

# kolibri ✓

## AUSWAHLKOMPONENTE

IP6 Abschlusspräsentation  
Ramona Marti, Lea Burki

[fhnw.ch](https://fhnw.ch)

**Hallo**  
Wir sinds...



Lea Burki



Ramona Marti

# Rückblick ZP

## Problem

- Begrenzte Möglichkeiten mit `select` & `datalist`
- Schwer umzugestalten
- Schlechtes Interaktionsdesign
- 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 OptionType

```
/**  
 * @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

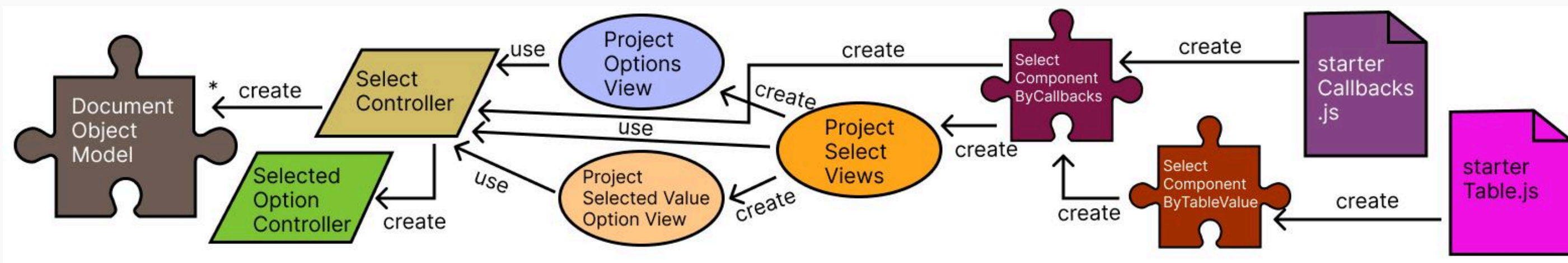
## Ausbau OptionsModel

```
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







# 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 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);  
};
```



# Decorator Patterns

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

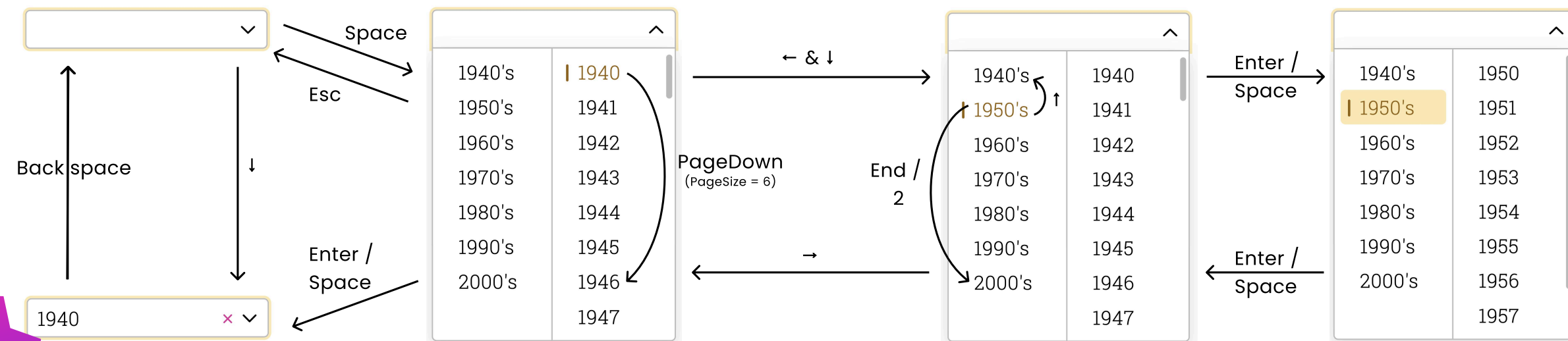
```
const interactionProjectorWithoutSelectionChange = (  
  rootElement,  
  ctrl,  
  pageSize = 10  
) => {  
  const currentCol = Observable(0);  
  interactionProjector(rootElement, ctrl, currentCol, pageSize);  
};
```

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





# Interaction





# Testing

## Automatisiert

7	tests in	<a href="#">projector/simpleForm/optionsModel</a>	ok
28	tests in	<a href="#">projector/selectComponent/optionsModel</a>	ok
9	tests in	<a href="#">projector/selectComponent/optionsController</a>	ok
6	tests in	<a href="#">projector/selectComponent/columnOptionsProjector</a>	ok
11	tests in	<a href="#">projector/selectComponent/columnOptionsComponent</a>	ok
26	tests in	<a href="#">projector/selectComponent/selectController</a>	ok
11	tests in	<a href="#">projector/selectComponent/selectProjector</a>	ok
38	tests in	<a href="#">projector/selectComponent/selectComponent</a>	ok
86	tests in	<a href="#">projector/selectComponent/interactionProjector</a>	ok
1	tests in	<a href="#">projector/selectComponent/selectProjector label (async)</a>	ok



# Testing

## Manuell / UI & Interaktion

OSX

Europe	Canada	Berlin
North America	France	Bern
Asia	Germany	Brugg
	Japan	Hamburg
	Swiss	L.A.
	United States	Marseille
		New York
		Ottawa

Windows

Europe	Canada	Berlin
North America	France	Bern
Asia	Germany	Brugg
	Japan	Hamburg
	Switzerland	L.A.
	United States	Marseille
		New York
		Ottawa

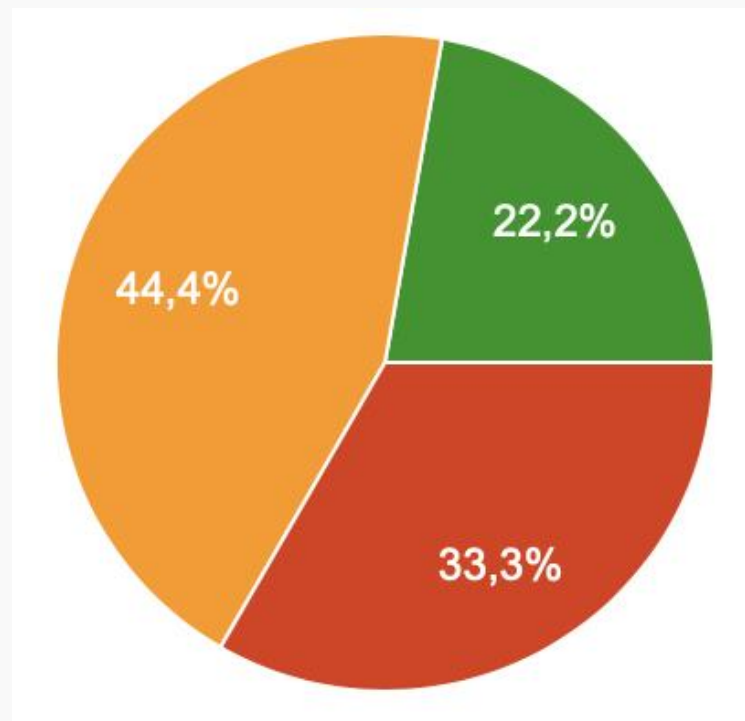
Scrollbar



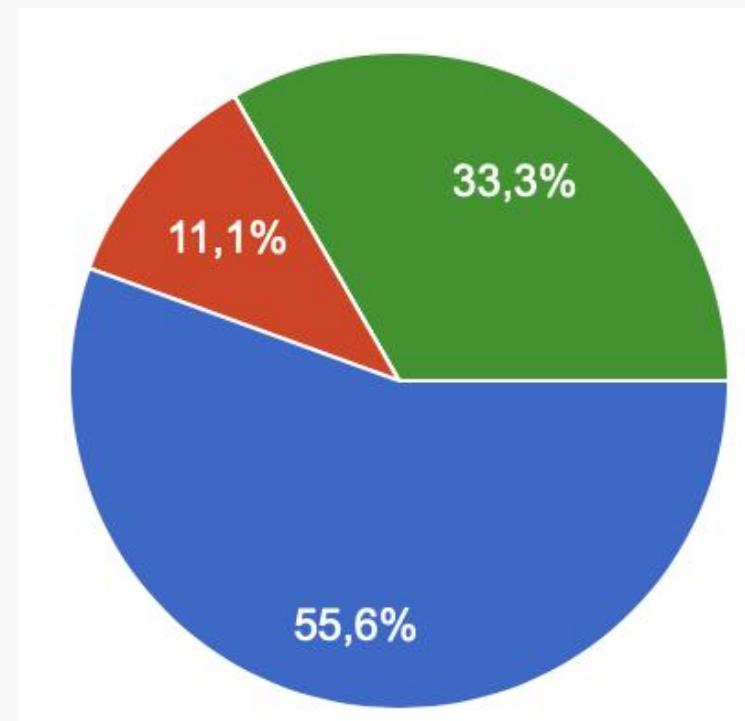
# User Tests

## Programmierer

Aufgabe - Select 1 Spalte



Aufgabe - Select 2 Spalte



- einfach
- eher einfach
- mittelmässig
- eher schwer
- kaum zu lösen



# 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



2022 / Feb / 28			✕ ^
Year	Month	Day	
2020's	2022	Jan	22
2010's	2021	Feb	23
2000's	2020	Mar	24
1990's	2019	Apr	25
1980's	2018	May	26
1970's	2017	Jun	27
1960's	2016	Jul	28

# Fazit

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





*Thank you*  
*Diskussion & Fragen*