The Progressive JavaScript Framework

WHY VUE.JS? GET STARTED GITHUB

vuejs.org/

Special Sponsor

Standard Library

Build APIs you need in minutes instead of days, for free.

Approachable

Already know HTML, CSS and JavaScript? Read the guide and start building things in no time!

Versatile

An incrementally adoptable ecosystem that scales between a library and a full-featured framework.

Performant

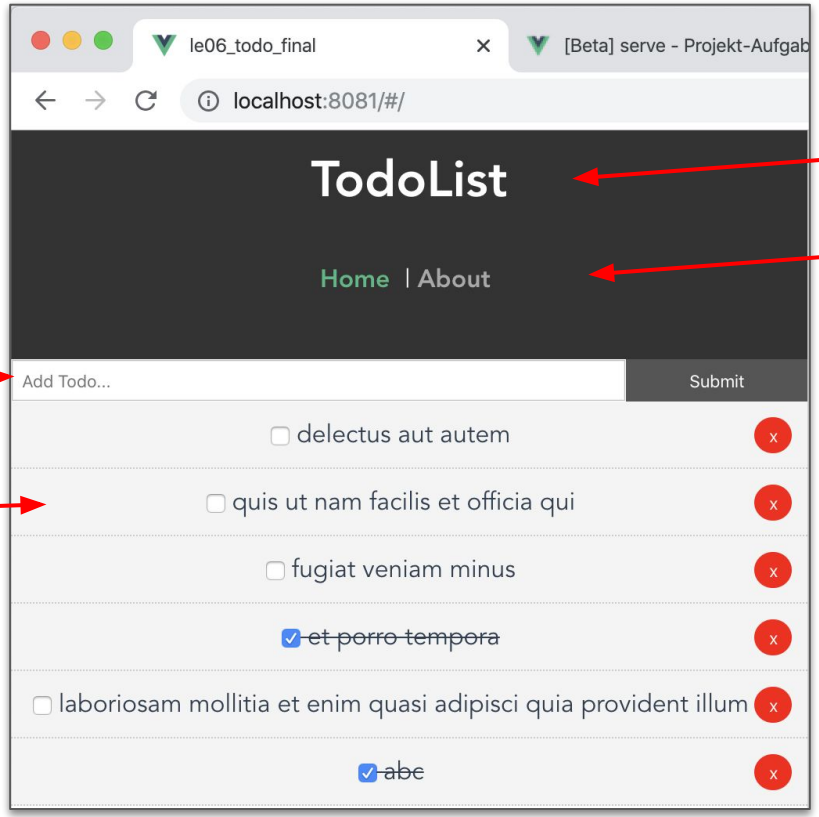
20KB min+gzip Runtime
Blazing Fast Virtual DOM
Minimal Optimization Efforts

Vorteile von Vue.js

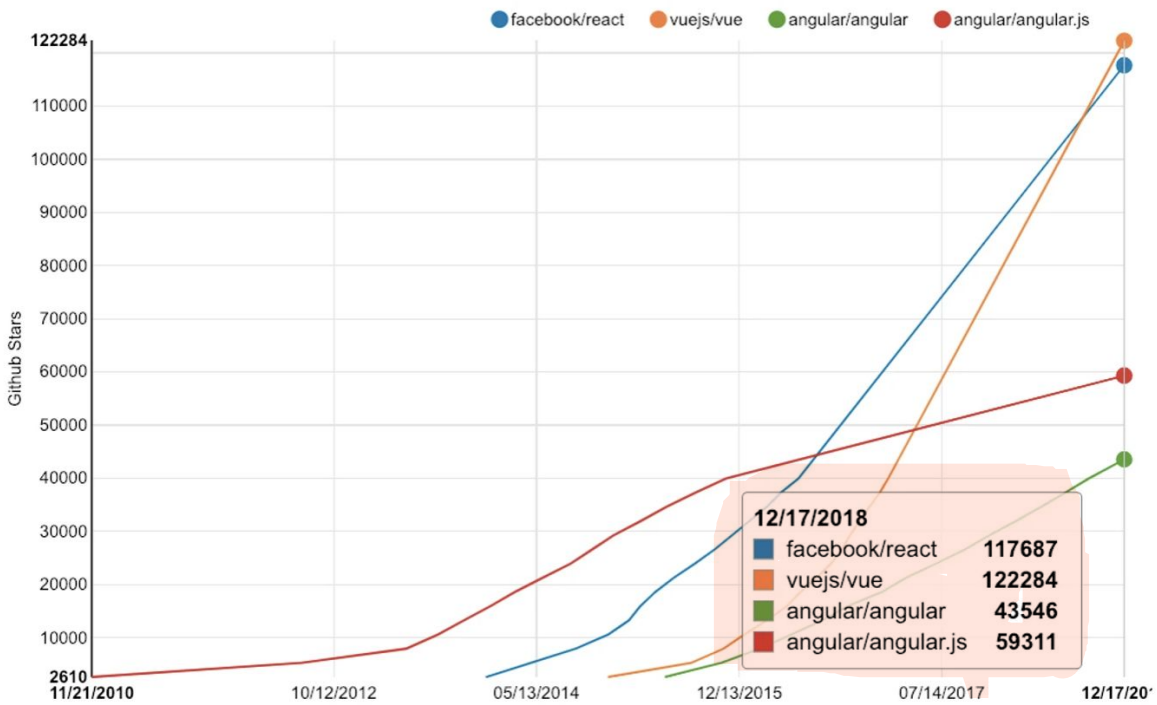
- Komplexe Single Page Application (SPA) sind mit Vue einfacher zu erstellen
- geringerer Lernaufwand als andere Frameworks
- performant und leichtgewichtig
- Code-Erzeugung für alte Browser (IE9) möglich
- Virtual DOM
- Stärkstes Wachstum des Interesses der Industrie
  - verglichen mit Angular und React
- Viele Beispiele
- Viele fertige Komponenten
- sehr gute Dokumentation
- große Community

vuejs.org/

Komponenten-Ansatz für SPA



Vue.js im Vergleich: Stärkstes Wachstum



Number of stars on GitHub projects for Angular, React, and Vue

The sizes of the libraries are as follows:

- Angular: 500+ KB
- React: 100 KB
- Vue: 80 KB

Chinese giants like Alibaba and Baidu picking Vue as their primary front-end JavaScript framework.

Vue should be your choice if you prefer simplicity, but also like flexibility.

https://www.codeinwp.com/blog/angular-vs-vue-vs-react/

Hello World in Vue.js

```
<div id="app">
  <h1>1. Thema {{title}}</h1>
  <h2>2. Unterthema {{title}}</h2>
  <p>Paragraph über {{title}}</p>
</div>

<script src="https://unpkg.com/vue/dist/vue.js"></script>

<script>
  new Vue({
    el: '#app',
    data: {
      title: "Hello World"
    }
  });
</script>
```

Wo soll gerendert werden?

Daten, die gerendert werden sollen

1. Template

2. Data

Data-Binding

Wie könnte man Vue.js selber implementieren?
mit Pattern Matching

```
<div id="app">
  <h1>1. Thema {{title}}</h1>
  <h2>2. Unterthema {{title}}</h2>
  <p>About {{title}}</p>
</div>
<script>
 class Vue {
  constructor( options ){
   Object.assign( this, options );
   const root = document.querySelector( this.el );
   const matches = root.innerHTML.matchAll( /({{(\w+)}})/gm );
   for (const match of matches) {
     const brackets = match[1]; // {{title}}
     const name = match[2];     // title
     root.innerHTML = root.innerHTML.replace( brackets, this.data[ name ] );
   }
  }
 }
</script>
```

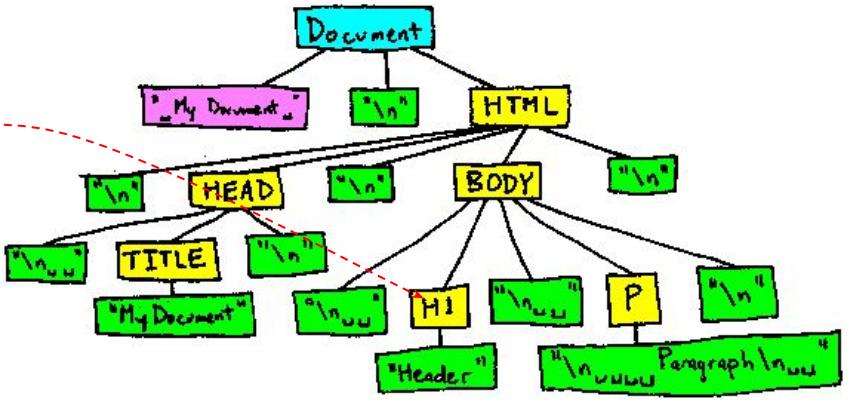
```
this.el = options.el;
this.data = options.data;
```

```
<script>
  new Vue({
    el: '#app',
    data: {
      title: "Hello World"
    }
  });
</script>
```

The `Object.assign()` method is used to copy the values of all enumerable own properties from one or more source objects to a target object. It will return the target object.

Vue.js und DOM - 2 Welten

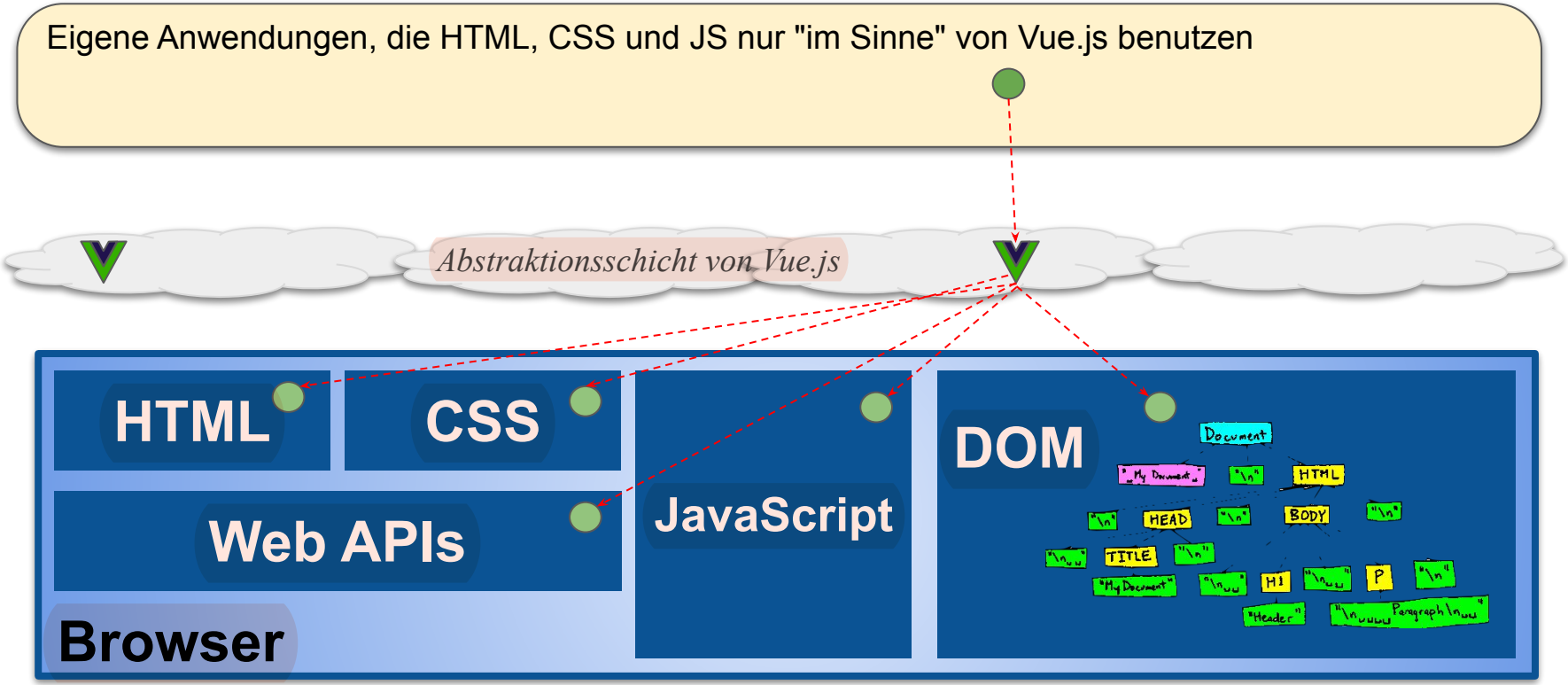
Eigene JS-Objekte und -Programme



- Mit JavaScript kann man den DOM **kontrollieren** und steuern.
- Mit JavaScript kann man dem DOM eine völlig **neue Semantik** geben.
- W3C **Web Components gehören zum DOM** und erweitern diesen.
- **Vue** bleibt **außerhalb** der DOM-Welt.
- Vue erzeugt eine **eigene Abstraktionsschicht**.

https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Whitespace_in_the_DOM

Frameworks sollen eine Wohlfühlwelt erzeugen, die den Programmierer von Unzulänglichkeiten und Problemen des Browsers abschirmt



Erstes Fazit

- Das **Vue.js**-Modell ist zunächst **unabhängig** von
  - W3C Custom Elements
  - W3C Shadow DOM
  - W3C Templates
  - ES6 Import

- Es implementiert einen **eigenen Komponentenansatz** unabhängig von W3C-Browser-Standards
- Vorteile?
- Nachteile?

LitElement integriert sich in den W3C-Komponenten-Standard

Gliederung der Vue-Einführung

1. Vue-Instanz

2. Vue-Komponenten

3. Vue-Komponenten in einer ".vue"-Datei

Vue-Instanz = Template + State + Logic

```
<div id="app">
  <h1>1. Thema {{title}}</h1>
  <h2>2. Unterthema {{title}}</h2>
  <p>Paragraph über {{title}}</p>
  <button @click="world">Change Title</button>
</div>

<script src="https://unpkg.com/vue/dist/vue.js"></script>

<script>
 new Vue({
  el: '#app',
  data: {
   title: 'Me'
  },
  methods: {
   world: function(){ this.title = 'World' }
  }
 });
</script>
```

1. Template

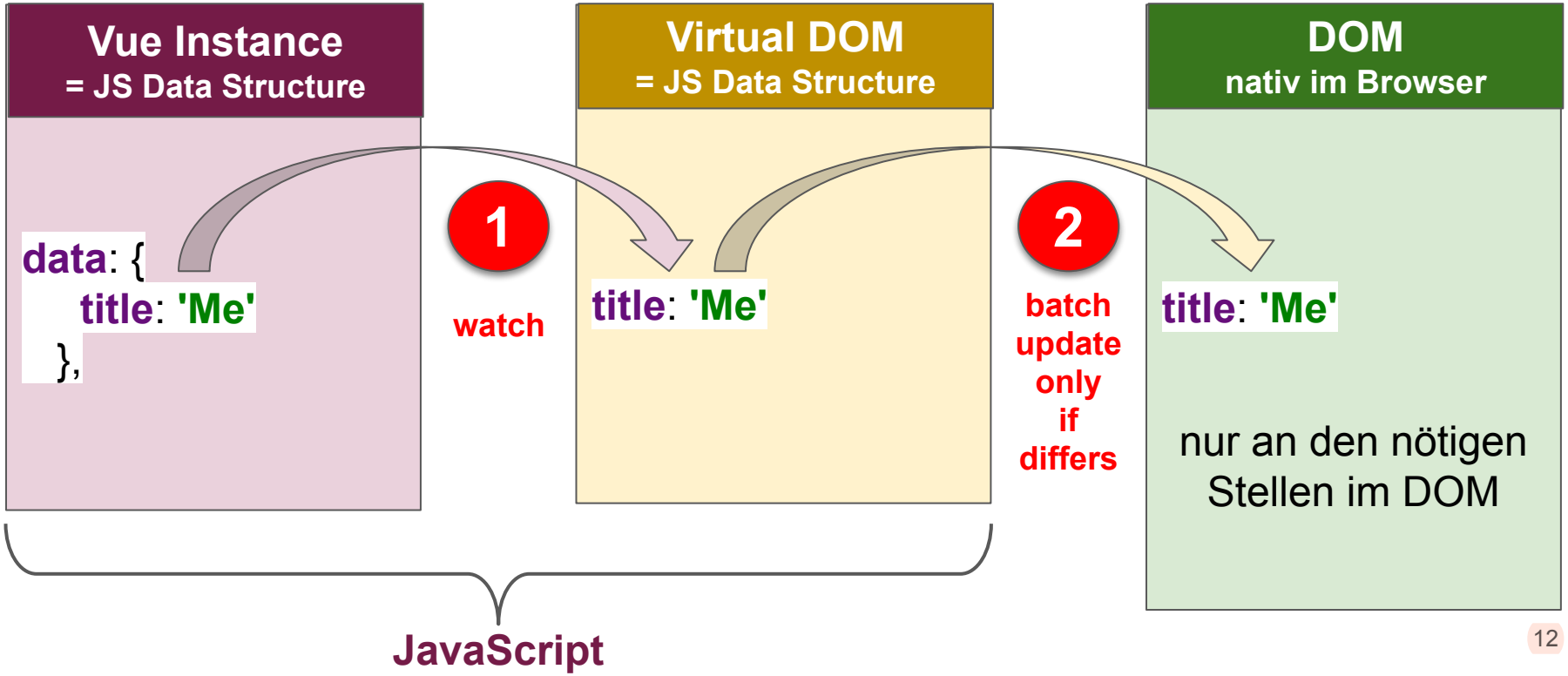
2. State

3. Logic

https://vuejs.org/v2/api/#v-on

Vue.js Virtual DOM

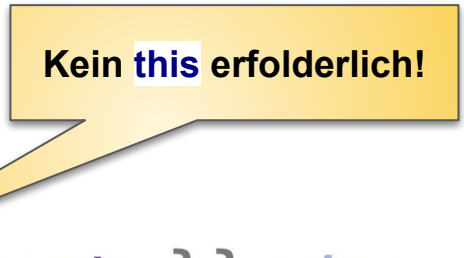
DOM Update ist die teuerste Operation



Vue.js Instance: Wichtige Begriffe

- Vue.js Template String Interpolation {{ }} + Expressions
- Vue.js Options
- Vue.js Instance Methods
- Events
- Two-way data binding
- State Management
- Message Bus
- Directives
  - v-on, v-bind, v-model, v-once, v-html, v-if, v-else, v-for, v-show
- Directive Modifiers
  - v-on:mousemove.stop === stopPropagation

Vue.js Expressions




```
<div id="app">
  <p>I have a {{ product }}</p>
  <p>{{ product + 's' }}</p>
  <p>{{ isWorking ? 'YES' : 'NO' }}</p>
  <p>{{ product.getSalePrice() }}</p>
</div>
```

https://www.vuemastery.com/pdf/Vue-Essentials-Cheat-Sheet.pdf

Vue.js **Options**: el + data + methods + computed

```
<div id="app">
  <button v-on:click="increment">+</button>
  <button v-on:click="decrement">-</button>
  <h1>{{ count }} + 3 = {{ count + 3 }} ist {{ result }}</h1>
</div>
<script src="https://unpkg.com/vue/dist/vue.js"></script>
<script>
 new Vue({
  el: '#app',
  data: {
    count: 0
  },
  methods: {
    increment: function(){ this.count += 1 },
    decrement: function(){ this.count -= 1 }
  },
  computed: {
    result: function() { return this.count + 3 > 9 ? " zweistellig " : " einstellig " }
  }
 });
</script>
```

Computed Properties

werden nur dann neu berechnet, wenn sich abhängige Daten ändern (hier **count**)

Methods

werden jedesmal neu berechnet

Das Objekt, das dem Konstruktor von Vue übergeben wird, ist das **Options**-Objekt.

https://vuejs.org/v2/api/#Options-Data

Exkurs: Vue Instance Method **$mount**

```
<div id="app">
   <h1>1. Thema {{title}}</h1>
   <h2>2. Unterthema {{title}}</h2>
   <p>Paragraph über {{title}}</p>
</div>

<script src="https://unpkg.com/vue/dist/vue.js"></script>

<script>
 new Vue({
   data: {
     title: "Hello World"
   }
}).$mount('#app')  // make DOM nodes
</script>
```

externes Template

ohne el-Property

https://vuejs.org/v2/api/#vm-mount

Exkurs: Vue Instance Method **$el**

ohne externes Template

internes Template

```
<body>
<script src="https://unpkg.com/vue/dist/vue.js"></script>
<script>
 const vm = new Vue({
  template: '<div><button @click="world">Click</button><h1>Hello {{name}}</h1></div>',
  data: {
    name: 'Me'
  },
  methods: {
    world: function(){ this.name = 'World' }
  }
 });

 vm.$mount();  // make DOM nodes

 document.body.appendChild( vm.$el ) // DIV inserted
</script>
</body>
```

https://vuejs.org/v2/api/#vm-el

Events

Kurzform:

```
<div id="app"  :mousemove="coordinates">
```

```
<div id="app" v-on:mousemove="coordinates">
  <h1>x = {{ x }}, y = {{ y }}</h1>
</div>

<script src="https://unpkg.com/vue/dist/vue.js"></script>

<script>
 new Vue({
   el: '#app',
   data: {
     x: 0,
     y: 0
   },
   methods: {
     coordinates: function( event ){ this.x = event.clientX; this.y = event.clientY;  }
   }
 });
</script>
```

mit der Vue.js-Direktive **v-on**

x = 128, y = 61

Actions / Events

Calls addToCart method on component:

```
<button v-on:click="addToCart">...
```

shorthand

```
<button @click="addToCart">...
```

Arguments can be passed:

```
<button @click="addToCart(product)">...
```

To prevent default behavior (e.g. page reload):

```
<form @submit.prevent="addProduct">...
```

Only trigger once:

```
<img @mouseover.once="showImage">...
```

`.stop`	Stop all event propagation
`.self`	Only trigger if event.target is element itself

Keyboard entry example:

```
<input @keyup.enter="submit">
```

Call onCopy when control-c is pressed:

```
<input @keyup.ctrl.c="onCopy">
```

Key modifiers:

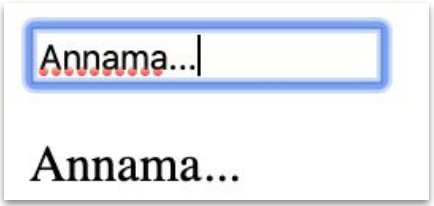
`.tab`	`.up`	`.ctrl`
`.delete`	`.down`	`.alt`
`.esc`	`.left`	`.shift`
`.space`	`.right`	`.meta`

Mouse modifiers:

`.left`	`.right`	`.middle`

https://www.vuemastery.com/pdf/Vue-Essentials-Cheat-Sheet.pdf

Two-way data binding mit v-model




```
<div id="app">
  <input type="text" v-model="name">
  <p>{{ name }}</p>
</div>

<script src="https://unpkg.com/vue/dist/vue.js"></script>

<script>
 new Vue({
   el: '#app',
   data: {
     name: 'Anna'
   }
 });
</script>
```

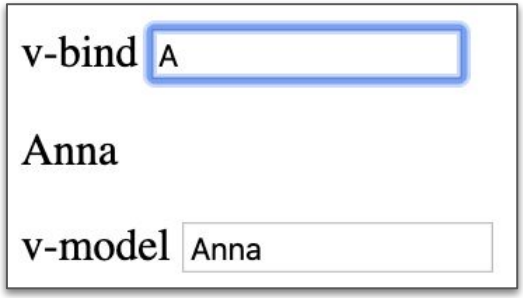
1

2

1. Änderungen im Input-Field werden zu den Daten propagiert.

2. Änderungen in den Daten werden ins Template propagiert.

One-way data binding mit v-bind




```
<div id="app">
  v-bind <input type="text" v-bind:value="name">
  <p>{{ name }}</p>
  v-model <input type="text" v-model="name">
</div>

<script src="https://unpkg.com/vue/dist/vue.js"></script>

<script>
 new Vue({
  el: '#app',
  data: {
    name: 'Anna'
  }
 });
</script>
```

Änderungen in den Daten werden ins Template propagiert, aber nicht umgekehrt.

v-model ⇔ (v-bind + v-on)

```
<div id="app">
  <input type="text" v-model="name">
  <p>{{ name }}</p>
</div>

<script src="https://unpkg.com/vue/dist/vue.js"></script>

<script>
 new Vue({
   el: '#app',
   data: {
     name: 'Anna'
   }
 });
</script>
```

```
<div id="app">
  <input type="text" v-bind:value="name"
                     v-on:input="name = $event.target.value">
  <p>{{ name }}</p>
</div>
<script src="https://unpkg.com/vue/dist/vue.js"></script>
<script>
 const app = new Vue({
   el: '#app',
   data: {
     name: 'Anna'
   }
 });
</script>
```

Binding mit v-bind

```
<a v-bind:href="url">...</a>
```

shorthand

```
<a :href="url">...</a>
```

True or false will add or remove attribute:

```
<button :disabled="isButtonDisabled”>...
```

If isActive is truthy, the class ‘active’ will appear:

```
<div :class="{ active: isActive }">...
```

Vue.js-Kurzform für: **v-bind:class**

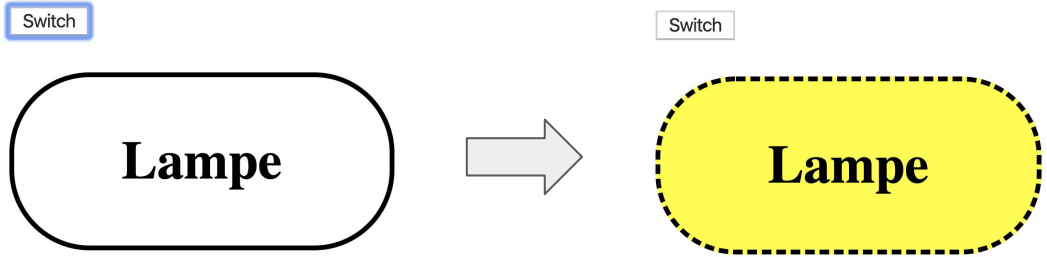
Style color set to value of activeColor:

```
<div :style="{ color: activeColor }">
```

https://www.vuemastery.com/pdf/Vue-Essentials-Cheat-Sheet.pdf

v-bind:class

```
<style>
  h1 {
     text-align: center;
     border: 3px solid;
     padding: 2rem;
     width: 10rem;
     border-radius: 3rem;
  }
  .on {
     background-color: yellow;
     border-style: dashed;
  }
</style>
<div id="app">
  <button @click="toggle">Switch</button>
  <h1 v-bind:class="{ on }">Lampe</h1>
</div>
```




```
<script src="https://unpkg.com/vue/dist/vue.js"></script>

<script>
 new Vue({
   el: '#app',
   data: {
     on: false
   },
   methods: {
     toggle: function(){
       this.on = ! this.on;
     }
   }
 });
</script>
```

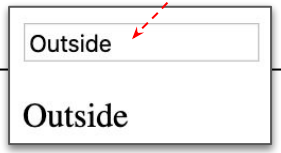
ES6-Kurzform für:

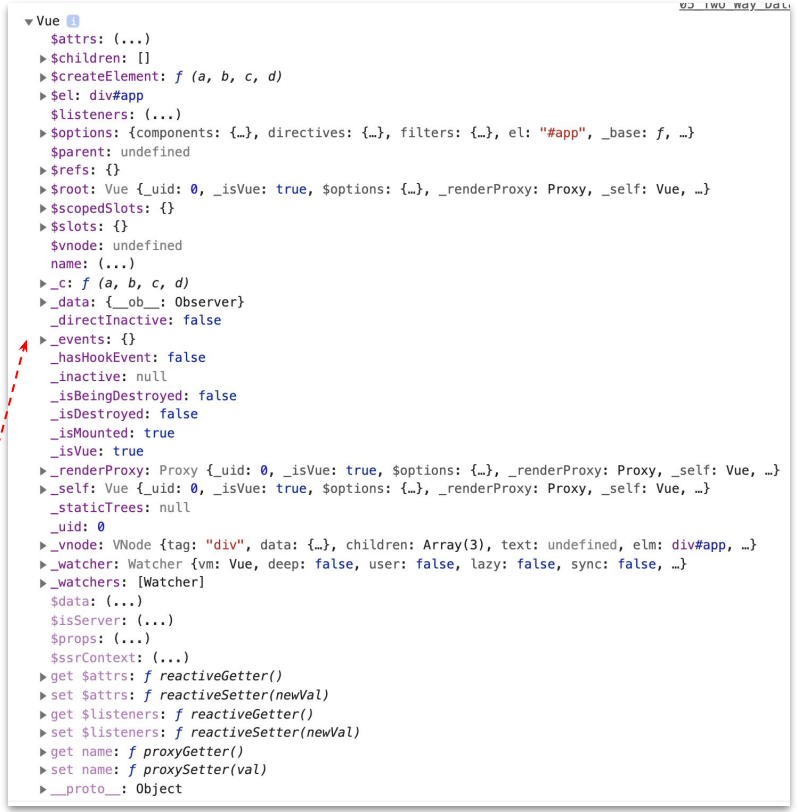
```
<h1 v-bind:class="{ on: on }">Lampe</h1>
```

Die Vue-Instanz

ist nicht gekapselt

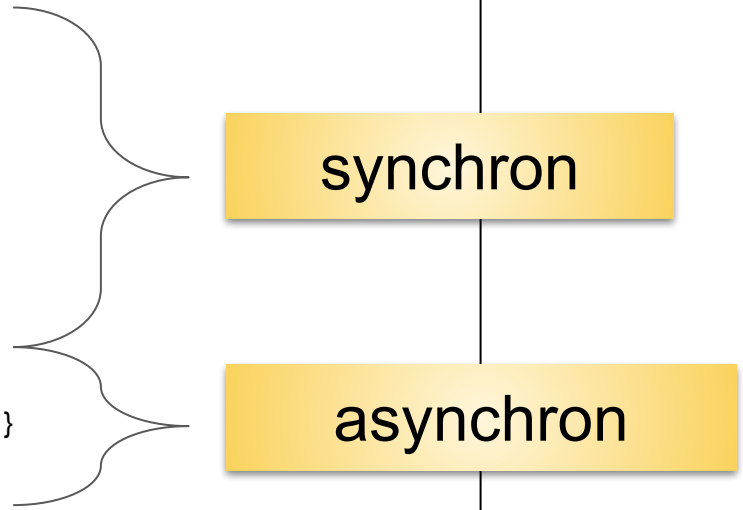
```
<div id="app">
  <input type="text" v-model="name">
  <p>{{ name }}</p>
</div>

<script src="https://unpkg.com/vue/dist/vue.js"></script>

<script>

 const app = new Vue({
   el: '#app',
   data: {
     name: 'Anna'
   }
 });

 console.log( app );
 app.$data.name = "Outside";

</script>
```





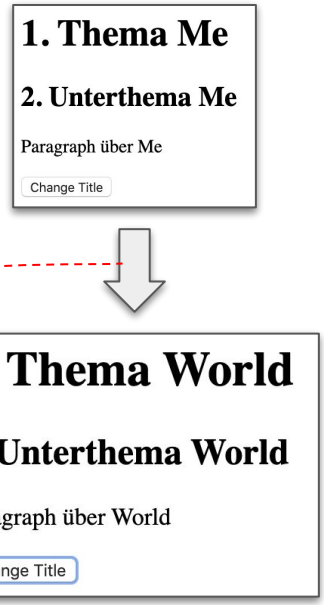
Asynchrones Update mit watch

```
<div id="app">
  <button v-on:click="increment">+</button>
  <button v-on:click="decrement">-</button>
  <h1>{{ count }} + 3 = {{ count + 3 }}: {{ result }}</h1>
</div>
<script src="https://unpkg.com/vue/dist/vue.js"></script>
<script>
 const sleep = seconds => new Promise(resolve => setTimeout(resolve, 1000 * seconds));

 new Vue({
  el: '#app',
  data: {
    count: 0,
    result: 'einstellig'
  },
  methods: {
    increment: function(){ this.count += 1 },
    decrement: function(){ this.count -= 1 }
  },
  watch: {
    count: async function( value) { await sleep(3);
      this.result = value + 3 > 9 ? 'zweistellig' : 'einstellig' }
  }
 });
</script>
```



Sparsame Updates

```
<div id="app">
  <h1>1. Thema {{title}}</h1>
  <h2>2. Unterthema {{title}}</h2>
  <p>Paragraph über {{title}}</p>
  <button @click="world">Change Title</button>
</div>
<script src="https://unpkg.com/vue/dist/vue.js"></script>
<script>
 new Vue({
  el: '#app',
  data: {
   title: 'Me'
  },
  methods: {
   world: function(){ this.title = 'World' }
  },
  updated: function(){ console.log( 'updated' ); }
 });
</script>
```



Vue Lifecycle-Methode "updated" wird nur einmal aufgerufen

Options / Lifecycle Hooks

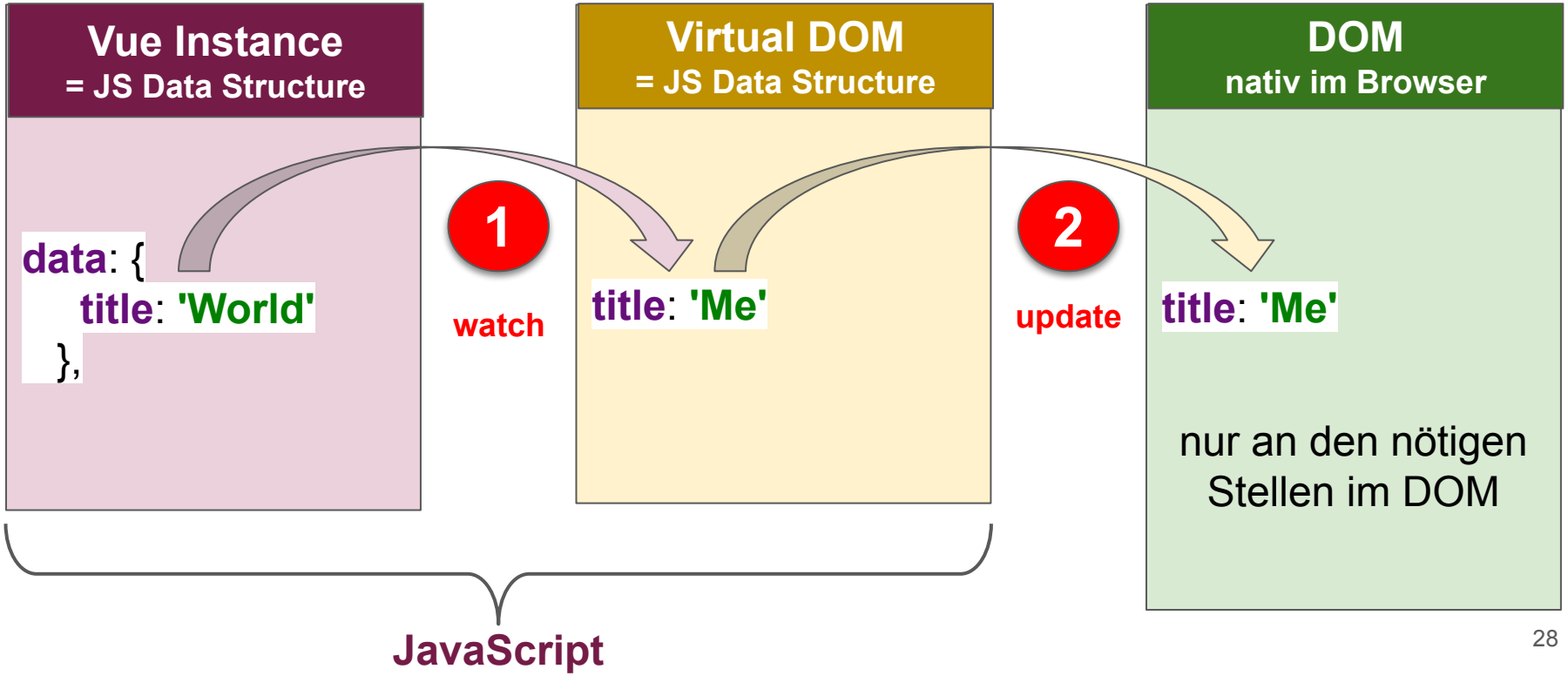
- beforeCreate
- created
- beforeMount
- mounted
- beforeUpdate
- **updated**
- activated
- deactivated
- beforeDestroy
- destroyed
- errorCaptured

https://vuejs.org/v2/api/#updated

Vue.js Virtual DOM mit Update

```
<button @click="world">Change Title</button>
```

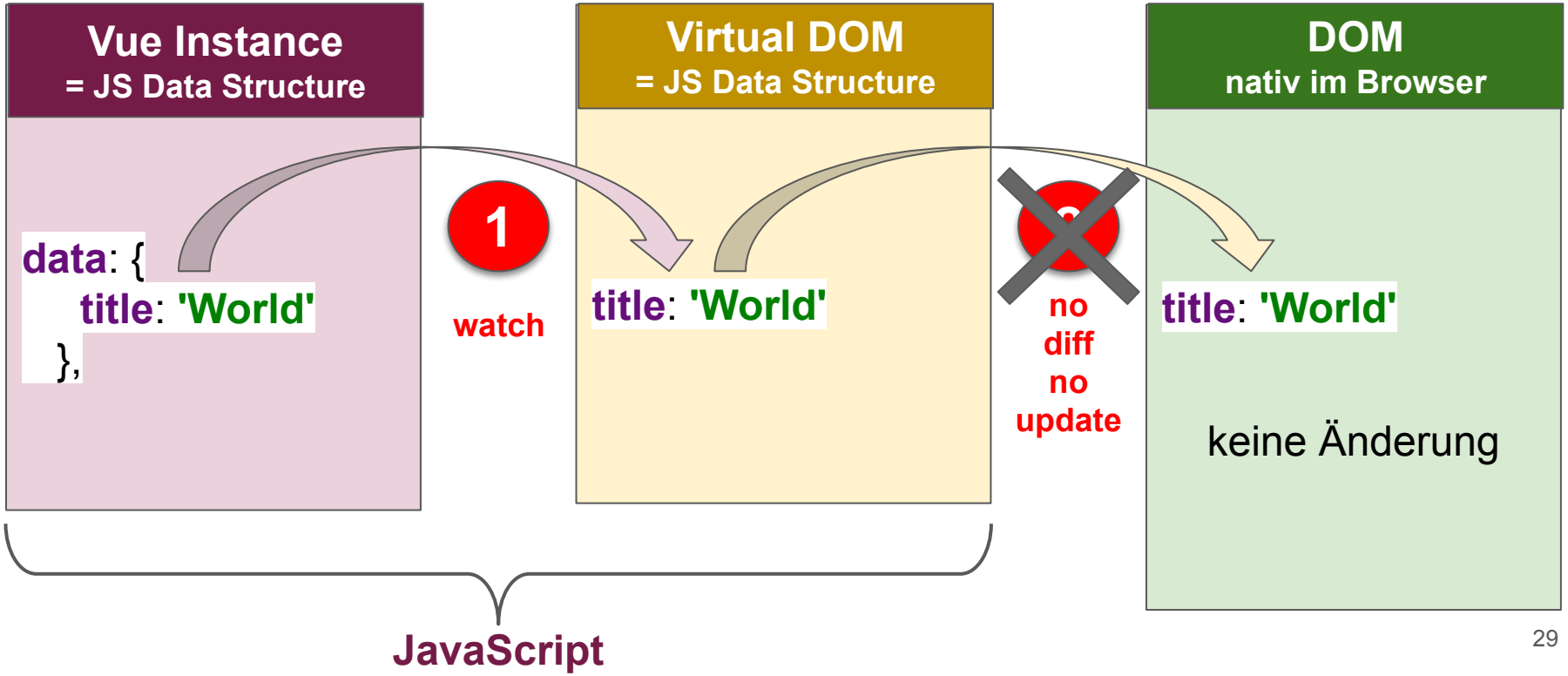
DOM Update ist die teuerste Operation



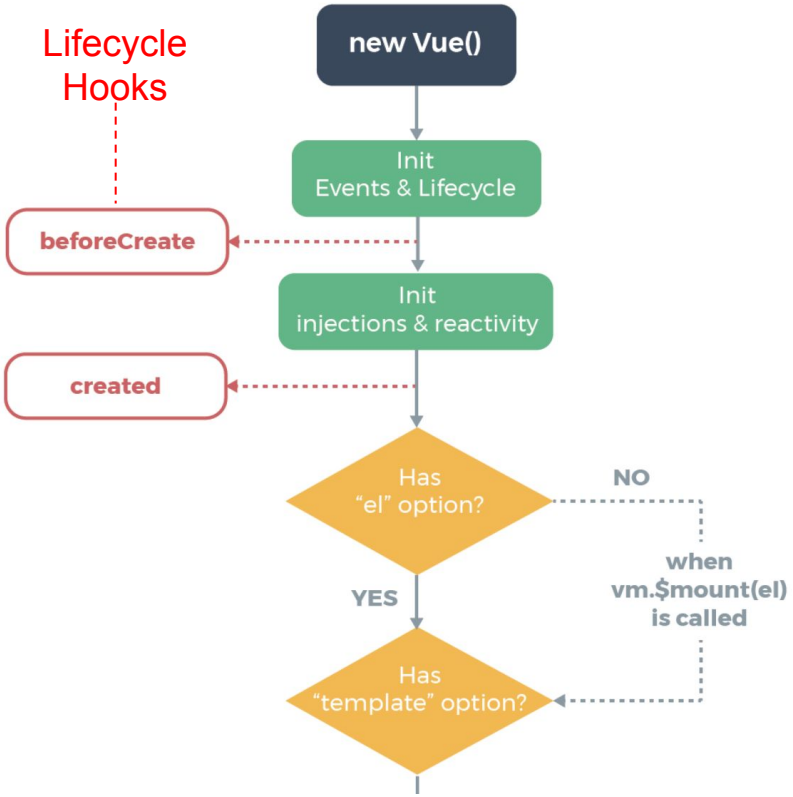
Vue.js Virtual DOM ohne Update

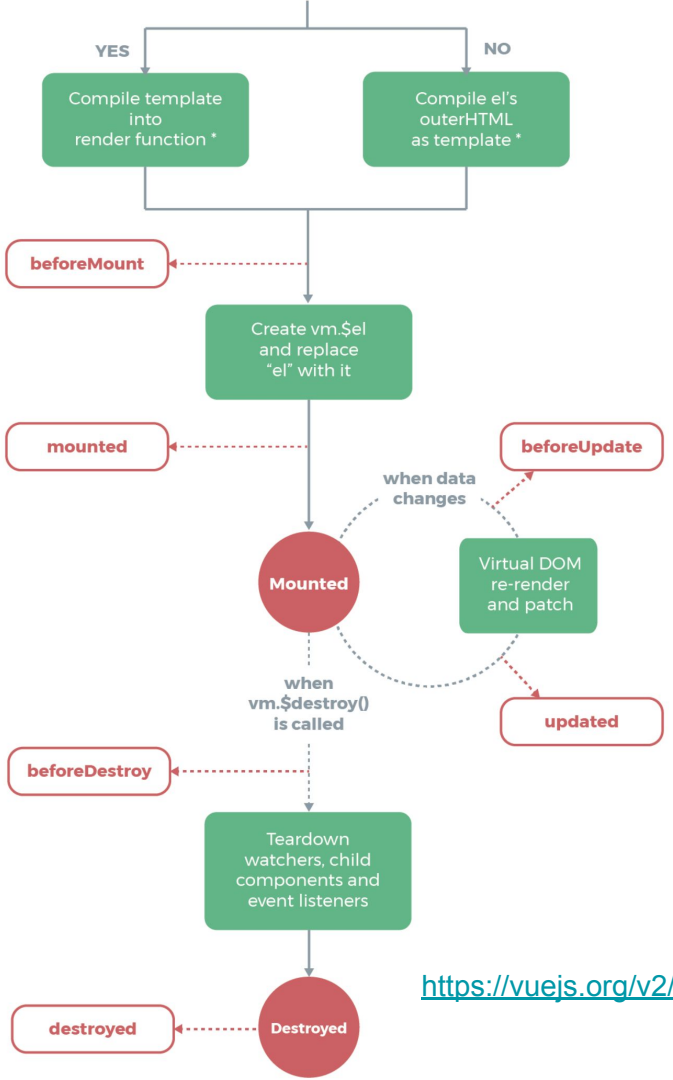
```
<button @click="world">Change Title</button>
```

DOM Update ist die teuerste Operation



Vue.js Lifecycle

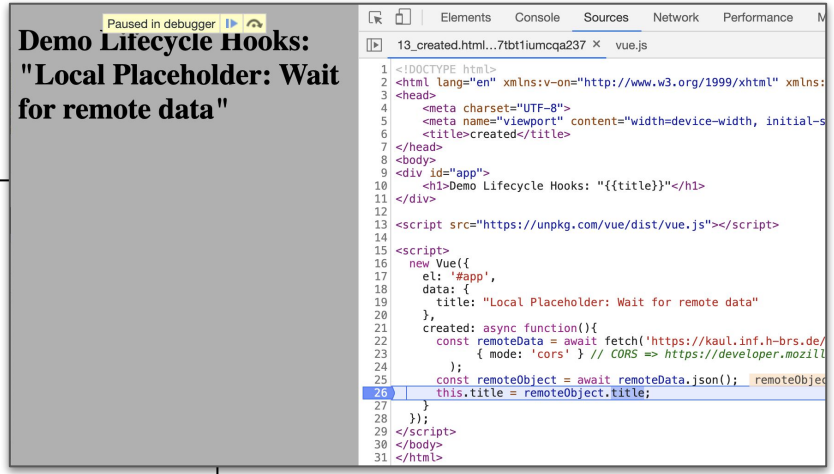




https://vuejs.org/v2/guide/instance.html

Lifecycle Method "created"

```
<div id="app">
  <h1>Demo Lifecycle Hooks: "{{title}}"</h1>
</div>

<script src="https://unpkg.com/vue/dist/vue.js"></script>

<script>
 new Vue({
  el: '#app',
  data: {
   title: "Local Placeholder: Wait for remote data"
  },
  created: async function(){
   const remoteData = await fetch('https://kaul.inf.h-brs.de/data/created-demo.json',
        { mode: 'cors' } // CORS => https://developer.mozilla.org/de/docs/Web/HTTP/CORS
     );
   const remoteObject = await remoteData.json();
   this.title = remoteObject.title;
  }
 });
</script>
```



1

2

Demo Lifecycle Hooks: "Title fetched from Server"

https://vuejs.org/v2/api/#created

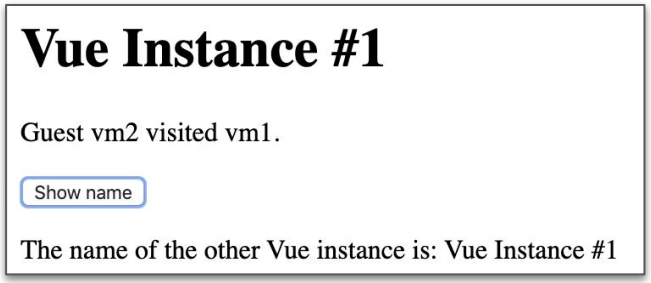
https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

Vue.js innere Referenzen mit ref

```
<div id="vm1">
  <h1>{{ name }}</h1>
  <p ref="paragraph"></p>
</div>

<div id="vm2">
  <button @click="showName">Show name</button>
  <p>{{ message }}</p>
</div>

<script src="https://unpkg.com/vue/dist/vue.js"></script>

<script>

 const vm1 = new Vue({
   el: '#vm1',
   data: {
     name: 'Vue Instance #1'
   }
 });
```

```
const vm2 = new Vue({
  el: '#vm2',
  data: {
    message: ''
  },
  methods: {
    showName: function() {
      this.message = "The name of the other Vue instance is: "
                   + vm1.$data.name;
      vm1.$refs.paragraph.textContent =
                              "Guest vm2 visited vm1.";
    }
  }
});

</script>
```

Zugriff von außen



Merke: **ref** ist das **href** für **Vue.js**-Substrukturen.

Share data between two Vue instances

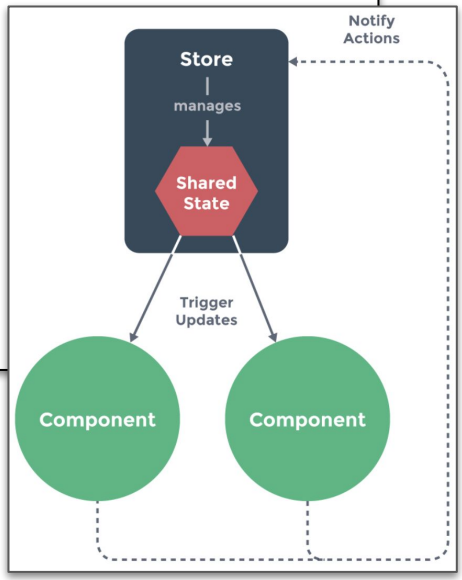

```
<div id="app-one">
  <button @click="showMessage">Show Message</button>
</div>

<div id="app-two">
  <p>{{ shared.message }}</p>
</div>

<script src="https://unpkg.com/vue/dist/vue.js"></script>

<script>

 const shared = {
   message: "Initial Message"
 };
```

```
new Vue({
  el:"#app-one",
  data:{
    shared
  },
  methods: {
    showMessage: function(){ this.shared.message =
                                                "Hello World!" }
  }
});

new Vue({
  el:"#app-two",
  data:{
    shared
  }
})

</script>
```

State Management

```
<div id="app-one">
  <button @click="showMessage">Show Message</button>
</div>

<div id="app-two">
  <p>{{ sharedState.state.message }}</p>
</div>

<script src="https://unpkg.com/vue/dist/vue.js"></script>

<script>

 const store = {
   state: {
     message: 'Hello!'
   },
   setMessageAction (newValue) {
     this.state.message = newValue
   },
   clearMessageAction () {
     this.state.message = ''
   }
 };
```

```
const vm1 = new Vue({
  el: '#app-one',
  data: {
    privateState: {},
    sharedState: store
  },
  methods: {
    showMessage: function(){
      this.sharedState.setMessageAction( "Hello World!" ) }
  }
});

const vm2 = new Vue({
  el: '#app-two',
  data: {
    privateState: {},
    sharedState: store
  }
});
```



https://vuejs.org/v2/guide/state-management.html

Ein zentraler Store löst das Problem zirkulärer Abhängigkeiten

```
<div id="firstname">
  <input type="text" v-model="name">
  <p>{{ name }} {{ lastname() }}</p>
</div>
<div id="lastname">
  <input type="text" v-model="name">
  <p>{{ firstname() }} {{ name }}</p>
</div>
<script src="https://unpkg.com/vue/dist/vue.js"></script>
<script>

 const app1 = new Vue({
   el: '#firstname',
   data: {
     name: 'Anna'
   },
   methods: {
     lastname: function(){ return app2.$data.name }
   }
 });
```

```
const app2 = new Vue({
  el: '#lastname',
  data: {
    name: 'Müller'
  },
  methods: {
    firstname: function(){ return app1.$data.name }
  }
});

</script>
```

```
▶ [Vue warn]: Error in render:                    vue.js:634
"ReferenceError: Cannot access 'app2' before
initialization"

(found in <Root>)
```

Message sending between two Vue instances

```
<div id="app-one">
  <button @click="sendMessage">sendMessage</button>
</div>
<div id="app-two">
  <p>{{message}}</p>
</div>

<script src="https://unpkg.com/vue/dist/vue.js"></script>

<script>

const bus = new Vue();

new Vue({
  el:"#app-one",
  methods:{
    sendMessage(){
      bus.$emit('send-message', "hello from app one!")
    }
  }
});
```

Vue Event Sender

Vue Instance Methods / Events

- vm.$on
- vm.$once
- vm.$off
- vm.$emit

```
new Vue({
  el:"#app-two",
  data:{
    message: null
  },
  mounted(){
    bus.$on("send-message", message =>
            this.message = message);
  }
 })
</script>
```

Vue Event Receiver

Vue Directives

Element inserted/removed based on truthiness:

```
<p v-if="inStock">{{ product }}</p>
```

```
<p v-else-if="onSale">...</p>
<p v-else>...</p>
```

Toggles the display: none CSS property:

```
<p v-show="showProductDetails">...</p>
```

Two-way data binding:

```
<input v-model="firstName" >
```

Directive Modifiers

`v-model.lazy="..."`	Syncs input after change event
`v-model.number="..."`	Always returns a number
`v-model.trim="..."`	Strips whitespace

https://www.vuemastery.com/pdf/Vue-Essentials-Cheat-Sheet.pdf

Vue Directives for List Rendering

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

key always recommended

To access the position in the array:

```
<li v-for="(item, index) in items">...
```

To iterate through objects:

```
<li v-for="(value, key) in object">...
```

Using v-for with a component:

```
<cart-product v-for="item in products"
              :product="item" :key="item.id">
```

https://vuejs.org/v2/api/#v-for

https://www.vuemastery.com/pdf/Vue-Essentials-Cheat-Sheet.pdf

Zusammenfassung Vue.js Instanz https://vuejs.org/v2/api/

Directives

- v-text
- v-html
- v-show
- v-if
- v-else
- v-else-if
- v-for
- v-on
- v-bind
- v-model
- v-slot
- v-pre
- v-cloak
- v-once

Special Attributes

- key
- ref
- is
- slot
- slot-scope
- scope

Options / Lifecycle Hooks

- beforeCreate
- created
- beforeMount
- mounted
- beforeUpdate
- updated
- activated
- deactivated
- beforeDestroy
- destroyed
- errorCaptured

Gliederung der Vue-Einführung

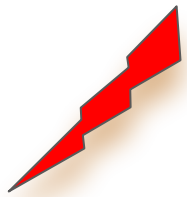
1. Vue-Instanz

2. Vue-Komponenten

3. Vue-Komponenten in einer ".vue"-Datei

Vue Components

- Vue-Instanzen sind **für genau den einen Webseitenbereich zuständig**, in den sie gemounted werden (mit der Vue-Option **el** oder mit der Vue-Instanzen-Methode **$mount**)
- Verschiedene Webseitenbereiche können **nicht von der gleichen Vue-Instanz** dirigiert werden.
- Vue-Instanzen sind
  - nicht mehrfach einsetzbar
  - **nicht wiederverwendbar**



https://vuejs.org/v2/guide/components.html

Vue instances are **not** re-usable

```
<body>
<div class="hello"></div>
<div class="hello"></div>
<div class="hello"></div>

<script src="https://unpkg.com/vue/dist/vue.js"></script>

<script>
  new Vue({
    el: '.hello',
    template: '<h1>Hello</h1>'
  });
</script>
</body>
```

wird nur im ersten DIV gemounted

Klassen-Selector

Vue-Instanzen sind genau einem DOM-Element eindeutig zugeordnet.

Zur mehrfachen Wiederverwendung in der gleichen Seite benötigt man **Vue Components**!

Vue Components: Re-usable Vue code within Vue

```
<div id="app">
  <hello></hello>
  <hello></hello>
  <hello></hello>
</div>

<script src="https://unpkg.com/vue/dist/vue.js"></script>

<script>
  Vue.component('hello', {
    template: '<h1>Hello</h1>'
  });

  new Vue({
    el: '#app'
  });

</script>
```

Nur innerhalb Vue

1. Gobale Registrierung der Komponente hello

2. Vue Apps definieren Bereiche, in denen die registrierten Komponenten verwendet werden dürfen

Jede Vue Component hat eigenen Zustand

```
<div id="app">
  <hello></hello>
  <hello></hello>
  <hello></hello>
</div>
<script>
  Vue.component('hello', {
    template: '<div><button @click="world">Click</button><h1>Hello {{name}}</h1></div>',
    data: function(){
      return {
        name: 'Me'
      }
    },
    methods: {
      world: function(){ this.name = 'World' }
    }
  });
  new Vue({
    el: '#app'
  });
</script>
```

ohne "el" Property

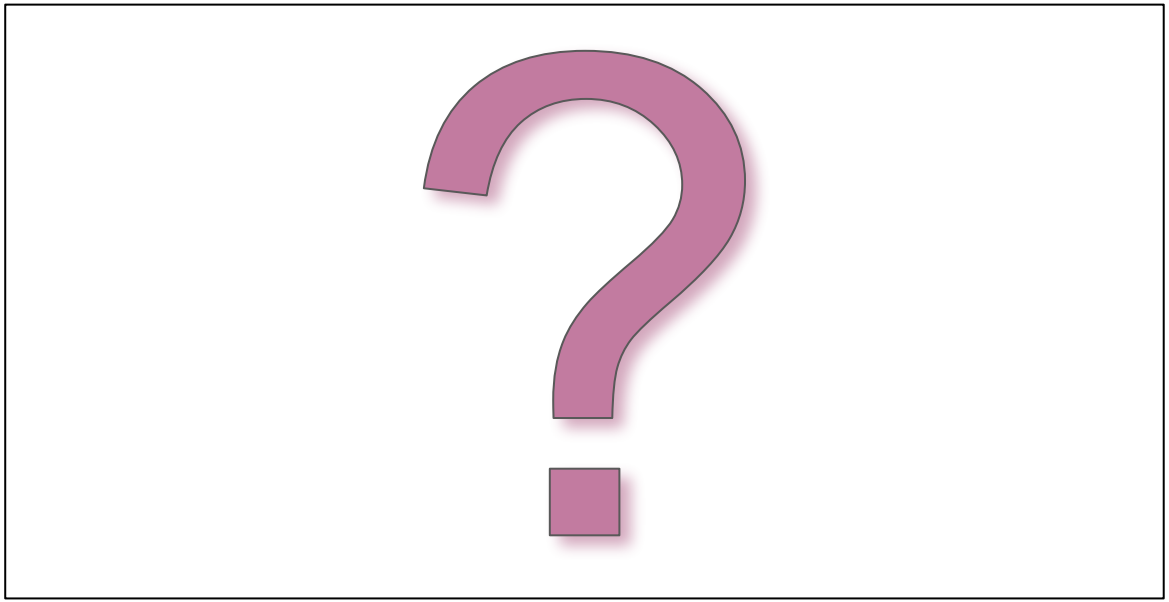
Function

`data` Must Be a Function

Since components are reusable Vue instances, they accept the same options as `new Vue`, such as `data`, `computed`, `watch`, `methods`, and lifecycle hooks. The only exceptions are a few root-specific options like `el`.

https://vuejs.org/v2/guide/components.html

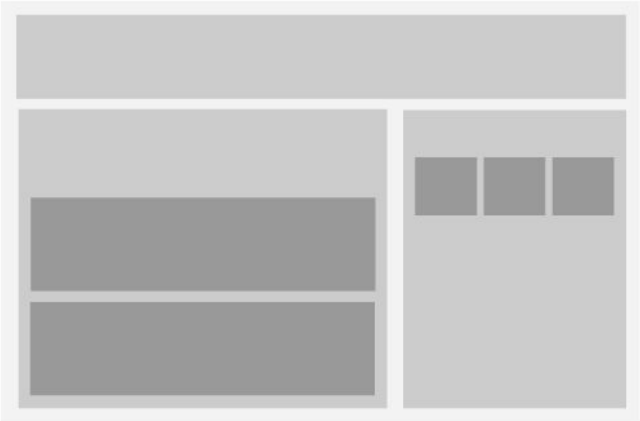
Aufgabe: Counter als Vue Component

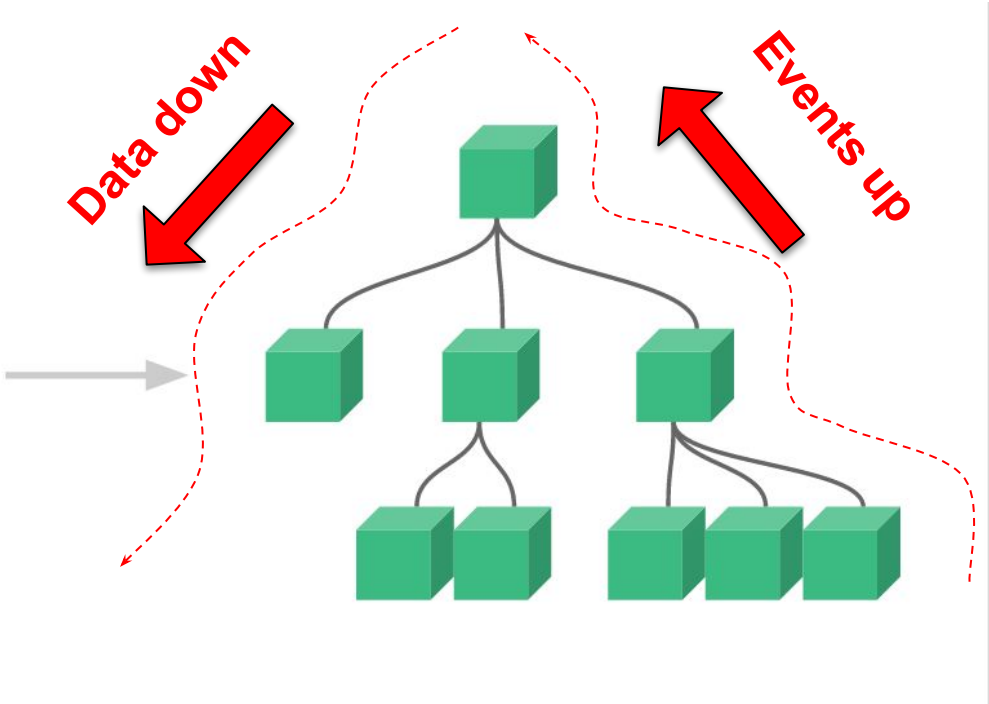


You clicked me 2 times.

You clicked me 0 times.

WebApp zerlegen in wiederverwendbare Komponenten





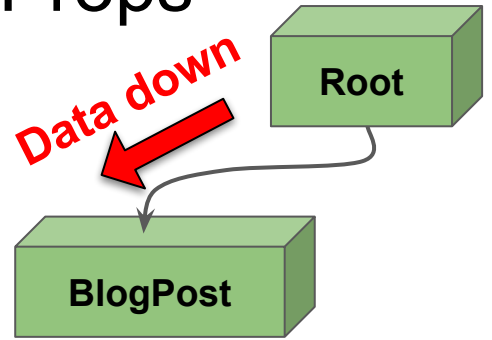
https://vuejs.org/v2/guide/index.html#Composing-with-Components

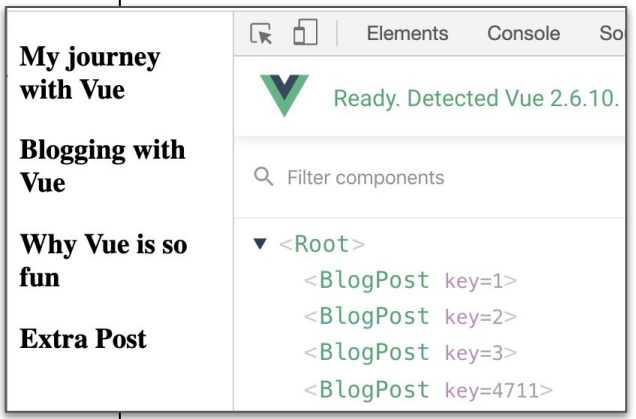
Passing Data to Child Components with Props

```
<div id="components-demo">
  <blog-post v-for="post in posts" v-bind:key="post.id" v-bind:title="post.title"></blog-post>
  <blog-post v-bind:key="4711" title="Extra Post"></blog-post>
</div>
<script src="https://unpkg.com/vue/dist/vue.js"></script>
<script>
 Vue.component('blog-post', {
   props: ['title'],
   template: '<h3>{{ title }}</h3>'
 });
 new Vue({
   el: '#components-demo',
   data: {
     posts: [
       { id: 1, title: 'My journey with Vue' },
       { id: 2, title: 'Blogging with Vue' },
       { id: 3, title: 'Why Vue is so fun' },
     ]
   }
 });
</script>
```

BlogPost

Root





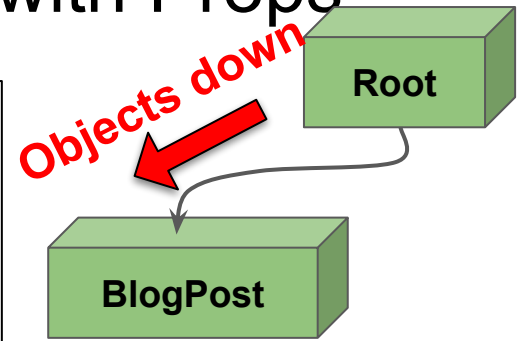
https://vuejs.org/v2/guide/components.html#Passing-Data-to-Child-Components-with-Props

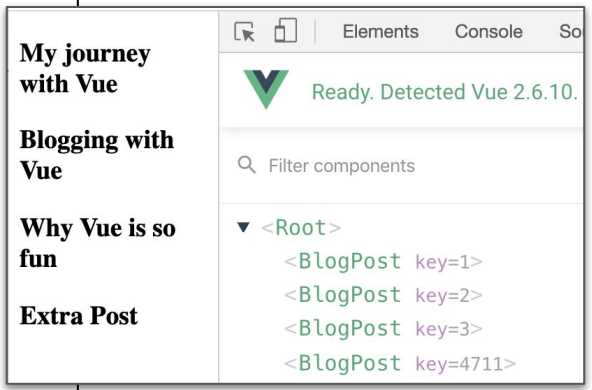
Passing **Objects** to Child Components with Props

```
<div id="components-demo">
  <blog-post v-for="post in posts" v-bind:key="post.id" v-bind:post="post"></blog-post>
  <blog-post v-bind:key="4711"
             v-bind:post='{"id":"1","title":"Extra Post","content":"E"}'></blog-post>
</div>
<script src="https://unpkg.com/vue/dist/vue.js"></script>
<script>
 Vue.component('blog-post', {
  props: ['post'],
  template: '<div><h3>{{ post.title }}</h3><p>{{ post.content }}</p></div>'
 });
 new Vue({
  el: '#components-demo',
  data: {
   posts: [
    { id: 1, title: 'My journey with Vue', content: 'A' },
    { id: 2, title: 'Blogging with Vue', content: 'B'  },
    { id: 3, title: 'Why Vue is so fun', content: 'C'  },
   ]
  }
 });
</script>
```

template darf nur eine Wurzel haben

Root





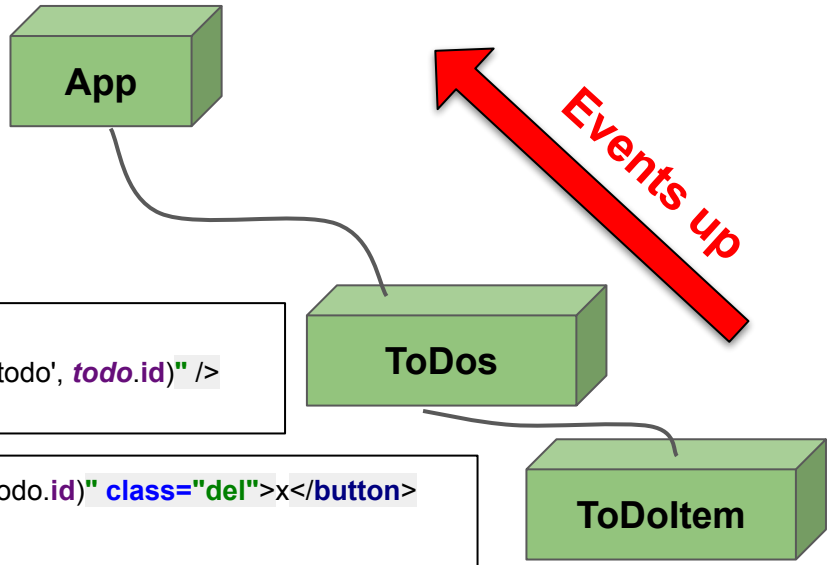
https://vuejs.org/v2/guide/components.html#Passing-Data-to-Child-Components-with-Props

Events up

```
<div id="app">
 <ToDos v-bind:todos="todos" v-on:del-todo="deleteToDo" />
</div>
// ...
methods: {
 deleteToDo(id){
   this.todos = this.todos.filter(todo => todo.id !== id);
 }
},
```

```
<div v-for="todo in todos" v-bind:key="todo.id">
   <TodoItem v-bind:todo="todo" v-on:del-todo="$emit('del-todo', todo.id)" />
</div>
```

```
<button @click="$emit('del-todo', todo.id)" class="del">x</button>
{{todo.title}}
```



Vue.js Component

```
Vue.component('my-component', {
  components: { Components that can be used in the template
    ProductComponent, ReviewComponent
  },
  props: {  → The parameters the component accepts
    message: String,
    product: Object,
    email: {
      type: String,
      required: true,
      default: "none"
      validator: function (value) {
        Should return true if value is valid
      }
    }
  },
  data: function() {  Must be a function
    return {
      firstName: 'Vue',
      lastName: 'Mastery'
    }
  },
  computed: {     Return cached values until
    fullName: function () { dependencies change
      return this.firstName + ' ' + this.lastName
    }
  },
  watch: {  Called when firstName changes value
    firstName: function (value, oldValue) { ... }
  },
  methods: { ... },
  template: '<span>{{ message }}</span>',
})       Can also use backticks for multi-line
```

https://www.vuemastery.com/pdf/Vue-Essentials-Cheat-Sheet.pdf

https://vuejs.org/v2/guide/components.html

Slot

Component template:

```
<div>
  <h2>I'm a title</h2>
    <slot>
      Only displayed if no slot content
    </slot>
</div>
```

Use of component with data for slot:

```
<my-component>
  <p>This will go in the slot</p>
</my-component>
```

https://www.vuemastery.com/pdf/Vue-Essentials-Cheat-Sheet.pdf

https://vuejs.org/v2/guide/components.html

Multiple Slots

Component template:

```
<div class="container">
  <header>
    <slot name="header"></slot>
  </header>
  <main>
    <slot>Default content</slot>
  </main>
  <footer>
    <slot name="footer"></slot>
  </footer>
</div>
```

Use of component with data for slot:

```
<app-layout>
<h1 slot="header">Page title</h1>
<p>the main content.</p>
<p slot="footer">Contact info</p>
</app-layout>
```

https://www.vuemastery.com/pdf/Vue-Essentials-Cheat-Sheet.pdf

https://vuejs.org/v2/guide/components.html

Dynamic Components mit v-bind:is


```
<div id="dynamic-component-demo" class="demo">
  <button
        v-for="tab in tabs"
        v-bind:key="tab"
        v-bind:class="['tab-button', { active: currentTab === tab }]"
        v-on:click="currentTab = tab"
  >{{ tab }}</button>

  <component
        v-bind:is="currentTabComponent"
        class="tab"
  ></component>
</div>

<script src="https://unpkg.com/vue"></script>
```

```
<script>
 Vue.component('tab-home', {
   template: '<div>Home content</div>'
 });
 Vue.component('tab-posts', {
   template: '<div>Posts content</div>'
 });
 Vue.component('tab-archive', {
   template: '<div>Archive content</div>'
 });

 new Vue({
   el: '#dynamic-component-demo',
   data: {
     currentTab: 'Home',
     tabs: ['Home', 'Posts', 'Archive']
   },
   computed: {
     currentTabComponent: function () {
       return 'tab-' + this.currentTab.toLowerCase()
     }
   }
 })
</script>
```

https://vuejs.org/v2/guide/components.html#Dynamic-Components

Gliederung der Vue-Einführung

1. Vue-Instanz

2. Vue-Komponenten

3. Vue-Komponenten in einer ".vue"-Datei

Kommandozeile mit Vue CLI

Vue CLI erlaubt "*.vue" - Dateien,
die aus **<template>**, **<script>** und **<style>** bestehen
und erst in HTML, JS und CSS transformiert werden müssen.
Für die Transformation benötigt man Vue CLI.

Vue CLI

🛠 Standard Tooling for Vue.js Development

```
npm install -g @vue/cli
     @vue/cli@4.1.1
     added 1173 packages from 641 contributors
                                    in 44.026s
> vue --version
    @vue/cli 4.1.1
```

App.vue

```
<template>
 <div id="app">
   <img alt="Vue logo" src="./assets/logo.png">
   <HelloWorld msg="Welcome to Your Vue.js App"/>
 </div>
</template>
```

1

```
<script>
import HelloWorld from './components/HelloWorld.vue'
export default {
 name: 'app',
 components: {
   HelloWorld
 }
}
</script>
```

2

```
<style>
#app {
 font-family: 'Avenir', Helvetica, Arial, sans-serif;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
 text-align: center;
 color: #2c3e50;
 margin-top: 60px;
}
</style>
```

3

https://cli.vuejs.org/

Vue Compiler

App.vue

```
<template>
 <div id="app">
   <img alt="Vue logo" src="./assets/logo.png">
   <HelloWorld msg="Welcome to Your Vue.js App"/>
 </div>
</template>
```

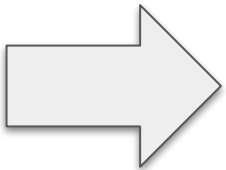
1

```
<script>
import HelloWorld from './components/HelloWorld.vue'
export default {
 name: 'app',
 components: {
   HelloWorld
 }
}
</script>
```

2

```
<style>
#app {
 font-family: 'Avenir', Helvetica, Arial, sans-serif;
 text-align: center;
 color: #2c3e50;
 margin-top: 60px;
}
</style>
```

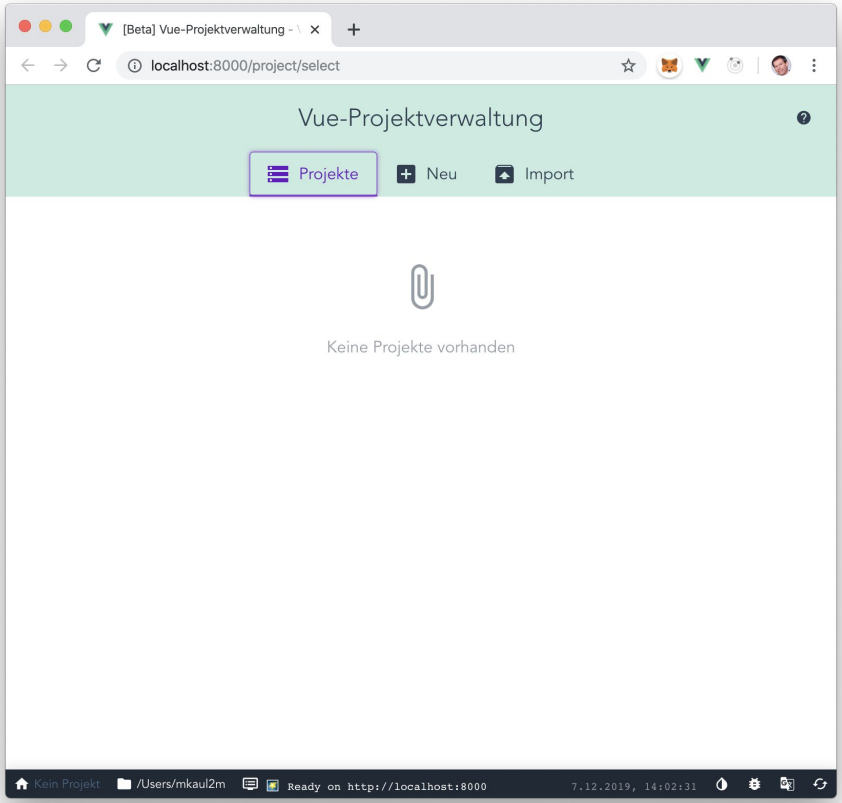
3



```
<style>
  #app {
      font-family: 'Avenir', Helvetica, Arial, sans-serif;
      text-align: center;
      color: #2c3e50;
      margin-top: 60px;
  }
</style>
<div id="app">
   <img alt="Vue logo" src="./assets/logo.png">
   <HelloWorld msg="Welcome to Your Vue.js App"/>
</div>
<script src="https://unpkg.com/vue/dist/vue.js"></script>
<script>
 Vue.component('HelloWorld', {
   props: ["msg"]
 });
 new Vue({
   el: '#app',
   components: {
     HelloWorld
   }
 });
</script>
```

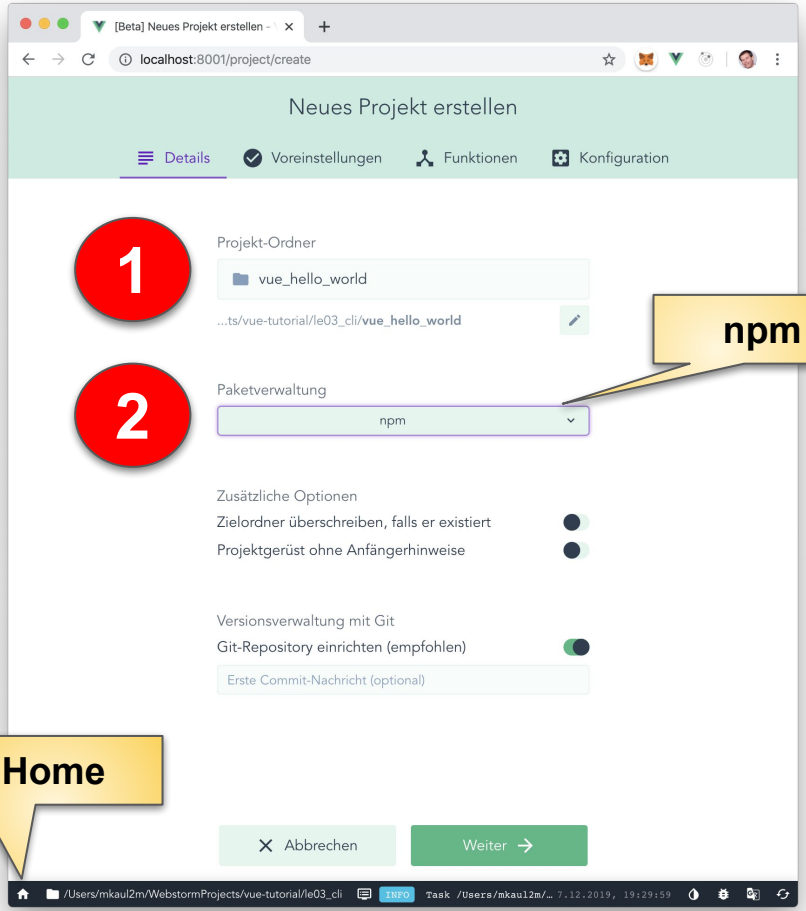
"*.vue"-Dateien werden vom Browser nicht unterstützt. Daher ist ein Build mit einer Compilation in Standard HTML+CSS+JS erforderlich.

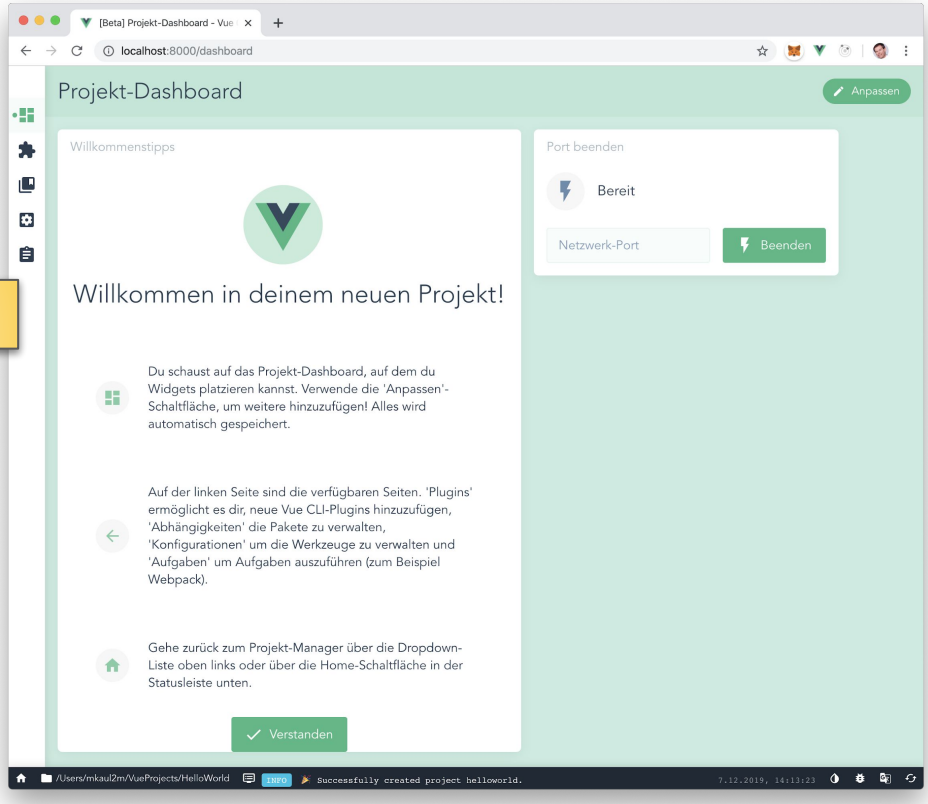
Vue GUI



Alternative zur Kommandozeile

https://cli.vuejs.org/guide/creating-a-project.html#using-the-gui





https://cli.vuejs.org/

Erste Vue App starten

1

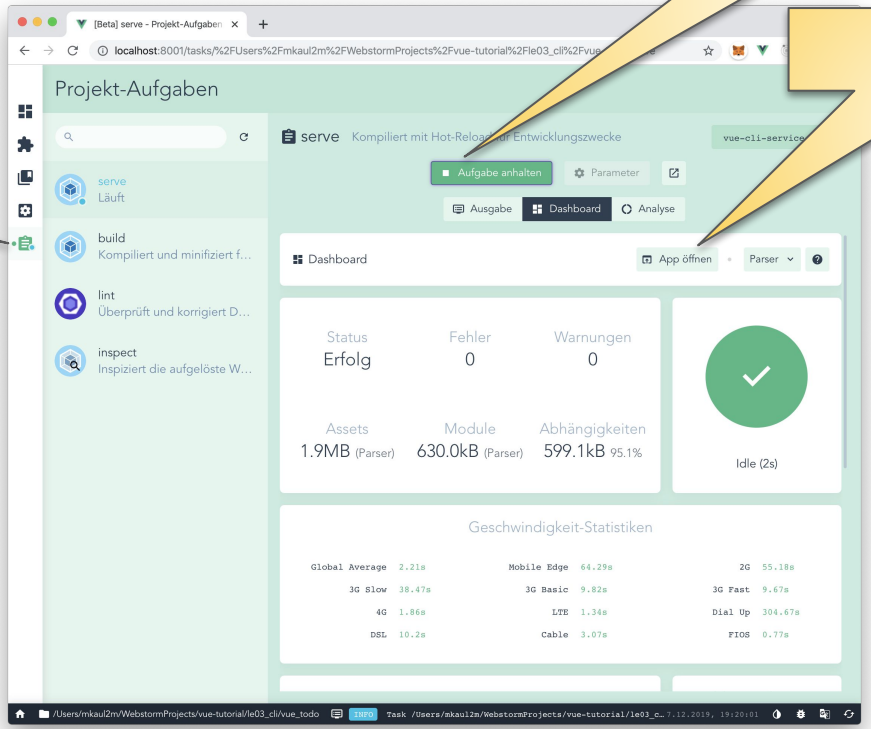
Aufgaben

2

Start Server

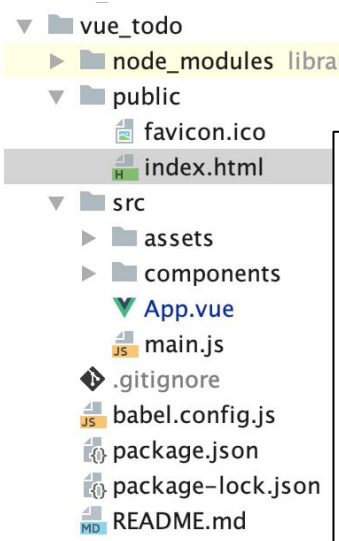
Öffne URL

3



https://youtu.be/Wy9q22isx3U?t=756

Generierte Default Vue Single Page Application (SPA)



index.html

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-widt
  <title>vue_todo</title>
 </head>
 <body>
  <noscript>
   <strong>We're sorry ...</strong>
  </noscript>
  <div id="app"></div>
  <!-- built files will be auto injected -->
 </body>
</html>
```

App.vue

```
<template>
 <div id="app">
  <img alt="Vue logo" src="./assets/logo.png">
  <HelloWorld msg="Welcome to Your Vue.js App"/>
 </div>
</template>
```

1

```
<script>
import HelloWorld from './components/HelloWorld.vue'
export default {
 name: 'app',
 components: {
  HelloWorld
 }
}
</script>
```

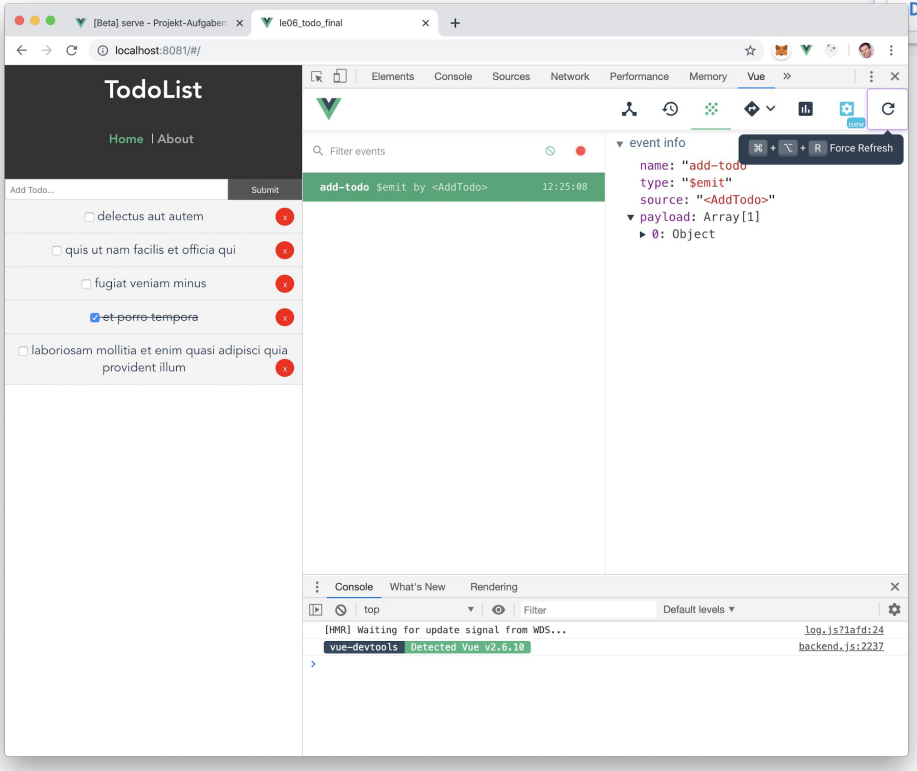
2

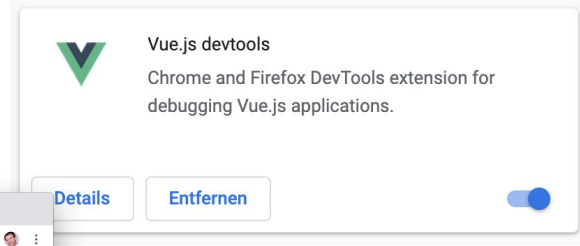
```
<style>
#app {
 font-family: 'Avenir', Helvetica, Arial, sans-serif;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
 text-align: center;
 color: #2c3e50;
 margin-top: 60px;
}
</style>
```

3

https://youtu.be/Wy9q22isx3U?t=756

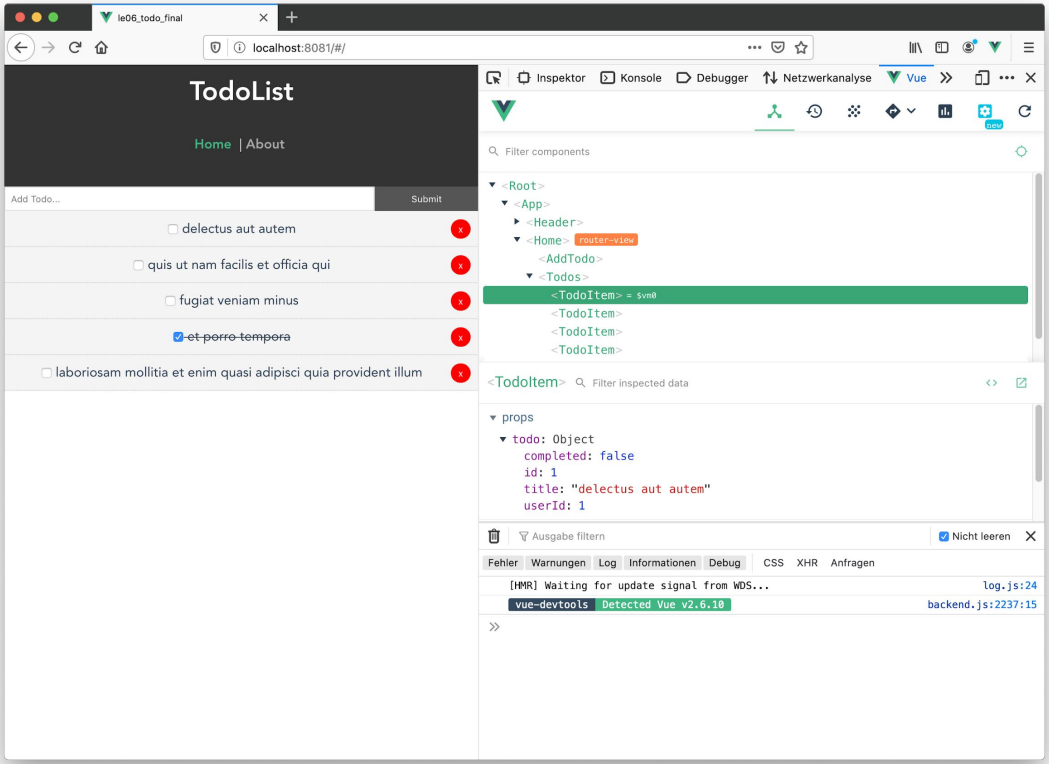
Chrome Extension Vue.js devtools

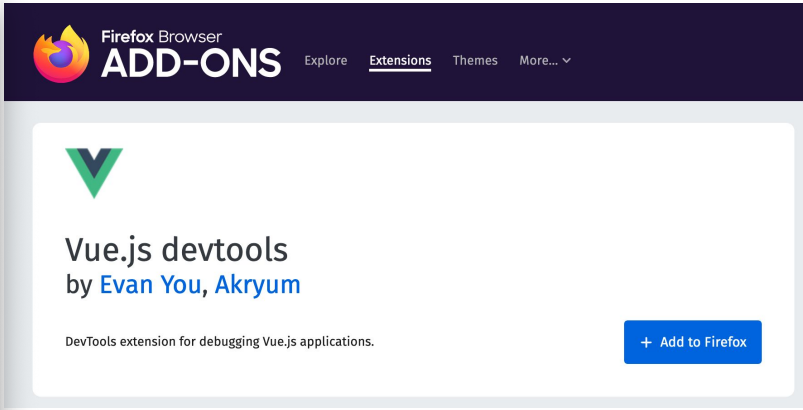




https://chrome.google.com/webstore/detail/vuejs-devtools/nhdogjmejiglipccpnnnanhbledajbpd

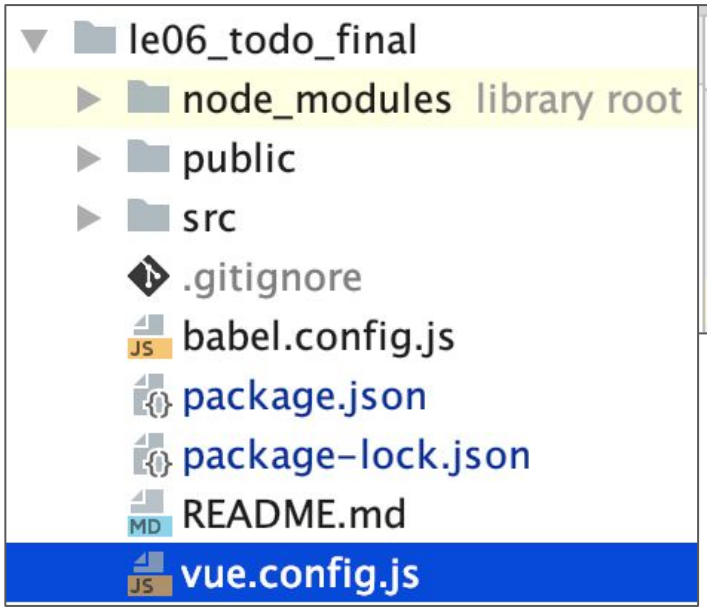
Firefox Extension Vue.js devtools





https://chrome.google.com/webstore/detail/vuejs-devtools/nhdogjmejiglipccpnnnanhbledajbpd

Debugging Vue.js



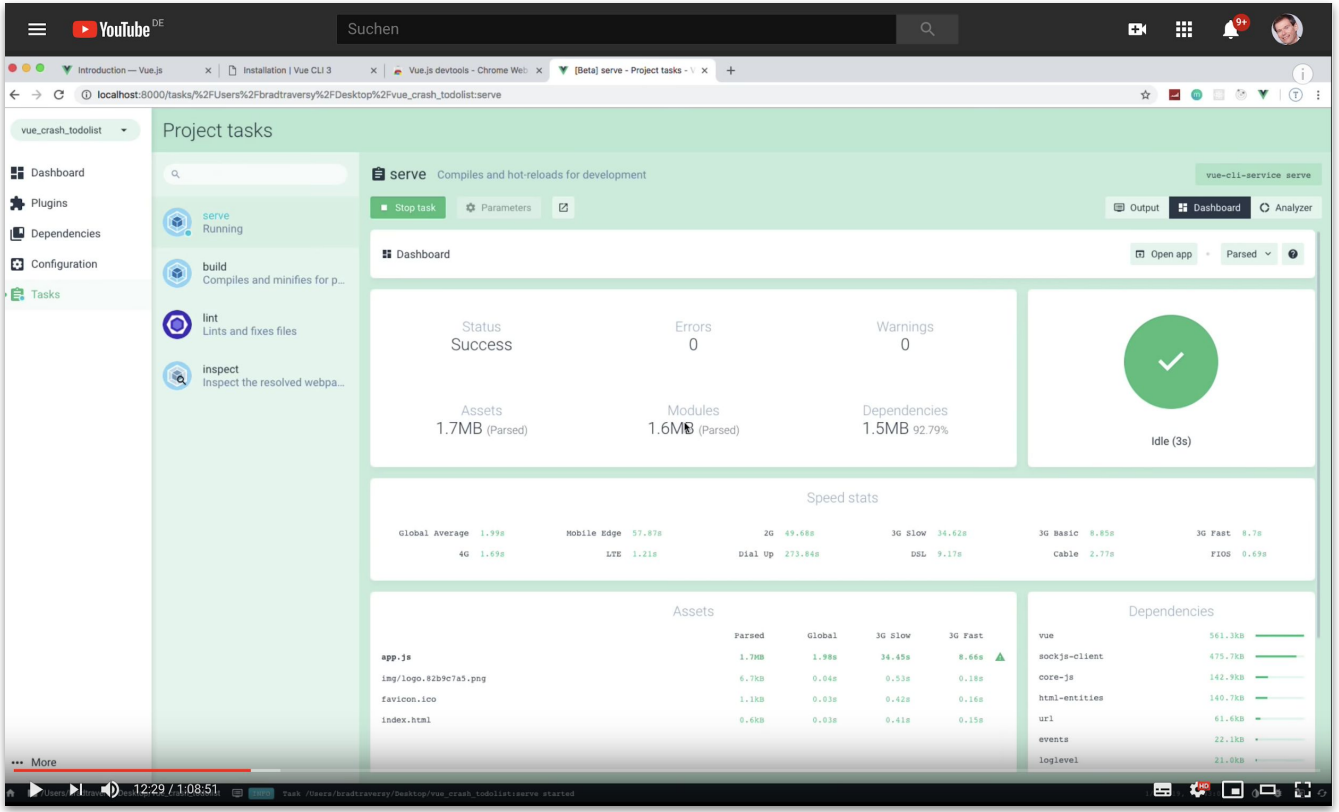
vue.config.js

```
1  module.exports = {
2    configureWebpack: {
3      devtool: 'source-map'
4    }
5  };
```

https://vuejs.org/v2/cookbook/debugging-in-vscode.html

https://youtu.be/lyGt1TmleoU?t=461

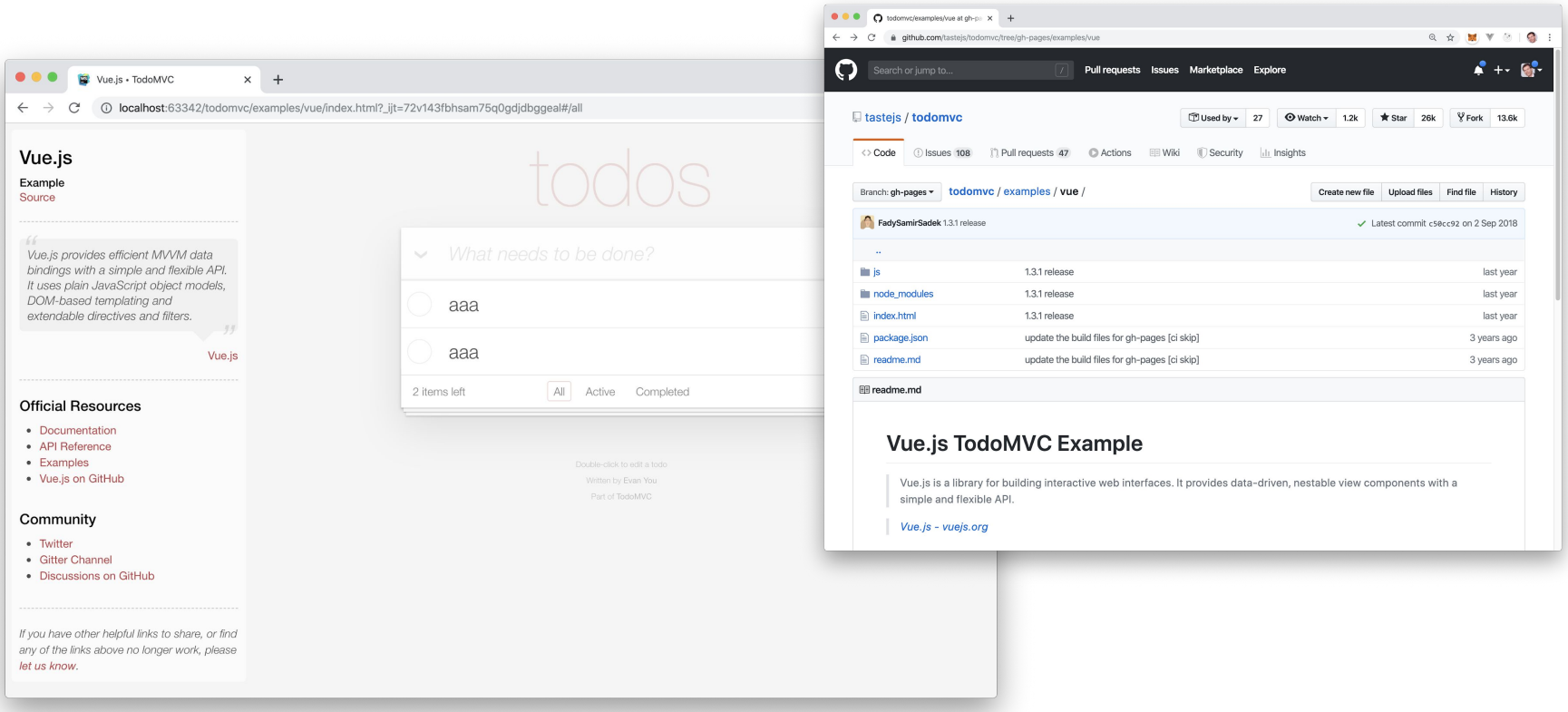
Tutorial zur Vue GUI auf Youtube



https://youtu.be/Wy9q22isx3U?t=756

Beispiel: ToDoMVC mit Vue.js

This is a fully spec-compliant TodoMVC implementation in under **120 effective lines of JavaScript** (excluding comments and blank lines).

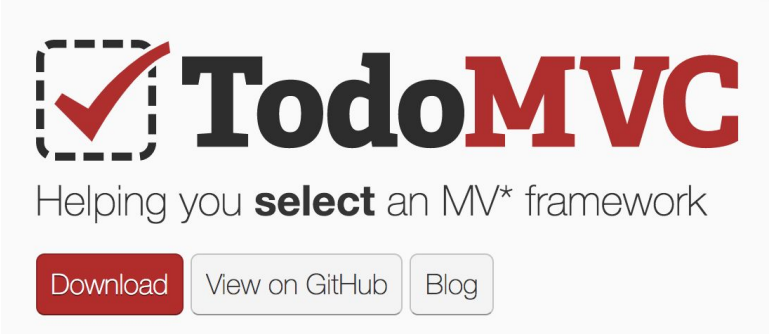


https://vuejs.org/v2/examples/todomvc.html

https://github.com/tastejs/todomvc/tree/gh-pages/examples/vue

Vergleich zwischen Frameworks

- ToDoMVC http://todomvc.com/



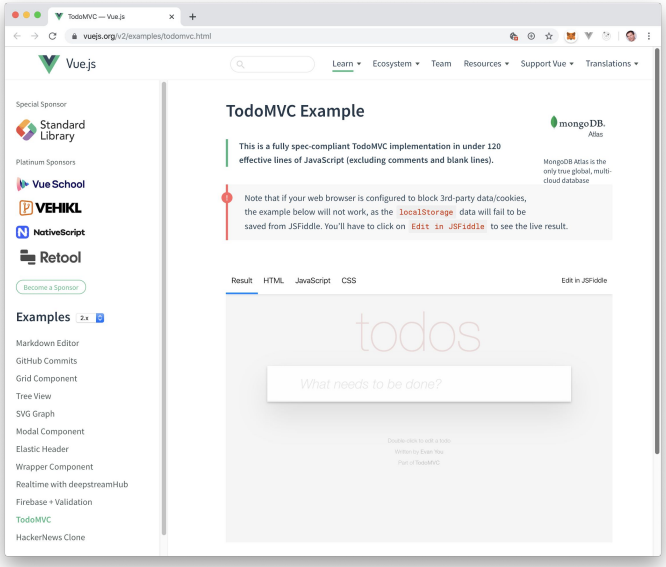
- HackerNews https://hnpwa.com/ a complete Progressive Web App

https://hnpwa.com/

Weitere Beispiele https://vuejs.org/v2/examples/todomvc.html

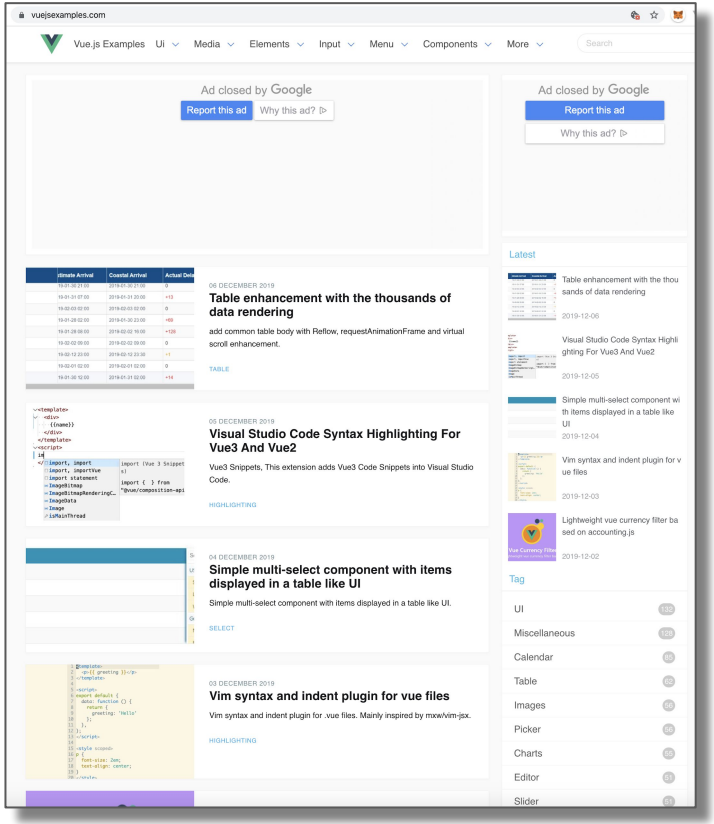
Examples

- Markdown Editor
- GitHub Commits
- Grid Component
- Tree View
- SVG Graph
- Modal Component
- Elastic Header
- Wrapper Component
- Realtime with deepstreamHub
- Firebase + Validation
- TodoMVC
- HackerNews Clone

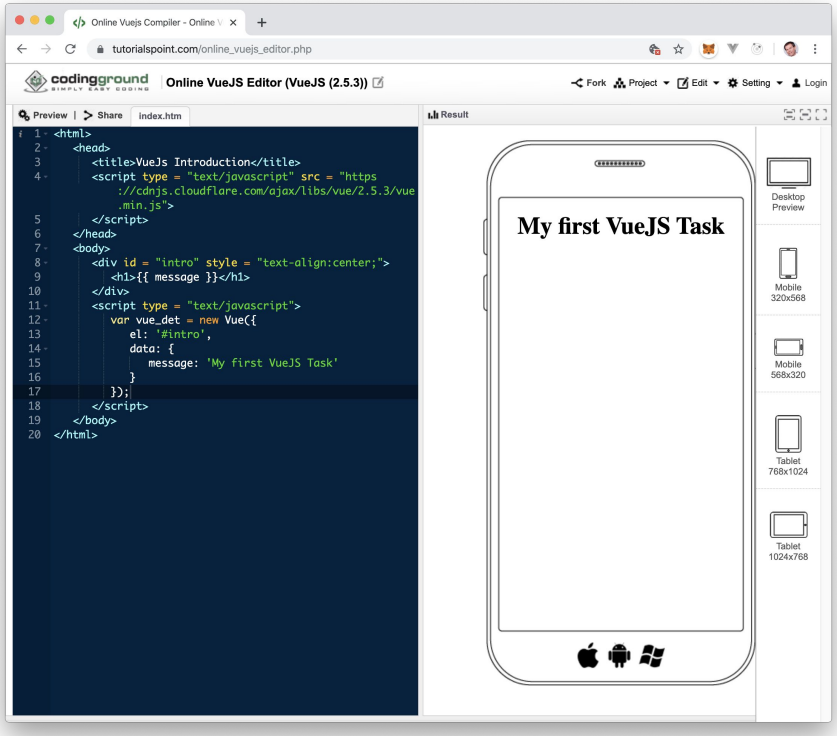


Vue.js Component Galleries

- Material Design Templates for Vue, Angular, React (Material UI)
  - **Material design templates for Vue**
    https://www.codeinwp.com/blog/material-design-templates-vue-angular-react/#Vue
- Interesting Vue UI Component Libraries
- Free and Premium VueJS Admin Templates Built With Bootstrap
- https://demos.creative-tim.com/vue-light-bootstrap-dashboard/#/admin/overview
- https://www.codeinwp.com/blog/best-vuejs-admin-templates-built-with-bootstrap/



Online VueJS Editor



https://www.tutorialspoint.com/online_vuejs_editor.php

The Progressive JavaScript Framework

WHY VUE.JS? GET STARTED GITHUB

vuejs.org/

Special Sponsor

Standard Library

Build APIs you need in minutes instead of days, for free.

Approachable

Already know HTML, CSS and JavaScript? Read the guide and start building things in no time!

Versatile

An incrementally adoptable ecosystem that scales between a library and a full-featured framework.

Performant

20KB min+gzip Runtime
Blazing Fast Virtual DOM
Minimal Optimization Efforts