Introducción al Máster en CSS, Responsive, SASS, Less, Flexbox, Grid y más

1. ¿Que aprehenderás en el máster?
2. HTML y CSS
3. Responsive
4. Animaciones
5. Preprocesador CSS SASS
6. Preprocesador LESS
7. Maquetación Flexbox
8. Maquetación CSS Grid Layout
9. Framework Bootstrap 4
10. Maquetación web profesional
11. Desarrollo web front end

Introducción CSS

1. Vincular CSS y HTML
2. Selectores
3. Fuentes y colores
4. Fondos y textos
5. Modelos de cajas y posicionamiento
6. Sombras y efectos
7. Posicionamientos
8. Pseudoclases, transiciones, animaciones
9. Proyecto Completo

1. Primeros pasos CSS

CSS son hojas de estilo en cascada (Cascading Stle Sheets). Es un lenguaje de estilos utilizado para dar diseño a la presentación de documentos HTML o XML.

Para linkear nuestro CSS con HTML simplemente ponemos la etiqueta link en el head de HTML, le especificamos la relacion stylesheet y el href al PATH relativo del archivo CSS.

Es importante tener nuestro CSS en un archivo aparte y no tenerlo nivel de página o a nivel de etiqueta.

2. Selectores CSS

Tipos de selectores:

- Selector universal: *
- Selectores de etiqueta: body, footer, etc
- Selectores de id: #ID
- Selectores de clases: .Clase
- Selectores de atributo: ETIQUETA["ATRIBUTO"]. Ejemplo: input[type="text"]{}
- Selectores hijo: PADRE > HIJO > ...n > OBJETIVO.

El nivel de especificidad ira de CSS->nivel de página->nivel de etiqueta o selector de etiquetas->clase->id o CSS de arriba hacia abajo. Con la propiedad !important al lado de una propiedad CSS dara mayor especificadad a todo el documento.

3. Fuentes y colores

Fuentes externas: https://fonts.google.com/ -> Google o cualquier otra página nos va a proporcionar un import CSS o un link HTML y la familia para aplicarle la fuente en el CSS.

Si descargamos una fuente y la importamos a nuestro archivo debemos utilizar la propiedad @font-face {font-family: NOMBRE_FAMILIA; src: url(PATH_RELATIVO);}

La propiedad font-family nos permite cambiar la familia de fuentes que queremos que despliegue el navegador. Es imporante que en caso que una fuente falle por motivo X tengamos una lista, ya que si falla en cargar la siguiente declarada en el font family se intentara cargar.

Puedo indicar dentro del font family el font-weight y font-style para esta fuente.

Puedo tener mas de un src para cargar la fuente al font face, en caso de que falle una cargue continua con el siguiente src.

Tenemos varios tipos de escalas para el font-size. Las mas utilizadas son px, em y %.

1 em = 16px.

100% = normal | 105% = normal + 5 % | 95% = normal - 5%.

Generador de colores CSS: https://htmlcolorcodes.com/es/

Existen los colores por default de HTML, código hexadecimal, rgb, rgba, hsl, hsla (a es alpha, para darle mayor o menor transparencia).

4. Fondos y textos

- background-repeat: Vamos a indicar si queremos que la imagne de fondo se repita para generar un patron en el fondo del sitio web.
- background-size: La cantidad de espacio del viewport que va a ocupar la imagen.
- background-attachment: Determina el comportamiento de la imagen, si va a acompañar el scroll o va a quedarse fija.
- background-position: Permite mover el eje que vamos a ver de la imagen.

5. Modelos de cajas - Margenes, paddings, alto ancho y bordes en CSS

El modelo de caja se compone del margen (separa los otros elementos HTML), border (genera un borde al contenido), padding (Genera un espaciado del contenido con el borde), y el contenido en sí.

Con la propiedad float vamos a poder posicionar cajas a nuestro gusto. Float me parmite flotar elementos en el eje Y.

Float es aplicable a todos los elementos de bloque.

Con la propiedad display podemos cambiar el tipo a en bloque, en linea, combinacion de ambos y si queremos ocultar ese elemento.

La propiedad clear nos permite resetear la propiedad de los flotados para volver a colocar elementos en bloque en flotado por defecto.

La propiedad la vamos a colocar en una clase la cual la vamos a asginar a un div luego de la caja flotada. También debemos hacer un float: none;

Float:none; tells the elements that you do not wish for it to float.

Clear tells other elements whether they should be allowed to float or not, and in the case of none, you're allowing floats on both sides. it's why when you use clear:both; that floating stops.

6. Sombras y efectos

La propiedad text/box-shadow agrega sombra a un elemento/contenido. Su nomenclatura es la siguiente: *-shadow: EJE-X-POSICION EJE-Y-POSICION CANTIDAD-DIFUMINADO COLOR.

Puedo ponerle la propiedad inset a un box shadow para que la sombra se genere dentro de la caja y no fuera.

border-radius generara un redondeo en las esquinas de la caja de X cantidad

7. Overflow y posicionamientos.

La propiedad overflow nos va a permitir controlar lo que se sale de una caja y lo que se queda dentro. Puedo indicar overflow para tanto el eje y, eje x, o ambos.

overflow: hidden; Todo lo que salga afuera se oculta.

overflow: visible; Es el por defecto. Por mas que se salga el contenido se sigue mostrando.

overflow: scroll; Agrega una barra de scroll lateral para ver todo el contenido

En general para los posicionamiento vamos a utilizar las propiedades top, right, left, bottom. Va a indicar la posicion en unidades de medida de donde queremos que se encuentre ese elemento.

- El posicionamiento estático mantendrá el div en la posición por default, sin poder modificar su posición.
- El posicionamiento relativo mantendrá el div en la posición por default o la que le indiquemos en el CSS.
- El posicionamiento absoluto es similar a la static, se utiliza normalmente cuando estamos trabajando dentro de otro elemento de tipo div. El absolute se va a posicionar de manera "absoluta", pero teniendo en cuenta el elemento que lo contiene. Si no lo contiene ningún elemento se va a tomar en cuenta el body (En este caso, si solo tiene el body es una buena práctica utilizar el default). El absolut no funciona si su div padre es static. Se va comportar como un elemento de línea.
- El posicionamiento fijo se va a colocar de manera relativa al viewport. Se va a mantener de manera fija en la posición de pantalla que le indicamos/que tiene por default. Si tenemos un scroll vertical y utilizamos el scroll, este elemento va a bajar conforme se utilice la barra de scroll.

- El posicionamiento sticky se caracteriza por tener que usar alguna de las propiedades de top, left, bottom, right. Se posiciona como un elemento de manera relativa y a la hora de mover la pantalla se utiliza como un elemento de manera fixed. Si trabajamos con safari debemos tener en cuenta una corrección, debemos agregar position: -webkit-sticky;. La posición sticky no funciona en Internet Explorer.

Hasta ahora trabajamos con el eje de X (horizontal) y el eje Y (vertical). El eje Z representa las capas de profundidad de un elemento dado.

En el eje Z se van a ir superponiendo los elementos según coincidan y según el orden en que se van agregando se van a ir superponiendo. Sin embargo, la superposición va a depender de varios factores. Los elementos static son elementos no posicionados, en cambio cuando es diferente la propiedad decimos que son posicionados. Por default se agregan de manera automática pero podemos personalizarlo (tanto negativo como positivo), también podemos personalizar tamaño del z-index.

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

8. Pseudoclases, transiciones, animaciones

Una pseudoclase CSS es una palabra clave que se añade a los selectores y que especifica un estado especial del elemento seleccionado. Por ejemplo, :hover aplicará un estilo cuando el usuario haga hover sobre el elemento especificado por el selector.

Las pseudoclases, junto con los pseudoelementos, permiten aplicar un estilo a un elemento no sólo en relación con el contenido del árbol de documento, sino también en relación a factores externos como el historial del navegador (:visited, por ejemplo), el estado de su contenido (como :checked en algunos elementos de formulario), o la posición del ratón (como :hover que permite saber si el ratón está encima de un elemento o no).

https://developer.mozilla.org/es/docs/Web/CSS/Pseudo-classes

https://developer.mozilla.org/es/docs/Web/CSS/Pseudo-elements

Para generar una transición vamos a utilizar la propiedad transition: PROPIEDAD_CSS TIEMPO;

https://developer.mozilla.org/es/docs/Web/CSS/CSS_Transitions/Using_CSS_transitions

Las animaciones se pueden lanzar de manera infinita, añadiendo o quitando una clase al elemento HTML, etc.

Las animaciones estan basadas en keyframes. Van del punto 0 al punto 100.

Para generar keyframes utilizamos el metodo CSS kreyframe. Ejemplo:

@keyframes nombre {

from {PROPIEDADES CON LAS QUE EMPIEZA}

to {PROPIEDADES CON LAS QUE TERMINA}

------ O PUEDO TAMBIEN UTILIZAR PORCENTAJES ------

0%{PROPIEDADES AL 0% DEL TIEMPO}

50%{PROPIEDADES AL 50% DEL TIEMPO}

100%{PROPIEDADES AL 100% DEL TIEMPO}

}

Puedo combinar from(0%) y to(100%) con porcentajes.

- animation-name: NOMBRE; -> Asignamos la animacion al elemento CSS
- animation-duration: TIEMPO; -> Tiempo que dura la animacion desde el from hasta finalizar el to.
- animation-iteration-count: CANTIDAD/infinite; -> Indico la cantidad de veces que vamos a repetir la animacion o si la queres hacer infinita.
- animation-timing-function: COMPORTAMIENTO; -> Cambio el comportamiento de como se va a comportar el CSS. Por ejemplo si quiero que arranque y finalice lento, y en el medio ir rapido, tomar el tiempo de la misma manera para arrancar, ir y terminar, etc.

https://developer.mozilla.org/es/docs/Web/CSS/CSS_Animations/Using_CSS_animations

Responsive Web Design

1. ¿Qué es el Responsive?
2. Maquetación con porcentajes
3. Media queris
4. Viewport
5. Adaptar web a cualquier pantalla
6. Proyecto Completo

1. ¿Qué es el Responsive Web Design o el Diseño Web Adaptable?

Definicion: https://es.wikipedia.org/wiki/Dise%C3%B1o_web_adaptable

El Responsive Web Design consiste de técnicas para que los estilos sean capaces de adaptarse a cualquier tipo de pantalla.

2. Maquetación en porcentajes

Para maquetear tengo que pensar en las cajas/containers como porcentajes y no como pixeles porque un porcentaje es adaptado/se adapta al tamaño de la caja que lo contiene, en cambio los pixeles son fijos. El porcentaje puede variar con respecto al elemento padre que lo contiene, con pixeles no.

3. Media queries

Lo importante del responsive es capturar los puntos de corte (ancho x largo en px) sobre los cuales se produce un cambio visual muy marcado y partir de ahi asiganrle estilos. Cuando hablamos de cambio hablamos a que el viewport llegue a cierto punto en el cual los elementos se vuelven ilegibles, no tiene un comportamiento adecuado, visualmente no nos gusta, etc.

Los media queries de declaran con @media(){} y le pasamos, por ejemplo, un max-width para que sepa que si el max es menor a esos px se ejecute este nuevo CSS. Entre las llaves llamamos a nuestros estilos y sobre escribimos los elementos que deseamos cambiar.

Para combinar mas de una condicion utilizamos el operador and y agregamos unos nuevos ().

4. Viewport

El viewport es la pantalla disponible del total del monitor. Por ejemplo la barra de tareas, pestañas, favoritos, etc. No son parte del viewport, es todo lo que el HTML renderiza.

La etiqueta <meta name="viewport" content="width=device-width, initial-scale=1.0"> vamos a poder implementar media queries segun el width de la pantalla donde se encuentre. Ademas podemos inidcarle la escala con la que empieza, en el ejemplo el initial scale x1.0, o sea, escala normal.

Para deshabilitar el zoom con los dos dedos podemos utilizar "user-scalable=no".

SASS

1. ¿Qué es un preprocesador CSS?
2. Instalar SASS
3. Variables y compilación
4. Anidación
5. Módulos
6. Mixins
7. Herencia
8. Operadores
9. Condicionales
10. Bucles

1. ¿Qué es un preprocesador css y para qué sirve?

https://developer.mozilla.org/es/docs/Glossary/CSS_preprocessor

https://sass-lang.com/

SASS es un preprocesador que añade capas de funcionalidades a CSS. Dotando a CSS de ciertas caracteristicas de un lenguaje de programación.

SASS cuenta con dos tipos de sintaxis.

SCSS: Requiere de puntos, comas, punto y coma, llaves, etc.

Sass: Parecida a Python, no requiere de llaves, punto y coma, etc.

2. Variables y compilación

Las variables se declaran con el signo '$':'Dato/s'.

La compilación del archivo SASS la realizamos desde una terminal de comandos. Dentro del proyecto ubicamos nuestro archivo SASS/SCSS realizamos el comando sass --watch ARCHIVO.SASS PATH*_ARCHIVODESTINO.css

La flag --watch le va a indicar a SASS que este atento a cambios dentro del archivo a compilar y que recompile en el mismo PATH si se produce un guardado. El watch durará siempre y cuando no cerremos la terminal de comandos.

Tambien la compilación genera un archivo .map

A source map is a .map file in JSON format that gets generated when your Sass files are compiled into CSS. It contains information that links each line of your CSS file (the output) to the corresponding line in its SCSS source file (the input). https://www.sitepoint.com/using-source-maps-debug-sass-chrome/#:~:text=A%20source%20map%20is%20a,source%20file%20(the%20input).

3. Módulos

Los modulos que son partes parciales de una gran clase comienzan con el _ en su nombre por convención, así le indicaremos a otros desarrolladores que ese archivo es una partial class.

Simplemente utilizamos @use 'PATH'; y ya estaremos importando el SASS. Los modulos parciales y las extensiones no son necesarias, simplemente ponemos el nombre del archivo.

4. Mixins con SASS

Los Mixins son "funciones" que por parámetros asignan un valor a una variable dependiendo desde donde la ejecutemos y los parámetros que les pasemos. Si no le pasamos una variable o no le asignamos un valor por defecto puede romper nuestro CSS. Las llamamos con el @include MIXIN() en el scope a utilizar.

Ejemplo:

```
@mixin theme($theme: DarkGray) {
background: $theme;
box-shadow: 0 0 1px rgba($theme, .25);
color: #fff;
}
.info {
@include theme;
}
.alert {
@include theme($theme: DarkRed);
}
```

.success {

@include theme($theme: DarkGreen);

5. Herencia

Los estilos heredables en CSS llevan delante el signo %. Para utilizarlo en otra clase utilizo la palabra reservada @extender %NOMBRE;

6. Operadores

SASS nos permite hacer operaciones matematicas dentro de CSS.

7. Condicionales

SASS nos permite hacer operaciones logicas dentro de CSS, incluye bucles y condicionales. Ambos comparten las operaciones +, -, and, or, not, *, /, %, <, <=, >, >=, ==, !=, = cuando este disponible.

Sintaxis: @if EXPRESION, @else EXPRESION, @else if EXPRESION

8. Bucles

Sintaxis: @for $VARIABLE from START_POINT through END_POINT, @each $v in $VAR, @while CONDITION

Para acceder a una variable en un parámetro utilizamos la nomenclatura #{$VARIABLE}

LESS

1. Introducción a LESS
2. Instalar LESS
3. Variables y compilación
4. Anidación
5. Mixins
6. Operadores
7. Condicionales
8. Bucles

1. LESS

https://lesscss.org/

Las variables en LESS se declaran con el @NOMBRE: DATO;

@{b}{DATOS} -> De esta manera puedo acceder a un elemento HTML guardado en una variable.

Para compilar el CSS utilizamos el comando: lessc ARCHIVO.LESS DESTINO.CSS. El destino lo crea LESS en caso de no existir.

2. Mixins y anidación

LESS también incluye las "funciones" mixin y la anidación.

Anidación de igual manera que SASS

Para delcarar un mixins utilizamos la siguiente sintaxis:
.NOMBRE_MIXIN(@PARAMETROS:VALORPORDEFECTO*);

3. Operadores, mixins de control y bucles

Misma cantidad de operadores que SASS.

.CLASE(@PARAMETROS) when (CONDICION){} Seria un if para ejecutar una clase. Si queremos una condición DENTRO de una clase simplemente utilizamos la palabra reservada if((CONDICIONES), [EJECUTA1, EJECUTA2, EJECUTA_N])

La siguiente sintaxis utilizamos para ejecutar una clase en bucle:

```
.loop(@cont) when (@cont > 0){
    .loop((@cont -1));
    CONDICIONES A EJECUTAR
}
```

La siguiente sintaxis para ejecutar codigo dentro de una clase en bucle:

```
each(range(4), {
  .col-@{value} {
    height: (@value * 50px);
  }
});
```

Flexbox

1. Introducción
2. Displays
3. Direcciones
4. Envoltorio
5. Orden
6. Grow y Shrink
7. Flex basis
8. Variables y funciones de CSS
9. Alineación
10. Maquetación completa en Flexbox

Links de interes y de practica:

https://developer.mozilla.org/es/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox

https://flexboxfroggy.com/#es

http://www.flexboxdefense.com/

1. Introducción

Flexbox es un módulo de CSS que define un nuevo modelo de cajas. Es una forma diferente a maquetar con los floats y los positions.

Optimiza y flexibiliza a la hora de maquetar y hacer responsive.

El diseño flex permite distribuir los elementos en dirección vertical u horizontal.

Se pueden flexibilizar los tamaños de manera automática. Todas estas cajas van a ir aumentando de tamaño en función de si tienen elementos al lado o no, etc.

Viene a "reemplazar" el flotado, es una tecnología muy utilizada por frameworks de CSS.

2. Conceptos Flexbox

Flebox tiene una particularidad, debemos indicarle la caja que va a contener nuestras cajas flexibles. No puedo trabajar con flexbox seleccionando etiquetas sueltas. Debo organizar todos los elementos dentro de una caja.

3. Display flex e inline-flex

https://developer.mozilla.org/es/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox#el_contenedor_flex

Un área del documento que contiene un flexbox es llamada contendedor flex. Para crear un contenedor flex, establecemos la propiedad del área del contenedor display como flex o inline-flex. Tan pronto como hacemos esto, los hijos directos de este contenedor se vuelven ítems flex. Como con todas las propiedades de CSS, se definen algunos valores iniciales, así que cuando creemos un contenedor flex todos los ítems flex contenidos se comportarán de la siguiente manera:

- Los ítems se despliegan sobre una fila (la propiedad flex-direction por defecto es row).
- Los ítems empiezan desde el margen inicial sobre el eje principal.
- Los ítems no se ajustan en la dimensión principal, pero se pueden contraer.
- Los ítems se ajustarán para llenar el tamaño del eje cruzado.
- La propiedad flex-basis es definida como auto.
- La propiedad flex-wrap es definida como nowrap.

Flex e inline-flex ambos aplican el diseño flexible a los hijos del contenedor. El contenedor con display:flex se comporta como un elemento a nivel de bloque, mientras que display:inline-flex hace que el contenedor se comporte como un elemento en línea.

4. Direcciones

Puedo setear la direccion de los elementos que componen mi flexbox como filas o columnas. Puedo indicar que sea una alineación de izquierda a derecha o reverse (derecha->izquierda) para filas o de arriba hacia abajo si es column y si es reverse de abajo hacia arriba.

La propiedad CSS justify-content define cómo el navegador distribuye el espacio entre y alrededor de los items flex, a lo largo del eje principal de su contenedor.

5. Envoltorio en flex

https://developer.mozilla.org/es/docs/Web/CSS/flex-wrap

https://developer.mozilla.org/es/docs/Web/CSS/flex-flow

Flexbox tiene un problema al declararse por defecto. No establece un maximo de elementos por linea. Lo que puede generar un overflow en caso de no correguirlo en el CSS.

Para que esto no suceda podemos utilizar el flex-wrap para que se comporte como un envoltorio y no suceda el overflow del width.

La propiedad flex-wrap de CSS especifica si los elementos "hijos" son obligados a permanecer en una misma línea o pueden fluir en varias líneas. Si la cobertura (wrap) está permitida, esta propiedad también te permite controlar la dirección en la cual serán apilados los elementos.

Tambien existe otra propiedad llamada flex-flow que une las propiedades flex-direction y flex-wrap

6. Orden

Existe la propiedad order que no es propia de FlexBox sino que debe utilizarse dentro de los elementos que compone el mismo. Dando un order vamos a esteblacer una jerarquía a los elementos para posicionarse en cierto lugar. Para tener un orden positivo debemos establecer el numero a todos los elementos. En caso de no tenerlo utilizamos numeros negativos para dar el orden.

7. Grow

https://developer.mozilla.org/es/docs/Web/CSS/flex-grow

La propiedad flex-grow de CSS especifica el factor de crecimiento de un elemento flexible (que tiene asignado display:flex), en su dirección principal. El factor de crecimiento especifica qué cantidad del espacio restante dentro del contenedor flexible, debería ocupar el item en cuestión.

Es una propiedad especialmente util a la hora de hacer responsive un flexbox. Es una propiedad utilizada dentro de los elementos hijos del flexbox.

flex-grow: NUMERO; -> El numero indica que tanto espacio va a ocupar el elemento en si, puedo especificar que uno elemento sea mas grande que otro poniendole un numero distinto. Se podria pensar como "Dale esta X a tanto y a Y dale N cantidad de veces mas espacio que a X"

El espacio restante es el tamaño del contenedor flexible menos el tamaño de todos los elementos flexibles juntos. Si todos los ítems dentro del contenedor tienen el mismo factor de crecimiento, todos los elementos reciben la misma cantidad del espacio restante. De lo contrario, el espacio restante se distribuye en función de los diferentes factores de crecimientos de cada item.

8. Shrink

https://developer.mozilla.org/es/docs/Web/CSS/flex-shrink

https://stackoverflow.com/questions/42820888/flex-shrink-not-working-as-expected

La propiedad CSS flex-shrink especifica el factor de contracción de un flex item. Los flex items se encogerán para llenar el contenedor de acuerdo a su número flex-shrink, cuando el tamaño por defecto de los flex items sea mayor al de su contenedor flex container.

flex-shrink is designed to distribute negative free space in the container.

In other words, it only works when flex items are big enough to overflow the container.

You're not having that problem here. There is no negative space. Therefore, I don't believe flex-shrink is having any effect on your layout.

flex-grow is consuming the positive free space and seems to be working fine.

You would need to set a flex-basis or add content to the items to put flex-shrink in play.

9. Flex basis, grow, calc y variables CSS

https://developer.mozilla.org/es/docs/Web/CSS/flex-basis

La propiedad de CSS flex-basis especifíca la base flexible, la cual es el tamaño inicial de un elemento flexible. Ésta propiedad determina el tamaño de una caja de contenidos a no ser que se haya especificado de otra forma usando box-sizing.

Puedo declarar variables propias de CSS con la sintaxis --NOMBRE: DATO; y llamarlas dentro de un scope con la etiqueta var(--VARIABLE);

10. Alineación horizontal

- justify-content: space-around; -> Mismo espacio entre ellos dejando un espacio con el margen
- justify-content: space-around; -> Mismo espacio entre ellos contando el mismo espacio con el margen
- justify-content: space-between; -> Espacio entre ellos sin contar el margen
- justify-content: center; -> Centra el contenido
- justify-content: flex-end; -> Alinear items flex desde el comienzo
- justify-content: flex-start; -> Alinear items desde el final
- justify-content: left;
- justify-content: right;

10. Alineación vertical

https://developer.mozilla.org/es/docs/Web/CSS/align-items

https://developer.mozilla.org/es/docs/Web/CSS/align-self

Grid Layout

1. Introducción a CSS Grid Layout
2. Columnas
3. Fracciones
4. Filas
5. Expansión
6. Maquetación completa
7. Template Areas
8. Grid Gap
9. De FlexBox a Grid
10. Responsive con Grid

1. ¿Qué es CSS Grid Layout?

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

CSS Grid no es mas que una nueva tecnología nativa de maqueteado CSS que, a diferencia de Flex, es bidireccional/bidimensional. Podemos crear una rejilla tanto horizontalmente como verticalmente en columnas y filas.

CSS Grid layout contiene funciones de diseño dirigidas a los desarrolladores de aplicaciones web. El CSS grid se puede utilizar para lograr muchos diseños diferentes. También se destaca por permitir dividir una página en áreas o regiones principales, por definir la relación en términos de tamaño, posición y capas entre partes de un control construido a partir de primitivas HTML.

Comparte propiedades con FlexBox, como align items, align self o justify content.

Puedo, una vez realizado el maqueteado Grid, utilizar FlexBox dentro de cada elemento de Grid para que su contenido se organice en cajas y pueda utilizar propiedades de Flex ya que son 100% adaptables al tamaño que lo contienen, a difrencia de Grid que si utilizo align items no va a ocupar el 100% del contenedor, solo lo que ocupe ese elemento en concreto.

2. Primeros pasos con CSS Grid Layout

Grid solamente actua dentro de la etiqueta que englobe los elementos que queremos utilizar en Grid.

Puedo tener Grid dentro de grid. Por ejemplo una section que es afectada por un grid del contenedor, esa section puede ser grid y que sus articles dentro se comporten como un sub-grid del padre. Puedo saltear un elemento padre para hacer grid un elemento mas especifico. Por ejemplo tener un contenedor grid, un <nav> y ese <nav> tener un <ul> para que se aplique grid a los <li>. Quedaria algo asi: Contenedcor(Grid)->Nav(Se acomoda al grid del contedor)->Ul(Genero un sub-grid para sus elementos dentro del "contenedor" nav, que a su vez esta contenido dentro del contenedor general. Nav en ningún momento tiene un display:grid).

display:grid; Dentro de la etiqueta que englobe.

grid-template-columns: X% Y% N%; Indico la cantidad de columnas y cuanto espacio va a ocupar cada una, podemos utilizar porcentajes o fr (fracción).

3. Fraccion y repeat

Para no repetir poner N cantidad de columnas puedo utilizar la propiedad repeat(CANTIDAD, TAMAÑO); Puedo combinar repeats en caso de tener distintas medidas.

Las fracciones permiten que se ajusten el tamaño automaticamente a la cantidad que tenga al lado. Es decir si tengo cuatro va a ser una fraccion de esas cuatro.

Existe la propiedad auto-fill para que automaticamente se genere un wrap en caso de no tener en claro cuantas columnas o filas ocupar si tengo un tamaño predeterminado que puede generar overflow.

4. Columnas y filas

grid-template-rows: Funciona igual que las columnas.

El tamaño de los elementos se adapta al espacio disponible de las filas y columnas

5. Expandir columnas

- grid-column-start: X; ->En que linea comienza el elemento.
- grid-column-end: X; -> En que linea de columna termina. En caso de enviar algun elemento hacia la linea de abajo se genera una nueva fila y el elemento empujado se coloca en esa nueva fila.
- grid-column: INICIO / FINAL; -> engloba el start y el end.

6. Expandir filas

- grid-row-start: X; -> En que linea comienza el elemento.
- grid-row-end: X; -> En que linea de row termina. En caso de enviar algun elemento hacia la linea de abajo se genera una nueva fila y el elemento empujado se coloca en esa nueva fila.
- grid-row: INICIO / FINAL; -> engloba el start y el end.

7. Grid Responsive

Grid es Responsive y dependiendo del media query podemos modificar la distribución del contenido en caso de ser necesario

8. Grid Template Areas

https://developer.mozilla.org/es/docs/Web/CSS/grid-template-areas

Con las Grid Template Areas vamos a evitar tener que estar expandiendo columnas y filas. De forma mucho mas rapida y responsive vamos a poder formar una maquetación.

grid-template-areas: Se organizan en la cantidad de filas disponible y en la cantidad de columnas. Encapsulo un nombre en una o varias filas/columnas con comillas, luego ese nombre se lo tengo que aplicar a un estilo css con la propiedad grid-area: nombre;

Ejemplo: 5 Columnas 2 filas

grid-template-areas: "cabecera cabecera cabecera cabecera cabecera"
"contenido contenido contenido contenido aside"

```
#cabecera{
    grid-area: cabecera;
}
#aside{
    grid-area: aside;
}
```

```
#contenido {
    grid-area: contenido;
}
```

De esta manera me ahorro la necesidad de escribir por cada CSS cuando va a ocupar o que lineas.

Puedo hacer con un solo juego de comillas, una vez se exceda la cantida de filas o columnas va hacia una nueva linea hacia abajo. En caso de que quede una celda vacia el siguiente elemento HTML ocupara esa celda.

Es recomendable para hacerlo mas legible arbrir comillas por cada fila y enter cuando hayamos terminado las columnas en esa fila.

Si quiero que el siguiente elemento que ocupe mi Grid Area sea cualquiera utilizo el '.', así le indico que el siguiente elemento HTML que encuentre se posicione alli o que se asigne un espacio vacio.

Bootstrap

1. Introducción a Bootstrap
2. Filas y columnas
3. Maquetación
4. Utilidades
5. Componentes

https://getbootstrap.com/docs/4.3/getting-started/introduction/

1. ¿Qué es Bootstrap y para que sirve?

Bootstrap es un framework para CSS. Un framework es un marco de trabajo que proporciona funcionalidades y herramientas ya desarrolladas.

Bootstrap es un toolkit open-source para desarrolladores de HTML, CSS y JS. Incluye SASS, responsive flexbox, grid, componentes pre construidos, etc.

Si quiero utilizar un width, padding, botones, justify-content, etc. Tengo todo en la documentación que indiqué más arriba. No voy a tomar nota salvo que sea necesario o esa algo muy especifico de como funciona bootstrap.

2. Filas y columnas CSS Grid y Flexbox

https://getbootstrap.com/docs/4.0/layout/grid/

Bootstrap utiliza un sistema Grid implementando flexbox y orientado al mobile-first. "Use our powerful mobile-first flexbox grid to build layouts of all shapes and sizes thanks to a twelve column system".

Los layouts suelen tener distintos tipos de tamaños. Suelen ser utilizados para hacer responsive la pagina.

El maximo de columnas en Bootstrap es de 12. Tengo la class column o row para indicar su posicionamineto y para poder utilizar este layout deben estar los elementos encapsulados en una clase container como indica la documentación.

3. Offset

https://getbootstrap.com/docs/4.0/layout/grid/#offsetting-columns

El offset nos va a permitir meter columnas de espaciado entre las diferentes columnas.

Las columnas de espaciado se colocan a la izquierda y no a la derecha.

4. Responsive Web Design

https://getbootstrap.com/docs/4.0/layout/overview/

El container-fluid va a tomar ser flexible la 100% del viewport. De esta manera los elementos que esten contenidos se va a adaptar a este viewport y nos hace el trabajo de responsive mucho mas facil y acotado.

Puedo colocar varias clases que nos da Bootstrap que ya tienen incorporada los media queris para manejar el tamaño de un elemento dependiendo de los pixeles. Ejemplo: class="col-md-12 col-sm-12" -> Así le indicamos que cuando llegue al punto de quiebre que tiene la categoria md (establecido en la documentación) ocupe 12 columnas y cuando llegue al punto de quiebre de sm ocupe 12 columnas.

Nos puede ser muy util para tener mas control y no dejar que se coloquen automaticamente.

La documentación de bootstrap nos provee documentación de los breakpoints establecidos. Es especialmente util para adaptar codigo CSS ajeno a bootstrap.

Bootstrap nos trae las propiedades hidden-TAMAÑO o hidden en general para todos los tamaños de pantalla.

https://getbootstrap.com/docs/4.3/utilities/display/

5. Utilidades en Bootstrap

https://getbootstrap.com/docs/4.3/utilities

Bootstrap nos da un monton de herramientas como altura, anchura, flotados, colores, fondos, sombras, etc. Todo esto esta disponible en la documentación simplemente ingresando el nombre en el buscador.

Para aplicar height en Bootstrap el elemento debe estar contenido dentro de una caja donde se especifique el height, ya sea puesto por nosotros o por bootstrap.

6. Componentes en Bootstrap

https://getbootstrap.com/docs/4.3/components

Los componentes son elementos muy comunes que hay dentro de una página web, por ejemplo, alertas, sliders, cartas, barras de progreso, etc. Bootstrap nos da estos elementos ya desarrollados para que podamos implementarlos.

Los aria-label son simplemente nombres que se guardan para indicar que va a hacer o para que sirve ese elemento HTML.

CAMBIOS EN BOOTSTRAP 5

Bootstrap 4	Bootstrap 5
Utilidades	**Utilidades**
pl-5,pr-5	ps-5,pe-5
rounded-left,rounded-right	rounded-start,rounded-end
float-left,float-right	float-start,float-end
ml-5,mr-5	ms-5,me-5
font-weight-bold	fw-bold
font-weight- light	fw-light
font-italic	fst-italic
Navbar	**Navbar**
data-toggle	data-bs-toggle
Alertas	**Alertas**
data-dismiss	data-bs-dismiss
close	btn-close (añade ya la cruz)
Badge	**Badge**
badge-success	bg-success
badge-pill	rounded-pill
Slider	**Slider**
data-slide	data-bs-slide
sr-only	visually-hiiden
data-slide-to	data-bs-slide-to
Jumbotron	**Jumbotron**
jumbrotron	No disponible en Bootstrap 5 CSS a mano: `.jumbotron { padding: 2rem 1rem; margin-bottom: 2rem; background-color: #e9ecef; border-radius: .3rem; }`
Modal	**Modal**
data-target	data-bs-target
Tooltip	**Tooltip**
data-placement	data-bs-placement
Popover	**Popover**
data-container	data-bs-container
data-content	data-bs-content
Formularios	**Formularios**
form-group	mb-3
No disponible en Bootstrap 4	Añadir a los label: form-label

Input-group-prepend	No disponible en Bootstrap 5
	Ahora solo hace falta esto:
	`<div class="input-group mb-3">`
	`<label class="input-group-text">No`

Tablas	**Tablas**
thead-dark	table-dark