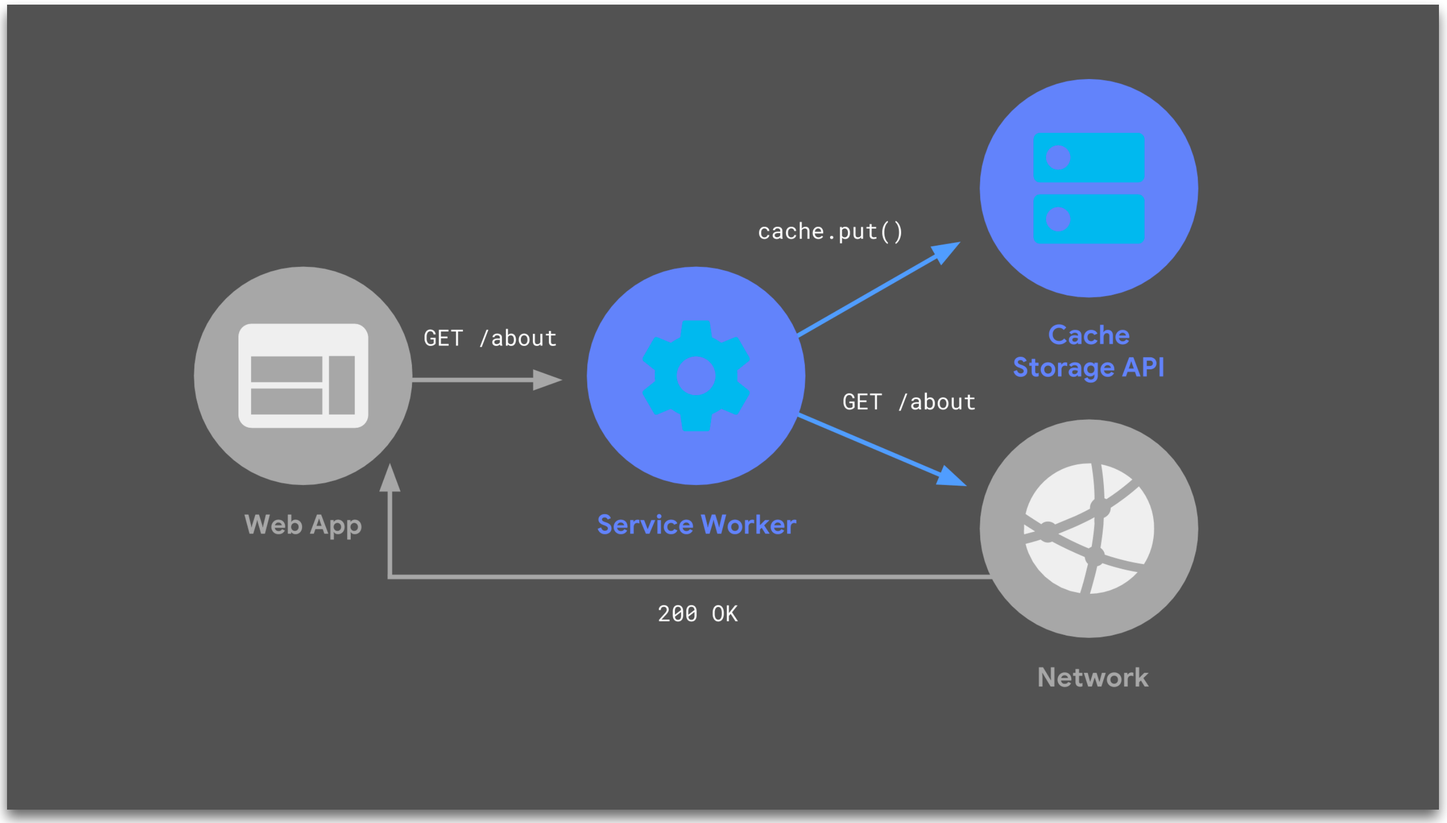


Image from Beyond SPAs: alternative architectures for your PWA (2018)
https://developers.google.com/web/updates/2018/05/beyond-spa

HTML References

➔ **HTML: HyperText Markup Language | MDN**
https://developer.mozilla.org/en-US/docs/Web/HTML

➔ **Latest version of HTML**
https://www.w3.org/TR/html/

➔ **WHATWG HTML Specification**
https://html.spec.whatwg.org/multipage/

➔ **Dive Into HTML5**
https://diveintohtml5.info/

➔ **HTML Dog: HTML, CSS and JavaScript tutorials**
https://htmldog.com/

➔ **Chapter 2 - A history of HTML**
https://www.w3.org/People/Raggett/book4/ch02.html