
CHAPTER 2

Basic web concepts

Dr.Negm Shawky

EMAIL: negmshawky@gmail.com

Contents

- HTTP
- URI
- MIME (Multipurpose Internet Mail Extensions)media types
- HTML

Basics

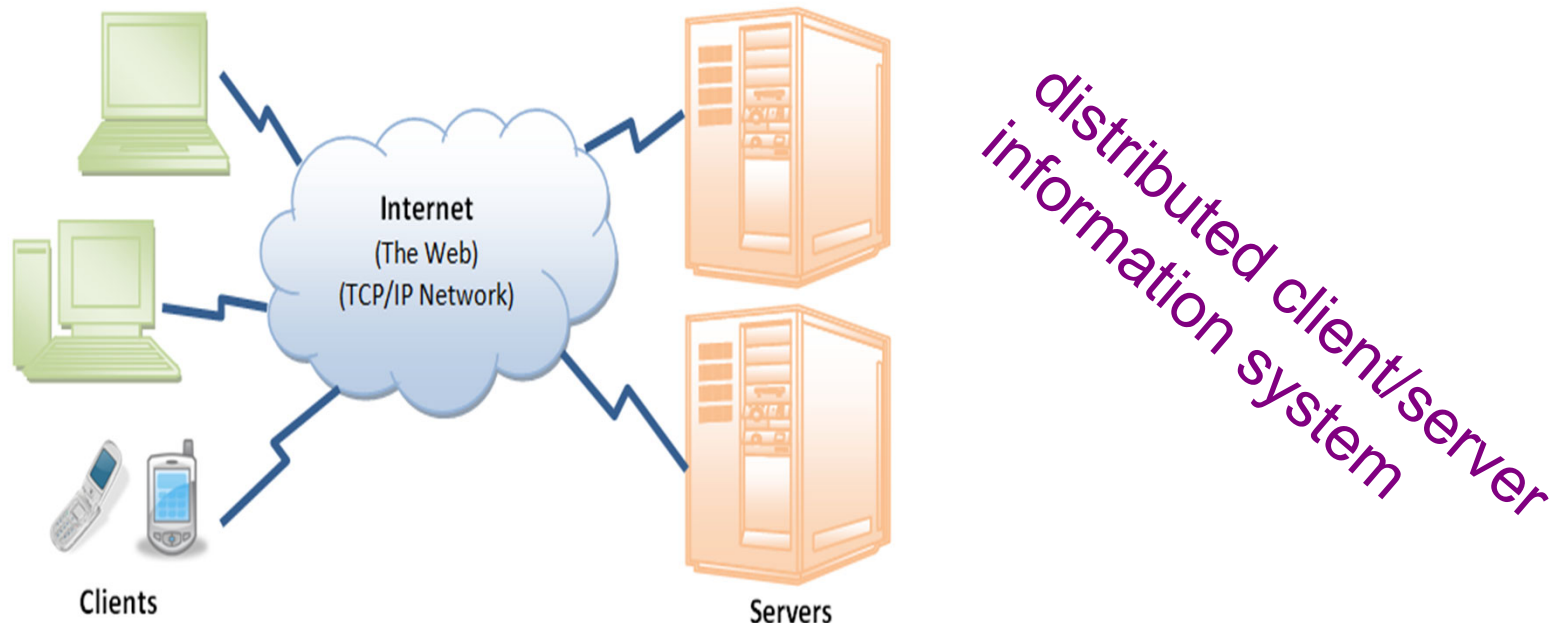
HTTP (HyperText Transfer Protocol)



How you can concentrate in the lecture

The WEB

- ❖ The **Web** is the common name used to refer to the **World Wide Web**, which is a subset of the Internet **consisting of the pages** that can be accessed by a **Web browser**.
- ❖ The **Web** is basically a system of Internet services that support specially formatted documents (text, graphics, audio, and video files). The documents are formatted in a markup language called HTML (*HyperText Markup Language*)



The WEB

- ❖ There are several applications called Web browsers that are used easily for helping us to access the Web; Two of the most popular applications Firefox and Microsoft's Internet Explorer.
- ❖ Many applications are running concurrently over the Web, such as e-mail, file transfer, audio & video streaming, and so on.
- ❖ For communication to take place between the client and the server, a specific application-level protocol such as HTTP, FTP, SMTP, POP, and etc. are used.

HyperText Transfer Protocol (HTTP)

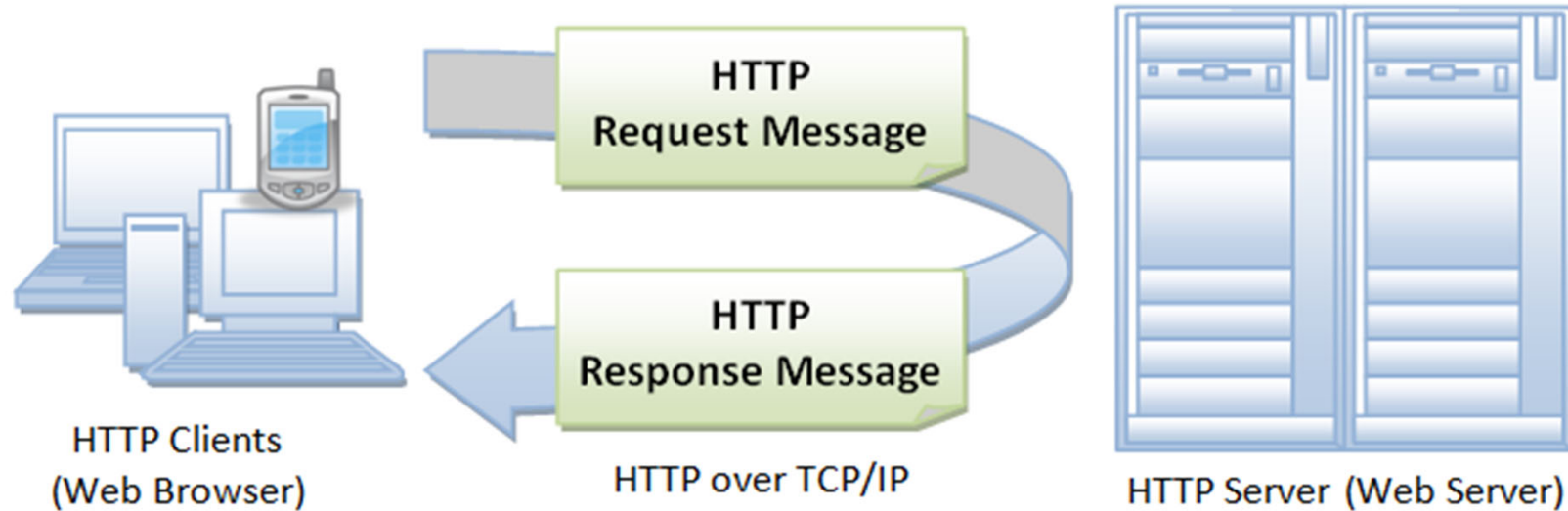
HTTP (Hypertext Transfer Protocol) is perhaps the **most popular application protocol used in the Internet** (or The WEB).

HTTP is the protocol for communicating between web clients and servers.

An HTTP client sends a request message to an HTTP server. The server, in turn, returns a response message.

In other words, HTTP is a *pull protocol*, the client *pulls* information from the server (instead of server *pushes* information down to the client).

HyperText Transfer Protocol (HTTP)



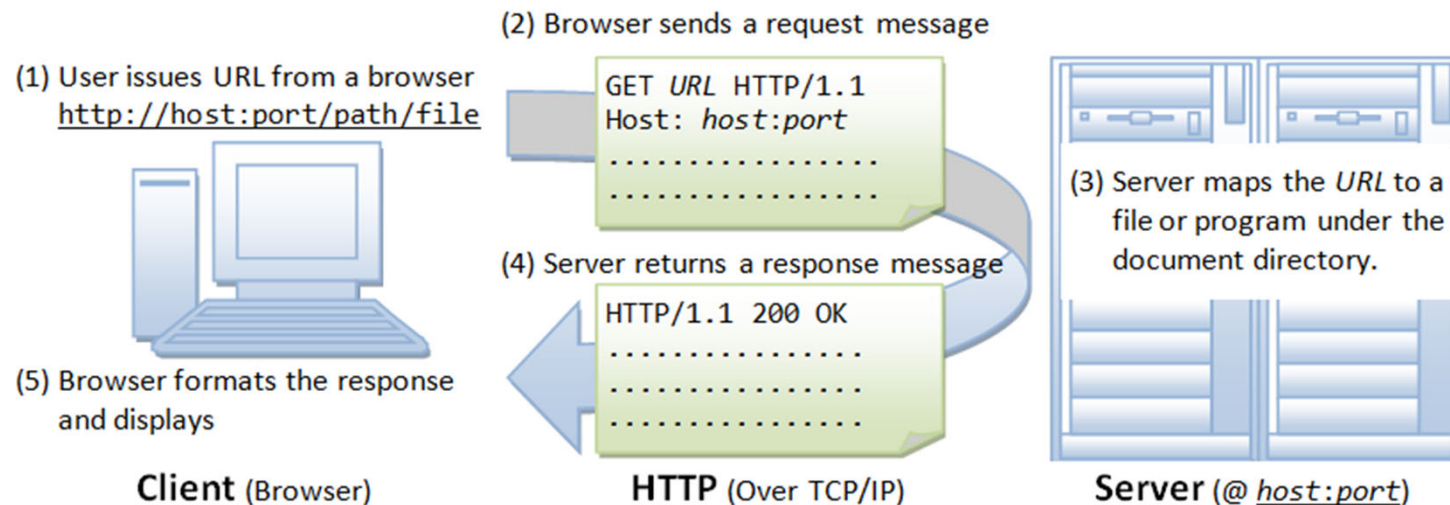
HTTP is a **stateless protocol**. In other words, the current request does not know what has been done in the previous requests.

Browser

- When you issue a **URL** from your browser to get a **web resource using HTTP**, e.g. you write on the browser bar <http://www.lifewire.com/index.html>,

the browser converts the URL into a *request message* and sends it to the HTTP server.

- The HTTP server interprets the request message, and returns you an appropriate response message, which is either the resource you requested or an error message.



URIs

- ❖ A **uniform resource identifier** (URI) is a string of characters in a particular syntax that identifies a **resource**:
 - A **file** on a server,
 - An **email** address,
 - A news **message**,...

URN and URL

- ❖ There are two types of URIs:
 - Uniform resource locators (URLs)
 - Uniform resource names (URNs).
- ❖ A URL is a way to determine a particular resource on the internet at a **particular location**
- ❖ URL specifies **the location** of an identified resource that is available and also **the mechanism** for retrieving it (how the resource can be obtained).

URLs

To identify the location of a resource on the internet using the URL .

This require of specifying **three main parts** :

- The **protocol** used to access a server (e.g.,FTP,HTTP),
- The **name of the server**,
- The **location of a file** on that server.

Uniform Resource Locator (URL)

- ❖ A URL (Uniform Resource Locator) that is used to uniquely identify a resource over the web has the following syntax:

Protocol://hostname:port/path-and-file-name

There are 4 parts in a URL:

Protocol: The application-level protocol used by the client and server, e.g., HTTP, FTP, and telnet.

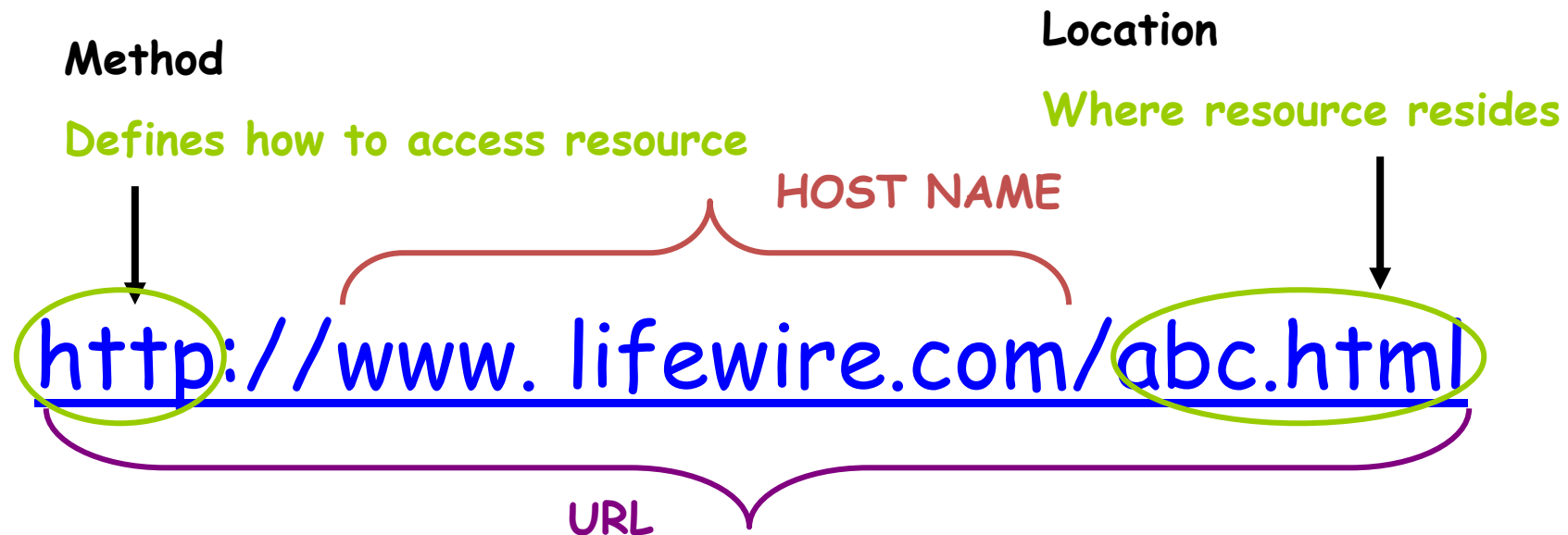
Hostname: The DNS domain name (e.g., www.lifewire.com) or IP address (e.g., 192.128.1.2) of the server.

Port: The TCP port number that the server is listening for incoming requests from the clients.

Path-and-file-name: The name and location of the requested resource, under the server document base directory.

Uniform Resource Locator (URL)

- ❖ URL includes protocol e.g. http://, ftp:// along with location to identify resource e.g. <http://www.lifewire.com/abc.html>.



Uniform Resource Locator (URL)

For example, in the URL

<http://www.lifewire.com/docs/index.html>,

- The communication **protocol** is **HTTP**;
- The **hostname** is **www.lifewire.com**.
- The **port number was not specified in the URL**, and takes on the default number, which is TCP port 80 for HTTP.
- The **path and file name** for the resource to be located is **"/docs/index.html"**.

Other examples of URL are:

<ftp://www.ftp.org/docs/test.txt>

<telnet://www.lifewire.com/>

The general form of a URL

❖ URL has the **general form** of the following syntax

protocol://username:password@hostname:port/path/filename?query#fragment

- The **protocol** is another word for what was called the **scheme** of the URI.
- The **username:password** is an **optional** username and password for the server.
- The **hostname** part of a URL is the name of the server that provides the resource you want.
- The **port number** is also **optional**.

The general form of a URL

protocol://username:password@hostname:port/path/filename?query#fragment

- The **path** points to a particular **directory** on the specified server.
- The **filename** points to a **particular file** in the directory specified by the path.
- The **query** string provides additional arguments for the server. It contains form data for input to programs running on the server.

The general form of a URL

Query Strings/Parameters

A simple URL with a query string might look like:

<http://www.example.com?search=ruby&results=10>

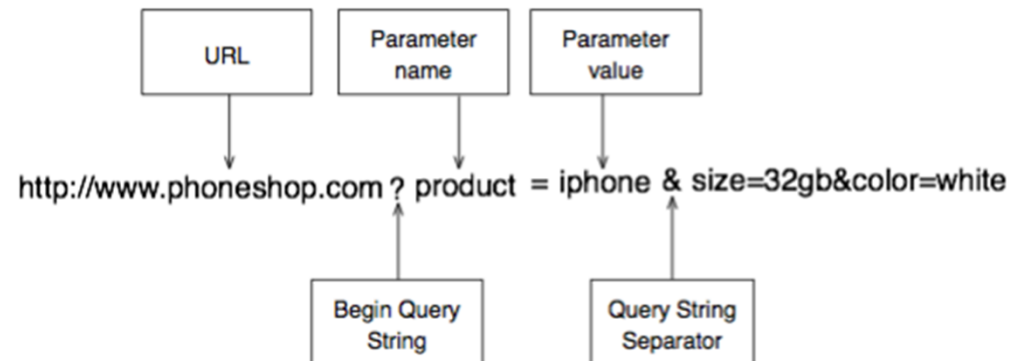
Query String Component	Description
?	This is a reserved character that marks the start of the query string
search=ruby	This is a parameter name/value pair .
&	This is a reserved character, when adding more parameters to the query string.
results=10	This is also a parameter name/value pair .

The general form of a URL

Query Strings/Parameters

Another example:

<http://www.phoneshop.com?product=iphone&size=32gb&color=white>



In the above example, name/value pairs in the form of `product=iphone`, `size=32gb` and `color=white` are passed to the server from the URL.

This is asking the `www.phoneshop.com` server to retrieve the resource file specified with a certain information (`product =iphone`, `size =32gb` and `color =white`).

The general form of a URL

protocol://username:password@hostname:port/path/filename?query#fragment

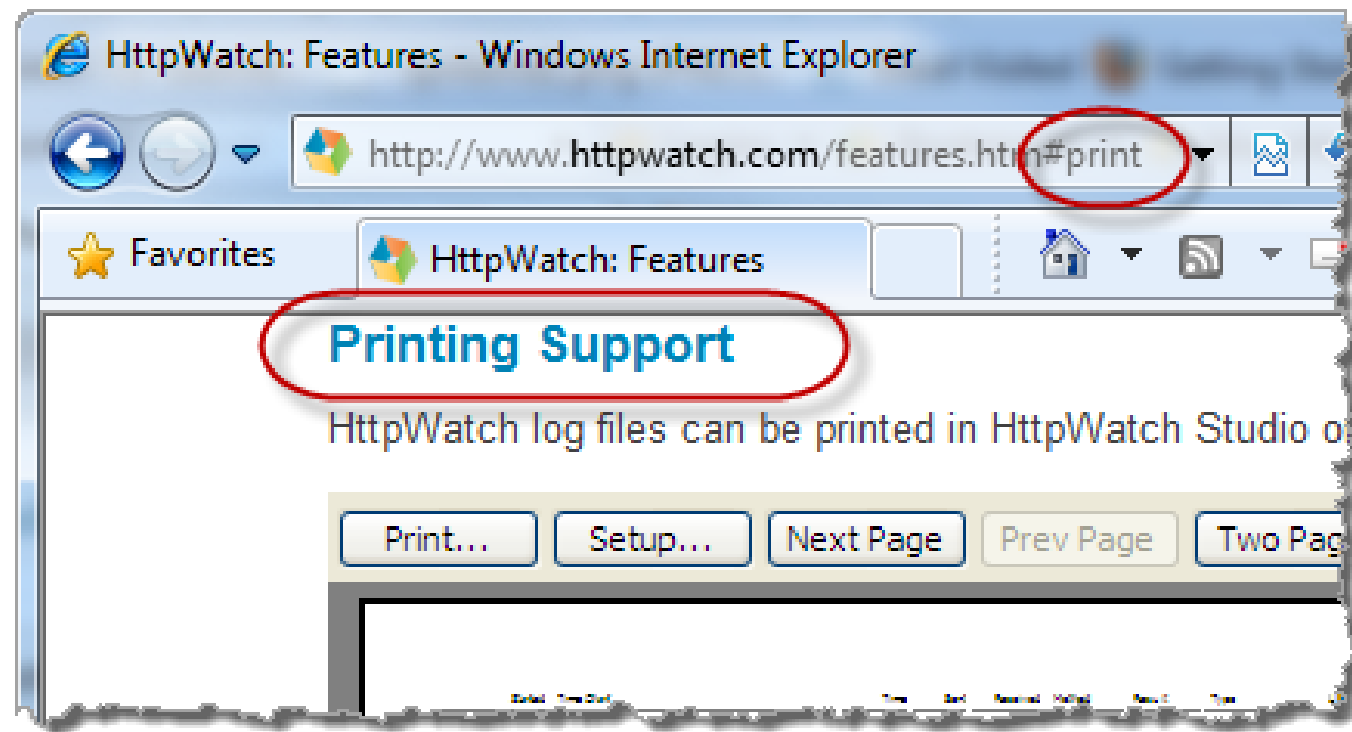
- Finally, the **fragment** is a short string of characters that refers to a resource that is subordinate to the primary resource i.e. (A **fragment** refers to a specific section of a web page and **is used as a reference to a particular part of the remote resource.**)
- A fragment usually appears at the end of a URL and begins with a hash (**#**) character followed by an identifier.
- The **fragment identifier** is used to specify a location within the resource.

The general form of a URL

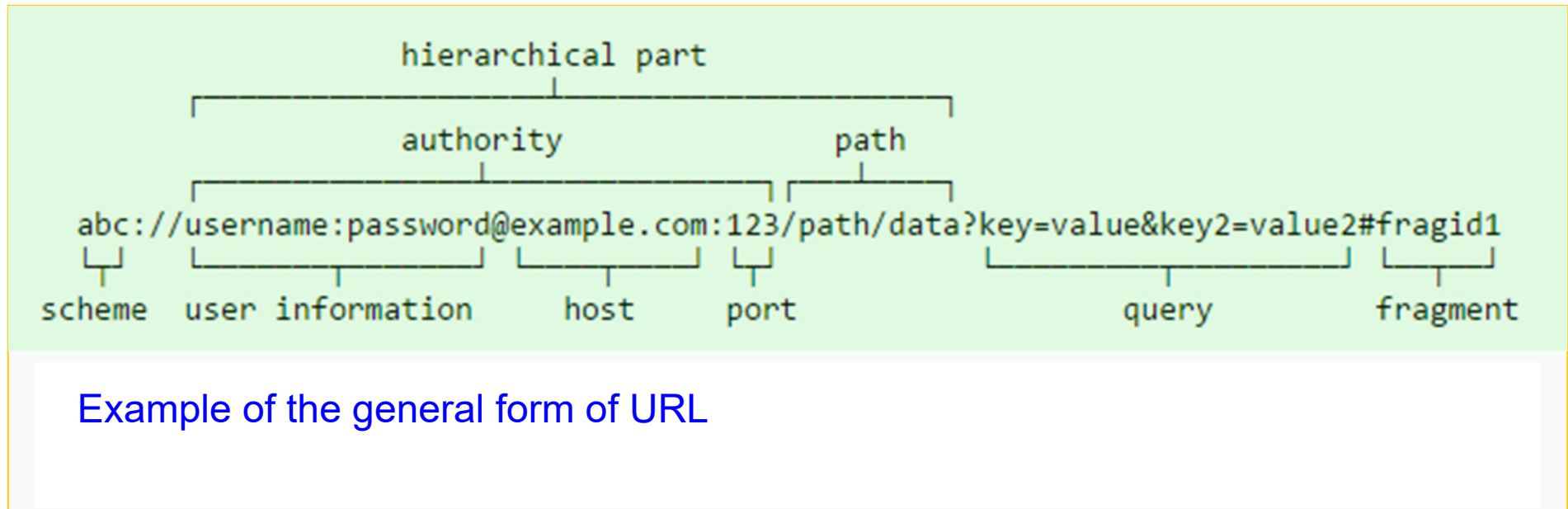
`http://www.httpwatch.com/features.htm#print`

HTML document
to download

Location within
document



The general form of a URL



URN

- ❖ A URN is a **name for a particular resource** but without implying its location or how to access it.
- ❖ URNs always **start with** the prefix **urn**
- ❖ A URN is like a **city's name**, while a URL is similar to a **city's latitude and longitude**.
- ❖ URN are the easy way to identify a resource without including a protocol and a location .

URNs

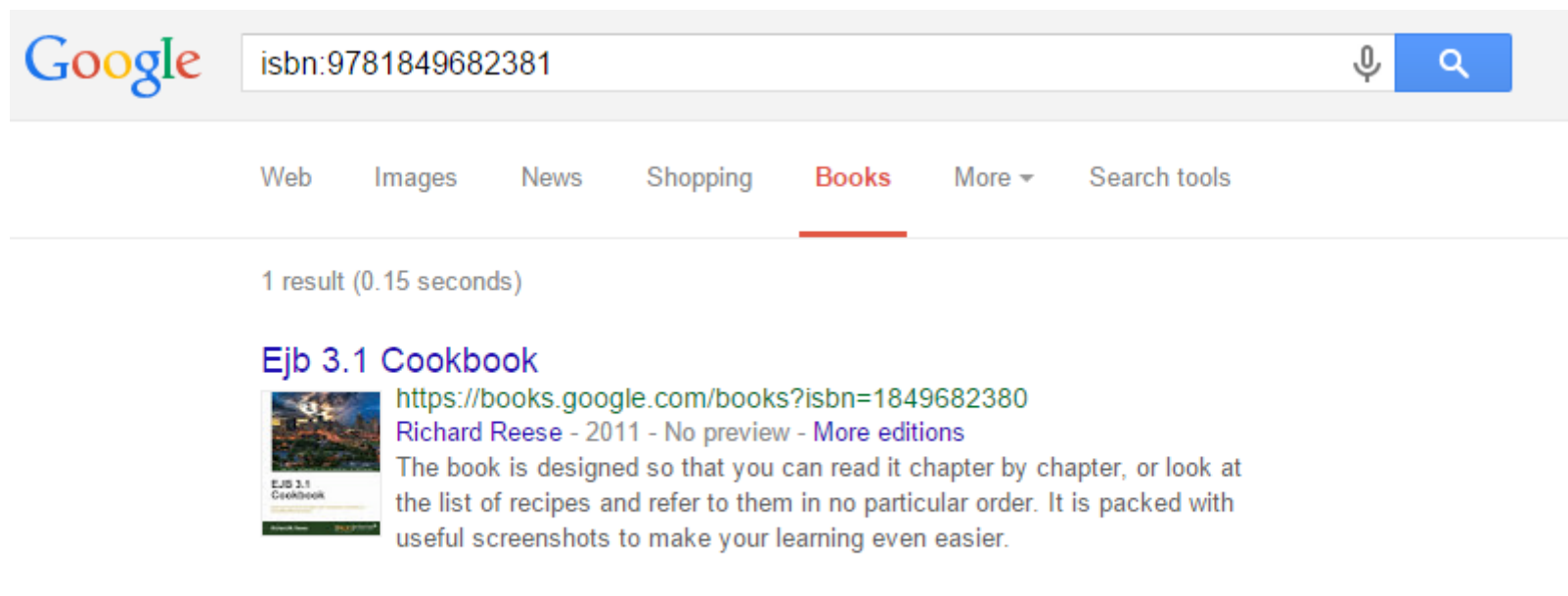
- ❖ A URN has the general form:

`urn: namespace:resource_name`

- The `namespace` is the name of a collection of certain kinds of `resources` maintained by some authority.
- The `resource_name` is the name of a resource within that collection.

Example

- The URN `urn:ISBN:9781849682381` identifies a resource in the `ISBN namespace` with the `9781849682381` i.e (identifies a book by its ISBN number).
- In this example Urn is used `identifier` to identify a book by its `ISBN number`.



Relative URLs

- ❖ A relative URL is a **type of URL** that specifies the location of a resource relative to the current page.
- ❖ It does not include the domain name or protocol, and **it is used to link to resources within the same website.**
- ❖ Relative URLs are **useful when you want to link to a resource on the same website without having to specify the full URL.**

Relative URLs

- ❖ In contrast, a **completely** specified URL is called an **absolute** URL.
- ❖ Of course, it's still important to understand **how relative links work**, so read on...
you would use the **following code** to create a link in HTML:

`Click Me`

linkhere.html would be the page **you want to link to it**, and **Click Me** would be the name related to link that the page will be displayed.

In the example above, we used a relative path.

Relative URLs(Absolute vs. Relative Paths/Links)

Relative Paths

index.html

/graphics/image.png

/help/articles/how-do-i-set-up-a-webpage.html

Absolute Paths

<http://www.mysite.com/index.html>

<http://www.mysite.com/graphics/image.png>

<http://www.mysite.com/help/articles/how-do-i-set-up-a-webpage.html>

The first difference you'll notice between the two different types of links is that **absolute paths** always include the **domain name** of the website, including **http://www.**, whereas **relative links** only point to a **file or a file path**.

When the **users click** a relative link, the browser take them to that **location on the same current site**.

Relative URLs(Absolute vs. Relative Paths/Links)

How about this example? Let's say we have domain <http://www.website.com> had a subfolder called **pictures**. Inside the pictures folder is a file called **pictures.html**. The full path to this page would be:

["http://www.website.com/pictures/pictures.html"](http://www.website.com/pictures/pictures.html).

Let's say in this **pictures.html** file, we have a link:

`More Pictures`

If someone clicked that, this similar to If you said

<http://www.website.com/pictures/morepictures.html>,

In this case both files are saved in the **pictures** subfolder.

Example of Relative URLs

❖ Suppose that while browsing the page:

<http://www.website.com/pictures/pictures.html> you click on this *hyperlink*: `` in that page

The browser cuts `pictures.html` off the end of `http://www.website.com/pictures/pictures.html` to get `http://www.website.com/pictures/`. Then it attaches `morepictures.htm` onto the end of `http://www.website.com/pictures/` to get

`http://www.website.com/pictures/morepictures.html`. Finally, it loads that document.

HTTP Protocol

As mentioned, when you **enter a URL** in the address box of the browser, the browser **translates the URL** into a request message according to the specified protocol; and **sends the request** message to the server.

For example, the browser translated the URL <http://www.lifewire.com/doc/index.html> into the following **request message**:

```
GET /docs/index.html HTTP/1.1
Host: www.lifewire.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
(blank line)
```

HTTP Protocol

- When this request message reaches the server, the server can take either one of these actions:
- The server interprets the request received, maps the request into a *file* under the server's document directory, and returns the file requested to the client.
- The server interprets the request received, maps the request into a *program* kept in the server, executes the program, and returns the output of the program to the client.
- When the request cannot be satisfied, the server returns an error message.

HTTP Protocol

An example of the HTTP **response message** is as shown:

HTTP/1.1 200 OK

Date: Sun, 18 Oct 2016 08:56:53 GMT

Server: Apache/2.2.14 (Win32)

Last-Modified: Sat, 20 Nov 2012 07:16:26 GMT

ETag: "10000000565a5-2c-3e94b66c2e680"

Accept-Ranges: bytes

Content-Length: 44

Connection: close

Content-Type: text/html

X-Pad: avoid browser bug

<html><body><h1>It works!</h1></body></html>

HTTP Protocol

□ The browser receives the response message, interprets the message and displays the contents of the message on the browser's window according to the media type of the response (as in the Content-Type response header).



□ Common media type include "text/plain", "text/html", "image/gif", "image/jpeg", "audio/mpeg", "video/mpeg", "application/msword", and "application/pdf".

HTTP Protocol

- ❑ In its idling state, an HTTP server **does nothing** but listening to the **IP address(es) and port(s)** specified in the configuration for incoming request.
- ❑ When a request arrives, the server **analyzes** the message header, **applies rules** specified in the configuration, and **takes** the appropriate **action**.

HTTP over TCP/IP

- ❑ You could also run multiple different applications in the same machine on different port numbers.
- What happens when a client issues a URL without explicitly stating the port number, e.g.,
<http://www.lifewire.com/docs/index.html>,
the browser will connect to the default port number 80 of the host www.lifewire.com.
- When you need to specify the port number in the URL, e.g.
<http://www.lifewire.com:8000/docs/index.html>
in this case the server is listening at port 8000 and not the default port 80.
- ❑ **In brief**, to communicate over TCP/IP, you need to know
(a) IP address or hostname, (b) Port number.


HTTP Specifications

- ❖ There are currently three versions of HTTP, namely,
 - HTTP/1.0
 - HTTP/1.1
 - HTTP/2.0
- ❖ The **original version**, HTTP/0.9 (1991), written by Tim Berners-Lee, is a simple protocol for **transferring raw data** across the Internet.




Tim Berners-Lee

HTTP Specifications

- ❖ HTTP/1.0 (1996) , improved the protocol by allowing MIME-like messages.

- ❖ HTTP 1.0 opens a new connection for every request.
- ❖ The primary improvement in HTTP 1.1 is connection reuse. These features were provided in HTTP/1.1 (1999) .
- ❖ HTTP 1.1 allows a browser to send many different requests over a single connection; the connection remains open until it is explicitly closed.
- ❖ The requests and responses are all asynchronous (a browser doesn't need to wait for a response to its first request before sending a second or a third).

HTTP Specifications

- ❖ HTTP/2.0 was officially released in 2015, and is focused on improving the protocol performance. 
- ❖ HTTP 2.0 executes a compression automatically of requests and responses.
- ❖ Add a Connection reset: when a closing of connection between a server and a client for some reason, thus immediately opening a new one.

MIME Media Types (Multipurpose Internet Mail Extensions)

- ❖ MIME is an extension of the original internet e-mail protocol, is used for sending multimedia data through Internet email. and lets people to exchange different kinds of data file on the internet: audio, video, images, application programs and other kinds.
- ❖ describe a file's contents so that a client software can recognize the difference between different kinds of data.

MIME Media Types

(Multipurpose Internet Mail Extensions)

- ❖ For example, a web browser uses MIME to tell whether a file is a GIF image or a text file.
- Servers insert the MIME header at the beginning of any web transmission.
- Clients use this header to select an appropriate “player” application for the type of data that the header indicates.

MIME Media Types (Multipurpose Internet Mail Extensions)

❖ Type and subtype:

MIME supports more than **100** predefined types of content.

Content types are classified at two levels: a **type** and a **subtype**.

The **type** shows very generally **what kind of data is contained**: is it a picture, text, or movie?

The **subtype** identifies the **specific type of data**: GIF image, JPEG image, TIFF image.

MIME Media Types (Multipurpose Internet Mail Extensions)

Example:

- HTML's **content type** is **text/html**; the type is text, and the subtype is html.
- The **content type** for a **GIF image** is **image/gif**; the type is image, and the subtype is gif.
- ❖ **Web servers** use MIME to identify the kind of data **they're sending**.
- ❖ **Web clients** use MIME to identify the kind of data **they're willing to accept**.

END

