🎨

CSS 기초 강의(3) - 애니메이션, 트랜지션, 반응형 웹

CSS 기초 강의(2)을 통해 CSS 기초 중 반드시 알아야하는 인라인/블록 속성과 박스모델에 대해 학습했습니다. 이 문서를 통해 애니메이션, 트랜지션, 반응형 웹에 대해 설명합니다.

CSS 애니메이션

CSS는 애니메이션 효과를 적용하는 기능을 제공합니다. 이를 이용하면 콘텐츠나 요소들을 더욱 생동감 있게 만들 수 있습니다. 애니메이션을 적용하기 위해서는 다음과 같은 속성을 이용합니다.

```
/* 애니메이션 속성 */
animation-name: move; /* 애니메이션 이름 */
animation-duration: 3s; /* 애니메이션 지속 시간 */
animation-timing-function: ease-in-out; /* 애니메이션의 타이밍 함수 */
animation-delay: 1s; /* 애니메이션 시작까지의 지연 시간 */
animation-iteration-count: infinite; /* 애니메이션 반복 횟수 */
animation-direction: alternate; /* 애니메이션 반복 방향 */
animation-fill-mode: forwards; /* 애니메이션 끝나고 상태 유지 */
```

위의 속성들을 이용하여 요소에 애니메이션 효과를 적용할 수 있습니다. 예를 들어, 다음과 같은 HTML과 CSS를 이용하여 요소를 왼쪽에서 오른쪽으로 이동하는 애니메이션을 적용할 수 있습니다.

```
<div class="box"></div>
```

```
.box {
  width: 100px;
  height: 100px;
  background-color: blue;
  animation-name: move;
  animation-duration: 2s;
  animation-timing-function: ease-in-out;
  animation-iteration-count: infinite;
}

@keyframes move {
  0% { transform: translateX(-200px); }
  50% { transform: translateX(0); }
  100% { transform: translateX(200px); }
}
```

위의 예시에서는 box 클래스를 가진 div 요소를 이용하여 애니메이션을 적용하였습니다. 이 애니메이션은 `@keyframes` 를 이용하여 정의하였습니다. 0%에서는 요소를 왼쪽에서 이동시키고, 50%에서는 멈추고, 100%에서는 오른쪽으로 이동시키도록 정의하였습니다.

CSS Transition

CSS의 transition은 요소의 스타일 변화를 부드럽게 처리하는 기능입니다. 예를 들어, 마우스 오버 효과를 부드럽게 처리하거나, 메뉴를 열고 닫을 때 스타일 변화를 부드럽게 처리하는 등 다양한 용도로 사용됩니다.

```
/* transition 속성 */
.box {
  transition: background-color 1s ease-in-out;
}
.box:hover {
  background-color: red;
}
```

위의 예시는 .box 클래스를 가진 요소에 transition을 적용하여 마우스 오버 시 배경색이 부드럽게 빨간색으로 변화하는 효과를 부여하는 방법입니다. `transition: background-color 1s ease-in-out;`는 배경색이 변경될 때 1초 동안 부드럽게 처리하는 것을 의미합니다. `ease-in-out`은 변

화가 시작할 때는 느리게, 끝나는 시점에는 다시 느리게 처리하여 부드러운 효과를 부여합니다.

transition 속성은 다양한 속성에 적용될 수 있습니다. 예를 들어, `color`, `transform`, `opacity` 등 다양한 속성에 적용하여 요소의 스타일 변화를 부드럽게 처리할 수 있습니다.

반응형 웹

반응형 웹은 다양한 기기에서 웹사이트를 이용할 때, 화면의 크기에 맞추어 적절하게 조정되는 웹사이트를 말합니다. 이를 위해서는 다음과 같은 기술과 방법들이 사용됩니다.

1. 미디어 쿼리(Media Queries)

미디어 쿼리는 화면의 크기와 방향 등에 따라 다른 CSS 스타일을 적용하는 방법입니다. 예를 들어, 모바일 기기에서는 화면이 작으므로 폰트 크기나 이미지 크기를 작게 하는 등 다양한 스타일을 적용할 수 있습니다.

```
/* 미디어 쿼리 */
@media (max-width: 600px) {
  /* 화면이 600px 이하일 때 적용되는 스타일 */
  body {
    font-size: 12px;
  }
}
```

위의 예시는 화면이 600px 이하일 때 body 요소의 폰트 크기를 12px로 조정하는 미디어 쿼리입니다.

2. 유동 그리드(Grid)

유동 그리드는 요소의 크기와 위치를 유연하게 조정하여 화면 크기에 맞춰 배치하는 방법입니다. 이를 이용하면 화면 크기에 따라 요소의 위치와 크기를 적절하게 조정하여 레이아웃을 구성할 수 있습니다.

```
/* 유동 그리드 */
.container {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
  grid-gap: 10px;
}
```

위의 예시는 .container 클래스를 가진 요소를 유동 그리드로 구성하는 방법입니다. `grid-template-columns` 속성을 이용하여 열의 개수와 너비를 정의하였습니다. `repeat(auto-fit, minmax(200px, 1fr))`는 자동으로 열의 개수를 조정하면서 최소 너비 200px을 유지하는 것을 의미합니다. `grid-gap` 속성을 이용하여 요소 간의 간격을 조정하였습니다.

그리드는 Grid Froggy(https://cssgridgarden.com/)에서 연습해보면 좋습니다!

추천 아티클: heropy(grid, flex)

추천 과제: 계산기의 숫자패드 부분을 그리드로 만들어보기.

3. 뷰포트(Viewport)

뷰포트는 사용자가 웹사이트를 볼 때 실제로 보이는 영역을 말합니다. 반응형 웹을 구현할 때는 뷰포트의 크기에 맞춰서 스타일을 조정해야 합니다.

```
<!-- 뷰포트 설정 -->
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

위의 예시는 뷰포트를 설정하는 방법입니다. `width=device-width`는 뷰포트의 너비를 기기의 너비와 같게 설정하는 것을 의미합니다. `initial-scale=1.0`은 초기 화면 배율을 1로 설정하는 것을 의미합니다.

위와 같은 기술과 방법들을 이용하여 반응형 웹을 구현할 수 있습니다. 이를 통해 사용자는 다양한 기기에서도 웹사이트를 적절하게 이용할 수 있습니다.