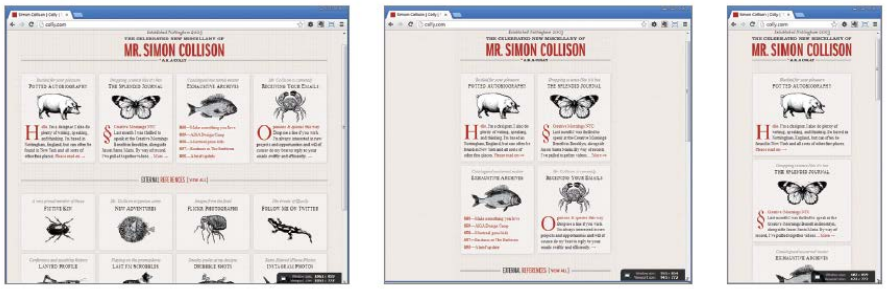
14장 반응형 웹

14.1 모바일 기기와 웹 디자인

(1) 반응형 웹 디자인

- 스마트폰이나 태블릿, 스마트 TV처럼 다양해지는 사용자 브라우저 환경에 따라 그때마다 웹 사이트를 따로 제작하는 데는 한계가 있다.
- 여러 크기의 브라우저 창에 맞게 사이트를 따로 제작하는 일은 비효율적입니다.
- 원래 웹사이트 내용을 그대로 유지하면서 다양한 화면 크기에 맞게 웹사이트를 표시하도록 해 보자 -> 반응형 웹 디자인(responsive web design)
- 반응형 웹 디자인은 화면 크기에 맞게 화면 요소들을 재배치하고 각 요소의 표시 방법만 다르게 해서 사이트를 구현



반응형 웹 디자인(http://colly.com)

(2) 반응형 웹의 장단점

- 모든 스마트 기기에서 접속 가능
- 가로 모드에 맞추어 레이아웃 변경 가능
- 사이트 유지 관리 용이
- 최신 브라우저에서만 지원된다. 하위 버전의 브라우저 사용자까지 고려할 수 없다는 단점도 있다.

(3) 모바일 기기를 위한 기본 다지기, 뷰포트

- 뷰포트란 스마트폰 화면에서 실제 내용이 표시되는 영역이다.

(4) 뷰포트 지정하기

- 320px 너비로 맞춰 모바일 사이트를 제작해도 스마트폰을 보면 아주 작게 표시됨
  → 모바일 브라우저의 기본 뷰포트 너비 980px
  → 웹 페이지를 무조건 980px 너비로 표시하려고 하기 때문

■ 해결방안: 뷰포트 크기나 배율을 조절해야 한다.

■ 기본형: <meta name="viewport" content="<속성1=값>,<속성2=값>,...">

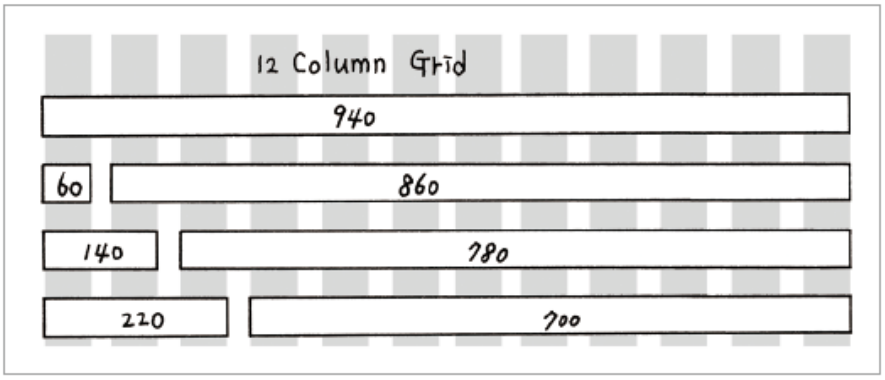
속성	설명	사용 가능한 값	기본 값
width	뷰포트 너비	device-width 또는 크기	브라우저 기본 값
height	뷰포트 높이	device-height 또는 크기	브라우저 기본 값
user-scalable	확대/축소 가능 여부	yes 또는 no	yes
initial-scale	초기 확대/축소 값	1~10	1
minimum-scale	최소 확대/축소 값	0~10	0.25
maximum-scale	최대 확대/축소 값	0~10	1.6

■ 예) <meta name="viewport" content="width=device-width, initial-scale=1">

14.2 가변 그리드 레이아웃

(1) 고정 그리드와 가변 그리드

■ 픽셀 단위를 사용했을 때의 단점을 극복하기 위해 미디어 쿼리를 이용해 기기 해상도별로 CSS 따로 정의, 그러나 기기별로 레이아웃이 달라질 수 있다.



■ 어느 기기에서나 동일한 레이아웃 유지하려면 -> 가변 레이아웃(Fluid Layout)
- 사이트 레이아웃을 픽셀 단위가 아닌 %로 지정한다.
- 브라우저 너비 값이 바뀔 때마다 웹 요소의 너비 값도 함께 바뀐다.

(2) 가변 그리드 레이아웃 만들기

■ 고정 그리드 레이아웃을 만들고 가변 그리드 레이아웃으로 바꾼다.

■ 전체를 감싸는 div 추가하기
- 가장 바깥에 새로운 태그를 추가하고 id 지정
- 각 요소들의 너비를 %(백분율)로 바꿀 때 기준이 되는 너비가 필요하기 때문
- 웹 문서 내용 전체의 크기나 배경색 등을 한꺼번에 조절할 수 있다.
- 브라우저 화면 크기에 상관없이 웹 문서의 내용을 중앙에 배치할 수 있다.

예) #wrapper {

```
width:96%;
margin:0 auto;
  }
  <div id="wrapper">
...
  </div>
```

■ 각 요소의 너비를 백분율(%)로 바꾸기

- 바깥의 #wrapper의 너비 값을 백분율 값으로 변환
- #wrapper 크기를 기준으로 나머지 요소들의 너비를 %로 계산
- (요소의 너비 / 콘텐츠 전체를 감싸는 요소의 너비) * 100
  예) article 600px → 600px / 960px * 100 = 62.5%
- 소수점 이하 숫자가 많다면 간단하게 소수점 이하 한두 자리로 표시
- 약간씩의 오차를 고려하여 실제 계산한 값보다 약간 작게 지정하는 것이 좋다.

[14/fluid-grid.html] 가변 그리드 레이아웃 만들기

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Fluid Grid Layout</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
          #wrapper {
                    width:96%;
                    margin:0 auto;
          }
          header {  /* 헤더 */
                    width: 100%;
                    height: 120px;
                    background-color: #066cfa;
                    border-bottom: 1px solid black;
          }
          .header-text{
                    font-size:32px;
                    color:white;
                    text-align:center;
                    line-height:120px;
          }
          .content {  /* 본문 */
                    float:left;
                    width:62.5%;
                    height:400px;
                    padding:1.5625%;
                    background-color:#ffd800;
          }
          .right-side {  /* 사이드 바 */
                    float:right;
                    width:31.25%;
                    height:400px;
                    padding:1.5625%;
                    background-color:#00ff90;
          }
          footer {  /* 푸터 */
                    clear:both;
                    width:100%;
                    height:120px;
                    background-color:#c3590a;
          }
</style>
```

```
</head>

<body>
        <div id="wrapper">
                <header>
                        <h1 class="header-text">가변 그리드 레이아웃</h1>
                </header>
                <section class="content">
                        <h4>본문</h4>
                </section>
                <aside class="right-side">
                        <h4>사이드바</h4>
                </aside>
                <footer>
                        <h4>푸터</h4>
                </footer>
        </div>
</body>
</html>
```

14.3 가변 레이아웃과 가변 요소

(1) 가변 글꼴

■ em 단위
- 부모 요소에서 지정한 폰트의 대문자 M의 너비를 1em으로 지정한 것으로 1em은 16px이다. (1em = 16px)
- 텍스트 크기를 픽셀(px) 단위로 지정하면 크기가 고정되기 때문에 화면 크기가 작은 기기에서는 매우 작게 표시된다.
- em 단위를 사용하면 해당 기기에 맞춘 픽셀 크기로 계산되어 표시된다.

■ rem 단위
- em 단위는 부모 요소의 글꼴을 기준으로 하기 때문에 중첩된 부모 요소의 글자 크기의 영향을 받는다. -> em 수치가 계속 달라진다는 단점이 있다.
- r은 루트(root)를 뜻하며 처음부터 기본 크기를 지정하기 때문에 중간에 기본 값이 바뀌지 않는다.

[14/fluid-font3.html] 가변 글꼴

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Fluid Grid Layout</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
body {
        font-size: 16px;
}

#wrapper {
        width: 96%;
        margin: 0 auto;
        font-size: 12px;
}

.header-text {
        font-size: 2rem;
```

```
        text-align: center;
}

.fluid-text {
        font-size: 1.5rem;
}
</style>
</head>

<body>
        <div id="wrapper">
                <header class="header-text">
                        <p>가변 그리드 레이아웃</p>
                        <!-- 루트 요소가 16px 이므로 2em=32px -->
                        <p class="fluid-text">가변 폰트</p>
                        <!-- 루트요소가 16px 이므로 1.5em=24px -->
                </header>
        </div>
</body>
</html>
```

(2) 가변 이미지

■ CSS 이용하기
예) .content img {
        max-width:100%;
        height:auto;
    }

■ <img> 태그에서 srcset 속성을 이용하면 화면 너비 값이나 픽셀 밀도에 따라 고해상도의 이미지 파일을 지정할 수 있다.

■ 예) <img src="images/pencil.jpg" srcset="images/pencil-hd.jpg 2x" alt="색연필">
<picture> 태그와 <source> 태그 - 상황별로 다른 이미지 표시하기

[14/fluid-img4.html] 가변 글꼴

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Fluid Image</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
</head>

<body>
    <picture>
        <source srcset="images/shop-large.jpg" media="(min-width:1024px)">
        <source srcset="images/shop-medium.jpg" media="(min-width:768px)">
        <source srcset="images/shop-small.jpg" media="(min-width:320px)">
        <img src="images/shop.jpg" alt="fill with coffee" style="width: 100%;">
    </picture>
</body>
</html>
```

(3) 가변 비디오

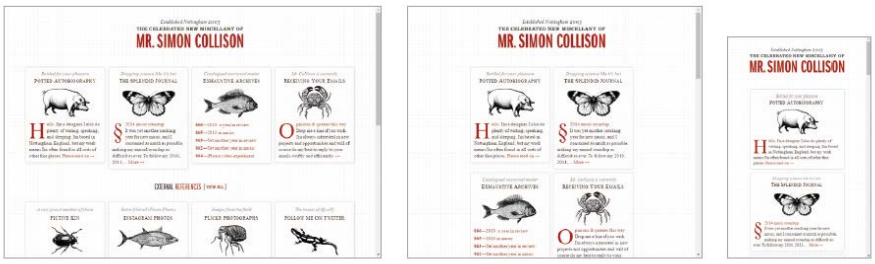
■ 화면의 너비가 달라질 때마다 비디오의 너비가 늘어나거나 줄어들 수 있도록 해야 한다.

```
예) video {
        max-width: 100%;
    }
```

14.4 미디어 쿼리

(1) 미디어 쿼리란?

- CSS3 모듈 중 하나로 사이트에 접속하는 장치에 따라 특정한 CSS 스타일을 사용하도록 해 주는 기능이다.
- 브라우저 창의 너비를 조절할 때마다 화면에 표시되는 칼럼 개수가 달라짐
- PC나 태블릿, 스마트폰의 웹 브라우저 화면 크기에 따라 사이트 레이아웃이 바뀜.



colly.com

(2) 미디어 쿼리 구문

- `@media [ONLY ¦ NOT] 미디어 유형 [AND 조건] * [AND 조건]`

연산자	설명
and	조건을 계속 추가할 수 있습니다.
,(쉼표)	동일한 스타일 유형을 사용할 미디어의 유형과 조건이 있다면 쉼표를 이용해 추가합니다.
only	미디어 쿼리를 지원하는 웹 브라우저에서만 조건을 인식하게 합니다. 이 키워드를 사용하면 미디어 쿼리를 지원하지 않는 웹 브라우저에서는 미디어 쿼리를 무시하고 실행하지 않습니다. IE에서는 미디어 쿼리를 제대로 인식하지 못하기 때문에 only 키워드를 사용하더라도 큰 의미가 없습니다.
not	not 다음에 지정하는 미디어 유형을 제외합니다. 예를 들어 'not tv'라고 지정한다면 TV를 제외한 미디어 유형에만 적용합니다.

- 미디어 유형의 종류

미디어 유형	사용 가능한 미디어
all	모든 미디어 유형
print	인쇄 장치
screen	컴퓨터 스크린(스마트폰 스크린 포함)
tv	음성과 영상이 동시 출력되는 TV
aural	음성 합성 장치(주로 화면을 읽어 소리로 출력해 주는 장치)
braille	점자 표시 장치
handheld	패드(pad)처럼 손에 들고 다니는 장치
projection	프로젝터
tty	디스플레이 기능이 제한된 장치(픽셀(px) 단위를 사용할 수 없음)
embossed	점자 프린터

```
@media screen and (min-width:200px) and (max-width:360px) {
...
}
```

(3) 미디어 쿼리의 조건

■ 웹 문서의 가로 너비와 세로 높이: 뷰포트의 너비와 높이

가로, 세로 값 설정하는 속성	설명
width, height	웹 페이지의 가로 너비, 세로 높이
min-width, min-height	최소 너비, 최소 높이
max-width, max-height	최대 너비, 최대 높이

[14/mq.html]

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Media Queries</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
body {
        background: url(images/bg0.jpg) no-repeat fixed;
        background-size: cover;
}

@media screen and (max-width:1024px) {
        body {
                background: url(images/bg1.jpg) no-repeat fixed;
                background-size: cover;
        }
}

@media screen and (max-width:768px) {
        body {
                background: url(images/bg2.jpg) no-repeat fixed;
```

```
                    background-size: cover;
          }
}

@media screen and (max-width:320px) {
          body {
                    background: url(images/bg3.jpg) no-repeat fixed;
                    background-size: cover;
          }
}
</style>
</head>

<body>


</body>
</html>
```

■ 단말기의 가로 너비와 세로 높이: 뷰포트의 너비와 높이

단말기의 가로, 세로 값을 설정하는 속성	설명
device-width, device-height	단말기의 가로 너비, 세로 높이
min-device-width, min-device-height	단말기의 최소 너비, 최소 높이
max-device-width, max-device-height	단말기의 최대 너비, 최대 높이

```
@media all and (min-device-width:320px) and (min-device-height:480px) {
   ...
}
```

■ 화면 회전 : 디바이스를 세로로 또는 가로로 보기

속성	설명
orientation: portrait	단말기 세로 방향
orientation: landscape	단말기 가로 방향

[14/mq-orientation.html]

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Media Queries</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
body {
          background-color: #eee;
}

@media screen and (orientation:landscape) {
          body {
                    background-color: orange;
          }
}
```

```
@media screen and (orientation:portrait) {
        body {
                background-color: yellow;
        }
}
</style>
</head>

<body>
        <h1>Meida Query</h1>
</body>
</html>
```

- 화면 비율 : 브라우저 화면 너비 값(width)을 높이 값(height)으로 나눈 것
- 단말기의 물리적 화면 비율 : 단말기 너비 값(device-width)을 높이로 나눈 것
- 색상당 비트 수 : 단말기에서 사용하는 최대 색상 비트 수
- 미디어 쿼리 중단점 만들기
  - **중단점(breakpoint)** : 서로 다른 CSS를 적용할 화면 크기
  - 대부분 기기의 화면 크기 기준.
  - 모든 기기를 반영할 수 없기 때문에 스마트폰과 태블릿, 데스크톱 정도로 구분
  - 모바일 퍼스트(mobile first) : 모바일 기기 레이아웃을 기본으로 작성 → 태블릿 & PC 레이아웃 작성
  - 미디어 쿼리 중단점은 개발자나 작업 조건에 따라 달라질 수 있다.

```
/* 스마트폰 세로 */
@media only screen and (min-width: 320px) { …… }

/* 스마트폰 가로 */
@media only screen and (min-width: 480px) { …… }

/* 태블릿 세로 */
@media only screen and (min-width: 768px) { …… }

/* 태블릿 가로 / 데스크톱 */
@media only screen and (min-width: 1024px) { …… }
```

14.5 미디어 쿼리를 사용해 사이트 구성하기

(1) 외부 CSS 파일 연결하기

- 각 조건별로 스타일시트 파일을 따로 저장한 후, <link> 태그나 @import 문을 사용해서 CSS 파일 연결한다.
- 기본형: <link href="css 파일 경로" rel="stylesheet" type="text/css" media="조건">
- 예) <link href="css/tablet.css" rel="stylesheet" type="text/css" media="screen and (min-width:321px) and (max-width:768px)">

(2) 웹 문서에서 직접 정의하기

- 〈style〉 태그 안에서 media 속성을 사용하여 조건과 그에 맞는 스타일 정의

- 기본형: 〈style media="미디어쿼리 조건"〉
  스타일 규칙들
  〈/style〉

  〈style〉
  @media 미디어쿼리 조건 {
  스타일 규칙들
  }
  〈/style〉

```
<style media=" screen and (max-width:320px) " >
  body {
    background-color: orange;
  }
</style>

<style>
  @media screen and (max-width:320px) {
    body {
      background-color: orange;
    }
  }
</style>
```

14.6 플렉서블 박스 레이아웃

(1) 플렉서블 박스 레이아웃과 새로운 용어

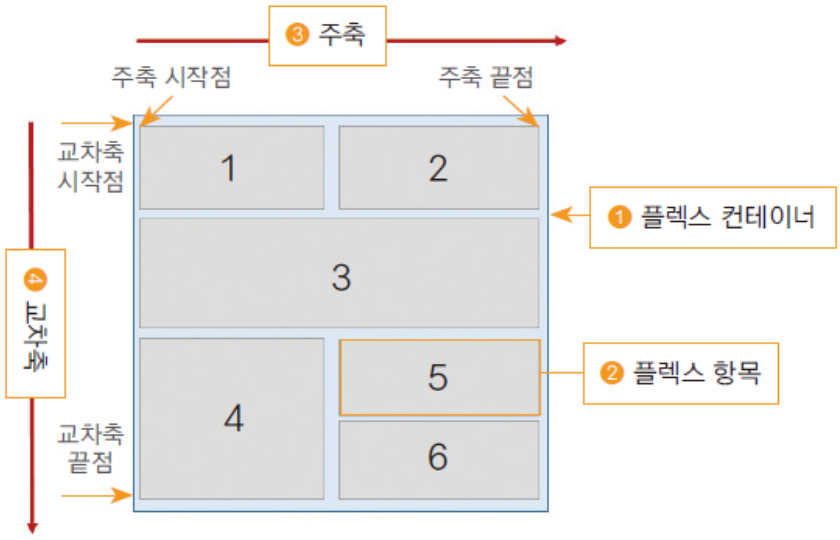
- 플렉서블 박스 레이아웃(flexible box layout)이란 그리드 레이아웃을 기본으로 해 플렉스 박스를 원하는 위치에 배치하는 것이다.

- 플렉스 컨테이너(flex container)

- 플렉스 항목(flex item)

- 주축(main axis)

- 교차축(cross axis)



(2) 플렉서블 박스 레이아웃 기본 속성

■ display 속성 - 플렉스 컨테이너 지정하기
- 기본형: display: flex ¦ inline-flex
  - flex: 플랙스 박스를 박스 레벨 요소로 정의한다.
  - inline-flex: 플랙스 박스를 인라인 레벨 요소로 정의한다.
- 배치하려는 웹 요소들이 있다면 그 요소들을 감싸는 부모 요소를 만들고 그 부모 요소를 플렉스 컨테이너로 만든다.

예) 〈style〉
```
    #container {
        display: flex;
    }
〈/style〉
〈div id=”container”〉
        ...
〈/div〉
```

■ display 속성과 브라우저 접두사
- 브라우저마다 플렉스 박스를 지원하는 방법이 달라 브라우저 접두사를 붙여야 한다.

예) .wrapper {
```
    display: -webkit-box;
    display: -moz-box;
    display: -ms-flexbox;
    display: -webkit-flex;
    display: flex;
}
```

■ flex-direction 속성 - 플렉스 방향 지정하기
- 기본형: flex-direction: row ¦ row-inverse ¦ colum ¦ colum-inverse

속성 값	설명
row	**주축을 가로로 교차축을 세로로 지정합니다. 플렉스 항목은 주축 시작점에서 끝점으로(왼쪽에서 오른쪽으로) 배치됩니다.★**
row-inverse	주축을 가로로 교차축을 세로로 지정합니다. 플렉스 항목은 주축 끝점에서 시작점으로(오른쪽에서 왼쪽으로) 배치됩니다.
column	주축을 세로로 교차축을 가로로 지정합니다. 플렉스 항목은 주축 시작점에서 끝점으로(위쪽에서 아래쪽으로) 배치됩니다.
column-inverse	주축을 세로로 교차축을 가로로 지정합니다. 플렉스 항목은 주축 끝점에서 시작점으로(아래쪽에서 위쪽으로) 배치됩니다.

[14/flex1.html] flex-direction 속성은 지정하지 않았을 경우에 기본 값인 row로 인식한다.

```
<!DOCTYPE html>

<html lang="ko">
<head>
<meta charset="utf-8">
<title>플렉스 박스</title>
<style>
#container {
		width: 500px;
		height: 300px;
		margin: 0 auto;
		display: flex;
		border: 2px solid black;
}

#container div {
		width: 200px;
		border: 2px solid black;
		background: #ccc;
}

h2 {
		font-size: 20px;
		font-weight: bold;
		padding: 20px;
}
</style>
</head>
<body>
		<div id="container">
			<div id="box1">
				<h2>1</h2>
			</div>
			<div id="box2">
				<h2>2</h2>
			</div>
			<div id="box3">
				<h2>3</h2>
			</div>
		</div>
</body>
</html>
```

[14/flex2.html] 플렉스 방향을 세로로 지정한 것이다.

```
<!DOCTYPE html>

<html lang="ko">
```

```
<head>
<meta charset="utf-8">
<title>플렉스 박스</title>
<style>
#container {
		width: 500px;
		height: 300px;
		margin: 0 auto;
		display: flex;
		flex-direction: column;
		border: 2px solid black;
}

#container div {
		width: 200px;
		border: 1px solid black;
		background: #ccc;
}

h2 {
		font-size: 20px;
		font-weight: bold;
		padding: 20px;
}
</style>
</head>
<body>
		<div id="container">
			<div id="box1">
				<h2>1</h2>
			</div>
			<div id="box2">
				<h2>2</h2>
			</div>
			<div id="box3">
				<h2>3</h2>
			</div>
		</div>
</body>
</html>
```

■ flex-wrap 속성 - 플렉스 항목을 한 줄 또는 여러 줄로 배치하기

- 기본형: flex-wrap: no-wrap ¦ wrap ¦ wrap-reverse

속성 값	설명
no-wrap	플렉스 항목들을 한 줄에 표시합니다. 기본 값입니다.★
wrap	플렉스 항목을 여러 줄에 표시합니다.
wrap-reverse	플렉스 항목을 여러 줄에 표시하되 기존 방향과 반대로 배치합니다.

[14/flex3.html] 플렉스 항목을 여러 줄에 표시한다.

```
<!DOCTYPE html>

<html lang="ko">
<head>
<meta charset="utf-8">
<title>플렉스 박스</title>
<style>
#container {
		width: 500px;
```

```
            height: 300px;
            margin: 0 auto;
            display: flex;
            -ms-flex-wrap: wrap;
            -webkit-flex-wrap: wrap;
            flex-wrap: wrap;
            border: 2px solid black;
}

#container div {
            width: 200px;
            border: 2px solid black;
            background: #ccc;
}

h2 {
            font-size: 20px;
            font-weight: bold;
            padding: 20px;
}
</style>
</head>
<body>
            <div id="container">
                        <div id="box1">
                                    <h2>1</h2>
                        </div>
                        <div id="box2">
                                    <h2>2</h2>
                        </div>
                        <div id="box3">
                                    <h2>3</h2>
                        </div>
            </div>
</body>
</html>
```

[14/flex4.html] 교차축의 끝점에서 시작점으로(아래 -> 위) 배치한다.

```
<!DOCTYPE html>

<html lang="ko">
<head>
<meta charset="utf-8">
<title>플렉스 박스</title>
<style>
#container {
            width: 500px;
            height: 300px;
            margin: 0 auto;
            display: flex;
            -ms-flex-wrap: wrap-reverse;
            -webkit-flex-wrap: wrap-reverse;
            flex-wrap: wrap-reverse;
            border: 2px solid black;
}

#container div {
            width: 200px;
            border: 2px solid black;
            background: #ccc;
}

h2 {
```

```
            font-size: 20px;
            font-weight: bold;
            padding: 20px;
}
</style>
</head>
<body>
            <div id="container">
                        <div id="box1">
                                    <h2>1</h2>
                        </div>
                        <div id="box2">
                                    <h2>2</h2>
                        </div>
                        <div id="box3">
                                    <h2>3</h2>
                        </div>
            </div>
</body>
</html>
```

■ flex-flow 속성 - 플렉스 방향과 여러 줄의 배치를 한꺼번에 지정하기
- 기본형: flex-flow: <플렉스 방향> <플렉스 줄 배치>
- 예) flex-flow: column wrap; /* 위에서 아래로, 여러 줄에 표시 */

■ order 속성 - 플렉스 항목의 배치 순서 바꾸기
- 기본형: order: 0 | 숫자

[14/flex-order.html] 1번 박스를 두 번째, 2번 박스를 세 번째, 3번 박스를 첫 번째로 배치한다.

```
<!DOCTYPE html>

<html lang="ko">
<head>
<meta charset="utf-8">
<title>플렉스 박스</title>
<style>
#container {
            width: 500px;
            height: 300px;
            margin: 0 auto;
            display: flex;
            border: 2px solid black;
}

#container div {
            width: 200px;
            border: 2px solid black;
            background: #ccc;
}

#box1 {
            order: 2;
}

#box2 {
            order: 3;
}

#box3 {
            order: 1;
}
```

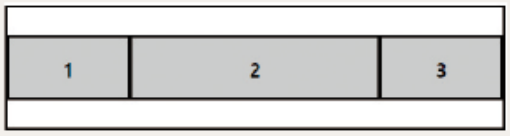
```
h2 {
        font-size: 20px;
        font-weight: bold;
        padding: 20px;
}
</style>
</head>
<body>
        <div id="container">
                <div id="box1">
                        <h2>box1</h2>
                </div>
                <div id="box2">
                        <h2>box2</h2>
                </div>
                <div id="box3">
                        <h2>box3</h2>
                </div>
        </div>
</body>
</html>
```

■ flex 속성 - 플렉스 항목 크기 조절하기

- 기본형: flex: [<flex-grow> <flex-shrink> <flex-basis>] ¦ auto ¦ initial

속성 값	설명
<flex-grow>	플렉스 항목의 너비를 얼마나 늘일지 숫자로 지정합니다.
<flex-shrink>	플렉스 항목의 너비를 얼마나 줄일지 숫자로 지정합니다.
<flex-basis>	플렉스 항목의 기본 크기를 지정합니다. width 속성처럼 너비 값을 지정할 수도 있고 0이나 auto를 지정할 수도 있습니다. 0일 경우, flex-grow와 flex-shrink의 인수 값을 함께 사용하고 auto일 경우, 플렉스 항목의 너비 값을 사용합니다.
initial	항목의 width/height 값에 의해 크기가 결정되는데 플렉스 컨테이너의 공간이 부족할 경우, 최소 크기까지 줄입니다.*
auto	항목의 width/height 값에 의해 크기가 결정되지만 플렉스 컨테이너의 공간에 따라 늘이거나 줄입니다.

```
<style>
  #box1 {
    flex:1 1 0; /* 늘이거나 줄이지 않음 */
  }
  #box2 {
    flex:2 2 0; /* 2배 늘이거나 2배 줄임 */
  }
</style>
```



(3) 플렉스 항목 배치를 위한 속성들

■ justify-content 속성 - 주축 기준의 배치 방법 지정하기

- 기본형: justify-content: flex-start ¦ flex-end ¦ center ¦ space-between ¦ space-

around

속성 값	설명	미리 보기
flex-start	주축의 시작점을 기준으로 배치합니다.	
flex-end	주축의 끝점을 기준으로 배치합니다.	
center	주축의 중앙을 기준으로 배치합니다.	
space-between	첫 번째 플렉스 항목과 마지막 플렉스 항목은 시작점과 끝점에 배치한 후 중앙 항목들은 같은 간격으로 배치합니다.	
space-around	모든 플렉스 항목들을 같은 간격으로 배치합니다.	

■ align-items 속성, align-self 속성 - 교차축 기준의 배치 방법 지정하기

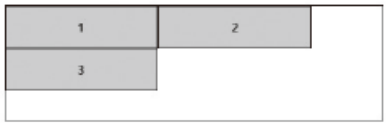
- align-items 속성과 align-self 속성을 이용하면 주축뿐만 아니라 교차축을 기준으로 배치 방법을 조절할 수 있다.
  - 기본형: align-items : stretch ¦ flex-start ¦ flex-end ¦ center ¦ baseline
- align-self 속성을 이용하면 플랙스 항목을 개별적으로 배치할 수 있다. 즉, 특정 플랙스 항목만 배치 방법을 바꿀 수 있다.
  - 기본형: align-self : auto ¦ stretch ¦ flex-start ¦ flex-end ¦ center ¦ baseline

속성 값	설명	미리 보기
stretch	플렉스 항목을 확장해 교차축을 꽉 채웁니다. 기본 값입니다.★	
flex-start	교차축의 시작점을 기준으로 배치합니다.	
flex-end	교차축의 끝점을 기준으로 배치합니다.	
center	교차축의 중앙을 기준으로 배치합니다.	
baseline	시작점과 글자 기준선이 가장 먼 플렉스 항목(미리보기에서는 2번 항목의 글자 크기가 가장 크기 때문에 2번의 글자 기준선이 가장 멀리 떨어져 있음)을 시작점에 배치합니다. 그리고 그 글자의 기준선과 다른 항목의 기준선을 맞추어 배치합니다.	

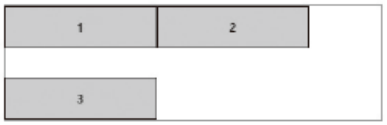
■ align-content 속성 - 여러 줄일 때의 배치 방법 지정하기

- 기본형: align-content: flex-start ¦ flex-end ¦ center ¦ space-between ¦ space-around

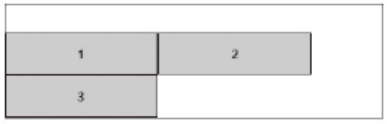
align-content : flex-start (기본 값)



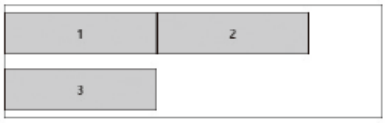
align-content : space-between



align-content : flex-end



align-content : space-around



align-content : center