

**TECNOLÓGICO NACIONAL DE MÉXICO**

**INSTITUTO TECNOLOGICO DE MERIDA**

**INGENIERÍA EN SISTEMAS COMPUTACIONALES**



**DESARROLLO FRONTEND**

**PROFESOR:** [**Héctor Jesús Cetina Cordero**](http://tecvi.itmerida.mx/user/view.php?id=1831&course=1)

**MAQUETACION CON POSITION Y Z-INDEX**

**Unidad**

**Grupo: 9SA**

**ALUMNO(A)**

* **Alcala Morales Aldo Gaspar**

**Fecha: 22/10/2019**

La propiedad **position** de [CSS](https://developer.mozilla.org/en-US/docs/CSS) especifica cómo un elemento es posicionado en el documento. Las propiedades [top](https://developer.mozilla.org/es/docs/Web/CSS/top), [right](https://developer.mozilla.org/es/docs/Web/CSS/right), [bottom](https://developer.mozilla.org/es/docs/Web/CSS/bottom), y [left](https://developer.mozilla.org/es/docs/Web/CSS/left) determinan la ubicación final de los elementos posicionados.

Tipos de posicionamiento

* Un **elemento posicionado**es un elemento cuyo valor [computado](https://developer.mozilla.org/en-US/docs/CSS/computed_value) de position es relative, absolute, fixed, o sticky. (En otras palabras, cualquiera excepto static).
* Un **elemento posicionado relativamente** es un elemento cuyo valor [computado](https://developer.mozilla.org/en-US/docs/CSS/computed_value) de position es relative. Las propiedades [top](https://developer.mozilla.org/es/docs/Web/CSS/top) y [bottom](https://developer.mozilla.org/es/docs/Web/CSS/bottom) especifican el desplazamiento vertical desde su posición original; las propiedades [left](https://developer.mozilla.org/es/docs/Web/CSS/left) y [right](https://developer.mozilla.org/es/docs/Web/CSS/right) especifican su desplazamiento horizontal.
* Un **elemento posicionado absolutamente** es un elemento cuyo valor [computado](https://developer.mozilla.org/en-US/docs/CSS/computed_value) de  position es absolute o fixed. Las propiedades [top](https://developer.mozilla.org/es/docs/Web/CSS/top), [right](https://developer.mozilla.org/es/docs/Web/CSS/right), [bottom](https://developer.mozilla.org/es/docs/Web/CSS/bottom), y  [left](https://developer.mozilla.org/es/docs/Web/CSS/left) especifican el desplazamiento desde los bordes del [bloque contenedor](https://developer.mozilla.org/en-US/docs/Web/CSS/All_About_The_Containing_Block) del elemento. (El bloque contenedor es el ancestro más cercano posicionado). Si el elemento tiene márgenes, se agregarán al desplazamiento.
* Un **elemento posicionado fijamente**es un elemento cuyo valor de  position [computado](https://developer.mozilla.org/en-US/docs/CSS/computed_value) es sticky. Es tratado como un elemento posicionado relativamente hasta que su [bloque contenedor](https://developer.mozilla.org/en-US/docs/Web/CSS/All_About_The_Containing_Block) cruza un límite establecido (como por ejemplo dando a [top](https://developer.mozilla.org/es/docs/Web/CSS/top) cualquier valor distinto de auto), dentro de su flujo principal (o el contenedor dentro del cual se mueve), punto al cual es tratado como "fijo" hasta que alcance el borde opuesto de su [bloque contenedor](https://developer.mozilla.org/en-US/docs/Web/CSS/All_About_The_Containing_Block).

La mayoría de las veces, los elementos absolutamente posicionados que tienen su [height](https://developer.mozilla.org/es/docs/Web/CSS/height) y [width](https://developer.mozilla.org/es/docs/Web/CSS/width) establecidos en auto son ajustados hasta acomodarse a su contenido. Sin embargo, elementos non-[replaced](https://developer.mozilla.org/en-US/docs/Web/CSS/Replaced_element) y absolutamente posicionados se pueden crear para llenar el espacio vertical disponible, especificando tanto [top](https://developer.mozilla.org/es/docs/Web/CSS/top) como [bottom](https://developer.mozilla.org/es/docs/Web/CSS/bottom), y dejando [height](https://developer.mozilla.org/es/docs/Web/CSS/height) sin especificar (es decir, auto). De igual manera se pueden utilizar para llenar el espacio horizontal disponible especificando tanto [left](https://developer.mozilla.org/es/docs/Web/CSS/left) como [right](https://developer.mozilla.org/es/docs/Web/CSS/right), y dando a [width](https://developer.mozilla.org/es/docs/Web/CSS/width) el valor de auto.

A excepción del caso anteriormente descrito (de elementos posicionados absolutamente rellenando el espacio disponible):

* Si ambos, top y bottom están especificados (técnicamente, no auto), top gana.
* Si ambos, left y right, están especificados,  left gana cuando es ltr (Inglés, japonés horizontal, etc.) y right gana cuando [direction](https://developer.mozilla.org/es/docs/Web/CSS/direction) es rtl (Persa, árabe, hebreo, etc.).

**Sintaxis**

La propiedad position es especificada como una palabra única elegida de la siguiente lista de valores.

Valores

**static**

El elemento es posicionado de acuerdo al flujo normal del documento. Las propiedades [top](https://developer.mozilla.org/es/docs/Web/CSS/top), [right](https://developer.mozilla.org/es/docs/Web/CSS/right), [bottom](https://developer.mozilla.org/es/docs/Web/CSS/bottom), [left](https://developer.mozilla.org/es/docs/Web/CSS/left), and [z-index](https://developer.mozilla.org/es/docs/Web/CSS/z-index) *no tienen efecto*. Este es el valor por defecto.

**relative**

El elemento es posicionado de acuerdo al flujo normal del documento, y luego es desplazado *con relación a sí mismo*, con base en los valores de top, right, bottom, and left. El desplazamiento no afecta la posición de ningún otro elemento; por lo que, el espacio que se le da al elemento en el esquema de la página es el mismo como si la posición fuera static. Este valor crea un nuevo [contexto de apilamiento](https://developer.mozilla.org/es/docs/Web/CSS/CSS_Positioning/entendiendo_z_index/El_contexto_de_apilamiento), donde el valor de z-index no es auto. El efecto que tiene relative sobre los elementos table-\*-group, table-row, table-column, table-cell, y table-caption no está definido.

**absolute**

El elemento es removido del flujo normal del documento, sin crearse espacio alguno para el elemento en el esquema de la página. Es posicionado relativo a su ancestro posicionado más cercano, si lo hay; de lo contrario, se ubica relativo al [bloque contenedor](https://developer.mozilla.org/en-US/docs/Web/CSS/All_About_The_Containing_Block) inicial. Su posición final está determinada por los valores de top, right, bottom, y left.

Este valor crea un nuevo [contexto de apilamiento](https://developer.mozilla.org/es/docs/Web/CSS/CSS_Positioning/entendiendo_z_index/El_contexto_de_apilamiento) cuando el valor de z-index no es auto. Elementos absolutamente posicionados pueden tener margen, y no colapsan con ningún otro margen.

**fixed**

El elemento es removido del flujo normal del documento, sin crearse espacio alguno para el elemento en el esquema de la página. Es posicionado con relación al [bloque contenedor](https://developer.mozilla.org/en-US/docs/Web/CSS/All_About_The_Containing_Block) inicial establecido por el [viewport](https://developer.mozilla.org/es/docs/Glossary/viewport), excepto cuando uno de sus ancestros tiene una propiedad transform, perspective, o filter establecida en algo que no sea none (ver [CSS Transforms Spec](https://www.w3.org/TR/css-transforms-1/#propdef-transform)), en cuyo caso ese ancestro se comporta como el bloque contenedor. (Notar que hay inconsistencias del navegador con perspective y filter contribuyendo a la formación del bloque contenedor.) Su posición final es determinada por los valores de top, right, bottom, y left.

Estos valores siempre crean un nuevo [contexto de apilamiento](https://developer.mozilla.org/es/docs/Web/CSS/CSS_Positioning/entendiendo_z_index/El_contexto_de_apilamiento). En documentos impresos, el elemento se coloca en la misma posición en *cada página*.

**sticky**

The element is positioned according to the normal flow of the document, and then offset relative to *its flow root and*[*containing block*](https://developer.mozilla.org/en-US/docs/Web/CSS/All_About_The_Containing_Block), including table-related elements, based on the values of top, right, bottom, and left. The offset does not affect the position of any other elements. This value always creates a new [stacking context](https://developer.mozilla.org/en/docs/Web/CSS/CSS_Positioning/Understanding_z_index/The_stacking_context). Note that sticky, by specification, will not work inside element with overflow: hidden or auto. (ref: [Github issue on W3C CSSWG](https://github.com/w3c/csswg-drafts/issues/865))

Formal syntax

static [|](https://developer.mozilla.org/es/docs/CSS/Value_definition_syntax#Single_bar) relative [|](https://developer.mozilla.org/es/docs/CSS/Value_definition_syntax#Single_bar) absolute [|](https://developer.mozilla.org/es/docs/CSS/Value_definition_syntax#Single_bar) sticky [|](https://developer.mozilla.org/es/docs/CSS/Value_definition_syntax#Single_bar) fixed

**Examples**

Relative positioning

Relatively positioned elements are offset a given amount from their normal position within the document, but without the offset affecting other elements. In the example below, note how the other elements are placed as if "Two" were taking up the space of its normal location.

**HTML**

<div class="box" id="one">One</div>

<div class="box" id="two">Two</div>

<div class="box" id="three">Three</div>

<div class="box" id="four">Four</div>

**CSS**

.box {

display: inline-block;

width: 100px;

height: 100px;

background: red;

color: white;

}

#two {

position: relative;

top: 20px;

left: 20px;

background: blue;

}

Abrir en CodePenAbrir en JSFiddle

Absolute positioning

Elements that are relatively positioned remain in the normal flow of the document. In contrast, an element that is absolutely positioned is taken out of the flow; thus, other elements are positioned as if it did not exist. The absolutely positioned element is positioned relative to its *nearest positioned ancestor* (i.e., the nearest ancestor that is not static). If a positioned ancestor doesn't exist, the initial container is used. In the example below, box "Two" has no positioned ancestor, so it is positioned relative to the the <body> of the document.

**HTML**

<div class="box" id="one">One</div>

<div class="box" id="two">Two</div>

<div class="box" id="three">Three</div>

<div class="box" id="four">Four</div>

**CSS**

.box {

display: inline-block;

width: 100px;

height: 100px;

background: red;

color: white;

}

#two {

position: absolute;

top: 20px;

left: 20px;

background: blue;

}

Abrir en CodePenAbrir en JSFiddle

Fixed positioning

Fixed positioning is similar to absolute positioning, with the exception that the element's [containing block](https://developer.mozilla.org/en-US/docs/Web/CSS/All_About_The_Containing_Block) is the *viewport*. This can be used to create a floating element that stays in the same position regardless of scrolling. In the example below, box "One" is fixed at 80 pixels from the top of the page and 10 pixels from the left. Even after scrolling, it remains in the same place relative to the viewport.

**HTML**

<div class="outer">

<p>

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam congue tortor eget pulvinar lobortis.

Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Nam ac dolor augue.

Pellentesque mi mi, laoreet et dolor sit amet, ultrices varius risus. Nam vitae iaculis elit.

Aliquam mollis interdum libero. Sed sodales placerat egestas. Vestibulum ut arcu aliquam purus viverra dictum vel sit amet mi.

Duis nisl mauris, aliquam sit amet luctus eget, dapibus in enim. Sed velit augue, pretium a sem aliquam, congue porttitor tortor.

Sed tempor nisl a lorem consequat, id maximus erat aliquet. Sed sagittis porta libero sed condimentum.

Aliquam finibus lectus nec ante congue rutrum. Curabitur quam quam, accumsan id ultrices ultrices, tempor et tellus.

</p>

<p>

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam congue tortor eget pulvinar lobortis.

Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Nam ac dolor augue.

Pellentesque mi mi, laoreet et dolor sit amet, ultrices varius risus. Nam vitae iaculis elit.

Aliquam mollis interdum libero. Sed sodales placerat egestas. Vestibulum ut arcu aliquam purus viverra dictum vel sit amet mi.

Duis nisl mauris, aliquam sit amet luctus eget, dapibus in enim. Sed velit augue, pretium a sem aliquam, congue porttitor tortor.

Sed tempor nisl a lorem consequat, id maximus erat aliquet. Sed sagittis porta libero sed condimentum.

Aliquam finibus lectus nec ante congue rutrum. Curabitur quam quam, accumsan id ultrices ultrices, tempor et tellus.

</p>

<div class="box" id="one">One</div>

</div>

**CSS**

.box {

width: 100px;

height: 100px;

background: red;

color: white;

}

#one {

position: fixed;

top: 80px;

left: 10px;

background: blue;

}

.outer {

width: 500px;

height: 300px;

overflow: scroll;

padding-left: 150px;

}

Abrir en CodePenAbrir en JSFiddle

Sticky positioning

Sticky positioning can be thought of as a hybrid of relative and fixed positioning. A stickily positioned element is treated as relatively positioned until it crosses a specified threshold, at which point it is treated as fixed until it reaches the boundary of its parent. For instance...

#one { position: sticky; top: 10px; }

...would position the element with id *one* relatively until the viewport were scrolled such that the element would be less than 10 pixels from the top. Beyond that threshold, the element would be fixed to 10 pixels from the top.

A common use for sticky positioning is for the headings in an alphabetized list. The "B" heading will appear just below the items that begin with "A" until they are scrolled offscreen. Rather than sliding offscreen with the rest of the content, the "B" heading will then remain fixed to the top of the viewport until all the "B" items have scrolled offscreen, at which point it will be covered up by the "C" heading, and so on.

You must specify a threshold with at least one of top, right, bottom, or left for sticky positioning to behave as expected. Otherwise, it will be indistinguishable from relative positioning.

**HTML**

<dl>

<div>

<dt>A</dt>

<dd>Andrew W.K.</dd>

<dd>Apparat</dd>

<dd>Arcade Fire</dd>

<dd>At The Drive-In</dd>

<dd>Aziz Ansari</dd>

</div>

<div>

<dt>C</dt>

<dd>Chromeo</dd>

<dd>Common</dd>

<dd>Converge</dd>

<dd>Crystal Castles</dd>

<dd>Cursive</dd>

</div>

<div>

<dt>E</dt>

<dd>Explosions In The Sky</dd>

</div>

<div>

<dt>T</dt>

<dd>Ted Leo &amp; The Pharmacists</dd>

<dd>T-Pain</dd>

<dd>Thrice</dd>

<dd>TV On The Radio</dd>

<dd>Two Gallants</dd>

</div>

</dl>

**CSS**

\* {

box-sizing: border-box;

}

dl > div {

background: #FFF;

padding: 24px 0 0 0;

}

dt {

background: #B8C1C8;

border-bottom: 1px solid #989EA4;

border-top: 1px solid #717D85;

color: #FFF;

font: bold 18px/21px Helvetica, Arial, sans-serif;

margin: 0;

padding: 2px 0 0 12px;

position: -webkit-sticky;

position: sticky;

top: -1px;

}

dd {

font: bold 20px/45px Helvetica, Arial, sans-serif;

margin: 0;

padding: 0 0 0 12px;

white-space: nowrap;

}

dd + dd {

border-top: 1px solid #CCC;

}

Abrir en CodePenAbrir en JSFiddle

ejemplo de position

Todo esto de position tal vez tenga un poco más de sentido con un ejemplo práctico. Abajo está la estructura real de una página.

.container {

position: relative;

}

nav {

position: absolute;

left: 0px;

width: 200px;

}

section {

*/\* position is static by default \*/*

margin-left: 200px;

}

footer {

position: fixed;

bottom: 0;

left: 0;

height: 70px;

background-color: white;

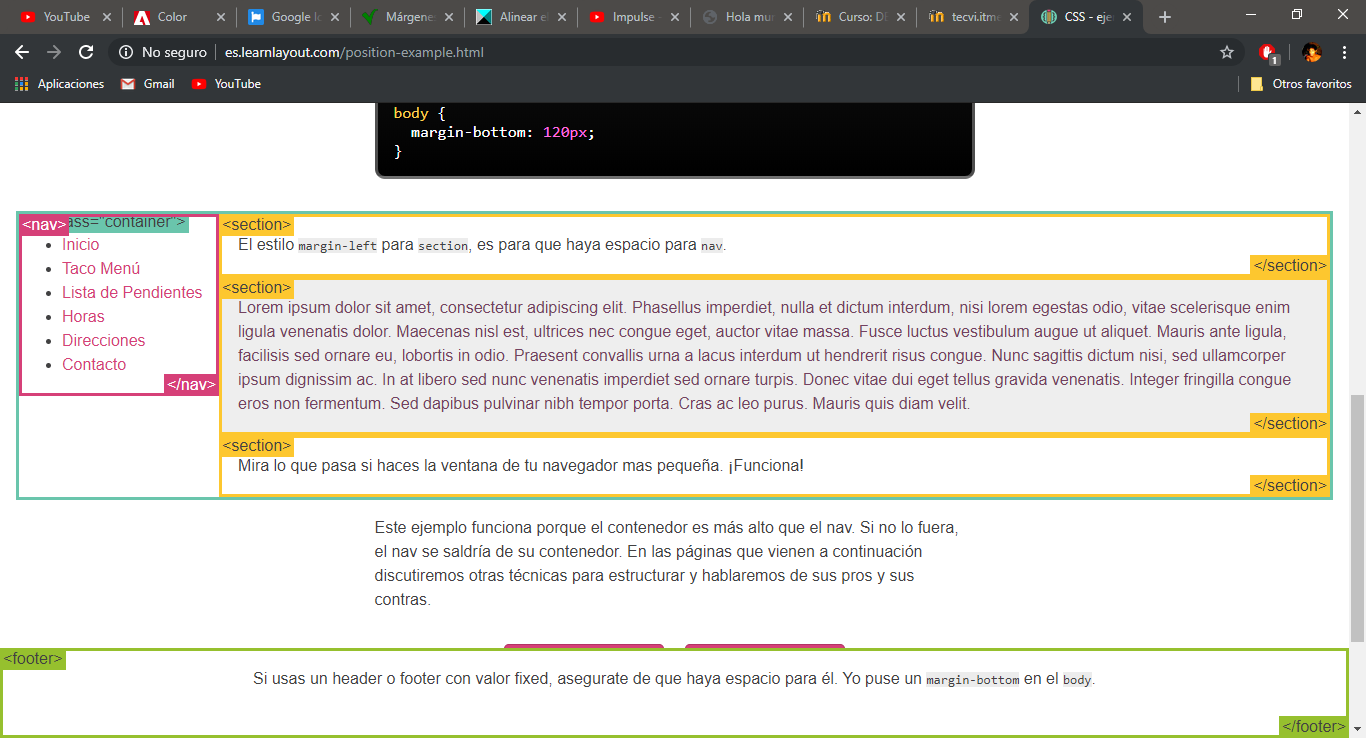
width: 100%;

}

body {

margin-bottom: 120px;

}



Este ejemplo funciona porque el contenedor es más alto que el nav. Si no lo fuera, el nav se saldría de su contenedor. En las páginas que vienen a continuación discutiremos otras técnicas para estructurar y hablaremos de sus pros y sus contras.

**Compatibilidad**

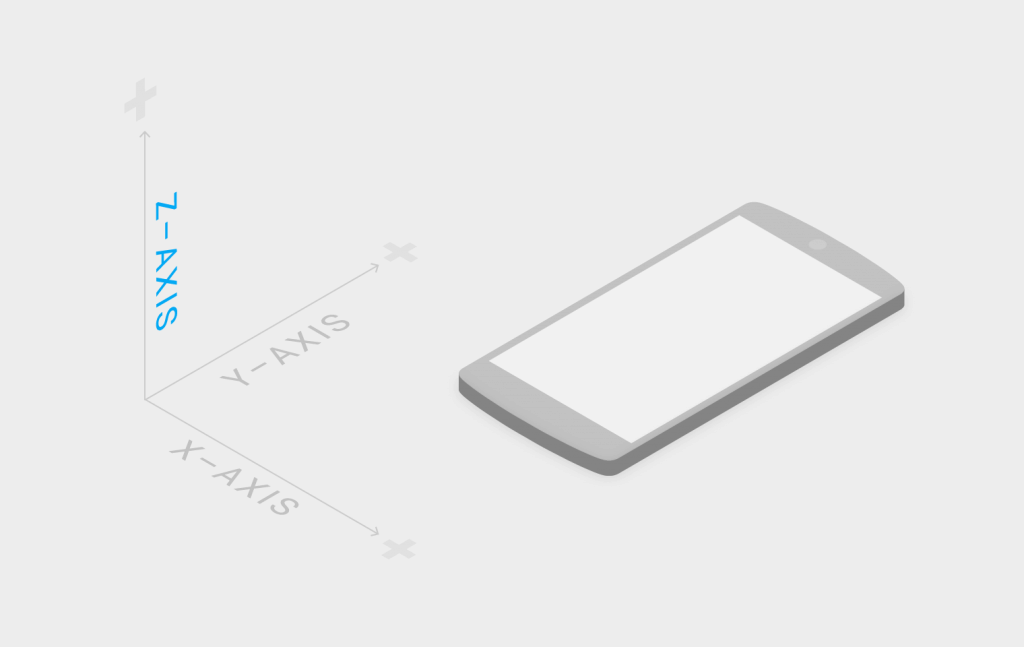
[Update compatibility data on GitHub](https://github.com/mdn/browser-compat-data)

|  | **Desktop** | | | | | | **Mobile** | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Chrome** | **Edge** | **Firefox** | **Internet Explorer** | **Opera** | **Safari** | **Android webview** | **Chrome para Android** | **Firefox para Android** | **Opera para Android** | **Safari en iOS** | **Samsung Internet** |
| **position** | Soporte completo1 | Soporte completo12 | Soporte completo1  Notas  Abrir | Soporte completo4 | Soporte completo4 | Soporte completo1 | Soporte completo≤37 | Soporte completo18 | Soporte completo4  Notas  Abrir | Soporte completoSi | Soporte completo1 | Soporte completoSi |
| **Absolutely-positioned flex children** | Soporte completo52 | Soporte completo12 | Soporte completo52 | Soporte completo10 | Soporte completo39 | Soporte completo11 | Soporte completo52 | Soporte completo52 | Soporte completo52 | Soporte completo41 | Soporte completo11 | Soporte completo6.0 |
| **fixed** | Soporte completo1 | Soporte completo12 | Soporte completo1  Notas  Abrir | Soporte completo7  Notas  Abrir | Soporte completo4 | Soporte completo1 | Soporte completo≤37 | Soporte completo18 | Soporte completo4  Notas  Abrir | Soporte completoSi | Soporte completo1 | Soporte completoSi |
| **Table elements as sticky positioning containers** | Soporte completo56 | Soporte completo16 | Soporte completo59 | Sin soporteNo | Soporte completo43 | Soporte completo8 | Soporte completo56 | Soporte completo56 | Soporte completo59 | Soporte completo43 | Soporte completo8 | Soporte completo6.0 |
| **sticky** | Soporte completo56 | Soporte completo16 | Soporte completo32Abrir | Sin soporteNo | Soporte completo43 | Soporte completo6.1  Prefijado  Abrir | Soporte completo56 | Soporte completo56 | Soporte completo32Abrir | Soporte completo43 | Soporte completo6.1  Prefijado  Abrir | Soporte completo6.0 |

# **Cómo utilizar z-index y los contextos de apilamiento**CSS

* [Juan Padial](https://cybmeta.com/sobre-mi)
* Publicado el 25 junio, 2018

La propiedad z-index en CSS se utiliza para **ordenar los elementos que se superpongan entre sí**. Con la propiedad z-index podemos controlar qué elemento iría encima y cual debajo, como si el documento tuviera **profundidad**, tres dimensiones en lugar de dos.

[](https://cybmeta.com/wp-content/uploads/2018/06/z-index-css.png)z-index CSS [Nick Babich: Top Design Moments for Mobile App](https://uxplanet.org/top-design-moments-for-mobile-app-e0178f06f6b3)

Puede haber muchos motivos para que dos elementos se superpongan. Por ejemplo, ¿quién no se ha visto lidiando con elementos posicionados de forma absoluta y fija que se superponen con otros?

## El orden z predeterminado

Antes de ver cómo utilizar z-index, vamos a repasar como se determina el orden z de elementos superpuestos.

El valor predeterminado de z-index es auto. El navegador irá ordenando los elementos en el **orden en el que aparezcan**, el primero quedará abajo y los siguientes elementos se irán apilando encima. Esta regla la aplica en este orden:

1. Primero posiciona el **elemento raíz** <html>
2. Luego los **elementos no posicionados** (cualquiera con el valor predeterminado position: static)
3. Luego los **elementos posicionados** (cualquiera con position: relative, position: absolute, position: fixed, position: sticky, o cualquier otro valor diferente a position: static).

Si al terminar este post tenéis más curiosidad, podéis ampliar leyendo [el caso de los elementos flotados](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Positioning/Understanding_z_index/Stacking_and_float).

### Ejemplo

Vamos a ver como los elementos no posicionados son colocados primero **aunque aparezcan los últimos en el HTML**:

<style>

.azul, .rojo, .verde {

position: absolute;

}

</style>

<div class="rojo">

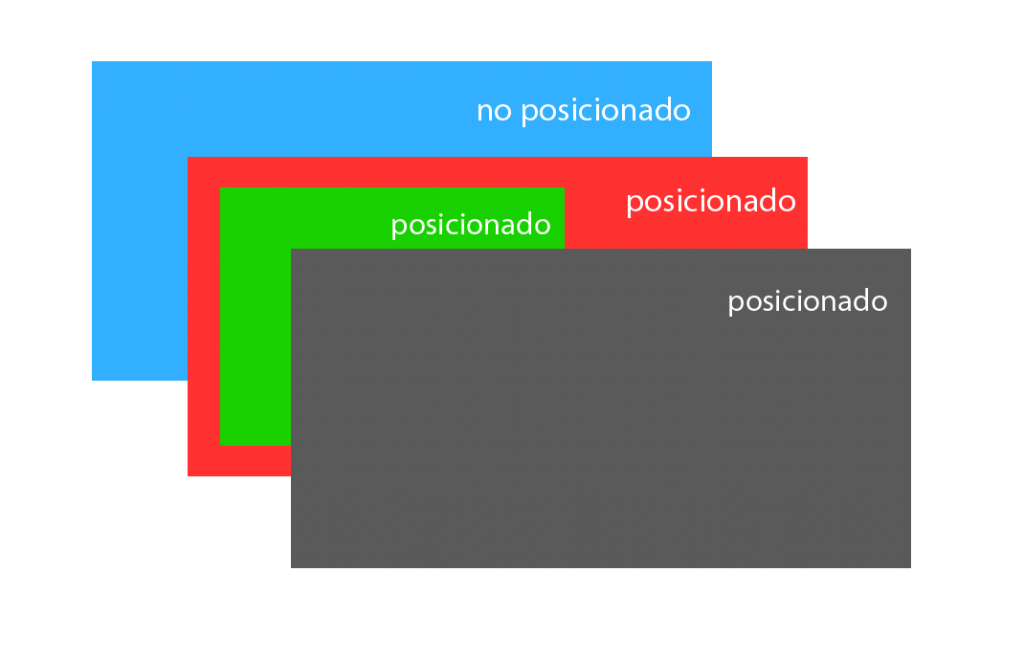
<div class="verde"></div>

</div>

<div class="gris"></div>

<div class="azul"></div>

El orden de apilamiento automático daría lugar al siguiente resultado:

[](https://cybmeta.com/wp-content/uploads/2018/06/stacking-predeterminado-1.png)Stacking predeterminado

Como ves, dejando los valores z-index en su valor predeterminado (z-index: auto), los elementos no posicionados se colocan siempre primero y después irán todos los demás.

## Cómo utilizar z-index

Con la propiedad z-index podemos cambiar el orden predeterminado que hemos visto antes. Simplemente se especifica el número de orden: z-index: 1, z-index: 2, etc. También puede tomar valores negativos: z-index: -1, z-index: -2, etc.

A mayor valor más arriba se representará el elemento. Si dos elementos tienen el mismo valor, el que sea definido antes se colocará primero.

Pero hay que tener dos cosas muy claras:

* La propiedad z-index **solo se aplica a elementos posicionados**. Los elementos no posicionados siempre se van a colocar uno en encima de otro en el orden en el que aparecen en el código.
* La propiedad z-index **no es heredable**.
* Definir la propiedad z-index con un **valor diferente a auto crea un stacking context**, un contexto de apilamiento. También se crea un stacking context en todos los elementos con position: fixed o sticky. Luego veremos qué significa esto.

### Ejemplo

Si ahora cambiamos el CSS anterior por este:

.gris, .rojo, .verde {

position: absolute;

}

.gris {

z-index: 2;

}

.verde {

z-index: 3;

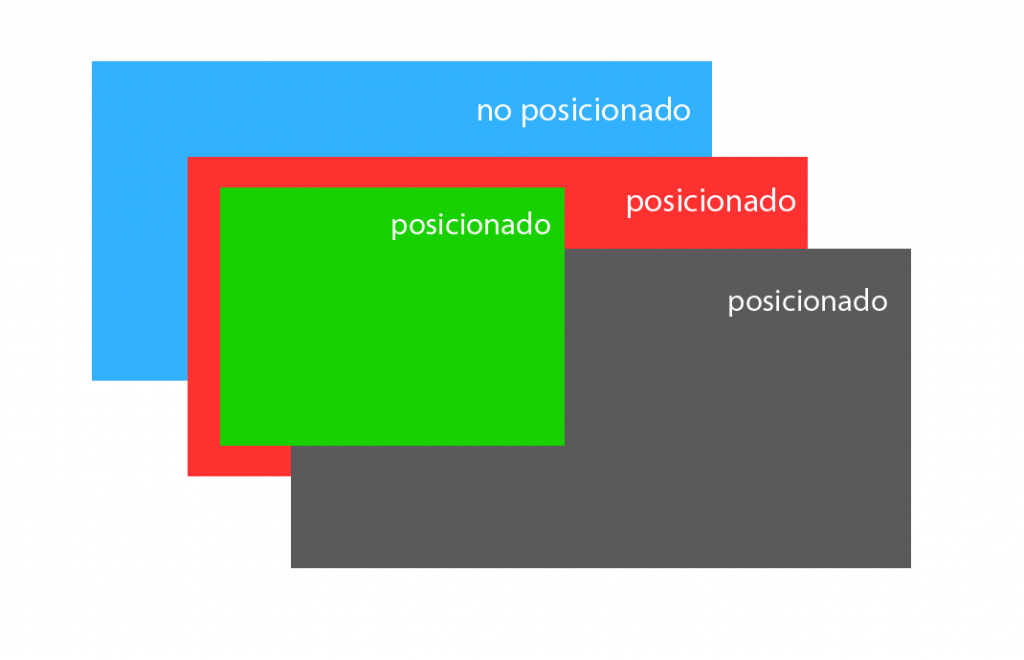
}

.azul {

z-index: 100; // Sin efecto, es un elemento no-posicionado.

}

Obtendremos este resultado:

[](https://cybmeta.com/wp-content/uploads/2018/06/z-index-ejemplo-1a.png)El elemento verde se ha posicionado en lo más alto de eje z

Como se puede ver, los elementos posicionados para los que se ha definido un z-index (el verde y el gris) se ordenan uno sobre otro sin importar que el elemento verde sea descendiente de otro elemento. Se dice que ambos elementos **se ordenan en el mismo contexto de apilamiento**, en el mismo stacking context.

Como se mencionó antes, el stacking context se crea en todos los elementos posicionados fixed o sticky, y en los posicionados con relative y absolute que además tengan un z-index diferente a auto. Y ese es nuestro caso, el elemento rojo está posicionado en absolute pero no tiene z-index, por tanto, este elemento no crea un nuevo contexto de apilamiento y su elemento hijo (el verde) se ordena en el contexto superior, que en este caso es el mismo contexto que el elemento gris.

Vamos a ver como se puede crear un nuevo stacking context. Imaginemos que introducimos un elemento más:

<div class="rojo">

<div class="verde"></div>

</div>

<div class="gris"></div>

<div class="amarillo"></div>

<div class="azul"></div>

Y le aplicamos este CSS:

.gris, .rojo, .verde, .amarillo {

position: absolute;

}

.amarillo {

z-index: 0;

}

.rojo {

z-index: 1;

}

.gris {

z-index: 2;

}

.verde {

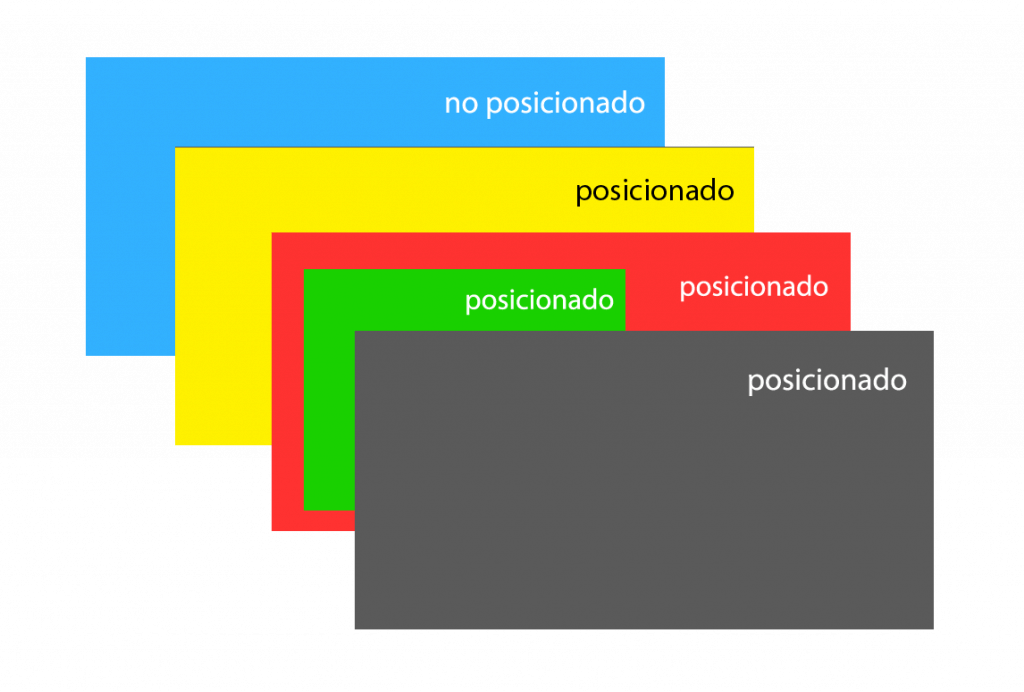
z-index: 3;

}

.azul {

z-index: 100; // Sin efecto, elemento no posicionado

}

[](https://cybmeta.com/wp-content/uploads/2018/06/nuevo-contexto-apilamiento.png)Se ha creado un nuevo contexto de apilamiento en el elemento rosa

Observa el resultado. De todos los elementos posicionados, el amarillo tiene el z-index más bajo y aparece debajo, luego aparece el rojo, hasta aquí bien. Pero fíjate como el verde ya no se superpone al gris aunque el z-index del elemento verde sea mayor.

Antes, el elemento verde y el gris se ordenaban en el mismo contexto de apilamiento, pero ahora, **el elemento rojo ha creado un nuevo contexto de apilamiento**, pues cumple dos reglas: es un elemento posicionado con absolute y tiene un z-index diferente de auto.

Así que el z-index del elemento verde ahora se aplica al contexto de apilamiento creado por su elemento contenedor.

El stacking context se crea en otras muchas más situaciones, tantas que cuesta recordarlas todas, entre las más habituales:

* Elemento raíz del documento
* Elementos position: absolute y position: relative con un valor z-index diferente de auto
* Elementos position: fixed y position: sticky
* Elementos con opacity menor a 1
* Elemento dentro de un **contenedor flex** y un valor z-index diferente de **auto**.

Referencias:

<http://es.learnlayout.com/position-example.html>

<https://cybmeta.com/como-se-utiliza-z-index>

<https://developer.mozilla.org/es/docs/Web/CSS/z-index>