**Reusability & Composition**

# - Mixins

**#**[**Basics**](https://vuejs.org/v2/guide/mixins.html#Basics)

Екі компонентте де қайталанатып операцияларды бір файлға салып оны mixen арқылы беруге болады

// mixen.js

export const myMixin = {

  created() {

    this.hello();

  },

  methods: {

    hello() {

      console.log("привет из примеси!");

    },

  },

};

// compoennt-1

import { myMixin } from "@/mixen/mixen.js";

export default {

  mixins: [myMixin],

};

// component-2

import { myMixin } from "@/mixen/mixen.js";

export default {

  mixins: [myMixin],

};

**#**[**Option Merging**](https://vuejs.org/v2/guide/mixins.html#Option-Merging)

mixen-ге жазылған данныйларды да шақыруға болады

компоненттерде қайталанған **переменныйлар** мен **функцияларды** ды mixen-ге жазып оны кез-келген компонентте шақыруға болады

// mixen.js

export const myMixin = {

  data() {

    return { message: "hello" };

  },

  created() {

    this.hello();

  },

  methods: {

    hello() {

      console.log("привет из примеси!");

    },

  },

};

<template>

  <div>

    <h2>icon</h2>

    {{ message }} {{ hello() }}

  </div>

</template>

<script>

// component-1

import { myMixin } from "@/mixen/mixen.js";

export default {

  mixins: [myMixin],

};

**#**[**Global Mixin**](https://vuejs.org/v2/guide/mixins.html#Global-Mixin)

mixen-ді глобалный main.js-ке жазып кез-келген компоненттен шақыруға болады

// main.js

Vue.mixin({

  methods: {

    hellow() {

      console.log("привет из примеси!2222");

    },

  },

});

// component-2

<template>

  <div>

    <h2>Click Btn</h2>

    {{ hellow() }}

  </div>

</template>

**#**[**Custom Option Merge Strategies**](https://vuejs.org/v2/guide/mixins.html#Custom-Option-Merge-Strategies) **???**

**Теңшелетін біріктіру опциялары стратегиясы**

Теңшелетін опцияларды біріктіру кезінде әдепкі стратегия қолданылады, ол жай ғана бір мәнді басқасымен ауыстырады. Теңшелетін опцияларды біріктіру кезінде жеке логиканы пайдаланғыңыз келсе, функцияны Vue.config.optionMergeStrategies ішіне қосыңыз:

Vue.config.optionMergeStrategies.myOption = function (toVal, fromVal) {

// return mergedVal

}

Нысан опцияларының көпшілігі үшін әдепкі стратегияны қолдануға болады, methods опциясы

var strategies = Vue.config.optionMergeStrategies

strategies.myOption = strategies.methods

Неғұрлым күрделі мысал vuex 1 біріктіру стратегиясы болуы мүмкін.x:

const merge = Vue.config.optionMergeStrategies.computed

Vue.config.optionMergeStrategies.vuex = function (toVal, fromVal) {

if (!toVal) return fromVal

if (!fromVal) return toVal

return {

getters: merge(toVal.getters, fromVal.getters),

state: merge(toVal.state, fromVal.state),

actions: merge(toVal.actions, fromVal.actions)

}

}

# - Custom Directives

v-model and v-show сияқты жаңадан өзің атрибуттар жасау

[**Intro**](https://vuejs.org/v2/guide/custom-directive.html#Intro)

"example"-деп аталатын жаңа атрибутты қолдану

<input v-example />

    <h2 v-example>Page</h2>

 // "example"-деп аталатын жаңа атрибут жасау

directives: {

    example: {

      inserted(el) {

        // el.focus();

        el.style.color = "red";

        console.log(el);

      },

    },

  },

// main.js

глобалный файлда "example"-деп аталатын жаңа атрибут жасау

Vue.directive("example", {

  inserted(el) {

    el.style.color = "red";

  },

});

[**Hook Functions**](https://vuejs.org/v2/guide/custom-directive.html#Hook-Functions)inserted - сияқты басқада функциялар бар

* bind: Директиваны элементке бірінші байланыстырғанда, бір рет шақырылады. Мұнда инициализация кодын қоюға болады.
* **inserted:** тектік элементтің ішіне қатысты элементті кірістіргеннен кейін шақырылады (ата-ананың өзі осы нүктеде әлі негізгі элемент ағашына тиесілі болмауы мүмкін екенін ескеріңіз).
* update: вызывается после обновления VNode компонента-контейнера, **но, возможно, до обновления дочерних элементов**.
* componentUpdated: вызывается после обновления как VNode компонента-контейнера, **так и VNode его потомков**.
* unbind: вызывается однократно, при отвязывании директивы от элемента.

[**Directive Hook Arguments**](https://vuejs.org/v2/guide/custom-directive.html#Directive-Hook-Arguments)

* el: -тен басқа да переменныйлерді қолдануға болады
* binding: Объект, содержащий следующие свойства:
  + name: Название директивы, без указания префикса v-.
  + value: Значение, переданное в директиву. Например, для v-my-directive="1 + 1" значением будет 2.
  + oldValue: Предыдущее переданное в директиву значение. Доступно только для хуков update и componentUpdated, и передаётся независимо от того, произошло ли в действительности его изменение.
  + expression: Выражение-строка, переданное в директиву. Например, для v-my-directive="1 + 1" это будет "1 + 1".
  + arg: Аргумент, переданный в директиву, в случае его наличия. Например, для v-my-directive:foo это будет "foo".
  + modifiers: Объект, содержащий модификаторы, если они есть. Например, для v-my-directive.foo.bar, объектом модификаторов будет { foo: true, bar: true }.
* vnode: Виртуальный элемент, созданный компилятором Vue. См. **[VNode API](https://ru.vuejs.org/v2/api/" \l "%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81-VNode)** для подробностей.
* oldVnode: Предыдущий виртуальный элемент, доступный только для хуков update и componentUpdated.

**Пример**

<div id="hook-arguments-example" v-demo:foo.a.b="message"></div>

Vue.directive('demo', {

bind: function (el, binding, vnode) {

var s = JSON.stringify

el.innerHTML =

'name: ' + s(binding.name) + '<br>' +

'value: ' + s(binding.value) + '<br>' +

'expression: ' + s(binding.expression) + '<br>' +

'argument: ' + s(binding.arg) + '<br>' +

'modifiers: ' + s(binding.modifiers) + '<br>' +

'vnode keys: ' + Object.keys(vnode).join(', ')

}

})

new Vue({

el: '#hook-arguments-example',

data: {

message: 'привет!'

}

})

**name:"demo"**

**value:"привет"**

**expression: "message"**

**argument:"foo"**

**...**

**#**[**Dynamic Directive Arguments**](https://vuejs.org/v2/guide/custom-directive.html#Dynamic-Directive-Arguments)

кейбір аргументтер динамикалық түрде(яғни ауыспалы) берілуі мүмкін, оныда алуға болады

<div id="dynamicexample">

<h3>Прокрутите страницу вниз</h3>

<p v-pin:[direction]="200">Элемент зафиксирован в 200px слева страницы.</p>

</div>

Vue.directive('pin', {

bind: function (el, binding, vnode) {

el.style.position = 'fixed'

var s = (binding.arg == 'left' ? 'left' : 'top')

el.style[s] = binding.value + 'px'

}

})

new Vue({

el: '#dynamicexample',

data: function () {

return {

direction: 'left'

}

}

})

[**Function Shorthand**](https://vuejs.org/v2/guide/custom-directive.html#Function-Shorthand)

кейде сокреченный түрде жазуға болады

Vue.directive("color-switch", function(el, binding) {

  el.style.backgroundColor = binding.value;

});

<template>

  <div id="app">

    <h2 v-color-switch="red">Page</h2>

  </div>

</template>

<script>

import SlotPage from "./view/SlotPage.vue";

export default {

  data() {

    return { red: "red" };

  },

[**Object Literals**](https://vuejs.org/v2/guide/custom-directive.html#Object-Literals)

binding -переменныйын қолданған кезде данный ларды обьект түрінде де беруге болады

<div v-demo="{ color: 'белый', text: 'привет!' }"></div>

Vue.directive('demo', function (el, binding) {

console.log(binding.value.color) // => "белый"

console.log(binding.value.text) // => "привет!"

})

# - Render Functions & JSX

## [Basics](https://vuejs.org/v2/guide/render-function.html#Basics)

**render-функции** — более низкоуровневую альтернативу шаблонам.

<script type="text/x-template" id="anchored-heading-template">

<h1 v-if="level === 1">

<slot></slot>

</h1>

<h2 v-if="level === 2">

<slot></slot>

</h2>

<h3 v-if="level === 3">

<slot></slot>

</h3>

<h4 v-if="level === 4">

<slot></slot>

</h4>

<h5 v-if="level === 5">

<slot></slot>

</h5>

<h6 v-if="level === 6">

<slot></slot>

</h6>

</script>

Осылай ұзақ жазфп жүрудің орнына **render-функции**  қолданып қысқа жазуға болады

Мысалы

// main.js

Vue.component("anchored-heading", {

  render: function(createElement) {

    return createElement(

      "h" + this.level, // имя тега

      this.$slots.default // массив дочерних элементов

    );

  },

  props: {

    level: {

      type: Number,

      required: true,

    },

  },

});

// компонент

  <anchored-heading :level="3">Привет, мир!</anchored-heading>

## [Узлы, деревья, и виртуальный DOM](https://ru.vuejs.org/v2/guide/render-function.html#%D0%A3%D0%B7%D0%BB%D1%8B-%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D1%8C%D1%8F-%D0%B8-%D0%B2%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9-DOM)

Vue.component("anchored-heading", {

  render: function(createElement) {

    return createElement(

      "h" + this.level, // имя тега

      "this.$slots.default" // бұл жерде тег ішіне жазғың келген нәрсені белгілеуге болады

    );

  },

### #[Виртуальный DOM](https://ru.vuejs.org/v2/guide/render-function.html#%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9-DOM)

### бұл жерде createElement орнына createNodeDescription - ты қолдануға болатындағы көрсетілген

Vue.component("anchored-heading", {

  render: function(createNodeDescription) {

    return createNodeDescription(

      "h" + this.level, // имя тега

      "this.$slots.default" // массив дочерних элементов

    );

  },

## [Аргументы createElement](https://ru.vuejs.org/v2/guide/render-function.html#%D0%90%D1%80%D0%B3%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D1%8B-createElement)

// @returns {VNode}

createElement( 'div', {},[

'Какой-то текст прежде всех остальных элементов.',

createElement('h1', 'Заголовок'),

createElement(MyComponent, {

props: {

someProp: 'foobar'

}

})

]

)

### #[Подробно об объекте данных](https://ru.vuejs.org/v2/guide/render-function.html#%D0%9F%D0%BE%D0%B4%D1%80%D0%BE%D0%B1%D0%BD%D0%BE-%D0%BE%D0%B1-%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%B5-%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)

### createElement - яғни элемент создать ететін кезде осыларды қолдануға болады

{

class: {

foo: true,

bar: false

},

style: {

color: 'red',

fontSize: '14px'

},

attrs: {

id: 'foo'

},

props: {

myProp: 'bar'

},

domProps: {

innerHTML: 'baz'

},

on: {

click: this.clickHandler

},

nativeOn: {

click: this.nativeClickHandler

},

directives: [

{

name: 'my-custom-directive',

value: '2',

expression: '1 + 1',

arg: 'foo',

modifiers: {

bar: true

}

}

],

scopedSlots: {

default: props => createElement('span', props.text)

},

slot: 'name-of-slot',

key: 'myKey',

ref: 'myRef',

refInFor: true

### }

### #[Полный пример](https://ru.vuejs.org/v2/guide/render-function.html#%D0%9F%D0%BE%D0%BB%D0%BD%D1%8B%D0%B9-%D0%BF%D1%80%D0%B8%D0%BC%D0%B5%D1%80)

    <anchored-heading :level="3">Привет, мир!</anchored-heading>

// main.js

Vue.component("anchored-heading", {

  render: function(createElement) {

    return createElement(

      "h" + this.level,

      {

        attrs: {

          name: "headingId",

          href: "#" + "headingId",

        },

        class: {

          foo: true,

          bar: false,

        },

        style: {

          color: "red",

          fontSize: "14px",

        },

      },

      "this.$slots.default"

    );

  },

### # [Ограничения](https://ru.vuejs.org/v2/guide/render-function.html#%D0%9E%D0%B3%D1%80%D0%B0%D0%BD%D0%B8%D1%87%D0%B5%D0%BD%D0%B8%D1%8F)

### Уникальный қылып жазуғада болады, Примерлер:

    <anchored />

// main.js

Vue.component("anchored", {

  render: function(createElement) {

    var myParagraphVNode = createElement("p", "hi");

    return createElement("div", [

      myParagraphVNode,

      myParagraphVNode,

    ]);

  },

});

**<div data-v-7ba5bd90="">**

**<p>hi</p>**

**<p>hi</p>**

**</div>**

    <anchored />

// main.js

Vue.component("anchored", {

  render: function(createElement) {

    return createElement(

      "div",

      Array.apply(null, { length: 20 }).map(function() {

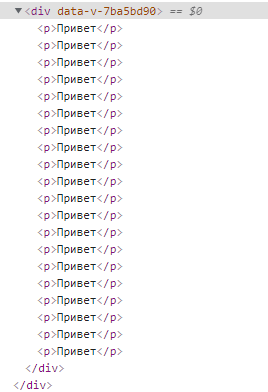
        return createElement("p", "Привет");

      })

    );

  },

});

****

## [Реализация возможностей шаблона с помощью JavaScript](https://ru.vuejs.org/v2/guide/render-function.html#%D0%A0%D0%B5%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F-%D0%B2%D0%BE%D0%B7%D0%BC%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D0%B5%D0%B9-%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D0%B0-%D1%81-%D0%BF%D0%BE%D0%BC%D0%BE%D1%89%D1%8C%D1%8E-JavaScript)

### # [v-if и v-for](https://ru.vuejs.org/v2/guide/render-function.html#v-if-%D0%B8-v-for)

**render , createElement -** ті цикл мен шарт ретінде істеу үшін

<anchored :items="items" />

  data() {

    return { items: [(name = "1"), (name = "45")] };

  },

// main.js

Vue.component("anchored", {

  props: ["items"],

  render: function(createElement) {

    if (this.items.length) {

      return createElement(

        "ul",

        this.items.map(function(item) {

          return createElement("li", item);

        })

      );

    } else {

      return createElement("p", "Ничего не найдено.");

    }

  },

});

**#** [**v-model**](https://ru.vuejs.org/v2/guide/render-function.html#v-model)

input-тағы данныйды алу жолы , render создать етуде мүндай бойынша дайын тәсіл жоқ, қолдан жасау керек

Мысал:

   <anchored :value="value" @input="inputF($event)" />

  data() {

    return { value: "asems" };

  },

  methods: {

    inputF(n) {

      console.log(n);

    },

  },

// main.js

Vue.component("anchored", {

  props: ["value"],

  render: function(createElement) {

    var self = this;

    return createElement("input", {

      domProps: {

        value: self.value,

      },

      on: {

        input: function(event) {

          self.$emit("input", event.target.value);

        },

      },

    });

  },

});

### #[События и модификаторы клавиш](https://ru.vuejs.org/v2/guide/render-function.html#%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D1%8F-%D0%B8-%D0%BC%D0%BE%D0%B4%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%82%D0%BE%D1%80%D1%8B-%D0%BA%D0%BB%D0%B0%D0%B2%D0%B8%D1%88)

### createElement арқылы түрлі позицияларды, істеу болады

Vue.component("buttons", {

  render: function(createElement) {

    return createElement(

      "button",

      {

        on: {

          click(event) {

            console.log(event);

          },

        },

      },

      "Click"

    );

  },

});

### 

### Vue on-мен бірге қолдануға болатын префикстерді ұсынады

| **Модификаторы** | **Префикс** |
| --- | --- |
| .passive | & |
| .capture | ! |
| .once | ~ |
| .capture.once или .once.capture | ~! |

Например:

on: {

'!click': this.doThisInCapturingMode,

'~keyup': this.doThisOnce,

'~!mouseover': this.doThisOnceInCapturingMode

}

   on: {

          click(event) {

            console.log(event);

          },

          "!click": () => {

            console.log("click");

          },

          "mouseover": () => {

            console.log("keyup");

          },

        },

### 

### [Слоты](https://ru.vuejs.org/v2/guide/render-function.html#%D0%A1%D0%BB%D0%BE%D1%82%D1%8B)

render: function (createElement) {

// `<div><slot></slot></div>` колдан жасамақшы болған элементтіңнің ішіне осылай жасағың келсе,

// this.$slots.default осылай істейсің,

return createElement('div', this.$slots.default)

}

[**this.$scopedSlots**](https://ru.vuejs.org/v2/api/#vm-scopedSlots): істеу үшін (**[this.$scopedSlots](https://ru.vuejs.org/v2/api/" \l "vm-scopedSlots)**:- тың не екенін slot бөлімінен қараңыз)

props: ['message'],

render: function (createElement) {

// `<div><child v-slot="props"><span>{{ props.text }}</span></child></div>`

return createElement('div', [

createElement('child', {

// передаём `scopedSlots` в объект data

// в виде { name: props => VNode | Array<VNode> }

scopedSlots: {

default: function (props) {

return createElement('span', props.text)

}

}

})

])

}

## [JSX](https://ru.vuejs.org/v2/guide/render-function.html#JSX)

render функциясын createElement арқылы беру, бұл тәсіл ұзақ әрі түсініксіз болуы мүмкін..

    <anchored />

Vue.component("anchored", {

  render: function(createElement) {

    return createElement(

      "anchored-heading",

      {

        props: {

          level: 1,

        },

      },

      [createElement("span", "Привет"), ", мир!"]

    );

  },

});

render функциясын JSX -тәсілімен беру

    <anchored />

Vue.component("anchored", {

  render() {

    return (

      <AnchoredHeading level={1}>

        <span>Привет, мир!</span>

      </AnchoredHeading>

    );

  },

});

## [Функциональные компоненты](https://ru.vuejs.org/v2/guide/render-function.html#%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5-%D0%BA%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D1%8B)

functional: true, - АРҚЫЛЫ КОМПОНЕНТ БЕРУ , бұл жағдайда render( яғни создать етілсе) берілсе, данныйлар тек қана context арқылы беріледі

Бұл тәсіл крмпонентке данны й берместен бұрын басқаруға мүмкіндік береді

Кейінгі көрсетулер үшін компоненттерді таңдауға мүмкіндік береді

render: function (createElement, context) {

// ...

}

functional: true, орнына , былай жазуға болады

<template functional>

</template>

**Пример**: тек context арқылы берілу бойынша

Vue.component("my-functional-button", {

  functional: true,

  render: function(createElement, context) {

    return createElement("button", context.props.contexts.data);

  },

});

    <my-functional-button :contexts="contexts" />

  data() {

    return {

      contexts: {

        data: "asems",

        children: "asemsasems",

      },

    };

  },

context арқылы берілу бойынша context- тен кейін жазылатын атрибуттар

* props: Объект со всеми переданными входными параметрами
* children: Массив дочерних виртуальных узлов
* slots: Функция, возвращающая объект со слотами
* scopedSlots: (2.6.0+) Объект, содержащий все переданные слоты с ограниченной областью видимости. Также предоставляет доступ к обычным слотам в качестве функций
* data: [**Объект данных**](https://ru.vuejs.org/v2/guide/render-function.html#%D0%9F%D0%BE%D0%B4%D1%80%D0%BE%D0%B1%D0%BD%D0%BE-%D0%BE%D0%B1-%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%B5-%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) целиком, переданный объекту вторым аргументом createElement
* parent: Ссылка на родительский компонент
* listeners: (2.3.0+) Объект, содержащий все зарегистрированные в родителе прослушиватели событий. Это просто псевдоним для data.on
* injections: (2.3.0+) Если используется опция **[inject](https://ru.vuejs.org/v2/api/" \l "provide-inject)**, будет содержать все разрешённые инъекции.

### # [Передача атрибутов и событий дочерним элементам/компонентам](https://ru.vuejs.org/v2/guide/render-function.html#%D0%9F%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B0-%D0%B0%D1%82%D1%80%D0%B8%D0%B1%D1%83%D1%82%D0%BE%D0%B2-%D0%B8-%D1%81%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9-%D0%B4%D0%BE%D1%87%D0%B5%D1%80%D0%BD%D0%B8%D0%BC-%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D0%)

Vue.component('my-functional-button', {

functional: true,

render: function (createElement, context) {

// Явная передача любых атрибутов, слушателей событий, дочерних элементов и т.д.

return createElement('button', context.data, context.children)

}

### })

context.data - Түрлі данныйларды беруге болады

context.children - Компонент арасындағы, жазуларды көрсету үшін slot орнына қолданады

егер осы тәсілмен қолданатын болсағыз data.attrs түрінде қолданылады

<template functional>

<button

class="btn btn-primary"

v-bind="data.attrs"

v-on="listeners"

>

<slot/>

</button>

</template>

### # [slots() vs children](https://ru.vuejs.org/v2/guide/render-function.html#slots-vs-children)

### children орнына slots() - қолдануға болады

### егер slot-тың аты болса мысалы "foo" деген оны context.slots().foo қолданылады

### егер болмаса context.slots().default қолданылады

    <my-functional-button>

      <template v-slot:foo>

        <h1>Здесь мог быть заголовок страницы</h1>

      </template>

      <p>второй</p>

    </my-functional-button>

Vue.component("my-functional-button", {

  functional: true,

  render: function(createElement, context) {

 //return createElement("button",context.data,context.slots().foo);

 return createElement("button", context.data,context.slots().default);

  },

});

## [Компиляция шаблонов](https://ru.vuejs.org/v2/guide/render-function.html#%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%86%D0%B8%D1%8F-%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D0%BE%D0%B2)

**render -** функциясының қалай жұмыс істейтіні көрсетілген, негізі оны білудің қажеті жоқ

# - Плагины

**плагин түрлері, плагиндерді осындай мақсаттарда істеуге болады**

* Добавляют глобальные методы и/или свойства,
* Добавляют глобальные объекты: директивы/фильтры/переходы и т.д.
* Добавляют опции компонентам через глобальную примесь, например **[vue-router](https://github.com/vuejs/vue-router" \t "_blank)**
* Добавляют методы экземпляра Vue через Vue.prototype.
* Библиотеки, предоставляющие собственные API

## [Использование плагинов](https://ru.vuejs.org/v2/guide/plugins.html#%D0%98%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D0%BF%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD%D0%BE%D0%B2)

// вызывает `MyPlugin.install(Vue)`

Vue.use(MyPlugin)

new Vue({

//... настройки

})

Бірнеше плпгинді бірден шақыруға болмайды, тек бір плагин шақырылады

Vue.use(MyPlugin, { someOption: true })

Плогинге қосымша параметр беруг еболады

router - сияқты Vue де тұрақты орнатылған плагиндерді Vue.use(MyPlugin) осылай шақырудың керегі жоқ. ал қолдан істелген плагиндерді

var VueRouter = require('vue-router')

// Не забудьте этот вызов!

Vue.use(VueRouter)

осылай шақырылуы керек

## [Создание плагина](https://ru.vuejs.org/v2/guide/plugins.html#%D0%A1%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-%D0%BF%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD%D0%B0)

MyPlugin.install = function(Vue, options) {

  // 1. добавление глобального метода или свойства (өзің қалаған (router-сияқты) тәсілдерді жасап шығару)

  Vue.myGlobalMethod = function() {

    // некоторая логика ...

  };

  // 2. добавление глобального объекта  (сustom Directives - қолдану)

  Vue.directive("my-directive", {

    bind(el, binding, vnode, oldVnode) {

      // некоторая логика ...

    },

    // ...

  });

  // 3. добавление опций компонентов (mixin - қолдану)

  Vue.mixin({

    created: function() {

      // некоторая логика ...

    },

    // ...

  });

  // 4. добавление метода экземпляра   (Vue.prototype - қолдану)

  Vue.prototype.$myMethod = function(methodOptions) {

    // Vue.prototype.$api = api;

    //$api-орнында $myMethod орналасқан мұны переменный ретінде қолданалады

    //api-орнында керекті орындалатын функция қолданылады

    // некоторая логика ...

  };

};

# - Фильтры

2-тәсілде қолданылады

<!-- mustache -->

{{ message | capitalize }}

<!-- v-bind -->

<div v-bind:id="rawId | formatId"></div>

2-тәсілде жасалады

Локальный

filters: {

capitalize: function (value) {

if (!value) return ''

value = value.toString()

return value.charAt(0).toUpperCase() + value.slice(1)

}

}

Глобальный

Vue.filter('capitalize', function (value) {

if (!value) return ''

value = value.toString()

return value.charAt(0).toUpperCase() + value.slice(1)

})

new Vue({

// ...

})

Локальный мен Глобальный дың аттары бірдец болып қалса Локальный күшке енеді

{{ message | filterA | filterB }}

Бірінші message мәнін алу арқылы filterA орындалады, сосын filterA мәнін алу арқылы filterB орындалады

фильтрды функция арқылы қолдануға болады, онда переменный беруге болады

{{ message | filterA('arg1', arg2) }}

Мұнда функцияға берілген переменныйлардың біріншісі message, екіншісі переменный 'arg1', үшінш і arg2