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

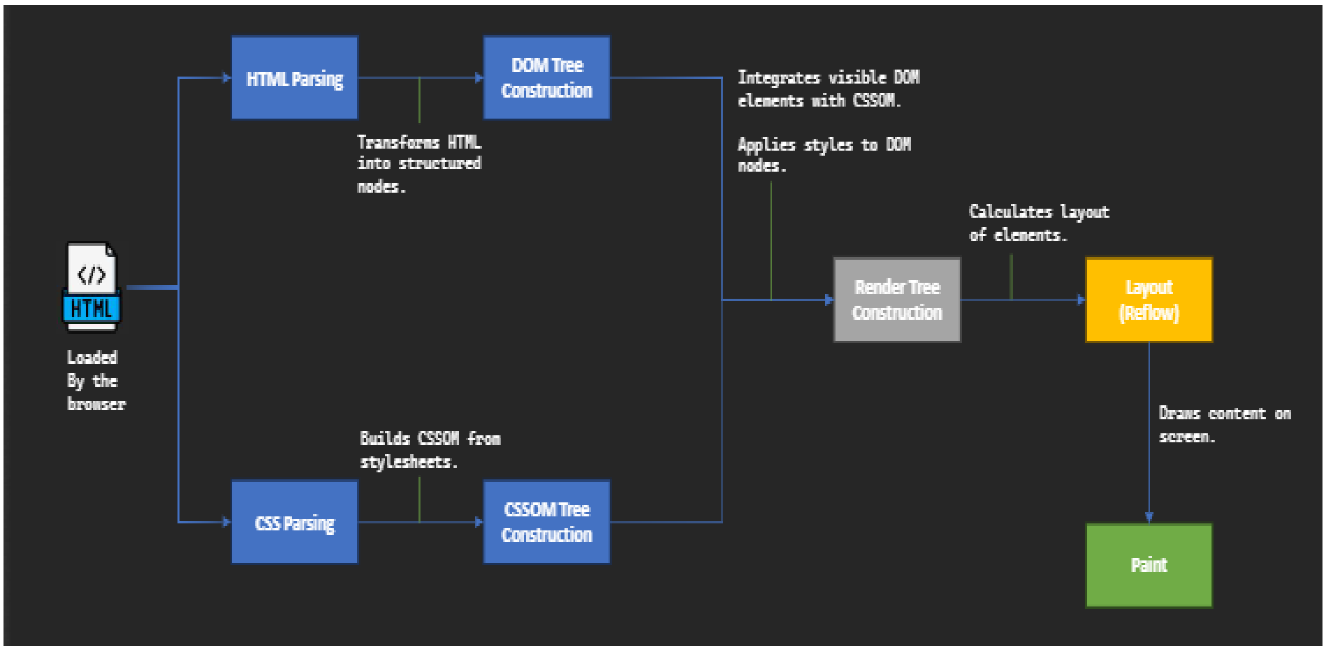
Relationship between HTML, CSS & JavaScript

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

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 (تحليل كود ال CSS)

- **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

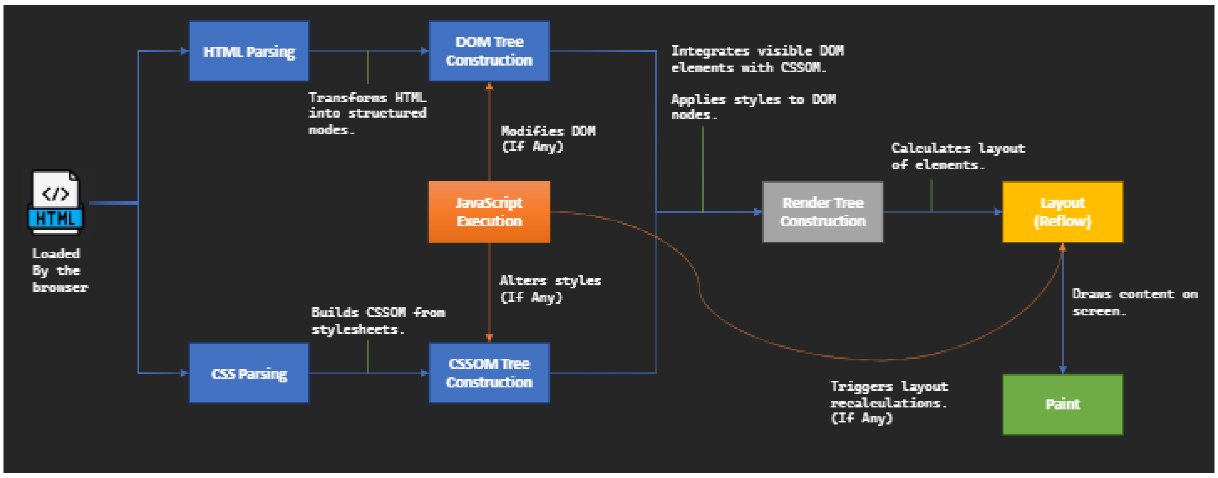
- **Process :** The browser **calculates** the exact position and size of each object on the page based on **DOM**, this process known as «layout» or «reflow». So layout process specialized **only** in calculating sizes & positions of elements
  This process can be affected by JavaScript, if the script 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 automaticly 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**

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…
- 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

- The ONLY thing that <span> and <div> elements share is that they are BOTH semantic **neutral** meaning