**Продвинутые темы**

**Жалпы Vue структурасы қалай жүзеге асады сол туралы**

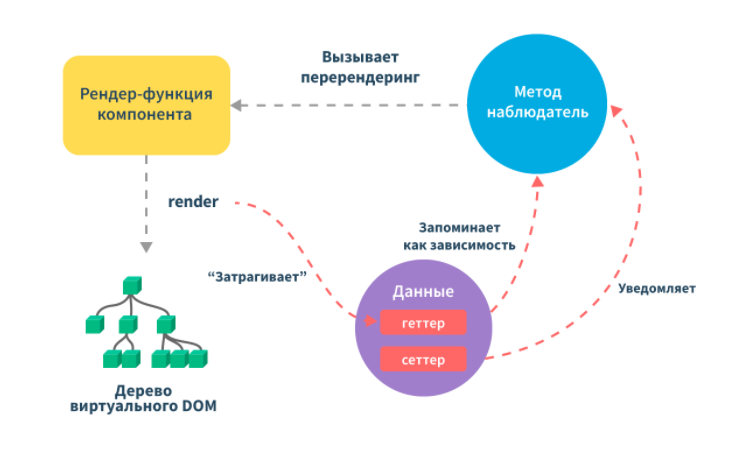
# Подробно о реактивности

Біз негіздердің көпшілігін қарастырдық, сондықтан тереңірек сүңгудің уақыты келді! Vue-дің ең көрнекті ерекшеліктерінің бірі оның көзге түспейтін реактивтілігі болып табылады. Модельдер қарапайым JavaScript нысандары болып табылады. Олар өзгерген сайын деректер көрінісі де өзгереді, бұл қолданба күйін басқаруды қарапайым және қарапайым етеді. Осыған қарамастан, реактивтілік механизмінің бірқатар ерекшеліктері бар, олармен танысу жалпы қателердің пайда болуын болдырмайды. Оқулықтың осы бөлімінде біз Vue реактивтілігі жүйесінің төмен деңгейде іске асырылуын егжей-тегжейлі қарастырамыз.

Vue JavaScript арқылы істеледі, Vue-дегі данныйлар өзгерсе JavaScript-те де кодтар өзгереді

## [Как отслеживаются изменения](https://ru.vuejs.org/v2/guide/reactivity.html#%D0%9A%D0%B0%D0%BA-%D0%BE%D1%82%D1%81%D0%BB%D0%B5%D0%B6%D0%B8%D0%B2%D0%B0%D1%8E%D1%82%D1%81%D1%8F-%D0%B8%D0%B7%D0%BC%D0%B5%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F)

**Өзгерістер қалай бақыланады**

****

## [Особенности отслеживания изменений](https://ru.vuejs.org/v2/guide/reactivity.html#%D0%9E%D1%81%D0%BE%D0%B1%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D0%B8-%D0%BE%D1%82%D1%81%D0%BB%D0%B5%D0%B6%D0%B8%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F-%D0%B8%D0%B7%D0%BC%D0%B5%D0%BD%D0%B5%D0%BD%D0%B8%D0%B9)

**Бақылау мүмкіндіктерін өзгерту**

JavaScript шектеулеріне байланысты Vue анықтай алмайтын өзгерістер түрлері бар. Дегенмен, реактивтілікті сақтау үшін оларды айналып өтудің жолдары бар.

var vm = new Vue({ }) сыртында тұрып ішіне данный қосу үшін тәсілдері

**#** [**Для объектов**](https://ru.vuejs.org/v2/guide/reactivity.html#%D0%94%D0%BB%D1%8F-%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BE%D0%B2)

Vue свойства-да қосуды немесе өшіруді анықтау алмайды. Vue даналау фазасында геттер/сеттер қосатындықтан, свойства data обьекінде болуы керек Vue оны түрлендіру және реактивті ету үшін. Мынау сияқты:

var vm = new Vue({

data: {

a: 1

}

})

// теперь `vm.a` — реактивное поле

Vue ішінде бұрыннан бар данаға жаңа түбірлік реактивті сипаттарды динамикалық түрде қосу мүмкін емес. Дегенмен, Vue.set (нысан, propertyName, мән) әдісі арқылы кірістірілген нысандарға реактивті сипатты қосуға болады.

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

Сондай-ақ жаһандық Vue.set лақап аты болып табылатын vm. $ Set данасы әдісін пайдалануға болады:

this.$set(this.someObject, 'b', 2)

Кейде бар нысанға бірнеше сипаттарды қосу қажет, мысалы Object.assign () немесе \_.extend (), бірақ Осылай жасасаңыз, қосылған сипаттар реактивті болмайды. Бұл мәселені шешу үшін бастапқы нысанның да, қоспа нысанының да өрістерін қамтитын жаңа нысан жасау керек:

this.someObject = Object.assign({}, this.someObject, { a: 1, b: 2 })

**#** [**Для массивов**](https://ru.vuejs.org/v2/guide/reactivity.html#%D0%94%D0%BB%D1%8F-%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2%D0%BE%D0%B2)

**Vue массивтегі келесі өзгерістерді бақылай алмайды:**

var vm = new Vue({

data: {

items: ['a', 'b', 'c']

}

})

vm.items[1] = 'x' // НЕ РЕАКТИВНО

vm.items.length = 2 // НЕ РЕАКТИВНО

Бірінші мәселені шешудің екі жолы бар, екеуі де vm.items [indexOfItem] = newValue сияқты әсер береді, сонымен қатар олар қолданба күйіне реактивті жаңартуларды қосады:

// Использовать Vue.set

Vue.set(vm.items, indexOfItem, newValue)

// Использовать Array.prototype.splice

vm.items.splice(indexOfItem, 1, newValue)

Можно использовать метод экземпляра **[vm.$set](https://ru.vuejs.org/v2/api/" \l "vm-set" \t "_blank)**, который является псевдонимом для глобального Vue.set:

vm.$set(vm.items, indexOfItem, newValue)

Для решения второй проблемы используйте splice:

vm.items.splice(newLength)

data: {

    items: { text: "Один", value: "А" },

  },

vm.$set(vm.items, "fffddd", "newValue");

//результат



## [Объявление реактивных свойств](https://ru.vuejs.org/v2/guide/reactivity.html#%D0%9E%D0%B1%D1%8A%D1%8F%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5-%D1%80%D0%B5%D0%B0%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D1%85-%D1%81%D0%B2%D0%BE%D0%B9%D1%81%D1%82%D0%B2)

Реактивті қасиеттерді жариялау

Vue сізге түбірлік реактивті қасиеттерді динамикалық түрде қосуға мүмкіндік бермейтіндіктен, бұл барлық түбірлік өрістерді ең болмағанда бос мәндермен бастапқыда инициализациялау керек дегенді білдіреді:

var vm = new Vue({

data: {

// объявляем поле message, содержащее пустую строку

message: ''

},

template: '<div>{{ message }}</div>'

})

// впоследствии задаём значение `message`

vm.message = 'Привет!'

Деректер опциясында хабар өрісін орнатпасаңыз, Vue көрсету функциясы жоқ сипатқа кіруге әрекеттеніп жатқаны туралы ескертеді.

Бұл шектеудің техникалық себептері бар: ол тәуелділікті есепке алу жүйесіндегі шеткі істердің бүкіл класын жоюға мүмкіндік береді, сонымен қатар Vue түрін тексеру жүйелерімен өзара әрекеттесуді жеңілдетеді. Бірақ одан да маңыздысы, бұл шектеу кодты сақтауды жеңілдетеді, өйткені деректер нысанын енді компонент күйінің схемасы ретінде қарастыруға болады. Компоненттің реактивті қасиеттерін алдын ала тізімдейтін кодты түсіну оңайырақ.

## [Асинхронная очередь обновлений](https://ru.vuejs.org/v2/guide/reactivity.html#%D0%90%D1%81%D0%B8%D0%BD%D1%85%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F-%D0%BE%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C-%D0%BE%D0%B1%D0%BD%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B9)

Асинхронды жаңарту кезегі

Кез келген жағдайда Vue DOM жаңартуларын асинхронды түрде жасайтынын есте сақтаңыз. Деректерде өзгеріс анықталған сайын, осы үшін буфер ретінде пайдаланылатын кезек жасалады және оқиғалар циклінің ағымдағы итерациясында («белгі») орын алатын кейінгі өзгерістер. Бір бақылаушы бірнеше рет атса, ол кезекке тек бір рет түседі. Буферді пайдалану және көшірмелерді жою арқылы DOM есептеуі мен манипуляциясы минимумға дейін сақталады. Оқиғалар циклінің келесі итерациясында Vue кезекті талдайды және нақты (бұдан әрі қайталанбайтын) жаңартуларды орындайды. Төмен деңгейде Promise.then, MutationObserver және setImmediate тапсырмаларды асинхронды кезекке қою үшін пайдаланылады және олар қол жетімді болмаса, setTimeout (fn, 0).

Сонымен, vm.someData = 'жаңа мән' кодын орындасаңыз, құрамдас бірден көрсетілмейді. Ол кезекті талдау кезінде келесі итерацияда жаңартылады. Көбінесе бұл мүмкіндікті елемеуге болады, бірақ кейде деректерді жаңартқаннан кейін DOM баратын күйді күту керек. Тікелей DOM манипуляциясы қажет емес және тұтастай алғанда жүйелерде өңделмеген деректер болатындай жобалаған дұрыс, бірақ кейде оны әлі де болдырмау мүмкін емес. Кейбір кодты DOM жаңартуы аяқталғаннан кейін ғана орындау үшін деректер өзгергеннен кейін кері қоңырауды Vue.nextTick (кері қоңырау) әдісіне жіберуге болады. Ол DOM жаңартылғаннан кейін шақырылады. Мысалға:

<div id="example">{{ message }}</div>

var vm = new Vue({

el: '#example',

data: {

message: '123'

}

})

vm.message = 'новое сообщение' // изменяем данные

vm.$el.textContent === 'новое сообщение' // false

Vue.nextTick(function () {

vm.$el.textContent === 'новое сообщение' // true

})

Сондай-ақ this арқылы да істеуге болады

Vue.component('example', {

template: '<span>{{ message }}</span>',

data: function () {

return {

message: 'не обновлено'

}

},

methods: {

updateMessage: function () {

this.message = 'обновлено'

console.log(this.$el.textContent) // => 'не обновлено'

this.$nextTick(function () {

console.log(this.$el.textContent) // => 'обновлено'

})

}

}

})

$ nextTick () Promise қайтаратындықтан, ES2017 жаңа асинхронды/күту синтаксисін пайдаланып бірдей нәтижеге қол жеткізе аласыз:

яғни async қолдануға болады

methods: {

updateMessage: async function () {

this.message = 'обновлено'

console.log(this.$el.textContent) // => 'не обновлено'

await this.$nextTick()

console.log(this.$el.textContent) // => 'обновлено'

}

}