DISEÑO DE INTERFACES WEB UD-2

CSS GRID

ÍNDICE

[INTRODUCCIÓN 3](#_Toc148035328)

[PROPIEDADES DE LA REJILLA 3](#_Toc148035329)

[A. display: grid 3](#_Toc148035330)

[B. grid-template-columns 4](#_Toc148035331)

[C. grid-template-rows 5](#_Toc148035332)

[D. grid-template 6](#_Toc148035333)

[E. grid-auto-rows 6](#_Toc148035334)

[F. column-gap y row-gap. 7](#_Toc148035335)

[G. gap 7](#_Toc148035336)

[H. UNIDAD fr 8](#_Toc148035337)

[I. minmax() 10](#_Toc148035338)

[J. repeat() 11](#_Toc148035339)

[I. auto-fit() 12](#_Toc148035340)

[J. grid-auto-flow 13](#_Toc148035341)

[K. grid-auto-columns 13](#_Toc148035342)

[PROPIEDADES DE LA CELDA 14](#_Toc148035343)

[A. grid-column-end 14](#_Toc148035344)

[B. grid-row-end 14](#_Toc148035345)

[C. grid-column y grid-row 15](#_Toc148035346)

[D. order 17](#_Toc148035347)

[PROPIEDADES DE ALINEACIÓN Y JUSTIFICACIÓN 18](#_Toc148035348)

[A. justify-self 18](#_Toc148035349)

[B. aling-self 19](#_Toc148035350)

[C. justify-items 21](#_Toc148035351)

[D. align-items 21](#_Toc148035352)

[E. justify-content 22](#_Toc148035353)

[F. align-content 26](#_Toc148035354)

[ÁREAS 30](#_Toc148035355)

# CSS GRID

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.

Al igual que las tablas, el grid layout permite a un autor alinear elementos en columnas y filas. Sin embargo, con CSS grid son posibles muchos más diseños y de forma más sencilla que con las tablas. Por ejemplo, los elementos secundarios de un contenedor de cuadrícula podrían posicionarse para que se solapen y se superpongan, de forma similar a los elementos posicionados en CSS.

Se parte de un HTML tipo:

Texto

Descripción generada automáticamente

Y de un CSS con el siguiente estilado:

Texto

Descripción generada automáticamente

## PROPIEDADES DE LA REJILLA

1. display: grid

Lo primero que vamos a tener que hacer sobre nuestro contenedor rejilla es aplicar la propiedad display de CSS pero en este caso con el valor grid.

Texto

Descripción generada automáticamente

Pero esta propiedad por si misma no va a hacer nada visual, aunque sí que ha convertido al elemento section en un elemento bidimensional como si fuera una tabla. En este caso una tabla de 1 columna y 8 filas.

Ahora debemos definir la cantidad de columnas y filas que compondrán nuestra rejilla y para ello utilizaremos grid-template-columns y grid-template-rows.

1. grid-template-columns

Esta propiedad nos va a servir para definir la cantidad de columnas que va a conformar nuestra rejilla, así como su tamaño.

Admite diferentes tipos de valores, pero empezaremos con los valores de px. En este caso a la propiedad le añadiremos tantos valores como columnas queramos crear y el valor determinará el tamaño de la columna.

Texto

Descripción generada automáticamente con confianza media

En este caso vamos a generar una rejilla de 5 columnas que tienen cada una un ancho de 120px.

Imagen que contiene Tabla

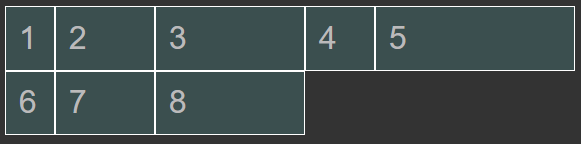
Descripción generada automáticamente

Como tenemos un total de 8 elementos los 5 primeros se colocan en una fila y los 3 restantes en la siguiente, siendo ambas filas del mismo tamaño.

Si colocamos tamaños diferentes el resultado será el siguiente:

Texto

Descripción generada automáticamente con confianza media



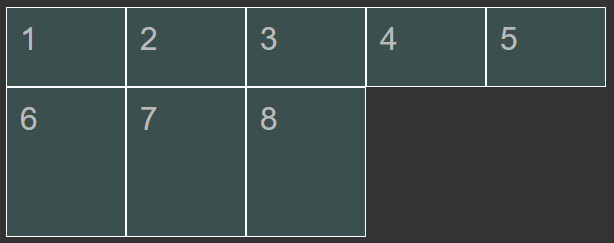
1. grid-template-rows

Esta propiedad nos va a servir para definir el número de filas y su tamaño.

Al igual que grid-templete-columns admite diferentes tipos de valores, pero vamos a empezar por los valores px.

Texto

Descripción generada automáticamente

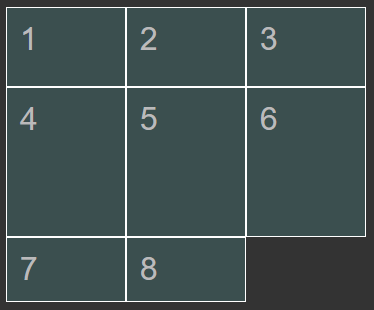


Hemos creado 2 filas siendo la primera una altura de 80px y la segunda de 150px.

En el caso de tener sólo 3 columnas, ocurriría lo siguiente:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente



Se genera una fila para los ítems 1, 2 y 3 que tiene una altura de 80px, otra fila para los ítems 4, 5 y 6 que tiene una altura de 150px y, por último, se genera una tercera fila para los 2 elementos restantes que no tiene una altura por defecto y que crecerá en función del tamaño que necesite la celda.

Por tanto, grid genera filas de forma automática si lo necesita para albergar a todos los elementos contenidos en el elemento padre que actúa de rejilla contenedora.

1. grid-template

Permite unir en una única propiedad CSS las dos propiedades grid-template. La sintaxis es número de filas, una barra inclinada y número de columnas.

Texto

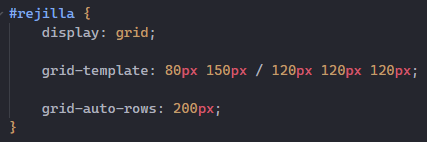
Descripción generada automáticamente

Calendario

Descripción generada automáticamente

1. grid-auto-rows

Esta propiedad permite establecer la altura que tendrán las filas que se generen de forma automática.

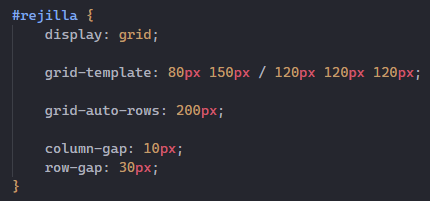




En este caso esa tercera fila que se genera de forma automática tendrá una altura definida a 200px.

1. column-gap y row-gap.

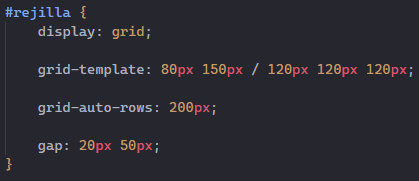
Permiten establecer un espaciado entre columnas (column-gap) y filas (row-gap).

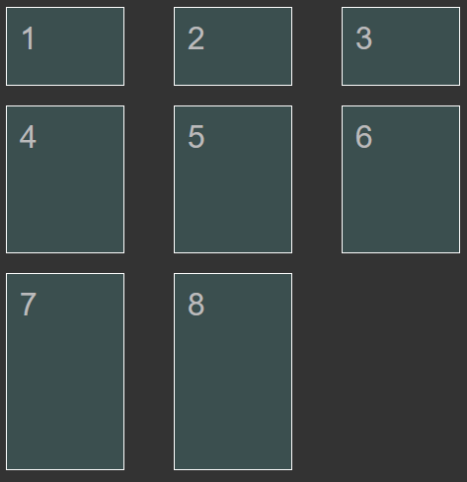




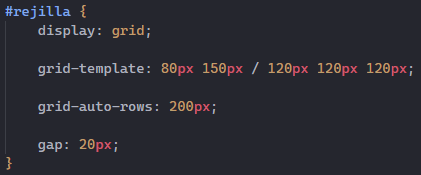
1. gap

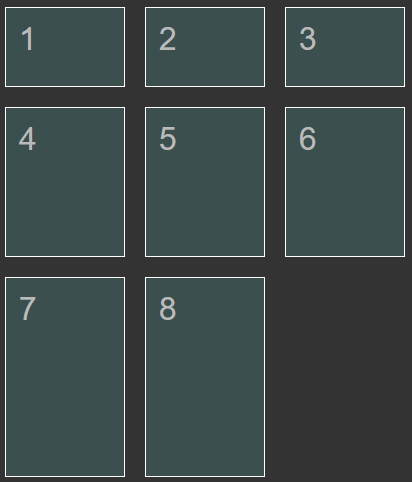
Es una propiedad que une las dos anteriores en una única, el primer valor afecta a la separación de filas y el segundo al de columnas.





En el caso de poner un valor único, este se aplicará tanto a la fila como a la columna.





1. UNIDAD fr

Como se ha dicho podemos utilizar diferentes valores para definir las filas y columnas: con px, con em, rem, %... y también con una unidad especial, el fr.

Esta unidad se utiliza para crear rejillas flexibles y una unidad de fr equivale a 1 fracción del espacio disponible.

Texto

Descripción generada automáticamente

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Lo que está sucediendo en este caso es que ahora nuestra rejilla ocupa todo el ancho disponible y las columnas se reparten este ancho (que es el ancho del contenedor – el ancho que ocupa el gap, en este caso 40px) en una fracción 1/1/2 es decir, la primera y la segunda se quedan con ¼ del ancho y la tercera se queda con la mitad.

Con esto hemos conseguido un diseño flexible ya que si reducimos el ancho disponible el espacio de cada columna se reorganiza.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Esto mismo se puede hacer con las filas, por ejemplo:

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente

De esta forma la primera fila ocupa 2 fracciones del espacio disponible, la segunda ocupa 1 fracción y la última, como es una fila generada automáticamente, utiliza el valor del grid-auto-rows que son 200px.

En el caso de tener un contenedor con una altura fija, por ejemplo 600px, el cálculo sería el siguiente:

* Altura real disponible: 600px – 200px (de la 3ª fila) – 40px (de los 2 gaps) = 360px.
* Altura para la fila 1: 2/3 de 360px = 240px.
* Altura para la fila 2: 1/3 de 360px = 120px.

Texto

Descripción generada automáticamente

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

1. Función minmax()

La función minmax acepta 2 parámetros, el primero el mínimo de px, em, %... que puede tener la columna/fila y el segundo el máximo que puede tener.

Texto

Descripción generada automáticamente

Esto quiere decir que la primera columna tendrá 60px como mínimo y 340px como máximo, mientras que las otras 2 tendrán siempre 300px.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Esta función por tanto nos ayuda a crear un diseño flexible.

1. Función repeat()

La función repeat acepta 2 parámetros, el primero el número de columnas/filas que se verán afectadas por la función y el segundo el número de fr, px, %... que se le asigna a esas columnas/filas.

Esto nos es realmente útil para disminuir la cantidad de código escrito y facilitar la comprensión y lectura de este cuando tenemos varias columnas/filas que son idénticas.

El código del apartado grid-template se podría resumir en:

Texto

Descripción generada automáticamente

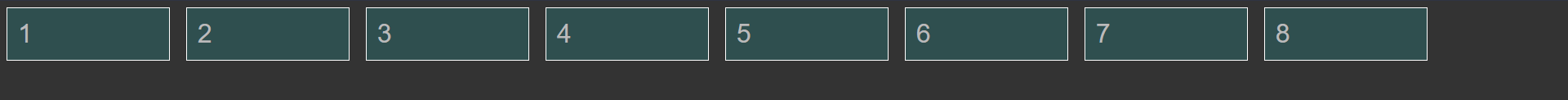
Esto generará 2 columnas de 1fr y una tercera columna de 2fr.

Podemos usar también la opción del auto-fill, esto lo que hará es crear tantas columnas como quepan en el ancho, por ejemplo:

Texto

Descripción generada automáticamente

Esto lo que va a hacer es generar tantas columnas de 200px como quepan en el ancho lo cual aportará cierta responsividad al diseño.



Interfaz de usuario gráfica, Gráfico

Descripción generada automáticamente con confianza media

Obviamente podemos juntar esto con columnas que fueran fijas, por ejemplo:

Texto

Descripción generada automáticamente

Gráfico

Descripción generada automáticamente

En este caso se generan primero las dos columnas de 600px y luego en el ancho disponible restante se generan las n columnas de 200px.

El problema que tiene este repeat con autofill es que hemos perdido la flexibilidad que nos daba el minmax() pero podemos utilizar ambas funciones en una única, por ejemplo:

Texto

Descripción generada automáticamente

Esto lo que hace es rellenar de columnas que tengan como mínimo 200px y como máximo automático para que siempre ocupe el ancho máximo de la pantalla.

Imagen que contiene Icono

Descripción generada automáticamente

1. Función auto-fit()

El problema del auto-fill es que si el si los elementos ocupan una única fila, esta no se rellena por completo teniendo en cuenta todo el ancho del contenedor, por ejemplo:

Texto

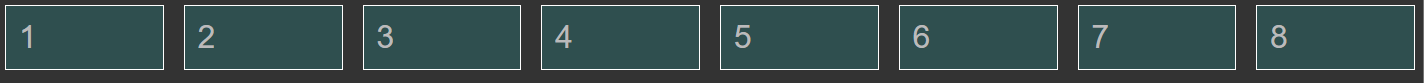
Descripción generada automáticamente



En este caso el auto-fill esta “generando” nuevas celdas pero que no se rellenan porque no hay elementos con los que hacerlos, para evitar este comportamiento, debemos usar la función auto-fit().

Texto

Descripción generada automáticamente



1. grid-auto-flow

El funcionamiento por defecto de grid es rellenar filas pero en determinados casos nos puede venir bien cambiar este flujo y que lo que se rellenen de forma automática sean nuevas columnas.

Para conseguir esto debemos usar la propiedad grid-auto-flow con el valor “column”.

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Gráfico

Descripción generada automáticamente

1. grid-auto-columns

Es el homólogo de grid-auto-rows pero para fijar el ancho de las columnas nuevas generadas al cambiar el grid-auto-flow a “column”.

Texto

Descripción generada automáticamente

1. PROPIEDADES DE LA CELDA
2. grid-column-end

Esta propiedad lo que va a hacer es definir el número de columnas que ocupa nuestra celda.

Una captura de pantalla de un celular con texto e imagen

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

1. grid-row-end

Al igual que podemos definir el número de columnas que ocupa una celda, podemos definir el número de filas que ocupa.

Texto

Descripción generada automáticamente

Gráfico de rectángulos

Descripción generada automáticamente

Si combinamos varias reglas CSS de este estilo podemos terminar generando huecos en nuestra rejilla, por ejemplo:

Texto

Descripción generada automáticamente

Gráfico de rectángulos

Descripción generada automáticamente con confianza media

Esto obviamente puede terminar siendo un problema en nuestro diseño web, así que en estos casos podemos utilizar el valor “dense” en la propiedad grid-auto-flow de la rejilla.

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

1. grid-column y grid-row

Podemos forzar que un elemento ocupe siempre una posición determinada dentro de la rejilla, por ejemplo:

Texto

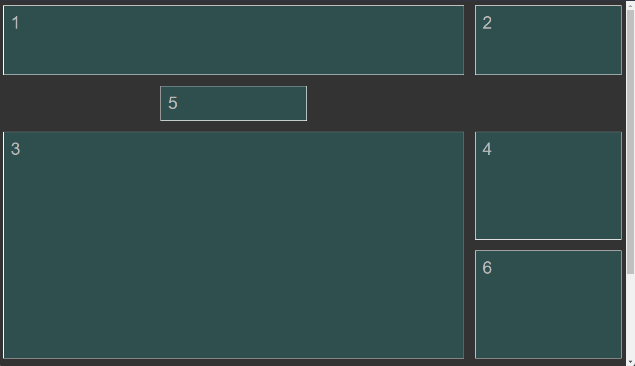
Descripción generada automáticamente

Esto lo que va a hacer es que el elemento 5 ocupe la posición de columna 2 – fila 2.

Gráfico de rectángulos

Descripción generada automáticamente

Y siempre se posicionará ahí, sea cual sea el ancho de la caja.



Estas propiedades realmente pueden llegar a integrar las propiedades de grid-column-end y grid-row-end. Para ello deberemos utilizar el separador de barra inclinada tras el primer número e indicar en qué columna o fila queremos que termine.

Texto

Descripción generada automáticamente

Gráfico de rectángulos

Descripción generada automáticamente

Lo que estamos diciendo:

* + En columna: “Empieza en la 2 y termina justo antes de la 3.”
  + En fila: “Empieza en la 2 y termina justo antes de la 3.”

Si por ejemplo usamos esta regla:

Texto

Descripción generada automáticamente

Sería similar a añadirle un grid-column-end: span 3; y nos producirá este efecto en la pantalla:

Gráfico, Gráfico de rectángulos

Descripción generada automáticamente

Así mismo podemos utilizar valores negativos, por ejemplo:

Texto

Descripción generada automáticamente

Gráfico, Gráfico de rectángulos

Descripción generada automáticamente

Esto es porque en su caso crecer hasta el borde 1 sería crecer hacia atrás y ocupar el primer hueco:

Gráfico, Gráfico de rectángulos

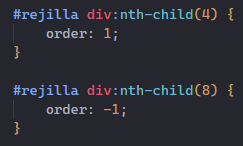
Descripción generada automáticamente

Al decirle que ocupe el borde -1 le estamos diciendo que ocupe desde su espacio hasta el final.

1. order

Al igual que ocurre en flex en CSS Grid tenemos una propiedad order que afecta al orden de los elementos en el flujo.

Por defecto todos los elementos tienen un order de 0.



Forma, Rectángulo

Descripción generada automáticamente

A igual order los elementos respetan su posicionamiento en el DOM.

## PROPIEDADES DE ALINEACIÓN Y JUSTIFICACIÓN

Lo primero es entender que alineación es en sentido vertical y justificación es en sentido horizontal.

1. justify-self

Es una propiedad de la celda y admite varios valores:

* *justify-self: strech*. Es el valor por defecto y hace que la celda ocupe el espacio de la celda.

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

* *justify-self: start*. El elemento se justifica al inicio de la celda, es decir, hacia la izquierda.

Texto

Descripción generada automáticamente

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

* *justify-self: end*. El elemento se justifica al final de la celda, es decir, hacia la derecha.

Texto

Descripción generada automáticamente

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

* *justify-self: center*. El elemento se sitúa en el centro de la celda.

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

1. aling-self

Es la propiedad que se usa para la alineación vertical de los elementos.

* *align-self: strech*. Es el valor por defecto. El elemento se alinea ocupando toda la altura de la celda.

Texto

Descripción generada automáticamente

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

* *align-self: start*. El elemento se alinea ocupando la parte superior de la celda.

Texto

Descripción generada automáticamente

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

* *align-self: end*. El elemento se alinea ocupando la parte inferior de la celda.

Texto

Descripción generada automáticamente

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

* *align-self: center*. El elemento se alinea ocupando la parte central de la celda.

Texto

Descripción generada automáticamente

Imagen que contiene Gráfico

Descripción generada automáticamente

Esta propiedad se puede usar conjunto con la de justificación, por ejemplo, si queremos que el elemento quede centrado en su celda podemos hacer lo siguiente.

Texto

Descripción generada automáticamenteInterfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

1. justify-items

Si queremos que todos los elementos tengan una misma justificación en el elemento rejilla utilizaremos esta propiedad junto con el valor que queramos darle (los mismos que en justify-self).

Texto

Descripción generada automáticamenteImagen que contiene Texto

Descripción generada automáticamente

1. align-items

Igual que el anterior provoca una alineación en todos los elementos de la página. Los valores que se pueden utilizar son los mismos que en align-self.

Texto

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

Al igual de forma individual estas reglas se pueden aplicar de forma conjunta, vamos a centrar todos los elementos en su celda:

Texto

Descripción generada automáticamenteImagen que contiene Cuadrado

Descripción generada automáticamente

1. justify-content

Puede darse el caso de que los elementos no ocupen todo el espacio de la rejilla.

Texto

Descripción generada automáticamente

Imagen que contiene Forma

Descripción generada automáticamente

Esto se puede arreglar gracias a la propiedad justify-content que justifica el contenido general de la rejilla con relación al espacio disponible.

Esta propiedad admite varios valores:

* *justify-content: center.* Justifica el contenido en el centro.

*Texto

Descripción generada automáticamente*

*Icono

Descripción generada automáticamente*

* *justify-content: flex-start.* Es el valor por defecto. Coloca los elementos justificados al inicio del contenedor.

Texto

Descripción generada automáticamente

Imagen que contiene Forma

Descripción generada automáticamente

* *justify-content: flex-end.* Justifica el contenido al final del contenedor.

Texto

Descripción generada automáticamente

Imagen que contiene Forma

Descripción generada automáticamente

* *justify-content: space-around.* Los ítems se alinean uniformemente de tal manera que el espacio entre dos ítems adyacentes es el mismo.

*Texto

Descripción generada automáticamente*

Imagen de la pantalla de un video juego

Descripción generada automáticamente con confianza baja

* *justify-content: space-between.* Los ítems se distribuyen uniformemente sobre la línea. El espaciamiento se hace de tal manera que el espacio adyacente entre dos ítems es el mismo. El borde del comienzo principal y el borde del final principal se alinean al ras con el borde del primer y último ítem respectivamente.

Texto

Descripción generada automáticamente

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

* *justify-content: space-evenly.* El espaciado es igual entre todos los elementos.

*Texto

Descripción generada automáticamente*

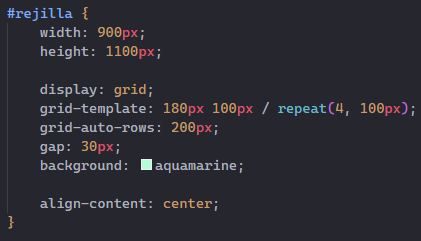
Imagen que contiene Icono

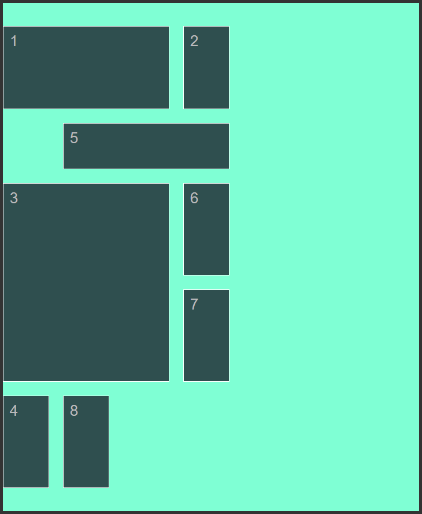
Descripción generada automáticamente

1. align-content

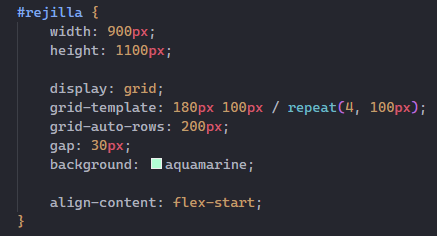
Con esta propiedad alinearemos el contenido.

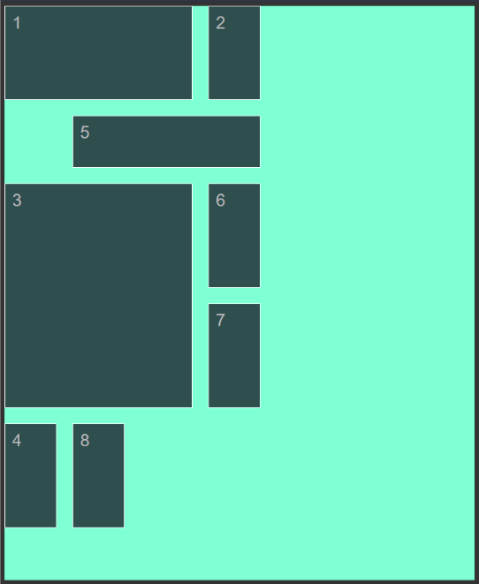
* *align-content: center.* Alinea el contenido en el centro.

**

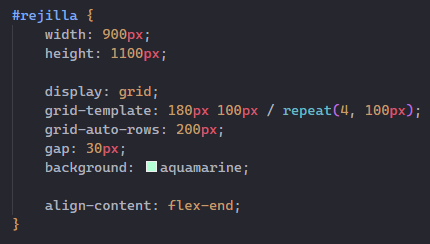
**

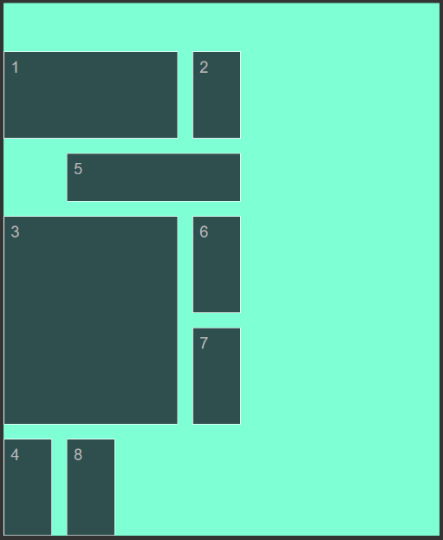
* *justify-content: flex-start.* Es el valor por defecto. Coloca los elementos alineados al inicio del contenedor.



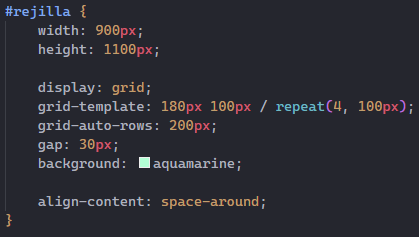


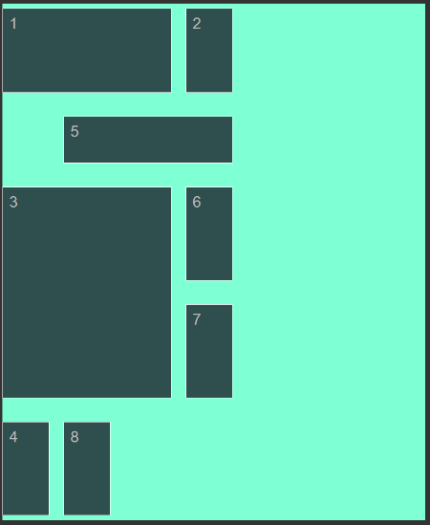
* *justify-content: flex-end.* Alinea el contenido al final del contenedor.



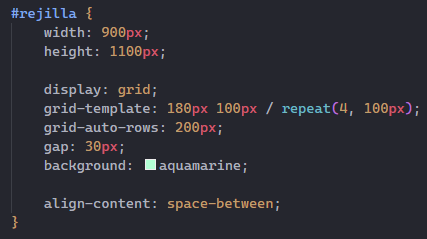


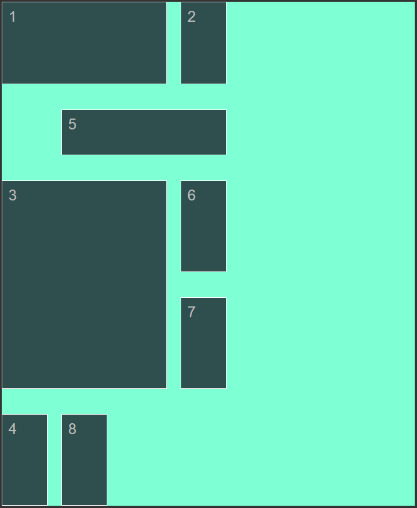
* *justify-content: space-around.* Los ítems se alinean uniformemente de tal manera que el espacio entre dos ítems adyacentes es el mismo.

**

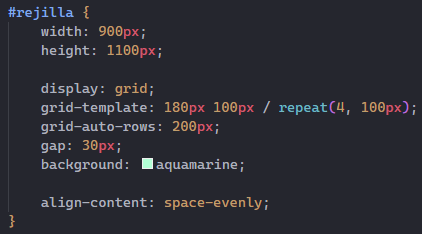


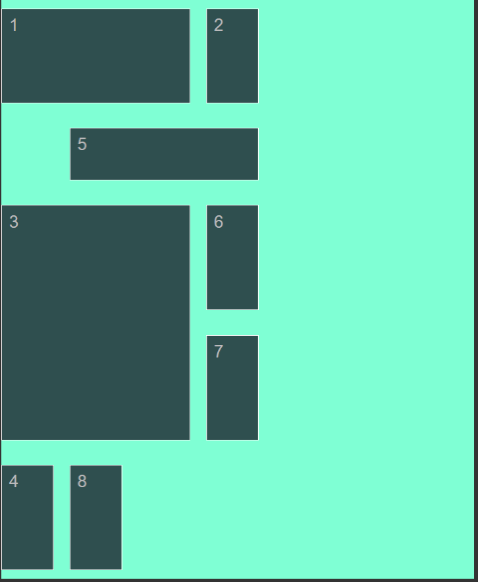
* *justify-content: space-between.* Los ítems se distribuyen uniformemente sobre la línea. El espaciamiento se hace de tal manera que el espacio adyacente entre dos ítems es el mismo. El borde del comienzo principal y el borde del final principal se alinean al ras con el borde del primer y último ítem respectivamente.





* *justify-content: space-evenly.* El espaciado es igual entre todos los elementos.

**



Al igual que pasa con las otras propiedades estas se pueden combinar.

Texto

Descripción generada automáticamente

Imagen de la pantalla de un video juego

Descripción generada automáticamente con confianza baja

## ÁREAS

Con CSS Grid podemos poner nombres a ciertos elementos de nuestro diseño. Esto nos permitirá trabajar con diferentes anchos de pantalla (media query).

Texto

Descripción generada automáticamente

Por ejemplo, en el siguiente ejemplo, vamos a poner nombre a las líneas que hay entre los elementos del grid:

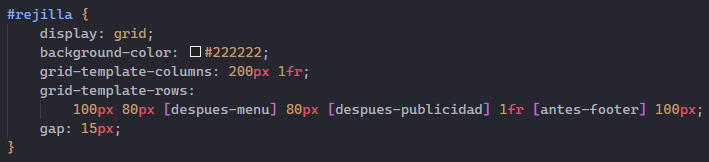


Esto nos crea este tipo de página:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Lo que nos permite es etiquetar zonas de la siguiente forma:



A la izquierda del primer valor pondríamos la etiqueta a la primera línea (el borde superior de la pantalla), entre el primer valor y el segundo, podríamos poner la etiqueta para la segunda línea (debajo de la primera fila del grid), y así sucesivamente podríamos etiquetar todas las líneas horizontales (en rows) o verticales (en columns). En nuestro caso hemos puesto nombre a las líneas 3ª, 4ª y 5ª.

Y ahora podemos usar esas etiquetas en las reglas de los elementos:

Texto

Descripción generada automáticamente

También podemos nombrar las áreas de nuestro diseño, con la propiedad: grid-area, definida en cada uno de nuestros elementos:

Texto

Descripción generada automáticamente

Una vez definidas las áreas, dentro de la rejilla dibujamos nuestro layout con la propiedad grid-template-areas, indicando qué coordenadas ocupan a nivel de fila y columna las áreas definidas, por ejemplo, el área cabecera (header) ocupa las dos columnas, por tanto indicamos las dos columnas de la primera fila son “header”. En la segunda fila el area menu también ocupa las dos columnas. “menu   menu”

En la tercera y cuarta fila, aside ocupa la primera columna… y así vamos completando todas las filas y columnas del grid indicando qué área iría en cada coordenada.

Texto

Descripción generada automáticamente

Con el uso de áreas ya no necesitamos indicar la propiedad grid-column / grid-row a nivel de elementos, así que podemos comentarlas.

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Una vez hecho esto, podemos aprovechar la media query para definir una distribución de áreas diferente cuando se alcance un tamaño determinado:

Texto

Descripción generada automáticamente

Dejamos sólo una columna y una fila, y que todos los elementos se apilen uno encima de otro cuando el ancho sea menor de 700px.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente