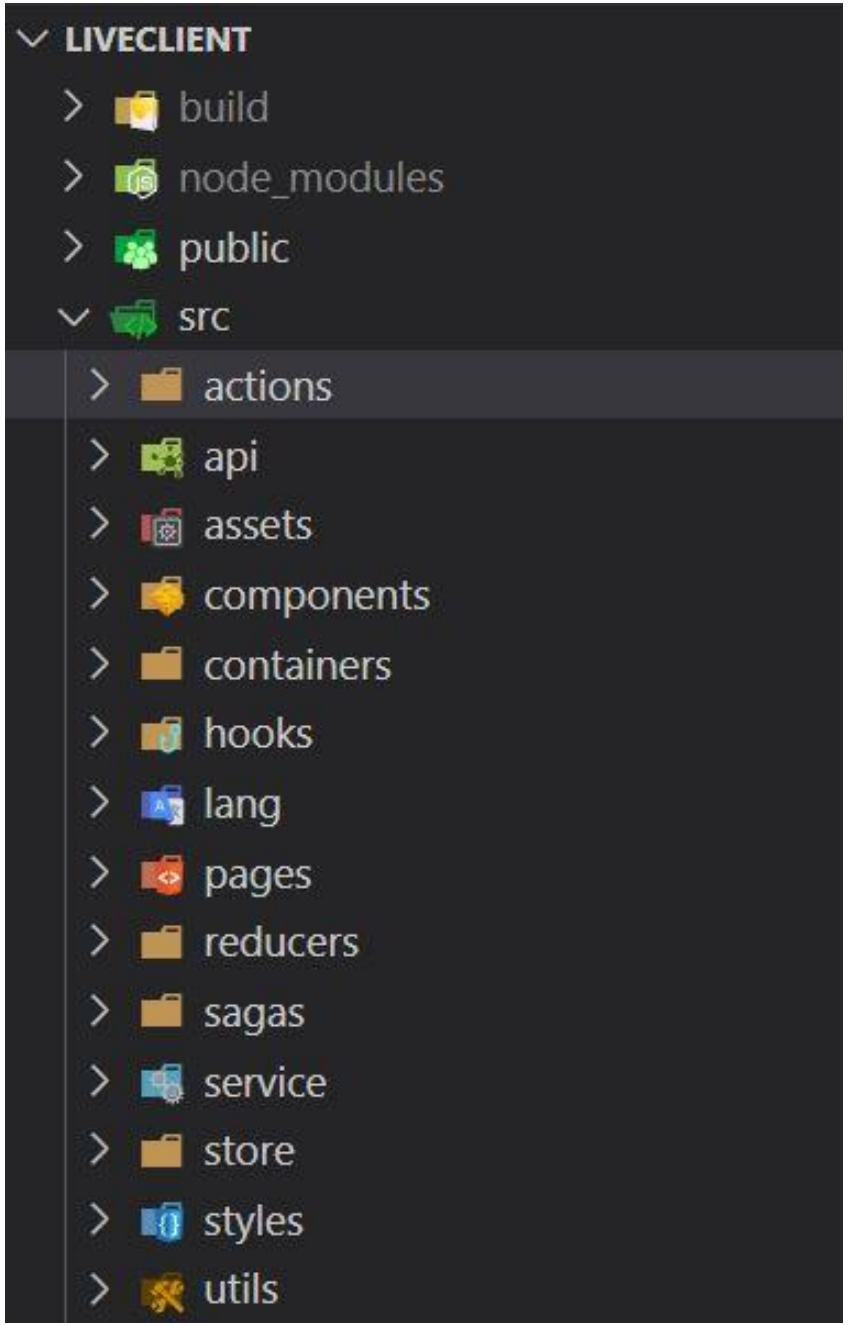
Kauction live page

REACT 디렉토리 구조



— // action creator (액션생성함수 모음), 상태에 어떤 변화가 필요할때 액션이 필요함

— // axios를 이용하여 서버와의 통신 로직을 담당

— // 애플리케이션의 사용되는 정적 파일들 (img, video등…)

— // 컴포넌트들의 모음 (순수 마크업만을 담당한다)

— // 컴포넌트에 데이터를 전달해주는 역할을 한다. (액션을 디스패치 해준다.)

— // custom hook 모음

— // 다국어처리

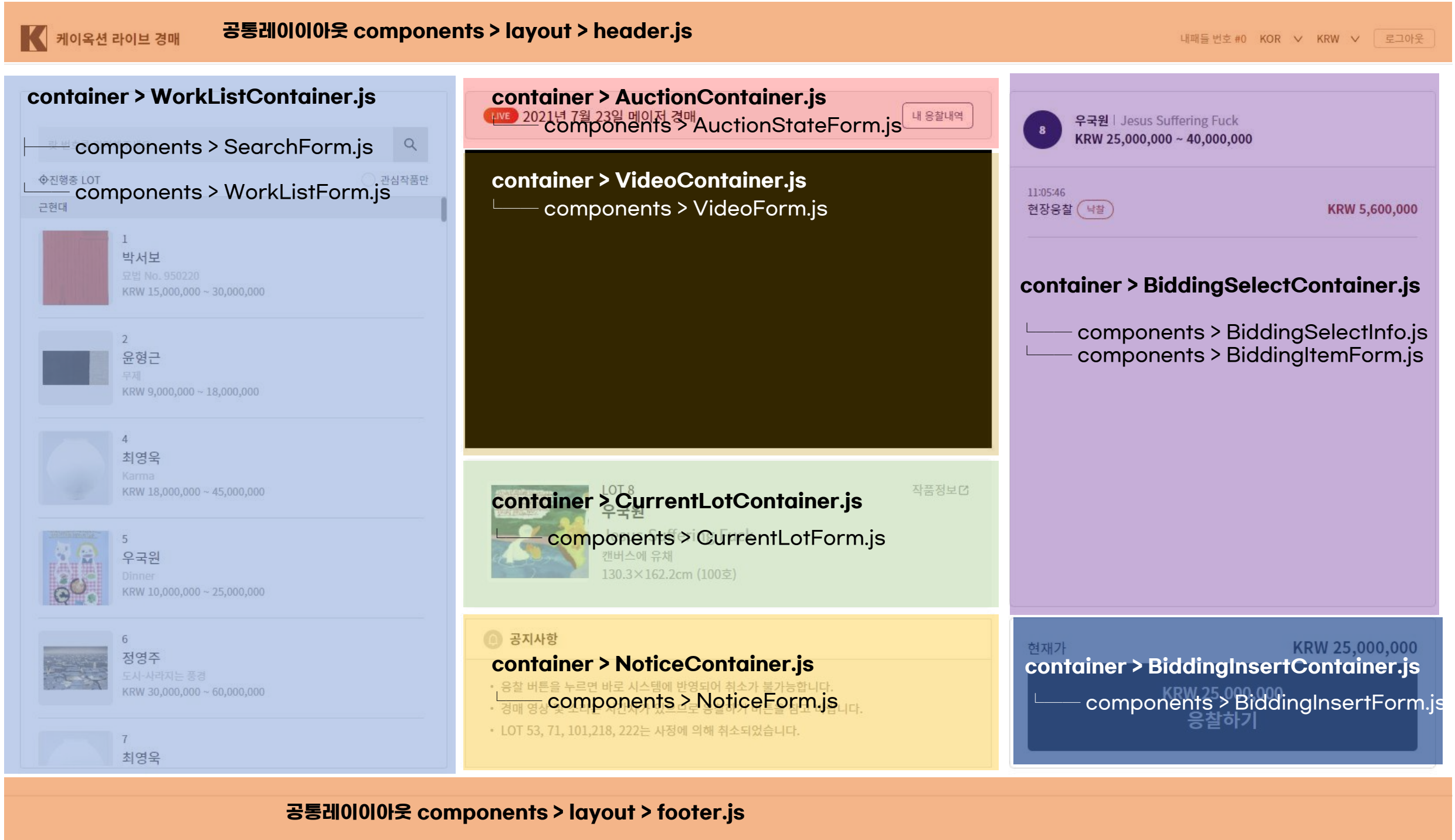
— // 라우팅으로 정리된 컴포넌트 폴더

— // 액션으로 인하여 만들어진 새로운 새로운 객체(state)를 리턴해주는 역할을 한다

— // 데이터 요청을 비동기 작업으로 사용하기 위한 redux- saga

— // state의 관리를 하는 전용 장소 (상태와 리듀서가 들어간다)

— // global css 정의 (전역으로 쓸 css)


케이옥션 라이브 경매
공통레이이아웃 components > layout > header.js
내패들 번호 #0 KOR KRW 로그아웃
container > WorkListContainer.js
components > SearchForm.js
components > WorkListForm.js
container > AuctionContainer.js
components > AuctionStateForm.js
container > VideoContainer.js
components > VideoForm.js
container > CurrentLotContainer.js
components > CurrentLotForm.js
container > NoticeContainer.js
components > NoticeForm.js
container > BiddingSelectContainer.js
components > BiddingSelectInfo.js
components > BiddingItemForm.js
container > BiddingInsertContainer.js
components > BiddingInsertForm.js
공통레이이아웃 components > layout > footer.js

Pallete.js 사용법

Liveclient > styles > Pallete.js

Pallette.js

liveclient > styles > JS palette.js > palette > blue

```
const palette = {
    orange: [
        '#FCF2EE',
        '#F9DFD3',
        '#FBB89B',
        '#F89569',
        '#F67137',
        '#F44E05',
        '#D63300',
        '#B42000',
        '#901100',
        '#700D00'
    ],
```

Pallette.js 사용예시

ex) Palette import 해온 후에 사용하고 싶은 위치에 ${palette.gray[2]}

```
import palette from '../styles/palette';

const Wrapper = styled.div`
    border: 1px solid ${palette.gray[2]};
    background-color: #F5F5F5;
    border-radius: 5px;
```

Redux and redux-saga

Liveclient > reducers, sagas

Redux

// 액션의 type을 정의, 액션 객체를 생성하기 위한 상수들이 저장되어 있음
// 액션 타입 => 액션 생성자 이름

```
// 공지사항
export const LOAD_NOTICE_REQUEST = 'auction/LOAD_NOTICE_REQUEST';
export const LOAD_NOTICE_SUCCESS = 'auction/LOAD_NOTICE_SUCCESS';
export const LOAD_NOTICE_FAILURE = 'auction/LOAD_NOTICE_FAILURE';
```

// 액션 생성자
// action에 해당하는 새로운 객체 (state/상태)를 리턴해준다..
// 불변성을 편리하게 관리해줄 수 있는 immer를 사용하였다.

```
const reducer = (state = initialState, action) => produce(state, (draft) => {
    switch(action.type) {
        case GET_SCHEDULE_REQUEST:
            draft.loading = true;
            draft.error = null;
            break
```

Redux-saga

// 비동기 작업을 하는 미들웨어가 필요할때 사용
// 특정 액션이 발생했을때 상태 값이나 응답 상태 등에 따라 다른 액션을 디스패치하거나 추가적인 로직을 적용해야 될때 사용할 수 있다.
// api 요청, 요청 실패시 재요청 등의 리덕스와 관계없는 코드를 미들웨어로 실행시킬 수 있다.
// yield는 중단점을 만든다.
// yield 뒤에 값이 있으면 그걸 객체의 value에 담아 return하고 멈춘다

// redux-saga/effects.

```
import { all, fork, put, takeLatest, call } from 'redux-saga/effects';
```

// all - 여러 사가를 합쳐주는 역할을 한다. All은 파라미터로 배열을 받는데, 배열에 있는것을 모두 실행시킨다.
// fork - 함수를 실행시키는 역할을 한다., 비동기이기 때문에 결과값을 기다려 주지 않는다.

```
export default function* auctionSaga() {
    yield all([
        fork(watchLoadSchedule),
        fork(watchLoadNotice),
    ]);
}
```

// redux-saga/effects.

```
import { all, fork, put, takeLatest, call } from 'redux-saga/effects';
```

// call - 함수를 동기적으로 실행시켜준다. 첫번째 파라미터는 함수, 두번째 파라미터는 해당함수에 넣을 인수이다.

```
const result:AxiosResponse = yield call(api.loadNotice, action.data);
```

//takeLatest - 기존에 진행중이던 작업이 있다면 취소처리하고 마지막으로 실행된 작업만 수행한다.

```
function* watchLoadSchedule() {
    yield takeLatest(GET_SCHEDULE_REQUEST, loadScheduleSaga)
}
```

// put - 특정액션을 dispatch하도록 한다.

```
yield put(wishInfoSelectSuccess(result.data.data.Table));
```

// redux-saga 함수 예시.

```
// 공지사항 조회
function* loadNoticeSaga(action: any) {
    try {
        const result:AxiosResponse = yield call(api.
        loadNotice, action.data);
        // eslint-disable-next-line eqeqeq
        if (result.data.resultCd === '00') {
            yield put(loadNoticeSuccess(result.data));
        } else {
            const err = new Error('please improve your
            code');
            throw err;
        }
    } catch (err) {
        console.error(err)
        yield put(loadNoticeFailure((err as any).message));
    }
}
```

React-intl 언어설정

Reat-intl 언어설정

// lang폴더에 ko.json, en.json 으로 정리

* 불러오는 방법

```
import { FormattedMessage } from "react-intl";
    <ButtonWrap onClick={showModal}>
        <FormattedMessage id='mybid'/>
    </ButtonWrap>
```

* 데이터가 있을경우 value값 넣기

```
<p><FormattedMessage
    id="SUCC_REG_DATE"
    values={{ reg_date: biddingInsert.
    reg_date}}
/></p>
```

이름	actions > action creator	api
공지사항 조회	loadNoticeRequest	loadNotice
경매스케줄 조회	getScheduleRequest	loadSchdule
응찰등록	biddingInsertRequest	biddingInsert
전체응찰내역조회	getBidsRequest	getBids
관심작품추가하기	likeWorkRequest	likeWork
관심작품목록가져오기	wishInfoSelectRequest	
언어변경	currencyRequest	
응찰내역업데이트 (취소나 삭제 됐을 경우)	bidsListUpdateRequest	
클레임 응찰내역	bidsListClaimRequest	
내응찰내역조회	getMyBidRequest	getMyBid