2025.5.2 ~ 5.9, Clab 프론트엔드 보고서(Goal2)

0. 성과

- 메인페이지
  : 반응형 디자인, 상호작용 및 그리드 연습
- Ableton
  : Symentic 태그로 구조 정리, 위치 및 크기 미세 조정, 네비게이션 바 애니메이션 추가

1. 배운 점

a. 반응형 디자인

반응형Responsive 디자인이란, 사용자마다 서로 다른 기기를 사용할 때 기기의 환경(화면 크기 등)에 맞게 실시간으로 적응하는 형태의 디자인이다.
비단 크기만을 다루지 않는 이유는 눈에 보이는 스크린만이 웹페이지의 대상이 아닐 수도 있기 때문이다.

- [개념] viewport ··· 반응형 디자인의 기본
  : 실제 보여지는 화면 크기(viewport)
  : 기본적으로 브라우저는 화면의 가로 크기를 범용 desktop 크기인 980px에 맞춘다. 때문에 meta 태그로 이를 기기 크기와 일치시키고, 스케일 또한 1:1비로(절대단위 px의 크기 일치) 일치시키는 것이 사용자 경험UX에 최선이다.
  : width=device-width는 DIPDevice Independent Pixels, 해상도와 무관한 물리적 크기 단위로 페이지 가로 길이를 기기 가로 길이에 맞춘다. · · · Explanation & Sample
  : initial-scale은 CSS에서 사용하는 단위 픽셀과 DIP 사이의 비율을 조절한다. maximun/mininum-scale이나 user-scalable이라는 속성도 있으나 권장하지 않는다.
  : 결론적으로, 화면 크기가 달라질 때 불편할 정도로 깨지는 게 아니라면 viewport는 항상 화면 크기에 맞추는 것이 좋다.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    …
```

```
    <!-- 반응형 viewport 설정 -->
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    …
  </head>
  …
```

- **[개념] media query … 반응형 디자인의 적용**

  : CSS 속성을 조건부로 적용하기 위한 질의문query. 좁게는 화면 크기부터 크게는 보여지는 매체Media까지 조건으로 설정 가능하다.

  : @을 앞에 붙여 사용하는 확장/앳 규칙At-rules 중 하나이다. @viewport도 사실 이 중 하나.

  : 매체 유형은 all(전부, 기본), screen(일반적인 디지털 기기), print(인쇄될 문서), projection(프로젝트 등 프레젠테이션 화면), tv, speech 등이 있다.

  : 문법과 대표적인 사용법은 예제와 함께 첨부한다.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    …
    <!-- print.css가 인쇄될 문서(print)에 적용되도록 함 -->
    <link rel="stylesheet" href="print.css" media="print">
    <style>
        /* link - media="print"는 아래와 같다. 이후 다른 쿼리도 html
에서도 그대로 사용 가능 */
        @media print {
            /* print.css의 내용들 */
        }
    </style>
  </head>
  …
```

- media-query: 여러 조건들

```
/* 가로 크기에 따른 조건; viewport 기준 */
@media (max-width: 600px) { ... } /* 가로가 600px보다 좁을(이하) 때
*/
@media (min-width: 601px) { ... } /* 가로가 601px보다 클(이상) 때 */
@media (min-width: 700px) { ... } /* 가로가 700px보다 클 때; 601px때
```

```
보다 나중에 적용 */
@media proejection and (min-width: 300px) and (max-width: 500px),
all and (min-width: 1000px) {
    /* 프로젝터 상에서 가로 크기가 300~500px이거나(,가 or)
    모든 기기에서(all) 1000px 이상일 때 적용 */
}

/* 기기가 놓인 상태(모바일, 태블릿 등)에 따른 조건 */
/* 그냥 min-width가 넓을 때로 감지해도 되나, 의미론적으로 편리 */
@media (orientation: portrait) { ... } /* 기기가 세로 방향(높이 > 너
비)일 때 */
@media (orientation: landscape) { ... } /* 기기가 가로 방향(높이 <
너비)일 때 */

/* 가로세로 비율에 따른 조건 */
@media (aspect-ratio: 16 / 9) { ... } /* 기기의 가로:세로 비율이
16:9일 때 */
```

- **[개념] viewport 기준 크기 단위**
  : px 처럼 직접 웹페이지 상에서 쓸 수 있는 단위이다.
  : vw, vh = viewport width, viewport height. 각각을 100%로 두었을 때 1vw = 1% or viewport width이다.
  : vmin, vmax = viewport의 너비/높이 중 작은(vmin)/큰(vmax) 값의 백분율.
  : 예시 - viewport 너비 100px마다 0.5px씩 커지는 폰트 크기. media-query로 최댓값 설정.

```
.responsive-font { font-size: calc(16px + 0.5vw); }
@media (min-width: 1000px) {
    .responsive-font { font-size: 21px; }
}
```

b. 애니메이션

연속적인 CSS 상태 변화를 나타내는 방법.

- **transition: (propertyName) (duration=0) (timing-function=ease) (delay=0)**
  : 연속적인 숫자형 값을 가지는 거의 모든 CSS 속성에 대해, 값 변화가 일어날 때 timing-function 방식에 따라 duration 시간 동안 연속적으로 변화시켜주는 속

성.
: 대상 속성의 단위는 같아야 한다.
: timing-function은 아래와 같다. 1~5는 Easing Functions라 한다.

1. ease(기본값): 느렸다가, 빨랐다가, 다시 느려짐. = cubic-bezier(0.25, 0.1, 0.25, 1.0)
2. linear: 속도 일정 = cubic-bezier(0.0, 0.0, 1.0, 1.0)
3. ease-in: 시작은 느리고 점점 빨라짐 = cubic-bezier(0.42, 0.0, 1.0, 1.0)
4. ease-out: 시작은 바르고 점점 느려짐 = cubic-bezier(0.0, 0.0, 0.58, 1.0)
5. ease-in-out: ease+시작과 끝이 강조되어 느려짐 = cubic-bezier(0.42, 0.0, 0.58, 1.0)
6. cubic-bezier(x2, y2, x3, y3): 조절점 $P_1 = (0,0), P_4 = (1,1)$일 때 2,3번째 조절점 $P_2 = (x_2, y_2), P_3 = (x_3, y_3)$에 따른 베지어 곡선을 duration 범위에 적용한다. 즉 정의역 $[0s, durations]$, 치역 $[이전값, 이후값]$ 의 베지어 함수의 결과값을 변화에 이용한다. → 곡선 시뮬레이터
7. steps(단계 수, 전환 시점): 각 단계에 전환 시점(start/end)에 주어진 단계만큼 값을 끊어서 변화시킨다.

- transform -> Reference
  : 초기 상대를 기준으로 3차원 상의(x,y=가로세로추 / z=화면의 법선 축) 이동(translae), 회전(rotate), 기울임(skew), 크기 조절(scale), 원근 조절(perspective)이 가능하다.
  : transform-origin으로 변환 기준점을 설정할 수 있다. 명시적으로 top left를 넣거나, 실제 값을 넣거나, center을 넣을 수 있다.
  : transition은 물론 keyframes를 이용한 커스텀 애니메이션 사용 시에도 연속값으로서 쓰인다.
  : 또한 top/left 등을 이용한 상대좌표 조절과 translate는 언뜻 비슷하나, translate는 리플로우Reflow, 레이아웃 갱신를 발생시키지 않고 단순히 그려지는 위치만 조절하여 렌더링 비용과 애니메이션 성능에서 우월하며 레이아웃을 어그러뜨리지도 않아 유용하다. 거기에 더해 다른 transform과 자연스러운 병용 또한 가능하다.
- @keyframes animation-name { steps / from-to { ... }} + animation: (name) (duration) ...
  : 현재 재생 비율(%)에서의 상태, 또는 시작(from)-끝(to) 상태 변화를 이용해 애니메이션을 정의할 수 있다.
  : animation 속성으로 @keyframes를 통해 정의한 애니메이션을 추가 옵션과 함

께 사용 가능하다.
: 참고자료

2. 막힌 부분

- 색 선택, '적당한' 크기 조절(디자인 감각 부족)
- 이벤트 리스너

3. 노하우나 팁

반응형 디자인에서 사용할 수 있는 여러 기법들

출처

- Web-dev

사진과 같은 요소 크기

- 대상: img, embed, object, video
- width: 100% (+ display: block)으로 사진이 부모 요소의 가로 크기에 맞추는 것을 기본값으로 설정한다.

레이아웃

display: grid, flexbox를 이용해 절대 크기(px)이 아닌, 요소 간 배치 관계로 viewport에 자연스레 종속되는 느슨한 레이아웃을 사용한다.

기기별 width 크기 정리(intial-scale=1)

- 모바일: 320 ~ 480px
- 태블릿: 768 ~ 1025px
- PC: 최소 960~1200px, max-width 설정으로 가독성 확보 고려

반응형 소스 변경

- picture 태그의 media, srcset 사용.

```
<picture>
  <source media="(min-width: 800px)" srcset="head.jpg, head-2x.jpg
2x">
```

```
  <source media="(min-width: 450px)" srcset="head-small.jpg, head-small-2x.jpg 2x">
  <img src="head-fb.jpg" srcset="head-fb-2x.jpg 2x" alt="a head carved out of wood">
</picture>
```

CSS 디자인 규칙

1. id 식별자(#)는 재사용성을 떨어뜨리므로 대신 클래스 식별자(.)를 권장한다.
2. !important는 정말 중요한 경우라도 자연스런 상속관계를 어지럽힐 수 있으므로 사용하지 않는다.
3. 모든 디자인에 앞서 먼저 페이지의 대략적인 형태를 구상한 뒤, 요소 간 디자인 상속관계를 고려하여 작성한다.
4. 크기는 절대 단위(px) 대신 em(부모 요소에 대한 백분율), rem(body에 대한 백분율), vw, vh와 같은 상대 단위를 사용한다.
5. 따옴표가 필요한 요소는 생략 가능하더라도 생략하지 않는다. (url 등)
6. 인자 개수에 따라 위치별 의미가 달라지는 단축 속성(font, border 등) 사용은 권장하지 않는다.

CSS 디자인 순서

1. 문서의 시각적 요소를 먼저 설계한다.
2. 필요한 문서 구조를 구축(html)하고 대략적인 레이아웃을 구성한다.
3. 필요한 CSS 속성들을 파악한다.
4. 서로 다른 media 환경에서도 동작할 수 있는 속성을 선별한다.
5. 상속 관계를 적극적으로 이용하고, 재사용성을 고려하여 CSS를 작성한다.
6. (+α, 권장) 서로 다른 media 환경에서의 시트를 디자인한다.