Hola! Continuamos profundizando en CSS. En esta ocasión exploramos las funciones en CSS.

Las funciones son unas de las caracteristicas que han sido incorporadas en CSS, que ademas ya estan soportadas por los navegadores.

Las funciones sirven para obtener valores que son usados por las propiedades. Hay un conjunto de funciones bien definidas en CSS, de las cuales discutimos algunas a continuacion:

rgb(): define un color en RGB. Un ejemplo se muestra seguidamente:

<!DOCTYPE html>

<html>

<head>

<style>

#p1 {

background-color:rgb(255,59,59);

}

</style>

</head>

<body>

<p>Ejemplo de color en RGB</p>

<p id="p1">Este color</p>

</body>

</html>

rgba(): define un color en RGB con transparencia.

hsl(): define un color en HSL (Hue-saturation-lightness).

hsla(): define un color en HSL con transparencia.

radial-gradient(): establece un gradiente radial en el fondo de la imagen, como se muestra en el siguiente ejemplo.

<!DOCTYPE html>

<html>

<head>

<style>

#gradiente1 {

height: 150px;

width: 200px;

background-image: radial-gradient(pink, gray, blue);

}

</style>

</head>

<body>

<h3>Ejemplo de Gradiente Radial</h3>

<div id="gradiente1"></div>

</body>

</html>

linear-gradient(): establece un gradiente lineal en el fondo de la imagen.

calc(): permite realizar operaciones matemáticas cuyos resulatdos se usan en la propiedades de CSS.

<!DOCTYPE html>

<html>

<head>

<style>

#c1 {

position: absolute;

left: 100px;

width: calc(100% - 200px);

border: 1px solid black;

background-color: gray;

}

</style>

</head>

<body>

<h1>Ejemplo de la función calc()</h1>

<div id="c1">Usando calc()</div>

</body>

</html>

attr(): devuelve el valor de un atributo de elemento seleccionado, como se muestra a continuación:

<!DOCTYPE html>

<html>

<head>

<style>

a:after {content: " (" attr(href) ")";}

</style>

</head>

<body>

<h1>Ejemplo de la función attr()</h1>

<p>Insert the value of the href attribute in parenthesis after each link:</p>

<p><a href="https://www.nextu.com">Visitanos en: </a></p>

</body>

</html>

var(): permite el uso de variables con valores especificos que puden ser usadas varias veces en el documento HTML. La variable se define anteponiendo dos guiones al nombre y se accde usando la funcion var(). Es ampliamante usada en la definicion de aspectos tales como colores, margenes, tipos de letras que empleamos en varios lugraes de nuestro sitio Web. A continuacion tenemos un ejemplo:

<!DOCTYPE html>

<html>

<head>

<style>

:root {

--color-principal-de-fondo: #FF3B3B;

--color-del-texto: yellow;

}

#div1 {

background-color: var( --color-principal-de-fondo);

padding: 5px;

color: var(--color-del-texto);

}

</style>

</head>

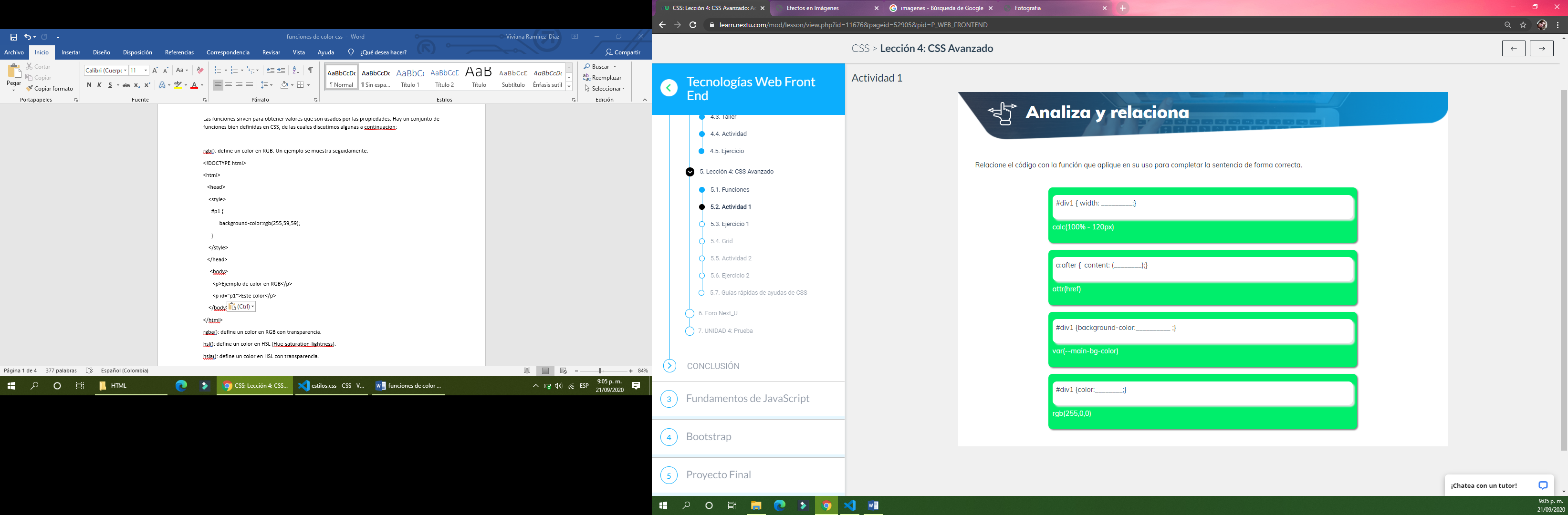
<body>

<h1>Uso de la función var()</h1>

<div id="div1">Ejemplo</div>

</body>

</html>



**Grid**

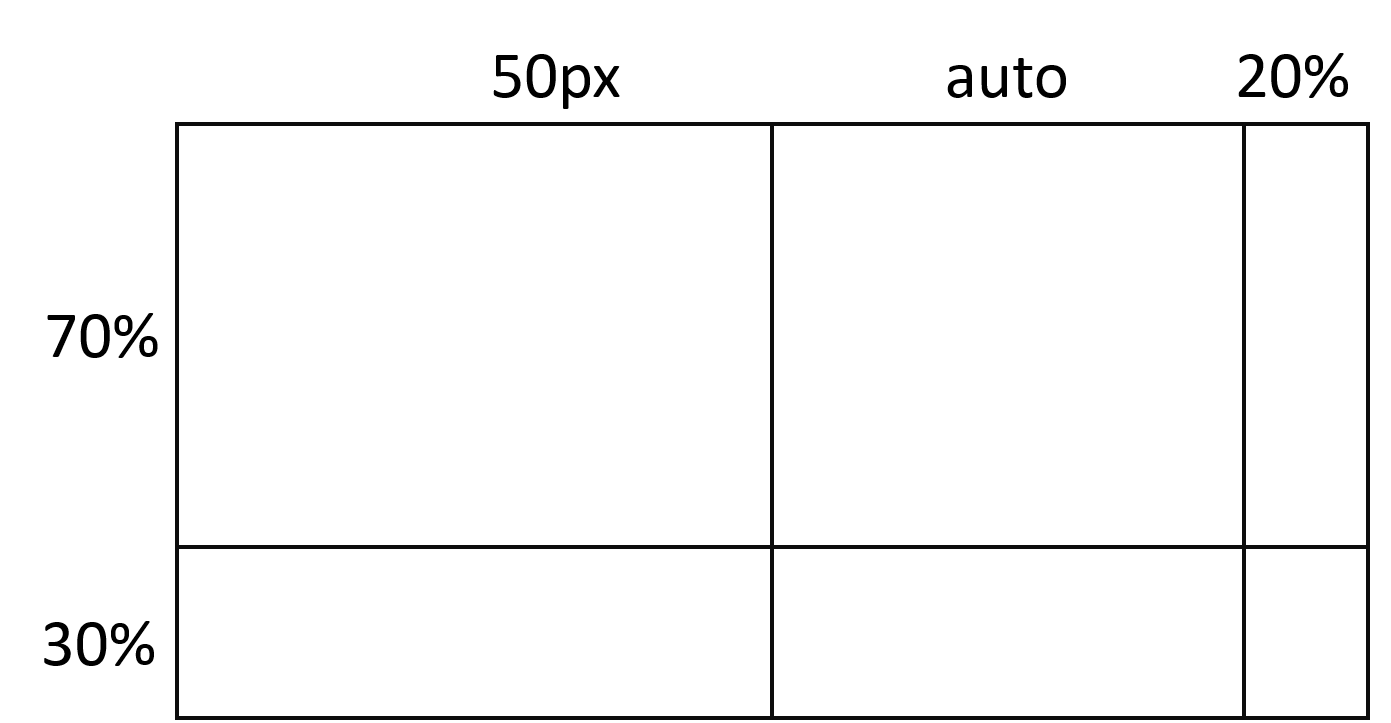
¡Hola! Te recibimos con mucho entusiasmo para seguir ampliando conocimientos en relación a las características avanzadas de CSS. En esta ocasión vamos a ver algunos aspectos relacionados con grids.

**¿Que es una grilla?**

Para facilitar el diseño de las páginas, CSS usa el concepto de **rejillas** para acomodar los diferentes elementos en un documento.

**Definiendo una grilla en CSS**

Supongamos que queremos definir una **grilla** con la dimensiones que se muestra en la **Figura 1**. La rejilla tiene una columna que mide **50px**, otra que mide el **20%** del espacio restante y una que se acomoda **automáticamente** (**auto**) en el espacio que sobra.



*Figura 1: Ejemplo de una rejilla.*

En CSS esta rejilla se construye con el siguiente código.

<style>

.rejilla {

display: grid;

grid-template-columns: 50px auto 20%;

grid-template-rows: 70% 30%;

}

</style>

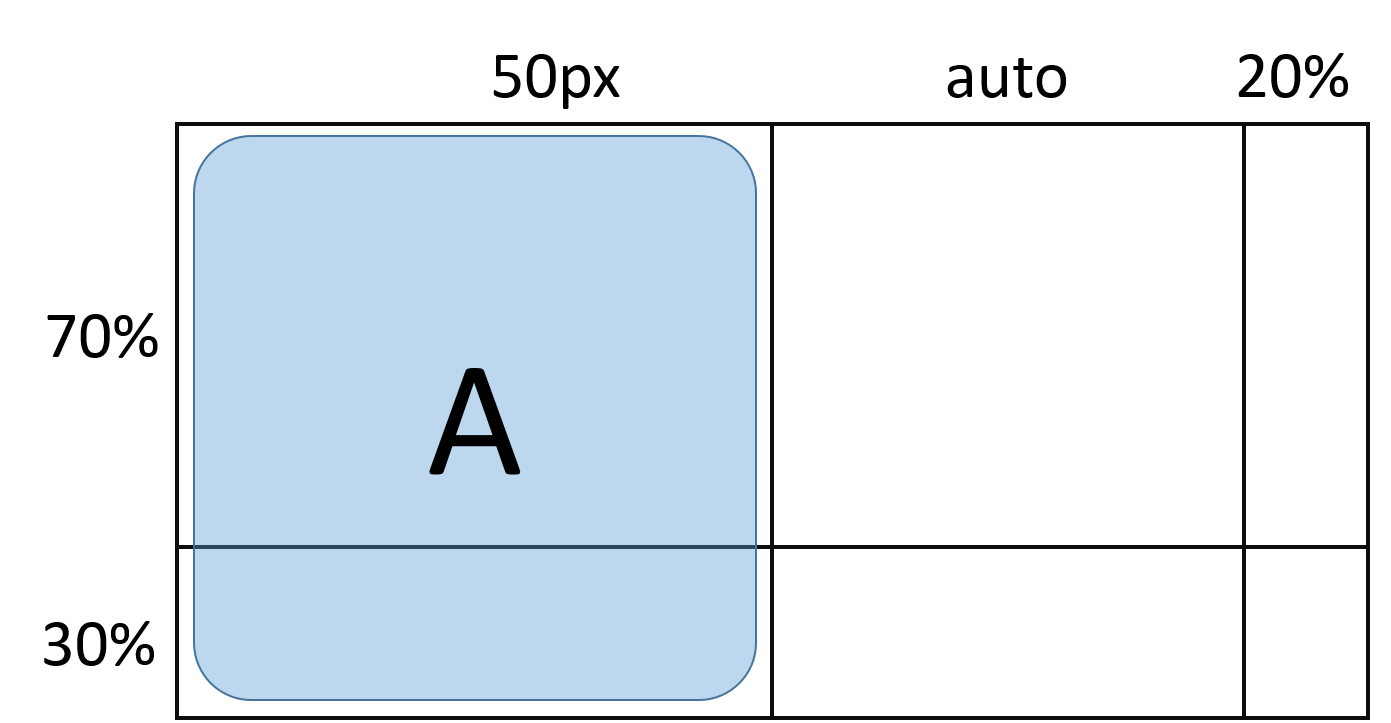
<div class="rejilla">

</div>

**Determinando elementos en la rejilla**

A continuación iremos examinando cómo se especifican elementos usando la rejilla que definimos previamente.

La**Figura 2** muestra cómo deseamos que se vea el **elemento A** en la rejilla.



*Figura 2: Posicionamiento del elemento A en la rejilla.*

Seguidamente tenemos el código para lograr dibujar el **elemento A** usando la rejilla.

<style>

.rejilla {

display: grid;

grid-template-columns: 50px auto 20%;

grid-template-rows: 70% 30%;

}

.elemento-a {

grid-row-start: 1;

grid-row-end: 3;

grid-column-start: 1;

grid-column-end: 2;

}

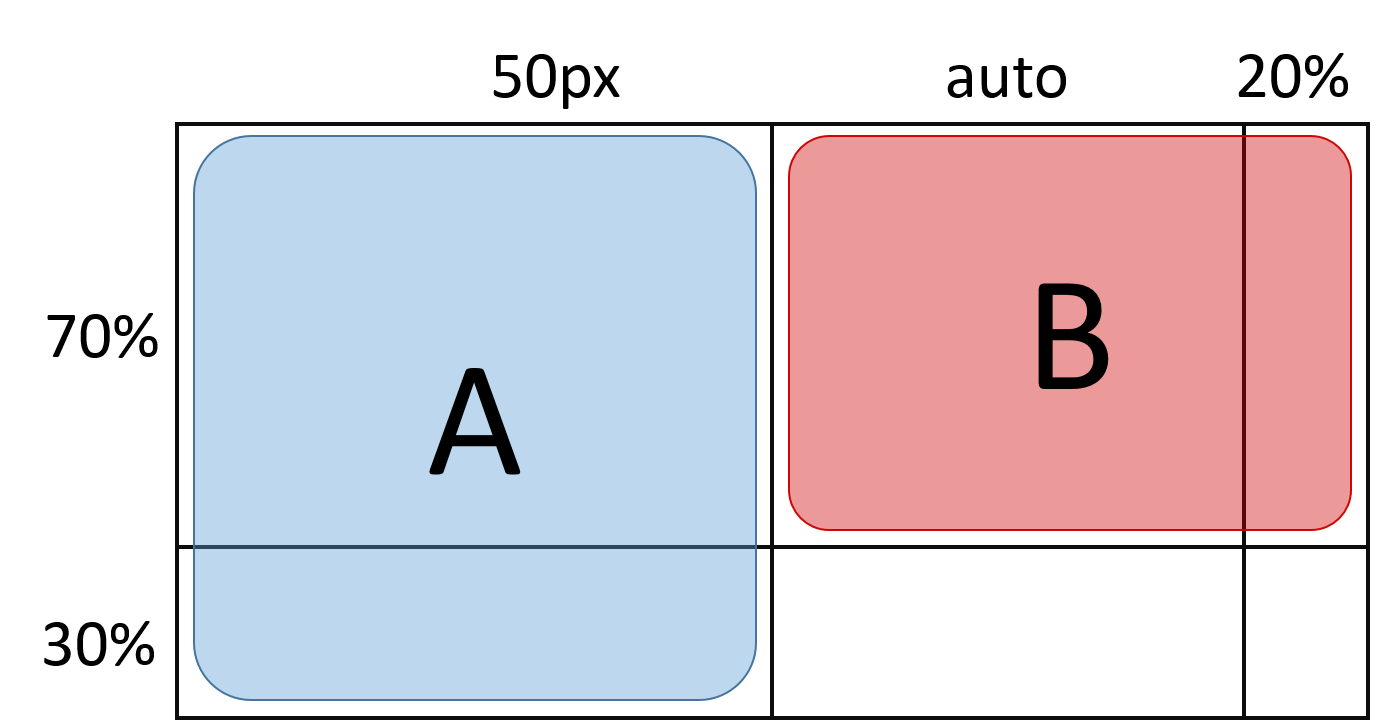
</style>

<div class="rejilla">

<div class="elemento-a">A</div>

</div>

En la **Figura 3** hemos incorporado otro elemento (**elemento B**) con la ayuda de la rejilla.



*Figura 3: Posicionamiento del elemento B en la rejilla.*

El siguiente código se usa para definir el **elemento B**.

<style>

.rejilla {

display: grid;

grid-template-columns: 50px auto 20%;

grid-template-rows: 70% 30%;

}

.elemento-a {

grid-row-start: 1;

grid-row-end: 3;

grid-column-start: 1;

grid-column-end: 2;

}

.elemento-b {

grid-row-start: 1;

grid-row-end: 2;

grid-column-start: 2;

grid-column-end: 4;

}

</style>

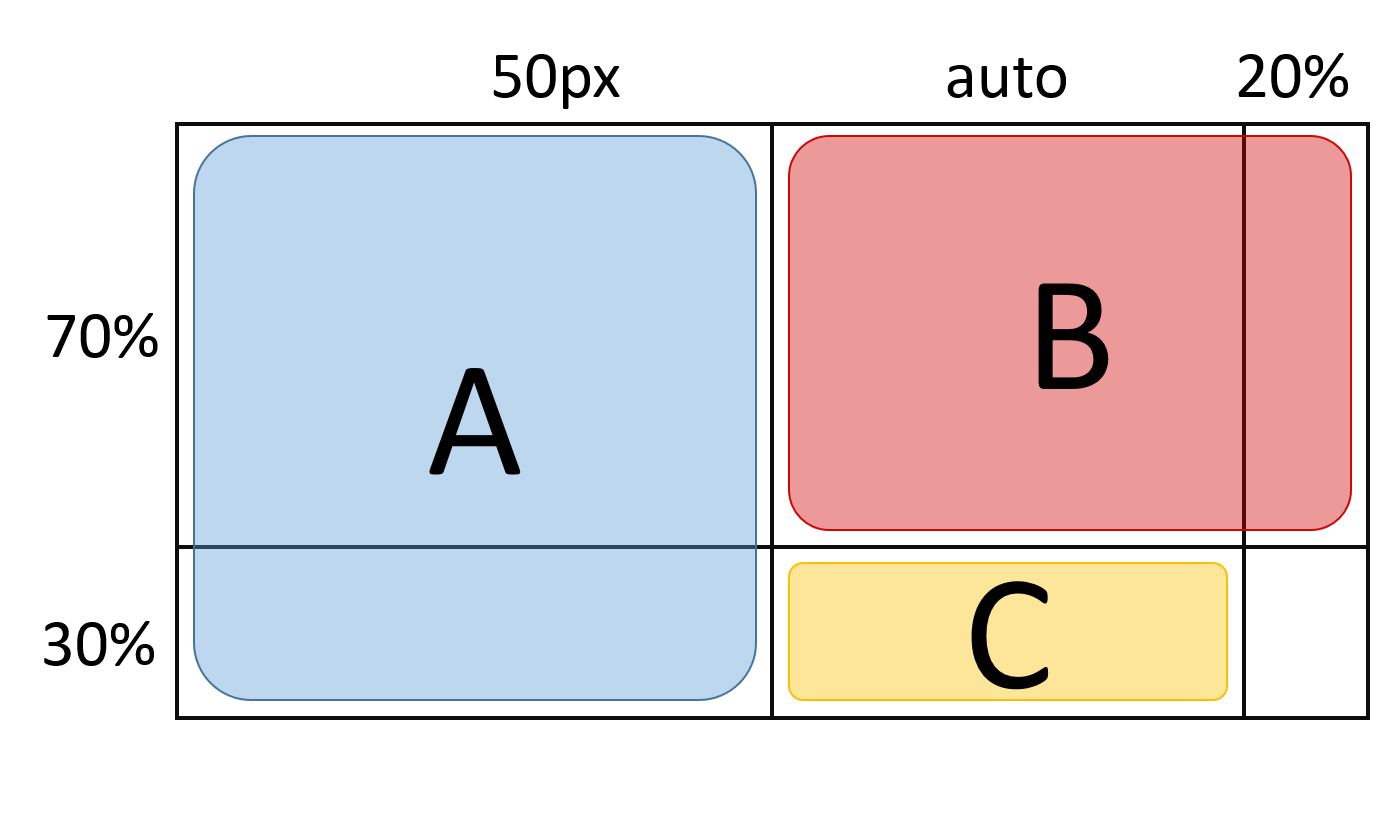
<div class="rejilla">

<div class="elemento-a">A</div>

<div class="elemento-b">B</div>

</div>

En la **Figura 4**, hemos sumado otro elemento (**elemento C**) a nuestro arreglo de ítems



*Figura 4: Posicionamiento del elemento C en la rejilla.*

El siguiente código se usa para definir el **elemento C**.

<style>

.rejilla {

display: grid;

grid-template-columns: 50px auto 20%;

grid-template-rows: 70% 30%;

}

.elemento-a {

grid-row-start: 1;

grid-row-end: 3;

grid-column-start: 1;

grid-column-end: 2;

}

.elemento-b {

grid-row-start: 1;

grid-row-end: 2;

grid-column-start: 2;

grid-column-end: 4;

}

.elemento-c {

grid-row-start: 2;

grid-row-end: 3;

grid-column-start: 2;

grid-column-end: 3;

}

</style>

<div class="rejilla">

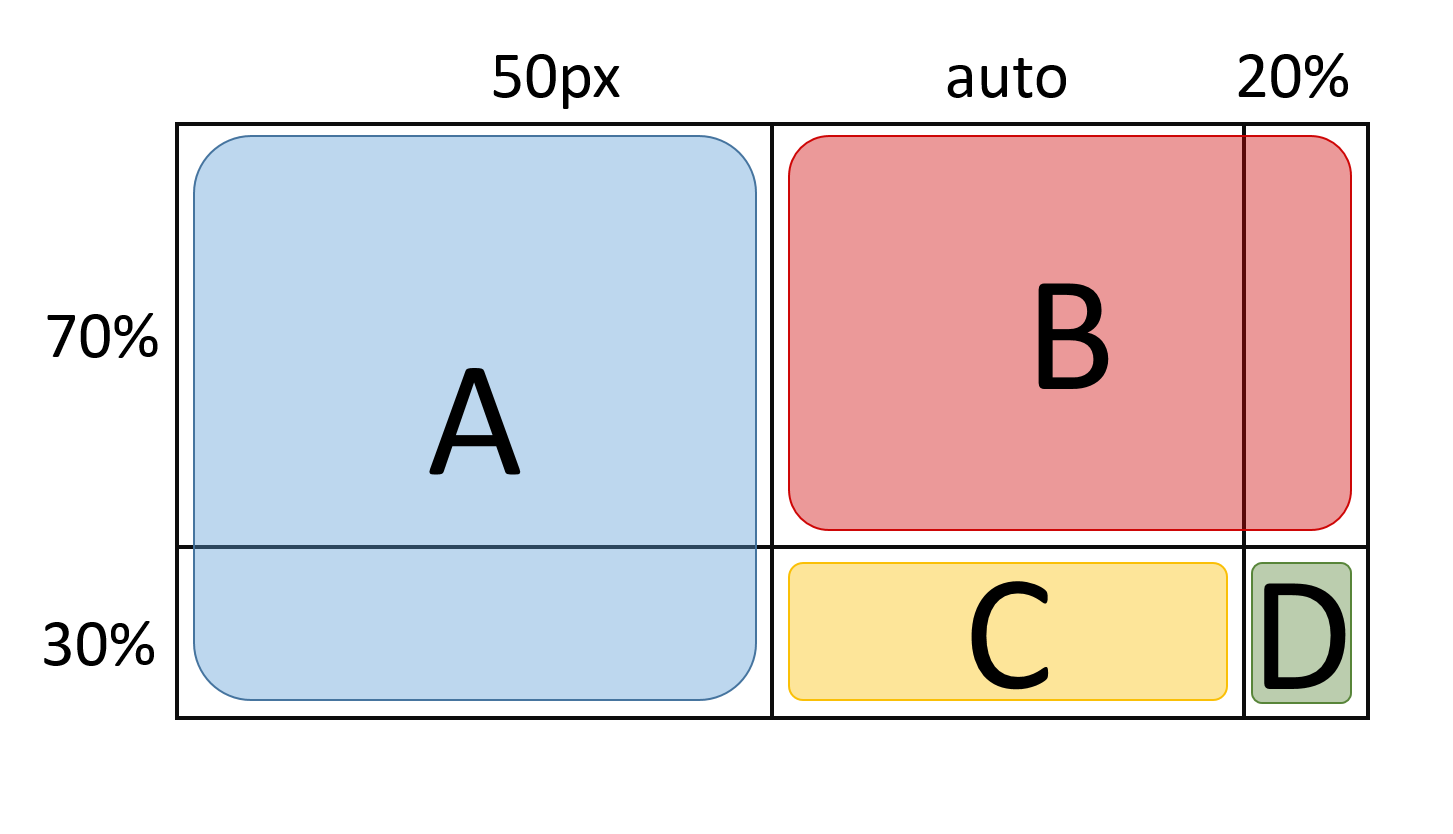
<div class="elemento-a">A</div>

<div class="elemento-b">B</div>

<div class="elemento-c">C</div>

</div>

La **Figura 5** muestra cómo deseamos que se vea el **elemento D** en la rejilla.



*Figura 5: Posicionamiento del elemento D en la rejilla.*

Seguidamente tenemos el código para lograr dibujar el **elemento D** usando la rejilla.

<style>

.rejilla {

display: grid;

grid-template-columns: 50px auto 20%;

grid-template-rows: 70% 30%;

}

.elemento-a {

grid-row-start: 1;

grid-row-end: 3;

grid-column-start: 1;

grid-column-end: 2;

}

.elemento-b {

grid-row-start: 1;

grid-row-end: 2;

grid-column-start: 2;

grid-column-end: 4;

}

.elemento-c {

grid-row-start: 2;

grid-row-end: 3;

grid-column-start: 2;

grid-column-end: 3;

}

.elemento-d {

grid-row-start: 2;

grid-row-end: 3;

grid-column-start: 3;

grid-column-end: 4;

}

</style>

<div class="rejilla">

<div class="elemento-a">A</div>

<div class="elemento-b">B</div>

<div class="elemento-c">C</div>

<div class="elemento-d">D</div>

</div>

**Guías Rápidas de Ayudas de CSS**

Estamos contentos de poder disfrutar esta lectura relacionada a las **guías rápidas de ayuda de CSS** contigo.

En Internet podrás conseguir varios sitios que resumen las propiedades de CSS, que te pueden ayudar cuando estés realizando tus tareas de codificación. Nosotros te proporcionamos algunos de ellos.

**Comprehensive CSS Cheat Sheet + PDF**

El sitio de [Comprehensive CSS Cheat Sheet](https://www.onblastblog.com/css3-cheat-sheet/" \t "_blank) ofrece una guía bastante completa de las propiedades de CSS agrupadas y categorizadas.

Ventajas

* Proporciona una versión descargable en formato pdf.
* Ofrece las propiedades de acuerdo a ciertas categorías.

Desventaja

* No hay ejemplos de los resultados de aplicar las propiedades.

**HTML CheatSheet**

En el sitio [HTML CheatSheet](https://htmlcheatsheet.com/css/) encontrarás una guía interactiva de las **propiedades de CSS**.

Ventaja

* Puedes interactuar con la propiedades de CSS, cambiando valores y viendo los resultados que obtienes al cambiar dichos valores.

Desventaja

* El sitio no contiene todas las propiedades de CSS.

**CSS Cheat Sheet por Adam Marsden**

El sitio de [CSS Cheat Sheet por Adam Marsden](https://adam-marsden.co.uk/css-cheat-sheet) ofrece otra guía rápida de las propiedades de CSS, que además se enlaza al tutorial completo del lenguaje de la gente de Mozilla.

Ventajas

* Tiene un menú que permite acceder a los diferentes componentes de CSS fácilmente.
* Proporciona acceso a información más detallada de cada componente de CSS, incluyendo ejemplos.

Desventaja

* Los ejemplos no se despliegan directamente en la guía rápida, sino en el Tutorial de CSS en el sitio de Mozilla.