INTERNET APPLICATION DEVELOPMENT

UNDERGRADUATE LECTURES

By
Jibran Rasheed Khan

PRELIMINARIES CONCEPTS **AND** TERMINOLOGIES

INTERNET

The **Internet** acronym for INTERconnected NETwork, is the global system of interconnected computer networks that uses the **Internet** protocol suite (TCP/IP) to communicate between networks or devices and publically accessible for all.

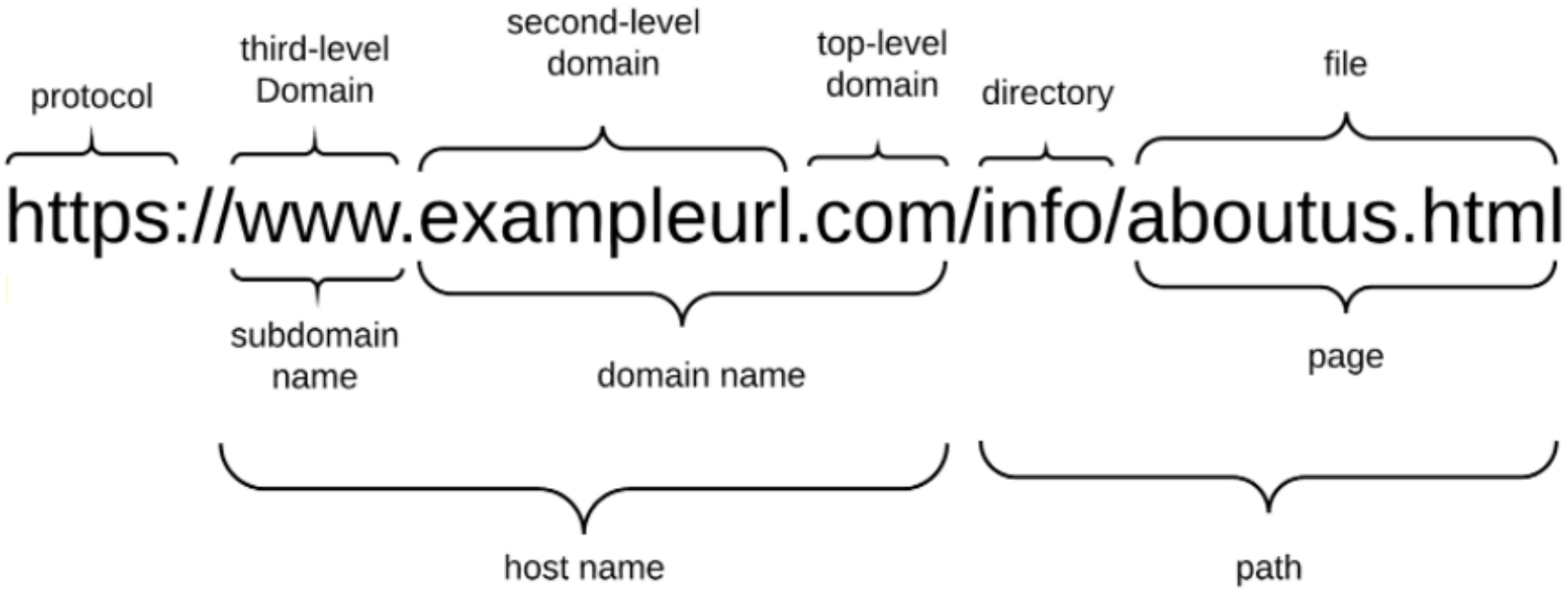
THE WEB

Web is the large transformable-information, which is publically available and accessible via internet.

- It is a service and an application of the Internet
- Web is based on internet.

UNIFORM RESOURCE LOCATORS (URL)

- Uniform Resource Locators (URL) specifies the Internet address of a file (any resource) stored on a host computer (server), connected to the Internet. URLs are translated into numeric addresses (IP Address) using a DNS.



WEB BROWSER

It is a software program that retrieves, presents, and traverses information resources on the Web. The primary function of a browser is to identify the URL and bring the information resource to user. To identify a web pages' exact location, a web browser relies on a URL. A URL, as described previously, is a four-part addressing scheme that tells the web browser:

- The transfer protocol to use for transporting the file
- The domain name of the computer on which the file resides
- The pathname of the folder or directory on the computer on which the file resides
- The name of the file

APPLICATION

Application software is a program or group of programs designed for end users to help people to perform an activity.

INTERNET APPLICATION

Internet Applications can be described as the type of **applications** which use the **internet** for operating successfully, that is, by using the **internet** for fetching, sharing and displaying the information from the respective server systems.

WEB SITE

A **website** is a single point of group of information (Related Information) globally accessible on interlinked globally accessible public network.

WEB PAGES

The **web page**(s) is a group of globally accessible, interlinked relevant pages under a single domain name.

WEB APPLICATION

A web application is a software or program which is accessible using any web browser, which are supported by major browsers.

WEB, WEBSITE AND ITS TYPE

By Nature

- Website

By Type

- Website
- Web Application
- Internet Application

By Dependency

- Partial (Website, Web App)
- Full (Internet Application)

By Operation

- Client Side (Website ,Web App)
- Server Side (Internet App)

DESKTOP APPLICATION

Any **software** application that can be installed on a single end-user device (Desktop or laptop) and used to perform specific task or activity

MOBILE APPLICATION

Any **software** application that can be installed on a single portable handed end-user devices (mobile or tablet) and used to perform specific task or activity.

WEB SERVICE

The **web services** provide a standard means of interoperating between different software applications running on a variety of platforms and frameworks.

- A web service is a software that allows different machines to interact with each other through a network.
- It is intended for computer to computer interaction.
- A web service does not necessarily have a UI since it is typically used as a component in an application
- A web service might used by a website or web app, but not compulsory.

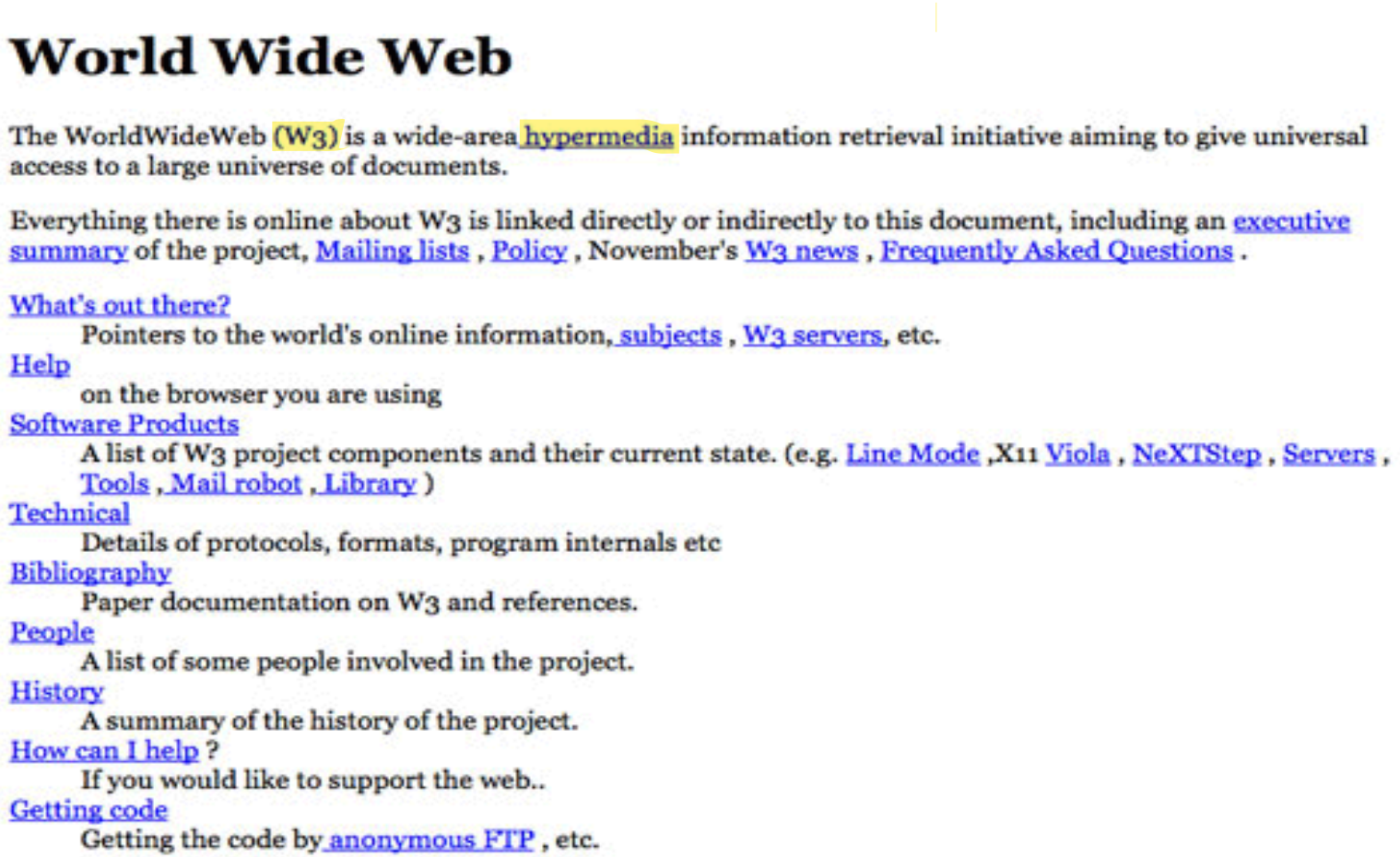
WEB TECHNOLOGY

Web technologies are tools and techniques that allow information on web and use in the process of communication between different types of devices over the internet to present, structure, and style them in an interactive manner.

THE FIRST WEB

- In August 1991, Tim Berners-Lee published the first website, a simple, text-based page with a few links. A copy from 1992 of the original page still exists online.

info.cern



World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.
Help
on the browser you are using
Software Products
A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)
Technical
Details of protocols, formats, program internals etc
Bibliography
Paper documentation on W3 and references.
People
A list of some people involved in the project.
History
A summary of the history of the project.
How can I help ?
If you would like to support the web..
Getting code
Getting the code by anonymous FTP , etc.

WEB REVOLUTION: WEB 1.0

- The initial building of the Web Based on posting printed media (books, news,…) online

Key areas

- protocols such as HTTP,
- open standard markup languages, such as HTML and XML,
- the first Web browsers,
- Web development platforms and tools,
- Web-centric software languages such as Java and Javascript,
- the creation of first Web sites,
- the commercialization of the Web and Web business models, …

WEB REVOLUTION: WEB 2.0

- The "Web 2.0" term coined and become popular after O'Reilly Media conference in 2004

Refers

- the changes in the ways people utilize the Web,
- NOT to a new wave of technology
- 2nd generation of Internet-based services that emphasize online collaboration and sharing among users

The major Web 2.0 include:

- – social networking,
- – social bookmarking,
- – media sharing (social media),
- – folksonomies,
- – lightweight collaboration (e.g., wikis),
- – mash-ups, …

WEB REVOLUTION: WEB 3.0

- A phrase coined by **John Markoff** of the New York Times in 2006
- Refers to the 3rd generation of Internet-based services that collectively comprise what might be called the **Intelligent Web**

The major factors

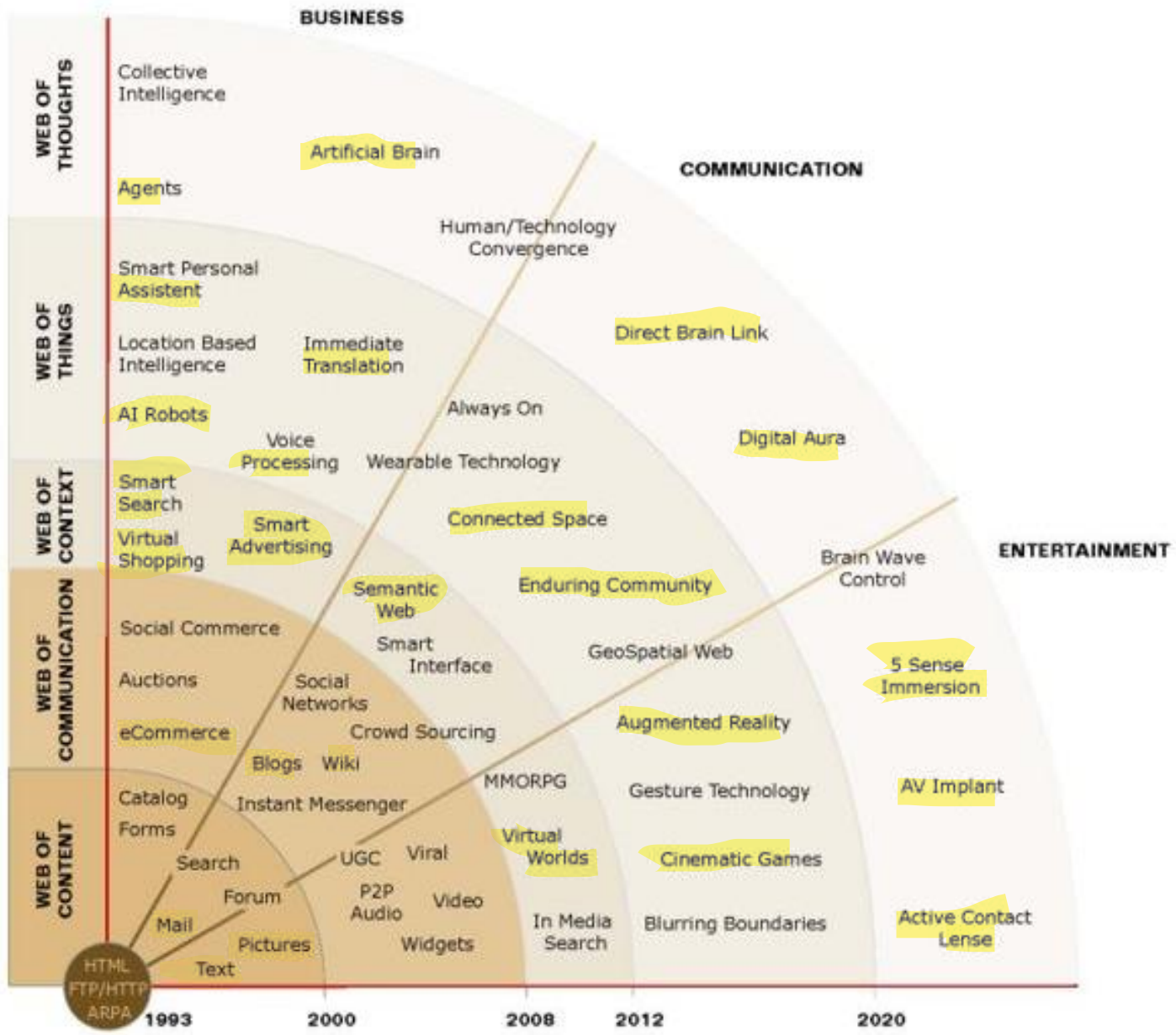
- Widespread use of AI-based technologies
  - Natural language processing,
  - Machine learning,
  - Rules-based inferences,
  - Personal agents,
  - Web mining,
- Web of Data – more effective use of the data on the Web
- Personalized Web

WEB REVOLUTION: WEB 4.0

WEB REVOLUTION: WEB 5.0

WEB REVOLUTION

	Traditional Web (Web1.0)	Social Web (Web2.0)	Semantic Web (Web3.0)
Average user	Browsing	Browsing Publishing Organizing	Browsing, Publishing, Organizing, Interacting
Communication style	One-way (e.g. reading a book)	One-way, Human-to-human (sharing)	One-way, Human-to-human, Human-to-machine (query-answering)
Data	Resources (syntactic data) - content and format mixed - documents hyperlinked	Resources, tags, metadata - content and format separated - data linked	Resources, tags, metadata - content and format separated - ontological data - data semantically linked
Data contributor	Webmaster or experienced user	Average user	Average user and web agents
Linking data	Hyperlinks	Different types of hyperlinks	Semantic links
Adding data	Composing HTML pages	Online publishing, tagging	Online publishing, tagging, machine generated data


BUSINESS
COMMUNICATION
ENTERTAINMENT
WEB OF THOUGHTS
WEB OF THINGS
WEB OF CONTEXT
WEB OF COMMUNICATION
WEB OF CONTENT
Collective Intelligence
Artificial Brain
Agents
Human/Technology Convergence
Smart Personal Assistent
Direct Brain Link
Location Based Intelligence
Immediate Translation
AI Robots
Always On
Digital Aura
Voice Processing
Wearable Technology
Smart Search
Connected Space
Smart Advertising
Virtual Shopping
Brain Wave Control
Semantic Web
Enduring Community
Social Commerce
Smart Interface
GeoSpatial Web
5 Sense Immersion
Auctions
Social Networks
Augmented Reality
eCommerce
Crowd Sourcing
Blogs
Wiki
MMORPG
Gesture Technology
AV Implant
Catalog
Instant Messenger
Forms
Virtual Worlds
Search
UGC
Viral
Cinematic Games
Forum
P2P Audio
Video
Mail
In Media Search
Blurring Boundaries
Active Contact Lense
Pictures
Widgets
Text
HTML FTP/HTTP ARPA
1993
2000
2008
2012
2020

WEB REVOLUTION SUMMARY

Era	Gen	Title	Description
1993-2000	Web 1.0	Web of Content	focused on initial building of the Web based on posting printed media (books, news,) online
2000-2008	Web 2.0	Web of Communication	focused on the advancement of the Web's ***front-end*** and the users' experience
2008-2012	Web 3.0	Web of Context	focused on (significant) improvement of the Web's ***backend***
2013-2020	Web 4.0	Web of Things	
2020-future	Web 5.0	Web of Thoughts	

COMPARISONS

- Web vs Internet
- Web vs Web Site (web node)
- Web Site vs Web Page
- Web Site vs Web App
- Web App vs Internet App
- Internet App vs Web App vs Desktop App vs Mobile app
- Web App vs Web Service
- Web Service vs Web Technology
- Web 1.0 vs 2.0 vs 3.0 vs 4.0 vs 5.0

TECHNOLOGY

Technology is the overall process of invention, innovation and diffusion of *technology* or processes.

OR

Technology is the collection of techniques, skills, methods, and processes used in the production of goods or services or in the accomplishment of objectives

PLATFORM

Platform is an environment which facilitates to build concern software application.

Or

Platform is simply the hardware or software for/on which the software is built.

Hardware Platform, includes the platform you are using to build/develop the application program such as developers machine and targeted hardware on which program will be installed on.

Software Platform, includes the software environment which uses to build/develop the application program such as developer's machine OS and IDE.

FRAMEWORK

Framework is a generic structure that provides a skeleton architecture/sketch with which specific software can be implemented.

INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)

Integrated Development Environment **(IDE)** is an application which helps you during the process of writing the code itself by automating many useful processes.

INTEGRATED DEVELOPMENT AND LEARNING ENVIRONMENT (IDLE)

Integrated Development & Learning Environment **(IDLE)** is an application which helps you during the process of writing the code itself by automating many useful processes.

APPLICATION PROGRAMMING INTERFACE (API)

API (Application Programming Interface), allows an application to interact with an external service using a simple set of commands.

To break down the name, the "Interface" is where different software components can interact.

Using an API allows developers to add specific functionalities to their applications and can speed up the development process

COMMON API ACTIONS

- GET (get a single item or a collection)
- POST (add an item to a collection)
- PUT (edit an item that already exists in a collection)
- DELETE (delete an item in a collection)

MICROSERVICES

The word "**microservices**" refers to the individual services in small part of application that is created for specific purpose and owned by one or more individual.

A **microservice architecture** is a software architecture style for modern web apps development where the application functionalities are broken up into smaller fragments called **services**.

SOFTWARE DEVELOPMENT KIT (SDK)

Software Development Kit is a collection of software development tools for a specific platform. It is set of robust features and functionalities which reduce the complexity of developing programs and applications. This can includes all sorts of things such as: Libraries, Code Samples, APIs, IDE's, Documentation, and other utilities.

PACKAGE

Package is a unit of distribution that can contain a library or module, program or any combination of them. Also, it could be a complete software application. It's a way to share your code or software with the community.

MODULE

A **Module** is a group of functions, types and classes that comes under a common namespace.

LIBRARY

Library refers to code, which provides functions that you can call from your own code to deal with common tasks

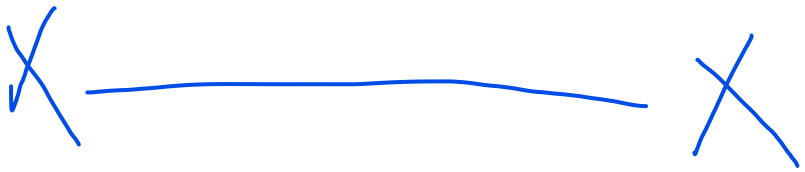
OR

A **library** is a set of functions which makes sense to be together and that can be used in a program or another library.

ASSIGNMENT

Find the at max five differences of the following.

- Library vs Module
- Module vs Framework
- Framework vs Architecture
- Package vs (Library, Module, Framework, Software)
- SDK vs IDE
- SDK vs API
- IDE vs IDLE



INTERNET APPLICATION ARCHITECTURE

The application architecture describes the interactions between applications, databases, and middleware systems on the Internet. It ensures that multiple applications work simultaneously.

TYPES OF ARCHITECTURE

- Single Tier
- Dual/Two Tier
- Three Tier
- N-Tier

ARCHITECTURE
ROLE OF BASIC LAYERS

Presentation tier

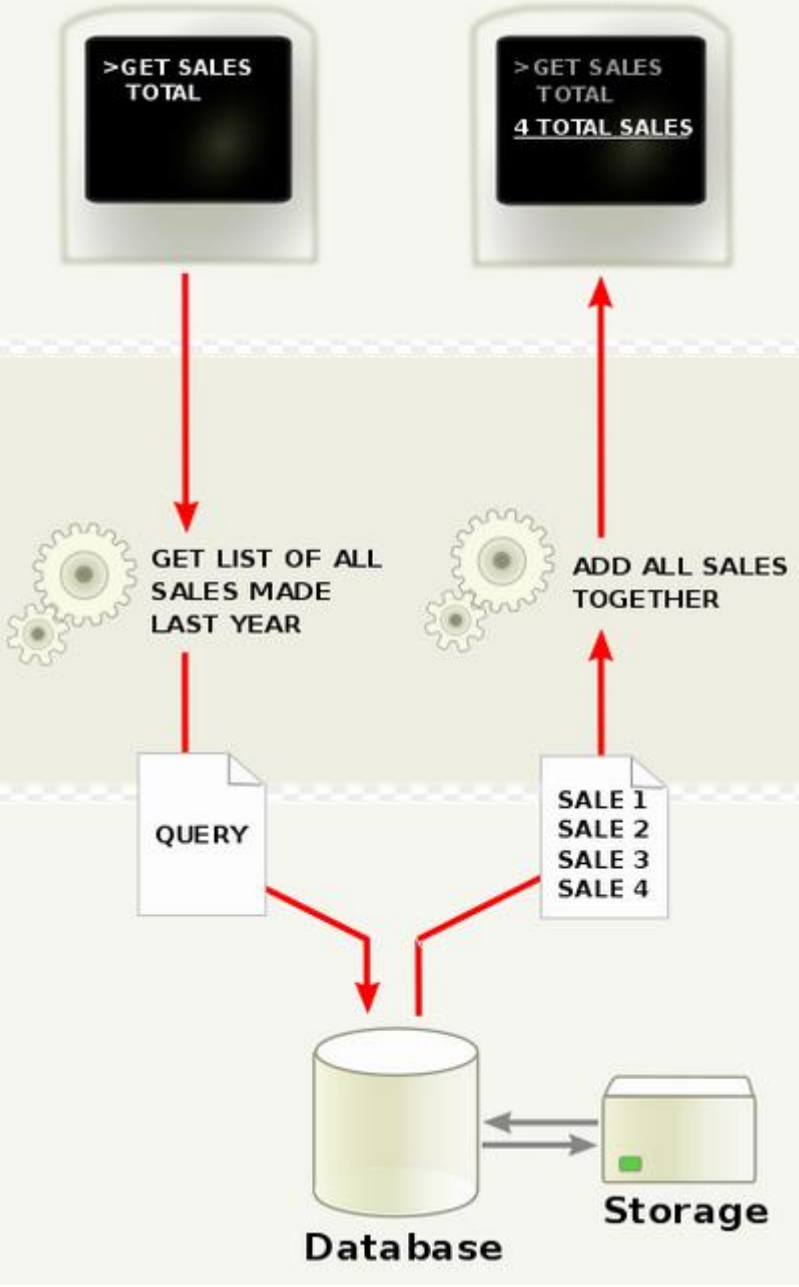
The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

Logic tier

This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

Data tier

Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.

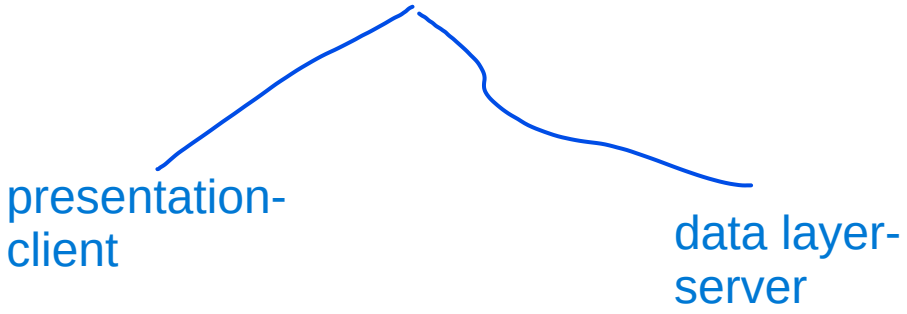


TYPES OF ARCHITECTURE
SINGLE TIER

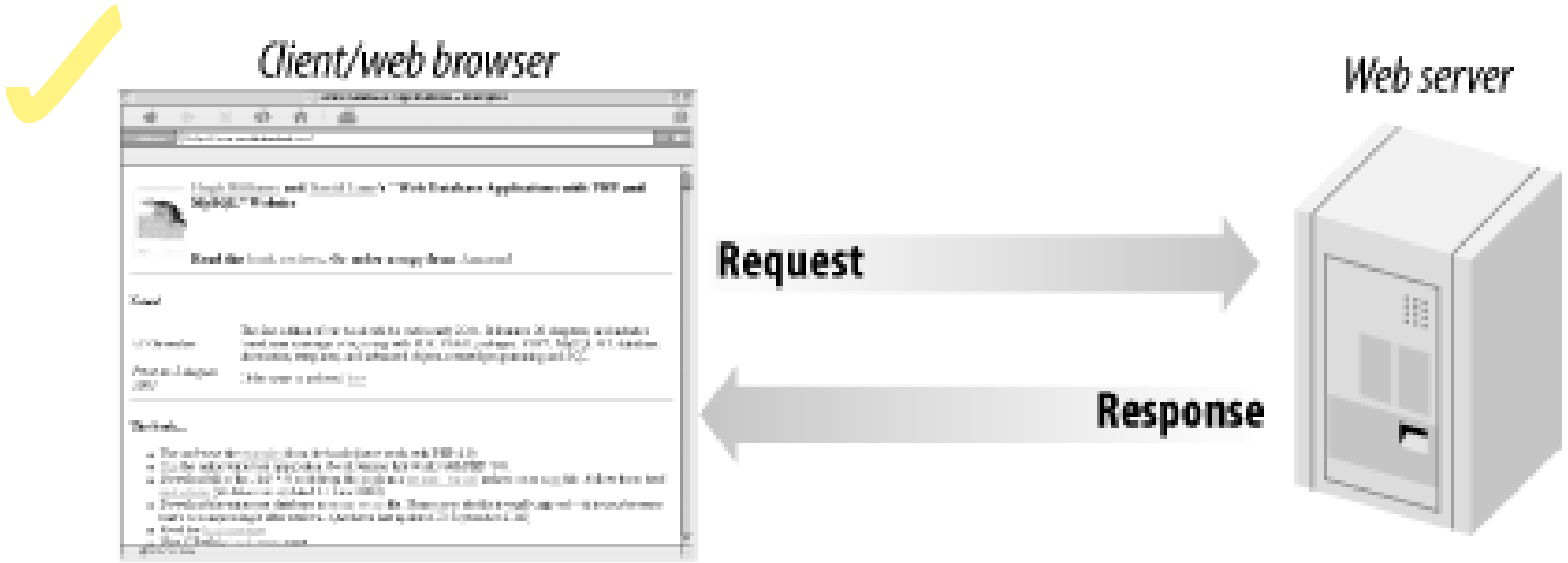
It involves putting all of the required components for a software application or technology on a **single** server or platform. It keeps all of the elements of an application, including the interface, Middleware and back-end data, in one place.

TYPES OF ARCHITECTURE
DUAL/TWO TIER

The architecture in which a **presentation** layer or interface runs on a **client**, and a data layer or data structure gets stored on a server. Separating these two components into different locations represents a two-tier architecture, as opposed to a single-tier architecture



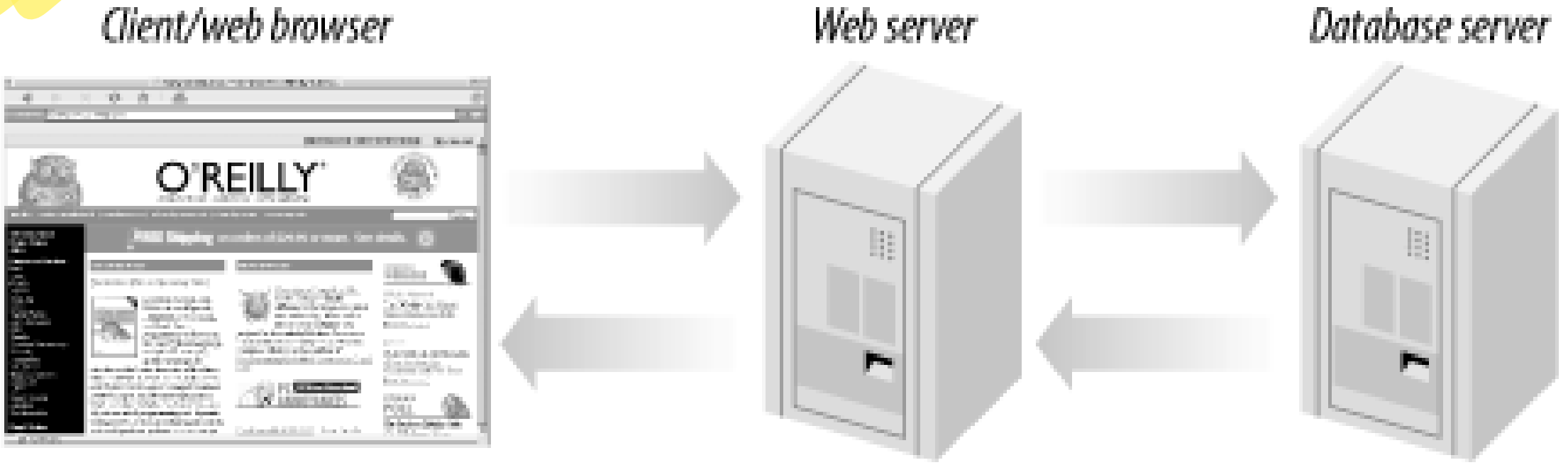
TYPES OF ARCHITECTURE
DUAL/TWO TIER



TYPES OF ARCHITECTURE
THREE TIER

Three-tier architecture is a well-established software application architecture that organizes applications into three logical and physical computing tiers: the presentation tier, or user interface; the application tier, where data is processed; and the data tier, where the data associated with the application is stored and managed.

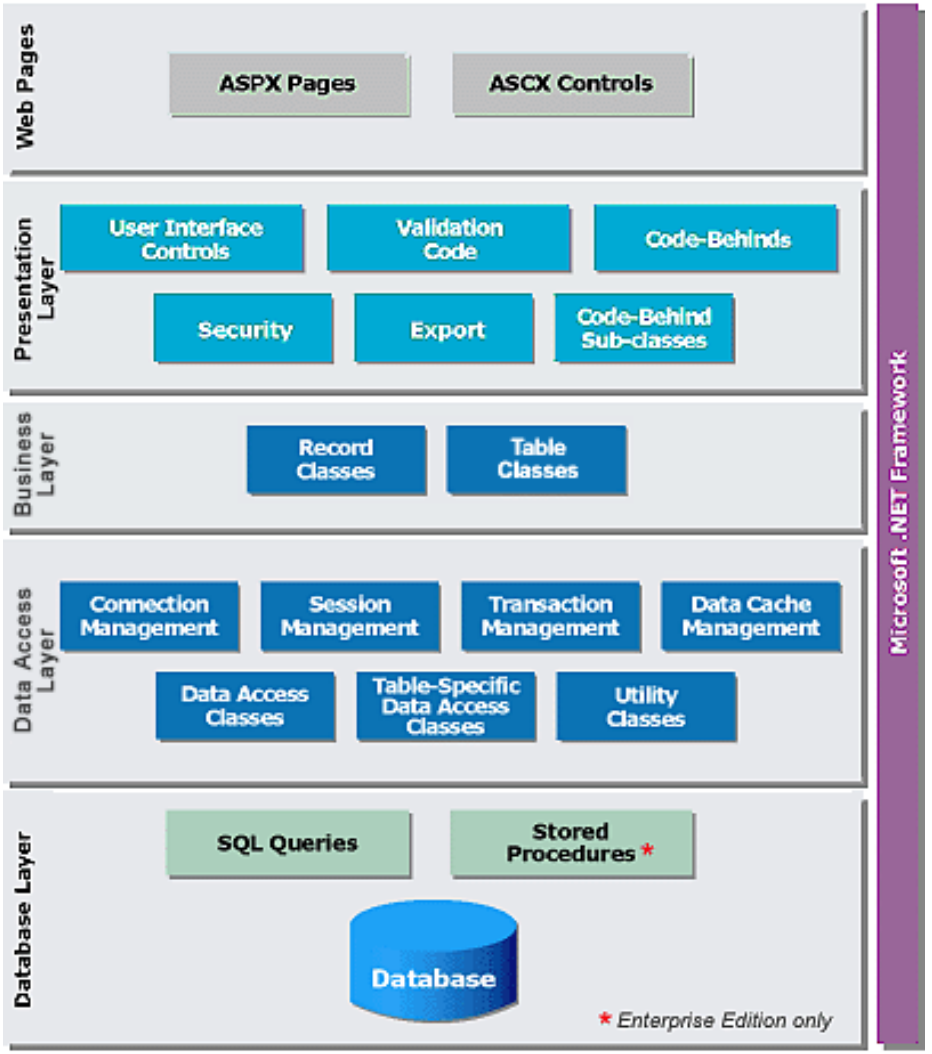
TYPES OF ARCHITECTURE
THREE TIER



TYPES OF ARCHITECTURE
MULTI-TIER

N-tier architecture is also called multi-tier architecture because the software is engineered to have the processing, data management, and presentation functions physically and logically separated.

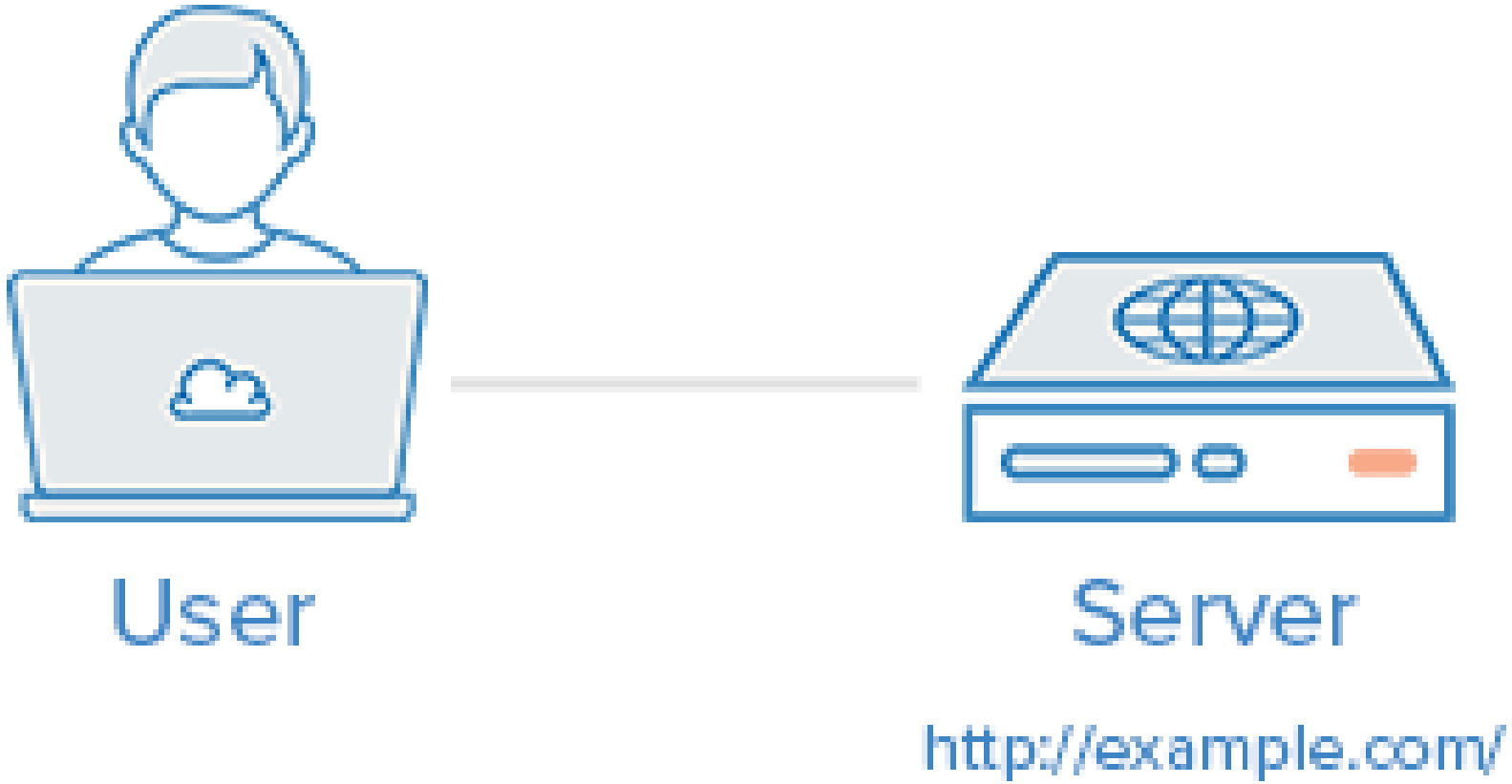
TYPES OF ARCHITECTURE
MULTI-TIER



WEB APPLICATION **DEPLOYMENT MODELS**

- Single Web and Database server
- One Web Server, One Database
- Two Web Servers, One Database (At a Machine Rather than the Web server) (Stateless)
- Two Web Server, Two Database
- Multiple Web Server, Multiple Databases

WEB DEPLOYMENT MODELS
ONE WEB & DATABASE SERVER



WEB APPLICATION MODELS
ONE WEB & DATABASE SERVER

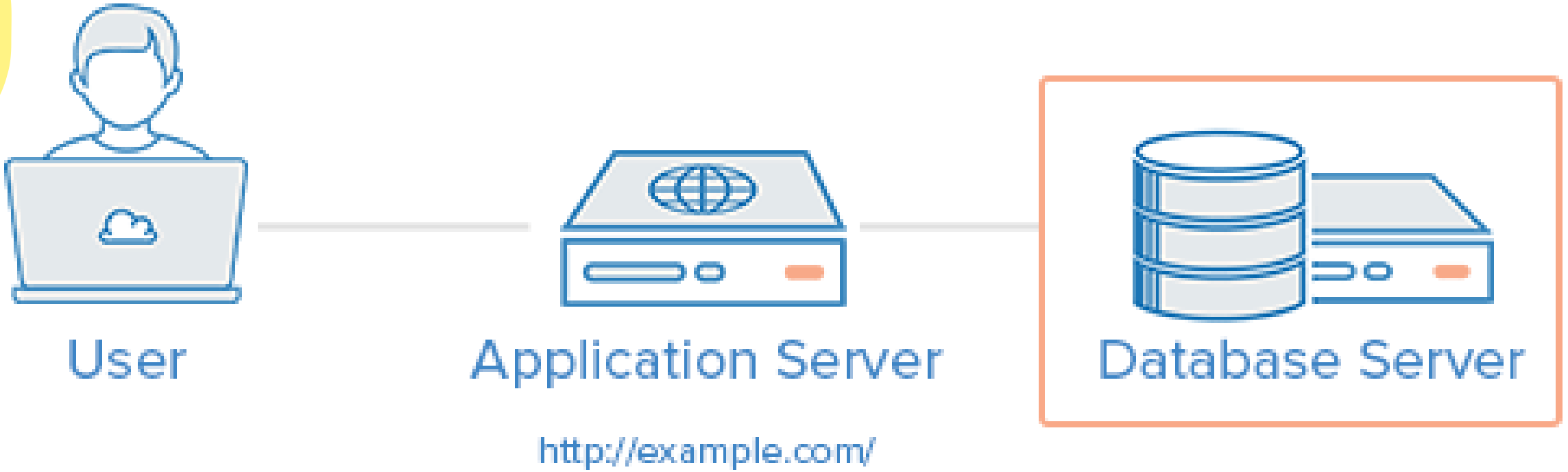
- This is the simplest and the most risky model,
- Only a single server responsible for both web and database.
- If server goes down, so does the app.
- If sever needs maintenance, so app goes down
- If server has any issue (App or database related), so app goes down.
- If application has any update or upgrade, so app goes down
- It is usually **only** suggest for using under **test project** or **personal practice**.

This model is strongly not recommended real-world scenarios,
but with respect to application need and scope

WEB APPLICATION MODELS
ONE WEB & DATABASE SERVER

- A common variation of this setup is a LAMP stack; Linux, Apache, MySQL, and PHP, on a single server
- Good for setting up an application quickly, as it is the simplest setup possible.
- Highly coupled component and the least isolation.
- It is less scalable
- It most difficult to determine the source (application or database) of poor performance.

WEB DEPLOYMENT MODELS
ONE WEB & **ONE** DATABASE SERVER

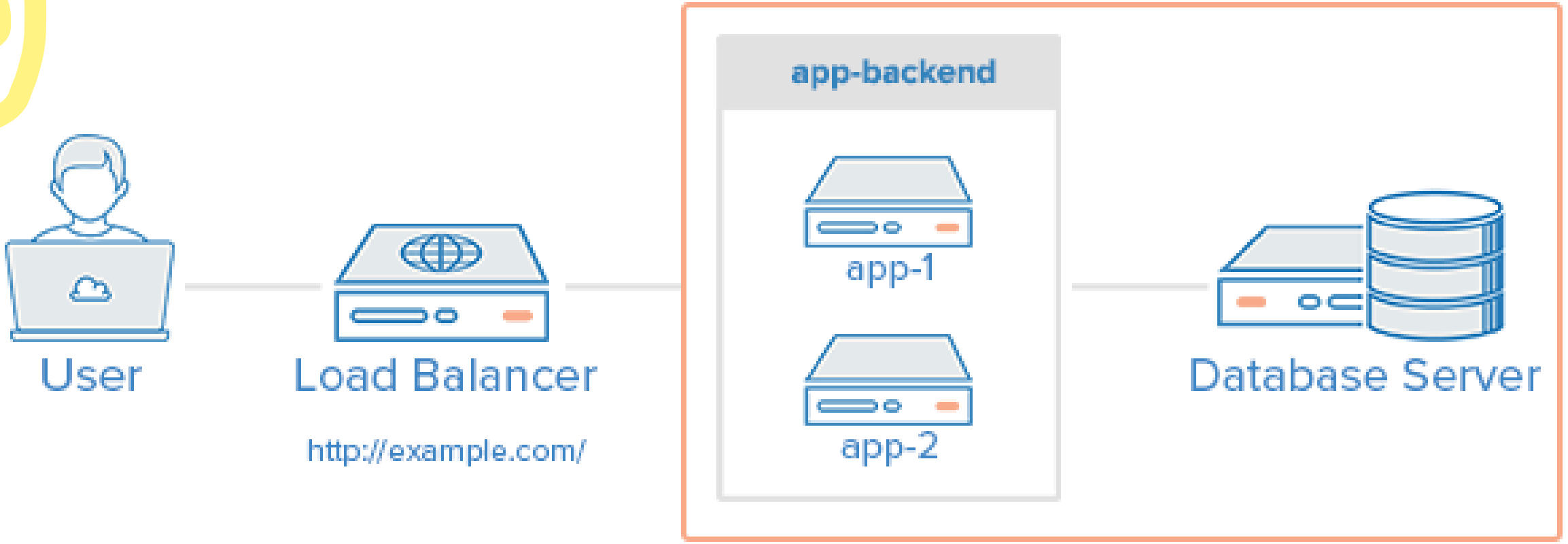


WEB DEPLOYMENT MODELS
ONE WEB & **ONE** DATABASE SERVER

- This is also the simplest and risky model, but much better than previous one
- Two servers web application and database respective responsible for their tasks or processes.
- If app server goes down, so it does not effect on database and vice versa.
- If app sever needs maintenance, so it does not disturb database, and vice versa.
- If app server has any issue, so it does not impact on database, and vice versa.
- If app has any update or upgrade, so it does not impact on database, and vice versa.
- It is usually **only** suggest for using under **small project**.

This model is not recommended real-world scenarios,
with respect to application **needs** and **scope**.

WEB DEPLOYMENT MODELS
DUAL WEB, **ONE** DATABASE SERVERS



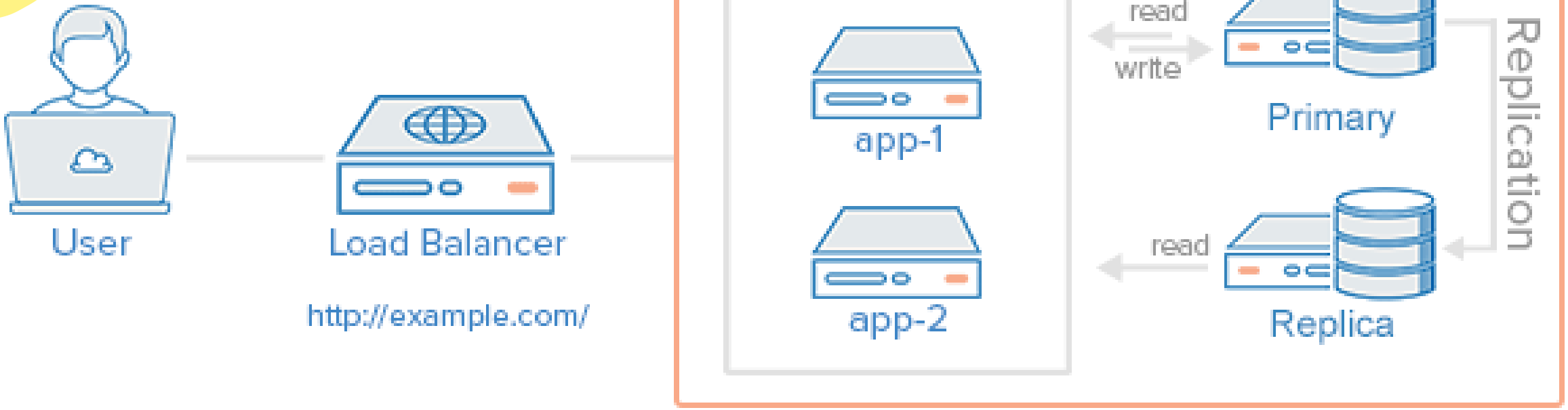
WEB APPLICATION MODELS
DUAL WEB, ONE DATABASE SERVERS

The idea is that webserver doesn't have to store any data: even when it gets information from a client, the webserver processes data, writes the data to the database (located on a physically separate machine) and forgets about it.

- With at least two web servers, you significantly reduce failure risks.
- If one of the web servers ever goes down, another one takes over immediately;
- All requests are automatically readdressed or redirected to the other server,
- Web app keeps running.
- Having only one database, still have performance risks: if it crashes, the entire system will crash as well.

This model is can be used in real-world scenarios,
with respect to application **needs** and **scope**.

WEB DEPLOYMENT MODELS
DUAL WEB, **DUAL** DATABASE SERVERS

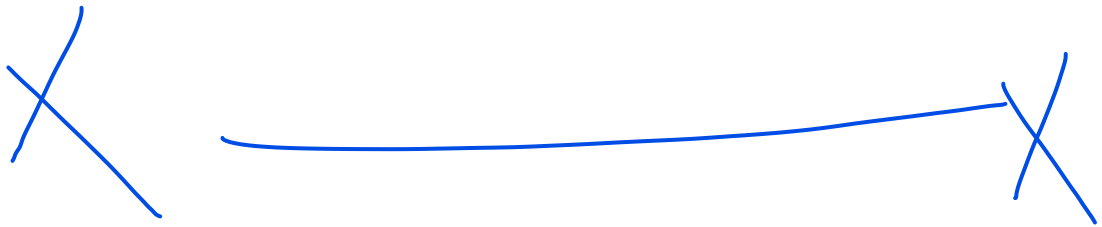


WEB DEPLOYMENT MODELS
DUAL WEB, **DUAL** DATABASE SERVERS

- This model considered to be more fail-proof: neither web servers nor databases have single points of failure.
- It usually installs **load balancers** that analyze all incoming requests and shrewdly allocate them to keep the workload under control.
- Two ways to use multi-databases options
  - **First** is to store identical data on each of your database server. Usually no more than 2 databases are needed in this case. When one is down, the other can replace it, loss-free.
  - **Second** is to evenly distribute data between your databases. The obvious advantage of storage space. But, this poses a risk of some data temporarily unavailable in the event of a database crash.
- To guarantee the **best web app performance**, commonly use as **first option** or in **a hybrid way** to replicate critical data while distributing the rest.

This model is the recommended in real-world scenarios,
with respect to application **needs** and **scope**.

WEB DEPLOYMENT MODELS
MULTI-WEB, AND **MULTI**-DATABASE SERVERS



WEB COMPONENTS

- **Structural Components –** The two major structural components of a web app are client and server sides.
  - **Client Component -** The client component is developed in CSS, HTML, and JS. As it exists within the user's web browser, there is no need for operating system or device-related adjustments. The client component is a representation of a web application's functionality that the end-user interacts with.
  - **Server Component -** The server component can be build using one or a combination of several programming languages and frameworks, including Java, .Net, NodeJS, PHP, Python, and Ruby on Rails. The server component has at least two parts; app logic and database. The former is the main control center of the web application while the latter is where all the persistent data is stored.
- **Non-Structural Component (UI/UX)–** This includes activity logs, dashboards, notifications, settings, statistics, etc. These components have nothing to do with the operation of a web application architecture. Instead, they are part of the interface layout plan of a web app.

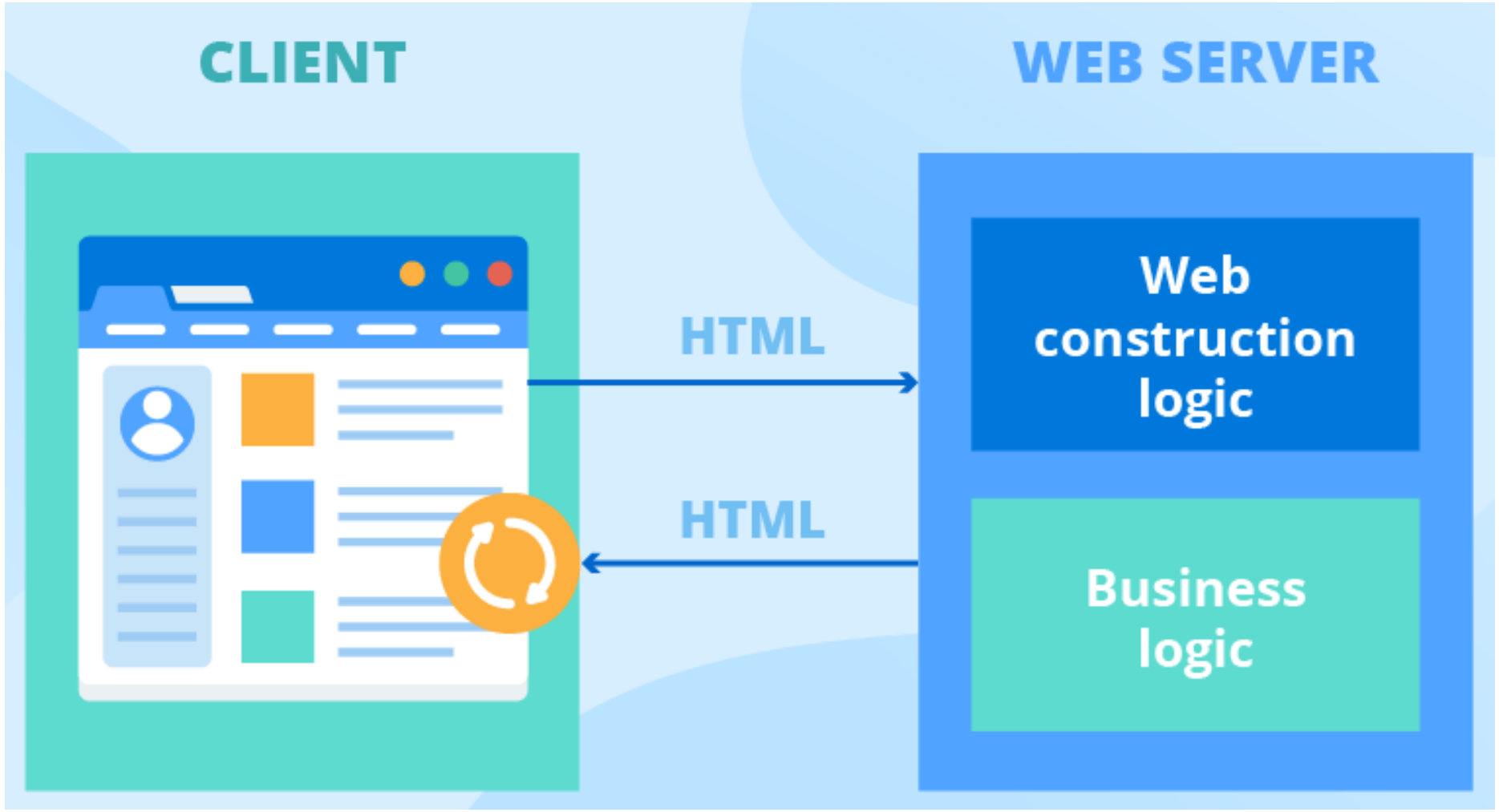
CODE DISTRIBUTION

- **Client-side Code -** The code that is in the browser and responds to some user input
- **Server-side Code -** The code that is on the server and responds to the HTTP requests

TYPES OF WEB APPLICATIONS ARCHITECTURE

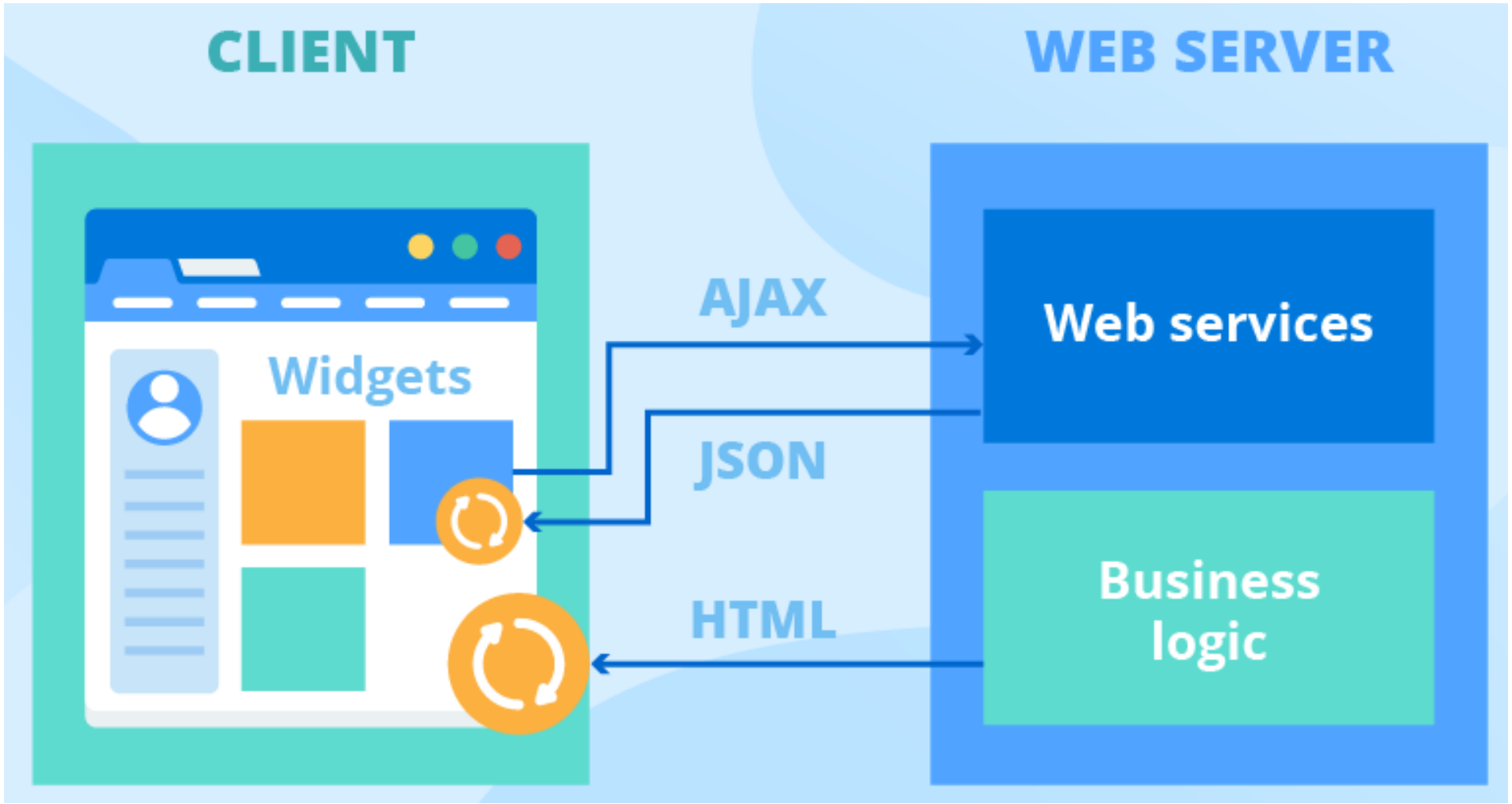
- Legacy HTML web app
- Widget web app (Js Widgets)
- Single-Page Applications (SPAs)
- Micro-services
- Monolithic
- Server-less

LEGACY WEB

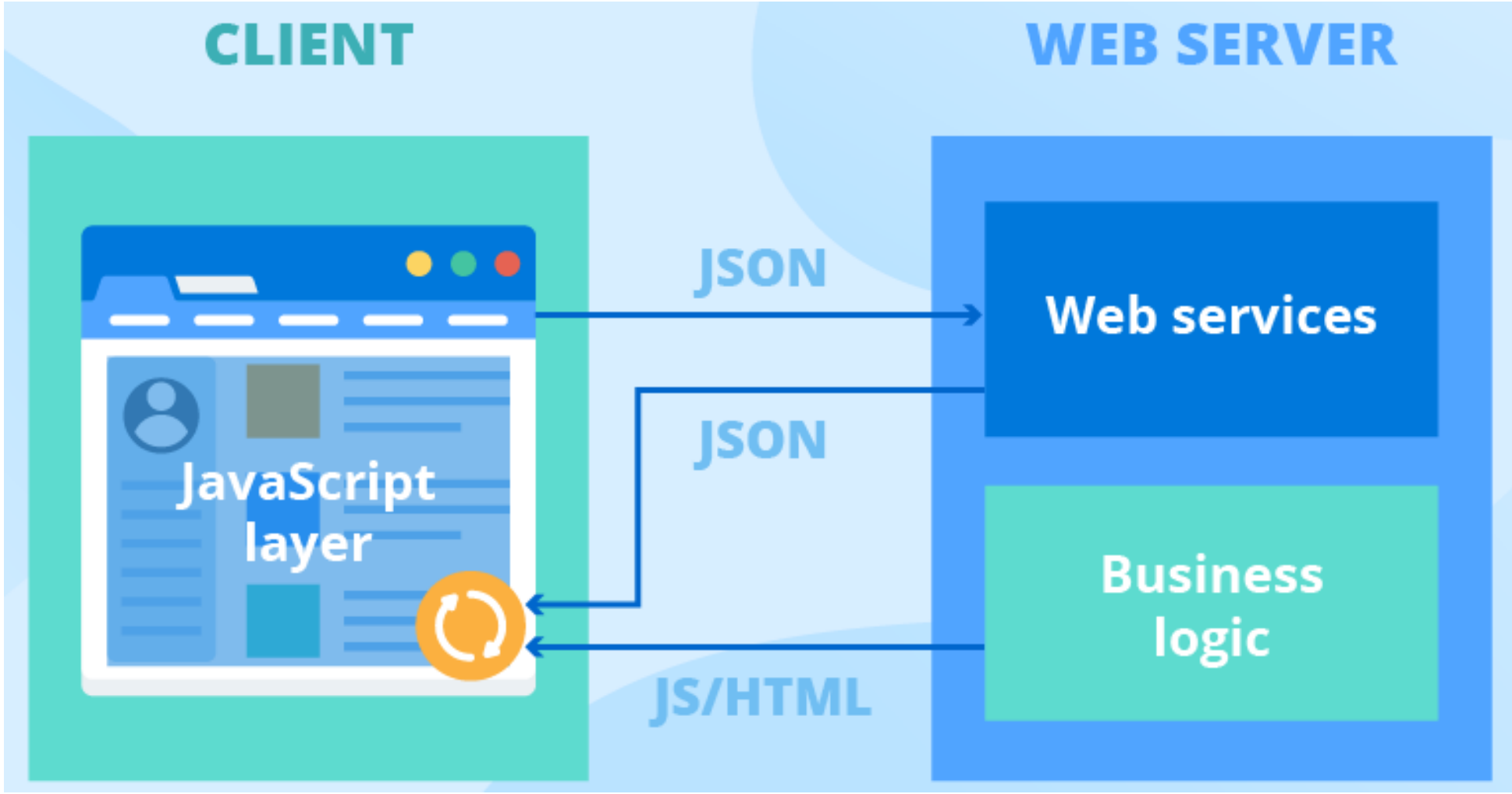

CLIENT
WEB SERVER
HTML
HTML
Web construction logic
Business logic

WEB WIDGET

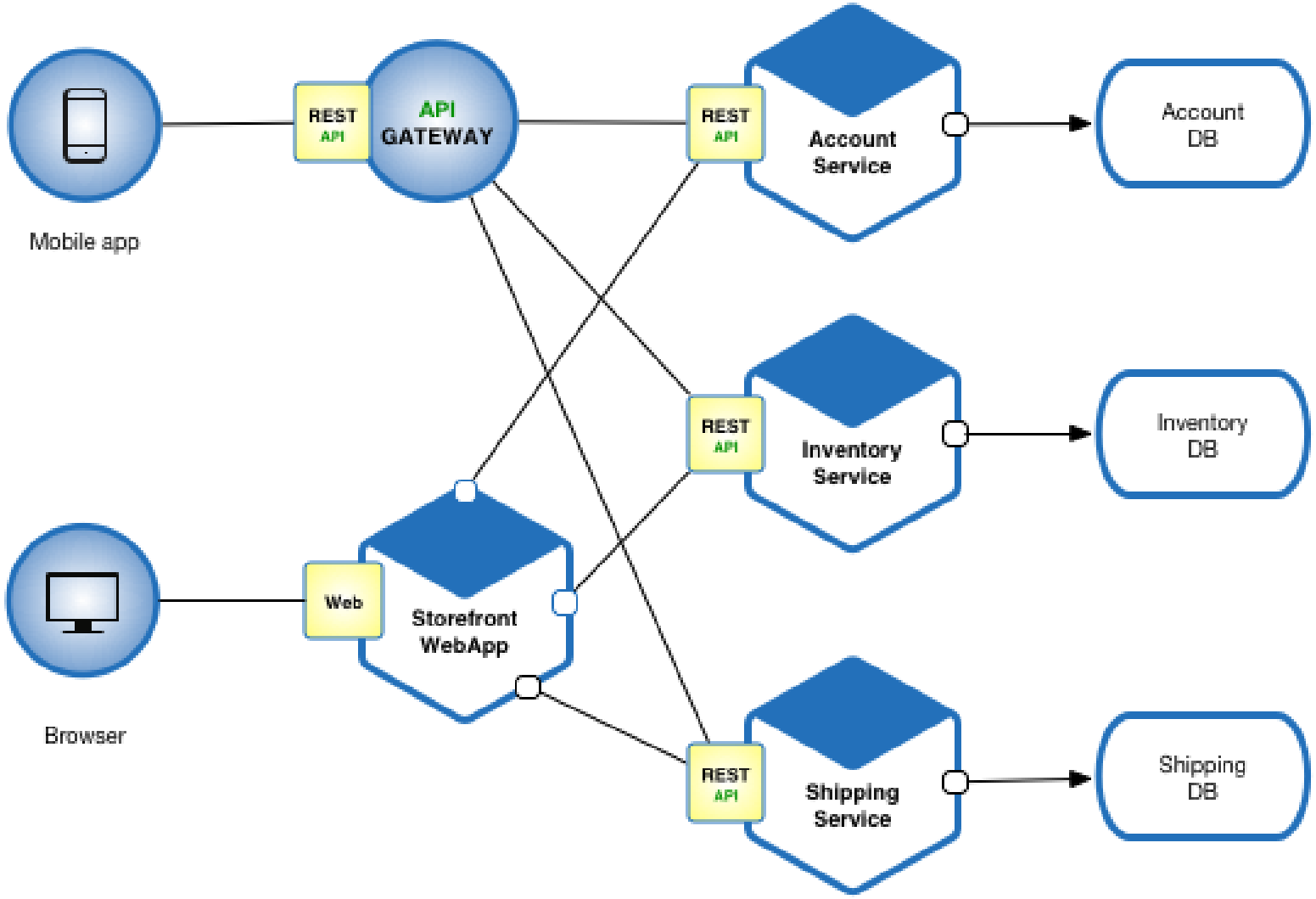
2


CLIENT
WEB SERVER
Widgets
AJAX
JSON
HTML
Web services
Business logic

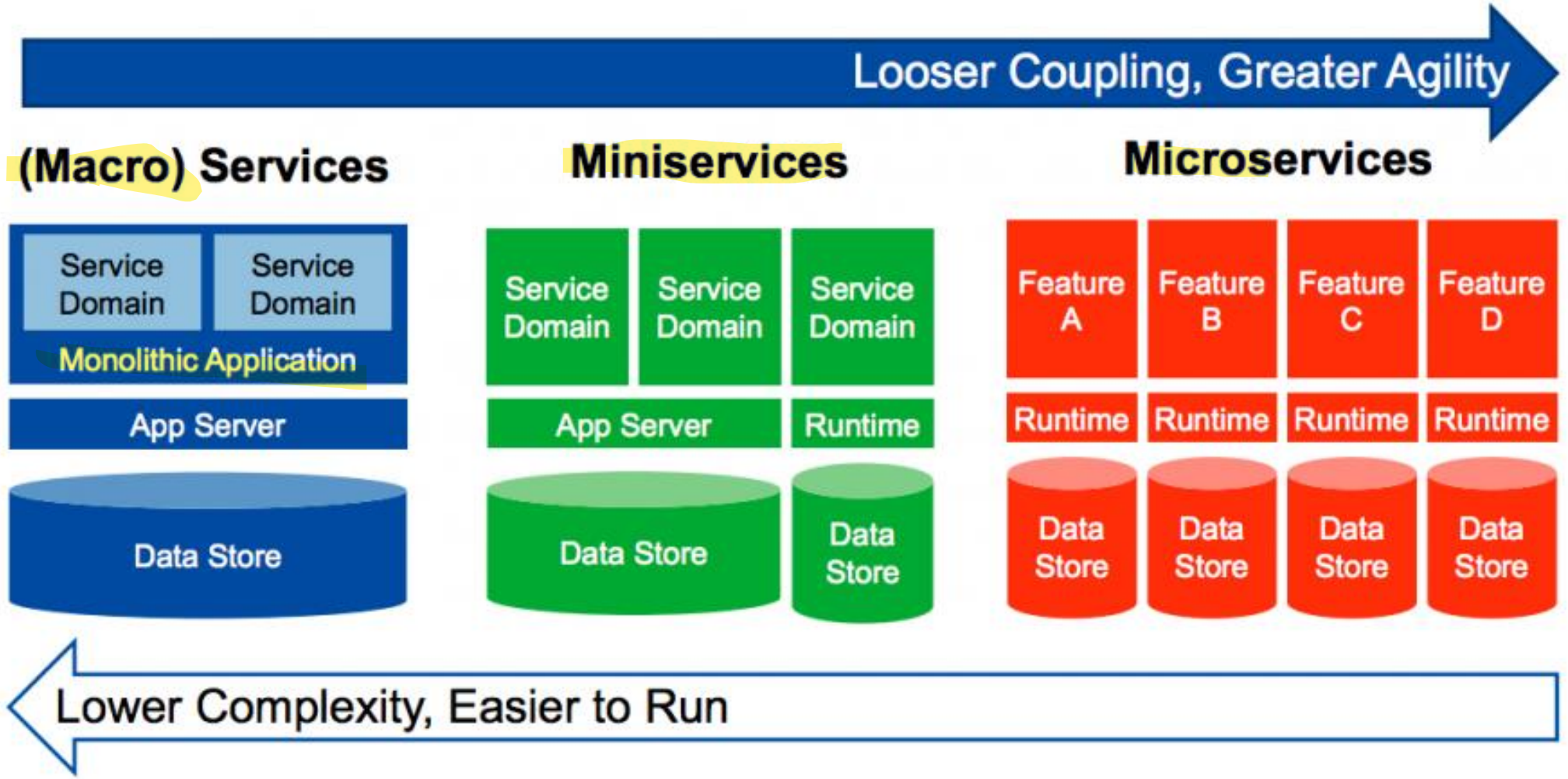
SINGLE PAGE


CLIENT
WEB SERVER
JSON
JSON
JS/HTML
JavaScript layer
Web services
Business logic

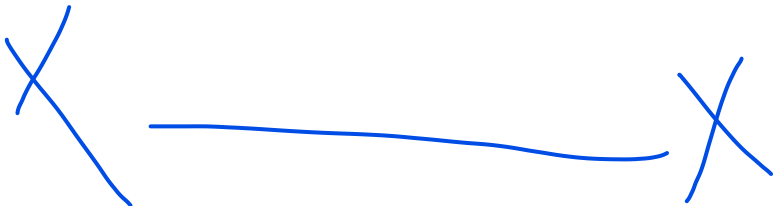
MICROSERVICE



MULTIGRAIN SERVICES



SERVERLESS



TYPES OF TECHNOLOGIES

Front-end Technologies

It is all about the exterior look and feel of an application required to presents and interact user.
By Structure, it consists of three (3) elements:

- HTML (Hypertext Markup Language) defining the structure of web layout
- CSS (Cascading Style Sheets) define the style of the elements & content
- JavaScript enabling the interactivity of the web page in which the app is running

Back-end Technologies

It is all about the inner workings of an application required to operate smoothly.
By Structure, it consists of four (4) elements:

- Programming languages (for example Python, PHP, JavaScript)
- Frameworks (for example Ruby on Rails, Flask, Django, Swift or Objective-C)
- Databases (for example, MongoDB and MySQL)
- Server providers (Apache, Nginx, etc.)

POPULAR FRONT-END TECHNOLOGIES

- HTML
- CSS
- Bootstrap
- W3.CSS
- JavaScript
- ES5
- HTML DOM
- JSON
- XML
- jQuery
- Angular
- React
- Backbone.js
- Express.js
- Ember.js
- Redux
- Storybook
- GraphQL
- Meteor.js
- Grunt
- Gulp

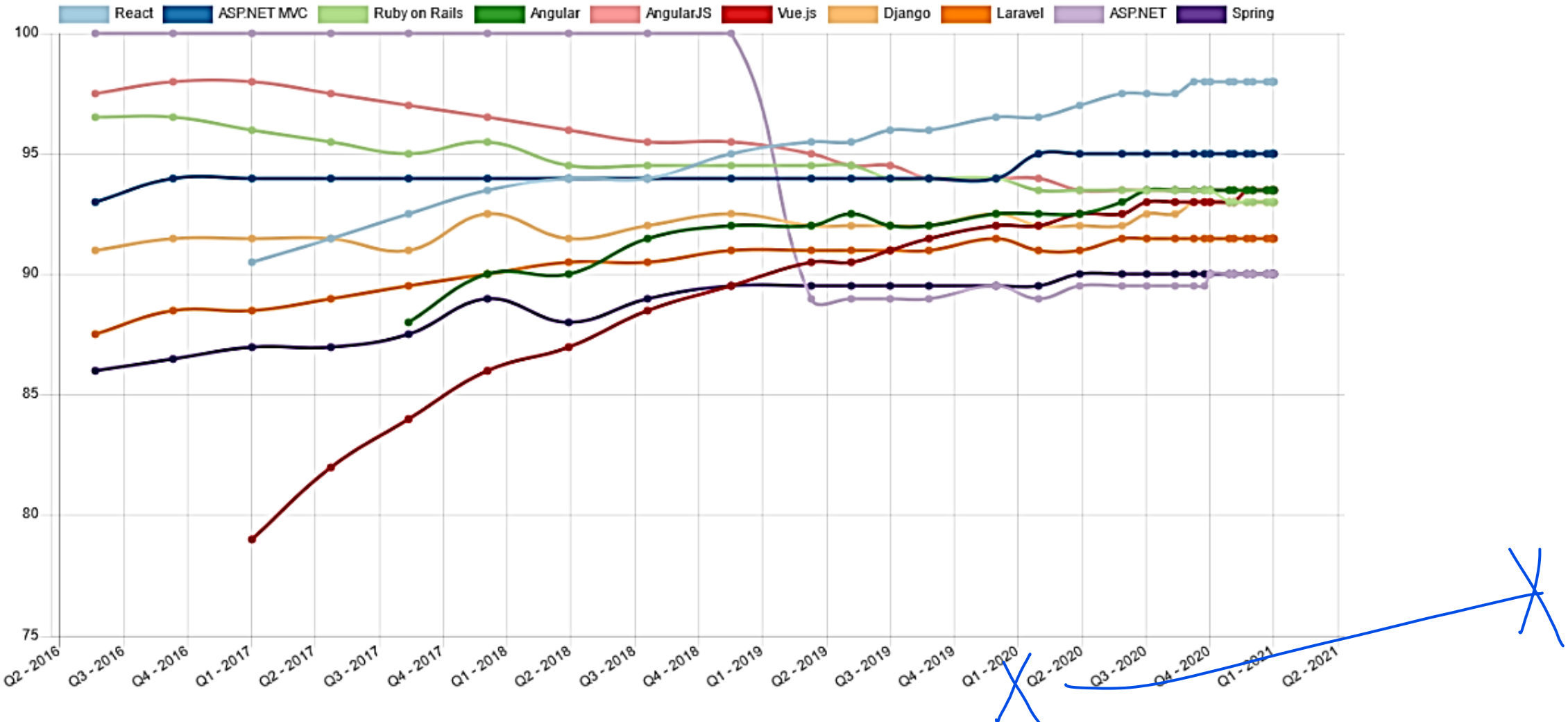
POPULAR BACK-END TECHNOLOGIES

- PHP
- ASP
- C++
- C#
- Java
- Python
- Node.js
- Ruby
- REST
- GO
- SQL/MS-SQL
- MySQL
- MongoDB
- Firebase.com
- Sass
- Less
- Parse.com
- PaaS (Azure and Heroku

POPULAR
TOP – 10 FRAMEWORKS

S.#	Framework	Latest Version	Language Use
1	React	16.12.0	JavaScript
2	ASP.NET MVC	5.2.7 /3.0.0 core	C#
3	AngularJS	4.0.0	JavaScript
4	Ruby on Rails	6.0.2.1	Ruby
5	Angular	9	JavaScript
6	Vue.js	2.6.10	JavaScript
7	Django	Django 1.9.2	Python
8	Laravel	5.5 LTS	PHP
9	ASP.NET		C#
10	Spring	5.1.4	Java

POPULAR FRAMEWORKS 2016-2021



STACK

In general, a **stack** is simply a container which represent the collection of things.

TECHNOLOGY STACK

A **technology stack**, also known as a **solutions stack**, **tech stacks**, **technology infrastructure**, or a **data ecosystem**, is a set of all tools, technologies and programming language that used to build and run an application. It consists of a software applications and frameworks as well.

POPULAR TECHNOLOGY STACK

- LAMP stack: JavaScript - Linux - Apache - MySQL - PHP
- LEMP stack: JavaScript - Linux - Nginx - MySQL - PHP
- MEAN stack: JavaScript - MongoDB - Express - AngularJS - Node.js
- Django stack: JavaScript - Python - Django - MySQL
- Ruby on Rails: JavaScript - Ruby - SQLite - PHP

FULL-STACK DEVELOPERS

A full stack web developer is a person who can develop both client and server software including intermediate program or services.

- Program a browser (like using JavaScript, jQuery, Angular, or Vue)
- Program a server (like using PHP, ASP, Python, or Node)
- Program a database (like using SQL, SQLite, or MongoDB)

HYPER-TEXT MARKUP LANGUAGE (HTML)

HTML

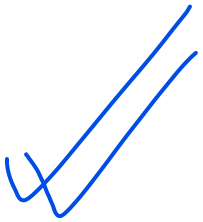
- **Hypertext** means a referencing or linking text, which indicating other piece of information that either exist within the document (internal reference) or outside documents (external reference).
- **Markup** means symbols, code that define the document text and its nature. It helps web browser to understand what is listed in document what are they representing. This is commonly known as Tags or Elements.
- **Language** means set of rules which defines how to write the language, what are the right ways and what is not acceptable, provides means on understanding the contents and how use language symbols, character and codes.

- **H**yper**t**ext **M**arkup **L**anguage (HTML) uses a markup system composed of elements which has their specific meaning and represent content.
- HTML represents only *what* is presented to a User, not responsible *how* it is presented (Visual Style) and how is engage and interact with user. Web browser reads the documents markup elements and presents only the documents content in human readable form.

HTML is not a programming language, it has no logic.
It is only a **structural** or **markup defination**.

HTML HISTORY

Version	Standard	Year	Release Date
1	n/a	1991	1994-01-01
2.0	RFC 1866	1995	1995-11-24
3.2	W3C: HTML 3.2	1997	1997-01-14
4.0	W3C: HTML 4.0	1998	1998-04-24
4.01	W3C: HTML 4.01	1999	1999-12-24
XHTML		2000	
5	WHATWG: HTML Living Standard	2014	2014-10-28
5.1	W3C: HTML 5.01	2016	2016-11-01



OLD vs NEW HTML

- The old HTML version does not has semantic tags. Thus, only browsers can read the structure and human can manually understand and meaning to the content in document or html page.
- The latest HTML version provides the semantic tags which offers machine-readability, understanding and meaning to the content in document.

ELEMENTS vs TAGS

- **Elements** are the name or definition of document which describe the type of content and what are their predefined properties that distinguish it in html structure and web browser representation that is more meaningful and understandable to human. Such as built-in predefined properties distinguish them e.g. size.
- Elements example: h1, p,
- **Tags** defines the boundaries of an element in a document structure. It helps web browser to understand and translate the Tag area in representative style. Tag itself nothing, but it become meaningful when uses some element to define it. There are only two types of Tags exists in HTML;
  - The pair tag that has opening and closing tag. They can hold content in them.
  - The single or void tag which has only opening tag and no closing tag. They can only represent attached content but cannot hold content.

HTML Tag is represented by angle brackets **<p>**. It contain an **element name**.

ELEMENTS vs TAGS: EXAMPLE

Elements

- Example: head, body, h1,h2, h3, h4, p, etc.

Tags

- Example: <></>, </>

HTML Tag Syntax:

- Pair Tag
  - Opening tag <elementName> e.g: <head>, <body>,
  - Closing tag </elementName> e.g: </head>, </body>,
- Single or Void Tag
  - Single Tag e.g: <img src="../images/home.jpg">
  - Void Tag </> e.g: </br>

BASIC HTML LAYOUT

- A HTML page may consist of several tags to hundreds of elements. A basic HTML Structure is shown below

```
<html>
  <head>
    <title>Page title</title>
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
  </body>
</html>
```

HTML TAG BY TYPES

HTML Tags can be distributed into two broad types

- **Block Elements**, these elements also known as stack elements. When you add these elements covers complete width screen display and appear below each other in a vertical stack form
- **Inline Elements**, only covers the length of its content and rest of the screen remain empty/unused. These elements with no content are called empty **elements**. Empty **elements do not have an end tag**, such as the
 element (which indicates a line break).

HTML TAGS BY CATEGORIES

- Basic Tags
- Meta Tags
- Link
- List
- Table
- Media (Image, Audio, Video)
- Form and Input
- Frame
- Style
- Semantic
- Formatting
- programming

BASIC HTML TAGs

Tag	Description
<!DOCTYPE>	Defines the document type
<html>	Defines an HTML document
<head>	Defines information about the document
<title>	Defines a title for the document
<body>	Defines the document's body
<h1> to <h6>	Defines HTML headings
<p>	Defines a paragraph
 	Inserts a single line break
<hr>	Defines a thematic change in the content
<!--...-->	Defines a comment

META and LINKS TAGs

Tag	Description
<head>	Defines information about the document
<meta>	Defines metadata about an HTML document
<base>	Specifies the base URL/target for all relative URLs in a document

Tag	Description
<a>	Defines a hyperlink anchor tag
<link>	Defines the relationship between a document and an external resource (most used to link to style sheets)
<nav>	Defines navigation links

LIST and TABLE TAGs

Tag	Description
	Defines an unordered list
	Defines an ordered list
	Defines a list item
<dl>	Defines a description list
<dt>	Defines a term/name in a description list
<dd>	Defines a description of a term/name in a description list

Tag	Description
<table>	Defines a table
<caption>	Defines a table caption
<th>	Defines a header cell in a table
<tr>	Defines a row in a table
<td>	Defines a cell in a table
<thead>	Groups the header content in a table
<tbody>	Groups the body content in a table
<tfoot>	Groups the footer content in a table

MEDIA TAGs

Tag	Description
<img>	Defines an image
<map>	Defines a client-side image-map
<area>	Defines an area inside an image-map
<canvas>	Used to draw graphics, on the fly, via scripting (usually JavaScript)
<figcaption>	Defines a caption for a <figure> element
<figure>	Specifies self-contained content
<picture>	Defines a container for multiple image resources
<svg>	Defines a container for SVG graphics
Tag	**Description**
<audio>	Defines sound content
<source>	Defines multiple media resources for media elements (<video>, <audio> and <picture>)
<track>	Defines text tracks for media elements (<video> and <audio>)
<video>	Defines a video or movie

STYLES and SEMANTICS TAGs

Tag	Description
<style>	Defines style information for a document
	Defines a section in a document
	Defines a section in a document
<header>	Defines a header for a document or section
<footer>	Defines a footer for a document or section
<main>	Specifies the main content of a document
<section>	Defines a section in a document
<article>	Defines an article
<aside>	Defines content aside from the page content
<details>	Defines additional details that the user can view or hide
<dialog>	Defines a dialog box or window
<summary>	Defines a visible heading for a <details> element
<data>	Links the given content with a machine-readable translation

PROGRAMMING and FRAME TAGs

Tag	Description
<script>	Defines a client-side script
<noscript>	Defines an alternate content for users that do not support client-side scripts
<object>	Defines an embedded object
<param>	Defines a parameter for an object

Tag	Description
<iframe>	Defines an inline frame

FORMATTING TAGs

Tag	Description
<abbr>	Defines an abbreviation or an acronym
<address>	Defines contact information for the author/owner of a document/article
<b>	Defines bold text
<bdo>	Overrides the current text direction
<cite>	Defines the title of a work
<code>	Defines a piece of computer code
<del>	Defines text that has been deleted from a document
<dfn>	Represents the defining instance of a term
	Defines emphasized text
<i>	Defines a part of text in an alternate voice or mood
<ins>	Defines a text that has been inserted into a document
<kbd>	Defines keyboard input
<mark>	Defines marked/highlighted text
<meter>	Defines a scalar measurement within a known range (a gauge)
<pre>	Defines preformatted text
<progress>	Represents the progress of a task
<s>	Defines text that is no longer correct
<small>	Defines smaller text
	Defines important text
<sub>	Defines subscripted text
<sup>	Defines superscripted text

- **DOCTYPES** tell browsers which version of HTML you are using.
- You can add comments to your code between the **<!--** and **-->** markers.
- The **id** and **class** attributes allow you to identify particular elements.
- The ** ** and **** elements allow you to group block-level and inline elements together.
- **<iframes>** cut windows into your web pages through which other pages can be displayed.
- The **<meta>** tag allows you to supply all kinds of information about your web page.
- Escape characters are used to include special characters in your pages such as **<**, **>**, and **©**

BASIC HTML PAGE

<html>

<body>

<h1>This is the Main Heading**</h1>**

** **This text might be an introduction to the rest of the page. And if the page is a long one it might be split up into several sub-headings.** **

<h2>This is a Sub-Heading**</h2>**

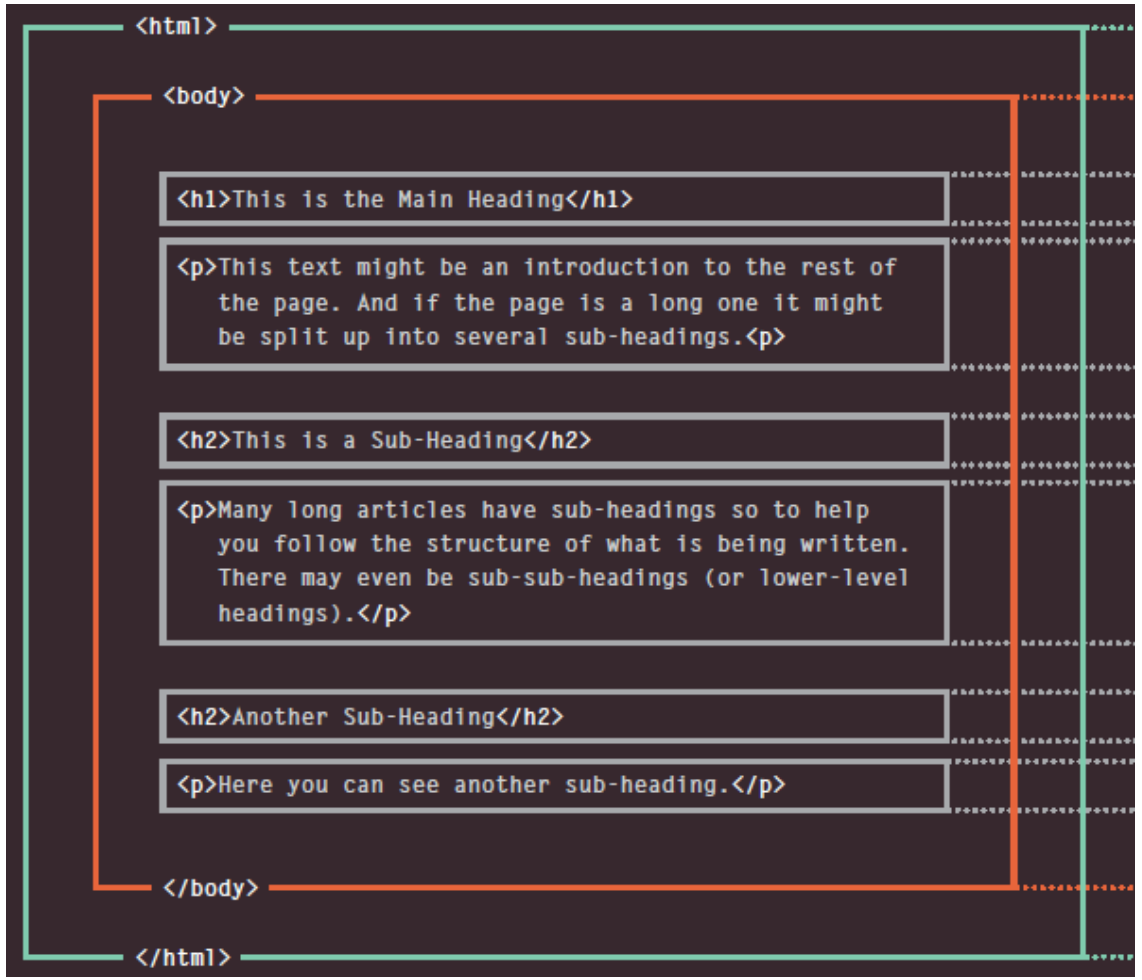
** **Many long articles have sub-headings so to help you follow the structure of what is being written. There may even be sub-sub-headings (or lower-level headings).** **

<h2>Another Sub-Heading**</h2>**

** **Here you can see another sub-heading.** **

</body>

</html>



TAG'S ATTRIBUTES

- Tag Attributes are the additional piece of information describing the tag. Each tag has individual set of attributes that could define. While having some common attributes are Class, ID etc.
- **Example**

```
<p lang="en-us">Paragraph in English</p>
```

- **lang** is an additional attribute
- **en-us** is value of attribute always given in **“ ”**. were as **en-us** define that it is in US English language.

UNDERSTANDING TAGS ROLE

```
<html>
<body>
<h1> Experience some useful HTML tags </h1>
<h2> PARAGRAPH </h2>
<p> A paragraph consists of one or more sentences that form a self-contained unit of discourse. The start of a paragraph is indicated by a new line.</p>
<h2> BOLD and ITALIC </h2>
<p>Inside a <strong> product</strong> description you might see some <b>key features</b> in bold. This is how we make a <em>word</em> appear <i>italic</i></p>
<h2> SUBSCRIPT AND SUPERSCRIPT </h2>
<p>On the 4<sup>th</sup> of September you will learn about CO<sub>2</sub>.</p>
<h2> Collapse White and empty space</h2>
<p>The HTML collapse     white space, it does not support.</p>
<h2> Line Break</h2>
<p>This paragraph appears<br/> in two line.</p>
<h2> Horizontal Rule</h2>
<p>This is how you can add horizontal line on html page.</p><hr/>
<h2> Quote Text</h2>
<blockquote> <p>This is block level quotation </p> </blockquote>
<p> this is inline <q>Quotation</q>
<h2> Updating content Text while leaving old one</h2>
<p>It was the <del>worst</del> <ins>best</ins> idea she had ever had.</p>
</body>
```

OUTPUT

Experience some useful HTML tags

PARAGRAPH

A paragraph consists of one or more sentences that form a self-contained unit of discourse. The start of a paragraph is indicated by a new line.

BOLD and ITALIC

strong b em italic

Inside a **product** description you might see some **key features** in bold. This is how we make a *word* appear *italic*.

SUBSCRIPT and SUPERSCRIPT

sup sub

On the 4th of September you will learn about CO_2.

Collapse White and empty space

The HTML collapse white space, it does not support.

OUTPUT

Line Break

This paragraph appears
in two line.

br

Horizontal Rule

This is how you can add horizontal line on html page.

---

hr

Quote Text

blockquote

This is block level quotation

q

this is inline “Quotation”

Updating content Text while leaving old one

del

It was the ~~worst~~ best idea she had ever had.

ins

EXPERIENCE SOME ADVANCE HTML TAGS

```
<!doctype html>
<html>
<body>
<h1> Experience some advance HTML tags </h1>
<h2>Working with List</h2>
<p>There are three types of list you can create in HTML <br/><code>UL</code> Defines an unordered list <br/> <code>OL</code> Defines order list<br/><code>DL</code> Defines defination or description list.<br/> let's have experince them.</p>
<h3> Unordered List</h3>
<ul> It uses <code>ul tag</code> to define list, and <code>li tag</code> to define list item
<li>Use <code>type</code> attributes to change or define different format.</li>
<li>for Unordered List, use type="disc" or "circle" or "square" or "none".</li>
</ul>
<h3> Ordered List</h3>
<ol type="i">It uses <code>ol tag</code> to define list, and <code>li tag</code> to define list item
<li>For Ordered List, use type="1" or "a" or "A" or "i" or "I".</li>
<li>Only with order list one can understand the sequence of item</li>
<li>use <code>reverse</code> keyword to show the list number in reverse order.</li>
<li>List uses <code>start tag</code> to define the staring number, such as start="5", list will start with 5 and so on</li>
<li>use <code>start tag</code> to define the staring point, such as start="5", list will start with 5 and so on</li>
</ol>
<h3> Defiination List</h3>
<dl>It uses <code>dl tag</code> to define list, <code>dt tag</code> to define data term, and <code>dd tag</code> for term defination.
<dt>Unordered or Ordered List</dt>
<dd>The unordered and ordered list has indentation by default.</dd>
<dt>Defination List</dt>
<dd>The defination list has no indentation</dd>
</dl>
```

OUTPUT

code

Experience some advance HTML tags

Working with List

There are three types of list you can create in HTML
`UL` Defines an unordered list
`OL` Defines order list
`DL` Defines defination or description list.
let's have experince them.

Unordered List

It uses `ul tag` to define list, and `li tag` to define list item

- Use `type` attributes to change or define different format.
- for Unordered List, use type="disc" or "circle" or "square" or "none".

Ordered List

It uses `ol tag` to define list, and `li tag` to define list item

i. For Ordered List, use type="1" or "a" or "A" or "i" or "I".
ii. Only with order list one can understand the sequence of item
iii. use `reverse` keyword to show the list number in reverse order.
iv. List uses `start tag` to define the staring number, such as start="5", list will start with 5 and so on
v. use `start tag` to define the staring point, such as start="5", list will start with 5 and so on

CASCADE STYLE SHEET (CSS)

CSS BREAKDOWN

Before driving into CSS, lets breakdown CSS to understand it

- **C –** Cascade – defines as a bunch of rules defined in a series or sequence that come one after other.
- **S –** Style – defines the appearance of web page content or element.
- **S –** Sheet – generally refers to the book or paper of definition.

WHAT IS CSS?

Cascading Style Sheets (**CSS**) is not a programming language, it is an abstract sheet of rules defining appearance of web pages elements.

- It requires abstract thinking and some creativity.
- It is simply about declaring rules, under various conditions we want certain things to happen.
- It is an appearance style language used to describe the presentation flow of a web page.

WHY TO STYLE ?

- HTML provides the contents of webpage and offers media to attach, usually in left align format.
- But, user cannot be attracted by simple content text in plain format.
- Human interaction require more interactive content to keep engage and deliver better.
- A better appearance keep people to stay longer at web page, serve more and generate more business.

PURPOSE OF CSS

Besides styling rules, the purpose of CSS the separation of concerns.

- Means keep the things separate and working seamless in collaboration. It allows the design, layout, and presentation of a web page separately from each other. This separation helps keeps source code readable, reliable and importantly avoid dependency and disturbance.
- **For example**, a designer can update styles separately from a developer who is creating the page structure or a web editor who is changing content on a page.

BRIEF HISTORY OF CSS VERSIONS

- CSS was first proposed by Håkon Wium Lie on **October 10, 1994.**
- The World Wide Web Consortium (W3C) first CSS Recommendation known as **CSS1** being released in **1996.**
- A proposal by Bert Bos was influential, and he became co-author of CSS1.
- As HTML grew, it came to encompass a wider variety of stylistic capabilities to meet the demands of web developers. This evolution gave the designer more control over site appearance, at the cost of more complex HTML.
- CSS has various levels and profiles (version). Each level of CSS builds upon the last, typically adding new features and typically denoted as CSS 1, CSS 2, CSS 3, and CSS 4 (coming sson).

CSS 1 - DECEMBER 17, 1996.

CSS1 has following capabilities and support for

- Font properties such as typeface and emphasis
- Color of text, backgrounds, and other elements
- Text attributes such as spacing between words, letters, and lines of text
- Alignment of text, images, tables and other elements
- Margin, border, padding, and positioning for most elements
- Unique identification and generic classification of groups of attributes

CSS2 - MAY 1998

CSS 2 includes a number of new capabilities like

- Absolute, Relative, and Fixed positioning of elements
- Z-index,
- Concept of media types,
- Support for aural style sheets

CSS level 2 revision 1, often referred to as "CSS 2.1", fixes errors in CSS 2

- CSS 2.1 first became a candidate recommendation on **February 25, 2004**.
- Reverted to a Working Draft on **June 13, 2005** for further review.
- Again became candidate recommendation on **19 July 2007** and then **updated twice in 2009**.
- CSS 2.1 was planned as the first and final revision of level 2 on W3C Recommendation on **7 June 2011**
- But low priority work on **CSS 2.2 began in 2015**.

CSS3 – 13 OCT, 2015

- CSS 3 is divided into several separate documents called "modules".
- Each module adds new capabilities or extends features defined in
- CSS 2, preserving backward compatibility.

Version Release Date

- CSS1 - 1996-12-17
- CSS2 - 1998-05-12
- CSS3 - 2015-10-13

CSS
HOW TO YOU USE IT

Typically, there are three ways to in-cooperate CSS with your web page(s).

1. **Inline**
2. **Internal**
3. **External**

INLINE CSS

- Inline CSS technique is the oldest practice that developer use.
- This is the hardest and heaviest approach to design webpage, especially in large website. It is fairly acceptable in small static one or two pages websites but not a good approach.
- In an Inline CSS, rule is explicitly define at each individual html tag and element. Thus, no consistency, high manageability, less controllability, large time requires.
- An **example** of inline CSS is mention below.

```
<p style= “Color: blue”> abc</p>
```

INTERNAL CSS

- Internal CSS technique is also one of past practices.
- Comparative to inline CSS, it is much better manageable and less complex approach.
- It is fairly acceptable in small to medium size websites but not a preferable approach in current era. In an internal CSS, rules become generalize than explicitly define at each individual html tag and element.
- Therefore, internal CSS implement on single webpage, each page require to define its CSS in order to work. Thus, this offer consistency, manageability, and controllability at some degree.
- An **example** of internal CSS is mention below.

```
<head>
<style type="text/css">
p { Color: blue; }
</style>
</head>
```

EXTERNAL CSS

- External CSS technique is the most preferable practice.
- Comparative to inline or internal CSS, it offers better manageable and less complex approach. It is most acceptable and applicable to any size and complexity of web structure.
- In an external CSS, rules are separated in from individual tag and individual web page. It offers reusability of rules, avoid extensive efforts, consistency, manageability, and controllability.
- An example of external CSS is mention below.

In HTML Page

```
<head>
<link rel="stylesheet" type="text/css"
href="style.css" />
</head>
```

In an separate file named “styel.css”

```
body { color: black;
        font-size: 12px;}
p { Color: blue; }
```

CSS
WHAT YOU CAN DO WITH IT

WHAT YOU CAN STYLE

- The designer can style each and every single tag of html and from blank space to rich media content on webpage.

WHAT YOU CAN USE TO STYLE

- To style an HTML web page you can use HTML tag name (element name), Tag's ID and Class values

WITH CSS WHAT YOU CAN DO

- Using CSS person can define and implement three things.
- **What to show**: covers which content should be visible.
- **Where to show**: covers the position of content.
- **How to show**: covers the appearance style

CSS
UNDERSTANDING RULE ADOPTION AND PRIORITY

Whenever there is conflict between the defined rules. The cascade considers three things to resolve the problem:

1. **Stylesheet origin**
   Where the styles come from. Your styles are applied in conjunction with the browser's default styles.
2. **Selector specificity**
   Which selectors take precedence over which.
3. **Source order**
   Order in which styles are declared in the stylesheet.

CSS
UNDERSTANDING STYLESHEET ORIGIN

- The stylesheets you add to web page are not the only ones the browser applies.
  1. **Author Stylesheet**:
     The style sheet added to webpage by a user/developer.
  2. **User agent stylesheet**:
     It is the browser's default styles.
- User agent stylesheet has ***lower priority***, so author styles **override** it.

CSS
UNDERSTANDING SPECIFICITY

If conflicting declarations can't be resolved based on their origin, the browser next tries to resolve them by looking at their ***specificity level***. Different types of selectors also have different specificities

The exact rules of specificity are:

- If a selector has more IDs, it wins (that is, it's more specific).
- If that results in a tie, the selector with the most classes wins.
- If that results in a tie, the selector with the most tag names wins.

CSS
UNDERSTANDING SPECIFICITY

```
html body header h1 {
  color: blue;
}
```

1 Four tags

```
body header.page-header h1 {
  color: orange;
}
```

2 Three tags and one class

```
.page-header .title {
  color: green;
}
```

3 Two classes

```
#page-title {
  color: red;
}
```

4 One ID

Selector	IDs	Classes	Tags	Notation
`html body header h1`	0	0	4	0,0,4
`body header.page-header h1`	0	1	3	0,1,3
`.page-header .title`	0	2	0	0,2,0
`#page-title`	1	0	0	1,0,0

THE CASCADED VALUES

- The browser follows these three steps—origin, specificity, and source order to resolve every property for every element on the page. A declaration that "wins" the cascade is called a *cascaded value*.

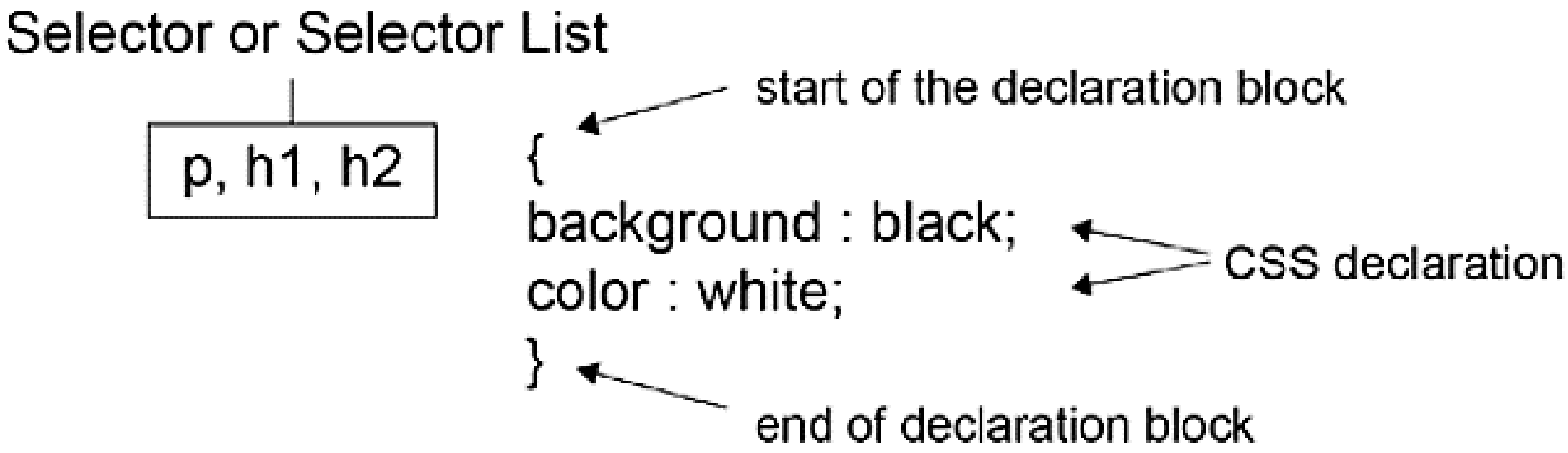
Suggestions

- **Don't use or avoid IDs in selector**.
  It is difficult to name each individual tag name that usually hard to manage and remember.
- **Don't use !important keyword.**
  This is even more difficult to override than an ID, and once you use it, you will need to add it every time you want to override the original declaration.

A key part of CSS development comes down to writing rules in such a way that they're predictable.
Use meaningful name of class

CSS
RULE ANATOMY

- Understand the CSS Rule anatomy



CSS
BASIC SELECTORS TYPES

There are basically four types of selectors in CSS

1. **ID selector**, #id
   Targets the element with the specified ID attribute. It has the ***highest priority*** and specificity of 1, 0, 0.
   Example: #sidebar.

2. **Class selector**, .class
   Targets elements that have the specified class name as part of their class attribute. It has the ***mid priority*** and a specificity of 0, 1, 0.
   Examples: .media; .nav-menu.

3. **Tag** or **Type selector**, tagName
   This selector matches the tag name of the elements to target. It has a ***low priority*** and a specificity of 0, 0, 1.
   Examples: p; h1; strong.

4. **Global** or **Universal selector**. *
   Targets all elements. It has the ***least priority*** and a specificity of 0, 0, 0.

CSS RULE
EXAMPLES

- **html { color: blue;}**
  Applies to all elements inside HTML tag
- **body {color: red;}**
  Applies to all elements inside body tag
- ***, *:before, *:after {color: inherit;}**
  Applies to all elements on HTML page
- **p { color: white;}**
  Applies to all p tag or elements on html page
- **#paragraph {color: green;}**
  Applies only to a tag name "paragraph" inside html page
- **.paragraph{color: black;}**
  Applies to all tag or elements having class name "paragraph" on html page
- **h1, h2, h3{ color: orange;}**
  Applies to all h1, h2 and h3 tags on html page. Grouping multiple selectors.
- **p, .paragraph, #paragraph{color: black;}**
  Applies to all p tags, class name "paragraph" and tag id name "paragraph" on html page. Grouping multiple type of selectors.

CSS
COMBINATOR SELECTORS

Combinator join multiple simple selectors into one complex selector. For example, **nav-menu li**. It indicates the targeted <li> is a descendant of an element that has the nav-menu class. The space between the two simple selectors is known as a ***descendant indicator***.

There are a few others combinatory types, each indicates a particular relationship between the elements.

- ***Child combinator*** (>) —Targets elements that are a direct descendant of another element.
  For Example: .parent > .child.
- ***Adjacent sibling combinator*** (+) —Targets elements that immediately follow another.
  For Example: p + h2.
- ***General sibling combinator*** (~) —Targets all sibling elements following a specified element.
  Example: li.active ~ li.
  This doesn't target siblings that appear before the indicated element.

CSS COMBINATOR SELECTORS EXAMPLES

- **div p { background-color: yellow; }**
  Descendant selector (space), applies to all <p> elements inside elements
- **div > p { background-color: yellow; }**
  Child selector (>), applies to all <p> elements that are children of a element
- **div + p { background-color: yellow; }**
  Adjacent sibling selector (+), applies to all <p> elements that are placed immediately after elements
- **div ~ p { background-color: yellow;}**
  General sibling selector (~), applies to all <p> elements that are siblings of elements, irrespective of immediate or adjacent siblings.

CSS
COMPOUND SELECTORS

- Multiple simple selectors can be joined (without spaces or other combinators) to form a *compound* selector.
- For Example,
  .dropdown.is-active {color: green}
  always targets but not alone .
- **.dropdown.is-active{color: red}**
  Applies to tags that have both class names in class attribute, but it does not apply any of class alone.
- **p.paragraph {color: yellow}**
  Applies to all p tags that have paragraph class, but it does not apply alone.
- **p#head{color: blue}**
  Applies to only p tags that have ID with head name, but it does not apply alone.

CSS: ADVANCE SELECTORS
PSEUDO-ANCHOR SELECTORS

- Anchor pseudo-classes provide various types of special effects to the element of HTML.
- It represent the state of links as unvisited, visited, or currently selected.
- Also, it enables to activate the HTML elements or apply a specified style to an element when the mouse pointer is kept over it or element is focused.
- **a:link { color: blue;}**
  Normal color of link when is unvisited or display on html page.
- **a:visited {color: #00FF00;}**
  New color of link when is visited clicked by user.
- **a:hover {color: #FF00FF;}**
  New color of link when is mouse enter to the content region of link or mouse hover on link content.
- **a:active {color: #0000FF;}**
  New color of link when content of link or link is selected by user.
- **input:focus {background-color: yellow;}**
  On getting focus of any input form field, change the background color of element.

CSS: ADVANCE SELECTORS
PSEUDO-CLASS SELECTORS

- Pseudo-class selectors are used to target elements when they're in a certain state. Pseudo element always begin with a colon (:) followed by the Pseudo element. It has the same specificity as a class selector (0,1,0).
- **ParentClass:first-child**
  Targets elements that are the first element within their parent element.
- **ParentClass:last-child**
  Targets elements that are the last element within their parent element.
- **ParentClass:only-child**
  Targets elements that are the only element within their parent element (no siblings).
- **ParentClass:nth-child(*an+b*)**
  Targets elements based upon their position among their siblings. The formula *an+b*, indicates which child elements to target, where a, b are integers and n indicate children number.

CSS: ADVANCE SELECTORS
PSEUDO-CLASS SELECTORS

Selector	Elements targeted	Result	Description
`:nth-child(n)`	0, 1, 2, 3, 4 ...		Every element
`:nth-child(2n)`	0, 2, 4, 6, 8 ...		Even elements
`:nth-child(3n)`	0, 3, 6, 9, 12 ...		Every third element
`:nth-child(3n+2)` starts	2, 5, 8, 11, 14 ...		Every third element beginning with element 2
`:nth-child(n+4)`	4, 5, 6, 7, 8 ...		All elements beginning with element 4
`:nth-child(-n+4)`	4, 3, 2, 1, 0 ...		First four elements

CSS: ADVANCE SELECTORS
PSEUDO-CLASS SELECTORS

- **li:first-child {background: yellow;}**
- Applies to all first child element (i.e.) of any list tag (e.g. , <ol>, etc.) on webpage
- **ul > :first-child { background: yellow; }**
  Applies to all first child element (i.e.) of each <ul> tag on webpage
- **p:first-of-type { background: red;}**
  Applies to all first <p> type element in its parent container on webpage
- **p:last-child {background: yellow;}**
  Applies to all last <p> element of in its parent container on webpage
- **p:last-of-type {background: yellow;}**
  Applies to all first <p> type element of its parent container on webpage
- **p:nth-child(odd) { background: red;}**
  Applies to all odd occurrence of <p> element in its parent container on webpage, from 1 or start.

CSS: ADVANCE SELECTORS
PSEUDO-CLASS SELECTORS

- **p:nth-child(even) { background: blue;}**
  Applies to all even occurrence of <p> element in its parent container on webpage from 1 or start.
- **p:nth-last-child(odd) { background: red;}**
  Applies to all odd occurrence of <p> element in its parent container on webpage, from n or end or last.
- **p:nth-last-child(even) { background: blue;}**
  Applies to all even occurrence of <p> element in its parent container on webpage, from n or end or last.
- **p:nth-of-type(2) {background: red;}**
  Applies to all second <p> type element of its parent container on webpage.
- **p:nth-child(3n+0) { background: red;}**
  Applies to all third <p> element in its parent container on webpage, starting from 0 index.
- **p:nth-last-child(3n+0) {background: red;}**
  Applies to all third <p> element in its parent container on webpage, starting from n index or end or last.
- **p:only-of-type {background: #ff0000;}**
  Applies to <p> element in its parent container on webpage, which has only (only one) <p> type element.

CSS: ADVANCE SELECTORS
PSEUDO-ELEMENT SELECTORS

Pseudo-elements are similar to pseudo-classes, it start with a double-colon (::) and has same specificity as a type or tag selector (0, 0, 1). It targets a certain part of the page that doesn't directly correspond to a particular element in the HTML. It targets portion of an element or use to inject content into the page *where the markup defines nothing*.

- **Element::first-letter** — Use to specify styles only for first character of text inside an element.
- **Element::first-line** — Use to specify styles only first line of text inside an element.
- **Element::before** —Used to insert text, images, or other shapes *before* the element.
- **Element::after**— Used to insert text, images, or other shapes *after* the element.
- **Element::selection** — Use to specify styles for any text the user has highlighted with their

CSS: ADVANCE SELECTORS
PSEUDO-ELEMENT SELECTORS

Examples

- **p::first-letter { color: blue; font-size: 16px;}**
  Applies to all p tags of webpage and change font size of first letter is 16px
- **p::first-line {color: green; font-variant: small-caps;}**
  Applies to all p tags of webpage and change the first line of paragraph content into small caps.
- **h1::before {content: url(smiley.gif);}**
  Applies to all h1 tags on webpage and add smiley picture before h1 heading.
- **h2::after {content: url(sad.gif);}**
  Applies to all h1 tags on webpage and add sad picture after h2 heading.
- **::selection { color: red; background: yellow; }**
  Applies to all tags on webpage change the content color and background color upon selecting.
- **p.intro::first-letter {color: green; font-size: 20px;}**
  Applies to only p tag having class name “intro” on webpage and change font size of first letter to 20px

The content property must be specified to make this element appear.

CSS
THE BOX MODEL

The browser's engine renders each HTML element as a rectangular box according to the standard **CSS basic box model**. There are two types of boxes.

- ***Inline box element***: simply known as inline elements that covers only the width of content. They appear in horizontal stack or Que.
- ***Block box element***: simply known as block elements that covers full the width of screen. They appear in vertical stack or Que.

Every box is composed of four parts (or *areas*); the *content are*, *padding area, border area*, and *margin area*.

- **Content** Area – Represent the content of the box, i.e. text and images
- **Padding** Area – Represents the space around the content as part of content.
- **Border** Area – Represents the edge of padding area (limit of content).
- **Margin** Area – Represents the space outside the border area. It is a gap between two elements.

They are also refers as four boxes Content-Box, Padding-Box, Border-Box, Margin box

EXAMPLE

- .fontStyle{ font-family: serif font-size: 16px; font-style: italic; font-weight: bold; text-align: center; color: blue;}
- .fontStyle_groupVals{ font: italic small-caps bold 16px serif;}

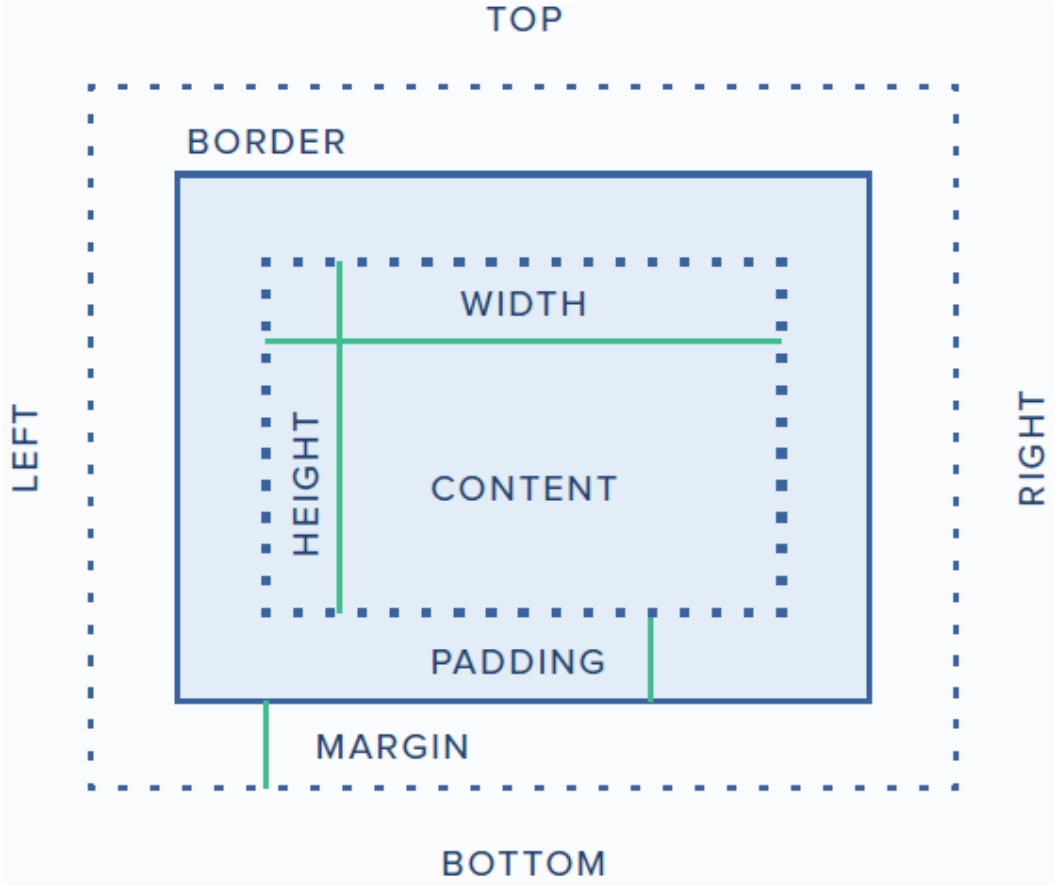
- .padding{ padding-top: 4px; padding-right: 3px; padding-bottom: 8px; padding-left: 6px;}

- .border{ padding-top: 2px; padding-right: 4px; padding-bottom: 6px; padding-left: 8px; }

- .margin{margin-top: 4px; margin-right: 5px; margin-bottom: 8px; margin-left: 10px;}
- .margin_shorthand4val{ margin:1px 2px 1px 2px; } /*1px from (top, bottom) and 2px (left, right)*/
- .margin_shorthand3val{ margin:1px 2px 1px;} /* Shorthand:1px from (top, bottom) and 2px (left, right)*/
- .margin_shorthand2val{ margin:1px 2px;} /* Shorthand:1px from (top, bottom) and 2px from(left, right)*/
- .margin_shorthand1val{ margin:1px;} /* Shorthand:1px margin from all sides, also known as group values */}

CSS
CALCULATING BOX SIZE

- **Box Width**
  Content width + padding-left + padding-right + border-left + border-right
- **Box Height**
  Content height + padding-top + padding-bottom + border-top + border-bottom



CSS: BOX MODEL
EXAMPLES

```
.boxModel_defaultSize{
font-size: 16px;
padding: 10px;
border: 2px;
margin: 5;
}
```

```
.boxModel_customSize {
font-size: 16px;
width: 350px;
height: 150px;
padding: 10px;
border: 2px;
margin: 5;
}
```

CSS
SIZE

- It is essential part of CSS layout, it defines how much space should an element cover on screen.
- There are different types of sizing schemes or units that CSS support.
- Broadly they are divide into two types;
  1. **Static (Absolute),** and
  2. **Dynamic (Relative and Viewport).**

CSS: SIZING
ABSOLUTE SIZING UNITS

- Absolute size units are like static or hard values that never change in any type of environment and screen size. Thus, with absolute unit, it require extensive work to explicitly define size of each element for every type of screen that fits best to it. The absolute sizing can be defined into various units, such as cm (centimeter), mm (millimeter), in (inch), px (pixel), pt (point), and pc (pica).

Unit	Description
cm	centimeters
mm	millimeters
in	inches (1in = 96px = 2.54cm)
px	pixels (1px = 1/96th of 1in)
pt	points (1pt = 1/72 of 1in)
pc	picas (1pc = 12 pt)

CSS: SIZING
RELATIVE SIZING UNITS

- Relative size units are dynamic values that applies in a hierarchical form. Means if size of root level or parent element changes then its descendant child elements adopt new proportional size automatically. It does not require extensive work to explicitly define size of each element for each screen.
- Please note that it does not ensure to fit every type of screen or device, but it make it easier to define size for multiple devices.
- The relative sizing can be defined into various units, such as em (emphemeral size), rem (root em size), ex (express size), ch (character size).

Unit	Description
em	Relative to the font-size of parent element
ex	Relative to the x-height of the current font (rarely used)
ch	Relative to the width of the "0" (zero)
rem	Relative to font-size of the root element

The em and rem units are practical in creating perfectly scalable layout.

CSS: SIZING
DYNAMIC SIZING UNITS

- Viewport size units are highly dynamic values that defines proportional part of screen. Means if size of root level or parent element changes then its descendant child elements adopt new proportional size automatically. It does not require extensive work to explicitly define size of each element for each screen.
- Also, it ensures that it fits to every type of screen or device, and it make it easier to define size for multiple devices. The viewport sizing can be defined into various units, such as vw (view width), vh (view height), vmin (view minimum), vmax (view maximum), % (proportion).

Unit	Description
vw	Relative to 1% of the width of the viewport
vh	Relative to 1% of the height of the viewport
vmin	Relative to 1% of viewport's smaller dimension
vmax	Relative to 1% of viewport's larger dimension
%	Relative to the parent element

Viewport = the browser window size. If the viewport is 50cm wide, 1vw = 0.5cm.

CSS POSITIONING

In presentation style, the position of elements appear to user is an essential part of interface design and layout. By default, every HTML element appear in left align and top to bottom flow format. Secondly, in every HTML page, the proportion of block elements with inline elements are much higher that covers about 80% by block element and only 20% by inline elements, i.e. 4:1 respectively. CSS allows to override these rules and present block element as inline block. Every HTML can be position on web page or screen in five (5) different types. Such as;

1. **Static:** D**efault or natural position of elements on the page**. Element appears where it call. The only reason to use static is to forcefully remove some positioning that got applied to an element outside of your control.
2. **Fixed: Removes elements from the flow of the HTML and allows them to be positioned anywhere. It** positioned relative to the browser window. Stayed where it call.
3. **Absolute: Removes the element from the flow of content and allows it to be positioned anywhere in relation to the HTML document.**
4. **Relative:** Position r**elative to the placement of the element within the** *flow* **of the document.**
5. **Sticky:** The element is positioned based on the user's scroll position. A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport, then it "sticks" in place like fixed.

CSS: POSITIONING
EXAMPLES

```
div.static {
  position: static;
  top: 10;
  padding: 5px;
  background-color: #cae8ca;
  border: 2px solid #4CAF50;
}
```

```
div.absolute {
  position: absolute;
  top: 30;
  padding: 5px;
  background-color: #cae8ca;
  border: 2px solid #4CAF50; }
```

```
div.fixed {
  position: fixed;
  top: 20;
  padding: 5px;
  background-color: #cae8ca;
  border: 2px solid #4CAF50;}
```

CSS: POSITIONING
EXAMPLES

```
div.relative {
  position: relative;
  top: 40;
  padding: 5px;
  background-color: #cae8ca;
  border: 2px solid #4CAF50;
}
```

```
div.sticky {
  position: sticky;
  top: 50;
  padding: 5px;
  background-color: #cae8ca;
  border: 2px solid #4CAF50;
}
```

CSS STACKING

Stacking is a vital property of visual effects and representation. It allows to presents multiple objects (elements) overlapping each other to show special effect, such 3D effects. The predefined staking order of HTML page element does not offer overlapping feature. The default document flow of HTML page is in vertical flow (from top to bottom direction), while some elements appears in horizontal stack with respect to their type. The default stacking order or z-index of HTML are same, except few positioning types.

Default Stacking Order

The following is the default staking order of browser when no z-index is applied to any element

- Root element (the <html> element)
- Non-positioned elements in the order they are called
- Positioned elements in the order they are defined

A non-positioned element is an element with the default position value static.
A positioned element is an element with any other position value such as absolute, relative, sticky or fixed.

CSS STACKING
WITH Z-INDEX

- Stacking can be defined by calling z-index property. Like many other things, CSS also allows to override the default z-index or stacking value by defining z-index property of element. Browser maintain the stacking layer of element with respect to their z-index value. Same z-index or stacking value create its own layer of stacking elements. Higher stacking value element(s) appears on top.

CSS STAKING WITHOUT Z-INDEX

- Yes, stacking can be done without using the z-index property, also known as auto stacking. This type of stacking is achieved by using the positioning property type. By default, the position type has priority of appearance order when they are called at some location. The browser when rendering the HTML page decides itself based on default properties which elements has higher staking order.

CSS: STACKING
EXAMPLES

```
div.stackOrder4 {
  top: 100;
  padding: 5px;
  background-color: #cae8ca;
  border: 2px solid #4CAF50;
  z-index: 4;
}
```

```
div.stackOrder5 {
  top: 120;
  padding: 5px;
  background-color: #cae8ca;
  border: 2px solid #4CAF50;
  z-index: 5;
}
```

```
div.stackOrder3 {
  top: 130;
  padding: 5px;
  background-color: #cae8ca;
  border: 2px solid #4CAF50;
  z-index: 3;
}
```

JAVASCRIPT

WHAT IS JAVASCRIPT?

WHAT JAVASCRIPT CAN DO?

- It can change all the HTML elements in the page
- It can change all the HTML attributes in the page
- It can change all the CSS styles in the page
- It can remove existing HTML elements and attributes
- It can add new HTML elements and attributes
- It can react to all existing HTML events in the page
- It can create new HTML events in the page

COMMENTS IN JAVASCRIPT

- **Single line comments - //**

- **Multi-line comments - /* comment here */**

JAVASCRIPT
VARIABLES ANATOMY

```
var iAmVariable = "Hello";
```

JAVASCRIPT
VARIABLES

There are 4 ways to declare a variable i.e. the var, let or const keywords, or without a keyword at all ("bare" declaration). Each determines resulting scope of the variable, or reassign ability in the case of const.

- **var — The most common variable.**
  Variables defined with **var** move to the top when code is executed. It can only be accessed within a function and can be reassigned. Also, it changes its data type w.r.t assigning data values. creates a function-scope variable.
- **const — usually use when require fix/unchanged values**
  Refers as the constant. Variable defined with const can not be reassigned and not accessible before they appear within the code. It either declare at the top or program or within specified function. creates a block-scope variable that cannot be reassigned
- **let — flexible constant variable commonly use in place of const**
  Similar to const, it allows the variable to be reassigned but cannot re-declared. creates a block-scope variable
- A bare declaration creates a global variable.

JAVASCRIPT
DATA TYPES

- **var** iamInt = 12;
  it defines *32-bit integer number ranging from -2,147,483,648 to 2,147,483,647*
- **var** iamLong = 9310141419482;
  it defines 64-bit number ranging from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
- **var** iamFloat = 5.5; *// it defines 32-bit floating-point number (decimal)*
- **var** iamDouble = 9310141419482.22; *// it defines 64-bit floating-point number*
- **var** iamBoolean = **true**; *// 1-bit Boolean data true or false, 0 or 1*
- **var** iamNotANumber = **NaN**; // non integer values
- **var** NaN_Example = 0/0; *// NaN: Division by Zero is not possible*
- **var** iamNotDefined; *// undefined: we didn't define it to anything yet*
- **var** iamString = "Hello World!" *// variable that's holds ASCII characters (A-Z, 0-9,!@#$, etc.)*

JAVASCRIPT
STRING & ITS OPERATIONS

charAt()	Returns a character at a specified position inside a string
charCodeAt()	Gives you the unicode of character at that position
concat()	Concatenates (joins) two or more strings into one
fromCharCode()	Returns a string created from the specified sequence of UTF-16 code units
indexOf()	Provides the position of the first occurrence of a specified text within a string
lastIndexOf()	Same as indexOf() but with the last occurrence, searching backwards
match()	Retrieves the matches of a string against a search pattern
replace()	Find and replace specified text in a string
search()	Executes a search for a matching text and returns its position
slice()	Extracts a section of a string and returns it as a new string
split()	Splits a string object into an array of strings at a specified position
substr()	Similar to slice() but extracts a substring depended on a specified number of characters
substring()	Also similar to slice() but can't accept negative indices
toLowerCase()	Convert strings to lower case
toUpperCase()	Convert strings to upper case
valueOf()	Returns the primitive value (that has no properties or methods) of a string object

JAVASCRIPT OPERATORS

Basic or Arithmetic Operators

+ **Addition**

- **Subtraction**

* **Multiplication**

/ **Division**

(...) **Grouping operator,** operations within brackets are executed earlier than those outside

% **Modulus (remainder)**

++ **Increment numbers**

-- **Decrement numbers**

Comparison Operators

=== **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**

? **Ternary operator**

JAVASCRIPT
OPERATORS

Logical Operators

- && Logical and
- || Logical or
- ! Logical not

Bitwise Operators

- & AND statement
- | OR statement
- ~ NOT
- ^ XOR
- << Left shift
- >> Right shift
- >>> Zero fill right shift

JAVASCRIPT
ESCAPE CHARACTERS

\'	Single quote
\"	Double quote
\\	Backslash
\b	Backspace
\f	Form feed
\n	New line
\r	Carriage return
\t	Horizontal tabulator
\v	Vertical tabulator

JAVASCRIPT DATE & TIME

Setting Dates

- **Date()**
  Creates a new date object with the current date and time
  y m d h m s ms
- **Date(2017, 5, 21, 3, 23, 10, 0)**
  Create a custom date object. The numbers represent year, month, day, hour, minutes, seconds, milliseconds. You can omit anything you want except for year and month.
- **Date("2017-06-23")**
  Date declaration as a string

Pulling Date and Time Values

- **getDate()**
  Get the day of the month as a number (1-31)
- **getDay() —** The weekday as a number (0-6)
- **getFullYear() —** Year as a four digit number (yyyy)
- **getHours() —** Get the hour (0-23)
- **getMilliseconds() —** The millisecond (0-999)
- **getMinutes() —** Get the minute (0-59)
- **getMonth() —** Month as a number (0-11)
- **getSeconds() —** Get the second (0-59)
- **getTime()**
  Get the milliseconds since January 1, 1970

JAVASCRIPT
ARRAYS

```
var iAmVariable = [“Hello”, “World”];
```

JAVASCRIPT
ARRAYS

- concat() — Join several arrays into one
- join() — Combine elements of an array into a single string and return the string
- indexOf() — Returns the primitive value of the specified object
- lastIndexOf() — Gives the last position at which a given element appears in an array
- pop() — Removes the last element of an array
- push() — Add a new element at the end
- reverse() — Sort elements in descending order
- slice() — Pulls a copy of a portion of an array into a new array
- splice() — Adds elements in a specified way and position
- sort() — Sorts elements alphabetically
- toString() — Converts elements to strings
- shift() — Remove the first element of an array
- unshift() — Adds a new element to the beginning

JAVASCRIPT
LOOPS

- **For loop**
  A standard for loop will execute until meet the final value/condition given is false.
- **For Of Loop**
  It will iterate over all items of collection without specifying initial, increment or final value as of in simple for loop.
- **For In Loop**
  It is iterating over object keys, not array indexes. Allows to access each property that belongs to an object
- **While Loop**
  A standard while loop will execute until the condition given is false.
- **Do While Loop**
  It will always execute at least one time, regardless of whether the condition is true or false:

JAVASCRIPT
LOOPS **LABELS**

- Continue
- Break (simple and named)

JAVASCRIPT FOR LOOPS EXAMPLES

Standard/Simple

```
for (var i = 0; i < 100; i++) {
console.log(i);
}
```

Stepping Increment

```
for (var i = 0; i < 100; i += 2) {
console.log(i);
}
```

On Array/Collection

```
var array = [‘Aslam', ‘babar', ‘chohan‘, ‘daniyal’];
for (var i = 0; i < array.length; i++) {
console.log(array[i]);
}
```

Decremented loop

- **for** (**var** i = 100; i >=0; i--) {
- console.log(i);
- }

JAVASCRIPT
LOOPS EXAMPLES

For…Of on Array Collection

```
var array = [‘Aslam', ‘babar', ‘chohan‘, ‘daniyal’];
for (let i of array) {
console.log(i);
}
```

For...Of on String

```
const string = "abc";
for (let chr of string) {
console.log(chr);
}
```

For..In Loop

```
var object = {"a":"foo", "b":"bar", "c":"baz"};
// `a` is inaccessible
Object.defineProperty(object , 'a', {
enumerable: false,
});
for (var key in object) {
if (object.hasOwnProperty(key)) {
console.log('object.' + key + ', ' + object[key]);
}
}
```

JAVASCRIPT
LOOPS WITH LABEL EXAMPLES

For with Continue Label

```
for (var i = 0; i < 3; i++) {
    if (i === 1) {
    continue;
    }
console.log(i);
}
```

While with Continue Label

```
var i = 0;
while (i < 3) {
    if (i === 1) {
    i = 2;
    continue;
    }
    console.log(i);
    i++;
}
```

JAVASCRIPT
LOOPS WITH BREAK EXAMPLES

With simple break

```
for(var i = 0; i < 5; i++){
    for(var j = 0; j < 5; j++){
    if(i == j) break;
    console.log(i, j);
    }
}
```

With named break

```
for(var i = 0; i < 5; i++){
nextLoop2Iteration:
    for(var j = 0; j < 5; j++){
    if(i == j) break nextLoop2Iteration;
    console.log(i, j);
    }
}
```

JAVASCRIPT
WHILE LOOPS EXAMPLES

Standard/Simple

```
var i = 101;
do {
console.log(i);
} while (i < 100);
```

Stepping Increment

- **var** i = 101;
- **do** {
- console.log(i);
- } while (i < 100);

On Array/Collection

```
var array = [‘Aslam', ‘babar', ‘chohan‘, ‘daniyal’];
while (i < array.length) {
console.log(array[i]);
i++
}
```

Decremented loop

```
var i = 10;
while (i > 0) {
```

- console.log(i);
- i--;
- }

JAVASCRIPT
CONDITIONAL STATEMENT

```
var animal = 'kitty';
var result = '';
if (animal == 'kitty') {
result = 'cute';
}
else {
result = 'still nice';
}
```

```
var animal = 'kitty';
var result = '';
if (animal === 'kitty') {
result = 'cute';
}
else {
result = 'still nice';
}
```

Ternary operators

```
var result = (animal === 'kitty') ? 'cute' : 'still nice';
```

JAVASCRIPT
SWITCH CASE

```
var value = 1;
switch (value) {
case 1:
console.log('Number is 1');
break;
case 2:
console.log('Number is 2');
break;
}
default:
console.log('Number does not match any case');
}
```

```
var animal = 'Lion';
switch (animal) {
case 'Dog':
console.log('The animal is "Dog" ');
break;
case 'Cat':
console.log(The animal is "Cat" ');
break;
default:
console.log('Animal does not match any case');
}
```

JAVASCRIPT
SWITCH CASE

```
function john() { return 'John'; }
function jacob() { return 'Jacob‘; }
switch (name) {
case john():
console.log('I will run if name === "John"');
break;
case 'Ja' + 'ne':
console.log(‘ The name is "Jane“ ');
break;
case  ‘Janifer‘+ john() + ' ' + jacob():
console.log('His name is equal to name too!');
break; }
```

```
var x = "c"
switch (x) {
case "a":
case "b":
case "c":
console.log("Either a, b, or c was selected.");
break;
case "d":
console.log("Only d was selected.");
break;
default:
console.log("No case was matched.");
break; }
```

NOTE: A case expression can be any kind of expression. It means you can use comparisons, function calls, etc. as case values.

JAVASCRIPT
FUNCTIONS

```
function foo(fname, lname) {
var name = fname + ‘ ’ +lname;
console.log(“Hello”, name);
}
```

JAVASCRIPT
OBJECTS

```
var person = {
firstName:"John",
lastName:"Doe",
age:20,
nationality:"German"
};
```

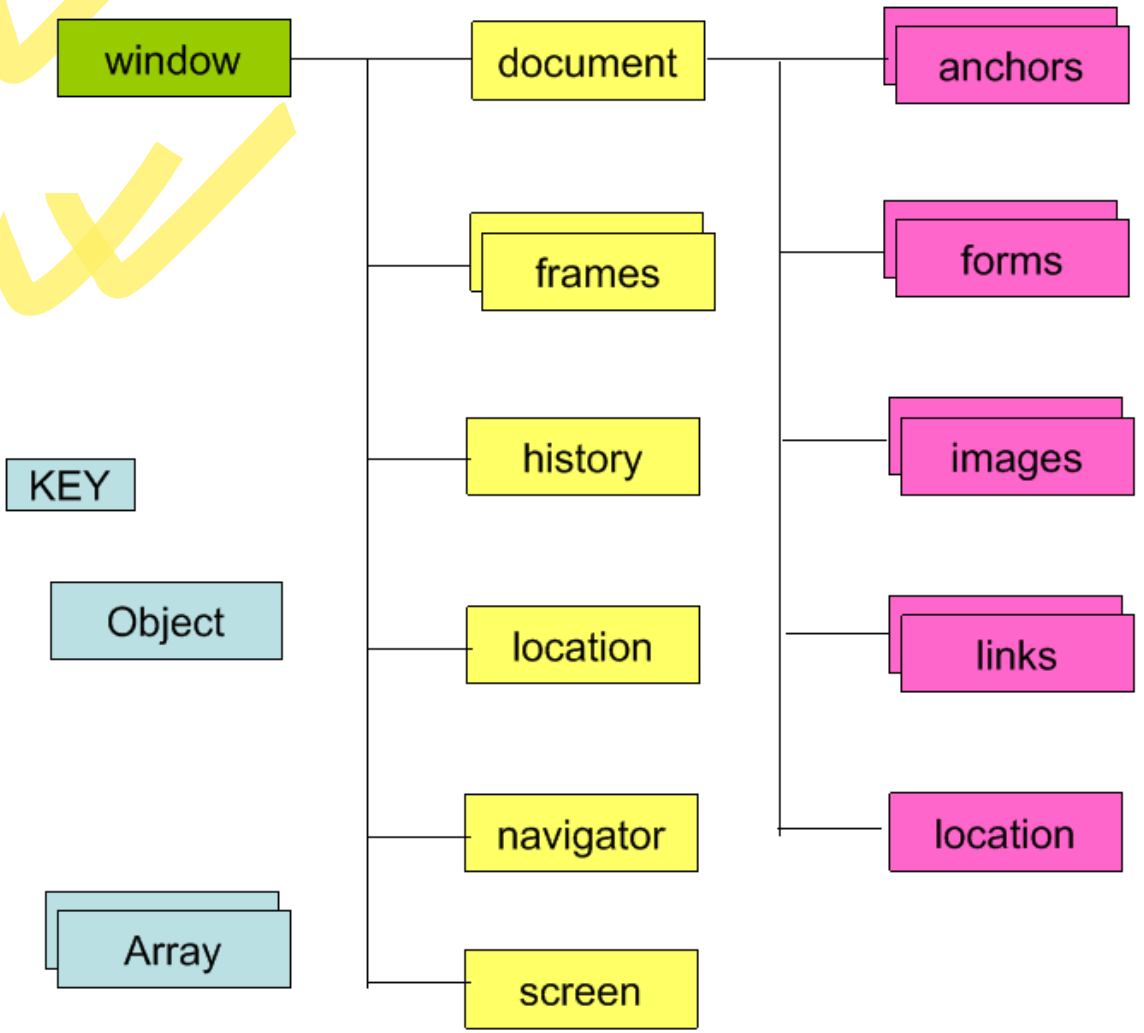
JAVASCRIPT
DATA INPUT & OUTPUT

- **console.log()**
  Writes information to the browser console, good for debugging purposes
- **document.write()**
  Write directly to the HTML document
- **window.alert()**
  Output data in an alert box in the browser window
- **window.confirm()**
  Opens up a yes/no dialog and returns true/false depending on user click
- **window.prompt()**
  Opens up a pop-up dialog box for taking user's input data.

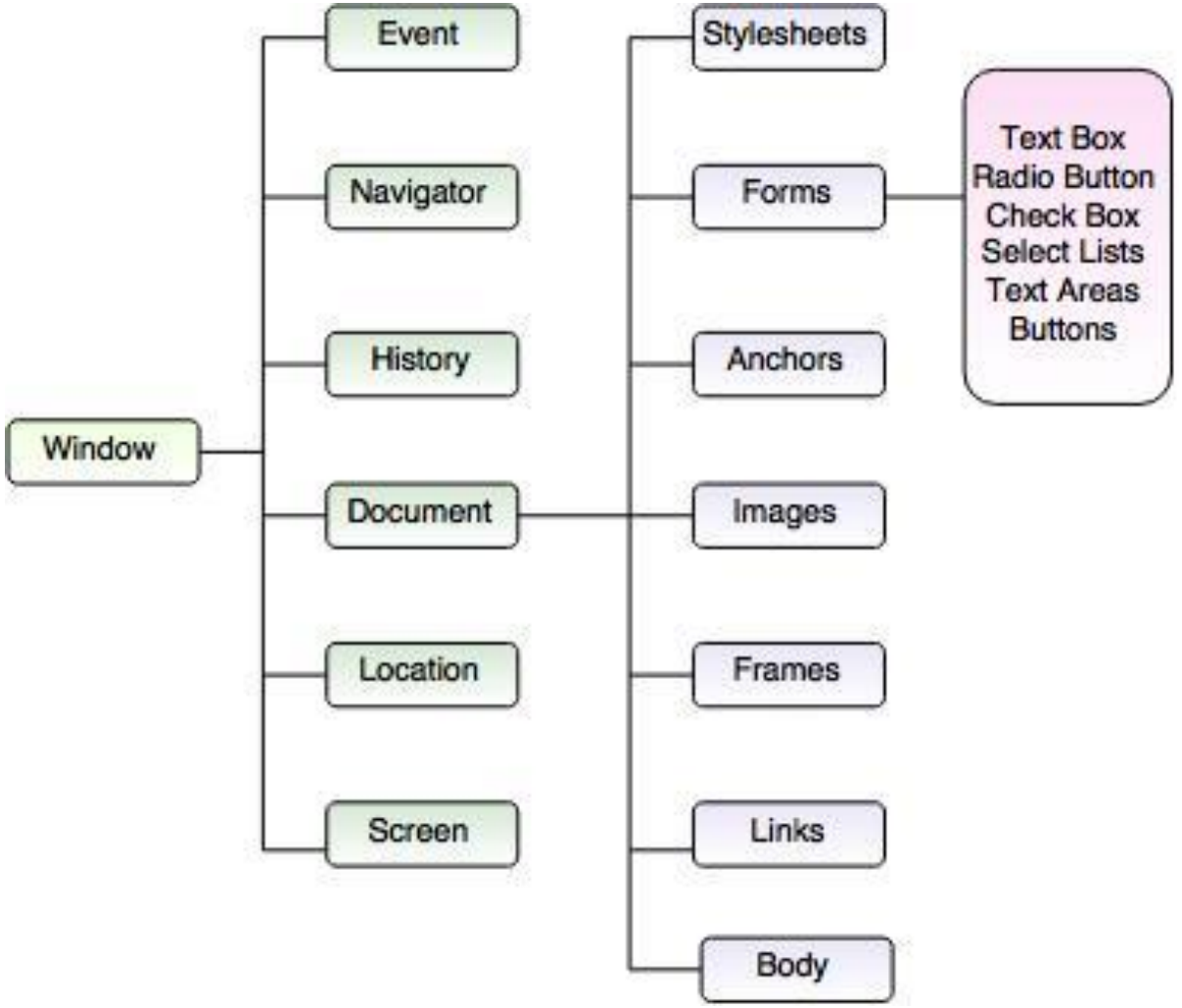
JAVASCRIPT
BOM OBJECTS

The BOM (Browser Object Model) contains objects that represent the current browser window and components; objects that model things like *history, device's screen*, etc.

- **Document:** represents current web page.
- **History:** represents pages in browser history.
- **Location:** represents URL of current page.
- **Navigator:** represents information about browser.
- **Screen:** represents device's display information.



JAVASCRIPT
WOM



JAVASCRIPT
WOM

- window.alert() Creates dialog box with message and an OK button
- window.blur() Remove focus from window
- window.close() Closes a browser window
- window.confirm() Creates dialog box with message, an OK button and a cancel button
- window.open() Opens new browser window with URL specified as parameter
- window.print() Tells browser that user wants to print contents of current page
- window.prompt() Creates dialog box for retrieving user input
- window.stop() Stop window from loading

JAVASCRIPT
DOM

The DOM is a W3C (World Wide Web Consortium) standard defines for accessing documents elements.

The standard is separated into 3 different types;

- Core DOM - standard model for all document types
- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents

The HTML DOM is a standard **object** model and **programming interface** for HTML defines:

- The HTML elements as **objects**
- The **properties** of all HTML elements
- The **methods** to access all HTML elements
- The **events** for all HTML elements

The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

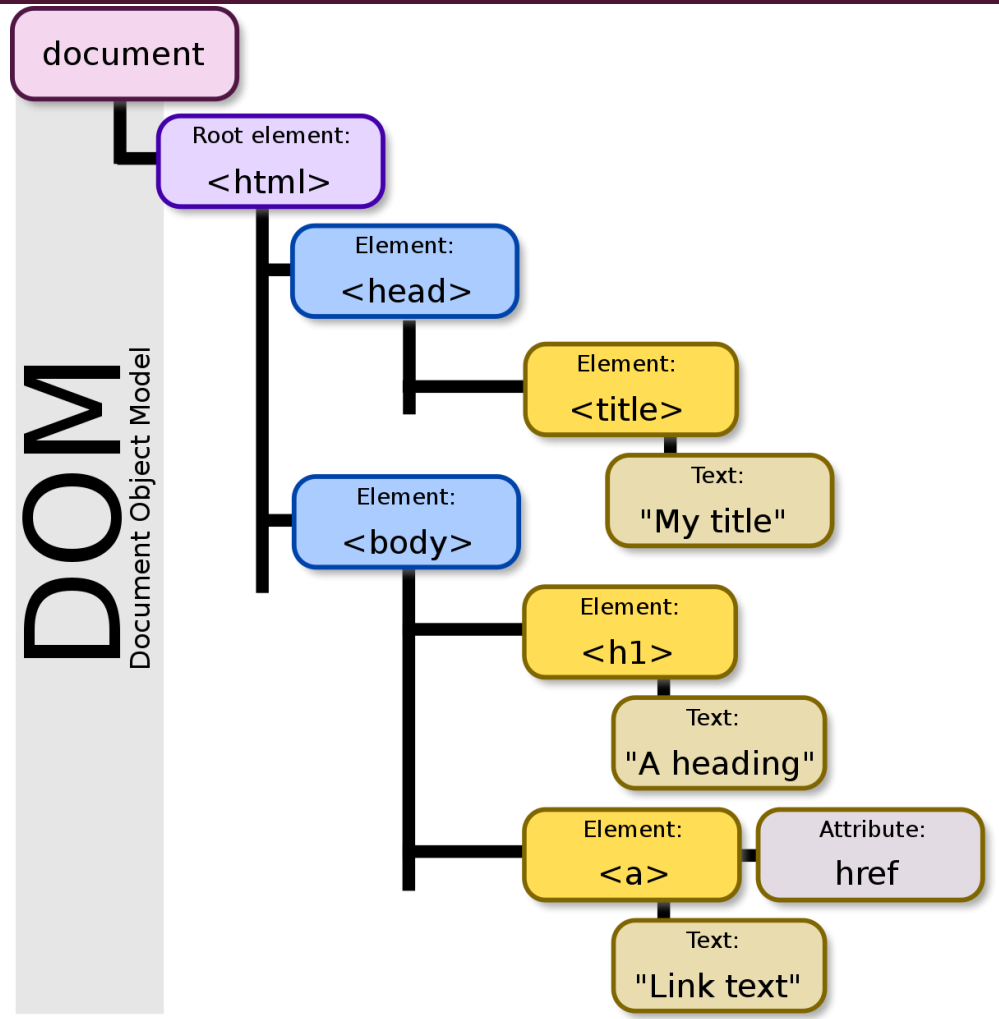
JAVASCRIPT
DOM: METHOD AND PROPERTIES

3 Methods

- **Find an element by element id**
  document.getElementById(id)
- **Find elements by class name**
  document.getElementsByClassName(name)
- **Find elements by tag name**
  document.getElementsByTagName(name)

3 Properties

- **Change the inside of an element**
  element.*innerHTML* = new content
- **Change the attribute value of an element**
  element.*attribute* = new value
- **Change the style of an element**
  element.style.*property* = new style



HOW YOU CAN USE IT?

- **Internal JS**

```
<script type="text/javascript">
Console.log("Hi, I am internal JS");
</script>
```

Internal JS can be called anywhere in HTML page.

- **External JS**

```
<script src="myscript.js"/>
```

EXAMPLES

- var introID = document.getElementById("intro");
- var pTag = document.getElementsByTagName("p");
- var infoClass = document.getElementsByClassName(“info");
- var pIntroCSS = document.querySelectorAll("p.intro");

JAVASCRIPT

Practice

ASP.NET MVC

REFER TO THE LECTURE UPLOADED AT GOOGLE CLASSROOM