



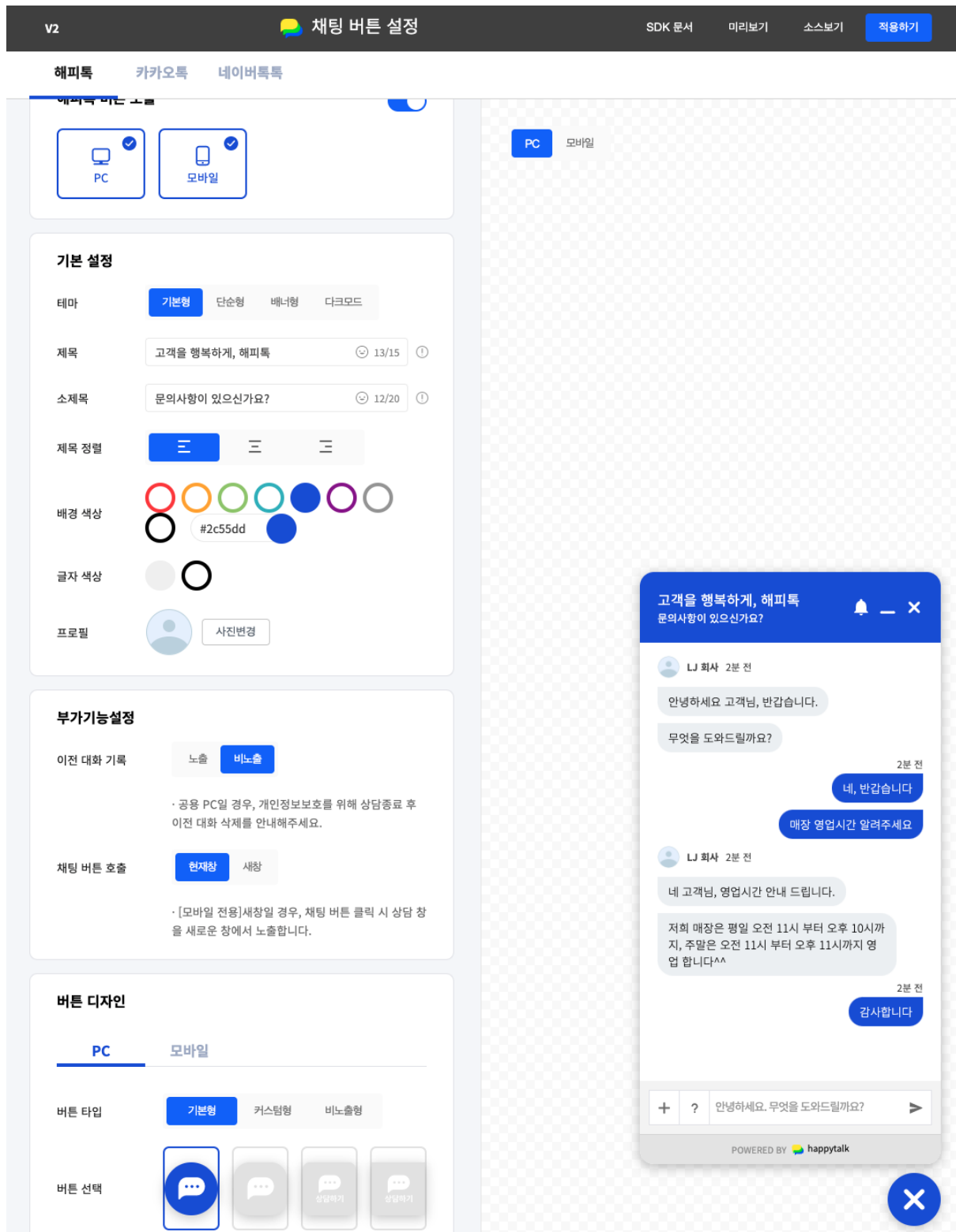


# happytalk-chat-front

 Created	@July 27, 2021 10:47 AM
 Tags	<div>Javascript SDK context api cypress emotion eslint next.js</div> <div>prettier react rollup.js socket.io-client typescript</div>
 Person	 Jaesung Lee



고객사 업체에 상담 채팅 솔루션을 제공하는 프로젝트입니다.

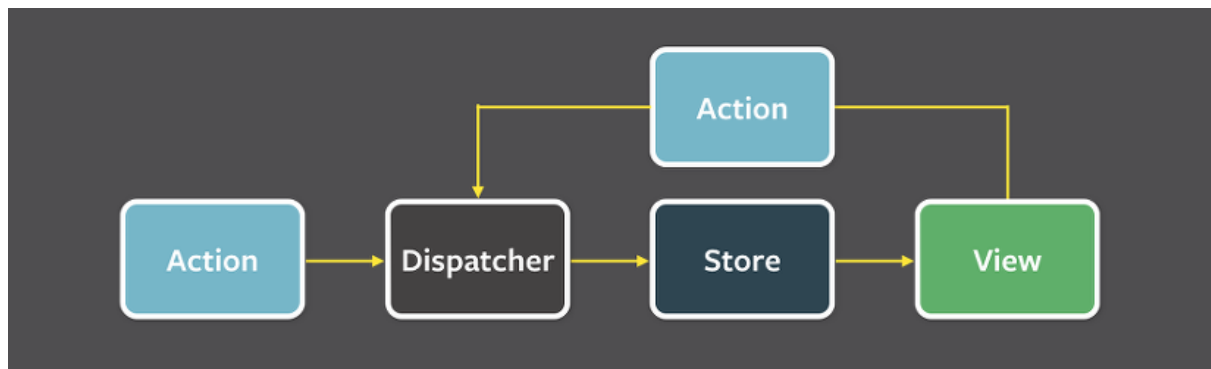
프로젝트에는 상담 채팅 버튼과 채팅창을 변경할 수 있는 60여가지의 에디터 기능과 고객과 상담원이 대화할 수 있는 채팅 기능을 제공합니다. 또한 고객사의 요청사항에 필요한 Javascript SDK 기능도 제공합니다.

## 개선 작업

기존 php 라라벨 프레임워크와 react 프로젝트를 하나의 서버에서 관리하는 형태로 인수인계 받아 서버 분리 작업 및 next.js 로 리팩토링 작업을 진행하였습니다. 다른 레포지토리에서 리팩토링을 진행하면서 버그 및 추가 기능사항등을 반영했으며 리팩토링이 끝난 뒤에 쿠버네티스로 인프라를 운영하기 위하여 Dockerfile 작성 뒤 서버개발자에게 제공하였습니다.

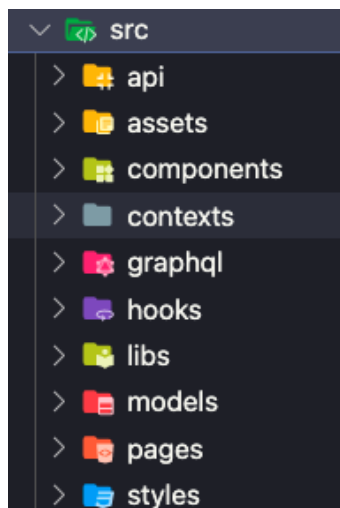
## Front-end Detail

리팩토링을 진행하며 외부 고객사 사이트에 가볍게 제공하기 위해 package 사용을 제한적으로 사용했으며 react에서 기본적으로 제공하는 Context API 를 활용하여 state 관리를 하였습니다.



단방향 데이터 흐름인 Flux 디자인 패턴 형태

## Scaffolding



- ▼ src ← 펼쳐주세요.
  - api → axios 통신

- assets → image 및 icon 보관
- ▼ components
  - @common → 공통 컴포넌트
  - error → 에러 관련 컴포넌트
  - chat → 채팅 관련 컴포넌트
  - editor → 에디터 관련 컴포넌트
- contexts → socket, chat, editor, auth 등 state 관리
- graphql → graphql 쿼리 작성 (최근 graphql 서버 생성, 추가 기능은 graphql로!!!)
- ▼ hooks
  - @common → 공통 hooks (useAuth, useOutsideClick, useInput 등)
  - chat → 채팅 관련 hooks 작성
  - editor → 에디터 관련 hooks 작성
- libs → axiosClient(interceptor handling), apolloClient, util 함수 작성 등
- models → 서버 응답 결과 및 context 상태 model type 정의
- pages → next routes
- styles → 공통 스타일 및 third part 라이브러리 styles

## Javascript SDK 제공

raw code형태로 코드 붙여넣기 할 수 있는 기능을 제공하여 고객사한테 제공했지만 운영하면서 불필요한 커뮤니케이션과 고객사의 개발자들이 잘 활용하지 못하는 사례를 접하여 좀 더 사용하기 쉬운 인터페이스로 변경하여 제공하기로 하였습니다.

<=> 소스보기

Deprecated

아래 소스를 복사하여 적용할 사이트의 </body> 위에 붙여넣기 하세요.

```
<script>
window.__ht_wc = window.__ht_wc || {};
window.__ht_wc.host = 'local-chat.happytalkio.com:3000';
window.__ht_wc.site_id = '4000001181'; // site_id
window.__ht_wc.site_name = 'MBI solution'; // 회사 이름
window.__ht_wc.category_id = '73559'; // 대분류 id
window.__ht_wc.division_id = '73560'; // 중분류 id

// 고정 및 Custom 파라미터 추가 영역, 파라미터가 여러개인 경우 ,(콤마)로 구분
// window.__ht_wc.params = 'site_uid=abcd1234,parameter1=param1';
// 앱 개발시, 웹뷰 Local Storage 사용가능하도록 처리

(function() {
  var ht = document.createElement('script');
  ht.type = 'text/javascript';
  ht.async = true;
  ht.src = ('https:' == document.location.protocol ? 'https://' : 'http://') +
    window.__ht_wc.host + '/web_chatting/tracking.js';
  var s = document.getElementsByTagName('script')[0];
  s.parentNode.insertBefore(ht, s);
})();
</script>
```

소스 복사하기

X

<=> 버튼 스크립트

X

아래 스크립트를 복사하여 적용할 사이트의 </head> 혹은 </body> 태그 안에 삽입해주세요.

```
<script id="happytalkSDK" src="http://local-
chat.happytalkio.com:3000/sdk/happytalk.chat.v2.min.js"></script>
<script type="text/javascript">
  var ht = new Happytalk({
    siteld: '4000001181',
    siteName: 'MBI solution',
    categoryId: '73559',
    divisionId: '73560',
    kakaoid: '@ntazbta4ntbiav8',
  });
</script>
```

\* 'SDK 문서' 활용 시 에디터에서 제공하는 기본적인 기능 외로 다양한 웹채팅 제어가 가능합니다.

스크립트 복사

```
// ex) basic example
<script id="happytalkSDK" src="https://design.happytalkio.com/sdk/happytalk.chat.v2.min.js"></script>
<script type="text/javascript">
  var ht = new Happytalk({
    siteId: 'SITE_ID',
    siteName: 'SITE_NAME',
    categoryId: 'CATEGORY_ID',
    divisionId: 'DIVISION_ID',
  });
</script>
```

## Javascript SDK

해피톡 웹채팅창을 사용하기 위한 안내서입니다. 최신 해피톡 Javascript Chat SDK 를 사용하기 위해서는 head 혹은 body 태그 안에 아래 스크립트 코드를 삽입해 주세요. 운영에 적용하기까지 4단계의 개발 과정을 거쳐 배포됩니다. 해피톡 대시보드에서 생성된 SITE\_ID, SITE\_NAME, CATEGORY\_ID, DIVISION\_ID 를 아래 예시처럼 삽입해 주세요. params에 site\_uid값을 넘길 경우 다른 <https://design.happytalkio.com/docs/basic>

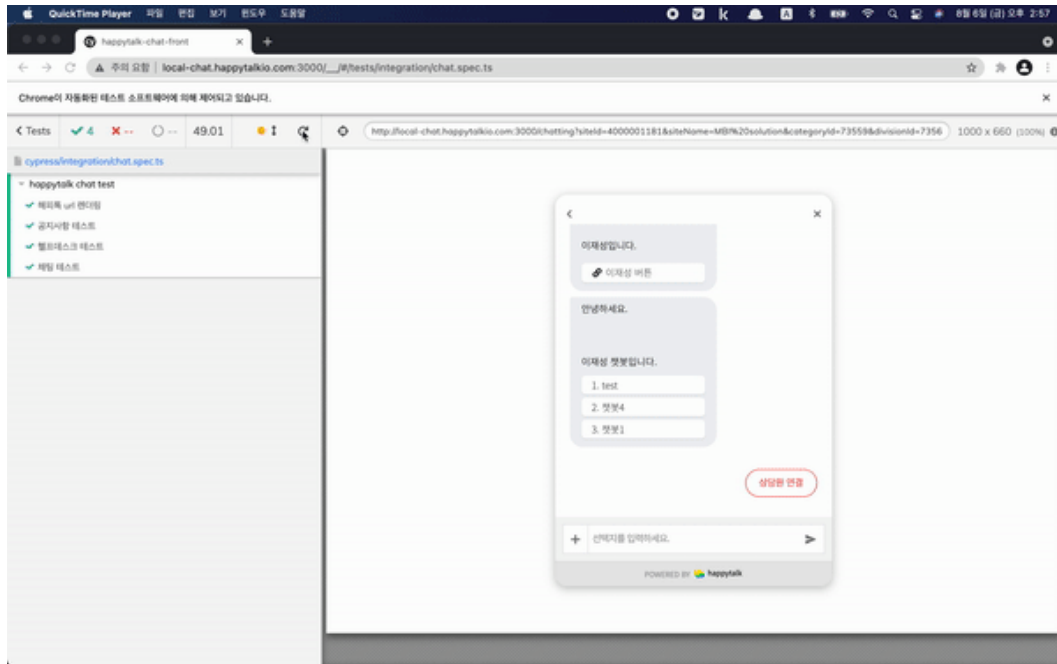
## Javascript Chat SDK 문서 참조

## HDS (Happytalk-Design-System) [링크](#)

회사에서 운영하고 있는 여러 도메인의 디자인 통일화를 위하여 npm으로부터 [happytalk-design-guide](#) 를 인스톨하여 사용하고 있습니다.

```
// ex)
import { Button, Calendar, Checkbox } from 'happytalk-design-guide';
```

## Cypress E2E 도입



운영 개발하면서 새로운 기능을 추가하거나 버그 수정하는 과정에서 일련의 QA 과정들을 반복해야하는 번거로움이나 사용자 행동을 예측한 end to end 테스트가 필요하다고 생각했습니다.

### 장점

- 러닝 커브가 낮아 도입하는데 하루, 이틀이면 충분함.
- 콘솔창에만 찍히는 시스템보다 UI 화면을 시각적으로 테스트하면서 확인해볼 수 있음.
- http 통신 로그도 남아서 어디 시점에서 api를 통신했는지 순차적으로 접근할 수 있음. (Timeline 기능)

### 단점

- 약간 무거운 감(?)이 있다.

### 도입하면서 겪었던 이슈

- cypress는 모든 테스트에서 기본적으로 session 과 localStorage 데이터가 리셋된다.

### ↓ 해결책 ↓

```
// cypress/support/commands.ts
let LOCAL_STORAGE_MEMORY = {};
```

```

Cypress.Commands.add('saveLocalStorage', () => {
  Object.keys(localStorage).forEach((key) => {
    LOCAL_STORAGE_MEMORY[key] = localStorage[key];
  });
});

Cypress.Commands.add('restoreLocalStorage', () => {
  Object.keys(LOCAL_STORAGE_MEMORY).forEach((key) => {
    localStorage.setItem(key, LOCAL_STORAGE_MEMORY[key]);
  });
});

```

```

// cypress/integration/chat.spect.ts
describe('happytalk chat test', () => {
  // 테스트간 localStorage 보존
  beforeEach(() => {
    cy.restoreLocalStorage();
  });

  afterEach(() => {
    cy.saveLocalStorage();
  });
});

```