Hochschule **Rosenheim**
University of Applied Sciences

Two-way data binding in MVC

Peter Kurfer, Thomas Mildner

Agenda

- What is data binding?
- History and concepts
- Two-way vs. One-way data binding
- Frameworks supporting data binding

Agenda

- Introduction to TypeScript
- Introduction Vue.js
- Data binding in Vue.js
- Exercise
- Edge cases

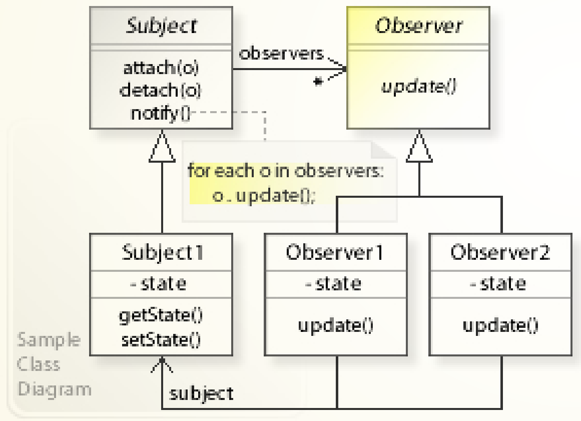
What is data binding?

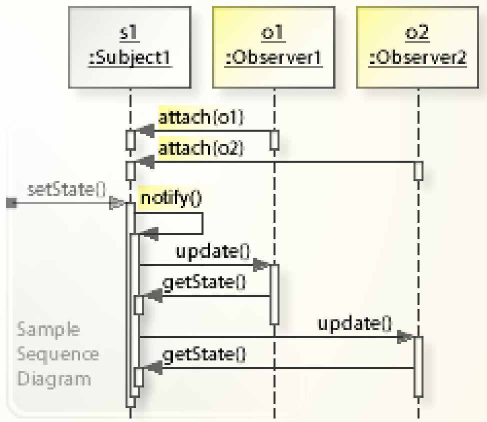
→ bind UI element to an application model

→ Software Design Pattern

→ Observer Pattern works often as underlying binding mechanism

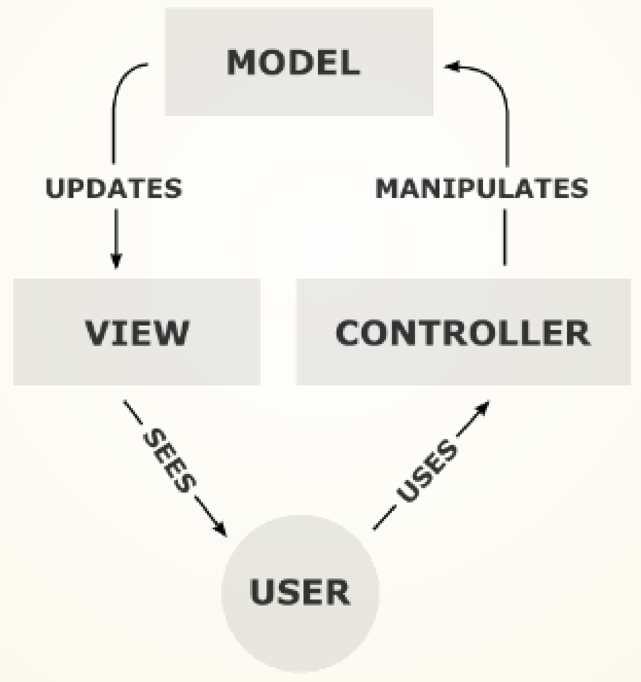
Observer Pattern





Hochschule **Rosenheim**
University of Applied Sciences

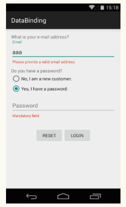
MVC Concept



Challenges for data binding

→ input validation

→ data type mapping



What is your e-mail address?
Email: aaa
Please provide a valid email address
Do you have a password?
No, I am a new customer. [✓ Yes, I have a password:]
Password
Mandatory field

History and concepts

Two-way vs. One-way data binding

→ different binding types are supported

→ choose binding type for suited use case

One-way « vs. » Two-way

One-way data binding

→ scope variable in HTML will be set to first value its model is bound to (first assignment)

→ bind the data from model to view

→ changes in the model are getting transported to the view but not vice versa

Pro and contra One-way data binding

Pro	Contra
only one direction of data flow	No automatic adaptation of data in the model, other components, the UI
easy to debug	invalid states in application
no gui validation / user input validation required	

Two-way data binding

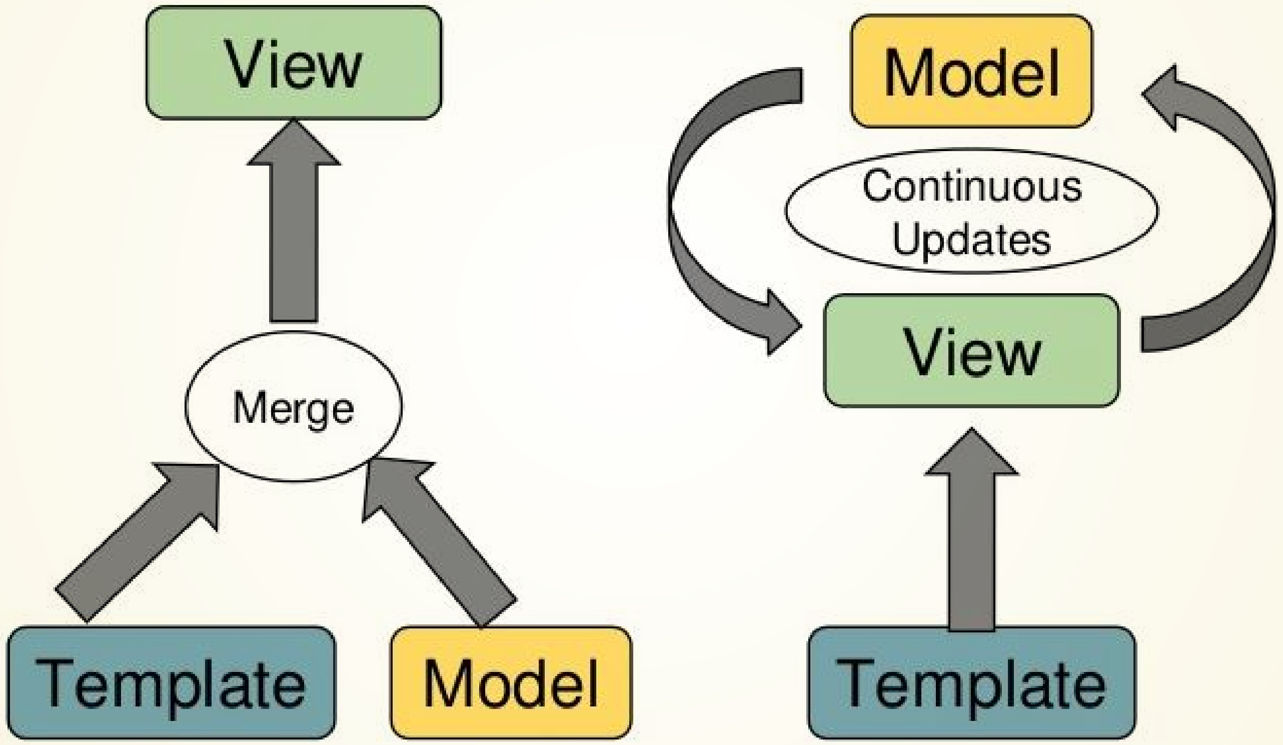
→ scope variable will change its value every time the the model value is changed and vice versa

→ bind the data from model to view and vice versa

Pro and contra Two-way data binding

Pro	Contra
bind multiple GUI elements to a single source of truth in the model	changes in model will cause a change in UI → Performance issue
data consistency guaranteed	input validation / data type matching
changes in data will be automatically added to UI → write less code for display logic	data manipulation / parsing works not very well → Performance issue

One vs. Two Way Data Binding



Frameworks supporting data binding

unity

Microsoft WPF

ANGULARJS by Google

BACKBONE.JS

Microsoft® .NET Windows Forms

Vue.js

Knockout.

ember

JavaFx

METEOR

OpenUI5

React

TypeScript

→ statically typed language

→ compiles to plain JavaScript

→ popular JS framework **Angular 2.0/IO** (not just compatible through typings but completely written in TypeScript)

Problems of JavaScript

→ JS was first developed as a language for client-side

→ Node.js marked JS as an emerging server-side technology

→ JS is difficult to maintain and hardly reusable

→ no object orientation, no strong type checks, no compiling checks

Solution = TypeScript

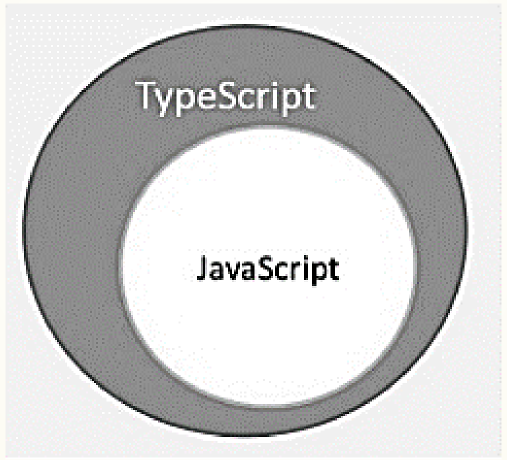
→ designed by Anders Hejlsberg (Designer of C# at Microsoft - 2012)

→ strongly typed, object orientated and compiled language

→ TypeScript is a superset of JavaScript

→ will be compiled to JavaScript

TypeScript is JavaScript plus some additional features



Features of TypeScript

Feature	Usage
TypeScript is just JavaScript	Only knowledge of JS required
Supports other JS Libraries	can be consumed of any JS Code. Can reuse all existing JS frameworks, tools and libraries
JavaScript is TypeScript	Any **.js** file can be renamed to **.ts** and compiled with other TypeScript Files
TypeScript is portable	Portable accross multiple browsers, devices and operating systems. → runs whereever JS runs

First example of TypeScript …

```
class Greeting {
    greet(): void {
        console.log("Hello World!!!")
    }
}

let g = new Greeting();
g.greet();
```

Variables and Compile Checks

```
let firstName: string = "John";
let score1: number = 50;
let score2: number = 42.50
let sum = score1 + score2
```

Compile Error:

```
let num: number = "hello"
```

⚡ Compiler error because `"hello"` is no `number`

Functions

```
function calc_discount(price: number, rate: number = 0.50): Uni
    let discount = price * rate;
    console.log("Discount Amount: ", discount);
}

calc_discount(1000)

// call method without default parameter
calc_discount(1000, 0.30)
```

Anonymous functions

```
let res = function (a: number, b: number) {
    return a * b;
};
console.log(res(12, 2))
```

Lambda Expressions

```
let foo = (x: number) => 10 + x;
console.log(foo(100))      //outputs 110
```

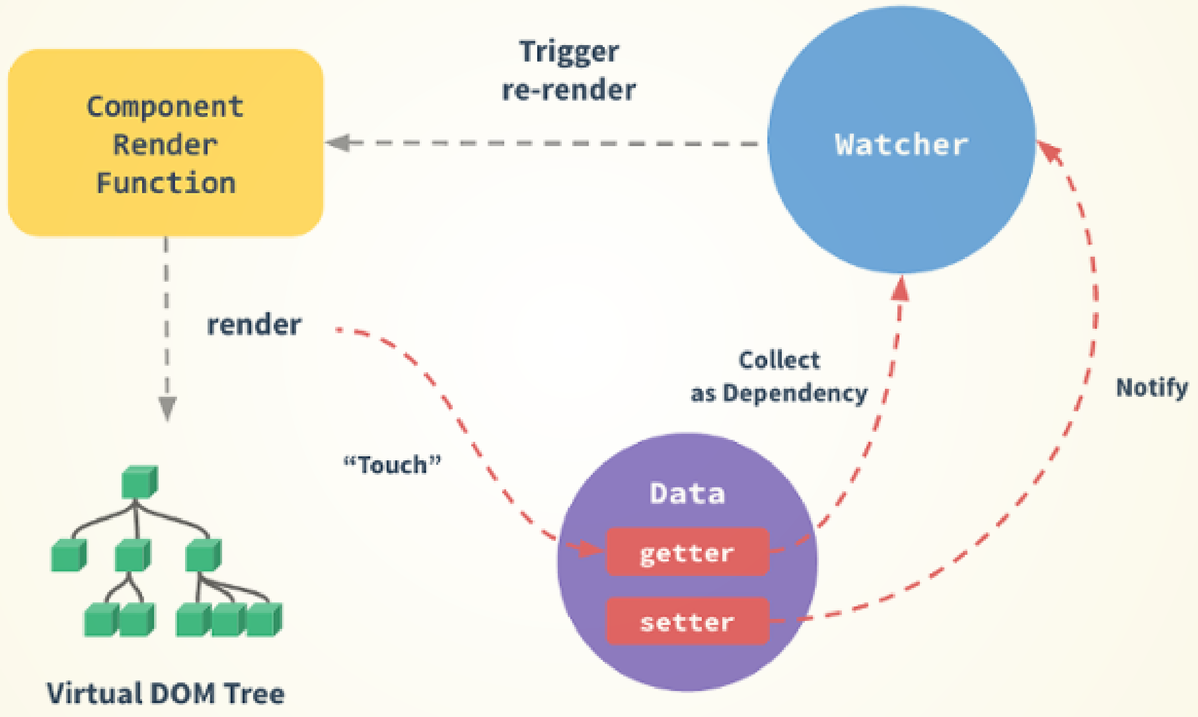
Vue.js

"Data binding" without Vue.js

Modify an HTML element from Vanilla JS:

```
<p id="test-id">Nothing to say</p>
<script>
    let pElem = document.getElementById("test-id");
    pElem.innerHTML = "Hello from JS";
</script>
```

Data binding in Vue.js



Quelle

Data binding in Vue.js - Template syntax

```
<div id="root">
  <p>The value is {{message}}</p>
</div>

<script>
new Vue({
    el: '#root',
    data: {
        message: 'test binding'
    }
})
</script>
```

Data binding in Vue.js - Attribute syntax

```
<div id="root">
  <input type="text" v-model='message'>
  <p>{{message}}</p>
</div>

<script>
new Vue({
  el: '#root',
  data: {
    message: 'initial value'
  }
});
</script>
```

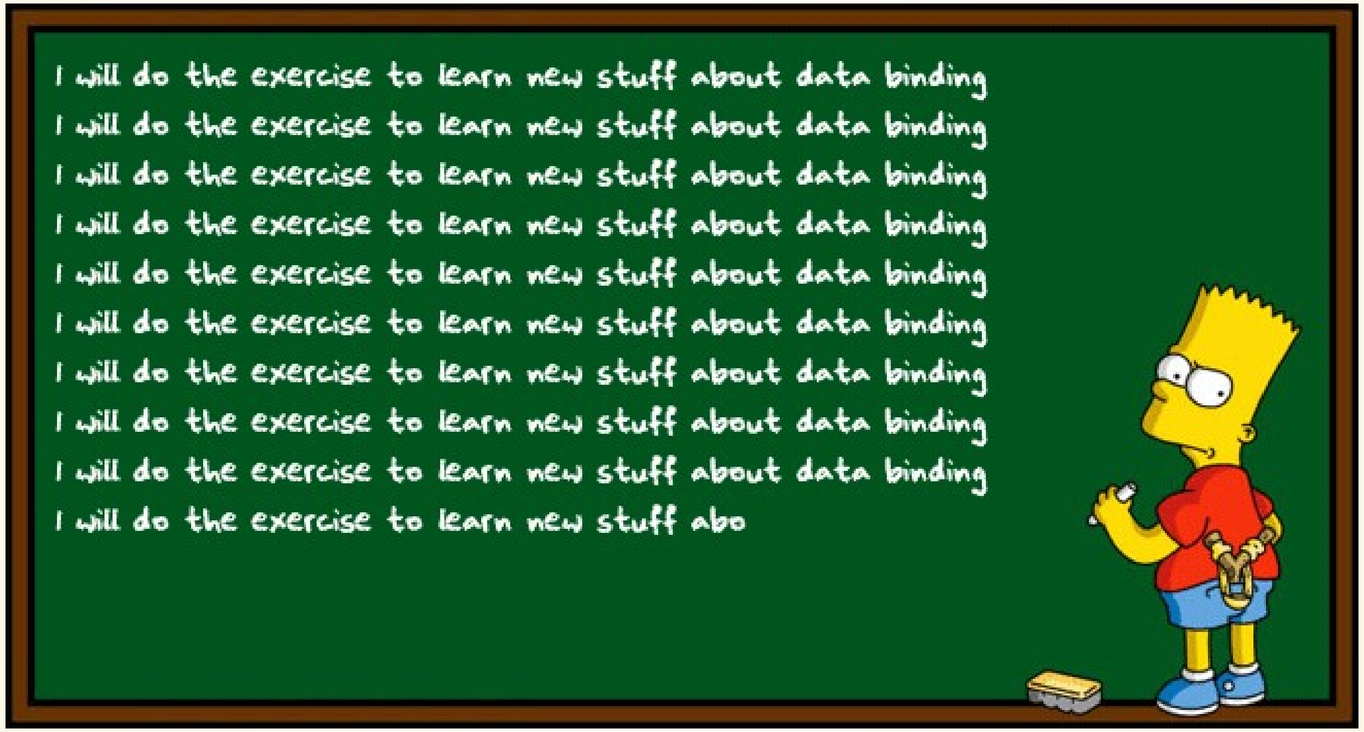
Data binding in Vue.js - Conditions

```
<div id="root">
  <p v-if="showParagraph">{{message}}</p>
  <p v-else>Paragraph is hidden</p>
</div>

<script>
  new Vue({
    el: '#root',
    data: {
      message: 'Hello :)',
      showParagraph: true
    }
  })
</script>
```

Data binding in Vue.js - Loops

```
<div id="root">
  <ul>
    <li v-for="item in items" :key="item.id">{{item.value}}</li
  </ul>
</div>

<script>
  new Vue({
    el: '#root',
    data: {
      items: [ { id: 1, value: 'Hello' }, { id: 2, value: 'Worl
    }
  })
</script>
```

Exercise


I will do the exercise to learn new stuff about data binding
I will do the exercise to learn new stuff about data binding
I will do the exercise to learn new stuff about data binding
I will do the exercise to learn new stuff about data binding
I will do the exercise to learn new stuff about data binding
I will do the exercise to learn new stuff about data binding
I will do the exercise to learn new stuff about data binding
I will do the exercise to learn new stuff about data binding
I will do the exercise to learn new stuff about data binding
I will do the exercise to learn new stuff abo

Edge cases of data binding in Vue.js

Dynamic properties

Vue.js does recognize new properties of objects (or when a property is deleted), when the instance is already initialized.

```
let vm = new Vue({
  data: {
    a: 1
  }
});

vm.b = 2
```

Dynamic properties

If required there's a "hack" to resolve the problem:

```
Vue.set(vm.someObject, 'b', 2)
```

Side note: because TypeScript enforces strict typing this edge case does not really matter for TypeScript (except you're using `any` but why should you 😉).

Reactive properties

```
let vm = new Vue({
  data: {
    message: ''
  }
});

//...

vm.message = 'Hello, World!';
```