**GUIA DE POSITION – FUNDAMENTOS**

Comencemos diciendo que:

* **Block** es un modelo de layout, son elementos de bloque que están uno debajo del otro.
* **Inline** es un modelo de layout, son elementos en línea que aparecen uno alado del otro.
* **Display Table** es un modelo de layout, no es el “Table” del HTML.
* **Position** es un modelo de layout.
* **Display Flex** es un nuevo modelo de layout.
* **Display Grid** es un nuevo modelo de layout. (Guía Pendiente…)

**Position** es un **modelo de layout** **(esto es la organización que tendrán los elementos para aparecer en la pantalla)** algo viejito, los nuevos modelos **Flexbox** y **Grid** han abierto un gran panorama de posibilidades y es lo más nuevo que se está utilizando en el desarrollo web, pero he tenido la necesidad de investigar y estudiar en qué consiste.

Antes de comenzar, hay que diferenciar y comprender que una cosa es el "**Modelo de caja**" y otra cosa es el "**Posicionamiento de caja**".

Esta guía trata el tema del **“Posicionamiento de caja”**.

**CONCEPTOS BÁSICOS**

* **Flujo:** El flujo [en HTML] es el orden en el que aparecen los elementos en la vista del navegador, y este mismo [Flujo] está basado en el orden en el que está escrito cada elemento en el código HTML.

**Ejemplo:** En código, primeramente, está escrito el elemento **"h1"**, luego un elemento **"p"**, luego un elemento **"img"** y, por último, un elemento **"p"**, con esta misma secuencia aparecerán los elementos en el navegador, éste es el comportamiento natural, pero si el último **párrafo** apareciera [en el navegador] antes que el **título**, entonces estaríamos rompiendo el flujo natural del HTML. Cuando hacemos RWD necesariamente rompemos con este flujo, por citar otro ejemplo.

**FLUJO ROTO**

*img { float: right; }*

*img { position: absolute; }*

**FLUJO NATURAL**

*<h1> Título </h1>*

*<p> Párrafo </p>*

*<img> Imagen*

*<p> Párrafo </p>*

* **Espacio:** Esto se refiere a aquel espacio que ocupa un elemento en el flujo. Este espacio en ocasiones puede Reservarse o No, dependiendo de las propiedades CSS que se utilicen.

**Ejemplo:** Tomando los elementos y el código anterior.

**ESPACIO NO RESERVADO**

*img {*

*float: right;*

*display: none;*

*}*

**ESPACIO RESERVADO**

*img {*

*float: right;*

*visibility: hidden;*

*}*

**PROPIEDADES DE POSICIONAMIENTO DE CAJAS (Antiguo modelo de layout)**

Hasta donde investigué y encontré, hay por lo menos 4 propiedades CSS comúnmente usadas para el posicionamiento de cajas. Estas 4 las muestro en este documento, pero empezaré por la más utilizada. La primer propiedad CSS que se utiliza comúnmente es “**position**”, y esta nos permite controlar la posición de un elemento en el flujo, y posee los siguientes 5 valores:

* **Static (valor por default)**
* **Relative**
* **Absolute**
* **Fixed**
* **Sticky (valor en fase experimental)**

Hay un concepto que es necesario comprender y recordar, es el siguiente:

* Según varios autores y profesionales, un **ELEMENTO** se encuentra **POSICIONADO** solo cuando posee los valores: **Relative**, **Absolute** o **Fixed**, **solo con estos valores se le puede considerar a un ELEMENTO como POSICIONADO**. Si un elemento posee "**Static**" no se considera posicionado.

**ELEMENTO NO POSICIONADO**

*div { position: static; }*

**ELEMENTO POSICIONADO**

*div { position: absolute; }*

*div { position: relative; }*

*div { position: fixed; }*

Cuando se comprende lo anteriormente dicho entonces pasamos a lo siguiente:

* Cuando un **ELEMENTO** se encuentra **POSICIONADO** se le puede sacar de su flujo natural. Todo **ELEMENTO** que está **POSICIONADO**, o sea, que posee cualquiera de los 3 valores antes mencionados, adquiere automáticamente 5 nuevas propiedades que sólo están disponibles para **ELEMENTOS** **POSICIONADOS**.

Estas nuevas propiedades tienen que ver con coordenadas, son las que se muestran a continuación:

* **top**: Para mover un elemento en relación a la coordenada superior.
* **bottom**: Para mover un elemento en relación a la coordenada inferior.
* **left**: Para mover un elemento en relación a la coordenada izquierda.
* **right**: Para mover un elemento en relación a la coordenada derecha.
* **z-index**: Es una ubicación respecto al eje Z, solo está disponible cuando se utiliza el valor "**z-index**".

Ahora sí, tomando en cuenta lo anterior, a continuación, se muestran los conceptos de los valores de esta propiedad CSS “**Position**”:

* **relative:** Cuando a un elemento se le da este valor aparentemente no sucede nada, **se queda en el mismo lugar (no se desmaqueta)**, visualmente no hay un cambio, pero internamente si. **Pueden utilizarse las propiedades que manipulan las coordenadas** (**top**, **bottom**, **left**, **right**), y pueden darse valores negativos. Ahora, las coordenadas para/de un **"Elemento Relative"** están en relación a su **posición inicial** y aunque se mueva, su **espacio inicial en el flujo queda reservado**.

Hay una cosa más que tomar en cuenta cuando se utiliza el valor "**relative**" en un elemento, y es el tema relacionado con la altura de ese elemento. Generalmente, las alturas se definen de acuerdo a su contenido. Por ejemplo, si utilizáramos la unidad "**%**" con la coordenada "**top**" (**top: 50%;**) para mover nuestro elemento de su posición inicial, **¿con respecto a quién o respecto a qué se separa nuestro elemento?** El cálculo se hace respecto al contenedor padre, o sea, aquel que contiene a nuestro "**Elemento Relative**", pero este cálculo solo puede realizarse cuando el contenedor padre posee un alto ya definido, sino posee medidas definidas entonces no tendrá efecto el cálculo. Eso no ocurre con “**left**” o “**right**”, aunque el contenedor padre no tenga un ancho definido, el "**Elemento Relative**" podrá moverse a la derecha o izquierda, porque por default es un "**display: block**“ [si es div] y todo elemento block ocupa todo el ancho (en el sentido horizontal) disponible de la pantalla.

Sintaxis: “**position: relative;**”

Conjunto: “**top: 10px|bottom 10px|left 10px| right 10px;**”

* **absolute:** Cuando a un elemento se le da este valor **se rompe la maquetación, se sale del flujo y su espacio inicial no queda reservado, se convierte en 0.** El ancho o dimensión que posee un "**Elemento Absolute**" es **respecto/definido al contenido que tenga**. En cuanto al cálculo o manejo de las coordenadas (**top**, **bottom**, **left**, **right**) sobre un " **Elemento Absolute** " es **respecto al ancestro más cercano que este posicionado**, o sea, que tenga declarado algún valor de posición (**relative**, **absolute**, **fixed**; es muy común que ese ancestro tenga **relative**). En otras palabras, primero busca a su padre, si su padre no está posicionado busca al padre de este padre, si este no está posicionado sigue haciendo la búsqueda hacia arriba, **si ninguno está posicionado se calcula respecto al “body” o viewport**.

Sintaxis: “**position: absolute;**”

Conjunto: “**top: 10px|bottom 10px|left 10px| right 10px;**”

* **fixed:** Este valor tiene muchas similitudes con el valor "**absolute**", pues también **pierde su tamaño en la maquetación del flujo, o sea, el espacio que ocupaba se pierde.** El ancho o dimensión que posee un “**Elemento Fixed**" es **respecto/definido al contenido que tenga**. En cuanto al cálculo o manejo de las coordenadas (**top**, **bottom**, **left**, **right**) **siempre son relativas a la ventana del navegador o viewport**. El "**Elemento Fixed**" se queda fijado/pegado en la posición que se le dé, aunque se le haga scroll, el elemento se quedara fijo/pegado en la posición dada.

Sintaxis: “**position: fixed;**”

Conjunto: “**top: 10px|bottom 10px|left 10px| right 10px;**”

**NOTA: Parece ser que estos valores se le pueden dar a contenedores padre y lo heredarían los hijos, pero también se le pueden aplicar a los mismos elementos hijos.**

Como mencioné al inicio del punto **“Propiedades De Posicionamiento de Cajas”**, hay por lo menos 4 propiedades CSS que comúnmente son usadas para posicionar cajas, inicié con la propiedad “**Position**” y ahora mostraré las otras 3 propiedades CSS que nos permiten posicionar cajas aparte de "**position**", a continuación, se mencionarán esas tres curiosas propiedades:

* **float:** Esta es otra propiedad CSS que se utiliza directamente escribiendo "**float**", no es un valor de la propiedad "**position**". **Cuando esta propiedad se le aplica/activa a un contenedor o elemento** **toma/muestra un comportamiento muy similar a como si utilizáramos "display:flex" y "flex-direction:row|row-reverse"**, pues pone los elementos en fila (horizontal) uno alado del otro, el problema es que esta propiedad junta mucho estos elementos y no deja un espacio de separación entre ellos, teniendo así que calcular manualmente “**margin’s**” o “**padding’s**” para lograr una separación uniforme. **Posee 2 valores: left (izquierda) y right (derecha).**

Sintaxis: “**float: left|right;**”

* **inline-block:** Este es un valor que pertenece a otra propiedad CSS llamada "**display**". **Cuando esta propiedad se le aplica/activa a un contenedor o elemento toma/muestra un comportamiento muy similar a como si utilizáramos "display:flex" y "flex-direction:row|row-reverse"**, pues pone los elementos en fila (horizontal) uno alado del otro, a comparación de "**float**", esta propiedad si agrega una pequeña separación entre los elementos que están juntos.

Sintaxis: “**display: inline-block;**”

* **z-index:** Es una propiedad que nos permite definir la posición en el eje z (profundidad) de **Elementos Posicionados**. Se considera una buena práctica separar los "**z-index**" por rangos amplios. Cuando se le quiere aplicar este valor a pseudoelementos solo pueden utilizarse dos valores: el 1 o el -1, siempre y cuando el elemento que genere estos pseudoelementos no tenga declarado un "**z-index**".

Sintaxis: “**z-index: 100;**”

Conjunto: “**top: 10px|bottom 10px|left 10px| right 10px;**”

**CSS**

*section { position: relative; }*

*.UNO { z-index: 1000 }*

*.DOS { z-index: 500 }*

*.TRES { z-index: 1 }*

**HTML**

*.section*

*div .UNO*

*div .DOS*

*div .TRES*

**NOTA: De estas 3 últimas, las primeras 2 propiedades tienen sus pros y sus contras, dependiendo de la situación y de la necesidad en el momento es la que se utilizara. En lo personal, he visto las diferencias en el video de donde saco esta información y concluyo que "FLEX-BOX" soluciona estos problemas.** **FLEX-BOX** **es muy útil, pero eso no significa que "float" e "inline-block" sean desechadas.**

**REFERENCIAS**

* Muchos Vídeos de EDteam relacionados con los temas.
* Algunas páginas que almaceno en mis marcadores de Google Chrome, en la sección Relacionado a páginas web>Utilidades web>Consejos Prácticos.
* Vídeos de otros profesores que están en Youtube, y cuyos vídeos almaceno en mi disco duro.