Tài liệu: CSS Grid Layout

Mục tiêu

- Hiểu rõ **CSS Grid Layout**, cách hoạt động, và vai trò trong việc tạo bố cục web 2D.
- Nắm vững các thuộc tính chính, cách sử dụng chúng để xây dựng giao diện linh hoạt và responsive.
- Cung cấp ví dụ minh họa và bài tập thực hành để áp dụng CSS Grid vào các dự án thực tế.

Đối tượng

- Sinh viên hoặc lập trình viên đã biết CSS cơ bản (selector, box model, media queries).
- Người muốn tạo các bố cục phức tạp, responsive mà không cần sử dụng framework như Bootstrap.
- Lập trình viên tìm hiểu cách áp dụng CSS Grid trong các dự án hiện đại (2025).

---

1. Định nghĩa

CSS Grid Layout là một hệ thống bố cục 2D trong CSS, cho phép lập trình viên tạo các lưới (grid) với các hàng (rows) và cột (columns) linh hoạt. Nó được thiết kế để xây dựng các giao diện phức tạp, từ bố cục toàn trang đến các thành phần nhỏ như gallery ảnh hoặc dashboard, với khả năng kiểm soát chính xác vị trí và kích thước của các phần tử.

---

2. Cách hoạt động

Cơ chế

- **Grid Container**: Phần tử cha được đặt `display: grid` hoặc `display: inline-grid`, tạo ra một lưới để chứa các phần tử con (grid items).
- **Grid Lines**: Các đường ngang và dọc chia lưới thành các ô. Mỗi hàng/cột được đánh số bắt đầu từ 1.
- **Grid Tracks**: Hàng hoặc cột trong lưới (được định nghĩa qua `grid-template-columns` và `grid-template-rows`).
- **Grid Items**: Các phần tử con trực tiếp của container, được đặt trong các ô lưới bằng cách sử dụng các thuộc tính như `grid-column`, `grid-row`, hoặc `grid-area`.
- **Cơ chế trình duyệt**: Trình duyệt tính toán kích thước và vị trí của các ô lưới dựa trên các thuộc tính được định nghĩa. CSS Grid tự động điều chỉnh bố cục dựa trên nội dung hoặc kích thước container, đặc biệt khi kết hợp với các hàm như `minmax()`, `auto-fit`, hoặc `auto-fill`.

Các thuộc tính chính

- **Container**:
  - `display: grid | inline-grid`: Kích hoạt lưới.
  - `grid-template-columns`: Xác định số lượng và kích thước cột.
  - `grid-template-rows`: Xác định số lượng và kích thước hàng.

    - `grid-template-areas`: Định nghĩa vùng (areas) trong lưới bằng tên.
    - `gap` (hoặc `row-gap`, `column-gap`): Khoảng cách giữa các ô.
    - `justify-items/align-items`: Căn chỉnh các grid items theo trục ngang/dọc trong ô.
    - `justify-content/align-content`: Căn chỉnh toàn bộ lưới trong container.
    - `grid-auto-columns/grid-auto-rows`: Xác định kích thước mặc định cho các cột/hàng tự động.
- **Grid Items**:
    - `grid-column` (hoặc `grid-column-start/grid-column-end`): Xác định vị trí cột của item.
    - `grid-row` (hoặc `grid-row-start/grid-row-end`): Xác định vị trí hàng của item.
    - `grid-area`: Đặt tên vùng hoặc xác định vị trí cụ thể (shorthand cho `grid-row` và `grid-column`).
    - `justify-self/align-self`: Căn chỉnh riêng lẻ cho từng item.

Lưu ý

- **Responsive**: Kết hợp với media queries hoặc các hàm như `minmax()`, `auto-fit` để tạo bố cục thích ứng.
- **Hiệu suất**: CSS Grid nhẹ và được tối ưu hóa bởi trình duyệt, phù hợp cho cả bố cục đơn giản và phức tạp.
- **Tương thích**: Hỗ trợ tốt trên các trình duyệt hiện đại (Chrome, Firefox, Edge, Safari từ 2017 trở đi). Kiểm tra trên **CanIUse.com** nếu cần hỗ trợ trình duyệt cũ.

---

3. Cú pháp

Container

```
.grid-container {
  display: grid;
  grid-template-columns: 100px 1fr 2fr; /* 3 cột: 100px, 1 phần, 2 phần */
  grid-template-rows: auto 200px; /* 2 hàng: tự động, 200px */
  gap: 20px; /* Khoảng cách giữa các ô */
  justify-items: center; /* Căn giữa ngang các item */
  align-items: start; /* Căn đầu dọc các item */
}
```

Grid Items

```
.grid-item {
  grid-column: 1 / 3; /* Chiếm từ cột 1 đến cột 3 */
  grid-row: 2 / 3; /* Chiếm hàng 2 */
  justify-self: stretch; /* Kéo dãn ngang */
  align-self: center; /* Căn giữa dọc */
}
```

Grid Areas

```
.grid-container {
  grid-template-areas:
    "header header"
    "sidebar main"
    "footer footer";
}
.item-header {
  grid-area: header;
}
```

---

4. Ví dụ minh họa

Ví dụ cơ bản: Lưới 3x2 đơn giản

Tạo một lưới 3 cột, 2 hàng với các ô có kích thước cố định và khoảng cách.

```
<!DOCTYPE html>
<html lang="vi">
<head>
  <meta charset="UTF-8">
  <title>CSS Grid Basic Example</title>
  <style>
    .grid-container {
      display: grid;
      grid-template-columns: repeat(3, 1fr); /* 3 cột đều nhau */
      grid-template-rows: 100px 100px; /* 2 hàng, mỗi hàng 100px */
      gap: 15px;
      padding: 20px;
      background-color: #f0f0f0;
    }
    .grid-item {
      background-color: #b2ebf2;
      padding: 10px;
      text-align: center;
      border-radius: 5px;
    }
  </style>
</head>
<body>
  <div class="grid-container">
    <div class="grid-item">Item 1</div>
    <div class="grid-item">Item 2</div>
    <div class="grid-item">Item 3</div>
    <div class="grid-item">Item 4</div>
```

```
    <div class="grid-item">Item 5</div>
    <div class="grid-item">Item 6</div>
  </div>
</body>
</html>
```

Giải thích:

- `repeat(3, 1fr)` tạo 3 cột có chiều rộng bằng nhau.
- `grid-template-rows: 100px 100px` tạo 2 hàng, mỗi hàng cao 100px.
- Các grid items tự động được đặt vào các ô theo thứ tự từ trái sang phải, trên xuống dưới.

Ví dụ nâng cao 1: Bố cục trang web với Grid Areas

Tạo một bố cục trang web gồm header, sidebar, main content, và footer bằng `grid-template-areas`.

```
<!DOCTYPE html>
<html lang="vi">
<head>
  <meta charset="UTF-8">
  <title>CSS Grid Advanced Example 1</title>
  <style>
    .grid-container {
      display: grid;
      grid-template-columns: 200px 1fr; /* Sidebar 200px, main tự giãn */
      grid-template-rows: 80px 1fr 60px; /* Header, main, footer */
      gap: 10px;
      height: 100vh;
      grid-template-areas:
        "header header"
        "sidebar main"
        "footer footer";
    }
    .header {
      grid-area: header;
      background-color: #333;
      color: white;
      padding: 20px;
      text-align: center;
    }
    .sidebar {
      grid-area: sidebar;
      background-color: #f0f0f0;
      padding: 20px;
    }
    .main {
      grid-area: main;
```

```
      background-color: #e0f7fa;
      padding: 20px;
    }
    .footer {
      grid-area: footer;
      background-color: #333;
      color: white;
      padding: 20px;
      text-align: center;
    }
  </style>
</head>
<body>
  <div class="grid-container">
    <header class="header">Header</header>
    <aside class="sidebar">Sidebar</aside>
    <main class="main">Nội dung chính</main>
    <footer class="footer">Footer</footer>
  </div>
</body>
</html>
```

Giải thích:

- `grid-template-areas` định nghĩa bố cục bằng tên vùng, giúp dễ dàng hình dung cấu trúc.
- `grid-template-columns` và `grid-template-rows` xác định kích thước các cột và hàng.
- `height: 100vh` đảm bảo lưới chiếm toàn bộ chiều cao viewport.

Ví dụ nâng cao 2: Gallery ảnh responsive với auto-fit

Tạo một gallery ảnh responsive sử dụng `auto-fit` và `minmax()`.

```
<!DOCTYPE html>
<html lang="vi">
<head>
  <meta charset="UTF-8">
  <title>CSS Grid Advanced Example 2</title>
  <style>
    .gallery {
      display: grid;
      grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
      gap: 15px;
      padding: 20px;
    }
    .gallery__item {
      background-color: #b2ebf2;
      padding: 10px;
```

```
      border-radius: 5px;
      text-align: center;
    }
    .gallery__item img {
      max-width: 100%;
      border-radius: 5px;
    }
    .gallery__item--large {
      grid-column: span 2; /* Chiếm 2 cột */
    }
  </style>
</head>
<body>
  <div class="gallery">
    <div class="gallery__item"><img src="https://via.placeholder.com/200"
alt="Image 1"></div>
    <div class="gallery__item gallery__item--large"><img
src="https://via.placeholder.com/400" alt="Image 2"></div>
    <div class="gallery__item"><img src="https://via.placeholder.com/200"
alt="Image 3"></div>
    <div class="gallery__item"><img src="https://via.placeholder.com/200"
alt="Image 4"></div>
  </div>
</body>
</html>
```

Giải thích:

- `repeat(auto-fit, minmax(200px, 1fr))` tạo các cột với chiều rộng tối thiểu 200px, tự động điều chỉnh số lượng cột dựa trên kích thước container.
- `grid-column: span 2` khiến một item chiếm 2 cột, tạo hiệu ứng bố cục không đều (masonry-like).
- Gallery tự động responsive mà không cần media queries.

---

5. Ứng dụng thực tế

- **Bố cục trang web**: Tạo các bố cục phức tạp như header, sidebar, main content, và footer mà không cần float hoặc framework.
- **Dashboard quản trị**: Sắp xếp các widget (biểu đồ, bảng dữ liệu) trong bố cục lưới linh hoạt, dễ dàng điều chỉnh trên các thiết bị.
- **Gallery ảnh**: Tạo các gallery responsive hoặc bố cục masonry cho portfolio, trang sản phẩm.
- **E-commerce**: Hiển thị danh sách sản phẩm trong lưới, với khả năng thay đổi số cột dựa trên kích thước màn hình.
- **Ứng dụng web**: Sử dụng CSS Grid để tạo các giao diện tương tác, như bảng điều khiển hoặc form nhập liệu phức tạp.

---

6. Ưu điểm và nhược điểm

Ưu điểm

- **Linh hoạt**: Hỗ trợ bố cục 2D, dễ dàng kiểm soát cả hàng và cột.
- **Responsive**: Kết hợp với `minmax()`, `auto-fit`, và media queries để tạo giao diện thích ứng.
- **Dễ bảo trì**: Cú pháp rõ ràng, giảm sự phụ thuộc vào các kỹ thuật phức tạp như float hoặc absolute positioning.
- **Hiệu suất cao**: Được tối ưu hóa bởi trình duyệt, nhẹ hơn nhiều framework CSS.

Nhược điểm

- **Khúc học tập**: Yêu cầu hiểu nhiều thuộc tính và cách chúng tương tác.
- **Tương thích trình duyệt cũ**: Không hỗ trợ tốt trên các trình duyệt rất旧 (như IE9). Tuy nhiên, các trình duyệt hiện đại (2025) đều hỗ trợ đầy đủ.
- **Phức tạp cho bố cục đơn giản**: Có thể quá mức cần thiết cho các bố cục tuyến tính (Flexbox phù hợp hơn).

---

7. Bài tập thực hành

1. **Lưới cơ bản**:
   - Tạo một lưới 4x2 với các ô có màu nền khác nhau, sử dụng `repeat()` và `gap`.
   - Thêm một item chiếm 2 cột bằng `grid-column: span 2`.
2. **Bố cục trang web**:
   - Xây dựng một trang web với bố cục gồm header (chiếm toàn bộ chiều rộng), sidebar (200px), main content, và footer, sử dụng `grid-template-areas`.
   - Thêm media query để chuyển sidebar thành full-width trên màn hình nhỏ (<600px).
3. **Gallery responsive**:
   - Tạo một gallery ảnh với 6 item, sử dụng `auto-fit` và `minmax(150px, 1fr)`.
   - Làm cho một item ngẫu nhiên chiếm 2 cột hoặc 2 hàng, tạo hiệu ứng masonry.
4. **Dashboard**:
   - Tạo một dashboard với 5 widget, sử dụng CSS Grid để sắp xếp linh hoạt.
   - Sử dụng `justify-items` và `align-items` để căn chỉnh nội dung trong các ô.

---

8. Tài nguyên bổ sung

- **MDN Web Docs**: Tài liệu về CSS Grid Layout (developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout).
- **CSS Tricks**: Hướng dẫn chi tiết và ví dụ về CSS Grid (css-tricks.com/snippets/css/complete-guide-grid).
- **Grid by Example**: Bộ sưu tập ví dụ thực tế về CSS Grid (gridbyexample.com).

- **CanIUse**: Kiểm tra tương thích trình duyệt (caniuse.com/css-grid).
- **CodePen**: Thực hành và chia sẻ mã (codepen.io).

---

9. Lưu ý

- **Tương thích trình duyệt**: CSS Grid được hỗ trợ đầy đủ trên Chrome, Firefox, Edge, Safari từ 2017. Nếu cần hỗ trợ IE10/11, sử dụng các thuộc tính đơn giản hơn hoặc polyfill.
- **Kết hợp với Flexbox**: Sử dụng CSS Grid cho bố cục 2D và Flexbox cho bố cục 1D (hàng hoặc cột) để đạt hiệu quả tối ưu.
- **Debug**: Sử dụng DevTools (Chrome/Firefox) để kiểm tra lưới bằng tính năng "Inspect Grid" (hiển thị grid lines).
- **Hiệu suất**: Tránh sử dụng quá nhiều ô hoặc thuộc tính phức tạp trong lưới lớn để đảm bảo hiệu suất.