

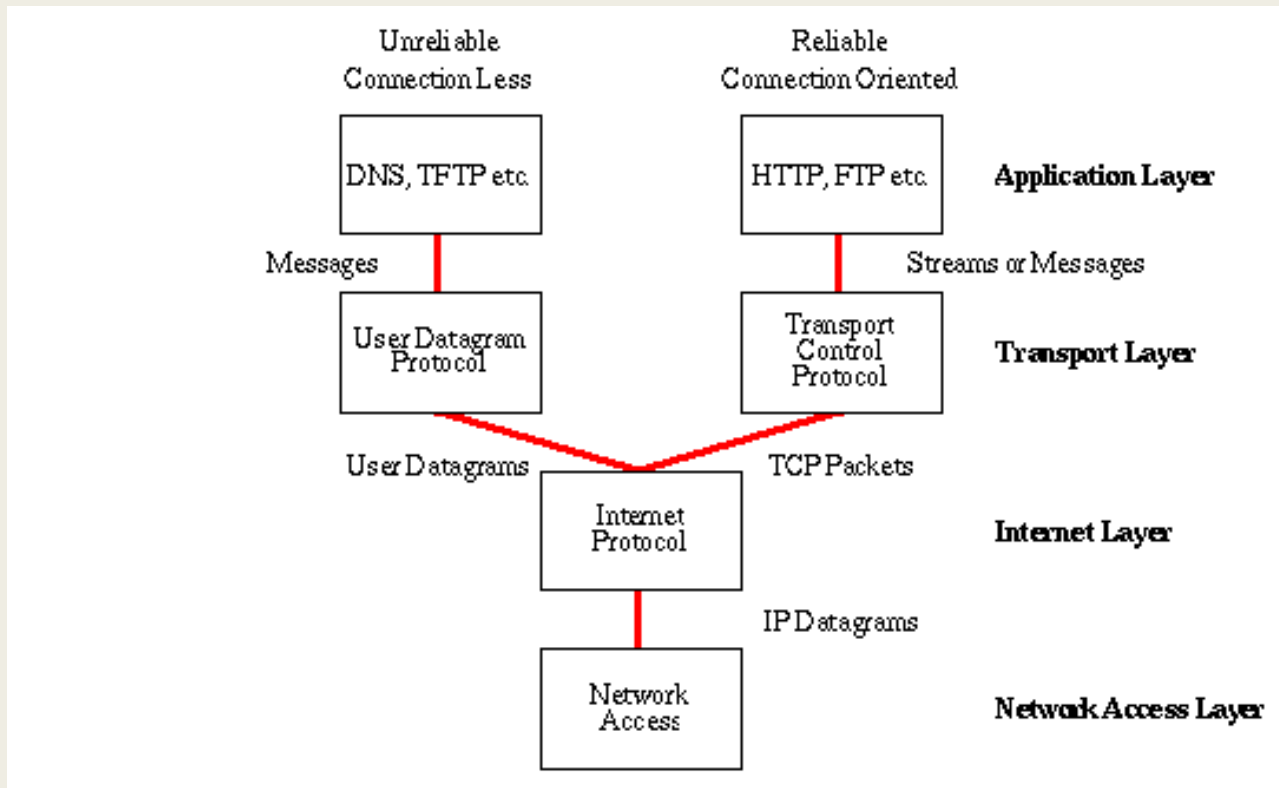
The image features two large, thick, black L-shaped brackets. One is positioned on the left side, with its vertical bar extending downwards and its horizontal bar extending to the right. The other is on the right side, with its vertical bar extending upwards and its horizontal bar extending to the left. These brackets frame the central text.

INTERNET BASICS

Context and a Little History

- The *internet* is a collection of computers connected by a hierarchical network.
- Messages are sent from one computer on the internet to another computer. Each message is broken up into sequenced datagrams and then reassembled on the other end.
- An IP address specifies the location of a computer(s). Originally 32 bits, it has been expanded to 128 bits for IPv6.
 - Ex: 17.255.8.103

TCP/IP



<https://www.w3.org/People/Frystyk/thesis/TcpIp.html>

Base Protocols

- Some low level protocols.
 - *TCP/IP – Transmission Control Protocol/Internet Protocol*
 - *UDP – User Datagram Protocol*

Sending data

- Messages get sent to a computer at the given address on a *port*.
- Services are assigned to certain ports and servers listen for incoming messages and respond as appropriate.

Some Common Protocols

- FTP/S – File Transfer Protocol/Secure
- SMTP – Simple Mail Transfer Protocol
- IMAP – Internet Message Access Protocol
- LDAP – Lightweight Directory Access Protocol
(distributed access to information services, user info)
- SSH – Secure Shell (Allow encrypted communication for remote command line access)

DNS

- Remembering an IP address is hard, so we want to have a way of specifying a location that is easy to remember. A Domain Name Server (DNS) takes an easy to remember *domain name* like www.google.com and returns the IP address. The program that requested the IP address will remember (cache) the address.
- Allows changing the IP address invisibly, but a potential security issue. If someone wanted to hijack a service they could do so by changing the associated IP address.
- Distributed for robustness,

URL

- Uniform Resource Locator combines
 - *Service*
 - *Domain Name*
 - *Path to resource*

Example:

<https://www.w3.org/People/Frystyk/thesis/Tcplp.html>

History

In the 1980s, the Internet was in existence and had moved beyond academia and military uses. One of the killer apps of that era was Email. An average person could send mail to another person without ever setting pen to paper. The details of that interaction were reasonably well hidden behind a graphical or text based interface. The average user really didn't have to know that there was a well-defined infrastructure called SMTP (Simple Mail Transfer Protocol) that managed the process. Similarly, there were a number of different server based applications like bulletin boards and games that would allow users to interact with each other without having to worry about the details.

History

A less commonly used protocol was FTP that supported getting or sending files. One primary reason for this was that a home user would typically connect to the Internet using a phone line and had significant restrictions on the speed of their data transfers. Businesses and educational institutes typically did not face that kind of slowdown because they could afford the expense of building a dedicated higher speed network. The average person was stuck and really couldn't do much. An Email message is usually small and can be sent over a phone line in seconds. Images are larger and would require minutes. Sound files could take an hour and video just wasn't an option. So the typical user, didn't have much call for sending and receiving files beyond mail.

History – The Problem

It was a different story though for academic researchers. There were large cooperative scientific ventures that were collecting large amounts of information. There were many files of raw data that needed to be organized. There were files that resulted from the processing of the data. And there were images, because a good image communicates results better than a pile of numbers. There were a couple problems that needed to be addressed. First, the data needed to be shared across long distances. A scientist could be part of a research group that was awarded time on an experimental machine thousands of miles away. FTP was often used to move data around, but it wasn't very convenient. Second, all of the files were stored without a lot of contextual information. There needed to be a way of exposing the data with context.

History – The Solution

HTML – Hypertext markup language has hyper links. From the earliest days, books have primarily been a linear form. While you can jump around in a book, there is typically a preferred order. With the advent of computers, the notion arose that we could structure things differently. We could have chunks of information (pages) with links between them. At any time, the user could explore by jumping to another page and then go back once they were done. This was perfect for organizing large collections of data. We can have descriptive pages with links off to images or raw data for easy access.

Browsers – HTML allows you to specify the format of information in a way that is independent of how it is being displayed. Remember, the information needed to be shared with many different groups. Each group was likely to have different computers with different display capabilities. So the HTML for a page gives instructions for how the information on a page is intended to look and a program called a browser renders the page on your screen. Today, we are much more finicky about making sure that web pages look the same from computer to computer and from browser to browser, but close enough was the early goal.

HTTP – A protocol for requesting and sending an HTML file, because there are protocols for everything.

Today

Why is it popular now? - At a fundamental level, simple webpages (text without much graphics or sound) could be accommodated at the telephone speeds. As time has gone on, the high-speed Internet connections that were only affordably available to larger users by businesses and educational institutions have become available to ordinary users.

Changes to the protocol and other methods now also allow web pages to go beyond their original static nature. This allows a wide range of services to be supported including streaming of videos.

Performance

Time required to send a file containing 3.5 million bytes of information (Roughly, the size of the bible, or three 600 by 600 pixel images, or a 3 minute mp3 song.)

- 1962 – Dial up telephone modem - 300 bps (32 hours)
- 1984 – Dial up telephone modem – 2400 bps (4 hours)
- 1994 – Dial up telephone modem – 19.2 Kbps (1/2 hour)
- 1998 – Dial up telephone modem – 56 Kbps (10 minutes)
- 1990s – DSL (a specialized phone line) – 512 Kbps (1 minute)
- 2000s – Cable modem – 1 Mbps (30 seconds)
- 2010s – Cable modem - 10 Mbps (3 seconds)
- 2017 - Cable modem – 50 Mbps (1/2 second)
- 2021 - Fiber Optic– 1 Gbps (0.03 seconds)

As a handy comparison, 3G wireless is about 1 Mbps and 4G wireless is about 10 Mbps.

Historical Note:

- Tim Berners-Lee at CERN (European Organization for Nuclear Research) is credited with creating the first browser/website in 1989.
- Another major step was the creation of the Mosaic browser in 1993 at the NCSA (National Center for Supercomputing Applications) located on the University of Illinois campus at Champaign-Urbana.



HTML INTRODUCTION

XML/HTML

- Both are markup languages and are used for the transfer/specification of information.
- Tag pairs create containers that can hold data (including more tag pairs).
 - `<tagname> content </tagname>`
 - *Tag names are not case sensitive, but safer to always use lower case.*
 - *Some tag pairs may work if closing tag is missing. Safer to always include*
 - *Some tags don't have content and do not need closing tag.*
- XML allows for the dynamic creation of new tags whereas HTML is fixed.
- Most recent version of HTML is HTML5 and is supported by all major browsers.
 - *Read about the differences here: [HTML vs HTML5](#)*

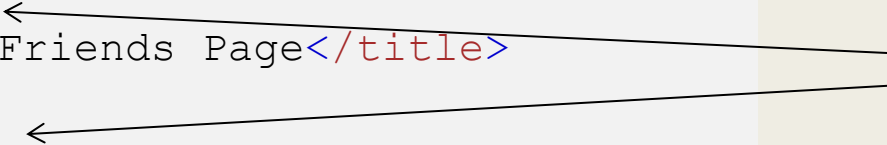
HTML Example

```
<!DOCTYPE html>
<html>
<head>
<title>Friends Page</title>
</head>
<body>

<h1>Justin</h1>
<p>
Justin is my best friend.
We both like rock and roll.
</p>

</body>
</html>
```

Matched tag pair



HTML Formatting

```
<!DOCTYPE html>  
<html>
```

Comment with type

```
  <head>  
    <title>Friends Page</title>  
  </head>
```

Meta Information

```
  <body>  
    <h1>Justin</h1>  
    <p>  
      Justin is my best friend.  
      We both like rock and  
roll.  
    </p>  
  </body>
```

The contents

```
</html>
```

Rendering HTML

- A Browser Chrome/FireFox/Safari can render HTML.
 - *Can open the browser directly on a file*
 - *Can create a server that will respond to a request.*
- Right click on a web page to view or inspect the source.
- Can change the HTML being displayed.
- Next page shows Chrome inspecting the file sample.html. The path includes spaces in the directory names which need to be recoded in the URL as %20

Justin

Justin is my best friend. We both like rock and roll.

Elements Console Sources Network Performance Memory Application >> ⚙️ ⋮ ✕

```
<!DOCTYPE html>
<html>
  <head>
    <title>Friends Page</title>
  </head>
  <body>
    <h1>Justin</h1>
    <p>
      Justin is my best friend.
      We both like rock and roll.
    </p>
  </body>
</html>
```

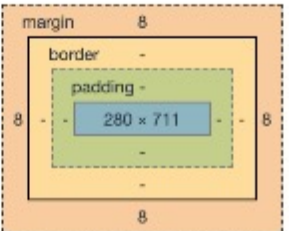
Styles Computed Layout Event Listeners >>

Filter :hov .cls + ⌵

element.style { }

body { user agent stylesheet

```
display: block;
margin: 8px;
}
```



html body


Console What's New ✕

Highlights from the Chrome 87 update

[New CSS Grid debugging tools](#)
Debug and inspect CSS Grid with the new CSS Grid debugging tools.

[New WebAuthn tab](#)
Emulate authenticators and debug the Web Authentication API with the new WebAuthn tab.

[Move tools between top and bottom panel](#)



HTML – Text Size

- Headings

- `<h1> Largest Heading </h1>`
- `<h6> Smallest Heading </h6>`

HTML – Text Style

- Italics/Bold

- `<i> Italicized content</i>`
- ` Bold content`

- Emphasis/Strong

- ` Italicized content usually`
- ` Bold content usually`

- Screen readers may pronounce text with vocal stress.

- Other tags are available, but maximal control of style is provided by CSS.

HTML – Attributes

- Attribute values can be specified inside the opening tag
 - `<tagname attribute="value"> Content<\tagname>`
- For safety, quote all attribute values. Double quotes are typically used, but single quotes work as well.
- Example Hyperlink
 - ` the link text`
- Example Image (No element)
 - ``

HTML – Images

- src – the image to use. Should use relative paths for safety.
- width – how wide is the image in pixels
- height – how high is the image in pixels
- alt – text to display instead of the image. Needed for ADA compliance.

- Example

```
<img src=\images\ball.jpg  
      height="200" width="100"  
      alt="a picture of a ball" >
```

Safer to quote the src value

HTML – Attributes

- Lang in html tag.
- Title in element tags gives a tooltip

HTML – Paragraph

- `<p> Content </p>`
 - *Content can be spread across lines.*
 - *Paragraph content is ragged right and fills the space.*
 - *Multiple spaces get squished.*
- `
`
 - *Signal a line break*
 - *Does not start a new paragraph*

HTML – Forcing Formatting

- `<pre> Content </pre>`
 - *Pre formatted content*
 - *Lines are preserved*
 - *Spaces are preserved.*
 - *Typically displayed in a mono font like courier*

HTML – Quote

- `<blockquote cite="referenceURL"> quoted text </blockquote >`
 - *Indented typically*
 - *Has cite attribute*
- `<q> quoted text </q>`
 - *Quoted inline*

HTML – Lists

■ Unordered

```
<ul>
```

```
    <li> Content </li>
```

```
    <li> Content </li>
```

```
    <li> Content </li>
```

```
<\ul>
```

■ Ordered: ...

HTML –Nested List Example

```
<ul>
  <li> Item 1 </li>
  <li> Item 2 </li>
  <li> Item 3
    <ul>
      <li> nested under 3 part 1 </li>
      <li> nested under 3 part 2 </li>
      <li> nested under 3 part 3 </li>
    </ul>
  </li>
</ul>
```

Compare this to markdown

HTML –Tables

```
<table>
<tr>
    <th> Header 1 </th>
    <th> Header 2 </th>
</tr>
<tr>
    <td> Content </td>
    <td> Content </td>
</tr>
</table>
```


HTML – Horizontal Rule

- `<hr>`



WEB SITE



SERVERS

Basic Servers (more later)

- Precondition: A live web server listening at the appropriate port.
- A request comes in as a URL.
- The web server looks at request and determines the actual location.
- The root directory of the web pages is most likely deep in the actual hierarchy.
 - *Allows for security and the ability to switch in a new set of web pages.*

Basic Servers (more later)

- If the URL ends at a folder, look for index.html and return that.
- If the URL ends at an html file, return that.

Dynamic HTML

- Add code to a web page that is executed by the browser and changes what is displayed (client side).
 - *May have safety concerns*
 - *Leads to wanting to cache data in the browser for later use. (Cookies and the like.)*
- Add code that is executed by the server and creates html code.
 - *May completely generate the HTML, but often splices in bits.*
 - *Security is still a concern*

Running locally

- If you have a web page on your local computer, you can open it with a browser and it will be rendered.
- As you change the code, you can refresh the browser and see the result.
- You can inspect and change the HTML with the browser, but that will not be reflected back to the local file.

Running on a Server

- As long as you restrict your self to web pages that are only client side, and your project is a repo on GitHub, there is a setting where GitHub will act as a server for the repo.
- Under settings – Select the branch and then the URL is displayed.

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Source

GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. [Learn more.](#)

None ▾

Save

Theme Chooser

Select a theme to publish your site with a Jekyll theme using the gh-pages branch. [Learn more.](#)

Choose a theme

Running on a Server

- There are other web hosting services that you can use.
 - *AWS (Amazon)*
 - *Azure (Microsoft)*
 - *Firebase (Google, also designed to work with Android Apps)*
 - *Heroku*
 - *... and many more*
- Watch for payment issues. If you have an educational account, what happens when it runs out. If they ask for a credit card, then they can charge it.

Web Sites-Structures

- A web site is going to be a collection of web pages and other resources. There are a number approaches that you can use to organize how people interact with your site.
- One of the most common is a hierarchical structure where you organize web pages by function or category. Similar functions may collected in subdirectories.
- Resources (like images, audio, video) may be collected in directories as well.

Web Sites - Hierarchical



More here [Website Structures](#)

Web Sites – Information Architecture

- Most large web sites undergo a process of discovery that is used to inform the structure that is used.
- Stakeholders and their goals can be used to drive design functionality.
- User profiling can be used to gain information about the capabilities and expectations of web site clients.
- Low fidelity prototypes can be used to gain information about a design before it is committed to code.
- You can read a short introduction to the process [here](#).

Web Sites – UX issues

- Give prompt feedback. Never make the user guess what is happening.
- Make error messages understandable. Don't make me retype information on an error.
- Use good signifiers. Don't make the user guess how they should interact.
- Use designs that are uncluttered and consistent.
- Put important controls on edges. (Edges act as guard rails.)

Web Sites – UX issues

- Organize the location of content by importance. Center, then top and left, then right, then bottom.
- Broad and shallow tree organization is often preferred.
- Be aware of number of clicks and pointer movements required to perform an operation. Confirmation clicks should be used when an operation is not easy to reverse.
- You can see examples here [Good and Bad UX design](#).
- [Nielsen's Heuristics for good UX design](#)

References

- [W3 Schools HTML](#) A good place to explore HTML. Has examples and quizzes.
- [Mozilla Developers Network Web Docs](#) Good tutorials and examples for HTML (and CSS)
- Design Links
 - [Website Structures](#)
 - [Information Architecture](#)
 - [Good and Bad UX design](#)
- Good books on design
 - [Norman - The Design of Everyday Things](#)
 - [Nielson - Mobile Usability](#)