**Ampliación de uso de estilos (UT2)**

|  |  |
| --- | --- |
| **Contadores** | Los contadores son variables en CSS que nos pueden ayudar a la hora de organizar los contenidos de la web y mostrar listas personalizadas con gran detalle a la hora de enumerar sus contenidos.  Tenemos varias propiedades para utilizar este tipo de variables:   * counter-reset – Crea o reinicia el valor de un contador. * counter-increment – Incrementa el valor de un contador. * content – Inserta un contenido personalizado por el desarrollador. * counter() – Añade el valor de un contador a un elemento.   Ejemplo: en este caso vamos a crear una variable consola para la página principal en el body. Cada vez que aparezca un elemento h2 incrementaremos el contador y además mostraremos un contenido delante del elemento.  <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=2">  <title>Web de pruebas</title>  <style>  body {  counter-reset: consola;  }  h2::before {  counter-increment: consola;  content: "Consola de videojuegos " counter(consola) ": ";  }  </style>  </head>  <body>  <h1>Contadores en CSS:</h1>  <h2>Xbox One X</h2>  <h2>Nintendo Switch</h2>  <h2>PS4 Pro</h2>  </body>  </html>  Otro ejemplo: creamos un contador para el ámbito de la página y llevar el control de las consolas, y además creamos otro contador para el ámbito de los h1 y llevar el control de los videojuegos asociados a esas consolas. En cada h1 el contador se pone a 0.  <!DOCTYPE html>  <html>  <head>  <style>  body {  counter-reset: consola;  }  h1 {  counter-reset: videojuegos;  }  h1::before {  counter-increment: consola;  content: "Consola "counter(consola) ". ";  }  h2::before {  counter-increment: videojuegos;  content: counter(consola) "."counter(videojuegos) " ";  }  </style>  </head>  <body>  <h1>Xbox One X</h1>  <h2>Halo Infinite</h2>  <h2>Gears 5</h2>  <h1>Nintendo Switch</h1>  <h2>The Legend Of Zelda: Breath Of The Wild</h2>  <h2>Super Mario Odyssey</h2>  <h1>PS4 Pro</h1>  <h2>God Of War</h2>  <h2>The Last Of Us 2</h2>  </body>  </html> |
| **Prioridades a la hora de aplicar reglas** | En ocasiones pueden existir conflictos entre diferentes reglas. Por ejemplo, podemos encontrarnos una clase que establezca un tamaño de fuente y además tener un tamaño de fuente específico para todo el texto del body. En estos casos CSS establece un orden de prioridades a la hora de resolver las reglas.  Ver las imágenes de origen |
| **Flexbox** | Flexbox es una propiedad esencial a la hora de maquetar o crear layouts en las páginas web. Permite evitar el uso de position o de float, que puede ser muy duro de implementar para el desarrollador. Esta propiedad nace de la necesidad de facilitar el maquetado, sobre todo a nivel horizontal, de los elementos de una página web.  Existen diferentes maneras de aplicar flexbox:   * Block, para las secciones de una página web. * Inline, para el texto. * Table, para tablas de dos dimensiones (filas y columnas). * Positioned, para especificar exactamente dónde queremos que aparezca un elemento.   Lo primero que necesitamos es definir un contenedor de tipo flex – Utilizará la propiedad display:flex  Ejemplo:  <!DOCTYPE html>  <html>  <head>  <style>  .flexcontenedor {  display: flex;  background-color: DodgerBlue;  }  .flexcontenedor > div {/\*Selector hijo. Hijos inmediatos.\*/  background-color: #f1f1f1;  margin: 10px;  padding: 20px;  font-size: 30px;  }  </style>  </head>  <body>  <div class="flexcontenedor">  <div>Elemento 1</div>  <div>Elemento 2</div>  <div>Elemento 3</div>  </div>  </body>  </html>  **Importante:**  En Sass .flexcontenedor {  > div {  }  }  **Propiedades fundamentales:**  Flex-direction: indicará la dirección en la que se apilarán los elementos. Tenemos varias opciones:  Column: apila los elementos verticalmente y en orden natural.  Ejemplo:  <!DOCTYPE html>  <html>  <head>  <style>  .flexcontenedor {  display: flex;  background-color: DodgerBlue;  flex-direction: column;  }  .flexcontenedor > div {/\*Selector hijo. Hijos inmediatos.\*/  background-color: #f1f1f1;  margin: 10px;  padding: 20px;  font-size: 30px;  }  </style>  </head>  <body>  <div class="flexcontenedor">  <div>Elemento 1</div>  <div>Elemento 2</div>  <div>Elemento 3</div>  </div>  </body>  </html>  Column-reverse: apila los elementos verticalmente y en sentido inverso.  <!DOCTYPE html>  <html>  <head>  <style>  .flexcontenedor {  display: flex;  background-color: DodgerBlue;  flex-direction: column-reverse;  }  .flexcontenedor > div {/\*Selector hijo. Hijos inmediatos.\*/  background-color: #f1f1f1;  margin: 10px;  padding: 20px;  font-size: 30px;  }  </style>  </head>  <body>  <div class="flexcontenedor">  <div>Elemento 1</div>  <div>Elemento 2</div>  <div>Elemento 3</div>  </div>  </body>  </html>  Row: apila los elementos horizontalmente de izquierda a derecha.  <!DOCTYPE html>  <html>  <head>  <style>  .flexcontenedor {  display: flex;  background-color: DodgerBlue;  flex-direction: row;  }  .flexcontenedor > div {/\*Selector hijo. Hijos inmediatos.\*/  background-color: #f1f1f1;  margin: 10px;  padding: 20px;  font-size: 30px;  }  </style>  </head>  <body>  <div class="flexcontenedor">  <div>Elemento 1</div>  <div>Elemento 2</div>  <div>Elemento 3</div>  </div>  </body>  </html>  Row-reverse: apila los elementos horizontalmente de derecha a izquierda.  <!DOCTYPE html>  <html>  <head>  <style>  .flexcontenedor {  display: flex;  background-color: DodgerBlue;  flex-direction: row-reverse;  }  .flexcontenedor > div {/\*Selector hijo. Hijos inmediatos.\*/  background-color: #f1f1f1;  margin: 10px;  padding: 20px;  font-size: 30px;  }  </style>  </head>  <body>  <div class="flexcontenedor">  <div>Elemento 1</div>  <div>Elemento 2</div>  <div>Elemento 3</div>  </div>  </body>  </html>  Flex-wrap: permite el ajuste automático cuando hay elementos que no quepan en el ancho del contenedor en el que se encuentren.  Wrap: los elementos pasan de línea si no caben  <!DOCTYPE html>  <html>  <head>  <style>  .contenedorflex {  display: flex;  flex-wrap: wrap;  background-color: DodgerBlue;  }  .contenedorflex > div {  background-color: #f1f1f1;  width: 100px;  margin: 10px;  text-align: center;  line-height: 75px;  font-size: 30px;  }  </style>  </head>  <body>  <div class="contenedorflex">  <div>1</div>  <div>2</div>  <div>3</div>  <div>4</div>  <div>5</div>  <div>6</div>  <div>7</div>  <div>8</div>  <div>9</div>  <div>10</div>  <div>11</div>  <div>12</div>  </div>  </body>  </html>  Nowrap: los elementos cambian de ancho automáticamente para adaptarse al ancho del contenedor. Es la opción por defecto.  Wrap-revese: aplica exactamente lo mismo que wrap, pero en sentido inverso.  Flex-flow: es una forma de reducir código al utilizar las propiedades flex-direction y flex-wrap.  Ejemplo:  Flex-flow: row wrap;  Justify-content: permite alinear horizontalmente los elementos que están dentro del contendor.  Center: alinea los elementos en el centro.  Ejemplo:  <!DOCTYPE html>  <html>  <head>  <style>  .contenedorflex {  display: flex;  flex-wrap: wrap;  justify-content: center;  background-color: DodgerBlue;  }  .contenedorflex > div {  background-color: #f1f1f1;  width: 100px;  margin: 10px;  text-align: center;  line-height: 75px;  font-size: 30px;  }  </style>  </head>  <body>  <div class="contenedorflex">  <div>1</div>  <div>2</div>  <div>3</div>    </div>  </body>  </html>  Flex-start – Posiciona los ítems al inicio del contenedor  .contenedorflex {  display: flex;  justify-content: flex-start;  }  Flex-end – Posiciona los ítems al final del contendor  . contenedorflex {  display: flex;  justify-content: flex-end;  }  Space-around – Deja espacio antes, después y entre medias de los ítems. Lo adapta al tamaño de línea (Probar ejemplo con muchos ítems).  . contenedorflex {  display: flex;  justify-content: space-around;  }  Space-between – Deja espacio entre medias de los ítems. Lo adapta al tamaño de línea (Probar ejemplo con muchos ítems).  Align-items: es una propiedad que permite alinear verticalmente los elementos de un contenedor. Debemos poner una altura al contenedor.  Para centrar los elementos a nivel vertical:  .contenendorflex {  display: flex;  align-items center;  }  Para alinear los elementos arriba del contenedor:  .contenedorflex {  display: flex;  height: 200px;  align-items: flex-start;  }  Para alinear los elementos abajo del contenedor:  .contenedorflex {  display: flex;  height: 200px;  align-items: flex-end;  }  Para rellenar automáticamente y adaptarse al tamaño del contenedor:  .contenedorflex {  display: flex;  height: 200px;  align-items: stretch;  }  Teniendo en cuenta su línea base de texto:  <html>  <head>  <style>  .contenedorflex {  display: flex;  height: 200px;  align-items: baseline;  background-color: DodgerBlue;  }  .contenedorflex > div {  background-color: #f1f1f1;  width: 100px;  margin: 10px;  text-align: center;  line-height: 75px;  font-size: 30px;  }  </style>  </head>  <body>  <div class="contenedorflex">  <div><h1>1</h1></div>  <div><h6>2</h6></div>  <div><h3>3</h3></div>  <div><small>4</small></div>  </div>  </body>  Align-content: esta propiedad se utiliza para alinear las líneas flex. Normalmente se suele consideran que un elemento flex es de una sola línea, pero puede haber excepciones. Cada línea se toma como un nuevo contenedor. Para crear rejillas multidimensionales utilizamos Grid, es decir, para layouts verticales.  El espacio sobrante se gestiona en relación al valor de la propiedad, siempre que haya más de una línea. Por ejemplo, con el valor flex-start, el elemento se situará pegado arriba y el espacio en blanco abajo, siempre que el alto sea mayor.  Se aplica a nivel vertical.  Ejemplo – Espacios iguales entre líneas  <!DOCTYPE html>  <html>  <head>  <style>  .contenedorflex {  display: flex;  height: 600px;  flex-wrap: wrap;  align-content: space-between;  background-color: DodgerBlue;  }  .contenedorflex > div {  background-color: #f1f1f1;  width: 100px;  margin: 10px;  text-align: center;  line-height: 75px;  font-size: 30px;  }  </style>  </head>  <body>  <div class="contenedorflex ">  <div>1</div>  <div>2</div>  <div>3</div>  <div>4</div>  <div>5</div>  <div>6</div>  <div>7</div>  <div>8</div>  <div>9</div>  <div>10</div>  <div>11</div>  <div>12</div>  </div>  </body>  </html>  Espacios antes, después y entre líneas:  .contenedorflex {  display: flex;  height: 600px;  flex-wrap: wrap;  align-content: space-around;  }  Las líneas flex se adaptan y ocupan el espacio restante:  .contenedorflex {  display: flex;  height: 600px;  flex-wrap: wrap;  align-content: stretch;  }  Las líneas flex se visualizan en medio del contenedor.  .contenedorflex {  display: flex;  height: 600px;  flex-wrap: wrap;  align-content: center;  }  Las líneas flex se visualizan al principio del contenedor:  .cotenendorflex {  display: flex;  height: 600px;  flex-wrap: wrap;  align-content: flex-start;  }  Las líneas flex se visualizan al final del contenedor:  .contenedorflex {  display: flex;  height: 600px;  flex-wrap: wrap;  align-content: flex-end;  }  **Centrado exacto**  Para centrar de manera perfecta un elemento tanto a nivel horizontal como a nivel vertical, lo ideal es combinar las propiedades justify-content y align-items  <!DOCTYPE html>  <html>  <head>  <style>  .contenedorflex{  display: flex;  justify-content: center;  align-items: center;  height: 300px;  background-color: DodgerBlue;  }  .contenendorflex>div {  background-color: #f1f1f1;  color: white;  width: 100px;  height: 100px;  }  </style>  </head>  <body>  <div class="contenedorflex">  <div></div>  </div>  </body>  </html>  Elementos hijos: los hijos directos de un contenedor flex automáticamente se convierten en elementos flex.  Tienen estas propiedades:  *Order*: determina el orden de los elementos dentro de un contenedor flex.  Ejemplo:  <div class="contenedorflex">  <div style="order: 3">1</div>  <div style="order: 2">2</div>  <div style="order: 4">3</div>  <div style="order: 1">4</div>  </div>  *Flex-grow*: especifica cuánto va a crecer un elemento flex con respecto al resto de elementos flex. Se identificará con un valor numérico.  Ejemplo: el tercer elemento flex crecerá 8 veces más rápido que el resto de los elementos.  <div class="contenedorflex">  <div style="flex-grow: 1">1</div>  <div style="flex-grow: 1">2</div>  <div style="flex-grow: 8">3</div>  </div>  *Flex-shrink*: es una propiedad que determina cuánto puede encogerse un elemento flex respecto al resto. Esta propiedad se identificará con un valor numérico.  Ejemplo: el tercer elemento no encogerá nada respecto al resto de elementos.  <div class="contenedorflex">  <div>1</div>  <div>2</div>  <div style="flex-shrink: 0">3</div>  <div>4</div>  <div>5</div>  <div>6</div>  <div>7</div>  <div>8</div>  <div>9</div>  <div>10</div>  </div>  *Flex-basis*: especifica la longitud inicial de un elemento flex hijo.  Ejemplo: el elemento 3 tendrá una longitud inicial de 250px.  <div class="contenedorflex">  <div>1</div>  <div>2</div>  <div style="flex-basis: 250px">3</div>  <div>4</div>  </div>  *Ahorrando código*: la propiedad flex puede aunar las propiedades flex-grow, flex-shrink y flex-basis.  Ejemplo: el elemento 3 no crecerá, no encogerá y tendrá una longitud inicial de 300px.  <div class="contenedorflex">  <div>1</div>  <div>2</div>  <div style="flex: 0 0 200px">3</div>  <div>4</div>  </div>  Align-self: especifica la alineación vertical de un elemento concreto dentro del contenedor flex. Esta propiedad sobrescribe a la propiedad align-items.  Ejemplo: el tercer elemento tendrá alineación vertical centrada.  <div class="cotenedorflex">  <div>1</div>  <div>2</div>  <div style="align-self: center">3</div>  <div>4</div>  </div>  Ejemplo:  <div class="contenedorflex">  <div>1</div>  <div style="align-self: flex-start">2</div>  <div style="align-self: flex-end">3</div>  <div>4</div>  </div>  Veremos ejemplos completos con Flexbox y Grid cuando estudiemos Media Queries. |

|  |  |
| --- | --- |
| **Media Queries** | Las Media Queries son reglas que se utilizan para comprobar parámetros del dispositivo en el que se está abriendo la página web y así poder adaptar la visualización para que sea óptima.  Se pueden utilizar diferentes tipos de Media Queries dependiendo del dispositivo al que vaya dirigida la página web:   * All -> Para todos los tipos de dispositivos. * Print -> Para impresoras. * Screen -> Para pantallas, tablets, teléfonos… * Speech -> Orientado a lectores de pantalla.   Podemos tener una serie de condiciones enlazadas con los **operadores and, not, only y or**.  Ejemplo:  @media (max-width: 600px) and (orientation: landscape) {  h1{  color: red;  }  }  **Ejemplos de Media Queries:**  Ejemplo 1: Cuando el ancho se mayor que 480px el color de fondo será verde.  @media screen and (min-width: 480px) {  body {  background-color: lightgreen;  }  }  Ejemplo 2: menú que se sitúa en la parte superior si la anchura es inferior a 480px.  <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <style>  .contenedor {  overflow: auto;  }  #principal {  margin-left: 4px;  }  #barralateral {  float: none;  width: auto;  }  #listamenu {  margin: 0;  padding: 0;  }  .elementomenu {  background: #CDF0F6;  border: 1px solid #d4d4d4;  border-radius: 4px;  list-style-type: none;  margin: 4px;  padding: 2px;  }  @media screen and (min-width: 480px) {  #barralateral {  width: 200px;  float: left;  }  #principal {  margin-left: 216px;  }  }  </style>  </head>  <body>  <div class="contenedor">  <div id="barralateral">  <ul id="listamenu">  <li class="elementomenu">Elemento menú 1</li>  <li class="elementomenu">Elemento menú 2</li>  <li class="elementomenu">Elemento menú 3</li>  <li class="elementomenu">Elemento menú 4</li>  <li class="elementomenu">Elemento menú 5</li>  </ul>  </div>  <div id="principal">  <h1>Hay que redimensionar la ventana para observar el efecto</h1>  <p>El menú flotará a la izquierda si el ancho del viewport es superior a 480px. Si la anchura es inferior a 480px,  el menú se encontrará en la parte superior.  </div>  </div>  </body>  </html>  Existen varios atributos importantes dentro de las Media Queries:   * Max-width: el ancho máximo de la zona que se está visualizando, normalmente la ventana del navegador. * Min-width: el ancho mínimo de la zona que se está visualizando, normalmente la ventana del navegador. * Min-height: el alto mínimo de la zona que se está visualizando, normalmente la ventana del navegador. * Max-height: el alto máximo de la zona que se está visualizando, normalmente la ventana del navegador. * Orientation: la orientación del navegador (landscape o portrait). Landscape se da cuando el ancho es mayor que el alto.   Más ejemplos con Media Queries:  Ejemplo: menú responsive.  <!DOCTYPE html>  <html lang="es">  <head>  <meta charset="utf-8">  <meta name="viewport" content="width=device-width, initial-scale=1">  <style>  \* { /\*En todos los elementos vamos a aplicar la propiedad box-sizin, es decir, el padding y border se incluirán en el ancho y alto del elemento\*/  box-sizing: border-box;  }  /\* Estilo del menú \*/  .menusuperior {  overflow: hidden;  background-color: #333;  }  /\* Estilo de los elementos del menú \*/  .menusuperior a {  float: left;  display: block;  color: #f2f2f2;  text-align: center;  padding: 14px 16px;  text-decoration: none;  }  /\* Efecto al situar el ratón encima de los enlaces \*/  .menusuperior a:hover {  background-color: #ddd;  color: black;  }  /\* En pantallas inferiores a 600px, el menú se visualizará verticalmente y sus elementos ocuparán el 100% del ancho \*/  @media screen and (max-width: 600px) {  .menusuperior a {  float: none;  width: 100%;  }  }  </style>  </head>  <body>  <h2>Menú responsive</h2>  <p>Si el ancho de la ventana del navegador es inferior a 600px, el menú se visualizará verticalmente.</p>  <div class="menusuperior">  <a href="#">Elemento 1</a>  <a href="#">Elemento 2</a>  <a href="#">Elemento 3</a>  </div>  </body>  </html>  Ejemplo: el texto se mostrará en verde cuando se visualice en la pantalla y negro cuando se imprima. Hay que recordar que los navegadores quitan los gráficos de fondo cuando vamos a imprimir una página web. Hay que elegir la opción adecuada del navegador para poder visualizarlo.  <!DOCTYPE html>  <html>  <head>  <style>  @media screen {  body {  color: green;  }  }  @media print {  body {  color: black;  }  }  </style>  </head>  <body>  <h1>Media Query que depende del dispositivo (pantalla o impresora).</h1>  <p>Texto verde cuando se visualiza por pantalla. Texto negro cuando se intenta imprimir.</p>  </body>  </html>  Ejemplo: en este caso la web se visualizará de manera diferente cuando la web se visualice en modo landscape (orientación horizontal) o portrait (orientación vertical).  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <style>  body {  background-color: lightgreen;  }  @media only screen and (orientation: landscape) {  body {  background-color: lightblue;  }  }  </style>  </head>  <body>  <p>Cuando el ancho de la ventana del navegador sea mayor que el alto de la ventana del navegador, el fondo se  visualizará en azul. Si el alto es mayor que el ancho, el fondo se visualizará en verde.</p>  </body>  </html>  Otro ejemplo con varias condiciones (Estos estilos para la clase "lateral" los visualizaríamos en anchuras de pantalla que cumplan las dos condiciones a la vez: que su anchura mínima sean 700 píxeles y las anchura máxima sea 800 píxeles. Es decir, el rango de 101 píxeles que van desde los 700 hasta los 800 píxeles).  Normalmente estos rangos de valores se colocan de menor a mayor. Es decir, si tenemos un rango de valores entre 0 y 600, colocaríamos esa media query antes que una que afecte al rango 601-800, por ejemplo.  @media (min-width: 700px) and (max-width: 800px){  .lateral{  width: 33%;  float: right;  background-color: #6ee;  }  }  Otro ejemplo con la condición screen añadida (como podemos ver, se pueden suceder una serie de condiciones:  @media screen and (min-width: 700px) and (max-width: 800px){  .lateral{  width: 33%;  float: right;  background-color: #6ee;  }  }  Veremos muchos más ejemplos con Flexbox y en el apartado de layouts. |
| **Layouts – Distribuciones de contenido de una página web** | Siguientes Unidades de Trabajo |
| **CSS Grid** | Siguientes Unidades de Trabajo |
| **Object fit** | Propiedad esencial para que los elementos se adapten adecuadamente a su contenedor.  Siguientes Unidades de Trabajo |