Web Anatomy

<HTML>

Dustin Menjivar Cornejo

Edición 21.06

Year: 2021
Edition: 21.06
Author: Dustin Menjivar Cornejo

Index

Requirements

Computer or mobile device

Code editor or IDE

- For PC, we suggest the visual studio code program
- For mobile, we suggest the Acode app

Visualizador

- In the case of using PC, it is a browser (We do not recommend IE)
- In the case of using a phone, it can be a browser or the same application we have downloaded

For the development of this material, we will use the "Visual Studio Code" program and the "Acode" application. If you have already worked with a different software, feel free to use the one that seems most comfortable for you.

Optional

- Basic computer course

Introduction to <HTML>

HyperText Markup Language or HTML, is the markup language used to structure web projects and applications.

The use of HTML is a standard, this language is supervised and established by the W3C (World Wide Web Consortium), which is in charge of establishing, updating and defining the structures and guidelines on the use of the language.

The standardization of the language allows us to have and greater compatibility between devices and in this way, any project can be supported on the device from which we view it, or failing that, have a simple way to make it compatible with devices that do not support it, that to tell the truth today, it is unlikely that we do not have a device that is not compatible with this web standard.

It should be noted that HTML is not a programming language itself, however, it is a markup language (layout or structured), which defines the structure of web projects and without this standard it would be complex to make a project that is compatible with the large number of resolutions and devices from which we connect today.

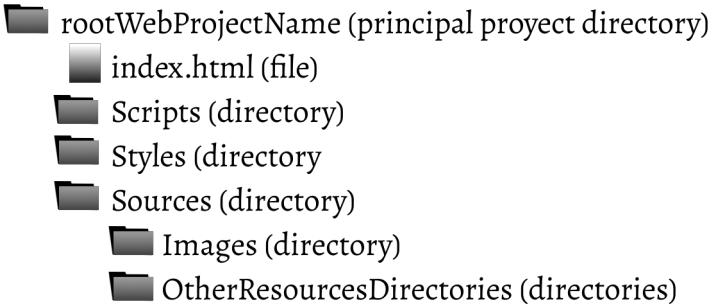
File management

Before we begin, we must define the essential, and that is to know how the organization of our files and directives will be.

By organizing our files we will obtain an easy to use and scale structure, because we will make a structure in such a way that allows the project to acquire new functionalities, directories and files in a simple way.

<!--We advise not to use the space, underscore or underscore character to name the project files or directories.-->

Example of a file handling structure:



Structure of an element

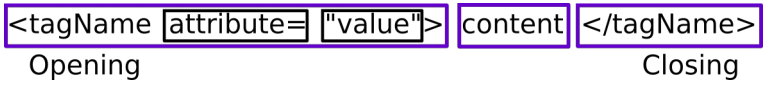
HTML is made up in its entirety by what we will understand as tags. Here is the first concept.

Tag

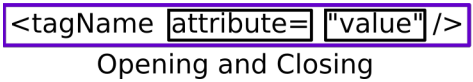
The tag, also called an element, box or container, is a block of space defined in the DOM, to which the content it will have is assigned and thus be able to dispose.

We can obtain similar results with different tags, but we must learn to use the appropriate one for each situation, taking into account its syntactics and semantics.

General structure of a tag



Particular structure of a tag



Attribute

It is a property assigned to the tag, an attribute can provide a semantic sense so that the element has a better structure, it can define a name which will be used as a reference on the server, or it can be a property that simply changes the visual appearance of the element. , although generally we will do the latter with another language.

Basic attribute

It is one that, when declared, requires a value to be able to apply the property to the element.

Boolean attribute

It is one that simply requires to be declared to apply its effect on the element.

Value

The value is the parameter that an attribute receives, and its function is to indicate to the attribute, the quality or specific data to apply to the element.

Container box model

Before starting to learn about html elements, we must know something that will help us to know how they will be laid out.

Block box

The elements that have the block property by default are those that will use the total width of the parent element, regardless of the size of the content it has, in addition to performing a line break.

Box online

The elements that have the inline property by default are those that will only use width of the size of the content that it has, in addition these elements are laid out one next to the other.

Parent and child element

In HTML, we will occasionally speak of a parent element and a child element, this simply refers to one element that is inside another.

HTML content structure

Any project structure that works with HTML must only contain tags and attributes with their respective values.

The styles, functionality or other extra characteristic, must be implemented later by means of files that have either only functionality or only styles (external or local files).

The elements used when structuring the code, in some cases is different from the aforementioned, this is because an element can be categorized as general or empty.

<tagName attribute= "value"> content </tagName>

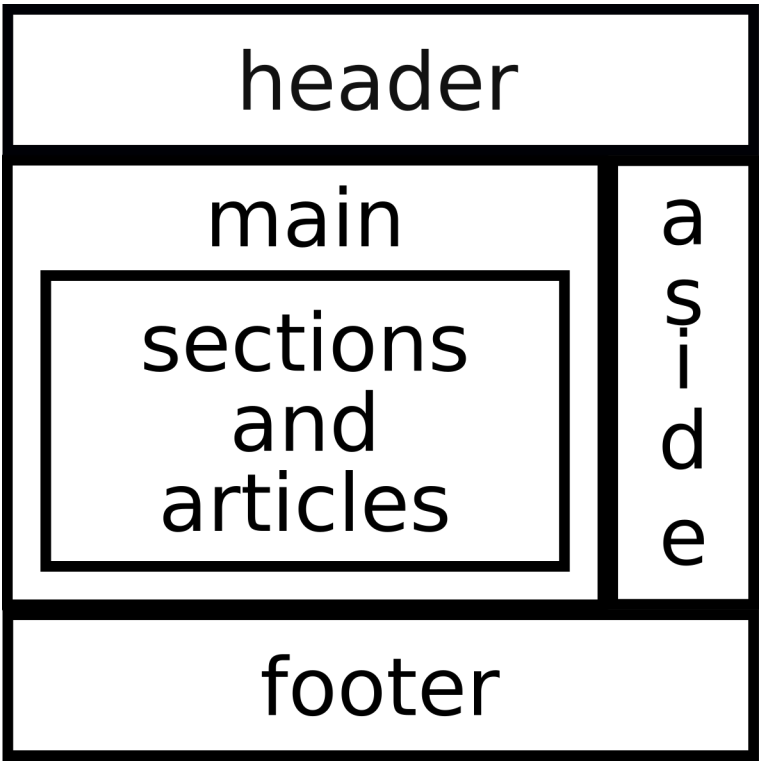
General element. They are the most common elements to find, because of the totality of tags in HTML, this group is the vast majority.

<tagName attribute= "value" />

Empty element. They are the elements that do not have content and also do not have a closing tag and sometimes, or attributes.

Basic structure of a project

Example of a web structure visible to the end user.



<!--We can use articles within sections, as vice versa, we only have to take into account the context in which we use them-->

Basic structure of a project (code)

```
<!doctype html>
<html lang= "languageAbrebiation" >
<head>
        <!--metaViewport and metadataTags-->
        <!--sourcesLinksTags and Icons links -->
        <title>ProjectName</title>
</head>
<body>
        <header></header>
        <nav></nav>
        <main>
                <!--Sections and Articles-->
        </main>
        <aside></aside>
        <footer></footer>
        <!--scriptsLinks -->
</body>
</html>
```

HTML content elements

Every HTML document has a basic structure, the function of each of them is explained below.

<!doctype html>

Document type. Tag found at the beginning of any file that uses HTML, its function is to inform the viewer that we will be working with an HTML language document.

<html> </html>

HTML. Tag which indicates that, within them, all our structure and content of the project will be found.

<head> </head>

Head of the document. This tag will have useful information for the browser.

<meta />

Metadata tag (Use explained in its respective topic).

Links to resources (Uses explained in their respective topic)

<body> </body>

Body of the project. It has the visible content (layout) of our website.

<header> </header>

Project header. Element indicating the header area of our site, usually containing our site logo and nav.

<nav> </nav>

Main Navigation. Tag that indicates the area where the navigation menu and its links are located.

<main> </main>

Main content of the site (information relevant to the user).

<section> </section>

Section. Content section related to the parent element.

** **

Simple container. Container without semantics.

<article> </article>

Article. Independent content section; We can visualize this element independent of the site, and even then it will not lose its meaning.

** **

Simple element. Element without semantics

<aside> </aside>

Secondary content. The content within this element is not relevant to the website, or is complementary, and can be from a secondary menu, to advertising.

<footer> </footer>

Footer. It contains complementary information, either copyright, contact information or other information of interest to the user.

Links to "script" resources. In general, the JavaScript are linked (Its use will be explained in its respective topic)

Text tags

Comments

<!--textToBeCommented-->

Comments are generally used to document our code, they do not affect the content of the site, because these are simply supporting texts (annotations). The annotations will be viewed only by the programmer.

Header levels

<h#> titleText </h#>

(The pound sign represents the heading level number.)

The header tags are used to indicate a title on our page. The tag only admits an integer value between one and six. In addition, we can only have a single "h1" per page.

This element is used to define the titles of either main sections, sub-sections or articles.

Pharagraphs

** pharagraphText **

The paragraph tag is the most useful, because with it we can define the vast majority of texts in our projects

Dates

<time datetime="dateOrHourOrBoth">dateOrHour<time/>

The dates can be used to indicate dates, hours or both, useful when we have articles or sections which present a date or time as a data of interest

Quotes

Citations are supportive when we use references from books, individuals, websites, characters, or other sources.

<blockquote> quoteText </blockquote>

We define an appointment, the label is in block format

<q> quotesText </q>

We define an appointment, the label is in online format

<cite> citationAuthor </cite>

Citation author. This modifier is added next to the citation, useful for naming the corresponding author of the quoted text

Text modifiers

The modifiers must be inside some text label, these modifiers, inside their body, will have the text which you want to modify.

** textToBeBold **

Bold font. (visual modifier)

** textToBeStrong **

Modifier similar to bold, with the characteristic of indicating to the browser that the text will be relevant to the search engine.

** textToBeItalic**

Italic. (visual modifier)

** textToBeEmphasized **

Emphasis. Modifier similar to italic or italic, with the characteristic of indicating to the browser that the text will be relevant to the search engine, but a little less than the text within a

<small>text</small>

Small. Less relevant text than normal text.

<pre>textToBePreformatted</pre>

Preformatted text. The text that we type inside the element will be displayed in exactly the same way it was written, useful for example when we want to show code examples.

<code>codeText</code>

Code text. The text inside the element corresponds to code text, it can be used together with the preformatted text.

**
**

Line break.

<wbr/>

Predefined line break. Make a line break, only when it is necessary. The dash '-' also fulfills this function.

^{text}

Superscript. An example is in mathematical equations.

_{text}

Subscript. An example descriptions of chemical compounds

<hr/>

Horizontal line. Insert a horizontal line into the layout (use the full width).

<strike>textToBeStriked</strike>
Strikethrough. The text inside this tag will get a strikethrough by a horizontal line in the center of the text

Symbols and Characters in HTML

To display special characters, we will use your code.

Using ascii codes

The first way to add symbols in our texts is by using the combination of keys and the ASCII code of the corresponding symbol.

Symbols using hexadecimal code

This way is the alternative way to add symbols by means of a combination of keys and ASCII code, this method is generally used in computers with gnu / linux

Explicit symbols in HTML using ASCII codes.

In this way we define an explicit way to include a symbol in our texts, this way has the advantage of showing the symbol that we exactly need.

&#asciiCode;

Structure to insert a symbol explicitly through an ascii code (Sometimes it may not work)

<!--To see other ways to represent symbols in HTML, we can investigate "html symbols and characters"-->

Links

Link tags

** textDisplayedToUser **

Simple link (Used to link to websites or files)

** textDisplayedToUser **

Link to an item on the site

** textDisplayedToUser **

Link to an item on a different site or page.

<link rel="icon" type="image/imageExtentionType" href="sourceRoute/fileName.imageExtentionType ">

Link to icon. Icon shown in the viewer tab or shown when installing our site through pwa. We recommend linking various sizes (16x16, 32x32, 57x57, 60x60, 72x72, 76x76, 96x96, 114x114, 120x120, 144x144, 152x152, 180x180, 192x192) taking into account that the image extension is supported.

<link rel= "stylesheet" href= "sourceRoute"/>

Link Style Resources (CSS)

<script src= "sourceRoute" > </script>

Link to a JavaScript file.

Link attributes

In "href" the value of a hash sign '#' is used to indicate that no address will be accessed.

./sourcesDirName/fileName.extention

When the link points to our system, the path can be relative to the current directory, or from the root directory.

https://absoluteSourceRoute

If the link sends to a website, it must have the absolute path (URL) and this must include its respective protocol. For security reasons, it is advisable not to use addresses that do not have the **https: //** protocol.

** textDisplayedToUser **

Attribute **target = "_blank"**, it is used to indicate that the link should be opened in a new tab.

Attribute **rel = "noopener"**, It is useful when untrusted links are opened, to indicate that they cannot alter the source document through the Window.opener property

Attribute **rel = ”noreferrrer”**, Prevents the browser from sending the page address or any other value when navigating to another page, as a referrer through the Referer: HTTP header.

** textDisplayedToUser **

Download. This boolean attribute has the function of allowing a file to be downloaded with a relative path. (works when site is running on server). If we add a value, this will be the name of the file that we download.

Using target = "_blank" without rel = "noreferrer" and rel = "noopener" makes the website vulnerable to exploit attacks via the **window.opener** API

Lists

List tag

** listItems **

Unnumbered list (Unordered list)

** listItems **

Numbered list (Ordered list)

<dl> listTermsAndDefinitions </dl>

Definitions list, one use could be when making a dictionary

List items

** listItemValue **

Add a new item to the list (ordered or unordered).

<dt> listTermItem </dt>

To add elements (or terms) to a definition list, we will use the following tag.

<dd> listTermDefinition </dd>

Add a description to the end of the list.

Listas anidadas

Nested lists are simply lists that are inside a "li" element of another list, either ordered or unordered. Nested lists are generally a combination of ordered lists and unordered lists.

```
<listTag>
    <li> listItems </li>
    <li>
        <listTag> listItems </listTag>
    </li>
</listTag>
```

Multimedia

Images

When working with images, the formats that we can find are: PNG, GIF, WEBP and JPEG or JPG, although the most recommended is the WEBP format.

****

Basic structure to embed images to our website

Optimizing image loading

This topic can be studied later. However, there is no problem in studying it at this time.

srcset= “sourceRoutes”

The resource's path set. Attribute with which we can indicate to the viewer, for example, depending on the size of the viewer "Viewport", to load one or another image

srcset= " sourceRoute1/ imageName.extention,

sourceRoute2/ imageName.extention,

sourceRoute3/ imageName.extention"

In the attribute, we add the different paths of the images and next to them, we must add a value (we will see it shortly)

Sometimes a viewer does not support a specific type of extension, to solve this, we include backup resources, with a different file extension

srcset= " sourceRoute1/ imageName.extention,

sourceRoute1Backup1/ imageName.extention1,

sourceRoute1Backup2/ imageName.extention2,

sourceRoute2/ imageName.extention,

sourceRoute2Backup1/ imageName.extention1,

sourceRoute2Backup2/ imageName.extention2"

To indicate to the set of routes of the resource, which route we want to take, at a given moment, we must add an extra parameter in the route of the resource, for this we have two options:

Use srcset via DPR

Indicate it by means of the DPR "Device Pixel Ratio" (the pounding indicates the DPR value)

srcset= " sourceRoute1/ imageName.extention #x"

Use srcset via viewport width

Indicate it by the width of the screen. (The value of the pixels can be modified at our discretion or convenience)

srcset= " sourceRoute1/ imageName.extention"

media="width: 320px,

sourceRoute1/ image2Name.extention"
media="width: 360px,

sourceRoute1/ image3Name.extention"
media="width: 480px"

When working with srcset through the width viewport, we must add an extra attribute to the image, which helps us to indicate to the viewer, when it should load the image

sizes= "(max-widt:320px) 320w,

(max-widt:360px) 360w,

(min-widt:361px) 480w"

loading= “lazy”

Extra attribute for an image. What it allows us is to tell the viewer to only load the image when it is close to finding it, otherwise, the viewer will not load it.

Structure example using viewport width

<img loading="lazy" alt="alt text" title=”title text” src="./Sources/Images/imageName.png" srcset="

./Sources/Images/imageName.webp 320w,

./Sources/Images/imageName.webp 360w,"

sizes=" (max-widt:320px) 320w,

(min-widt:360px) 360w”>

<picture></picture>

Optional container, this element can be used to indicate more parameters for an image to be loaded

<picture>

<source srcset= “sourceRoute” media= “(max-width:

screenWidthValue)” />

</picture>

Video

<video> videosSources </video>

Base structure, contains the routes of the video with different extensions, this tag can be used with local files.

Video elements

<source src= "videoRoute" type= "video/videoExtention">

Add a video resource, we should include several types of formats of the same file to increase compatibility with different types of viewers

Video attributes

autoplay

Autoplay. Boolean attribute

controls

Boolean attribute that activates controls for the user.

loop

Repeat video. Boolean attribute that indicates that the video, once finished, will be played again

muted

Mute the video. Boolean attribute

poster= "sourceRoute"

Miniature. Add a thumbnail image to the video

preload= "propertyValue"

Indicates what we want to preload from the video, for the user

Values that can be defined in the preload attribute.

- **none**

  Do not preload any information

- **metadata**

  Just the metadata

- **auto**

  Allow to preload the video

Audio

<audio> audioSources </audio>

Base structure, contains the routes of the audio with different extensions, this tag can be used with local files.

Audio elements

<souce src=”sourceRoute” type= “audio/audioExtention”>

Add an audio resource, we should include several types of formats of the same file to increase compatibility with different types of viewers

Audio attributes

autoplay

Autoplay. Boolean attribute

controls

Controls Boolean attribute that indicates the activation of the controls for the user.

loop

Repeat audio. Boolean attribute that indicates that the audio should be played again, once finished

muted

Boolean attribute. Mute the audio. Boolean attribute

poster= "sourceRoute"

Boolean attribute. Add a thumbnail image to audio

preload= "propertyValue"

Indicate what we want to preload from the audio, for the user, the value can take different values

Valores de la propiedad precargar.

- **none**

  Do not preload any information

- **metadata**

  Just the metadata

- **auto**

  Allow preloading of audio

Iframe

Element used, generally. To embed content from other websites, be it videos, maps or something else, generally we will not have to structure this code because the sites already provide it to us.

<!--It is recommended not to use too many iframes as they have a strong impact on resource usage-->

<iframe src="url"></iframe>

Basic structure of the tag, keep in mind that depending on the resource, it may have more attributes.

If we want to embed YouTube videos to our website we recommend using the code following the example shown below (It will be useful to later control the video with JavaScript)

From our url of the video, we will only need the id of the video, which, generally the text that is after the word **watch =**

As an example, we have the following youtube address, which we obtain through the common way of copying the url of a video

https://www.youtube.com/watch?v=jNQXAC9IVRw

So following the example **OurVideoId= jNQXAC9IVRw**

So our iframe should look like the following

<iframe src="https://www.youtube.com/embed/**OurVideoId**?enablejsapi=1&html5=1" allowfullscreen="" frameborder="0"></iframe>

The text **?enablejsapi=1&html5=1** It is an attribute which enables us to use the youtube iframe api and thus be able to control the video through the use of scripts with JavaScript

<!--We regret not being able to bring more content on this topic. If necessary, later editions of this manual will add more content on the use of the "iframe" element.-->

Figure y Figcaption

Figure element is used to contain multimedia, poems, quotes, code of some language or other functions and uses.

The figure can also be explained, as an element that has optional related content, which means that if the parent container has, or not, the figure, it will not lose meaning and will always convey its idea in a complete way.

<figure>

mainMultimediaContent

<figcaption>Content</figcaption>

</figure>

<figcaption>complementaryContent</figcaption>

Supplementary content. Used to add

information of interest or complementary information

related to the content of the “figure”, it is generally

a text that complements the content of the "figure".

Tables

The tables, as can be assumed, we will use them

solely for tabulating data in an orderly and well-structured manner.

Table tag

<table> tableContent </table>

Elements of a table

<tr> tableColumns </tr>

Row. Add a row to the table

<td> cellData </td>

Table data. Add a table data to the row

In the table, you need to add useful information, to recognize what the data displayed is about. We will add said information with the following labels:

<caption> describedTableTitle </caption>

Table title. Describes the content that the table, this label is used as the first child of the table.

<thead> headerCells </thead>

Table header. Located after the title of the table, it will own the header cells.

<th> nameOfValueType </th>

Header cells. These cells have the name of the data the column will contain.

<tbody> dataRowsAndColumnsCells </tbody>

Table body. Inside this container, we will have all the data that we tabulate.

<tfoot> tableRowAndTableData </tfoot>

Table foot.

Cell attributes

rowspan= " intValue "

Cell size. Specify to a cell, what amount of rows will use (Expands vertically).

colspan= " intValue "

Cell size. Specify to a cell, how many columns it will use (They are expanded horizontally).

Grouping columns

Visually we have a complete table and it is correct, but semantically, we only have a set of rows with cells. To add the corresponding semantics to the columns, we must use a couple of elements which group the cells in the form of a column. Adding these elements is totally optional, but it helps to have better semantics in the table

<colgroup> tableColums </colgroup>

Indicating the grouping of columns. This label is a child of the table (within it it has the column elements of the table)

<col/>

Table columns. With this element we indicate the number of semantic columns that we have in our table. We must use a label for each semantic column that we have in the table.

span= " numberOfColumns "

If we need a semantic column to use more than one visual column, we will use the following attribute on the table column element (enter an integer value).

Forms

Form tags

<form> formContent </form>

Base structure of the form tag.

Atributos de un formulario

action= "sourceRoute"

Attribute of the form with which we indicate the element which will be in charge of processing the form data, usually it is the url of a file that is on our server.

method= "methodType"

Attribute of the form, defined after the "action", used to send said data. (there are two methods).

- POST
- GET

Form elements

All (or the vast majority) of the data entries in a form will be received using the same type of tag. To differentiate the type of data entered, we will define the value of the type of said input.

<input id="inputIdName" type="inputType" name= "referentialNameOfTheInput" value= "inputValue">

Base structure of the data entry tag. Keep in mind that it depends on the functionality that we will use with the input, we can omit or add any of the attributes

<label for= "inputIdName"> labelText </input>

To add a text which will be related to the input, we can use an element, which must have a "for" attribute, and its value must be the input identifier, so we will relate them to each other.

<select> optionsItems </select>

Selection list

<option value="optionValue" > optionText </option>

Add a new item to the selection list

multiple

Boolean attribute. To indicate that we can make multiple selection, we add the attribute "multiple" to the drop-down list

<opgroup label= “text” > optionsToBeGrouped </opgroup>

Group of options. By having extensive selection lists, we can group their options to facilitate their understanding, with the "label" attribute we add the text referring to the group

<datalist list= “inputListAttributeValue” > optionsItems

</datalist>

Data list. Selection list with filter by character match. For its use it requires an input of type list and with the list attributes

The options within the data list are structured in the same way as the selection list (options and groups of options), due to this we do not think it necessary to repeat your explanation (If you want to add it in subsequent versions)

<fieldset> groupedFormElements </fieldset>

Group elements within a form.

<legend> fieldSetTitle </legend>

Add a text that indicates the content of the "fieldset"

<textarea> </textarea>

Text area. Area for the user to enter text, we can use it as the body of the message when we make a contact form

<button> </button>

Button. Generally used to add functionality through JavaScript, the subject of functionality through JavaScript can be learned with our JavaScript manual and course

Atributos de los elementos del formulario

required

For any type of data that the user will enter, we can indicate that it is a required data.

placeholder= “textDisplayedToUser”

Position marker. Add a text similar to the label, this has the advantage of being as content, and it also removes itself when selecting the entry (it would be like a ghost text). Useful for string entries.

readonly

Read only. The input should have a set value before applying this attribute. The input value will be sent to the server, but the value cannot be modified.

disable

Read only and disabled. The input should have a value before applying this attribute. The value of the ticket will not be sent.

min= “value” max= “value”

Attributes to define the minimum and maximum range of a numeric entry (enter an integer value)

minlenght= “value” maxlenght= “value”

Attributes to define the minimum and maximum characters in a text string input (enter an integer value)

autofocus

Focus. With this attribute we simulate a user click on a specific entry. It is activated when the form is loaded, its function is that it directs us directly to fill in the indicated field

Input attributes

value= "inputTextValue"

The elements, since they are input elements, will have values or texts that correspond to the type of data that the input requires, if we want to modify it we can use the following attribute.

Placeholder. It is recommended to use this way of indicating the type of input whenever possible, instead of displaying a text with the value attribute

(We will only remember this attribute, since it has already been explained previously in this topic).

(It is found in the Attributes of Form Elements topic, you can go back a bit to find its use)

type= "inputType"

For the inputs, the most important thing we must know is their type, because with this, we will identify what type of data we will receive.

Possible property values based on the data to receive:

type= "text" Text

type= "password" Password

type= “number” Number

type= “tel” Phone number

type= "email" E-mail

type= "date" Date

type= "time" Time, Time in hour format

type= “month” Month

type= “week” Week

type= “url” Web address

type= "checkbox" Checkbox (one or more)

type= "radio" Single checkbox.

<radiogroup> radioInputs <\radiogroup> It is important that the single checkboxes are grouped by this element so that they have the expected functionality

type= “file” Upload files. Requires functionality with JS.

type= "color" Box to select a color

type= "range" min= "minValue" max= "maxValue" step= “valuePerStep” Range, shows a "slider" where we can select one of the defined values.

type= "submit" value= "text" Add a button that sends the data entered in the form to the server

type= "button" value= "text" Button. Show a button (used to implement functionality with JS)

type= "reset" Restart. Input that has the function of resetting the values of the form fields.

maxlength= "maximumInputLength" Attributes that help define a maximum limit of characters for the inputs, we can use the following attribute

name= "inputVarName" If we will work with servers, we need to send the data and indicate a reference to what type of information it is receiving, which is solved by the following attribute

<!--Take into account that when working with forms that will connect to servers, the form elements should have the name and value attributes-->

Metadata

Metadata is optional information (but highly recommended)

<meta attribute=”value”/>

Basic structure of the metadata tag

<meta name="viewport" content="width=device-width, initial scale=1, user- scalable=no" />

One of the most important metadata. Indicates that the page fits the width of the screen, optionally we add the attributes **minimun-scale = 1, maximum-scale = 1.**

Atributos de los metadatos

charset= "characterCodification"

Charset. Used to indicate the character encoding of the page, usually "utf-8" will be used

name= "metadateType"

Name. Used to indicate the type of metadata to work with

content= "nameAttributeDescriptionText"

Contents. This attribute will be found next to any name tag, its function is to have the text or words that describe the name attribute

Values of the "name" attribute (metadata type)

name= "copyright"

Rights. We will indicate that a page is under the law of some entity.

Indicate that the page is indexable or not indexable. (Allows search engines to find a page on the web)

- **name= "robots" content= "index"**
- **name= "robots" content= "noindex"**

name="description" content=" pageDescriptionText "

Description of our website. Make a description of the site with a maximum (approximate) of 170 characters.

name=" author " content=" authorName "

Name of the author, company or website owner.

name=" keywords " content= "pageKeywods"

Keywords about our website.

Open Graph Protocol

Open graph protocol, is a tool created by facebook developers, to be able to standardize the elements of a site when its address is shared.

This protocol allows us to share an address with a set of data such as a title, an image and a description, previously defined by us.

This protocol is for optional use (If you like in later versions we add and explain its use in the manual)

The official site for how to use this protocol is:

https://opengraphprotocol.org

Sources

Any version and subsequent improvement of this manual will be held based on feedback and support that our community contributes to us, thus will have information at the forefront and of high quality

La version 21.06 de este manual utilizó las fuentes:

- DorianDesings https://www.youtube.com/watch?v=eesyGnJwfAY&list=PLROIqh_5RZeB92ME1GFyeqDVOa-gLoYbd
- FalconMasters https://www.youtube.com/watch?v=cqMfPS8jPys&list=PLhSj3UTs2_yVHt2DgHky_MzzRC58UHE4z
- Fazt https://www.youtube.com/watch?v=rbuYtrNUxg4
- SoyDalto https://www.youtube.com/watch?v=kN1XP-Bef7w
- yacklyon https://www.youtube.com/watch?v=PvTRboH_iqg&list=PLg9145ptuAijps8rns9muRGmxlhO1vOe1
- yacklyon https://www.youtube.com/watch v=Qla90rIkOFc&list=PLg9145ptuAijj9GoHPTcYT8IoQAOchM-n

Message to the reader.

We, Kixne ™, care about being able to provide you with the best possible information, therefore, we will strive to keep all our content, at the forefront and with high quality.

Our content is free. But if you want to support us in any way, in order to continue imparting knowledge, you can do so in various ways, either through suggestions, through contributions using our contacts and official sites, or if you like, you can see more forms of support on our site

https://kixne.github.io/web

The development of knowledge is one of the most important decisions for all people. We thank you for choosing us, and for entrusting us with your personal and professional development. We look forward to having you with us again.

We wish you a good day. See you soon!