**CSS avanzado**

Diseño de interfaces web

**Contenidos** *1. Media queries* 2. Tipos de diseño web

3. Metaetiqueta *viewport*

4. Sombras

5. Gradientes

6. Selector de atributos

7. Transiciones y transformaciones

8. Animaciones

U.T.3. CSS avanzado

9. Flexbox, modelo de caja flexible 10. Centrado vertical y horizontal 11. Preprocesadores CSS

***1. Media queries***

**1. Media queries**

Las *media queries* se introdujeron en CSS3 para dar respuesta a las necesidades del **diseño web responsivo** (o en inglés *responsive*). Mediante ellas podemos definir **estilos condicionales**, aplicables únicamente en determinadas situaciones.

Utilizamos ***media queries*** para establecer **estilos diferentes para cada ancho de pantalla**. **@media (min-width: ... ) { ... }**

**1.1. Sintaxis**

La sintaxis general de una *media query* es la siguiente:

**@media not|only media\_type and (media\_feature and|or|not media\_feature) { /\* Código CSS \*/**

**}**

La sintaxis consiste de un **tipo de medio opcional** (impreso, pantalla o hablado [útil para los equipos adaptados y accesibles]) y una o más expresiones de **características de medios**. Se pueden combinar varias consultas empleando operadores lógicos. No se distingue entre mayúsculas y minúsculas.

**1.1.1. Tipos de medios o *Media Types***

Entre los **tipos de medios** o ***media types***, tenemos:

● **all**: Todos los dispositivos. (Valor por defecto)

● **print**: Material impreso y modo de vista previa de impresión. ● **screen**: Destinado a las pantallas.

● **speech**: Destinado a sintetizadores de voz.

**1.1.2. Propiedades o *Media Feature* principales**

Entre las **características de medios** principales o ***media features***, tenemos: ● **height**: Altura de la superficie a renderizar en el dispositivo de salida. ● **width**: Ancho de la superficie a renderizar en el dispositivo de salida. ● **max-height / min-height**

● **max-width / min-width**

**1.1.3. Palabras clave not, only y and:**

● **not**: invierte el significado de una consulta de medios completa. ● **only**: evita que navegadores antiguos que no admiten *queries* apliquen los estilos indicados. No tiene ningún efecto en navegadores modernos. ● **and**: combina varios tipos de medios o características.

Todas son opcionales. Sin embargo, si se usan not u only, también se debe especificar el tipo de medio.

**1.1. Sintaxis**

**@media not|only media\_type and (media\_feature and|or|not media\_feature) { /\* Código CSS \*/**

**}**

<!-- Media query CSS en un elemento link -->

<**link** rel="stylesheet" media="(max-width: 600px)" href="example.css" />

<!-- Media query CSS en una hoja de estilo -->

<**style**>

@media (max-width: 600px) {

.sidebar {

display: none;

}

}

<**/style**>

**1.1. Sintaxis**

Entrada *CSS media queries* en MDN Web Docs.

**1.2. Cómo funciona una *media query***

**Actividad**

Crea las reglas necesarias utilizando *media queries* para mostrar un título **h1** con contenido “Escritorio”, “Tableta” o “Móvil” en función del tamaño del *viewport* (Escritorio > 1024px, Tableta <= 1024px y Móvil <= 480px). Únicamente para pantallas.

**1.2. Cómo funciona una *media query***

La siguiente *media query* especifica un estilo para ser utilizada cuando la ventana tiene un ancho entre **450** y **800** píxeles:

@media screen and (min-width: 450px) and (max-width: 800px) {

...

}

**1.3. Otras propiedades o *media features*** A parte del ancho y alto, hay muchas otras propiedades que se pueden emplear:

**Propiedad Descripción**

**width** Anchura del ***viewport****.*

**height** Altura del ***viewport****.*

**aspect-ratio** Relación de aspecto entre la anchura y la altura del ***viewport****.* (p. ej., **(aspect-ratio: 1/1)**)

**orientation** Orientación del ***viewport****.* (**portrait** o **landscape**)

**resolution** Densidad de píxeles del dispositivo. (p. ej., **(resolution: 150dpi)**) **scan** Renderizado de la pantalla del dispositivo (entrelazado o progresivo). **grid** Si el dispositivo es grid o bitmap. (Ej. grid: dispositivo braille. Ej. bitmap: proyector)

**1.3. Otras propiedades o *media features***

**Propiedad Descripción**

**update-frequency** Velocidad de actualización del dispositivo.

**overflow-block** Cómo maneja el dispositivo el contenido que excede los límites del ***viewport*** a lo largo del eje del bloque.

**overflow-inline** Cómo maneja el dispositivo el contenido que excede los límites del ***viewport*** a lo largo del eje en línea.

**color** Componente de número de bits por color del dispositivo, o cero si el dispositivo no es a color.

**color-index** Número de entradas en la tabla de búsqueda de color del dispositivo, o cero si el dispositivo no usa una tabla.

**monochrome** Bits por píxel en el *buffer* de marco monocromático del dispositivo, o 0 si el dispositivo no es monocromático.

**hover** Si se puede posicionar el puntero sobre los elementos.

**1.4. Operadores lógicos**

Las *media queries* nos permiten utilizar **operadores lógicos** para **comprobar si se cumple una condición**. **Operador AND**

El siguiente código aplicará estilos cuando la pantalla tenga un ancho mínimo de **700px** y la orientación sea horizontal (*landscape*).

@media (min-width: 700px) and (orientation: landscape) { ... }

**Operador NOT**

El siguiente código aplicará estilos cuando no se cumpla la condición especificada, es decir a cualquier elemento que no sea una pantalla convencional ni sea monocromo.

@media not screen and (monochrome) { ... }

**1.5. *Mobile first***

Una de las dudas principales que se plantean a la hora de usar las ***media queries*** es si definir el estilo general del CSS para escritorio y reservar los condicionales para los móviles o hacerlo a la inversa. Estamos hablando de las corrientes ***Desktop First*** (escritorio primero) y ***Mobile First*** (móvil primero).

**Mobile First Desktop First**

**1.5. *Mobile first***

Se ha discutido mucho sobre este asunto, pero poca gente duda ya de que lo más interesante es trabajar con ***Mobile First***, es decir, tener unos estilos para móvil como base (pequeñas resoluciones) e ir creciendo poco a poco añadiendo *media queries* para dispositivos más grandes.

**Actividad 3.3**

Con esta actividad vamos a practicar el uso de *media queries*. Para ello tendrás que crear una página con 4 estilos distintos.

El primer estilo estará destinado a equipos de **escritorio**, el segundo a dispositivos tipo ***tablet*** y el tercero a **móviles**. Los estilos se incluirán dentro del elemento **<style>**.

El último y cuarto estilo estará orientado al medio de **impresión**. Los estilos se incluirán en un fichero externo.

**Actividad 3.3. *Media queries***

**Versión móvil Versión *tablet* Versión escritorio**

**Actividad 3.3. *Media queries***

**Elementos comunes a la versión móvil, *tablet* y escritorio:**

● Cabecera: compuesta por 2 cajas con fondo **naranja (#ff9900)**

● Contenido principal: compuesto por 1 caja con fondo **verde (#c4d20a)**.

● Contenido relacionado: compuesto por 3 cajas **azules (#3d85c6)**.

● Pie de página: compuesto por 4 cajas **grises (#595959)**.

Las cajas no deben tener contenido. Pero deberán tener un tamaño y distribución distintos en función del tamaño de pantalla del dispositivo desde el que se visualice la página.

**Actividad 3.3. *Media queries***

**Versión impresión:**

Deberá mostrarse únicamente texto (sin imágenes), donde se incluya el título de la página, título de una sección, algunos párrafos y el pie de página.

**Nota**: todo el contenido debe ser inventado (p.ej., a través de *Lorem ipsum*)

**2. Tipos de diseño**

**2. Tipos de diseño**

A la hora de crear una interfaz web existen diferentes tipos de diseño. A continuación citaremos algunos de los más conocidos:

● Diseño fijo

● Diseño elástico

● Diseño líquido o fluido

● Diseño responsivo o adaptable

● Diseño flexible

**2.1. Diseño fijo**

El diseño fijo dispone de **medidas fijas** que no son modificadas para los distintos dispositivos. Por lo tanto, se trata de un diseño que no cumple las normas de un diseño web responsivo o *responsive web design*.

**2.2. Diseño elástico**

Los diseños elásticos se crean utilizando **medidas relativas** en lugar de medidas absolutas. El ancho de la página no es fijo. El tamaño de la página y sus elementos están declarados en relación al tamaño del texto, que se suele especificar en em. La principal ventaja de este diseño es que cuando se varía el tamaño del texto, el diseño se ajusta perfectamente al texto, sin desbordamientos.

Ancho fijo Ancho NO

fijo

**2.3. Diseño líquido o fluido**

En este diseño, los elementos de una página web se comportan como el agua. Se extienden por la superficie de la página web como el agua por una superficie y se adaptan al recipiente que los contiene. En general este es el comportamiento natural de una página web si no se hace nada.

Ejemplo: primera página web de la historia, se comporta bien a cualquier resolución.

**2.3. Diseño líquido o fluido**

En el diseño líquido el ancho de la página no es fijo, el tamaño de la página y sus elementos están declarados en relación al ancho de la ventana (que se suele especificar en porcentaje). La principal ventaja de este diseño es que se adapta a los diferentes dispositivos de visualización.

**2.4. Diseño responsivo o adaptable**

El **diseño web responsivo** o ***responsive web design*** (también llamado **adaptable** o ***adaptative***) se basa en el uso de ***media queries***. Gracias a esta técnica podemos definir estilos condicionales con puntos de ruptura o puntos de interrupción, aplicables únicamente en determinadas situaciones.

**Resoluciones**

**Pequeñas Medianas Grandes Extra grandes** 320x480 768x1024 1280x800 1440x900 640x480 800x600 1280x1024 1600x900 1024x600 1360x768 1680x1050

1024x768 1366x768 1920x1080

**2.4. Diseño responsivo o adaptable**

**2.4. Diseño responsivo o adaptable**

El **diseño web responsivo** o ***responsive web design*** (también llamado **adaptable** o ***adaptative***) se basa en el uso de ***media queries***. Gracias a esta técnica podemos definir estilos condicionales con puntos de ruptura o puntos de interrupción, aplicables únicamente en determinadas situaciones.

**2.4. Diseño responsivo o adaptable**

**2.5. Diseño flexible**

Consiste en utilizar las propiedades CSS **min-width** y **max-width** para que las anchuras de los bloques puedan adaptarse dentro de unos mínimos y máximos.

**2.5. ¿Qué tipo de diseño utilizo?**

Qué tipo de método utilizar para nuestros proyectos es un tema muy debatido en la red porque crear un sitio web adaptable exige, en muchos casos, **combinar algunas de las técnicas que hemos estudiado**. Lo más habitual es declarar más de una media query para adaptar el diseño a múltiples dispositivos. Pero el número de puntos de interrupción y cómo se implementan estas técnicas **depende del diseño de nuestro**

**sitio web**.

**Autoevaluación**

Ponte a prueba con los siguientes ejercicios donde practicar algunos de los diseños que hemos visto hasta ahora:

● Maquetación diseño fijo

● Maquetación diseño elástico

● Maquetación diseño líquido

● Maquetación diseño adaptable 1

● Maquetación diseño adaptable 2

**3. Metaetiqueta *viewport***

**3. Metaetiqueta viewport**

HTML5 introdujo un método para permitir a los diseñadores tomar el control sobre el *viewport* a través de la etiqueta <meta>.

Debemos incluir el siguiente elemento en todas nuestras páginas: <**meta** name="viewport" content="width=device-width, initial-scale=1">

**3. Metaetiqueta viewport**

Incluyendo la etiqueta anterior le damos al navegador instrucciones sobre cómo controlar las dimensiones y la escala de la página.

La parte **width=device-width** establece el ancho de la página para seguir el ancho de la pantalla del dispositivo (que varía según el dispositivo).

La parte **initial-scale=1.0** establece el nivel de *zoom* inicial cuando el navegador carga la página por primera vez.

**3. Metaetiqueta viewport**

Además, cuando trabajamos con webs responsivas es importante definir un **viewport** adecuado, de lo contrario es muy probable que la página no lea correctamente las **media queries** y que se vea en formato muy reducido, siendo necesario hacer *zoom* para ver el contenido.

**3. Metaetiqueta viewport**

A continuación vemos un ejemplo de una página **sin** y **con** la metaetiqueta:

**3. Metaetiqueta viewport**

Disponemos de los siguientes parámetros en la metaetiqueta **viewport**.

● **width**: anchura virtual (emulada) de la pantalla o anchura del **viewport**. ● **height**: altura virtual de la pantalla o altura del **viewport**. ● **initial-scale**: la escala inicial del documento.

● **minimum-scale**: la escala mínima que se puede establecer en el documento. ● **maximum-scale**: la escala máxima configurable en el documento. ● **user-scalable**: si se permite al usuario hacer *zoom*.

**3. Metaetiqueta viewport**

Ejemplos de etiqueta **viewport** que **NO permiten** escalar la página: <**meta** name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">

<**meta** name="viewport" content="width=device-width, initial-scale=1, user-scalable=no"> Ejemplo de etiqueta **viewport** que **permite** escalar la página: <**meta** name="viewport" content="width=device-width,initial-scale=1.0">

**4. Sombras**

**4. Sombras**

Gracias a las propiedades de CSS3 **text-shadow** y **box-shadow** podemos aplicar sombras en los textos y contenedores para mejorar nuestros diseños (siempre que no abusemos del recurso).

**4.1. Propiedades text-shadow y box-shadow**

El formato es el mismo para las dos propiedades. A continuación se muestra un ejemplo de uso:

/\* desplazamiento-x | desplazamiento-y | difuminado | color \*/

text-shadow: 3px 4px 0px #fff;

/\* desplazamiento-x | desplazamiento-y | difuminado | grosor | color \*/ box-shadow: 10px 10px 14px 2px rgba(0,0,0,0.47);

Veamos por ejemplo qué significan los valores de la propiedad **text-shadow** del ejemplo anterior:

● **3px**: sombra que se proyecta hacia la derecha del texto.

● **4px**: sombra que se proyecta hacia abajo del texto.

● **0px**: radio de desenfoque.

● **#fff**: color de la sombra. Se puede aplicar también en RGBA y HSLA. Para aplicar sombras arriba y a la izquierda se deben utilizar valores negativos.

**4.1. Propiedades text-shadow y box-shadow** Para incluir varias sombras podemos hacerlo separándolas por comas: text-shadow: 3px 4px 0px #fff, 2px 2px 1px #ccc;

En este tipo de propiedades es importante incluir los prefijos de los navegadores.

**Prefijo Navegador**

**-moz-** Firefox

**-webkit-** Safari y Chrome

**-o-** Opera

**-khtml-** Konqueror

**-ms-** Internet Explorer

**-chrome-** Google Chrome

**4.1. Propiedades text-shadow y box-shadow**

<**p** class="shadow-1">Texto con sombra</**p**>

<**p** class="shadow-2">Texto en contenedor con sombra</**p**>

<**p** class="shadow-3">Texto en contenedor con sombra interior</**p**>

<**p** class="shadow-4">Texto con dos sombras</**p**>

.shadow-1 { text-shadow: 3px 3px 2px rgba(152, 150, 207, 0.7); }

.shadow-2 { box-shadow: 10px 10px 14px 2px rgba(0,0,0,0.47); }

.shadow-3 { box-shadow: 5px 1px 4px 2px #45f875 inset; }

.shadow-4 { text-shadow: 3px 4px 0px grey, 2px 2px 1px yellow; }

**4.2. Generadores de sombra *online***

Para facilitarnos la tarea de crear sombras podemos utilizar generadores de sombras tanto para los textos como para las cajas o contenedores.

A continuación tenemos algunos ejemplos de herramientas *online*:

● Generador de sombras en textos: https://css3gen.com/text-shadow/ ● Generador de sombras en cajas:

https://cssgenerator.org/box-shadow-css-generator.html

**4.2. Generadores de sombra *online***

Intenta reproducir, mediante el uso de box-shadow, las siguientes sombras:

Algunas propiedades de las cajas:

● background-color: Crimson;

● width: 400px;

● height: 40px;

● padding: 10px;

**5. Gradientes**

**5. Gradientes**

Los gradientes permiten añadir efectos de colores degradados a nuestros diseños. Están disponibles desde CSS3 y se configuran como fondos, por lo que usaremos la propiedad **background**. Disponemos de 3 tipos de gradientes: **lineal**, **radial** y **cónico**.

**linear-gradient() radial-gradient() conic-gradient()**

**5.1. Gradiente lineal**

El gradiente lineal se define mediante la función **linear-gradient()** y permite crear fondos degradados en una dirección específica.

**5.1. Gradiente lineal**

background: linear-gradient(dirección | ángulo, color, color, color, ...); background: linear-gradient(dirección | ángulo, color tamaño, color tamaño, ...); background: linear-gradient(color, color, color...);

● **Dirección | ángulo**: indica la dirección o ángulo del gradiente. Puede ser especificado usando las palabras clave **to top**, **to bottom**, **to left** y **to right**. También puede ser indicado por un ángulo para declarar una dirección específica del gradiente.

● **Color**: permite crear una lista con varias paradas para ir cambiando el color del gradiente. Se pueden definir tantos colores como se necesiten.

● **Tamaño**: altura a la que comienza a cambiar el color del gradiente.

**5.1. Gradiente lineal**

Ejemplo de **linear-gradient()**:

background: linear-gradient(90deg, rgba(176,174,238,1) 0%, rgba(29,30,31,1)); background: linear-gradient(to right, red,brown,orange,yellow,blue,green,violet,pink); background: linear-gradient(to top, #000, #ccc);

background: linear-gradient(to right,red 300px,brown 400px,orange 700px);

**5.2. Gradiente radial**

El gradiente radial nos permite crear degradados con formas circulares. La sintaxis para los gradientes radiales es muy parecido al anterior. En este caso, debemos usar la función **radial-gradient()** y un nuevo atributo para la forma.

**5.2. Gradiente radial**

background: radial-gradient(forma, color, color, color...); background: radial-gradient(forma tamaño, color, color, color...); background: radial-gradient(forma, color tamaño, color tamaño...); background: radial-gradient(color, color, color...);

Significado de las propiedades:

● **forma**: existen dos tipos de forma: **circle** y **ellipse** (el valor por defecto será **ellipse**). De forma optativa se puede indicar el tamaño de la forma. ● **color**: permite crear una lista con varias paradas para ir cambiando el color del gradiente. Se pueden definir tantos colores como se necesiten.

● **tamaño**: para cada color se puede indicar la posición en la que comienzan las transiciones.

**5.2. Gradiente radial**

Ejemplo de **radial-gradient()**:

background: radial-gradient(ellipse, rgba(176,174,238,1) 0%, rgba(29,30,31,1)); background: radial-gradient (circle,red,brown,orange,yellow,blue);

background: radial-gradient(circle,red 300px,brown 400px,orange 700px);

background: radial-gradient(#000, #ccc);

**5.3. Gradiente cónico**

El gradiente cónico crea una imagen que consta de un degradado con transiciones de color giradas alrededor de un punto central (en lugar de irradiar desde el centro). En este caso, debemos usar la función **conic-gradient()**.

**5.3. Gradiente cónico**

Ejemplo de **conic-gradient()**:

/\* Un gradiente cónico girado 45 grados, comenzando en azul y terminando en rojo \*/ background: conic-gradient(from 45deg, blue, red);

/\* Cuadro violeta azulado: degradado de azul a rojo, solo el cuadrante inferior derecho es visible ya que el centro del degradado cónico está en la esquina superior izquierda. \*/

background: conic-gradient(from 90deg at 0 0, blue, red);

/\* Rueda de color \*/

background: conic-gradient(

hsl(360, 100%, 50%), hsl(315, 100%, 50%), hsl(270, 100%, 50%), hsl(225, 100%, 50%), hsl(180, 100%, 50%), hsl(135, 100%, 50%), hsl(90, 100%, 50%), hsl(45, 100%, 50%), hsl(0, 100%, 50%)

);

**5.4. Generadores de gradientes**

Para facilitarnos la tarea de la creación de gradientes podemos utilizar un generador de gradientes *online* que nos proporcione el código CSS necesario para nuestro diseño.

A continuación tenemos algunos ejemplos de herramientas *online*:

● cssgradient.io

● css3gen.com/gradient-generator

**5.5. Patrones de gradientes**

Podemos aplicar diseños muy trabajados creando patrones sin necesidad de cargar imágenes con mucho peso. Existen diversos artistas dedicados a hacer este tipo de diseños. A continuación mostramos algunos ejemplos:

● leaverou.github.io/css3patterns

**6. Selector de atributos**

**6. Selector de atributos**

Los selectores de atributos permiten elegir elementos HTML en función de sus atributos y/o valores de esos atributos. Los tipos de selectores de atributos son los siguientes:

● **[nombre\_atributo]**, selecciona los elementos que tienen establecido el atributo llamado nombre\_atributo independientemente de su valor.

● **[nombre\_atributo=valor]**, selecciona los elementos que tienen establecido un atributo llamado nombre\_atributo con un valor igual a valor.

● **[nombre\_atributo~=valor]**, selecciona los elementos que tienen establecido un atributo llamado nombre\_atributo y al menos uno de los valores del atributo es la palabra "valor".

**6. Selector de atributos**

● **[nombre\_atributo|=valor]**, selecciona los elementos que tienen establecido un atributo llamado nombre\_atributo, cuyo valor es una lista de valores, donde alguno comienza por valor o valor-.

● **[nombre\_atributo$=valor]**, selecciona aquellas etiquetas cuyo atributo acabe en ese valor. ● **[nombre\_atributo^=valor]**, selecciona aquellas etiquetas cuyo atributo comience por ese valor.

● **[nombre\_atributo\*=valor]**, el atributo **nombre\_atributo** existe y contiene valor.

**6. Selector de atributos**

**Ejemplos**

**Atributo Descripción**

**[href]** El atributo **href** existe en la etiqueta.

**[href="#"]** El atributo **href** existe y su valor es igual al texto **#**.

**[href\*="ejemplo"]** El atributo **href** existe y su valor contiene el texto **ejemplo**. **[href^="https://"]** El atributo **href** existe y su valor comienza por **https://**.

**[href$=".pdf"]** El atributo **href** existe y su valor termina en **.pdf**.

**[class~="ejemplo"]** El atributo **class** contiene una lista de valores, que contiene **ejemplo**.

**[lang|="es"]** El atributo **lang** contiene una lista de valores, donde alguno es igual a **es** o empieza por **es-**.

**6. Selector de atributos**

**Ejemplos**

Selecciona aquellas imágenes con extensión **png**:

img[src$=png]

Se muestran de color azul todos los enlaces que tengan un atributo **class**, sin importar su valor: a[class] { color: blue; }

Se muestran de color azul todos los enlaces que tengan un atributo **class** con el valor **externo**: a[class="externo"] { color: blue; }

**6. Selector de atributos**

**Ejemplos**

Se muestran de color azul todos los enlaces que apunten al sitio “http://www.ejemplo.com”. a[href="http://www.ejemplo.com"] { color: blue; }

Selecciona los elementos de la página con atributo **lang** igual a **en**, es decir, los elementos en inglés. \*[lang=en] { ... }

Selecciona los elementos de la página con atributo **lang** empiece por **es**, (**es**, **es-ES**, **es-AR**, etc.) \*[lang|="es"] { color : red }

**6. Selector de atributos**

**Ejercicios**

Ponte a prueba con los siguientes ejercicios:

● Ejercicio 1

● Ejercicio 2

● Ejercicio 3

● Ejercicio 4

● Ejercicio 5

● Ejercicio 6

**6. Selector de atributos**

**Actividad 3.1.**

Crea un documento HTML con varias imágenes. La direcciones de las imágenes serán obtenidas a través de internet, es decir, el valor **href** apuntará a una ruta remota. Algunas de las imágenes contendrán un valor **href** inexistente.

Utiliza los selectores de atributos, la pseudo-clase **:not** y tus conocimientos de CSS para dibujar un borde rojo (**border: 3px solid red;**) a todas las imágenes que no tengan el atributo **alt** o lo tengan sin valor.

**6. Selector de atributos**

**Actividad 3.2.**

Utiliza los selectores de atributos y el pseudo-elemento **after** para añadir un icono de pdf (como el que se muestra a continuación) justo detrás de todos los enlaces que apunten a un pdf.

Icono: https://cdn1.iconfinder.com/data/icons/Keyamoon-IcoMoon--limited/16/file-pdf.png **7. Transiciones y transformaciones**

**7.1. Transiciones**

Las transiciones en CSS nos permiten aplicar un cambio de estilo gradual a los elementos del documento HTML. Además, nos ofrecen la ventaja de poder especificar el tiempo para que se produzca la transformación entre estilos, de esta forma podríamos utilizarlas para dar un efecto de animación.

Todos los parámetros para aplicar las transiciones se pueden establecer en una sola línea y también mediante propiedades por separado.

Sintaxis:

transition: [Propiedad a modificar] [Duración] [Tipo de animación] [Retardo];

**7.1. Transiciones**

Ejemplo:

<**div** class="box"></**div**>

.box {

width: 200px;

height: 200px;

background-color: #C0392B;

transition: background-color 1s linear; }

.box:hover {

background-color: #3F51B5;

}

El efecto de transición comenzará cuando la propiedad CSS especificada (background-color) cambie de valor.

**7.1. Transiciones**

**Propiedades a modificar**: A continuación vemos algunas de las propiedades que podemos modificar.

**Propiedades a modificar**

**all bottom line-height padding background-color left margin stroke border right opacity text-shadow border-radius box-shadow word-spacing vertical-align color width letter-spacing visibility top height fill z-index**

**7.1. Transiciones**

**Duración en segundos**: especifica el número de segundos que durará la animación. P. ej.: **3s** (3 segundos).

**Tipo de animación**: esta propiedad es opcional y sirve para indicar la velocidad de la animación. Algunas de las posibilidades son las siguientes:

● **linear**: la velocidad de la animación es uniforme en todo el recorrido.

● **ease**: la velocidad se acelera al inicio, luego se retarda un poco y se acelera al final de nuevo. ● **ease-in**: la animación empieza lentamente y va aumentando progresivamente. ● **ease-out**: la animación empieza muy rápida y va descendiendo progresivamente. ● **ease-in-out**: la animación empieza y acaba lentamente, y es en la parte central del recorrido donde la velocidad es más rápida.

Algunos ejemplos adicionales aquí.

**7.1. Transiciones**

**Retardo**: tiempo (en segundos) que el navegador espera antes de poner en marcha la animación.

Se especifica el número de segundos a esperar seguido de la "**s**" (ejemplo: **1s**).

**7.1. Transiciones**

Realiza el cuestionario sobre transiciones del aula virtual.

**7.2. Transformaciones**

Las transformaciones CSS nos permiten rotar, torcer, escalar o desplazar los elementos de nuestra página web. Este tipo de propiedades de CSS3 son muy interesantes para convertir el lenguaje de hojas de estilo en un sistema capaz de realizar todo tipo de efectos visuales.

Las dos propiedades que nos sirven para definir las transformaciones son **transform** y **transform-origin**.

**7.2. Transformaciones**

Las **transformaciones** CSS nos permiten **rotar**, **torcer**, **escalar** o **desplazar** los elementos de nuestra página web. Este tipo de propiedades de CSS3 son muy interesantes para realizar todo tipo de efectos visuales.

Las dos propiedades que nos sirven para definir las transformaciones son: ● **transform-origin**: La posición de origen que se utilizará para la transformación es por defecto el lado superior izquierdo del elemento. ● **transform**: La posición de origen para realizar la transformación es el eje central del elemento.

**7.2. Transformaciones**

Las transformaciones que podemos aplicar son las siguientes:

● **Scale**: modifica el tamaño de los elementos.

● **Translate**: cambia la posición del elemento. Izquierda, derecha, arriba o abajo. ● **Rotate**: gira o rota los elementos en grados.

● **Skew**: distorsiona los elementos.

● **Matrix**: mueve o transforma los elementos con precisión de píxel.

● **Perspective**: establece la distancia entre el usuario y el plano z, la perspectiva desde la que estaría el espectador si la interfaz bidimensional fuera tridimensional.

**7.2. Transformaciones**

La propiedad **transform** se utiliza junto con la propiedad **transition** vista en el punto anterior:

<**div** id="box"></**div**>

#box {

width: 100px;

height: 100px;

background-color: #c0392b;

-webkit-transition: 1s linear;

transition: 1s linear;

}

#box:hover {

-webkit-transform: scale(0.5);

transform: scale(0.5);

}

**7.2. Transformaciones** Ejemplo con múltiples tipos de transformaciones y transiciones:

**8. Animaciones**

**8. Animaciones**

Con CSS3 podemos realizar animaciones de elementos HTML sin usar *JavaScript*. **¿Qué son las animaciones CSS?**

● Una animación permite que un elemento cambie gradualmente de un estilo a otro.

● Podemos cambiar tantas propiedades CSS y tantas veces como queramos. ● Para utilizar una animación CSS, primero debemos especificar algunos fotogramas clave de la animación.

● Los fotogramas clave contienen los estilos que tendrá el elemento en determinados momentos.

**8. Animaciones**

Con CSS3 podemos realizar animaciones de elementos HTML sin usar *JavaScript*. **¿Qué son las animaciones CSS?**

● Una animación permite que un elemento cambie gradualmente de un estilo a otro.

● Podemos cambiar tantas propiedades CSS y tantas veces como queramos. ● Para utilizar una animación CSS, primero debemos especificar algunos fotogramas clave de la animación.

● Los fotogramas clave contienen los estilos que tendrá el elemento en determinados momentos.

**8.1. La regla @keyframes**

Al especificar estilos CSS dentro de la regla **@keyframes**, la animación cambiará progresivamente del estilo actual al nuevo en determinados momentos.

Para que una animación funcione, se debe vincular la animación a un elemento.

**8.1. La regla @keyframes**

El siguiente ejemplo vincula la animación "example" al elemento <div>. La animación durará 4 segundos y gradualmente cambiará su color de fondo de rojo a amarillo:

<**h1**>Animación CSS</**h1**>

<**div**></**div**>

<**p**><**b**>Nota:</**b**> Cuando una animación concluye, vuelve a su estilo inicial.</**p**>

/\* The animation code \*/

@keyframes example {

from { background-color: red; } to { background-color: yellow; } }

/\* The element to apply the animation to \*/ div {

width: 100px;

height: 100px;

background-color: red;

animation-name: example;

animation-duration: 4s;

}

**8.1. La regla @keyframes**

La propiedad **animation-duration** define cuánto tiempo debe tardar una animación en completarse. Si no se especifica la propiedad de duración, no se producirá ninguna animación, porque el valor predeterminado es 0 (segundos).

En el ejemplo anterior, hemos especificado cuándo cambiará el estilo utilizando las palabras clave "**from**" y "**to**" (que representa 0% (inicio) y 100% (final)).

**8.1. La regla @keyframes**

También es posible utilizar el porcentaje. Al usar el porcentaje, puede agregar tantos cambios de estilo como desee.

<**h1**>Animación CSS</**h1**>

<**div**></**div**>

<**p**><**b**>Nota:</**b**> Cuando una animación concluye, vuelve a su estilo inicial.</**p**>

div {

width: 100px;

height: 100px;

background-color: red;

animation-name: example;

animation-duration: 4s;

}

@keyframes example {

0% { background-color: red; } 25% { background-color: yellow; } 50% { background-color: blue; } 100% { background-color: green; } }

**8.1. La regla @keyframes**

El siguiente ejemplo cambiará tanto el color de fondo como la posición del elemento <div> cuando la animación esté completa al 25%, al 50% y nuevamente cuando la animación esté completa al 100%:

/\* The animation code \*/

@keyframes example {

0% { background-color: red; left:0px; top:0px; }

25% { background-color: yellow; left:200px; top:0px; }

50% { background-color: blue; left:200px; top:200px; }

75% { background-color: green; left:0px; top:200px; }

100% { background-color: red; left:0px; top:0px; }

}

/\* ... \*/

**9. Flexbox, modelo de caja flexible**

**9. Flexbox, modelo de caja flexible**

El modelo de **caja flexible** o ***flexbox*** nos ofrece un mecanismo con el que dividir el espacio disponible en la ventana, y crear las filas y columnas necesarias para la implementación del diseño de una página web.

Este modelo flexible no necesita utilizar los métodos tradicionales de posicionamiento de cajas (**static**, **relative**, **absolute**, **float**...) sino que organiza los elementos utilizando contenedores flexibles.

**9.1. display: flex**

Para utilizar las propiedades de flexbox es necesario crear un **contenedor padre** que configure las características de los elementos hijos.

A este contenedor padre debemos declararle la propiedad y valor **display: flex;** Ejemplo:

#container {

display: flex;

}

**9.1. display: flex**

<**h1**>Display: flex</**h1**>

Ejemplo:

<**div** id="container">

<**div** class="box">1</**div**>

<**div** class="box">2</**div**>

<**div** class="box">3</**div**>

<**div** class="box">4</**div**>

<**div** class="box">5</**div**>

</**div**>

#container { display: flex; }

.box {

width: 100%; height: 150px; ...

}

**9.2. flex-wrap: nowrap | wrap | wrap-reverse**

Podemos indicar qué elementos flexibles se deben trasladar cuando no hay suficiente espacio en el contenedor mediante la propiedad **flex-wrap**. Sus valores son los siguientes:

● **nowrap**: los elementos no pasan a la siguiente fila y se reduce su anchura para mostrarlos. Este es el **valor por defecto.**

● **wrap**: los elementos pasan a la **siguiente fila** y conservan su anchura.

● **wrap-reverse**: los elementos pasan a la siguiente fila, pero en **sentido inverso** al de su declaración.

Ejemplo:

#container {

display: flex;

flex-wrap: wrap;

}

**9.3. justify-content**

Define la justificación horizontal de los elementos hijos de un contenedor flexible. Acepta los siguiente valores:

● **flex-start**: justificado a la **izquierda** del contenedor. Valor **por defecto**. ● **flex-end**: justificado a la **derecha** del contenedor.

● **center**: **centra** los elementos en el contenedor.

● **space-between**: añade un **espacio idéntico entre** los elementos. El primer elemento está alineado a la izquierda del contenedor, y el último, a la derecha.

● **space-around**: añade espacio, de forma regular, **alrededor** de los elementos del contenedor.

**9.3. justify-content**

#container{

display: flex;

justify-content: flex-start;

}

**9.4. align-items**

La propiedad **align-items** actúa sobre el eje vertical. Para utilizar esta propiedad primero se debe definir la propiedad y valor **flex-direction: row**.

Acepta los siguiente valores:

● **stretch**: los elementos se amplían verticalmente para ocupar toda la altura disponible en el contenedor. Este es el **valor por defecto**.

● **flex-start**: los elementos se colocan en la parte superior del contenedor. ● **flex-end**: los elementos se colocan en la parte inferior del contenedor. ● **center**: los elementos se colocan en el centro del contenedor.

**9.4. align-items**

#container{

display: flex;

}

#flex-start {

align-items: flex-start;

flex-direction: row;

}

**Ejercicio**

Haciendo uso de flex y de las propiedades anteriores, intenta replicar un bloque de información como el siguiente:

**10. Centrado vertical y horizontal**

**10. Centrado vertical y horizontal**

Existen diversas formas de **centrar horizontal y verticalmente** un elemento en un contenedor a través de CSS. Cada método puede ser adecuado en una situación concreta. Por ello es necesario conocer cómo funciona cada método para distinguir qué técnica utilizar en cada momento.

**10.1. Centrar elementos con flexbox**

Este método funciona con todos los elementos posicionados dentro de un contenedor. Por tanto, es válido para imágenes, textos, vídeos, etc.

Para emplear este método, el contenedor debe definir su propiedad **display: flex**. Para centrar horizontal y verticalmente el contenido utilizaremos las propiedades **justify-content: center** y **align-items: center**.

.container {

display: flex;

justify-content: center;

align-items: center;

}

**10.2. Centrar texto con line-height y text-align**

Mediante las propiedades **line-height** y **text-align** podemos alinear textos. Este método sólo va a ser válido para el centrado de párrafos, encabezados, etc.

div {

height: 200px;

text-align: center;

background-color: #DCDCDC;

}

div h1 {

line-height: 200px;

}

**10.3. Centrar imágenes y contenedores con transform: translate**

Este método puede alinear **imágenes y contenedores** vertical y horizontalmente. **No es válido para la alineación de textos** ya que no tiene en cuenta el tamaño de la letra.

.container {

height: 300px;

border: 1px solid black;

position: relative;

}

.container img {

position: absolute;

left: 50%;

top: 50%;

transform: translate(-50%, -50%);

-webkit-transform: translate(-50%, -50%);

}

**10.4. Alinear texto verticalmente a una imagen con vertical-align**

Con esta técnica podemos **alinear un texto a una imagen de forma vertical**. Esta propiedad admite los valores **top**, **middle** y **bottom**, entre otros valores.

img {

vertical-align: middle;

}

**10.5. Alinear verticalmente un checkbox con un label**

Si el tamaño de texto no coincide con el tamaño de un **checkbox**, estos elementos no se alinean verticalmente. Para conseguirlo, utilizaremos **vertical-align: middle** y definiremos el mismo tamaño de fuente en ambos elementos.

label, input{

font-size: 28px;

vertical-align: middle;

}

**11. Preprocesadores CSS**

**11. Preprocesadores CSS**

La sencillez sintáctica de CSS hace que el lenguaje sea algo limitado. Cambiar el valor de una propiedad y que sea igual en muchos documentos es trabajoso. Además, no podemos usar **condicionales** o **estructuras anidadas**. Para facilitar estas y otras muchas tareas, tenemos los **preprocesadores CSS**.

**11. Preprocesadores CSS**

Los preprocesadores CSS permiten desarrollar **estilos más mantenibles** con **sintaxis más ricas**. Los ficheros de los preprocesadores son parecidos a los de CSS y **deben compilarse** para traducirse a hojas de estilo reconocibles por los navegadores.

**11. Preprocesadores CSS**

Existen varios preprocesadores como *Sass*, *Less, Stylus* o *PostCSS*. Todos ellos son muy válidos y ofrecen características similares. No obstante, el procesador más extendido en la actualidad es *Sass*.

**11.1. Extensiones archivos**

En función del tipo de preprocesador que utilicemos, sus archivos fuente tendrán las siguientes extensiones:

● Sass: **.scss** o **.sass**

● Less: **.less**

● Stylus: **.styl**

**11.2. Ejemplos en Sass**

**11.2.1. Variables**

**Sintaxis .scss Sintaxis .sass**

$font-stack: Helvetica, sans-serif; $primary-color: #666;

body {

font-family: $font-stack;

color: $primary-color;

}

body {

font-family: Helvetica, sans-serif; color: #666;

}

$font-stack: Helvetica, sans-serif $primary-color: #666

body

font-family: $font-stack

color: $primary-color

**CSS compilado**

**11.2.2. Estructuras anidadas**

**Sintaxis .scss Sintaxis .sass CSS compilado**

nav {

ul {

margin: 0;

padding: 0;

list-style: none; }

li {

display: inline-block; }

}

nav

ul

margin: 0

padding: 0

list-style: none li

display: inline-block

nav ul {

margin: 0;

padding: 0;

list-style: none; }

nav li {

display: inline-block; }

**11.2.3. Operadores**

**Sintaxis .scss Sintaxis .sass CSS compilado**

$size: 400px;

article {

float: left;

width: $size / 960px \* 100%; height: $size;

border: 1px solid red; }

aside {

float: right;

width: $size / 960px \* 100%; height: $size;

border: 1px solid blue; }

**11.2.4. Mixins**

$size: 400px

article

float: left

width: $size / 960px \* 100% height: $size

border: 1px solid red

aside

float: right

width: $size / 960px \* 100% height: $size

border: 1px solid blue

article {

float: left;

width: 41.6666666667%; height: 400px;

border: 1px solid red; }

aside {

float: right;

width: 41.6666666667%; height: 400px;

border: 1px solid blue; }

**Sintaxis .scss Sintaxis .sass CSS compilado**

@mixin transform($property) { -webkit-transform: $property; -ms-transform: $property; transform: $property;

}

.box {

@include transform(rotate(30deg)); width: 200px;

height: 100px;

background-color: red;

}

@mixin transform($property) -webkit-transform: $property -ms-transform: $property

transform: $property

.box

@include transform(rotate(30deg)) width: 200px

height: 100px

background-color: red

.box {

-webkit-transform: rotate(30deg); -ms-transform: rotate(30deg); transform: rotate(30deg); width: 200px;

height: 100px;

background-color: red;

}

**11.3. Continúa aprendiendo**

Conoce más sobre los preprocesadores *Sass* y *Less* en los siguientes enlaces:

● Aprende Sass

● Aprende Less