디자인 및 기능 설명서

목차.1 출처

Ameen Fahmy
@ameenfahmy

Owen Young
@owenyoungg

Efe Kurnaz
@efekurnaz

배경 사진 출처
unsplash.com

폰트명
Metropolis

폰트 출처
1001fonts.com

폰트명
Neo둥근모

폰트 출처
noonnu.cc

목차.2 디자인

① 첫번째 매뉴 PORTFOLIO 는 아이디어를 연상하는 전구 배경을 사용했고, 호버 했을시 위 배경과 잘 어우러지는 반딧불이 효과를 적용함.

② 두번째 매뉴 INTERVIEW 는 제목부터 흔하디 흔한 About Me 대신 귀여운 아바타와 인터뷰를 하는 컨셉에따라 작명했고, 호버 했을시 카메라 셔터가 터지는 듯한 효과이후 본인의 ~~아름다운~~ 사진이 뛰워짐.

③ 세번째 매뉴 GALLERY 는 미술관 입구를 연상하는 배경을 사용했고, 호버 했을시 화려한 배경색과 유사한 그라데이션을 적용한 빗방울이 떨어짐.

공용.

④ 호버 했을시 매뉴 폰트 색상 변경 및 크기 증가.

④ 클릭 했을시 해당 매뉴의 좌우 폭이 100% 증가하고 매뉴 제목 또한 넓어지는 매뉴 폭을 따라 증가하다 최후 0.5초간 단독으로 증가함으로 입체적 효과를 줌.

⑤ 클릭 애니메이션이 끝남과 동시에 Fade-out 효과를 주고 Fade-out 효과 종료후 해당 매뉴의 컨텐츠 환면에 Fade-in 효과를 적용함.

목차.3 메인 매뉴 기능

menu[0], menu[2]는
동일 구조임으로 생략.

```
menu[0].addEventListener('mouseenter', particleHandler);

function particleHandler(){
    mTitle[0].classList.add('menu-hover');

    const particleContainer = document.createElement('div');
    particleContainer.classList.add('particle-container');
    menu[0].appendChild(particleContainer);

    let particleCount = 150;
    let i = 0;

    while(i < particleCount){
        const particle = document.createElement('span');
        particle.classList.add('particle')
        particleContainer.appendChild(particle);

        let particleSize = Math.random() * 5;
        let particlePosX = Math.floor(Math.random() * window.innerWidth);
        let particleDelay = Math.random() * -20;
        let particleDuration = Math.random() * 5;

        particle.style.width = 5 + particleSize + 'px';
        particle.style.height = 5 + particleSize + 'px';
        particle.style.left = particlePosX + 'px';
        particle.style.animationDelay = particleDelay + 's';
        particle.style.animationDuration = 6 + particleDuration + 's';

        i++
    }

    menu[0].onmouseleave = ()=> {
        menu[0].removeChild(particleContainer);
        mTitle[0].classList.remove('menu-hover');
    }
}
```

menu[0]에 mouse-enter시 이벤트 발생

menu[0]의 (mTitle[0]) == (제목)에 클래스 부여
(.menu-hover) = font-size : 소폭 증가

 태그 생성후 (.particle-container) 클래스 부여.
클래스 주입된 를 menu[0]의 자손으로 지정.
해당 클래스는 추후 반딧불의 컨테이너 역할로 사용.

(particleCount) = 반딧불이의 총 개수
및 하단 while 문의 break 역할

 태그 생성후 (.particle) 클래스 부여
해당 클래스는 태그에 반딧불이 색상 적용
클래스 주입된 은 상단에 미리 생성된
(.particle-container) 의 자손으로 지정.
자손 지정된 (.particle)에 Math.random 메소드로
크기, 생성 위치, 생성 속도, 지속 시간 등을 지정.

menu[0] 에서 mouse-leave시 이벤트 발생
menu[0]의 자손인 (.particle-container) 삭제
(container) 가 삭제되면서 자손인 (.particle) 또한
자동적으로 삭제함. (mTitle[0])의 (.menu-hover)
삭제후 본래 지정 폰트 크기로 돌아감

```
menu[1].addEventListener('mouseenter', shutterHandler);

function shutterHandler(){
    mTitle[1].classList.add('menu-hover');

    const shutter = document.createElement('div');
    const picture = document.createElement('div');
    shutter.classList.add('shutter');
    picture.classList.add('picture');
    menu[1].appendChild(shutter);
    menu[1].appendChild(picture);


    setTimeout(() => {
        picture.classList.add('show');
    }, 300);

    menu[1].onmouseleave = ()=>{
        menu[1].removeChild(shutter);
        menu[1].removeChild(picture);
        mTitle[1].classList.remove('menu-hover');
    }
}
```

menu[1]에 mouse-enter시 이벤트 발생

 태그 2개 생성 후 첫 에 (.shutter) 클래스 부여.
두번째 는 (.picture) 클래스 부여 이후 두 클래스를
menu[1]의 자손으로 지정. 자손으로 지정됨가 동시에
(.shutter) = animation 0.3s {opacity: 0% to 100%}
0.3초 뒤 플래시 효과 이후 (.picture)에 (.show) 클래스
추가함으로 프로필 사진이 등장

menu[1]에 mouse-leave시 이벤트 발생
menu[1]의 자손인 (.shutter)와 (.picture)을 삭제
menu[1]의 제목인 (mTitle[1])의 (.menu-hover)
클래스 삭제후 본래 지정 폰트 크기로 돌아감.

목차.3 메인 매뉴 기능 - 2


```
<section class="menu menu-1">
<section class="menu menu-2">
<section class="menu menu-3">
```

```
menuWrap.addEventListener('click', clickHandler);

function clickHandler(e) {
    let elem = e.target;
    while(!elem.classList.contains('menu')){
        elem = elem.parentNode;
        if(elem.nodeName == "BODY") {
            elem = null;
            return;
        }
    }

    for(i=0; i<menu.length; i++) {
        mTitle[i].classList.add('hide');
        if(elem.classList.contains('menu-'+(i+1))) {
            mTitle[i].classList.remove('hide');
            mTitle[i].classList.add('menu-click');
            menu[i].classList.add('expand');
            doSetTimeout(i);
        }
    }

    if(elem.classList.contains('menu-1')){
        sliderSelector(0)
    }else if(elem.classList.contains('menu-3')){
        sliderSelector(1)
    }
}
```

```
function doSetTimeout(i){
    setTimeout(() => {
        menu[i].classList.add('fade-out');
        setTimeout(()=>{
            menuWrap.classList.add('hide');
            content[i].classList.add('flex');
            content[i].classList.add('fade-in');
            entry();
        },1400);
    }, 1000);
}
```

(menuWrap)을 click시 이벤트 발생.
(menuWrap)은 이전 menu[0], [1], [2] 를 감싸는 부모 요소입니다.

event.target 한 요소에 (.menu) 클래스를 찾을때 까지 target된 해당 요소의 값을 그의 상위노드 값으로 바꾸고, 만일 상위노드로 등반중 <BODY> 를 만난다면 null값을 대입하고 return으로 종료한다.

menu의 길이만큼 for문을 돌아 모든 (mTitle[i])의 제목에 (.hide) 클래스를 부여해 display : none 값을주고, 상위 보라 영역에서 찾은 (.menu) 클래스의 클래스 목록중 (.menu-(i+1))의 클래스가 존재한다면 해당 (선택된) (menu-(i+1))의 (mTitle)에 부여된 (.hide) 를 회수하고 (.menu-click) = (font-size : 증가) 을 부여함.
menu[i]에 (.expand) = (flex : 100%) 클래스 부여.
doSetTimeout(i) 메소드 실행.

클릭된 요소에 (.menu-1)의 클래스명이 존재한다면 sliderSelector(0) 의 파라미터 값을 가진 메소드 실행
이외 (.menu-3)라면 sliderSelector(1) 으로 실행.

1초 뒤 menu[i]의 (.fade-out) = {animation 1.5s opacity : 100% to 0%} 클래스 추가, 이후 menu[i]의 opacity가 0% 에 도달했을 1.4초 뒤에 (menuWrap)에 (.hide) 부여.
(menuWrap)이 display : none 으로 가려진 동시점 (content[i])에 (.flex) = (display : flex(block))와 (.fade-in) = {animation 1.5s opacity : 0% to 100%} 의 두 클래스를 부여.

목차4. 홈메이드 슬라이더

```
function sliderSelector(i){

    const slider = document.querySelectorAll('.slider');
    const sliderControl = document.querySelectorAll('.slider-control');

    slider[i].addEventListener('mousemove', sliderHover);

    let sliderTime;

    function sliderHover() {
        clearTimeout(sliderTime);
        sliderControl[i].classList.add('show');
        sliderControl[i].classList.add('slider-in');
        sliderControl[i].classList.remove('slider-out');

        sliderTime = setTimeout(() => {
            sliderControl[i].classList.remove('slider-in');
            sliderControl[i].classList.add('slider-out');
            sliderTime = setTimeout(() => {
                sliderControl[i].classList.remove('show');
                }, 400);
            }, 1000);
    }
```

파란 태두리 내부 (slider) 영역이며 mouse-move 시 이벤트 발생

이벤트 발생시 (sliderControl)에 추가
(.show) = display : block
(.slider-in) = animation 0.5s
{opacity: 0% to opacity: 100%}

이벤트 발생후 1초뒤 호버 감지 안될시 재사용을 위해 (.slider-in) 제거후
(.slider-out) = animation 0.5s
{opa: 100% to opa: 0%} 추가.
(slider-out) 실행 0.4초 뒤
opacity: 0% 시점 (.show) 제거.
(sliderControl) = display : none

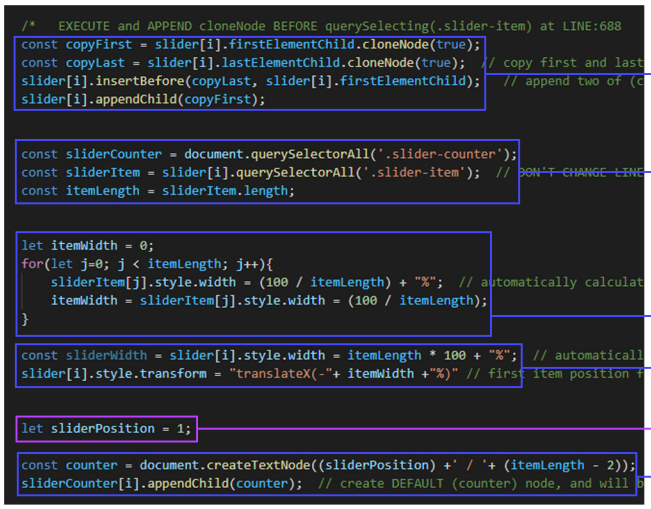
이벤트 재발생시 기존 예약되어있던 Timeout 초기화


```
sliderControl[i].addEventListener('mouseenter', sliderBtnEnter);

function sliderBtnEnter() {
    clearTimeout(sliderTime);
    sliderControl[i].classList.add('show');
}

sliderControl[i].addEventListener('mouseleave', sliderBtnLeave);

function sliderBtnLeave() {
    sliderTime = setTimeout(() => {
        sliderControl[i].classList.remove('slider-in');
        sliderControl[i].classList.add('slider-out');
        sliderTime = setTimeout(() => {
            sliderControl[i].classList.remove('show');
        }, 400);
    }, 1000);
}
```

(sliderControl) 영역에 mouse-enter시 이벤트 발생.
상위 보라영역 (sliderTime) 을 초기화 후
(sliderControl) 에 (.show) 추가함으로
영구적 display: block 유지.

(sliderControl) 영역에서 mouse-leave시 이벤트 발생.
재사용을 위해 (.slider-in) 제거 및 (.slider-out) 추가.
이후 0.4초뒤 (.show)를 제거함으로 opacity: 0%
지점에서 (sliderControl) = display : none;

목차4. 홈메이드 슬라이더 - 2



```
/*   EXECUTE and APPEND cloneNode BEFORE querySelecting(.slider-item) at LINE:688
const copyFirst = slider[i].firstElementChild.cloneNode(true);
const copyLast = slider[i].lastElementChild.cloneNode(true);  // copy first and last
slider[i].insertBefore(copyLast, slider[i].firstElementChild);   // append two of (c
slider[i].appendChild(copyFirst);

const sliderCounter = document.querySelectorAll('.slider-counter');
const sliderItem = slider[i].querySelectorAll('.slider-item');  // DON'T CHANGE LINE
const itemLength = sliderItem.length;

let itemWidth = 0;
for(let j=0; j < itemLength; j++){
    sliderItem[j].style.width = (100 / itemLength) + "%";  // automatically calculat
    itemWidth = sliderItem[j].style.width = (100 / itemLength);
}

const sliderWidth = slider[i].style.width = itemLength * 100 + "%";  // automaticall
slider[i].style.transform = "translateX(-"+ itemWidth +"%)" // first item position f

let sliderPosition = 1;

const counter = document.createTextNode((sliderPosition) +' / '+ (itemLength - 2));
sliderCounter[i].appendChild(counter);  // create DEFAULT (counter) node, and will b
```

(slider) 는 슬라이더 속에 있는 모든 아이템들의 부모 요소.
(slider[i])의 첫째 와 마지막 자손을 복사 후 본사된
첫 자손은 최후방에 배치, 반대로 본사된 마지막 자손은
최전방으로 배치함으로 차후 슬라이더 왕복시
setTimeout()을 활용한 속임수로 마지막 칸에서
첫 칸으로 혹은 첫 칸에서 마지막 칸으로 이동할때 쓰임.

(sliderItem)은 슬라이더 속 내용물을 담당한다.
(itemLength)속에 배열로 불러온 아이템들의 길이를 담는다.

(itemLength) 만큼 for문을 돌려 모든
(sliderItem)의 width 값을 100 / (itemLenth) 한다.
부모인 (slider)의 기준인 width : 100%를 (item)의 개수
만큼 나눈다면 (item)의 width는 실제 보이는
슬라이더의 (틀) 안에서 자동으로 꽉 채워진다.

(itemLength) * 100 = (slider)의 width이다
이또한 (item)의 개수가 +-@ 에 따라 자동으로 적용함

(sliderPosition) 슬라이더의 위치를 알려줄 소중한 친구

(sliderCounter) == (페이지 위치 UI)
의 최초 위치 값으로 (1 / ??) 으로 지정함

```
sliderControl[i].addEventListener('click', sliderBtnClick); // click event for (slider-buttons

function sliderBtnClick(e) {
    elem = e.target;

    while(!elem.classList.contains('slider-button')){
        elem = elem.parentNode;
        if(elem.nodeName == "BODY") {
            elem = null;
            return;
        }
    }

    if(elem.classList.contains('next-btn')) {

        slider[i].style.transition = "0.8s ease";
        slider[i].style.transform = "translateX(-"+ (sliderPosition + 1) * itemWidth + "%)";

        if(sliderPosition === (itemLength - 2)){
            sliderPosition = 0;  // set (sPosition) to (0) here, and will eventually become (1)
            sliderControl[i].style.pointerEvents = "none";  // prevent spam clicking so the (c
            setTimeout(() => {
                slider[i].style.transition = "0s";
                sliderControl[i].style.pointerEvents = "auto";
                slider[i].style.transform = "translateX(-"+ sliderPosition * itemWidth + "%)";
            }, 750);
        }
        sliderPosition++;
    }

    if(elem.classList.contains('prev-btn')) {

        slider[i].style.transition = "0.8s ease"
        slider[i].style.transform = "translateX(-"+ (sliderPosition - 1) * itemWidth + "%)";

        if(sliderPosition === 1){
            sliderPosition = itemLength - 1;  // subtract (-1) from (itemLength) here and will
            sliderControl[i].style.pointerEvents = "none";  // prevent spam clicking so the (c
            setTimeout(()=> {
                slider[i].style.transition = "0s";
                sliderControl[i].style.pointerEvents = "auto";
                slider[i].style.transform = "translateX(-"+ sliderPosition * itemWidth + "%)";
            }, 750);
        }
        sliderPosition--;
    }

    const counter = document.createTextNode((sliderPosition)+' / '+ (itemLength - 2)); // subtr
    sliderCounter[i].appendChild(counter);  // create after (counter) and append to (sliderCoun

    if(sliderCounter[i].contains(counter)) {  // remove before (counter) from LINE:705
        sliderCounter[i].removeChild(sliderCounter[i].firstChild);
    }
}
```

(sliderControl)에 click시 이벤트 발생

event.target 이 (.slider-button)의
클래스를 찾을때 까지 작동

event.target에 (.next-btn)의 클래스가 있다면
우측으로 (sliderPosition + 1) * (itemWidth) % 이동.
(sliderPosition)이 마지막 아이템 위치에서 발동됐다면
(sliderPosition)에 = 0의 위치를 대입후 하단에서
0.75초간의 setTimeout을 기다리는 동안 코드블럭 끝의
sliderPosition++ 가 먼저 성행 적용됨으로
현재 Position 값은 1, 최종으로 setTimeout이
실행됨으로 첫 아이템의 자리로 이동됨.

상단 (.next-btn) 우측의 반대인 (.prev-btn) 좌측 버튼

(counter)텍스트 노드속에 현재 (sliderPosition)의
위치 값을 대입후, (sliderCounter) == (페이지 위치 UI)
의 자손으로 지정. 그리고 (sliderCounter) 속에 이전에
지정 되었던 (counter) 노드를 삭제함으로
click 이벤트 발생시마다 (counter)를 재갱신.

목차.5 INTERVIEW 디자인 및 기능

포토샵으로
배경 수제작

캐릭터 제작은
avatarsinpixels.com
에서 무료 제작후
표정만 수정하여 사용.

선택을 감싸는 부모
(userBox)

대화 하는 느낌을 주기위해 좌측에서
우측으로 텍스트 출력 및 한줄당
1초간의 간격을 두어 출력함.

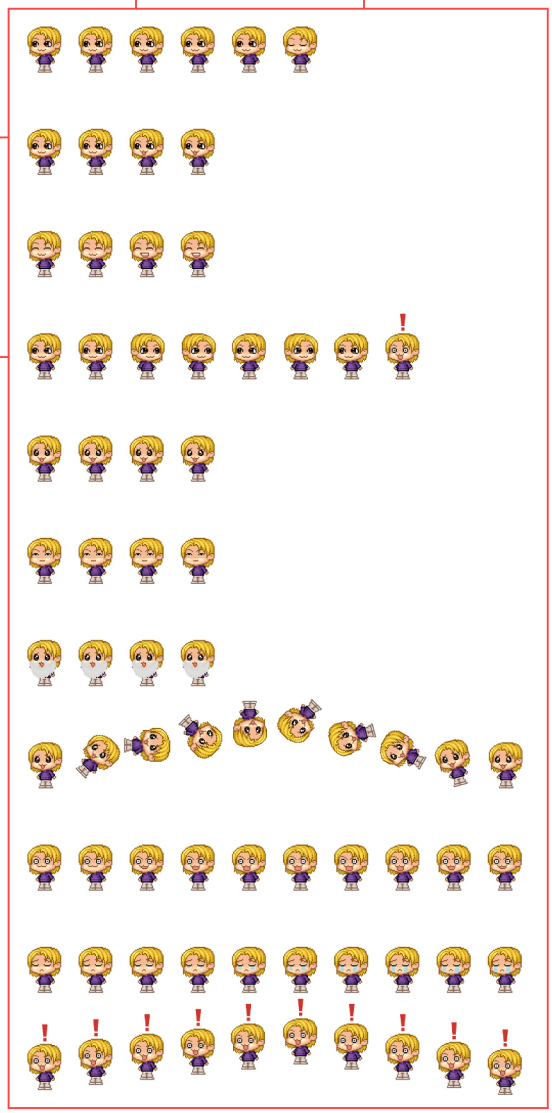
선택지에 따라 캐릭터의
애니메이션 및 대사 출력함

애니메이션을 위해
canvas에 사용할 한장의
sprite sheet입니다.

```
const canvas = document.getElementById('canvas');
const context = canvas.getContext('2d');
const canvasWidth = canvas.width = 250;
const canvasHeight = canvas.height = 500;

const hopeSheet = new Image();
hopeSheet.src = "./img/hope-sprite.png";

const frameWidth = 250;
const frameHeight = 500;

let ImageFrame = 0;
let staggerFrames = 0;
const spriteAnimations = [];
const animationStates = [
    {
        name: 'idle',
        frames: 6,
    },
    {
        name: 'talk',
        frames: 4,
    },
```

canvas에 사용할
sprite sheet를 불러옴

불러온 sprite속 각각 이미지의
X/Y 좌표값을 Object 형태로 저장.
이후 선택에 따라 해당 Object속
좌표값을 canvas로 출력.



스테이지 1 의 메소드

(userBox)에 click시 이벤트 발생

```
function select_1(){
    userBox.onclick = (e)=>{
        clearLog(e)

        if(e.target.classList.contains('A')){
            charState('cute', '40');
            animate();
            textBox();
            charLog('아니 이런 누추한 곳에 귀하신 분이');
            userLog('엣헴', 'A');
            select_2();

        }else if(e.target.classList.contains('B')){
            charState('angry', '30');
            animate();
            textBox();
            charLog('아니 이런 귀한 곳에 누추한 분이');
            userLog('힝..', 'B')
            select_2();
        }
    }
}
```

사용자가 선택하여 이벤트 발동시
이전 캐릭터의 말풍선과 유저의 선택을
전부 삭제 시키는 clearLog() 메소드

사용자가 선택한 요소에 (.A)
클래스가 존재한다면 실행.

sprite속의 좌표값을 담은 Object의
좌표값을 parameter로 불러와 canvas에
적용하고 애니메이션의 초당 프레임을
설정하는 charState() 메소드.
canvas를 실행하는 animate() 메소드.

캐릭터의 말풍선 생성 textBox() 메소드.
캐릭터의 텍스트 생성 charLog() 메소드.

사용자가 선택할 텍스트와
해당 선택의 클래스를 parameter로
설정후 생성하는 userLog() 메소드.

생성된 사용자의 선택은
스테이지 2의 규칙을 따름.