# Ash open interface elements

## Global pattern

### Header

#### Header


Contains: title(fund-name), amount(share price), amount(aum), main section

CSS:

```
.fund-name {
    font-size: 24px;
    font-weight: 900;
}

.share-price, .aum {
    font-weight: 100;
}
```

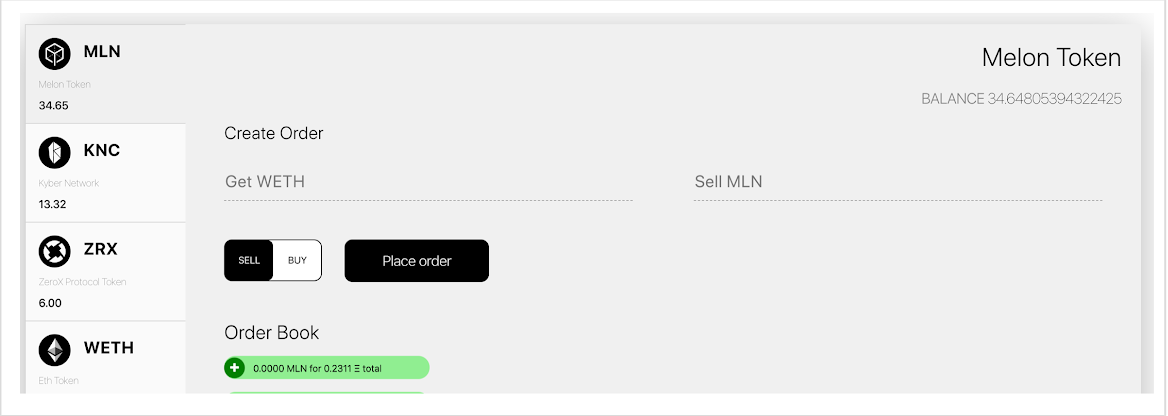
Description:

Display fundamental information like fund name, share price and assets under management.

Should be fixed in position and always visible independent of section changes.

# main(trading desk)

## main(trading desk)




Contains: list(assets), pattern(token-trade)

CSS:

```
.fund-view {
    background: rgb(250, 250, 250);
    height: auto;
    max-height: calc(100vh - 80px);
    overflow-y: scroll;
    border-bottom: 1px solid lightgrey;
    color: grey;
    cursor: pointer;
}
.asset-view {
    background: #f0f0f0;
    width: 100%;
    overflow-y: scroll;
    height: calc(100vh - 80px);
}
```

Description:

Use to enable fund interaction. Main interaction the open interface implements is the trading desk pattern.

Possible future implementations of the main section will center around donations (giveth module) and financial product integration.

pattern(token-trade)

# pattern(token-trade)


Contains: title(token), amount(token-balance), input(make-order), button(toggle), button(place-order), list(order-book)

CSS:

```
.asset-view {
    background: #f0f0f0;
    width: 100%;
    overflow-y: scroll;
    height: calc(100vh - 80px);
}

.asset-view>h1, .asset-view>h3 {
    text-align: right;
    margin-right: 24px;
    font-weight: 300;
}
```

Description:

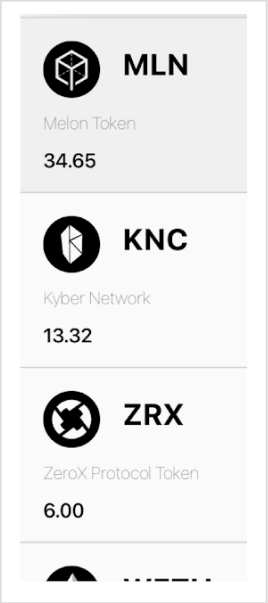
Provide an easy overview of all information pertaining to the selected asset.

Enable placing
buy/sell orders as well as picking an order from the order book.

# Building blocks

## Lists

### list(assets)




Contains: list(assets-element)

CSS:

```
.fund-view {
    background: rgb(250, 250, 250);
    height: auto;
    max-height: calc(100vh - 80px);
    overflow-y: scroll;
    border-bottom: 1px solid lightgrey;
    color: grey;
    cursor: pointer;
}
```

Description:

Display all available assets. Start top to bottom with assets in fund vault in descending order, to zero amount assets.

Highlight selected asset.

Order book

## list(order-book)


Contains: list(order-book-element)

CSS:

```
.orders {
    margin-left: 48px;
    font-size: 12px;
}
.order {
    border-radius: 50px;
    width: 256px;
    cursor: pointer;
    margin-top: 16px;
    color: white;
    display: inline-block;
}
.order.add {
    background-color: lightgreen;
    color: black
}
.order.remove {
    background-color: #ff3939;
}
```

Description:

List buy and sell orders, use traditional green/red differentiation.

title(balance)

title(balance)


CSS:

```
.asset-view>h3 {
    text-align: right;
    margin-right: 24px;
    font-weight: 300;
}
```

Description:

Display descriptor in grey and amount in black to provide easy to consume guided information architecture.

# Input

## input(make-order)


CSS:

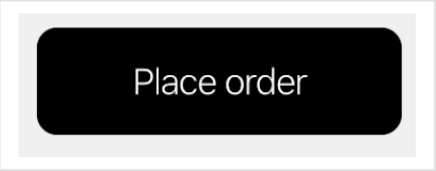
```
.input-value {
    height: 45px;
    width: calc(50% - 40px);
    font-size: 21px;
    background: none;
    border: none;
    border-bottom: 1px dashed darkgray;
}
.input-container {
    justify-content: space-between;
    align-items: center;
    display: flex;
    padding: 48px;
    padding-top: 24px;
    padding-bottom: 0;
}
```

Description:

Use explicit empty-container messages to illustrate intended use.

# Buttons

## button(place-order), button(toggle)





CSS:

```
button {
    background: white;
    border-radius: 0;
    box-shadow: none;
    outline: none;
    border: 1px solid black;
    padding: 16px;
    width: calc(20% - 40px);
    font-size: 18px;
    font-weight: bolder;
    cursor: pointer;
    min-width: 100px;
}
.toggle {
    height: 50px;
    width: 120px;
    border-radius: 10px;
    border: 1px solid #000;
    background-color: #fff;
    cursor: pointer;
}
```

Description:

Use black button elements for execution of action.

To keep in line with button color doctrine use black for active state, white to indicate inactive state.

## Loading elements

### loader(order-book), loader(order)

CSS:

```
.loading {
    height: calc(100vh - 80px);
    overflow-y: hidden;
    border-bottom: 1px solid lightgrey;
    color: grey;
    width: 100%;
    display: flex;
    justify-content: center;
    align-items: center;
}
```

Description:

Use loader elements as placeholder for content getting loaded from the blockchain.

Use loader elements in modal state to bridge waiting time while transacations are settling on the blockchain and guide the user experience.