**Регистрация компонентов**

[Именование компонентов](https://ru.vuejs.org/v2/guide/components-registration.html#%D0%98%D0%BC%D0%B5%D0%BD%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D0%BA%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D0%BE%D0%B2)

Vue.component('my-component-name', { /\* ... \*/ })

Vue.component('MyComponentName', { /\* ... \*/ })

[Глобальная регистрация](https://ru.vuejs.org/v2/guide/components-registration.html#%D0%93%D0%BB%D0%BE%D0%B1%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F-%D1%80%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80%D0%B0%D1%86%D0%B8%D1%8F)

Vue.component('component-a', { /\* ... \*/ })

Vue.component('component-b', { /\* ... \*/ })

<div id="app">

<component-a></component-a>

<component-b></component-b>

</div>

[**Локальная регистрация**](https://ru.vuejs.org/v2/guide/components-registration.html#%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F-%D1%80%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80%D0%B0%D1%86%D0%B8%D1%8F)

var ComponentB = { /\* ... \*/ }

var ComponentC = { /\* ... \*/ }

new Vue({

el: '#app',

components: {

'component-a': ComponentA,

'component-b': ComponentB

}

})

Что локально компоненты **не будут доступны** в дочерних компонентах.

[**Модульные системы**](https://ru.vuejs.org/v2/guide/components-registration.html#%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5-%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B)

Глобальный компоненттер **Base, App или V. сөздерімен басталуы дұрыс, шатаспау үшін**

Пример:

BaseButton

BaseIcon

BaseInput

[Локальная регистрация в модульной системе](https://ru.vuejs.org/v2/guide/components-registration.html#%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F-%D1%80%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80%D0%B0%D1%86%D0%B8%D1%8F-%D0%B2-%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%D0%BD%D0%BE%D0%B9-%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B5)

Әрбір компонент жеке файлда тұрса локальный кезде жеке жеке импорт жасайды

import ComponentA from './ComponentA'

import ComponentC from './ComponentC'

export default {

components: {

ComponentA,

ComponentC

},

// ...

}

[Автоматическая глобальная регистрация базовых компонентов](https://ru.vuejs.org/v2/guide/components-registration.html#%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F-%D0%B3%D0%BB%D0%BE%D0%B1%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F-%D1%80%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80%D0%B0%D1%86%D0%B8%D1%8F-%D0%B1%D0%B0%D0%B7%D0%BE%D0%B2%D1%8B%D1%85)

import BaseButton from './components/BaseButton.vue'

import BaseIcon from './components/BaseIcon.vue'

import BaseInput from './components/BaseInput.vue'

export default {

components: {

BaseButton,

BaseIcon,

BaseInput

}

}

**Осыларды бірден импорт жасау үшін gloval.js пайдаланылады**

import Vue from "vue"; // './components/gloval.js'

//Base деп басталатын компоненттердің қасында тұратын js файл осы компоненттерді бірден глобальный импорт жасау үшін керек

const requireComponent = require.context(".", false, /\_Base-[\w-]+\.vue$/);

requireComponent.keys().forEach((fileName) => {

  const componentConfig = requireComponent(fileName);

  const componentName = fileName

    .replace(/^\.\/\_/, "")

    .replace(/\.\w+$/, "")

    .split("-")

    .map((kebab) => kebab.charAt(0).toUpperCase() + kebab.slice(1))

    .join("");

  Vue.component(componentName, componentConfig.default || componentConfig);

});

# Входные параметры

## [Prop Casing (camelCase vs kebab-case)](https://vuejs.org/v2/guide/components-props.html#Prop-Casing-camelCase-vs-kebab-case)

Vue.component('blog-post', {

// - postTitle осылай Name беру camelCase деп аталады

props: ['postTitle'],

template: '<h3>{{ postTitle }}</h3>'

# })

// - post-title осылай Name беру kebab-case деп аталады

# <blog-post post-title="hello!"></blog-post>

## [Prop Types](https://vuejs.org/v2/guide/components-props.html#Prop-Types)

# props: ['title', 'likes', 'isPublished', 'commentIds', 'author']

# props -ты массив емес обьект түрінде шақырған дұрысрақ

props: {

title: String,

likes: Number,

isPublished: Boolean,

commentIds: Array,

author: Object,

callback: Function,

contactsPromise: Promise // или любой другой конструктор

# }

## [Passing Static or Dynamic Props](https://vuejs.org/v2/guide/components-props.html#Passing-Static-or-Dynamic-Props)

# <blog-post v-bind:title="post.title"></blog-post>

### #[Passing a Number](https://vuejs.org/v2/guide/components-props.html#Passing-a-Number)

# <blog-post v-bind:likes="42"></blog-post>

### #[Passing a Boolean](https://vuejs.org/v2/guide/components-props.html#Passing-a-Boolean)

# <blog-post v-bind:is-published="false"></blog-post>

### #[Passing an Array](https://vuejs.org/v2/guide/components-props.html#Passing-an-Array)

# <blog-post v-bind:comment-ids="[234, 266, 273]"></blog-post>

### #[Passing an Object](https://vuejs.org/v2/guide/components-props.html#Passing-an-Object)

<blog-post

v-bind:author="{

name: 'Veronica',

company: 'Veridian Dynamics'

}"

# ></blog-post>

# // <blog-post v-bind:author="post.author"></blog-post>

### #[Passing the Properties of an Object](https://vuejs.org/v2/guide/components-props.html#Passing-the-Properties-of-an-Object)

post: {

id: 1,

title: 'My Journey with Vue'

# }

<blog-post v-bind="post"></blog-post>

<blog-post

v-bind:id="post.id"

v-bind:title="post.title"

# > </blog-post>

## [One-Way Data Flow](https://vuejs.org/v2/guide/components-props.html#One-Way-Data-Flow)

props: ['initialCounter'],

data: function () {

return {

counter: this.initialCounter

}

# }

props: ['size'],

computed: {

normalizedSize: function () {

return this.size.trim().toLowerCase()

}

# }

## [Prop Validation](https://vuejs.org/v2/guide/components-props.html#Prop-Validation)

  <my-component

        prop\_one="ass"

        v-bind:prop\_two="12"

        prop\_three="12"

        prop\_sex="danger"

      >

      </my-component>

# 

Vue.component("my-component", {

  props: {

    // Просто проверка типа (`null` и `undefined` проходят проверку для любого типа)

    prop\_one: String,

    // Несколько допустимых типов

    prop\_two: [Number, Boolean],

    // // Обязательное значение строкового (String) типа

    prop\_three: {

      type: String,

      required: false,

    },

    // // Число со значением по умолчанию

    prop\_foo: {

      type: Number,

      default: 100,

    },

    // // Объект

     // data: {

    //   type: Object,

    //   default: () => {},

    //   required: true,

    // },

    // // Объект со значением по умолчанию

    prop\_five: {

      type: Object,

      // Для объектов или массивов значения по умолчанию

      // должны возвращаться из функции

      default: function () {

        return { message: "hello" };

      },

    },

    // // Пользовательская функция для валидации

    prop\_sex: {

      required: false,

      validator: function (value) {

        // Значение должно соответствовать одной из этих строк

      return ["success", "warning", "danger"].indexOf(value) !== -1;

      },

  // prop\_seven: {

    //   validator: Person,

    // },

    },

  },

  template:

    "<h1>{{ prop\_one }} | {{ prop\_two }} | {{ prop\_three }} | {{prop\_foo}} | {{prop\_five.message}} |{{prop\_sex}}</h1>",

});

// function Person(value) {

//   return ["success", "warning", "danger"].indexOf(value) !== -1;

// }

## ??? [Non-Prop Attributes](https://vuejs.org/v2/guide/components-props.html#Non-Prop-Attributes)

который требует указания атрибута data-date-picker на элементе input. Мы можем добавить этот атрибут к нашему экземпляру компонента:

<bootstrap-date-input data-date-picker="activated"></bootstrap-date-input>

### ??? [Replacing/Merging with Existing Attributes](https://vuejs.org/v2/guide/components-props.html#Replacing-Merging-with-Existing-Attributes)

### ??? [Disabling Attribute Inheritance](https://vuejs.org/v2/guide/components-props.html#Disabling-Attribute-Inheritance)

**Custom Events**

## [Event Names](https://vuejs.org/v2/guide/components-custom-events.html#Event-Names)

 <component-a

      v-on:enlarge-text="postFontSize += $event"

//enlarge-text -деп жазылса emit- қалай жазылуы керек

    />

    {{ postFontSize }}

**Былай жазылса дұрыс болмайды**

<button v-on:click="$emit('enlargeText', 10)">

      Click

    </button>

**Осылай жазса дұрыс болады, яғни (enlarge-text )**

<button v-on:click="$emit('enlarge-text', 10)">

      Click

    </button>

## [Customizing Component v-model](https://vuejs.org/v2/guide/components-custom-events.html#Customizing-Component-v-model)

Компонент ішінде тұрып v-model -ге настройка жасау үшін, және оның значениесін родительскыйға жіберу

Vue.component('base-checkbox', {

model: {

prop: 'checked', input - тың өзгерісін үстау үшін

event: 'change'

},

props: {

checked: Boolean input - тың нәтижесін көрсету үшін

},

template: `

<input

type="checkbox"

v-bind:checked="checked"

v-on:change="$emit('change', $event.target.checked)"

>

`

})

<base-checkbox v-model="lovingVue"></base-checkbox>

    <base-checkbox v-model="lovingVue"></base-checkbox>

    {{ lovingVue }} // компонент ішіндегі инпуттың value-сін родительскыйда тұрып білу үшін керек

data() {

    return {

      lovingVue: null,

    };

  },

<template>

  <div>

    <input

      type="checkbox"

      v-on:change="$emit('change', $event.target.checked)"

    />

    {{ checked }}

  </div>

</template>

<script>

export default {

  model: {

    prop: "checked", //input - тың өзгерісін үстау үшін

    event: "change", //input - тың өзгерісін главныйға жіберу үшін

  },

  props: {

    checked: Boolean, //input - тың нәтижесін көрсете алу үшін

  },

};

</script>

## [Binding Native Events to Components](https://vuejs.org/v2/guide/components-custom-events.html#Binding-Native-Events-to-Components)

компонент ішіндегі собитияларының функциясын родительскыйде шақыру

яғни сыртта тұрып іштегі жағдаяттарды анықтап сонымен сырттан функция беру

оны істеу үшін **native** қолданылады

    <base-button @click.native="onFocus()" />

 methods: {

    onFocus() {

      console.log("alert");

    },

  },

<template>

  <div>

    <button>Click Btn</button>

  </div>

</template>

**native**- кей жағдайларда проблема байқатады оны реттеу үшін , қате шығармау үшін, **native** **-** емес басқаны  **$listeners**-ті қолданады

 <base-input v-on:focus="onFocus"></base-input>

  methods: {

    onFocus() {

      console.log("alert");

    },

  },

<template>

  <div>

    <label>

      {{ label }}

      <input v-bind="$attrs" v-bind:value="value" v-on="inputListeners" />

    </label>

  </div>

</template>

<script>

export default {

  inheritAttrs: false,

  props: ["label", "value"],

  computed: {

    inputListeners: function() {

      var vm = this;

      return Object.assign({}, this.$listeners, {

        input: function(event) {

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

        },

      });

    },

  },

};

</script>

## [.sync Modifier](https://vuejs.org/v2/guide/components-custom-events.html#sync-Modifier)

**родительскыйдегі** переменныйдың мәнін **компоненттің** ішіндегі кнопка арқылы өзгерткіміз келсе осы тәсілді қолданамыз

 //родительскый - де

<base-sync :val="number" v-on:sync-text="number=$event"/>

data() {

    return {

      number: 12,

    };

  },

# 

  //components -ішінде

<button @click="$emit('sync-text', val + 12)">Click</button>

props: ["val "],

# егер жоғарыдағы тәсілді қысқа әрі ыңғайлы тәсілде жасау үшін .sync  тәсілімен істеу керек ол төмендегідей

 //родительскый - де

<base-sync v-bind:val.sync="number" />

// v-bind:title.sync="doc.title + '!' -осылай жазса қате болады, өйткені артық нәрселер көбейсе , операция қиындап қате шығарады

data() {

    return {

      number: 12,

    };

  },

//components -ішінде

<button @click="$emit('update:val', val + 12)">Click</button>

props: ["val "],

# .sync  тәсілімен істеген кезде обьект жіберетін болса осылай қысқа қайтаруға болады және общый обьектный данный жібергің келседе осылай істеуге болады v-bind.sync="obj"

  <base-sync v-bind.sync="obj" />

 //v-bind.sync="{ title: doc.title }"' -осылай жазса қате болады, өйткені артық нәрселер көбейсе , операция қиындап қате шығарады

    {{ obj.val }}

data() {

    return {

      obj: { val: 23 },

    };

  },

# 

//components -ішінде

<button @click="$emit('update:val', val + 15)">Click</button>

props: ["val "],

# 

**Slots**

## [Slot Content](https://vuejs.org/v2/guide/components-slots.html#Slot-Content)

//родительскый

<navigation-link url="/profile">

Ваш профиль

# </navigation-link>

# //дочерный компонент

<a

v-bind:href="url"

class="nav-link"

>

<slot></slot>

# </a>

//родительскый

<navigation-link url="/profile">

<!-- slot родительскый-ға иконку қойсада көрсетеді -->

<span class="fa fa-user"></span>

Ваш профиль

# </navigation-link>

//родительскый

<navigation-link url="/profile">

<!-- slot родительскый-ға компонент қойсада көрсетеді компонент-->

<font-awesome-icon name="user"></font-awesome-icon>

Ваш профиль

# </navigation-link>

# Егер компонент ішіне  <slot>, қойылмасалюбой қойылған контент будет просто проигнорирован. (көрінбейді)\

## [Compilation Scope](https://vuejs.org/v2/guide/components-slots.html#Compilation-Scope)

компонент ішіндегі данныйларды, родительскый де шақырылған <компонент тегінің> арасында {{ көрсете }} алмаймыз алмаймыз

    <base-slot>Example{{ date }}</base-slot>

  <div>

    <h2>icon</h2>

    <slot></slot>

  </div>

  data() {

    return { data: "deloo" };

  },

## [Fallback Content](https://vuejs.org/v2/guide/components-slots.html#Fallback-Content)

<!-- page  -->

<base-slot> {{ date.n }}</base-slot>

data() {

    return { date: { n: "Отправить" } };

  },

<!-- component  -->

<button><slot></slot></button>

<!-- <button><slot>Отправить</slot></button> -->

## [Named Slots](https://vuejs.org/v2/guide/components-slots.html#Named-Slots)

## Гловныйда v-slot:="header" арқылы данныйды белгілеп, name="header" арқылы оны компонентте шақырып алуға болады

**v-slot - тек** template - те ғана шақыра алады

<base-slot> <!-- page  -->

      <template v-slot:header>

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

      </template>

      <template v-slot:default>

        <p>Параграф для основного контента.</p>

        <p>И ещё один.</p>

      </template>

      <template v-slot:footer>

        <h3>Некая контактная информация</h3>

      </template>

</base-slot>

 <div class="container"> <!-- component  -->

      <header>

        <slot name="header"></slot>

      </header>

      <main>

        <slot></slot>

      </main>

      <section>

        <slot name="header"></slot>

      </section>

      <footer>

        <slot name="footer"></slot>

      </footer>

    </div>

## [Scoped Slots](https://vuejs.org/v2/guide/components-slots.html#Scoped-Slots)

## компоненттердегі данныйларды родительскыйда көрсету,

## компоненттердегі данныйларға 3 -тәсілмен доступ алу үшін

<base-slot>   //родительскый

      <template #default="{user}">

        {{ user.lastName }}

      </template>

      <template #action="{user}"> //slot-name арқылы беру

        {{ user.firstName }}

      </template>

      <template #main="slotProps">

        {{ slotProps.user.main }}//қосымша переменный енгізу арқылы алу

      </template>

</base-slot>

### 

  //компонент

<slot :user="user"> </slot>

      <slot name="action" :user="user"> </slot>

      <slot name="main" :user="user"> </slot>

data() {

    return {

      user: {

        firstName: "Отправить",

        lastName: "Coxронить",

        main: "SlotProps",

      },

    };

  },

### # [Abbreviated Syntax for Lone Default Slots](https://vuejs.org/v2/guide/components-slots.html#Abbreviated-Syntax-for-Lone-Default-Slots)

//родительскый

<base-slot v-slot:default="{ user }">

<!-- <base-slot v-slot="{user}">     -->

<!-- <base-slot v-slot:default="suser"> -->

      {{ user.main }}

<!-- {{ suser.user.main }} -->

 </base-slot>

  //компонент

<slot :user="user"> </slot>

Компонент ішіне template жазған кезде компонентке слот берсең қате болады

<base-slot v-slot:default="suser">

      {{ suser.user.main }}

      <template v-slot:action="{ user }">

        {{ user.main }}

      </template>

</base-slot>

Оның орнына бірнеше компонент жазсаң соларға слот берсең дұрыс болады

   <base-slot>

      <template v-slot:default="suser">

        {{ suser.user.main }}

      </template>

      <template v-slot:action="{ user }">

        {{ user.main }}

      </template>

    </base-slot>

### # [Destructuring Slot Props](https://vuejs.org/v2/guide/components-slots.html#Destructuring-Slot-Props)

<current-user v-slot="{ user }">

{{ user.firstName }}

# </current-user>

<current-user v-slot="{ user: person }">

{{ person.firstName }}

# </current-user>

<current-user v-slot="{ user = { firstName: 'Гость' } }">

{{ user.firstName }}

# </current-user>

## [Dynamic Slot Names](https://vuejs.org/v2/guide/components-slots.html#Dynamic-Slot-Names)

<base-layout>

<template v-slot:[dynamicSlotName]>

...

</template>

# </base-layout>

# родительскый мен компонентті байланыстыратын v-slot-тың (атын) Name-ін JavaScript арқылы қолдан беруге болады

// родительскый

<base-icon>

      <template v-slot:[datas]>{{ message }}</template>

    </base-icon>

 data() {

    return { datas: "action", message: "hello" };

  },

< компонент

<slot name="action"></slot>

# 

## [Named Slots Shorthand](https://vuejs.org/v2/guide/components-slots.html#Named-Slots-Shorthand)

# Например, v-slot:header можно записать как #header:

<base-layout>

<template #header>

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

</template>

<p>Параграф для основного контента.</p>

<p>И ещё один.</p>

<template #footer>

<p>Некая контактная информация</p>

</template>

# </base-layout>

<!-- **Қате** Это выкинет предупреждение -->

**<current-user #="{ user }">**

{{ user.firstName }}

# </current-user>

**<current-user #default="{ user }">**

{{ user.firstName }}

# </current-user>

## [Other Examples](https://vuejs.org/v2/guide/components-slots.html#Other-Examples)

Пример: **компонент** ішіндегі данныйларды **родительскый** де шақырып шарт беріп, сол шарт арқылы данныйларды шығару

//родительскый

<base-slot>

      <template v-slot:todo="{ todo }">

        <span style="color:red;" v-if="todo.isComplete">✓ </span>

        {{ todo.text }}

      </template>

</base-slot>

 //компонент

  <ul>

      <li v-for="todo in arr" v-bind:key="todo.id">

        <slot name="todo" v-bind:todo="todo">

          {{ todo.text }}

        </slot>

      </li>

    </ul>

 arr: [

        {

          id: 1,

          text: "Помыть посуду",

        },

        {

          id: 2,

          text: "Вынести мусор",

          isComplete: true,

        },

        {

          id: 3,

          text: "Подстричь газон",

          isComplete: false,

        },

      ],

# 

## [Deprecated Syntax](https://vuejs.org/v2/guide/components-slots.html#Deprecated-Syntax)

**v-slot белгісін жақсарту үшін slot и slot-scope белгілері жасап шығарылды, Бұл белгіле Vue2-де болғанымен Vue3-те өшірілді (есірді)**

### #[Named Slots with the slot Attribute](https://vuejs.org/v2/guide/components-slots.html#Named-Slots-with-the-slot-Attribute)

// родтительскый

<base-slot>

      <template slot="header">

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

      </template>

      <p>Параграф для основного контента.</p>

      <p>И ещё один.</p>

<!-- бұлар нәтижеде көрінбецді өйткені компоненттер бөлімінде бұларды шықырып тұрған slot  жоқ-->

      <template slot="footer">

        <p>Некая контактная информация</p>

      </template>

</base-slot>

   // компонент

<slot name="header"></slot>

    <slot name="footer"></slot>

блай беруге да болады

// родтительскый 2

 <base-slot>

      <h1 slot="header">Здесь мог быть заголовок страницы</h1>

      <p>Параграф для основного контента.</p>

      <p>И ещё один.</p>

      <p slot="footer">Некая контактная информация</p>

    </base-slot>

### #[Scoped Slots with the slot-scope Attribute](https://vuejs.org/v2/guide/components-slots.html#Scoped-Slots-with-the-slot-scope-Attribute)

# slot-scope - арқылы да жазса болады slot-ты бірге қолданған жақсы

# { user }- мына жерде жақшаны алып тастаса неге болмайды

v-slot:default="{user}"> - жақшамен жазылса компоненттен алынатын данныйды сразу шақырады

{{user}} - корсететін кезде осылай сразу көрсетеміз

v-slot:default="users"> - егер жақшасыз жазылса компоненттен шақырылатын данныйды осы тырнақша ішіндегі переменныйдың (переменный ды обьект ретінде алып) ішіне кіргізіп шақырады

{{users.user}} - корсететін кезде осылай көрсетеміз

# 

    <base-slot>

      <template slot="default" slot-scope="{ user }">  <!--{ user }- мына жерде жақшаны алып тастаса неге болмайды-->

        <!-- <template slot-scope="{ user }"> -->

        <!-- <span slot-scope="{ user }"> -->

        {{ user.firstName }}

        <!-- </span> -->

      </template>

    </base-slot>

# 

    <slot :user="user"></slot>

  data() {

    return {

      user: {

        firstName: "Отправить",

        lastName: "Coxронить",

        main: "SlotProps",

      },

},

},

**Dynamic & Async Components**

## [keep-alive with Dynamic Components](https://vuejs.org/v2/guide/components-dynamic-async.html#keep-alive-with-Dynamic-Components)

**keep-alive** - деген бір компоненттен екіншісіне өткен кезде жазылған( мысалы input - та ) данныйларды сақтап қалу

*https://www.youtube.com/watch?v=09n2945JW\_0*

<template>

  <div>

    <keep-alive>

        <tr v-bind:is="component"></tr>

    </keep-alive>

    <button @click="ComponentCreate()">Companent Create</button>

  </div>

</template>

<script>

import BlogPostRow from "@/components/BlogPostRow";

import NewBlock from "@/components/NewBlock";

export default {

  data() {

    return {

      component: "blog-post-row",

      componen: "new-block",

    };

  },

  components: {

    BlogPostRow,

    NewBlock,

  },

  methods: {

    ComponentCreate() {

      this.component === "new-block"

        ? (this.component = "blog-post-row")

        : (this.component = "new-block");

    },

  },

};

</script>

 что <keep-alive> требует, чтобы у всех переключаемых компонентов было

* задано имя,
* либо через опцию name компонента,
* либо через локальную/глобальную регистрацию.

## [Async Components](https://vuejs.org/v2/guide/components-dynamic-async.html#Async-Components)

**Компонентті ассинхронный қылу үшін компоненттің ішіне осы лар кжазылады**

Vue.component('async-example', function (resolve, reject) {

setTimeout(function () {

// Передаём определение компонента в коллбэк resolve

resolve({

template: '<div>Я — асинхронный!</div>'

})

}, 1000)

})

**, немес**

Vue.component('async-webpack-example', function (resolve) {

// Специальный синтаксис require укажет Webpack

// автоматически разделить сборку на части

// для последующей асинхронной загрузки

require(['./my-async-component'], resolve)

### })

CallBack - арқылы компонентті глобальный шақыру

    <async-webpack-example />

Vue.component(

  "async-webpack-example",

  // Динамический импорт возвращает Promise

  () => import("./components/asyng")

);

CallBack - арқылы компонентті локальный шақыру

    <my-component />

  components: {

    myComponent: () => import("@/components/asyng"),

  },

### # [Handling Loading State](https://vuejs.org/v2/guide/components-dynamic-async.html#Handling-Loading-State)

<async-webpack-example />

import LoadingComponent from "@/components/asyng\_\_load.vue";

import ErrorComponent from "@/components/asyng\_\_error.vue";

components: {

    AsyncWebpackExample: () => ({

      // Загружаемый компонент. Значение должно быть Promise

      component: import("@/components/asyng.vue"),

      // Компонент загрузки, используемый пока загружается асинхронный компонент

      loading: LoadingComponent,

      // Компонент ошибки, используемый при неудачной загрузке

      error: ErrorComponent,

      // Задержка перед показом компонента загрузки. По умолчанию: 200 мс.

      delay: 200,

      // Компонент ошибки будет отображаться, если таймаут

      // был указан и время ожидания превышено. По умолчанию: Infinity.

      timeout: 3000,

    }),

  },

Жоғарыдағы мысал негізі орындалмады, оның орнына астындағы осы мысалды қолдануға болады

    <async-web></async-web>

import NumberBoxComponent from "@/components/asyng.vue";

import LoadingComponent from "@/components/asyng\_\_load.vue";

import ErrorComponent from "@/components/asyng\_\_error.vue";

 AsyncWeb: () => ({

      component: new Promise((resolve /\* reject\*/) => {

        setTimeout(() => {

          resolve(NumberBoxComponent);

          // reject(ErrorComponent);

        }, 3000);

      }),

      loading: LoadingComponent,

      delay: 200,

    }),

Осы мысалдың html- түрінде жазылғаны

const LoadingComponent = { template: `<h1>Loading...</h1>` }

const NumberBoxComponent = { template: `<h1>123123</h1>` }

const AsyncComponent1 = () => ({

component: new Promise((resolve) => {

setTimeout(() => {

resolve(NumberBoxComponent)

}, 1000)

}),

loading: LoadingComponent,

})

const AsyncComponent2 = () => ({

component: new Promise((resolve) => {

setTimeout(() => {

resolve(NumberBoxComponent)

}, 1000)

}),

loading: LoadingComponent,

delay: 500

})

new Vue({

el: '#app',

components: {

'async-component1': AsyncComponent1,

'async-component2': AsyncComponent2

}

})

<script src="https://cdnjs.cloudflare.com/ajax/libs/vue/2.4.4/vue.min.js"></script>

<div id="app">

<async-component1></async-component1>

<async-component2></async-component2>

</div>

**Handling Edge Cases**

## [Element & Component Access](https://vuejs.org/v2/guide/components-edge-cases.html#Element-amp-Component-Access)

### # [Accessing the Root Instance](https://vuejs.org/v2/guide/components-edge-cases.html#Accessing-the-Root-Instance)

Мысалы бір компоненттің ішінде осындай даный лар бар

data: {

foo: 1

},

computed: {

bar: function () { /\* ... \*/ }

},

methods: {

baz: function () { /\* ... \*/ }

}

, осы данный ларды бірінші главный Vue - ға main.js -қа тіркеп қою керек

import api from "./utils/COMPONENT"

Vue.prototype.$api = api;

сосын ол данный ларды кез келген жерден , кез келген компоненттен шақыруға болады,

this.$root.foo

// Установка данных в корневом экземпляре

this.$root.foo = 2

// Использование вычисляемых свойств корневого экземпляра

this.$root.bar

// Вызов методов корневого экземпляра

this.$root.baz()

.$root - дегенді жазбаяқ шақыру керек, егер шықпай жатса .$root - ты жазу керек , негізі .$root крайный жағдай да қолданылады

### # [Accessing the Parent Component Instance](https://vuejs.org/v2/guide/components-edge-cases.html#Accessing-the-Parent-Component-Instance)

родительскый ішінде осындай данный болса

 data() {

    return { foo: 13 };

  },

оны this.$parent-арқылы компонент ішінен осылай шақыруға болады

this.$parent.foo

### #[Accessing Child Component Instances & Child Elements](https://vuejs.org/v2/guide/components-edge-cases.html#Accessing-Child-Component-Instances-amp-Child-Elements)

<base-input ref="usernameInput"></base-input>

this.$refs.usernameInput

<input ref="input">

methods: {

// Используется родителем для фокуса на input

focus: function () {

this.$refs.input.focus()

}

}

this.$refs.usernameInput.focus()

    <input type="text" ref="input" />  // ref - деген белгілеп алу үшін input деген ref-тің аты, қандай атпен белгіліеу керек екенін жазу, киін шықырғанда керек болады

    <button @click="fucus">Click</button>

  methods: {

    fucus() {

      this.$refs.input.focus();// refs -деген ref-ті шақыру үшін input алдында ref-ке белгілеп қойған name

    },

  },

### #[Dependency Injection](https://vuejs.org/v2/guide/components-edge-cases.html#Dependency-Injection)

Vuex -сияқты переменныйларды главныйға айналдырып сосын оларды қалаған жерден шақыруға болатын қылу

  provide() { // главный переменный ге айналдыру үшін осылай жазу керек

    return {

      getMap: 12234,

    };

  },

    {{ getMap }}

  inject: ["getMap"], //главный элементтке айналған переменный ды шақыру үшін

## [Programmatic Event Listeners](https://vuejs.org/v2/guide/components-edge-cases.html#Programmatic-Event-Listeners)

Прослушивать событие с помощью **$on(eventName, eventHandler)**

Прослушивать событие только один раз с помощью **$once(eventName,eventHandler)**

Прекращать прослушивание события с помощью **$off(eventName, eventHandler)**

// родительскый

    <base-icon v-on:title="searchText($event)"> </base-icon>

methods: {

    searchText(n) {

      console.log(n);

    },

  },

// компонент

<button @click="votes()">Click</button>

methods: {

    votes() {

      this.$emit("title", 13);

    },

  },

  created() {

    this.$on("title", () => {

      // $on - компоненттің ішіндегі created-ті қайта қайта орындауға мүмкіндік береді, title - бұл қайсы компоненттен шақырылып жатқанын анықтау үшін

      console.log("created-on");

    });

    this.$once("title", () => {

      // $once - компоненттің ішіндегі created-ті бір рет қан орындауға бұйырады

      console.log("created-once");

      this.$off("title"); // $off - компоненттің ішіндегі created-ті қалаған жерден орындалуын тоқтату үшін керек

    });

  },

## [Circular References](https://vuejs.org/v2/guide/components-edge-cases.html#Circular-References)

### ##[Recursive Components](https://vuejs.org/v2/guide/components-edge-cases.html#Recursive-Components) ???

компонент ішінен компонент шақыруға байланысты

name: 'unique-name-of-my-component'

Vue.component('unique-name-of-my-component', {

// ...

})

name: 'stack-overflow',

template: '<div><stack-overflow></stack-overflow></div>'

### #[Circular References Between Components](https://vuejs.org/v2/guide/components-edge-cases.html#Circular-References-Between-Components) ???

компоненттер бір-бірінің орындалуын шарт етіп қойса , екеуіде орындалмай қалады , мұндай жағдайды реттеу үшін require немесе import тәсілдері қолданылады

### Система модулей видит, что ей нужен A, но сначала A нуждается в B, но B нуждается в A, но A нуждается в B, и т. д.

в которой она может сказать «A нуждается в B иногда, но нет необходимости разрешать B сначала».

beforeCreate: function () {

this.$options.components.TreeFolderContents = require('./tree-folder-contents.vue').default

}

Или вы можете использовать асинхронный import компонента:

components: {

TreeFolderContents: () => import('./tree-folder-contents.vue')

}

## [Alternate Template Definitions](https://vuejs.org/v2/guide/components-edge-cases.html#Alternate-Template-Definitions)

### #[Inline Templates](https://vuejs.org/v2/guide/components-edge-cases.html#Inline-Templates)

Компоненттен template - ті ұстап ішіндегі данныйларды алу

 // родителскый -де тұрып компонен ішіндегі данныйларды алу

<base-slot inline-template>

      <div>

        <h2>dfdff{{ folders }}</h2>

      </div>

 </base-slot>

// компонент

<script>

export default {

  data() {

    return {

      folders: 21,

    };

  },

### #[X-Templates](https://vuejs.org/v2/guide/components-edge-cases.html#X-Templates)

компоненттегі данныйларды осы тәсілменде шақыруға болады

CLI - тәсілмен қылуға болмады ???

### <hello-world></hello-world>

<script type="text/x-template" id="hello-world-template">

<p>Привет, привет, привет</p>

</script>

Vue.component('hello-world', {

template: '#hello-world-template'

})

## [Controlling Updates](https://vuejs.org/v2/guide/components-edge-cases.html#Controlling-Updates)

### # [Forcing an Update](https://vuejs.org/v2/guide/components-edge-cases.html#Forcing-an-Update)

қолдан обновит жасау үшін .$forceUpdate немес .$mount()

    <h1>{{ Math.random() }}</h1>

    <button @click="update()">Click</button>

  methods: {

    update() {

      this.$forceUpdate();

     // this.$mount();

    },

  },

### #[Cheap Static Components with v-once](https://vuejs.org/v2/guide/components-edge-cases.html#Cheap-Static-Components-with-v-once)

қолдан обновит жасалса өзгермейтін **блок**  қылу үшін v-once қолданылады

    <h1>{{ Math.random() }}</h1>

    ||

    <h1 v-once>{{ Math.random() }}</h1>

  methods: {

    update() {

      this.$forceUpdate();

     // this.$mount();

    },

  },