12. 반응형 디자인

뷰포트 설정 (`<meta viewport>`)

모바일 시대의 웹 개발에서 가장 중요한 기본 설정 중 하나는 `viewport` **메타 태그**다.
이 태그는 브라우저가 **페이지를 어떻게 스케일링하고 표시할지**를 정의한다.

---

1. 기본 문법

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

---

2. 주요 속성 설명

속성	의미
`width`	뷰포트의 너비를 설정 (`device-width`: 디바이스 화면 너비와 일치)
`initial-scale`	초기 확대/축소 비율 (1.0은 100%)
`maximum-scale`	최대 확대 허용 비율
`minimum-scale`	최소 축소 허용 비율
`user-scalable`	사용자의 확대/축소 허용 여부 (`yes` / `no`)

---

3. 사용 예시

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

```
<!-- 확대/축소를 금지할 경우 (접근성에는 좋지 않음) -->
<meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">
```

---

4. 왜 필요한가?

과거:

- 모바일 브라우저는 기본적으로 **데스크톱 사이트처럼 표시**함
- 그래서 화면이 작아도 **전체 레이아웃이 축소되어 표시**됨

현재:

- 뷰포트 태그로 "기기의 실제 너비에 맞춰 페이지를 그리라"고 지시
- 반응형 레이아웃(CSS Media Query) 적용이 **정확하게 작동**함

---

5. width 속성의 값

값	의미
`device-width`	실제 디바이스의 화면 너비 (픽셀 단위)
숫자 (예: 640)	고정된 가상 뷰포트 너비 지정

> 대부분의 경우 `device-width`를 사용하는 것이 표준이다.

---

6. 초기 배율 (`initial-scale`)

- 사용자가 페이지에 처음 진입할 때 확대/축소 비율
- `1.0`이면 원래 크기, `0.5`면 50% 축소

---

7. 예외적으로 설정하지 않았을 때

```
<!-- viewport가 없을 때 모바일에서의 현상 -->
- 전체 웹사이트가 축소되어 표시됨
- 버튼, 텍스트가 매우 작아짐
- 반응형 디자인이 적용되지 않음
```

결론

> `<meta name="viewport">`는 **모바일 웹 개발의 핵심 요소**로,
> 모든 반응형 웹페이지에는 **무조건 포함되어야 하는 기본 설정**이다.
> `width=device-width, initial-scale=1.0` 조합은 대부분의 경우 기본값으로 안전하고 권장된다.

미디어 쿼리

미디어 쿼리는 CSS에서 **디바이스의 특성(화면 크기, 해상도, 방향 등)에 따라 다른 스타일을 적용**할 수 있도록 하는 기능이다.
이 기능은 **반응형 웹 디자인(Responsive Web Design)**의 핵심으로, 다양한 화면에서 일관되고 유연한 UI를 구현하게 해준다.

---

1. 기본 문법

```
@media media-type and (condition) {
  /* 조건이 참일 때 적용될 CSS */
}
```

예시:

```
@media screen and (max-width: 768px) {
  body {
    background-color: lightgray;
  }
}
```

- 화면 너비가 768px 이하일 때만 적용

---

2. 주요 미디어 유형 (media type)

미디어 유형	설명
`all`	모든 장치에 적용 (기본값)
`screen`	디스플레이 장치 (PC, 모바일 등)
`print`	인쇄용 스타일
`speech`	스크린 리더 등 음성 출력 장치

---

3. 조건 (media features)

대표 조건들:

조건	설명
`width` / `height`	뷰포트의 너비 또는 높이
`min-width` / `max-width`	최소/최대 너비
`orientation`	`portrait` (세로), `landscape` (가로)
`resolution`	화면 해상도 (dpi 또는 dppx)
`hover`	포인팅 장치가 hover 가능한지 여부
`pointer`	포인팅 장치의 정밀도 (`fine` , `coarse` , `none`)

---

4. 자주 사용하는 미디어 쿼리 패턴

📱 모바일 우선 전략 (Mobile First)

```
/* 기본 모바일용 스타일 먼저 */
body {
  font-size: 14px;
}

/* 화면이 커질수록 덮어씀 */
@media (min-width: 768px) {
  body {
    font-size: 16px;
  }
}

@media (min-width: 1024px) {
  body {
    font-size: 18px;
  }
}
```

---

5. 여러 조건 결합

```
@media screen and (min-width: 600px) and (orientation: landscape) {
  /* 가로 방향이고 너비가 600px 이상일 때만 적용 */
}
```

---

6. 범위 지정 예시

```
@media (min-width: 480px) and (max-width: 767px) {
  /* 태블릿 전용 스타일 */
}
```

---

7. `not`, `only`, `and`, `,` 키워드

키워드	기능
`and`	여러 조건 동시 만족
`,` (쉼표)	여러 미디어 조건 중 하나라도 만족하면 적용 (OR 조건)
`not`	해당 조건이 **거짓일 때** 적용
`only`	구형 브라우저에서 잘못된 해석 방지용 (사실상 의미 없음)

예시

```
@media not screen and (max-width: 768px) {
  /* 768px 이하가 아닌 모든 경우 */
}
```

---

8. 실전 예: 반응형 카드 레이아웃

```
.card {
  width: 100%;
}

@media (min-width: 600px) {
  .card {
    width: 48%;
  }
}

@media (min-width: 1024px) {
  .card {
    width: 30%;
  }
}
```

---

9. 단위 주의

- `%`, `px`, `em`, `rem` 모두 사용 가능
- 뷰포트 기준에서는 `px`이 가장 일반적

---

10. 결론

> 미디어 쿼리는 **다양한 화면 조건을 감지하고 적절한 스타일을 적용하는 방식**이다.
> 반응형 웹을 만들기 위해서는 필수 요소이며,
> `@media`를 적절히 활용하면 **모바일, 태블릿, 데스크탑 모두에서 자연스러운 UI/UX**를 구현할 수 있다.

단위 선택: `%`, `vw`, `vh`, `em`, `rem`

CSS에서 길이, 크기, 위치, 여백 등을 지정할 때는 **다양한 단위**를 사용할 수 있다.
특히 반응형 웹 개발이나 유연한 디자인을 위해 고정 단위(px) 외에
상대 단위인 `%`, `vw`, `vh`, `em`, `rem` 등을 적절히 사용하는 것이 중요하다.

---

1. `%`: 부모 요소에 대한 백분율 비율

- **부모 요소의 너비/높이 대비 백분율로 크기를 지정**
- 레이아웃에서 자주 사용

```
.container {
  width: 100%;   /* 부모 너비의 100% */
  height: 50%;   /* 부모 높이의 50% */
}
```

- 글자 크기에서도 사용 가능하나, 예측하기 어려워 잘 쓰이지 않음

---

2. `vw`, `vh`: 뷰포트 기반 단위

단위	의미
`1vw`	**뷰포트 너비의 1%**
`1vh`	**뷰포트 높이의 1%**
`1vmin`	뷰포트의 **짧은 쪽**의 1%
`1vmax`	뷰포트의 **긴 쪽**의 1%

예시:

```
.hero {
  width: 100vw;    /* 뷰포트 너비 전체 */
  height: 100vh;   /* 뷰포트 높이 전체 */
}
```

- `vw` / `vh`는 **전체 화면에 꽉 차는 배경**이나 **스크롤 없는 영역 설정** 등에 사용

---

3. `em`: 요소 자신의 글꼴 크기 기준

- `1em` = 해당 요소의 `font-size`
- 중첩되면 누적됨 (부모의 `font-size` 영향을 받음)

예시:

```
body {
  font-size: 16px;
}

p {
  font-size: 1.5em;  /* 24px */
}

p span {
  font-size: 2em;    /* 2 * 1.5em = 3em = 48px */
}
```

> 중첩 구조에서는 예측이 어려워지는 단점 있음

---

4. `rem`: 루트 요소(`html`)의 글꼴 크기 기준

- **절대 기준점**: `html` 태그의 `font-size`
- 중첩과 상관없이 항상 동일 기준

예시:

```
html {
  font-size: 16px;
}

h1 {
  font-size: 2rem;   /* 32px */
}

button {
  padding: 0.5rem 1rem;  /* 8px 16px */
}
```

> `rem`은 **디자인 시스템 또는 접근성 향상**을 위한 크기 기준으로 추천됨

---

5. 단위 비교 요약

단위	기준	특징
`%`	부모 요소	상대적 레이아웃 구현
`vw` / `vh`	뷰포트 너비/높이	반응형 배경, 화면 꽉 채우기
`em`	현재 요소의 `font-size`	중첩될수록 누적됨
`rem`	루트 요소의 `font-size`	중첩과 무관, 예측 가능

단위	기준	특징
`px`	고정 크기 (절대 단위)	정밀한 컨트롤, 비반응형

---

6. 언제 어떤 단위를 사용할까?

목적	추천 단위
반응형 전체 너비/높이	`vw`, `vh`
내부 요소 비율 유지	`%`
타이포그래피 계층화	`rem`
컴포넌트 내부 상대 크기	`em`
정확한 픽셀 제어	`px`

---

7. 실전 조합 예시

```
html {
  font-size: 16px;
}

.card {
  width: 90%;
  max-width: 600px;
  padding: 1rem;
}

.hero {
  height: 100vh;
  font-size: 2rem;
}
```

결론

> `%`, `vw`, `vh`, `em`, `rem`은 고정 단위(px)보다 더 **유연하고 반응형 친화적인 CSS를 작성**할 수 있도록 도와준다.
> 목적에 맞는 단위를 적절히 선택하는 것이 **현대 웹 디자인의 핵심 전략**이다.

콘텐츠 크기 조정: `max-width`, `min-height`, etc.

CSS에서는 요소의 크기를 고정하거나 유연하게 제한할 수 있도록 다양한 **최소/최대 크기 속성**을 제공한다.
이러한 속성들은 반응형 웹 디자인이나 콘텐츠 레이아웃에서 **콘텐츠가 지나치게 작아지거나 커지지 않도록 제어**할 수 있는 중요한 도구다.

---

1. `max-width`: 최대 너비 제한

- 요소가 **너무 넓어지는 것**을 방지
- 특히 이미지, 텍스트 블록, 카드 등에 유용

```
.container {
  width: 100%;
  max-width: 800px;
}
```

> 위 예제는 **전체 너비를 사용하되**, 최대 800px까지만 확장되도록 제한함
> → 작은 화면에선 유동적으로 줄어들고, 큰 화면에선 800px까지만 늘어남

---

2. `min-width`: 최소 너비 설정

- 요소가 **지정된 너비보다 작아지지 않도록** 강제

```
.box {
  min-width: 300px;
}
```

> 뷰포트나 부모 요소가 작더라도 **최소 300px 너비는 확보**

---

3. `max-height`: 최대 높이 제한

- 콘텐츠가 너무 길어져서 레이아웃이 무너지는 것을 방지

```
.image {
  height: auto;
  max-height: 400px;
}
```

> 이미지나 박스가 자동으로 늘어나되, **400px 이상 커지지 않도록 제어**

---

4. `min-height`: 최소 높이 보장

- 콘텐츠가 적더라도 **일정한 높이를 유지**할 수 있게 설정

```
.card {
  min-height: 200px;
}
```

> 내용이 짧아도 **일관된 카드 레이아웃 유지**

---

5. 실전 예: 반응형 카드 레이아웃

```
.card {
  width: 100%;
  max-width: 500px;
  min-height: 250px;
  padding: 1rem;
  background-color: #f5f5f5;
}
```

- 작은 화면에서는 `width: 100%`
- 너무 커지면 `max-width`로 제한
- 내용이 적어도 최소 높이 보장

---

6. 자동 높이/너비 (`auto`)

- 브라우저가 콘텐츠에 따라 자동으로 계산

```
img {
  width: 100%;
  height: auto;
}
```

> 이미지 비율 유지하면서 부모 너비에 맞춤

---

7. 비교 요약

속성	의미	주 용도
`width`	고정 너비	특정 크기 강제
`min-width`	최소 너비	너무 좁아지지 않도록
`max-width`	최대 너비	너무 커지지 않도록
`height`	고정 높이	특정 높이 강제
`min-height`	최소 높이	일정 높이 확보
`max-height`	최대 높이	너무 길어지지 않도록
`auto`	콘텐츠 기준 자동	유동적 대응

---

8. 반응형 디자인에서의 조합

```
section {
  width: 90%;
  max-width: 1200px;
  min-width: 300px;
  margin: 0 auto;
}
```

- 너무 작거나 너무 커지지 않도록 유도
- 가운데 정렬(`margin: 0 auto`)과 함께 자주 사용

결론

> `min-*`, `max-*` 속성은 **고정된 레이아웃보다 유연하고 적응력 있는 디자인**을 가능하게 해준다.
> 다양한 해상도, 콘텐츠 양, 사용자 설정에 대응하는 데 필수적인 구성 요소다.

반응형 이미지: `object-fit`, `srcset`

현대 웹에서 이미지는 **디바이스 해상도, 비율, 뷰포트 크기**에 따라 유연하게 조정되어야 한다.
이를 위해 CSS의 `object-fit`과 HTML의 `srcset` 속성은 **반응형 이미지 구현의 핵심 도구**다.

---

1. `object-fit` (CSS)

개요

`object-fit`은 **이미지나 비디오 요소가 할당된 컨테이너 박스에 어떻게 맞춰질지를 결정**하는 속성이다.

문법

```
img {
  object-fit: contain | cover | fill | none | scale-down;
}
```

주요 값

값	설명
`fill`	기본값, 비율 무시하고 요소에 맞춤
`contain`	**비율 유지**, 요소에 **완전히 들어오도록** 축소
`cover`	**비율 유지**, 요소를 **완전히 채우도록** 확대
`none`	이미지 원본 크기 그대로 표시
`scale-down`	`none`과 `contain` 중 더 작은 쪽 선택

예시

```
.image {
  width: 300px;
  height: 200px;
  object-fit: cover;
}
```

> 이미지가 지정된 영역을 넘치지 않도록 **크롭되며 꽉 차게** 보여줌

---

2. `object-position` (CSS 보조 속성)

- `object-fit`과 함께 사용
- 이미지를 **수직/수평으로 정렬** (기본: `center center`)

```
img {
  object-fit: cover;
  object-position: top left;
}
```

---

3. `srcset` (HTML)

개요

`srcset` 속성은 HTML `<img>` 요소에서 **해상도나 화면 너비에 따라 다른 이미지를 로딩**하도록 지정하는 속성이다.
고해상도 대응(Retina), 데이터 절약, 성능 최적화에 매우 유용하다.

기본 문법

```
<img
  src="image-default.jpg"
  srcset="image-480.jpg 480w, image-800.jpg 800w"
  sizes="(max-width: 600px) 480px, 800px"
  alt="반응형 이미지">
```

설명

- `srcset`: 이미지와 해당 이미지가 적용될 너비 조건(`파일 경로 [공백] 크기w`)
- `sizes`: 각 미디어 조건별로 표시할 이미지의 너비를 알려줌
- 브라우저는 이 정보로 최적의 이미지를 자동 선택함

Retina 대응 예시 (`x` 단위 사용)

```
<img
  src="image@1x.jpg"
  srcset="image@2x.jpg 2x, image@3x.jpg 3x"
  alt="고해상도 대응 이미지">
```

> 브라우저는 디바이스의 **픽셀 밀도(dpr)**를 기준으로 적절한 이미지 선택

`picture` 요소와의 조합 (고급)

```
<picture>
  <source media="(min-width: 768px)" srcset="image-large.jpg">
  <source media="(max-width: 767px)" srcset="image-small.jpg">
  <img src="fallback.jpg" alt="예시 이미지">
</picture>
```

> **뷰포트 크기별로 완전히 다른 이미지**를 제공 가능

---

4. 실전 예시

```
<img
  src="default.jpg"
  srcset="
    small.jpg 600w,
    medium.jpg 1200w,
    large.jpg 2000w"
  sizes="(max-width: 600px) 100vw,
         (max-width: 1200px) 50vw,
         1000px"
  style="width: 100%; height: auto; object-fit: cover;"
  alt="반응형 이미지">
```

---

5. `object-fit` vs `background-size`

항목	`object-fit`	`background-size`
대상	`<img>`, `<video>` 등	배경 이미지 (`background-image`)
사용 위치	CSS 속성	CSS 속성
주요 값	`contain`, `cover`, `fill` 등	`contain`, `cover`, `auto` 등
포지셔닝	`object-position`	`background-position`

결론

> ✅ `object-fit`: 이미지의 **표시 방식 제어 (crop, fill, fit 등)**
> ✅ `srcset`: 디바이스 환경에 따라 **적절한 리소스를 선택 로딩**

둘을 조합하면 **디자인 일관성 유지 + 로딩 최적화 + 고해상도 대응**이 가능하다.