CSS13: VISIBILIDAD visibilidad, solapamiento y otras cosas 18/02/2019

Elementos q se solapan:

Cuando los elementos son colocados fuera del flujo normal de la página, pueden solapar otros elementos.

Si dos elementos se solapan, se verá por encima el que esté colocado último en el código HTML.

La propiedad z-index:

Permite especificar el orden de la pila de elementos, que elemento debe estar delante y cuál detrás independientemente del orden en el código.

La propiedad z-index puede tener valores negativos.

Valores: auto, number (establece el valor al valor deseado, permite números negativos), initial, inherit.

Ocultando elementos:

Se puede esconder un elemento de dos formas:

Display: none; el elemento escondido no ocupa espacio, la página se muestra como si el elemento no existiera.

Visibility: hidden; esconde elemento, pero mantiene su espacio en el layout.

Propiedad visibility:

Visible, hidden, collapse, initial, inherit

Propiedad display:

Nos permite mostrar u ocultar elementos, así como mostrarlos de forma diferente o cómo se ven en su estado normal.

Puede tener muchos valores… pag 13

Overflow:

Cuando un contenido no cabe en in contenedor, podemos indicar qué queremos hacer con la parte que sobresale.

Propiedades: visible(por defecto), hidden (sobrado es cortado), scroll (el sobrante es cortado pero se añade una barra de scroll para poder desplazarse), auto (si el elemento no cabe, automáticamente se pondrá la barra), initial, inherit.

Menús desplegables:

Con las propiedades mencionadas anteriormente (display y overflow) y con los modos de posicionamiento relative y absolute.

En HTML debemos tener listas para los menús y sublistas para submenús.

En el CSS los submenús tendrán position: absolute y display: none.

Al pasar el ratón sobre el padre del submenú, éste cambiará su display a initial.

<ul>

<li>

<ul>

<li></li>

<li></li>

</ul>

</li>

</ul>

Si queremos hacer menús que se desplieguen lentamente, con una transición no usaremos el display none si no que jugaremos con el ancho o alto del submenú.

Se debe poner overflow a hidden para evitar los elementos del submenú sobresalgan cuando éste está cerrado.

Clip:

Permite recortar un elemento que tenga posicionamiento absoluto.

Valores: auto (por defecto), shape (se recorta el elemento, el único valor permitido es:rect (top, right, left, bottom), initial, inherit.

Cambiando el cursor:

Tb exite la posibilidad de cambiar el cursor del ratón a nuestro antojo.

Se hace mediante la propiedad cursor, que puede tener los siguientes valores: url, auto, crosshair, default, pointer, move, e-resize, zoom-in, ..

Con el valor url podemos poner el cursor que nos apetezca a partir de una imagen.

Con ico, png y gif suele funcionar, porque hay que utilizar imágenes pequeñas (32px por 32px).

Ejemplo: cursor: url(‘imágenes/foto.jpg’), auto;

Flotando elementos:

Cuando flotamos un elemento, se puede colocar a la izquierda o a la derecha, permitiendo a otro elementos rodearlos.

Se usa para colocar imágenes en artículos

Antes del flex y el gri, se usaba tb para maquetar (hacer columnas). Hoy en día no se recomienda.

Propiedades: clear (especifica en qué lado o lados no se permiten elementos flotados).

Float (especifica si un elemento debe flotar o no).

Valores: left, right, both, initial, inherit. En el float en vez de both es none.

<div class=’limpiador’></div>

.limpiador{

Clear.both;

}

Los elementos flotan horizontalmente, lo q significa que solo … pag 34

Si flotamos un elemento a la derecha o izquierda, el texto q se encuentre a continuación se colocará a su alrededor, aunque el texto esté contenido en párrafos.

Valores para la propiedad float:

None, left, right, initial, inherit.

Valores para la propiedad clear: este hace todo lo contrario al float.

None, left (no se permiten elementos flotantes a la izquierda), right, both (no se permite elementos flotantes en ambos lados), initial, inherit.

Problema con el float:

Suele suceder que, si el texto no llega a rodear la imagen completamente, está puede sobresalir (overflow).

Las imágenes con float y los textos sin float.

Se podrá resolver: indicando el overflow: hidden. El navegador no cortará el elemento flotante, en su lugar aumentará la altura del contenedor para que éste entre.

Lo mejor sería añadir hasta que se rellene.

**CSS14: GRID Layout** El modelo de tabla flexible de CSS3

El módulo Grid Layout ofrece un sistema de posicionamiento de los elementos basados en una estructura de tabla, con filas y columnas.

De esta forma es más sencillo para diseñadores ubicar los elementos en la posición deseada.

No sustituye a flex, se complementan.

Una tabla o parrilla (grid) consiste en un elemento padre, con un conjunto de elementos hijos.

Existen contenedores grid que tienen filas columnas celdas áreas

Elementos grid que se colocan en las diferentes posiciones de la parrilla.

Contenedores y elementos:

Para hacer q un elemento se comporte como un contenedor grid, debemos ponerle la propiedad display:grid o inline-grid.

Los elementos contenidos en un grid-container.

Si mofidicamos la propiedad grid-template-colums podemos hacer más columnas. Ejemplo grid—auto auto auto;

Espacio entre columnas grid-column-gap y el espacio entre filas grid- row-gap y grid-gap.

Se puede ajustar el espacio entre filas y columnas usando las propiedas grid-column-gap, grid-row-gap

Grid-template-columns define el número de columnas en nuestro layout pudiendo definir además el ancho para cada columna (incluso con el valor auto). Recomendación del profe que por lo menos uno sea auto.

Dicha propiedad es una lista de valores separados por espacios, donde cada valor es el ancho de la columna respectiva.

Grid-template-rows define alto para cada fila.

Solo la deberíamos usar en casos concretos (por ejemplo, una galería fotográfica...)

El alto de los elementos depende del contenido.

No se utiliza casi nunca.

**19/02/2019**

El display: grid Hace una tabla responsiva.

Alineando el contenido:

La propiedad justify-content se usa para alinear horizontalmente toda la parrilla dentro del contenedor. Valores: start, end, center, space-around y space-between.

Align-content: se usa para alinear verticalmente. Valores: start, end, center, space-around y space-between.

Colocando elementos:

Un contenedor grid contiene elementos grid, por defecto un elemento por cada columna y fila.

Además, se colocan automáticamente.

Sin embargo, se pueden colocar los elementos libremente en filas y columnas, aunque no les corresponda por su orden en el HTML.

Se pueden poner elementos que ocupen más de una fila, más de una columna o un área con varias filas y columnas.

Podemos referirnos al número de fila y columna donde queremos ubicar dicho elemento:

Grid-column-start: columna inicial.

Grid-column-end: columna final.

Grid-column-:

Las líneas entre columnas column lines y las líneas entre filas rows lines.

Podemos tener un Grid dentro de grid.

Grid-column:

Es la propiedad abreviada para grid-column-start + grid-column-end

Span hablas de celdas

Grid-row:

Es la propiedad abreviada para grid-row-start + grid-row-end

Grid área, permite definir de forma abreviada (grid-row-start, grid-colmn-start, grid-row-end, grid-column-end).

Si usamos la propieda grid-area en un elemento grid, no usaremos otras.

Definiendo áreas:

También se usa para dar nombres a los elementos de la parrilla.

Una vez q un elemento tiene un nombre, puede ser referenciado en la propiedad grid-template-areas del contenedor grid para colocarlo libremente. Pones en el css el grid-area y puedes ponerle el nombre que quieras sin comillas y luego en el grid-template-areas utilizas ese nombre que hemos declarado utilizando las comillas simples.

Cada fila se define entre comillas ’’.

Las columnas se separarán por un espacio dentro de las comillas.

Ejemplo: .grid-container{

Display: grid;

Grid-template-columns: 20% auto 20%;

Grid-template-areas:

‘he he lo’

‘.ma ma’ el punto es para rellenar lo que falte

‘.ma ma’ necesita tener las mismas columnas

‘.ma ma’

‘fo fo fo’

}

Sobre el ejemplo anterior podríamos usar las propiedades grid-template-columns y grid-template-rows para ajustar los anchos y altos de cada una de las zonas de nuestro grid.

No indicaría el alto (excepto en algunos casos como galerías de imágenes), puesto que los altos de las filas deberían ir en relación del contenido.

El ancho que hay dentro del contenedor no se debería indicar con medidas absolutas (px) puesto que se produciría overflow.

https://james-priest.github.io/grid-critters-code/