LAYOUT.

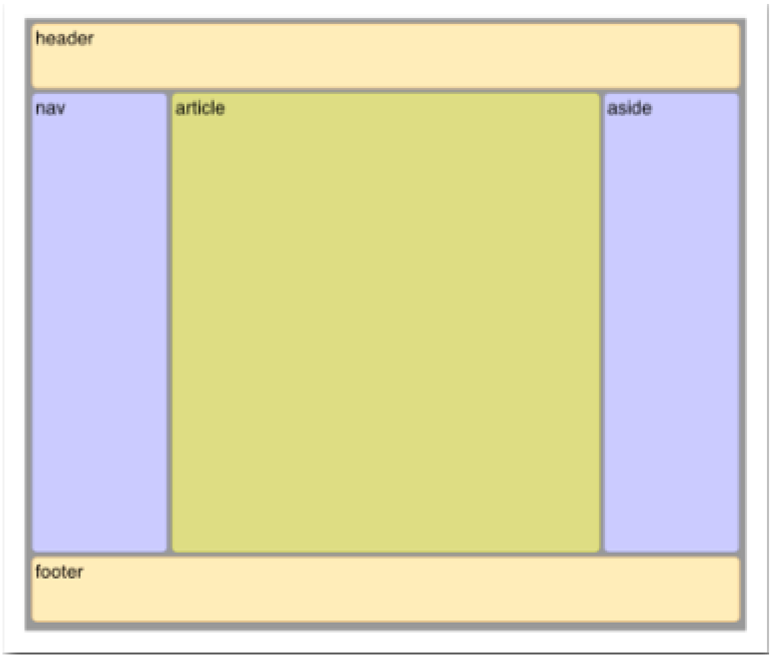
The layout of a web interface is the way to arrange the visual elements on the page. The layout determines the position of each of the elements that make up the web interface (header, menus, banners, content, etc.).

In addition, the layout also includes a series of decisions that affect the following characteristics:

- Margin size.
- Size and position of images and figures.
- Number of columns (or areas) in which the page is divided.
- Areas intentionally left white.

Holy Grail Layout

The Holy Grail refers to a web page layout which has multiple, equal height columns that are defined with style sheets. It is commonly desired and implemented, although the ways in which it can be implemented with current technologies all have drawbacks. Because of this, finding an optimal implementation has been likened to searching for the elusive Holy Grail.



History of layout

Before we get started working on the topic of layout directly, it is useful to understand a bit of HTML and CSS history.

In the not-so-distant-past, most HTML documents were long-form prose interspersed with lists and sometimes tables of information. HTML was used to present technical documentation, corporate communication, instructions and manuals, lists of files, and occasionally emails or notices. The "layout" needs were minimal to non-existent.

Over time, however, users began to prettify their documents by adjusting font sizes and font faces. And this was primarily done in HTML itself, which caused problems. Thus the first impetus for CSS was to separate any style information from the HTML document itself. However, the use cases those times were all still primarily text-based. The goals of CSS were modeled after the "master sheets" of some word processors.

Before the first specification for CSS had been agreed upon, Web developers began a transition to creating different types of documents. These documents were more like fancy magazine pages - images, not text, were the centerpiece. Decorative graphics abounded, advertisements showed up. And even as CSS2 and later CSS3 were being written, Web development changed again - Web pages became more interactive, the term "Web application" was coined. Many Web sites bore far more similarity to the control panel of a microwave than to a magazine article, much less the page of a book. And finally along came smart phones and a whole new "mobile Web" focus for Web sites, Web pages, and Web applications.

During its short lifetime, CSS has often played "catch up" especially with respect to layout. Here is a short list of techniques and CSS properties used historically for layout:

- tables and "slicing"
- absolute positioning
- floats and clear
- css columns
- css tables
- media queries
- flexbox
- grid

Techniques for making layouts

Why won't this work?

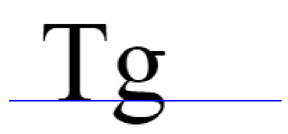
When newbie developers are groping around CSS blindly, they often stumble upon a variety of CSS properties that could be used to alter the positioning or size of an element such as left, top, and margin. However, when using the properties, these developers get confused because the properties fail to behave consistently. Sometimes the properties work, sometimes they don't, sometimes they do the opposite of what they are doing in a different rule. We have not covered properties like left and top yet, but we have introduced margin and the intrepid readers may have already discovered in their exercises that margin can have some unexpected behavior. Why is that?

The answer has to do with the two CSS properties: display and position. The display property, in particular, has different default values for different tags. Some tags start with display:block, and others are display:inline. They behave very differently. These two properties (display and position) often change how an element responds to certain other layout properties. And when this is not understood, then it may seem random to a developer struggling to get stuff to work.

So, let's start with understanding a very important difference between block and inline display. And that begins with the baseline.

Baseline

The text "baseline" is a key concept to understanding how the browser makes its layout decisions.



In the image, we see two text characters placed next to each other, the blue line indicating the baseline. The baseline determines how and where the characters are positioned. Note that the tail of the "g" hangs below the baseline.

The baseline is never drawn by the browser, it is not exposed directly to you as a developer, and CSS only may have some properties related to it. However, the baseline governs the placement of all inline elements.

Display: block versus inline

The browser, for example, will give every <p> tag gets a new line, but <a> tags do not.

This is the key distinction between the "block" level elements (like the <p> tag) and the "inline" elements (like the <a> tag). Here is a quick table of the default values for some of the tags we've already learned.

Bloque	**Línea**
p	a
h1	span
div	q
blockquote	i
ul	b
ol	label
li	input

Here are some differences between the block - level and inline elements.

Block Level

- appears below and to the left of their block level neighbors (like a carriage return on a typewriter going to the next new line)
- will expand to fill the width of the parent container by default
- respects all margin properties
- can have its width property set, which will make it narrower and cause its children to wrap, but not crop. (We'll cover this later)
- takes on the height of all its children (pending certain exceptions) as long as its own height is unset. (We will cover setting the height later)
- ignores the vertical-align property

Inline Elements

- Simply appear to the right of their preceding inline neighbor. They do not drop to the next line unless they must "wrap".
- by default, the width is simply the width of the content of the element, plus any padding
- ignore top and bottom margin settings
- ignore width and height properties
- are subject to vertical-align property as well as CSS white-space settings
- support padding, but any padding-top or padding-bottom does not contribute to the calculation of the height of the text line it sits upon

- cleave to the baseline where they are being placed

The last bullet about inline elements is one of the most important to understand. Inline elements cleave to the baseline. This is very important to understand why inline elements are positioned vertically the way they are. It also contributes to why they ignore top and bottom margins. Note that making an inline element "bigger" with padding will certainly keep its neighbors away horizontally. But if there is a neighboring text line above or below, it can only be kept at bay with the line-height property, not margins or padding.

Below we see a span that has padding, margin-top, and background-color applied, but no extra room is being made for it above or below, so its background is overlapping the lines above and below.

https://codepen.io/paqui-molina/pen/yLLqxqd

display property
https://codepen.io/paqui-molina/pen/OBYdOW

Horizontal and vertical centering

Horizontal centering.

Inline
To center an inline element, we use the text-align property of its parent.

p {text-align: center; } /* the text and any inline children of this element will be centered */

Block

First, you may recall that block level elements take the width of their parent by default.
If the element is the same width as its parent, it is already centered.
So the first step is to limit the width of the element. Setting the width property directly is not generally a good practice, but we'll just do that and discuss sizing at length later.

div {width: 200px; }

Now that we've sized the element, how to center it?

Width margin:
If set to auto, then the left and right margins will center the element. This is the simplest and best way of centering a block level element. So the full solution is to set the width and apply auto to the left and right margins (or to all margins).

div {width: 200px; margin: auto; }

Vertical centering.

Line
Inline elements respect the vertical-align property. This determines how the inline element is aligned relative to the baseline it is being laid upon. This may or may not solve your vertical centering conundrum.

Block
There is no margin:auto approach to vertical centering. There are some complicated systems that folk have developed, but the shortest and best answer to vertical centering: use flexbox.

Position property

The 'left', 'top', 'right', and 'bottom' properties

In CSS, there are four properties which can be used to adjust or set the position of an element. Those properties are: left, top, right, and bottom.

However, using those properties by themselves will have no effect on any element.

The reason these properties don't work by default is that they only work when applied to positioned elements. And positioned elements are those that have had their position property changed from the default.

The position property

The position property tells how an item is positioned on the page and how it responds to position adjustment properties (left, top, right and bottom).

https://www.w3schools.com/css/css_positioning.asp

https://developer.mozilla.org/es/docs/Web/CSS/position

z-index property

https://www.w3schools.com/cssref/pr_pos_z-index.asp
https://developer.mozilla.org/es/docs/Web/CSS/z-index

float property

https://www.w3schools.com/cssref/pr_class_float.asp
https://www.w3schools.com/cssref/pr_class_clear.asp

Clear property

https://www.w3schools.com/cssref/pr_class_clear.asp

Overflow property

https://www.w3schools.com/cssref/pr_pos_overflow.asp