HTML Concepts

Table Of Content :

What is HTML

- HTML stands for Hyper Text Markup Language . It's the standard markup language (لغة وصفية) used to create web pages and applications
- HTML tells the web browser how to display content, including texts, images, and other forms of multimedia

Note :

HTML is **NOT** a programming language, it's a markup language used to structure content in the web

Why Learn HTML

- **Foundation of web development** : HTML is the **BUILDING BLOCK** of all web development, it is essentiel for web design, web development, and web maintenance (صيانة)
- **Versatility** (متعددة الاستخدامات) : It can be used with CSS and JavaScript to create dynamic web pages

Key Components of an HTML Document

- **<!DOCTYPE>** Declaration : Specifies the document type and version of HTML For HTML5, we use <!DOCTYPE html>.
- **<html>** Element: The root element that contains **all other HTML elements**.
- **<head>** Section: Contains **meta-informations** about the document, such as the title, character set, stylesheets, and other resources.
- **<body>** Section: Contains the **content of the web page**, such as text, images, and other media

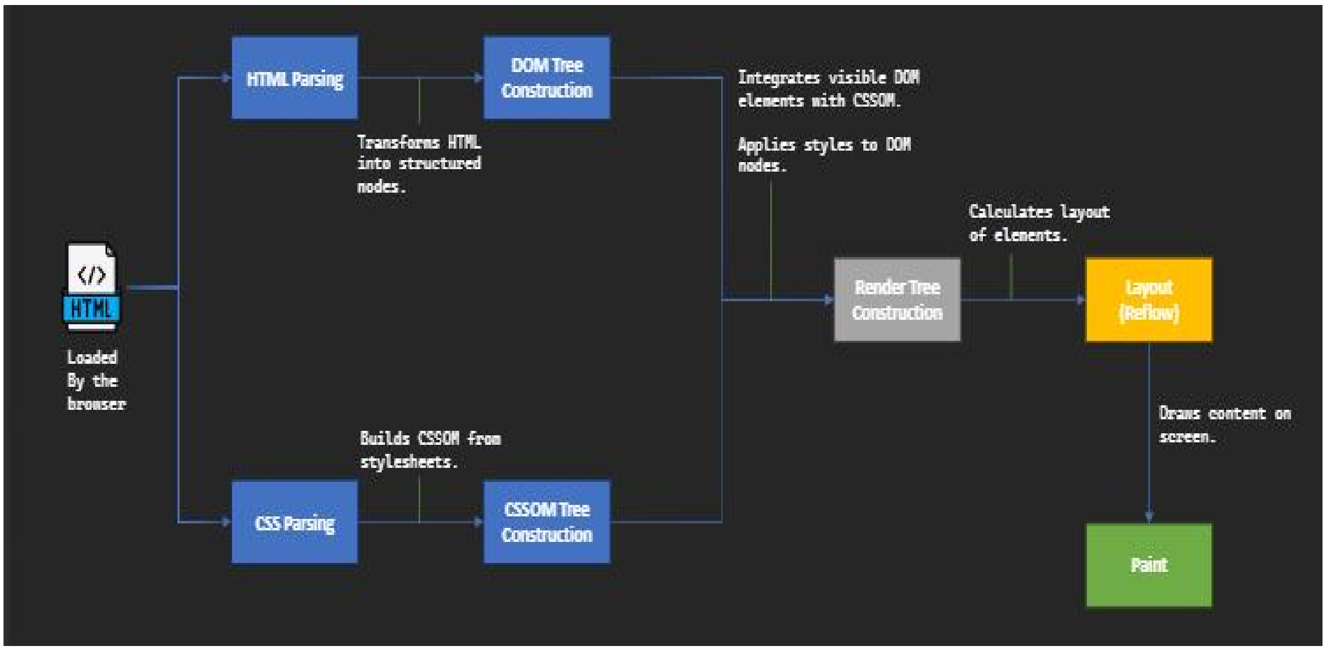
Relationship between HTML, CSS & JavaScript

We have 3 Core Web Technologies :

- HTML (Hyper Text Markup Language): The backbone of any web page, responsible for **structuring content**
- CSS (Cascading Style Sheets) : Defines, the presentation , formatting & layout
- JavaScript : Adds **interactivity** to web pages, works with both HTML & CSS to create a complete web page experience

So in short :

- HTML : Provides the **content structure**
- CSS : **Styles** the content
- JavaScript : Makes the content **interactive** يجعل المحتوى تفاعليًا


HTML Parsing
DOM Tree Construction
Transforms HTML into structured nodes.
Integrates visible DOM elements with CSSOM.
Applies styles to DOM nodes.
Calculates layout of elements.
HTML
Loaded By the browser
Render Tree Construction
Layout (Reflow)
Draws content on screen.
Builds CSSOM from stylesheets.
CSS Parsing
CSSOM Tree Construction
Paint

How Browsers Render HTML ? No JavaScript

كيف يقوم المتصفح بعرض ال HTML

Step 1 : HTML Parsing (تحليل كود ال HTML)

- **Process :** When a browser loads an HTML document, it reads or « **parses** » the **HTML code** to understand the **structure** and the **content** of the web page
- **Outcome (Result) :** The browser converts HTML tags into DOM (Document Object Model) nodes, resulting a « DOM tree »
  Note : The browser converts HTML into **Tree data structure** because it shows the relationships between tags

```
Document
|
└── DOCTYPE: html
|
└── html (lang="en")
```

```
    │
    ├── head
    │   ├── meta (charset="UTF-8")
    │   ├── meta (name="viewport", content="width=device-width, initial-scale=1.0")
    │   ├── title: "This is my page title"
    │   └── style: [CSS rules]
    │
    └── body
        ├── h1: "Welcome to HTML"
        ├── p: "This is my [b: 'First'] paragraph."
        ├── p: "This is my [b: 'Second'] paragraph"
        └── p: "This is my third [b: 'paragraph']"
```

In this tree, each HTML tag is represented as a node with parent-child relationships mirroring the HTML's nested structure.

Step 2 : CSS Parsing (CSS تحليل كود ال)

- **Process :** Like HTML, the browser in the same time parses the **CSS code** to determine the **styling of various HTML elements**
- **Outcome (Result) :** The browser converts CSS informations into CSSOM (CSS Object Model) nodes, resulting a « CSSOM Tree »

```
CSSOM Tree
│
└── style rules
    │
    ├── body {font-family: Arial, sans-serif; background-color: #f0f0f0;}
    ├── h1 {color: blue;}
    ├── p {color: #333; font-size: 16px;}
    └── b {color: red;}
```

- So now we have **two seperated Trees** one for HTML structure and the other for CSS styling

Step 3 : Constructing the Render Tree تصميم شجرة العرض

- **Process :** The browser now combines the DOM Tree and the CSSOM Tree to form the **Render Tree** which represents the visual layout of the web page. **Only elements that are actually visible** (those that affect the layout التصميم and not set to display : none) are included
- **Outcome :** The render tree includes all visual elements of the page, like text and colors, all according to CSS rules

DOM Tree + CSSOM Tree = Render Tree

```
Render Tree
|
└── body (font-family: Arial, sans-serif; background-color: #f0f0f0)
    |
    ├── h1 (color: blue): "Welcome to HTML"
    ├── p (color: #333; font-size: 16px): "This is my [b (color: red): 'First'] pa
ragraph."
    ├── p (color: #333; font-size: 16px): "This is my [b (color: red): 'Second'] p
aragraph"
    └── p (color: #333; font-size: 16px): "This is my third [b (color: red): 'para
graph']"
```

Not in the RenderTree :

- Non Visual Elements : Head, Title …etc
- Nodes Hidden via Display : None

Step 4 : Layout Process عملية التصميم

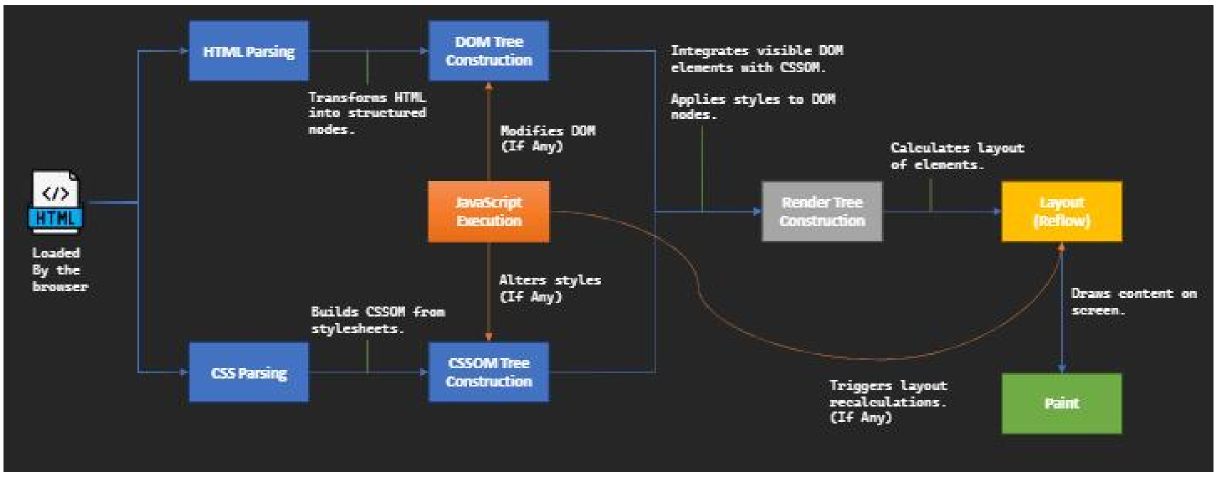
- **Process :** The browser **calculates** the exact position and size of each object on the page based on **DOM**, this process known as «**layout**» or «**reflow**». So layout process specialized **only** in calculating sizes & position of elements
  This process can be affected by JavaScript, if the script contains the geometry of elements (like changing the size or position of elements)
- **Outcome :** Determines how elements are spatially positioned on the screen

Step 5 : Painting

- **Process :** The final step is painting, where the **render tree** is converted into actual pixels on the screen.
- **Outcome :** The visual representation of the page is displayed to the user

How Browsers Render HTML ? With JavaScript

كيف يقوم المتصفح بعرض ال HTML



Step 1 : HTML Parsing (تحليل كود ال HTML)

Step 2 : CSS Parsing (تحليل كود ال CSS)

Step 3 : Executing JavaScript

- **Process :** JavaScript execution can occur during initial **parsing** (HTML & CSS) if scripts are synchronous متزامن or after HTML parsing if scripts are asynchronous غير متزامن
- JavaScript can modify both the **DOM** & **CSSOM** during or after their construction. So in parsing it's preffered to **NOT** have JavaScript Code, it's better to include it **after parsing**
- **Outcome (Result) :** JavaScript may add, remove or modify elements in the DOM which may necessite recalculating the CSSOM and re-rendering the Render Tree
- JavaScript may affect also **Layout changes** or even **repaints** depending on the nature of the DOM manipulation

Step 4 : Constructing the Render Tree تصميم شجرة العرض

Step 5 : Layout Process عملية التصميم

Step 6 : Painting

Important Note on JavaScript's Impact :

Because JavaScript makes the content interactive that means the content can be changed in the run time, that can cause :

- **Reflows & Repaints :** JavaScript can cause performance issues if not handled correctly, as recalculating the CSSOM it can lead to frequent reflows and repaints

Note :

- Efficient JavaScript coding practices is **essential** to ensure smooth, efficient rendering by the browser

What are Heading Elements :

What are Headings : العناوين

- Headings are HTML elements designed to organize the content by defining **titles** and **subtitles** on a web page
- Hierarchy of headings : HTML provides 6 levels of headings, **<h1>** is the most bigger one, down to **<h6>** which is the smallest one
- Headings play a big role in **optimizing Search Engine (SEO),** by adding **ONLY** a **single** <h1> tag per page to define the most important title . This tag represents the central topic and increase the website visibility
- Headings break down content into manageable sections making the web page easier to read

Note :

1. Always **maintain the logical order** without skipping heading levels to preserve content structure
2. Do **NOT** use <h1> tags for different sections in your webpage, because it can confuse both users and search engines about the structure and the importance of the content
3. Use HTML headings only for headings. Do not use headings to make text BIG or **bold**

- Almost all elements in HTML (<h1> <p> ….) have their properties(attributes - خصائص), for example in headings elements we have the **style attribute** that contains : font – color ….

What is Paragraph :

- The HTML <p> tag is a **block level element** (starts a new line + takes the full width) that defines paragraphs which improves the overall structure of the page

Note :

- HTML paragraphs **ignores** all the whitespaces, but having a lot of them will **slows** HTML document so you should always **avoid unecessary whitespaces**
- **NEVER** use 2 nested paragraphs because the browser will automatically close the outer paragraph that consumes **all the width** so here the browser opens the inner paragraph which leads to errors

What is Break :

- The HTML <br> tag is an element used to insert line breaks in text
- Unlike paragraph tags, <br> tags does **not** create any additional margin around the break, that means <br> tags do not take all the width like paragraphs so <br> tags are **Inline elements**
- The <br> tag is an **empty element / tag** that means it does not have a closing tag

Note :

1. <br> tags should be used **wisely (بحكمة),** because excessive use of <br> tags does not add any semantic meaning to the text, unlike paragraph tags <p> which indicate a block of **related content**, so using paragraph tags is **better** for content meaning, both to users & web technologies such as engines & screen readers resulting improving both readability and optimizing search engines
2. <br> tags does **not** have any attributes

- To create a space between paragraphs use CSS properties like margin or padding on <p> tags instead of using <br> tags to make the content semantically correct

- Using <p> tags rather than multiple <br> tags to seperate blocks of text is generally considered better practice for several reasons :

1. **Semantic Meaning** : the <p> tag provides semantic meaning to the text, this helps search engines and screen readers to understand the structure of the content more effectively
2. **Maintaining Document Structure** : Paragraph tags helps to maintain a clear structure of the document. This structure helps both users and developers to understand how content is organized where each <p> tags represent a **block level element**, signifying a related section of text, unlike <br> tags that simply break lines without indicating any structural division
3. **Styling and CSS** : when using <p> tags it easier to apply **CSS styles** such as padding, margin, line-height, text alignment and other typographical styles. Unlike <br> that does not provide any attributes which can lead to maintenance challenges.
4. **Readability and maintenance** : Code readability improves noticeably when using <p> tags because the purpose of each section of text becomes clearer, this is very helpful when maintaining a website, as developers can quickly understand how text is grouped and structured. The excessive use of <br> tags can make the HTML document hard to understand, making it difficult to maintain
5. **Avoiding Bad Practices** : Using <br> tags between paragraphs is considered a **bad practice** because it mixes content with presentation. The seperation of content from presentation is a **fundamental web design principle** , achieved by using HTML elements for structure like <p> tags for paragraphs and CSS for presentation like spacing and layout

What is Preformatted tag :

- The HTML <pre> tag is used to display preformatted text
- Unlike <p> tags, preformatted tags **preserves both** spaces and line breaks in the text, making it very useful when formatting is **important**
- <pre> tag is a **block level element** that means it start a new line and takes the full width
- The HTML <pre> tag can contain only **inline elements**, not block-level elements

Disadvantages of using <pre> tags :

- **Limited Styling :** the <pre> tag preserves whitespaces and line breaks which can make difficult to apply some CSS styles
- **Accessibility :** preformatted text may not be accessible to screen readers, it can be harder for users with disabilities to navigate and understand the content.
- **Content Overflow فيض المحتوى :** If the content within the <pre> tag is too long, it may cause horizontal scrolling or overflow issues, especially on smaller screens.

When <pre> tag is useful for preserving formatting, it's important to consider these disadvantages and use it wisely.

What are Horizontal Rules :

- The HTML <hr> tag stands for « Horizontal Rule » it's used to create a break between paragraph-level, it's by default represented as a **horizontal line**
- It's a versatile tag (متعددة الاستعمالات) that helps to divide content sections on web pages
- The <hr> tag is a **self closing tag (empty tag)** that does not contain any content and **cannot** contain other HTML elements
- The <hr> tag is a **block level element** that means it start a new line and takes all the width
- It's can be styled using CSS to change it appearence, such as it height (thickeness), width, color and style of the line.

What is Span Element :

- The HTML <span> tag is a **versatile inline element** used primarily for styling or **marking up** a part of a text while **keeping** the semantic meaning
- The <span> tag serves as a **small container** for styling
- Unlike block level elements, which starts a new line and takes the full width, the <span> tag is an **inline element**, this means it does **not** cause a line break, and only takes as much width as needed.
- It useful when there is a need to apply **CSS styles or JavaScript actions** to a part of the text

- We can use **ONLY** inline HTML elements Inside <span> tags, it obviously because <span> tag is an inline element.
- Unlike headers, the <span> element is a semantic-**neutral** meaning, it represent any additional semantic meaning on it own, it is just a **silent container**
- The <span> tag is **NOT** a replacement for tags, while it used for small scale styling within the text, **<p>** tags are primarly used to define paragraphs and provide semantic meaning to text blocks

What is Div Element :

- The HTML <div> tag stands for Division or Divider, it a versatile **block level element** used in web design that means the **layout** of the web page
- <div> element is a **BIG container**, we can use it for structuring and grouping a section of HTML elements in the web page and applying css styles to each group
- Div element is a semantic **neutral** meaning, it does **not** represent anything on it own but serves as a **container** for other HTML elements
- We can use almost **ALL** other HTML elements Inside <div> tags like paragraphs, links… because <div> is a block level element, also we can use another <div> tags inside the main <div> tag
- We can add « id » property to **ALL** HTML elements , it useful when **javascript** want to access and modify the content of an HTML element

  Here, the most important question….

What is the difference between <span> element and <div> element ?

<span>	**<div>**
Inline element	Block level element
Contains ONLY inline elements	Contains almost ALL HTML elements
Small Container	Big Container

- **In short, we use <div> tag for structural purposes that requires block level divisions, and <span> tag for inline styling of the text without affecting the overall layout of the web page**
- The **ONLY** thing that **<span>** and **<div>** elements share is that they are **BOTH** semantic **neutral** meaning

HTML Comments :

- HTML comments provides a way for developers to include notes, explanations and disable code temporarily without deleting it
- HTML comments are marked by **< !-- -->** anything placed between these tags will not appear in the browser and does not affect the web page functionality
- Especially in HTML, comments **SLOWS** the speed of the web page. Use comments **ONLY** if there is a **strong** need to type a comment because they take space.
- Comments should be **meagniful** and **minimal** because HTML is a downloaded document by the browser so every **comment** and **white space** **increase** the file size that affect the load time of a web page especially in environments with **slow connection**
- For very hight traffic **websites** or **applications** where every byte counts , consider to stripping (removing) comments in the production version of the HTML files, if comments are VERY neccesary try to use **compression algorithms** such as **GZIP** or **Brotli** which are very effective at reducing the size of the text-based files including HTML, these technologies are very useful when compressing HTML content, making the size added by comments even **less significant**
- To comment out a line just press on **Ctrl + /** and **Ctrl + /** again to uncomment it

HTML <b> vs <strong> :

- The HTML **<b>** and **<strong>** tags are both used to make text visually **bold,** but they have different semantic meanings
- For **<b>** tag, it purely used to get the text bold that does **not** have any semantic meaning
- The **<strong>** tag is used to make text visually bold and to indicate that the words between <strong> tags have a **strong importance, seriousness and urgency** that means **<strong>** tag have a **semantic meaning** that means screen readers will read text between <strong> tag in a stressful way

HTML italic <i> vs emphasize <em> :

- The HTML **<i>** and **<em>** tags are both used to make text visually *italic,* but they have different semantic meanings
- <i> tag stands for italic it's used to display text in an *italic* style, with a **neutral semantic** meaning
- <em> tag stands for emphasize (تاكيد) text, it used to make text italic and it's have **stress** meaning that means screen readers will read it in a stressful way on the words between <em> tag but less than <strong>, so <em> tag have a **semantic meaning**

HTML Underline <u> tag :

- The HTML <u> tag stands for underline, it's used to underline text
- Traditionally <u> tag was used to emphasize (تاكيد) text, but it usage was involved to just make the text underlined, so <u> tag does **NOT** have any semantic meaning

HTML <small> tag :

- The HTML <small> tag is used to **decrease** the font size of the text in HTML documents, but it purpose extends beyond just visual presentation
- <small> tag meant to indicate that the text is **less important** than other texts of the web page such as legal text, disclamers, copyright informations or any fine details that are not the main focus of the web page
- While <small> tag modifies the visual appearance of text, it also carries **semantic meaning,** we can use <span> tag with small font size to display text smaller but the text will **NOT** have any semantic meaning

HTML <mark> tag :

- The HTML <mark> tag is used to highlight text within a document
- It useful for drawing attention to specific parts of text throught a background color (yellow by default), it essential in a particular context such as during searches

HTML <del> vs <ins> :

- The HTML <del> and <ins> tags are used to indicate text modifications in a document
- **<del> tag :**
  1. Stands for **deletion**, it's used to mark text that has been removed from the document
  2. It's displayed with a ~~strikethrough~~, indicating a deletion
- **<ins> tag :**
  1. Stands for **insertion**, it's used to indicate text that has been added to a document
  2. It's displayed with an underline, highlighting the addition
- Both **<del>** and **<ins>** tags affect visual presentation of the text and provides a way to document changes, making them useful in edits, updates, and corrections. Also, they provide a **semantic meaning** that can be useful in search engines

HTML <sub> vs <sup> :

- The HTML **<sub>** tag stands for **subscript**, while **<sup>** tag stands for **superscript**
- **<sub> tag :**
  1. It's used to create **subscript** text, which appears slightly **below the baseline** of the normal text line like : H_2O
  2. It's ofen used in chemical formulas or mathematical expressions
- **<sup> tag :**
  1. It's used to create **superscript** text, which appears slighly **above the baseline** of the normal text line like : 1^{st}
  2. It's often used in mathematical exponents (x^2) or references in documents
- These tags **affect the line height slightly** but are crucial for presenting academic contexts where formatting is necessary to **convey** (give) informations correctly.

HTML <img> Tag :

- The HTML <img> tag is essential to illustrates images in the browser
- When we get a picture from internet we call it : absolute url / external url
- <img> have it's attributes / properties like src = source & alt = alternate
- Alternate is the desription of the image
- Alternative is also used to help blind people by adding the description of the image so the screen readers can read the description so blind people can imagine the illustration of image
- Title attribute gives you extra informations about image by using a tool tip text
- Alternate attribute shown **ONLY** when image is **LOST** with a placeholder of image
- We can upload pictures using Relative path (images that are located in the same machine : Images/Image.JPG) or by external URL
- Always set width and height bcs if you don't set w & h the browser load the image with default size
- The **default** size unit in images is px so 50px is same as 50
- Image Element is an **inline element**(Non-Block Level Elements)**,** that means we can put it inside a paragraph, list, cell inside a table **…** otherwise you must set a <br> line to put images line by line
- The images Folder **MUST** be in the same folder of the HTML document So they can be shown in the browser, otherwise only the **placeholder** will appear with the alternative
- In **HTML**, Units are **implicitly pixels (px),** we don't need to set the unit So height = 500 px is same as height = 500, while in CSS we **MUST** specify the unit (px / %), so if we write height = 500 is **NOT** valid in CSS and the browser will ignore it instead write : height = 500px
- If we want for example the quarter size of the original picture we can just type in width property (width : 25%) and to keep the aspect ratio you just write in the height (height : auto) so the browser will automatically calculates the aspect ratio all depending on the width
- <img> does not have a self closing tag it's used to display various image formats such as : JPEG, PNG, GIF….

How Browsers deal with images ?

Detailed Process to Download images :

1.Parsing HTML :

- While the browser parses the HTML document it's constructs the DOM tree
- During the HTML parsing, when an image tag with a source attribute found, the browser recognizes that this image needs to be fetched(downloaded). The parsing continues while these resources are being fetched

2.Sending Request :

The browser puts the images in a queue and sends an HTTP request for each image, then he opened a new thread to download the image in the background to initialize the download of image ressources needed for a webpage, **ASYNCHRONOUSLY** he continues parsing the HTML document

3.Downloading Image :

- Image files are downloaded in the background, allowing the browser to continue with other tasks such as parsing the entire HTML document, loading CSS and JavaScript
- The downloading process does not block the browser from performing these other tasks unless spesifically instructed to do so, for example you can configure certain tags to instruct the browser to not complete rendering until you download this (image/video…)

4.Rendering Image :

- The rendering of the image happens after the image data has fully received and decoded(converts image from binary to image again)
- It's recommanded to set the dimensions of the image (width & height) so the browser preserves the space in the web page, so when the image is downloaded, the browser put it in it preserved space to prevent **cumulative layout shifts CLS** (the movement of the content in the web page from place to place) so Lower CLS = smoother user experience

5.Cashing :

- All browsers have cashing policy, so while cashing does not directly impact the rendering of the web page on the first visit, it significantly speeds up image loading for next visits, because images were cashed in the **local drive** (the browser stores them in a compressed binary format not as normal .jpg or .png files) instead of bring them from servers, so the browser uses the cached copy from the local disk that why cashing makes websites **load faster**

- From time to time, the browser refresh the cash memory depending on the cash policy

See this path on your laptop :

C:\Users\pc\AppData\Local\Google\Chrome\User Data\Default\Cache\Cache_Data

- Rendering the web page to ensure the web page becomes visible to the user as soon as possible, so the web page is displayed gradually to the user
- So for each image founded, the browser makes HTTP request, that means if the browser founds **10 images in the HTML document**, the browser makes **10 http's requests**, each one is seperateD from the other

ASYNCHRONOUS Rendering and Loading : العرض المتزامن و التحميل

- The key to modern web browsers performance is their ability to do many of these tasks **ASYNCHRONOUSLY** :
    1. **Non-Blocking Ressources :** Images are loaded in a way that **does NOT block the rendering of other elements whatever the size of the image is,** browsers can handle various resources like (Images – Videos – JS file – CSS file…) so when we have **block level elements**, we download them **first** then we download the inline elements
    2. **Progressive Rendering :** Browsers try to render progressively, which means they display parts of the webpage as soon as they become ready.This approach improves the user experience by making the page seem faster and responsive as soon as possible

IMPORTANT NOTES :

- Always set the dimensions to the <img> tag to **prevent layout shift .**
- The images does not affect the rendering process whatever the size of the image is, **BUT they affect the speed of downloading the image**, so always use image compression that does not affect the quality of the image
- Image optimization involves compressing the image file size without significantly degrading its quality, so converting images on the right format, and loading them based on the user device, is essential in web development because speeds up web page loading times

and improves the overall performance and user experience

- Images Formats :
  .PNG images size are way bigger than .JPG images and .SVG is the smallest one PNG > JPG > SVG
  So always use the right image format sizes, because in finally whatever you chose whether it was the image format or size will works BUT always think of the performance of the web page
- Its **NOT** a recommand practice to use vector images for icons and simple graphics when using images in the web design
- To improve web page load time, it's **HIGHLY RECOMMANDED** to compress the size of the image to reduce the amount of data transferred over the network
- The primary reason for converting simple images such as icons into SVG format is for hight performance without loss of quality
- For more speeding images in the web page, it's recommanded to use Content Delivery Network (CDN) like Cloud Flare to enable it in your website. A real Example :
  You're in Morocco , your website can have many users from different countries so the benefit of CDN is that he stores copies of the images on geographically distributed servers, it automatically happens when subscribing with Cloud Flare
  So CDN speeds the web page loading , so the images are delivred from the closest server to the user's location, so that enhance the overall user experience

Other Benefits of using CDN :

- Cashes images
- Handle images size that means it giving you the most optimized size for each image

The browser in HTTP/1.1 must send HTTP request for each image, so the benefit of using HTTP2 is, when hosting your website and use HTTP2 protocol, this allows you to download more than one image in just one connection so this is a significant improvement of HTTP1.1 that obligate you to have a seperate connection to each image so **10 images = 1 connection** without it **10 images = 10 connections** that is called **request multiplexing (تعدد الإرسال في الطلبات)**

In short :

- In **HTTP/1.1**, if your webpage has 10 images : the browser opens **multiple separate connections** to download them.
- In **HTTP/2**, the browser opens **only one connection**, but it can **send and receive all 10 image requests simultaneously** (معا) through that single connection, this is called **request multiplexing**. It's **very useful** with websites with many **assets**(images)

So using CDN with HTTP2 is way better for the web page performance

- Efficient use of asynchronous loading, careful management of resources priorities and the best use of images are the fundamental basics of modern web development

Lazy Loading in HTML

Lazy Loading = تاجيل تحميل الصور حتى الاقتراب الى نطاق الرؤية

- The loading = "lazy" attribute in HTML is a native browser feature that enables lazy loading of images and iframes
- The Lazy Loading is important for improving the webpage load times so instead of sending 10 HTTP requests for 10 images, the browser sends only 2 HTTP requests for the first 2 images then as much as the user scrolls down the other images begins to download, this enhance user experience by deferring (تاجيل) the loading of off-screen resources(the images that does not appear in the screen yet)
- Lazy Loading is a **design pattern** that focuse on loading resources until they are needed
- The lazy attribute tells the browser to not download the images only if the user **scrolls to it** or the images are **about to enter to the view port** (نطاق الرؤية), this reduces the initial page load time (وقت تحميل الصفحة الأولي)

Benefits of using Lazy Loading :

- **Improved Performance : Reduces** the initial load time of the web page
- **Reduce Bandwidth usage :** Saves data for users and server resources by loading **fewer** resources
- **Enhance user experience :** Allows users to interact (يتفاعل) with the visible content **faster** while off screen images load as needed

Best Practices :

- Use loading = "lazy" for images and iframes that are below the fold (خارج نطاق الشاشة)
- Ensure that the first images that appears in top area are **NOT** lazy loading, as this could delay (تأخير) their visibility
- **Use loading = "lazy" for :**
  1. **Images that are rarely viewed by users**
  2. **Images in a hidden tab**
  3. **Images that appears in the bottom of the web page**

The most important question is :

How the browser knows when an element enters the viewport ?

Modern browsers use an internal mechanism very similar to the **Intersection Observer API** — an API that detects when an element is **visible within the viewport** (the visible part of the web page).

When the image's position enters or gets near the visible area, the browser **starts downloading it** — that's how lazy loading works.

Note :

- Always combine **lazy loading** with other performance optimazation techniques such as **image compression** using the appropriate image formats
- Everything applied to images is also applied to iframes

Eager vs Auto Loading

- HTML provides an attribute called **loading** that can be used to **control the loading behavior** of img and iframe tags
- It s used to **defer** (تاجيل) the loading of resources until they are needed, which can reduce initial page load time, save bandwidth, and improves user interaction speed
- Loading attribute have 3 modes
  1. Lazy Loading
  2. Eager Loading
  3. Auto Loading
- **Eager** Loding **forces** the browser to load the image immediately, whenever it s appears is in the web page

- **Automatic** Loading means that the browser will decide if it will use the **Lazy** Loading Mode or **Eager** Loading Mode

Always set the loading as **Eager** in the top images, that tells the browser that this image should be shown immediately

Benefits of using loading Eager

1. Immediate Content Availability Essential for above the fold content ensuring that is loaded instantly to enhance user experience
2. Ensures critical resources to load immediately, even under slow connection

It s NOT recommanded to use auto loading, you should decide to use **Eager** or **Lazy** loading

Best Practices

- Use loading **Eager** when for important images that are in the **first** viewport or **near** to the top of the page
- Use loading **Lazy** to images and iframes that are below the fold or not immediately necessary

To get the HTML Boilerplate (Basic starting structure قالب جاهز), you can type just « ! » and this code will be generated automatically :

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

</body>
</html>
```