

CS472 WAP

Lecture 1: Intro to the internet and HTML

Maharishi International University - Fairfield, Iowa

All rights reserved. No part of this slide presentation may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying or recording, or by any information storage and retrieval system, without permission in writing from Maharishi International University.



Wholeness Statement

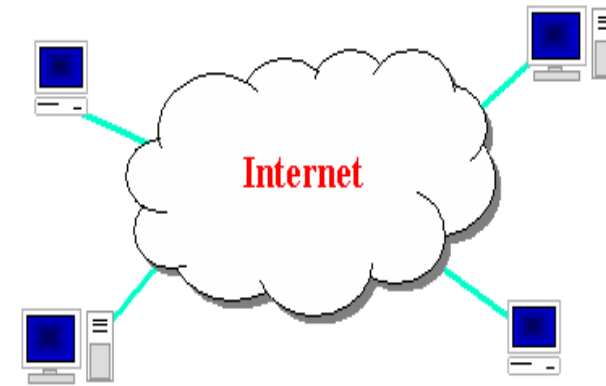
In this lecture we introduce the basic technologies that make up the Internet, the World Wide Web, and the Hyper Text Markup Language (HTML). We will see that many technologies are built on top of other technologies.

Life is found in layers and the TM Technique gives us access to the full range of our awareness and thoughts.

The Internet

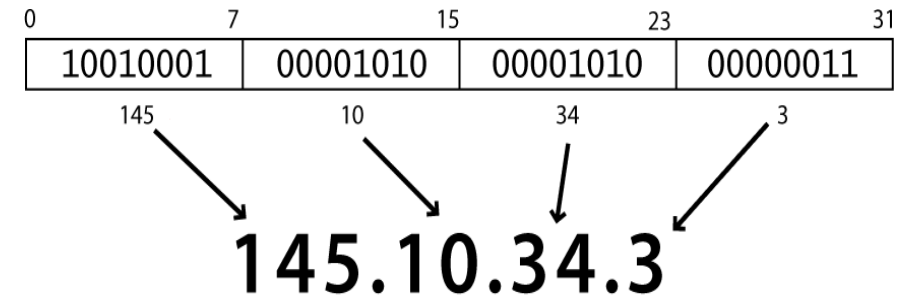
A connection of computer networks using the Internet Protocol (IP) layers of communication protocols:

1. IP
2. TCP/UDP
3. HTTP/FTP/...



Internet Protocol (IPv4)

- The underlying system of communication for all data sent across the Internet.
 - Each device has a 32-bit IP address written as four 8-bit numbers (0-255)
- There are two types of IP addresses
 - servers often have static IP address
 - Users usually get a dynamic IP address from their ISP
- Find out your local IP address:
 - In a terminal, type: `ipconfig` (Windows) or `ifconfig` (Mac/Linux)
- IPv6 addresses are 128-bit IP address written in hexadecimal and separated by colons.
 - An example IPv6 address could be written like this: `3ffe:1900:4545:3:200:f8ff:fe21:67cf`



Transmission Control Protocol (TCP)

- TCP (Transmission Control Protocol) is a standard that defines how to establish and maintain a network conversation via which application programs can exchange data. TCP works with the Internet Protocol (IP).
- Guaranteed message delivery on top of IP
 - IP is stateless – no memory of sent messages or success of delivery
 - TCP is stateful – tracks status of messages until determine success of delivery
- Some programs (games, streaming media programs) use simpler UDP protocol instead of TCP

Multiplexing

- Multiple programs using the same IP address, by using a **port**, a number given to each program or service
 - port 80: default (HTTP)
 - port 443: for secure connection (HTTPS)
 - port 21: File transfer (FTP)
 - [more common ports](#)

Domain Name System (DNS)

- A set of servers that map written names to IP addresses
 - Example: www.miu.edu → 192.249.118.206

Many systems maintain a local DNS cache called a **host** file:

- Windows: C:\Windows\system32\drivers\etc\hosts
- Mac: /etc/hosts
- Linux: /etc/hosts

Hypertext Transport Protocol (HTTP)

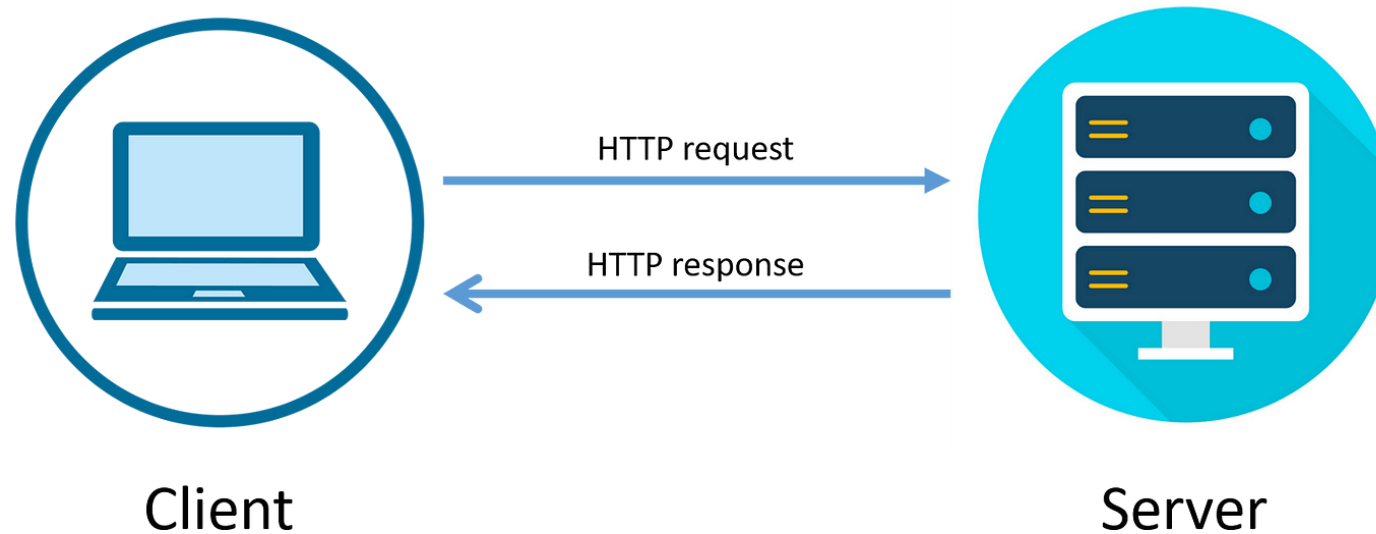
HTTP is the underlying protocol used by the World Wide Web and this protocol defines **how messages are formatted**, and what actions Web servers and clients should take in response to various **commands**.



http: //

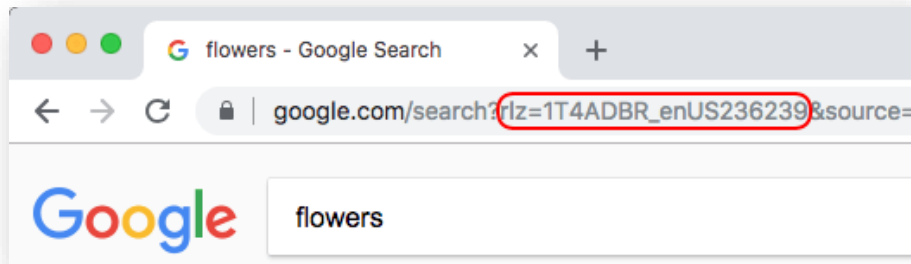
HTTP Request

1. The client sends an HTTP request to the server
2. The server sends a response back

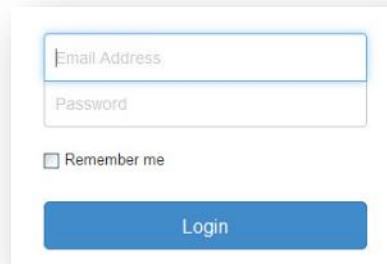


How to send a Request?

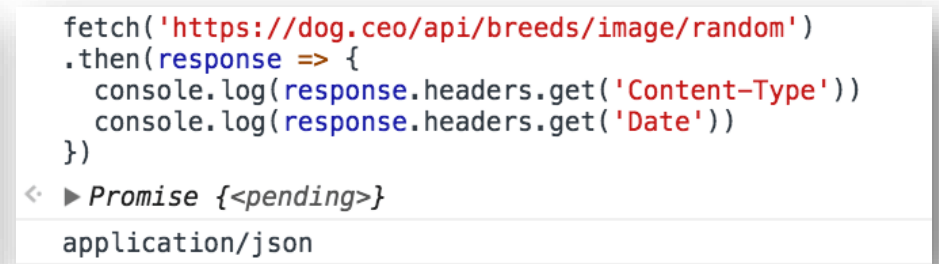
1. From a Browser, enter URL (only GET)
2. From an HTML page, create form and submit button (only GET, POST)
3. From JavaScript (Fetch API)
4. From a software ([Postman](#))
5. From VS code extension



1



2



3

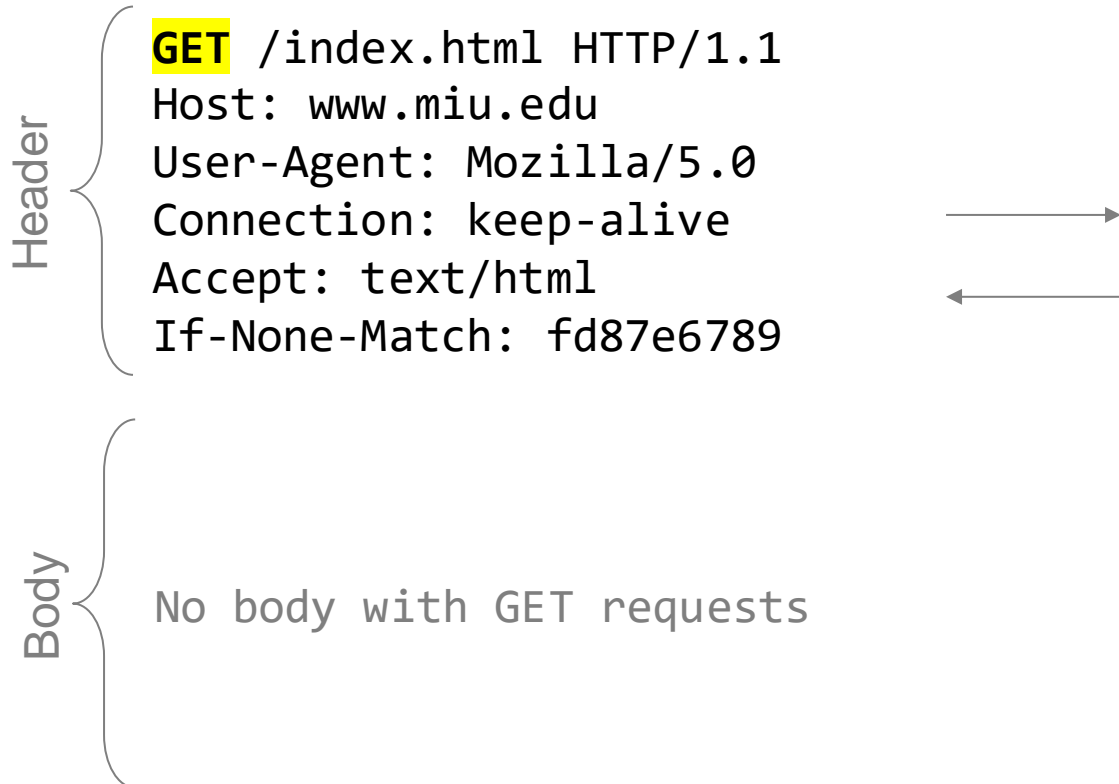
HTTP Verbs

- **GET** Retrieves data from the server
- **POST** Submits data to the server
- **PUT** Replace data on the server
- **DELETE** Delete data from the server

...

Example

Request



Response

HTTP/1.1 200 OK
Content-Length: 16824
Server: Apache
Content-Type: text/html
Date: Monday 8 Dec 2020
Etag: h64w175

<!DOCTYPE html>
<html>
...



Demo: Inspect req/res in Chrome devtools, and Rest Client.

HTTPS: Hypertext Transport Protocol Secure

- A secure version of HTTP
- Encrypts data between your browser and the server
- Uses TLS (Transport Layer Security) for security
- With HTTPS:
 - **Encryption:** Encrypts sensitive data
 - **Authentication:** Confirms you're communicating with the real website
 - **Data Integrity:** Ensures data isn't altered during transfer

Encryption vs. Hashing

Encryption is a two-way function, what is encrypted can be decrypted with the proper key. Usually a key and salt are used.

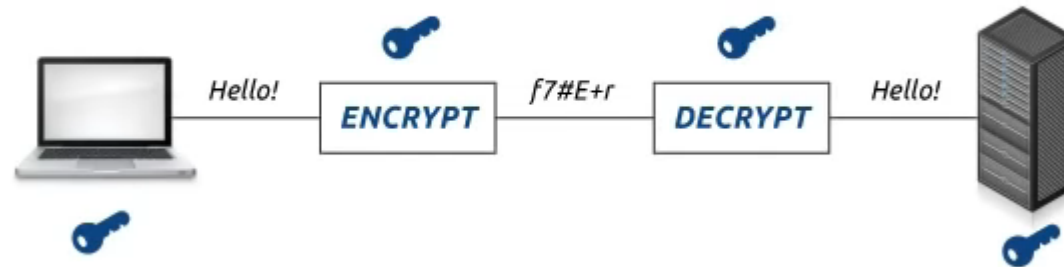
Example: base64 (atob, btoa), RSA, Crypto, AES

Hashing is a one-way function that scrambles plain text to produce a unique message digest (most of time with a fixed size). There is no way to reverse the hashing process to reveal the original plain text.

Example: MD5, SHA1/256/512, PBKDF2, BCrypt

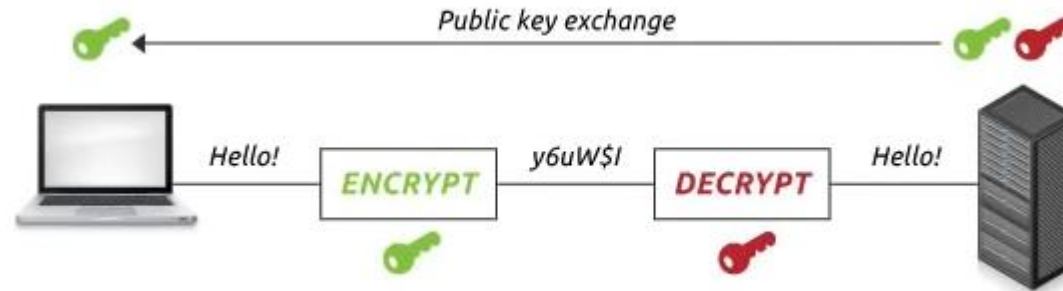
Symmetric Encryption

Symmetric cryptography is a type of encryption where the same key (secret key) is used to both encrypt and decrypt the data. Usually it's very fast because it uses 256 bits key.



Asymmetric Encryption

Asymmetric cryptography (public key cryptography), uses public and private keys to encrypt and decrypt data. **Public key** is used to encrypt the data, and the **Private key** is used to decrypt the data. This is more secure because it uses a 2048 key, but it is slow.



TLS (Transport Layer Security)

A standard security technology for establishing an encrypted connection between a web server and a client. This connection ensures that all data passed between the web server and the client remains private.



Digital Certificate

A certificate is an electronic credentials used to assert the online identities of individuals, applications and other entities on a network. They are similar to an ID card (passport, driving license, diploma). A certificate is signed by (Certificate Authority CA).

A certificate is a file contains:

- Identity of owner
- Public Key of the owner
- Signature of the certificate issuer →
- Expiration date

A signature is always made with the private key of issuer and can be validated with the public key of the issuer.

Certificate Issuer

A certificate authority (CA) validates the identity of the certificate holder.

There are two types of CA:

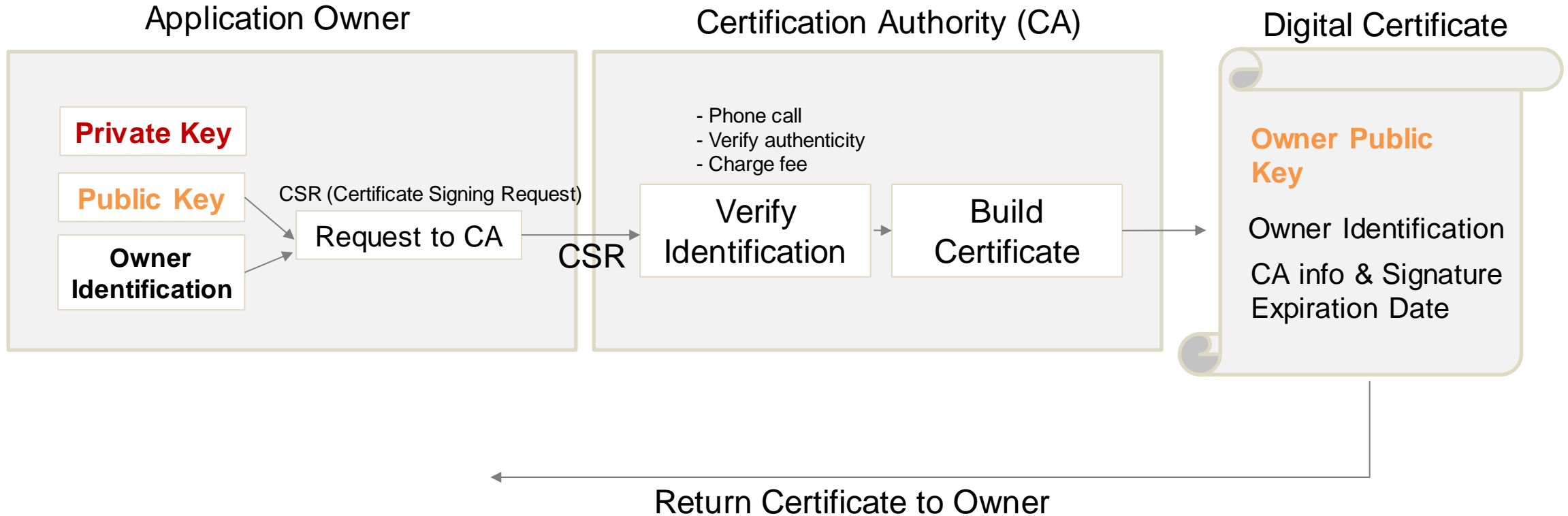
RCA: Trusted Root Certification Authorities (CA)

ICA: Intermediate Certification Authorities (They own certificates from CA) (a security layer so private key for RCA is inaccessible)

All Root CA are pre-defined in the browser, so any client can validate the certificate (like a hotline between the browser and the issuer).



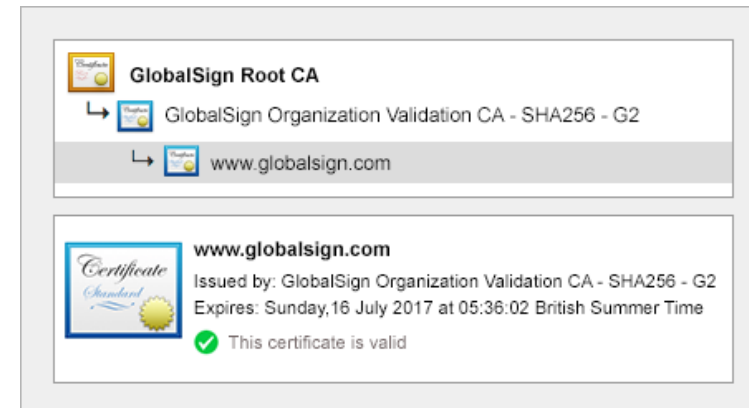
How Digital Certificates Are Created?



Using Digital Certificates

When a client contacts an application owner via TLS, the first thing they receive is the digital certificate. Clients check the validity of this certificate (if it was issued to the claimed app owner, not expired, the CA signature).

Usually a **chain of certificates** are sent to the client (the owner certificate signed by ICA, and ICA certificate signed by Root CA), since the client has the public key for all Root CAs, it validates the ICA certificate first and makes sure it is a certified entity, and then use its public key in that certificate to validate the owner certificate.



TLS Handshake

Client: I want to connect to `https://miu.edu`

Server: Sure, here's my certificate containing my asymmetric public key. It was signed by Comodo CA.

Client: validates the signature of the certificate using the public key of trusted issuers (Comodo CA), if valid, it creates a new symmetric secret key and encrypts it with the asymmetric public key of MIU in the certificate.

Server: decrypts the asymmetric public key with its asymmetric private key to get the symmetric secret key of the client.

All future communication is encrypted and decrypted with the symmetric secret key.

If the client were to connect to the same server the next day, a **new secret key** would be created.

Web Communication In a Nutshell

We generally use the **asymmetric system** to exchange the secret key and then **symmetric system** to encrypt and decrypt messages.

- The server sends its public key to the client
- The client generates a secret key and encrypt it with the server's public key and sends it to the server
- The server with its private key will decrypt and read the secret key

All future communication will be done using the symmetric encryption/decryption using the secret key (faster).

Uniform Resource Locator (URL)

Anchor: jumps to a given section of a web page

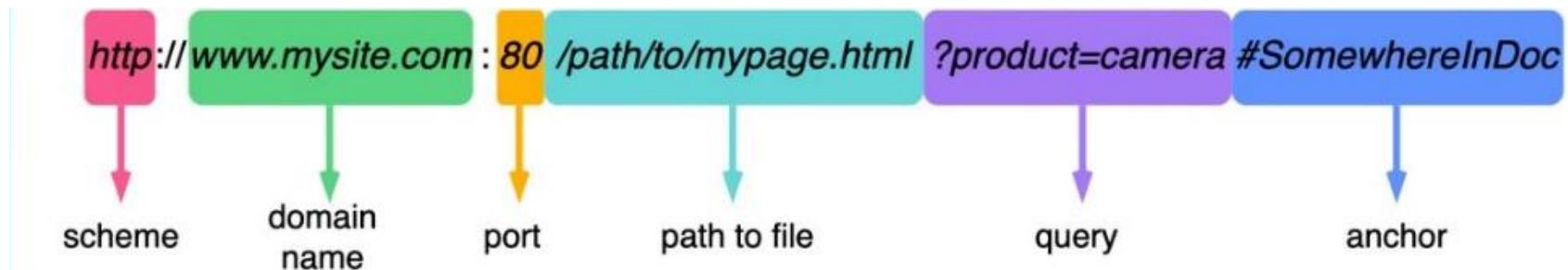
`http://www.miu.edu/download/index.html#downloads`

Port: for web servers on ports other than the default 80

`http://www.cs.miu.edu:8080/mscs/wap.txt`

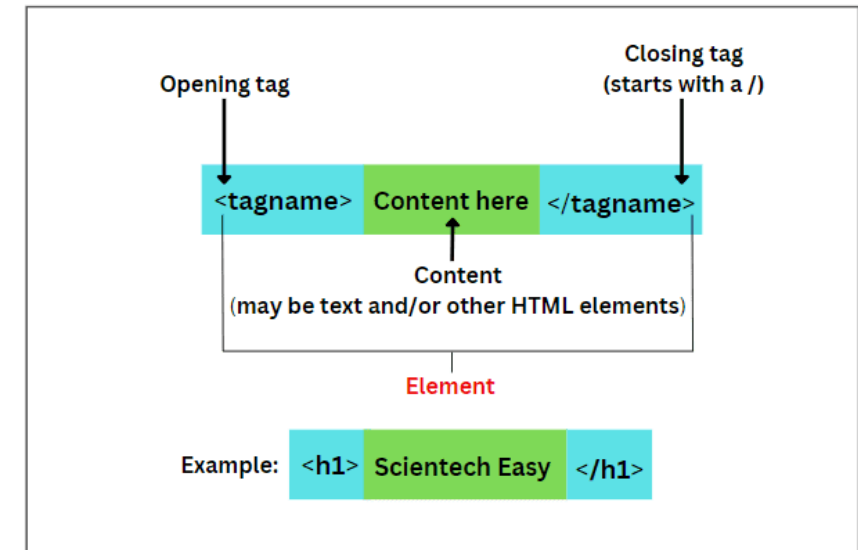
Query string: a set of parameters passed to a web program

`http://www.google.com/search?q=miu&start=10`



Hypertext Markup Language (HTML)

- Describes the content and structure of information on a web page with **tags**: opening and **closing tags** or **self-closed tags**.
- Each tag and its content is called an **element**
 - Syntax: **<tag>**content **</tag>**
 - Example: **<p>**This is a paragraph**</p>**
- Most whitespace is insignificant in HTML
 - (ignored or collapsed to a single space)
- The newest version is **HTML5**
- Each tag, can be configured with **attributes**



Structure of an HTML5 page

- An HTML page is saved into a file ending with extension **.html**
- **DOCTYPE** tag tells browser the HTML version.
 - In this case, HTML5.
- A standard HTML5 page has two sections:
 - **head** describes the page and
 - **body** contains the page's contents

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title></title>
  </head>
  <body>
    Page contents
  </body>
</html>
```

Metadata

- Data about data - Information describes the page itself
 - Placed in the head section of your HTML page
- meta tags often have both the name and content attributes
- HTML5 introduced a way for designers to control the viewport
 - (the user's visible area of a web page), through the <meta> tag

```
<meta charset="utf-8"/>
```

```
<meta name="description" content="Learn HTML" />
```

```
<meta name="keywords" content="HTML,CSS" />
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0"/>
```

Favorites Icon (favicon)

- The link tag, in the head section, attaches another file to the page
- In this case, an icon for the browser title bar and bookmarks
 - E.g. https://www.w3schools.com/tags/tryit.asp?filename=tryhtml5_link_sizes

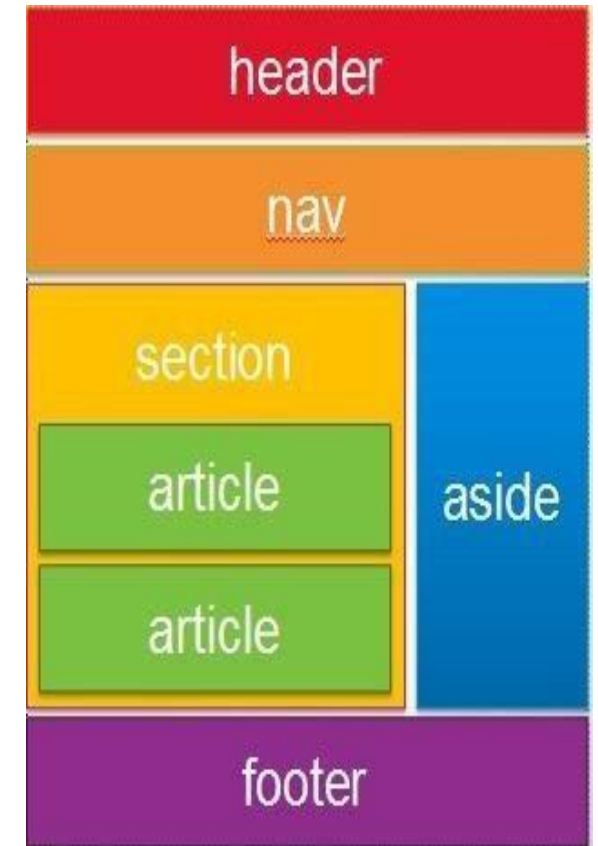
Syntax: `<link href="filename" type="MIME type" rel="relationship"/>`

Example: `<link href="yahoo.gif" type="image/gif" rel="shortcut icon"/>`



HTML5 New Semantic Elements

- They generally have no default outward appearance on the page, instead they give insight into the structure of the page.
 - **section**: defines a section in a document.
 - **header**: specifies a header for a document or section.
 - **footer**: specifies a footer for a document or section.
 - **nav**: defines a set of navigation links.
 - **aside**: defines some content aside from the content it is placed in (like a sidebar).
 - **article**: specifies independent, self-contained content.
 - [More Semantic Tags](#)



Absolute vs. Relative URL

Absolute URL: A complete address containing all the parts needed to find a specific file or web page on the internet.

Relative URL: It provides directions to find a resource relative to the current webpage. A relative URL skips some parts of the absolute URL, assuming the browser can fill in the gaps based on the current webpage location.

The **<base>** tag specifies the base URL and/or target for all relative URLs in a document.

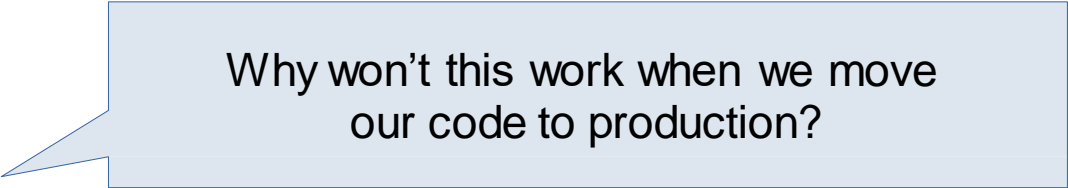
Relative vs Absolute URL

- **Relative URL**

- index.html (path relative)
- graphics/image.png (path relative)
- ./graphics/image.png (path relative)
- ../about.html (directory relative)
- ../../stories/stories.html (directory relative)
- /image.png (root relative)

- **Absolute URL**

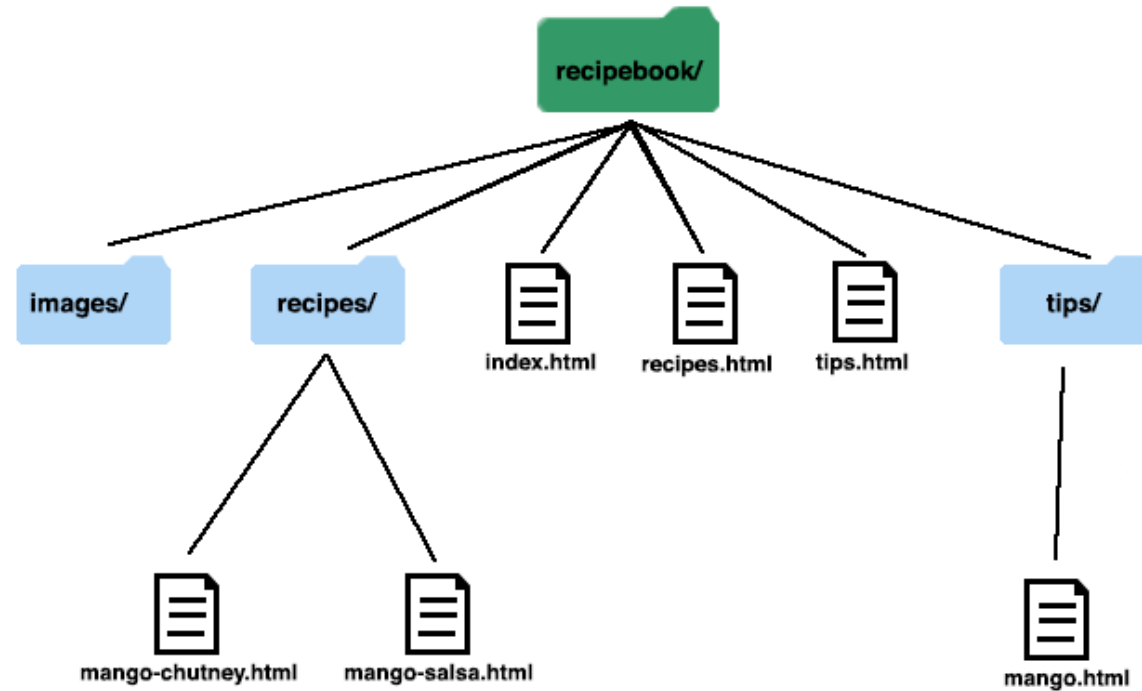
- <http://www.mysite.com>
- C:\website\images\image.png



Why won't this work when we move our code to production?

Exercise

From mango-chutney.html



Link to index.html:

Link to mango-salsa.html:

Link to mango.html:

Images ``

- Inserts a graphical image into the page (inline)
 - Another get request
 - The **src** attribute specifies the image URL
 - HTML5 also requires an **alt** attribute
 - describing the image
 - The **title** attribute is an optional tooltip
 - (on ANY element)



```

```

Links <a>

- Links, or "anchors", to other pages (inline)
 - **href** attribute can be absolute or relative URL
 - Anchors are inline elements

In HTML5, you can wrap links around “block-level” elements

<p>

```
<a href="story1.html">Bruce Wayne, the richest
man in Gotham City, is the alter ego of
Batman. Bam! </a>
```

</p>

Block Level Elements



- A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).
- Examples of block-level elements:

`<div>`

`<h1> - <h6>`

`<p>`

`<form>`

Paragraph <p>



- Creates a paragraph in a block of text
 - Placed in the body of the page
 - See examples at [w3schools](https://www.w3schools.com/html/html_tags.asp)

<p>

Lorem quis lorem. Pellentesque ultrices nunc id mauris
posuere pulvinar.

</p>

Headings <h1>, <h2>, ..., <h6>

- Headings are used to give a title (heading) to major areas of the page (block)

Note: only use one <h1> tag per page, as it describes “the page”

```
<h1>Maharishi University</h1>
```

```
<h2>Department of Computer Science</h2>
```

```
<h3>WAP Course</h3>
```

Maharishi University

Department of Computer Science

WAP Course

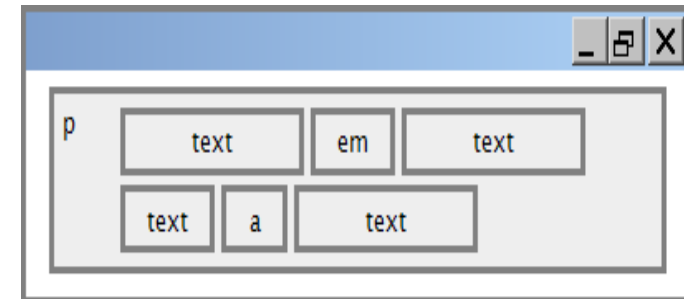
Inline Elements



- An inline element does not start on a new line and only takes up as much width as necessary.
- Examples of inline elements:
 - ``
 - `<a>`
 - ``

Block vs Inline Elements

- Block elements contain an entire large region of content
 - Examples: paragraphs, lists, table cells
 - The browser places a margin of whitespace between block elements for separation, normally a newline
- Inline elements affect a small amount of content
 - Examples: bold text, code fragments, images
 - The browser allows many inline elements to appear on the same line
 - Must be nested inside a block element



Line Break `
`

- Forces a line break in the middle of a block element (inline)

`<p>`Teddy said it was a hat, `
` So I put it on .`</p>`

`<p>`Now Daddy's saying, `
` Where the heck's the toilet plunger gone?`</p>`

`
` should not be used to separate paragraphs or used multiple times in a row to create spacing

- Warning: Don't over-use br
 - (guideline: ≥ 2 in a row is bad)

Nesting Tags

- Tags must be correctly nested
 - a closing tag must match the most recently opened tag

The browser may render it correctly anyway, but it is invalid HTML

- Bad nesting:
 - `<p> What a lovely </p> day `
- Good nesting:
 - `<p> What a lovely day </p>`

Comments <!-- -->

- Comments to document your HTML file
 - or "comment out" text
- Useful at the top of page and for disabling code

```
<!-- This is a comment -->
```

```
<p>
```

```
WAP courses are <!-- NOT --> a lot of fun!
```

```
</p>
```

Computer Code `<code>`



- A short section of computer code (usually shown in a fixed-width font) (inline)

WI

WI

The `ul` and `ol` tags make lists.

`<p>`

The `<code>ul</code>` and **`<code>ol</code>`** tags make lists.

`</p>`

Preformatted Text `<pre>`



- A large section of pre-formatted text (block)
 - a fixed-width font (usually Courier), and it preserves both spaces and line breaks.

`<pre>`

**Steve Jobs speaks loudly reality
distortion Apple fans bow down**

`</pre>`

```
Steve Jobs speaks loudly reality
distortion Apple fans bow down
```

Unordered Lists `` ``



- `ul` represents a bulleted list of items (block)
- `li` represents a single item within the list (block)

```
<ul>  
  <li>No shoes</li>  
  <li>No shirt</li>  
  <li>No problem!</li>  
</ul>
```

- No shoes
- No shirt
- No problem!

Ordered List



- ol represents a numbered list of items (block)
- We can make lists with letters or Roman numerals using CSS (later)

```
<p>Concise business model:</p>
<ol>
  <li>Find customers</li>
  <li>Make profits</li>
</ol>
```

Concise business model:

1. Find customers
2. Make profit!

Nested Lists



```
<ul>
  <li>Simpsons:
    <ul>
      <li>Homer</li>
      <li>Marge</li>
    </ul>
  </li>
  <li>Family Guy:
    <ul>
      <li>Peter</li>
      <li>Lois</li>
    </ul>
  </li>
</ul>
```

Important: nested lists are inside list items.
Not directly inside the other list!

- Simpsons:
 - Homer
 - Marge
- Family Guy:
 - Peter
 - Lois

HTML Tables <table> <tr> <td>



- A 2D table of rows and columns of data (block element)
- table defines the overall table, tr each row, and td each cell's data
- tables are intended for displaying row/column data sets

```
<table>
  <tr>
    <td>1,1</td>
    <td>1,2 okay</td>
  </tr>
  <tr>
    <td>2,1 real wide</td>
    <td>2,2</td>
  </tr>
</table>
```

1,1	1,2 okay
2,1 real wide	2,2

- Table Header <th> and Caption <caption> tags
- The **rowspan** and **colspan** attributes
- Don't use Tables for Layout.

Table Header <th> and Caption <caption>



- the cells in a row are headers
 - Appear bold by default
- A caption labels the meaning of the table

Column 1	Column 2
1,1	1,2 okay
2,1 real wide	2,2

```
<table>
  <caption>My important data</caption>
  <tr>
    <th>Column 1</th>
    <th>Column 2</th>
  </tr>
  <tr>
    <td>1,1</td>
    <td>1,2 okay</td>
  </tr>
  <tr>
    <td>2,1 real wide</td>
    <td>2,2</td>
  </tr>
</table>
```

The rowspan and colspan attributes



colspan makes a cell occupy multiple columns
rowspan makes a cell occupy multiple rows

```
<table>
  <tr>
    <th>Column 1</th>
    <th>Column 2</th>
    <th>Column 3</th>
  </tr>
  <tr>
    <td colspan="2">1,1-1,2</td>
    <td rowspan="3">1,3-3,3</td>
  </tr>
  <tr>
    <td>2,1</td>
    <td>2,2</td>
  </tr>
  <tr>
    <td>3,1</td>
    <td>3,2</td>
  </tr>
</table>
```

Column 1 **Column 2** **Column 3**

1,1-1,2

2,1

2,2

1,3-3,3

3,1

3,2

Column 1	Column 2	Column 3
1,1-1,2		1,3-3,3
2,1	2,2	
3,1	3,2	

Main point

The Hypertext Markup Language uses tags to demarcate different sections of a text. An HTML page always starts with a `<html>` tag, inside of which it has a `<head>` tag to describe the page and a `<body>` tag of the contents that will be displayed. These are the tags you will use for every HTML page. This is a foundational concept

Well begun is half done. Start with a good foundation and build upon that.

CONNECTING THE PARTS OF KNOWLEDGE WITH THE WHOLENESS OF KNOWLEDGE

Layers of Abstraction

1. HTML is the basis of web programming; every web page is composed of HTML.
 2. To be an effective web programmer you also must understand the deeper underlying realities of HTTP, TCP, and DNS.
-
3. Transcendental consciousness is when our mind is in contact with the deepest underlying reality, the unified field.
 4. Impulses within the Transcendental field: the infinite dynamism of the unified field constantly expresses itself as all the layers of the universe
 5. Wholeness moving within itself: In Unity Consciousness, one experiences that all these layers are ultimately composed of pure consciousness, our own Pure Awareness.
- 