¿Qué es Grid?

Grid o **CSS Grid Layout** es un sistema de diseño en CSS que permite crear estructuras complejas de manera sencilla y eficiente. Con Grid, los desarrolladores web pueden organizar y distribuir los elementos dentro de un contenedor en un diseño de cuadrícula (grid) bidimensional, es decir, tanto en filas como en columnas.

Características principales de CSS Grid

1. **Estructura Bidimensional**: CSS Grid permite controlar tanto las filas (horizontal) como las columnas (vertical) en un contenedor, lo que facilita la creación de diseños complejos. A diferencia de Flexbox, que es un sistema de diseño unidimensional (solo filas o solo columnas), Grid trabaja en dos dimensiones.
2. **Contenedores y Elementos**:
    - Un contenedor Grid se define mediante la propiedad `display: grid;`.
    - Los elementos dentro del contenedor pueden ser asignados a celdas específicas utilizando las propiedades `grid-column` y `grid-row`.
3. **Tamaño de las celdas**:
    - Las celdas de la cuadrícula pueden tener tamaños fijos o automáticos. Usamos propiedades como `grid-template-columns` y `grid-template-rows` para definir el tamaño de las filas y columnas.
    - También es posible usar unidades flexibles, como fracciones (`fr`), para distribuir el espacio de manera equitativa o según necesidades específicas.
4. **Áreas de la cuadrícula**:
    - CSS Grid permite crear áreas dentro de la cuadrícula utilizando `grid-template-areas`. Esto ayuda a definir nombres de áreas (como un diseño de encabezado, barra lateral, contenido principal, etc.) y luego asignar los elementos a esas áreas de manera más visual.
5. **Alineación**:
    - Ofrece un control preciso sobre la alineación de los elementos dentro de las celdas de la cuadrícula, tanto horizontal como verticalmente, con propiedades como `align-items`, `justify-items`, `align-self` y `justify-self`.
6. **Responsive Design**:
    - CSS Grid se adapta perfectamente a diseños responsivos. Es posible cambiar el número de filas y columnas o sus tamaños según el tamaño de la pantalla mediante media queries, lo que facilita la creación de diseños que se ajustan a diferentes dispositivos.

Ejemplos

- **`display: grid;`**: Define un contenedor como un "grid" y activa las propiedades de la cuadrícula en sus elementos hijos.

- **`grid-template-columns`** y **`grid-template-rows`**: Definen cuántas columnas y filas tendrá la cuadrícula, así como su tamaño. Por ejemplo:

```
css
Copiar
grid-template-columns: 1fr 2fr 1fr; /* Tres columnas con tamaños
relativos */
grid-template-rows: 100px 200px;   /* Dos filas con tamaños específicos
*/
```

- **`grid-column`** y **`grid-row`**: Permiten a los elementos hijo ocupar una o más columnas o filas. Ejemplo:

```
css
Copiar
grid-column: 1 / 3; /* El elemento ocupa desde la primera columna hasta
la tercera */
grid-row: 2 / 4;    /* El elemento ocupa desde la segunda fila hasta la
cuarta */
```

- **`grid-template-areas`**: Permite organizar el diseño mediante nombres de áreas. Ejemplo:

```
css
Copiar
grid-template-areas:
  "header header header"
  "sidebar content content"
  "footer footer footer";
```

- **`gap`**: Define el espacio entre las filas y columnas de la cuadrícula. Ejemplo:

```
css
Copiar
gap: 10px;
```

Ventajas de usar CSS Grid

1. **Mayor control sobre el diseño**: Permite controlar tanto las filas como las columnas de forma precisa, lo que lo convierte en una excelente opción para diseños complejos.
2. **Flexibilidad**: Grid se adapta a distintos tamaños de pantalla y dispositivos con facilidad, lo que facilita la creación de diseños responsivos.
3. **Simplicidad**: Aunque puede parecer complicado al principio, CSS Grid facilita la creación de estructuras que, de otra manera, requerirían muchos elementos flotantes o trucos con otras técnicas de diseño.
4. **Organización clara**: Gracias a su enfoque en áreas y celdas, los diseños pueden ser más fáciles de leer y mantener.

Validación de formularios

La validación de formularios se puede hacer tanto en el lado del cliente (con JavaScript) como en el servidor. La validación del lado del cliente ayuda a mejorar la experiencia del usuario, pero siempre debe complementarse con una validación en el servidor por razones de seguridad.

```html
<form id="miFormulario">
  <label for="nombre">Nombre:</label>
  <input type="text" id="nombre" name="nombre" required>

  <label for="email">Correo Electrónico:</label>
  <input type="email" id="email" name="email" required>

  <label for="edad">Edad:</label>
  <input type="number" id="edad" name="edad" min="18" required>

  <button type="submit">Enviar</button>
</form>
```

```javascript
document.getElementById('miFormulario').addEventListener('submit', function(event) {
  // Previene el envío del formulario si hay un error
  event.preventDefault();

  // Obtener los valores de los campos
  const nombre = document.getElementById('nombre').value;
  const email = document.getElementById('email').value;
  const edad = document.getElementById('edad').value;

  // Validación: Verificar que los campos no estén vacíos
  if (!nombre || !email || !edad) {
    document.getElementById('mensajeError').style.display = 'block';
    return; // Salir de la función si hay error
  }

  // Validar el formato del correo electrónico
  const emailRegex = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/;
  if (!emailRegex.test(email)) {
    alert('Por favor ingresa un correo electrónico válido');
    return;
  }

  // Validar la edad (mayor o igual a 18)
  if (edad < 18) {
    alert('Debes ser mayor de 18 años');
    return;
  }

  // Si todo es correcto, se puede enviar el formulario
  alert('Formulario enviado con éxito');
  document.getElementById('miFormulario').reset(); // Reiniciar el formulario
  document.getElementById('mensajeError').style.display = 'none';
});
```

Uso de localStorage

`localStorage` es una API de JavaScript que permite almacenar datos de manera persistente en el navegador del usuario. Estos datos no tienen fecha de expiración, lo que significa que permanecen en el navegador hasta que se borren explícitamente.

```
document.getElementById('miFormulario').addEventListener('submit', function(event) {
  event.preventDefault();

  const nombre = document.getElementById('nombre').value;
  const email = document.getElementById('email').value;
  const edad = document.getElementById('edad').value;

  if (!nombre || !email || !edad) {
    document.getElementById('mensajeError').style.display = 'block';
    return;
  }

  const emailRegex = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/;
  if (!emailRegex.test(email)) {
    alert('Por favor ingresa un correo electrónico válido');
    return;
  }

  if (edad < 18) {
    alert('Debes ser mayor de 18 años');
    return;
  }

  // Guardar los datos en localStorage
  localStorage.setItem('nombre', nombre);
  localStorage.setItem('email', email);
  localStorage.setItem('edad', edad);

  alert('Formulario enviado y datos guardados');

  document.getElementById('miFormulario').reset();
  document.getElementById('mensajeError').style.display = 'none';
});
```