**CSS 레이아웃 입문서**

* [Overview: CSS 레이아웃](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout)
* [다음](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Normal_Flow)

이 문서에서는 이전 모듈에서 이미 다뤘던 CSS 레이아웃 기능, 예를 들어 다양한 [display (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/display) 속성값의 차이 등을 복습하고, 이번 과정에서 다룰 몇 가지 CSS 개념을 소개합니다.

|  |  |
| --- | --- |
| **선결 사항:** | HTML의 기초 ([HTML에 대한 소개](https://developer.mozilla.org/ko/docs/Learn/HTML/Introduction_to_HTML))와 CSS 작동 방식에 대한 개념([CSS 소개](https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps)를 공부하세요.) |
| **목표:** | CSS 페이지 레이아웃 기술에 대한 개요를 제공합니다. 개별 기술은 후속 자습서에서 보다 자세하게 학습할 수 있습니다. |

CSS 페이지 레이아웃 기술은 웹페이지에 포함될 요소들을 가져와 일반적인 레이아웃 흐름에서의 기본 위치, 주변의 다른 요소, 부모 컨테이너, 기본 뷰포트 및 윈도우 등의 요소와 관련하여 해당 요소가 어디에 위치할지 제어할 수 있습니다. 우리가 이번 과정에서 자세하게 다룰 페이지 레이아웃 기술은 다음과 같습니다.

* 일반 흐름(normal flow)
* [display (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/display) 속성
* 플렉스박스(Flexbox)
* 그리드(Grid)
* 플로트(Floats)
* 포지셔닝
* 테이블 레이아웃
* 다단 레이아웃

각각의 기술은 저마다 용도와 장단점이 있습니다. 어떤 기술도 단독으로 사용하도록 설계되지는 않았습니다. 각 레이아웃 기술이 어떤 용도로 설계된 것인지 이해하게 되면 해당 작업에 가장 적합한 도구가 어떤 것인지 파악할 수 있습니다.

[**일반 흐름(normal flow)**](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Introduction#%EC%9D%BC%EB%B0%98_%ED%9D%90%EB%A6%84normal_flow)

일반 흐름(normal flow)은 페이지 레이아웃을 제어하기 위해 아무것도 하지 않을 경우, 브라우저가 기본값으로 HTML 페이지를 배치하는 방법을 말합니다. 간단한 HTML 예를 살펴봅시다.

HTMLPlayCopy to Clipboard

<p>나는 고양이를 사랑한다.</p>

<ul>

<li>고양이 먹이를 사세요</li>

<li>운동</li>

<li>기운내 친구야</li>

</ul>

<p>여기가 끝!</p>

기본적으로 브라우저는 이 코드를 다음과 같이 표시합니다.

Play

HTML이 소스 코드에 나타나는 순서 그대로 표시되는지, 첫 번째 단락과 정렬되지 않은 목록, 두 번째 단락 등 요소들이 서로 겹쳐있는지 확인하세요.

요소들이 서로 나란히 표시되는 요소를 **인라인(inline)** 요소라고 하며, 다른 요소 아래 하나씩 표시되는 요소를 **블록(block)** 요소라고 합니다.

**참고:** 블록 요소 콘텐츠가 배치되는 방향을 블록 방향이라고 합니다. 영어와 같은 언어 상에서 블록 방향은 수직 방향이 되며, 이는 가로쓰기 모드입니다. 일본어와 같은 세로 쓰기 모드가 있는 언어에서는 블록 방향은 수평 방향이 됩니다. 인라인 방향은 문장과 같은 인라인 콘텐츠가 흘러가는 방향을 말합니다.

페이지의 많은 요소에서 일반 흐름(normal flow)은 필요한 레이아웃을 정확히 생성합니다. 그러나 더 복잡한 레이아웃의 경우, CSS에서 사용할 수 있는 몇 가지 도구를 사용하여 이 기본 동작을 변경해야 합니다. 잘 구조화된 HTML 문서로 시작하는 것이 아주 중요한 이유는 기본 레이아웃과 충돌하지 않는 레이아웃으로 작업할 수 있기 때문입니다.

CSS에서 요소가 배치되는 방식을 변경시키는 메서드는 다음과 같습니다.

* [**display (en-US)**](https://developer.mozilla.org/en-US/docs/Web/CSS/display)**속성** — block, inline 또는 inline-block과 같은 기준 속성값은 일반 흐름(normal flow) 속에서 요소가 동작하는 방식을 변경할 수 있습니다. 예를 들면 블록 요소를 인라인 요소로 동작하게 할 수도 있습니다. (더 자세한 정보는 [CSS 박스 유형 (en-US)](https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/The_box_model#types_of_css_boxes)을 참조하세요). 또한 display 속성값 지정을 통해 활성화될 수 있는 전체 레이아웃 메서드도 존재합니다. 그 예로는 [CSS 그리드](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Grids)와 [플렉스박스](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Flexbox)가 있으며, 부모 요소 내부에 자식 요소가 배치되는 방식을 변경할 수 있습니다.
* **플로트(Floats)** — [float](https://developer.mozilla.org/ko/docs/Web/CSS/float) 속성의 값을 예로 left로 적용하면 흔히 매거진 레이아웃에 속한 이미지가 텍스트를 자신의 주변에 떠 있게 하는 방식과 같이 블록 수준 요소가 어떤 요소의 한쪽 측면으로 밀려나도록 할 수 있다.
* [**position**](https://developer.mozilla.org/ko/docs/Web/CSS/position)**속성** — 다른 박스 내부에 있는 박스의 배치를 정밀하게 제어할 수 있습니다. 일반 흐름에서는 정적(static) 포지셔닝이 기본값이지만, 브라우저 뷰포트에 고정하는 등 다른 값을 사용하여 요소를 다르게 배치할 수 있습니다.
* **테이블 레이아웃** — HTML 테이블의 일부 스타일을 지정하기 위해 고안된 기능으로, display: table 관련 속성을 사용하여 테이블 관련 속성을 비테이블 요소에서도 사용할 수 있습니다.
* **다단 레이아웃** — [다단 레이아웃](https://developer.mozilla.org/ko/docs/Web/CSS/CSS_multicol_layout) 속성을 사용하면 마치 신문처럼 블록 콘텐츠를 단 형태로 배치되도록 만들 수 있습니다.

[**디스플레이(display) 속성**](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Introduction#%EB%94%94%EC%8A%A4%ED%94%8C%EB%A0%88%EC%9D%B4display_%EC%86%8D%EC%84%B1)

CSS 상에서 페이지 레이아웃을 구현하는 주요 방법은 display 속성에 포함된 모든 속성의 값을 지정하는 것입니다. 이 속성을 사용하면 무언가를 표시하는 기본값을 변경할 수 있습니다. 일반 흐름(normal flow)상의 모든 요소에는 기본 display 속성값, 즉 요소가 작동하도록 설정된 기본 방식을 가지고 있습니다. 예를 들어 영어로 된 단락이 다른 대상 요소 바로 밑에 표시되는 것은 그들 요소의 스타일이 display: block으로 지정되었기 때문입니다. 단락 내부 일부 텍스트 주변에 링크를 만들면, 해당 링크는 나머지 텍스트와 함께 인라인을 유지하며 새 줄로 넘어가는 행갈이를 하지 않습니다. 이는 [<a>](https://developer.mozilla.org/ko/docs/Web/HTML/Element/a) 요소가 기본값으로 display: inline이기 때문입니다.

이러한 기본값 display 속성을 변경할 수 있습니다. 예를 들어 [<li>](https://developer.mozilla.org/ko/docs/Web/HTML/Element/li) 요소는 기본값으로 display: block이므로 영어 문서상에서 다른 대상 요소 바로 밑에 표시됩니다. display 속성값을 inline으로 변경하면 문장에서 단어가 나란히 표시되는 것처럼 목록 항목이 나란히 표시됩니다. 모든 요소에 대해 display 속성값을 변경할 수 있다는 것은 HTML 요소들이 어떻게 보일지 신경 쓰지 않고, 해당 HTML 요소의 의미론적 의의를 선택할 수 있다는 뜻입니다. 표시 방식은 사용자가 변경할 수 있는 부분입니다.

한 항목을 block에서 inline으로 바꾸거나, 그 반대로 바꿔 기본 표시 방식을 변경할 수 있는 것 외에도, 하나의 display 속성값으로 시작해서 몇 가지 더 복잡한 레이아웃 방법들도 있습니다. 그러나 이러한 방법을 사용할 때는 추가 속성을 호출해야 합니다. 레이아웃의 목적이 무엇인지 논할 때 가장 중요한 두 가지 속성값은 display: flex와 display: grid이다.

[**플렉스박스(flexbox)**](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Introduction#%ED%94%8C%EB%A0%89%EC%8A%A4%EB%B0%95%EC%8A%A4flexbox)

플렉스박스(flexbox)는 [유연한 박스 레이아웃](https://developer.mozilla.org/ko/docs/Web/CSS/CSS_flexible_box_layout) CSS 모듈의 줄임말로, 행이나 열을 한 차원으로 쉽게 레이아웃할 수 있도록 설계되었습니다. 플렉스박스를 사용하려면 레이아웃하려는 요소의 부모 요소에 display: flex를 적용하면 모든 직계 자식이 *플렉스 항목*이 됩니다. 간단한 예시를 통해 이를 확인할 수 있습니다.

[**display:flex 설정하기**](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Introduction#displayflex_%EC%84%A4%EC%A0%95%ED%95%98%EA%B8%B0)

아래 HTML 마크업은 wrapper 클래스를 가진 컨테이너 요소를 제공하고, 그 요소 내부에는 세 개의 [<div> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/div) 요소를 가지고 있습니다. 기본적으로 이러한 요소는 영어 문서상에서 블록 요소로 서로 아래에 표시됩니다.

그러나 부모 요소에 display: flex를 추가하면 이제 세 항목이 열로 정렬됩니다. 이것은 요소들이 *플렉스 항목*이 되었을 뿐만 아니라 플렉스박스가 설정하는 일부 초기 값의 영향을 받았기 때문입니다. 부모 요소의 [flex-direction](https://developer.mozilla.org/ko/docs/Web/CSS/flex-direction) 속성의 초기 값이 row이기 때문에 행으로 표시됩니다. 그리고 부모 요소의 [align-items](https://developer.mozilla.org/ko/docs/Web/CSS/align-items) 속성의 초기 값이 stretch이기 때문에 모두 높이가 늘어나는 것처럼 보입니다. 즉, 항목이 플렉스 컨테이너의 높이까지 늘어나는데, 이 경우 가장 높은 항목에 의해 정의됩니다. 항목은 모두 컨테이너의 시작 부분에 정렬되며 행 끝에 여분의 공간이 남습니다.

CSSPlayCopy to Clipboard

.wrapper {

display: flex;

}

HTMLPlayCopy to Clipboard

<div class="wrapper">

<div class="box1">하나</div>

<div class="box2">둘</div>

<div class="box3">셋</div>

</div>

Play

[**플렉스 속성 설정하기**](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Introduction#%ED%94%8C%EB%A0%89%EC%8A%A4_%EC%86%8D%EC%84%B1_%EC%84%A4%EC%A0%95%ED%95%98%EA%B8%B0)

*플렉스 콘테이너*에 적용할 수 있는 속성 외에도 *플렉스 항목*에 적용할 수 있는 속성도 있습니다. 이러한 속성은 무엇보다도 항목이 *플렉스*되는 방식을 변경하여 사용 가능한 공간에 따라 항목을 확장하거나 축소할 수 있습니다.

간단한 예로 모든 하위 항목에 [flex](https://developer.mozilla.org/ko/docs/Web/CSS/flex) 속성을 추가하고 값을 1로 지정할 수 있습니다. 이렇게 하면 끝에 공간이 남지 않고, 모든 항목이 커져서 컨테이너를 채우게 됩니다. 공간이 더 많으면 항목이 더 넓어지고, 공간이 더 적다면 항목이 더 좁아집니다. 또한 마크업에 다른 요소를 추가하면 해당 요소를 위한 공간을 만들기 위해 다른 항목이 모두 작아지며, 모든 항목이 계속해서 모든 공간을 차지하게 됩니다.

CSSPlayCopy to Clipboard

.wrapper {

display: flex;

}

.wrapper > div {

flex: 1;

}

HTMLPlayCopy to Clipboard

<div class="wrapper">

<div class="box1">하나</div>

<div class="box2">둘</div>

<div class="box3">셋</div>

</div>

Play

**참고:** 지금까지 플렉스박스에서 가능한 기능에 대해 간략히 소개했습니다. 더 자세한 내용은 [flexbox](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Flexbox) 편을 참조하세요.

[**그리드 레이아웃**](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Introduction#%EA%B7%B8%EB%A6%AC%EB%93%9C_%EB%A0%88%EC%9D%B4%EC%95%84%EC%9B%83)

플렉스박스는 일차원 레이아웃을 위해 설계된 반면, 그리드 레이아웃은 행과 열로 정렬하는 이차원 레이아웃을 위해 설계되었습니다.

[**display:grid 설정하기**](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Introduction#displaygrid_%EC%84%A4%EC%A0%95%ED%95%98%EA%B8%B0)

플렉스박스와 마찬가지로 디스플레이 속성에 display: grid라는 특정 값을 지정하면 그리드 레이아웃으로 활성화할 수 있다. 아래 예시에서는 플렉스 예시와 유사한 마크업을 사용하여 컨테이너와 일부 자식 요소를 사용합니다. display: grid를 사용하는 것 외에도 [grid-template-rows (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/grid-template-rows)와 [grid-template-columns](https://developer.mozilla.org/ko/docs/Web/CSS/grid-template-columns) 속성을 사용하여 부모 요소에 대한 행과 열 칸을 정의합니다. 각각 1fr값이 지정된 3개의 열과 100px이 지정된 2개의 행을 정의했습니다. 그리드에서 생성한 셀에 자동으로 배치되므로 자식 요소에 규칙을 적용할 필요가 없습니다.

CSSPlayCopy to Clipboard

.wrapper {

display: grid;

grid-template-columns: 1fr 1fr 1fr;

grid-template-rows: 100px 100px;

grid-gap: 10px;

}

HTMLPlayCopy to Clipboard

<div class="wrapper">

<div class="box1">하나</div>

<div class="box2">둘</div>

<div class="box3">셋</div>

<div class="box4">넷</div>

<div class="box5">다섯</div>

<div class="box6">여섯</div>

</div>

Play

[**그리드에 항목 배치하기**](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Introduction#%EA%B7%B8%EB%A6%AC%EB%93%9C%EC%97%90_%ED%95%AD%EB%AA%A9_%EB%B0%B0%EC%B9%98%ED%95%98%EA%B8%B0)

그리드가 있으면 위에 표시된 자동 배치 동작에 의존하지 않고, 명시적으로 항목을 그리드에 배치할 수 있습니다. 아래 다음 예제에서는 동일한 그리드를 정의했지만, 이번에는 세 개의 하위 항목이 정의했습니다. [grid-column (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/grid-column)과 [grid-row (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/grid-row)를 사용해서 각 항목의 시작 및 끝 줄을 설정했습니다. 이렇게 하면 항목이 여러 칸에 걸쳐있습니다.

CSSPlayCopy to Clipboard

.wrapper {

display: grid;

grid-template-columns: 1fr 1fr 1fr;

grid-template-rows: 100px 100px;

grid-gap: 10px;

}

.box1 {

grid-column: 2 / 4;

grid-row: 1;

}

.box2 {

grid-column: 1;

grid-row: 1 / 3;

}

.box3 {

grid-row: 2;

grid-column: 3;

}

HTMLPlayCopy to Clipboard

<div class="wrapper">

<div class="box1">하나</div>

<div class="box2">둘</div>

<div class="box3">셋</div>

</div>

Play

**참고:** 이 두 예제는 그리드 레이아웃의 강력한 기능을 보여주는 작은 샘플에 불과합니다. 자세한 내용은 [그리드 레이아웃](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Grids) 문서를 참조하세요.

이 가이드의 나머지 부분에서는 페이지의 기본 레이아웃 구조로 보기엔 덜 중요하지만, 특정 작업을 수행하는 데 여전히 도움이 될 수 있는 다른 레이아웃 방법을 다룹니다. 각 레이아웃 작업의 특성을 이해함으로써 디자인의 특정 구성 요소를 볼 때, 해당 구성 요소에 가장 적합한 레이아웃 유형이 명확하게 드러나는 경우가 많습니다.

[**플로트(Floats)**](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Introduction#%ED%94%8C%EB%A1%9C%ED%8A%B8floats)

요소를 플로팅 하면 일반 흐름(normal flow)에서 해당 요소와 그 뒤에 따르는 블록 수준 요소의 동작이 변경됩니다. 플로팅 된 요소는 왼쪽이나 오른쪽으로 이동되어 일반 흐름(normal flow)에서 벗어나게 되며 주변 콘텐츠가 그 주위에 떠 있습니다.

이 [float](https://developer.mozilla.org/ko/docs/Web/CSS/float) 속성은 네 가지 값을 가질 수 있습니다.

* left — 요소를 왼쪽으로 띄웁니다.
* right — 요소를 오른쪽으로 띄웁니다.
* none — 플로팅을 전혀 지정하지 않습니다. 이것이 기본값입니다.
* inherit — 플로팅 속성의 값을 요소의 부모 요소에서 상속하도록 지정합니다.

아래 예제에서 <div>를 왼쪽으로 띄우고, 오른쪽에 [margin](https://developer.mozilla.org/ko/docs/Web/CSS/margin)을 두어 주변 텍스트를 밀어냅니다. 이렇게 하면 텍스트가 박스 요소를 감싸는 효과를 얻을 수 있으며, 최신 웹 디자인에서 사용되는 플로트에 대해 알아야 할 대부분의 내용을 알 수 있습니다.

HTMLPlayCopy to Clipboard

<h1>간단한 플로트 예제</h1>

<div class="box">플로트</div>

<p>

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla luctus aliquam

dolor, eu lacinia lorem placerat vulputate. Duis felis orci, pulvinar id metus

ut, rutrum luctus orci. Cras porttitor imperdiet nunc, at ultricies tellus

laoreet sit amet. Sed auctor cursus massa at porta. Integer ligula ipsum,

tristique sit amet orci vel, viverra egestas ligula. Curabitur vehicula tellus

neque, ac ornare ex malesuada et. In vitae convallis lacus. Aliquam erat

volutpat. Suspendisse ac imperdiet turpis. Aenean finibus sollicitudin eros

pharetra congue. Duis ornare egestas augue ut luctus. Proin blandit quam nec

lacus varius commodo et a urna. Ut id ornare felis, eget fermentum sapien.

</p>

CSSPlayCopy to Clipboard

.box {

float: left;

width: 150px;

height: 150px;

margin-right: 30px;

}

Play

**참고:** 플로트에 대한 설명은 [플로트 및 해제](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Floats) 속성 편에서 충분히 다룹니다. 플렉스박스 및 그리드 레이아웃과 같은 기술이 나오기 이전에 플로트는 열 레이아웃을 만드는 방법으로 사용되었습니다. 웹에서 여전히 이러한 방법을 접할 수 있으며, [레거시 레이아웃 방법](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Legacy_Layout_Methods)에서 그 내용을 다룰 것입니다.

[**포지셔닝 기술**](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Introduction#%ED%8F%AC%EC%A7%80%EC%85%94%EB%8B%9D_%EA%B8%B0%EC%88%A0)

포지셔닝을 사용하면 요소를 일반 흐름(normal flow)에서 배치되는 위치에서 다른 위치로 이동할 수 있습니다. 포지셔닝은 페이지의 기본 레이아웃을 만드는 방법이 아니라, 페이지에서 특정 항목의 위치를 관리하고 미세 조정하는 데 사용됩니다.

그러나 [position](https://developer.mozilla.org/ko/docs/Web/CSS/position) 속성에 의존하는 특정 레이아웃 패턴을 얻는 데 유용한 기술이 있습니다. 포지셔닝를 이해하면 일반 흐름(normal flow)과 항목을 정상적인 흐름에서 벗어나는 것이 무엇을 의미하는지 이해하는 데도 도움이 됩니다.

알아야 할 다섯 가지 유형의 포지셔닝이 있습니다.

* **정적(static) 포지셔닝**은 모든 요소에 기본값으로 부여된 속성입니다. 이는 "문서 레이아웃 흐름에서 요소를 정상 위치에 배치하는 것이며, 특별히 더 살펴볼 것은 없다는 것"을 의미합니다.
* **상대(relative) 포지셔닝**을 사용하면 페이지에서 요소의 위치를 수정하여 일반 흐름의 위치에 비해 상대적으로 이동하고, 페이지의 다른 요소와 겹치도록 만들 수 있습니다.
* **절대(absolute) 포지셔닝**은 요소를 페이지의 일반적인 레이아웃 흐름에서 완전히 벗어나 마치 별도의 레이어에 있는 것처럼 이동시킵니다. 여기에서 가장 가까운 위치에 있는 상위요소의 가장자리를 기준으로 한 위치를 고정할 수 있습니다.(다른 상위 요소가 없는 경우, <html>이 기준이 됨.) 이 기능은 서로 다른 콘텐츠 패널이 서로 겹쳐져 있고, 원하는 대로 표시되거나 숨겨지는 탭 상자나, 기본적으로 화면 밖에 있지만 제어 버튼을 사용하여 화면에서 슬라이드되도록 만들 수 있는 정보 패널과 같은 복잡한 레이아웃 효과를 만드는 데 유용합니다.
* **고정(fixed) 포지셔닝**은 다른 요소가 아닌 브라우저 뷰포트를 기준으로 요소를 고정한다는 점을 제외하면 절대 포지셔닝과 매우 유사합니다. 이 방법은 나머지 콘텐츠가 스크롤될 때, 항상 화면의 같은 위치에 유지되는 영구 탐색 메뉴와 같은 효과를 만드는 데 유용합니다.
* **스티키(sticky) 포지셔닝**은 뷰포트에서 사전에 정의된 오프셋에 도달할 때까지 요소가 position: relative와 같이 작동하고, 그 시점부터는 position: fixed와 같이 작동하는 새로운 포지셔닝 방법입니다.

[**간단한 포지셔닝 예제**](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Introduction#%EA%B0%84%EB%8B%A8%ED%95%9C_%ED%8F%AC%EC%A7%80%EC%85%94%EB%8B%9D_%EC%98%88%EC%A0%9C)

이러한 페이지 레이아웃 기술에 친숙해지도록 몇 가지 간단한 예시를 보여드리겠습니다. 예제는 모두 다음과 같은 동일한 HTML 구조(제목 뒤에 세 개의 단락이 있는)를 사용합니다.

HTMLPlayCopy to Clipboard

<h1>포지셔닝</h1>

<p>나는 기본 블록 수준 요소입니다.</p>

<p class="positioned">나는 기본 블록 수준 요소입니다.</p>

<p>나는 기본 블록 수준 요소입니다.</p>

이 HTML은 다음 CSS를 사용하여 스타일이 지정됩니다.

CSSPlayCopy to Clipboard

body {

width: 500px;

margin: 0 auto;

}

p {

background-color: rgb(207, 232, 220);

border: 2px solid rgb(79, 185, 227);

padding: 10px;

margin: 10px;

border-radius: 5px;

}

렌더링된 출력은 다음과 같습니다.

Play

[**상대(relative) 포지셔닝**](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Introduction#%EC%83%81%EB%8C%80relative_%ED%8F%AC%EC%A7%80%EC%85%94%EB%8B%9D)

상대 포지셔닝을 사용하면 일반 흐름에서 항목을 기본 위치에서 주어지는 오프셋만큼 간격을 띄울 수 있습니다. 즉, 아이콘을 약간 아래로 이동하여 텍스트와 일직선이 되도록 하는 등의 작업을 수행할 수 있습니다. 이를 위해 다음 규칙을 추가하여 상대 포지셔닝을 추가할 수 있습니다.

CSSPlayCopy to Clipboard

.positioned {

position: relative;

top: 30px;

left: 30px;

}

여기서는 중간 단락에 [position](https://developer.mozilla.org/ko/docs/Web/CSS/position) 값을 relative로 지정합니다. 이것은 그 자체적로는 아무것도 하지 않으므로 [top (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/top) 및 [left (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/left) 속성도 추가합니다. 이는 영향을 받는 요소를 아래쪽과 오른쪽으로 이동시키는 역할을 합니다. 이는 기대했던 것과 반대되는 것처럼 보일 수 있지만, 요소가 왼쪽과 위쪽에서 밀려서 오른쪽과 아래로 이동한다고 생각하면 됩니다.

이 코드를 추가하면 다음과 같은 결과가 나타납니다.

CSSPlayCopy to Clipboard

.positioned {

position: relative;

background: rgba(255, 84, 104, 0.3);

border: 2px solid rgb(255, 84, 104);

top: 30px;

left: 30px;

}

Play

[**절대(absolute) 포지셔닝**](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Introduction#%EC%A0%88%EB%8C%80absolute_%ED%8F%AC%EC%A7%80%EC%85%94%EB%8B%9D)

절대 포지셔닝은 일반 흐름(normal flow)에서 요소를 완전히 제거하고, 대신 컨테이너 블록의 가장자리로부터 오프셋을 사용하여 요소를 배치하는 데 사용됩니다.

원래의 포지셔닝 되지 않은 예제로 돌아가서 다음 CSS 규칙을 추가하여 절대 위치 지정을 구현할 수 있습니다.

CSSPlayCopy to Clipboard

.positioned {

position: absolute;

top: 30px;

left: 30px;

}

여기서는 중간 단락에 [position](https://developer.mozilla.org/ko/docs/Web/CSS/position) 값을 absolute로 지정하고, 이전과 동일한 [top (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/top)과 [left (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/left)속성을 지정합니다. 이 코드를 추가하면 다음과 같은 결과가 생성됩니다.

CSSPlayCopy to Clipboard

.positioned {

position: absolute;

background: rgba(255, 84, 104, 0.3);

border: 2px solid rgb(255, 84, 104);

top: 30px;

left: 30px;

}

Play

아주 많이 달라졌습니다! 이제 배치된 요소가 페이지 레이아웃의 나머지 부분과 완전히 분리되어 그 위에 배치됩니다. 다른 두 단락은 이제 위치가 지정된 형제 단락이 존재하지 않는 것처럼 함께 배치됩니다. [top (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/top) 및 [left (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/left) 속성은 상대 포지셔닝 요소에 대해 미치는 것과는 다르게 절대 포지셔닝 요소에 대해 영향을 줍니다. 이 경우 오프셋은 페이지의 상단과 왼쪽에서 계산되었습니다. 이 컨테이너가 되는 부모 요소를 변경할 수 있으며, [포지셔닝](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Positioning)에 관한 과정에서 이에 대해 살펴보겠습니다.

[**고정(fixed) 포지셔닝**](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Introduction#%EA%B3%A0%EC%A0%95fixed_%ED%8F%AC%EC%A7%80%EC%85%94%EB%8B%9D)

고정 포지셔닝은 절대 포지셔닝과 같은 방식으로 문서 일반 흐름에서 요소를 제거합니다. 그러나 오프셋이 컨테이너에서 적용되는 대신 뷰포트를 기준으로 적용됩니다. 항목이 뷰포트 기준으로 고정되기 때문에 페이지가 아래로 스크롤될 때 고정된 상태로 유지되는 메뉴와 같은 효과를 만들 수 있습니다.

이번 예제에서는 페이지를 스크롤할 수 있도록 세 단락의 텍스트와 position: fixed가 지정된 박스가 포함되어 있습니다.

HTMLPlayCopy to Clipboard

<h1>고정 포지셔닝</h1>

<div class="positioned">고정</div>

<p>단락 1.</p>

<p>단락 2.</p>

<p>단락 3.</p>

CSSPlayCopy to Clipboard

.positioned {

position: fixed;

top: 30px;

left: 30px;

}

Play

[**스티키(sticky) 포지셔닝**](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Introduction#%EC%8A%A4%ED%8B%B0%ED%82%A4sticky_%ED%8F%AC%EC%A7%80%EC%85%94%EB%8B%9D)

스티키 포지셔닝은 포지셔닝 유형 중 남은 마지막 포지셔닝 방법입니다. 상대 포지셔닝과 고정 포지셔닝을 혼합한 것입니다. 항목에 position: sticky가 지정된 경우, 해당 항목은 뷰포트 기준에서 오프셋에 도달할 때까지 일반 흐름으로 스크롤 됩니다. 이 시점에서 position: fixed가 적용된 것처럼 "고정"됩니다.

CSSPlayCopy to Clipboard

.positioned {

position: sticky;

top: 30px;

left: 30px;

}

Play

**참고:** 포지셔닝에 대해 더 알아보고 싶다면, [포지셔닝](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Positioning) 문서를 참조하세요.

[**테이블 레이아웃**](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Introduction#%ED%85%8C%EC%9D%B4%EB%B8%94_%EB%A0%88%EC%9D%B4%EC%95%84%EC%9B%83)

HTML 테이블은 테이블 형식의 데이터를 표시하는 데 적합하지만, 수년 전, 즉 기본적인 CSS조차 브라우저에서 안정적으로 지원되기 전에는 웹 개발자가 전체 웹 페이지 레이아웃에도 테이블을 사용하여 머리글, 바닥글, 열 등을 다양한 테이블 행과 열에 넣곤 했습니다. 이 방식은 당시에는 효과가 있었지만, 테이블 레이아웃이 유연하지 않고, 마크업이 매우 무거우며, 디버깅이 어렵고, 의미상 잘못된 점(예: 스크린 리더 사용자가 테이블 레이아웃을 탐색하는 데 문제가 있음) 등 많은 문제점을 가지고 있었습니다.

테이블 마크업을 사용할 때, 웹 페이지에서 테이블이 보이는 방식은 테이블 레이아웃을 정의하는 일련의 CSS 속성에 의해 표시됩니다. 이러한 동일한 속성은 테이블이 아닌 요소를 레이아웃하는 데 사용할 수 있으며, 이를 "CSS 테이블 용법"이라고 설명하기도 합니다.

아래 예시는 이러한 사용법을 보여줍니다. 레이아웃에 CSS 테이블을 사용하는 것은 현재로서는 레거시 방법으로 간주되어야 하며, 플렉스박스 또는 그리드를 지원하지 않는 구형 브라우저를 지원할 때만 사용해야 한다는 점에 유의해야 합니다.

예제를 살펴보겠습니다. 우선, HTML 양식을 만드는 간단한 마크업입니다. 각 입력 요소에는 레이블이 있으며, 단락 안에 캡션도 포함했습니다. 각 레이블/입력 요소 쌍은 레이아웃을 위해 [<div> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/div) 요소에 둘러싸여 있습니다.

HTMLPlayCopy to Clipboard

<form>

<p>우선, 이름과 나이를 말씀해주세요</p>

<div>

<label for="이름">이름:</label>

<input type="text" id="이름" />

</div>

<div>

<label for="성">성:</label>

<input type="text" id="성" />

</div>

<div>

<label for="나이">나이:</label>

<input type="text" id="나이" />

</div>

</form>

CSS의 경우 [display (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/display) 속성을 사용하는 것을 제외하면 대부분 평범합니다. [<form>](https://developer.mozilla.org/ko/docs/Web/HTML/Element/form)과 [<div> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/div), [<label>](https://developer.mozilla.org/ko/docs/Web/HTML/Element/label), [<input> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input)은 각각 테이블, 테이블 행, 테이블 셀처럼 표시되도록 지정되었습니다. 기본적으로 HTML 테이블 마크업처럼 작동하여 기본적으로 레이블과 입력이 멋지게 정렬됩니다. 그런 다음 약간의 크기 조정, 여백 등을 추가하여 모든 것이 좀 더 멋지게 보이도록 하면 완료됩니다.

캡션 단락에 display: table-caption;을 지정하여 테이블 [<caption>](https://developer.mozilla.org/ko/docs/Web/HTML/Element/caption)처럼 동작하도록 하고, 또한 캡션이 소스의 <input> 요소 앞에 있지만 스타일 지정 목적으로 캡션이 테이블 하단에 위치하도록 하기 위해 caption-side: bottom;이 지정되어 있음을 알 수 있습니다. 이를 통해 약간의 유연성을 확보할 수 있습니다.

CSSPlayCopy to Clipboard

html {

font-family: sans-serif;

}

form {

display: table;

margin: 0 auto;

}

form div {

display: table-row;

}

form label,

form input {

display: table-cell;

margin-bottom: 10px;

}

form label {

width: 200px;

padding-right: 5%;

text-align: right;

}

form input {

width: 300px;

}

form p {

display: table-caption;

caption-side: bottom;

width: 300px;

color: #999;

font-style: italic;

}

이렇게 하면 다음과 같은 결과가 나옵니다.

Play

이 예제는 [CSS 테이블 예제.html](https://mdn.github.io/learning-area/css/styling-boxes/box-model-recap/css-tables-example.html)에서 실시간으로 볼 수 있고, [소스 코드](https://github.com/mdn/learning-area/blob/master/css/styling-boxes/box-model-recap/css-tables-example.html)도 참조할 수 있습니다.

**참고:** 이 페이지의 다른 주제와 달리 표 레이아웃은 레거시 적용으로 인해 이 과정에서 더 이상 다루지 않습니다.

[**다단 레이아웃**](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Introduction#%EB%8B%A4%EB%8B%A8_%EB%A0%88%EC%9D%B4%EC%95%84%EC%9B%83)

다단 레이아웃 CSS 모듈은 신문에서 텍스트가 흐르는 방식과 유사하게 콘텐츠를 열로 배치하는 방법을 제공합니다. 사용자가 위아래로 스크롤해야 하기 때문에 웹 환경에서는 열을 위아래로 읽는 것이 덜 유용하지만, 그럼에도 불구하고 콘텐츠를 열로 배열하는 것은 유용한 기술이 될 수 있습니다.

한 블록을 다단 컨테이너로 바꾸려면 [column-count (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/column-count) 속성을 사용하여, 브라우저에 원하는 열 수를 지정하거나 [column-width (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/column-width) 속성을 사용하여 지정된 너비의 열로 컨테이너를 최대한 많이 채우도록 지시합니다.

아래 예제에서는 container 클래스에 해당하는 <div> 요소를 포함하는 HTML 블록으로 시작합니다.

HTMLPlayCopy to Clipboard

<div class="container">

<h1>다단 레이아웃</h1>

<p>단락 1.</p>

<p>단락 2.</p>

</div>

해당 컨테이너에 200픽셀의 열 너비를 사용하므로, 브라우저에서 200픽셀 열을 최대한 많이 만들게 됩니다. 열 사이에 남은 공간은 모두 공유됩니다.

CSSPlayCopy to Clipboard

.container {

column-width: 200px;

}

Play

[**요약**](https://developer.mozilla.org/ko/docs/Learn/CSS/CSS_layout/Introduction#%EC%9A%94%EC%95%BD)

이 글에서는 알아야 할 모든 레이아웃 기술에 대해 간략하게 요약했습니다. 각 개별 기술에 대한 자세한 내용을 계속 읽어보세요!