



Module 1: Introduction to Web

Module Overview

This module explains about the aspect of web development and its use for full stack development



Module Objective

At the end of this module, students should be able to demonstrate appropriate knowledge, and show an understanding of the following:

- Web Technology
- HTTP Protocol
- URN
- Web Services
- Web applications
- Full Stack development
- Use of Python, Django Ruby on Rails in full stack development



Web technology Fundamental

ppj

www is online store of onformation. It is s/m of creating,organizinf and linking documents

What is the www?

WWW is stands for World Wide Web.

- The World Wide Web (WWW) is a global information medium which users can read and write via computer connected to the internet.

- The Web, or World Wide Web, is basically a system of Internet servers that support specially formatted documents. The documents are formatted in a markup language called HTML (Hypertext Markup Language) that supports links to other documents, as well as graphics, audio, and video files.

- In short, World Wide Web (WWW) is collection of text pages, digital photographs, music files, videos, and animations you can access over the Internet.

- Web pages are primarily text documents formatted and annotated with Hypertext Markup Language (HTML). In addition to formatted text, web pages may contain images, video, and software components that are rendered in the user's web browser as coherent pages of multimedia content.

- The terms Internet and World Wide Web are often used without much distinction. However, the two are not the same.
- The Internet is a global system of interconnected computer networks. In contrast, the World Wide Web is one of the services transferred over these networks. It is a collection of text documents and other resources, linked by hyperlinks and URLs, usually accessed by web browsers, from web servers.
- There are several applications called Web browsers that make it easy to access the World Wide Web; For example: Firefox, Microsoft's Internet Explorer, Chrome Etc.
- ① • Users access the World-Wide Web facilities via a client called a browser, which provides transparent access to the WWW servers. User can access WWW via two way such as:
w browser, HTML webpage, website

History of WWW:

- Tim Berners-Lee, in 1980 was investigating how computer could store information with random links. In 1989, while working at European Particle Physics Laboratory, he proposed to idea of global hypertext space in which any network-accessible information could be referred to by single "universal Document Identifier". After that in 1990, this idea expanded with further program and knows as World Wide Web.

Internet and WWW

- The Internet, linking your computer to other computers around the world, is a way of transporting content. The Web is software that lets you use that content...or contribute your own. The Web, running on the mostly invisible Internet, is what you see and click on in your computer's browser.

What is The Internet?

- The Internet is a massive network of networks, a networking infrastructure. It connects millions of computers together globally, forming a network in which any computer can communicate with any other computer as long as they are both connected to the Internet. Information that travels over the Internet does so via a variety of languages known as protocols. So, we can say that Internet is network of computer which connect to together and any computer communicate with any other computer.

What is The Web (World Wide Web)?

- The World Wide Web, or simply Web, is a way of accessing information over the medium of the Internet. It is an information-sharing model that is built on top of the Internet.
- The Web uses the HTTP protocol, only one of the languages spoken over the Internet, to transmit data. The Web also utilizes browsers, such as Internet Explorer or Firefox, to access Web documents called Web pages that are linked to each other via hyperlinks. Web documents also contain graphics, sounds, text and video.

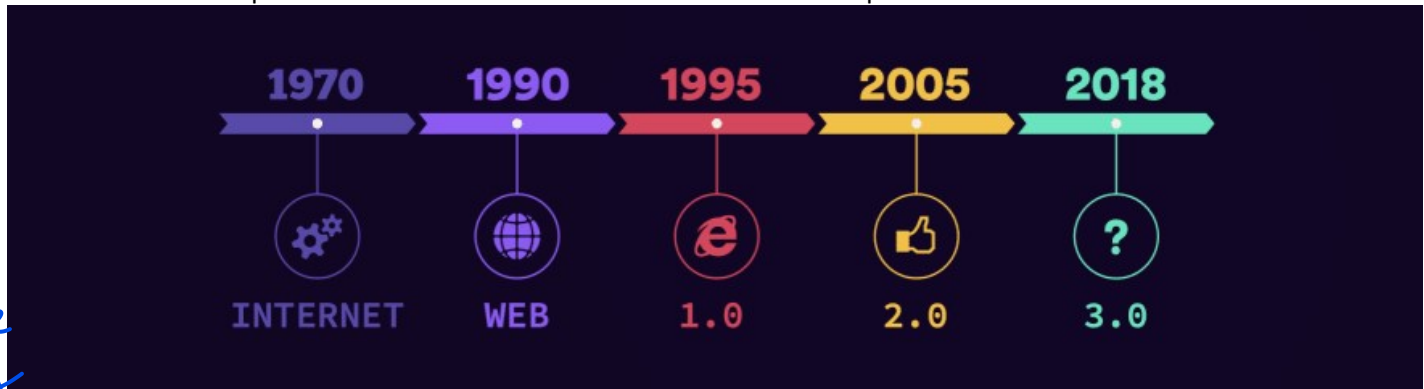
Different between Internet and WWW

- The Web is a Portion of The Internet. The Web is just one of the ways that information can be disseminated over the Internet. The Internet, not the Web, is also used for email, which relies on SMTP, Usenet news groups, instant messaging and FTP. So the Web is just a portion of the Internet.

Evolution of the Web / Web 1.0 / 2.0 / 3.0

② The internet has inarguably been the most important technology revolution in the history of mankind and fortunately, we have been in the right generation to keep up and observe the wide impact that it has on the world. However, the web that we know today has seen many phases, broadly categorized into three phases - Web 1.0, Web 2.0, and Web 3.0.

Let's dive a little deeper and discover how the web evolved from its inception to what we know it as now.

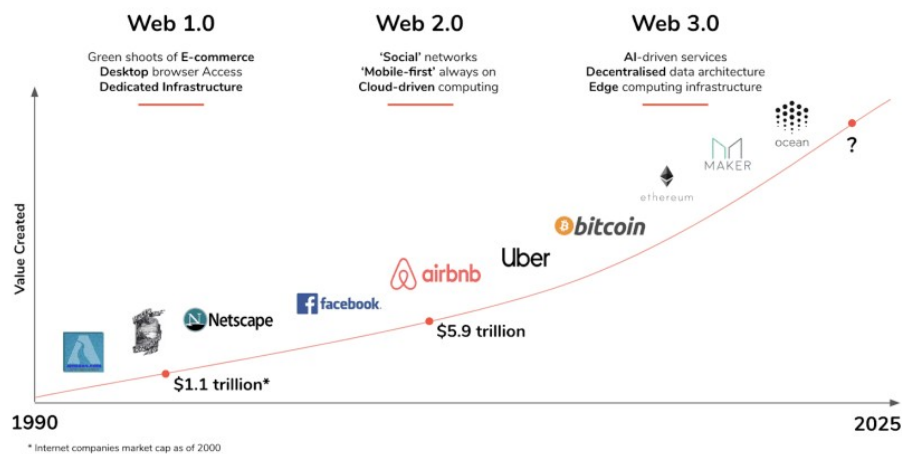


Web 1.0

Web 1.0 was the **first stage of the World Wide Web** revolution, usually referred to as **read-only web**. This was how the Internet we know today started in the first place, where the websites were merely **informational and comprised entirely static content**; they were only linked together by hyperlinks and lacked **any interactive content or design elements**. Moreover, this was the era when only text emails could be written and sent, one could not even upload or attach any **images or pictures**. Although, personal pages were quite common, consisting mainly of static pages hosted on ISP-run web servers, or on free web hosting services. Interestingly, it cost the user as per pages viewed. It had directories that enabled the user to retrieve a particular piece of information.

In a nutshell, Web 1.0 was a **content delivery network (CDN)** that enabled to showcase of the piece of information on the websites where users passively receive information without being given the opportunity to **post reviews, comments, and feedback**. The content here was served from the server's file system and the pages were built using *Server Side Includes (SSI)* or *Common Gateway Interface (CGI)*. Interestingly, frames and tables were used to position and align the elements on a webpage back then!

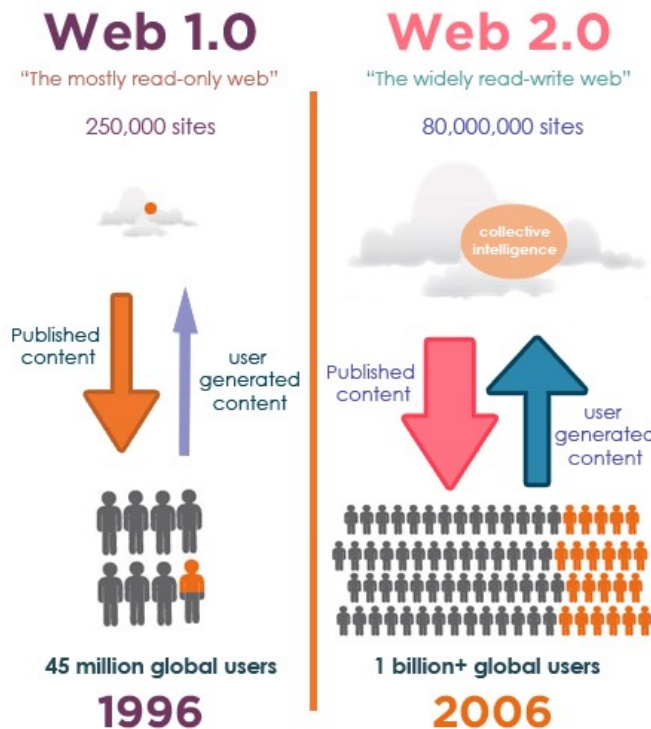
The Evolution of the Web



It spawned the **dot-com bubble**, which lasted from 1995 to 2000 and included many internet-based businesses. These were the companies that emerged with Web 1.0:

Web 2.0

Web 2.0 was the second stage of the evolution of the web, also called **the read-write web** and it was the phase when websites grew in **terms of user interaction**. It was the period when websites became more focused on user-generated content, usability, and interoperability for end-users, leading them to become the - **participative social web**.



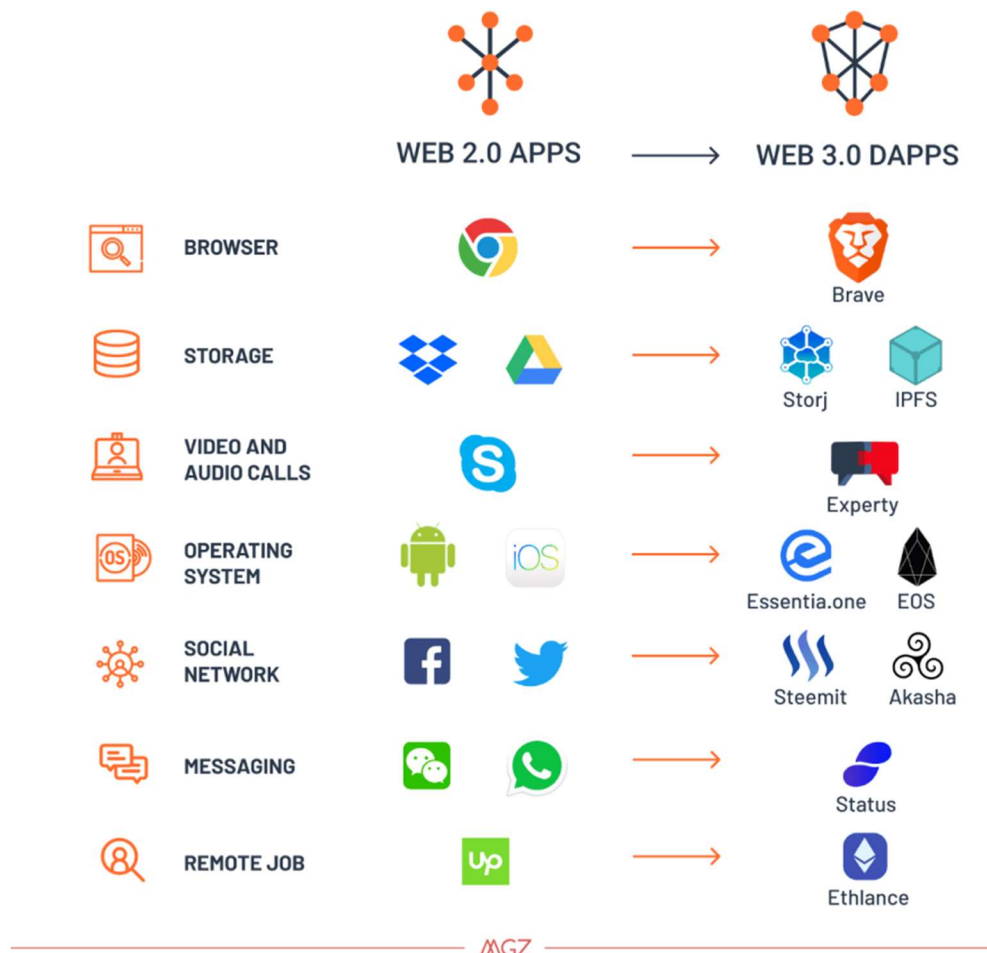
During Web 2.0, terms like **blogs, social media, and video streaming** gained popularity. This time period is also acknowledged for the ease with which **music and video clips could be exchanged**. It opened doors to podcasting, blogging, tagging, curating with RSS, **social bookmarking, social networking, social media, web content voting**, etc. It was the birthplace of **Youtube, Wiki, Flickr, Facebook, and so on**. Also, blogging became popular with the introduction of WordPress which started as a PHP & MySQL-led blogging platform and has now advanced to become a full content managed system (CMS) which power over a quarter of the web and e-Commerce completely revolutionized the way we shop.



Thus, Web 2.0 brought a fundamental shift where people were allowed to share their perspectives, opinions, thoughts, and experiences via a number of online tools and platforms. It brought us the concept - 'Web as Platform', where software applications are built upon the Web as opposed to upon the desktop. This was when websites began using web browser technologies such as AJAX and Javascript frameworks. This period continued to see the origin of APIs (Application Programming Interface) - a software intermediary that allows two applications to communicate with one another.

Web 3.0

Web 3.0 is the next generation of web, also termed as the executable web or read- write-execute web. It began with the onset of dynamic applications, interactive services, and "machine-to-machine" interaction. It is used to describe many evolutions of web usage and interaction between various paths. Data is not owned in this case, but rather shared, with services displaying different views for the same web/data.

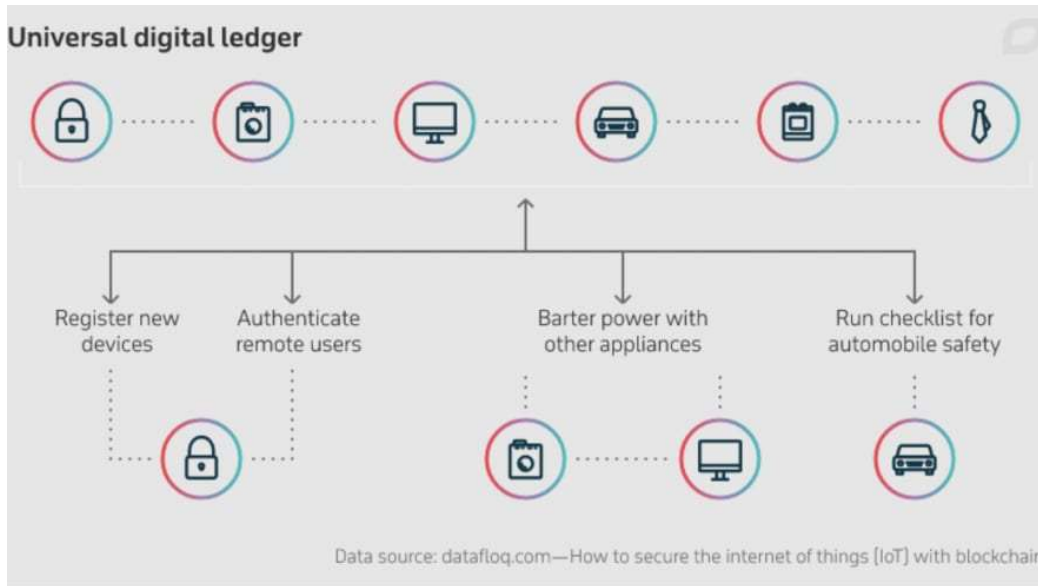


It has also been referred to as **Semantic Web** to describe a web in which machines would process content in a humanlike way where all data would be connected and understood both contextually and conceptually, leading to the herald of Artificial Intelligence and Machine Learning. With this, information is more connected thanks to semantic metadata. As a result, the user experience evolves to another level of connectivity that leverages all the available information. Two key terms associated with this phase are - semantic markup and web services. Apart from defining an item's appearance, semantic markup aids in its description. It guides in the search for other matched items based on similar attributes. Semantics, as opposed to Web 2.0, focuses on easy searching. You can now simply enter a keyword into Google Search and it will intelligently suggest related words.

Web 3.0 – What next?

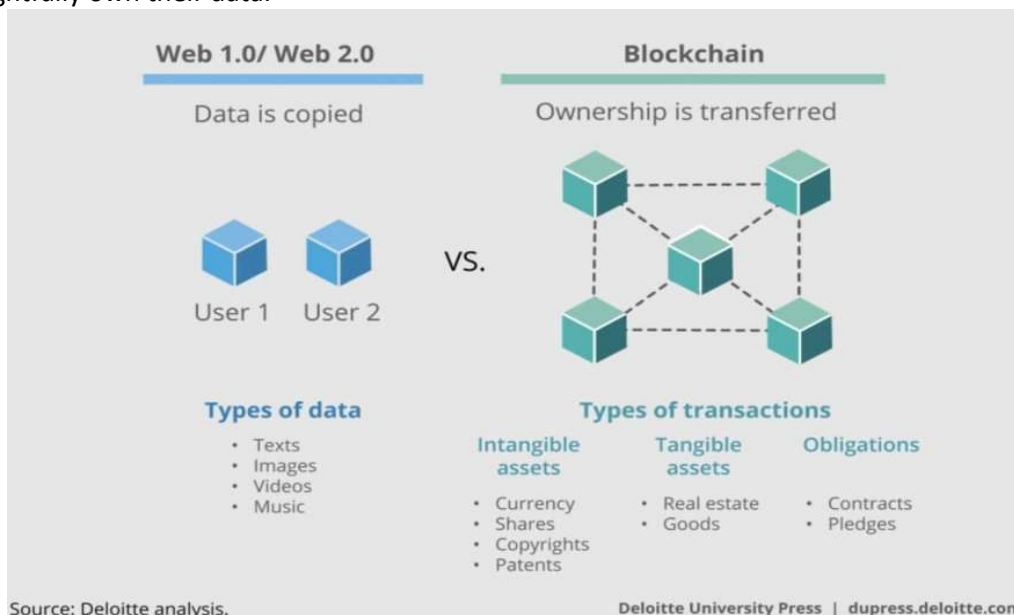
Universal applications

The future of Web 3.0 points to universal applications which can be read and used by a large number of devices and software types, making the ways in which we indulge business and leisure increasingly convenient.



The decentralized web

The rise of technologies such as distributed ledgers and blockchain storage will enable data decentralization and the creation of a transparent and secure environment, subverting Web 2.0's centralization, surveillance, and exploitative advertising. Decentralized infrastructure and application platforms will displace centralized tech giants, allowing individuals to rightfully own their data.



Indeed, one of the most significant implications of decentralization and blockchain technology is in the area of data ownership and compensation. As we move toward Web 3.0 and the technologies that support it would mature and become scalable, thus, the decentralized blockchain protocol will allow individuals to connect to an internet where they can own and be properly compensated for their time and data, eclipsing an exploitative and unjust web in which giant, centralized repositories own and profit from it.

World Wide Web Consortium

an association, typically of several companies.

The World Wide Web Consortium was created in October 1994 to lead the World Wide Web to its full potential by developing common protocols that promote its evolution and ensure its interoperability. W3C has around 350 Member organizations from all over the world and has earned international recognition for its contributions to the growth of the Web.

Background

In October 1994, Tim Berners-Lee, inventor of the Web, founded the World Wide Web Consortium (W3C) at the Massachusetts Institute of Technology, Laboratory for Computer Science [MIT/LCS] in collaboration with CERN, where the Web originated, with support from DARPA and the European Commission. For further information on the joint initiative and the contributions of CERN, INRIA, and MIT, please see the statement on the joint World Wide Web Initiative. In April 1995, INRIA (Institut National de Recherche en Informatique et Automatique) became the first European W3C host, followed by Keio University of Japan (Shonan Fujisawa Campus) in Asia in 1996. In 2003, ERCIM (European Research Consortium in Informatics and Mathematics) took over the role of European W3C Host from INRIA. W3C also pursues an international audience through its Offices worldwide.

W3C's Goals

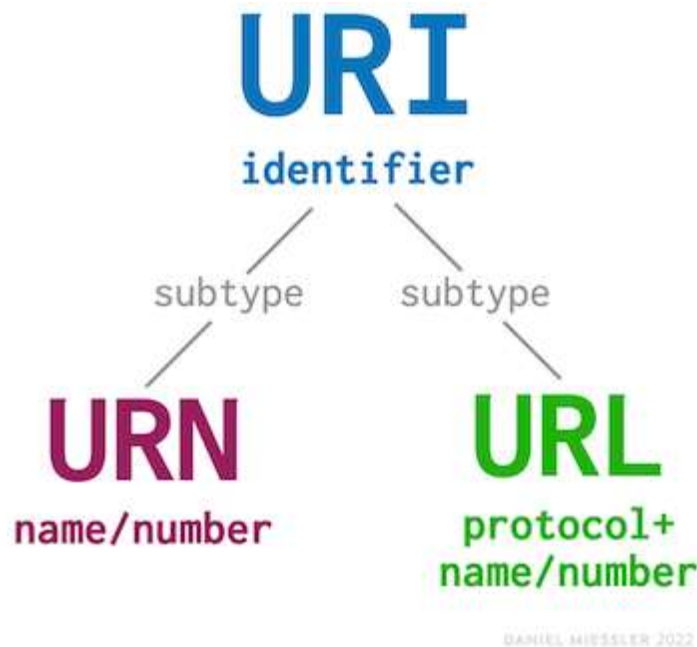
W3C's long term goals for the Web are:

1. Universal Access: To make the Web accessible to all by promoting technologies that take into account the vast differences in culture, languages, education, ability, material resources, access devices, and physical limitations of users on all continents;
2. Semantic Web: To develop a software environment that permits each user to make the best use of the resources available on the Web;
3. Web of Trust: To guide the Web's development with careful consideration for the novel legal, commercial, and social issues raised by this technology

web standards



Unified Resource Identifier (URI)



What is a URI?

A URI — short for “Uniform Resource Identifier” — is a sequence of characters that distinguishes one resource from another.

For example, foo://example.com:8042/over/there?name=ferret#nose is a URI containing a scheme name, authority, path, query and fragment. A URI does not need to contain all these components. All it needs is a scheme name and a file path, which can be empty.

Here’s another example of a URI: telnet://192.0.2.16:80/. In this example, “telnet” is the scheme name and the numbers after the double slash make up the authority. The path is empty, which is why no characters come after the slash

Types of URI

There are two types of URIs: URNs and URLs.

A Uniform Resource Name (URN) is a persistent and location-independent identifier that follows the “urn” scheme. In this context, “persistent” means that the URN persists in identifying the same resource over time. Here’s an example of a URN provided by the RFC 3986: urn: oasis:names:specification:docbook:dtd:xml:4.1.2

A URL, on the other hand, is a location-dependent identifier that is not necessarily persistent. Meaning, URLs are not required to identify the same resource over time. URLs also do not follow the “urn” scheme. Now that we have a brief understanding of URIs, URLs, and URNs, let’s take a closer look at what URLs are below.

What is a URL?

A URL — short for “Uniform Resource Locator” — is a specific type of identifier that not only identifies the resource but tells you how to access it or where it’s located. For example, a URL might contain ftp:// or https://. This tells you that

the resource can be located and accessed via File Transfer Protocol (FTP) or Hypertext Transfer Protocol Secure (HTTPS). Here are some examples of URLs:

- ftp://ds.internic.net/internet-drafts/draft-ietf-uri-irl-fun-req-02.txt
- https://blog.hubspot.com/website/application-programming-interface-api
- http://www.ietf.org/rfc/rfc2396.txt
- https://offers.hubspot.com/how-to-run-seo-audit?hubs_post-cta=anchor&hsCtaTracking=f55ac8df-26f8-41f5-b63a-fa80e97d2fec%7Cfe8e963d-d682-4a22-b84e-52f7d60e4786

You'll notice that these look similar to the URI examples mentioned above. That's because they contain many of the same components, including path and query.

However, a URL contains unique components, such as protocol and domain, as well. In the last URL example, "https://" is the protocol. "Offers" is the subdomain, and "hubspot.com" the domain name. "How-to-run-seo-audit" is the path, and the question mark and everything after makes up the query.

Now let's take a closer look at the difference between a URI and a URL.

What is the difference between URIs and URLs?

The key difference between URIs and URLs is that URIs are identifiers, whereas URLs are locators. In other words, a URI simply identifies the resource. It does not describe or imply how to locate the resource. A URL does.

URLs are therefore a specific subset of URIs. Meaning, all URLs are URIs, but not all URIs are URLs.

The most common analogy used to understand the difference between URIs and URLs is comparing a person's name vs. their address. A person's name is like a URI because it identifies the person without providing any information on how to locate them. An address, however, identifies the person as resident of that address and provides their physical location. That's why it's like a URL.

The other major difference is that URIs can be used to identify and differentiate HTML, XML, and other files from each other. URLs, on the other hand, can only be used to identify and locate web pages.

Identifying and Locating Resources on the Web

With millions of resources available on the web, it's important to understand the proper way of identifying and locating individual resources. While URIs are used to identify resources with either a name or location, URLs are a subset of URIs that identify resources by how you access them.

Unified Resource Name (URN)

A Uniform Resource Name (URN) is a type of URI that uses the specific naming scheme of `urn:` —like `urn: isbn:0-486-27557-4` or `urn: isbn:0-395-36341-1`.

URIs	URNs	URLs
A name, name and location, or both	a name or number	a name/number with location
A name, a name and address, or both	Someone's name or address	Someone's name and address
An ISBN number	An ISBN number	mailto://dan@ft.io
micah@google.com	micah@google.com	mailto://dan@ft.io
dan@ft.io	ftp.google.com	ftp://ftp.google.com

So which should I use?

URI vs. URL is one of the most timeless of all the geek debates. If you ever hear someone correct someone—in one direction or another—you'll often hear the drawing of Katana swords right after.

So which is correct?

When most people mention a domain the HTTP protocol is implied, making it a URL.

The answer is that it depends on what's being discussed. If you're talking about an everyday website domain, like google.com, it's best to use URL instead of URI because URL is more specific.

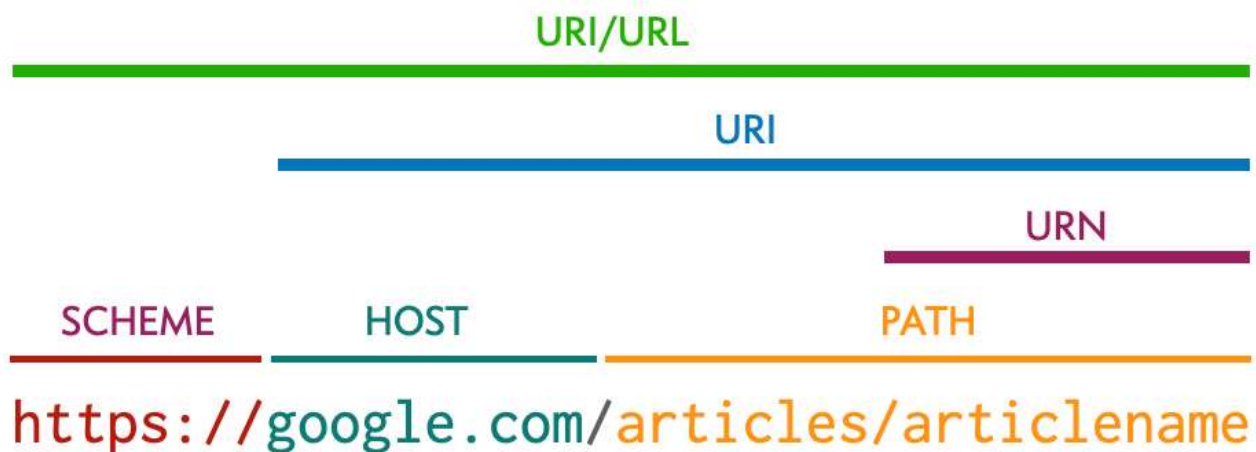
It's usually best to be as specific as possible, so URL is better than URI even when both are technically true.

If you're talking to someone from San Francisco, for example, and they ask you where you live, you wouldn't tell them you live in San Francisco, or in California. You'd respond with your neighborhood, because that's the level of detail they were asking for.

Technically google.com is a URI, just like technically you live in California, but google.com is also a URL and you also live in San Francisco.

In other words, in 99% of everyday cases, you should use URL instead of URI because both are technically true but URL is more specific!

URL Structures

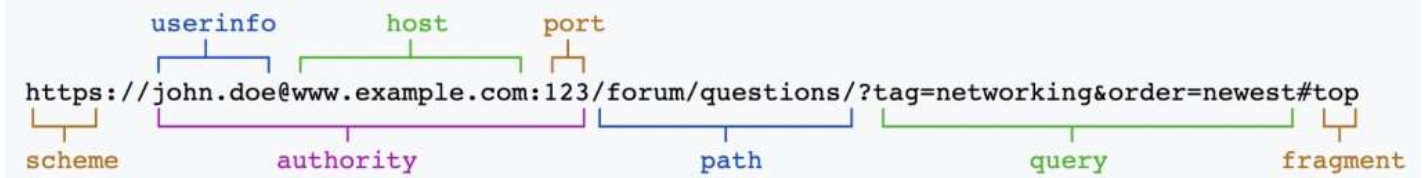


DANIEL MIESSLER 2022

URLs have their own specific structure as well. In that structure you have the following components:

1. The Scheme, which is the protocol that you're using to interact.

2. The Authority, which is the target you're accessing. This breaks down into user info, host, and port.
3. The Path, which is the resource you're requesting on the host.
4. The Query, which are the parameters being used within the web application.
5. The Fragment, which is the target to jump to within a given page.



Schemes can include: HTTP, HTTPS, FTP, MAILTO, IRC, FILE, etc. HTTP and HTTPS are usually used to reach internet resources, but they can point to local (on-network or on-computer) resources as well.

The FILE scheme refers to a file located on the local computer, and it looks for the file at the path that's provided. The host can also include a port designation **that overrides the default port for the specified protocol. For example:**

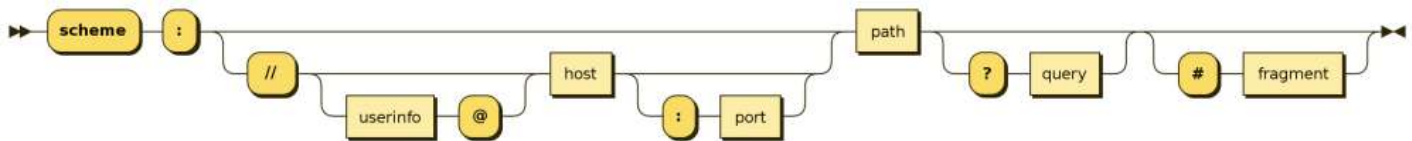
`https://google.com`

will go to the host google.com on port 443 because 443 is the default port for HTTPSs. But if you specify the port like so:

`https://google.com:9023`

the client will attempt to connect to connect to port 9023 using the HTTPS protocol instead.

Finally, URLs also have query parameters and fragment identifiers.



Query parameters indicate an argument being passed into a web application, such as a search function for a webpage, like:

`https://google.com/search?s=bing`

This would search for the word "bing" on a function called search on Google.

Fragments allow you to jump to a specific part of the page from the URL, like so:

`https://google.com/results.html#worse`

This would jump to a hyperlink on the page labeled "worse" on the page named results.html.

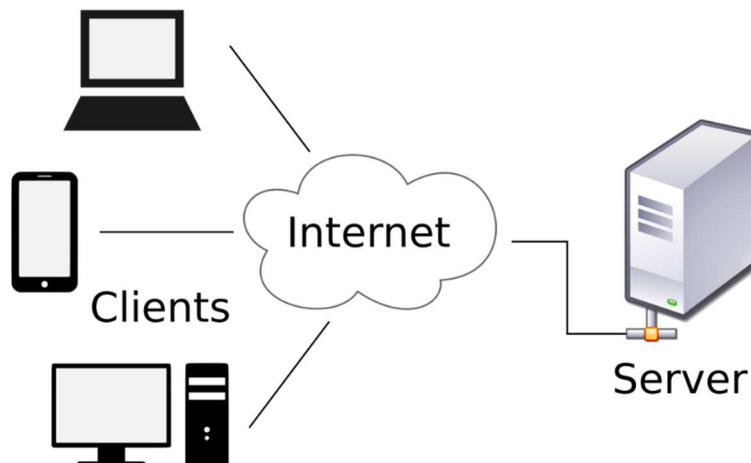


Web Server

Web Server

Introduction to Web Servers: A web server can be referred to as either the hardware (the computer) or the software (the computer application) that helps to deliver content that can be accessed through the Internet. A web server is what makes it possible to be able to access content like web pages or other data from anywhere as long as it is connected to the internet. The hardware houses the content, while the software makes the content accessible through the internet.

The most common use of web servers is to host websites but there are other uses like data storage or for running enterprise applications. There are also different ways to request content from a web server. The most common request is the Hypertext Transfer Protocol (HTTP), but there are also other requests like the Internet Message Access Protocol (IMAP) or the File Transfer Protocol (FTP).



How Web Servers Work

The Basic Process

Let's say that you are sitting at your computer, surfing the Web, and you get a call from a friend who says, "I Just read a great article! Type in this URL and check it out. It's at <http://www.howstuffworks.com/web-server.htm>." So you type that URL into your browser and press return. And magically, no matter where in the world that URL lives, the page pops up on your screen. At the most basic level possible, the following diagram shows the steps that brought that page to your screen:


Your browser formed a connection to a Web server, requested a page and received it. Behind the Scenes If you want to get into a bit more detail on the process of getting a Web page onto your computer screen, here are the basic steps that occurred behind the scenes:

The browser broke the URL into three parts:

1. The protocol ("http")
2. The server name ("www.howstuffworks.com")
3. The file name ("web-server.htm")

- The browser communicated with a name server to translate the server's name "www.howstuffworks.com" into an IP Address, which it uses to connect to the server machine.
- The browser then formed a connection to the server at that IP address on port 80.


Note: Any server machine makes its services available to the Internet using numbered ports, one for each service that is available on the server. For example, if a server machine is running a Web server and an FTP Server, the web server would typically be available on port 21. Clients connect to a service at a specific IP address and on a specific port.

- 
- Following the HTTP protocol, the browser sent a GET request to the server, asking for the file "http://www.howstuffworks.com/web-server.htm."
 - The server then sent the HTML text for the Web page to the browser.
 - The browser reads the HTML tags and displays the page onto your screen.

Web Server Features and Types

Features of Web Server

Here, we will explain the features of web server and characteristics; like as

- 
- ✓ **Web Server** can support enlarge data storage support, so it is capable to make multiple websites.
 - Easy to configure log file set up, enabling where to hold all log files. (Log files help to analyses web traffic and more)
 - ✓ It helps to control bandwidth to regulate network traffic, so due to this it can avoid the down time while flowing high volume web traffic.
 - Easy to make FTP websites, because it helps to move enlarge files from one site to another site.
 - Easy to set up website configuration and directory security
 - Easy to make virtual directories, and then help to map them along with physical directories.
 - ✓ Easy to set up of custom error pages **configuration** that means it helps to view user friendly error messages on your website, when your website is getting any issues like as 404 Error will be displayed if web pages do not present.
 - Can be specified default documents for example if any file has not specified with its name, then default documents will be displayed.

- ✓ It is enabled with Server-side web scripting so it allows user to make dynamic websites. Few **types of server-side scripting languages** are PHP, ASP, Ruby, Perl, Python, and more.

Advantages of Web Server

In this section, we will spread light on the different **benefits of web server**; such as –

- ✓ You have to right for implementing to **server-side** scripting languages like as PHP, Ruby etc.
- It is enabled with your coding style because your website structure can be represented as your predefined paths instead of your computer system's directory structure.
- It helps to provide you useful knowledge for making communication with your hosting provider, if you want to know about how it works in live environment.
- It helps to monitor download speed of any web App as well as performance.
- You have to permission for viewing URL construction and broken links etc.
- It provides better transparency in transaction in between your website and server while hosting.
- Your local website plays role as like the live one. For instance, you can configure directory security; test your custom error pages etc. before committing them to the production environment.
- It also delivers best insight in way HTTP communication.
- It is more control able and flexible.
- It is hold in protective infrastructure.
- Easy to handle all applications
- ✗ You have to permission to set up of customizes server as per your needs.

Disadvantages of Web Server

There are few **limitations of web server**; below explain each one –

- ✗ It may be more expensive compare to use of electronic website hosting.
- Harder to customize hosting service
- Due to overwhelm of server at any time, it can get down your website.
- It is comfortable to use only online enterprise, but if you want to adopt it for own companies which are more superior and requires customized design and style then it is not smart choice for that purpose.

- To need more protection to sites, like as credit card required at own base for e-commerce online store

Types of Web Server

In the computing language, the term 'Server' refers to either a computer program or hardware or a software device that provides functionality or other devices or programmes by **responding to the requests made by the network**.

A server basically refers to a certain computer system that receives requests from the web and sends this requested information to these clients.

It can broadly be understood as a device which is equipped with and backed by certain specific programs that enable it to offer services to other devices that it shares this network with.

A Web Server mainly refers to server hardware or a software device that stores the web content and is used to host the web sites and **produce the same as results when requested by the clients on the World Wide Web.**

To store, process and deliver the web pages when requested by the clients is the most prominent and technically the key feature and purpose of a Web Server.

These web servers generally tend to carry one or more websites. The requests sent by the clients over the World Wide Web are generally processed by the Web Server over Hypertext Transfer Protocol (HTTP) and the Web Pages are mostly delivered as HTML documents.

Web servers are at the very core of the concept of **web hosting**. A web server is always connected to the internet and each of these connected servers has a certain unique address.

The hosting providers are able to manage multiple domains on a single server because of the web servers.

Different types of web servers are available in the market for the developers to choose from, depending upon their preferences. The most prominent types of Web Servers available in the market are:

1. Apache HTTP Server Web Server
2. Internet Information Services (IIS) Web Server
3. Lighttpd Web Server
4. Sun Java System Web Server
5. Jigsaw Server Web Server
6. LiteSpeed server Web Server
7. Node.js Web Server

Apache HTTP Web Server



Developed by the Apache Software Foundation, the Apache HTTP Web Server is one of the most **popular Web Servers**. It is **a free and open-source cross-platform** web server software that was released under the terms of Apache License

2.0.

The key benefit of the Apache HTTP Web Server is that it tends to support almost all types of operating systems; be it Windows, Linux, Mac OS, Unix, FreeBSD and many more.

It is because of this that approximately around 60% of machines are run on the Apache HTTP Web Server.

Another distinguishing feature of the Apache Web Server is that it allows for a certain degree of customization very easily.

It owes this feature to its modular structure and the fact that it is open-source.

This implies that one has the freedom and scope to add new modules to the server as per his or her convenience and make any required modifications.

Some other noteworthy features of the Apache HTTP Server are:

- It can be easily installed on multiple platforms and function properly.
- Out of all the types of web servers available, the Apache HTTP servers are much more stable and easier in terms of functionality, especially the latest released versions which are capable of handling a distinguishingly greater number of client requests when compared to the other versions.

Internet Information Services Web Server (IIS)



The Internet Information Services Web Server is an extensible Web Server by Microsoft which was created with the Windows NT family. It is because of this that Microsoft maintains this product and it works with each and every Windows Operating System Platform.

The Internet Information Services Web Server supports the HTTP, HTTP/2, HTTPS, FTP, SMTP, and NNTP but unlike Apache is not open-source. This makes the process of customization a little complicated and modifications are not that easy.

Lighttpd Web Server

The Lighttpd Web Server is another open-source Web Server. This Web server along with being very flexible and secure is specifically designed and optimized for high performance, speed-critical environments.

Lighttpd gets its name from a portmanteau of 'light' and 'httpd' and has been nicknamed 'lighty'. Lighttpd defines itself in terms of security, speed, compliance, and flexibility.

A distinguishing feature of the Lighttpd Web Server is the efficiency that it offers especially in the case of servers suffering from load problems.

Sun Java System Web Server

Sun Microsystems used the brand Sun Java System to market computer software and this brand superseded the Sun ONE. Basically, the Sun Java System Web Server is a web server from Sun that runs under Windows, Solaris and HP- UX and is known for supporting the JavaServer Pages (JSP) technology, Server- Side JavaScript (SSJS) and Java Servlets. The Sun Java Web Server can be distinguished as a reliable, easy-to-use and secure web server specifically designed for large or medium sites.

Jigsaw Web Server

The Jigsaw Web Server is

an object-oriented, full-functioning Web Server that offers an array of distinguishing features along with an advanced architecture that is written in Java.

The birth of the Jigsaw Web Server was primarily for the purpose of experimentation of new technology and was not intended to be a full-fledged release. But the later versions of the Web Server took a leap and focused more on the development of better features.

LiteSpeed Web Server



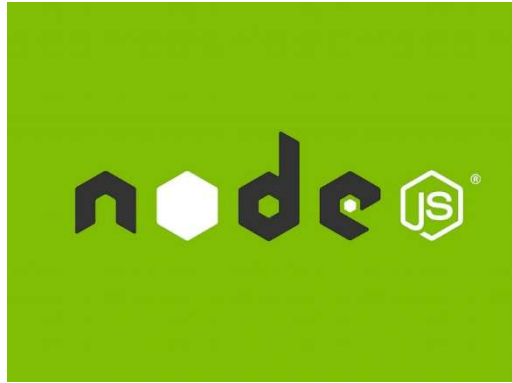
The LiteSpeed Web Server which was released in 2003 became one of the most popular Web Servers in a short span of few years.

It was developed by the privately held LiteSpeed Technologies and was a web server that came with Open Source variants as well.

The LiteSpeed Web Servers are quite compatible with the features of the Apache Web servers since they use similar configurations.

Owing to this feature, the LiteSpeed Web Servers are capable of directly loading the Apache configuration files and can replace all Apache functions.

Node.js Web Server



Node.js is known for executing the JavaScript code outside of a browser. It is an open-source, cross-platform, JavaScript runtime environment and enables developers to use JavaScript for writing commands.

Some of the prominent features of Node.js Web Server are:

- Owing to the fact that it is built on Google Chrome's V8 JavaScript Engine, it is very fast in code execution.
- Even though it is based on the Single-threaded model, the Node.js Web Server is highly scalable which gives it an upper hand over the other traditional Web Servers.

Every client comes with a different set of requirements and it is based on these requirements that the Web Hosting companies decide which Web servers to opt for.

Web clients

The web client is the front end or the user side of the web architecture. It can be a web browser or a web application that communicates through hypertext transfer protocol (HTTP) to format and transmit data such as documents, images, videos and audio files from a web server to the end-user.

The web client is connected to web servers through the Internet and it provides end-users the interface to interact with the web servers. It requests for data or web content through HTTP and the web server responds to the web client using the same protocol. Some of his key features are accessibility to all users, load content faster, mobile compatibility, efficient error handling, and effective navigation.

What are the examples of web clients?

These are examples that allow users to perform different tasks using the web.

- The popular web browsers such as Google Chrome, Internet Explorer, Opera, Firefox and Safari are examples that allow users to access any website through the Internet.
- Zoom is another example that allows users to create and join meetings and web conferences through the Internet.
- 3CX is another one that allows users to easily make calls, send chat messages, view the status of the colleagues and host video conferences online.

- Go Anywhere MFT web client assists users to perform secure file transfers and collaborate with employees and colleagues over the Internet.
- Collaborator is a web client designed for peer review tasks such as creating reviews as an author, reviewing or observing other reviewers and inspecting review materials.

Web browsers

According to Mozilla.org

A web browser takes you anywhere on the internet, letting you see text, images and video from anywhere in the world.

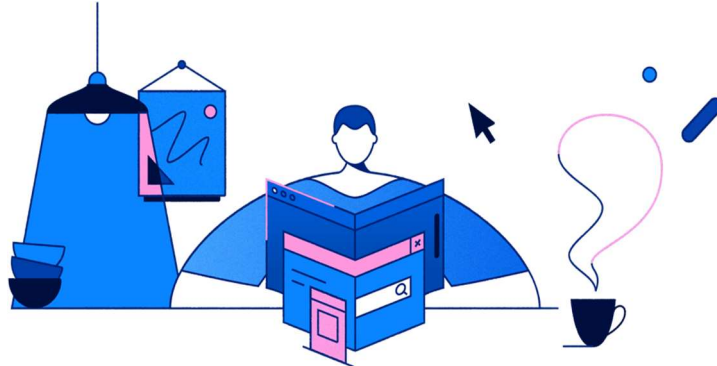


Figure reference: <https://www.mozilla.org>

The web is a vast and powerful tool. Over the course of a few decades, the internet has changed the way we work, the way we play and the way we interact with one another. Depending on how it's used, it bridges nations, drives commerce, nurtures relationships, drives the innovation engine of the future and is responsible for more memes than we know what to do with.

It's important that everyone has access to the web, but it's also vital that we all understand the tools we use to access it. We use web browsers like Mozilla Firefox, Google Chrome, Microsoft Edge and Apple Safari every day, but do we understand what they are and how they work? In a short period of time we've gone from being amazed by the ability to send an email to someone around the world, to a change in how we think of information. It's not a question of how much you know anymore, but simply a question of what browser or app can get you to that information fastest. In a short period of time, we've gone from being amazed by the ability to send an email to someone around the world, to a change in how we think about information.

How does a web browser work?

A web browser takes you anywhere on the internet. It retrieves information from other parts of the web and displays it on your desktop or mobile device. The information is transferred using the Hypertext Transfer Protocol, which defines how text, images and video are transmitted on the web. This information needs to be shared and displayed in a consistent format so that people using any browser, anywhere in the world can see the information.

Sadly, not all browser makers choose to interpret the format in the same way. For users, this means that a website can look and function differently. Creating consistency between browsers, so that any user can enjoy the internet, regardless of the browser they choose, is called web standards.

When the web browser fetches data from an internet connected server, it uses a piece of software called a rendering

engine to translate that data into text and images. This data is written in Hypertext Markup Language (HTML) and web browsers read this code to create what we see, hear and experience on the internet.

Hyperlinks allow users to follow a path to other pages or sites on the web. Every webpage, image and video has its own unique Uniform Resource Locator (URL), which is also known as a web address. When a browser visits a server for data, the web address tells the browser where to look for each item that is described in the html, which then tells the browser where it goes on the web page.



Hyper Text Transfer Protocol

Hyper Text Transfer Protocol

HTTP means Hypertext Transfer Protocol. HTTP is the underlying protocol used by the World Wide Web and this protocol defines **how messages are formatted and transmitted**, and what actions Web servers and browsers should take in response to various commands. For **example, when you enter a URL in your browser, this actually sends an HTTP command to the Web server directing it to fetch and transmit the requested Web page**. The other main standard that controls how the World Wide Web works is HTML, which covers how Web pages are formatted and displayed.



HTTP is called a stateless protocol because each command is executed independently, without any knowledge of the commands that came before it. This is the main reason that it is difficult to implement Web sites that react intelligently to user input.

HTTPS: A similar abbreviation, HTTPS means Hyper Text Transfer Protocol Secure. Basically, it is the secure version of HTTP. Communications between the browser and website are encrypted by Transport Layer Security (TLS), or its predecessor, Secure Sockets Layer (SSL).

Basic aspects of HTTP

HTTP is generally designed to be simple and human readable, even with the added complexity introduced in HTTP/2 by encapsulating HTTP messages into frames. HTTP messages can be read and understood by humans, providing easier testing for developers, and reduced complexity for newcomers.

HTTP is extensible

Introduced in HTTP/1.0, HTTP headers make this protocol easy to extend and experiment with. New functionality can even be introduced by a simple agreement between a client and a server about a new header's semantics.

HTTP is stateless, but not session less

HTTP is stateless: there is no link between two requests being successively carried out on the same connection. This immediately has the prospect of being problematic for users attempting to interact with certain pages coherently, for example, using e-commerce shopping baskets. But while the core of HTTP itself is stateless, HTTP cookies allow the use of stateful sessions. Using header extensibility, HTTP Cookies are added to the workflow, allowing session creation on each HTTP request to share the same context, or the same state.

HTTP and connections

A connection is controlled at the transport layer, and therefore fundamentally out of scope for HTTP. HTTP doesn't require the underlying transport protocol to **be connection-based; it only requires it to be reliable**, or not lose messages (at minimum, presenting an error in such cases). Among the two most common transport protocols on the Internet, **TCP is reliable and UDP isn't**. HTTP therefore relies on the TCP standard, which is connection-based.

Before a client and server can exchange an HTTP request/response pair, they must establish a TCP connection, a process which requires several round-trips. The default behavior of HTTP/1.0 is to open a separate TCP connection for each HTTP request/response pair. This is less efficient than sharing a single TCP connection when multiple requests are sent in close succession.

In order to mitigate this flaw, HTTP/1.1 introduced pipelining (which proved difficult to implement) and persistent connections: the underlying TCP connection can be partially controlled using the Connection header. HTTP/2 went a step further by multiplexing messages over a single connection, helping keep the connection warm and more efficient.

Experiments are in progress to design a better transport protocol more suited to HTTP. For example, Google is experimenting with **QUIC which builds on UDP** to provide a more reliable and efficient transport protocol.

What can be controlled by HTTP

This extensible nature of HTTP has, over time, allowed for more control and functionality of the Web. Cache and authentication methods were functions handled early in HTTP history. The ability to relax the origin constraint, by contrast, was only added in the 2010s.

Here is a list of common features controllable with HTTP:

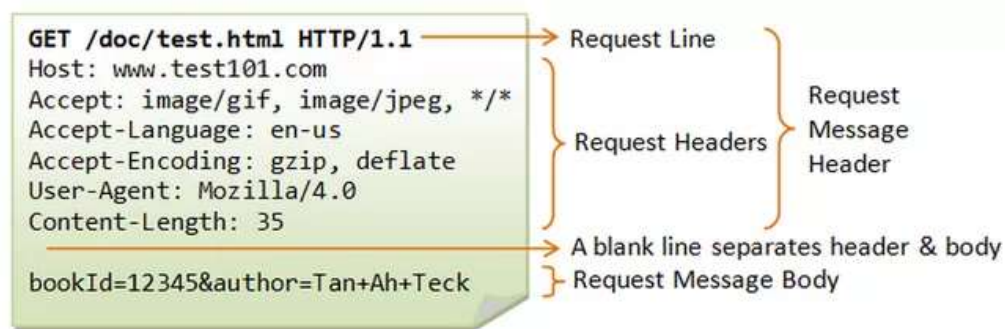
- **Caching** How documents are cached can be controlled by HTTP. The server can instruct proxies and clients about what to cache and for how long. The client can instruct intermediate cache proxies to ignore the stored document.
- **Relaxing the origin constraint** to prevent snooping and other privacy invasions, Web browsers enforce strict separation between Web sites. Only pages from the same origin can access all the information of a Web page. Though such a constraint is a burden to the server, HTTP headers can relax this strict separation on the server side, allowing a document to become a patchwork of information sourced from different domains; there could even be security-related reasons to do so.
- **Authentication** Some pages may be protected so that only specific users can access them. Basic authentication may be provided by HTTP, either using the WWW-Authenticate and similar headers, or by setting a specific session using HTTP cookies.
- **Proxy and tunneling** Servers or clients are often located on intranets and hide their true IP address from other computers. HTTP requests then go through proxies to cross this network barrier. Not all proxies are HTTP proxies. The SOCKS protocol, for example, operates at a lower level. Other protocols, like ftp, can be handled by these proxies.
- **Sessions** Using HTTP cookies allows you to link requests with the state of the server. This creates sessions, despite basic HTTP being a state-less protocol. This is useful not only for e-commerce shopping baskets, but also for any site allowing user configuration of the output.

How Hyper Text Transfer Protocol works?

How HTTP Works

HTTP is an application layer protocol built on top of TCP that uses a client-server communication model. HTTP clients and servers communicate through request and response messages. The three main HTTP message types are GET, POST, and HEAD.

- **HTTP GET:** Messages sent to a server contain only a URL. Zero or more optional data parameters may be appended to the end of the URL. The server processes the optional data portion of the URL, if present, and returns the result (a web page or element of a web page) to the browser.
- **HTTP POST:** Messages place any optional data parameters in the body of the request message rather than adding them to the end of the URL.
- **HTTP HEAD:** Requests work the same as GET requests. Instead of replying with the full contents of the URL, the server sends back only the header information (contained inside the HTML section).



The browser initiates communication with an HTTP server by initiating a TCP connection to the server. Web browsing sessions use server port 80 by default, although other ports such as 8080 are sometimes used instead.

After a session is established, you trigger the sending and receiving of HTTP messages by visiting the web page.

HTTP is what's called a *stateless system*. This means that, unlike other file transfer protocols such as FTP, the HTTP connection is dropped after the request completes. So, after your web browser sends the request and the server responds with the page, the connection closes.

HTTP MIME Types

A media type (also known as a Multipurpose Internet Mail Extensions or MIME type) indicates the nature and format of a document, file, or assortment of bytes. MIME types are defined and standardized in IETF's RFC 6838.

The Internet Assigned Numbers Authority (IANA) is responsible for all official MIME types,

Structure of a MIME type

A MIME type most-commonly consists of just two parts: a type and a subtype, separated by a slash (/) — with no whitespace between:

type/subtype

The type represents the general category into which the data type falls, such as video or text.

The subtype identifies the exact kind of data of the specified type the MIME type represents. For example, for the MIME type text, the subtype might be plain (plain text), html (HTML source code), or calendar (for iCalendar/.ics) files.

Each type has its own set of possible subtypes. A MIME type always has both a type and a subtype, never just one or the other. An optional parameter can be added to provide additional details:

type/subtype; parameter=value

For example, for any MIME type whose main type is text, you can add the optional charset parameter to specify the character set used for the characters in the data. If no charset is specified, the default is ASCII (US-ASCII) unless overridden by the user agent's settings. To specify a UTF-8 text file, the MIME type text/plain; charset=UTF-8 is used. MIME types are case-insensitive but are traditionally written in lowercase. The parameter values can be case-sensitive.

Types

There are two classes of type: **discrete and multipart**. Discrete types are types which represent a single file or medium, such as a single text or music file, or a single video.

A **multipart type** is one which represents a document that's comprised of multiple component parts, each of which may have its own individual MIME type; or, a multipart type may encapsulate multiple files being sent together in one transaction. For example, multipart MIME types are used when attaching multiple files to an email.

Discrete types

The discrete types currently registered with the IANA are:

Application

Any kind of binary data that doesn't fall explicitly into one of the other types; either data that will be executed or interpreted in some way or binary data that requires a specific application or category of application to use. Generic binary data (or binary data whose true type is unknown) is application/octet-stream. Other common examples include application/pdf, application/pkcs8, and application/zip. (Registration at IANA)

Audio

Audio or music data. Examples include audio/mpeg, audio/vorbis. (Registration at IANA)

Example

Reserved for use as a placeholder in examples showing how to use MIME types. These should never be used outside of sample code listings and documentation. example can also be used as a subtype; for instance, in an example related to working with audio on the web, the MIME type audio/example can be used to indicate that the type is a placeholder and should be replaced with an appropriate one when using the code in the real world.

Font

Font/typeface data. Common examples include font/woff, font/ttf, and font/otf. (Registration at IANA)

Image

Image or graphical data including both bitmap and vector still images as well as animated versions of still image formats such as animated GIF or APNG. Common examples are image/jpeg, image/png, and image/svg+xml. (Registration at IANA)

Model data for a 3D object or scene. Examples include model/3mf and model/vrml. (Registration at IANA)

Text

Text-only data including any human-readable content, source code, or textual data such as comma-separated value (CSV) formatted data. Examples include: text/plain, text/csv, and text/html. (Registration at IANA)

Video

Video, data or files, such as MP4 movies (video/mp4). (Registration at IANA) For text documents without a specific subtype, text/plain should be used. Similarly, for binary documents without a specific or known subtype, application/octet-stream should be used.

Multipart types

Multipart types indicate a category of document broken into pieces, often with different MIME types; they can also be used — especially in email scenarios — to represent multiple, separate files that are all part of the same transaction. They represent a composite document.

Except for multipart/form-data, used in the POST method of HTML Forms, and multipart/byteranges, used with 206 Partial Content to send part of a document, HTTP doesn't handle multipart documents in a special way: the message is transmitted to the browser (which will likely show a "Save As" window if it doesn't know how to display the document).

There are two multipart types:

Message

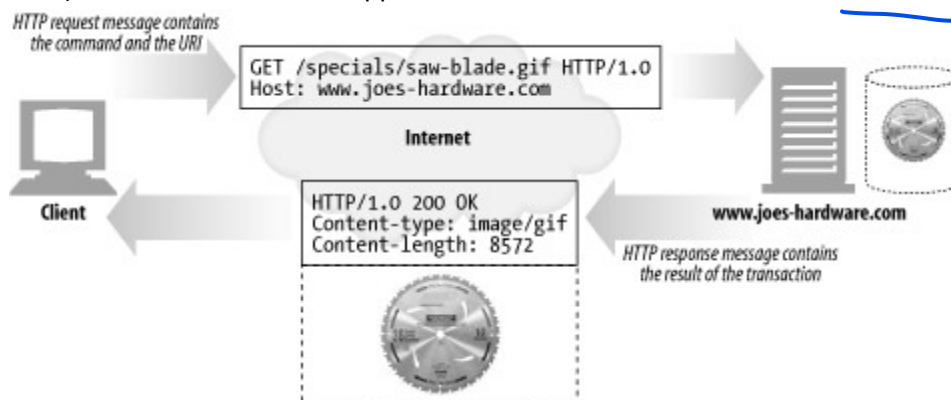
A message that encapsulates other messages. This can be used, for instance, to represent an email that includes a forwarded message as part of its data, or to allow sending very large messages in chunks as if it were multiple messages. Examples include message/rfc822 (for forwarded or replied-to message quoting) and message/partial to allow breaking a large message into smaller ones automatically to be reassembled by the recipient. (Registration at IANA)

multipart

Data that consists of multiple components which may individually have different MIME types. Examples include multipart/form-data (for data produced using the FormData API) and multipart/byteranges (defined in RFC 7233, section 5.4.1 and used with HTTP's 206 "Partial Content" response returned when the fetched data is only part of the content, such as is delivered using the Range header). (Registration at IANA)

HTTP Transaction HTTP Methods The HTTP Status Codes

An HTTP transaction consists of a request command (sent from client to server), and a response result (sent from the server back to the client). This communication happens with formatted blocks of data called *HTTP messages*



Methods

HTTP supports several different request commands, called *HTTP methods*. Every HTTP request message has a method. The method tells the server what action to perform (fetch a web page, run a gateway program, delete a file, etc.). table lists five common HTTP methods.

HTTP method	Description
GET	Send named resource from the server to the client.
PUT	Store data from client into a named server resource.
DELETE	Delete the named resource from a server.
POST	Send client data into a server gateway application.
HEAD	Send just the HTTP headers from the response for the named resource.

Status Codes

Every HTTP response message comes back with a status code. The status code is a three-digit numeric code that tells the client if the request succeeded, or if other actions are required. A few common status codes are shown in table

HTTP status code	Description
200	OK. Document returned correctly.
302	Redirect. Go someplace else to get the resource.
404	Not Found. Can't find this resource.

HTTP also sends an explanatory textual “reason phrase” with each numeric status code (see the response message in above figure. The textual phrase is included only for descriptive purposes; the numeric code is used for all processing.

The following status codes and reason phrases are treated identically by HTTP software:

200 OK
200 Document attached
200 Success
200 All's cool, dude



Web Applications

A Web application (Web app) is an application program that is stored on a remote server and delivered over the Internet through a browser interface. Web services are Web apps by definition and many, although not all, websites contain Web apps. According to Web.AppStorm editor Jarel Remick, any website component that performs some function for the user qualifies as a Web app.

Web applications can be designed for a wide variety of uses and can be used by anyone; from an organization to an individual for numerous reasons. Commonly used Web applications can include webmail, online calculators, or e-commerce shops. Some Web apps can be only accessed by a specific browser; however, most are available no matter the browser.

How Web applications work

Web applications do not need to be downloaded since they are accessed through a network. Users can access a Web application through a web browser such as Google Chrome, Mozilla Firefox or Safari.

For a web app to operate, it needs a Web server, application server, and a database. Web servers manage the requests that come from a client, while the application server completes the requested task. A database can be used to store any needed information.

Web applications typically have short development cycles and can be made with small development teams. Most Web apps are written in JavaScript, HTML5, or Cascading Style Sheets (CSS). Client-side programming typically utilizes these languages, which help build an application front-end. Server-side programming is done to create the scripts a Web app will use. Languages such as Python, Java, and Ruby are commonly used in server-side programming.

Front-end and Back-end Web Development

Advantages:

Web applications have many different uses, and with those uses, comes many potential benefits. Some common benefits of Web apps include:

1. Allowing multiple users access to the same version of an application.
2. Web apps don't need to be installed.
3. Web apps can be accessed through various platforms such as a desktop, laptop, or mobile.
4. Can be accessed through multiple browsers.

Web Application vs. other application types

Within the mobile computing sector, Web apps are sometimes contrasted with native apps, which are applications that are developed specifically for a particular platform or device and installed on that device. However, the two are not mutually exclusive. Native applications are applications typically downloaded and made specifically for the type of device it is downloaded on. Native apps can commonly make use of the device-specific hardware, such as a GPS or camera on a mobile native app.

Programs that combine the two approaches are sometimes referred to as hybrid applications. Hybrid apps work similar to a Web app but are installed to the device as a native app would be. Hybrid apps can also take advantage of device-specific resources by using internal APIs. Downloaded native apps can sometimes operate offline; however, hybrid apps don't have this functionality. A hybrid app will typically share similar navigation elements are a Web app since they are based on Web apps.


Front-end and Back-end Web Development

Front- and back-end development are the two main specialization areas within web development. Front-end handles what is visible to visitors while back-end handles the background processes. Both are equally important in making a functional website.

What is front-end development?

Front-end development is the programming of all visible elements to visitors/users of websites and web applications. Programmers who specialize in front-end development are said to be specialists in client-side web development, as their work focuses on what clients see.

Front-end specialists' work encompasses all the essential elements of functional, dynamic websites, including:

- 
- Layout/design
 - Buttons
 - Internal links
 - Images
 - Navigation


To excel in their work, front-end developers need a sense of how the different surface-level site elements combine into a cohesive, easily-usable whole. They need good rapport with the back-end developers who make data from the server available. They also usually need to communicate with web designers to make their aesthetic ideas fully functional. Front-end languages, frameworks, and libraries.

Anyone can learn to code, and to become a web developer you need to know at least a few programming languages to write active code. Front-end specialists must master the "big three": HTML, CSS, and Java. These are among the easiest programming languages to learn and form the front-end's foundation.

What is back-end development?

Back-end development is the area of web development focused on all of the necessary background processes occurring at the "server side" of websites, web apps, and mobile apps. What is interesting about this area of web development is how important it is despite being "behind the scenes" and invisible to lay people.

Back-end web developers' work includes the following concepts:

- 
- Database management
 - Communication between servers and applications through Application Programming Interfaces (APIs)
 - Back-end logic
 - Accessibility and security policy compliance

In order to properly create and maintain websites, back-end web developers need to work closely with front-end developers. This allows for smooth communication between different databases, servers, and applications and therefore smooth delivery of a final product to clients at the visible end.

Back-end languages, frameworks, and databases

Back-end web development, as one might expect, draws upon a slightly different set of tools than front-end web development. For instance, even though JavaScript is often used at the server-end of web development, CSS and HTML are absent. The programming languages that back-end web developers most often use include:

- Python
- PHP
- Ruby
- JavaScript
- Java

Python is a favorite of many back-end developers due to its general-purpose applicability and usefulness in automation. Back-end web developers often use frameworks to automate some back-end tasks necessary to keep sites running quickly and securely. Some of the most popular include:

- Django
- Laravel
- Flask
- Ruby on Rails



Without these frameworks, common website functions such as infinite scrolling would not be possible.

Libraries are also used in back-end development to allow for more flexibility. For instance, some popular Python libraries used by back-end specialists include:

- FastApi
- Pillow
- TensorFlow

Using prewritten code from libraries decreases the tedium and repetition of delegating many basic back-end coding tasks.



Web Technology

In introduction part we already learn about web technologies and its application, in this section we will discuss about web services in details

Web Services

Web Services represents an architectural structure that allows communication between applications. The use of eXtensible Markup Language (XML)- based Technology allows exploitation of services without needing to know what platform or language was used to create those services.

Web Services represent an architectural structure that allows communication between applications. A service can be invoked remotely or be used to employ a new service together with another services. Beyond the basic Web Services structure, this paper will be presenting some standard technologies to build Web Services, such as: Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP) and Universal Description, Discovery and Integration (UDDI). These technologies are based in eXtensible Markup Language (XML), and they allow service reutilization and invocation, without needing to know the underlying language or platform

What are Web Services?

Web Services are modularized applications that can be described, published and invoked under a network, commonly being WEB-based. I.e., Web Services are interfaces that describes a collection of operations that are accessible through the network by throwing standard XML messages. Web Services allow service integration in a fast, effective way.

A Web Service is a software component that depends on the implementation and platform. It can be described using a service description language, published in a registry and discovered by a standard mechanism. Thus, it can be invoked from an Application Program Interface (API) through the network, built in round of another services.

Web Services contain the best accomplishments of the Web-based component development. Like its software components counterparts, Web Services contains black-box functionalities that can be reused without caring in what language it will be used.

The Web Services terminology describes specific functionalities of the exposed business, usually through Internet connections with the purpose to provide some way to exploit this service.

The exposure of services enrolls:

- Identify or define value functions in the process or business.
- Define a service-based interface to the processes.
- Describe these interfaces in a Web-based form.

To make a service available in the Web is commonly necessary:

- Publish the service interface, allowing users find and use it.

- Accept requests and send responses using the XML message protocol.
- Establish a binding between external requests and business functions implementation.

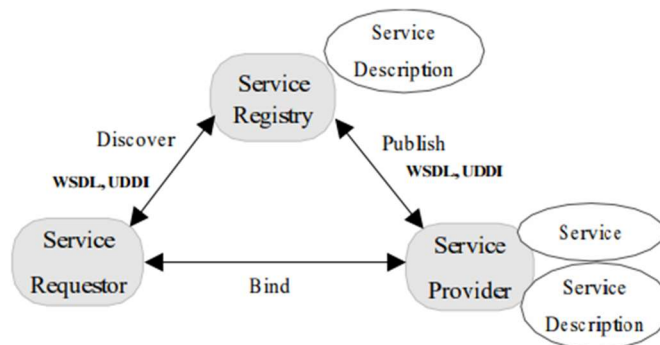
By using a message-form document, a program sends a request to a Web Service through the network and (optionally) receives some response, also in a XML-based message-form document. The usage of XML raises the importance of provide such extensible mechanism to describe the messages and its contents. Furthermore, Web-based services standards define the format of the message and mechanisms to publish and discover Web Services interfaces.

Web Services are not accessed via specific object model protocols such as Distributed Component Object Model (DCOM), Remote Method Invocation (RMI) or Internet Inter-ORB Protocol (IIOP); actually, they are accessed through transport protocols like HTTP, FTP, etc.

Web Services Architecture

In a Web Service architecture, a service description covers all the necessary details to grant the service interaction, including messages' format, the transport protocols and physical location. This interface hides service implementation details, allowing that the same service can be used independently of the underlying hardware or language, thus making Web Services-based applications component-oriented, making those components available for reuse. The Web Services architecture presented in following figure is built upon a three-role interaction:

- **Service Provider:** this is the service's owner from the business perspective. From the architectural approach, this is the platform that is accessed in the service request. It is also the entity that creates the Web Service, being responsible to make its description in some standard format and publish its details in a central registry. How web services work.
- **Service Requestor:** it is an application that invokes or initializes some interaction with the service. It could be a web browser or even a non-user interface program such as another Web Service. By using the service description it's possible to discover and invoke Web Services
- **Services Registry:** it's the place where service providers publish their service descriptions. Service Requestors search the Registries, fetching binding and description information both during the development time (static bindings) or run time (dynamic bindings).



Web Service Components

SOAP – Simple Object Access Protocol

SOAP (Simple Object Access Protocol) is a message protocol that allows distributed elements of an application to communicate. SOAP can be carried over a variety of lower-level protocols, including the web-related Hypertext Transfer Protocol (HTTP). SOAP defines a header structure that identifies the actions that various SOAP nodes are expected to take on the message, in addition to a payload structure for carrying information.

The concept of routing a message through a string of nodes that perform different functions is how SOAP supports things like addressing, security and format-independence. Essentially, the headers identify roles, which in turn provide the SOA features which SOAP then routes to. Stringing messages through a sequence of steps is uncommon in today's microservice-centric development environments.

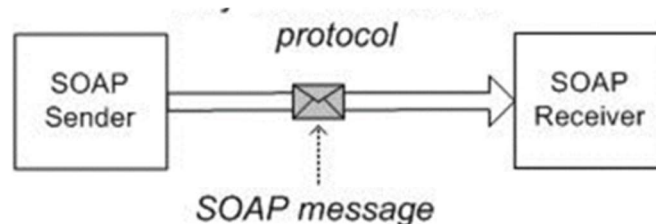
SOAP Advantages and Disadvantages

SOAP is an integral part of the service-oriented architecture (SOA) and the Web services specifications associated with SOA. Because it allows the sender to create a message route based on the logical services that have to be applied to the message on the way to its destination, it lends itself to providing secure and compliant connections, controlling access, offering reliable delivery and failure recovery, and supporting dynamic service discovery. SOA without SOAP is difficult to imagine.

SOAP's messages are defined at a high level in XML, but most SOAP applications use Web Services Definition Language (WSDL), which is authored in XML. The XML structure of SOAP makes it handy for applications that expect their information to be provided in XML form, and the fact that SOAP can ride on a variety of network protocols, including HTTP, means it's easily passed through firewalls, where other protocols might require special accommodation.

The data structure of SOAP is based on XML, which is similar in many ways to the HTML used to define web pages. Like HTML, XML is largely human-readable, which makes it fairly easy to understand a SOAP message, but also makes the messages relatively large in comparison to the Common Object Request Broker Architecture (CORBA) and its Remote Procedure Call (RPC) protocol that will accommodate binary data.

The biggest disadvantage of SOAP (and SOA overall) is that it's a heavyweight protocol for a heavyweight architecture. The notion of a message passing through a string of nodes to be processed by each seems to mix protocols and service bus architectural models for software, and neither of those two are considered optimal for microservice-based development as popularly used today.



SOAP APIs

SOAP is a protocol that's almost always used in the context of a Web Services/SOA framework. As such, it's application programming interface (API) is typically hidden by the higher-level interface for SOA. There are SOA API middleware tools available for nearly all modern programming languages, and Microsoft offers a variety of .NET SOAP/SOA tools.

SOAP vs REST

SOAP is designed to break traditional monolithic applications down into a multi-component, distributed form without losing security and control. In contrast, REST is a model of distributed computing interaction based on the HTTP protocol and the way that web servers support clients. REST over HTTP is almost always the basis for modern microservices development and communications. RESTful APIs uses HTTP requests to GET, PUT, POST and DELETE data.

REST/HTTP is simple, flexible, lightweight, and offers little beyond a way of exchanging information. SOAP can ride on HTTP as well, but it connects the elements of a complex set of distributed computing tools (the Web Services and SOA framework) as well as application components, and this forms a part of a total service-oriented framework.

Future of Simple Object Access Protocol

SOAP was the first widely used protocol for connecting web services in a Service Oriented Architecture (SOA). Today, nearly all modern development of distributed applications is based on RESTful principles. SOAP is almost always confined to legacy applications and projects, and over time its use is declining.

REST – Representational State Transfer

REST (Representational State Transfer) is an architectural style for developing web services. REST is popular due to its simplicity and the fact that it builds upon existing systems and features of the internet's Hypertext Transfer Protocol (HTTP) in order to achieve its objectives, as opposed to creating new standards, frameworks and technologies.

Advantages and disadvantages of REST

There are several advantages to using REST. They are as follows:

- **Resource-based.** A primary benefit of using REST, from both a client and server perspective, is that REST interactions are based on constructs which are familiar to anyone accustomed to using HTTP. Employing a resource-based approach, REST defines how developers interact with web services.
- **Communication.** REST-based interactions communicate their status through numerical HTTP status codes. REST APIs use these HTTP status codes to detect errors and ease the API monitoring process. They include the following:
 - 404 error indicates that a requested resource wasn't found;
 - 401 status response code is triggered by an unauthorized request;
 - 200 status response code indicates that a request was successful; and
 - 500 error signals an unrecoverable application fault on the server.
- **Familiarity.** Most developers are already familiar with key elements of the REST architecture, such as Secure Sockets Layer (SSL) encryption and Transport Layer Security (TLS).
- **Language-independent.** When creating RESTful APIs or web services, developers can employ any language that uses HTTP to make web-based requests. This capability makes it easy for programmers to choose the technologies they prefer to work with and that best suit their needs.

Pervasive. The popularity of REST is due to its widespread use in both server- and client-side implementations. For example, on the server side, developers can employ REST-based frameworks, including Restlet and Apache CXF. On the client side, developers can employ a variety of frameworks (i.e., jQuery, Node.js, Angular, EmberJS, etc.) and invoke RESTful web services using standard libraries built into their APIs.

Web APIs. When it comes to caching, RESTful services employ effective HTTP mechanisms. For example, by providing many endpoints, a REST API makes it easier for developers to create complex queries that can meet specific deployment needs.



Disadvantages of REST are as follows:

Architecture. Developers working with REST frequently encounter limitations due to its architecture design. These include multiplexing multiple requests over a single TCP connection, having different resource requests for each resource file, server request uploads, and long HTTP request headers, which cause delays in webpage loading.

Stateless applications. Since HTTP does not store state-based information between request-response cycles, the client must perform state management tasks. This makes it difficult for programmers to implement server updates without the use of client-side polling or other types of webhooks that send data and executable commands from one app to another.

Definition. Developers generally disagree over defining REST-based designs. As an architectural style, REST lacks a clear reference implementation or a definitive standard that designates whether a specific design can be defined as RESTful. This also leads to uncertainty over whether a given web API conforms to REST-based principles.

Data overfetching/underfetching. RESTful services frequently return large amounts of unusable data combined with relevant information, typically the result of multiple server queries. These inefficiencies also increase the time it takes for a client to return all the required data.

Alternatives to REST

Alternate technologies for creating SOA-based systems or creating APIs for invoking remote microservices include XML over HTTP (XML-RPC), CORBA, RMI over IIOP and the Simple Object Access Protocol (SOAP).

In general, every technology has benefits and drawbacks. However, a unique feature of REST is that instead of requiring that developers work with custom protocols for client-server message exchanges, REST insists that the best way to implement network-based web services is to use the basic construct of the network protocol itself, which in terms of the internet is HTTP.

This is an important component, as REST is not intended to apply just to the internet; rather, its principles are intended to apply to all protocols, including WebDav and FTP.

WSDL – Web Services Description Language

Web Services Description Language (WSDL) is a format for describing a Web Services interface. It is a way to describe services and how they should be bound to specific network addresses. WSDL has three parts:

- Definitions
- Operations
- Service bindings



Definitions are generally expressed in XML and include both data type definitions and message definitions that use the data type definitions. These definitions are usually based upon some agreed upon XML vocabulary. This agreement could be within an organization or between organizations. Vocabularies within an organization could be designed specifically for that organization. They may or may not be based on some industry-wide vocabulary. If data type and message definitions need to be used between organizations, then most likely an industry-wide vocabulary will be used. For more on XML vocabularies, XML, however, is not necessary required for definitions. The OMG Interface Definition Language (IDL), for example, could be used instead of XML. If a different definitional format were used, senders and receivers would need to agree on the format as well as the vocabulary. Nevertheless, over time, XML-based vocabularies and messages are likely to dominate. XML Namespaces are used to ensure uniqueness of the XML element names in the definitions, operations, and service bindings.

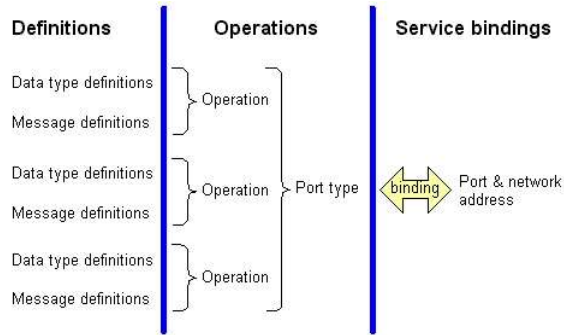
Operations describe actions for the messages supported by a Web service. There are four types of operations:

- One-way: Messages sent without a reply required
- Request/response: The sender sends a message and the receiver sends a reply.
- Solicit response: A request for a response. (The specific definition for this action is pending.)
- Notification: Messages sent to multiple receivers. (The specific definition for this action is pending.)

Operations are grouped into port types. Port types define a set of operations supported by the Web service.

Service bindings connect port types to a port. A port is defined by associating a network address with a port type. A collection of ports defines a service. This binding is commonly created using SOAP, but other forms may be used. These other forms could include CORBA Internet Inter-ORB Protocol (IIOP), DCOM, .NET, Java Message Service (JMS), or WebSphere MQ to name a few.

The following figure shows the relationship of the basic parts of WSDL:



UDDI – Universal Description, Discovery and Integration

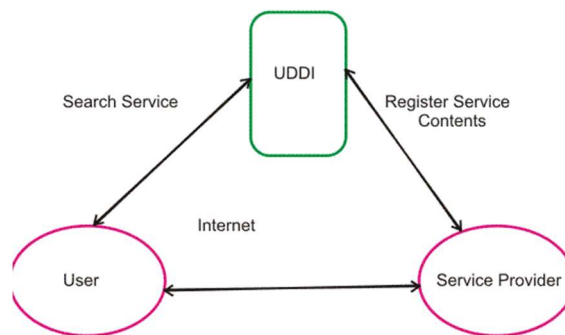
UDDI (Universal Description, Discovery, and Integration) is an XML-based registry for businesses worldwide to list themselves on the Internet. Its ultimate goal is to streamline online transactions by enabling companies to find one another on the Web and make their systems interoperable for e-commerce. UDDI is often compared to a telephone book's white, yellow, and green pages. The project allows businesses to list themselves by name, product, location, or the Web services they offer.

Microsoft, IBM, and Ariba spearheaded UDDI. The project now includes 130 companies, including some of the biggest names in the corporate world. Compaq, American Express, SAP AG, and Ford Motor Company are all committed to UDDI, as is Hewlett-Packard, whose own XML-based directory approach, called e-speak, is now being integrated with UDDI.

While the group does not refer to itself as a standards body, it does offer a framework for Web services integration. The UDDI specification utilizes World Wide Web Consortium (W3C) and Internet Engineering Task Force (IETF) standards such as XML, HTTP, and Domain Name System (DNS) protocols. It has also adopted early versions of the proposed Simple Object Access Protocol (SOAP) messaging guidelines for cross platform programming.

In November 2000, UDDI entered its public beta-testing phase. Each of its three founders - Microsoft, IBM, and Ariba - now operates a registry server that is interoperable with servers from other members. As information goes into a registry server, it is shared by servers in the other businesses. The UDDI beta is scheduled to end in the first quarter of 2001. In the future, other companies will act as operators of the UDDI Business Registry.

UDDI registration is open to companies worldwide, regardless of their size.





What is web development

Web development refers in general to the tasks associated with developing websites for hosting via intranet or internet. The web development process includes web design, web content development, client-side/server-side scripting and network security configuration, among other tasks.

In a broader sense, web development encompasses all the actions, updates, and operations required to build, maintain and manage a website to ensure its performance, user experience, and speed are optimal.

Web development is the coding or programming that enables website functionality, per the owner's requirements. It mainly deals with the non-design aspect of building websites, which includes coding and writing markup.

Web development ranges from creating plain text pages to complex web-based applications, social network applications and electronic business applications.

The web development hierarchy is as follows:

- Client-side coding.
- Server-side coding.
- Database technology.

Most web developments use Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript to develop websites.

HTML defines the basic framework of a website – the foundation upon which everything else is built upon. It forms the blocks that define a page's layout, format, and critical components. Although it is theoretically possible to code a website on HTML only, it will be just a barebone site with no functions unless it's enriched with CSS and JavaScript. Also, even simple style modifications such as changing the color of a button require a lot of coding to be executed using HTML only.

CSS is used to style the content of a website using a small set of files that are kept across the entire site. This way, whenever a change must be applied to say, consistently change the color of all the buttons found in every page of the website, a web dev needs to edit only a single file in CSS.

The JavaScript programming language is used to take care of the interactivity of many unique website elements. It can be used to create effects that alter the appearance of icons and drop-down menus, add animations, games, and other interactive elements.

Web developers are usually divided into front-end devs, back-end devs, and full-stack devs. A front-end dev takes care of all the visual aspects of the website (layout, navigation bar, etc.), its interactivity, and binds together all its elements.

Back-end devs take care of less visible tasks that ensure the website runs smoothly, such as managing the website's hosting services, database, and applications. Back-end devs might need to engineer solution to server issues by using additional server-side languages such as Python, Ruby, Java, and PHP.

Why Python is suitable for web development

Developers in different fields use Python for various things, including web development, deep learning, and artificial intelligence (AI). Using Python for websites will help you compete with giants like Google, Facebook, and Microsoft because they all use the programming language.

What is python?

Python is an adaptable and highly efficient programming language. This is because Python for websites offers dynamic typing capabilities. This versatile programming language lets developers create scientific applications, system applications with graphics, games, command-line utilities, web applications, and so much more. Every time we create a website with Python, it is high-functioning and very successful.

Another benefit of creating a website with Python is that you can use and distribute it for free. Python is known as an open-source product, which means you can get all the coding information you need on the Internet. Other open-source products include Perl, Apache, and Linux. Also, copying, embedding, and distributing Python in your products is unrestricted. This makes it extremely useful in the world of digital systems, which is important for your business because it gives you flexibility in the marketplace by allowing your company to interact with industry sectors. Ultimately, this will help your business be successful for a long time to come.

Python is a well-known software that is used everywhere. Many small companies and developers rely on it, and companies like Facebook, Google, Dropbox, Microsoft, Mozilla, and Intel all have websites built with Python. Besides the big companies, different development fields use Python for website development as well.

How web development fields use python

- **Web development:** Web development relates to creating, deploying, and operating all applications and programming interfaces on the Internet. We can use Python for web development to create web-based applications. Generally, we use a combination of Python and JavaScript to accomplish this.
- **Machine learning:** Machine learning helps computers take in data without being programmed. Python is used to set up the computer, and the language is incorporated throughout its learning process.
- **Artificial intelligence (AI):** AI is used to quickly and efficiently analyze and process large amounts of data. It can be used to give personalized suggestions to potential customers. Python is one of the most popular programming languages for AI.
- **Deep learning:** This is part of AI, and its goal is to process data in a way similar to that of the human brain. Python for web development allows developers in this field to work with robotics and image recognition.
- **Internet of things:** Python helps devices like cameras and games connect to the Internet quickly.

Remaining adaptive in today's fluid marketplace will ensure your company's success and longevity.

Is python good for web development?

Python is known and used around the world. Hundreds of thousands of developers and businesses use Python for web development. Huge companies like Red Hat, Yandex, Google, Microsoft, and Facebook all rely on Python. JetBrains developed PyCharm, which is the most popular integrated development environment for Python. According to JetBrains, web development is second only to machine data analysis.

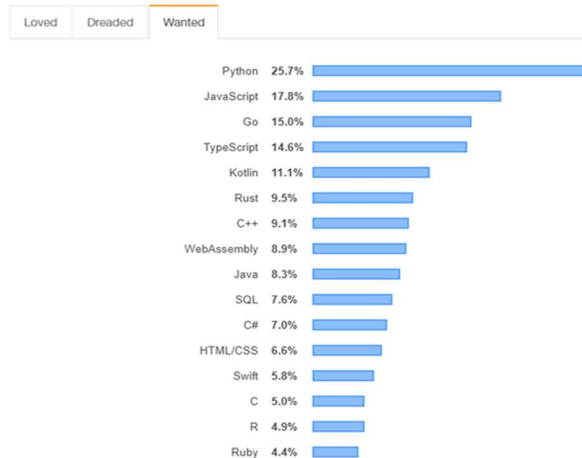
What do you use Python for?



Most developers use Python primarily for web development and machine learning.

According to the Stack overflow, for the second year in a row, Python is the programming language in the highest demand. This means that Python is the programming language that developers say they want to learn. So, why use Python for web development?

Most Loved, Dreaded, and Wanted Languages



This diagram shows the most in-demand programming languages. As you can see, Python is at the top of the list.

Why do we prefer using python for website development?

Developers who know how to build a website with Python are in such high demand because Python is a powerful and highly advanced programming language. However, it is extremely difficult to find a good web development company in the Denver, Philadelphia or Austin areas that use Python for website creation. Of course, there are many other programming languages that developers can choose to use. But because Python is one of the more advanced

and complex languages in the world of website design and development, programmers who know Python come with hefty salary demands. Unfortunately, the average web design company cannot afford to pay these developers' annual salaries, which means that only large companies can usually pay for projects to make a website with Python.

This is why smaller companies tend to find it so difficult to compete with larger companies, like Microsoft and Facebook, who have websites built with Python. Usually, web designers in Denver and Philadelphia download WordPress templates and modify them slightly. This is typically accomplished via a PHP programmer who either works in-house, or the work is outsourced to a company in another country such as India. This template is then sold at a high price as a "custom-designed" website. While this manipulation should not be legal, it happens daily. We have already written several articles on problems with websites like Wix and WordPress, so feel free to check them out.

At Direct Line Development, we do things differently. We give you a truly customized website by using only the best developers who know the languages in the highest demand today. Our results help our clients' businesses shine and showcase our teams' skills. Python web development allows us to do this.

1. **Easy to learn and use:** Compared to other programming languages, Python is one of the easiest to learn and use. Because it is relatively short, using Python for web development and debugging it is not difficult.
2. **Perfect for building prototypes:** Because the language is easy to access, it is not difficult to ensure that the program works properly. You can build prototypes to test out the code.
3. **Flexible:** Python was not originally created to fulfill a need. Because of this, its structure is not specific. Instead, it can integrate with several other programming languages.
4. **Fantastic frameworks:** Python can use many different frameworks for building apps and websites. This saves you the tedious task of having to start from scratch.
5. **Versatile and runs on every platform:** Whether you are developing a website or managing cloud infrastructures, Python web development can perform numerous functions.
6. **Django** – the ultimate weapon for Python: Django is a framework of Python. Python web development with Django allows web developers to skip getting started and instead focus on parts of their application that are new. Django has the most features of any framework out there.

Web Framework

It is a software framework that was developed in order to simplify the web development process and make it easier to build a website. It includes templating capabilities that allow you to present information within a browser, provides an environment for scripting how information flows and also contains many applications programming interfaces (APIs) for gaining access to underlying data resources. Most frameworks also provide tools in order for web developers to build a content management system (CMS) for managing digital information on websites and the Internet.

A web application or development framework can be considered as a pre-built structure that handles the more repetitive processes and features involved with developing a website. This means that a web developer will spend most of their time interacting with the different parts of the web framework through the use of code.

Some of the features associated with web frameworks include web caching, authentication and authorization procedures, database mapping and configuration and URL mapping.

The advantages and disadvantages of web frameworks

The most notable advantages for web developers that use a web framework are due to the fact that it is open source, efficient, has a good level of support, has a high level of security and includes an integration feature.

Being open-source means that web frameworks are very cost effective for both the developer and the client. This doesn't mean that they aren't of good quality. Most of the popular web frameworks used by developers are free for use. Efficiency is another important advantage for web developers. This is because web frameworks eliminate the need to write a lot of repetitive code allowing developers to build websites and applications much quicker. A web framework also comes with a support team and advanced security features meaning that you can be rest assured knowing that if an issue does arise, it will be handled by a team of experts. One of the most helpful features of a web framework is its integration feature that has the ability to allow developers link other tools such as databases to the framework.

However, saying that, developers often note that the biggest disadvantages with using a web framework are due to it being very limited in terms of making changes to the web framework. Everything from coding paradigms and design is very restrictive. This could cause frustration for web developers. Another disadvantage is that some developers find it difficult to learn the language behind web frameworks. Oftentimes, web developers will learn the framework but will not know the coding language behind it making it difficult for them to fully understand everything about the web framework itself.

Python Web Framework

1. Django

About: Django, an open-source framework, is a popular high-level web framework in Python which supports rapid web development and design.

Some of its features are-

- Django helps developers avoid various common security mistakes.
- With this framework, developers can take web applications from concept to launch within hours.
- The user authentication system of this framework provides a secure way to manage user accounts and passwords.

2. CherryPy

About: CherryPy is a popular object-oriented web framework in Python. The framework allows building web applications in a much simpler way.

Some of its features are-

- A powerful configuration system for developers and deployers alike.
- Built-in profiling, coverage, and testing support.
- Built-in tools for caching, encoding, sessions, authentication, static content etc.
- A reliable, HTTP/1.1-compliant, WSGI thread-pooled webserver.

- A flexible plugin system.

3. TurboGears

About: TurboGears is a Python web application framework. The next version, TurboGears 2, is built on top of several web frameworks, including TurboGears 1, Rails and Django.

Some of its features are:

- It is designed to be a web application framework suitable for solving complex industrial-strength problems.
- It has a transaction manager to help with multi-database deployments.
- It officially supports MongoDB as one of the primary storage backends.
- It provides support for multiple template engines.

4. Flask

About: Flask is a popular Python web framework used for developing complex web applications. The framework offers suggestions but doesn't enforce any dependencies or project layout.

Some of its features are-

- Flask is flexible.
- The framework aims to keep the core simple but extensible.
- It includes many hooks to customise its behaviour.

5. Web2Py

About: Written in Python, Web2Py is a free, open-source web framework for agile development of secure database-driven web applications. It is a full-stack framework.

Some of its features are-

- It is designed to guide a web developer to follow good software engineering practices, such as using the Model View Controller (MVC) pattern.
- Web2Py automatically addresses various issues that can lead to security vulnerabilities by following well-established practices.
- The framework includes a Database Abstraction Layer (DAL) that writes SQL dynamically.

6. Bottle

About: Bottle is a fast, simple and lightweight WSGI micro web framework for Python web applications. The framework has no other dependencies than the standard Python library.

Some of its features are-

- Bottle runs with Python 2.7 and 3.6+.
- It has a fast and Pythonic *built-in template engine* and support for mako, jinja2 and cheetah templates.
- The framework has convenient access to form data, headers, file uploads, cookies, and other HTTP-related metadata.
- Built-in HTTP development server as well as support for bjoern, Google App Engine, fapws3, cherrypy or any other WSGI capable HTTP server.

7. Falcon

About: Falcon is a WSGI library for building speedy web APIs and app backends. The framework has CPython 3.5+ and PyPy 3.5+ support. Falcon complements more general Python web frameworks by providing extra reliability, flexibility, and performance.

Some of its features are-

- It includes a highly optimised and extensible codebase.
- Easy access to headers as well as bodies through the request and response objects.
- The framework provides DRY request processing via middleware components and hooks.

8. CubicWeb

About: Written in Python, CubicWeb is a free and open-source semantic web application framework. It empowers developers to efficiently build web applications by reusing components (called cubes) and following the well-known object-oriented design principles.

Some of its applications are-

- It has a query language named RQL, similar to W3C's SPARQL.
- It includes a library of reusable components that fulfil common needs.

9. Quixote

About: Quixote is a framework for writing Web-based applications using Python. The goal of this framework is to provide flexibility and high-performance during web development.

Some of its features are-

- Flexibility and high-performance.
- Quixote includes Python Template Language for producing HTML with Python code.

10. Pyramid

About: Pyramid is a lightweight and open-source Python web framework. The framework provides only the core tools needed for nearly all web applications: mapping URLs to code, security, and serving static assets (files like JavaScript and CSS).

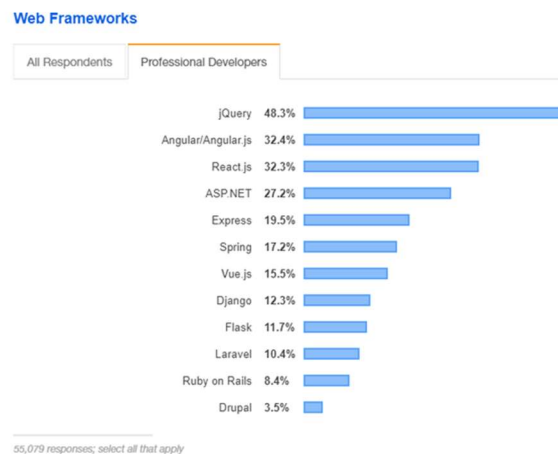
Some of its features are-

- Support for Python 3.8 and 3.9.
- New security APIs to support a massive overhaul of the authentication and authorisation system.

Which web framework is better?

Why django and what does it have to do with python web development?

Django is a backend framework of Python, and it is used to create high-level websites. Python web development with Django allows developers to focus on creating their websites, without starting from the beginning. Not only is Django the most universal and functional Python language in 2019, but it is also one of the top 10 most popular frameworks in the world. This is because Django web development with Python is quick and easy to use, very secure, works on any number of web application projects, and has been around for a long time.



These are the most popular web frameworks in the world. Django, a Python framework, is in the top 10. [Stackoverflow chart from 2019 survey](#) Several popular companies use Django as their main development framework. These include sites for:

- Spotify
- NASA
- Reddit
- Quora

Django web development with Python is one of the most effective ways to build popular websites because it can create highly scalable websites or ones where the audience is constantly growing. Of course, there are many more sites created with Django. We have only selected a few of the most popular ones.

WHY DJANGO IS THE PERFECT FRAMEWORK FOR PYTHON WEBSITES?

Combining Python and Django leads to amazing results and chances to develop fast websites, regardless of traffic flow. Take a look at each of these benefits of Django web development with Python.

- **It's fast and simple.** Django can simplify life for developers so that they can focus on creating a website with Python instead of starting from scratch.

- **It's secure.** Django comes with lots of security features, including SQL injection and cross-scripting. This helps avoid the most common security issues.
- **It suits any web application project.** Whether you are making a simple website or a huge one, Django is compatible. It also works on a lot of different computers, including Macs and PCs.
- **An incredible community of developers.** When other developers have problems with the code, they can reach out to a community of developers who are willing to help them with their problems. These people are also willing to help you learn how to implement the code.
- **SEO-friendly framework** – Django is compatible with SEO services. For example, if you move your website to a new location, you can use Django to tell search engine bots that you have moved. You can also use Django web development with Python to create a sitemap that will allow more of your pages to rank in search engine results pages. Finally, Django can reduce the amount of time it takes for your website to load, which will help it rank higher in Google.
-



Introduction to Full Stack Development/ Roadmap of Full stack development

What is Full Stack Development?

A full stack developer is someone who works with both the front end and back end of a web application. The front end is responsible for the visual look and feel of the website, while back end is responsible for the behind-the-scenes logic and infrastructure of the site. Let's take a closer look at the terms front end and back end.

Front end Development (Client side)

Everything on a web page from the logo, to the search bar, buttons, overall layout and how the user interacts with the page was created by a front end developer. Front end developers are in charge of the look and feel of the website. Front end developers also have to make sure the website looks good on all devices (phones, tablets, and computer screens). This is called Responsive Web Design.

Back end Development (Server side)

Back end development refers to the part of the application that the user does not see. Back end developers focus on the logic of the site, creating servers, and working with databases and API's (Application Programming Interfaces). For example, the content and layout for a form would be created on the client side, but when the users submits their information it gets processed on the server side (back end).

Skills You Need as a Full Stack Web Developer

Let's take a look at the technologies and tools you will need to learn to become a full stack developer.

HTML

HTML stands for HyperText Markup Language. HTML displays the content on the page like buttons, links, headings, paragraphs, and lists.

CSS

CSS stands for Cascading Style Sheets. CSS is responsible for the style of your web page including colors, layouts, and animations. Responsive design is essential in creating websites that look good on all devices. Accessibility is the practice of making sure that everyone can easily use your web sites. You do not want to create web sites that cannot be used by those who use assistive technologies like screen readers.

JavaScript

You use JavaScript with HTML and CSS to create dynamic and interactive web pages and mobile applications. Examples of JavaScript include animations, count down clocks, drop down mobile menus, and showing/hiding information when a user clicks on an element on the page.

CSS Frameworks, Libraries, and Preprocessors

These tools were created to help speed up the development process. Instead of writing all custom CSS, you can use a framework's catalog of CSS classes in your web pages.

There is no need to learn them all, but here is a list of a few options:

- Bootstrap
- Tailwind CSS
- Bulma
- Materialize
- Semantic UI

CSS preprocessors like Sass and Less allow you to add logic and functionality to your CSS. These tools make your CSS clean and easy to work with.

JavaScript libraries and frameworks

These frameworks and libraries allow you to save time and do more with less code.

Here are some popular options:

- React
- Angular
- Vue

It is not necessary to learn them all. You should just research which technology is used in your area and start learning that one well.

Databases

It is important for a full stack developer to know how to work with databases. A database in a web application is a place to store and organize your project's data.

There are many types of databases to learn but here are some popular options.

- SQL
- MySQL
- PostgreSQL
- MongoDB

Back-end languages

Languages you can use for back-end development include Java, Python, Node, and PHP. It is not necessary to learn all of these languages, but instead focus on one to get started.

There are also many tech stacks for both front end and backend development. Here are a few popular ones.

- **MEAN stack (MongoDB, Express, Angular and Node)**
- **MERN stack (MongoDB, Express, React and Node)**
- **NodeJS**
- **LAMP stack (Linux, Apache, MySQL, and PHP)**

Testing and Debugging skills

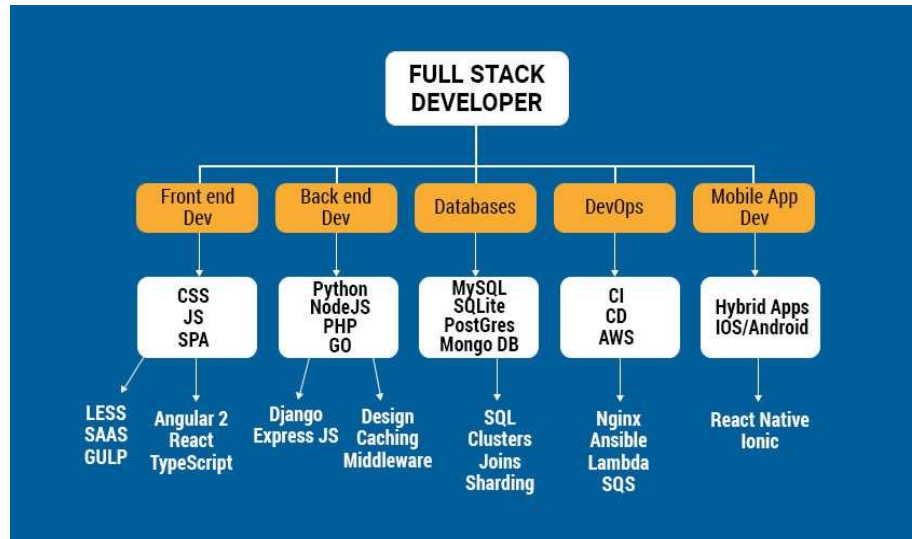
As you are developing your application, there will be errors in your code that need fixing. Debugging is the act of identifying those errors ("bugs") and fixing them. Testing is another important skill to learn. Writing tests for your code is a way to ensure that your code is doing what it is supposed to do.

Version control

Version control is a way to track and manage changes to the project's code. Git is a popular software that you can use to track your code. If you mess up a lot of things in your code, you can use Git to go back to a previous version of your code instead of manually rewriting everything. Learning Git also allows you to collaborate with others on a team and make changes to the same code base from different locations.

Problem Solving

The most important skill for any developer is knowing how to problem solve. Companies and clients are looking for you to provide solutions. It is important to learn how to tackle a problem, break it down into smaller manageable pieces, and troubleshoot the issue in these web applications.



Introduction to Ruby on Rails

Rails is an open-source web application framework for creating database-driven online applications quickly. In other words, compared to other similar tools, Rails is a technology that allows anyone to easily build out strong web apps with minimal work. Rails is a web framework for Ruby that uses Ruby on both the front and back ends to interface with the database and show data to users. Rails leverages the “Model, View, Controller” or MVC style of structure to send information back and forth between users and the database. While Rails was not the first MVC framework, it was one of the first widely used implementations of the organizational technique, and many credit it with popularizing this method of online application structure.

1. Ruby on Rails is a full-stack web app building framework made with the Ruby programming language. Ruby has clean, simple syntax without tons of parenthesis, brackets, or semi-colons. **Github, Shopify, Calendly, Dev.to** were built using Rails.

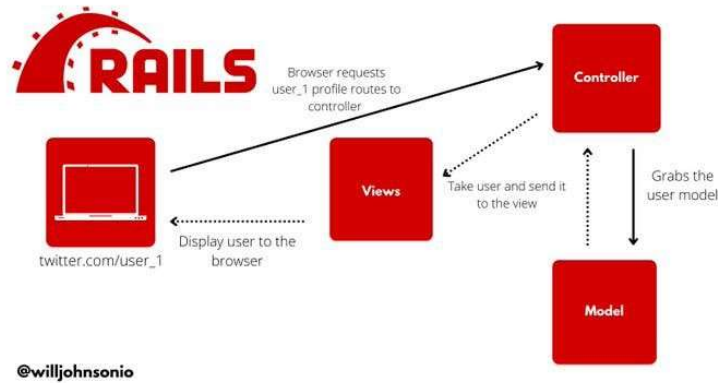
Ruby



@willjohnsonio

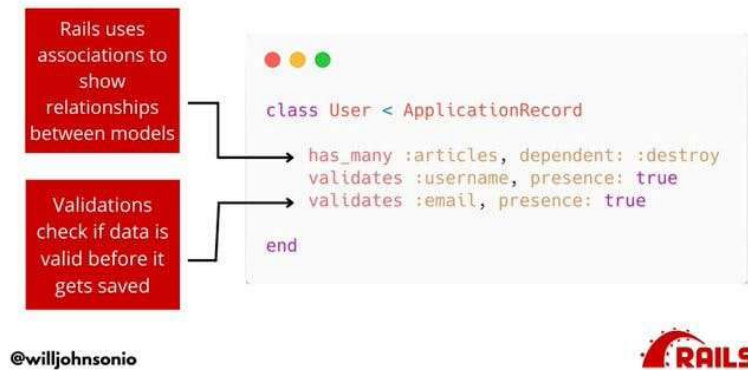


2. Rails use the MVC(Model. View. Controller) programming pattern to organize and separate your code. This makes it easier to know where things are and how they are connected which gives you more time to focus on building features users need



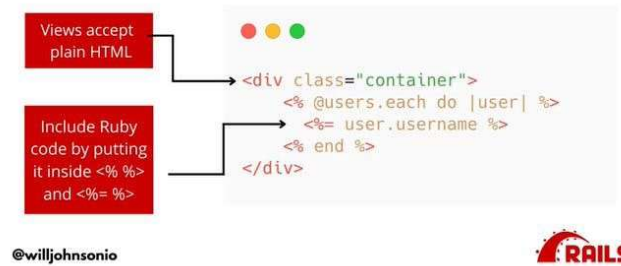
3. Models represent the data in your apps. You define models to make sure any new models match before being saved. You can also associate different models with each other. For example users and their comments and/or articles.

Model



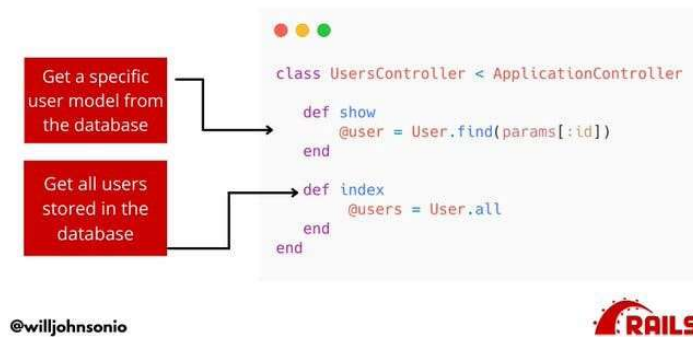
4. Views is what gets displayed to users. It's written just like plain HTML. You can you embedded Ruby(eRB) to put Ruby code inside of your views to display data from the database. You can display multiple users username writing just one line of code!

View

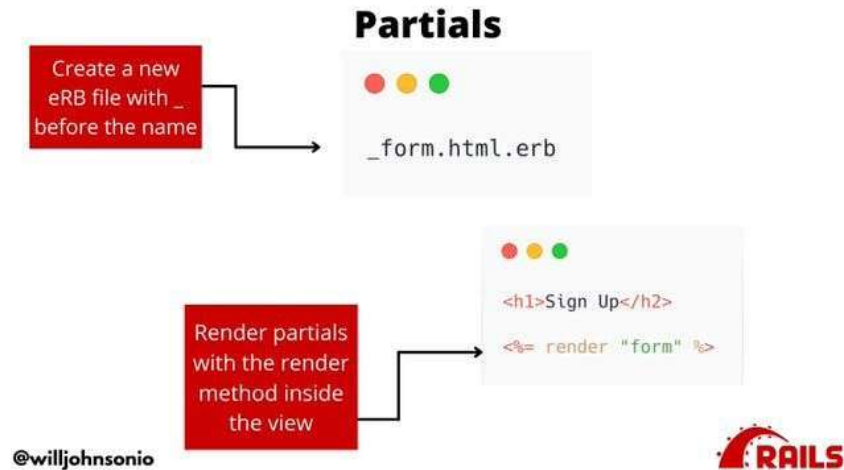


5. Controllers are where you tell your app what to do with your data and what to show to your views. You can grab all users from the database and tell Rails to display them after a user logs-in or click a link to the user index page.

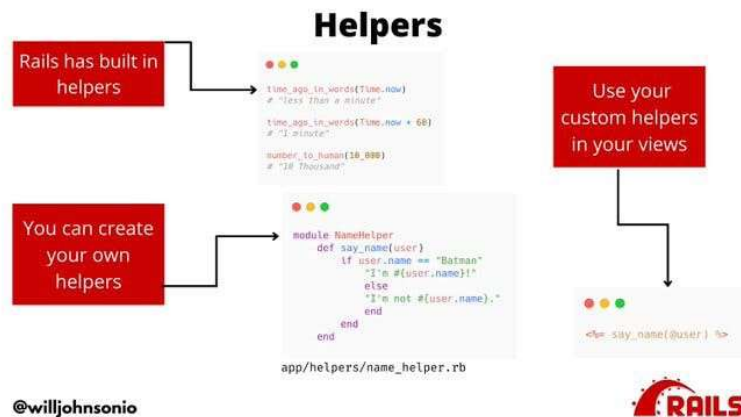
Controller



6. Partials are Views you can share to keep you from re-writing the same code multiple times. A user sign-up and user edit are both forms with emails, passwords, etc. You can write the form once and use it in any view with the render keyword and eRB syntax.

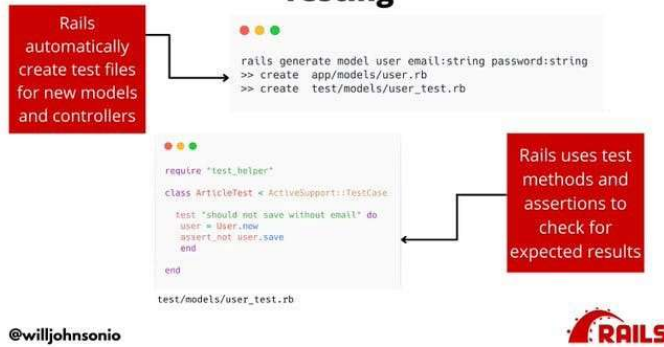


7. Helpers are another way Rails keeps you from rewriting the same code. Rails has built-in helpers for common things like formatting numbers. You can make your own helpers in the app/helpers folder and use them in your views with eRB.



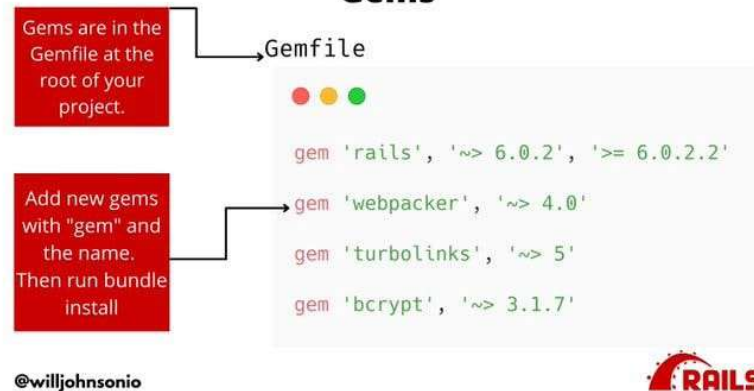
8. Testing is made easy in Rails. A test folder is made with each new Rails app. Each time you create a new model or controller it creates a test file as well. Testing library minitest is built in. Use the test method with assertions to check for expected outcomes.

Testing



9. Gems are like plugins you add to Rails to gain new functionality. Some gems handle complex things like authentication and payments. Gems are located in Gemfile in the root of your Rails app.

Gems



10. Routing tells your application where things go.

Each path is linked to an action inside the controller. The users path shows all users which is linked to the index action inside the controller. Which matches the users index view.

CRUD, Paths, & Actions

HTTP	Path	Controller#Action	What happens
GET	/users	user#index	Display all users
GET	/users/new	user#new	Form to create new user
POST	/users	user#create	Save new user to the database
GET	/users/:id	user#show	Show a certain user
GET	/users/:id/edit	user#edit	Form to edit a certain user
PATCH/PUT	/users/:id	user#update	Update & save a certain user
DELETE	/users/:id	user#destroy	Delete a certain user

@willjohnsonio



Benefits of Ruby on Rails

1. Safe & Secure

When it comes to an online business, safety and security is a major concern. But with RoR, you don't need to worry about it as it has multiple built-in safety measures. Not just this, it supports both behavior-driven development and test-driven development.

2. Cost-Effective

One of the most commonly faced issues by startups is timeline and budget. They plan to finish off the projects in a cost-effective manner and launch on time. This is where RoR comes in, it is an open-source framework that works perfectly on Linux. Moreover, it is an open-source software operating system that is free completely.

3. Easy to Maintain

Ruby on Rails is all about effective coding and reliability. The components that come along with this technology allow the development companies to enhance their overall test codes, coding styles and remove the bugs on very short notice. All of these benefits make Ruby on Rails development easy to manage and maintain. If you are thinking about where you will find the RoR developer, there are multiple waiting in the market.

4. Improves Productivity

The most amazing part of RoR is that it is short and expressive at the same time. And when RoR development company combines it with numerous available third-party libraries, it helps in building features quickly. Hence, it makes the RoR as one of the most productive and easy programming languages for developers.

5. Automated Tested

Obviously! You don't want to indulge yourself in the testing process after development. All you want to do is get done with the project as soon as possible. If this is your aim RoR should be your choice. It is keen on testing & has a test automation process. On top of this, the Ruby on Rails development company tests the entire code before implementation. Not just this, to test the RoR, you do not need a third-party testing tool.

6. Full Stack Framework

Either you are working in the healthcare industry, Oil & Gas, or details with logistics, RoR is a full-stack framework that is suitable for almost all types of industry verticals.



Exercise

1. Prepare a document to show how Hyper Text Transfer Protocol works and illustrate using an example.
2. Prepare a case study on Python web development framework and explain its merit over other frameworks
3. Prepare a document for Ruby on rails explaining its advantages.
4. Prepare a report on web services and its applications in web developments