# UNIDAD DE TRABAJO 4: IMPLANTACIÓN DE CONTENIDO MULTIMEDIA.

Tabla de contenido

[UNIDAD DE TRABAJO 4: IMPLANTACIÓN DE CONTENIDO MULTIMEDIA. 1](#_Toc30704270)

[1. Licencias 2](#_Toc30704271)

[1.1 Con derechos de autor: 2](#_Toc30704272)

[1.2 Creative Commons 2](#_Toc30704273)

[1.3 De dominio público 3](#_Toc30704274)

[2. PROGRAMAS PARA LA EDICIÓN DE IMÁGENES. 3](#_Toc30704275)

[2.1 Implementación de imágenes con HTML5 5](#_Toc30704276)

[3. Programas para convertir audio 6](#_Toc30704277)

[3.1 Ejemplo de implementación de audio 8](#_Toc30704278)

[4. Programas de conversión de video 8](#_Toc30704279)

[4.1 Ejemplo de implementación 12](#_Toc30704280)

[5. CANVAS 12](#_Toc30704281)

[6. Propiedad Storage (localStorage y sessionStorage). 13](#_Toc30704282)

[6.1 EL OBJETO LOCALSTORAGE 14](#_Toc30704283)

[6.2 El objeto sessionstorage 15](#_Toc30704284)

[6.3 Implementación 15](#_Toc30704285)

[6.4 Forma Simplificada 15](#_Toc30704286)

[Ejemplo de localStorage: 15](#_Toc30704287)

[EJEMPLO DE SESSIONSTORAGE 16](#_Toc30704288)

[7. GRID 17](#_Toc30704289)

[7.1 Propiedad inline-grid 18](#_Toc30704290)

[7.2 Hueco entre elementos: Row-gap y column-gap 19](#_Toc30704291)

[7.3 Column lines y Row lines 20](#_Toc30704292)

[7.4 Grid-template-columns 22](#_Toc30704293)

[7.5 Grid-template-rows 23](#_Toc30704294)

[7.6 Justify-content 24](#_Toc30704295)

[7.7 Align-content 25](#_Toc30704296)

[7.8 Grid-area 25](#_Toc30704297)

[8. Imágenes responsive 27](#_Toc30704298)

# **Licencias**

Con respecto a las imágenes existen 3 tipos distintos de licencias:

* Con derechos de autor (@copyright).
* Creative Commons.
* De dominio público.

## **Con derechos de autor:**

Protege el contenido en su totalidad. El contenido pertenece al autor o al usuario al que se le ha cedido el uso de su explotación.

Los derechos de autor surgen al crear la obra y no necesitan de un registro previo.

Una variante del copyrigth es el **Copyleft**, que permite la libre alteración y distribución de las copias de las imágenes, siempre que se garantice el libre acceso a las mismas.

## **Creative Commons**

Es un tipo de licencia creada especialmente para internet. Es compatible con los derechos de autor, pero ofrece derechos específicos a otras personas. En cualquier obra hay que reconocer la autoría.

Tipos de licencias:

* Reconocimiento: Se permite cualquier explotación de la obra, así como la creación de obras derivadas y su distribución. Hay que reconocer la autoría.

Imagen que contiene dibujo

Descripción generada automáticamente

* Reconocimiento – No comercial: Se permite cualquier explotación de la obra siempre que no tenga fines comerciales.
* Imagen que contiene dibujo, señal

  Descripción generada automáticamente
* Reconocimiento – No comercial – Compartir igual: No se permite el uso comercial ni de la obra original ni de las derivadas. Además, la distribución se debe hacer manteniendo el tipo de licencia original.

Imagen que contiene dibujo, señal, plato

Descripción generada automáticamente

* Reconocimiento – No comercial – Sin obra derivada: No está permitida la comercialización de la obra original ni la creación de obras derivadas, independientemente de su finalidad.

Imagen que contiene dibujo, señal, plato, calle

Descripción generada automáticamente

* Reconocimiento – Compartir igual: Se permite el uso comercial de la obra, así como de las obras derivadas, siempre que se mantenga la licencia que regula la obra original.

Imagen que contiene dibujo, señal, plato

Descripción generada automáticamente

* Reconocimiento – Sin obra derivada: Se permite el uso comercial de la obra, pero no la creación de obras derivadas.

Imagen que contiene dibujo, señal, medidor, plato

Descripción generada automáticamente

Para poder obtener una licencia hemos de visitar la web oficial:

<https://creativecommons.org/choose/?lang=es_CO>

Aquí tendremos que rellenar un formulario con una serie de campos para poder elaborar la licencia según las especificaciones que queramos.

Captura de pantalla de un celular

Descripción generada automáticamente

## **De dominio público**

Creaciones cuyo periodo de protección de derechos de autor ha expirado. Pueden ser utilizadas sin reconocer la autoría, sin pagar a nadie y para el fin que se estime oportuno. La obra de Cervantes.

## **PROGRAMAS PARA LA EDICIÓN DE IMÁGENES.**

Todos conocen las soluciones del paquete de Adobe para la edición de imágenes, como Photoshop o illustrator.

Pero si bien son herramientas muy potentes, su uso esta más enfocado a la edición profesional.

Las alternativas gratuitas que hemos utilizado en clase para este propósito son las siguientes:

* Paint.net: este es un editor sencillo y ligero para Windows que nos va a permitir de una manera simple trabajar con capas e imágenes. Para instalarlo es necesario descargarlo de su página oficial en el siguiente [enlace](https://www.getpaint.net/index.html) e instalarlo en nuestro equipo. No es multiplataforma y solo se encuentra para Windows.

Una captura de pantalla de una red social

Descripción generada automáticamente

* La segunda herramienta utilizada en clase es GIMP. Esta es una alternativa mucho más potente que Paint.net y nos va a permitir hacer la mayoría de trabajos que necesitemos con imágenes. Para instalarlo solo es necesario ir a su web en el siguiente [enlace](https://www.gimp.org/) y descargarse la última versión e instalarlo. Tiene la ventaja de ser multiplataforma.

Captura de pantalla de un celular con texto e imagen

Descripción generada automáticamente

## **Implementación de imágenes con HTML5**

Vamos a tratar sobre la etiqueta <img>. La cual se utiliza para insertar una imagen en una página web. Es necesario poner los atributos alt (texto alternativo) y title, para así tener una mayor accesibilidad.

Un ejemplo de utilización:

<h2>Etiqueta img</h2>

<img src="img/imagen1.jpg" alt="Imagen the witcher" title="Geralt de Rivia">

[Salida en el navegador](imagenes/subir/fichero.html)

Ahora vamos a tratar sobre dos etiquetas que sirven para etiquetar una imagen. Estas etiquetas son:

**<figure>** la cual contendrá todo lo relacionado con la imagen.

**<figcaption>** Esta etiqueta sería el pie de página.

Ejemplo de uso:

<figure>

<img src="img/imagen2.jpg" alt="imagen the witcher" title="imagen the witcher">

<figcaption>Este es el pie de foto de la imagen</figcaption>

</figure>

[Salida en el navegador](figureyfigcaption/subir/fichero.html)

**Etiqueta picture:** Esta etiqueta nos va a permitir ofrecer distintas imágenes dependiendo del dispositivo. Se usa de una forma muy similar a las media queries, ya que vamos a especificar las medidas en las cuales va a ser visible la imagen. Es importante que se especifique la etiqueta img por si el navegador no soporta la etiqueta picture.

Vamos con un ejemplo de uso:

<picture>

<source media="(min-width: 650px)" srcset="imagen1.jpg">

<source media="(min-width: 465px)" srcset="imagen2.jpg">

<img src="imagenRetrocompatible.jpg" alt="Imagen retrocompatible" style="width:auto;">

</picture>

[Salida en el navegador.](picture/fichero.html)

Para comprobar el ejemplo es necesario disminuir la pantalla del navegador.

## **Programas para convertir audio**

Al igual que hay que tratar las imágenes puede ser necesario que tratemos los elementos de audio que incorporemos en nuestra página web. Sobretodo será necesario incluir distintos formatos para hacer que sea completamente compatible.

Para hacer estas conversiones de audio hemos usado el programa Audacity. Este software es uno de los más conocidos para audio y nos va a permitir transformar el formato del audio. Para ello lo primero va a ser descargarlo e instalarlo desde su web:

Captura de pantalla de un celular

Descripción generada automáticamente

Una vez descargado hay que ejecutar el instalador:

Captura de pantalla de un celular

Descripción generada automáticamente

El proceso de instalación es todo siguiente, hasta que se termine el instalador y nos permita ejecutar la aplicación.

Para poder usarlo es necesario tener un audio descargado y abrirlo en la aplicación:

Imagen que contiene captura de pantalla

Descripción generada automáticamente

Una vez lo tenemos ahí para poder transformalo hay que ir de nuevo a archivo y elegir la opción de exportar, aquí nos salen las opciones de formatos en que podemos exportar.

A nosotros nos va a interesar los formatos ogg y wav.

Captura de pantalla de un celular

Descripción generada automáticamente

Exportamos a los dos formatos que queremos y ya tendríamos el audio preparado para que sea compatible.

## **Ejemplo de implementación de audio**

Ahora que ya tenemos en todos los formatos los audios, vamos a ver un ejemplo de implementación.

<audio id=”audio1” controls>

<source src=”audio/audio1.mp3” type=”audio/mpeg”>

<source src=”audio/audio2.ogg” type=”audio/ogg”>

<source src=”audio/audio3.wav” type=”audio/wav”>

Tu navegador no soporta esta etiqueta HTML5

</audio>

Ahora utilizamos unas funciones de Javascript para implementar unos botones de control del audio:

<div> <button onclick="document.getElementById('audio1').play()">Reproducir el Audio</button> <button

onclick="document.getElementById('audio1').pause()">Pausar el Audio</button> <button

onclick="document.getElementById('audio1').volume+=0.1">Aumentar el Volumen</button> <button

onclick="document.getElementById('audio1').volume-=0.1">Disminuir el Volumen</button>

</div>

[Salida en el Navegador](compatibilidadAudio/fichero.html)

## **Programas de conversión de video**

Al igual que el audio el video tiene los mismos problemas de compatibilidad. Por ello debemos dar al usuario más de una opción por si su navegador no soporta algún formato de video.

Los formatos de video que utilizaremos para dar la compatibilidad son:

**MP4** = MPEG 4: utiliza el codec de vídeo H264 y el codec de audio AAC.

**WebM** = WebM: utiliza el codec de vídeo VP8 y el codec de audio Vorbis.

**Ogv** = Ogv: utiliza el codec de vídeo Theora y el codec de audio Vorbis.

Al igual que hemos utilizado el Audacity para poder transformar los formatos de los audios, ahora vamos a ver que programas disponemos para poder realizar la conversión:

* VLC: este programa es conocido como uno de los mejores a la hora de reproducción de video, pero también nos permite el convertir a distintos formatos un video.

Lo primero que debemos hacer es instalarlo, para ello hay que ir a su web y descargarnos el instalador:

Captura de pantalla de un celular

Descripción generada automáticamente

Una vez descargado hacemos click en el instalador y lo ejecutamos:

Captura de pantalla de un celular

Descripción generada automáticamente

Es un proceso de siguiente siguiente, sin necesidad de hacer ningún cambio de lo que viene por defecto. Al terminar nos permitirá ejecutar el programa:

Captura de pantalla de un celular

Descripción generada automáticamente

Ahora vamos con el proceso de convertir un video. Para ello abrimos la aplicación:

* Hacemos click sobre **medio** y elegimos la opción de **convertir**.

Captura de pantalla de un celular

Descripción generada automáticamente

* Añadimos el video que queremos convertir y hacemos click sobre convertir:

Captura de pantalla de un celular

Descripción generada automáticamente

Al hacer click sobre convertir pasamos a otra pantalla en donde debemos elegir el formato que queremos para el video y la carpeta de destino:

Captura de pantalla de un celular

Descripción generada automáticamente

Al hacer click en iniciar habremos completado el proceso y ya estará el video convertido en la carpeta de salida que le hayamos puesto.

## Ejemplo de implementación

Una vez transformados los videos en el formato correcto ya podemos implementarlo en una web. El código es el siguiente:

<video id=video1 width=300 height=200 autoplay=autoplay controls loop=loop>

<source src=video/video1.mp4 type=video/mp4>

<source src=video/video2.ogv type=video/ogg>

<source src=video/video3.webm type=video/webm>

Tu navegador no soporta esta etiqueta HTML5

</video>

Ahora vamos a implementar unos botones para controlar el video con javascript:

<button onclick="document.getElementById('video1').play()">Reproducir el Vídeo</button> <button

onclick="document.getElementById('video1').pause()">Pausar el Vídeo</button> <button

onclick="document.getElementById('video1').volume+=0.1">Aumentar el Volumen del Vídeo</button>

<button onclick="document.getElementById('video1').volume-=0.1">Disminuir el Volumen del Vídeo</button> </div>

[Salida en navegador](compatibilidadVideo/fichero.html)

## **CANVAS**

Canvas es una etiqueta de html5 que nos permite crear un lienzo en el que a través de javascript y funciones vamos a poder dibujar.

Para explicar el funcionamiento, voy a desarrollar un ejemplo básico:

Dibujar una línea:

var canvas = document.getElementById('canvas1');

var contexto = canvas.getContext("2d");

contexto.moveTo(200, 200);

contexto.lineTo(400, 200);

Dibujar Texto:

var canvas1 = document.getElementById('canvas1');

var contexto1 = canvas1.getContext("2d");

contexto1.font = "30px Arial";

contexto1.fillText("HOLA", 600, 200);

}

Dibujar Círculo:

var canvas2 = document.getElementById('canvas1');

var contexto2 = canvas2.getContext("2d");

contexto2.beginPath();

contexto2.arc(600, 70, 40, 0, 2 \* Math.PI);

Dibujar na imagen:

var canvasvar1 = document.getElementById("canvas1");

var contexto1 = canvasvar1.getContext("2d");

var imagen1 = document.getElementById("kamelot1");

contexto1.drawImage(imagen1, 5, 5);

[Ejemplo completo en el navegador.](canvas/subir/fichero.html)

## Propiedad Storage (localStorage y sessionStorage).

* Esta especificación nos va a proveer de dos objetos que nos van a servir para la **persistencia de datos** de manera estructurada en el lado del cliente.
* Es un mecanismo similar a las Cookies pero con la diferencia de que **no hay que enviar el contenido almacenado al servidor con cada petición**.
* Hay que tener en cuenta que solo se pueden almacenar **cadenas de texto**. Pero podemos almacenar objetos convirtiéndolos a formato JSON

Principales características:

* - A diferencia de las cookies el **límite de almacenamiento** es mucho mayor, al menos de 5MB.
* - Los datos se almacenan en pares nombre/valor.
* - Es posible almacenar grandes cantidades de datos sin afectar el rendimiento de la página web.
* - Los datos no se incluyen en cada petición al servidor, solamente se envían cuando es necesario
* - El almacenamiento web es más seguro y más rápido.

Compatibilidad con navegadores:

Captura de pantalla de un celular

Descripción generada automáticamente

Nosotros podemos comprobarlo con el siguiente script:

Captura de pantalla con letras y números

Descripción generada automáticamente

## EL OBJETO LOCALSTORAGE

* **Es uno de los dos objetos que nos proporciona la estandaridación webStorage.**
* **Lo característico de este objeto es que nos va a permitir almacenar datos en el lado del cliente de manera persistente (no tiene fecha de expiración). Es decir, los datos se mantendrán pese a:**

- Cerremos el navegador

- Abramos la web con otra ventana.

## El objeto sessionstorage

* Al igual que el anterior, este objeto nos va a permitir almacenar información en el lado del cliente. La peculiaridad de este objeto es que esta información solo estará disponible mientras este activa la sesión del navegador. Por ejemplo si cerramos el navegador o si lo abrimos en otra ventana perderemos la información.

## Implementación

## 

* Estos dos Objetos funcionan de una manera muy similar. Se implementan en javascript y comparten la mayoría de funciones. Las más importantes son las siguientes:

- sessionStorage.setItem(clave, valor);

- sessionStorage.getItem(clave);

- sessionStorage.revokeItem(clave);

* Interesante será usar la propiedad length para saber el número de los elementos que estamos guardando.

## Forma Simplificada

* Además de la forma que acabamos de ver para utilizar los objetos localStorage y sessionStorage hay otra forma de interactuar con ellos mediante javascript que es más sencilla.
* Podemos así almacenar datos de la siguiente forma:

- sessionStorage.clave = valor;

* O podemos borrar el contenido que llevamos almacenado con:

- sessionStorage.clear();

## **Ejemplo de localStorage:**

/\*Esto es para localStorage \*/

document.addEventListener('DOMContentLoaded', inicio);

var nombre;

var pwd;

function inicio(){

document.getElementById('guardar').addEventListener('click', guardarValores);

document.getElementById('mostrar').addEventListener('click', imprimirValores);

document.getElementById('borrar').addEventListener('click', borrarDatos);

}

function guardarValores(){

nombre = document.getElementById('idnombre').value;

pwd = document.getElementById('idpassword').value;

localStorage.nombre = nombre; //Guardamos el nombre || podemos hacerlo con localStorage.setItem(nombre)

localStorage.pwd= pwd;

}

function imprimirValores(){

if (localStorage.nombre != undefined && localStorage.pwd != undefined) {

document.getElementById('pantalla').innerHTML= "El nombre es: " +localStorage.nombre + " La contraseña es: "+ localStorage.pwd;

}

else{

document.getElementById('pantalla').innerHTML= "No hay datos para mostrar";

}

}

function borrarDatos(){

localStorage.clear();

}

[SALIDA EN EL NAVEGADOR](localstorage/index.html)

## **EJEMPLO DE SESSIONSTORAGE**

/\*Esto es para sessionStorage \*/

document.addEventListener('DOMContentLoaded', inicio);

var nombre;

var pwd;

function inicio(){

document.getElementById('guardar').addEventListener('click', guardarValores);

document.getElementById('mostrar').addEventListener('click', imprimirValores);

document.getElementById('borrar').addEventListener('click', borrarDatos);

}

function guardarValores(){

nombre = document.getElementById('idnombre').value;

pwd = document.getElementById('idpassword').value;

sessionStorage.nombre = nombre; //Guardamos el nombre || podemos hacerlo con sessionStorage.setItem(nombre)

sessionStorage.pwd= pwd; //guaradmos la password

}

function imprimirValores(){

if (sessionStorage.nombre != undefined && sessionStorage.pwd != undefined) {

document.getElementById('pantalla').innerHTML= "El nombre es: " +sessionStorage.nombre + " La contraseña es: "+ sessionStorage.pwd;

}

else{

document.getElementById('pantalla').innerHTML= "No hay datos para mostrar";

}

}

function borrarDatos(){

sessionStorage.clear(); //Borrara los datos almacenados

}

[SALIDA EN EL NAVEGADOR](sessionstorage/index.html)

## **GRID**

Grid es una de las opciones que tiene la propiedad display de CSS. Esta enfocado a diseños en rejilla, es decir cuando tenemos que colocar elementos que ocupan varias filas y varias columnas.

Hay que tener una cosa en cuenta y es que esta propiedad no se hereda, con lo que será necesario especificar, a los hijos de un contenedor Grid como display grid si lo necesitaramos (por ejemplo para un centrado perfecto del texto de uno de los hijos).

Para crear una disposición básica de grid hay que espedificar la propiedad:

Display: grid.

Vamos con un ejemplo.

.contenedorgrid{

display: grid;

width: 80%;

background-color: blueviolet;

padding: 20px;

grid-template-columns: auto auto auto;

&\_\_elementohijo{

background-color: greenyellow;

padding: 30px;

text-align: center;

border: 2px solid black;

}

}

Como vemos, al contenedor padre le ponemos el **display: grid**. Además con la propiedad **grid-template-columns**, nos permite especificar el número de columnas.

Hay que ser consecuente con las columnas que añadimos y el número de elementos de que disponemos como hijos del contenedor.

Vamos a ver el html:

<h2>Ejemplo grid básico</h2>

<div class="contenedorgrid">

<div class="contenedorgrid\_\_elementohijo">1</div>

<div class="contenedorgrid\_\_elementohijo">2</div>

<div class="contenedorgrid\_\_elementohijo">3</div>

<div class="contenedorgrid\_\_elementohijo">4</div>

<div class="contenedorgrid\_\_elementohijo">5</div>

<div class="contenedorgrid\_\_elementohijo">6</div>

<div class="contenedorgrid\_\_elementohijo">7</div>

</div>

Como podemos ver se ha utilizado la metodología bem para los nombres de las clases.

[SALIDA EN EL NAVEGADOR](gridBasico/subir/fichero.html)

Con esto ya hemos realizado un ejemplo básico del funcionamiento de grid.

## **Propiedad inline-grid**

La propiedad inline-grid en el contenedor padre, nos va a servir para que se cree un contenedor que se adapta al contenido, no utiliza el ancho completo del navegador.

Vamos con un ejemplo:

.contenedorgrid{

display: inline-grid;

background-color: blueviolet;

padding: 20px;

grid-template-columns: auto auto auto;

&\_\_elementohijo{

background-color: greenyellow;

padding: 30px;

text-align: center;

border: 2px solid black;

}

}

De momento el HTML del ejemplo es el mismo que el del ejemplo anterior.

[SALIDA EN EL NAVEGADOR](inlinegrid/subir/fichero.html)

## **Hueco entre elementos: Row-gap y column-gap**

Estas propiedades nos permiten establecer el espacio entre elementos dentro de un contenedor grid:

Imagen que contiene biombo, edificio

Descripción generada automáticamente

Se puede acortar y ahorrar código usando la propiedad gap, a modo de resumen de row-gap y column-gap. Siendo el primero de los tamaños que se especifica el tamaño de separación de las filas y el segundo el de las columnas.

Vamos ahora con un ejemplo:

.contenedorgrid{

display: inline-grid;

background-color: blueviolet;

padding: 20px;

grid-template-columns: auto auto auto;

gap: 60px 60px; /\* Primero row segundo column \*/

&\_\_elementohijo{

background-color: greenyellow;

padding: 30px;

text-align: center;

border: 2px solid black;

}

}

[SALIDA EN EL NAVEGADOR](propiedadGap/subir/fichero.html)

## **Column lines y Row lines**

Estas propiedades nos van a servir para colocar elementos en una disposición grid, son para que nos hagamos una idea, similar a las propiedades row-span y column-span.

Imagen que contiene biombo

Descripción generada automáticamente

Para poder implementarlo se utilizan las propiedades **grid-column-start y grid-column-end o también grid-row-start y grid-row-end.**

Vamos ahora con un ejemplo de uso:

.contenedorgrid{

display: inline-grid;

background-color: blueviolet;

padding: 20px;

grid-template-columns: auto auto auto;

&\_\_elementohijo{

background-color: greenyellow;

padding: 30px;

text-align: center;

border: 2px solid black;

}

&\_\_elemento1{

grid-column-start: 2;

grid-column-end: 3;

}

}

Es necesario especificar al elemento que queremos modificar. Por ello en el HTML le damos una clase adicional al elemento1.

<h2>Ejemplo grid-columns</h2>

<div class="contenedorgrid">

<div class="contenedorgrid\_\_elementohijo contenedorgrid\_\_elemento1">1</div>

<div class="contenedorgrid\_\_elementohijo">2</div>

<div class="contenedorgrid\_\_elementohijo">3</div>

<div class="contenedorgrid\_\_elementohijo">4</div>

<div class="contenedorgrid\_\_elementohijo">5</div>

<div class="contenedorgrid\_\_elementohijo">6</div>

<div class="contenedorgrid\_\_elementohijo">7</div>

</div>

[SALIDA EN NAVEGADOR](colocacionDeElementos/subir/fichero.html)

Ahora vamos con un ejemplo de **grid-row.**

.contenedorgrid{

display: inline-grid;

background-color: blueviolet;

padding: 20px;

grid-template-columns: auto auto auto;

&\_\_elementohijo{

background-color: greenyellow;

padding: 30px;

text-align: center;

border: 2px solid black;

}

&\_\_elemento1{

grid-row-start: 1;

grid-row-end: 3;

}

}

[SALIDA EN EL NAVEGADOR](grid-row/subir/fichero.html)

## **Grid-template-columns**

Ya se enunció antes esta propiedad, pero ahora vamos a hablar sobre ella con un poco más de detenimieto.

Es una propiedad que permite definir el número de columnas del grid. También permite definir el ancho de cada una de estas columnas. El ancho de estas columnas se especifica por separado como valor de la propiedad grid-template-columns.

Ejemplo – grid con 4 columnas de ancho automático

**grid-template-columns: auto auto auto auto;**

Ejemplo – grid con 3 columnas con un ancho prefijado

**Grid-template-columns: 120px 100px 200px 40px;**

.contenedorgrid{

display: grid;

background-color: blueviolet;

padding: 20px;

grid-template-columns: auto auto auto;

&\_\_elementohijo{

background-color: greenyellow;

padding: 30px;

text-align: center;

border: 2px solid black;

}

}

.contenedorgrid1{

display:grid;

background-color: blueviolet;

padding: 20px;

grid-template-columns: 200px 300px 400px;

&\_\_elementohijo{

background-color: greenyellow;

padding: 30px;

text-align: center;

border: 2px solid black;

}

}

[SALIDA EN NAVEGADOR](grid-template-coumns/subir/fichero.html)

## **Grid-template-rows**

Esta propiedad permite definir la altura de cada una de las filas del grid.

Vamos con un ejemplo:

.contenedorgrid{

display: grid;

background-color: blueviolet;

padding: 20px;

grid-template-columns: auto auto auto;

&\_\_elementohijo{

background-color: greenyellow;

padding: 30px;

text-align: center;

border: 2px solid black;

}

}

.contenedorgrid1{

display:grid;

background-color: blueviolet;

padding: 20px;

grid-template-columns: auto auto auto;

grid-template-rows: 80px 200px 400px;

&\_\_elementohijo{

background-color: greenyellow;

padding: 30px;

text-align: center;

border: 2px solid black;

}

}

[SALIDA EN EL NAVEGADOR](grid-template-rows/subir/fichero.html)

## **Justify-content**

Es una propiedad que permite alinear horizontalmente los elementos grid dentro del contenedor. Es importante saber que el ancho total de los elementos grid debe ser inferior al ancho del contenedor para que podamos observar su efecto.

Justify-content puede tomar una serie de valores: ·

* **space-evenly:** añade el mismo espacio entre columnas y alrededor de ellas.
* **· space-around:** añade el mismo espacio alrededor de las columnas. Añade a la izquierda y derecha de cada elemento y si son adyacentes, se suman entre ellos los espacios. ·
* **space-between:** añade el mismo espacio entre las columnas. · center: alinea todos los elementos grid en el centro del contenedor. ·
* **start:** alinea todos los elementos grid al inicio del contenedor. ·
* **end:** alinea todos los elementos grid al final del contenedor. ·
* **stretch:** opción por defecto. Rellena el espacio disponible.

Captura de pantalla de un celular

Descripción generada automáticamente

Vamos con un ejemplo de uso:

.contenedorgrid{

display: grid;

background-color: blueviolet;

padding: 20px;

grid-template-columns: auto auto auto;

justify-content: space-evenly;

&\_\_elementohijo{

background-color: greenyellow;

padding: 30px;

text-align: center;

border: 2px solid black;

}

}

[SALIDA EN EL NAVEGADOR](justify-content/subir/fichero.html)

## **Align-content**

Es una propiedad que permite alinear verticalmente los elementos grid dentro del contenedor.

Puede tomar los mismos valores que justify-content: ·

* **space-evenly:** añade el mismo espacio entre columnas y alrededor de ellas. ·
* **space-around:** añade el mismo espacio alrededor de las columnas.
* · **space-between:** añade el mismo espacio entre las columnas. · center: alinea todos los elementos grid en el centro del contenedor.
* · **start:** alinea todos los elementos grid al inicio del contenedor.
* **· end:** alinea todos los elementos grid al final del contenedor.

Vamos con un ejemplo:

|  |
| --- |
| @mixin justifycontent($valor) { |
|  | justify-content: $valor; |
|  |  |
|  | } |

[SALIDA EN NAVEGADOR](align-content/subir/fichero.html)

## Grid-area

Se puede utilizar como manera de simplificar las propiedades grid-row-start, grid-column-start, grid-row-end y grid-column-end

Vamos con un ejemplo:

.contenedorgrid{

display: grid;

grid-template-columns: repeat(4, auto);

grid-template-rows: repeat(4, auto);

background-color: #2196f3;

gap: 10px;

padding: 10px;

&\_\_elementohijo{

background-color: grey;

text-align: center;

padding: 20px 0;

font-size: 30px;

}

&\_\_elemento1{

//grid-column: 1 / 5;

grid-area: 1 / 1 / 3 / 4;

}

&\_\_elemento4{

grid-area: 3 / 1 /5 /2;

}

&\_\_elemento6{

grid-area: 3/ 3/5 /5;

}

/\* colocación de elementos \*/

&\_\_elem1{

grid-area: 1 /3 / 2 /4 ;

}

&\_\_elem2{

grid-area: 2/3/3/5;

}

&\_\_elem3{

grid-area: 1/1/2/2;

}

&\_\_elem4{

grid-area: 1/2/2/5;

}

&\_\_elem5{

grid-area: 2/1/3/2

}

&\_\_elem6{

grid-area: 2/2/3/3;

}

}

[SALIDA EN NAVEGADOR](grid-area/subir/fichero.html)

## **Imágenes responsive**

Vamos a ver una serie de soluciones para este problema:

· Establecer width como porcentaje y height a auto

&\_\_imagenresponsive{

width: 100%;

height: auto;

}

· Establecer la anchura máxima a 100%. Esto obligará a que nunca se pueda escalar por encima del 100%

.imagenresponsivemaxwidth{

max-width: 100%;

height: auto;

display: block;

margin: auto;

}

· Imágenes de fondo responsivas: debemos establecer la propiedad background-size a contain

.contenedorimagen{

display: block;

margin: auto;

border: 2px solid black;

width: 70%;

min-height: 500px;

background-image: url("img/horse.jpg");

margin-top: 40px;

background-repeat: no-repeat;

background-size: contain;

}

**Propiedad Object-fit.**

Propiedad esencial para que los elementos se adapten adecuadamente a su contenedor.

Mediante object-fit: cover es posible recortar la imagen en sus laterales y seguir manteniendo su proporción de aspecto.

El valor scale-down, escala la imagen para que se adapte, pero sin perder su proporción de aspecto.

Sin object-fit:cover

img {

width:400px;

height:800px;

}

<img src="ColosoDeRodas.jpg" alt="Coloso de Rodas">

Con object-fit:cover

img {

width:400px;

height:800px;

object-fit:cover;

}

<img src="ColosoDeRodas.jpg" alt="Coloso de Rodas">

[SALIDA EN NAVEGADOR DE LOS EJEMPLOS ANTERIORES](imagenesResponsivas/subir/fichero.html)