15. 반응형 웹과 HTML

15.1 뷰포트 설정 (`<meta name="viewport">`)

— 반응형 웹의 핵심: 모바일 화면에서의 렌더링 범위와 배율 조정

`<meta name="viewport">` 태그는 모바일 브라우저가 HTML 문서를 어떻게 표시할지를 정의하는 중요한 메타 태그다. 뷰포트(viewport)란 사용자의 스크린에서 실제 콘텐츠가 보여지는 영역을 의미하며, 이 태그를 사용하지 않으면 모바일에서 데스크톱 기준으로 웹페이지가 렌더링되어 축소된 화면이 보이게 된다.

✅ 1. 기본 문법

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

✅ 2. 속성 설명

속성 이름	설명
`width`	뷰포트의 너비를 설정. 일반적으로 `device-width` 사용
`initial-scale`	처음 로딩 시 줌(확대/축소) 배율 (1.0 = 100%)
`maximum-scale`	최대 줌 배율
`minimum-scale`	최소 줌 배율
`user-scalable`	사용자의 줌 동작 허용 여부 (`yes` / `no`)

🔹 예시 1: 표준적인 반응형 웹

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

- 대부분의 반응형 웹사이트가 사용하는 가장 기본적인 설정
- 뷰포트를 **기기 너비에 맞추고**, **기본 확대율은 100%**

🔹 예시 2: 확대/축소를 제한하는 경우

```
<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0,
user-scalable=no">
```

- 사용자가 줌(핀치 줌 등)을 하지 못하게 제한
- 주의: **접근성 측면에서 비추천** (시각 장애인을 고려해야 함)

🔹 예시 3: 데스크탑 뷰 유지 (💥 절대 사용하면 안 되는 예)

```
<!-- 모바일에서 데스크톱처럼 보이게 함 → 사용성 ↓↓↓ -->
<meta name="viewport" content="width=1024">
```

- 모바일에서도 1024px 기준으로 축소해서 보여주므로 **폰트가 작게 보이고 사용이 불편**
- 최신 모바일 친화 웹에서는 거의 사용하지 않음

✅ 3. 시각적 비교

설정 없음	`<meta name="viewport">` **있음**
모바일 화면이 축소되어 전체 페이지가 보임	모바일 기기 너비에 맞게 확대되어 콘텐츠가 잘 보임
터치 요소 작고 클릭 불편	터치 요소 정상 크기로 렌더링됨
기본 980px 기준 뷰포트	실제 기기 너비 사용 (예: 375px)

✅ 4. CSS와의 연계

- `@media` 쿼리를 이용한 **반응형 디자인**을 적용할 때 필수

```
@media (max-width: 768px) {
  body {
    font-size: 14px;
  }
}
```

📌 만약 `<meta name="viewport">`가 없다면 이 media query는 **의미가 없어짐**.
브라우저가 "768px 이하"라는 조건을 **적용하지 않음**.

✅ 5. 실전 팁

- 반드시 `<head>` 안에 위치시켜야 함
- `width=device-width, initial-scale=1.0`은 거의 필수
- 접근성 향상을 위해 `user-scalable=no`는 **되도록 지양**

✅ 요약

> `<meta name="viewport">`는 모바일 환경에서 웹페이지의 **레이아웃과 확대 배율을 제어하는 메타 태그**로,
> 반응형 웹을 만들기 위해 **가장 먼저 추가해야 하는 요소**이다.
> `width=device-width`와 `initial-scale=1.0` 조합은 거의 모든 모바일 웹의 표준 시작점이다.

15.2 picture 요소 (`<picture>`, `srcset`, `media`)

— 반응형 웹을 위한 이미지 최적화 태그의 핵심

`<picture>` 태그는 다양한 화면 크기나 해상도에 따라 **자동으로 다른 이미지 소스를 선택**할 수 있게 해 주는 HTML5 요소다. 주로 다음의 두 가지 목적에 사용된다:

1. **디바이스 크기별 이미지 제공** (모바일 vs 데스크탑)
2. **고해상도(Retina) 또는 낮은 대역폭 사용자에 맞는 이미지 제공**

✅ 1. 기본 구조

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

- `<source>`: 조건에 맞는 이미지 소스를 제공
- `media`: CSS의 `@media`와 동일한 조건 사용
- `srcset`: 해당 조건에 맞는 이미지 경로
- `<img>`: fallback 이미지. 모든 조건에 해당하지 않으면 사용됨

✅ 2. `srcset` + `sizes`를 `<img>` 단독으로 쓰는 예

```
<img
  src="default.jpg"
  srcset="small.jpg 480w, medium.jpg 800w, large.jpg 1200w"
  sizes="(max-width: 600px) 480px, (max-width: 1024px) 800px, 1200px"
  alt="예제 이미지">
```

항목	설명
`srcset`	각 이미지와 그 이미지의 너비 지정
`sizes`	현재 뷰포트에 대해 어느 너비의 이미지를 선택해야 할지 지정

📌 브라우저는 해당 조건을 기반으로 **최적의 이미지**를 자동 선택해 로딩한다.

✅ 3. Retina 대응 (2x 해상도 대응)

```
<img
  src="image.png"
  srcset="image.png 1x, image@2x.png 2x"
  alt="레티나 대응 이미지">
```

- 브라우저는 디바이스 픽셀 비율(`window.devicePixelRatio`)에 따라 고해상도 이미지를 선택함

✅ 4. WebP와 JPG 자동 선택

```
<picture>
  <source srcset="image.webp" type="image/webp">
  <source srcset="image.jpg" type="image/jpeg">
  <img src="image.jpg" alt="브라우저에 따라 형식 선택">
</picture>
```

- WebP를 지원하는 브라우저는 더 가벼운 `.webp` 사용
- 지원하지 않는 경우 `.jpg`로 fallback

✅ 5. `<picture>`의 동작 순서

브라우저는 아래 순서대로 판단한다:

1. 위에서 아래로 `<source>` 태그를 **순차적으로** 확인
2. 조건(media, type 등)이 **최초로 일치하는** `<source>`**를 사용**
3. 일치하는 것이 없으면 `<img>`의 `src`를 사용

✅ 6. CSS와의 차이점

항목	`<picture>`	CSS `background-image`
접근성 (`alt`) 제공	✅ 지원	❌ 미지원
SEO 및 크롤링 대상	✅ 포함	❌ 제외
이미지 최적화 자동 선택	✅ 지원	❌ 수동 처리 필요
반응형 처리	✅ 뛰어남	⚠️ 미디어쿼리 따로 필요

✅ 7. 실전 예제: 고해상도 대응 + 반응형

```
<picture>
  <source srcset="banner@2x.webp 2x, banner.webp 1x" type="image/webp" media="(min-width: 768px)">
  <source srcset="banner@2x.jpg 2x, banner.jpg 1x" type="image/jpeg" media="(min-width: 768px)">
  <img src="banner.jpg" alt="배너 이미지">
</picture>
```

- 브라우저가 WebP 지원 & 해상도 2x → `banner@2x.webp`
- WebP 미지원 → `banner@2x.jpg`
- 모바일 → `<img src="banner.jpg">`

✅ 요약

핵심 요소	설명
`<picture>`	반응형 & 형식별 이미지 분기
`<source>`	조건 기반 소스 제공 (media, type)
`srcset`, `sizes`	자동 최적화 & 고해상도 대응
`<img>`	fallback 요소 + 접근성 제공

✅ 마무리 Tip

- `srcset`은 `<img>`와 `<source>` 모두에서 사용됨
- `<picture>`는 `<video>`와 매우 유사한 구조
- `<canvas>`와 함께 쓰면 동적 렌더링 가능

15.3 미디어쿼리 연동

— HTML과 CSS의 연계를 통한 반응형 웹 디자인의 핵심 기술

미디어쿼리(Media Query)는 HTML 요소 자체가 아니라 **CSS에서 사용하는 조건문**이다.
브라우저의 **뷰포트 크기**, **해상도**, **장치 방향**, **색상 심도** 등 다양한 특성에 따라
다른 스타일을 적용할 수 있도록 한다.

HTML에서는 `<link>` 또는 `<style>` 태그를 통해 **미디어쿼리를 연동한 CSS를 적용**한다.

✅ 1. 외부 스타일시트에서의 미디어쿼리 연동

```
<link rel="stylesheet" href="style-desktop.css" media="screen and (min-width: 1024px)">
<link rel="stylesheet" href="style-mobile.css" media="screen and (max-width: 767px)">
```

- 브라우저 너비가 1024px 이상이면 `style-desktop.css` 적용
- 767px 이하일 경우 `style-mobile.css` 적용
- 불필요한 리소스를 조건적으로 로딩할 수 있다는 점에서 **성능상 이점**

✅ 2. 내부 스타일 (`<style>`) 태그와 함께 사용

```
<style>
  body {
    font-size: 16px;
  }

  @media (max-width: 768px) {
    body {
      font-size: 14px;
    }
  }
</style>
```

- 뷰포트가 768px 이하일 때만 font-size를 작게 조정
- **HTML 내부에서 직접 CSS 조건부 적용 가능**

✅ 3. `<picture>` 또는 `<source>`에서 media 속성과 연동

```
<picture>
  <source srcset="banner-large.jpg" media="(min-width: 1024px)">
  <source srcset="banner-small.jpg" media="(max-width: 1023px)">
  <img src="banner-default.jpg" alt="배너 이미지">
</picture>
```

- HTML 구조 자체가 **media query**에 따라 다른 이미지 리소스를 선택
- `media` 속성은 CSS의 미디어쿼리와 **문법이 동일**

✅ 4. 자주 쓰는 Media Query 조건 정리

조건	예시	설명
최대 너비	`@media (max-width: 768px)`	768px 이하일 때 적용
최소 너비	`@media (min-width: 1024px)`	1024px 이상일 때 적용

조건	예시	설명
해상도	`@media (min-resolution: 2dppx)`	레티나(2x) 이상에서만 적용
방향	`@media (orientation: portrait)`	세로 모드일 때 적용
색상	`@media (prefers-color-scheme: dark)`	다크모드 사용자에게 적용

✅ 5. 실전 예시: 반응형 내비게이션

```
<style>
  nav ul {
    display: flex;
  }

  @media (max-width: 768px) {
    nav ul {
      flex-direction: column;
    }
  }
</style>

<nav>
  <ul>
    <li>홈</li>
    <li>소개</li>
    <li>연락처</li>
  </ul>
</nav>
```

- 데스크톱에서는 가로 메뉴
- 모바일에서는 세로 메뉴

✅ 6. 반응형 디자인의 핵심 흐름

1. HTML 구조: 시맨틱하게, 의미 중심으로 설계
2. CSS 스타일: 미디어쿼리로 각 해상도에 맞는 스타일 정의
3. `<meta name="viewport">` 설정: 모바일 기준의 뷰포트 확정
4. `<picture>` 및 `srcset`: 이미지 리소스도 반응형으로 관리
5. 조건부 `<link>` 및 `<style>` 활용: 성능 최적화까지 고려

✅ 요약

요소	설명
`<link media="">`	조건부 외부 스타일시트 로딩
`<style> + @media`	HTML 내 인라인 반응형 스타일 정의
`<source media="">`	HTML 콘텐츠 자체의 조건별 분기 (이미지 등)
미디어쿼리 핵심 역할	기기별 레이아웃 최적화, 성능 개선, 접근성 향상

15.4 모바일 최적화 마크업 전략

— 모바일 친화적인 HTML 구조와 작성법의 모든 것

모바일 퍼스트(Mobile First)는 더 이상 선택이 아닌 **표준 전략**이다.
HTML 마크업 자체도 **모바일 환경에서 빠르게 렌더링되고, 사용성이 높으며, 접근성까지 고려**되어야 한다.

아래는 모바일 최적화를 위한 핵심 마크업 전략을 체계적으로 정리한 것이다.

✅ 1. 뷰포트 설정은 필수

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

- **모바일 디바이스의 실제 너비**에 맞춰 콘텐츠가 렌더링되도록 함
- 이 설정이 없으면 데스크탑 레이아웃이 축소되어 보이며 사용성 최악

✅ 2. 모바일에 적합한 구조 설계

항목	전략
레이아웃	시맨틱 태그 사용 (`<main>`, `<section>`, `<article>`)
콘텐츠 길이	짧은 단락, 핵심 문장 위주
이미지	`srcset`, `<picture>` 활용하여 고해상도와 저해상도 대응
내비게이션	`<nav>`는 명확하게, 가능한 토글 방식으로 구현

✅ 3. 버튼과 입력 필드 크기 고려 (터치 최적화)

- **버튼 최소 크기: 44px × 44px** (애플 권장)
- `type="button"` 또는 `type="submit"` 명시
- `<label for="id">`을 이용한 넓은 클릭 영역 확보

```html
<label for="agree">
  <input type="checkbox" id="agree"> 동의합니다
</label>
```

✅ 4. 폰트 크기와 단위

- CSS에서 `px` 대신 `rem`, `em` 사용 권장
- `<meta name="viewport" ...>`가 없으면 폰트가 너무 작아짐

```css
body {
  font-size: 1rem; /* 기준 폰트 크기 16px */
}
```

✅ 5. 이미지 및 미디어 대응

- `img`는 **반응형 스타일 적용** 필수

```css
img {
  max-width: 100%;
  height: auto;
}
```

- `<picture>` 또는 `srcset`을 통해 디바이스 해상도별 이미지 제공

✅ 6. `<input>`의 모바일 최적화 type 설정

```html
<input type="email"> <!-- 이메일 키보드 표시 -->
<input type="tel">   <!-- 숫자 키패드 표시 -->
<input type="number"> <!-- 숫자 입력 전용 -->
```

- 사용자가 **터치로 입력할 때 편한 인터페이스** 제공
- 키패드 자동 전환으로 UX 향상

✅ 7. 폼 요소에 자동완성 및 도움말 추가

```html
<input type="email" autocomplete="email" placeholder="이메일 주소">
<input type="password" autocomplete="current-password">
```

- 빠른 입력을 돕고, 브라우저 자동완성 기능 활성화

✅ 8. 로딩 속도 최적화

항목	전략
이미지	WebP, AVIF 등 사용
스크립트	`defer`, `async`로 로딩 최적화
CSS	`@media`를 이용해 조건부 로딩
폰트	너무 많은 외부 폰트 지양

```
<script src="main.js" defer></script>
```

✅ 9. 화면 리플로우 방지 (배치 변화 최소화)

- 레이아웃 요소에 **고정 높이 또는 비율 설정** → 렌더링 안정성 증가
- `<img>`에 `width`와 `height` 속성 직접 명시

```
<img src="img.jpg" alt="..." width="400" height="300">
```

✅ 10. 다크모드 대응 (선택)

```
@media (prefers-color-scheme: dark) {
  body {
    background: #111;
    color: #eee;
  }
}
```

- 사용자 기기의 다크모드 환경에 따라 자동 적용

✅ 11. 접근성과 SEO 고려

- `<header>`, `<main>`, `<nav>`, `<footer>` 구조 준수
- `alt`, `aria-label`, `role`, `label` 등 추가 마크업으로 **스크린 리더 대응**
- `<meta name="description">`, `<title>`, `<h1>`을 포함한 의미 있는 마크업 구조

✅ 12. 테스트 도구

도구	설명
Lighthouse	구글의 성능/접근성/SEO 분석

도구	설명
Chrome DevTools	모바일 뷰포트 시뮬레이션
BrowserStack / LambdaTest	다양한 기기에서 실제 테스트
Axe	접근성 검사 플러그인

✅ 요약 정리

전략 항목	핵심 요약
뷰포트 설정	`<meta name="viewport">`는 필수
시맨틱 구조	`<main>`, `<nav>` 등으로 의미 부여
반응형 콘텐츠	이미지, 폰트, 버튼 등 모두 크기 대응
폼 UX 개선	`type`, `autocomplete`, `label` 적극 활용
접근성 보강	`aria`, `role`, `alt`, `label` 사용
성능 최적화	script 최적 로딩, 리소스 최소화