Park Innovation

Full Stack Dev Bootcamp

Daniel Awde - Park Innovation

Section 1 of 8

DEFINITIONS AND HISTORY

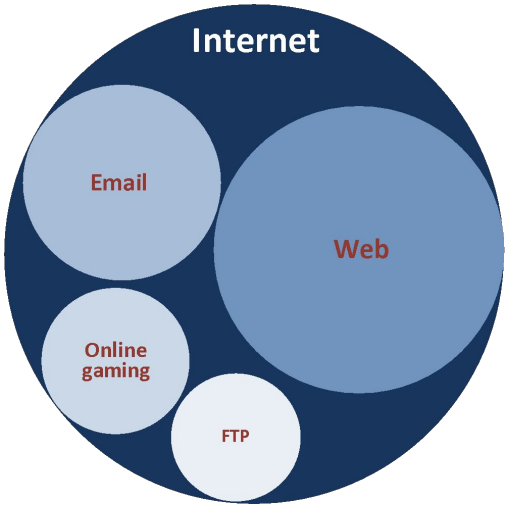
Daniel Awde - Park Innovation

Internet = Web?

The answer is no

The World-Wide Web (WWW or simply the Web) is certainly what most people think of when they see the word “internet.”

But the WWW is only a subset of the Internet.



Communication Definitions

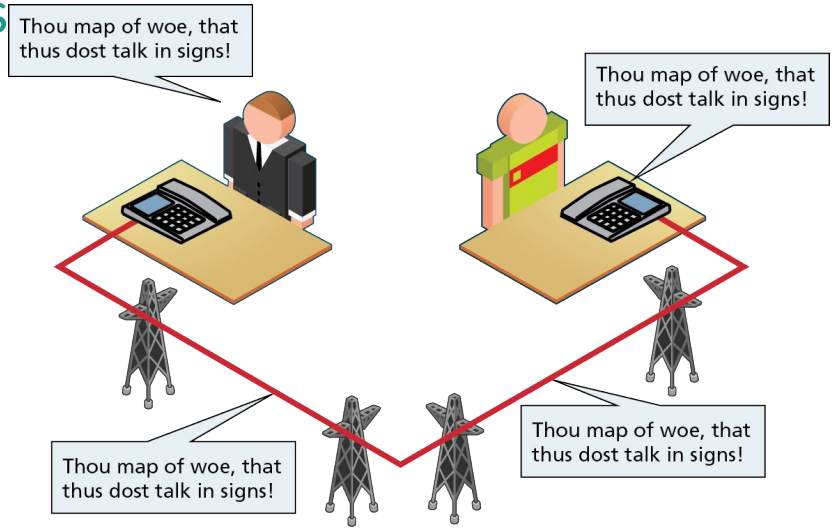
We will begin with the telephone

Telephone networks provide a good starting place to learn about modern digital communications.

In the telephone networks of old, calls were routed through operators who physically connected caller and receiver by connecting a wire to a switchboard to complete the circuit.

Circuit Switching

A **circuit switching** establishes an actual physical connection between two people through a series of physical switches



Circuit Switching

Its Limitations

Circuit Switching Weaknesses

- You must establish a link and maintain a dedicated circuit for the duration of the call
- Difficult to have multiple conversations simultaneously
- Wastes bandwidth since even the silences are transmitted

ARPANET

The beginnings of the Internet

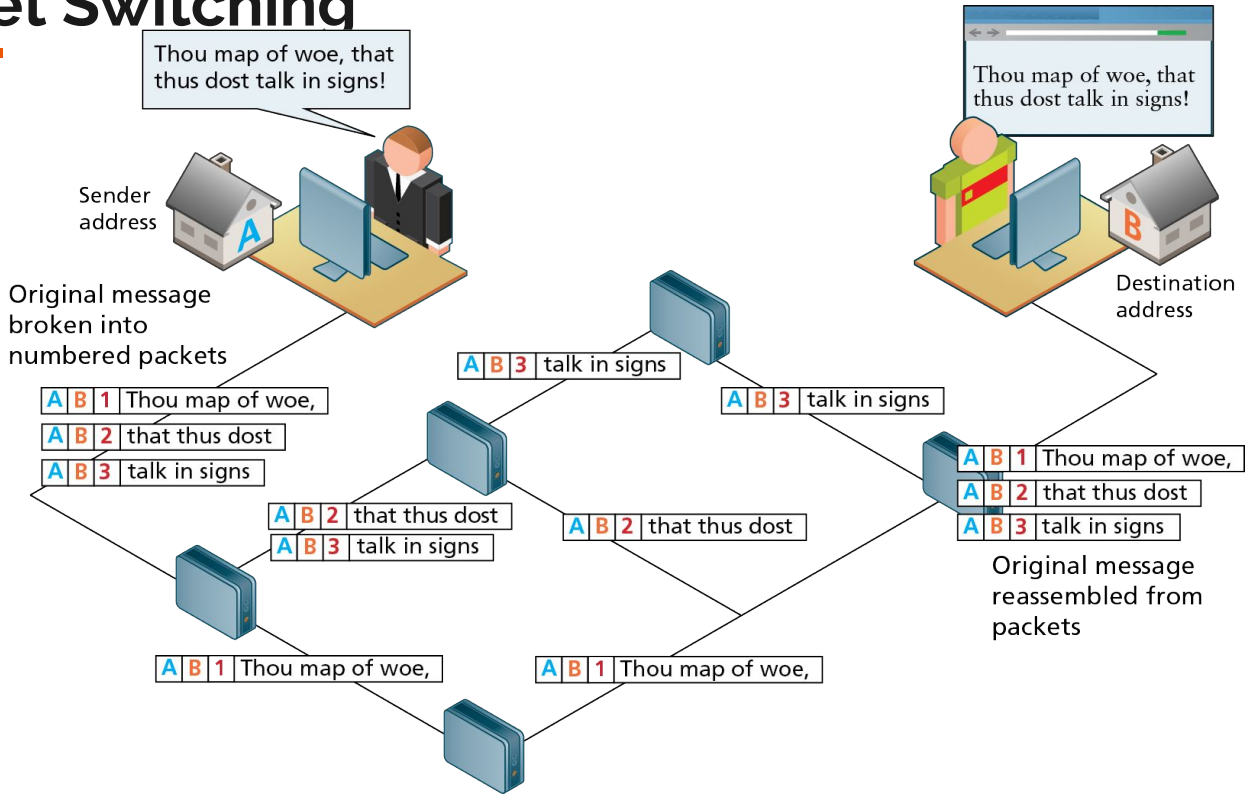
The research network ARPANET was created. In the 1960s

- ARPANET did not use circuit switching
- it used **packet switching**

A packet-switched network does not require a continuous connection. Instead it splits the messages into smaller chunks called **packets** and routes them to the appropriate place based on the destination address.

The **packets** can take different routes to the destination.

Packet Switching



Packet Switching

Isn't this more complicated?

While **packet switching** may seem a more complicated and inefficient approach than **circuit switching**, it is:

- more robust (it is not reliant on a single pathway that may fail) and
- a more efficient use of network resources (since a circuit can communicate multiple connections).

Short History of the Internet

Perhaps not short enough

The early ARPANET network was funded and controlled by the United States government, and was used exclusively for academic and scientific purposes.

The early network started small with just a handful of connected campuses in 1969 and grew to a few hundred by the early 1980s.

TCP/IP

Rides to the rescue

To promote the growth and unification of the disparate networks a suite of protocols was invented to unify the networks together.

By 1981, new networks built in the US began to adopt the **TCP/IP** (**Transmission Control Protocol / Internet Protocol**) communication model (discussed in the next section), while older networks were transitioned over to it.

Core Features of the Web

Shortly after that initial proposal Berners-Lee developed the main features of the web:

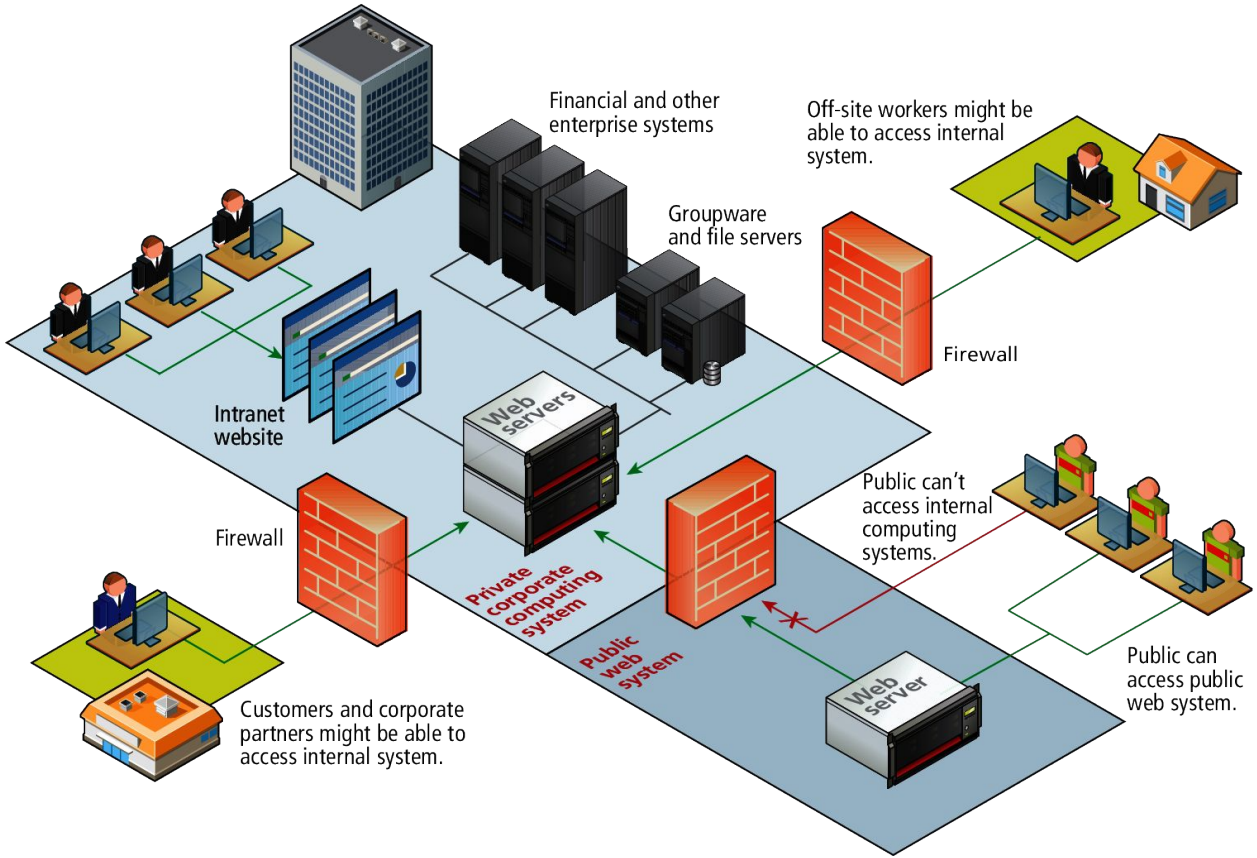
1. A URL to uniquely identify a resource on the WWW.
2. The HTTP protocol to describe how requests and responses operate.
3. A software program (later called web server software) that can respond to HTTP requests.
4. HTML to publish documents.
5. A program (later called a browser) to make HTTP requests from URLs and that can display the HTML it receives.

W3C
The World Wide Web Consortium

Also in late 1994, Berners-Lee helped found the **World Wide Web Consortium** (**W3C**), which would soon become the international standards organization that would oversee the growth of the web.

This growth was very much facilitated by the decision of CERN to not patent the work and ideas done by its employee and instead left the web protocols and code-base royalty free.

Intranet versus Internet



Daniel Awde - Park Innovation

Static Web Sites

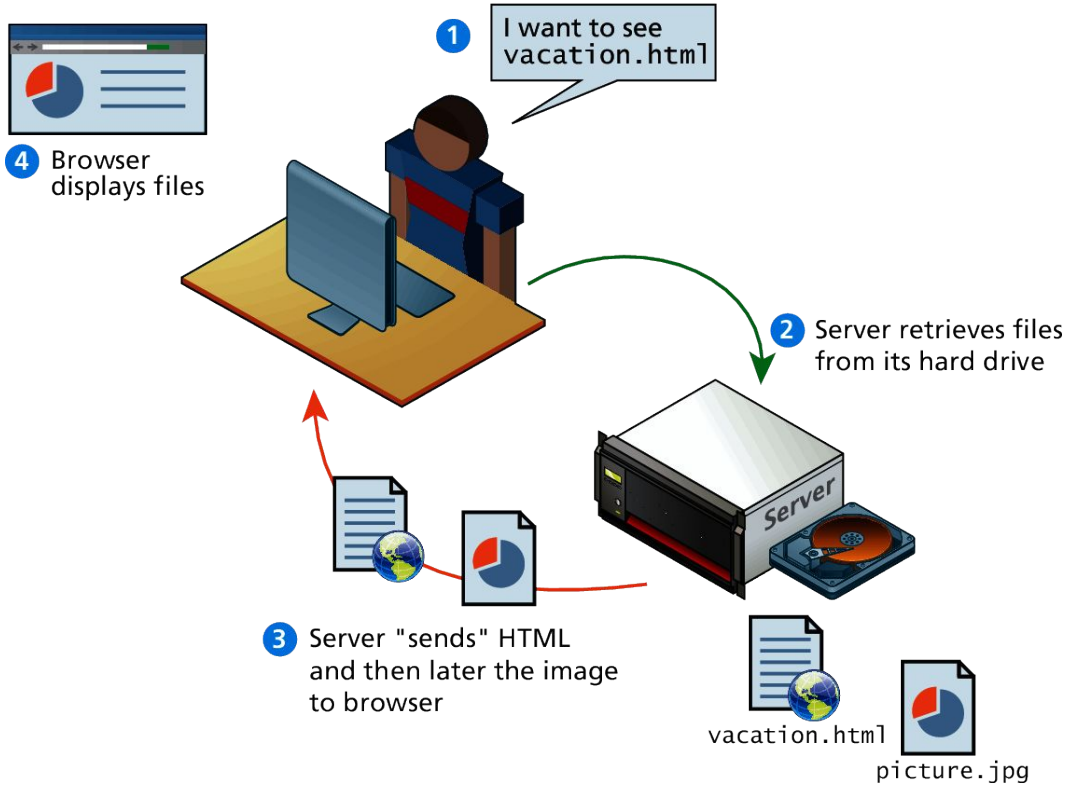
Partying Like It's 1995

In the earliest days of the web, a **webmaster** (the term popular in the 1990s for the person who was responsible for creating and supporting a web site) would publish web pages, and periodically update them.

In those early days, the skills needed to create a web site were pretty basic: one needed knowledge of the HTML markup language and perhaps familiarity with editing and creating images.

This type of web site is commonly referred to as a **static web site**, in that it consists only of HTML pages that look identical for all users at all times.

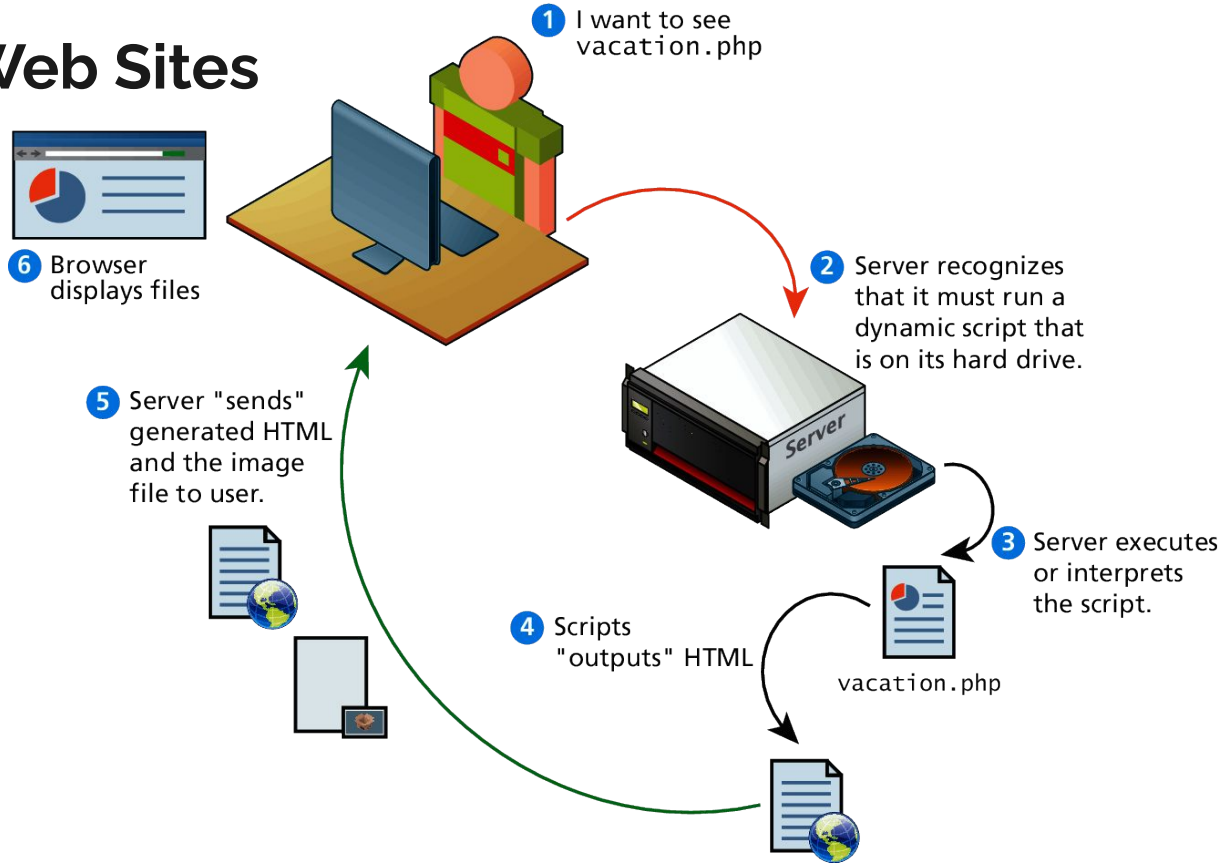
Static Web Sites



Dynamic Web Sites

Within a few years of the invention of the web, sites began to get more complicated as more and more sites began to use programs running on web servers to generate content dynamically.

Dynamic Web Sites



Dynamic Web Sites

What are they?

These server-based programs would read content from databases, interface with existing enterprise computer systems, communicate with financial institutions, and then output HTML that would be sent back to the users' browsers.

This type of web site is called here in this book a **dynamic web site** because the page content is being created at run-time by a program created by a programmer; this page content can vary for user to user.

Web 2.0 and Beyond

In the mid 2000s, a new buzz-word entered the computer lexicon: **web 2.0**.

This term had two meanings, one for users and one for developers.

Web 2.0

Its meaning for users

For the users, Web 2.0 referred to an interactive experience where users could contribute *and* consume web content, thus creating a more user-driven web experience.

Web 2.0

Its meaning for developers

For software developers, Web 2.0 also referred to a change in the paradigm of how dynamic web sites are created.

Programming logic, which previously existed only on the server, began to migrate to the browser.

This required learning Javascript, a rather tricky programming language that runs in the browser, as well as mastering the rather difficult programming techniques involved in asynchronous communication.

Section 2 of 8

INTERNET PROTOCOLS

Daniel Awde - Park Innovation

What's a Protocol?

The internet exists today because of a suite of interrelated communications protocols.

A **protocol** is a set of rules that partners in communication use when they communicate.

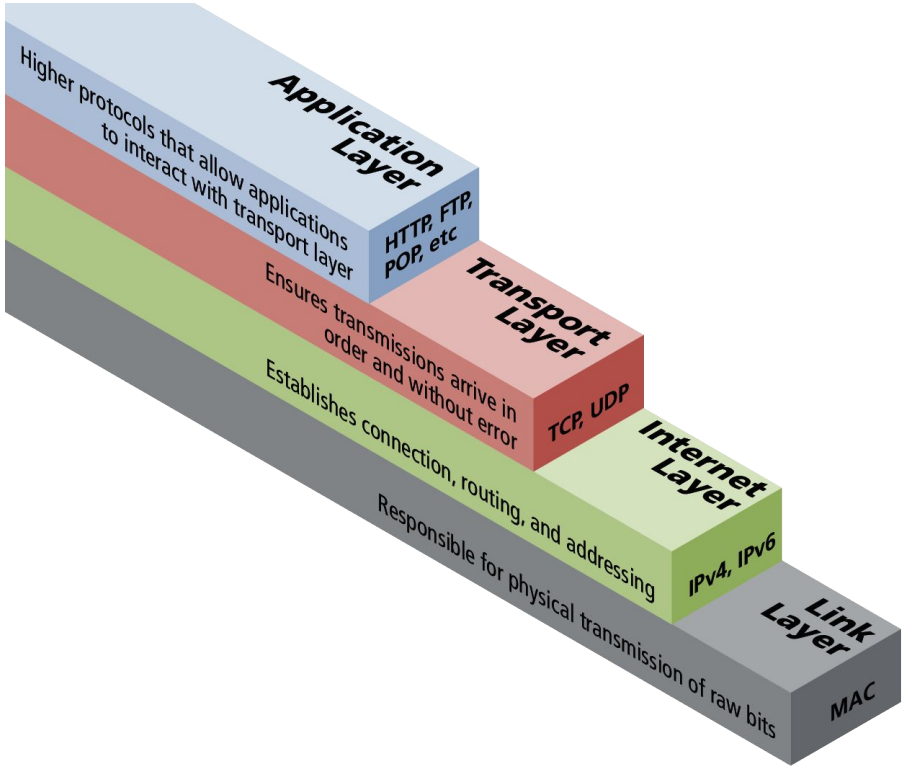
A Layered Architecture

The TCP/IP Internet protocols were originally abstracted as a four-layer stack.

Later abstractions subdivide it further into five or seven layers.

Since we are focused on the top layer anyhow, we will use the earliest and simplest **four-layer network model**.

Four Layer Network Model



Link Layer

Save this for your networking course

The **link layer** is the lowest layer, responsible for both the physical transmission across media (wires, wireless) and establishing logical links.

It handles issues like packet creation, transmission, reception and error detection, collisions, line sharing and more.

Internet Layer

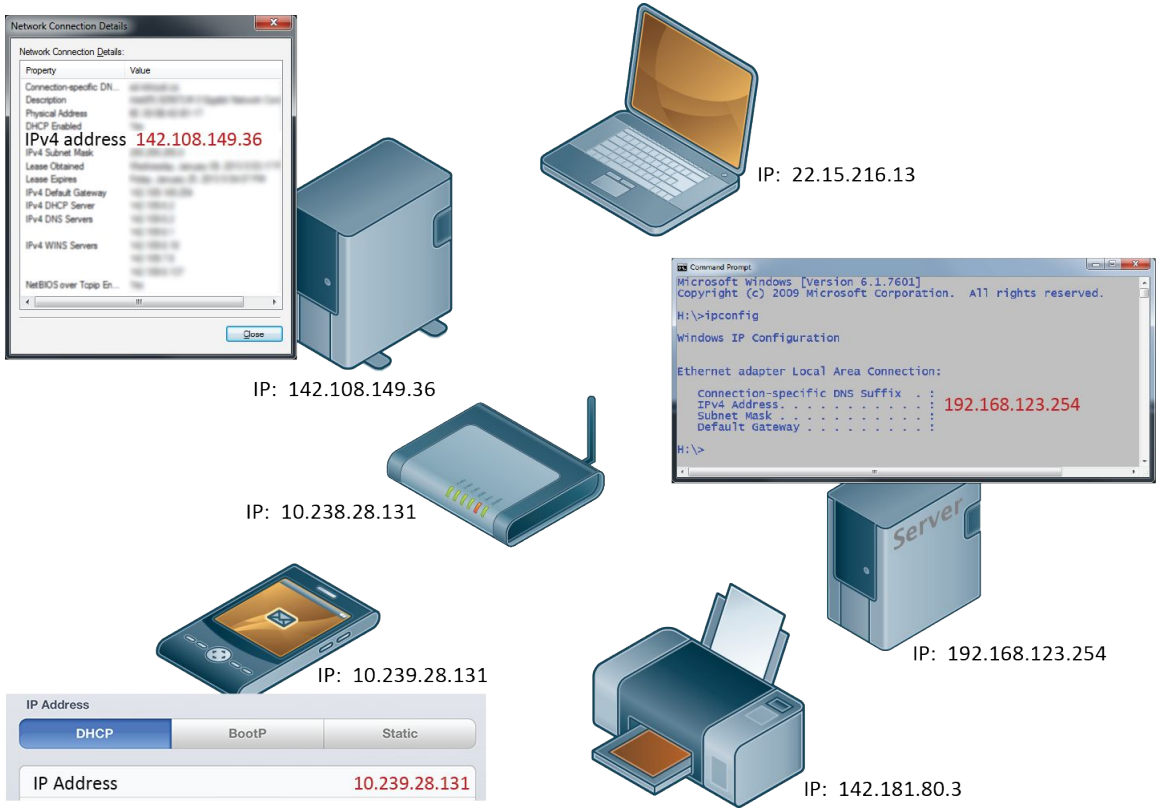
The **internet layer** (sometimes also called the IP Layer) routes packets between communication partners across networks.

Internet Protocol (IP)

The Internet uses the **Internet Protocol** (**IP**) addresses to identify destinations on the Internet.

Every device connected to the Internet has an **IP address**, which is a numeric code that is meant to uniquely identify it.

IP addresses and the Internet



IP Addresses

Two types

IPv4 addresses are the IP addresses from the original TCP/IP protocol.

In IPv4, 12 numbers are used (implemented as four 8-bit integers), written with a dot between each integer.

Since an unsigned 8-bit integer's maximum value is 255, four integers together can encode approximately 4.2 billion unique IP addresses.

IP Addresses

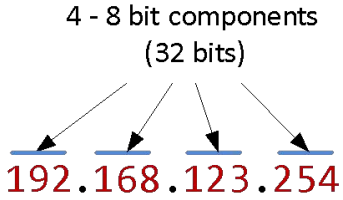
Two types

To future proof the Internet against the 4.2 billion limit, a new version of the IP protocol was created, **IPv6**.

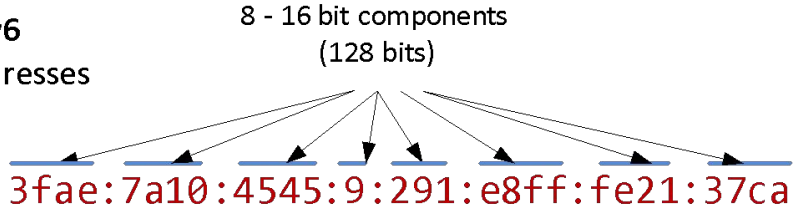
This newer version uses eight 16-bit integers for 2^{128} unique addresses, over a billion billion times the number in IPv4.

These 16-bit integers are normally written in hexadecimal, due to their longer length.

IPv4
2^{32} addresses



IPv6
2^{128} addresses



IP Addresses

Inside of networks is different

Your IP address will generally be assigned to you by your Internet Service Provider (ISP).

In organizations, large and small, purchasing extra IP addresses from the ISP is not cost effective.

In a local network, computers can share a single IP address between them.

Transport Layer

The **transport layer** ensures transmissions arrive, in order, and without error.

This is accomplished through a few mechanisms.

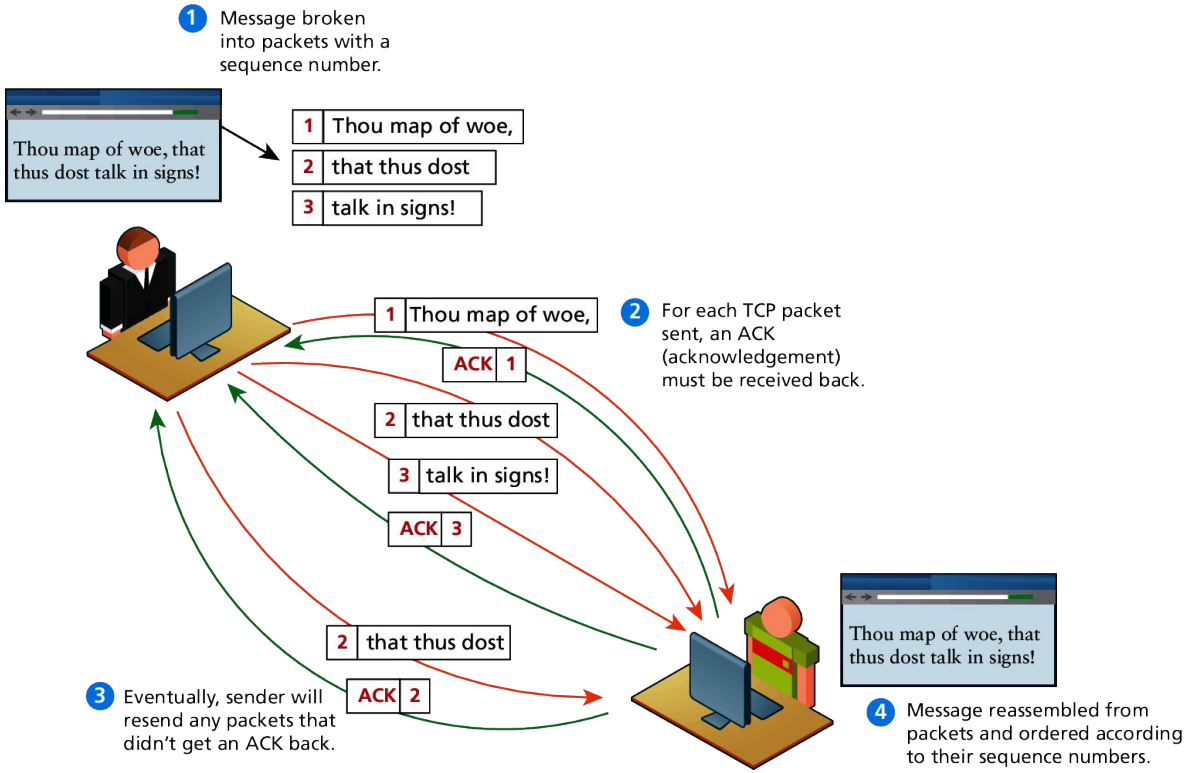
First, the data is broken into packets formatting according to the **Transmission Control Protocol** (**TCP**).

Transport Layer

Secondly, each packet is acknowledged back to the sender so in the event of a lost packet, the transmitter will realize a packet has been lost since no ACK arrived for that packet.

That packet is retransmitted, and although out of order, is reordered at the destination.

TCP Packets


1 Message broken into packets with a sequence number.
Thou map of woe, that thus dost talk in signs!
1 Thou map of woe,
2 that thus dost
3 talk in signs!
1 Thou map of woe,
2 For each TCP packet sent, an ACK (acknowledgement) must be received back.
ACK 1
2 that thus dost
3 talk in signs!
ACK 3
2 that thus dost
ACK 2
3 Eventually, sender will resend any packets that didn't get an ACK back.
Thou map of woe, that thus dost talk in signs!
4 Message reassembled from packets and ordered according to their sequence numbers.

Application Layer

With the **application layer**, we are the level of protocols familiar to most web developers.

Application layer protocols implement process-to-process communication and are at a higher level of abstraction in comparison to the low-level packet and IP addresses protocols in the layers below it.

Examples: HTPP, SSH, FTP, DNS, POP, SMTP.

Section 3 of 8

CLIENT-SERVER MODEL

Daniel Awde - Park Innovation

Client-Server Model

What is it?

The web is sometimes referred to as a client-server model of communications.

In the **client-server model**, there are two types of actors: clients and servers.

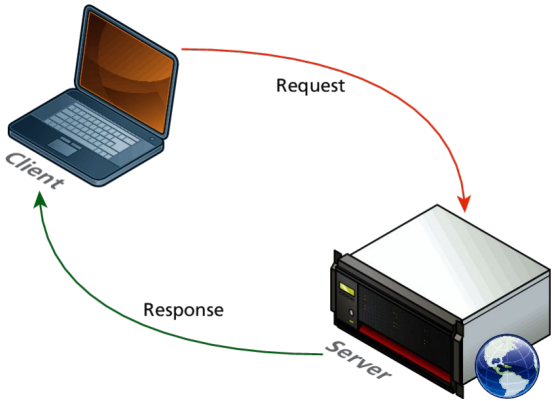
The **server** is a computer agent that is normally active 24 hours a day, 7 days a week (or simply 24/7), listening for queries from any client who make a request.

A **client** is a computer agent that makes requests and receives responses from the server, in the form of response codes, images, text files, and other data.

Request-Response Loop

Within the client-server model, the **request-response loop** is the most basic mechanism on the server for receiving requests and transmitting data in response.

The client initiates a **request** to a server and gets a **response** that could include some resource like an HTML file, an image or some other data.



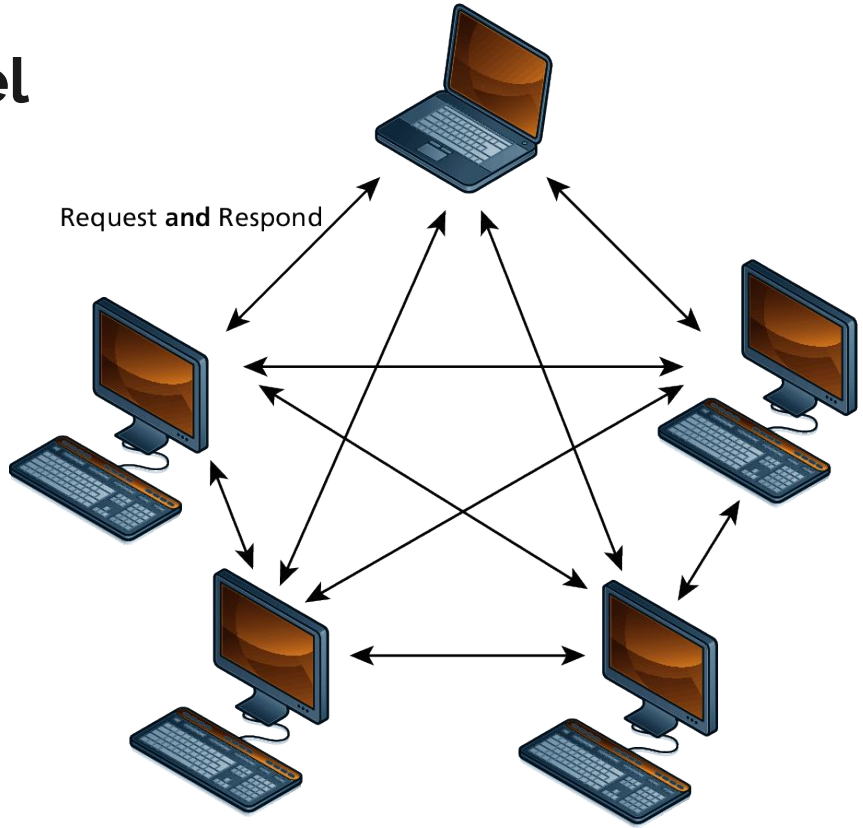
The Peer-to-Peer Alternative

Not actually illegal

In the **peer-to-peer model** where each computer is functionally identical, each node is able to send and receive directly with one another.

In such a model each peer acts as both a client and server able to upload and download information.

Peer-to-Peer Model



Server Types

A server is rarely just a single computer

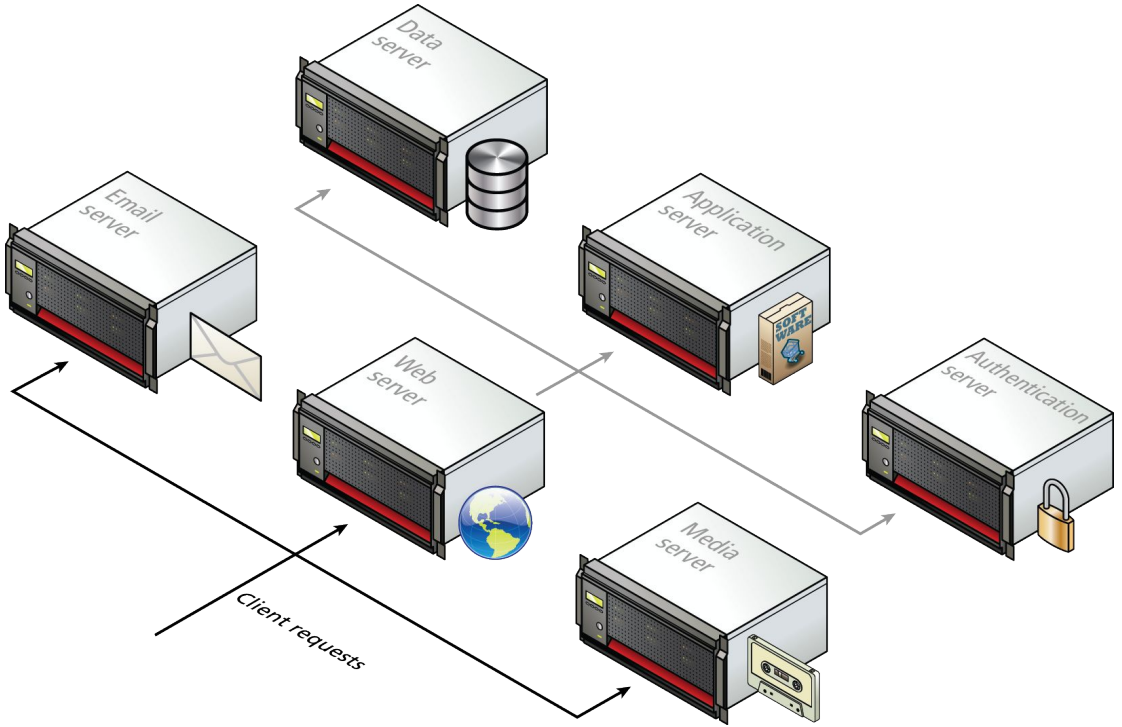
Earlier, the server was shown as a single machine, which is fine from a conceptual standpoint.

Clients make requests for resources from a URL; to the client, the server *is* a single machine.

However, most real-world web sites are typically not served from a single server machine, but by many servers.

It is common to split the functionality of a web site between several different types of server.

Server Types



Real-World Server Installations

Not only are there different types of servers, there is often replication of each of the different server types.

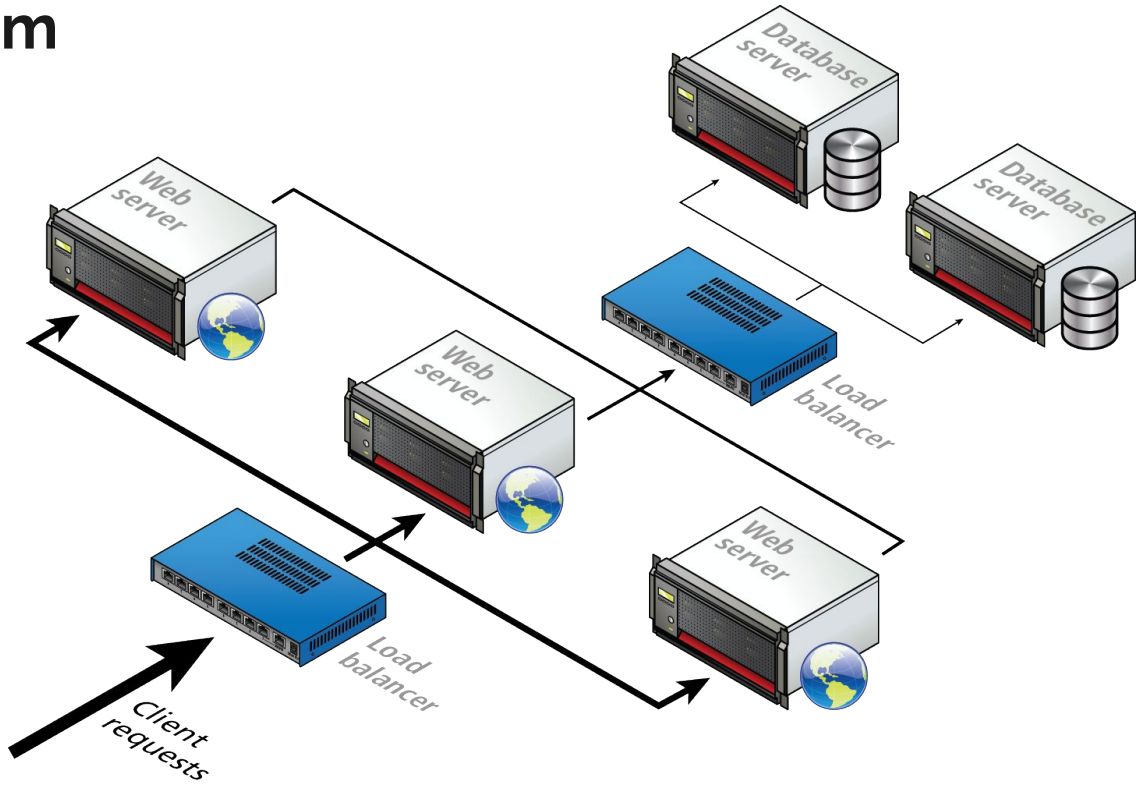
A busy site can receive thousands or even tens of thousands of requests a second; globally popular sites such as Facebook receive millions of requests a second.

Server Farms

Have no cows

A single web server that is also acting as an application or database server will be hard-pressed to handle more than a few hundred requests a second, so the usual strategy for busier sites is to use a **server farm**.

Server Farm


Database server
Database server
Web server
Web server
Load balancer
Web server
Load balancer
Client requests

Server Farms

The goal behind server farms is to distribute incoming requests between clusters of machines so that any given web or data server is not excessively overloaded.

Special routers called **load balancers** distribute incoming requests to available machines.

Server Farms

Even if a site can handle its load via a single server, it is not uncommon to still use a server farm because it provides **failover redundancy**.

That is, if the hardware fails in a single server, one of the replicated servers in the farm will maintain the site's availability.

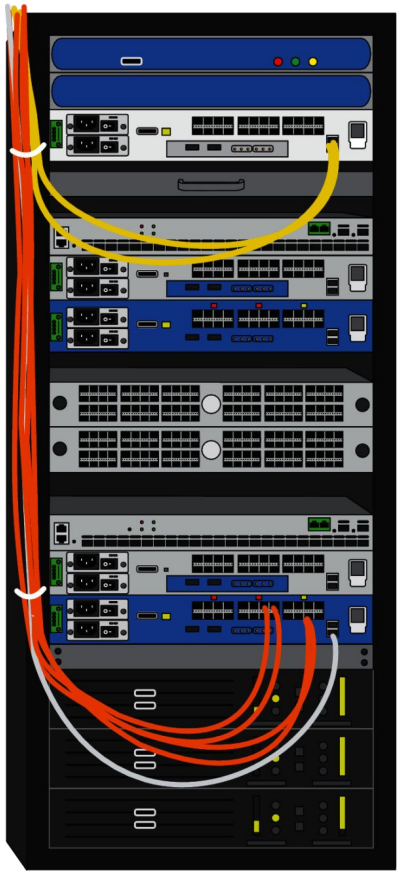
Server Racks

In a server farm, the computers do not look like the ones in your house.

Instead, these computers are more like the plates stacked in your kitchen cabinets.

That is, a farm will have its servers and hard drives stacked on top of each other in **server racks**.

A typical server farm will consist of many server racks, each containing many servers.



Fiber channel switches

Rack management server

Test server

Keyboard tray and flip-up monitor

Patch panel

Production web server

Production data server

RAID HD arrays

Patch panel

Production web server

Production data server

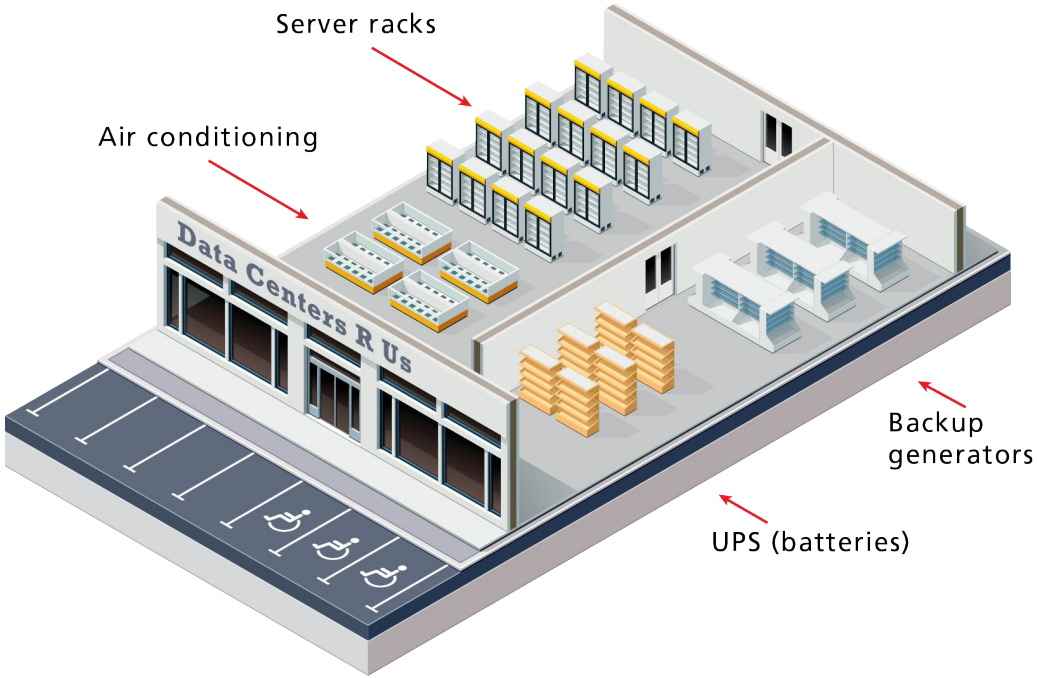
Batteries and UPS

Daniel Awde - Park Innovation

Data Centers

Server farms are typically housed in special facilities called **data centers**.

Hypothetical Data Center



Data Centers

Where are they?

To prevent the potential for site down times, most large web sites will exist in mirrored data centers in different parts of the country, or even world.

As a consequence, the costs for multiple redundant data centers are quite high, and only larger web companies can afford to create and manage their own.

Most web companies will instead lease space from a third-party data center.

Commercial Web Hosting

It is also common for the reverse to be true – that is, a single server machine may host multiple sites.

Large commercial web hosting companies such as GoDaddy, Blue Host, Dreamhost, and others will typically host hundreds or even thousands of sites on a single machine (or mirrored on several servers).

Section 4 of 8

WHERE IS THE INTERNET?

Daniel Awde - Park Innovation

Is the Internet a Cloud?

The Internet is often visually represented as a cloud, which is perhaps an apt way to think about the Internet given the importance of light and magnetic pulses to its operation.

Is the Internet a Cloud?

No

It is important to recognize that our global network of networks does not work using magical water vapor, but is implemented via

millions of kilometers of copper wires and fiber optic cables, as well as via

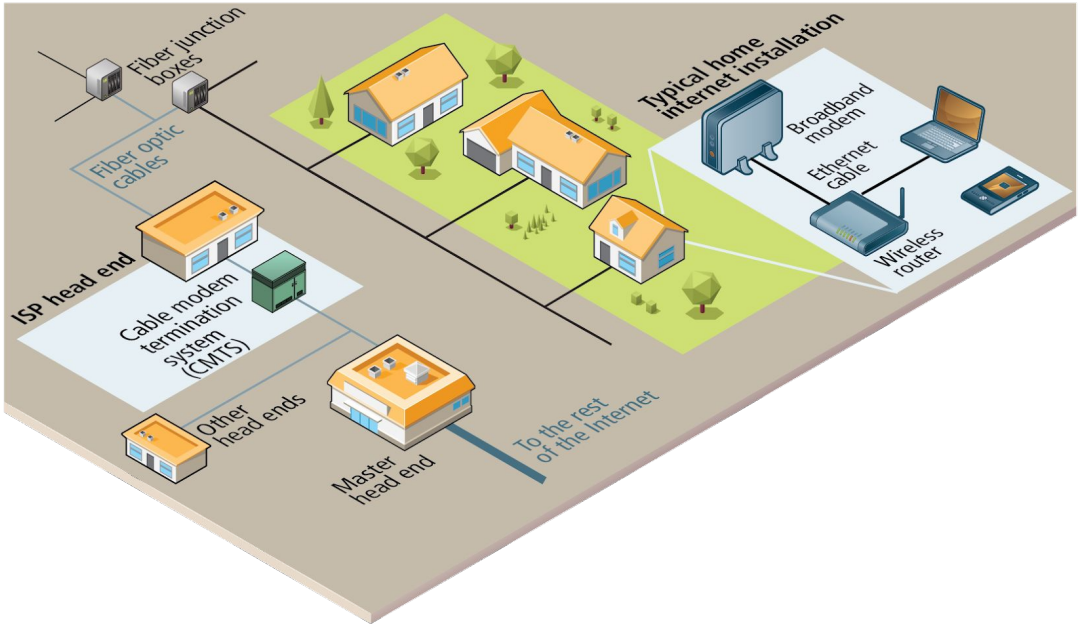
hundreds of thousands of server computers

and probably an equal number of routers, switches, and other networked devices,

along with many thousands of air conditioning units and specially-constructed server rooms and buildings.

From the Computer to the Local Provider

Our main experience of the hardware component of the Internet is that which we experience in our homes.



In the House

The **broadband modem** (also called a cable modem or DSL modem) is a bridge between the network hardware outside the house (typically controlled by a phone or cable company) and the network hardware inside the house.

These devices are often supplied by the ISP.

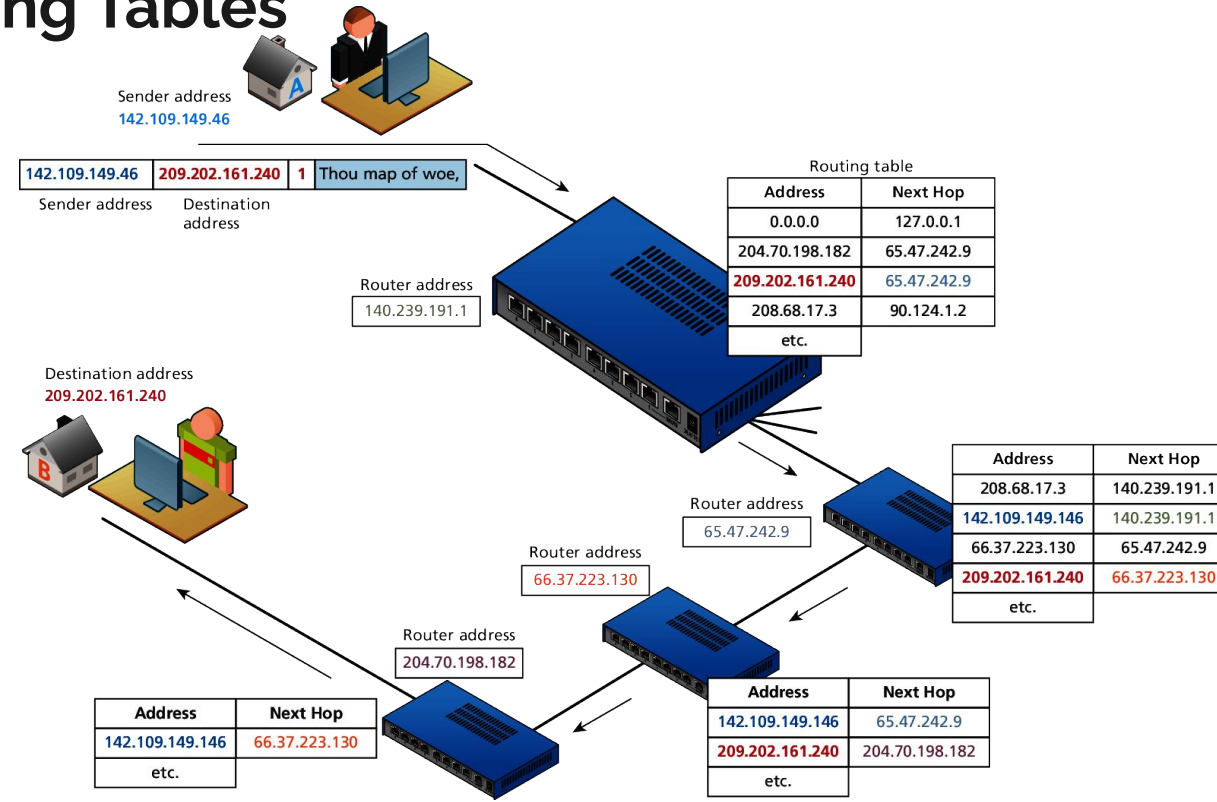
Routers

The wireless router is perhaps the most visible manifestation of the Internet in one's home, in that it is a device we typically need to purchase and install.

Routers are in fact one of the most important and ubiquitous hardware devices that makes the Internet work.

At its simplest, a **router** is a hardware device that forwards data packets from one network to another network.

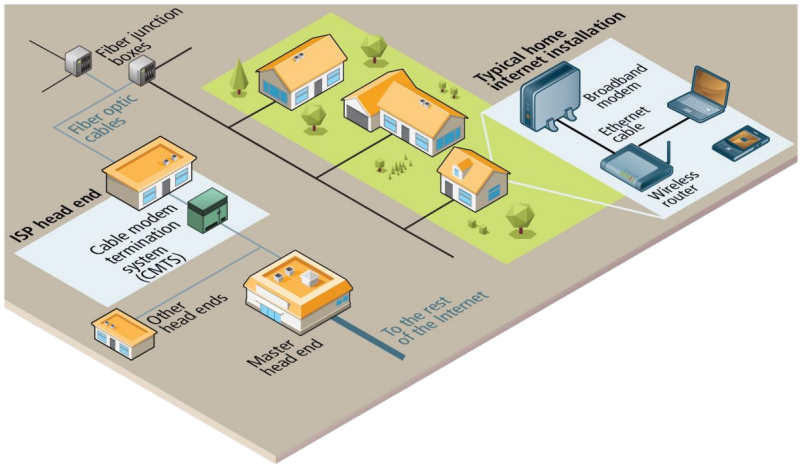
Routers and Routing Tables



Out of the House

Once we leave the confines of our own homes, the hardware of the Internet becomes much murkier.

In the illustration, the various neighborhood broadband cables (which are typically using copper, aluminum, or other metals) are aggregated and connected to fiber optic cable via fiber connection boxes.



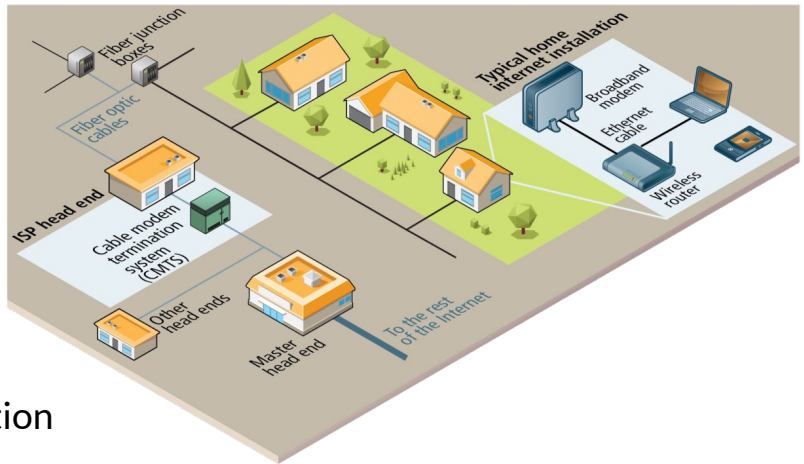
Fiber Optic Cable

Fiber optic cable (or simply optical fiber) is a glass-based wire that transmits light and has significantly greater bandwidth and speed in comparison to metal wires.

In some cities (or large buildings), you may have fiber optic cable going directly into individual buildings; in such a case the fiber junction box will reside in the building.

To the Provider

These fiber optic cables eventually make their way to an ISP's **head-end**, which is a facility that may contain a **cable modem termination system** (CMTS) or a digital subscriber line access multiplexer (DSLAM) in a DSL-based system.



From the Local Provider to the Ocean

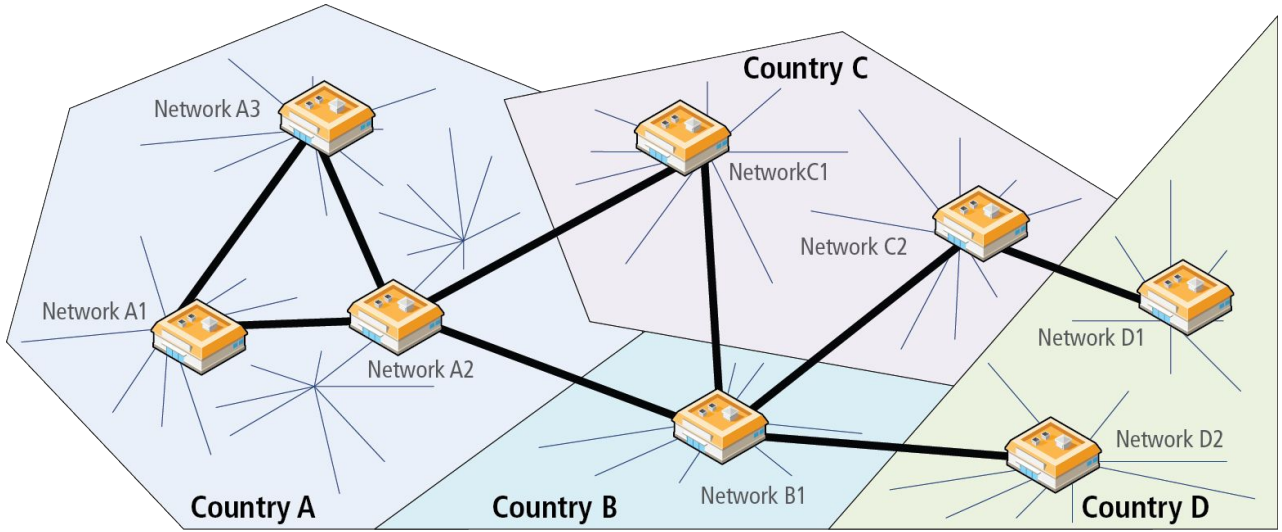
Eventually your ISP has to pass on your requests for Internet packets to other networks.

This intermediate step typically involves one or more regional network hubs.

Your ISP may have a large national network with optical fiber connecting most of the main cities in the country.

Some countries have multiple national or regional networks, each with their own optical network.

Connecting different networks within and between countries



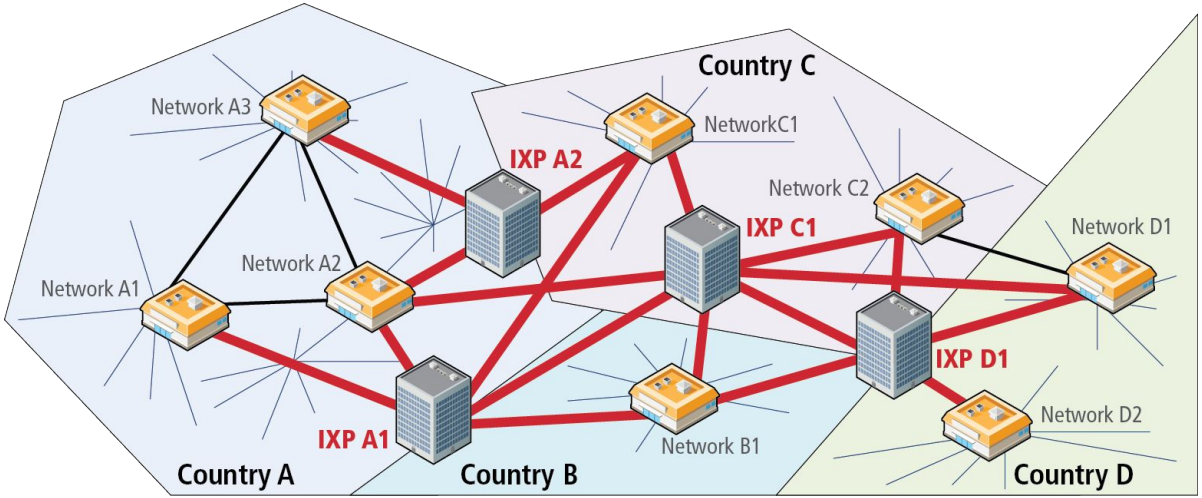
Internet Exchange Points

Connecting different networks

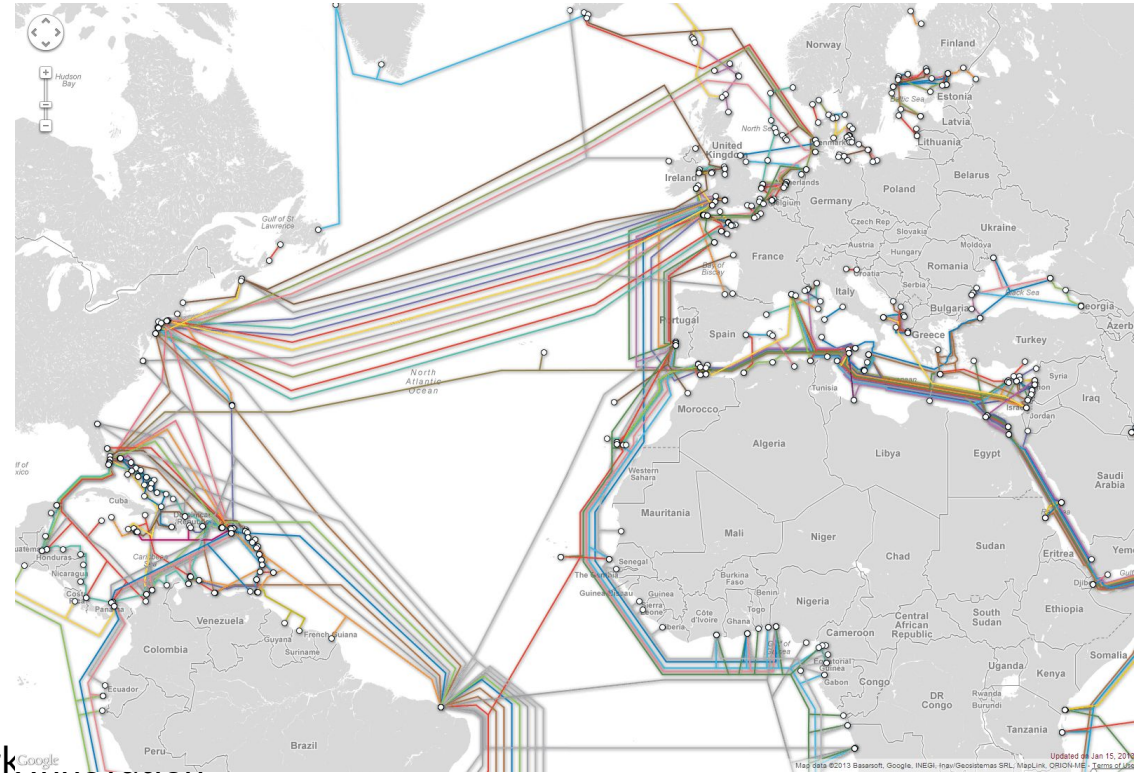
This type of network configuration began to change in the 2000s, as more and more networks began to interconnect with each other using an **Internet Exchange Point** (**IX** or **IXP**).

These IXPs allow different ISPs to **peer** with one another (that is, interconnect) in a shared facility, thereby improving performance for each partner in the peer relationship.

National and regional networks using Internet Exchange Points



Undersea fiber optic lines (courtesy TeleGeography)



Section 5 of 8

DOMAIN NAME SYSTEM (DNS)

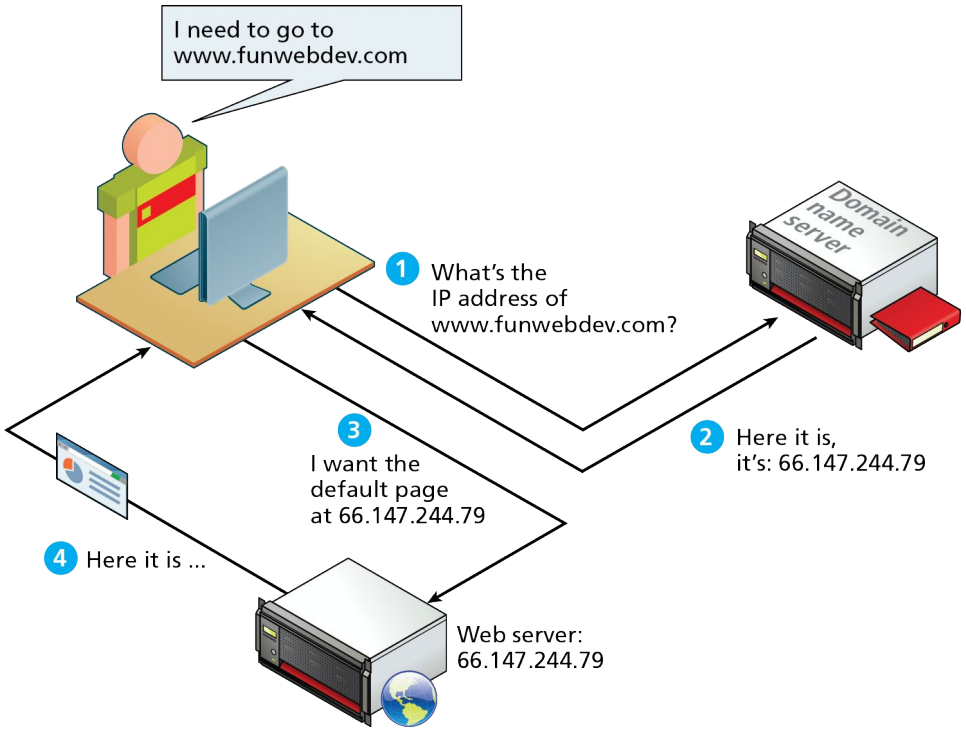
Daniel Awde - Park Innovation

Domain Name System

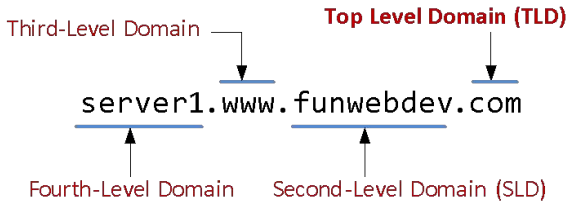
Why do we need it?

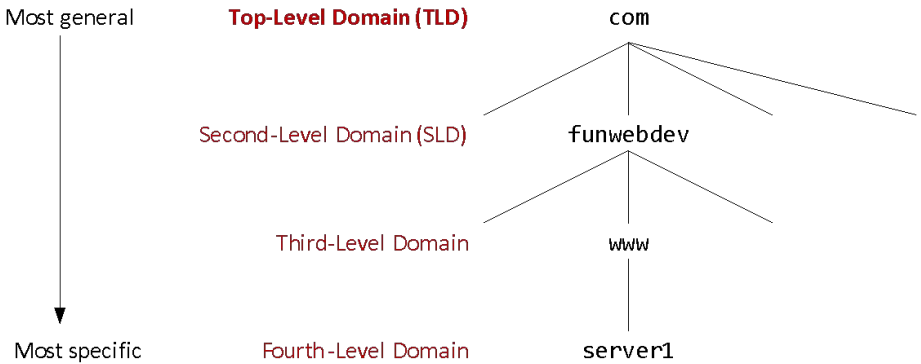
As elegant as IP addresses may be, human beings do not enjoy having to recall long strings of numbers. Instead of IP addresses, we use the **Domain Name System (DNS)**

DNS Overview



Domain Levels





Types of TLDs

Generic top-level domains (gTLD)

Country code top-level domain (ccTLD)

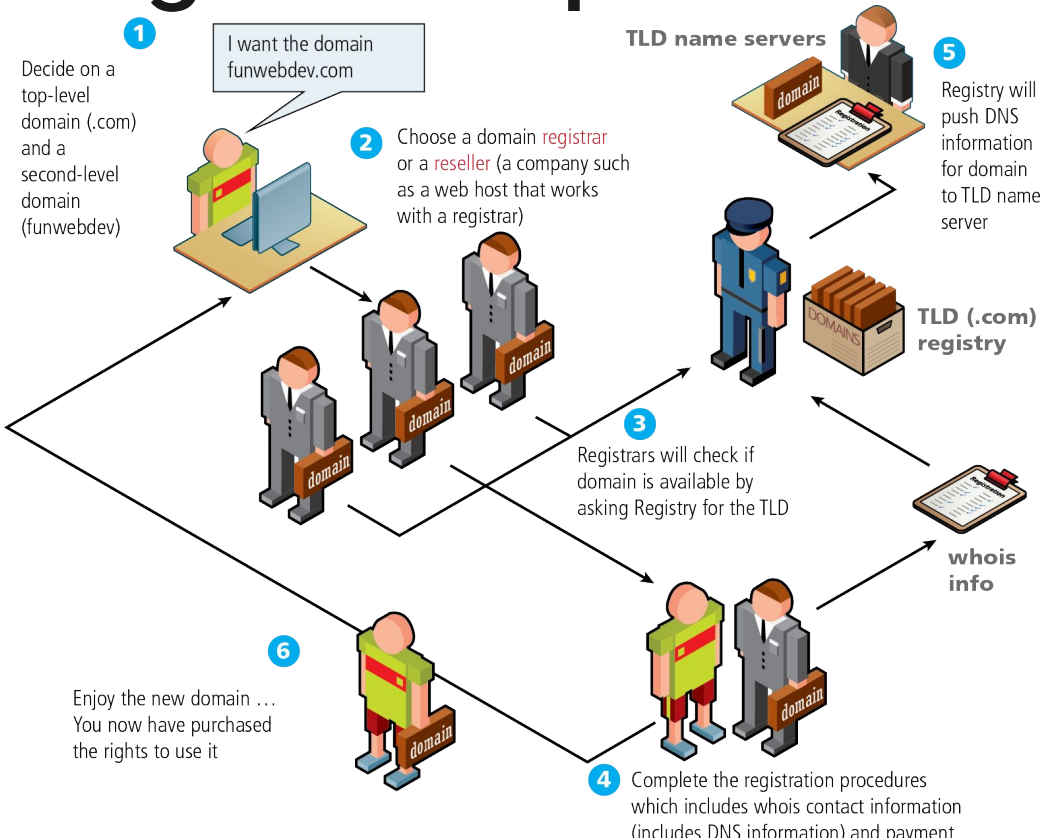
Name Registration

How are domain names assigned?

Special organizations or companies called **domain name registrars** manage the registration of domain names.

These domain name registrars are given permission to do so by the appropriate generic top-level domain (gTLD) registry and/or a country code top-level domain (ccTLD) registry.

Domain name registration process



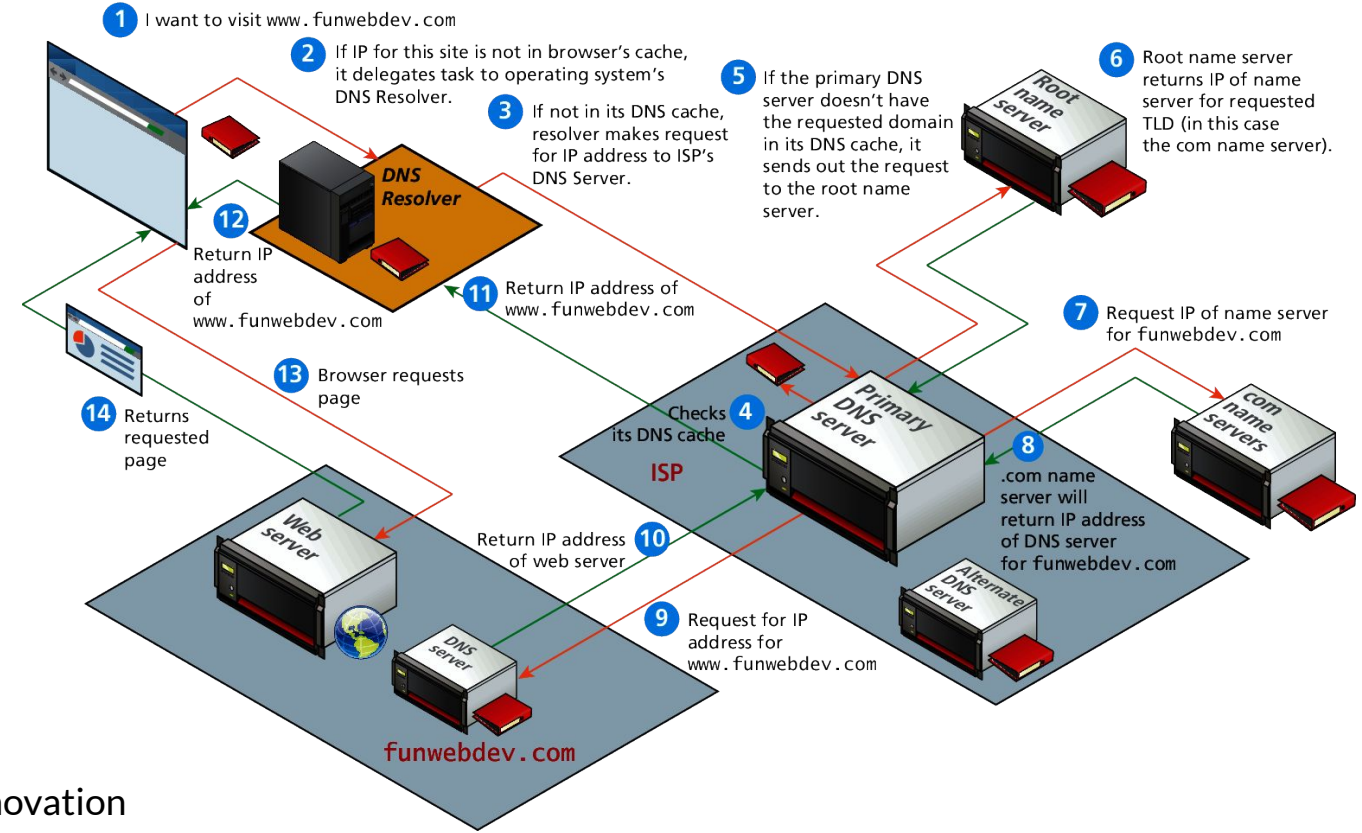
DNS Address Resolution

While domain names are certainly an easier way for users to reference a web site, eventually, your browser needs to know the IP address of the web site in order to request any resources from it.

The Domain Name System provides a mechanism for software to discover this numeric IP address.

This process is referred to here as **address resolution**.

Domain name address resolution process



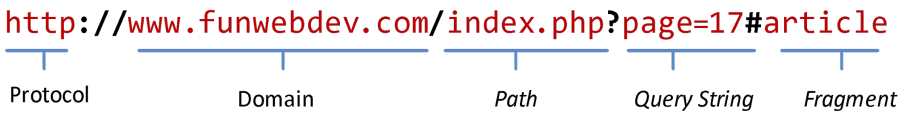
Section 6 of 8

UNIFORM RESOURCE LOCATORS (URL)

URL Components

In order to allow clients to request particular resources from the server, a naming mechanism is required so that the client knows how to ask the server for the file.

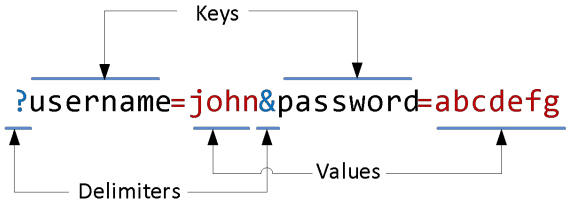
For the web that naming mechanism is the **Uniform Resource Locator** (URL).



Query String

Query strings will be covered in depth when we learn more about HTML forms and server-side programming.

They are the way of passing information such as user form input from the client to the server. In URL's they are encoded as key-value pairs delimited by “&” symbols and preceded by the “?” symbol.



Section 7 of 8

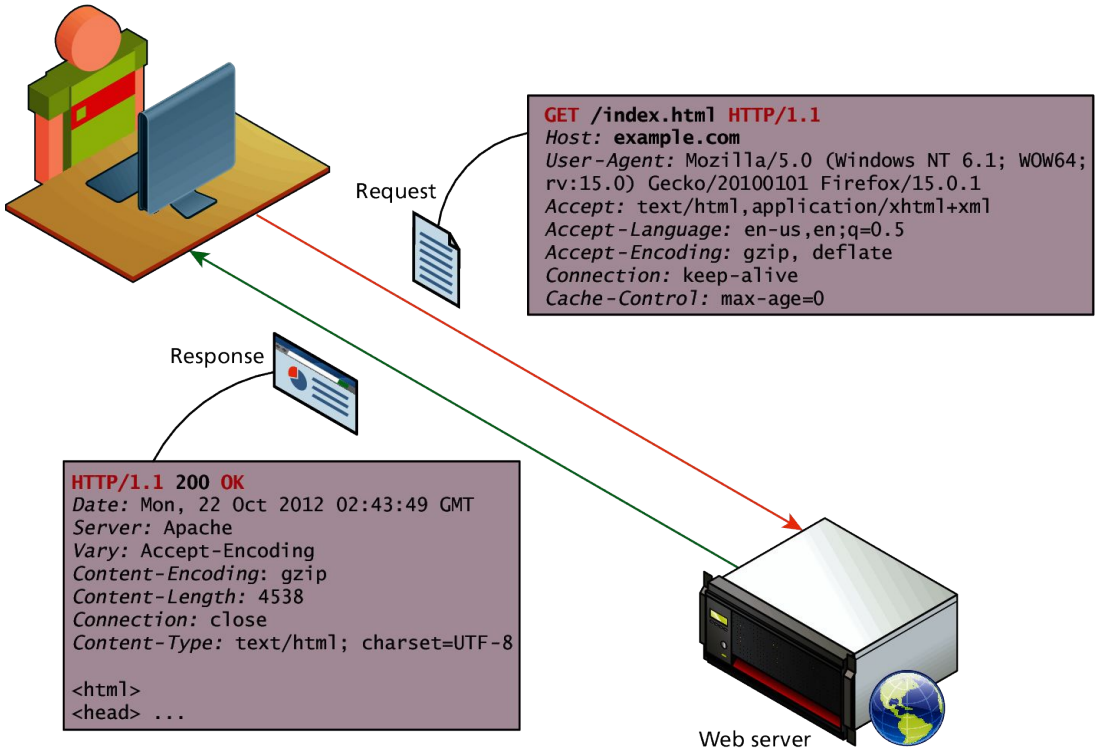
HYPERTEXT TRANSFER PROTOCOL (HTTP)

HTTP

The HTTP protocol establishes a TCP connection on port 80 (by default).

The server waits for the request, and then responds with a response code, headers and an optional message (which can include files).

HTTP


Request
GET /index.html HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:15.0) Gecko/20100101 Firefox/15.0.1
Accept: text/html,application/xhtml+xml
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cache-Control: max-age=0
Response
HTTP/1.1 200 OK
Date: Mon, 22 Oct 2012 02:43:49 GMT
Server: Apache
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 4538
Connection: close
Content-Type: text/html; charset=UTF-8
<html>
<head> ...
Web server

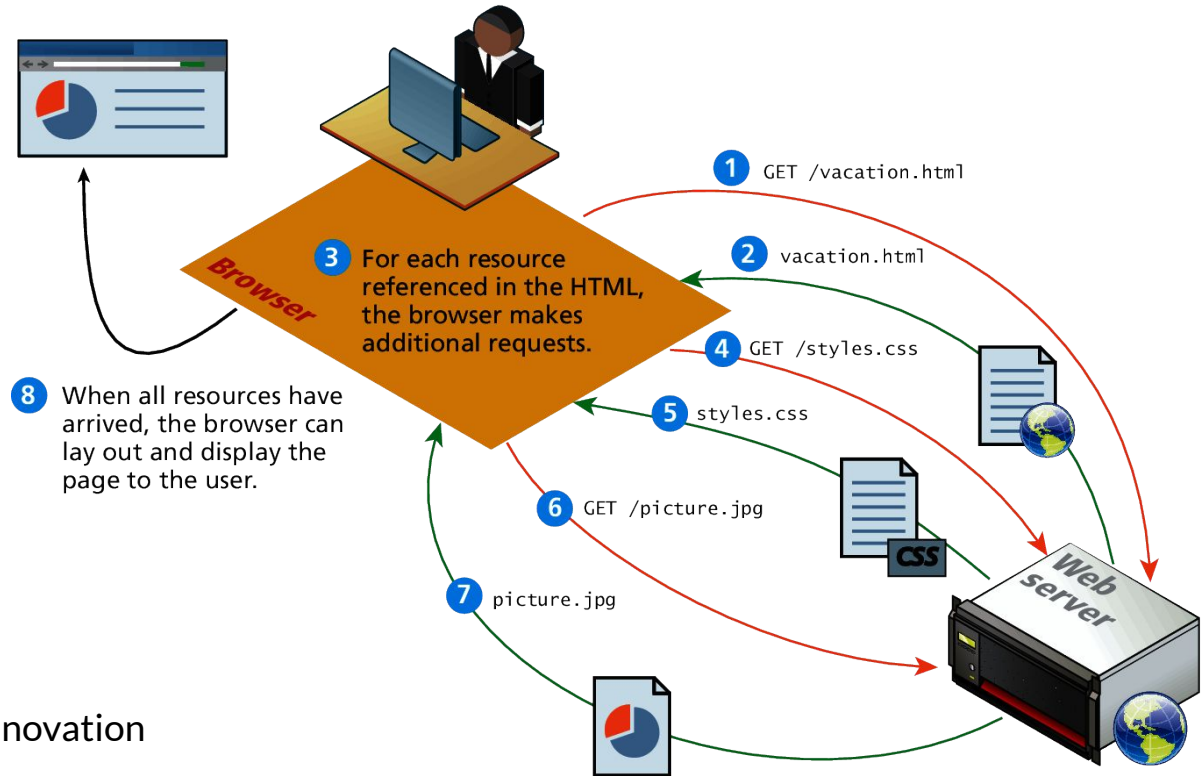
Web Requests

While we as web users might be tempted to think of an entire page being returned in a single HTTP response, this is not in fact what happens.

In reality the experience of seeing a single web page is facilitated by the client's browser which requests the initial HTML page, then parses the returned HTML to find all the resources referenced from within it, like images, style sheets and scripts.

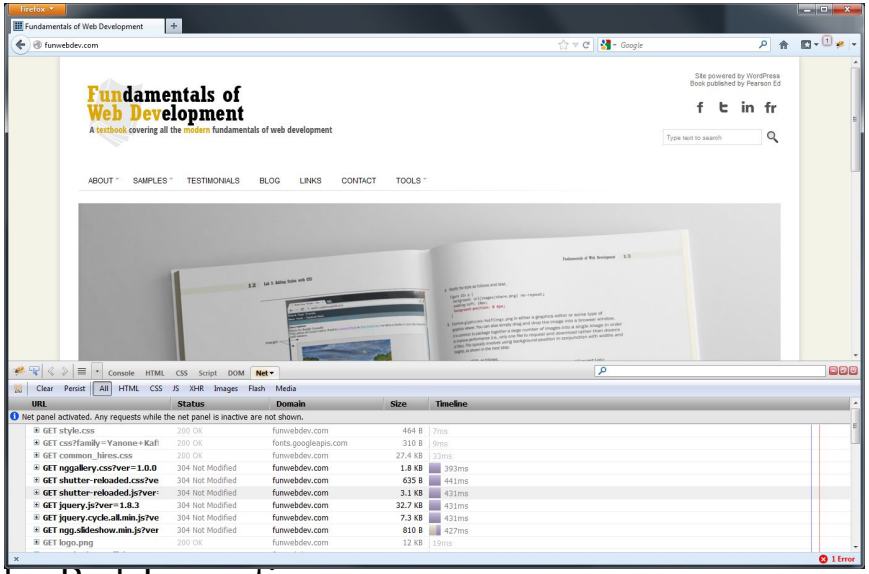
Only when all the files have been retrieved is the page fully loaded for the user

Browser parsing HTML and making subsequent requests



Browser Tools for HTTP

Modern browsers provide the developer with tools that can help us understand the HTTP traffic for a given page.

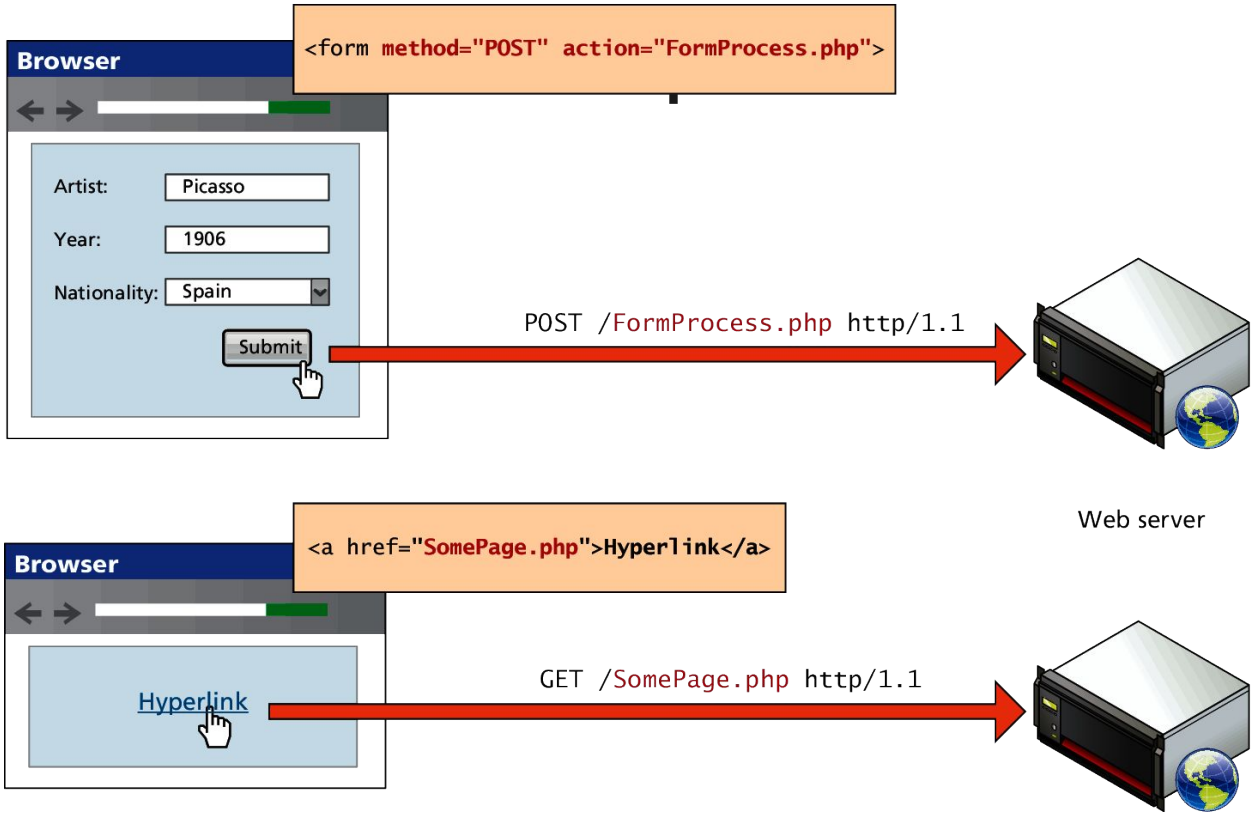


HTTP Request Methods

The HTTP protocol defines several different types of requests, each with a different intent and characteristics.

The most common requests are the GET and POST request, along with the HEAD request.

Other requests, such as PUT, DELETE, CONNECT, TRACE and OPTIONS are seldom used, and are not covered here.


Browser
<form method="POST" action="FormProcess.php">
Artist:
Picasso
Year:
1906
Nationality:
Spain
Submit
POST /FormProcess.php http/1.1
Web server
Browser
Hyperlink
Hyperlink
GET /SomePage.php http/1.1

Section 8 of 8

WEB SERVERS

Daniel Awde - Park Innovation

Web Servers

A **web server** is, at a fundamental level, nothing more than a computer that responds to HTTP requests.

Web Stack

Regardless of the physical characteristics of the server, one must choose an application stack to run a website.

This stack will include an operating system, web server software, a database and a scripting language to process dynamic requests.

LAMP Software Stack

Throughout this textbook we will rely on the **LAMP software stack**, which refers to the Linux operating system, Apache web server, MySQL database, and PHP scripting language

WISA software stack

Many corporations, for instance, make use of the Microsoft **WISA software stack**, which refers to Windows operating system, IIS web server, SQL Server database, and the ASP.NET server-side development technologies.

HTML 1: Overview

Daniel Awde - Park Innovation

Section 1 of 6

HTML DEFINED + ITS HISTORY

Daniel Awde - Park Innovation

Brief History of HTML

Did we mention that this will be brief?

- ARPANET of the late 1960s
- jump quickly to the first public specification of the HTML by Tim Berners-Lee in 1991
- HTML's codification by the World-Wide Web Consortium (better known as the **W3C**) in 1997.

HTML Syntax

What is a markup language?

HTML is defined as a **markup language**.

- A markup language is simply a way of annotating a document in such a way to make the annotations distinct from the text being annotated.
- The term comes from the days of print, when editors would write instructions on manuscript pages that might be revision instructions to the author or copy editor.

Markup

What is it again?

At its simplest, **markup** is a way to indicate *information about the content*

- This “information about content” in HTML is implemented via **tags** (aka elements).
- The markup in the previous slide consists of the red text and the various circles and arrows on the one page, and the little yellow sticky notes on the other.
- HTML does the same thing but uses textual tags.

What is the W3C?

Standards

The W3C is the main standards organization for the World Wide Web.

To promotes compatibility the W3C produces **recommendations** (also called **specifications**).

In 1998, the W3C turned its attention to a new specification called XHTML 1.0, which was a version of HTML that used stricter XML (Extensible Markup Language) syntax rules.

XHTML

Partying like it's 1999

The goal of XHTML with its strict rules was to make page rendering more predictable by forcing web authors to create web pages without syntax errors.

XHTML

You too can be strict

The XML-based syntax rules for XHTML are pretty easy to follow.

The main rules are:

- lowercase tag names,
- attributes always within quotes,
- and all elements must have a closing element (or be self-closing).

XHTML

Two versions

To help web authors, two versions of XHTML were created:

XHTML 1.0 Strict and **XHTML 1.0 Transitional**.

- The **strict** version was meant to be rendered by a browser using the strict syntax rules and tag support described by the W3C XHTML 1.0 Strict specification.
- The **transitional** recommendation is a more forgiving flavor of XHTML, and was meant to act as a temporary transition to the eventual global adoption of XHTML Strict.

Standards Movement

Following a standard is a good thing

During much of the 2000s, the focus in the professional web development community was on standards: that is, on limiting oneself to the W3C specification for XHTML.

XHTML 2.0 and WHATWG

Where did it go?

In the mid 2000s, XHTML 2.0 proposed a revolutionary and substantial change to HTML.

- backwards compatibility with HTML and XHTML 1.0 was dropped.
- Browsers would become significantly less forgiving of invalid markup.

At around the same time, a group of developers at Opera and Mozilla formed the **WHATWG** (Web Hypertext Application Technology Working Group) group within the W3C.

This group was not convinced that the W3C's embrace of XML and its abandonment of backwards-compatibility was the best way forward for the web.

HTML5

The new hotness

By 2009, the W3C stopped work on XHTML 2.0 and instead adopted the work done by WHATWG and named it HTML5.

HTML5

Three main aims

There are three main aims to HTML5:

- Specify unambiguously how browsers should deal with invalid markup.
- Provide an open, non-proprietary programming framework (via Javascript) for creating rich web applications.
- Be backwards compatible with the existing web.

HTML5

It evolves

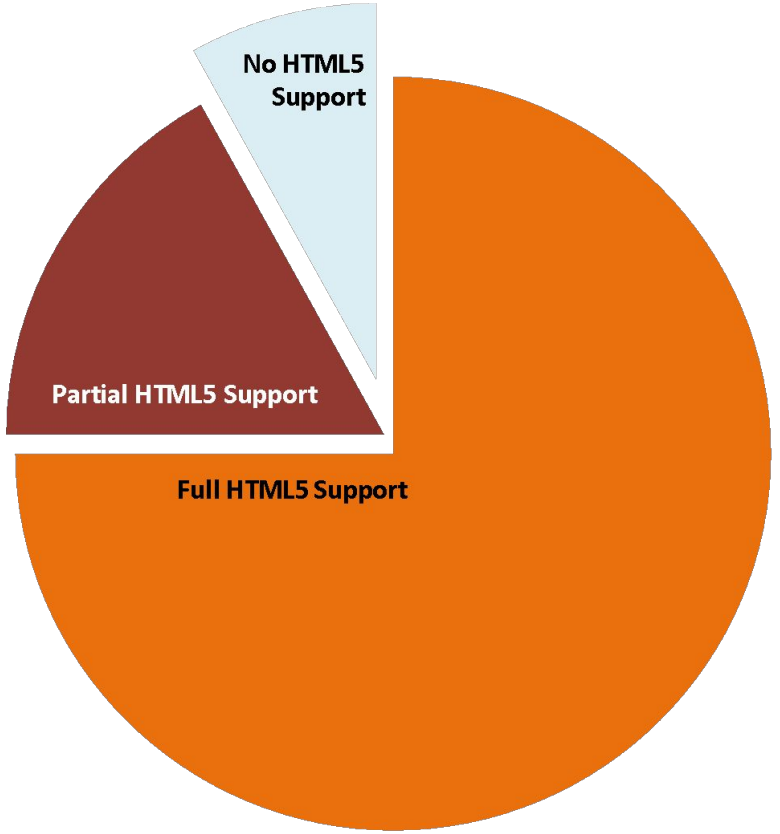
While parts of the HTML5 are still being finalized, all of the major browser manufacturers have at least partially embraced HTML5.

Certainly not all browsers and all versions support every feature of HTML5.

This is in fact by design. HTML in HTML5 is now a living language: that is, it is a language that evolves and develops over time.

As such, every browser will support a gradually increasing subset of HTML5 capabilities

HTML5 Support in Browsers



Section 2 of 6

HTML SYNTAX

Daniel Awde - Park Innovation

Elements and Attributes

More syntax

HTML documents are composed of textual content and HTML elements.

An **HTML element** can contain text, other elements, or be empty. It is identified in the HTML document by tags.

HTML elements can also contain attributes. An **HTML attribute** is a name=value pair that provides more information about the HTML element.

In XHTML, attribute values had to be enclosed in quotes; in HTML5, the quotes are optional.

What HTML lets you do

- Insert images using the **<img>** tag
- Create links with the **<a>** tag
- Create lists with the **<ul>, ** and **<li>** tags
- Create headings with **<H1>, <H2>, …, <H6>**
- Define metatdata with **<meta>** tag
- And much more…

Elements and Attributes

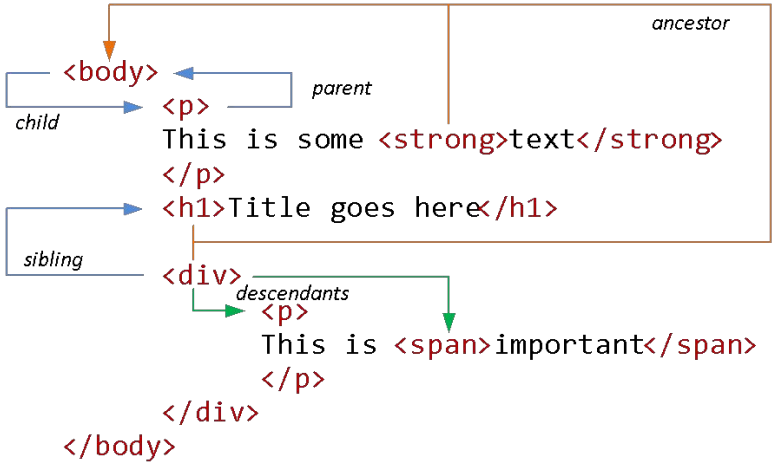
Nesting HTML elements

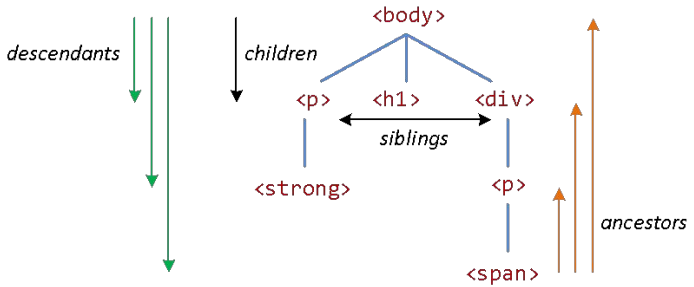
Often an HTML element will contain other HTML elements.

In such a case, the container element is said to be a parent of the contained, or child, element.

Any elements contained within the child are said to be **descendents** of the parent element; likewise, any given child element, may have a variety of **ancestors.**

Hierarchy of elements

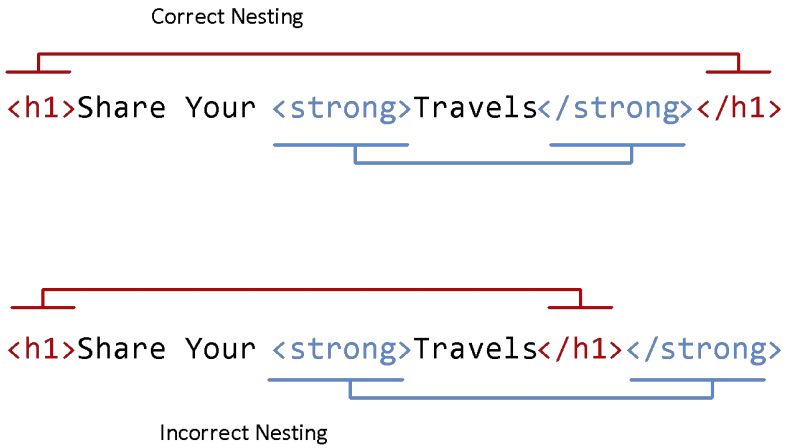




Nesting HTML elements

In order to properly construct a hierarchy of elements, your browser expects each HTML nested element to be properly nested.

That is, a child's ending tag must occur before its parent's ending tag.



Section 3 of 6

SEMANTIC MARKUP

Daniel Awde - Park Innovation

Semantic Markup

What does it mean?

Over the past decade, a strong and broad consensus has grown around the belief that HTML documents should **only** focus on the structure of the document.

Information about how the content should look when it is displayed in the browser is best left to CSS (Cascading Style Sheets).

Semantic Markup

As a consequence, beginning HTML authors are often counseled to create **semantic HTML** documents.

That is, an HTML document should not describe how to visually present content, but only describe its content's structural semantics or meaning.

Structure

Structure is a vital way of communicating information in paper and electronic documents.

All of the tags that we will examine in this presentation are used to describe the basic structural information in a document, such as articles, headings, lists, paragraphs, links, images, navigation, footers, and so on.

Semantic Markup

Its advantages

Eliminating presentation-oriented markup and writing semantic HTML markup has a variety of important advantages:

Maintainability. Semantic markup is easier to update and change than web pages that contain a great deal of presentation markup.

Faster. Semantic web pages are typically quicker to author and faster to download.

Accessibility. Visiting a web page using voice reading software can be a very frustrating experience if the site does not use semantic markup.

Search engine optimization. Semantic markup provides better instructions for search engines: it tells them what things are important content on the site.

Section 4 of 6

STRUCTURE OF HTML

Daniel Awde - Park Innovation

Simplest HTML document

```
<!DOCTYPE html >
<title >A Very Small Document </title>
<p>This is a simple document with not much content  </p>
```

The <title> element (Item 1) is used to provide a broad description of the content. The title is not displayed within the browser window. Instead, the title is typically displayed by the browser in its window and/or tab.

A more complete document

1 DOCTYPE

(short for **Document Type Definition**)

Tells the browser (or any other client software that is reading this HTML document) what type of document it is about to process.

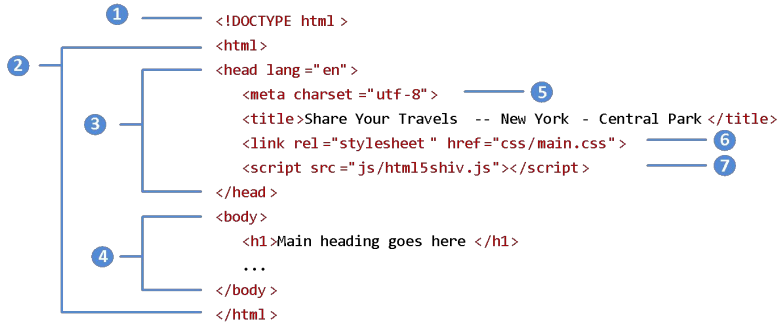
Notice that it does not indicate what version of HTML is contained within the document: it only specifies that it contains HTML.

HTML, Head, and Body

HTML5 does not require the use of the <html>, <head>, and <body>.

However, in XHTML they were required, and most web authors continue to use them.

The <html> element is sometimes called the **root element** as it contains all the other HTML elements in the document.



Head and Body

HTML pages are divided into two sections: the **head** and the **body**, which correspond to the <head> and <body> elements.

The head contains descriptive elements *about* the document

The body contains content that will be displayed by the browser.

Inside the head

There are no brains

5

You will notice that the <head> element contains a variety of additional elements.

The first of these is the <meta> element. Our example declares that the character encoding for the document is UTF-8.

Inside the head

No brains but metas, styles and javascripts

6. Our example specifies an external CSS style sheet file that is used with this document.

7. It also references an external Javascript file.

Section 5 of 6

QUICK TOUR OF HTML

Daniel Awde - Park Innovation

Why a quick tour?

HTML5 contains many structural and presentation elements – too many to completely cover in this presentation.

Rather than comprehensively cover all these elements, this presentation will provide a quick overview of the most common elements.

Sample Document

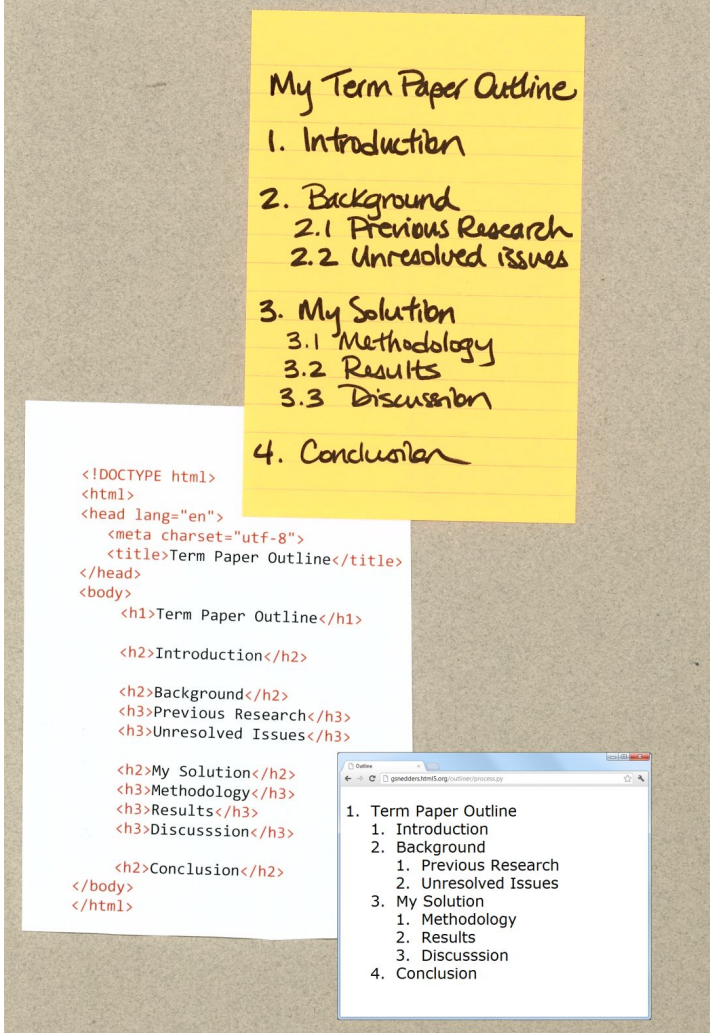
```
<body>
   <h1>Share Your Travels </h1>
   <h2>New York  - Central Park </h2>
   <p>Photo by Randy Connolly </p>
   <p>This photo of Conservatory Pond in
      <a href="http://www.centralpark .com/">Central Park </a>
      New York City was taken on October  22, 2011 with a
      <strong>Canon EOS  30D</strong> camera .
   </p>
   <img src="images/central -park.jpg" alt="Central Park " />

   <h3>Reviews</h3>
   <div>
      <p>By Ricardo on  <time>September  15, 2012</time></p>
      <p>Easy on the HDR buddy .</p>
   </div>

   <div>
      <p>By Susan on  <time>October  1, 2012</time></p>
      <p>I love Central Park .</p>
   </div>

   <p><small>Copyright  &copy; 2012 Share Your Travels </small></p>
</body>
```

1 Headings

<h1>, <h2>, <h3>, etc

HTML provides six levels of heading (**h1, h2, h3**, …), with the higher heading number indicating a heading of less importance.

Headings are an essential way for document authors use to show their readers the structure of the document.



Headings

The browser has its own default styling for each heading level.

However, these are easily modified and customized via CSS.

Headings

Be semantically accurate

In practice, specify a heading level that is semantically accurate.

Do not choose a heading level because of its default presentation

- e.g., choosing <h3> because you want your text to be bold and 16pt

Rather, choose the heading level because it is appropriate

- e.g., choosing <h3> because it is a third level heading and not a primary or secondary heading

2 Paragraphs

<p>

Paragraphs are the most basic unit of text in an HTML document.

Notice that the **<p>** tag is a container and can contain HTML and other **inline HTML elements**

inline HTML elements refers to HTML elements that do not cause a paragraph break but are part of the regular “flow” of the text.

6 Divisions

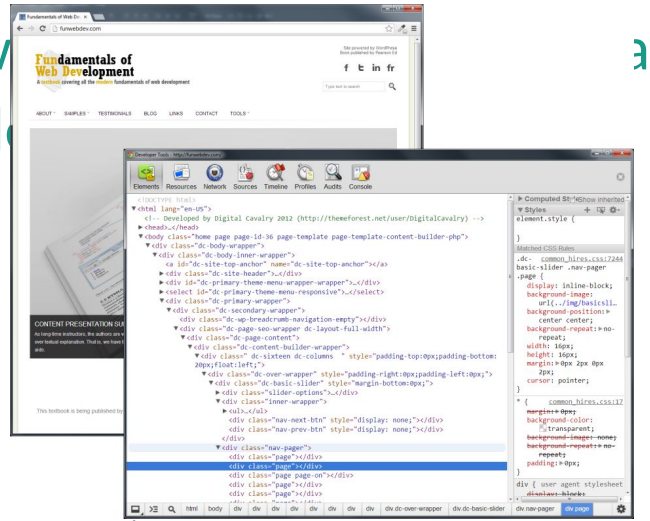
This ** ** tag is also a container element and is used to create a logical grouping of content

- The element has no intrinsic presentation.
- It is frequently used in contemporary CSS-based layouts to mark out sections.

Using div elements

Can you say “div-tastic”

HTML5 has a variety of new semantic elements (which we will examine later) that can be used to reduce somewhat the confusing mass of div within divs w al of contemporary web d



3 Links

<a>

Links are created using the **<a>** element (the “a” stands for anchor).

A link has two main parts: the destination and the label.

```
<a href="http://www.centralpark.com">Central Park</a>
```

Destination

Label (text)

```
<a href="index.html"><img src="logo.gif" /></a>
```

Label (image)

Types of Links

You can use the anchor element to create a wide range of links:

- Links to external sites (or to individual resources such as images or movies on an external site).
- Links to other pages or resources within the current site.
- Links to other places within the current page.
- Links to particular locations on another page.
- Links that are instructions to the browser to start the user's email program.
- Links that are instructions to the browser to execute a Javascript function.

Different link destinations

Link Text

Some guidance … or … don't "Click Here"

Links with the label "Click Here" were once a staple of the web.

Today, such links are frowned upon, since:

- they do not tell users where the link will take them
- as a verb "click" is becoming increasingly inaccurate when one takes into account the growth of mobile browsers.

Instead, textual link labels should be descriptive.

~~"Click here to see the race results"~~

"Race Results" or **"See Race Results".**

URL Absolute Referencing

For external resources

When referencing a page or resource on an external site, a full **absolute reference** is required: that is,

- the protocol (typically, http://),
- the domain name,
- any paths, and then finally
- the file name of the desired resource.

URL Relative Referencing

An essential skill

We also need to be able to successfully reference files within our site.

This requires learning the syntax for so-called **relative referencing**.

When referencing a resource that is on the same server as your HTML document, then you can use briefer relative referencing. If the URL does not include the “http://” then the browser will request the current server for the file.

URL Relative Referencing

If all the resources for the site reside within the same **directory** (also referred to as a **folder**), then you can reference those other resources simply via their filename.

However, most real-world sites contain too many files to put them all within a single directory.

For these situations, a relative pathname is required along with the filename.

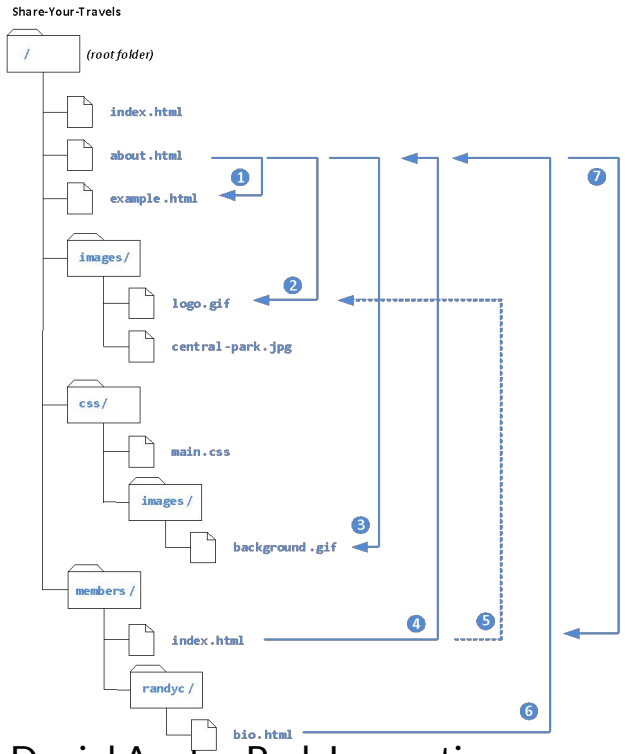
The **pathname** tells the browser where to locate the file on the server.

Pathnames

Pathnames on the web follow Unix conventions.

- Forward slashes (“/”) are used to separate directory names from each other and from file names.
- Double-periods (“..”) are used to reference a directory “above” the current one in the directory tree.

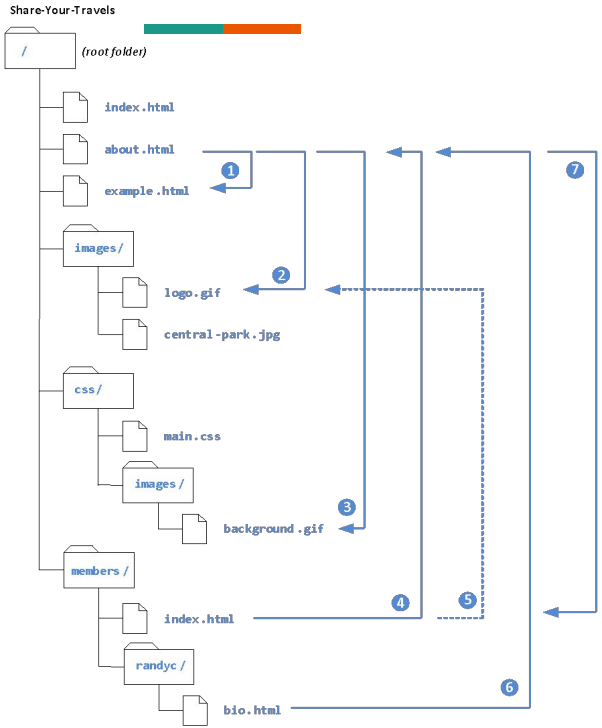
URL Relative Referencing



	Relative Link Type	Example
1	**Same Directory** To link to a file within the same folder, simply use the file name.	To link to example.html from about.html (in Figure 2.18), use: `<a href="example.html">`
2	**Child Directory** To link to a file within a subdirectory, use the name of the subdirectory and a slash before the file name.	To link to logo.gif from about.html, use: `<a href="images/logo.gif">`
3	**Grandchild/Descendant Directory** To link to a file that is multiple subdirectories *below* the current one, construct the full path by including each subdirectory name (separated by slashes) before the file name.	To link to background.gif from about.html, use: `<a href="css/images/background.gif">`

4	**Parent/Ancestor Directory**
	
	Use "../" to reference a folder *above* the current one. If trying to reference a file several levels above the current one, simply string together multiple "../".
		To link to about.html from index.html in members, use:
		
		`<a href="../about.html">`
		
		To link to about.html from bio.html, use:
		
		`<a href="../../about.html">`

URL Relative Referencing



5 Sibling Directory

Use "../"to move up to the appropriate level, and then use the same technique as for child or grandchild directories.

To link to logo.gif from index.html in members, use:

```
<a href="../images/about.html">
```

To link to background.gif from bio.html, use:

```
<a href="../../css/images/background.gif">
```

6 Root Reference

An alternative approach for ancestor and sibling references is to use the so-called **root reference** approach. In this approach, begin the reference with the root reference ("/") and then use the same technique as for child or grandchild directories. **Note that these will only work on the server! That is, they will not work when you test it out on your local machine.**

To link to about.html from bio.html, use:

```
<a href="/about.html">
```

To link to background.gif from bio.html, use:

```
<a href="/images/background.gif">
```

7 Default Document

Web servers allow references to directory names without file names. In such a case, the web server will serve the default document, which is usually a file called index.html (apache) or default.html (IIS). **Again, this will only generally work on the web server.**

To link to index.html in members from about.html, use either:

```
<a href="members">
```

Or

```
<a href="/members">
```

Inline Text Elements

Do not disrupt the flow

Inline elements do not disrupt the flow of text (i.e., cause a line break).

HTML5 defines over 30 of these elements.

e.g., <a>,
, ,

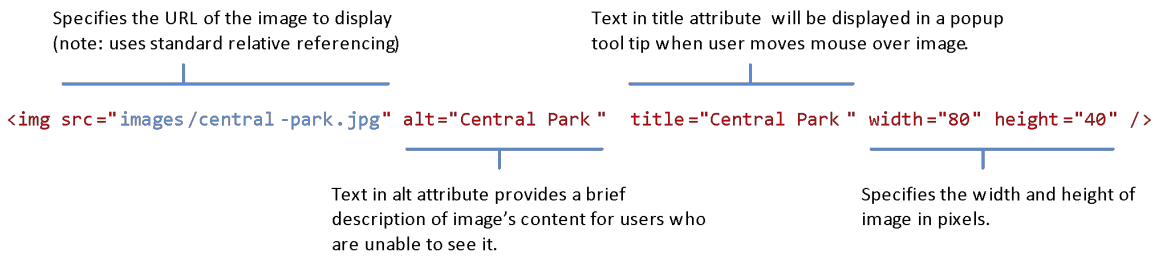
Images

While the **<img>** tag is the oldest method for displaying an image, it is not the only way.

For purely decorative images, such as background gradients and patterns, logos, border art, and so on, it makes semantic sense to keep such images out of the markup and in CSS where they more rightly belong.

But when the images are content, such as in the images in a gallery or the image of a product in a product details page, then the <img> tag is the semantically appropriate approach.

Images


Specifies the URL of the image to display
(note: uses standard relative referencing)
Text in title attribute will be displayed in a popup
tool tip when user moves mouse over image.

Text in alt attribute provides a brief
description of image's content for users who
are unable to see it.
Specifies the width and height of
image in pixels.

Lists

HTML provides three types of lists

Unordered lists. Collections of items in no particular order; these are by default rendered by the browser as a bulleted list.

Ordered lists. Collections of items that have a set order; these are by default rendered by the browser as a numbered list.

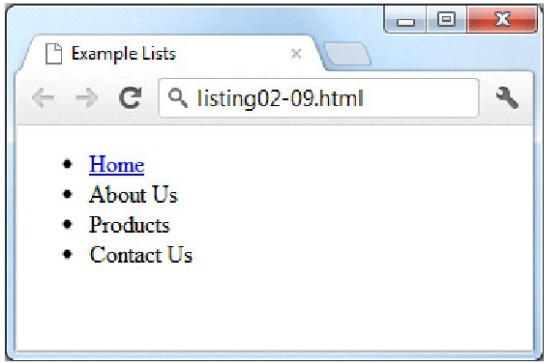
Definition lists. Collection of name and definition pairs. These tend to be used infrequently. Perhaps the most common example would be a FAQ list.

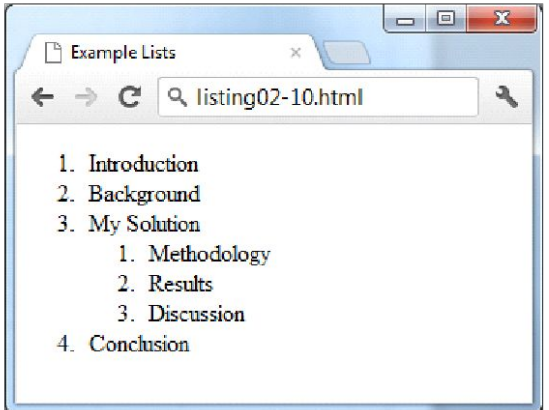
Lists

Notice that the list item element can contain other HTML elements

```
<ul>
   <li><a href="index.html">Home</a></li>
   <li>About Us</li>
   <li>Products</li>
   <li>Contact Us</li>
</ul>
```

```
<ol>
   <li>Introduction</li>
   <li>Background</li>
   <li>My Solution</li>
   <li>
      <ol>
         <li>Methodology</li>
         <li>Results</li>
         <li>Discussion</li>
      </ol>
   </li>
   <li>Conclusion</li>
</ol>
```





Character Entities

These are special characters for symbols for which there is either no way easy way to type in via a keyboard (such as the copyright symbol or accented characters) or which have a reserved meaning in HTML (for instance the “<” or “>” symbols).

They can be used in an HTML document by using the entity name or the entity number.

e.g.,   and ©

Section 6 of 6

HTML SEMANTIC ELEMENTS

Daniel Awde - Park Innovation

HTML5 Semantic Elements

Why are they needed?

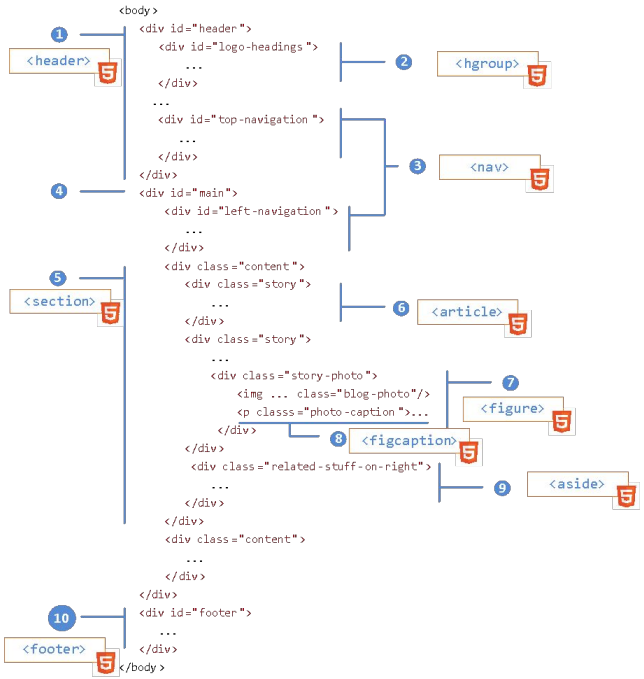
One substantial problem with modern, pre-HTML5 semantic markup:

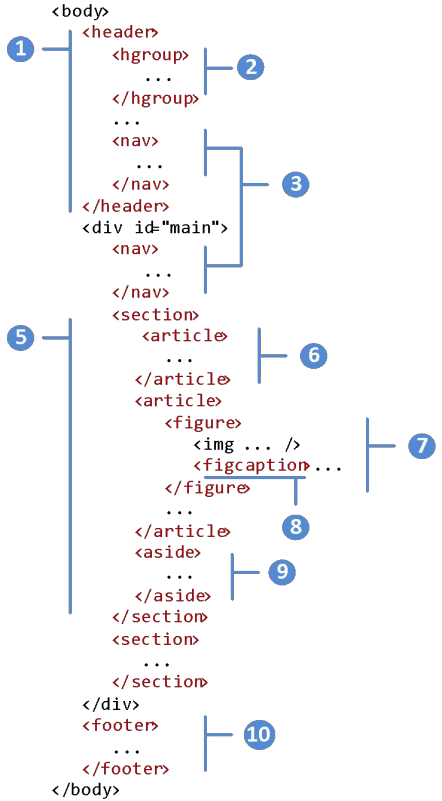
> most complex web sites are absolutely packed solid with elements.

Unfortunately, all these elements can make the resulting markup confusing and hard to modify.

Developers typically try to bring some sense and order to the chaos by using id or class names that provide some clue as to their meaning.

XHTML versus HTML5





Header and Footer

<header> <footer>

Most web site pages have a recognizable header and footer section.

Typically the **header** contains

- the site logo
- title (and perhaps additional subtitles or taglines)
- horizontal navigation links, and
- perhaps one or two horizontal banners.

Header and Footer

<header> <footer>

The typical footer contains less important material, such as

- smaller text versions of the navigation,
- copyright notices,
- information about the site's privacy policy, and
- perhaps twitter feeds or links to other social sites.

Header and Footer

Both the HTML5 <header> and <footer> element can be used not only for *page* headers and footers, they can also be used for header and footer elements within other HTML5 containers, such as <article> or <section>.

```
<header>
   <img src="logo.gif" alt="logo" />
   <h1>Fundamentals of Web Development</h1>
   ...
</header>
<article>
   <header>
       <h2>HTML5 Semantic Structure Elements </h2>
            <p>By <em>Randy Connolly</em></p>
            <p><time>September 30, 2012</time></p>
   </header>
   ...
</article>
```

2 Heading Groups

<hgroup>

The <hgroup> element can be used to group related headings together within one container.

```
<header>
   <hgroup>
       <h1>Chapter Two: HTML 1</h1>
            <h2>An Introduction</h2>
   </hgroup>
</header>
<article>
   <hgroup>
      <h2>HTML5 Semantic Structure Elements </h2>
          <h3>Overview</h3>
   </hgroup>
</article>
```

3 Navigation

`<nav>`

The **<nav>** element represents a section of a page that contains links to other pages or to other parts within the same page.

Like the other new HTML5 semantic elements, the browser does not apply any special presentation to the <nav> element.

The <nav> element was intended to be used for major navigation blocks, presumably the global and secondary navigation systems.

Navigation

```
<header>
        <img src="logo.gif" alt="logo" />
        <h1>Fundamentals of Web Development</h1>
        <nav role="navigation">
                <ul>
                        <li><a href="index.html">Home</a></li>
                        <li><a href="about.html">About Us</a></li>
                        <li><a href="browse.html">Browse</a></li>
                </ul>
        </nav>
</header>
```

5 6

Articles and Sections

<article> <section>

The **<article>** element represents a section of content that forms an independent part of a document or site; for example, a magazine or newspaper article, or a blog entry.

The **<section>** element represents a section of a document, typically with a title or heading.

Articles and Sections

According to the W3C, **<section>** is a much broader element, while the **<article>** element is to be used for blocks of content that could potentially be read or consumed independently of the other content on the page.

Sections versus Divs

How to decide which to use

The WHATWG specification warns readers that the <section> element is **not** a generic container element. HTML already has the element for such uses.

When an element is needed only for styling purposes or as a convenience for scripting, it makes sense to use the element instead.

Another way to help you decide whether or not to use the <section> element is to ask yourself if it is appropriate for the element's contents to be listed explicitly in the document's outline.

> If so, then use a <section>; otherwise use a .

7 8

Figure and Figure Captions

<figure> <figcaption>

The W3C Recommendation indicates that the <figure> element can be used not just for images but for any type of *essential* content that could be moved to a different location in the page or document and the rest of the document would still make sense.

Figure and Figure Captions

Note however …

The **<figure>** element should **not** be used to wrap every image.

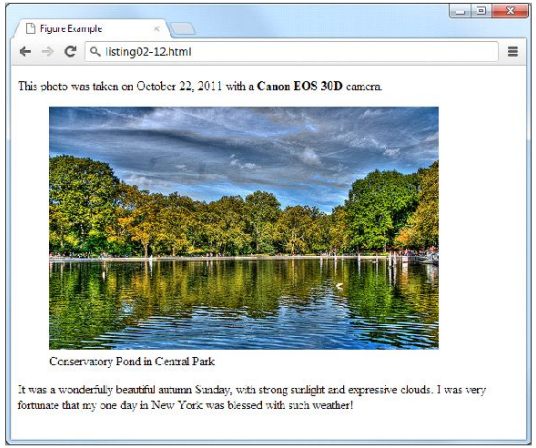
For instance, it makes no sense to wrap the site logo or non-essential images such as banner ads and graphical embellishments within <figure> elements.

Instead, only use the <figure> element for circumstances where the image (or other content) has a caption and where the figure is essential to the content but its position on the page is relatively unimportant.

Figure and Figure Captions

Figure could be moved to a different location in document
…
But it has to exist in the document (i.e., the figure isn't optional)

```
<p>This photo was taken on October  22, 2011 with a Canon EOS  30D camera.</p>
<figure>
  <img src="images/central-park.jpg" alt="Central Park " /><br/>
  <figcaption>Conservatory Pond in Central Park </figcaption>
</figure>
<p>
It was a wonderfully beautiful autumn Sunday  , with strong sunlight and
expressive clouds . I was very fortunate that my one day in New York was
blessed with such weather !
</p>
```



9 Aside

<aside>

The **<aside>** element is similar to the **<figure>** element in that it is used for marking up content that is separate from the main content on the page.

But while the **<figure>** element was used to indicate important information whose location on the page is somewhat unimportant, the **<aside>** element “represents a section of a page that consists of content that is tangentially related to the content around the aside element.”

The **<aside>** element could thus be used for sidebars, pull quotes, groups of advertising images, or any other grouping of non-essential elements.

INTRODUCING TABLES

Daniel Awde - Park Innovation

HTML Tables

A grid of cells

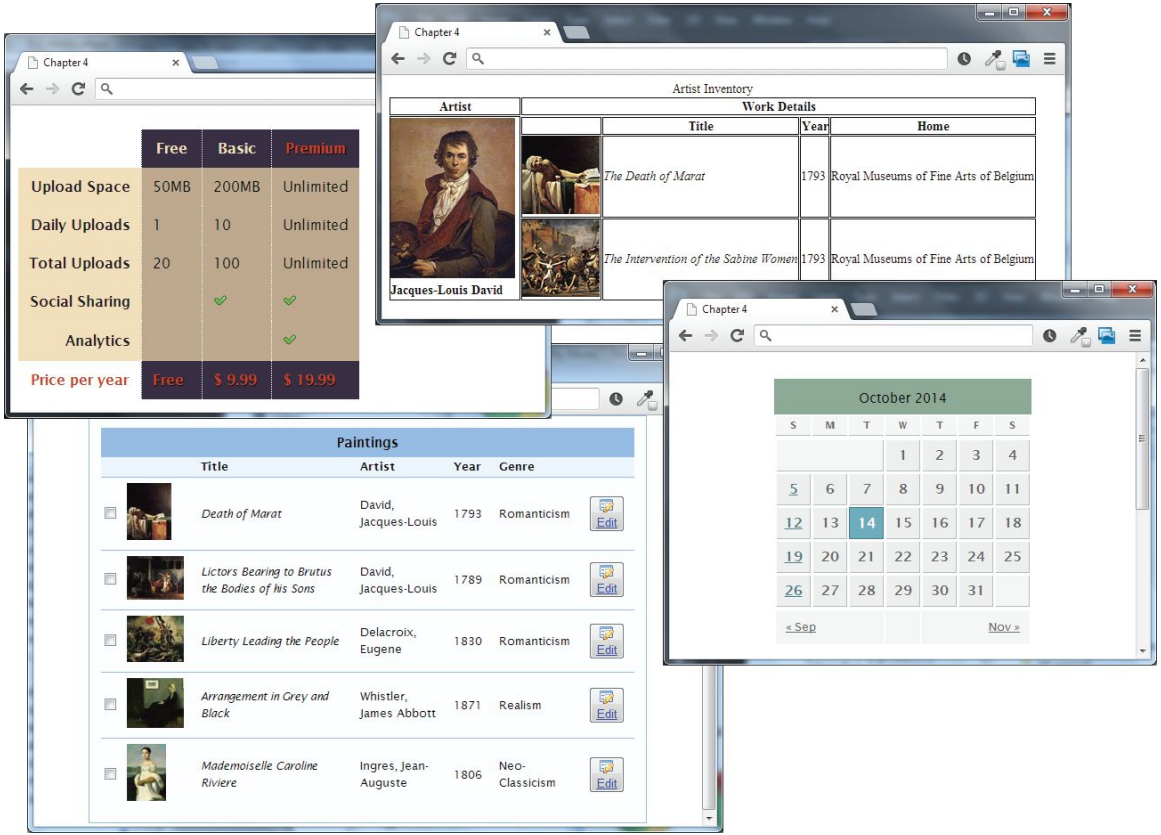
A table in HTML is created using the **<table>** element

Tables can be used to display:

- Many types of content
  - Calendars, financial data, lists, etc...
- Any type of data
  - Images
  - Text
  - Links
  - Other tables

HTML Tables

Example usages



Daniel Awde - Park Innovation

Tables Basics

Rows and cells

- an HTML **<table>** contains any number of rows (**<tr>**)
- each row contains any number of table data cells (**<td>**)
- Content goes inside of **<td></td>** tags

```
<table>
        <tr>
                <td>The Death of Marat</td>
        </tr>
</table>
```

content

A basic Example

<table>
<tr><td>The Death of Marat</td><td>Jacques-Louis David</td><td>1793</td><td>162cm</td><td>128cm</td></tr>
<tr><td>Burial at Ornans</td><td>Gustave Courbet</td><td>1849</td><td>314cm</td><td>663cm</td></tr>
</table>

```
<table>
    <tr>
        <td>The Death of Marat</td>
        <td>Jacques-Louis David</td>
        <td>1793</td>
        <td>162cm</td>
        <td>128cm</td>
    </tr>
    <tr>
        <td>Burial at Ornans</td>
        <td>Gustave Courbet</td>
        <td>1849</td>
        <td>314cm</td>
        <td>663cm</td>
    </tr>
</table>
```

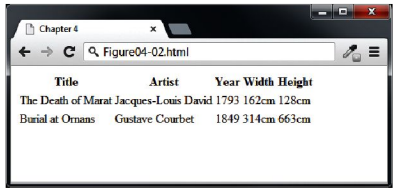
With Table Headings

<table>

<tr>	Title <th>	Artist <th>	Year <th>	Width <th>	Height <th>
<tr>	The Death of Marat <td>	Jacques-Louis David <td>	1793 <td>	162cm <td>	128cm <td>
<tr>	Burial at Ornans <td>	Gustave Courbet <td>	1849 <td>	314cm <td>	663cm <td>

th

```
<table>
   <tr>
      <th>Title</th>
      <th>Artist</th>
      <th>Year</th>
      <th>Width</th>
      <th>Height</th>
   </tr>
   <tr>
      <td>The Death of Marat</td>
      <td>Jacques-Louis David</td>
      <td>1793</td>
      <td>162cm</td>
      <td>128cm</td>
   </tr>
   <tr>
      <td>Burial at Ornans</td>
      <td>Gustave Courbet</td>
      <td>1849</td>
      <td>314cm</td>
      <td>663cm</td>
   </tr>
</table>
```



Why Table Headings

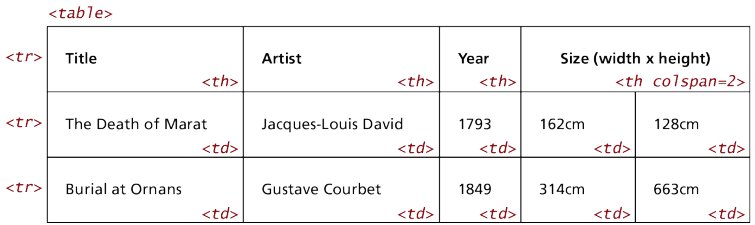
A table heading <th>

- Browsers tend to make the content within a <th> element bold
- <th> element for accessibility (it helps those using screen readers)
- Provides some semantic info about the row being a row of headers

Spanning Rows and Columns

Span Span Span a Row

Each row must have the same number of <td> or <th> containers. If you want a given cell to cover several columns or rows,




```
<table>
  <tr>
    <th>Title</th>
    <th>Artist</th>
    <th>Year</th>
    <th colspan="2">Size (width x height)</th>
  </tr>
  <tr>
    <td>The Death of Marat</td>
    <td>Jacques-Louis David</td>
    <td>1793</td>
    <td>162cm</td>
    <td>128cm</td>
  </tr>
  ...
</table>
```

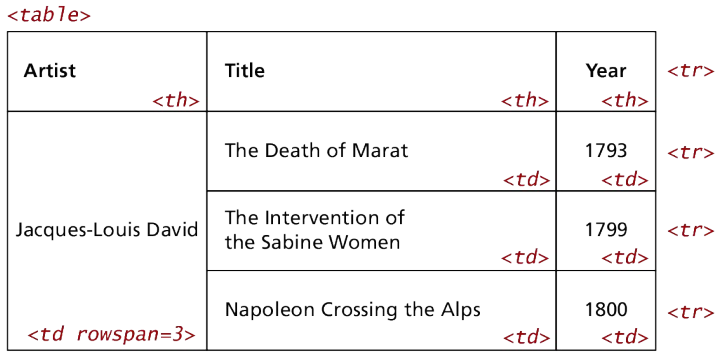
Notice that this row now only has four cell elements.

use the **colspan or rowspan attributes**

Using Tables for Layout

It works in many situations

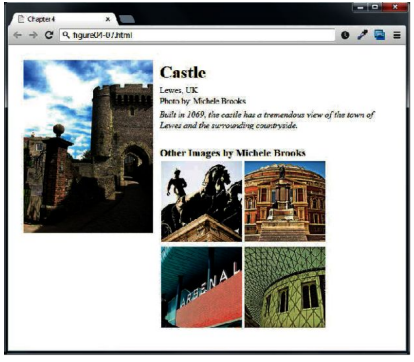
- Popular in 1990s
- Results in table bloat
- Not semantic
- Larger HTML pages
- Browser quirks

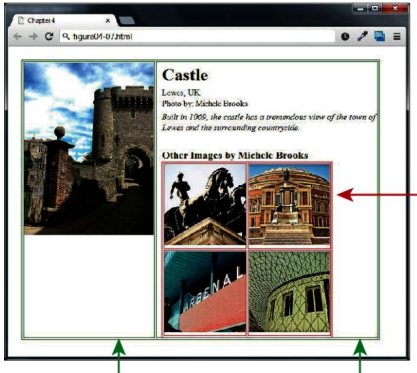



```
<table>
   <tr>
      <th>Artist</th>
      <th>Title</th>
      <th>Year</th>
   </tr>
   <tr>
      <td rowspan="3">Jacques-Louis David</td>
      <td>The Death of Marat</td>
      <td>1793</td>
   </tr>
   <tr>
      <td>The Intervention of the Sabine Women</td>
      <td>1799</td>
   </tr>
   <tr>
      <td>Napoleon Crossing the Alps</td>
      <td>1800</td>
   </tr>
   ...
</table>
```

Notice that these two rows now only have two cell elements.

Example Table layouts






```
<table>
  <tr>
    <td>
      <img src="images/959.jpg" alt="Castle"/>
    </td>
    <td>
      <h2>Castle</h2>
      <p>Lewes, UK</p>
      <p>Photo by: Michele Brooks</p>
      <p>Built in 1069, the castle has a tremendous
         view of the town of Lewes and the
         surrounding countryside.
      </p>

      <h3>Other Images by Michele Brooks</h3>

      <table>
        <tr>
          <td><img src="images/464.jpg" /></td>
          <td><img src="images/537.jpg" /></td>
        </tr>
        <tr>
          <td><img src="images/700.jpg" /></td>
          <td><img src="images/828.jpg" /></td>
        </tr>
      </table>
    </td>
  </tr>
</table>
```

Additional table tags

- <caption>
- <col>,<colgroup>
- <thead>
- <tfoot>
- <tbody>

A title for the table is good for accessibility.

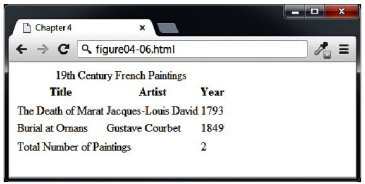
These describe our columns, and can be used to aid in styling.

Table header could potentially also include other <tr> elements.

Yes, the table footer comes *before* the body.

Potentially, with styling the browser can scroll this information, while keeping the header and footer fixed in place.

```
<table>
  <caption>19th Century French Paintings</caption>
  <col class="artistName" />
  <colgroup id="paintingColumns">
      <col />
      <col />
  </colgroup>

  <thead>
     <tr>
        <th>Title</th>
        <th>Artist</th>
        <th>Year</th>
     </tr>
  </thead>

  <tfoot>
     <tr>
        <td colspan="2">Total Number of Paintings</td>
        <td>2</td>
     </tr>
  </tfoot>

  <tbody>
     <tr>
        <td>The Death of Marat</td>
        <td>Jacques-Louis David</td>
        <td>1793</td>
     </tr>
     <tr>
        <td>Burial at Ornans</td>
        <td>Gustave Courbet</td>
        <td>1849</td>
     </tr>
  </tbody>

</table>
```



STYLING TABLES

Daniel Awde - Park Innovation

Styling Tables

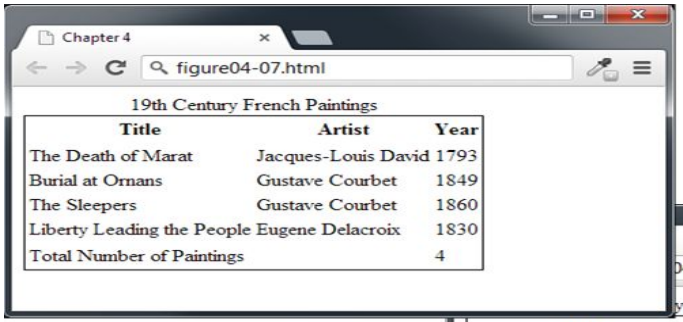
The old way's deprecated

In HTML5 it is left to CSS, However legacy support for deprecated HTML attributes still exist

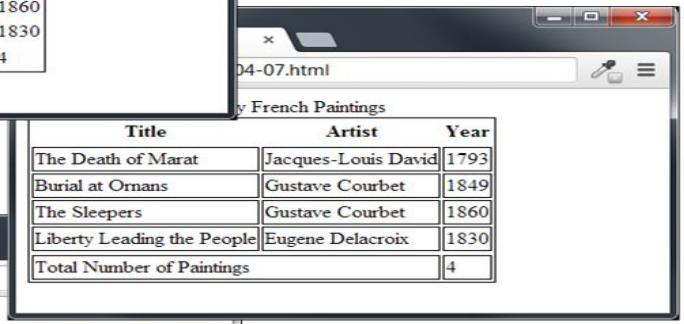
- **width, height**—for setting the width and height of cells
- **cellspacing**—for adding space between every cell in the table
- **cellpadding**—for adding space between the content of the cell and its border
- **bgcolor**—for changing the background color of any table element
- **background**—for adding a background image to any table element
- **align**—for indicating the alignment of a table in relation to the surrounding container

Styling Tables

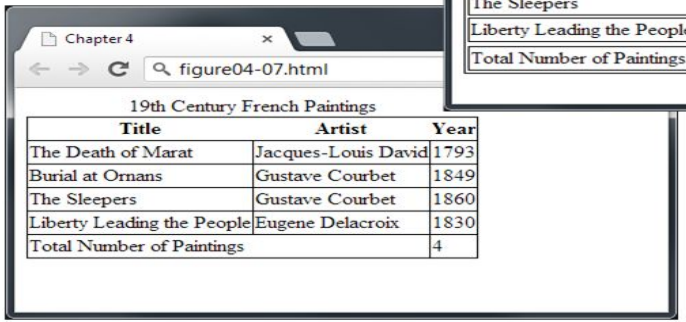
Borders




```
table {
    border: solid 1pt black;
}
```



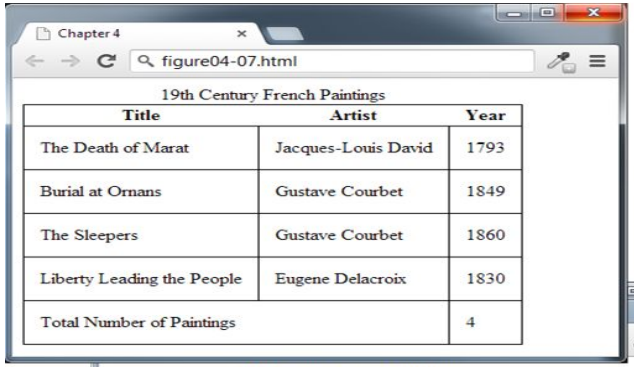

```
table {
    border: solid 1pt black;
}
td {
    border: solid 1pt black;
}
```



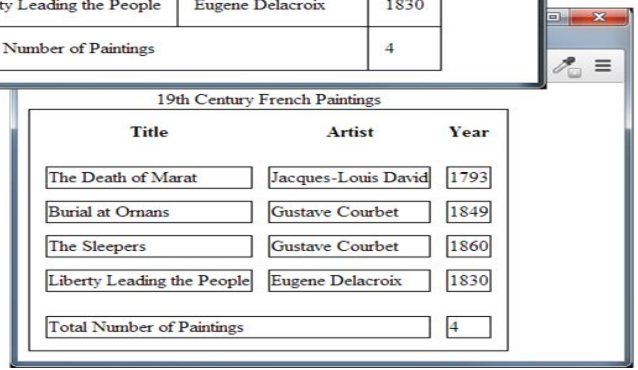

```
table {
    border: solid 1pt black;
    border-collapse: collapse;
}
td {
    border: solid 1pt black;
}
```

Styling Tables

Padding and spacing



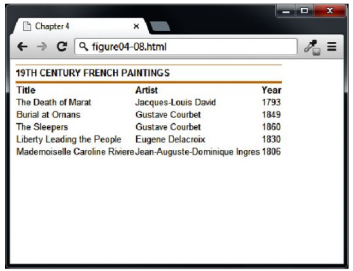

```
table {
   border: solid 1pt black;
   border-collapse: collapse;
}
td {
   border: solid 1pt black;
   padding: 10pt;
}
```



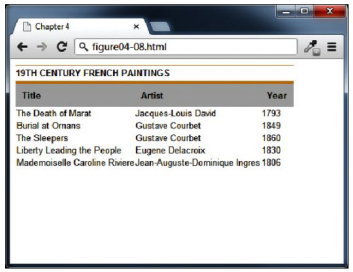

```
table {
   border: solid 1pt black;
   border-spacing: 10pt;
}
td {
   border: solid 1pt black;
}
```

Styling Tables

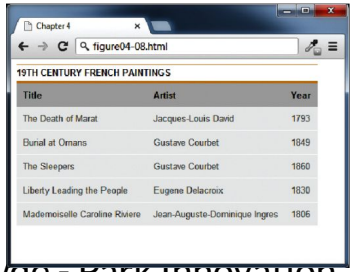
Examples




```
table {
   font-size: 0.8em;
   font-family: Arial, Helvetica, sans-serif;
   border-collapse: collapse;
   border-top: 4px solid #DCA806;
   border-bottom: 1px solid white;
   text-align: left;
}
caption {
   font-weight: bold;
   padding: 0.25em 0 0.25em 0;
   text-align: left;
   text-transform: uppercase;
   border-top: 1px solid #DCA806;
}
```



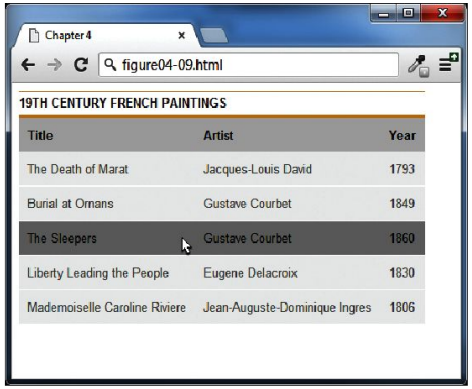

```
thead tr {
   background-color: #CACACA;
}
th {
   padding: 0.75em;
}
```



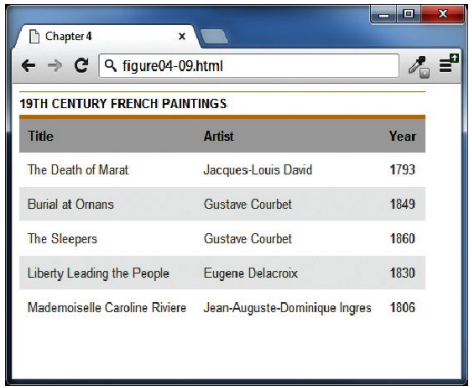

```
tbody tr {
   background-color: #F1F1F1;
   border-bottom: 1px solid white;
   color: #6E6E6E;
}
tbody td {
   padding: 0.75em;
}
```

Nth-Child

Nifty Table styling tricks: hover effect and zebra-stripes




```
tbody tr:hover {
   background-color: #9e9e9e;
   color: black;
}
```




```
tbody tr:nth-child(odd) {
   background-color: white;
}
```

INTRODUCING FORMS

Daniel Awde - Park Innovation

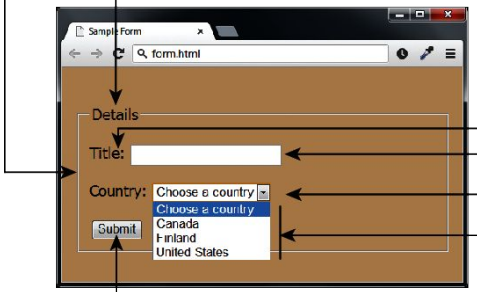
HTML Forms

Richer way to interact with server

Forms provide the user with an alternative way to interact with a web server.

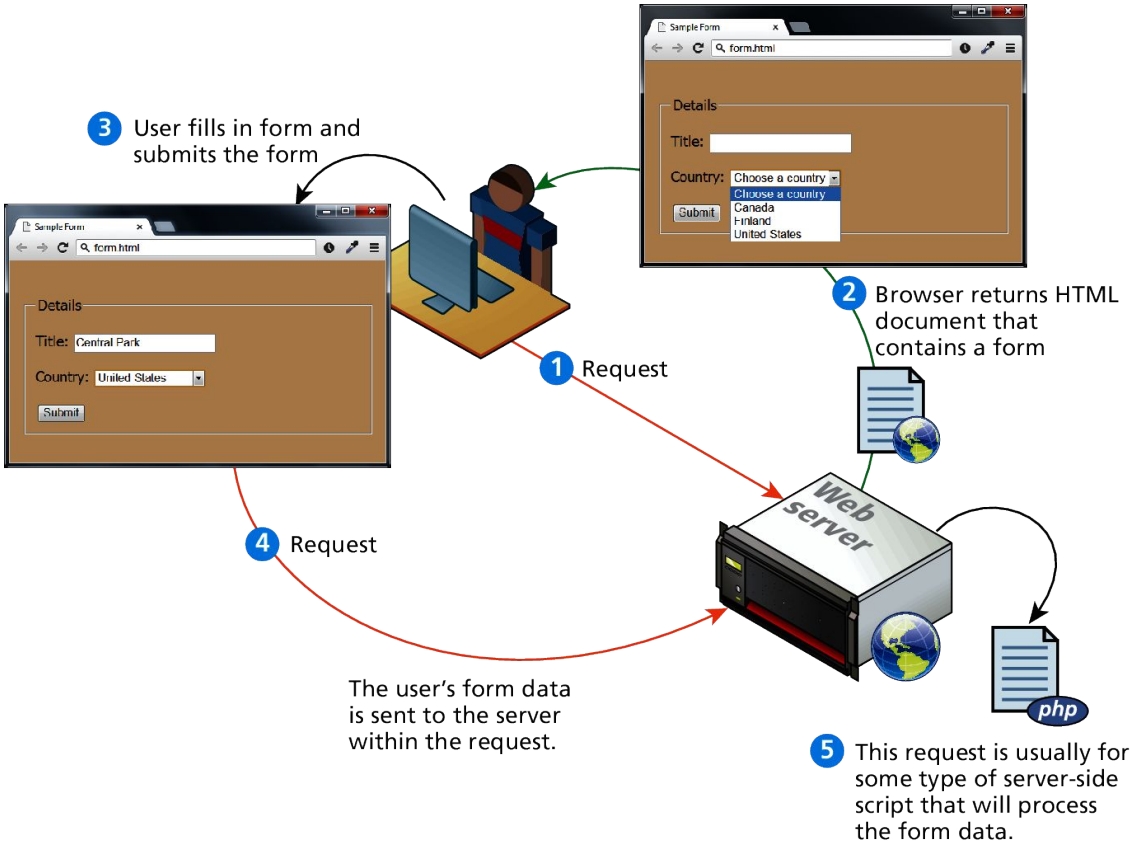
- Forms provide rich mechanisms like:
  - Text input
  - Password input
  - Options Lists
  - Radio and check boxes

Form Structure




```
<form method="get" action="process.php">
 <fieldset>
  <legend>Details</legend>
  <p>
   <label>Title: </label>
   <input type="text" name="title" />
  </p>
  <p>
   <label>Country: </label>
   <select name="where">
    <option>Choose a country</option>
    <option>Canada</option>
    <option>Finland</option>
    <option>United States</option>
   </select>
  </p>
  <input type="submit" />
 </fieldset>
</form>
```

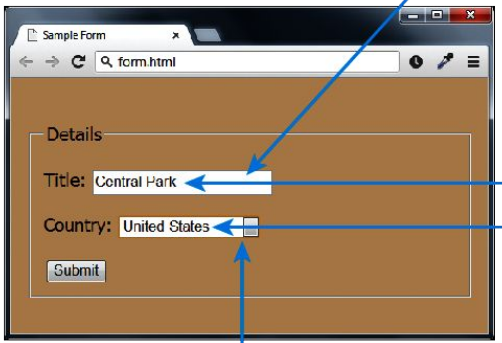
How forms interact with servers



Query Strings

At the end of the day, another string

```
<input type="text" name="title" />
```

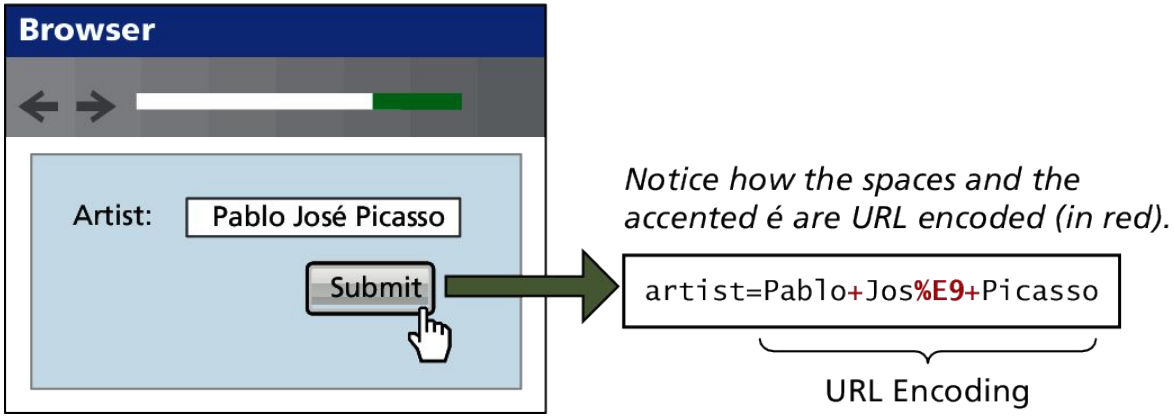



```
title=Central+Park&where=United+States
```

```
<select name="where">
```

URL encoding

Special symbols

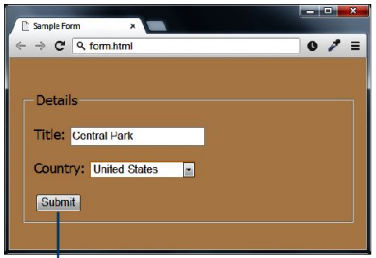


<form> element

Two essential features of any form, namely the **action** and the **method** attributes.

- The **action** attribute specifies the URL of the server-side resource that will process the form data
- The **method** attribute specifies how the query string data will be transmitted from the browser to the server.
  - GET
  - POST

GET vs POST




```
<form method="get" action="process.php">
```

```
GET /process.php?title=Central+Park&where=United+States http/1.1
```

querystring

```
<form method="post" action="process.php">
```

```
POST /process.php http/1.1
Date: Sun, 20 May 2012 23:59:59 GMT
Host: www.mysite.com
User-Agent: Mozilla/4.0
Content-Length: 47

title=Central+Park&where=United+States
```

HTTP Header

querystring

GET vs POST

Advantages and Disadvantages

- Data can be clearly seen in the address bar.
- Data remains in browser history and cache.
- Data can be bookmarked
- Limit on the number of characters in the form data returned.

POST

- Data can contain binary data.
- Data is hidden from user.
- Submitted data is not stored in cache, history, or bookmarks.

FORMS CONTROL ELEMENTS

Daniel Awde - Park Innovation

Form-Related HTML Elements

Type	Description
<button>	Defines a clickable button.
<datalist>	An HTML5 element form defines lists to be used with other form elements.
<fieldset>	Groups related elements in a form together.
<form>	Defines the form container.
<input>	Defines an input field. HTML5 defines over 20 different types of input.
<label>	Defines a label for a form input element.
<legend>	Defines the label for a fieldset group.
<option>	Defines an option in a multi-item list.
<optgroup>	Defines a group of related options in a multi-item list.
<select>	Defines a multi-item list.
<textarea>	Defines a multiline text entry box.

Text Input Controls

Type	Description
text	Creates a single line text entry box. **<input type="text" name="title" />**
textarea	Creates a multiline text entry box. **<textarea rows="3" ... />**
password	Creates a single line text entry box for a password **<input type="password" ... />**
search	Creates a single-line text entry box suitable for a search string. This is an HTML5 element. **<input type="search" ... />**
email	Creates a single-line text entry box suitable for entering an email address. This is an HTML5 element. **<input type="email" ... />**
tel	Creates a single-line text entry box suitable for entering a telephone. This is an HTML5 element. **<input type="tel" ... />**
url	Creates a single-line text entry box suitable for entering a URL. This is an HTML5 element. **<input type="url" ... />**

Text Input Controls

Classic

```
<input type="text" ... />
```

Text:

```
<textarea>
  enter some text
</textarea>
```

```
<textarea placeholder="enter some text">
</textarea>
```

TextArea: enter some text

TextArea: Enter some text

```
<input type="password" ... />
```

Password:

Password: ••••

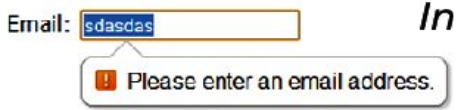
Text Input Controls

HTML5

```
<input type="search" placeholder="enter search text" ... />
```

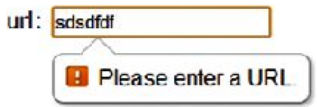

```
<input type="email" ... />
```

In Opera



In Chrome

```
<input type="url" ... />
```




```
<input type="tel" ... />
```

Daniel 

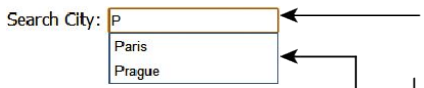
tion

HTML5 advanced controls

Pattern attribute

```
<input type="text" ... placeholder="L#L #L#" pattern="[a-z][0-9][a-z] [0-9][a-z][0-9]"  />
```

datalist




```
<input type="text" name="city" list="cities"  />

<datalist id="cities">
   <option>Calcutta</option>
   <option>Calgary</option>
   <option>London</option>
   <option>Los Angeles</option>
   <option>Paris</option>
   <option>Prague</option>
</datalist>
```

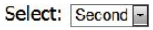
Select Lists

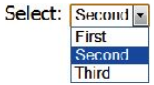
Chose an option, any option.

- **<select>** element is used to create a multiline box for selecting one or more items
  - The options are defined using the **<option>** element
  - can be hidden in a dropdown or multiple rows of the list can be visible
  - Option items can be grouped together via the **<optgroup>** element.

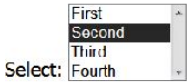
Select Lists

Select List Examples






```
<select name="choices">
  <option>First</option>
  <option selected>Second</option>
  <option>Third</option>
</select>
```




```
<select size="3" ... >
```

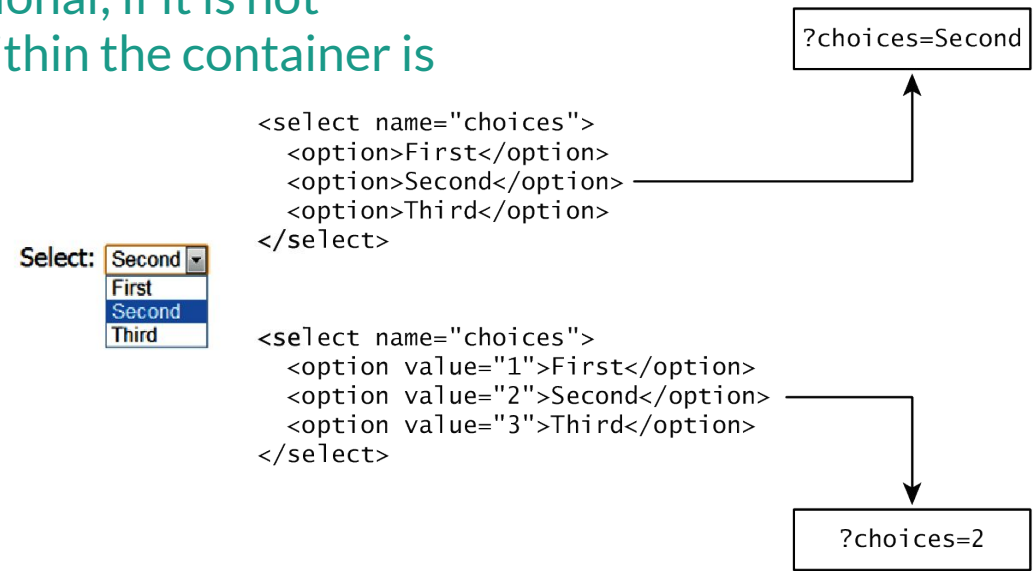

```
<select … >
 <optgroup label="North America">
   <option>Calgary</option>
   <option>Los Angeles</option>
 </optgroup>
 <optgroup label="Europe">
   <option>London</option>
   <option>Paris</option>
   <option>Prague</option>
 </optgroup>
</select>
```

Which Value to send

Select Lists Cont.

The **value** attribute of the <option> element is used to specify what value will be sent back to the server.

The value attribute is optional; if it is not specified, then the text within the container is sent instead



Radio Buttons

Radio buttons are useful when you want the user to select a single item from a small list of choices and you want all the choices to be visible

- radio buttons are added via the **<input type="radio">** element
- The buttons are mutually exclusive (i.e., only one can be chosen) by sharing the same name attribute
- The checked attribute is used to indicate the default choice
- the value attribute works in the same manner as with the <option> element

Radio Buttons


```
<input type="radio" name="where" value="1">North America<br/>
<input type="radio" name="where" value="2" checked>South America<br/>
<input type="radio" name="where" value="3">Asia
```

Checkboxes

Checkboxes are used for getting yes/no or on/off responses from the user.

- checkboxes are added via **the <input type="checkbox”>** Element
- You can also group checkboxes together by having them share the same name attribute
- Each checked checkbox will have its value sent to the server
- Like with radio buttons, the checked attribute can be used to set the default value of a checkbox

Checkboxes

I accept the software license ☑

```
<label>I accept the software license</label>
<input type="checkbox" name="accept" >
```

Where would you like to visit?
☑ Canada
☐ France
☑ Germany

```
<label>Where would you like to visit? </label><br/>
<input type="checkbox" name="visit" value="canada">Canada<br/>
<input type="checkbox" name="visit" value="france">France<br/>
<input type="checkbox" name="visit" value="germany">Germany
```

Button Controls

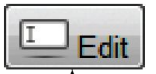
Type	Description
<input type="submit">	Creates a button that submits the form data to the server.
<input type="reset">	Creates a button that clears any of the user's already entered form data.
<input type="button">	Creates a custom button. This button may require Javascript for it to actually perform any action.
<input type="image">	Creates a custom submit button that uses an image for its display.
<button>	Creates a custom button. The <button> element differs from <input type="button"> in that you can completely customize what appears in the button; using it, you can, for instance, include both images and text, or skip server-side processing entirely by using hyperlinks. You can turn the button into a submit button by using the type="submit" attribute.

Button Controls

```
<input type="submit"  />
```


```
<input type="reset"  />
<input type="button" value="Click Me" />
```

```
<input type="image" src="appointment.png" />
```



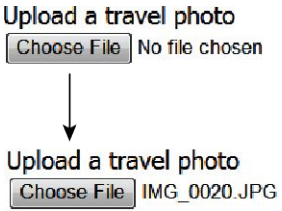

```
<button>
   <a href="email.html">
      <img src="images/email.png" alt=""/>
      Email
   </a>
</button>
```

```
<button type="submit"  >
   <img src="images/edit.png" alt=""/>
   Edit
</button>
```

Specialized Controls

I'm so special

- <input type=hidden>
- <input type=file>




```
<form method="post" enctype="multipart/form-data" ... >
   ...
   <label>Upload a travel photo</label>
   <input type="file" name="photo" />
   …
</form>
```

Number and Range

Typically input values need be **validated**. Although server side validation is required, optional client side pre-validation is good practice.

The number and range controls Added in HTML5 provide a way to input numeric values that **eliminates the need for JavaScript numeric validation!!!**

Number and Range

Rate this photo:
2

```
<label>Rate this photo: <br/>
<input type="number" min="1" max="5" name="rate" />
```

Grumpy Ecstatic

```
Grumpy
<input type="range" min="0" max="10" step="1" name="happiness" />
Ecstatic
```

Rate this photo:

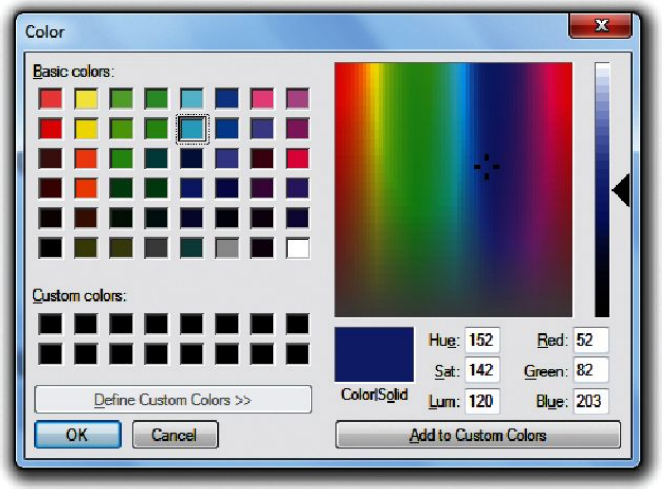
Grumpy Ecstatic

Controls as they appear in browser
that doesn't support these input types

Color

Background Color:

```
<label>Background Color: <br/>
<input type="color" name="back" />
```



Background Color:

Control as it appears in browser that doesn't support this input type

Date and Time Controls

Dates and times often need validation when gathering this information from a regular text input control.

From a user's perspective, entering dates can be tricky as well: you probably have wondered at some point in time when entering a date into a web form, what format to enter it in, whether the day comes before the month, whether the month should be entered as an abbreviation or a number, and so on.

HTML5 Date and Time Controls

Date:


```
<label>Date: <br/>
<input type="date" ... />
```

Time:

02:02 AM

```
<input type="time" ... />
```

DateTime:

2013-03-08 05:46 UTC

```
<input type="datetime" ... />
```

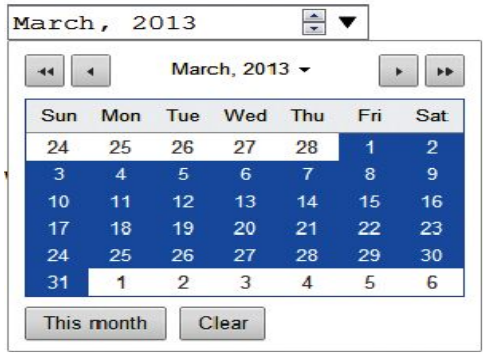
DateTime Local:

2013-03-13 12:02

```
<input type="datetime-local" ... />
```

HTML5 Date and Time Controls

Month:




```
<input type="month" ... />
```

Week:


```
<input type="week" ... />
```

HTML Controls

Type	Description
date	Creates a general date input control. The format for the date is "yyyy-mm-dd".
time	Creates a time input control. The format for the time is "HH:MM:SS", for hours:minutes:seconds.
datetime	Creates a control in which the user can enter a date and time.
datetime-local	Creates a control in which the user can enter a date and time without specifying a time zone.
month	Creates a control in which the user can enter a month in a year. The format is "yyyy-mm".
week	Creates a control in which the user can specify a week in a year. The format is "yyyy-W##".

Other Controls

You mean there's more

- The <progress> and <meter> elements can be used to provide feedback to users,
    - but requires JavaScript to function dynamically.
- The <output> element can be used to hold the output from a calculation.
- The <keygen> element can be used to hold a private key for public-key encryption

TABLE AND FORM ACCESSIBILITY

Daniel Awde - Park Innovation

Web Accessibility

Not all web users are able to view the content on web pages in the same manner.

The term **web accessibility** refers to the assistive technologies, various features of HTML that work with those technologies, and different coding and design practices that can make a site more usable for people with visual, mobility, auditory, and cognitive disabilities.

In order to improve the accessibility of websites, the W3C created the **Web Accessibility Initiative (WAI)**

- Web Content Accessibility Guidelines

Web Content Accessibility Guidelines

- Provide text alternatives for any nontext content so that it can be changed into other forms people need, such as large print, braille, speech, symbols, or simpler language.
- Create content that can be presented in different ways (for example simpler layout) without losing information or structure.
- Make all functionality available from a keyboard.
- Provide ways to help users navigate, find content, and determine where they are.

Accessible Tables

1. Describe the table's content using the <caption> element

2. Connect the cells with a textual description in the header

```
<table>
   <caption>Famous Paintings</caption>
   <tr>
      <th scope="col">Title</th>
      <th scope="col">Artist</th>
      <th scope="col">Year</th>
      <th scope="col">Width</th>
      <th scope="col">Height</th>
   </tr>
   <tr>
      <td>The Death of Marat</td>
      <td>Jacques-Louis David</td>
      <td>1793</td>
```

Accessible Forms

Recall the <fieldset>, <legend>, and <label> elements.

Each <label> element should be associated with a single input element.