반응형 웹디자인

스마트폰이 대중화되면서 초기에는 PC용 웹 사이트와 모바일 사이트를 따로 제작했습니다.
하지만 스마트폰, 태블릿 PC 등 다양한 해상도의 휴대용 디바이스가 나오면서 각 디바이스에 대응하는 웹 사이트를 따로따로 만든 사이트가 아닌 한 개의 웹 사이트로 모든 디바이스에 대응할 수 있는 반응형 웹디자인(Responsive Web Design) 사이트가 나왔습니다.

반응형 웹이란?
디스플레이(기기) 종류에 따라 화면의 크기가 자동으로 최적화되도록 조절되는 웹 페이지를 의미합니다.
HTML5 시대가 도래됨에 따라 기능 완성 형태의 새로운 마크업 언어의 등장으로 스마트한 웹 페이지 구현이 가능합니다.

왜 반응형 웹인가?
1. 유지보수가 간편하다!
   모바일 버전과 데스크톱 버전 같이 2개의 웹사이트를 만들게 되면, 수정시에도 2번작업을 해야함.
   하지만 반응형 웹은 모바일버전, 태블릿버전, 데스크톱 버전 등 모든 디자인을 하나의 HTML과 CSS 파일에서 작업하기 때문에 유지보수가 간편하다.
2. 마케팅에 유리하다!
   반응형 웹이라는 기술을 이용하여 마케팅 웹사이트를 개발하면 환경이나 기기에 따라 최적화된 구조로 웹사이트를 변경하여 보여줄 수 있기 때문에 전달하고자 하는 내용을 확실하게 전달할 수 있음.
   그리고 두가지 버전의 웹사이트를 만들지않아도 되기 때문에 비용측면에서도 상당한 절감효과가 있음.
3. 검색 엔지 최적화가 가능하다!
   검색엔진 최적화란 포털사이트 또는 검색사이트에서 사용자가 특정 키워드로 검색을 했을 때 나오는 웹사이트 검색 결과의 상위노출을 뜻함.
   만약 모바일 버전 PC 버전 같이 2개의 웹사이트가 있으면 주소가 2개이기 때문에 검색결과에서 상위권에 배치되기 어려움.
   하지만 반응형 웹은 1개의 주소와 1개파일(HTML)만 있어 검색결과에 잘 노출될 수 있음.
4. 미래 지향적 기술이다!
   시대가 변할수록 기기가 다양해지고, 고정형태의 웹사이트는 잘려서 보이는 문제가 발생.
   하지만 반응형 웹은 웹사이트의 구조를 환경에 따라 최적화된 구조로 바꾼 웹사이트를 보여줌.
   앞으로 어떤 화면의 기기가 나올지 모르는 상황에서 반응형웹은 웹기술 중 가장 미래지향적이다.

사례로 알아보는 반응형 웹

워드프레스 같은 CMS(콘텐츠 관리 시스템 : 사용자가 웹사이트에서 콘텐츠를 생성, 관리 및 수정할 수 있도록 하는 소프트웨어 응용 프로그램) 도구를 이용해 제작한 웹사이트를 판매하는 사이트에서도 대부분의 상품이 반응형웹으로 제작되어 있음.

해외 반응형 웹 사례 –보스턴글로브 **(http://www.bostonglobe.com)**
신문사 웹사이트는 내용이 많고 복잡해 디자인 설계가 어려워서 고정크기로 웹사이트를 제작하는

경우가 많음.
하지만 보스턴글로브는 반응형웹사이트로 개발하여 흰바탕에 글자위주로 디자인하고 각영역을 회색의 선으로만 구분하여 간단한 디자인으로 구성.

국내 반응형 웹사례 – 중앙그룹 **(https://www.joonganggroup.com)**
와이드한 이미지와 블랙앤화이트 계열로 깔끔하게 구성하여 주목성, 디자인, 기획력까지 돋보이는 디자인. 다양한 인터랙션으로 사용자에게 신선한 경험을 제공함.
화면이 작아지면 기존의 메뉴영역을 감추고 다양한 토글버튼을 제공하여 사용자 편리성을 강조함.

반응형 웹 제작의 핵심 기술

가변그리드 : 픽셀의 한계를 대신하여 적용
가변이란? 어떠한 객체/물체 또는 사물이 늘어나거나 줄어 들거나 성질이 변하는것.
가변그리드는 웹사이트를 제작할 때 화면의 크기에 관계없이 자유롭게 늘어나거나 줄어들 수 있도록 픽셀(px) 대신 퍼센트(%)로 제작하는 기술.

미디어쿼리 – 화면의 크기와 환경 감지 및 웹사이트를 변경하는 기술
미디어 쿼리(Media Queries)란? 컴퓨터나 기기에게 '너는 어떤 종류의 미디어니?' 또는 '미디어의 화면크기는 어느정도나 되니?' 라고 미디어에게 질문하고 감지한 후 웹사이트를 변경하는 기술.
CSS3의 속성 중 하나. 미디어 쿼리를 사용해 적용 스타일을 기기마다(화면 크기마다) 전환함.
컴퓨터나 기기의 환경 또는 종류를 알아야 그 기기에 맞게 웹사이트의 구조를 바꿀 수 있기 때문에 반응형 웹을 제작할 때 반드시 필요함.
[only 또는 not] @media [미디어 유형] [and 또는 ,콤마] (조건문) {실행문}

뷰포트 – 화면을 제어하는 기술
뷰포트(Viewport)는 화면에 보이는 영역을 제어하는 기술로 수많은 기기의 화면크기를 감지해야 할 때 꼭 필요한 기술.
데스크톱은 사용자가 지정한 해상도가 보이는 영역이 되지만 스마트기기는 기본설정값이 보이는 영역으로 지정됨. 미디어쿼리를 사용해도 스마트기기의 실제 화면크기를 감지할 수 없어 반응형 웹제작할때는 뷰포트가 반드시 필요.

모바일 환경과 해상도의 이해

모바일 환경은 2000년대 무선 인터넷폰 시절로 거슬러 올라갑니다. PC 환경만큼은 아니지만 그동안 수많은 모바일 운영체제가 존재했습니다. 모든 운영체제를 알 필요는 없지만, 최근 스마트폰 열풍을 가지고 온 대표적인 두 주자인 iOS와 안드로이드 운영체제의 환경과 해상도는 반드시 알아야 합니다.

스마트폰의 대중화로 모바일 전용 사이트가 만들어지고 있습니다. 각 해상도를 정확하게 이해하고 있어야만 그에 맞는 사이트를 구축할 수 있습니다.
매년이 아닌 매달 새로운 해상도의 신제품이 나오기 때문에 어느 해상도가 가장 우선인지는 서비스하는 목표에 따라 달라질 수 있으나, 최저 해상도는 아이폰 3GS의 320px과 갤럭시 S2의 480px을 기준으로 봅니다.

하지만 여기서 아주 흥미로운 사실이 있습니다. 디바이스의 해상도와 디바이스에 내장된 브라우저의 기본 표현 해상도가 다르다는 점입니다. 아이폰 전 제품의 가로값은 320px이며, 갤럭시 S3는 360px이고 갤럭시 노트는 400px로 보통 디바이스 해상도의 절반값을 갖고 있습니다.

기본적으로 해상도를 기준으로 디바이스를 구분하고 있으며, 다양한 해상도가 있지만 아래 표를 기준으로 하고 있습니다.

가로폭	**명칭**
480px	스마트폰
800px	태블릿
1024px	넷북
1600px	데스크톱

미디어쿼리 MedaiQuery

미디어쿼리의 기본문법

@media [only 또는 not] [미디어유형] [and 또는 ,콤마] (조건문) {실행문}
ex) @media only screen and (max-width:786px){width:100%}

[only 또는 not]
only 키워드는 미디어쿼리를 지원 하는 브라우저에서만 해석하게 해주는 키워드입니다.
not은 다음에 오는 조건을 부정하는 키워드 입니다. 예를 들어 not tv일 경우 tv를 제외한 모든 미디어 유형에만 적용됩니다.

[미디어유형]

기기명	설명
all	모든장치
print	인쇄방치
screen	컴퓨터 화면 장치 또는 스마트 기기의 화면
tv	영상과 음성이 동시에 출력되는 장치
projection	프로젝터 장치
handheld	손에 들고 다니는 소형장치
speech	음성 출력 장치
aural	음성 합성 장치(화면을 읽어 소리로 출력해 주는 장치)
embossed	점자 인쇄 장치(화면을 읽어 종이에 점자를 찍어내는 장치)
tty	디스플레이 기능이 제한된 장치
braille	점자 표시 장치

※ handheld는 소형 기기라는 의미가 있지만 스마트 기기는 screen을 사용

조건문의 종류

조건문	설명	조건값	min/max사용
width	웹페이지의 가로 너비값	width 속성에서 사용할 수 있는 모든 속성값	사용함
height	웹페이지의 세로 높이값		
device-width	기기의 가로 너비값		
device-height	기기의 세로 높이값		
orientation	기기의 화면 방향	portrait(세로) landscape(가로)	사용 안 함
aspect-ratio	화면 비율	브라우저 화면 비율(1), 브라우저 종횡비(16/9), 브라우저 해상도(1280/720)	사용함
device-aspect-ratio	단말기 브라우저의 화면 비율	기기 화면 비율(1), 기기 종횡비(16/9), 기기 해상도(640/320)	
color	기기의 비트 수	8(bit 단위)	
color-index	기기의 색상 수	128(색상 수 단위)	
monochrome	기기가 흑백일 때 픽셀당 비트 수	1(bit 단위)	
resolution	기기의 해상력	300dpi/dpcm	
scan	TV의 스캔 방식	progressive/interlace	
grid	기기의 그리드/비트맵	0(비트맵 방식)/1(그리드 방식)	사용안함

미디어쿼리 작성시 주의사항

1. 띄어쓰기 주의하여 입력하기
@media [only 또는 not] [미디어유형] [and 또는 ,콤마] (조건문) {실행문}

2. 조건문 접두사인 min을 사용할때는 반드시 작은 순서대로 작성 (min은 최소 또는 그이상이라는 뜻으로, 점차 커지는 것을 의미하기 때문에)
   @media (min-width:480px){실행문}
   @media (min-width:768px){실행문}
   @media (min-width:1024px){실행문}

3. 조건문 접두사인 max 사용시 큰순서대로 작성 (max는 최대 또는 그 이하라는 뜻으로, 점차 작아지는 것을 의미하기 때문에)
   @media (max-width:1200px){실행문}
   @media (max-width:1024px){실행문}
   @media (max-width:768px){실행문}

① 대형 해상도의 PC : 2560px * 1600px
② 와이드 해상도의 PC : 1920px * 1200px
③ 일반 해상도의 PC : 1280px * 960px
④ 낮은 해상도의 PC, 테블릿 : 1024px * 768px
⑤ 모바일 가로, 테블릿 : 480px ~ 767px
⑥ 모바일 : ~ 480px

뷰포트 속성

속성명	속성값	속성 설명
width	device-width, 양수	뷰포트의 너비를 지정.
height	device-height, 양수	뷰포트의 높이를 지정.
initial-scale	양수	뷰포트의 초기 배율을 지정(기본값은 1) 1보다 작은 값을 사용하면 축소된 페이지를 표시하고, 1보다 큰 값을 사용하면 확대된 페이지를 표시
user-scalable	yes,no	뷰포트의 확대/축소 여부를 지정(기본값은 yes). 반대로 no를 설정하면 사용자가 페이지를 확대 할 수 없음.
minimum-scale	양수	뷰포트의 최소 축소 비율을 지정(기본값은 0.25).
maximum-scale	양수	뷰포트의 최대 확대 비율을 지정(기본값은 5.0).

※ minimum-scale과 maximum-scale 속성은 각각 값을 1.0으로 지정할 경우 user-scalable을 'yes'로 지정하더라도 사용자가 화면을 확대하거나 축소할 수 없음.

1) 반응형 웹디자인을 위한 기본 설정

기기의 가로 해상도 상황별로 총 6가지 디자인으로 이뤄져 있습니다. 각 해상도에 따라 변경되는 콘텐츠의 레이아웃과 노출 여부를 체크해야 합니다.

① 대형모니터
② 와이드 모니터
③ 일반 모니터
④ 태블릿 PC
⑤ 소형 태블릿 PC
⑥ 스마트폰

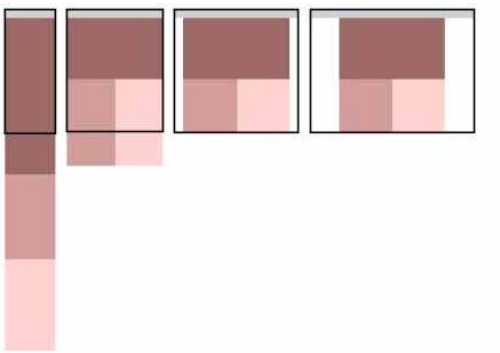
사실 3가지(PC, 태블릿, 스마트폰) 정도만 구분하면 되지만 예제를 디자인하고 구성할 때는 좀 더 복잡하더라도 사용자가 편리하게 사용할 수 있는 사이트를 구축해야 합니다.

2) 가장 많이 사용하는 유동형 패턴(Mostly Fluid)

Mostly Fluid 패턴은 반응형 웹디자인에서 가장 많이 사용하는 패턴입니다.
중대형 화면에서 중간 크기의 화면까지는 여백 정도만 조정하고, 작은 화면이 되면 그리드가 움직이며 콘텐츠를 수직으로 배치합니다.

- Wide PC에서는 가변이 아닌 고정형으로 가고 이보다 작아지면 내부는 가변적으로 변하게끔 설정하는 패턴입니다.

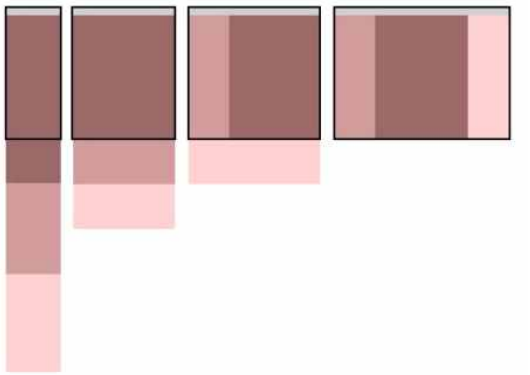
- 태블릿 PC에서는 여백을 없애고 가로가 100%인 상태로 콘텐츠를 노출합니다.
- 모바일에서는 작은 크기에서 최적화하기 위해 column들을 한 개씩 세로로 정렬(vertical align) 합니다.



```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport"content="width=device-width, initial-scale=1.0">
<title>Mostly Fluid</title>
<style>
* {margin: 0; padding: 0;}
#wrap {width: 80%; margin: 0 auto; }
#box_a {height: 400px; background: #aaa;}
#box_b {float: left; width: 50%; height: 400px; background: #bbb;}
#box_c {float: right; width: 50%; height: 400px; background: #ccc;}
@media screen and (max-width: 1024px) {
    #wrap {width: 90%;}
}
@media screen and (max-width: 768px) {
    #wrap {width:100%;}
}
@media screen and (max-width: 480px) {
    #box_b {float: none; width: 100%;}
    #box_c {float: none; width: 100%;}
}
</style>
</head>
<body>
<div id="wrap">
    <div id="box_a"></div>
    <div id="box_b"></div>
    <div id="box_c"></div>
</div>
</body>
</html>
```

3) 컬럼 드롭 패턴(Column Drop)

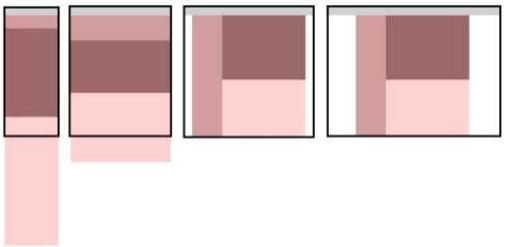
Column Drop 패턴에서는 가로로 정렬(Inline align)되어 있던 요소가 해상도에 따라 세로로 정렬되는 모습을 볼 수 있습니다.



```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport"content="width=device-width, initial-scale=1.0">
<title>Column Drop</title>
<style>
* {margin: 0; padding: 0;}
#box_a {float: left; width: 30%; height:400px; background: #aaa;}
#box_b {float: left; width:50%; height:400px; background: #bbb;}
#box_c {float: right; width: 20%; height: 400px;  background: #ccc;}

@media screen and (max-width: 1024px) {
    #box_b {width: 70%;}
    #box_c {clear: left; float: none; width: 100%;}
}
@media screen and (max-width: 768px) {
    #box_a {float: none; width: 100%;}
    #box_b {float: none; width: 100%;}
}
</style>
</head>
<body>
<div id="box_a"></div>
<div id="box_b"></div>
<div id="box_c"></div>
</body>
</html>
```

4) 시프터 패턴(Shifter)

세로와 가로 정렬이 혼합된 모습으로 PC 웹 사이트와 모바일 웹 사이트 사이의 GNB 영역(Global Navigation Bar : 페이지마다 항상 위치하게 되는 주메뉴 영역)의 변화가 가장 큽니다. 자칫하면 다른 사이트로 보일 수도 있으나, 이질감이 느끼지 않게 룩엔픽(Look & Feel)을 유지하면서 변형한다면 오히려 사용자들에게 참신하고 좋은 느낌을 줄 수 있습니다. Shifter 패턴은 Mostyle Fluid 패턴처럼 큰 화면에서는 고정 픽셀이고 작은 화면에서는 퍼센트 형으로 변경되는 특징이 있습니다. #box_a 요소는 GNB 역할로 왼쪽에 있다가 모바일 버전으로 가면서 위로 자리를 변경하게 됩니다. 레이아웃을 크게 변경하면 내부에 들어가는 요소의 디자인도 많이 변경해야 하므로 번거롭고, 새롭게 디자인해야 한다는 단점이 있습니다.



```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport"content="width=device-width, initial-scale=1.0">
<title>Layout Shifter</title>
<style>
* {margin: 0; padding: 0;}
#wrap {width: 80%; margin: 0 auto; }
#box_a {float: left; width: 20%; height: 400px; background: #aaa}
#box_b {float: left; width: 80%; height: 200px; background: #bbb;}
#box_c {float: left; width: 80%; height: 200px;  background: #ccc;}

@media screen and (max-width: 1024px) {#wrap {width: 90%;}}
@media screen and (max-width: 768px) {
    #wrap {width: 100%;}
    #box_a {float: none; width: 100%;}
    #box_b {float: none; width: 100%;}
    #box_c {float: none; width: 100%;}
}
</style>
</head>
<body>
<div id="wrap">
    <div id="box_a"></div>
    <div id="box_b"></div>
    <div id="box_c"></div>
</div>
</body>
</html>
```

5) 작은 리스트 패턴(Tiny Tweaks)

PC부터 모바일까지 모두 같은 형태로 디자인하는 방식으로, 간단한 사이트 또는 리스트 형태로 꾸며진 사이트에 사용할 패턴으로 적합니다.

Tiny 패턴은 PC 버전부터 모바일까지 같은 형태로 작아지기만 하는 레이아웃이므로 자칫하면 꽤 지루하게 보일 가능성이 높습니다. 따라서 내부에 있는 리스트를 변경함으로써 지루함을 없애는 게 이 패턴의 노하우입니다. 예시로 만든 코드 역시 내부의 리스트가 가로 5개에서 4개, 3개, 2개로 점차 줄어들게 했습니다.

우선 ul, li 요소로 리스트를 구성했으면 ul 요소는 width 값을 80%로 설정하고 가운데 정렬을 했습니다.
지금부터가 중요한 부분입니다. 우서 내부에 들어가는 li 요소를 가로로 배치하기 위해 float 속성을 left로 설정했습니다. 이렇게 하면 한 줄에 몇 개의 li 요소가 노출되는가는 li 요소의 width 값에 따라 달라집니다. li 요소의 width 속성의 합이 ul 요소의 width 값보다 커지면 자동으로 줄바꿈되어서 나열되기 때문입니다.

정확히 4개만 노출되기를 바란다면 margin 값까지 고려해서 계산해야 합니다.

① 5개 노출되는 wide PC의 경우
(100%-10%)/5=18%;
100% : li 요소를 감싼 ul 요소의 너비
10% : 각 li 요소의 margin 값의 합
5 : li 요소의 개수

② max-width:1024
(100%-8%)/4=23%;

③ max-width:800px
(100%-6%)/3=31.333 ...%;

④ max-width:480px
(100%-4%)/2=48%;

이런 식으로 반응형에 따른 리스트 개수를 조절하는 방법을 활용하면 갤러리 리스트 또는 네비게이션 메뉴에서 활용할 수 있습니다.

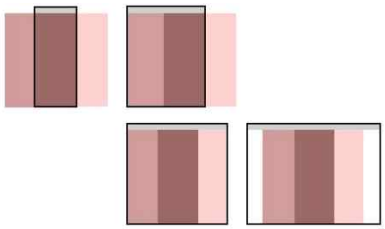
```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport"content="width=device-width, initial-scale=1.0">
<title>Tiny Tweaks</title>
<style>
* {margin:0; padding:0;}
body {font-family:Arial, sans-serif;}
ul {list-style:none;}
#box_a {height:400px; background:#aaa;}
.list_man {margin:0 auto; padding-top:50px; width:80%;}
.list_man li {float:left; margin:1%; width:18%; background:#fff;}

@media screen and (max-width: 1024px) { /* li 4개 */
    .list_man {width:95%;}
    .list_man li {margin:1%; width:23%;}
}
@media screen and (max-width: 768px) { /* li 3개 */
    .list_man li {margin:1%; width:31.333%;}
}
@media screen and (max-width: 480px) { /* li 2개 */
    .list_man li {margin:1%; width:48%;}
}
</style>
</head>
<body>
<div id="box_a">
    <ul class="list_man">
        <li>a</li>
        <li>b</li>
        <li>c</li>
        <li>d</li>
        <li>e</li>
        <li>f</li>
        <li>g</li>
        <li>h</li>
        <li>i</li>
        <li>j</li>
    </ul>
</div>
</body>
</html>
```

6) 오프 캔버스 패턴(Off Canvas)

PC 웹 사이트를 모바일 디바이스로 가져오면서 콘텐츠의 크기와 양을 줄이지 않는 새로운 방법으로, 일부 영역을 숨겨서 사용자의 제스쳐 혹은 액션에 따라 숨겨뒀던 영역을 보여주는 방식입니다. 작은 디바이스에서 많은 것을 보여주는 방법보다 더 편한 UX를 적용해 사용하며 페이스북 모바일 웹, 앱 등에서 많이 사용하고 있는 패턴입니다.

오프 캔버스 패턴은 단순하게 HTML과 CSS로 이뤄지는 경우는 없습니다. 대부분 자바스크립트와 제이쿼리 등을 이용해 작업합니다.

max-width 800px 영역에서 각 부분의 width 값의 합은 절대로 100%가 될 수가 없으며 이는 UI 개발을 하는 이에게 피로를 주는 주범 중 하나입니다. 그러므로 간단하게는 세 개 중 한 개를 33.334%로 설정해 해결할 수 있습니다.



```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport"content="width=device-width, initial-scale=1.0">
<title>Off Canvas</title>
<style>
* {margin: 0; padding: 0;}
body {overflow: hidden;}
#wrap {width: 80%; margin: 0 auto; }
#wrap_box {width: 100%;}
#box_a {float: left; width: 30%; height: 400px; background: #aaa;}
#box_b {float: left; width: 40%; height: 400px; background: #bbb;}
#box_c {float: left; width: 30%; height: 400px; background: #ccc;}
@media screen and (max-width: 1024px) { #wrap {width: 100%;} }
@media screen and (max-width: 768px) {
    #wrap_box {position: absolute; width: 150%;}
    #box_a {width: 33.333%;}
    #box_b {width: 33.334%;}
    #box_c {width: 33.333%;}
}
@media screen and (max-width: 480px) {
  #wrap_box {
            width: 300%;
            margin-left: -100%;
        }
}
</style>
</head>
<body>
<div id="wrap">
    <div id="wrap_box">
        <div id="box_a"></div>
        <div id="box_b"></div>
        <div id="box_c"></div>
    </div>
</div>
</body>
</html>
```