Unidad N°3 Posicionamiento: Flexbox-Grid. Una forma de guardar nuestros proyectos

.

Tabla de contenidos

1. Presentación y objetivo

Presentación

En CSS pudimos ver en los cursos anteriores que contamos con varias formas de posicionar elementos: utilizando la propiedad position- utilizando flexbox o grid. En esta unidad revisaremos estas últimas dos formas y agregaremos detalles sobre más herramientas que nos proveen estas propiedades. También hablaremos de una herramienta que nos va a permitir guardar nuestros proyectos de forma remota, pudiendo acceder y compartirlos de una manera simple y segura.

Objetivo

Que el alumno conozca las propiedades de posicionamiento que ofrece CSS3 y la herramienta git que nos va a permitir la carga de proyectos.

2. Hablemos un poco de posicionamiento

Cuando hablamos de cómo se encuentran posicionados los elementos, a menudo hacemos referencia a una de las maneras en que están distribuidos los elementos de un sitio. Para ello inicialmente en CSS3 contamos con la propiedad position que nos provee de propiedades para ir ubicando a los elementos del sitio en distintas posiciones dentro del navegador.

Estas propiedades con sus variantes (static- relative- absolute- fixed- sticky) nos permiten hacer que el elemento se ubique en un plano y en una coordenada (top- right- left- bottom) del navegador específica. Pero tiene como defecto que vuelve a nuestro sitio propenso a que falle, ya que al agregar nuevos elementos o modificar apenas el tamaño de pantalla, la ubicación que realizamos con esta propiedad se modificará. También nos hace difícil el poder alinear elementos en relación al elemento padre que lo contiene, entre otras cosas.

Por este motivo, se recomienda dejar el uso de position para detalles específicos como: solapar algún texto a una imagen, o hacer fijo alguna sección al scrollear el sitio. Para los posicionamientos generales del sitio, recomendamos las siguientes propiedades ofrecidas por CSS: Flexbox y Grid.

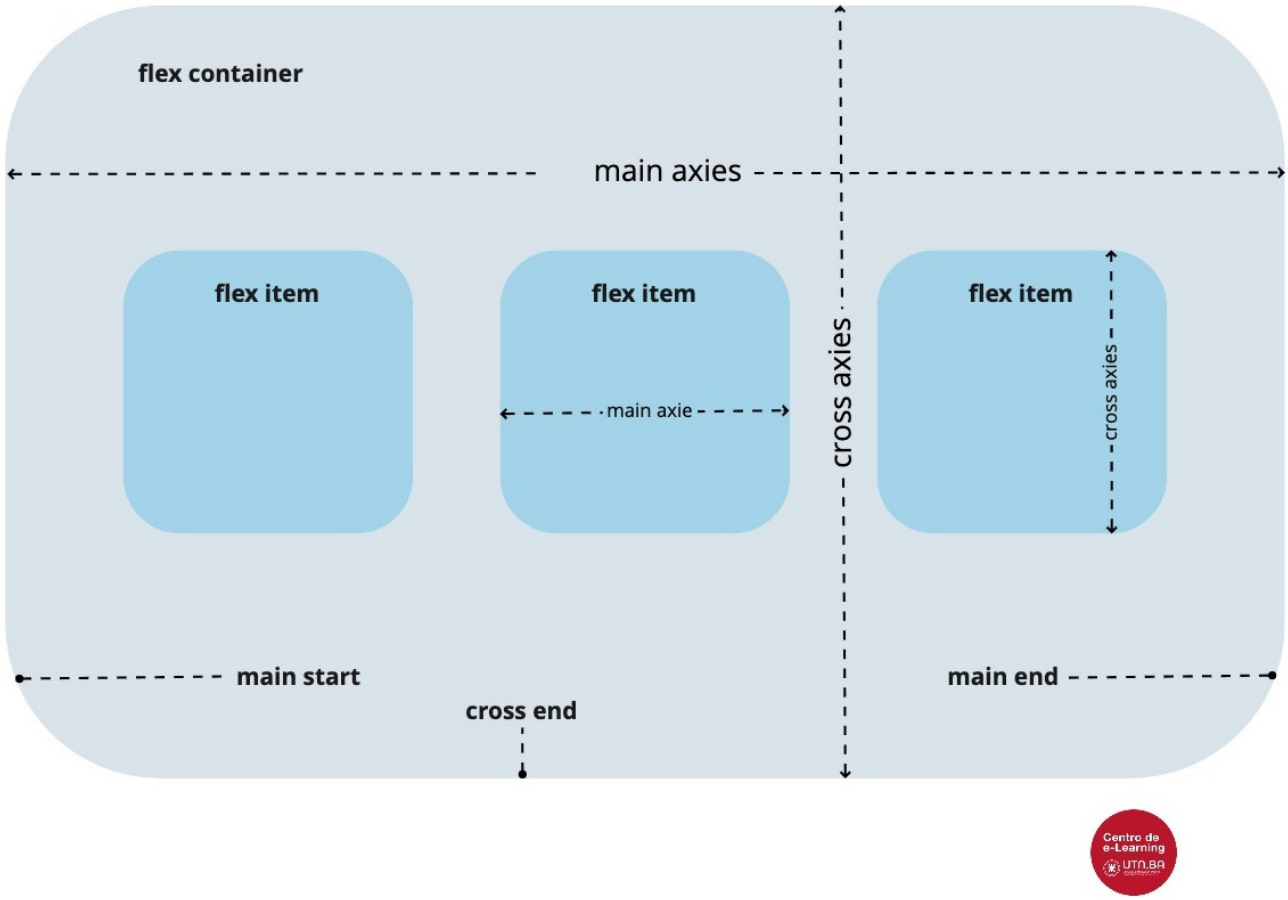
3. Flexbox

Flexbox es un método de diseño de página que se basa en ubicar elementos en filas o columnas.

La idea principal detrás del diseño flexible es darle al contenedor la capacidad de modificar el ancho/alto (y el orden) de sus elementos para llenar mejor el espacio disponible (principalmente para adaptarse a todo tipo de dispositivos de visualización y tamaños de pantalla). Un contenedor flexible expande los elementos para llenar el espacio libre disponible, o los reduce para evitar el desbordamiento. Flexbox es una herramienta muy avanzada para poder crear diseños de maquetación con características mejoradas y necesarias en el día de hoy, donde es tan importante tener una estética cuidada y una gran adaptabilidad a distintos formatos de pantalla. Este módulo se invoca por medio de la propiedad display y contiene una gran cantidad de propiedades que nos provee de distintas características a los elementos.

Al colocar en la propiedad display el valor flex o inline-flex, hacemos que el elemento al que se asignó dicha propiedad y sus hijos sean elementos flexibles. Estos se distribuyen con respecto a dos ejes y nos permiten poder modificar las dimensiones y el orden de los elementos hijos para acomodarlos de distintas maneras controladas por el desarrollador.

Pero .. ¿cuáles son estos ejes? Para poder explicarlo de una manera clara, miremos el siguiente gráfico:



Los elementos se distribuirán siguiendo el main-axis (desde el main-start hasta el main-end) o el cross-axis (desde el cross-start al cross-end).

El eje principal (main axis) es el eje que corre en la dirección en que se colocan los elementos ,es decir de forma horizontal o vertical. El inicio y el final de este eje se denominan inicio principal (main start) y final principal (main end).

El eje transversal (cross axis) es el eje que corre perpendicular a la dirección en la que se colocan los elementos. El inicio y el final de este eje se denominan inicio transversal (cross start) y extremo cruzado (cross end).

Ahora que tenemos una idea de cómo se distribuye flexbox, veamos los valores y propiedades que nos ofrece.

3.1. Valores de flexbox

justify-content: Nos sirve para alinear los contenedores hijos en el eje horizontal, haciendo que queden: centrados, al inicio, al final, o indicar la manera en la que se separan entre sí, etc.

aling-items: Nos sirve para alinear los contenedores hijos en el eje vertical, haciendo que queden: centrados, arriba, abajo, centrados, etc.

flex-direction: Indicamos si queremos que se posicionen los elementos en forma de filas (horizontal) o en forma de columna (vertical).

flex-grow: Especifica qué cantidad del espacio restante dentro del contenedor flexible, debería ocupar el ítem en cuestión. Acepta valores sin unidades que sirven como proporción, pero no acepta valores negativos. Esto quiere decir que, por ejemplo, si todos los elementos tienen el flex-grow establecido en 1, el espacio restante en el contenedor se distribuirá por igual a todos los hijos. Si uno de los hijos tiene un valor de 2, ese hijo ocuparía el doble de espacio que cualquiera de los otros (o lo va a intentar, al menos).

flex-wrap: Permite “envolver” a los componentes hijos para que no cambien su tamaño mientras se achique la pantalla, sino que cuando vayamos achicando la pantalla se desplace hacia abajo manteniendo el tamaño del componente.

flex-shrink: Define la habilidad de un ítem de encogerse si es necesario. Al igual que flex-grow, no acepta valores negativos y utiliza valores sin unidad que sirven como proporción.

flex-basis: Define el tamaño predeterminado de un elemento antes de que se distribuya el espacio restante. Se acepta cualquier valor de longitud (px,rem,%,etc) o palabras clave (auto, content, min-content,max-content, fit-content).

flex-flow: Es una abreviatura de las propiedades flex-direction y flex-wrap que nos permite definir con esta opción el eje principal y transversal del contenedor.

gap: Controla el espacio entre elementos.Tenemos que tener en cuenta que este espacio solo se da entre elementos que no están en los bordes exteriores: solo el espacio entre los elementos.

4. Grid

CSS Grid Layout (también conocido como "Grid" o "CSS Grid") es un sistema de diseño basado en cuadrículas que, al igual que Flexbox, cambia por completo la forma en que diseñamos las interfaces de usuario.

A diferencia de Flexbox, este método utiliza un diseño bidireccional que, al ser una cuadrícula, permite tomar valores de ambos ejes (vertical y horizontal) e incluso permite que se combine con flexbox sin problema, trabajando en completo equilibrio.

Grid es el primer módulo CSS creado específicamente para resolver los problemas de diseño que presentaban las propiedades como position o incluso float.

Para invocar a este método y acceder a todas sus propiedades y valores, debemos definir en el elemento contenedor padre display:grid.

Para poder ver y saber la forma de utilizar las propiedades que nos provee grid, podemos clasificarlas en dos grandes grupos: aquellas que son para utilizar en los componentes padres, y las propiedades por elementos hijos.

4.1. Propiedades para contenedores padres

Empezando con la propiedad display:grid; estas propiedades deben ser definidas en los contenedores padres para que cumplan su funcionalidad. Ellos son:

- grid-template-columns: Esta propiedad establece la cantidad y el ancho de las columnas en que se va a “dividir” el elemento padre. Puede utilizar cualquier unidad de medida para definirlo (%, px, vh, vw) e incluso podemos utilizar una unidad que nos presenta grid, llamada fr (fracción).

La unidad fr representa una fracción del espacio disponible en el contenedor de la cuadrícula, es decir que se expanden y se encogen de acuerdo al espacio disponible.

- grid-template-rows: Esta propiedad establece la cantidad y el ancho de las filas en que se va a “dividir” el elemento padre. Al igual que grid-template-columns, podemos utilizar cualquier unidad de medida para definirlo (%, px,vh,vw) e incluso también la unidad fr.
- grid-template: Esta propiedad “unifica” las propiedades grid-template-column / grid-template-row.
- grid-template-areas: Define la cuadrícula (filas y columnas) haciendo referencia a los nombres de las grid-areas que se especificaron en los elementos hijos.

Si repetimos el nombre de un grid-area causa que el contenido (el elemento que tiene definido ese nombre de grid-area) abarque esas celdas. En caso que queramos dejar una celda vacía, podemos colocar un punto (.).

Ejemplo de la sintaxis:

```
.item-a {
  grid-area: header;
}
.item-b {
  grid-area: main;
}
.item-c {
  grid-area: sidebar;
}
.item-d {
  grid-area: footer;
}

.container {
  display: grid;
  grid-template-columns: 50px 50px 50px 50px;
  grid-template-rows: auto;
  grid-template-areas:
    "header header header header"
    "main main . sidebar"
    "footer footer footer footer";
}
```

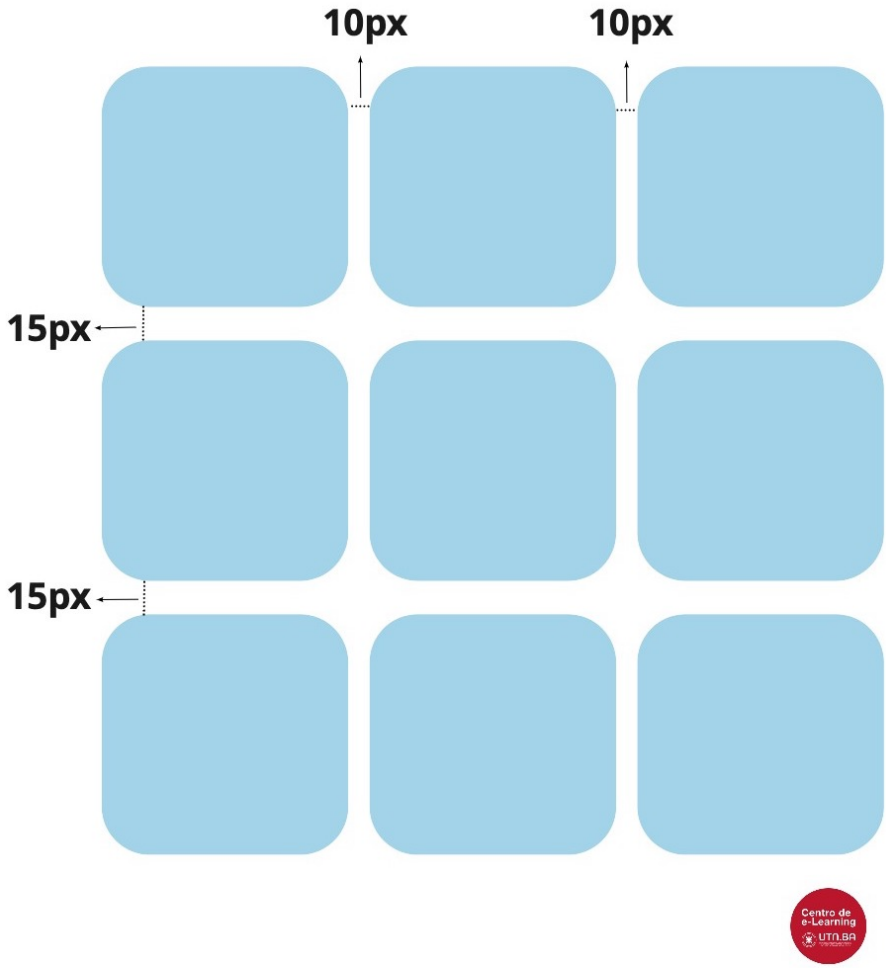
- grid-column-gap / grid-row-gap: Propiedades que definen el tamaño de las líneas de las grillas, es decir que definen el ancho entre columnas y filas (el espacio entre columnas). Debemos tener en cuenta que esto sucede entre las columnas y filas y no por fuera.

Ejemplo

Código

```
.container {
  grid-template-columns: 100px 50px 100px;
  grid-template-rows: 80px auto 80px;
  column-gap: 10px;
  row-gap: 15px;
}
```

Representación gráfica



- grid-gap: Esta propiedad “unifica” las propiedades grid-column-gap y grid-row-gap.
- justify-items: Propiedad que nos permite alinear los elementos hijos en el plano horizontal. Puede tomar los valores :
  - start (alinea a los elementos del lado izquierdo de la celda)
  - end (alinea a los elementos del lado derecho de la celda)
  - center (centra los elementos en la celda)

    - stretch (toma todo el ancho de la celda)
- align-items: Propiedad similar a justify-ítems pero con la diferencia que la alineación de los elementos hijos es en el plano vertical. Puede tomar los valores :
    - start (alinea los elementos en la parte superior de la celda-al inicio de la misma).
    - end (alinea los elementos en la parte inferior de la celda-al final de la misma).
    - center (centra los elementos en la parte central de la celda).
    - stretch (llena todo el alto de la celda- valor por default)
- place-items: Esta propiedad “unifica” las propiedades align-items y justify-items.

4.2. Propiedades para contenedores hijos

Tal cual como indica la frase, las siguientes propiedades de grid son aquellas que se definen en los elementos hijos (lo que hace que en el elemento padre debamos como mínimo colocar la propiedad display:grid).

Ellas son:

- grid-column: Esta propiedad define qué columnas ocupa cada elemento de la grilla. Para ello, utilizamos números enteros donde enumeramos las columnas de izquierda a derecha.
- grid-row: Esta propiedad define qué filas ocupa cada elemento de la grilla. Para ello, utilizamos números enteros donde enumeramos las filas de izquierda a derecha.

- grid-area:Esta propiedad permite asignar un nombre al elemento para luego utilizarlo en la propiedad del componente padre grid-template-area; como también funciona como una propiedad que “unifica” las propiedades grid-row y grid-column.

- justify-self: Propiedad similar al justify-items con la diferencia que solo alinea al elemento en el cual está definido y no al resto de los componentes hijos.

Puede tomar los siguientes valores:

- start (alinea al elemento del lado izquierdo de la celda)
- end (alinea al elemento del lado derecho de la celda)
- center (centra al elemento en la celda)
- stretch (el elemento toma todo el ancho de la celda)
- align-self: Propiedad similar al align-items con la diferencia que solo alinea al elemento en el cual se está definido y no al resto de los componentes hijos.

Puede tomar los siguientes valores:

- start (alinea al elemento en la parte superior de la celda-al inicio de la misma).
- end (alinea al elemento en la parte inferior de la celda-al final de la misma).
- center (centra al elemento en la parte central de la celda).
- stretch (llena todo el alto de la celda- valor por default).
- place-self (“unifica” las propiedades align-self y justify-self).

5. Introducción a Git

Git es un sistema de control de versiones distribuidas, creada por Linus Torvalds y de código abierto (Open Source). Este sistema nos permite a los desarrolladores descargar un software, realizar cambios y subir la versión que hemos modificado.

Por un lado, esto nos va a permitir poder subir nuestro proyecto a un sitio para almacenarlos, priorizando la existencia del proyecto más allá de nuestra computadora. Por otra parte, nos permite ir actualizando los cambios como versiones, pudiendo recuperar los antiguos cambios en caso que deseemos, y también actualizar el proyecto con los cambios realizados.

Algo que tenemos que tener en cuenta es que para manejar Git utilizaremos lo que se llama terminal, que es un programa que ya se encuentra en las computadoras que nos sirven para comunicarnos con ellas.

Desde allí vamos a poder comunicarnos con la computadora e indicarle, mediante comandos, subir el proyecto y sus actualizaciones a medida que vayamos modificándolo.

Los comandos más utilizados son:

- git init: Comando que va a crear una carpeta en nuestra computadora (un repositorio local); que nos servirá para guardar todos los archivos que queramos subir al sitio de almacenamiento.
- git add: Comando que usamos para indicar que archivos vamos a subir al sitio de almacenamiento. Crea una especie de lista donde podemos agregar todos los archivos creados o podemos agregar algún archivo en particular que queramos subir.
- git commit -M “rotulo”: Comando que nos va a permitir “rotular” ese grupo de archivos a subir, nos sirve para hacer una referencia de lo que se está subiendo y permite que a futuro, cuando queramos ver los cambios que se subieron, podamos ubicarlos por medio de ese mensaje.
- git remote add origin http://sitio-de-almacenamiento: Comando que nos permite indicar la ruta donde se tiene que subir ese proyecto o esos cambios. Con esto establecemos una conexión con el sitio de almacenamiento.
- git push :Comando que nos permite subir los archivos indicados al sitio de almacenamiento.

Ahora que ya tenemos una idea de qué es Git y para qué nos sirve, vamos a instalar este software en nuestra computadora para poder empezar a utilizarlo. Pero antes, aprendamos un poco más…

5.1. GitHub: qué es y cómo creamos nuestro primer repositorio

Como dijimos en el anterior punto, Git (entre otras cosas) nos permite subir nuestro proyecto a un sitio para almacenarlo.

Estos sitios son llamados "social coding” y colocan a cada proyecto en un repositorio, que son carpetas donde se encuentran todos los archivos correspondientes a nuestro proyecto.Esto hará que, con la ayuda de Git, podamos ir modificando el proyecto y subiendo los cambios, teniendo así el código de forma remota, lo que nos va a evitar perder el mismo en caso de algún altercado con nuestra computadora.

Otro beneficio que nos traen estos sitios “social coding” es la posibilidad de trabajar en equipo, pudiendo así compartir y hacer colaborativo un proyecto de una manera más ágil.

Existen hoy en día varios de estos sitios; pero en este curso utilizaremos GitHub

Para iniciar el uso de github, te invitamos a crearte un usuario ingresando al siguiente sitio: https://github.com/ (https://github.com/)

5.2. Creando nuestro primer repositorio y subiendo nuestros primeros archivos

¡Ya estamos listos para crear nuestro primer repositorio!

Como primer paso, vamos a instalar Git en nuestra computadora. Para ello, los invitamos a ver el siguiente video que contiene las instrucciones necesarias para instalarlo correctamente.

Ahora que ya tenemos instalado Git, podemos crear nuestro primer repositorio en GitHub. Para ello, los invitamos a seguir los pasos del siguiente video.

Una vez que ya tenemos creado nuestro repositorio, podemos ir a nuestra terminal y empezar con los comandos que nos provee Git para subir el proyecto:

Tengamos en cuenta que la forma de poder aprender estos conceptos es poniéndolos en práctica, por lo que los invitamos a subir los proyectos que tengan armados y proyectos ficticios para que puedan familiarizarse con estas nuevas herramientas.

6. En resumen

En esta unidad tuvimos la oportunidad de repasar y ver en profundidad las propiedades que nos brindan ***grid*** y ***flexbox***, ayudándonos a definir estilos que son muy complejos de resolver con propiedades como ***position*** u otras.

Por otra parte, aprendimos sobre un sistema de control de versionado, como utilizarlo y la forma en que podemos hacer uso de él para guardar nuestros proyectos de forma remota.

De esta forma no solamente empezamos a poder comunicarnos entre desarrolladores y ser parte de un grupo de trabajo; sino que también tenemos la posibilidad de guardar nuestros proyectos y poder acceder a ellos desde cualquier equipo, evitando que perdamos información y que la vida de nuestro proyecto se limite a nuestra computadora.

7. Bibliografía utilizada y sugerida

https://developer.mozilla.org/es/docs/Learn/CSS/CSS_layout/Flexbox (https://developer.mozilla.org/es/docs/Learn/CSS/CSS_layout/Flexbox)

https://www.w3schools.com/css/css3_flexbox.asp (https://www.w3schools.com/css/css3_flexbox.asp)

https://lenguajecss.com/css/maquetacion-y-colocacion/flexbox/ (https://lenguajecss.com/css/maquetacion-y-colocacion/flexbox/)

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

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

https://css-tricks.com/snippets/css/complete-guide-grid/ (https://css-tricks.com/snippets/css/complete-guide-grid/)