무언가를 만들었다

이인용@UPnL

42차 워크샵

180428

예상 청중

- 웹을 잘 모르는 사람
  - 신기해하기 (?)
  - '웹에서 이런 것들도 되는구나'를 알고 가기
- 웹을 잘 아는 사람
  - 신기해하기 (??)
  - 얘가 뭐로 뭘 짰는지 한 번 들어보기

육하원칙

- 누가
- 언제
- 어디서
- 무엇을
- 어떻게
- 왜

육하원칙

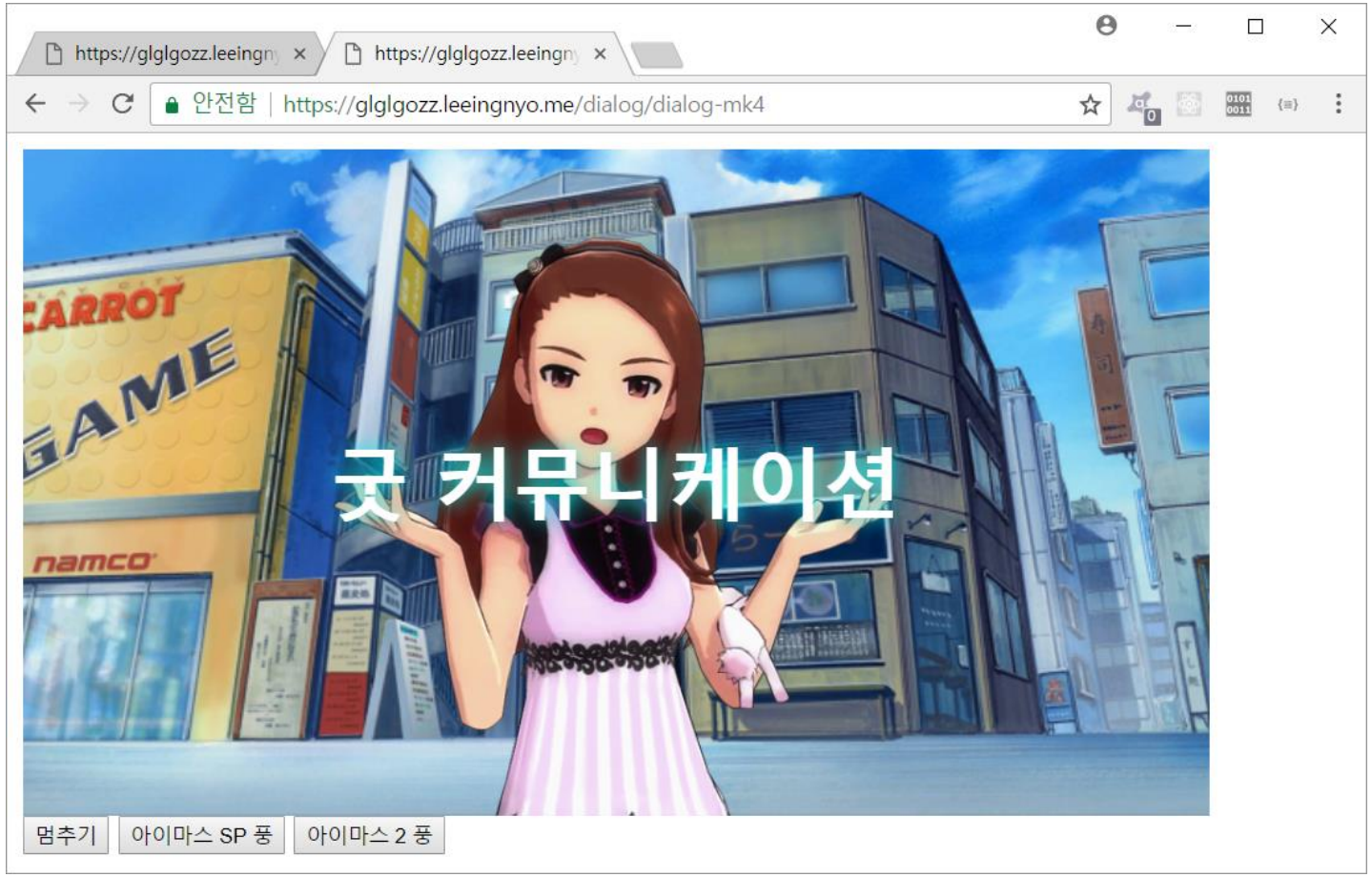
- 내가
- 몇 달에 걸쳐서
- 짬이 나는 곳에서
- 무엇을
- 어떻게
- 왜

무엇을

- https://glglgozz.leeingnyo.me/dialog/dialog-mk4
- 같이 참여해주세요

- 혹시 위 url이 안 되면 아래로 찾아봐주세요
- https://glglgozz.leeingnyo.me/dialog/old/mk4/dialog-mk4
- 조작법
  - 방향키나 마우스 클릭
  - z가 스킵

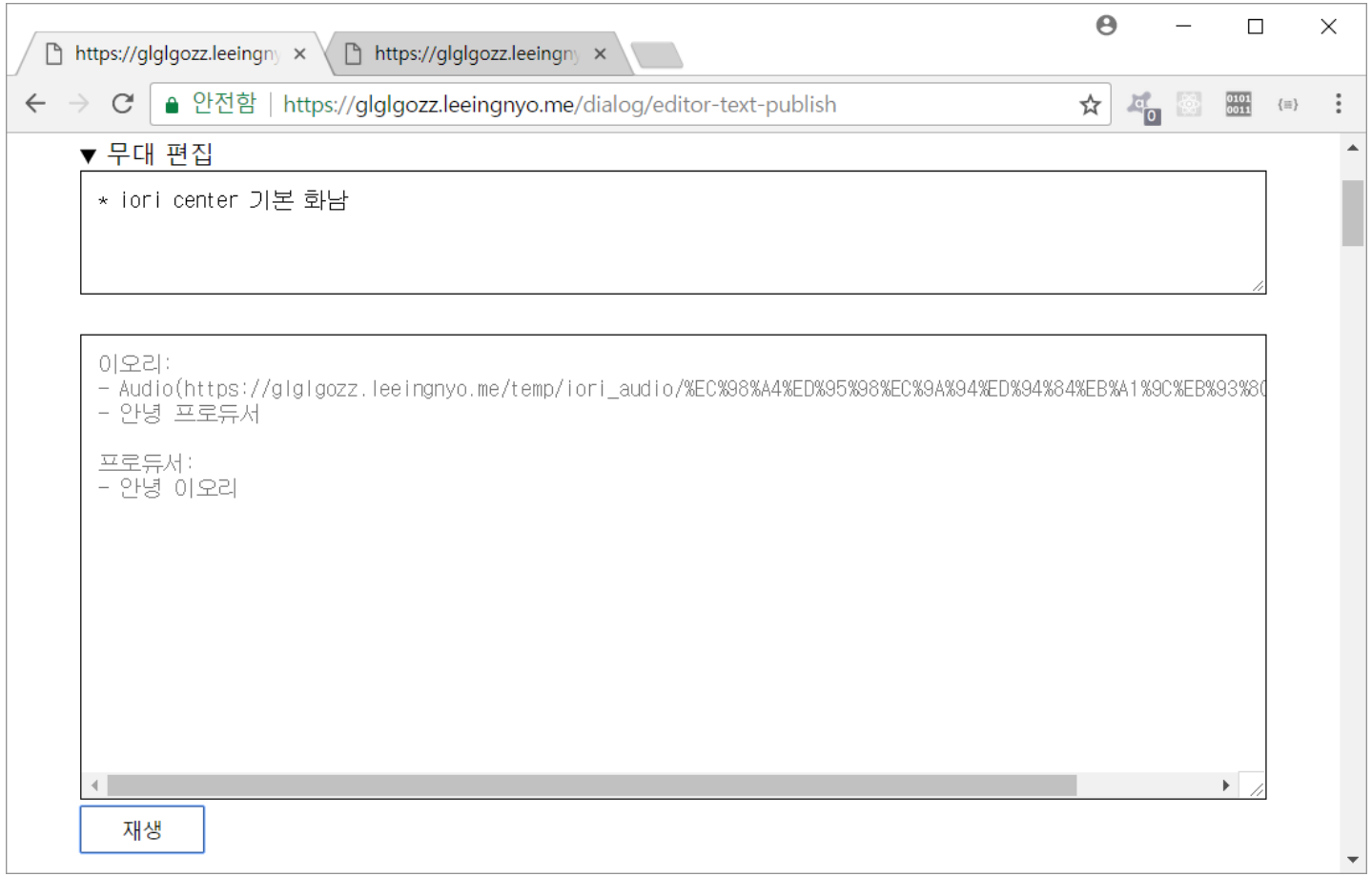
무엇을


https://glglgozz.leeingn
https://glglgozz.leeingn
안전함 | https://glglgozz.leeingnyo.me/dialog/dialog-mk4
CARROT
GAME
namco
굿 커뮤니케이션
멈추기
아이마스 SP 풍
아이마스 2 풍

무엇을

- 편집기도 있습니다
- https://glglgozz.leeingnyo.me/dialog/editor-text-publish
- https://glglgozz.leeingnyo.me/dialog/iori-one-day-script.txt

무엇을


https://glglgozz.leeingny
https://glglgozz.leeingny
안전함 | https://glglgozz.leeingnyo.me/dialog/editor-text-publish
▼ 무대 편집
* iori center 기본 화남
이오리:
- Audio(https://glglgozz.leeingnyo.me/temp/iori_audio/%EC%98%A4%ED%95%98%EC%9A%94%ED%94%84%EB%A1%9C%EB%93%8
- 안녕 프로듀서
프로듀서:
- 안녕 이오리
재생

무엇을

```
이오리:
- Audio(https://glglgozz.leeingnyo.me/temp/iori_audio/%EC%98%A4%ED%95%9E
- 안녕 프로듀서

- Animation(iori, 뭐냐, 기본)
- 지금 목 마르니까 가서 내가 마실 걸 사와!
wait 1300
- Animation(iori, 기본, 기본)
-  뭔진 알고 있겠지?
프로듀서:
- Animation(iori, 깍지, 미소)
- 아 당연히 알고 있지. 이오리가 원하는 건...
choice
ㄴ> 100% 포도 주스 { "type": "change", "normal": "normal" } -> no
ㄴ> 100% 오렌지 주스 { "type": "change", "normal": "good" } -> correct
ㄴ> 100% 프로듀서 주스 { "type": "change", "normal": "bad" } -> mad

:no
프로듀서:
- 상큼한 100% 포도
- Animation(iori, 기본, 극혐)
-  주스잖아?
이오리:
- Animation(iori, 기본, 빡침)
- 키이잇! 아니거든. 이 바보 프로듀서!
```

어떻게

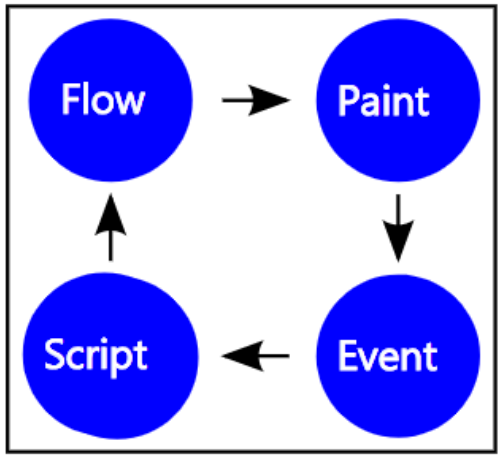
- HTML과 CSS 그리고 Javascript로!
- It's vanilla!

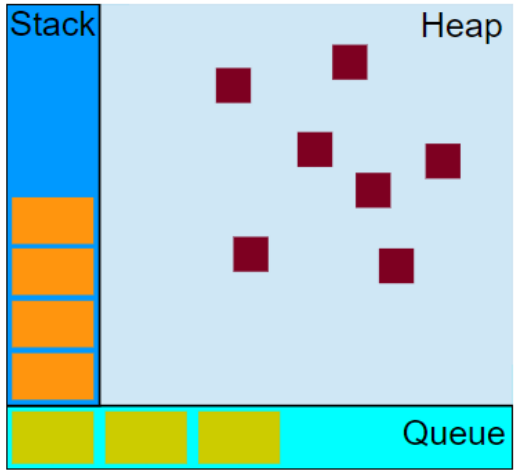
어떻게..?

- JS Event Loop로 FSM 구현
- JS Promise로 입력 대기, 스킵, 타임 오버 구현
- CSS 적극(?) 활용
- CSS Animation 활용
- PausableTimer 로 일시정지 구현

JS Event Loop

- 다 설명하다간... 간단히 설명
- 이벤트 발생 (다운 완료, 버튼 클릭, 포커스, 마우스 무브 등)
- 큐에 적재. 콜백이 스택에 올라가고 해당 스택을 빠져나갈 때까지 다른 이벤트들은 큐에 대기



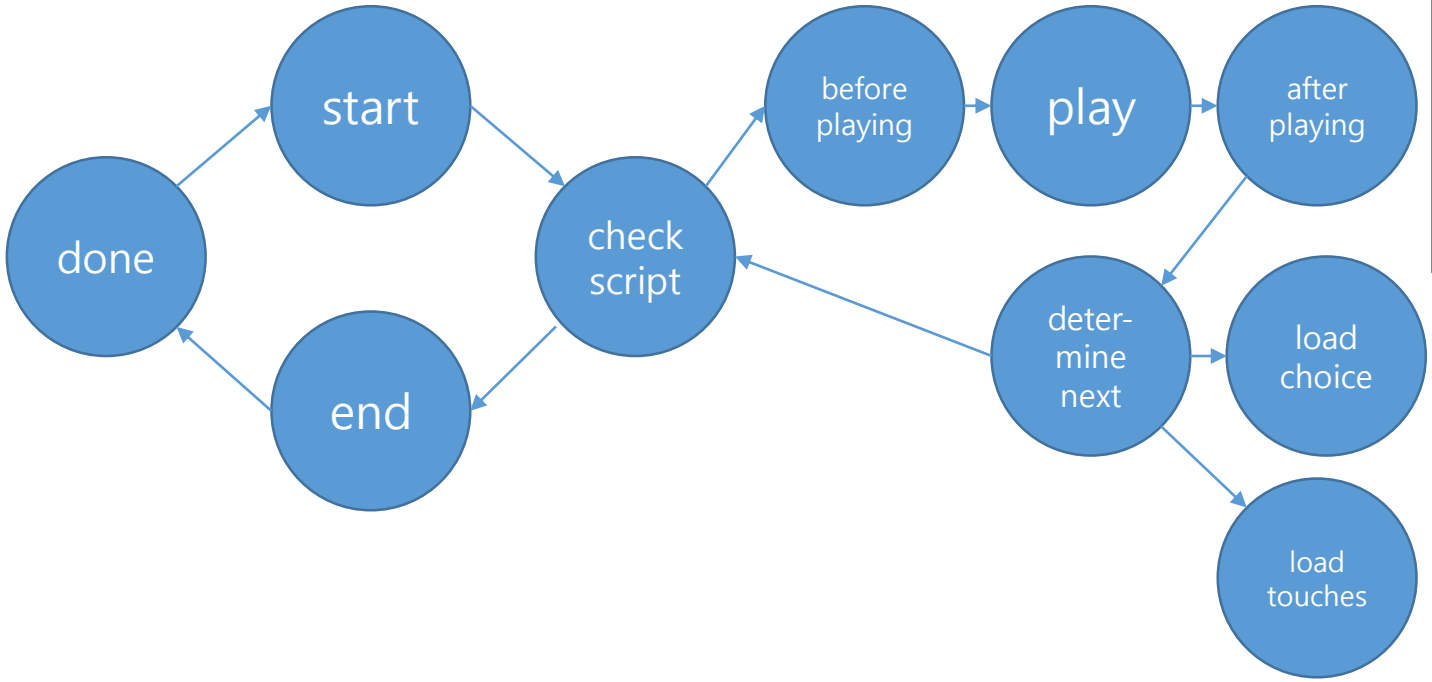


JS Event Loop로 FSM

- 각 state를 event로 정의
- 해당 event callback 끝에 다음으로 넘어갈 event를 발생시킴
- ex)

  start (무대 설정, bgm 재생) → check script 이벤트 발행
  check script (작업) → has next script ? before play : end 이벤트 발행

JS Event Loop로 FSM




```
state.addEventListener('check script', function () {
  logger.log('check script id ₩'%s₩'', currentScriptId);
  if (!(currentScriptId && scripts[currentScriptId])) {
    logger.log('no such script');

    dispatchEvent('end');
    return;
  }

  currentScript = scripts[currentScriptId];
  logger.log('current script is set');

  dispatchEvent('before playing');
});

//=============//

state.addEventListener('before playing', async function () {
  // before playing script
  // bgm change
```

생략

JS Event Loop로 FSM

- 이벤트 발행하기
  - EventTarget 에 event listener를 추가해준다.
  - eventTarget.dispatchEvent(event 객체) 로 이벤트를 발행할 수 있다.
- 이슈 발생
  - EventTarget 종류로는 Element, document, window가 있는데, 뒤의 2개는 콘솔에서 접근할 수 있고, Element를 document.createElement() 로 생성하면 document 트리에 추가되어 어찌됐든 접근할 수 있게 됨
- 숨길 방법은 없는가?

JS Event Loop로 FSM

- DocumentFragment

> `DocumentFragment`s 는 DOM 노드들 입니다. 이것들은 메인 DOM 트리의 일부가 되지 않습니다. 일반적인 유즈 케이스는 다큐먼트 조각을 생성하고, 엘리먼트들을 다큐먼트 조각에 추가하고 그 다큐먼트 조각을 DOM 트리에 추가하는 것입니다. DOM 트리 내에서 다큐먼트 조각은 그것의 모든 자식들로 대체됩니다.

```
function () {
  var state = document.createDocumentFragment();
}
// Reference Error: state is not defined
```

- 인데, 만들면서 보니까 createElement 해도 숨겨지는데..? 뭐지 ㅋ 암튼 무슨 이슈가 있어서 이거 씀

JS Event Loop로 FSM

- 장점
  - 그냥 함수로 호출했으면 콜스택이 터질 수도 있다. 하지만 Event Loop는 이벤트가 쌓이고, 해당 이벤트의 콜 스택이 제거되고 다음 이벤트의 콜 스택이 진행되니 터질 가능성이 낮아졌다
  - 그냥 함수 호출 말고 state 변수를 직접 바꾸거나 했으면 복잡한 시스템을 만들었어야 했지만, 이미 있는 것을 활용할 수 있게 됐다.

JS Promise

- 좋은 것. 이제는 없이 코딩하는 것이 힘들다. 콜백 헬...
- Promise는 다음과 같은 역할을 하는 객체이다.
  - Promise 객체는 생성할 때 함수를 하나 받는데, 그 함수 안에서 resolve나 reject를 할 수 있다.
  - resolve를 하면 then(func)의 func가 호출되며, reject를 하면 catch(func)의 func가 호출된다.

```
var promise = new Promise(function (resolve, reject) {
    setTimeout(function () { resolve(1); }, 1000);
});
promise.then(value => console.log(value)); // 1초 후에 resolve가 되니 1이 출력된다.
```

- 이렇게 asynchronous 한 일을 처리할 때 사용한다 (통신, IO).

JS Promise

- Promise에는 다음과 같은 일을 하는 함수도 있다.
  - Promise.race([Promise 객체의 배열])
    - 인자로 받은 객체들 중 하나라도 resolve 되면 그것의 값을 then()으로 받는다.
  - Promise.all([Promise 객체의 배열])
    - 인자로 받은 모든 객체들이 resolve 되면 그 값들을 배열로 받아 then()으로 받는다.
- Promise.race([1초 뒤에 1 resolve, 2초 뒤에 2 resolve])
  .then(value => console.log(value));
- 출력 결과는?
- 1이 먼저 resolve 되었으니 1

JS Promise

- Promise.all([1초 뒤에 1 resolve, 2초 뒤에 2 resolve])
  .then(value => console.log(value));
- 2초 뒤에 Array [1, 2] 가 출력된다. (모두 resolve 되면 넣은 순서대로의 결과 배열이 나옴)
- 이것들을 가지고 한 일
  - 해당 스크립트 아이템 모두 재생
  - 아이템 모두 재생되기 전에 스킵
  - 입력 대기 중에 타임아웃

JS Promise 활용

- 해당 스크립트 아이템 모두 재생
  - Promise.all([텍스트 뿌리기 promise, 음성 파일 재생 promise, 캐릭터 동작 변경 promise]).then(() => {
    다음 행동
    });
- 아이템 모두 재생되기 전에 스킵
  - Promise.race([Promise.all([아이템들]), 누르면 null resolve하는 promise])
    .then(value => {
    value가 array면 스킵 안 한 것, null이면 스킵한 것
    });

JS Promise 활용

- 입력 대기 중에 타임아웃
  - Promise.race([선택하면 선택한 것으로 resolve, 5초 뒤에 null로 resolve)])
    .then(value => {
    value 가 null이면 timeout 아니면 선택한 것으로 처리
    });

```
// wait for all items played
//  or pressing skip key
//  or touching the screen
Promise.race([
  Promise.all(currentScript.items.map(function (item, index) {
    return new Promise(function (resolve, reject) {
      switch (item.type) {
        case 'text': {
          registeredItems.push(timer.setTimeout(function () {
            var textCall = null, speakingEffectCall = null;
            if (!currentScript.hasAudio) {
```

CSS 활용

- CSS 적극(?) 활용

할 줄 아는 사람도 있겠지만
감히 어렵다고 말할 수 있다

이오리

지금 목 마르니까 가서 내가 마실 걸 사와!

위 3개는 다 같은 구조의
DOM입니다.

CSS 활용

- CSS Animation 활용
  - 선택할 때 반짝거리는 것, 선택하면 선택지가 떨어지는 것 모두 CSS Animation 으로 처리
  - Javascript는 class 추가 등으로 CSS 발동에만 관여

일시정지

- Web 에는 유니티처럼 object deactivate 같은 것이 없다.
- 근데 기획자가 일시정지를 만들어 달라고 함

- 일시정지에 커뮤니케이션 플레이어에 일어나야 할 일들
  - 다른 state 로 넘어가면 안 됨
  - 시간이 멈춰야 함

일시정지

- 일시정지 때 state에 event 가 발생하면 다른 곳에 쌓아뒀다가 resume 때 발생시키자
- Timer는 Web 환경에서 Global Timer가 돌고 있고 이를 제어하는 법은 없어서, 따로 제어하는 것을 만들게 됨
- PausableTimer 매커니즘
  - Global Timer에 등록
  - deactivate 할 때 Global Timer에서 해제하면서 남은 시간을 기억
  - activate 할 때 기억했던 시간으로 다시 Global Timer에 등록
  - 원리도 간단하고
  - https://github.com/Leeingnyo/PausableTimer 코드도 간단

왜?

- 이 프로젝트는 우선적으로 아이돌 마스터 커뮤니케이션들을 웹으로 포팅하는 것에 초점이 맞춰졌습니다
- 옛날 게임의 커뮤니케이션을 저장, 번역하는데 사진 캡쳐 후 코멘트 달기나 동영상으로 녹화하는 방법을 쓰고 있는데,
- 이는 분기가 있는 커뮤니케이션을 모두 보지 못 하거나 어색하게 보는 치명적인 단점이 있다
- 그래서 체험이 쉽게 하려고 웹으로 포팅 중!

왜?

- 침체기?인 아이돌 마스터 본가를 살리기 위해 유저들한테 직접 커뮤니케이션을 짜보게 하려고
  - ~~뭐가 안 나온다고 징징대지 말고 꼬우면 니네가 해봐라~~
- 창작 활동을 편하게 해주기 위해 에디터 개발

남은 일

- 3D 모델 적용
  - 하다못해 Live 2D 적용
  - 지금은 하나의 이미지지만 처리는 포즈와 표정으로 나누었음
- 아이돌 별로 Text to Speech 를 만들어서 대사를 말하게 하기
  - 스크립트에 실제로 읽을 일본어도 같이 적게 하는 법을 생각해야겠네...
- 둘 다 끔찍하게 오래 걸리고 혼자서는 힘들 것 같아서 안 할 것 같다

남은 일 (현실적인)

- 에디터 더 편하게 개발
- 만든 커뮤니케이션을 저장, 공유할 수 있는 서비스 만들기
- 커뮤니케이션 뿐만 아니라 아예 게임을 통으로 포팅해오기

미래에는

- 이 프로젝트는 덕질로 공부/개발하기 시리즈의 두번째 작품!

- 다음 발표 후보
  - 위키 사이트 만들기
  - 1 대 1 실시간 리듬 전략 게임

- 덕질하세요!

공부하고 싶으신 분은 보세요

- JS Event Loop
  - https://developer.mozilla.org/ko/docs/Web/Guide/Events/Overview_of_Events_and_Handlers
  - https://developer.mozilla.org/ko/docs/Web/JavaScript/EventLoop
  - http://meetup.toast.com/posts/89
- JS Promise
  - https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/Promise
  - https://developers.google.com/web/fundamentals/primers/promises?hl=ko

공부하고 싶으신 분은 보세요

- CSS Animation
  - https://www.w3schools.com/css/css3_animations.asp
  - https://developer.mozilla.org/ko/docs/Web/CSS/CSS_Animations/Using_CSS_animations