Exploration — Introduction to the Web and Web Applications

Introduction

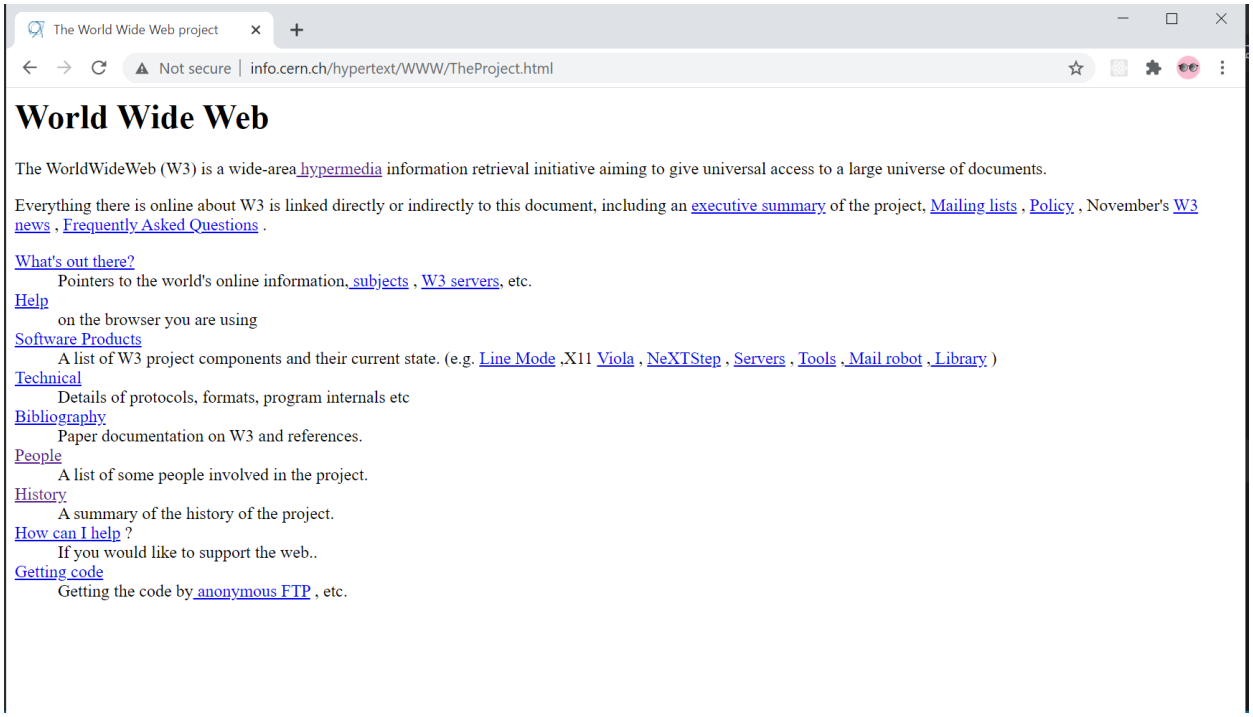
In this course, we will explore concepts, techniques, and technologies needed to build modern web applications and sites. Technologies for the frontend of a web app include URLs, HTML, CSS, JavaScript, and REACT. Technologies for the backend include HTTP, JavaScript, Express, Mongoose, REST, MongoDB. We will also talk about Browsers, Clients, Servers, Web Servers, etc. In this first exploration, we will take a high level look at some of the most important web technologies and see how these fit together in developing web applications. For now, don't worry if the high-level description of some of the technologies seems vague. Anyone first encountering web development comes across a veritable alphabet soup of acronyms. Let's get familiar with the ingredients of this soup!

The Start of the Web

The World Wide Web, or W3, or WWW, or simply the web, started out as **a project to build (http://info.cern.ch/hypertext/WWW/TheProject.html)** "a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents."

The project introduced 3 core technologies for building such an information retrieval system:

1. **Hypertext Transfer Protocol** or **HTTP**: A protocol that describes how to exchange messages. Specifically, it is a protocol that describes the details of how:
   - A program called the client can send a request (e.g., to retrieve a document) to
   - Another program called the server which sends back a response (e.g., containing the requested document) back to the client.
2. **Uniform Resource Locator** or **URL**: A naming infrastructure to represent documents, or more formally resources, located on the web. We commonly call a URL a web address.
   - To request a document, a client uses the document's URL to send an HTTP request to that server that hosts the requested document.
   - Typically the client and server programs run on different machines. In such cases, the URL identifies both the machine on which the server is running as well as the specific document among all the other documents hosted by that server. The URL naming infrastructure thus allows retrieval of documents that are located across geographically distant places, i.e., across a wide area.
3. **Hypertext Markup Language** or **HTML**: A markup language for describing documents that can be retrieved over the web.
   - The term **hypertext** means text which contains links to other texts.
   - HTML supports describing **hypermedia** documents, i.e., documents that besides text, can also contain other media such as graphics, video, and sound.
   - A client interprets the HTML in a document it receives in the HTTP response from the server and displays the document.

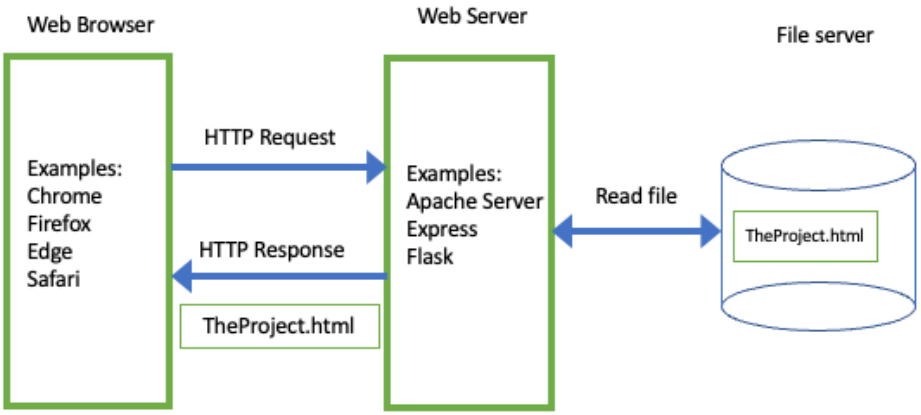


The website for the World Wide Web Project is one of the earliest webpages ever created. The URL is **http://info.cern.ch/hypertext/WWW/TheProject.html (http://info.cern.ch/hypertext/WWW/TheProject.html)** . The website is hosted at CERN, a

European research center in Switzerland. The World Wide Web Project was started by Tim Berners-Lee at CERN in the late 1980's.

A typical scenario for retrieving a document on the web will be as follows:

- The client is a web browser.
- A user enters a URL in the browser.
- The browser sends an HTTP request to the appropriate web server as identified by the URL.
- The server receives the HTTP request and determines the document that has been requested.
- The server sends back the document in an HTTP response to the browser.
- The browser interprets the HTML document and displays its contents.



A web browser sends an HTTP request to the server for the resource identified by the URL. The server responds by sending the HTML document in the HTTP response.

The Evolution of the Web

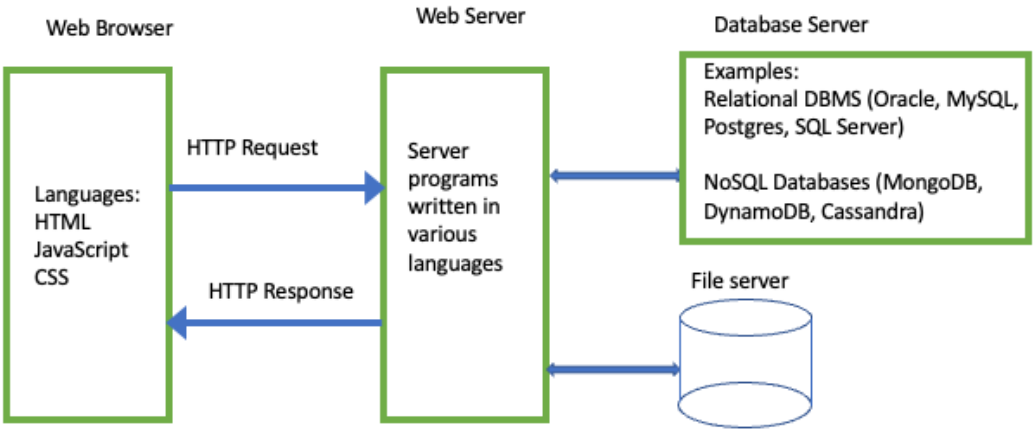
Create, Read, Update and Delete Operations

The initial focus of the web was on retrieval of documents. However, very quickly the need was identified that the web should go beyond supporting **information retrieval** to encompassing **information modification**. The web quickly evolved to support create, update and delete operations, in addition to the original support for read operations. **CRUD** is a common acronym used for Create, Read, Update and Delete. Sometimes the term "Retrieve" is used instead of "Read" in this acronym but the essential meaning remains the same.

Server Side Programming and Web Servers

To process HTTP requests for CRUD operations on arbitrary resources, business logic needed to be encoded in programs to, e.g., buy books, make payments, update wish lists, etc. **Web Servers**

were created to support writing and running programs in different programming languages to process HTTP requests. To process a request a program may call other programs, e.g., a database server to look up some data. Note that a program thus may act as a server for the HTTP request from the web browser, while simultaneously acting as a client for the program, e.g., a database server, that it calls.



In most web applications, the server program acts as a client to send requests to other servers, e.g., database servers.

Separating Style from Content with CSS

As the web grew, developers wanted more control over the look and feel of their webpages. This caused HTML to become more complex. The need was also identified for displaying webpages differently based on screen differences. This lead to efforts to separate the styling aspects of a webpage, e.g., colors, fonts, etc., from its structure. **Cascading Style Sheets** or **CSS** is a result of this effort. CSS is a language used for describing the presentation of documents on the web while HTML continues to be used for describing the structure of the documents. Multiple webpages on a website can have the same look and feel by sharing CSS files, while the same webpage can have different presentation for different screen sizes by using a different CSS file based on the screen size.

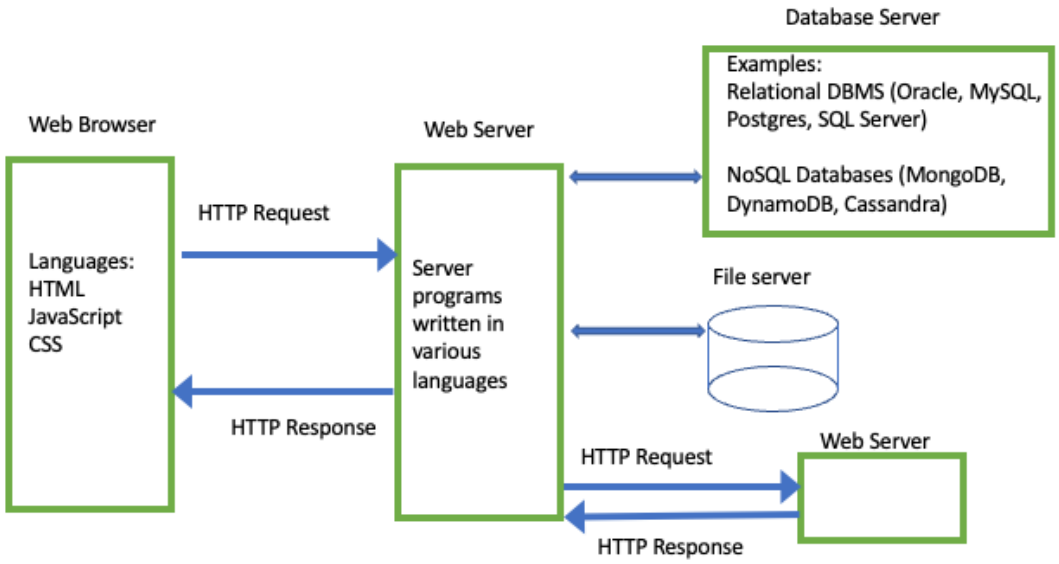
Client-Side Interactivity with JavaScript

In the early days, webpages were static. Once a page was loaded in the browser, it could not execute any code. Any interactivity in a web application required sending a request to the server where the necessarily processing would be done by a program running on the server and the result of the processing sent back as a response for the browser to display. A need was quickly identified for running programs in the browser to support full-blown applications with rich user interfaces and good performance that did not require round-trip communication to the server for every user interaction. In 1995 this lead to the development of **JavaScript**, a new programming language

which can be executed by web browsers. Along with HTML and CSS, JavaScript is a core technology for developing web applications. Overtime the use of JavaScript has grown beyond the browser. JavaScript is used as a general purpose programming language and is also very popular for implementing server side programs. In this course, we will use JavaScript for both client-side programs, i.e., programs that run in the web browser, and for server-side programs, i.e., programs that run in the Web Server to process HTTP requests received from the client and send back HTTP responses to the client.

Application to Application Interaction Using Web Services

The web project started with the aim to support document retrieval where the end users were humans. As the web grew in popularity, developers recognized that the web can be used to write applications where the end user of the document is not a human, but a program that needs information from the server. This lead to the development of **web services.** Web services use URLs for resource identification and HTTP as the protocol for communication with the client. However, the difference from traditional web applications is that documents sent in the response may be in formats other than HTML. The reason is that the end user of web servers is not necessarily a human who wants the document displayed in a browser. Instead, the end user may be a program which will interpret the document to get some information. Document formats such as **JSON** and **XML** are commonly used by web services. These formats convey information in a manner that is easy to interpret by a program. In many cases, a web application may process a request for an HTML document from a web browser by calling other web services, getting back data in a non-HTML format, and then processing this data for inclusion in an HTML document to send back to the web browser.



A server program may call other servers, including web services over HTTP, to process an HTTP request.

Summary

Three core technologies, URL, HTTP, and HTML were introduced by the World Wide Web project. Over time, other technologies, particularly CSS and JavaScript, have also become fundamental in building web applications. Next we start looking at the core technologies in more depth, starting with HTTP.

Additional Resources

Here are some references to learn more about the topics we discussed in this exploration.

- The Wikipedia page for the World Wide Web gives a good history and overview of the web. **World Wide Web (https://en.wikipedia.org/wiki/World_Wide_Web)**
- Tim Berners-Lee's proposal for the WWW project is available here **Information Management: A Proposal (https://www.w3.org/History/1989/proposal.html)**