Kolibri

AUSWAHLKOMPONENTE

IP6 Abschlusspräsentation

Ramona Marti, Lea Burki

fhnw.ch

Hallo

Wir sinds...

Lea Burki

Ramona Marti

Rückblick

Problem

- Begrenzte Möglichkeiten mit `select & datalist`
- Schwer umzugestalten
- Schlechtes Intraktionsdesign
- Bibliotheken mit vielen Abhängigkeiten

Ziel

- Generalisierte Auswahlkomponente
- Ansprechendes Design
- Wiederverwendbare Komponenten
- Effiziente und benutzerfreundliche Bedienung

Simple Input

Rückblick

```
/**
 * @typedef SimpleOptionType
 * @property { String } value  - selectable value of the input
 * @property { String? } label - visible label of the input
 */
```

```
const SimpleOptionsModel = () ⇒ {
   const list    = [];
   const listObs = ObservableList(list);
   return {
      getList   : () ⇒ [...list],
      getObsList: () ⇒ listObs,
   };
};
```

Simple Input

```
/**
 * @typedef SimpleOptionsControllerType
 * @property { () ⇒ Array<SimpleOptionType> }      getOptions
 * @property { (option: SimpleOptionType) ⇒ void } addOption
 * @property { (option: SimpleOptionType) ⇒ void } delOption
 * @property { (cb: ConsumerType<SimpleOptionType>) ⇒ void } onAddOption
 * @property { (cb: ConsumerType<SimpleOptionType>) ⇒ void } onDelOption
 */



/**
 * @typedef { SimpleInputControllerType<String> & SimpleOptionsControllerType } SimpleInputWithOptionsControllerType
 */
```

Aufbau

Ausbau OptionsType

```
/**
 * @typedef SimpleOptionType
 * @property { String } value  - selectable value of the input
 * @property { String? } label - visible label of the input
 */
```

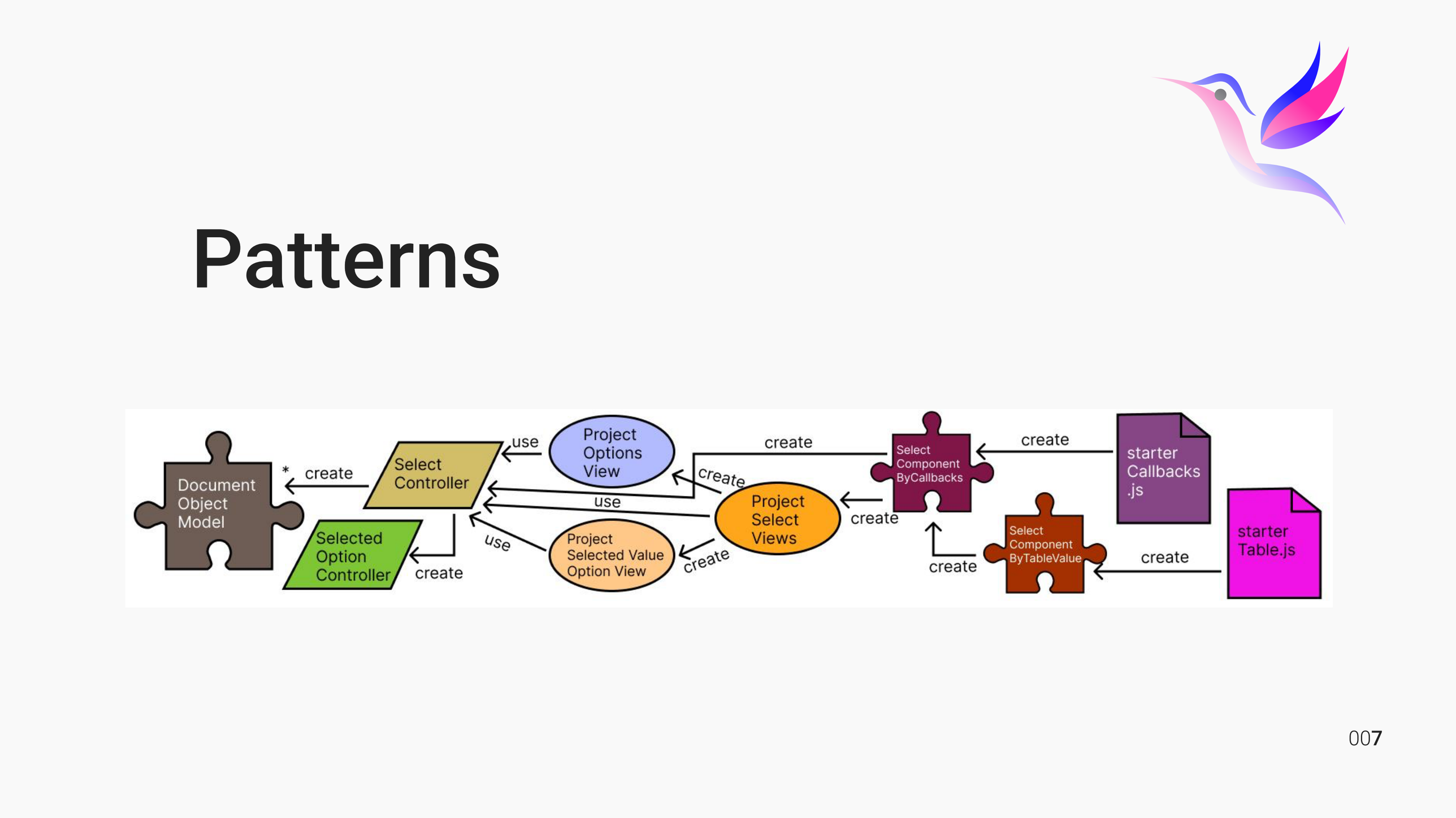
```
/**
 * @typedef OptionType
 * @property { () ⇒ String } getValue          - selectable value of the input
 * @property { () ⇒ String } getLabel          - visible label of the input
 * @property { () ⇒ String } getId             - unique identifier of the option
 * @property { (OptionType) ⇒ Boolean } equals - true if label and value are the same
 */
```

Aufbau

Ausbau OptionsModel

```
const SimpleOptionsModel = () => {
   const list    = [];
   const listObs = ObservableList(list);
   return {
      getList   : () => [...list],
      getObsList: () => listObs,
   };
};
```

```
const OptionsModel = () => {
   const list         = [];
   const listObs      = ObservableList(list);
   const optionsCache = {};
   const addOption    = (option) => {
      /* check null option */
      /* check cache & add option if needed */
      /* add cached option if not contained */
   };
   return {
      getList   : () => [...list],
      getObsList: () => ({ ...listObs, add: addOption })
   };
};
```



Patterns

Document Object Model
*
create
Select Controller
use
Project Options View
create
Select Component ByCallbacks
create
starter Callbacks .js
use
Project Select Views
create
Selected Option Controller
create
use
Project Selected Value Option View
create
create
Select Component ByTableValue
create
starter Table.js

Decorator Patterns

```
const SelectComponentByCallbacks = (
   attributes,
   callbackArray
) ⇒ {
   const ctrl           = SelectController(
      attributes,
      callbackArray.length
   );
   const [viewElement]  = projectSelectViews(ctrl);
   const [labelElement] = viewElement.children;
   /* component logic */
   return {
      getSelectController: () ⇒ ctrl,
      getComponentView   : () ⇒ viewElement,
      getLabelElement    : () ⇒ labelElement,
   };
};
```

```
const SelectComponentByTableValues = (
   attributes,
   optionsTable,
   sortColumnOptionsAlphabetical = false
) ⇒ {
   /* mapping between table and callbacks */
   const component = SelectComponentByCallbacks(
      attributes,
      callbacks);
   return {
      ...component,
   };
};
```

Decorator Patterns

```
const interactionProjector = (
    rootElement,
    selectController,
    currentColumn,
    pageSize = 10,
    autoClose = true
) => {
    // definition of interactions
};
```

```
const interactionProjectorWithoutSelectionChange = (
    rootElement,
    ctrl,
    pageSize = 10
) => {
    const currentCol = Observable(0);
    interactionProjector(rootElement, ctrl, currentCol, pageSize);
};
```

```
const interactionProjectorWithSelectionChange = (
    rootElement,
    ctrl,
    pageSize = 10
) => {
    const currentCol = Observable(0);
    // change selection with cursor position
    interactionProjector(rootElement, ctrl, currentCol, pageSize, false);
};
```

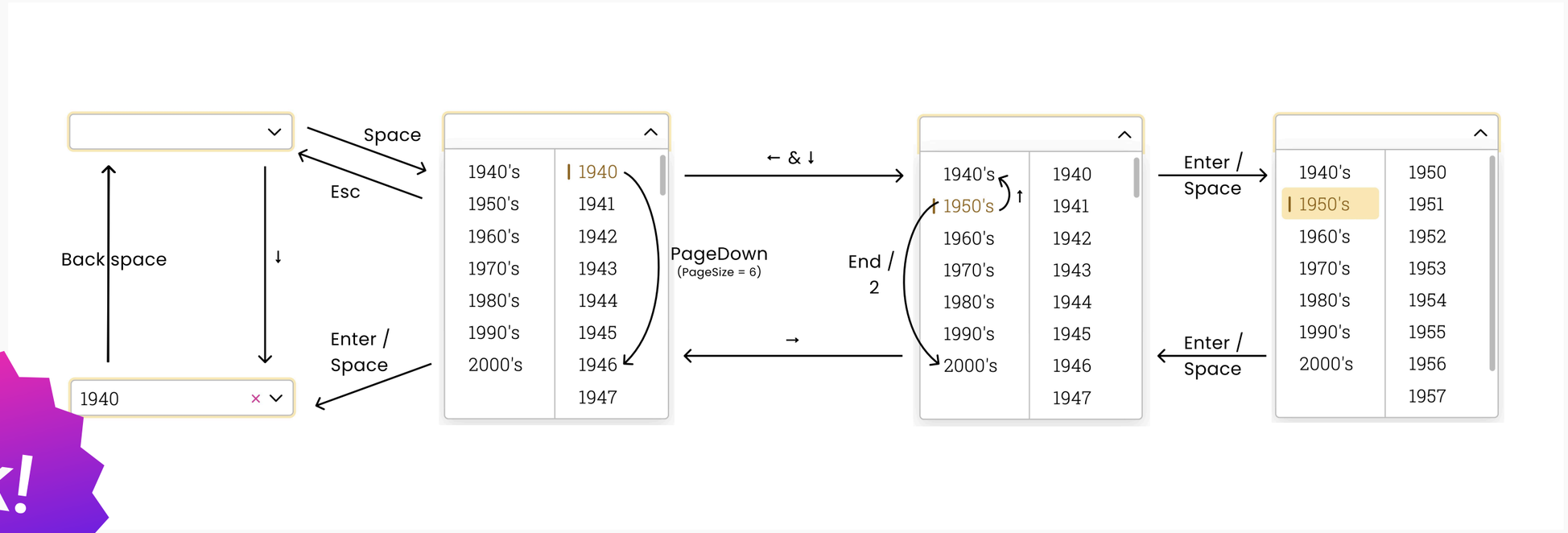
Decorator Patterns

```
const Option = (value, label) => {
   /* option logic & equals definition */
   let id = "";
   return {
      getValue: () => value,
      getLabel: () => label,
      getId   : () => id,
      equals  : equals,
   }
};
```

```
const ValueOption = (value, label = "") => {
   const optionLabel = ( !label || label === "" )
                        ? value
                        : label;
   return Option(value, optionLabel);
};
```

```
const CategoryOption = (label) => {
   return Option("", label);
};
```

Interaction



Link!
Inter-aktionen

Testing

Automatisiert

7	tests in	projector/simpleForm/optionsModel	ok
28	tests in	projector/selectComponent/optionsModel	ok
9	tests in	projector/selectComponent/optionsController	ok
6	tests in	projector/selectComponent/columnOptionsProjector	ok
11	tests in	projector/selectComponent/columnOptionsComponent	ok
26	tests in	projector/selectComponent/selectController	ok
11	tests in	projector/selectComponent/selectProjector	ok
38	tests in	projector/selectComponent/selectComponent	ok
86	tests in	projector/selectComponent/interactionProjector	ok
1	tests in	projector/selectComponent/selectProjector label (async)	ok

Testing

Manuell / UI & Interaktion

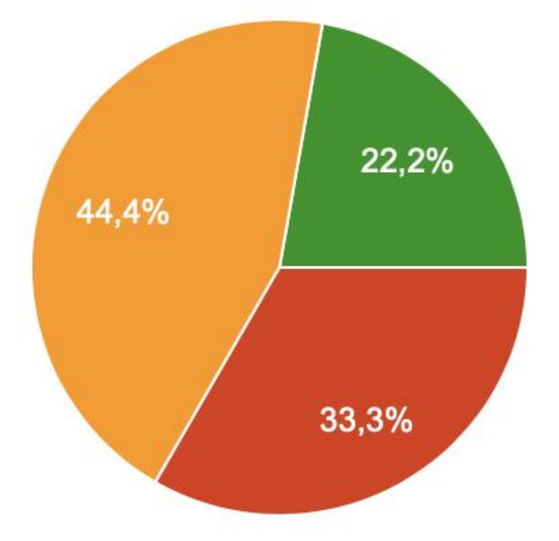
OSX

Windows

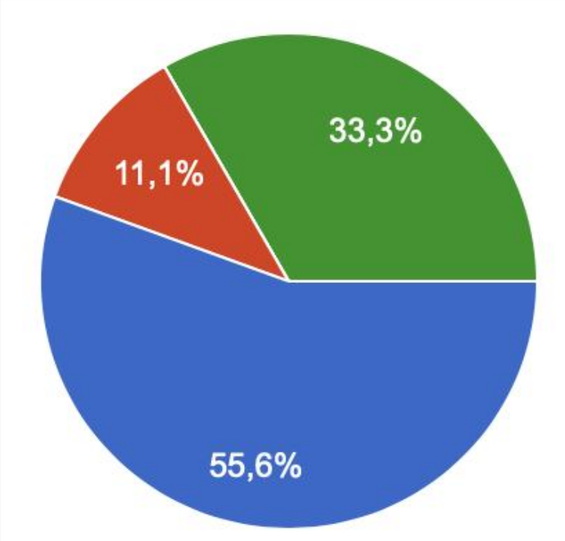
User Tests

Programmierer

Aufgabe - Select 1 Spalte



Aufgabe - Select 2 Spalte



User Tests

Endanwender

100%

Meinten, Sie konnten die gewünschte Auswahl schneller treffen, als wenn sie durch eine lange Liste hätten scrollen müssen?

User Tests

Endanwender

100%

Meinten, Sie konnten die gewünschte Auswahl schneller treffen, als wenn sie durch eine lange Liste hätten scrollen müssen?

50%

Fanden die Filteroptionen sehr hilfreich.

50%

Fanden die Filteroptionen ungewohnt, nach Angewöhnungszeit jedoch hilfreich

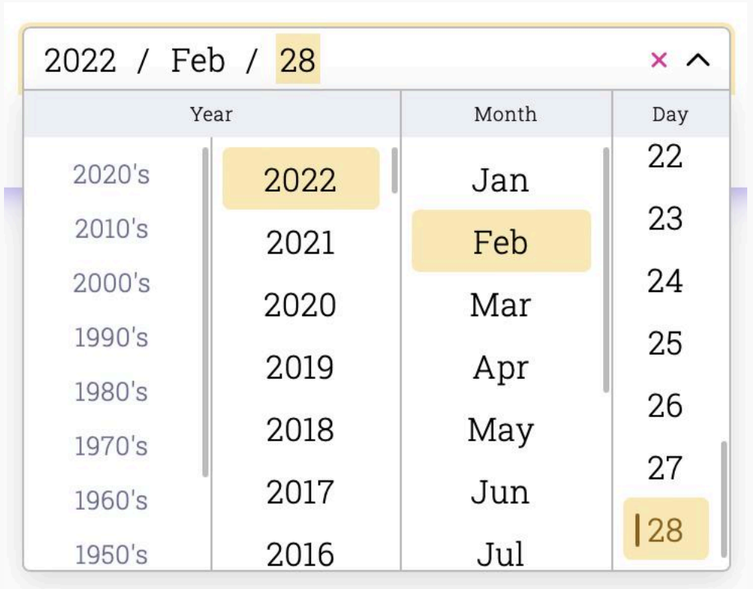
0%

Fanden die Filteroptionen überflüssig

Future

Inkl. Bonus

- Einbindung / Übernahme in Kolibri
- Weitere User-Tests
- Performance weiter verbessern
- Erstellen neuer Projektoren



LIVE DEMO
Bonus

Fazit

- Benutzerfreundlichkeit
- Flexibilität
- Stabile Basis für zukünftige Erweiterungen
- Usertest Timing

Thank you

Diskussion & Fragen