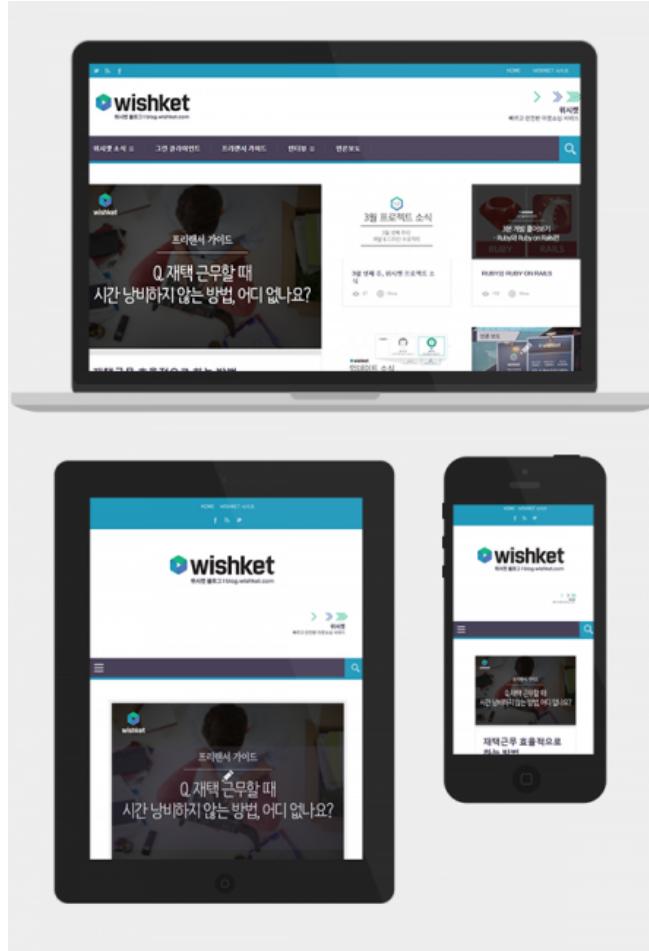


CSS 3 심화 과정



@ IT Korea 강사 이정훈

반응형 웹(Responsive Web)



반응형 웹이란 다양한 화면 크기에 맞게 웹사이트를 표시 할수 있는 방법입니다

적응형 vs 반응형



적응형 웹은 기본적으로 클라이언트의 정보를 미리 받아 표시합니다.

그래서 크기를 줄였다 늘였다 해도 레이아웃이 변하지 않습니다.

반응형 웹은 브라우저를 줄이면 레이아웃 자체가 변합니다.

미디어 쿼리

Media Queries

The Modern Gentleman



미디어쿼리란, CSS3모듈로 사이트 접속하는 장치에 따라,
특정한 CSS스타일을 사용하도록 해주는 기능입니다. *<https://mediaqueri.es>

미디어 쿼리 구문

@media [only | not] 미디어 유형 [And 조건문] * [...]

기본적인 미디어 쿼리를 사용시 사용되는 구문 형태입니다.

Only는 미디어 쿼리를 지원하는 웹 브라우저에서만 조건을 인식하며,
not은 뒤의 미디어유형을 제외한 나머지 미디어유형에만 적용됩니다.

미디어 유형의 종류

- **all** : 모든 미디어 유형에서 사용할 CSS를 정의
- **print** : 인쇄 장치에서 사용할 CSS를 정의
- **screen** : 컴퓨터 스크린에서 사용할 CSS를 정의, 스마트폰도 포함
- **tv** : 음성과 영상이 동시 출력되는 TV에서 사용할 CSS를 정의
- **aural** : 음성 합성장치에서 사용할 CSS를 정의
- **braille** : 점자 표시장치에서 사용할 CSS를 정의
- **handheld** : 패드처럼 한손에 들고 다니는 장치를 위한 CSS를 정의
- **projection** : 프로젝터를 위한 CSS를 정의
- **tty** : 디스플레이 기능이 제한된 장치에 맞는 CSS를 정의
- **embossed** : 점자 프린터를 위한 CSS를 정의

```
<!doctype html>
<html lang="ko">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title>미디어쿼리</title>
    <style>
        body{
            background-color:gray;
        }
        #logo img{
            width:100%;
            margin-top:150px;
        }

        @media screen and (max-width: 960px) {
            body { background-color:green; }
        }

        @media screen and (max-width:500px) {
            body {background-color:yellow; }
        }

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

가로너비와 세로너비를 설정하는 속성입니다. Width, height 가 있으면, min-width(height), max-width(height) 와 같이 최대,최소너비설정이 간능합니다.

단말기 가로 너비와 세로 높이

- **device-width, device-height**
- **min-device-width, min-device-height**
- **max-device-width, max-device-height**

화면 비율

- **aspect-ratio** : 화면 비율 (width / height)
- **min-aspect-ratio** : 최소 화면 비율
- **max-aspect-ratio** : 최대 화면 비율

단말기의 물리적 화면 비율

- **device-aspect-ratio** : 단말기 화면 비율 (width / height)
- **device-min-aspect-ratio** : 단말기 최소 화면 비율
- **device-max-aspect-ratio** : 단말기 최대 화면 비율

```
<!doctype html>
<html lang="ko">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title>미디어쿼리</title>
    <style>
        body{
            background-color:gray;
        }
        #logo img{
            width:100%;
            margin-top:50px;
        }
    @media screen and (orientation:landscape) {
        body {background-color:orange;}
    }
    @media screen and (orientation:portrait) {
        body {background-color:yellow;}
    }
    </style>
</head>
<body>
    <div id="logo">
        
    </div>
</body>
</html>
```

화면을 회전하는 속성입니다. portrait와 landscape 두 가지가 있습니다.

미디어 쿼리 적용하기

1. CSS 파일 연결하기

```
<link rel="stylesheet" type="text/css" href="css 파일경로" media="미디어 쿼리 조건">  
  
<link rel="stylesheet" type="text/css" href="css/print.css" media="print">  
  
<link rel="stylesheet" type="text/css" href="css/tablet.css" media="screen and (min-width:321px)">  
  
<link rel="stylesheet" type="text/css" href="css/tablet.css" media="screen and (min-width:321px) and (max-width:768px)">
```

```
@import url("css/tablet.css") screen and (min-width:321px) and (max-width:768px)
```

미디어 쿼리를 직접적으로 적용하는 방법은 두가지로 나뉩니다.

외부에서 css 파일을 연결시키는 방법과 웹문서에 직접 정의하는 방법이죠.

먼저 위의 두가지 방법은 외부에서 연결하는 두가지 방법입니다. link와 @import 입니다.

2. 웹 문서에서 직접 정의하기

```
<!--첫번째 방법-->

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

<!--두번째 방법-->

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

웹문서에서 직접 정의 하는 방법은 두가지가 있습니다.

첫번째로 <style> 태그 안에 media 속성을 사용하여 조건을 지정하고, 조건에 맞게 스타일을 지정하는방법, 두번째로 스타일 선언시 미디어문을 사용해 조건별로 스타일을 지정해 놓고 선택적으로 스타일을 적용하는 방법입니다.

뷰포트 지정하기

```
<meta name="viewport" content="width=device-width">  
|  
<meta name="viewport" content="width=device-width, user-scalable=yes,  
initial-scale=1.0, maximum-scale=3.0">
```

스마트폰에서 실제 내용이 표시되는 영역을 뷰포트라고 합니다.

미디어 쿼리를 이용해서 뷰포트에 맞게 스타일을 정의했다 하더라도, 화면에서 내용이 아주 작
게 표시되는것을 보게 됩니다. 따라서 표기되는 내용의 크기와 배율을 조절해야합니다.

이때 뷰포트를 지정해줍니다.

width : 뷔포트의 너비를 지정합니다. 기본값은 device-width.

height : 뷔포트의 높이를 지정합니다. 기본값은 device-height.

initial-scale : 초기 배율을 나타내며, 기본값은 1입니다. 1보다 작은값이면 축소된 페이지를 표시하고, 1보다 큰 값은 확대된 페이지를 표시합니다.

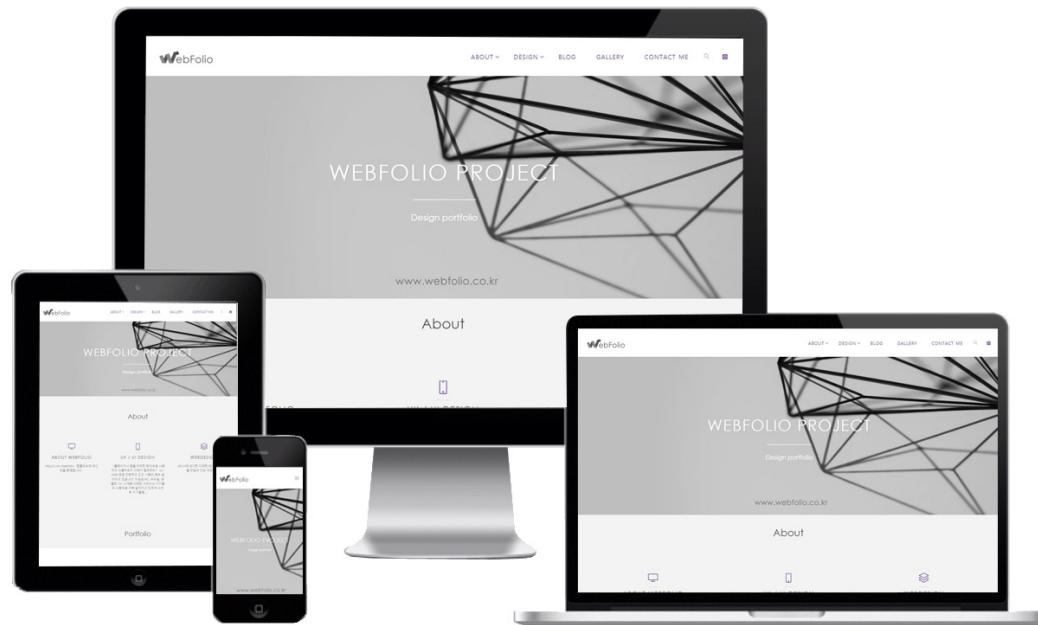
user-scalable : 사용자가 페이지를 확대/축소할수있는지의 여부를 지정합니다. 기본값은 yes, no로 지정하면 사용자가 확대축소를 할수없습니다.

minimum-scale : 사용자가 축소할수있는 최솟값을 지정합니다. 기본값은 0.25이며, 가로값을 기준으로 합니다.

maximum-scale : 사용자가 확대할수있는 최댓값을 지정합니다. 기본 값은 5.0 입니다.

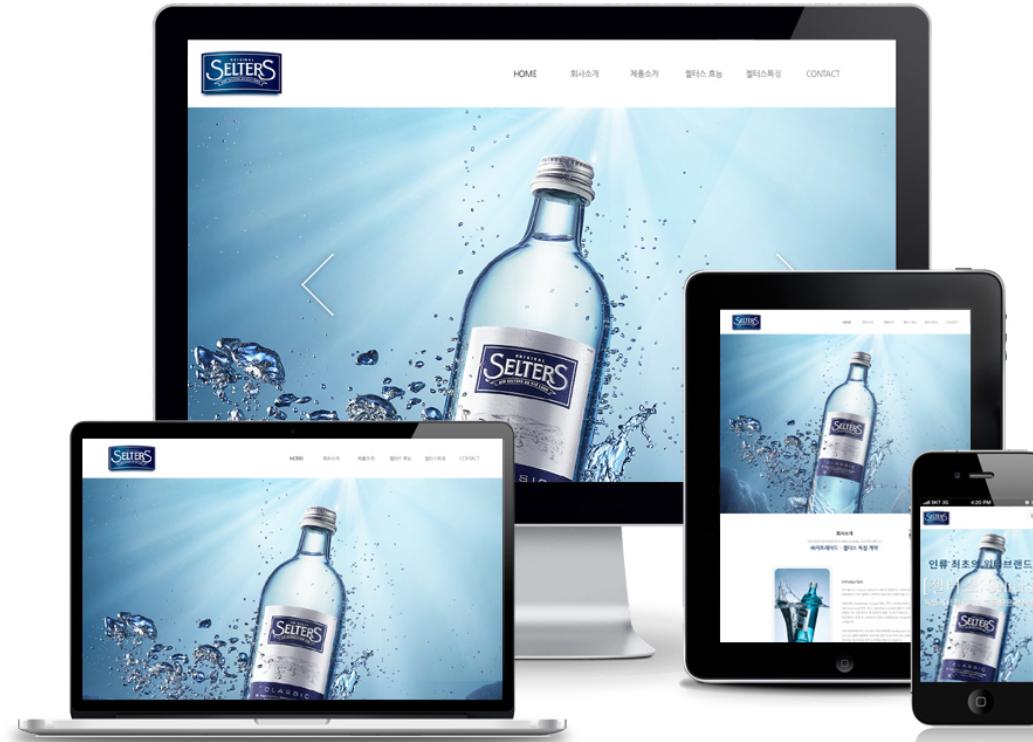
유동형 레이아웃 만들기

그리드 시스템



그리드 시스템이란, 레이아웃을 정할때 자주 사용하는 기준입니다.
굳이 따를 필요는 없지만, 화면을 단순하면서도 규칙적으로 배열을 하기 때문에
대부분의 사이트에서 사용하는 방법입니다.

유동형 레이아웃



유동형 레이아웃은 사이트 모든 요소를 상대적인 크기로 지정해서
브라우저 크기에 따라 탄력적으로 보여주는 방법입니다.
CSS 하나만 정의하면 되기에 매우 유연한 방법입니다.

1. div 추가하기

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title>너비가 고정된 레이아웃</title>
    <style>
        body{
            background:#ccc;
        }
        header{
            width:960px;
            height:120px;
            background-color:#f5f4e0;
            background:url(back.png) no-repeat left center;
            border-bottom:1px solid black;
        }
        header h1{
            padding-top:30px;
            padding-left:120px;
        }
        .header1{
            font-size:40px;
            font-family: "Comic Sans MS";
        }
        article {
            float:left;
            width:600px;
            padding:20px;
        }
        aside {
            float:right;
            width:300px;
            padding:8px;
            background-color:#d7e4e0;
            border-left:1px dotted black;
        }
        footer{
            clear:both;
            background:#677b75;
            padding:10px;
            text-align:center;
        }
    </style>
</head>
```

```
<body>
    <header>
        <h1 class="header1">Fixed-width Layout</h1>
    </header>
    <section>
        <article>
            <p>실제 내용이 표시됩니다</p>
            <p>실제 내용이 표시됩니다</p>
            <p>실제 내용이 표시됩니다</p>
            <p>실제 내용이 표시됩니다</p>
            <p>실제 내용이 표시됩니다</p>
        </article>
        <aside>
            <p>사이드바의 내용이 표시됩니다</p>
            <p>사이드바의 내용이 표시됩니다</p>
        </aside>
    </section>
    <footer>
        <address>제작자 연락처 정보</address>
        <p>저작권 정보 및 주소 </p>
    </footer>
</body>
</html>
```

2. 요소의 너비를 백분율로 바꾸기

```
#wrapper{  
    width:96%;  
    margin:0 auto;  
    background-color:white;  
}
```

```
article {  
    float:left;  
    width:62.5%;  
    padding:20px;  
}
```

```
aside {  
    float:right;  
    width: 뭘까요? 계산해보자! ;  
    padding:8px;  
    background-color:#d7e4e0;  
    border-left:1px dotted black;  
}
```

px값을 %로 바꿀때 반드시 96%로 해야만 하는것은 아닙니다.

px값에서 일반적으로 계산이 쉽게 1/10, 1/100 크기로 줄여서 계산합니다.

article 또는 aside 값이 #wrapper 값 안에 포함되는 것이므로, wrapper 크기를 기준으로 백분율 값을 계산합니다

3. 패딩과 마진을 백분율로 바꾸기

```
header h1{  
    padding-top:30px;  
    padding-left:120px;  
}
```

```
article {  
    float:left;  
    width:600px;  
    padding:20px;  
}
```

```
aside {  
    float:right;  
    width:300px;  
    padding:8px;  
    background-color:#d7e4e0;  
    border-left:1px dotted black;  
}
```

```
footer{  
    clear:both;  
    background:#677b75;  
    padding:10px;  
    text-align:center;  
}
```

마찬가지로 padding 과 margin 값도 백분율로 변경시켜줍니다.

4. 글자 크기를 em 단위로 변환하기

```
.header1{  
    font-size:40px;  
    font-family: "Comic Sans MS";  
}
```

글꼴 12pt

글꼴 16px

글꼴 1em

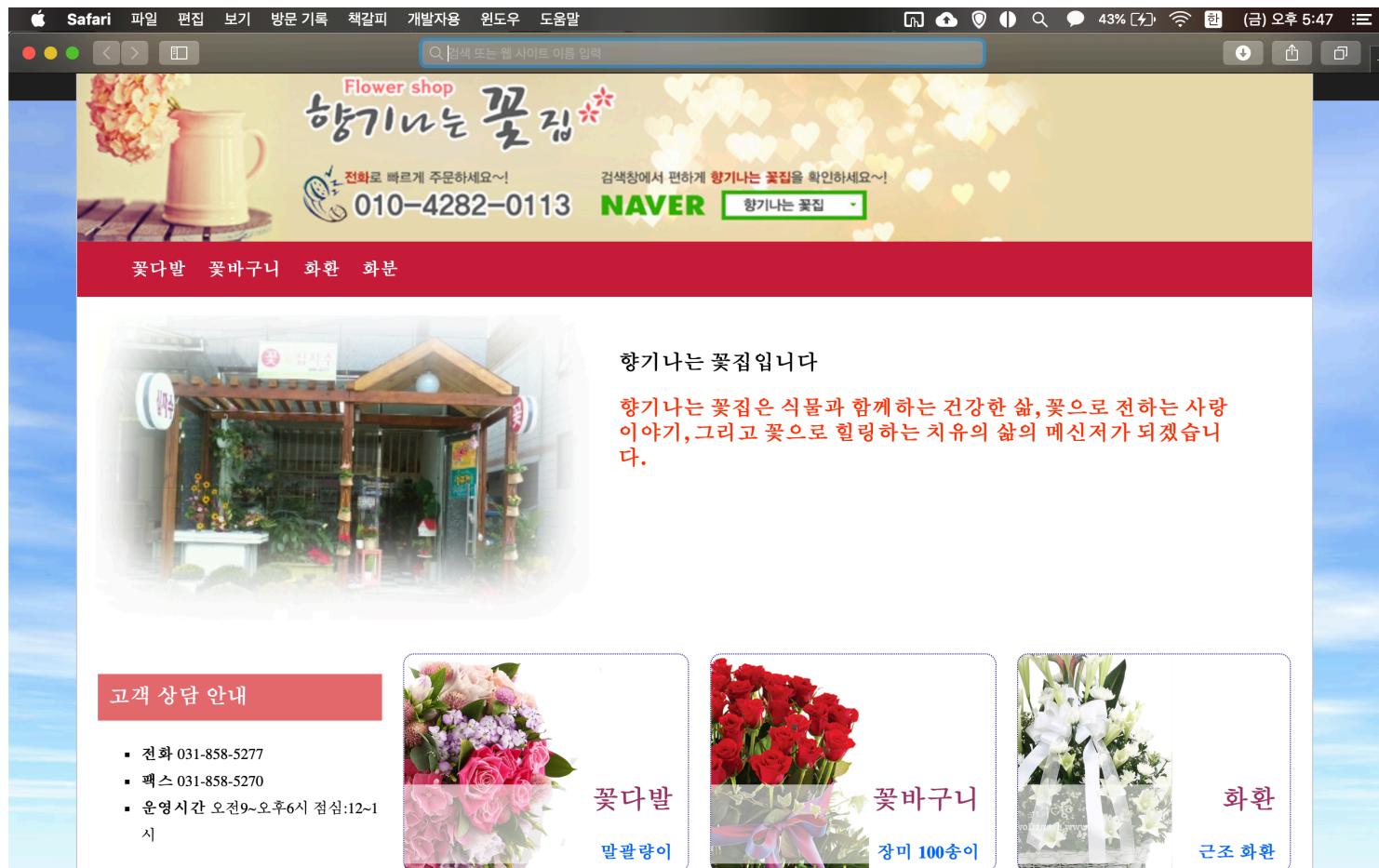
웹 페이지 작성시 글자 크기는 주로 픽셀 단위를 씁니다.

px은 모니터의 해상도를 기준으로 해서 N-screen 시대에는 적합하지 않습니다.
16px 이 1em에 해당합니다. 즉 $em = \text{글자크기}/\text{px} / 16/\text{px}$ 이라는 공식이 나오지요.

유동형 이미지와 비디오

```
<!DOCTYPE html>
<html lang="ko">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width">
<title>애완견 돌보기</title>
<style>
#wrapper{
    width: 96%;
    margin:0 auto;
}
video{
    max-width:100%;
}
header{
    background-color:#069;
    padding:10px;
    overflow:hidden;
}
header h1{
    width:40%;
    float:left;
    font-size:2.5em;
    font-family:"돋움", "굴림";
    color:#fff;
}
header nav{
    width:60%;
    float:right;
}
nav ul {
    list-style-type:none;
    margin-top:20px;
}
nav ul li{
    display:inline;
    float:left;
    margin-top:20px ;
    margin-right:20px;
}
nav ul a, nav ul li a:visited{
    font-family:"돋움", "굴림";
    font-size:16px;
    color:white;
    text-decoration:none;
}
section {
    border:5px solid #333;
    padding:30px;
}
footer{
    padding:10px;
    background-color:#333;
    color:white;
    text-align:center;
}
</style>
</head>
<body>
<header>
<h1>애완견 종류</h1>
<nav>
<ul>
<li><a href="#">애완견 종류</a></li>
<li><a href="#">입양하기</a></li>
<li><a href="#">건강돌보기</a></li>
<li><a href="#">더불어살기</a></li>
</ul>
</nav>
</header>
<section>
<article>
<h2>귀염둥이들의 일상</h2>
<video width="600" height="450" controls id="puppy">
    <source src="mypuppy.ogv">
    <source src="mypuppy.mp4">
    <source src="mypuppy.webm">
</video>
</article>
</section>
<footer>
    <p>Copyright 2012 funnycom</p>
</footer>
</body>
</html>
```

최종 실습 : 반응형 웹 직접 만들어 보기

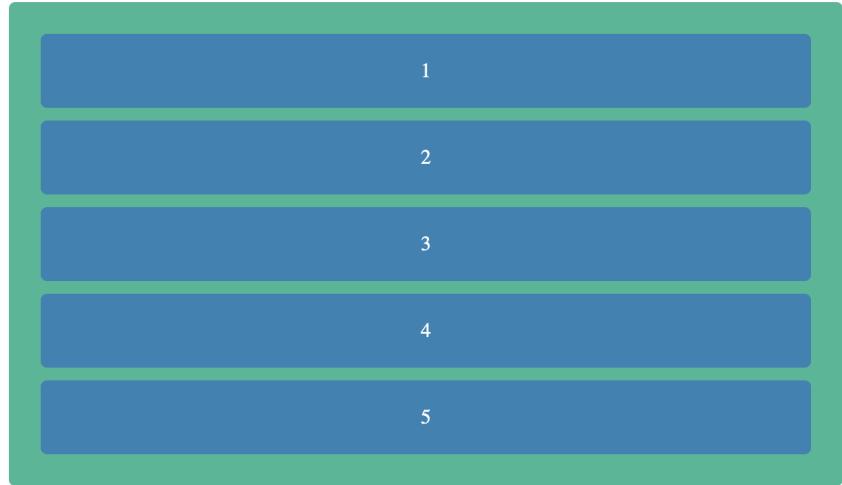


1. 그리드 레이아웃 구상하기
2. 전체 구조 만들기
3. CSS 리셋하기
4. 화면 중앙에 내용 전체 배치하기
5. 미디어 쿼리 추가하기
6. 내용 입력하기
7. 메뉴부분에 내용 입력하기
8. 꽃집 소개 이미지 및 텍스트 입력
9. 고객상담을 위한 연락처와 추천 제품 입력하기
10. 푸터 입력하고 마무리하기

Flexbox

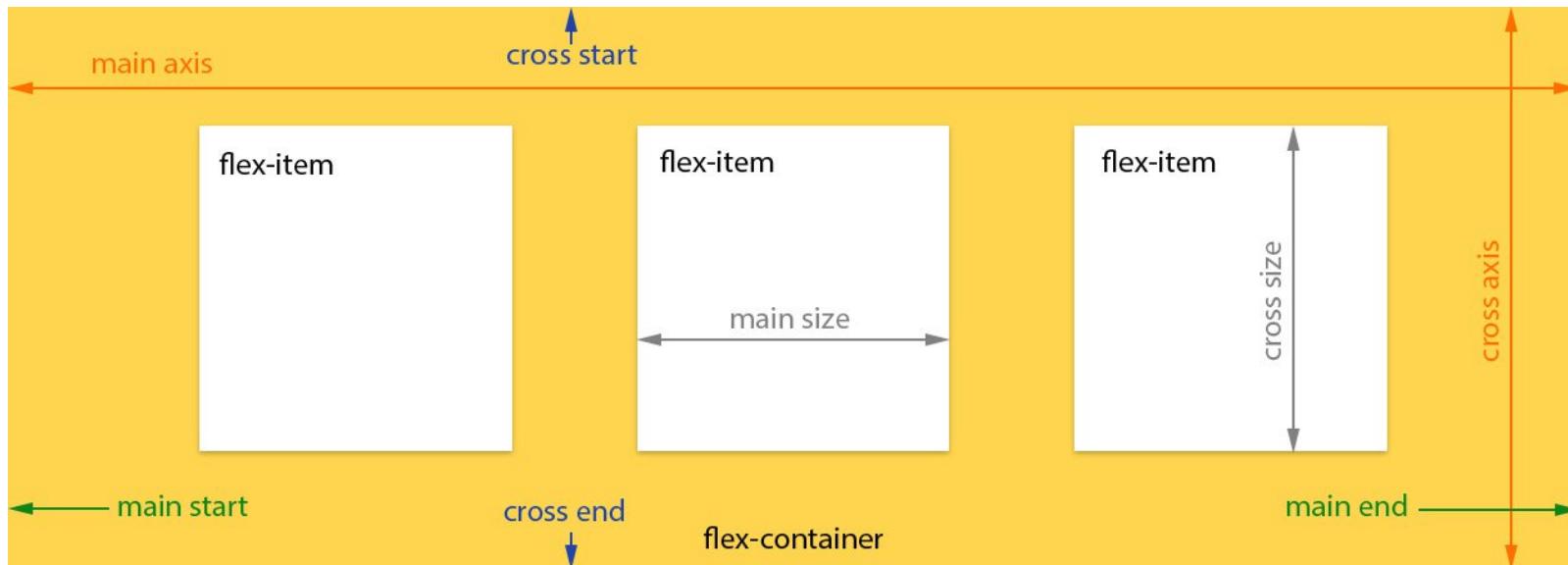
```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Flexbox Layout Example</title>
  <style>
    .flex-container {
      margin: 10px;
      padding: 15px;
      border-radius: 5px;
      background: #60B99A;
    }

    .flex-item {
      margin: 10px;
      padding: 20px;
      color: #fff;
      text-align: center;
      border-radius: 5px;
      background: #4584b1;
    }
  </style>
</head>
<body>
  <div class="flex-container">
    <div class="flex-item">1</div>
    <div class="flex-item">2</div>
    <div class="flex-item">3</div>
    <div class="flex-item">4</div>
    <div class="flex-item">5</div>
  </div>
</body>
</html>
```



Flexbox란 기존 layout보다 세련된 방식으로 모던 웹에 적용시키는 새로운 layout방식입니다. 요소의 사이즈가 불명확하거나 동적으로 변화할때도 유연한 레이아웃을 만들수있습니다.

Flexbox layout 구조



flexbox를 사용하기 위해서 HTML 부모요소의 display 속성에 flex를 지정한다.

부모요소는 flex-container가 부모요소가 된다.

부모요소가 inline 요소일경우, inline-flex를 지정한다.

이 두가지는 반드시 부모요소에 지정해야하며, 자식요소는 자동으로 flex item이 된다.

Flexbox-container

1. flex-direction

```
.flex-container {  
    flex-direction: row;  
}
```

```
.flex-container {  
    flex-direction: row-reverse;  
}
```

```
.flex-container {  
    flex-direction: column;  
}
```

```
.flex-container {  
    flex-direction: column-reverse;  
}
```

direction은 flex-container의 주축(main-axis) 방향을 설정한다.

2. flex-wrap

```
<!DOCTYPE html>
<html>
<head>
<title>Flexbox</title>
<meta charset="UTF-8">
<style>
.flex-container {
  width: 500px;
  margin: 10px;
  padding: 15px;
  border-radius: 5px;
  background: #60B99A;

  display: flex;
  flex-wrap: nowrap;
}

.flex-item {
  margin: 10px;
  padding: 20px;
  color: #fff;
  text-align: center;
  border-radius: 5px;
  background: #4584b1;
}
</style>
</head>
<body>
<div class="flex-container">
  <div class="flex-item">11111</div>
  <div class="flex-item">22222</div>
  <div class="flex-item">33333</div>
  <div class="flex-item">44444</div>
  <div class="flex-item">55555</div>
</div>
</body>
</html>
```

flex-wrap 속성은 flex 컨테이너의 복수 flex-item 을 1행 또는 복수행으로 배치합니다.
Flex 컨테이너의 width보다 flex item들의 width 합계가 더 큰 경우, 한 줄로 표현할 것인지,
여러 줄로 표현할 것인지를 정합니다.

```
.flex-container {  
    flex-wrap: wrap;  
    flex-wrap: wrap-reverse;  
    flex-wrap: nowrap;  
}
```

1. **nowrap** : 개행을 하지 않음
2. **wrap** : flex item들의 width의 합계가 flex 컨테이너의 width보다 큰 경우, flex item을 복수행에 배치한다. 기본적으로 좌에서 우로, 위에서 아래로 배치된다
3. **wrap-reverse** : flex-wrap: wrap;과 동일하나 아래에서 위로 배치된다.

1	2	3	4	5	6	7	8
9	10	11	12				

9	10	11	12				
1	2	3	4	5	6	7	8

3. justify-content

```
.flex-container {  
    justify-content: flex-start;  
    justify-content: flex-end;  
    justify-content: center;  
    justify-content: space-between;  
    justify-content: space-around;  
}
```

1. flex-start



2. flex-end



3. center



4. space-between



5. space-around



flex container의 main axis를 기준으로 flex item을 수평 정렬한다.

Flexbox-container

1. order

```
.flex-item {  
    order: 정수값;  
}
```



flex item의 배치 순서를 지정한다. HTML 코드를 변경하지 않고 order 속성값을 지정하는 것으로 간단히 재배치할 수 있다. 기본 배치 순서는 flex container에 추가된 순서이다. 기본값은 0이다.

2. flex-grow

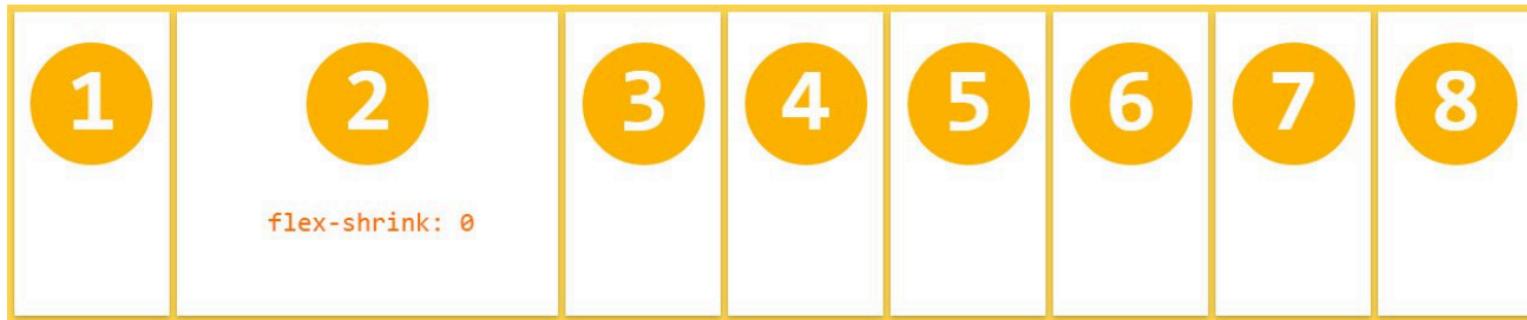
```
.flex-item {  
    flex-grow: 양의 정수값;  
}
```



flex item의 너비에 대한 확대 인자(flex grow factor)를 지정한다. 기본값은 0이고 음수값은 무효하다. 동일한 값을 지니면, 고르게 나오며, 다를 때 차이가 생긴다.

3. flex-shrink

```
.flex-item {  
    flex-shrink: 양의 정수값;  
}
```



`flex item`의 너비에 대한 축소 인자(`flex shrink factor`)를 지정한다. 기본값은 1이고 음수값은 무효하다. 0을 지정하면 축소가 해제되어 원래의 너비를 유지한다. 기본적으로 모든 `flex item`은 축소된 상태로 지정(기본값 1)하고 두번째 `flex item`만 축소를 해제(`flex-shrink: 0;`)하면 원래의 너비를 유지한다.

4. flex-basis

```
.flex-item {  
    flex-basis: auto | <width>;  
}
```



flex item의 너비 기본값을 px, % 등의 단위로 지정한다. 기본값은 auto이다.