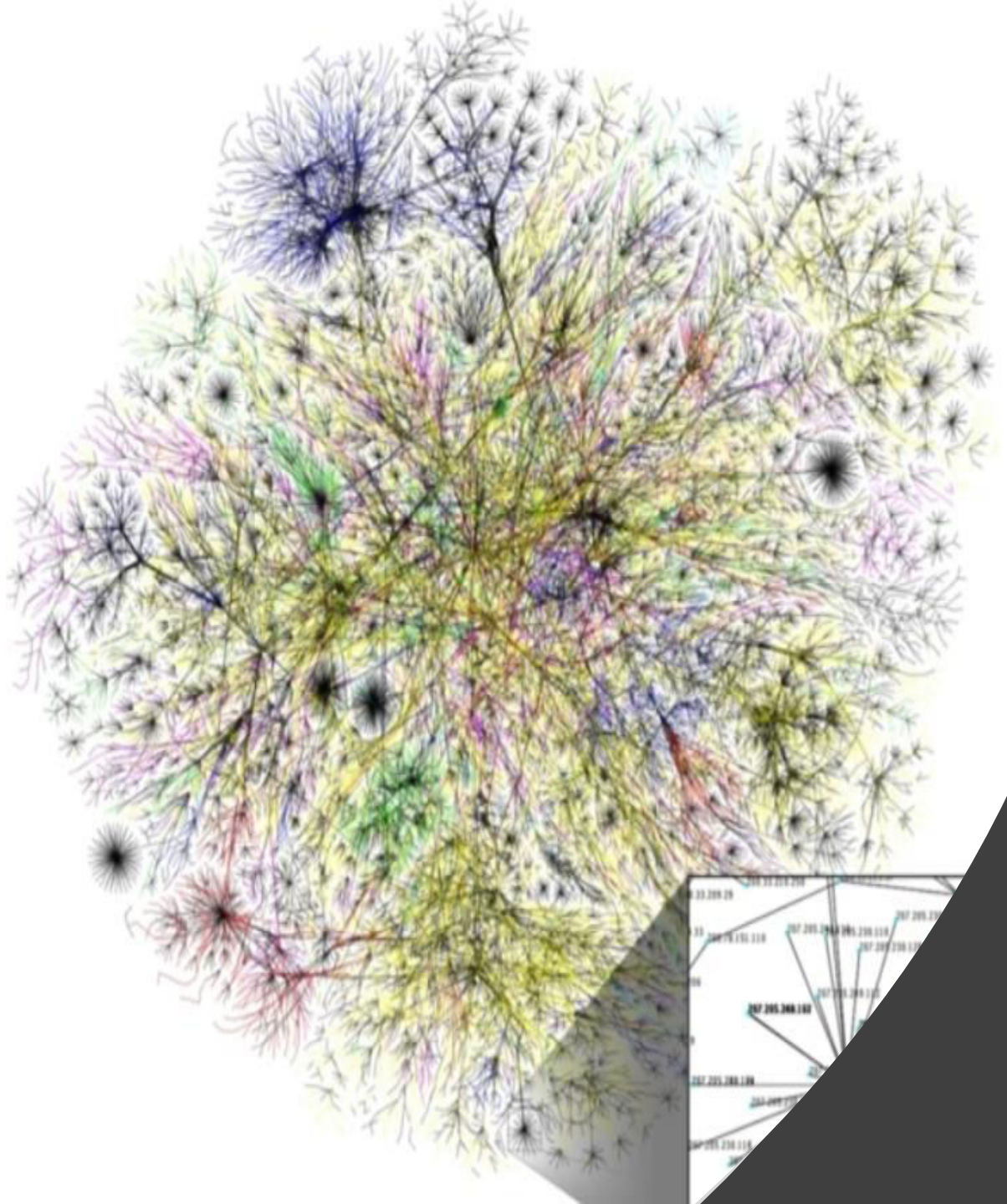


Web Design and Development I

Chapter 1

Basics of the Internet and the web

Outlines of the Chapter:

- The Internet
- World Wide Web (WWW)
- Application Layer Protocols
- Web browser
- Web Site Development Basics
- Web and internet Terminologies

Discussion Points

- What is Internet ?
- How internet Works ?
- What is World Wide Web (WWW)?
- What is a website ?
- What is URL ?
- What is a web Browser ?

1. What is the Internet ?

- The Internet is a global Connection of networks that enables computers of all kinds to directly and transparently communicate and share services throughout the world.
- It is a *network of networks* that consists of millions of

  private,

  public,

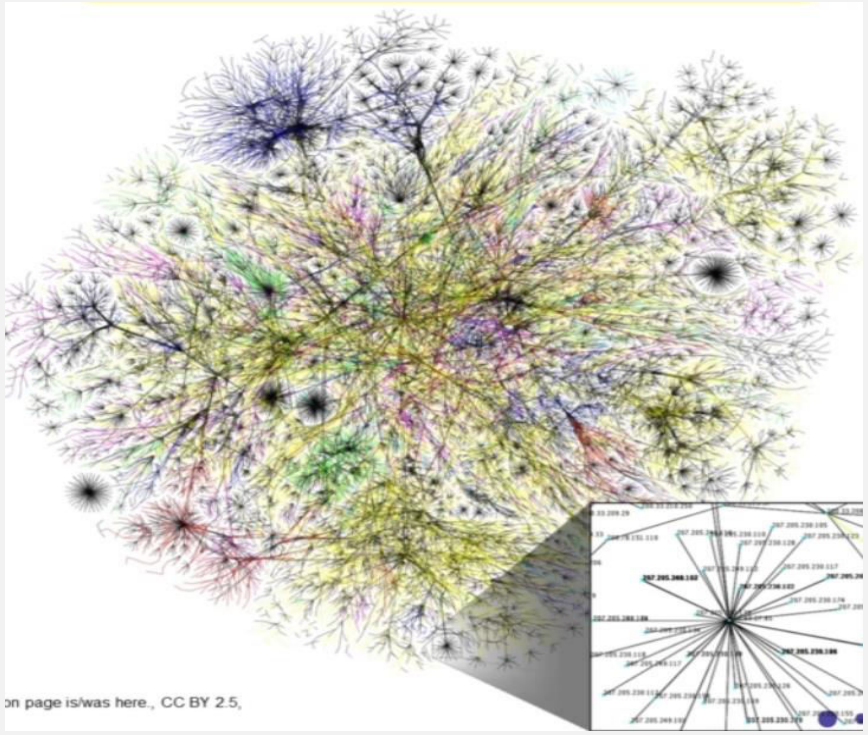
  academic,

  business,

  government networks

of local to global scope, linked by a broad array of *electronic*, *wireless,* and *optical* networking technologies.

- The Internet carries an extensive range of information resources and services.



(Continued…)

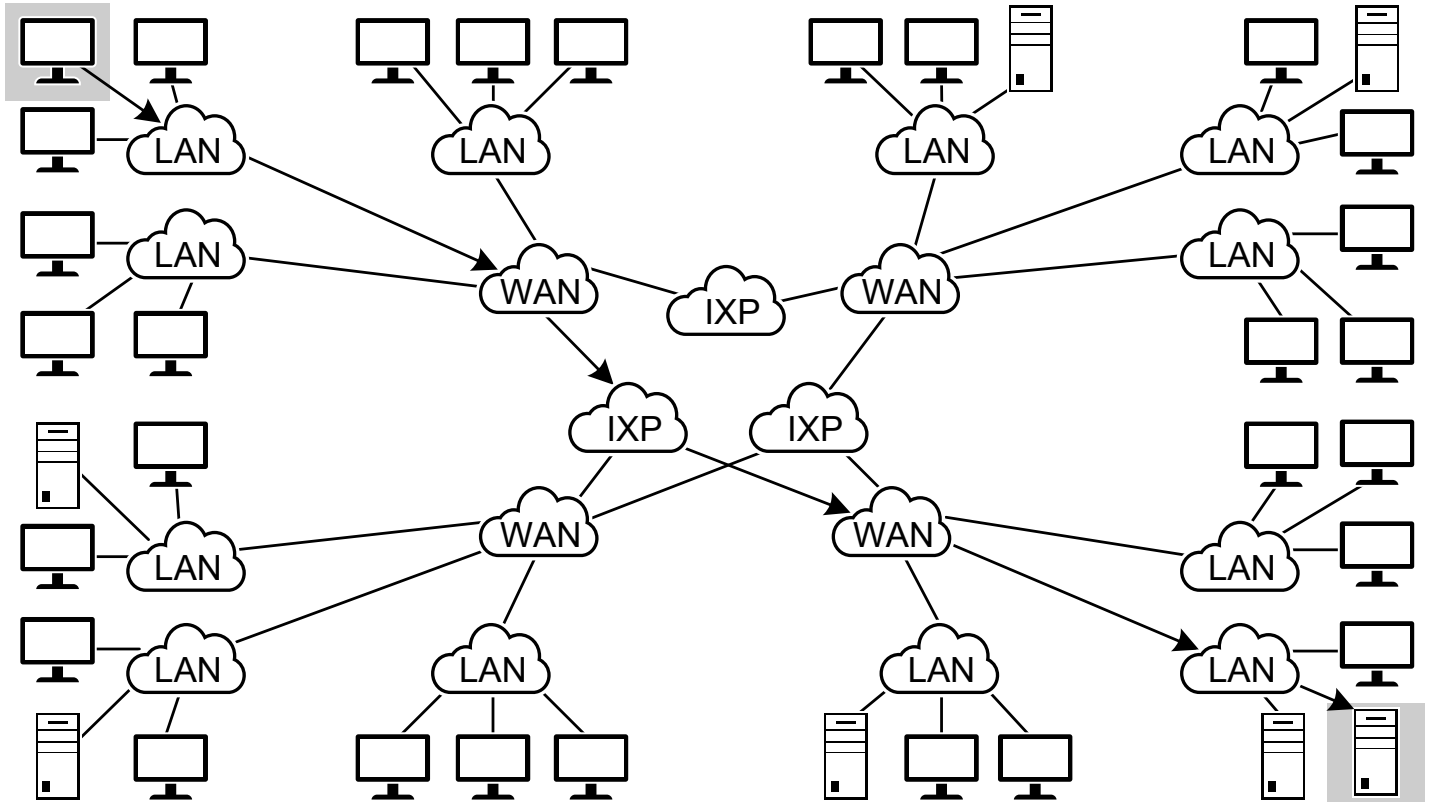
- In 1969 the precursor of Internet is born**: ARPAnet.**
  - ARPA = Advanced Research Projects Agency sponsored by the American Department of

  Defense (DOD).
  - Designed to connect military research centers.
  - Distributed computer system able to survive a nuclear attack.
- **Problem:** ARPAnet could connect only networks of the same type.
- In 1970, ARPA starts developing the Transmission Control Protocol / Internet Protocol (TCP/IP), a technology for connecting networks of different types (produced by different companies).
- Other networks appear, such as CSNET and BITNET.
- The Internet = a network of networks.
- People around the world share ideas, information, comments, and stories.

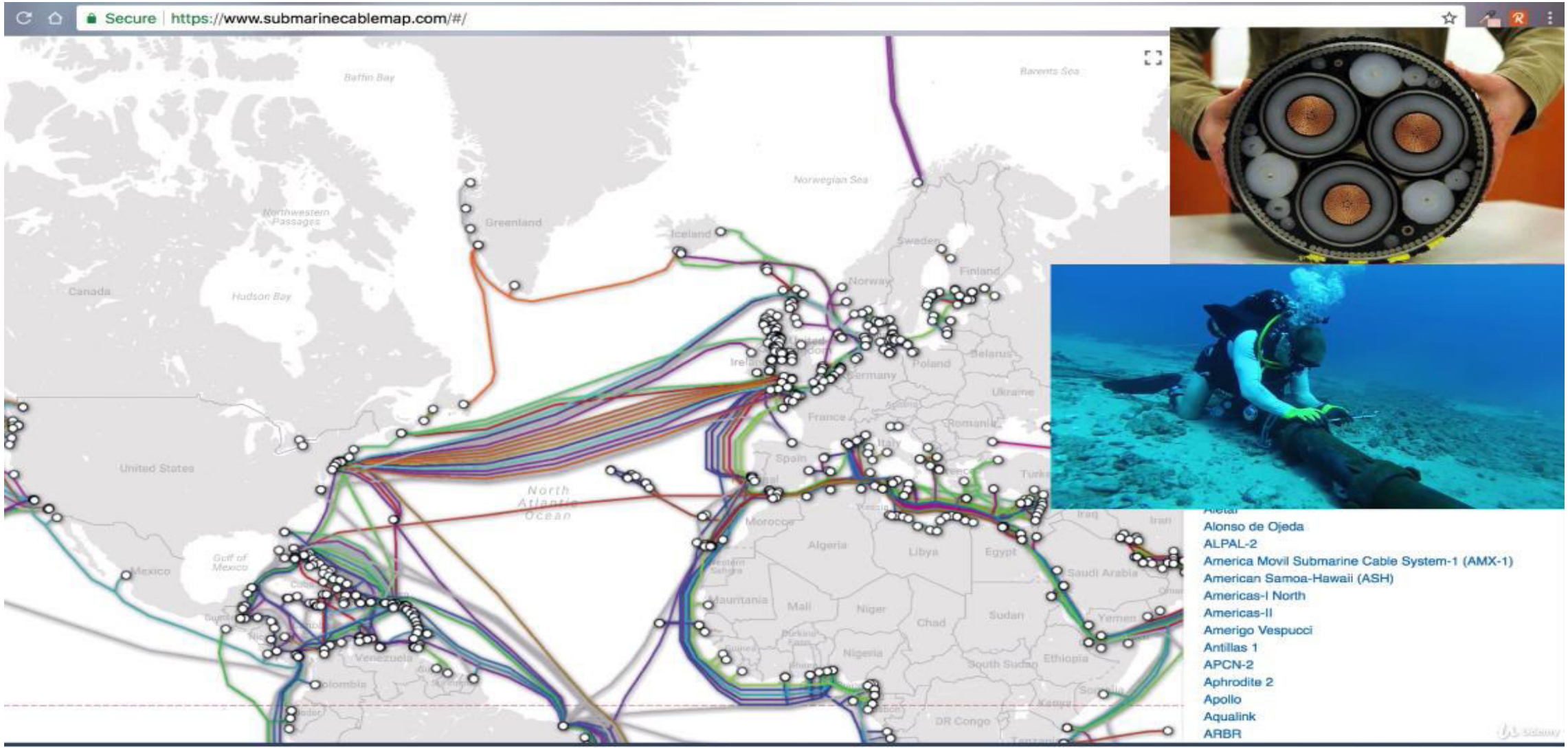
(Continued…)

- The Internet is a global system of interconnected computer networks and a short form for INTERconnected NETwork.
- Each network is independent from its neighbors, and so ultimately, no one country or organization "owns" the Internet.
- The Internet's design also mandates that all hosts (computers) connected through the network are equal, or peers. "no single point of failure" goal since there are no easily-targeted "master" servers required to keep traffic moving.

The architecture of the Internet



Internets global network with submarine backbone

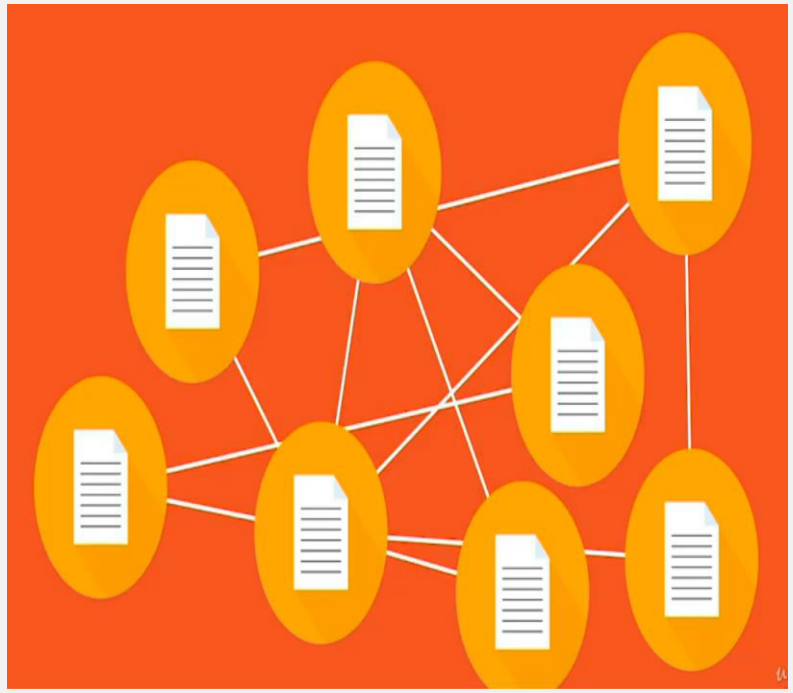


Popular services over the internet:

- Email (electronic mail) is the most popular service.
- Web browsing to find information.
- Newsgroups :
  - are collections of messages on various subject
- FTP (File Transfer Protocol).
  - You can copy files from one computer to another over the Internet.
- Telnet or remote login.
  - Permits your computer to log onto another computer and use it as if you were there.
- Chatrooms.
  - You can exchange messages with other
- Internet services for companies:
  - e-commerce, etc.

2. What is WWW?

- WWW stands for **World Wide Web**.
- A technical definition of the World Wide Web is – **All the resources and users on the Internet that are using the Hypertext Transfer Protocol (HTTP).**
- WWW (World Wide Web): It is a **collection of interlinked documents** that are accessible over the Internet.
- It consists of millions of web pages that contain text, images, voice and videos.



(Continued…)

- The World Wide Web allows computer users to locate and view multimedia-based documents (i.e., documents with text, graphics, animations, audios or videos) on almost any subject.
- Even though the Internet was developed more than three decades ago, the introduction of the World Wide Web is a relatively recent event.
- In 1990, Tim Berners-Lee of CERN (the European Laboratory for Particle Physics) developed the World Wide Web and several communication protocols that form the backbone of the Web.

(Continued…)

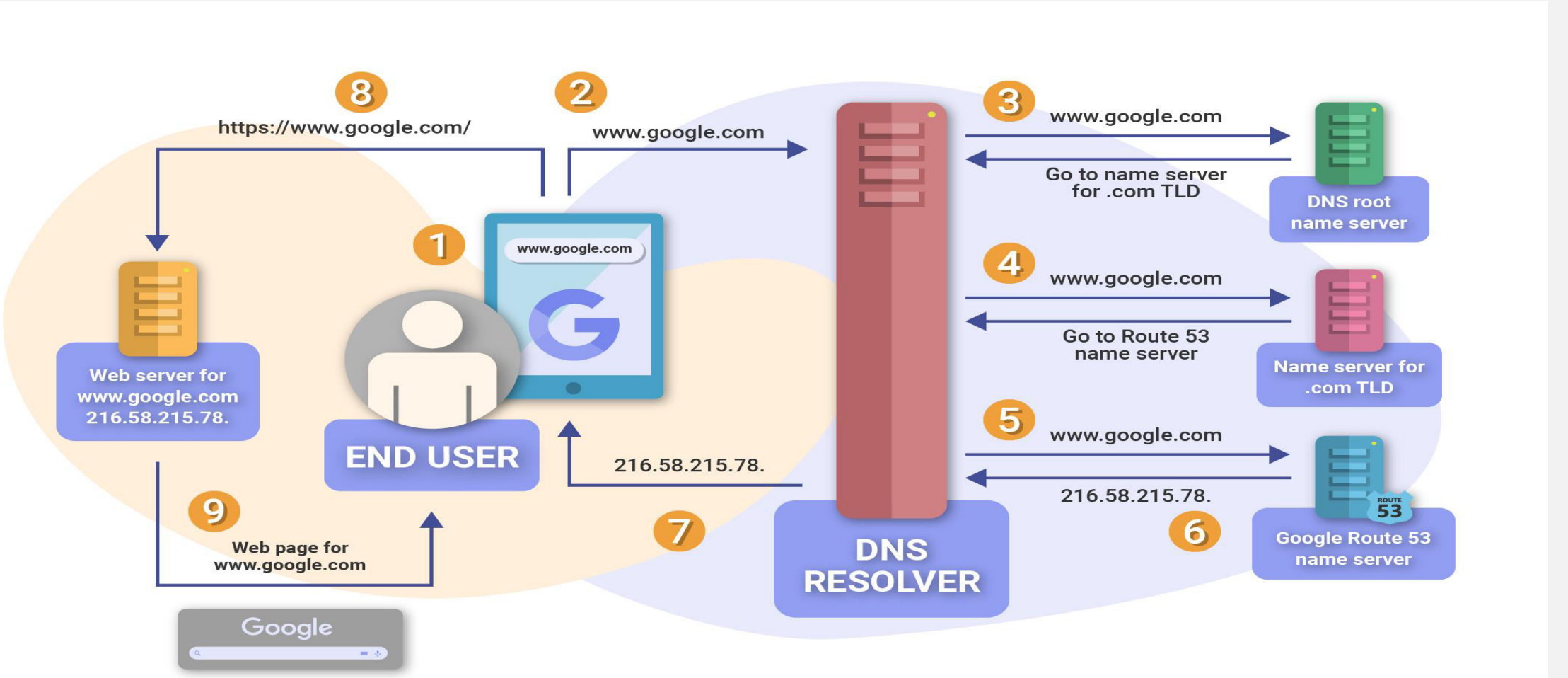
- The **backbone** of the World Wide Web is the Hypertext Markup Language, often simply referred to as **HTML.**
- The Web is known as a ***client-server system***. Your computer is the client and the remote computers that store electronic files are the servers.
- The Web is a ***distributed network***.

That means there is no central computer for the World Wide Web. Any server on the Web can be accessed directly by any client. If a server on the World Wide Web malfunctions, it doesn't affect the performance of other servers.

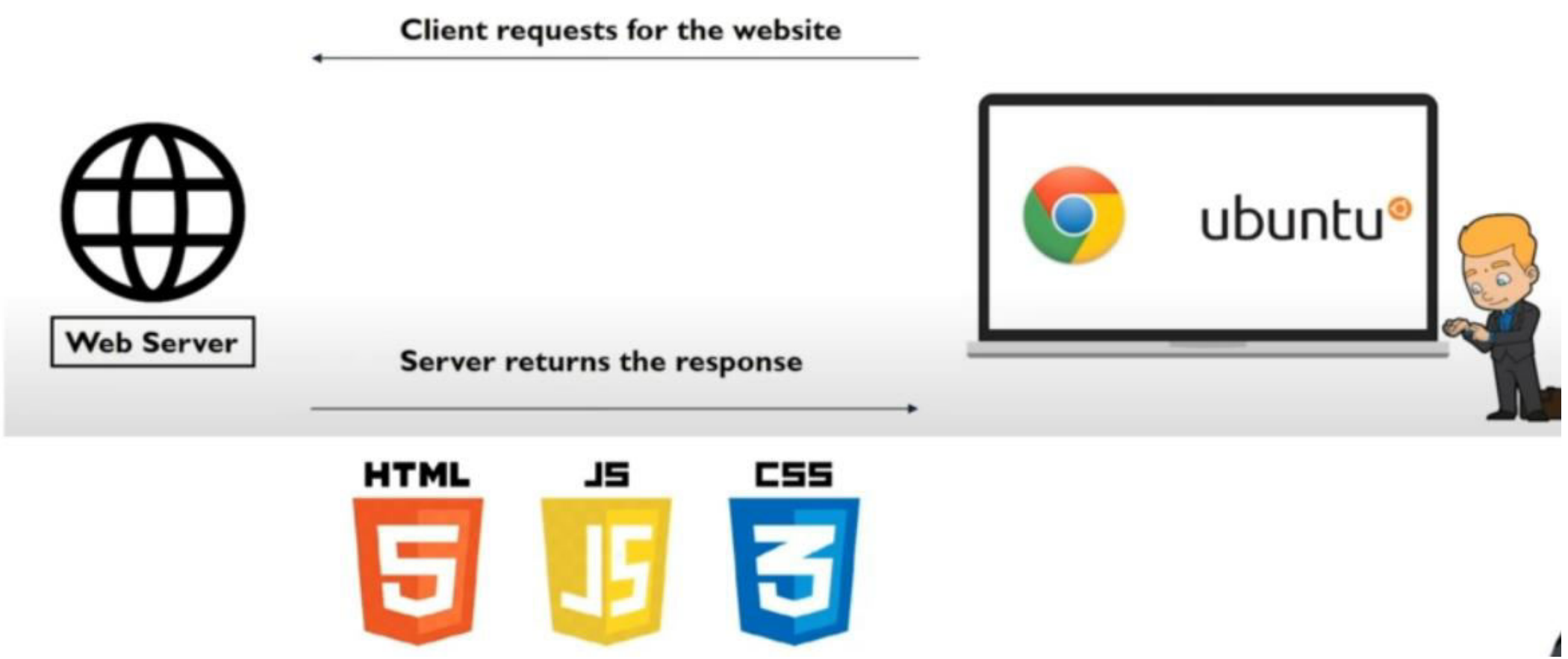
(Continued…)

- Most of the documents on the World Wide Web are written in **Hypertext Markup Language (HTML).**
- HTML provides instructions for the client software on how the document should be displayed.
- HTML also contains information about how to link up to other documents on the Web.

How internet Works ?



How a website works



3. Application Layer Protocols

The protocols in this layer deliver all of the Internet services we're familiar with:

- **HTTP** (HyperText Transfer Protocol, TCP port 80): transmits web pages across the Internet
- **FTP (**File Transfer Protocol, TCP ports 20 & 21): used for moving files from one machine to another
- **SMTP** (Simple Mail Transfer Protocol, TCP port 25): for sending e-mail messages
- **POP3** (Post Office Protocol, version 3, TCP port 110): for retrieving e-mail messages from a e-mail server
- **NNTP** (Network News Transfer Protocol, TCP port 119): network newsgroups
- **Telnet** (TCP port 23): allows users to log on to remote computers just as if they were sitting in front of the remote computer
- **DNS** (Domain Name Service, UDP port 53): translates the machine name part of the URL into IP addresses.

3.1 HTTP and HTTPs?

- **Hypertext Transfer Protocol (HTTP)** is a communications protocol. Its use for retrieving inter-linked text documents (hypertext) led to the establishment of the **World Wide Web.**
- HTTP is a request/response standard between a client and a server.
- A client is the end user, the server is the web site. The client making a HTTP request—using a web browser, or other end-user tool—is referred to as the ***user agent***. The responding server—which stores or creates *resources* such as HTML files and images—is called the ***origin server***

3.1.1 HTTP

- an HTTP client initiates a request.
- It establishes a Transmission Control Protocol (TCP) connection to a particular port on a host (port 80 by default).
- An HTTP server listening on that port waits for the client to send a request message. Upon receiving the request, the server sends back a status line, such as "HTTP/1.1 200 OK", and a message of its own, the body of which is perhaps the requested file, an error message, or some other information.
- Resources to be accessed by HTTP are identified using Uniform Resource Locators (URLs)) using the http: or https URI schemes.

3.1.2 HTTPS

- Hyper Text Transfer Protocol Secure (HTTPS) is the secure version of HTTP, the protocol over which data is sent between your browser and the website that you are connected to.
- The 'S' at the end of HTTPS stands for 'Secure'. It means all communications between your browser and the website are encrypted.
- HTTPS is often used to protect highly confidential online transactions like online *banking* and *online shopping* order forms.
- Web browsers such as Internet Explorer, Firefox and Chrome also display a padlock icon in the address bar to visually indicate that HTTPS connection is in effect.
- This ensures reasonable protection from eavesdroppers but is weak with *man-in-the-middl*e attacks.

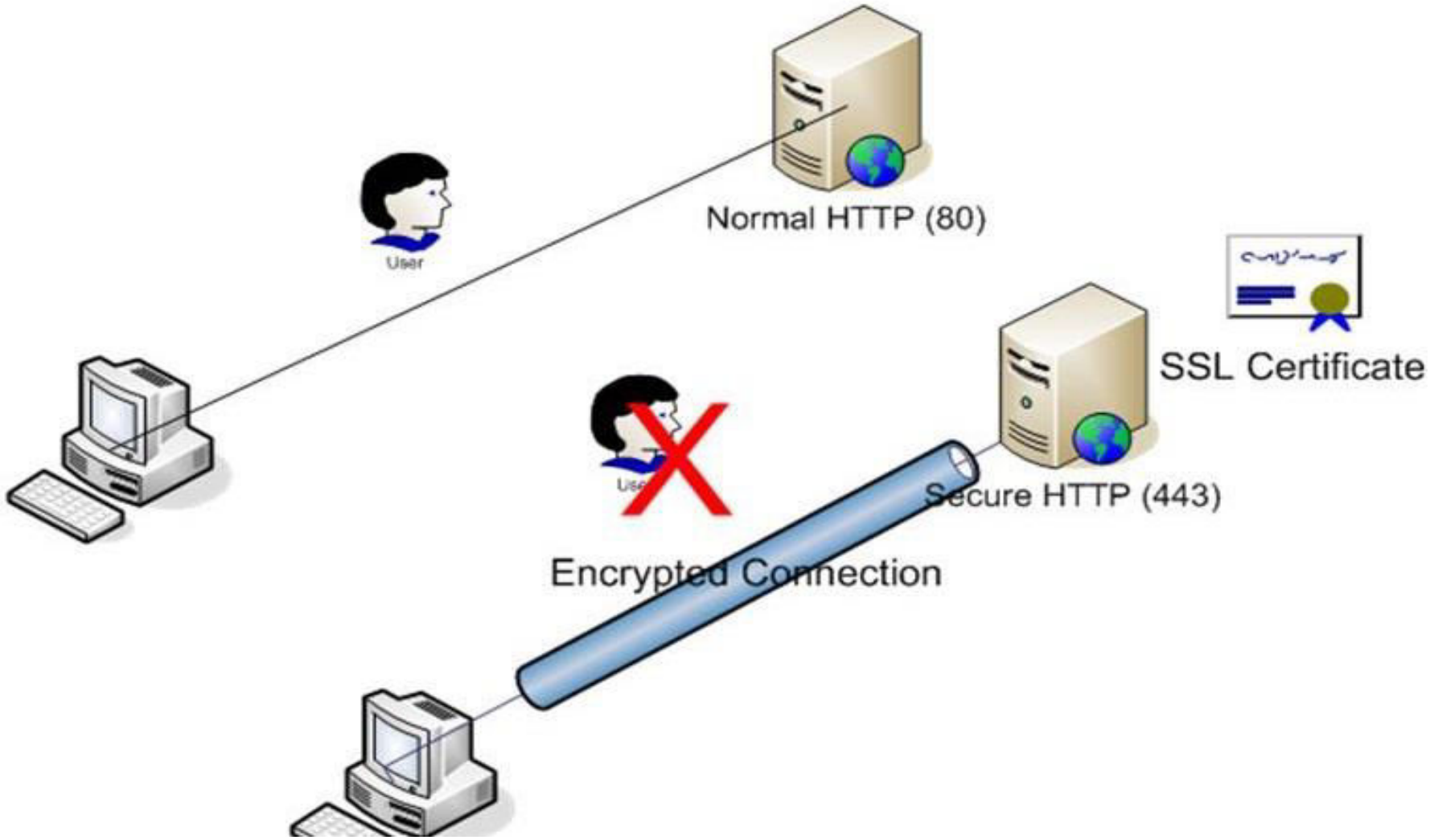
(Continued...)

- SSL certificates contain the computer owner's "public key."
- The owner shares the public key with anyone who needs it. Other users need the public key to encrypt messages to the owner.
- The owner sends those users the SSL certificate, which contains the public key.
- The owner does not share the private key with anyone.
- The security during the transfer is called the Secure Sockets Layer (SSL) and Transport Layer Security (TLS).
- The procedure for exchanging public keys using SSL Certificate to enable HTTPS, SSL and TLS is called Public Key Infrastructure (PKI).

HTTP VS HTTPS (Continued…)

	
→Exchanging confidential information for this reason, HTTPS Was developed.	→Its a System for transmitting and receiving information across the Internet.
→Transfer encrypted information	→If you don't want to Enter any sensitive information, Then HTTP:// is just fine.
→When you Enter password, credit card number, or other financial information, you should always look HTTPS://	→ Its use for any blogs, entertainment purpose.
→ HTTPS is HTTP-within-SSL/TLS.	→Its a Protocol and used to transfer data with Header from client.
→ Invented by Netscape corporation.	→ Invented by Sir Timothy John.
→Uses Port 443.	→ It will use Port 80.

HTTP vs HTTPS (Continued…)



4. What is DNS ?

- The domain name is basically the address of your website, a nickname for its IP address.
- Domains can be assigned to IPs by a Domain Name Registry headed by the Internet Assigned Numbers Authority (IANA).
- When we type in a URL, an application-layer service called the **Domain Name Service (DNS)** translates the human-friendly URL into the computer-friendly IP address for us.
- DNS performs this translation by consulting the databases maintained by the Domain Name Registrars. DNS is responsible for translating "www.apple.com" into "17.112.152.32" whenever we type that into a browser address bar. DNS is the "phone book" of the Internet: look up a name, and find its number.
- **TCP**/**IP**, or the Transmission Control Protocol/Internet Protocol, is a suite of communication protocols used to interconnect network devices on the internet.**TCP**/**IP** can also be used as a communications protocol in a private network (an intranet or an extranet).

5. Uniform Resource Locators (URLs)

- Humans being humans, prefer not to have to remember IP addresses like "17.112.152.32".
- Instead, we'd rather use a little piece of text like "www.CBE.com", which is called a **Uniform Resource Locator, or URL** (pronounced "You-Are-Elle", or sometimes "Earl").
- In fact, both 17.112.152.32 and www.CBE.com refer to the same Internet host: CBE Computer's webserver.
- URL's are used for more than just websites; they're also used for e-mail addresses, FTP servers, network newsgroups, or any other service running on a computer connected to the Internet.

(Continued…)

- A URL is made up of many parts. For example, consider the following:

http://www.FTI.edu/registrat/grader/index.html

- **http** is the application-layer protocol. In this case HTTP, the protocol used by the World Wide Web.
- **www.FTI.edu** is the **Fully Qualified Domain Name (FQDN).** In turn, the FQDN is made up of pieces as well (from right to left, or most general to most specific).

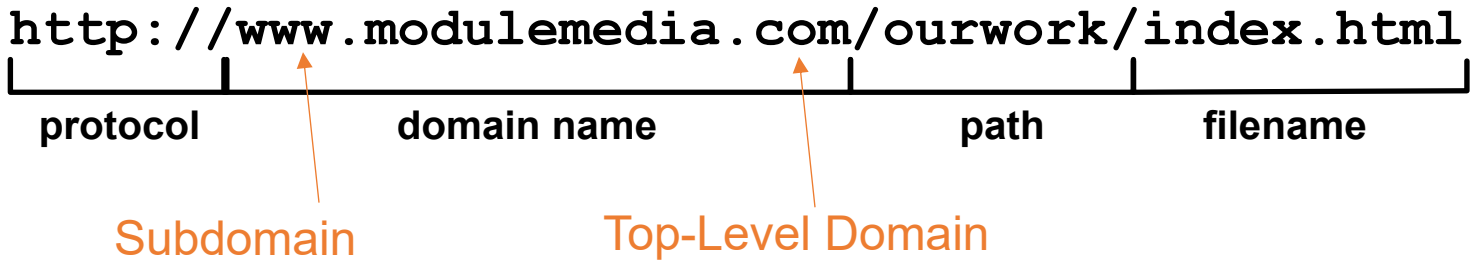
- **.com** is the **Top Level Domain (TLD**), in this case meaning “educational institute"
- **FTI** is the specific company/institute name, chosen by FTI Computer.
- **www** is the name of the webserver hosting FTI Computer's website.
- **/Reg/grade/index.html** is the path to the requested HTML page on the webserver. This path and file is used by the webserver to figure out which file to return to a browser. Other application-layer protocols might not need a path or file name.

The components of an HTTP URL

- Protocols determine network language
  - Usually http or https for web browsers

- Other Common Protocols:
  - `https://` (secure)
  - `ftp://` (file transfer)
  - `mailto://` (e-mail address)
  - `file://` (local files)

The components of an HTTP URL



- Subdomains
  - www (web server, sometimes optional)
  - ftp (file server)
  - en (“English” on Wikipedia)
  - www.cs (“Computer Science”, “web” at Pitt)

- Top-Level Domains (TLD)
  - .com, .org, .net, .gov, .mil
  - .us, .ca, .uk, .jp
  - newer ones: .biz, .info, …

(Continued…)

Top **Level Domains**

There are only a few top level domains, and these are controlled by one of the Internet self-regulating authorities(ICANN [www.icann.org]).

The most common TLDs are as follows:

.com: companies or commercial ventures

.net: ISPs or other companies that work with the Internet

.org: non-profit organizations and institutions

.mil: military

.edu: education institutions

.gov: government

Recently, ICANN specified some new TLDs that haven't really caught on much yet:

.info, .biz, .pro, .museum, etc.

TLDs are also assigned to countries:

- **.et**: Ethiopia
- **.us:** United States
- .**uk:** United Kingdom

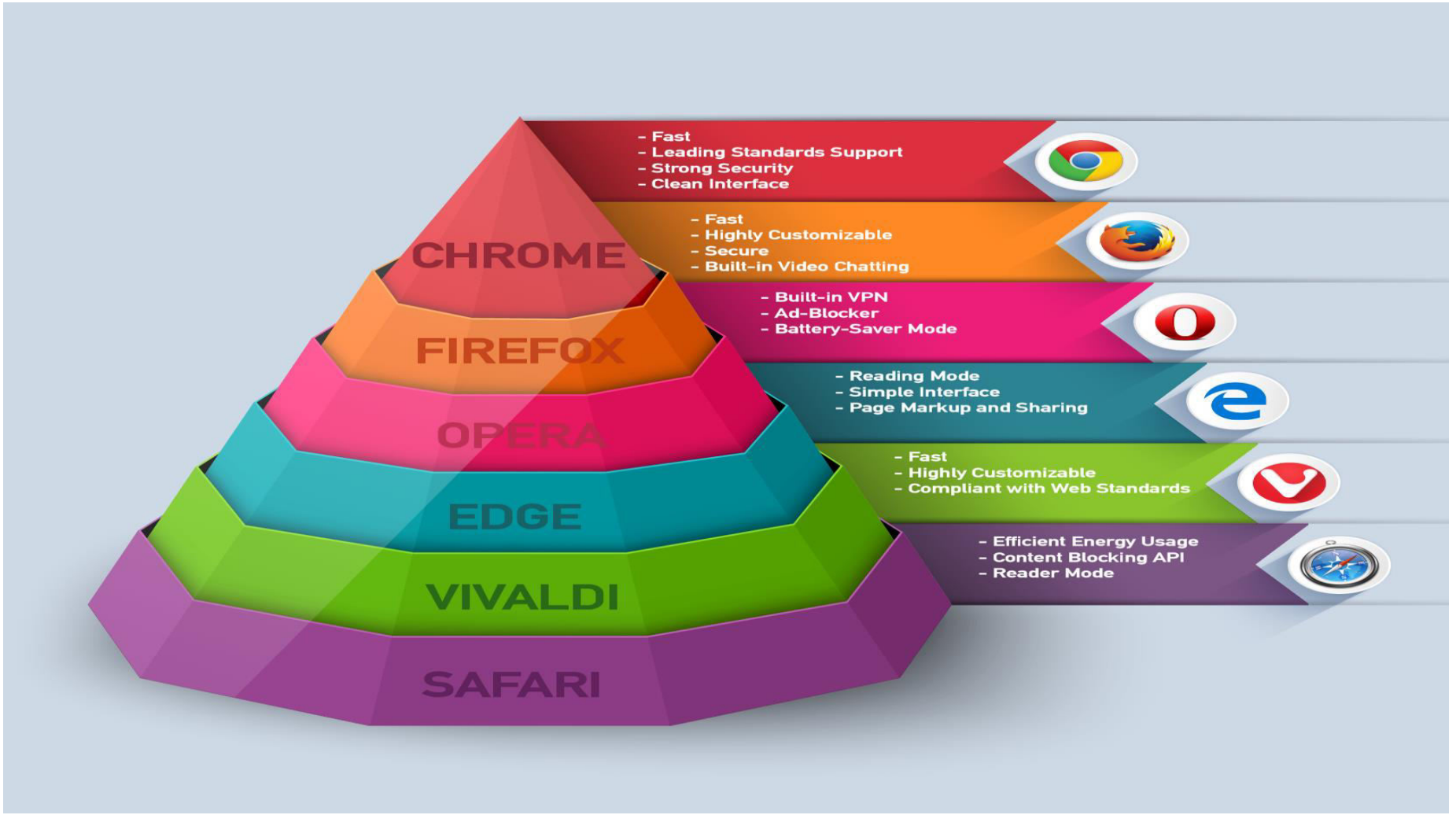
6. What is Web Browser?

- A web browser (also referred to as an Internet browser or simply a browser) is **application software for accessing the World Wide Web or a local website**.
- A web browser takes you anywhere on the internet, letting you see text, images and video from anywhere in the world.
- Web browsers are used primarily for displaying and accessing websites on the internet, as well as other content created using languages such as Hypertext Markup Language (HTML) and Extensible Markup Language (XML).
- Browsers translate web pages and websites delivered using Hypertext Transfer Protocol (HTTP) into human-readable content. They also have the ability to display other protocols and prefixes, such as secure HTTP (HTTPS), File Transfer Protocol (FTP), email handling (mailto:), and files (file:).
- In addition, most browsers also support external plug-ins required to display active content, such as in-page video, audio and game content.

(Continued…)

- Currently you must be using any sort of Web browser while you are navigating through a website On the Web, when you navigate through pages of information, this is commonly known as **web browsing or web surfing.**

Web Browser Comparison

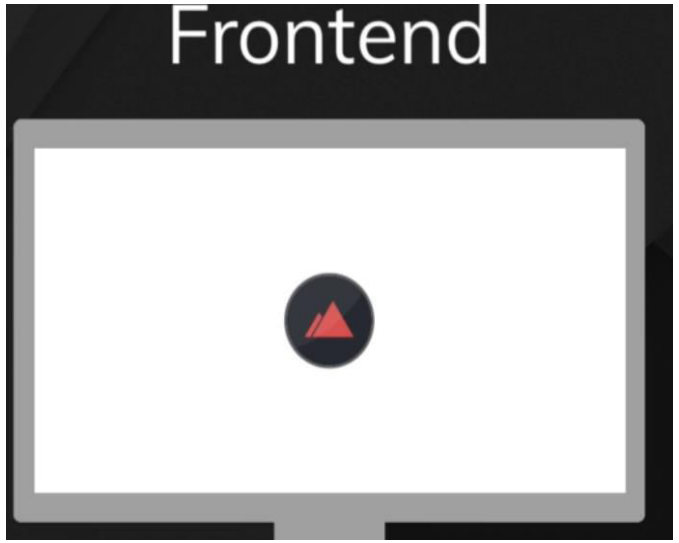


7. Web Site Development Basics

- Who's a web developer?
  - Those websites which you scroll to are created by none other than web developers
- In Web Development we have 3 paths:
  - Frontend developer
  - Backend developer
  - Full-stack developer

7.1 Frontend developer

- Front-End is the UI (User Interface), it deals with the website’s overall appearance, on how interactive and dynamic it is.
- For mastering it, get clear with all the elements of HTML, CSS, and JavaScript



Frontend development

- Features of website that are immediately observable & accessible by the end-user / client is known as frontend development.
- A frontend developer is in charge of everything you see on a website be it styling, graphics, text, alignment, colors and so on whose end goal is to improve user experience and make it as seamless as possible.
- Frontend developers also contribute to overall design and aesthetic, along with debugging.
- The major goal of frontend developers is to make websites responsive

Front-end technologies

HTML, CSS, and JavaScript

Basic requirements to build a website

Hypertext Markup Language(HTML)

- HTML, in full hypertext markup language, **a formatting system for displaying material retrieved over the Internet**.
- Each retrieval unit is known as a Web page (from World Wide Web), and such pages frequently contain hypertext links that allow related pages to be retrieved.

Understanding HTML with an example

- HyperText Markup Language
- A standard markup language for giving a static skeleton to web application
- It's a well standardized system

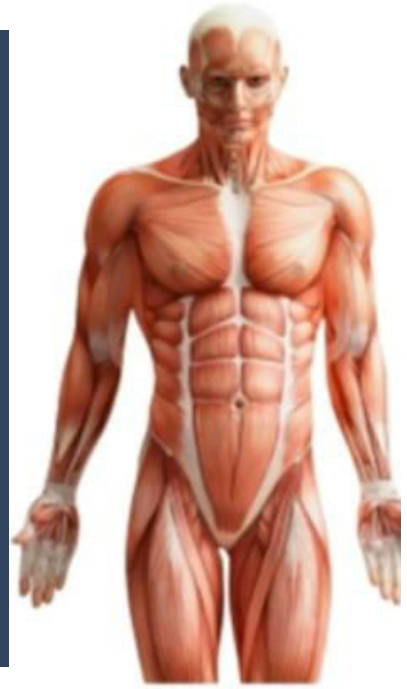
Cascading Style sheets (CSS)

- CSS helps us to **control the text color, font style, the spacing between paragraphs, sizing of columns, layout designs, and many more**.
- It is independent of HTML, and we can use it with any XML-based markup language. It is recommended to use CSS because the HTML attributes are being deprecated.

Understanding CSS with an example

CSS

{..}



- Cascading Style Sheets
- It is a style sheet language used to handle the presentation of the web page containing HTML.
- It make our beautiful and modern looking

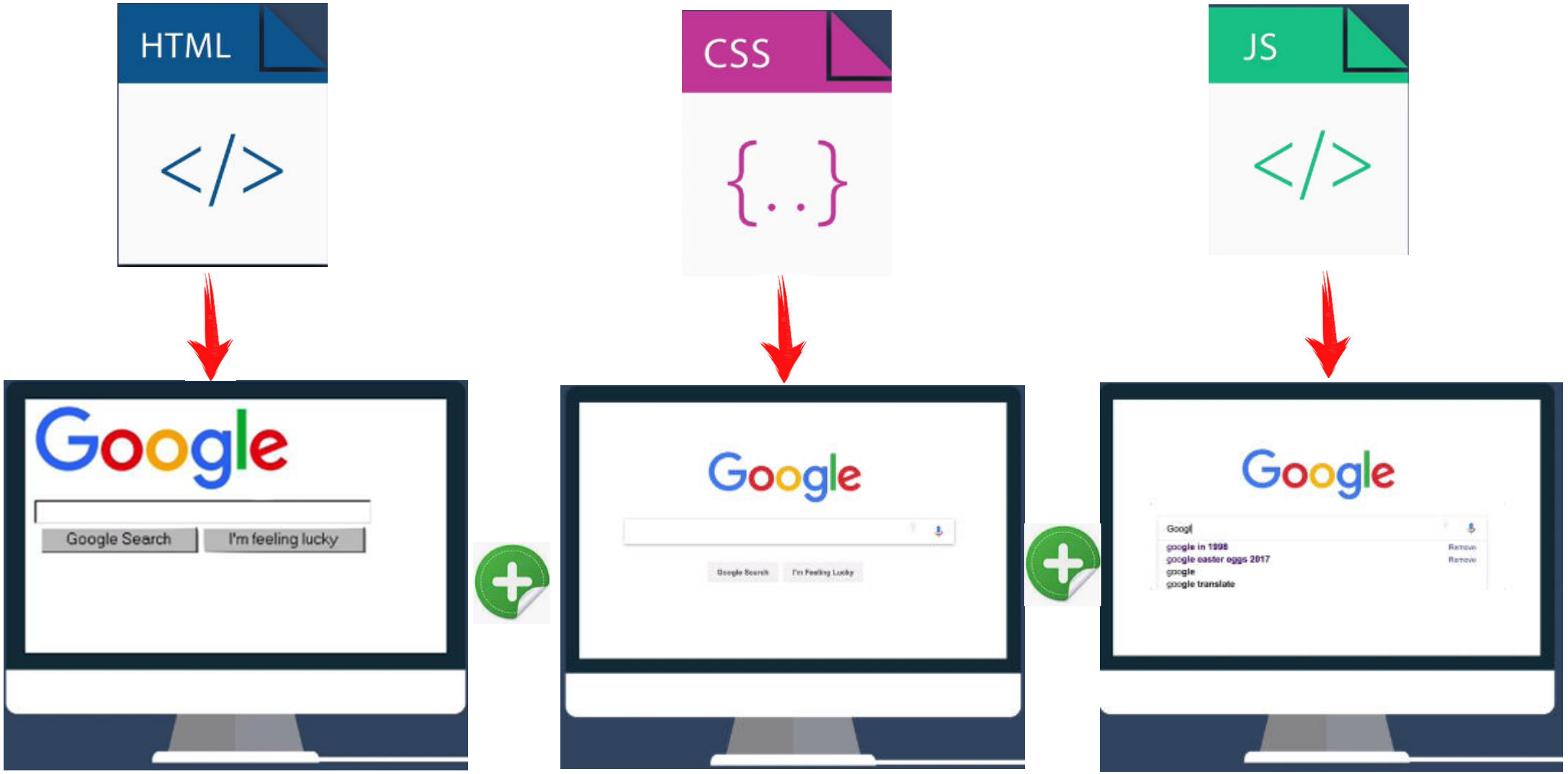
Java Script (JS)

- What is JavaScript and why it is used?
- JavaScript is **a text-based programming language used both on the client-side and server-side that allows you to make web pages interactive**.
- Where HTML and CSS are languages that give structure and style to web pages, JavaScript gives web pages interactive elements that engage a user

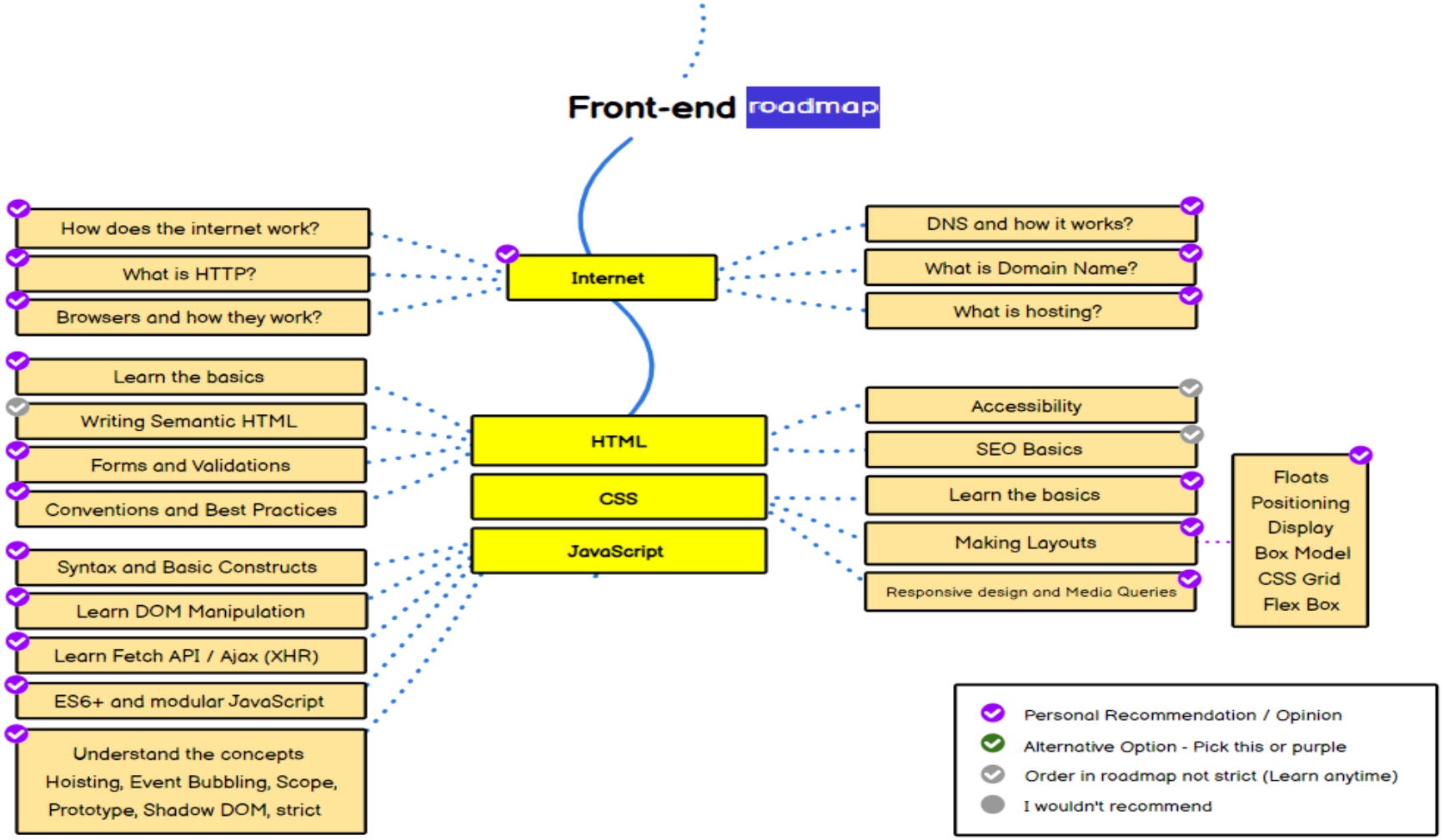
Understanding JS with an example

- JavaScript
- It is a high-level dynamic interpreted programming language
- It allows client-side scripting to create completely dynamic web applications and websites

Frontend With Example



Frontend Roadmap



7.2 Backend developer

- Back-end developers are the experts who build and maintain the mechanisms that process data and perform actions on websites.
- Unlike front-end developers, who control everything you can see on a website, back-end developers are involved in data storage, security, and other server-side functions that you cannot see

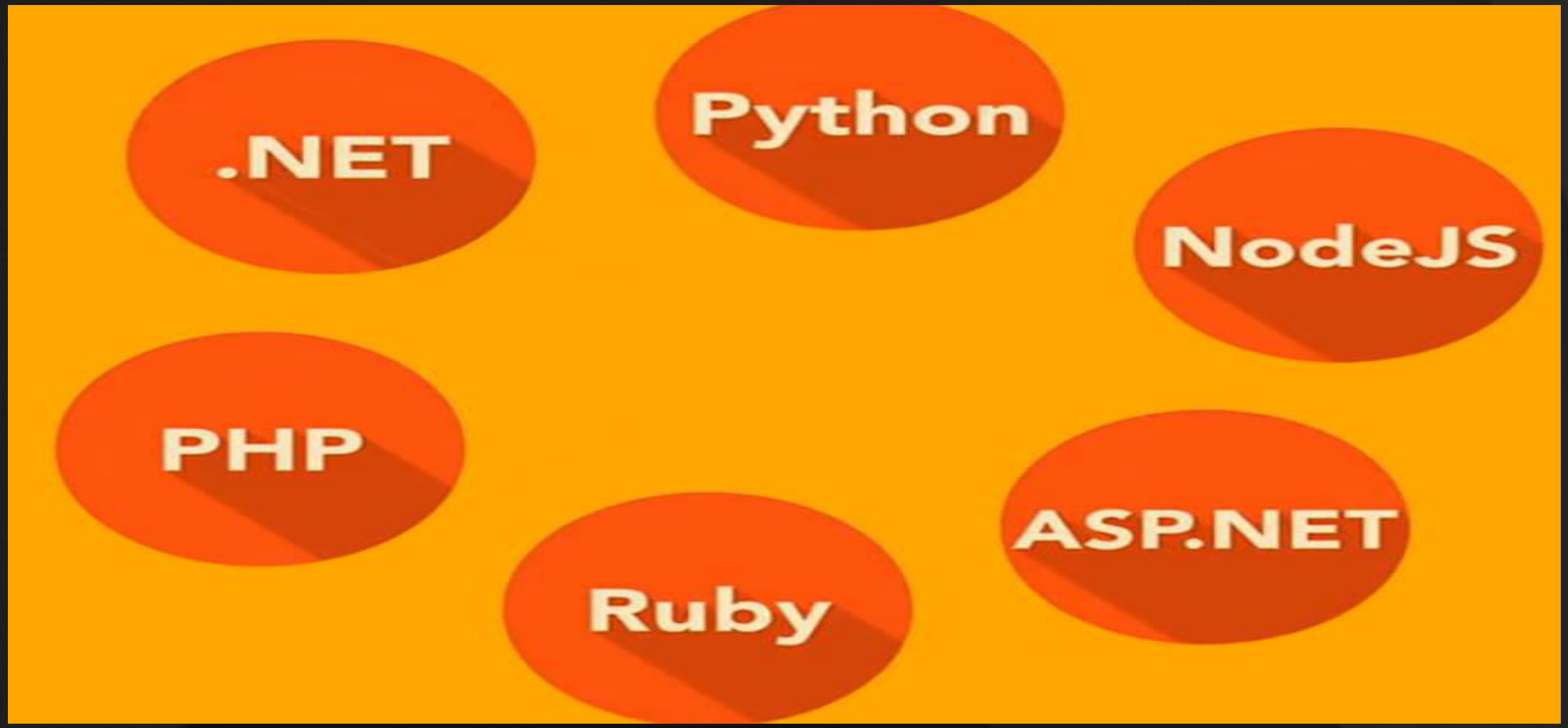
Backend development

- Web development that occurs at the back end of programs is accurately termed as backend development.
- It covers server-side web application logic and integration and activities like writing APIs, creating libraries, and working with system components
- Backend developers create code that allows a database and an application to communicate with one another

Backend

- Stores data
- Allows for a dynamic website
- Holds together frontend components

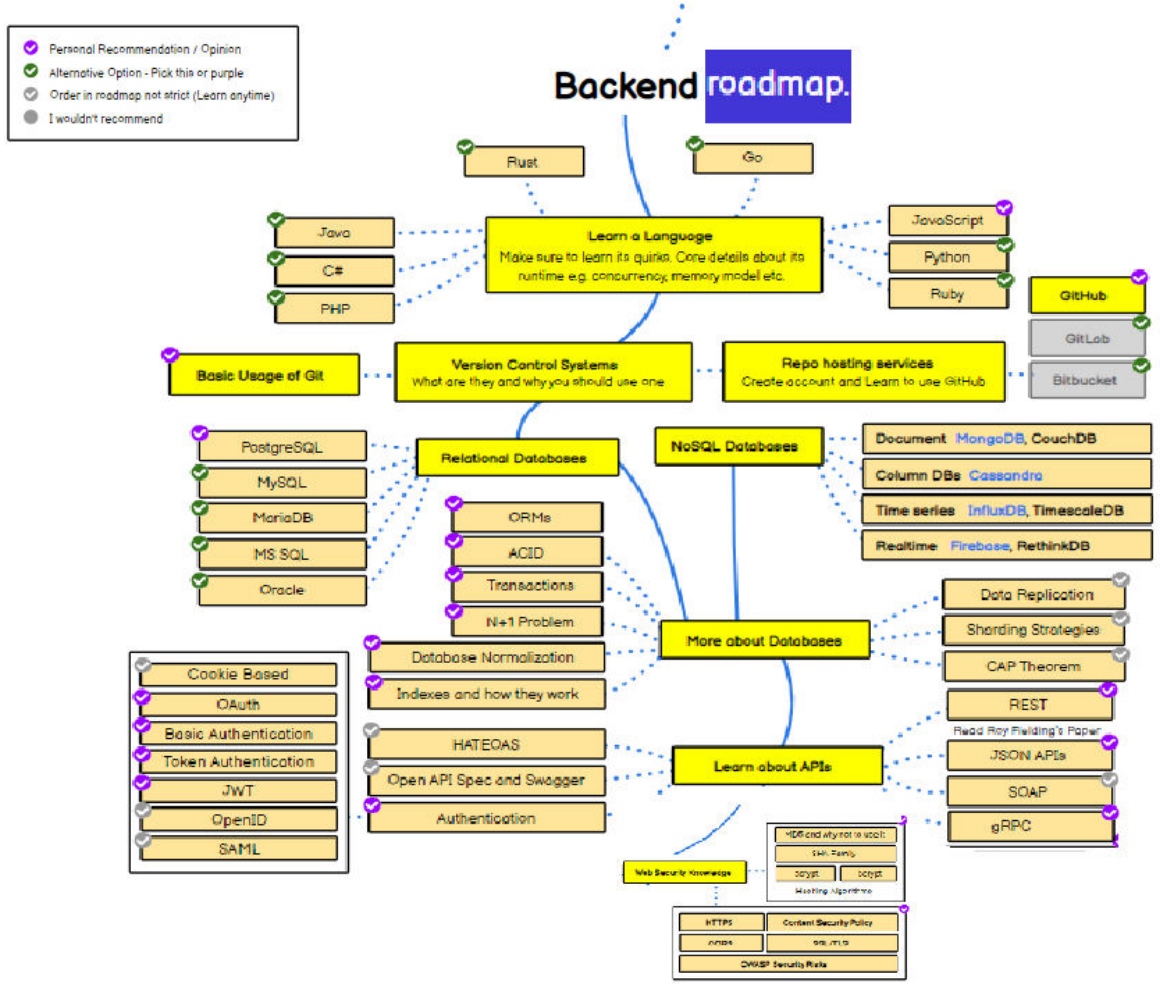
Common Backend Languages



Commonly Used Databases

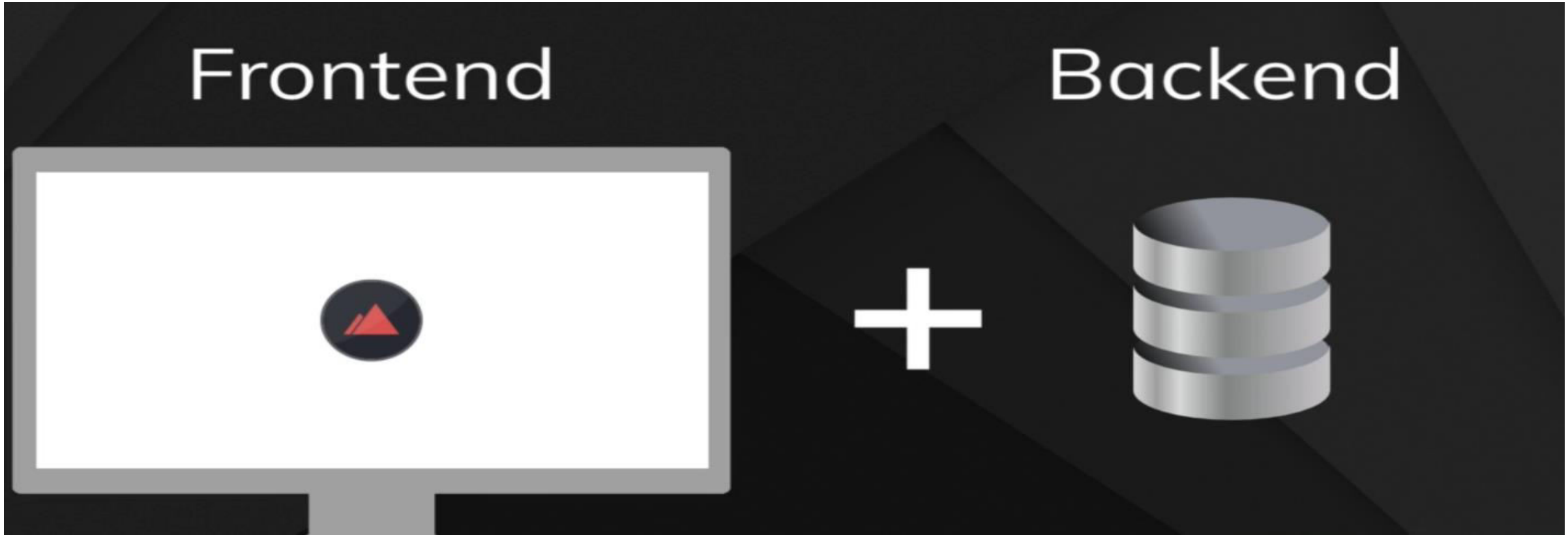
- Firebase
- MongoDB
- MySQL

Back end road map



7.3 Full-stack developer

- A **full-stack web developer** looks after both the front-end and the back-end parts.

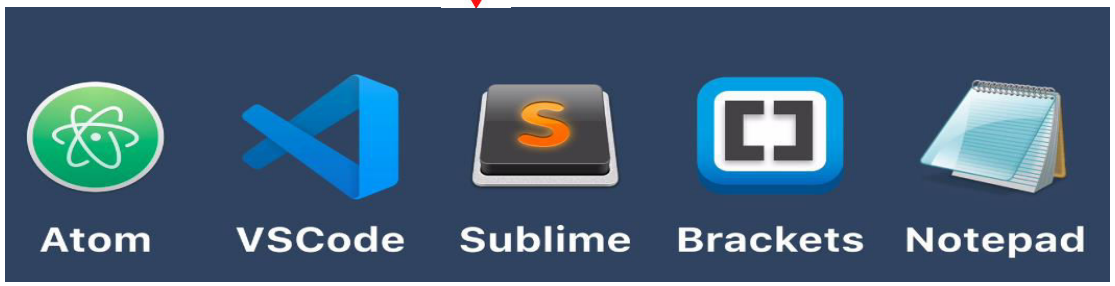


Fullstack development

- Fullstack developers are "jack of all trades"
- They are skilled in both frontend and backend languages and can work with multiple frameworks


JS
HTML
CSS

7.4 Setup for Web Development



Sample web page

https://techcrunch.com

TC

Startups
Apps
Gadgets
Events
Videos

Crunchbase
More

Search

Apple
Transportation
Gaming
Security

Login / Sign up

Review: Apple Watch Series 4

A larger screen, improved tracking and serious health tools make the best smartwatch even better

Brian Heater

The Latest

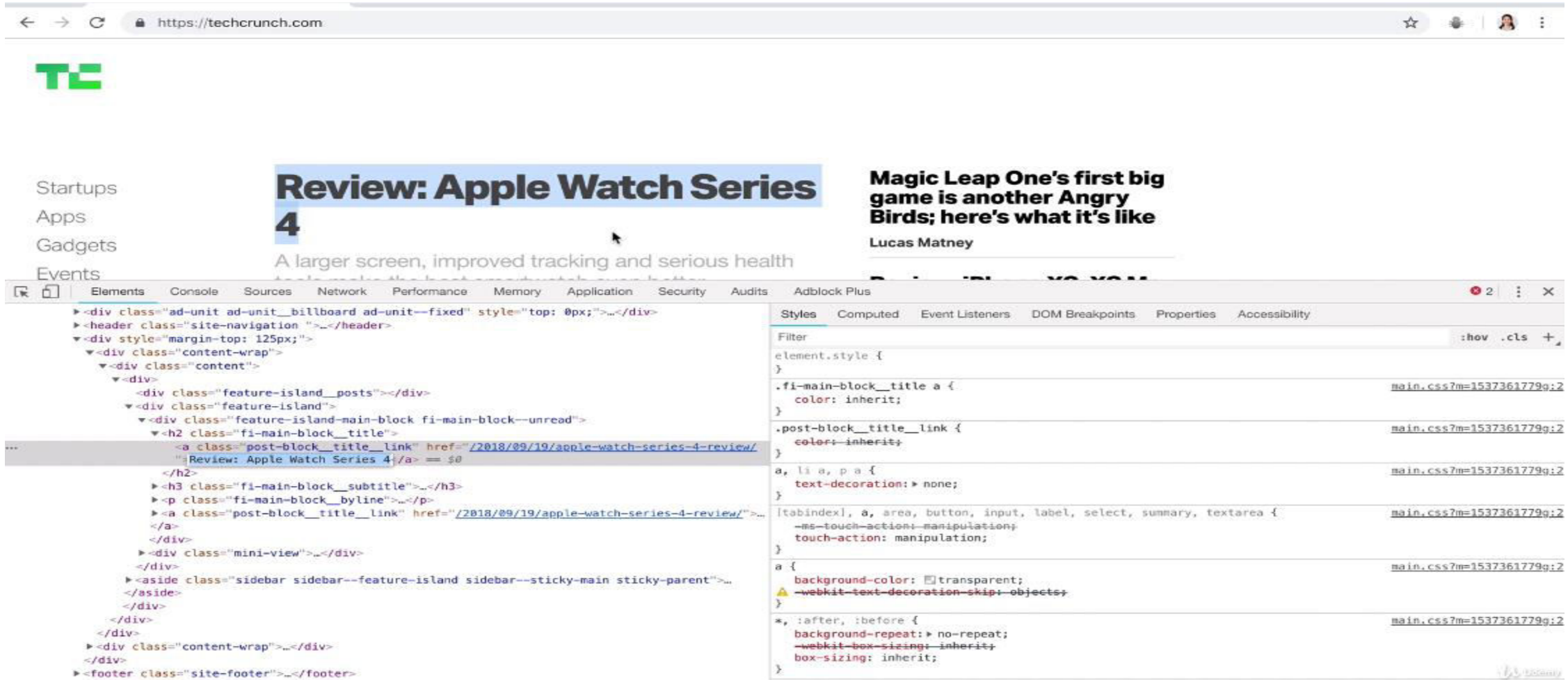
Magic Leap One's first big game is another Angry Birds; here's what it's like
Lucas Matney

Review: iPhone XS, XS Max and the power of long-term thinking
Matthew Panzarino

Sony announces the PlayStation Classic, its own mini retro console
Taylor Hatmaker

Peter Thiel's argument that Silicon Valley has been 'brainwashed' by higher education is tired
Connie Loizos

Inspecting source code of a web page



Web Contents

- Web content is the textual, visual or aural content that is encountered as part of the user
- experience on websites.
- It may include, among other things: text, images, sounds, videos andanimations.

HTML web content

- Even though we may embed various protocols within web pages, the "web page" composed of "html" (or some variation) content is still the dominant way whereby we share content.
- And while there are many web pages with localized proprietary structure (most usually, business websites), many millions of websites abound that are structured according to a common core idea.

Types of Website Content

- A website, or individual web page, can be static or dynamic.
  - Static Web Site
  - Dynamic Web Site

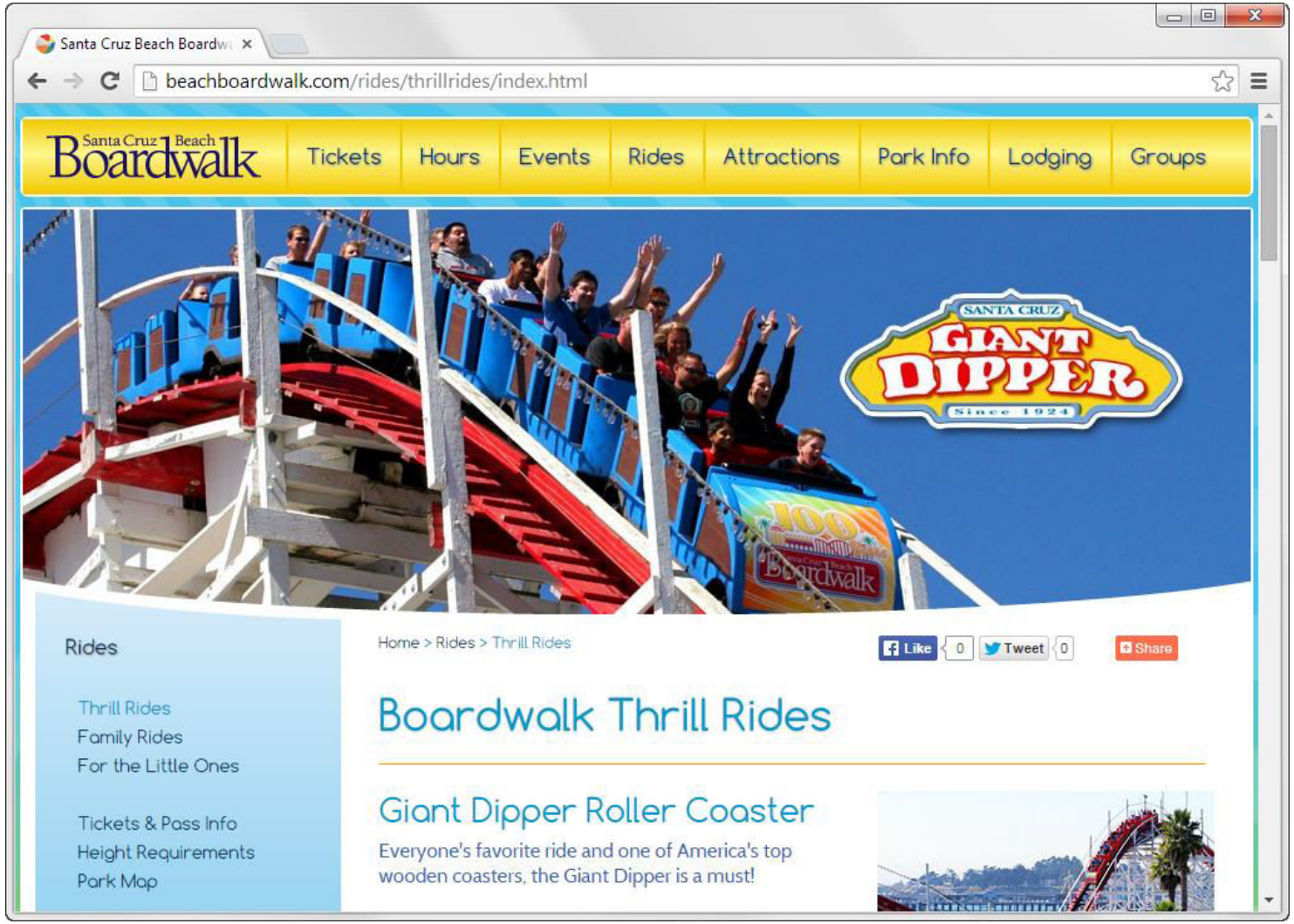
Static Web Site

- A static website or static web page contains information that doesn't change automatically. It remains the same, or static, for every viewer of the site.
- A static web page (sometimes called a flat page) is a web page that is delivered to the user exactly as stored.
- A static web page displays the same information for all users, from all contexts, subject to modern capabilities of a web server to negotiate content-type or language of the document where such versions are available and the server is configured to do so.
- Static web pages are often HTML documents stored as files in the file system and made available by the web server over HTTP.
- Static Web pages are very simple in layout and informative in context.

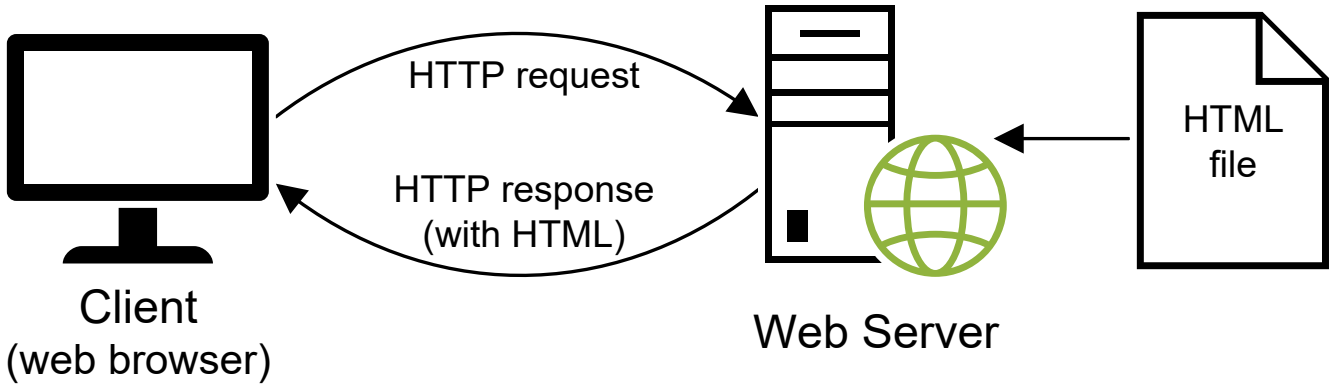
Advantages and **Disadvantages** of Static Web Site

- **Advantages**
  - No programming skills are required to create a static page.
  - Inherently publicly cacheable (i.e. a cached copy can be shown to anyone).
  - No particular hosting requirements are necessary.
  - Can be viewed directly by a web browser without needing a web server or application server, for example directly from a CDROM or USBDrive
- **Disadvantages**
  - Any personalization or interactivity has to run client-side (ie. In the browser), which is restricting.
  - Maintaining large numbers of static pages as files can be impractical without automated tools

A static web page



How a web server processes a static web page



Dynamic Web Sites

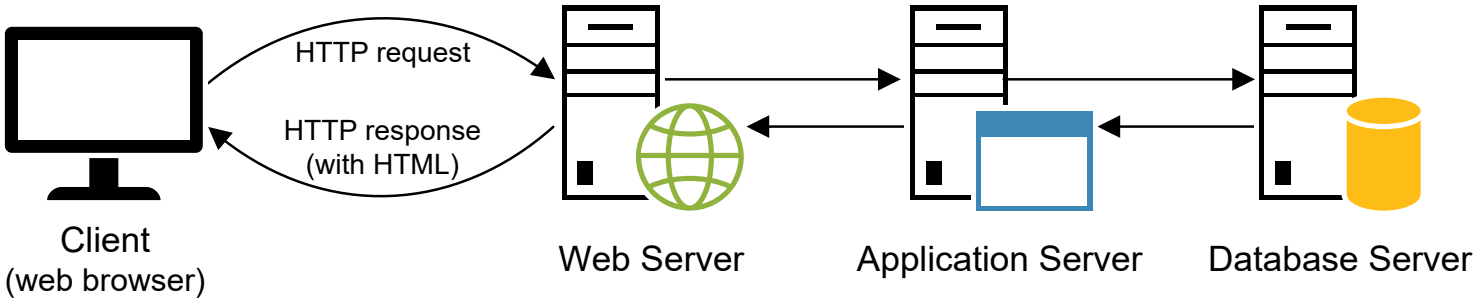
- A dynamic website or dynamic web page contains information that changes, depending on the viewer, the time of the day, the time zone, the viewer's native language, and other factors.
- A dynamic web page is a kind of web page that has been prepared with fresh information (content and/or layout), for each individual viewing.
- It is not static because it changes with the time (ex. news content), the user (ex. preferences in a login session), the user interaction (ex. web page game), and the context (parametric customization), or any combination of the foregoing.
- A dynamic website can contain client-side scripting or server-side scripting to generate the changing content or a combination of both scripting types. These sites also include HTML programming for the basic structure. The client-side or server-side scripting takes care of the guts of the site.

Application areas of Dynamic Website

- Dynamic web page is required when the following necessities arise:
  - Need to change main pages more frequently to encourage clients to return to the site.
  - Long list of products/services offered that are also subject to upgradation
  - Introducing sales promotion schemes from time to time
  - Need for a more sophisticated ordering system with a wide variety of functions
  - Tracking and offering personalized services to clients.
  - Facility to connect Web site to the existing back-end system.

A dynamic web page at amazon.com

How a web server processes a dynamic web page



Static vs Dynamic

Static

- HTML + CSS + JavaScript
- No processing of content on server
- Loading time is fast
- No interaction with databases.
- Static does NOT mean that the page never changes – it's just **pre-built during development!**

Dynamic

- Backend language (e.g. Node.js, PHP)
- Slower than static websites
- Updates and interaction with databases are possible
- Dynamic does NOT mean that there's no HTML page being served – it's just **built dynamically for each request**

8. Web and internet Terminologies

Website:

- A website is a set of web pages consisting of text, audio and video. Web servers host websites.

Web Page:

- Web pages are resources of information. They are generally created in the HTML format and provide the web users with navigational abilities through hyperlinks to other web pages on the web.

Home Page:

- The term home page is used to refer to the page that is the default page of any website. It is the main page of a complex website.

Web Browser:

- A web browser is a software application that facilitates user interaction with the text, audio, video and other information that is located on the web.

Bounce Rate

- A website's bounce rate is the percentage of people who leave the site from the same page they entered the site, without clicking through to any other pages.

Client-Side

Client-side refers to scripts that are run in a viewer's browser, instead of on a web server (as in server-side scripts). Client-side scripts are generally faster to interact with, though they can take longer to load initially.

Server-Side

- Server-side refers to scripts run on a web server, as opposed to in a user's browser. Server-side scripts often take a bit longer to run than a client-side script, as each page must reload when an action is taken.

Content Management System

- Also known as a CMS, the Content Management System is a backend tool for managing a site's content that separates said content from the design and functionality of the site.
- Using a CMS generally makes it easier to change the design or function of a site independent of the site's content. people who aren't programmers, saving on the cost of hiring a programmer because you can add/edit/delete content yourself.

Web and internet Terminologies

Hyperlink

- A hyperlink is a link from one web page to another, either on the same site or another one. Generally these are text or images, and are highlighted in some way (text is often underlined or put in a different color or font weight). The inclusion of hyperlinks are the “hyper” part of “hypertext.”

Meta Data

- Meta data is the data contained in the header that offers information about the web page that a visitor is currently on. The information contained in the meta data isn’t viewable on the web page (except in the source code). Meta data is contained within meta tags.

Navigation

- Navigation refers to the system that allows visitors to a website to move around that site. Navigation is most often thought of in terms of menus, but links within pages, related links, and any other links that allow a visitor to move from one page to another are included in navigation.

Open Source

- Open source refers to the source code of a computer program being made available to the general public. Open source software includes both web-based and desktop applications. Open source programs are generally free or very low cost and are developed by teams of people, sometimes comprised mostly of volunteers.

Plug-In

- A plug-in is a bit of third party code that extends the capabilities of a website. It's most often used in conjunction with a CMS or blogging platform. Plug-ins are a way to extend the funtionality of a website without having to redo the core coding of the site. Plugins can also refer to bits of third-party software installed within a computer program to increase its functionality.

Script

- Generally refers to a portion of code on an HTML page that makes the page more dynamic and interactive. Scripts can be written in a variety of languages, including JavaScript.

Web Server

- A web server is a computer that has software installed and networking capabilities that allow it to host web sites and pages and make them available to internet users located elsewhere. There are a few different setups that can be used for a web server, including the LAMP setup mentioned earlier.

Web Standards

- Standards are specifications recommended by the World Wide Web Consortium for standardizing website design. The main purpose of web standards is to make it easier for both designers and those who create web browsers to make sites that will appear consistent across platforms.
- This can be checked through various validation services, most commonly the one from W3C.

ASP

- A coding language that is compatible with Windows servers. Normally used for increased functionality on a website or to work with a database. It works in conjunction with html and html variants.

BLOG

- An online journal or diary and a very popular current method of sharing your thoughts with the world. It is also very popular as a marketing tool. This article is found within Thinking IT's blog.

ISP

- Internet Service Provider. The company that provides you with internet access (connection) and related services is your ISP.

SEARCH ENGINE

- A programme that collects, stores, arranges and normally ranks the various resources available on the internet. It is most commonly on a website and used to find other websites – much like the yellow pages is used in the brick and mortar world.

SITEMAP: This is an index to all the content on a website. It is normally accessible from at least the front page of the site and is used for two purposes: to help people find what they are looking for on the site and to help search engines find all your links.

- **Cache**: Web browsers maintain a cache of recently visited web pages. Some of them use an external proxy web cache, which is a server program through which web requests pass. This enables the browsers to cache frequently visited pages. Even search engines make available already indexed web pages through their caches.
- **Web Cookie**: Also known as an HTTP cookie, it is piece of text that is exchanged between the web client and the web server. It is sent by the web server to the web client and returned unchanged by the client each time it accesses the server.
- **Session**: It is an exchange of information between a computer and its user. It is established for a certain period of time after which it ends.
- **Hyperlink**: A reference in a document to another section of the document or to another document is termed as a hyperlink. Hyperlinks are used to redirect the user from one section of a page content to another.

- **Spamming**: The act of sending unsolicited bulk messages over an email system is known as spamming. It is an undesirable use of the electronic messaging systems.
- **Phishing:** It is a fraudulent activity of acquiring the sensitive information by the use of a fake identity during electronic communication. It is implemented by means of emails and instant messages wherein a user is lured to enter his/her details, which are actually captured by a fraudulent website.
- **Hacking:** Hacking is the activity of programmatically gaining access to a computer application that is otherwise inaccessible. The act of gaining an unauthorized access to a computer is known as hacking. Hacking of passwords that leads to breach of email privacy is a threat to communication over the Internet. Internet crimes refer to all the criminal activities that are carried over the Internet.
- **Email Virus:** It is a computer code that is transmitted through an email in the form of an attachment. The email attachment causes the destruction of some of the files on the receiver computer's hard disk and is programmatically emailed to the contacts in the address book of the receiver.

Chapter 2
Introduction to HTML

Basics of HTML
- Tools used for HTML formatting
- Understanding header, paragraph, table, image, lists, hyperlink, frame and layout tags

HTML/XHTML

HTML (HyperText Markup Langage) is the language used to create web page documents.

The updated version, XHTML (eXtensible HTML) is essentially the same language with stricter syntax rules. It is common to see HTML and XHTML referred to collectively as (X)HTML

- Extensible HTML.
- Almost identical to HTML 4.01.
- It is stricter and cleaner version of HTML.
- HTML defined as an XML application.
- XHTML was published as a w3c recommendation in January of 2000.

.

XML

- **XML (Extensible Markup Language)**
  - Much like HTML.
  - To carry data, not to display data.
  - Tags are not predefined.
  - It is a W3C recommendation.

  - **Difference between XML and HTML**
  - XML is not a replacement for HTML.
  - XML and HTML designed for different goal.
    - XML is to transport and store data with the focus of what the data is. (Carry)
    - HTML is to display data with the focus of how the data looks. (Displaying)

HTML

- HTML stands for **H**yper **T**ext **M**arkup **L**anguage
- HTML is not a programming language, it is a **markup language**
- A markup language is a set of **markup tag**
- HTML uses **markup tags** to describe web pages
- Simple to write and simple to learn
- The markup tags tell the Web browser how to display the page
- An HTML file must have an .htm or .htm l file extension
- An HTML file can be created using a simple text editor such as Note pad

HTML versions

- HTML version Time line
  - **Version** **Year**
  - HTML 1991
  - HTML 2.0 1995
  - HTML 3.2 1997
  - HTML 4.01 1999
  - XHTML 2000
  - HTML5 2014

HTML Tags

- HTML markup tags are usually called HTML tags.
- HTML tags are keywords surrounded by **angle brackets** like <html>
- HTML tags normally **come in pairs** like and
- The first tag in a pair is the **start tag,** the second tag is the **end tag**
- Start and end tags are also called **opening tags** and **closing tags**

Basic HTML Document Structure

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

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

------ HTML Basics

- The text between **<html>** and **</html>** describes an HTML document
- The text between **<head>** and **</head>** provides information about the document
- The text between **<title>** and **</title>** provides a title for the document
- The text between **<body>** and **</body>** describes the visible page content
- The text between **<h1>** and **</h1>** describes a heading
- The text between ** ** and ** ** describes a paragraph

HTML Elements

- HTML documents are defined by HTML elements. An HTML element is everything between the start tag and the end tag. The start tag is often called the opening tag. The end tag is often called the closing tag.
  - An HTML element starts with a **start tag / opening tag.**
  - An HTML element ends with an **end tag / closing tag.**
  - The **element content** is everything between the start and the end tag.
  - Some HTML elements have **empty content.**
  - Empty elements are **closed in the start tag.**
  - Most HTML elements can have **attributes.**

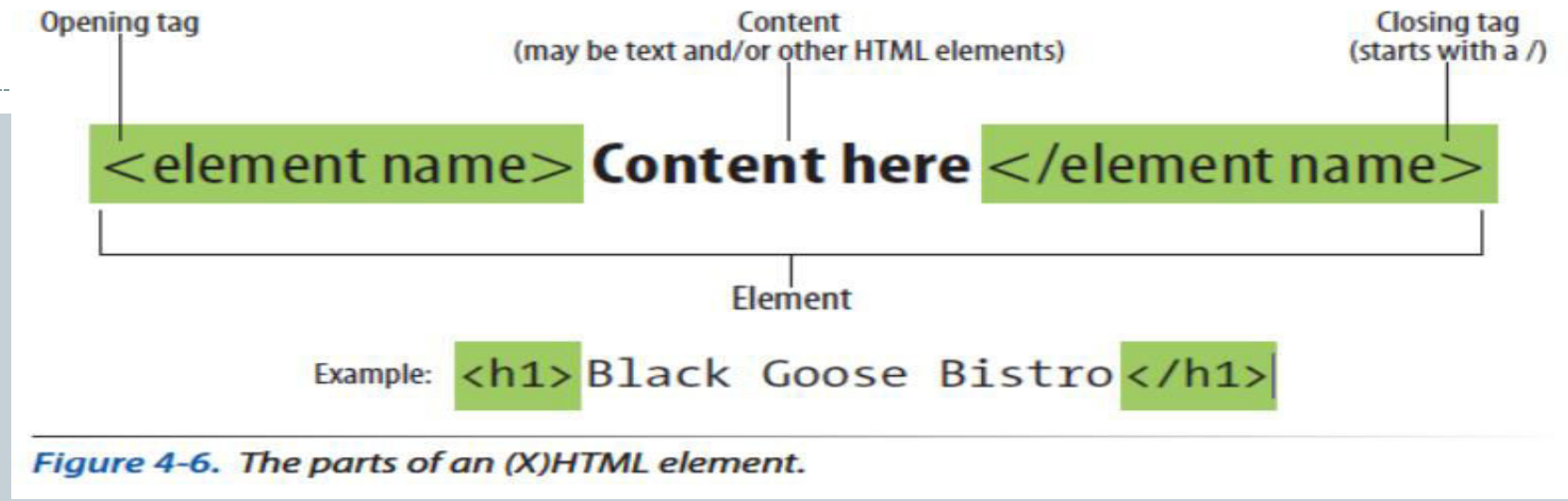


Figure 4-6. The parts of an (X)HTML element.

OPENING TAG	ELEMENT CONTENT	CLOSING TAG
	This is a paragraph.	
	This is a link.	

Nested HTML Elements

- Most HTML elements can be nested (can contain other HTML elements). The following example contains three HTML elements. Notice that the <p> element is nested in the <body> element, which in turn is nested in the <html> element.

Example

```
<html>

<body>
<p>This is my first paragraph.</p>
</body>

</html>
```

HTML Attributes

- HTML elements can have **attributes**
- Attributes provide **additional information** about an element
- Attributes are always specified in **the start tag**
- Attributes come in name/value pairs like: **name="value"**

Attribute Example
HTML links are defined with the <a> tag. The link address is specified in the href attribute:
Example
This is a link

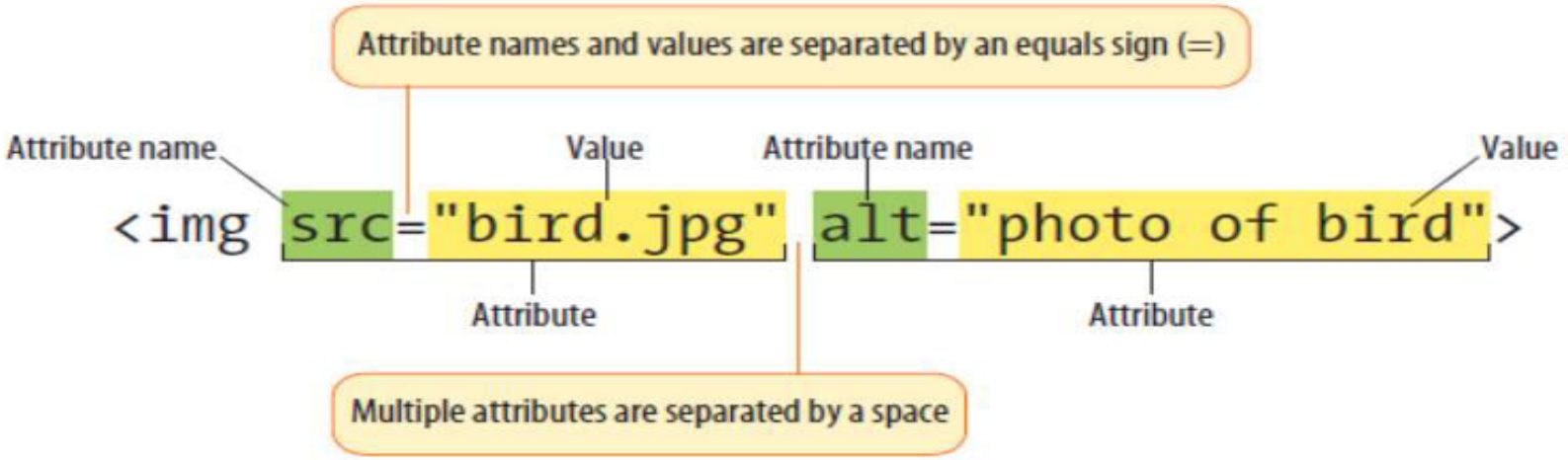


Figure 4-11. An element with attributes.

Always Quote Attribute Values

Attribute values should always be enclosed in quotes. Double style quotes are the most common, but single style quotes are also allowed.

Some attributes

- Below is a list of some attributes that are standard for most HTML elements:
- **Attribute Value Description**
- Class *classname* Specifies a classname for an element
- Id *id* Specifies a unique id for an element
- Style *style_definition* Specifies an inline style for an element
- Title *tooltip_text* Specifies extra information about an element (displayed as a tool tip)

HTML Headings

- Headings are defined with the <h1> to <h6> tags.
- <h1> defines the most important heading. <h6> defines the least important heading.
- **Example**
- <h1>This is a heading</h1>
  <h2>This is a heading</h2>
  <h3>This is a heading</h3>
- **Headings Are Important**
- Use HTML headings for headings only. Don't use headings to make text **BIG** or **bold**.
- Search engines use your headings to index the structure and content of your web pages.
- Since users may skim your pages by its headings, it is important to use headings to show the document structure.

HTML Lines

- The <hr /> tag creates a horizontal line in an HTML page

The hr element can be used to separate content:
Example

```
<p>This is a paragraph</p>
<hr />
<p>This is a paragraph</p>
<hr />
<p>This is a paragraph</p>
```

HTML Comments

• Comments can be inserted into the HTML code to make it more readable and understandable. Comments are ignored by the browser and are not displayed.
Example

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

HTML Paragraphs

- Paragraphs are defined with the tag.
- **Example**
- This is a paragraph
  This is another paragraph

HTML Line Breaks

Use the
 tag if you want a line break (a new line) without starting a new paragraph:

Example

This is
a para
graph with line breaks

The
 element is an empty HTML element. It has no end tag.

**
 or
**

•With HTML, you cannot change the output by adding extra spaces or extra lines in your HTML code.

•The browser will remove extra spaces and extra lines when the page is displayed. Any number of lines counts as one line, and any number of spaces count as one space.

HTML Text Formatting Tags

Tag	Description
<b>	Defines bold text
<big>	Defines big text
	Defines emphasized text
<i>	Defines italic text
<small>	Defines small text
	Defines strong text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<ins>	Defines inserted text
<del>	Defines deleted text

HTML "Computer Output" Tags

- **Tag Description**
- <code> Defines computer code text
- <kbd> Defines keyboard text
- <samp> Defines sample computer code
- <tt> Defines teletype text /Not supported HTML5
- <var> Defines a variable
- <pre> Defines preformatted text

HTML Citations, Quotations, and Definition Tags

- **Tag Description**
- <abbr> Defines an abbreviation
- <acronym> Defines an acronym/ Not supported HTML5
- <address> Defines contact information for the author/owner of a document
- <bdo> Defines the text direction/ dir="rtl"
- <blockquote> Defines a long quotation
- <q> Defines a short quotation
- <cite> Defines a citation
- <dfn> Defines a definition term

Bdo =Bi-Directional Override.

The HTML Tag Should NOT be Used

- The tag is deprecated in HTML 4, and removed from HTML5.
- The World Wide Web Consortium (W3C) has removed the tag from its recommendations. In HTML 4, style sheets (CSS) should be used to define the layout and display properties for many HTML elements.
- The example below shows how the HTML could look by using the tag:
- **Example**
-

  This paragraph is in Arial, size 5, and in red text color.

Using the HTML Style Attribute

- It is time consuming and not very practical to style HTML elements using the style attribute.
- **The preferred way to add CSS to HTML, is to put CSS syntax in separate CSS files.**
- **HTML Style Example - Background Color**
- The background-color property defines the background color for an element:
- **Example**
- <html>
  <body style="background-color:yellow;">
  <h2 style="background-color:red;">This is a heading</h2>
  This is a paragraph.
  </body>
  </html>

HTML Style Example - Font, Color and Size

- The font-family, color, and font-size properties defines the font, color, and size of the text in an element:
- **Example**
- <html>
  <body>
  <h1 style="font-family:verdana;">A heading</h1>
  A paragraph.
  </body>
  </html>
- The font-family, color, and font-size properties make the old <font> tag obsolete.

HTML Style Example - Text Alignment

- The text-align property specifies the horizontal alignment of text in an element:

Example

- <html>
  <body>
  <h1 style="text-align:center;">Center-aligned heading</h1>
  This is a paragraph.
  </body>
  </html>

Deprecated Tags and Attributes

- In HTML 4, several tags and attributes were deprecated. Deprecated means that they will not be supported in future versions of HTML.

These tags and attributes should be avoided:

Tags	**Description**
	Deprecated. Defines centered content
<font> and <basefont>	Deprecated. Defines HTML fonts
<s> and <strike>	Deprecated. Defines strikethrough text
<u>Deprecated.	Defines underlined text

Attributes	**Description**
Align	Deprecated. Defines the alignment of text
Bgcolor	Deprecated. Defines the background color
Color	Deprecated. Defines the text color

For all of the above: Use styles instead!

HTML Lists

Unordered List:

- Item
- Item
- Item
- Item

Ordered List:

1. First item
2. Second item
3. Third item
4. Fourth item

A Description List/Definition Lists

Coffee

- black hot drink

Milk

- white cold drink

Unordered List:

```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Output

- Coffee
- Tea
- Milk

```
<ul style="list-style-type:square">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Output

- Coffee
- Tea
- Milk

Unordered List: **style** attribute

style attribute can be added to an **unordered list**, to define the style of the marker:

Style	Description
list-style-type:disc	**The list items will be marked with bullets (default)**
list-style-type:circle	**The list items will be marked with circles**
list-style-type:square	**The list items will be marked with squares**
list-style-type:none	**The list items will not be marked**

Ordered HTML Lists

```
<ol>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Output

1. Coffee
2. Tea
3. Milk

```
<ol type="A">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Output

A. Coffee
B. Tea
C. Milk

```
<ol type="I">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Output

I. Coffee
II. Tea
III. Milk

Ordered HTML Lists - The Type Attribute

- A **type** attribute can be added to an **ordered list**, to define the type of the marker:

Type	Description
type="1"	The list items will be numbered with numbers (default)
type="A"	The list items will be numbered with uppercase letters
type="a"	The list items will be numbered with lowercase letters
type="I"	The list items will be numbered with uppercase roman numbers
type="i"	The list items will be numbered with lowercase roman numbers

HTML Description Lists

- A description list is a list of terms, with a description of each term.
- The **<dl>** tag defines the description list, the **<dt>** tag defines the term (name), and the **<dd>** tag describes each term:

```
<dl>
  <dt>Coffee</dt>
  <dd>- black hot drink</dd>
  <dt>Milk</dt>
  <dd>- white cold drink</dd>
</dl>
```

Output

Coffee
- black hot drink

Milk
- white cold drink

Nested HTML Lists

- List can be nested (lists inside lists):

```
<ul>
  <li>Coffee</li>
  <li>Tea
    <ul>
      <li>Black tea</li>
      <li>Green tea</li>
    </ul>
  </li>
  <li>Milk</li>
</ul>
```

Output

- Coffee
- Tea
  - Black tea
  - Green tea
- Milk

Horizontal Lists

- HTML lists can be styled in many different ways with CSS.
- One popular way, is to style a list to be displayed horizontally:

HTML CSS JavaScript PHP

```
<!DOCTYPE html>
<html>
<head>
<style>
ul#menu li {
    display:inline;
}
</style>
</head>
<body>

<h2>Horizontal List</h2>

<ul id="menu">
  <li>HTML</li>
  <li>CSS</li>
  <li>JavaScript</li>
  <li>PHP</li>
</ul>

</body>
</html>
```

HTML Links - Hyperlinks

HTML links are hyperlinks.

A hyperlink is a text or an image you can click on, and jump to another document.

In HTML, links are defined with the **<a>** tag:

HTML Links - Syntax

In HTML, links are defined with the **<a>** tag:

- *link text*

Example

Visit our HTML tutorial

The **href** attribute specifies the destination address (http://www.w3schools.com/html/)

Local Links

- The example above used an absolute URL (A full web address).
- A local link (link to the same web site) is specified with a relative URL (without http://www....).
- **Example**
- HTML Images

HTML Links - Colors

- When you move the mouse over a link, two things will normally happen:
- The mouse arrow will turn into a little hand
- The color of the link element will change
- By default, a link will appear like this (in all browsers):
- An unvisited link is underlined and blue
- A visited link is underlined and purple
- An active link is underlined and red
- You can change the default colors, by using styles:

HTML Links - The target Attribute

- The **target** attribute specifies where to open the linked document.
- This example will open the linked document in a new browser window or in a new tab:
- **Example**
- Visit W3Schools!

HTML Links - The target Attribute

Target Value	Description
_blank	Opens the linked document in a new window or tab
_self	Opens the linked document in the same frame as it was clicked (this is default)
_parent	Opens the linked document in the parent frame
_top	Opens the linked document in the full body of the window
framename	Opens the linked document in a named frame

HTML Links - Image as Link

- It is common to use images as links:
- **Example**
-
  <img src="smiley.gif" alt="HTML tutorial" style="width:42px;height:42px;border:0">

HTML Links - Create a Bookmark

- HTML bookmarks are used to allow readers to jump to specific parts of a Web page.
- Bookmarks are practical if your website has long pages.
- To make a bookmark, you must first create the bookmark, and then add a link to it.
- When the link is clicked, the page will scroll to the location with the bookmark.
- **Example**
- First, create a bookmark with the id attribute:
- <h2 id="tips">Useful Tips Section</h2>
- Then, add a link to the bookmark ("Useful Tips Section"), from within the same page:
- Visit the Useful Tips Section
- Or, add a link to the bookmark ("Useful Tips Section"), from another page:
- **Example**
- Visit the Useful Tips Section

HTML Images

- **HTML Images Syntax**
- In HTML, images are defined with the **<img>** tag.
- The <img> tag is empty, it contains attributes only, and does not have a closing tag.
- The src attribute specifies the URL (web address) of the image:
- <img src="*url*" alt="*some_text*">

The alt Attribute

- The alt attribute specifies an alternate text for an image, if the image cannot be displayed.
- The alt attribute provides alternative information for an image if a user for some reason cannot view it (because of slow connection, an error in the src attribute, or if the user uses a screen reader).
- If a browser cannot find an image, it will display the alt text:
- **Example**
- <img src="wrongname.gif" alt="HTML5 Icon" style="width:128px;height:128px;">

HTML Screen Readers

- A screen reader is a software program that can read what is displayed on a screen.
- Screen readers are useful to people who are blind, visually impaired, or learning disabled.
- Screen readers can read the **alt** attribute.

Image Size - Width and Height

- You can use the **style** attribute to specify the width and height of an image.
- The values are specified in pixels (use px after the value):
- **Example**
- <img src="html5.gif" alt="HTML5 Icon" style="width:128px;height:128px;">
- Alternatively, you can use **width** and **height** attributes. Here, the values are specified in pixels by default:
- **Example**
- <img src="html5.gif" alt="HTML5 Icon" width="128" height="128">

Width and Height or Style?

- Both the width, height, and style attributes are valid in the latest HTML5 standard.
- We suggest you use the style attribute. It prevents styles sheets from changing the original size of images:
- **Example**
- <!DOCTYPE html>
<html>
<head>
<style>
img {
width:100%;
}
</style>
</head>
<body>

<img src="html5.gif" alt="HTML5 Icon"
style="width:128px;height:128px;">
<img src="html5.gif" alt="HTML5 Icon" width="128" height="128">

</body>
</html>

Images in Another Folder

- If not specified, the browser expects to find the image in the same folder as the web page.
- However, it is common to store images in a sub-folder. You must then include the folder name in the src attribute:
- **Example**
- <img src="/images/html5.gif" alt="HTML5 Icon" style="width:128px;height:128px;">

Image Floating

- Use the CSS float property to let the image float.
- The image can float to the right or to the left of a text:
- **Example**
-
  <img src="smiley.gif" alt="Smiley face"
  style="float:right;width:42px;height:42px;">
  The image will float to the right of the text.

  <img src="smiley.gif" alt="Smiley face"
  style="float:left;width:42px;height:42px;">
  The image will float to the left of the text.

Image Maps

- Use the <map> tag to define an image-map. An image-map is an image with clickable areas.
- The name attribute of the <map> tag is associated with the <img>'s usemap attribute and creates a relationship between the image and the map.
- The <map> tag contains a number of <area> tags, that defines the clickable areas in the image-map:
- **Example**
- <img src="planets.gif" alt="Planets" usemap="#planetmap" style="width:145px;height:126px;">

  <map name="planetmap">
  <area shape="rect" coords="0,0,82,126" alt="Sun" href="sun.htm">

HTML Forms

- HTML forms are used to collect user input.
- The **<form>** element defines an HTML form:
- <form>

  .

  form elements

  .

  </form>

- HTML forms contain **form elements**.
- Form elements are different types of input elements, checkboxes, radio buttons, submit buttons, and more.

The <input> Element

- The **<input>** element is the most important **form element**.
- The <input> element has many variations, depending on the **type** attribute.
- Here are the types used in this chapter:
- Type Description text Defines normal text input radio Defines radio button input (for selecting one of many choices) submit Defines a submit button (for submitting the form)

HTML5 Input Types

- HTML5 added several new input types:
- color
- date
- datetime
- datetime-local
- email
- month
- number
- range
- search
- tel
- time
- url
- week

Text Input

- **<input type="text">** defines a one-line input field for **text input**:
- **Example**
- <form action="action_page.php">
- <label for="firstname"> First name :</label>
  <input type="text" value="Mickey" Id ="Firstname" >
  Last name:
  <input type="text" name="lastname" value="Mouse">

  <input type="submit" value="Submit">
  </form>

Input Type: password

- **<input type="password">** defines a **password field**:
- **Example**
- <form>
  User name:
  <input type="text" name="username">
  User password:
  <input type="password" name="psw">
</form>

Input Type: number

- The **<input type="number">** is used for input fields that should contain a numeric value.
- You can set restrictions on the numbers.
- Depending on browser support, the restrictions can apply to the input field.
- **Example**
- <form>
  Quantity (between 1 and 5):
  <input type="number" name="quantity" min="1" max="5">
  </form>

Input Type: date

- The **<input type="date">** is used for input fields that should contain a date.
- Depending on browser support, a date picker can show up in the input field.
- **Example**
- <form>
  Birthday:
  <input type="date" name="bday">
  </form>

Input Type: range

- The **<input type="range">** is used for input fields that should contain a value within a range.
- Depending on browser support, the input field can be displayed as a slider control.
- **Example**
- <form>
  <input type="range" name="points" min="0" max="10">
  </form>

Radio Button Input

- **<input type="radio">** defines a **radio button**.
- Radio buttons let a user select ONE of a limited number of choices:
- **Example**
- <form>
  <input type="radio" name="gender" value="male" checked> Male
  <input type="radio" name="gender" value="female"> Female
  <input type="radio" name="gender" value="other"> Other
  </form>

Input Type: checkbox

- **<input type="checkbox">** defines a **checkbox**.
- Checkboxes let a user select ZERO or MORE options of a limited number of choices.
- **Example**
- <form>
  <input type="checkbox" name="vehicle1" value="Bike"> I have a bike
  <input type="checkbox" name="vehicle2" value="Car"> I have a car
  </form>

Input Type: email

- The **<input type="email">** is used for input fields that should contain an e-mail address.
- Depending on browser support, the e-mail address can be automatically validated when submitted.
- Some smartphones recognize the email type, and adds ".com" to the keyboard to match email input.
- **Example**
- <form>
  E-mail:
  <input type="email" name="email">
  </form>

The Submit Button

- **<input type="submit">** defines a button for **submitting** a form to a **form-handler**.
- The form-handler is typically a server page with a script for processing input data.
- The form-handler is specified in the form's **action** attribute:
- **Example**
- <form action="action_page.php">
  First name:
  <input type="text" name="firstname" value="Mickey">
  Last name:
  <input type="text" name="lastname"
value="Mouse">

  <input type="submit" value="Submit">
</form>

The Action Attribute

- The **action attribute** defines the action to be performed when the form is submitted.
- The common way to submit a form to a server, is by using a submit button.
- Normally, the form is submitted to a web page on a web server.
- In the example above, a server-side script is specified to handle the submitted form:
- <form **action="action_page.php**">

- If the action attribute is omitted, the action is set to the current page.

The Method Attribute

- The **method attribute** specifies the HTTP method (**GET** or **POST**) to be used when submitting the forms:
- <form action="action_page.php" **method="get"**>
- or:
- <form action="action_page.php" **method="post"**>

When to Use GET?

- You can use GET (the default method):
- If the form submission is passive (like a search engine query), and without sensitive information.
- When you use GET, the form data will be visible in the page address:
- action_page.php?firstname=Mickey&lastname=Mouse
- GET is best suited to short amounts of data. Size limitations are set in your browser.

Grouping Form Data with <fieldset>

- The **<fieldset>** element groups related data in a form.
- The **<legend>** element defines a caption for the <fieldset> element.
- **Example**
- <form action="action_page.php">
  <fieldset>
  <legend>Personal information:</legend>
  First name:
  <input type="text" name="firstname" value="Mickey">
  Last name:
  <input type="text" name="lastname" value="Mouse">

  <input type="submit" value="Submit">
  </fieldset>
  </form>

The <select> Element (Drop-Down List)

- The **<select>** element defines a **drop-down** list:
- **Example**
- <select name="cars">
  <option value="volvo“ selected >Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
  </select>

The <textarea> Element

- The **<textarea>** element defines a multi-line input field (**a text area**):
- **Example**
- <textarea name="message" rows="10" cols="30">
  The cat was playing in the garden.
  </textarea>

The <button> Element

- The **<button>** element defines a clickable **button**:
- **Example**
- <button type="button" onclick="alert('Hello World!')">Click Me!</button>

HTML5 <input>: Attributes

- HTML5 added the following attributes for <input>:
- autocomplete
- autofocus
- form
- formaction
- formenctype
- formmethod
- formnovalidate
- formtarget
- height and width
- list
- min and max
- multiple
- pattern (regexp)
- placeholder
- required
- step
- and the following attributes for <form>:
- autocomplete
- novalidate

The required Attribute

- The required attribute is a boolean attribute.
- When present, it specifies that an input field must be filled out before submitting the form.
- The required attribute works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.
- **Example**
- A required input field:
- Username: <input type="text" name="usrname" required>

The autofocus Attribute

- The autofocus attribute is a boolean attribute.
- When present, it specifies that an <input> element should automatically get focus when the page loads.
- **Example**
- Let the "First name" input field automatically get focus when the page loads:
- First name:<input type="text" name="fname" autofocus>

The pattern Attribute

- The pattern attribute specifies a regular expression that the <input> element's value is checked against.
- The pattern attribute works with the following input types: text, search, url, tel, email, and password.
- **Tip:** Use the global title attribute to describe the pattern to help the user.
- **Tip:** Learn more about regular expressions in our JavaScript tutorial.
- **Example**
- An input field that can contain only three letters (no numbers or special characters):
- Country code: <input type="text" name="country_code" pattern="[A-Za-z]{3}" title="Three letter country code">

The placeholder Attribute

- The placeholder attribute specifies a hint that describes the expected value of an input field (a sample value or a short description of the format).
- The hint is displayed in the input field before the user enters a value.
- The placeholder attribute works with the following input types: text, search, url, tel, email, and password.
- **Example**
- An input field with a placeholder text:
- <input type="text" name="fname" placeholder="First name">

The HTML <video> Element

- To show a video in HTML, use the **<video>** element:
- **Example**
- <video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
  Your browser does not support the video tag.
  </video>

The HTML <audio> Element

- To play an audio file in HTML, use the **<audio>** element:
- **Example**
- <audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
  Your browser does not support the audio element.
  </audio>

Iframe Syntax

- The syntax for adding an iframe is:
- <iframe src="*URL*"></iframe>
- The **src** attribute specifies the URL (web address) of the iframe page.

Iframe - Set Height and Width

- Use the **height** and **width** attributes to specify the size.
- The attribute values are specified in pixels by default, but they can also be in percent (like "80%").
- **Example**
- <iframe src="demo_iframe.htm" width="200" height="200"></iframe>

Iframe - Remove the Border

- By default, an iframe has a black border around it.
- To remove the border, add the style attribute and use the CSS border property:
- **Example**
- <iframe src="demo_iframe.htm" style="border:none"></iframe>
- With CSS, you can also change the size, style and color of the iframe's border:
- **Example**
- <iframe src="demo_iframe.htm" style="border:5px dotted red"></iframe>

Use iframe as a Target for a Link

- An iframe can be used as the target frame for a link.
- The **target** attribute of the link must refer to the **name** attribute of the iframe:
- **Example**
- <iframe src="demo_iframe.htm" name="iframe_a"></iframe>
  W3Schools.com

HTML Layout - Using Tables

- The simplest and most popular way of creating layouts is using HTML <table> tag. These tables are arranged in columns and rows, so you can utilize these rows and columns in whatever way you like.

Main Title of Web Page	
Menu HTML CSS JavaScript	Content goes here
Copyright © 2011 W3Schools.com	

More on Tables

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Layout using Tables</title>
</head>
<body>
<table width="100%" border="0">
 <tr>
  <td colspan="2" bgcolor="#b5dcb3">
   <h1>This is Web Page Main title</h1>
  </td>
 </tr>
 <tr valign="top">
  <td bgcolor="#aaa" width="50">
   <b>Main Menu</b><br />
   HTML<br />
   PHP<br />
   PERL...
  </td>
  <td bgcolor="#eee" width="100" height="200">
    Technical and Managerial Tutorials
  </td>
 </tr>
 <tr>
  <td colspan="2" bgcolor="#b5dcb3">
   <center>
   Copyright © 2007 Tutorialspoint.com
   </center>
  </td>
 </tr>
</table>
</body>
</html>
```


1300px
800px
400px
300px

HTML Layouts - Using DIV

- The element is a block level element used for grouping HTML elements. While the tag is a block-level element, the HTML element is used for grouping elements at an inline level.
- Although we can achieve pretty nice layouts with HTML tables, but tables weren't really designed as a layout tool. Tables are more suited to presenting tabular data.

```
<div style="width:100%">
    <div style="background-color:#b5dcb3;
  width:100%">
        <h1>This is Web Page Main title</h1>
    </div>
```

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Layouts using DIV, SPAN</title>
</head>
<body>
<div style="width:100%">
  <div style="background-color:#b5dcb3; width:100%">
     <h1>This is Web Page Main title</h1>
  </div>
  <div style="background-color:#aaa; height:200px;width:100px;float:left;">
     <div><b>Main Menu</b></div>
     HTML<br />
     PHP<br />
     PERL...
  </div>
  <div style="background-color:#eee; height:200px;width:350px;float:left;">
    <p>Technical and Managerial Tutorials</p>
  </div>
  <div style="background-color:#aaa; height:200px;width:100px;float:right;">
     <div><b>Right Menu</b></div>
     HTML<br />
     PHP<br />
     PERL...
  </div>
  <div style="background-color:#b5dcb3;clear:both">
  <center>
     Copyright © 2007 Tutorialspoint.com
  </center>
  </div>
</div>
</body>
</html>
```

Website Layout Using HTML5

Website Layout Using HTML5

- HTML5 offers new semantic elements that define different parts of a web page:
- **<header>** - Defines a header for a document or a section
- **<nav>** - Defines a container for navigation links
- **<section>** - Defines a section in a document
- **<article>** - Defines an independent self-contained article
- **<aside>** - Defines content aside from the content (like a sidebar)
- **<footer>** - Defines a footer for a document or a section

```
<!DOCTYPE html>
<html>
<head>
<style>
header {
    background-color:black;
    color:white;
    text-align:center;
    padding:5px;
}
nav {
    line-height:30px;
    background-color:#eeeeee;
    height:300px;
    width:100px;
    float:left;
    padding:5px;
}
section {
    width:350px;
    float:left;
    padding:10px;
}
footer {
    background-color:black;
    color:white;
    clear:both;
    text-align:center;
    padding:5px;
}
</style>
</head>
```

```
<body>

<header>
<h1>City Gallery</h1>
</header>

<nav>
London<br>
Paris<br>
Tokyo
</nav>

<section>
<h1>London</h1>
<p>London is the capital city of England. It is the
 most populous city in the United Kingdom,
with a metropolitan area of over 13 million inhabitants.</p>
<p>Standing on the River Thames, London has been
a major settlement for two millennia,
its history going back to its founding by the Romans, w
ho named it Londinium.</p>
</section>

<footer>
Copyright © W3Schools.com
</footer>

</body>
</html>
```

What is CSS?

- **C**ascading **S**tyle **S**heets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages **presentable.**
- it provides a powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

CSS – Syntax

- A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A style rule is made of three parts:

  Selector: **A**n HTML tag at which a style will be applied.
  Property: attribute of HTML tag.
  Value: Values are assigned to properties.

- CSS Style Rule Syntax:

  selector { property: value }

Example: table{ border : 1px solid #C00; }

define selectors

- Selectors can be defined in various ways

1. **The Type Selectors**

   h1 {
   color: #36CFFF;
   }

2. **The Class Selectors – using class attribute**

   . black {
   color: #000000;
   }
   The color effect of this will be applied to all tags with black class.

…. define selectors

- to make it a bit more particular

e.g.

```
h1. black {
   color: #000000;
   }
```

This rule renders the content in black for only <h1> elements with class attribute set to *black*.

– You can apply more than one class selectors to a given element. Consider the following example: <p class="center bold">

…. define selectors

3. The ID Selectors - You can define style rules based on the *id* attribute.

```
#black {
   color: #000000;
   }
```

This rule renders the content in black for every element with *id* attribute set to *black.*

- can make it a bit more particular like below

```
h1#black {
            color: #000000;
          }
```

…. define selectors

4. The Child Selectors

```
body > p {
   color: #000000;
   }
```

This rule will render all the paragraphs in black if they are a direct child of the <body> element. Other paragraphs put inside other elements like or <td> would not have any effect of this rule.

CSS – Inclusion

There are four ways to associate styles with your HTML document. Most commonly used methods are inline CSS and External CSS.

1. **Embedded CSS -The <style> Element**

You can put your CSS rules into an HTML document using the <style> element. This tag is placed inside the <head>...</head> tags.

Here is the generic syntax:

```
<head>
<style type="text/css" media="...">
        Style Rules.....
</style>
</head>
```

… CSS – Inclusion

2. Inline CSS -The *style* Attribute

Rules will be applied to that element only. Here is the generic syntax:

<element style="... style rules.... ">

Example:

<h1 style ="color:#36C; "> This is inline CSS </h1>

3. External CSS -The <link> Element

The <link> element can be used to include an external stylesheet file in your HTML document. An external style sheet is a separate text file with **.css** extension.

… CSS – Inclusion

Here is the generic syntax of including external CSS file:-

```
<head>
<link type="text/css" href="..." media="..." />
</head>
```

Example

Consider a simple style sheet file with a name *mystyle.css* having the following rules:

```
h1, h2, h3 {
   color: #36C;
   font-weight: normal;
   letter-spacing: . 4em;
   margin-bottom: 1em;
   text-transform: lowercase;
   }
```

Now you can include this file *mystyle.css* in any HTML document as follows:

```
<head>
<link type="text/css" href="mystyle.css" media="all" />
</head>
```

4. Imported CSS -@import Rule

@import is used to import an external

stylesheet in a manner similar to the <link>.

- Here is the generic syntax of @import rule.

```
<head>
<style>@import "URL";</style>
</head>
```

Example:

```
<head>
   <style>@import "mystyle.css";</style>
</head>
```

CSS Comments

You can use /* ....*/ to comment multi-line blocks

Example

```
/* This is an external style sheet file */
h1, h2, h3 {
color: #36C;
font-weight: normal;
letter-spacing: . 4em;
margin-bottom: 1em;
text-transform: lowercase;
}
/* end of style rules. */
```

CSS – Background

- You can set the following background properties of an element:
- The **background-color** property is used to set the background color of an element.
- The **background-image** property is used to set the background image of an element.
- The **background-repeat** property is used to control the repetition of an image in the background.

… CSS – Background

- The **background-position** property is used to control the position of an image in the background.
- The **background-attachment** property is used to control the scrolling of an image in the background.
- The **background** property is used as a shorthand to specify a number of other background properties.

Set the Background Color

Following is the example, which demonstrates how to set the background color for an element.

```
<p style="background-color: yellow; ">
This text has a yellow background color.
</p>
```

Set the Background Image

```
<table style="background-image:
url(/images/pattern1. gif); ">
<tr><td>
This table has background image set.
</td></tr>
</table>
```

Repeat the Background Image

- The following example demonstrates how to repeat the background image if an image is small. You can use *no-repeat* value for the *background-repeat* property if you don't want to repeat an image.
- By default, the *background-repeat* property will have a *repeat* value.

```
<table style="background-image: url(/images/pattern1.
   gif); background-repeat: repeat; ">
   <tr><td>
   This table has background image which repeats
   multiple times.
   </td></tr>
   </table>
```

- The following example which demonstrates how to repeat the background image vertically.

```
<table style="background-image:
url(/images/pattern1. gif);
background-repeat: repeat-y; ">
<tr><td>
This table has background image set which
will repeat vertically.
</td></tr>
</table>
```

The following example demonstrates how to repeat the background image horizontally.

```
<table style="background-image:
 url(/images/pattern1. gif);
 background-repeat: repeat-x; ">
 <tr><td>
 This table has background image set which
 will repeat horizontally.
 </td></tr>
 </table>
```

- **Set the Background Image Position**
  The following example demonstrates how to set the background image position 100 pixels away from the left side.
  <table style="background-image: url(/images/pattern1. gif); background-position: 100px; ">
  <tr><td>
  Background image positioned 100 pixels away from the left.
</td></tr>
  </table>

Border Width

- The border-width property specifies the width of the four borders.
- The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick.
- The border-width property can have from one to four values (for the top border, right border, bottom border, and the left border).
- **Example**
- p.one {
  border-style: solid;
  border-width: 5px;
  }
  p.two {
  border-style: solid;
  border-width: medium;
  }

Border Color

- The border-color property is used to set the color of the four borders.
- The color can be set by:
- name - specify a color name, like "red"
- Hex - specify a hex value, like "#ff0000"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- transparent
- The border-color property can have from one to four values (for the top border, right border, bottom border, and the left border).
- If border-color is not set, it inherits the color of the element.
- **Example**
- p.one {
  border-style: solid;
  border-color: red;
  }

Border - Individual Sides

- From the examples above you have seen that it is possible to specify a different border for each side.
- In CSS, there is also properties for specifying each of the borders (top, right, bottom, and left):
- **Example**
- p {
  border-top-style: dotted;
  border-right-style: solid;
  border-bottom-style: dotted;
  border-left-style: solid;
  }

Border - Individual Sides

- p {
  border-style: dotted solid;
  }
- So, here is how it works:
- If the border-style property has four values:
- **border-style: dotted solid double dashed;**
  - top border is dotted
  - right border is solid
  - bottom border is double
  - left border is dashed

Border - Shorthand Property

- As you can see from the examples above, there are many properties to consider when dealing with borders.
- To shorten the code, it is also possible to specify all the individual border properties in one property.
- The border property is a shorthand property for the following individual border properties:
- border-width
- border-style (required)
- border-color
- **Example**
- p {
  border: 5px solid red;
  }

CSS Margins

- **CSS Margin Properties**
- The CSS margin properties are used to generate space around elements.
- The margin properties set the size of the white space OUTSIDE the border.
- This element has a margin of 80px.
- The CSS margin properties set the size of the white space OUTSIDE the border.
- **Note:** The margins are completely transparent - and cannot have a background color! With CSS, you have full control over the margins. There are CSS properties for setting the margin for each side of an element (top, right, bottom, and left).

---CSS Margins

- **Margin - Individual Sides**
- CSS has properties for specifying the margin for each side of an element:
- margin-top
- margin-right
- margin-bottom
- margin-left
- All the margin properties can have the following values:
- auto - the browser calculates the margin
- *length* - specifies a margin in px, pt, cm, etc.
- *%* - specifies a margin in % of the width of the containing element
- inherit - specifies that the margin should be inherited from the parent element

- **Example**
- p {
  margin-top: 100px;
  margin-bottom: 100px;
  margin-right: 150px;
  margin-left: 80px;
  }
- The following example lets the left margin be inherited from the parent element:

- **Example**
- div.container {
  border: 1px solid red;
  margin-left: 100px;
  }

  p.one {
  margin-left: inherit;
  }

Margin - Shorthand Property

- To shorten the code, it is possible to specify all the margin properties in one property.
- The margin property is a shorthand property for the following individual margin properties:
- margin-top
- margin-right
- margin-bottom
- margin-left
- **Example**
- p {
  margin: 100px 150px 100px 80px;
  }

CSS Padding Properties

- The CSS padding properties are used to generate space around content.
- The padding properties set the size of the white space between the element content and the element border.

Padding - Individual Sides

- CSS has properties for specifying the padding for each side of an element:
- padding-top
- padding-right
- padding-bottom
- padding-left
- All the padding properties can have the following values:
- *length* - specifies a padding in px, pt, cm, etc.
- *%* - specifies a padding in % of the width of the containing element
- inherit - specifies that the padding should be inherited from the parent element

- p {
  padding-top: 50px;
  padding-right: 30px;
  padding-bottom: 50px;
  padding-left: 80px;
  }

- p {
  padding: 50px 30px 50px 80px;
  }

CSS Text

- **Text Color**
- **The color** property is used to set the color of the text.
- h1 {
  color: green;
  }
- **Text Alignment**
- The text-align property is used to set the horizontal alignment of a text.
- A text can be left or right aligned, centered, or justified. h1 {
  text-align: center;
  }
  h2 {
  text-align: left;
  }

- **Text Decoration**
- The text-decoration property is used to set or remove decorations from text.
- The value text-decoration: none; is often used to remove underlines from links:
- **Example**
- ```
  a {
    text-decoration: none;
  }
  ```
- ```
  h1 {
    text-decoration: overline;
  }

  h2 {
    text-decoration: line-through;
  }

  h3 {
    text-decoration: underline;
  }
  ```

Text Indentation

- The text-indent property is used to specify the indentation of the first line of a text:
- **Example**
- ```
  p {
      text-indent: 50px;
  }
  ```
- **Letter Spacing**
- The letter-spacing property is used to specify the space between the characters in a text.
- ```
  h1 {
      letter-spacing: 3px;
  }
  ```
- **Line Height**
- The line-height property is used to specify the space between lines:

```
p.small {
      line-height: 0.8;
  }
```

- **Word Spacing**
- The word-spacing property is used to specify the space between the words in a text.
- h1 {
  word-spacing: 10px;
  }

CSS Fonts

- The CSS font properties define the font family, boldness, size, and the style of a text.
- The font family of a text is set with the font-family property.
- The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font, and so on.
- p {
  font-family: "Times New Roman", Times, serif;
  }

Font Style

- The font-style property is mostly used to specify italic text.
- This property has three values:
- normal - The text is shown normally
- italic - The text is shown in italics
- oblique - The text is "leaning" (oblique is very similar to italic, but less supported)
- **Example**
- p.normal {
  font-style: normal;
  }

Font Size

- The font-size property sets the size of the text.
- Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs.
- Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs.
- The font-size value can be an absolute, or relative size.
- Absolute size:
- Sets the text to a specified size
- Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
- Absolute size is useful when the physical size of the output is known
- Relative size:
- Sets the size relative to surrounding elements
- Allows a user to change the text size in browsers

- h1 {
  font-size: 40px;
  }

  h2 {
  font-size: 30px;

CSS3 Shadow Effects

- With CSS3 you can add shadow to text and to elements.
  - text-shadow
  - box-shadow
- **CSS3 Text Shadow**
- The CSS3 text-shadow property applies shadow to text.
- In its simplest use, you only specify the horizontal shadow (2px) and the vertical shadow (2px):
- Next, :add a blur effect to the shadow
- Then, :add a color to the shadow
- **Example**

- h1 {
  text-shadow: 2px 2px 5px red;
  }

CSS Links

- Links can be styled with any CSS property (e.g. color, font-family, background, etc.).
- In addition, links can be styled differently depending on what **state** they are in.
- The four links states are:
- **a:link - a normal, unvisited link**
- **a:visited - a link the user has visited**
- **a:hover - a link when the user mouses over it**
- **a:active - a link the moment it is clicked**
- When setting the style for several link states, there are some order rules:
- a:hover MUST come after a:link and a:visited
- a:active MUST come after a:hover

CSS Links

- **Example**

```
/* unvisited link */
    a:link {
       color: red;
    }
    /* visited link */
    a:visited {
       color: green;
    }
    /* mouse over link */
    a:hover {
       color: hotpink;
    }
    /* selected link */
    a:active {
       color: blue;
    }
```

An Image as The List Item Marker

- The list-style-image property specifies an image as the list item marker:
- **Example**
- ul {
  list-style-image: url('sqpurple.gif');
  }

Position The List Item Markers

- The list-style-position property specifies whether the list-item markers should appear inside or outside the content flow:
- **Example**
- ul {
  list-style-position: inside;
  }

List - Shorthand property

- The list-style property is a shorthand property. It is used to set all the list properties in one declaration:
- **Example**
- ul {
  list-style: square inside url("sqpurple.gif");
  }
- When using the shorthand property, the order of the property values are:
- list-style-type (if a list-style-image is specified, the value of this property will be displayed if the image for some reason cannot be displayed)
- list-style-position (specifies whether the list-item markers should appear inside or outside the content flow)
- list-style-image (specifies an image as the list item marker)
- If one of the property values above are missing, the default value for the missing property will be inserted, if any.

CSS Tables

Table Borders

- To specify table borders in CSS, use the border property.
- **Example**
- table, th, td {
  border: 1px solid black;
  }

Collapse Table Borders

- The border-collapse property sets whether the table borders should be collapsed into a single border

```
table {
    border-collapse: collapse;
}
```

CSS Tables

- Width and height of a table are defined by the width and height properties.
- **Example**
- ```
  table {
    width: 100%;
  }

  th {
    height: 50px;
  }
  ```
- **Horizontal Alignment**
- The text-align property sets the horizontal alignment (like left, right, or center) of the content in <th> or <td>.
- **Example**
- ```
  th {
    text-align: left;
  }
  ```

CSS Tables

- **Vertical Alignment**
- The vertical-align property sets the vertical alignment (like top, bottom, or middle) of the content in <th> or <td>.
- By default, the vertical alignment of the content in a table is middle (for both <th> and <td> elements).
- **Example**
- td {
  height: 50px;
  vertical-align: bottom;
  }

CSS Tables

Table Padding

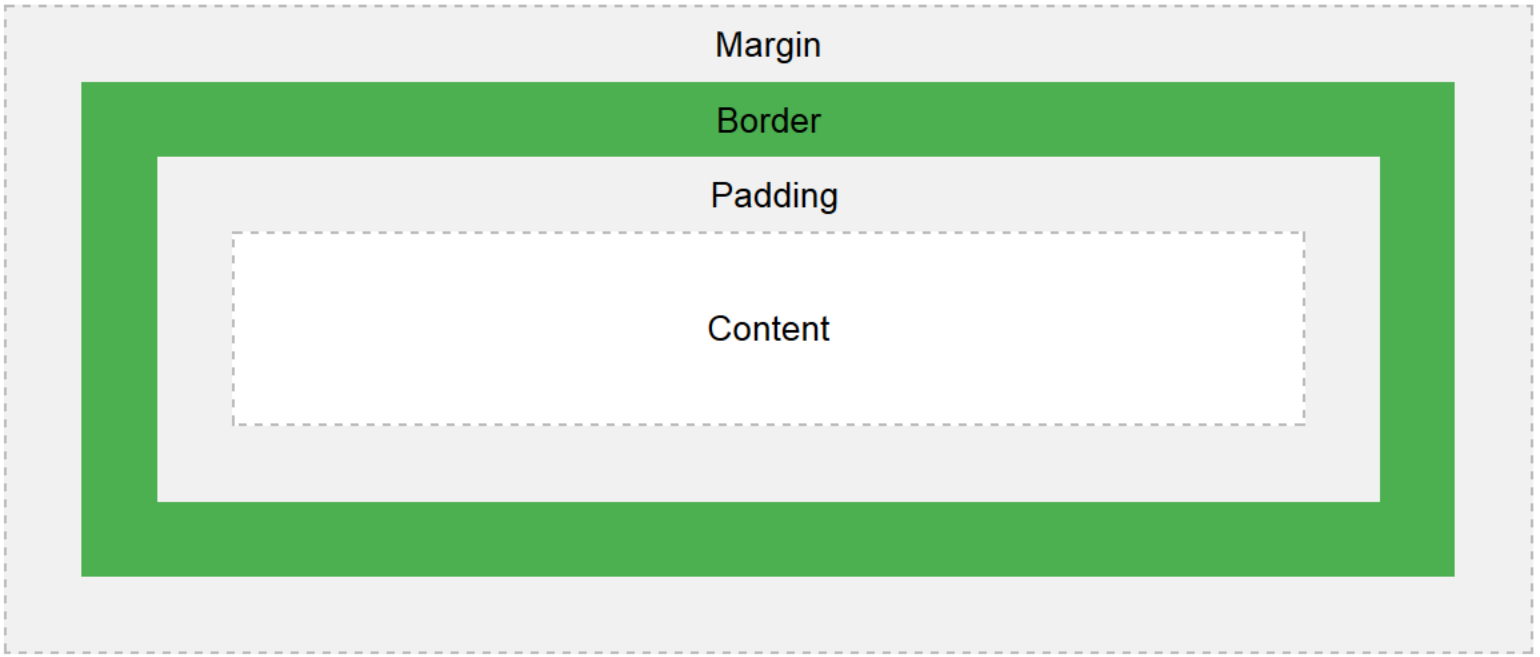
- To control the space between the border and the content in a table, use the padding property on <td> and <th> elements:
- th, td {
  padding: 15px;
  text-align: left;
  }

Horizontal Dividers

- Add the border-bottom property to <th> and <td> for horizontal dividers:
- **Example**
- th, td {
  border-bottom: 1px solid #ddd;
  }

The CSS Box Model

- All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.
- The CSS box model is essentially a box that wraps around every HTML element. It consists
  c
  c



The CSS Box Model

Explanation of the different parts:

1. **Content** - The content of the box, where text and images appear
2. **Padding** - Clears an area around the content. The padding is transparent
3. **Border** - A border that goes around the padding and content
4. **Margin** - Clears an area outside the border. The margin is transparent

- The box model allows us to add a border around elements, and to define space between elements.

- **Example**
- ```
  div {
    width: 300px;
    padding: 25px;
    border: 25px solid navy;
    margin: 25px;
  }
  ```
  ```

Width and Height of an Element

- In order to set the width and height of an element correctly in all browsers, you need to know how the box model works.
- **Important:** When you set the width and height properties of an element with CSS, you just set the width and height of the **content area**. To calculate the full size of an element, you must also add padding, borders and margins. Assume we want to style a element to have a total width of 350px:
- **Example**
- ```
  div {
    width: 320px;
    padding: 10px;
    border: 5px solid gray;
    margin: 0;
  }
  ```
  ```

Width and Height of an Element

- Here is the math:
- 320px (width)
  + 20px (left + right padding)
  + 10px (left + right border)
  + 0px (left + right margin)
  = 350px
- The total width of an element should be calculated like this:
- Total element width = width + left padding + right padding + left border + right border + left margin + right margin
- The total height of an element should be calculated like this:
- Total element height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

The display Property

- The display property specifies if/how an element is displayed.
- Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline.

Block-level Elements

- A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can). Examples of block-level elements:
- ,<h1> - <h6>,<p>,<form>,<header>,<footer>,<section>

Inline Elements

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

The position Property

- The position property specifies the type of positioning method used for an element.
- There are four different position values:

1. static
2. relative
3. fixed
4. absolute

- Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.

position: static;

- HTML elements are positioned static by default.
- Static positioned elements are not affected by the top, bottom, left, and right properties.
- An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:
- **Example**
- div.static {
  position: static;
  border: 3px solid #73AD21;
  }

position: relative;

- An element with position: relative; is positioned relative to its normal position.
- Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.
- **Example**
- div.relative {
  position: relative;
  left: 30px;
  border: 3px solid #73AD21;
  }

position: fixed;

- An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.
- A fixed element does not leave a gap in the page where it would normally have been located.
- Notice the fixed element in the lower-right corner of the page. Here is the CSS that is used:
- **Example**
- div.fixed {
  position: fixed;
  bottom: 0;
  right: 0;
  width: 300px;
  border: 3px solid #73AD21;
  }

position: absolute;

- An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).
- However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.
- **Note:** A "positioned" element is one whose position is anything except static.
- **Example**
- div.relative {
  position: relative;
  width: 400px;
  height: 200px;
  border: 3px solid #73AD21;
  }

```
div.absolute {
    position: absolute;
    top: 80px;
    right: 0;
    width: 200px;
    height: 100px;
    border: 3px solid #73AD21;
}
```

CSS Forms

Styling Input Fields

- Use the width property to determine the width of the input field:
- **Example**

```
input {
    width: 100%;
}
```

- The example above applies to all <input> elements. If you only want to style a specific input type, you can use attribute selectors:
- input[type=text] - will only select text fields
- input[type=password] - will only select password fields
- input[type=number] - will only select number fields

Padded Inputs

- Use the padding property to add space inside the text field.
- **Tip:** When you have many inputs after each other, you might also want to add some margin, to add more space outside of them:

```
input[type=text] {
    width: 100%;
    padding: 12px 20px;
    margin: 8px 0;
    box-sizing: border-box;
}
```

CSS Forms

Bordered Inputs

- Use the border property to change the border size and color, and use the border-radius property to add rounded corners:
- **Example**
- input[type=text] {
  border: 2px solid red;
  border-radius: 4px;
  }

Colored Inputs

- Use the background-color property to add a background color to the input, and the color property to change the text color:
- **Example**
- input[type=text] {
  background-color: #3CBC8D;
  color: white;
  }

HTML Lists and CSS List Properties

- In HTML, there are two main types of lists:
- unordered lists (<ul>) - the list items are marked with bullets
- ordered lists (<ol>) - the list items are marked with numbers or letters
- The CSS list properties allow you to:
- Set different list item markers for ordered lists
- Set different list item markers for unordered lists
- Set an image as the list item marker
- Add background colors to lists and list items

Different List Item Markers

- The list-style-type property specifies the type of list item marker.
- The following example shows some of the available list item markers:
- **Example**
- ul.a {
  list-style-type: circle;
  }

  ul.b {
  list-style-type: square;
  }

Example on Creating Menu using CSS

```
<!doctype html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>CSS Only Navigation Menu</title>
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <link rel="stylesheet" href="style.css">
</head>
<body>
   <ul id="menu">
      <li><a href="#">Home</a></li>
      <li>
         <a href="#">About ↓</a>
         <ul class="hidden">
```

```
 <li><a href="#">Who We Are</a></li>
            <li><a href="#">What We Do</a></li>
          </ul>
        </li>
        <li>
          <a href="#">Portfolio ↓</a>
          <ul class="hidden">
            <li><a href="#">Photography</a></li>
            <li><a href="#">Web & User Interface Design</a></li>
            <li><a href="#">Illustration</a></li>
          </ul>
        </li>
        <li><a href="#">News</a></li>
        <li><a href="#">Contact</a></li>
    </ul>
</body>
</html>
```

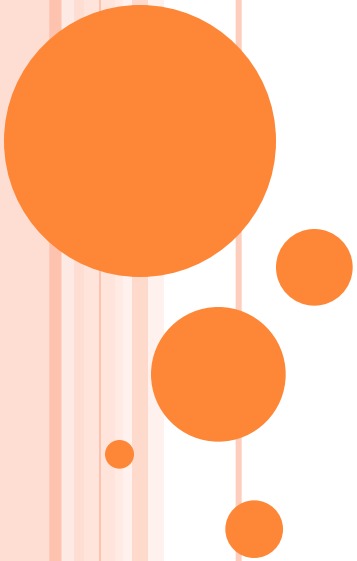
style.css

```
/*Strip the ul of padding and list styling*/
ul {
   list-style-type:none;
   margin:0;
   padding:0;
   position: absolute;
}
/*Create a horizontal list with spacing*/
li {
   display:inline-block;
   float: left;
   margin-right: 1px;
}
/*Style for menu links*/
li a {
```

```
display:block;
    min-width:140px;
    height: 50px;
    text-align: center;
    line-height: 50px;
    font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
    color: #fff;
    background: #2f3036;
    text-decoration: none;
}
/*Hover state for top level links*/
li:hover a {
    background: #19c589;
}
/*Style for dropdown links*/
li:hover ul a {
```

```
background: #f3f3f3;
    color: #2f3036;
    height: 40px;
    line-height: 40px;
}
/*Hover state for dropdown links*/
li:hover ul a:hover {
    background: #19c589;
    color: #fff;
}
/*Hide dropdown links until they are needed*/
li ul {
    display: none;
}
```

```
/*Make dropdown links vertical*/
li ul li {
    display: block;
    float: none;
}
/*Prevent text wrapping*/
li ul li a {
    width: auto;
    min-width: 100px;
    padding: 0 20px;
}
/*Display the dropdown on hover*/
ul li a:hover + .hidden, .hidden:hover {
    display: block;
}
```



Introduction to JavaScript

Why Study JavaScript?

- JavaScript is one of the **3 languages** all web developers **must** learn:
- 1. **HTML** to define the content of web pages
- 2. **CSS** to specify the layout of web pages
- 3. **JavaScript** to program the behavior of web pages

JavaScript

- JavaScript is the programming language of HTML and the Web.
- Programming makes computers do what you want them to do.
  - **Change HTML Content**
  - **Change HTML Attributes**
  - **Change HTML Styles (CSS)**
  - **Validate Data**
- JavaScript and Java are completely different languages, both in concept and design.

JavaScript Where To

- JavaScript can be placed in the <body> and the <head> sections of an HTML page.
- **The <script> Tag**
- In HTML, JavaScript code must be inserted between <script> and </script> tags.

  Example
- <script>
  document.getElementById("demo").innerHTML = "My First JavaScript";
  </script>

JavaScript in <head>

- In this example, a JavaScript function is placed in the <head> section of an HTML page.
- The function is invoked (called) when a button is clicked:
- **Example**
- <!DOCTYPE html>
  <html><head>
  <script>
  function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
  }
  </script>
  </head>
  <body>
  <h1>My Web Page</h1>
  A Paragraph
  <button type="button" onclick="myFunction()">Try it</button>
  </body>
  </html>

JavaScript in <body>

- In this example, a JavaScript function is placed in the <body> section of an HTML page.
- The function is invoked (called) when a button is clicked:
- **Example**
- <!DOCTYPE html>
  <html>
  <body>
  <h1>My Web Page</h1>
  A Paragraph
  <button type="button" onclick="myFunction()">Try it</button>
  <script>
  function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
  }
  </script>

  </body>
  </html>

- It is a good idea to place scripts at the bottom of the <body> element.
  This can improve page load, because script compilation can slow down the display.

External JavaScript

- Scripts can also be placed in external files:

myScript.js

- function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
  }
- External scripts are practical when the same code is used in many different web pages.
- JavaScript files have the file extension **.js**.
- To use an external script, put the name of the script file in the src (source) attribute of a <script> tag:
- **Example**
- <!DOCTYPE html>
  <html>
  <body>
  <script src="myScript.js"></script>
  </body>
  </html>

External JavaScript

- You can place an external script reference in <head> or <body> as you like.
- The script will behave as if it was located exactly where the <script> tag is located.
- External scripts cannot contain <script> tags.

External JavaScript Advantages

- Placing JavaScripts in external files has some advantages:
- It separates HTML and code
- It makes HTML and JavaScript easier to read and maintain
- Cached JavaScript files can speed up page loads

JavaScript Output

- JavaScript does NOT have any built-in print or display functions.

JavaScript Display Possibilities

- JavaScript can "display" data in different ways:
  - Writing into an alert box, using **window.alert()**.
  - Writing into the HTML output using **document.write()**.
  - Writing into an HTML element, using **innerHTML**.
  - Writing into the browser console, using **console.log()**.

Using window.alert()

- You can use an alert box to display data:
- **Example**
- <!DOCTYPE html>
  <html>
  <body>

  <h1>My First Web Page</h1>
  My first paragraph.

  <script>
  window.alert(5 + 6);
  </script>

  </body>
  </html>

Using document.write()

- For testing purposes, it is convenient to use **document.write()**:
- **Example**
- <!DOCTYPE html>
  <html>
  <body>
  <h1>My First Web Page</h1>
  My first paragraph.
  <script>
  document.write(5 + 6);
  </script>
  </body>
  </html>
- Using document.write() after an HTML document is fully loaded, will **delete all existing HTML**:
- <button onclick="document.write(5 + 6)">Try it</button>

Using innerHTML

- To access an HTML element, JavaScript can use the **document.getElementById(id)** method.
- The **id** attribute defines the HTML element. The **innerHTML** property defines the HTML content:
- **Example**
- <!DOCTYPE html>
  <html>
  <body>
  <h1>My First Web Page</h1>
  My First Paragraph

  <script>
  document.getElementById("demo").innerHTML = 5 + 6;
  </script>
  </body>
  </html>

Using console.log()

- In your browser, you can use the **console.log()** method to display data.
- Activate the browser console with F12, and select "Console" in the menu.
- **Example**
- <!DOCTYPE html>
  <html>
  <body>
  <h1>My First Web Page</h1>
  My first paragraph.
  <script>
  console.log(5 + 6);
  </script>
  </body>
  </html>

JavaScript Syntax

- JavaScript **syntax** is the set of rules, how JavaScript programs are constructed

JavaScript Programs.

- A **computer program** is a list of "instructions" to be "executed" by the computer.
- In a programming language, these program instructions are called **statements**.
- JavaScript is a **programming language**.
- JavaScript statements are separated by **semicolons**.
- **Example**
- var x = 5;
  var y = 6;
  var z = x + y;

JavaScript Syntax

JavaScript Statements

- JavaScript statements are composed of:
- Values, Operators, Expressions, Keywords, and Comments.

JavaScript Values

- The JavaScript syntax defines two types of values: **Fixed values** and **variable values**.
- Fixed values are called **literals**. Variable values are called **variables**.

JavaScript Literals

- The most important rules for writing fixed values are:
- **Numbers** are written with or without decimals:
- 10.50
  1001
- **Strings** are text, written within double or single quotes:
- "John Doe"
  'John Doe'

JavaScript Syntax

JavaScript Variables

- In a programming language, **variables** are used to **store** data values.
- JavaScript uses the **var** keyword to **declare** variables.
- An **equal sign** is used to **assign values** to variables.
- In this example, x is defined as a variable. Then, x is assigned (given) the value 6:
- var x;
  x = 6;

JavaScript Operators

- JavaScript uses an **assignment operator** (=) to **assign** values to variables:
- var x = 5;
  var y = 6;
- JavaScript uses **arithmetic operators** (+ - * /) to **compute** values:
- (5 + 6) * 10

JavaScript Syntax

JavaScript Expressions

- An expression is a combination of values, variables, and operators, which computes to a value.
- The computation is called an **evaluation.**
- For example, 5 * 10 evaluates to 50:
- 5 * 10
- Expressions can also contain variable values:
- x * 10
- The values can be of various types, such as numbers and strings.
- For example, "John" + " " + "Doe", evaluates to "John Doe":

JavaScript Keywords

- JavaScript **keywords** are used to identify actions to be performed.
- The **var** keyword tells the browser to create a new variable:
- var x = 5 + 6;
  var y = x * 10;

JavaScript Syntax

JavaScript Comments

- Not all JavaScript statements are "executed".
- Code after double slashes **//** or between **/*** and ***/** is treated as a **comment**.
- Comments are ignored, and will not be executed:
- var x = 5; // I will be executed

  // var x = 6; I will NOT be executed

JavaScript Character Set

- JavaScript uses the **Unicode** character set.
- Unicode covers (almost) all the characters, punctuations, and symbols in the world.

JavaScript Syntax

JavaScript Identifiers

- Identifiers are names.
- In JavaScript, identifiers are used to name variables (and keywords, and functions, and labels).
- The rules for legal names are much the same in most programming languages.
- In JavaScript, the first character must be a letter, an underscore (_), or a dollar sign ($).
- Subsequent characters may be letters, digits, underscores, or dollar signs.
- Numbers are not allowed as the first character. This way JavaScript can easily distinguish identifiers from numbers.

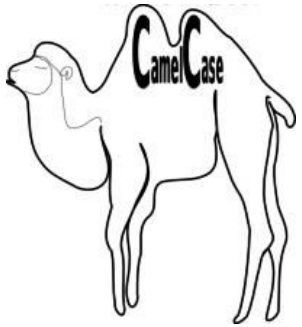
JavaScript Syntax

JavaScript is Case Sensitive

- All JavaScript identifiers are **case sensitive**.
- The variables **lastName** and **lastname**, are two different variables.
- lastName = "Doe";
  lastname = "Peterson";
- JavaScript does not interpret **VAR** or **Var** as the keyword **var**.

JavaScript Syntax

JavaScript and Camel Case



- Historically, programmers have used three ways of joining multiple words into one variable name:
- **Hyphens:**
- first-name, last-name, master-card, inter-city.
- **Underscore:**
- first_name, last_name, master_card, inter_city.
- **Camel Case:**
- FirstName, LastName, MasterCard, InterCity.
- In programming languages, especially in JavaScript, camel case often starts with a lowercase letter:
- firstName, lastName, masterCard, interCity.
- Hyphens are not allowed in JavaScript. It is reserved for subtractions.

JavaScript Statements

- In HTML, JavaScript statements are "instructions" to be "executed" by the web browser.
- This statement tells the browser to write "Hello Dolly." inside an HTML element with id="demo":
- **Example**
- document.getElementById("demo").innerHTML = "Hello Dolly.";

JavaScript Programs

- Most JavaScript programs contain many JavaScript statements.
- The statements are executed, one by one, in the same order as they are written.
- In this example x, y, and z are given values, and finally z is displayed:
- **Example**
- var x = 5;
  var y = 6;
  var z = x + y;
  document.getElementById("demo").innerHTML = z;

Example

```
<!DOCTYPE html>
<html>
<body>
<p>JavaScript code (or just JavaScript) is a list of
   JavaScript statements.</p>
<p id="demo"></p>
<script>
var x = 5;
var y = 6;
var z = x + y;
document.getElementById("demo").innerHTML = z;
</script>
</body>
</html>
```

JavaScript Statements

Semicolons ;

- Semicolons separate JavaScript statements.
- Add a semicolon at the end of each executable statement:
- a = 5;
  b = 6;
  c = a + b;
- When separated by semicolons, multiple statements on one line are allowed:
- a = 5; b = 6; c = a + b;

JavaScript White Space

- JavaScript ignores multiple spaces. You can add white space to your script to make it more readable.
- The following lines are equivalent:
- var person = "Hege";
  var person="Hege";
- A good practice is to put spaces around operators (= + - * /):
- var x = y + z;

JavaScript Statements

JavaScript Line Length and Line Breaks

- For best readability, programmers often like to avoid code lines longer than 80 characters.
- If a JavaScript statement does not fit on one line, the best place to break it, is after an operator:
- **Example**
- document.getElementById("demo").innerHTML = "Hello Dolly.";

JavaScript Statements

JavaScript Code Blocks

- JavaScript statements can be grouped together in code blocks, inside curly brackets {...}.
- The purpose of code blocks is to define statements to be executed together.
- One place you will find statements grouped together in blocks, are in JavaScript functions:
- **Example**
- function myFunction() {
  document.getElementById("demo").innerHTML = "Hello Dolly.";
  document.getElementById("myDIV").innerHTML = "How are you?";
  }

JavaScript Statements

JavaScript Keywords

- JavaScript statements often start with a **keyword** to identify the JavaScript action to be performed.
- Here is a list of some of the keywords you will learn about in this tutorial:

Keyword	Description
break	Terminates a switch or a loop
continue	Jumps out of a loop and starts at the top
debugger	Stops the execution of JavaScript, and calls (if available) the debugging function
do ... while	Executes a block of statements, and repeats the block, while a condition is true
for	Marks a block of statements to be executed, as long as a condition is true
function	Declares a function
if ... else	Marks a block of statements to be executed, depending on a condition
return	Exits a function
switch	Marks a block of statements to be executed, depending on different cases
try ... catch	Implements error handling to a block of statements
var	Declares a variable

JavaScript Comments

- JavaScript comments can be used to explain JavaScript code, and to make it more readable.
- JavaScript comments can also be used to prevent execution, when testing alternative code.

Single Line Comments

- Single line comments start with //.
- Any text between // and the end of the line, will be ignored by JavaScript (will not be executed).
- **Example**
- var x = 5; // Declare x, give it the value of 5
  var y = x + 2; // Declare y, give it the value of x + 2

JavaScript Comments

- **Multi-line Comments**
- Multi-line comments start with /* and end with */.
- Any text between /* and */ will be ignored by JavaScript.
- This example uses a multi-line comment (a comment block) to explain the code:
- /*
  The code below will change
  the heading with id = "myH"
  and the paragraph with id = "myP"
  in my web page:
  */
- Using comments to prevent execution of code, is suitable for code testing.
- Adding // in front of a code line changes the code lines from an executable line to a comment.

JavaScript Comments

- **Multi-line Comments**
- Multi-line comments start with /* and end with */.
- Any text between /* and */ will be ignored by JavaScript.
- This example uses a multi-line comment (a comment block) to explain the code:
- /*
  The code below will change
  the heading with id = "myH"
  and the paragraph with id = "myP"
  in my web page:
  */
- Using comments to prevent execution of code, is suitable for code testing.
- Adding // in front of a code line changes the code lines from an executable line to a comment.

JavaScript Variables

- JavaScript variables are containers for storing data values.
- In this example, x, y, and z, are variables:
- **Example**
- var x = 5;
  var y = 6;
  var z = x + y;
- All JavaScript **variables** must be **identified** with **unique names**.
- These unique names are called **identifiers**.

JavaScript Identifiers

- Identifiers can be short names (like x and y), or more descriptive names (age, sum, totalVolume).
- The general rules for constructing names for variables (unique identifiers) are:
- Names can contain letters, digits, underscores, and dollar signs.
- Names must begin with a letter
- Names can also begin with $ and _ (but we will not use it in this tutorial)
- Names are case sensitive (y and Y are different variables)
- Reserved words (like JavaScript keywords) cannot be used as names

The Assignment Operator

- In JavaScript, the equal sign (=) is an "assignment" operator, not an "equal to" operator.
- This is different from algebra. The following does not make sense in algebra:
- x = x + 5
- In JavaScript, however, it makes perfect sense: it assigns the value of x + 5 to x.
- (It calculates the value of x + 5 and puts the result into x. The value of x is incremented by 5.)
- The "equal to" operator is written like == in JavaScript.

JavaScript Data Types

- JavaScript variables can hold numbers like 100, and text values like "John Doe".
- In programming, text values are called text strings.
- JavaScript can handle many types of data, but for now, just think of numbers and strings.
- Strings are written inside double or single quotes. Numbers are written without quotes.
- If you put quotes around a number, it will be treated as a text string.
- **Example**
- var pi = 3.14;
  var person = "John Doe";
  var answer = 'Yes I am!';

Declaring (Creating) JavaScript Variables

- Creating a variable in JavaScript is called "declaring" a variable.
- You declare a JavaScript variable with the **var** keyword:

```
var carName;
```

- After the declaration, the variable has no value. (Technically it has the value of **undefined**)
- To **assign** a value to the variable, use the equal sign:

```
carName = "Volvo";
```

- You can also assign a value to the variable when you declare it:
- var carName = "Volvo";
- var person = "John Doe", carName = "Volvo", price = 200;

EXAMPLE

```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Variables</h1>
<p>Create a variable, assign a value to it, and display
   it:</p>
<p id="demo"></p>
<script>
var carName = "Volvo";
document.getElementById("demo").innerHTML =
   carName;
</script>
</body>
</html>
```

Value = undefined

- In computer programs, variables are often declared without a value. The value can be something that has to be calculated, or something that will be provided later, like user input.
- A variable declared without a value will have the value **undefined**.
- The variable carName will have the value undefined after the execution of this statement:
- **Example**
- var carName;

Re-Declaring JavaScript Variables

- If you re-declare a JavaScript variable, it will not lose its value.
- The variable carName will still have the value "Volvo" after the execution of these statements:
- **Example**
- var carName = "Volvo";
  var carName;

JavaScript Arithmetic

- As with algebra, you can do arithmetic with JavaScript variables, using operators like = and +:
- **Example**

  var x = 5 + 2 + 3;
- You can also add strings, but strings will be concatenated (added end-to-end):
- **Example**

  var x = "John" + " " + "Doe";

JavaScript Arithmetic Operators

- Arithmetic operators are used to perform arithmetic on numbers (literals or variables).

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement

JavaScript Assignment Operators

- Assignment operators assign values to JavaScript variables.

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

JavaScript Comparison and Logical Operators

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to

Operators and Operands

- The numbers (in an arithmetic operation) are called **operands**.
- The operation (to be performed between the two operands) is defined by an **operator**.
- Operand Operator Operand 100 + 50

JavaScript Data Types

- JavaScript variables can hold many **data types**: numbers, strings, arrays, objects and more:

```
var length = 16;                                // Number
var lastName = "Johnson";                       // String
var cars = ["Saab", "Volvo", "BMW"];            // Array
var x = {firstName:"John", lastName:"Doe"};     // Object
```

- JavaScript evaluates expressions from left to right. Different sequences can produce different results:
- var x = 16 + 4 + "Volvo";
- var x = "Volvo" + 16 + 4;

JavaScript Has Dynamic Types

- JavaScript has dynamic types. This means that the same variable can be used as different types:
- **Example**
- var x; // Now x is undefined
  var x = 5; // Now x is a Number
  var x = "John"; // Now x is a String

JavaScript Strings

- A string (or a text string) is a series of characters like "John Doe".
- Strings are written with quotes. You can use single or double quotes:

- **Example**
- var carName = "Volvo XC60"; // Using double quotes
  var carName = 'Volvo XC60'; // Using single quotes

JavaScript Numbers

- JavaScript has only one type of numbers.
- Numbers can be written with, or without decimals:

- **Example**
- var x1 = 34.00; // Written with decimals
  var x2 = 34; // Written without decimals

JavaScript Booleans

- Booleans can only have two values: true or false.
- **Example**
- var x = true;
  var y = false;

JavaScript Arrays

- JavaScript arrays are written with square brackets.
- Array items are separated by commas.
- The following code declares (creates) an array called cars, containing three items (car names):
- **Example**
- var cars = ["Saab", "Volvo", "BMW"];

- Array indexes are zero-based, which means the first item is [0], second is [1], and so on.

JavaScript Objects

- JavaScript objects are written with curly braces.
- Object properties are written as name:value pairs, separated by commas.
- **Example**
- var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
- The object (person) in the example above has 4 properties: firstName, lastName, age, and eyeColor.

JavaScript Functions

- A JavaScript function is a block of code designed to perform a particular task.
- A JavaScript function is executed when "something" invokes it (calls it).

```
function myFunction(p1, p2) {
    return p1 * p2;              // The function returns
the product of p1 and p2
}
```

```
<!DOCTYPE html>
<html>
<body>
<p>This example calls a function which performs a
   calculation, and returns the result:</p>
<p id="demo"></p>
<script>
function myFunction(a, b) {
    return a * b;
}
document.getElementById("demo").innerHTML =
   myFunction(4, 3);
</script>
</body>
</html>
```

JavaScript Function Syntax

- A JavaScript function is defined with the **function** keyword, followed by a **name**, followed by parentheses **()**.
- Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).
- The parentheses may include parameter names separated by commas:
  (*parameter1, parameter2, ...*)
- The code to be executed, by the function, is placed inside curly brackets: **{}**

```
function name(parameter1, parameter2, parameter3) {
    code to be executed
}
```

JavaScript Function Syntax

- Function **parameters** are the **names** listed in the function definition.
- Function **arguments** are the real **values** received by the function when it is invoked.
- Inside the function, the arguments behave as local variables.

```
function myFunction(p1, p2) {
    return p1 * p2;              // The function returns the
product of p1 and p2
}

document.getElementById("demo").innerHTML =
myFunction(4, 3);
```

Function Invocation

- The code inside the function will execute when "something" **invokes** (calls) the function:
  - When an event occurs (when a user clicks a button)
  - When it is invoked (called) from JavaScript code
  - Automatically (self invoked)

- <input type="button" onClick="compare()" Value="Compute" />

Function Return

- When JavaScript reaches a **return statement**, the function will stop executing.
- If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.
- Functions often compute a **return value**. The return value is "returned" back to the "caller":

Example

Calculate the product of two numbers, and return the result:

```
var x = myFunction(4, 3);        // Function is called, return value will end up in x

function myFunction(a, b) {
    return a * b;                // Function returns the product of a and b
}
```

The result in x will be:

```
12
```

Conditional Statements

- Conditional statements are used to perform different actions based on different conditions.
- Very often when you write code, you want to perform different actions for different decisions.
- You can use conditional statements in your code to do this.
- In JavaScript we have the following conditional statements:
- Use **if** to specify a block of code to be executed, if a specified condition is true
- Use **else** to specify a block of code to be executed, if the same condition is false
- Use **else if** to specify a new condition to test, if the first condition is false
- Use **switch** to specify many alternative blocks of code to be executed

The if Statement

- Use the **if** statement to specify a block of JavaScript code to be executed if a condition is true.
- Syntax

```
if (condition) {
block of code to be executed if the condition is true
 }
```

EXAMPLE

```
<!DOCTYPE html>
<html>
<body>
<p>Display “Not Allowed!" if the age is less than 18</p>
<p id=“ag"></p>
<script>
Var age=14;
if (age < 18) {
    document.getElementById(“ag").innerHTML = “Not Allowed!";
}
</script>
</body>
```

The else Statement

- Use the **else** statement to specify a block of code to be executed if the condition is false.
- **Syntax**

if (*condition*) {
    block of code to be executed if the condition is true
} else {
    block of code to be executed if the condition is false
}

EXAMPLE

```
<!DOCTYPE html>
<html>
<body>
<p>Display “Not Allowed!" if the age is less than 18 else print Allowed</p>
<p id=“ag"></p>
<script>
Var age=14;
if (age < 18) {
    document.getElementById(“ag").innerHTML = “Not Allowed!";
}
else{
    document.getElementById(“ag").innerHTML = “Allowed!";
}
</script>
</body>
```

The else if Statement

- Use the **else if** statement to specify a new condition if the first condition is false.
- **Syntax**

if (*condition1*) {
    block of code to be executed if condition1 is true
} else if (*condition2*) {
    block of code to be executed if the condition1 is false and condition2 is true
} else {
    block of code to be executed if the condition1 is false and condition2 is false
}

EXAMPLE

```
<!DOCTYPE html>
<html>
<body>
<p> Comparing Numbers</p>
<p id=“comp"></p>
<script>
Var x=14;
Var y=14;
if (x > y) {
    document.getElementById(“comp").innerHTML =x+ “is gratest";
}
else if (y> x) {
document.getElementById(“comp").innerHTML =y+ “is gratest";
}
else {
document.getElementById(“comp").innerHTML = “both are equal";
}
</script>
</body>
```

The JavaScript Switch Statement

- The switch statement is used to perform different actions based on different conditions.
- Use the switch statement to select one of many blocks of code to be executed.
- Syntax

```
switch(expression) {
    case n:
        code block
        break;
    case n:
        code block
        break;
    default:
        default code block
}
```

- This is how it works:
- The switch expression is evaluated once.
- The value of the expression is compared with the values of each case.
- If there is a match, the associated block of code is executed.

The break Keyword

- When the JavaScript code interpreter reaches a **break** keyword, it breaks out of the switch block.
- This will stop the execution of more code and case testing inside the block.

The default Keyword

- The **default** keyword specifies the code to run if there is no case match:

Example

```
<!DOCTYPE html>
<html>
<body>
<input id="myInput" type="text" value="Tutti Frutti">
<button onclick="checkFruit()">Check Fruit</button>
<p id="demo"></p>
<script>
function checkFruit() {
  var text;
  var fruits = document.getElementById("myInput").value;
  switch(fruits) {
    case "Banana":
      text = "Banana is good!";
      break;
    case "Orange":
      text = "I am not a fan of orange.";
      break;
    case "Apple":
      text = "How you like them apples?";
      break;
    // add the default keyword here
  }
  document.getElementById("demo").innerHTML = text;
}
</script>
</body>
```