7장
효과적인 레이아웃을 위한 CSS 속성 다루기

7.1 플렉스 박스 레이아웃으로 1차원 레이아웃 설계하기

7.2 그리드 레이아웃으로 2차원 레이아웃 설계하기

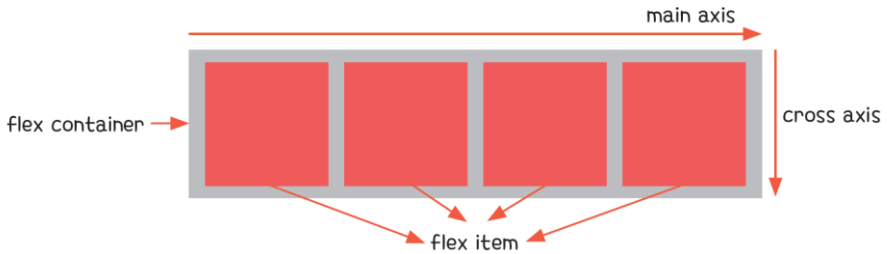
7.3 반응형 웹을 위한 미디어 쿼리 사용하기

7.1 플렉스 박스 레이아웃으로 1차원 레이아웃 설계하기

1. 플렉스 박스 레이아웃 살펴보기

- 1차원 방식으로 효과적으로 레이아웃을 설계할 수 있도록 고안된 스타일

그림 7-1 플렉스 박스 레이아웃의 구성 요소

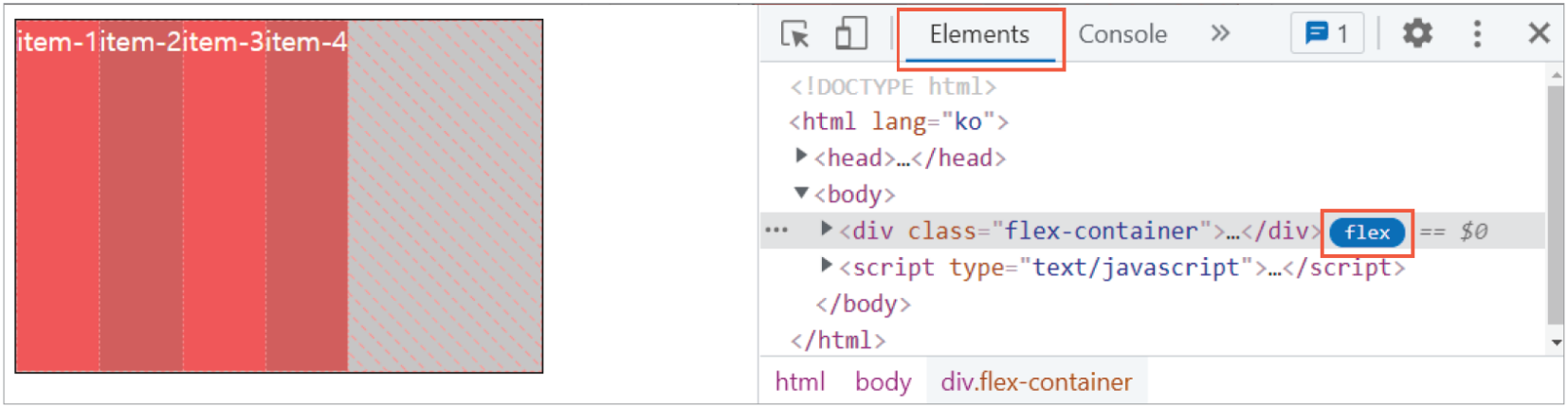


- 구성 요소
  - 주축(main axis) : 플렉스 박스의 진행 방향과 수평한 축
  - 교차축(cross axis) : 주축과 수직하는 축
  - 플렉스 컨테이너(flex container) : display 속성값으로 flex나 inline-flex가 적용된 요소
  - 플렉스 아이템(flex item) : 플렉스 컨테이너와 자식 관계를 이루는 태그 요소

7.1 플렉스 박스 레이아웃으로 1차원 레이아웃 설계하기

- 레이아웃 확인 방법
  - 개발자 도구의 Elements 탭 > flex 아이콘 클릭

그림 7-2 크롬 브라우저에서 플렉스 박스 레이아웃 확인



7.1 플렉스 박스 레이아웃으로 1차원 레이아웃 설계하기

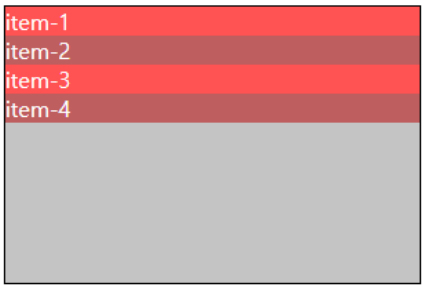
- 레이아웃 확인 예제

Html 코드

```html
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>flex basic</title>
  <link rel="stylesheet" href="flex-basic.css">
</head>
<body>
  <div class="flex-container">
    <div class="flex-item">item-1</div>
    <div class="flex-item">item-2</div>
    <div class="flex-item">item-3</div>
    <div class="flex-item">item-4</div>
  </div>
</body>
</html>
```

css 코드

```css
.flex-container{
  width:300px;
  height:200px;
  background-color: #c4c4c4;
  border:1px solid black;
}
.flex-item{
  color: white;
  background-color: #ff5252; /* 기본 배경 색상 */
}
.flex-item:nth-child(2n){
  background-color: #bf5e5e;
}
```



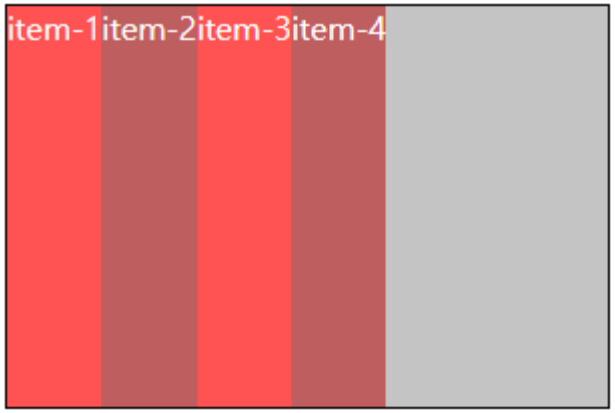
7.1 플렉스 박스 레이아웃으로 1차원 레이아웃 설계하기

2. 플렉스 박스 레이아웃의 기본 속성

- display 속성 : flex, inline-flex 값을 지정하면 해당 요소가 플렉스 컨테이너로 설정

형식 `display:flex; /* inline-flex */`

```
  <link rel="stylesheet" href="flex-basic.css">
  <style>
    .flex-container{
      display:flex;
    }
  </style>
</head>
<body>
  <div class="flex-container">
    <div class="flex-item">item-1</div>
    <div class="flex-item">item-2</div>
    <div class="flex-item">item-3</div>
    <div class="flex-item">item-4</div>
```



7.1 플렉스 박스 레이아웃으로 1차원 레이아웃 설계하기

- flex-direction 속성 : 플렉스 박스 레이아웃의 주축 방향을 지정

  형식 `flex-direction:<속성값>;`

  - row : 주축 방향을 왼쪽에서 오른쪽으로 지정
  - row-reverse : 주축 방향을 오른쪽에서 왼쪽으로 지정
  - column : 주축 방향을 위쪽에서 아래쪽으로 지정
  - column-reverse : 주축 방향을 아래쪽에서 위쪽으로 지정

7.1 플렉스 박스 레이아웃으로 1차원 레이아웃 설계하기

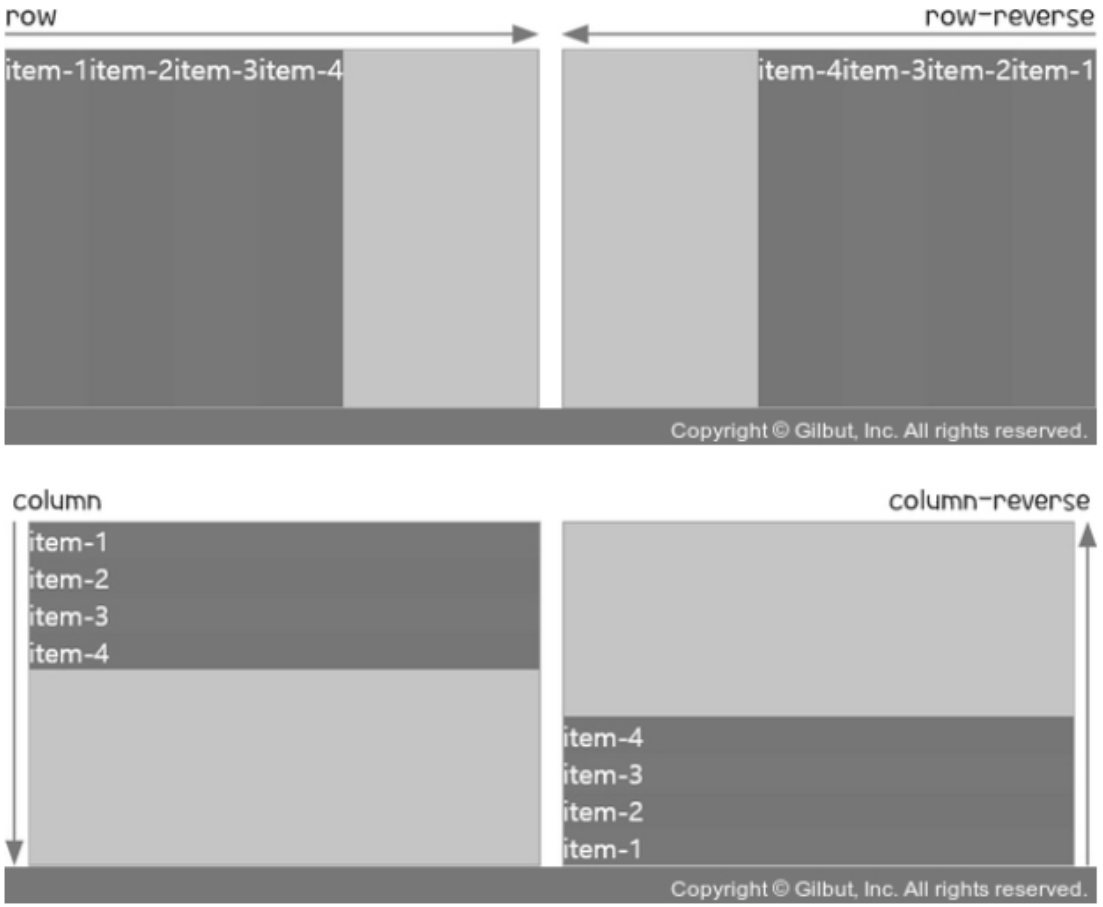


그림 7-5 flex-direction 속성값에 따른 주축 방향과 플렉스 아이템 배치

7.1 플렉스 박스 레이아웃으로 1차원 레이아웃 설계하기

- flex-wrap 속성 : 플렉스 아이템이 플렉스 컨테이너 영역을 벗어날 때 어떻게 처리할지 지정

  형식 `flex-wrap:<속성값>;`

  - nowrap : 플렉스 아이템이 플렉스 컨테이너를 벗어나도 무시
  - wrap : 플렉스 아이템이 플렉스 컨테이너를 벗어나면 줄 바꿈
  - wrap-reverse : 플렉스 아이템이 플렉스 컨테이너를 벗어나면 wrap의 역방향으로 줄 바꿈

7.1 플렉스 박스 레이아웃으로 1차원 레이아웃 설계하기

norap일때

wrap일때

Wrap-reverse일때

7.1 플렉스 박스 레이아웃으로 1차원 레이아웃 설계하기

- flex-flow 속성 : flex-direction과 fled-wrap 속성을 한 번에 사용할 수 있는 단축 속성

형식 flex-flow:<flex-direction> <flex-wrap>;

```
<style>
  .flex-container{
    display:flex;
    flex-direction: column;
    flex-wrap: nowrap;
  }
</style>
```

```
<style>
  .flex-container{
    display:flex;
    flex-flow:column nowrap;
  }
</style>
```

7.1 플렉스 박스 레이아웃으로 1차원 레이아웃 설계하기

3. 플렉스 박스 레이아웃의 정렬 속성

- justify-content 속성 :플렉스 아이템을 모두 주축 방향으로 정렬

  형식 `justify-content:<속성값>;`

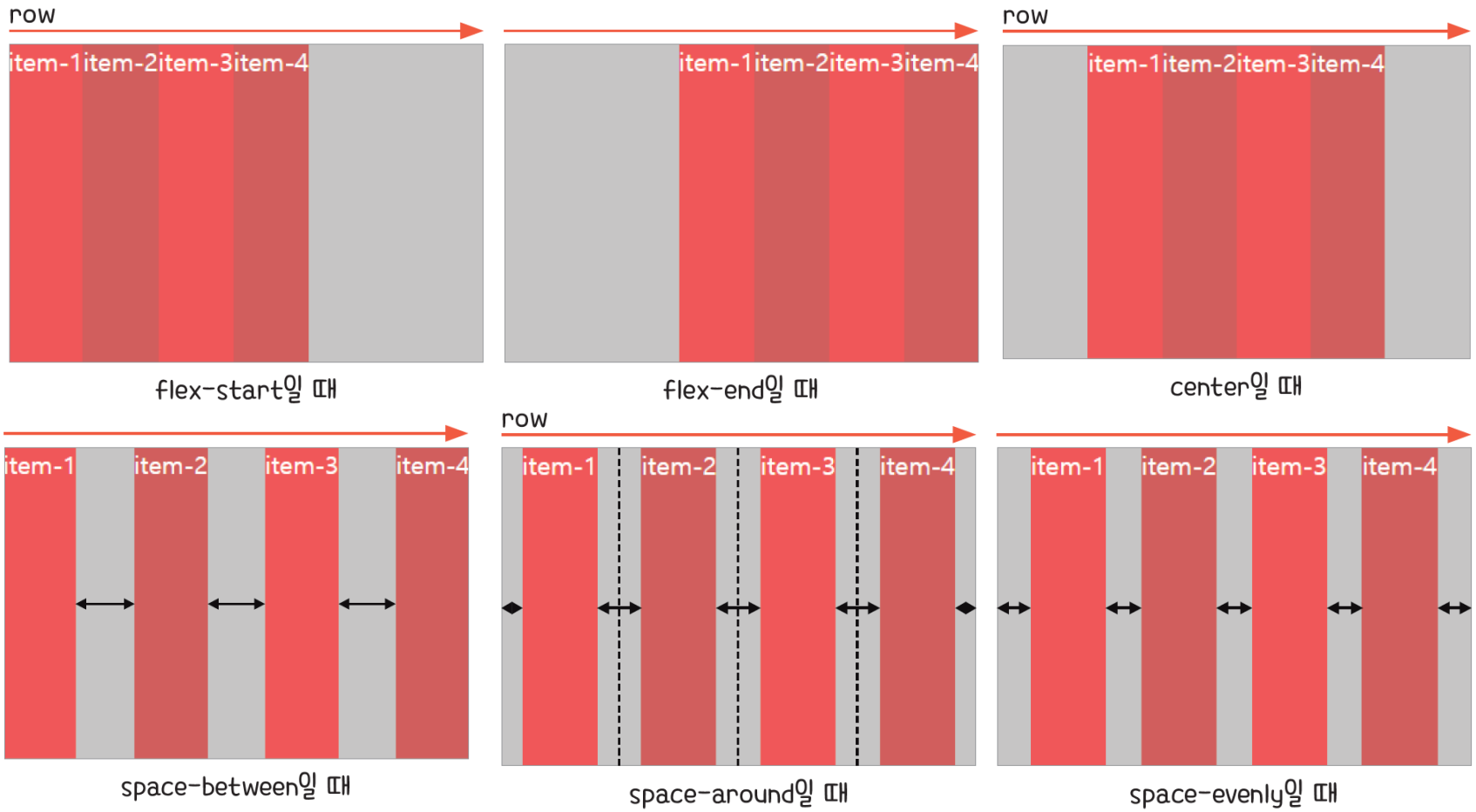
  - flex-start : 주축 방향의 시작을 기준으로 정렬
  - flex-end : 주축 방향의 끝을 기준으로 정렬
  - center : 주축 방향의 중앙에 정렬
  - space-between : 플렉스 아이템 사이의 간격이 균일하도록 정렬
  - space-around : 플렉스 아이템의 둘레(around)가 균일하도록 정렬
  - space-evenly : 플렉스 아이템 사이와 양끝의 간격이 균일하도록 정렬

7.1 플렉스 박스 레이아웃으로 1차원 레이아웃 설계하기

```
  <style>
    .flex-container{
      display:flex;
      justify-content:flex-start; /* flex-end, center, space-between, space-around, space-evenly */
    }
  </style>
</head>
<body>
  <div class="flex-container">
    <div class="flex-item">item-1</div>
    <div class="flex-item">item-2</div>
    <div class="flex-item">item-3</div>
    <div class="flex-item">item-4</div>
  </div>
```

7.1 플렉스 박스 레이아웃으로 1차원 레이아웃 설계하기

그림 7-8 주축의 방향이 row일 때 justify-content 속성값에 따른 정렬 결과



7.1 플렉스 박스 레이아웃으로 1차원 레이아웃 설계하기

- align-items 속성 : 플렉스 아이템을 교차축 방향으로 정렬

  형식 `align-items:<속성값>;`

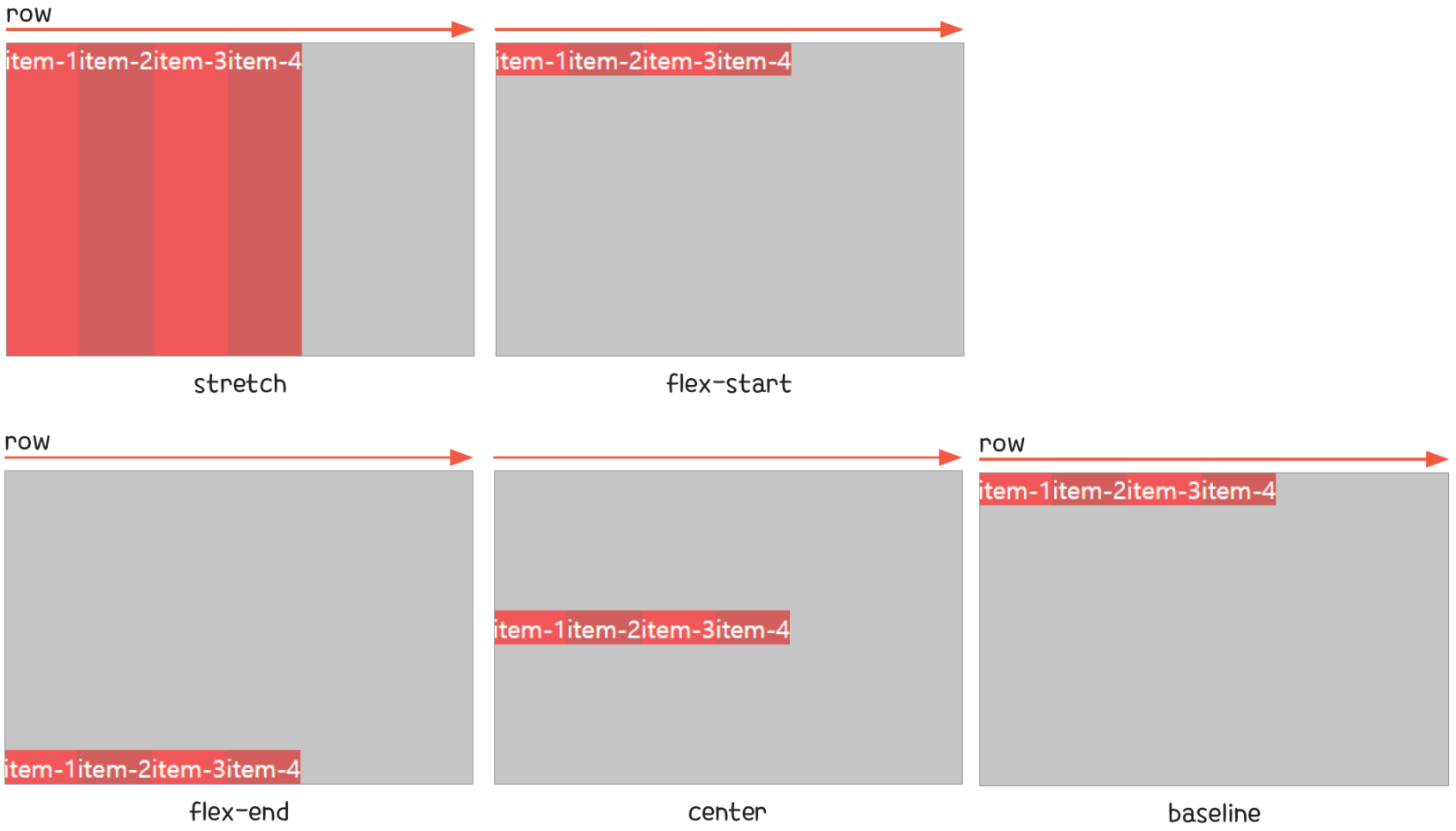
  - stretch : 교차축 방향으로 플렉스 아이템의 너비나 높이가 늘어남
  - flex-start : 교차축 방향의 시작을 기준으로 정렬
  - flex-end : 교차축 방향의 끝을 기준으로 정렬
  - center : 교차축 방향의 중앙을 기준으로 정렬
  - baseline : 플렉스 아이템의 baseline을 기준으로 정렬

7.1 플렉스 박스 레이아웃으로 1차원 레이아웃 설계하기

```
  <style>
    .flex-container{
      display:flex;
      align-items:stretch; /* flex-start, flex-end, center, baseline */
    }
  </style>
</head>
<body>
  <div class="flex-container">
    <div class="flex-item">item-1</div>
    <div class="flex-item">item-2</div>
    <div class="flex-item">item-3</div>
    <div class="flex-item">item-4</div>
  </div>
```

7.1 플렉스 박스 레이아웃으로 1차원 레이아웃 설계하기

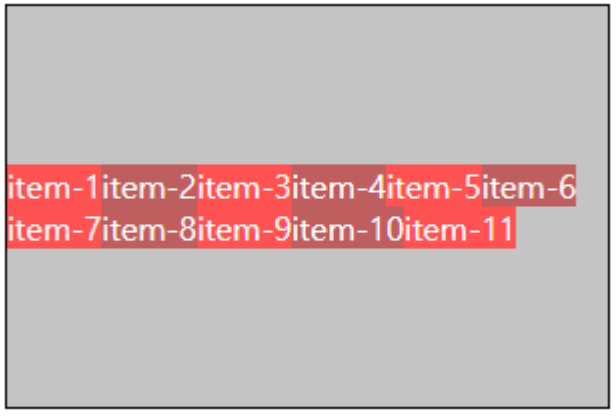
그림 7-9 주축의 방향이 row일 때, align-items 속성값에 따른 정렬 결과



7.1 플렉스 박스 레이아웃으로 1차원 레이아웃 설계하기

- align-content 속성 : 플렉스 아이템이 두 줄 이상일 때 교차축 방향으로 정렬(flex-wrap 속성때문)

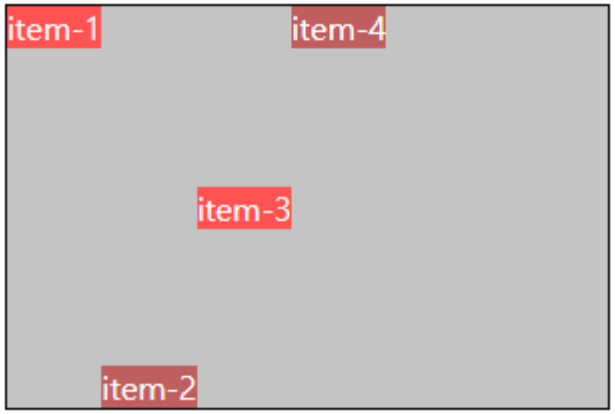
```
  <style>
    .flex-container{
      display:flex;
      flex-wrap:wrap;
      align-content:center;
    }
  </style>
</head>
<body>
  <div class="flex-container">
    <div class="flex-item">item-1</div>
    <div class="flex-item">item-2</div>
    <div class="flex-item">item-3</div>
    <div class="flex-item">item-4</div>
    <div class="flex-item">item-5</div>
    <div class="flex-item">item-6</div>
    <div class="flex-item">item-7</div>
    <div class="flex-item">item-8</div>
    <div class="flex-item">item-9</div>
    <div class="flex-item">item-10</div>
    <div class="flex-item">item-11</div>
  </div>
```



7.1 플렉스 박스 레이아웃으로 1차원 레이아웃 설계하기

- align-self 속성 : align-items 속성으로 플렉스 아이템을 한 번에 정렬하지 않고 각각 정렬하고 싶을 때 사용

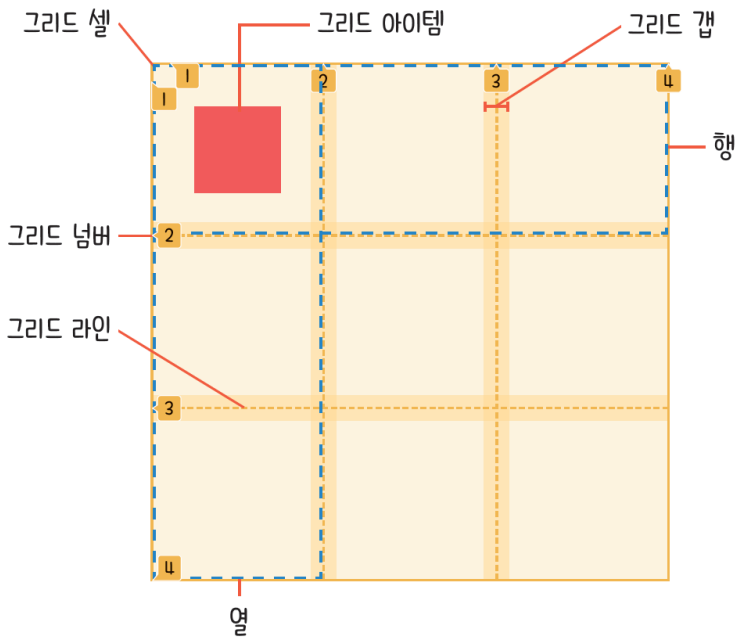
```
  <style>
    .flex-container{
      display:flex;
    }
    .flex-item:nth-child(1){
      align-self:flex-start;
    }
    .flex-item:nth-child(2){
      align-self:flex-end;
    }
    .flex-item:nth-child(3){
        align-self:center;
    }
    .flex-item:nth-child(4){
      align-self:baseline;
    }
  </style>
</head>
<body>
  <div class="flex-container">
    <div class="flex-item">item-1</div>
    <div class="flex-item">item-2</div>
    <div class="flex-item">item-3</div>
    <div class="flex-item">item-4</div>
  </div>
```



7.2 그리드 레이아웃으로 2차원 레이아웃 설계하기

1. 그리드 레이아웃 살펴보기

그림 7-10 그리드 레이아웃 구성 요소

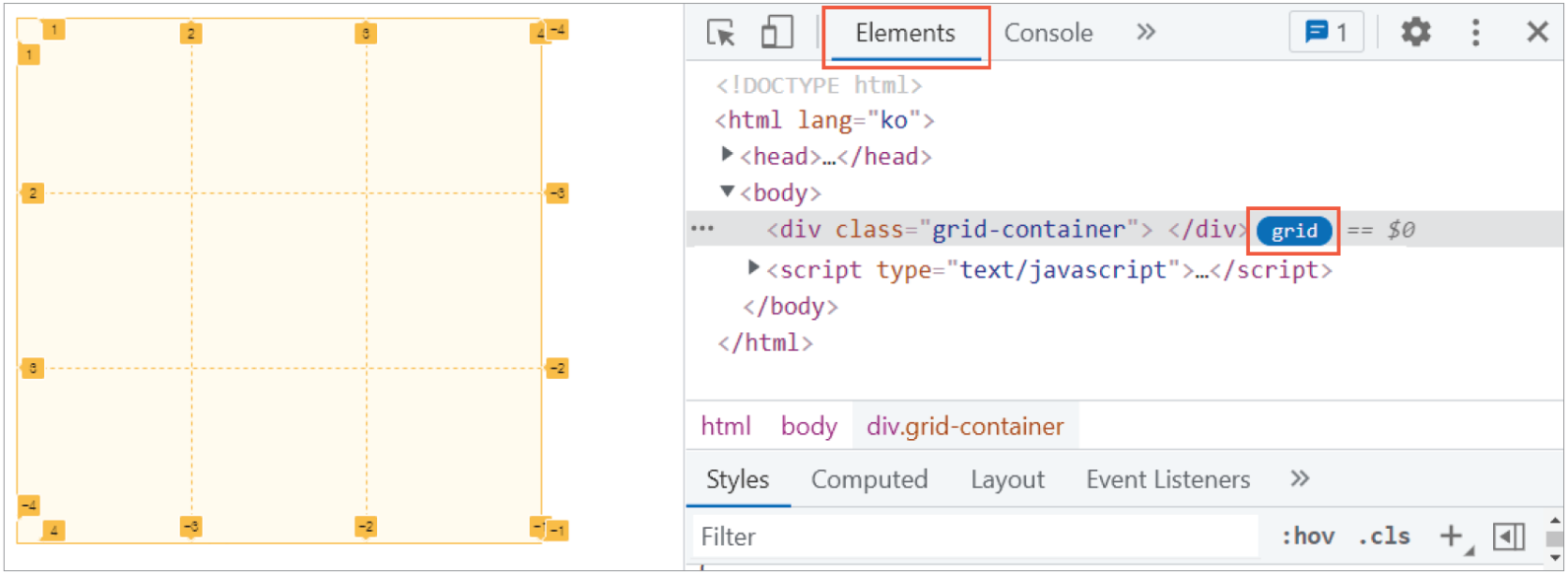


- 행(row) : 그리드 레이아웃의 가로줄
- 열(column) : 그리드 레이아웃의 세로줄
- 그리드 셀(grid cell) : 행과 열이 만나서 이루어지는 하나의 공간
- 그리드 갭(grid gap) : 그리드 셀과 그리드 셀 사이의 간격
- 그리드 아이템(grid item) : 그리드 셀 안에 포함되는 콘텐츠
- 그리드 라인(grid line) : 그리드 행과 열을 그리는 선
- 그리드 넘버(grid number) : 그리드 라인에 붙는 번호
- 그리드 컨테이너(grid container) : 그리드 아이템을 묶는 부모 요소

7.2 그리드 레이아웃으로 2차원 레이아웃 설계하기

- 레이아웃 확인 방법
  - 개발자 도구의 Elements 탭 > grid 아이콘 클릭

그림 7-11 크롬 브라우저에서 그리드 레이아웃 확인



7.2 그리드 레이아웃으로 2차원 레이아웃 설계하기

2. 그리드 레이아웃의 기본 속성

- display 속성 : 속성값을 grid, inline-grid로 지정하면 그리드 레이아웃을 만들 수 있음

  형식 `display:grid; /* inline-grid */`

- grid-template-columns와 grid-template-rows 속성 : 그리드 레이아웃의 행과 열을 지정

  형식 `grid-template-columns:<1열값> <2열값> ...;`

  `grid-template-rows:<1행값> <2행값> ...;`

- row-gap과 column-gap 속성 : 그리드 셀과 셀 사이의 간격을 지정

  형식 `row-gap:<크기>;`

  `column-gap:<크기>;`

7.2 그리드 레이아웃으로 2차원 레이아웃 설계하기

3. 그리드 레이아웃의 정렬 속성

- align-items 속성 : 그리드 아이템 전체를 셀의 세로 방향으로 정렬
  - stretch : 그리드 아이템이 그리드 셀을 꽉 채우도록 크기를 늘림
  - start : 그리드 아이템을 그리드 셀의 맨 위에 배치
  - center : 그리드 아이템을 그리드 셀의 세로 방향 중간에 배치
  - end : 그리드 아이템을 그리드 셀의 맨 아래에 배치
- align-self 속성 : 각각의 그리드 아이템을 셀의 세로 방향으로 정렬

7.2 그리드 레이아웃으로 2차원 레이아웃 설계하기

- justify-items 속성 : 그리드 아이템 전체를 셀의 가로 방향으로 정렬
  - stretch : 그리드 아이템을 그리드 셀이 꽉 차도록 늘림
  - start : 그리드 아이템을 그리드 셀의 왼쪽 끝에 배치
  - center : 그리드 아이템을 그리드 셀의 가로 방향 중간에 배치
  - end : 그리드 아이템을 그리드 셀의 오른쪽 끝에 배치
- justify-self 속성 : 각각의 그리드 아이템을 셀의 가로 방향으로 정렬

7.2 그리드 레이아웃으로 2차원 레이아웃 설계하기

- place-item 속성 : align-items와 justify-items 속성을 한 번에 사용할 수 있는 단축 속성

  형식 `place-items:<align-items> <justify-items>;`

- place-self 속성 : align-self와 justify-self 속성을 한 번에 사용할 수 있는 단축 속성

  형식 `place-self:<align-self> <justify-self>;`

  예

```
place-items:center end; /* align-items:center, justify-items:end */
place-self:center end; /* align-self:center, justify-self:end */
```

7.2 그리드 레이아웃으로 2차원 레이아웃 설계하기

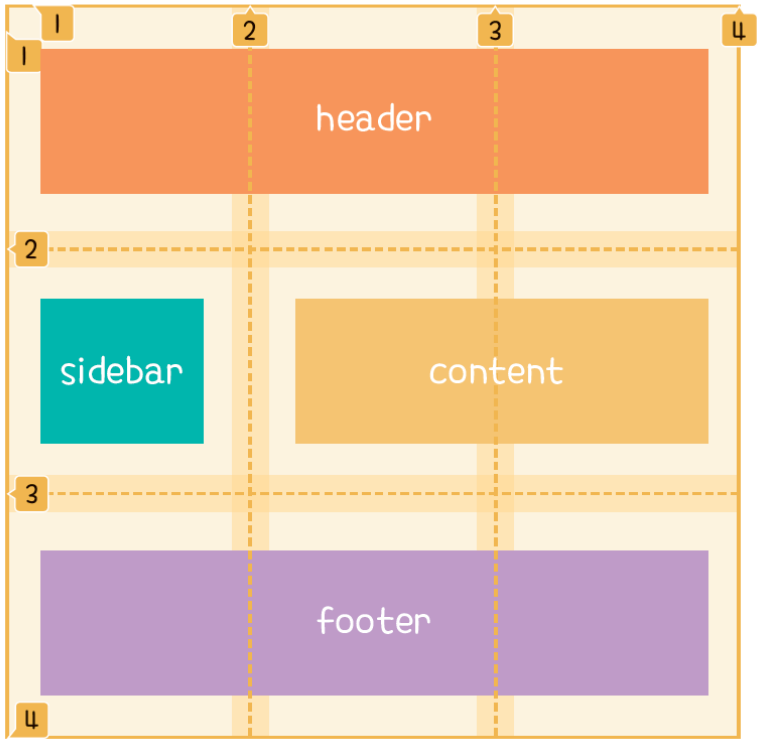
4. 그리드 레이아웃의 배치 속성

- grid-template-areas 속성 : 그리드 레이아웃에서 행과 열을 이름으로 지정
- grid-area 속성 : 그리드 아이템에 이름을 지정

형식 grid-area:<행과 열 이름>;

```
<style>
  .grid-container{
    display:grid;
    grid-template-areas:
    "header header header"
    "sidebar content content"
    "footer footer footer";
  }
  #header{
    grid-area:header;
  }
  #sidebar{
    grid-area:sidebar;
  }
```

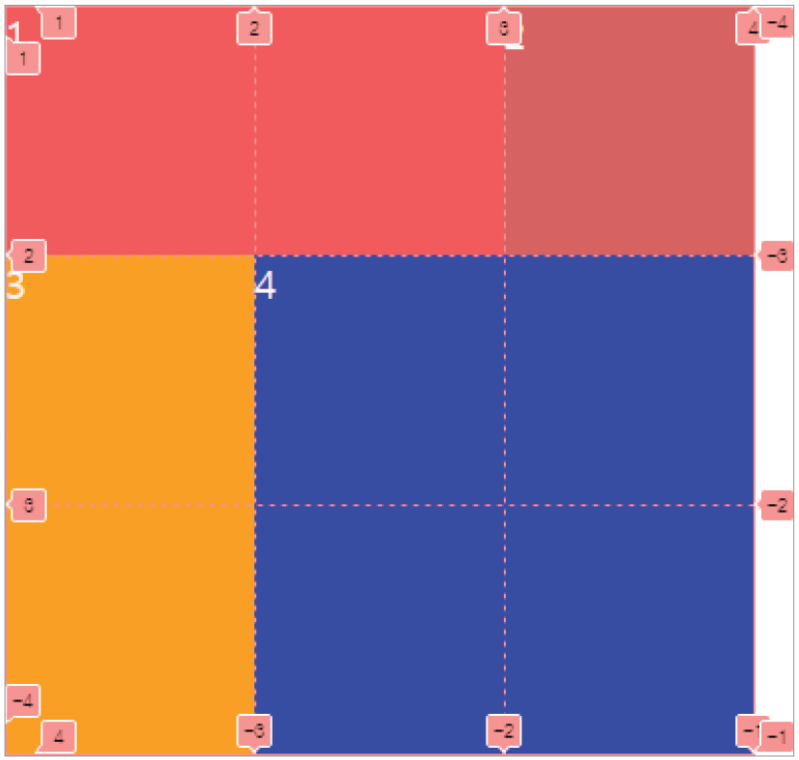
그림 7-18 그리드 레이아웃 예시



7.2 그리드 레이아웃으로 2차원 레이아웃 설계하기

- grid-column-start, grid-column-end 속성 : 그리드 레이아웃에서 열의 시작 번호와 끝 번호를 지정
- grid-row-start, grid-row-end 속성 : 그리드 레이아웃에서 행의 시작과 끝 번호를 지정
- 그리드 라인 : 그리드 컨테이너를 구성하는 행과 열을 그리는 선
- 그리드 넘버 : 그리드 라인에 있는 고유한 번호

그림 7-19 그리드 넘버



7.2 그리드 레이아웃으로 2차원 레이아웃 설계하기

- grid-column 속성 : grid-column-start와 grid-column-end 속성을 한 번에 사용할 수 있는 단축 속성
- grid-row 속성 : grid-row-start와 grid-row-end 속성을 한 번에 사용할 수 있는 단축 속성

형식 grid-column:<start> <end>;

grid-row:<start> <end>;

형식 grid-column:<start>/span <열 개수>;

grid-row:<start>/span <행 개수>;

7.3 반응형 웹을 위한 미디어 쿼리 사용하기

1. 미디어 쿼리란

- 사이트에 접속하는 미디어 타입과 특징, 해상도에 따라 다른 스타일 속성을 적용할 수 있게 하는 기술

그림 7-20 모질라 사이트의 반응형 웹

7.3 반응형 웹을 위한 미디어 쿼리 사용하기

2. 뷰포트 알아보기

- 뷰포트 : 웹 페이지가 접속한 기기에서 보이는 실제 영역의 크기

그림 7-22 해상도가 작은 기기에서 보이는 화면

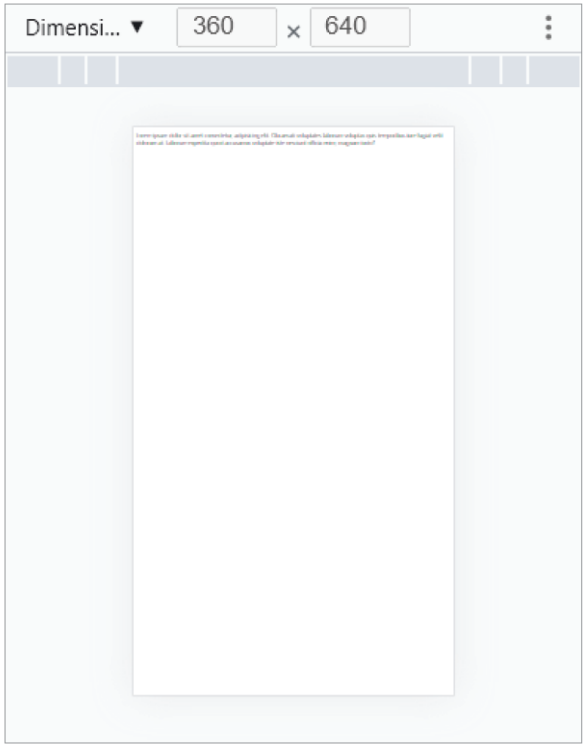


그림 7-23 뷰포트 적용 후 화면

7.3 반응형 웹을 위한 미디어 쿼리 사용하기

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

표 7-7 content 속성값

속성값	설명
width	뷰포트의 너비를 설정합니다. 보통 device-width로 설정합니다.
height	뷰포트의 높이를 설정합니다. 잘 사용하지 않습니다.
initial-scale	뷰포트의 초기 배율을 설정합니다. 1이 기본값이며 1보다 작으면 축소 값, 1보다 크면 확대 값으로 설정됩니다.
minimum-scale	뷰포트의 최소 축소 비율을 설정합니다. 기본으로 0.25가 적용되어 있습니다.
maximum-scale	뷰포트의 최대 확대 비율을 설정합니다. 기본으로 5.0이 적용되어 있습니다.
user-scalable	뷰포트의 확대 또는 축소 여부를 설정합니다. yes 또는 no로 지정하는데, no로 지정하면 화면을 확대 또는 축소할 수 없습니다.

7.3 반응형 웹을 위한 미디어 쿼리 사용하기

3. 미디어 쿼리의 기본 문법

형식 `@media <not|only> <media type> and (<media feature>)`

```
    <and|or|not> (<media feature>){
      /* CSS 코드; */
    }
```

- not/only
  - not : 뒤에 오는 모든 조건을 부정
  - only : 미디어 쿼리를 지원하는 기기만

7.3 반응형 웹을 위한 미디어 쿼리 사용하기

- media type : 미디어 쿼리가 적용될 미디어 타입
  - all : 모든 기기
  - print : 인쇄 장치(예: 프린터기)
  - screen : 컴퓨터 화면 장치 또는 스마트 기기
  - speech : 스크린 리더기 같은 보조 프로그램으로 웹 페이지를 소리 내어 읽어 주는 장치

7.3 반응형 웹을 위한 미디어 쿼리 사용하기

- media feature : 미디어 쿼리가 적용될 미디어 조건

- 사용할 수 있는 조건은 18가지이며 실무에서 자주 사용하는 부분은 아래와 같다.

  - min-width <화면 너비> : 미디어 쿼리가 적용될 최소 너비
  - max-width <화면 너비> : 미디어 쿼리가 적용될 최대 너비
  - orientation portrait : 세로 모드, 뷰포트의 세로 높이가 가로 너비보다 큰 경우
  - orientation landscape : 가로 모드, 뷰포트의 가로 너비가 세로 높이보다 큰 경우

7.3 반응형 웹을 위한 미디어 쿼리 사용하기

```
@media only screen and (min-width:360px) and (max-width:720px){
  /* CSS 코드 */
}
```

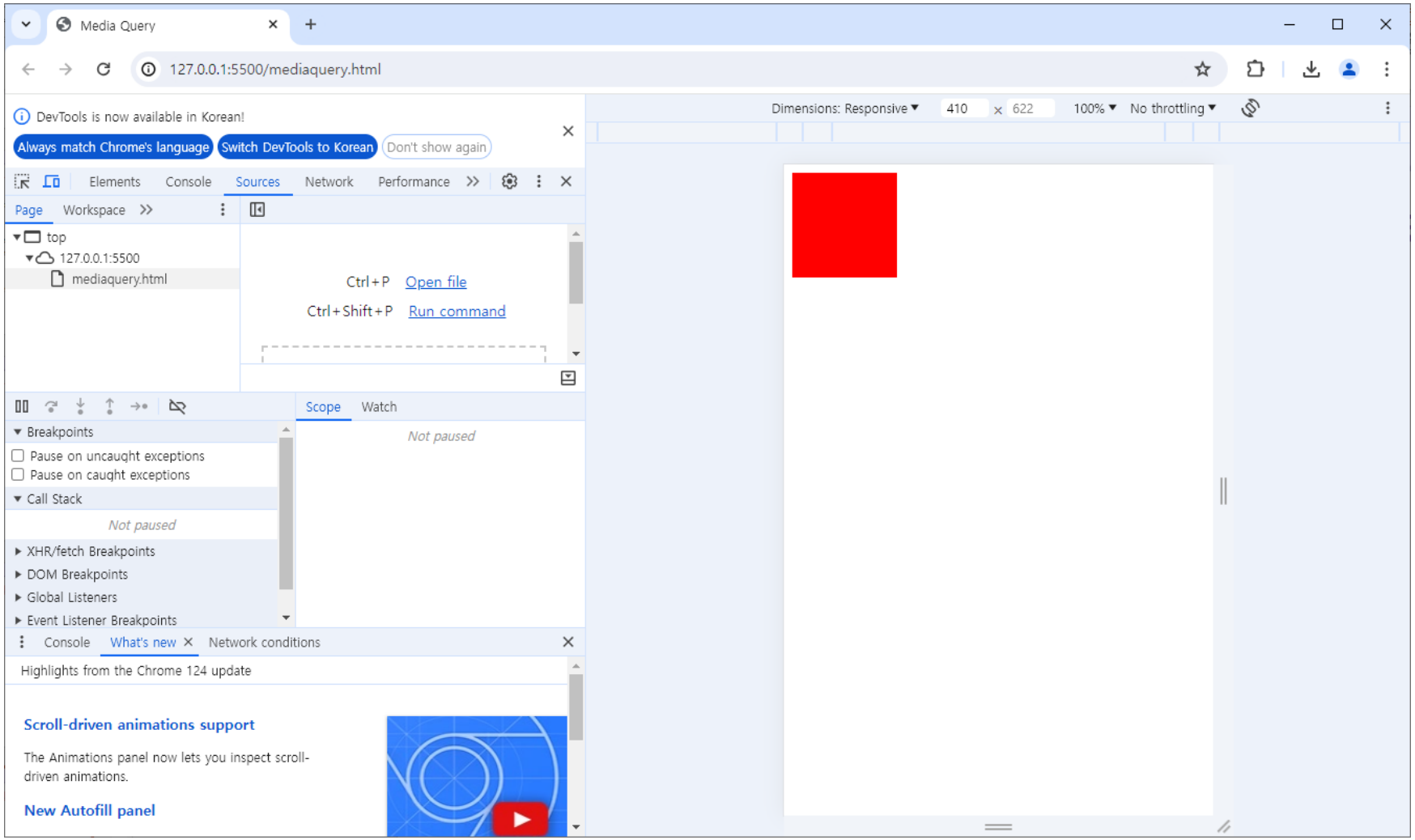
```
@media (min-width:360px) and (max-width:720px){
  /* CSS 코드 */
}
```

```
@media only all and (min-width:360px) {
  /* CSS 코드 */
}
```

7.3 반응형 웹을 위한 미디어 쿼리 사용하기 과제

```html
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Media Query</title>
  <style>
    div{
      width:100px;
      height:100px;
      background-color: red;
    }
    @media only screen and (min-width: 420px){
      div{
        background-color: blue;
      }
    }
  </style>
</head>
<body>
  <div></div>
</body>
</html>
```

7.3 반응형 웹을 위한 미디어 쿼리 사용하기 과제



7.3 반응형 웹을 위한 미디어 쿼리 사용하기 과제