LEMONCODE

HTML

Introducción a Flexbox

Flexbox es una herramienta de CSS que nos ayuda a crear diseños adaptables y posicionar elementos de forma sencilla. Sus propiedades permiten colocar elementos incluso cuando las dimensiones son desconocidas o cambian dinámicamente.

Para utilizar Flexbox, necesitamos dos tipos de elementos: los contenedores y los ítems. Es importante saber esto porque algunas propiedades se aplican solo al contenedor, mientras que otras se definen para los ítems dentro del contenedor.

Los ítems se pueden distribuir en dos ejes: el principal y el transversal o secundario. Por defecto, el eje principal va de izquierda a derecha y el transversal de arriba a abajo. Sin embargo, existen propiedades que pueden cambiar esta orientación, especialmente en idiomas que se leen de derecha a izquierda o de arriba abajo.

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta chartset="UTF-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <link rel="stylesheet" href="style.css" />
        <title>Intro Flexbox</title>
    </head>
    <body>
        <div class="container">
            <div class="box box-1">1</div>
            <div class="box box-2">2</div>
            <div class="box box-3">3</div>
            <div class="box box-4">4</div>
            <div class="box box-5">5</div>
        </div>
    </body>
</html>
```

```css
/* style.css */

:root {
    --primary: #845ec2;
    --secondary: #dbe7ed;
    --accent: #39829b;
}
```

```
.container {
    background-color: var(--tertiary);
}

.box {
    background-color: var(--primary);
    color: var(--secondary);
    font-size: 24px;
    text-align: center;
    padding: 20px;
    margin: 20px;
    width: 50px;
}
```

Para empezar a utilizar Flexbox, debemos indicar el tipo de display que tendrá el contenedor. Utilizamos la propiedad `display: flex` para definir que el contenedor es de tipo flex y que sus elementos internos se moverán según las reglas de Flexbox. Los elementos se colocarán de forma horizontal por defecto.

```
/* style.css */

:root {
    --primary: #845ec2;
    --secondary: #dbe7ed;
    --accent: #39829b;
}

.container {
    display: flex;
    background-color: var(--tertiary);
}

.box {
    background-color: var(--primary);
    color: var(--secondary);
    font-size: 24px;
    text-align: center;
    padding: 20px;
    margin: 20px;
    width: 50px;
}
```

Podemos cambiar la dirección de los elementos utilizando la propiedad `flex-direction`. Por ejemplo, si establecemos `flex-direction: row-reverse`, los elementos se girarán en sentido inverso. También podemos establecer que los elementos se distribuyan en forma vertical utilizando `flex-direction: column` o su versión inversa con `flex-direction: column-reverse`.

```
/* style.css */
```

```
:root {
    --primary: #845ec2;
    --secondary: #dbe7ed;
    --accent: #39829b;
}

.container {
    display: flex;
    flex-direction: row-reverse;
    background-color: var(--tertiary);
}

.box {
    background-color: var(--primary);
    color: var(--secondary);
    font-size: 24px;
    text-align: center;
    padding: 20px;
    margin: 20px;
    width: 50px;
}
```

Si el espacio horizontal es insuficiente, los elementos se apilarán uno encima del otro. Podemos controlar este comportamiento utilizando la propiedad `flex-wrap`. Con `flex-wrap: nowrap`, los elementos se mantienen en una sola línea. Si queremos que los elementos salten a una nueva línea cuando no hay suficiente espacio, utilizamos `flex-wrap: wrap`. También existe la opción `flex-wrap: wrap-reverse`, que además de cambiar de línea, gira los elementos.

```
/* style.css */

:root {
    --primary: #845ec2;
    --secondary: #dbe7ed;
    --accent: #39829b;
}

.container {
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    background-color: var(--tertiary);
}

.box {
    background-color: var(--primary);
    color: var(--secondary);
    font-size: 24px;
    text-align: center;
    padding: 20px;
    margin: 20px;
```

```
    width: 50px;
}
```

En lugar de utilizar las propiedades por separado, podemos abreviarlas usando `flex-flow`, que combina `flex-direction` y `flex-wrap`. Por ejemplo, `flex-flow: row wrap` tiene el mismo efecto que `flex-direction: row` y `flex-wrap: wrap`.

```
/* style.css */

:root {
    --primary: #845ec2;
    --secondary: #dbe7ed;
    --accent: #39829b;
}

.container {
    display: flex;
    flex-flow: row wrap;
    background-color: var(--tertiary);
}

.box {
    background-color: var(--primary);
    color: var(--secondary);
    font-size: 24px;
    text-align: center;
    padding: 20px;
    margin: 20px;
    width: 50px;
}
```

Para ordenar los elementos dentro del contenedor, utilizamos la propiedad `justify-content` en el eje principal. Algunos valores comunes son `flex-start`, que ordena los elementos desde el principio del eje, `flex-end`, que los ordena desde el final, y `center`, que los centra. También existen `space-between`, que coloca los elementos pegados a los bordes y distribuye el espacio restante de manera equitativa, y `space-around`, que distribuye los elementos de forma equitativa en el espacio disponible.

```
/* style.css */

:root {
    --primary: #845ec2;
    --secondary: #dbe7ed;
    --accent: #39829b;
}

.container {
    display: flex;
    flex-flow: row wrap;
```

```
    justify-content: space-around;
    background-color: var(--tertiary);
}

.box {
    background-color: var(--primary);
    color: var(--secondary);
    font-size: 24px;
    text-align: center;
    padding: 20px;
    margin: 20px;
    width: 50px;
}
```

Para ordenar los elementos en el eje secundario, utilizamos la propiedad `align-items`. Por ejemplo, `align-items: flex-end` alinea los elementos en el extremo inferior del eje secundario, mientras que `align-items: flex-start` los alinea en el extremo superior. Con `align-items: center`, los elementos se centran verticalmente en el espacio disponible.

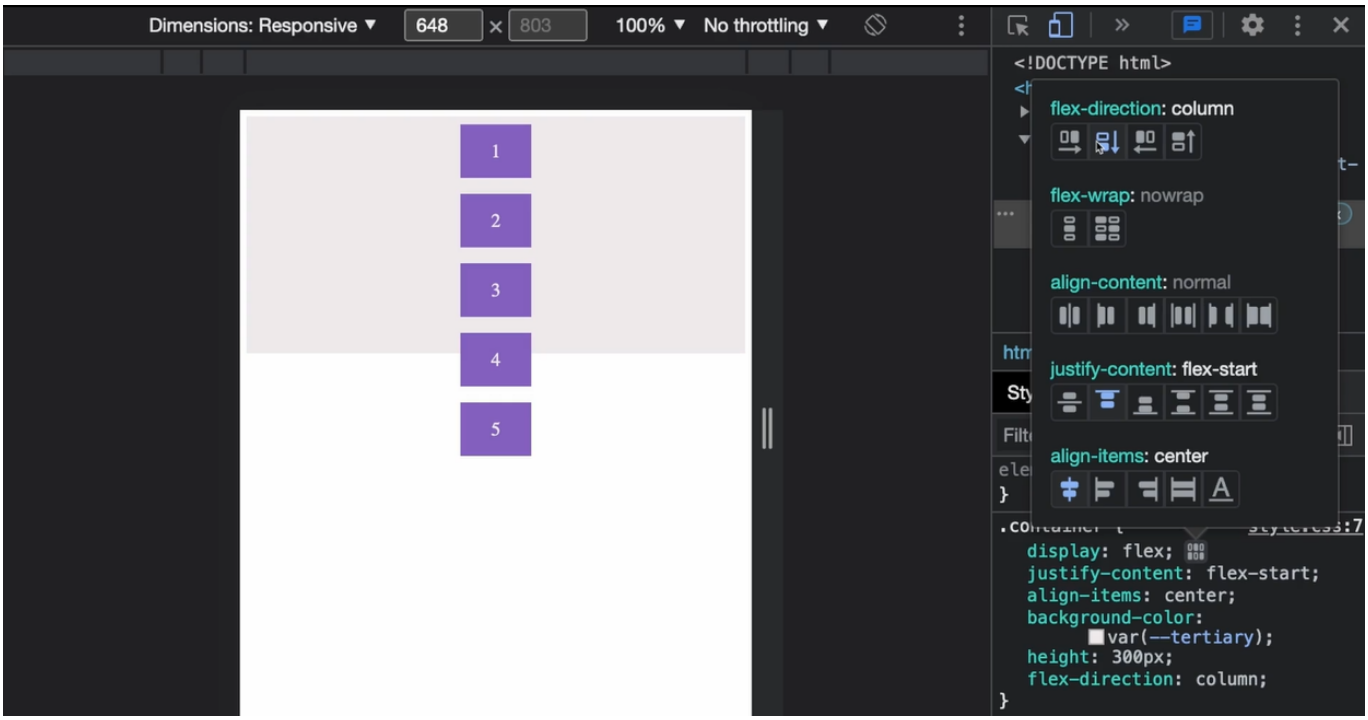
```
/* style.css */

:root {
    --primary: #845ec2;
    --secondary: #dbe7ed;
    --accent: #39829b;
}

.container {
    display: flex;
    justify-content: space-around;
    align-items: flex-end;
    background-color: var(--tertiary);
    height: 300px;
}

.box {
    background-color: var(--primary);
    color: var(--secondary);
    font-size: 24px;
    text-align: center;
    padding: 20px;
    margin: 20px;
    width: 50px;
    height: 50px;
}
```

Es importante tener en cuenta que `justify-content` se aplica al eje principal y `align-items` al eje secundario. Para ayudar a comprender y experimentar con estas propiedades, los navegadores suelen proporcionar herramientas visuales en sus "developer tools".



Ya hemos visto cómo posicionar los elementos utilizando Flexbox. Además, podemos determinar el orden en el que se ubican los elementos dentro del contenedor. Sin embargo, es importante tener en cuenta que si modificamos el orden de los elementos con Flexbox, estos cambios no se reflejarán en el DOM. Esto significa que los lectores de pantalla seguirán leyendo el orden real de los elementos en el DOM, no los cambios realizados con Flexbox. Esto es crucial para garantizar la accesibilidad.

La propiedad que nos permite controlar el orden de los elementos es `order`. Esta propiedad acepta un número como valor. Si un elemento no tiene definida la propiedad `order`, se ubicará al principio del eje, es decir, hacia la izquierda en el eje principal. Esto es independiente de la propiedad `order` de los demás elementos.

Veamos un ejemplo para entenderlo mejor. Supongamos que tenemos un contenedor con la clase "container" que tiene elementos con las clases "box1", "box2", "box3", "box4" y "box5". Si queremos modificar el orden del elemento 2, podemos agregar la propiedad `order` y asignarle un valor, por ejemplo, 5. El valor de `order` representa el peso que le damos a ese elemento. Cuanto mayor sea el valor, más a la derecha se ubicará en el eje principal. Por ejemplo, si le asignamos a la clase "box1" un `order` de 3, se posicionará a la izquierda del elemento 2. En cambio, si a la clase "box3" le asignamos un `order` de 4, se posicionará entre el 1 y el 2.

Es importante mencionar que si hay elementos que no tienen definida la propiedad `order`, se ubicarán al principio del eje, ya que les estamos indicando que no tienen suficiente peso para desplazarse hacia la derecha.

```
/* style.css */

:root {
    --primary: #845ec2;
    --secondary: #dbe7ed;
    --accent: #39829b;
}

.container {
```

```
    display: flex;
    justify-content: flex-start;
    align-items: center;
    background-color: var(--tertiary);
}

.box {
    background-color: var(--primary);
    color: var(--secondary);
    font-size: 24px;
    text-align: center;
    padding: 20px;
    margin: 20px;
    width: 50px;
}

.box1 {
    order: 3;
}

.box2 {
    order: 9;
}

.box3 {
    order: 4;
}

.box4 {
}

.box5 {
}
```

Para explorar todas las propiedades de Flexbox y experimentar con ellas, puedes utilizar herramientas online como "Flexbox Playground" en Codepen. También te recomiendo un divertido juego llamado "Flexbox Froggy", donde puedes practicar todas estas propiedades. El objetivo del juego es posicionar una rana sobre una hoja utilizando Flexbox.