Компоненти. v-model

Батьківський компонент передає *дочірньому* дані через *властивості-атрибути*

```
//-------- Дочірній компонент ----------
<template>
      . . . Опис розмітки . . .
</template>

<script>
  export default {

    .............

    props: {

      властивість_1 : {

       … опис властвості …

      }

     .............

  },

</script>
<style> . . . . . . . . . . . . . </style>
```

```
//-------- Батьківський компонент ----------
<template>
    . . . . . . . . . . . . . . . . . . . .
  <дочірній
     властивість = “ значення_string ”
     : властивість = “ обчислюване_значеня ”
   />
    . . . . . . . . . . . . . . . . . . . .
</template>

<script>
     . . . . . . . . . . . . . . . . . . .
</script>
<style>
   . . . . . . . . . . . . . . . . . . .
</style>
```

Батьківський компонент передає *дочірньому* дані через *властивості*

```
//-------- Дочірній компонент ----------
<template>
      . . . Опис розмітки . . .
</template>

<script>
  export default {

    ............

    props: {

      властивість_1 : {

      … опис властвості …

      }

      ............

  },

</script>
<style> . . . . . . . . . . . . . </style>
```

```
//-------- Батьківський компонент ----------
<template>
    . . . . . . . . . . . . . . .
  <дочірній
    властивість = “ значення_string ”
    : властивість = “ обчислюване_значеня ”
  />
    . . . . . . . . . . . . . . . . . . . .
</template>

<script>
    . . . . . . . . . . . . . . . . . . . .
</script>
<style>
   . . . . . . . . . . . . . . . . . . . .
</style>
```

батьківський підготував дані і передає у дочірній

Батьківський компонент передає *дочірньому* дані через *властивості*

```
//-------- Дочірній компонент ----------
<template>
      . . . Опис розмітки . . .
</template>

<script>
  export default {

    ............

    props: {

       властивість_1 : {

       … опис властвості …

      }

      ............

  },

</script>
<style> . . . . . . . . . . . . . . </style>
```

```
//-------- Батьківський компонент ----------
<template>
    . . . . . . . . . . . . . . . . . . . .
  <дочірній
     властивість = “ значення_string ”
     : властивість = “ обчислюване_значеня ”
    />
    . . . . . . . . . . . . . . . . . . . .
</template>

<script>
     . . . . . . . . . . . . . . . . . . .
</script>
<style>
    . . . . . . . . . . . . . . . . . . .
</style>
```

1)батьківський передає дочірньому компоненту дані через властивості-атрибути у ***props*** дочірнього
2)при присвоєнні нових значень у батьківському компоненті дочірній автоматично оновлюється

Передача вхідних параметрів-властивостей (передача компоненту даних)

Дочірній компонент

BlogPost.vue components U

src > components > BlogPost.vue > {} "BlogPost.vue" > template > div.t

```vue
<template>
  <div class="task-container">
    <div>{{ title }}</div>
    <div>{{ likes }}</div>
    <button @click="onLike">Like</button>
    <button @click="onDislike">Dislike</button>
  </div>
</template>

<script>
export default {
  name: 'BlogPost',

  props: {
    title: {
      type: String,
      default: 'Guest',
    },
    likes: {
      type: Number,
      default: 0,
    },
  },

  methods: {…
  },

<style lang="css" scoped>…
```

Батьківський компонент

App.vue src M

src > App.vue > {} "App.vue"

You, 42 seconds ago | 1 author (You)

```vue
<template>
  <blog-post title="Моя подорож з Vue" :likes="25" />
</template>

<script>
import BlogPost from './components/BlogPost.vue'

export default {
  name: 'App',
  components: {
    BlogPost,
  },
}
</script>

<style>…
```

Моя подорож з Vue 25 Like Dislike

Передача даних від дочірнього компонента у батьківський

Дочірній компонент

```
<template>
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”
    methods: {
      метод () {
        ......
        this.$emit(‘назва_події’, дані )
        ......
      }
    },
    ......
 }
</script>
<style>  ... </style>
```

1.Дочірній компонент генерує подію і за потреби передає дані

Передача даних від дочірнього компонента у батьківський

Дочірній компонент

```
<template>
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”
    methods: {
      метод () {
        ......
        this.$emit(‘назва_події’, дані )
        ......
      }
    },
    ......
 }
</script>
<style>  ... </style>
```

1.Дочірній компонент генерує подію і за потреби передає дані

Батьківський компонент

```
<template>
  . . . . .
  <дочірній_компонент
      @назва_події=“ обробник_події ”
  />
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”
    methods: {
      .......
      обробник_події ( дані ) {
        .......
      },
      .......
    },
    ......
 }
</script>
<style>  ... </style>
```

2. Для дочірнього компонента призначаємо *обробника* події

Передача даних від дочірнього компонента у батьківський

Дочірній компонент

```
<template>
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”
    methods: {
      метод () {
        ......
        this.$emit(‘назва_події’, дані )
        ......
      }
    },
    ......
 }
</script>
<style>  ... </style>
```

1.Дочірній компонент генерує подію і за потреби передає дані

Батьківський компонент

```
<template>
  . . . . .
  <дочірній_компонент
      @назва_події=“ обробник_події ”
  />
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”
    methods: {
      ......
      обробник_події ( дані ) {
        ......
      },
      ......
    },
    ......
 }
</script>
<style>  ... </style>
```

2. Для дочірнього компонента призначаємо обробника події

3. Викликається обробник події

Передача даних від дочірнього компонента у батьківський

Дочірній компонент

```
<template>
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”
    methods: {
      метод () {
        ......
        this.$emit(‘назва_події’, дані )
        ......
      }
    },
    ......
 }
</script>
<style>  ... </style>
```

1.Дочірній компонент генерує подію і за потреби передає дані

Батьківський компонент

```
<template>
  . . . . .
  <дочірній_компонент
      @назва_події=“ обробник_події ”
  />
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”
    methods: {
      ......
      обробник_події ( дані ) {
        ......
      },
      ......
    },
    ......
 }
</script>
<style>  ... </style>
```

2. Для дочірнього компонента призначаємо обробника події

3. Викликається обробник події

4. Обробник події отримує передані разом з подією дані

PostCard.vue components U

src > components > PostCard.vue > { } "PostCard.vue"

```vue
<template>
  <div class="container">
    <div class="logo-container">
      <img v-if="cardData.logo" :src="cardData.logo"
      class="logo" />
      <div v-else class="logo">{{ firstLetter }}</div>
    </div>
    <div class="content">
      <div>{{ cardData.userName }}</div>
      <div>{{ cardData.message }}</div>
      <div>
        <button @click="$emit('like',cardData.id)">
          Like
        </button>
        <span>{{ cardData.likes }}</span>
        <button @click="$emit('dislike',cardData.id)">
          DisLike
        </button>
      </div>
    </div>
  </div>
</template>

<script>
export default {…
}
</script>
```

PostsPanel.vue components U

src > components > PostsPanel.vue > { } "PostsPa

```vue
<template>
  <post-card
    v-for="card in postsList"
    :key="card.id"
    :card-data="card"
    @like="onLike"
    @dislike="onDislike"
  />
</template>

<script>
import PostCard from './PostCard.vue'
export default {
  components: { PostCard },
  name: 'PostsPanel',

  props: {…
  },

  methods: {
    onLike(cardId) {…
    },
    onDislike(cardId) {…
    },
  }
}
</script>
```

PostCard.vue components U

Дочірній компонент

src > components > PostCard.vue > { } "PostCard.vue"

```vue
<template>
  <div class="container">
    <div class="logo-container">
      <img v-if="cardData.logo" :src="cardData.logo"
      class="logo" />
      <div v-else class="logo">{{ firstLetter }}</div>
    </div>
    <div class="content">
      <div>{{ cardData.userName }}</div>
      <div>{{ cardData.message }}</div>
      <div>
        <button @click="$emit('like',cardData.id)">
          Like
        </button>
        <span>{{ cardData.likes }}</span>
        <button @click="$emit('dislike',cardData.id)">
          DisLike
        </button>
      </div>
    </div>
  </div>
</template>

<script>
export default {…
}
</script>
```

1. Дочірній компонент генерує подію і передає дані

PostsPanel.vue components U

Батьківський компонент

src > components > PostsPanel.vue > { } "PostsPa

```vue
<template>
  <post-card
    v-for="card in postsList"
    :key="card.id"
    :card-data="card"
    @like="onLike"
    @dislike="onDislike"
  />
</template>

<script>
import PostCard from './PostCard.vue'
export default {
  components: { PostCard },
  name: 'PostsPanel',

  props: {…
  },

  methods: {
    onLike(cardId) {…
    },
    onDislike(cardId) {…
    },
  }
}
</script>
```

PostCard.vue components U

Дочірній компонент

src > components > PostCard.vue > { } "PostCard.vue"

```vue
<template>
  <div class="container">
    <div class="logo-container">
      <img v-if="cardData.logo" :src="cardData.logo"
      class="logo" />
      <div v-else class="logo">{{ firstLetter }}</div>
    </div>
    <div class="content">
      <div>{{ cardData.userName }}</div>
      <div>{{ cardData.message }}</div>
      <div>
        <button @click="$emit('like',cardData.id)">
          Like
        </button>
        <span>{{ cardData.likes }}</span>
        <button @click="$emit('dislike',cardData.id)">
          DisLike
        </button>
      </div>
    </div>
  </div>
</template>

<script>
export default {…
}
</script>
```

1. Дочірній компонент генерує подію і передає дані

PostsPanel.vue components U

Батьківський компонент

src > components > PostsPanel.vue > { } "PostsPa

```vue
<template>
  <post-card
    v-for="card in postsList"
    :key="card.id"
    :card-data="card"
    @like="onLike"
    @dislike="onDislike"
  />
</template>

<script>
import PostCard from './PostCard.vue'
export default {
  components: { PostCard },
  name: 'PostsPanel',

  props: {…
  },

  methods: {
    onLike(cardId) {…
    },
    onDislike(cardId) {…
    },
  }
}
</script>
```

2.На дочірній компонент призначаємо обробника

PostCard.vue components U

src > components > PostCard.vue > { } "PostCard.vue"

```vue
<template>
  <div class="container">
    <div class="logo-container">
      <img v-if="cardData.logo" :src="cardData.logo"
      class="logo" />
      <div v-else class="logo">{{ firstLetter }}</div>
    </div>
    <div class="content">
      <div>{{ cardData.userName }}</div>
      <div>{{ cardData.message }}</div>
      <div>
        <button @click="$emit('like',cardData.id)">
          Like
        </button>
        <span>{{ cardData.likes }}</span>
        <button @click="$emit('dislike',cardData.id)">
          DisLike
        </button>
      </div>
    </div>
  </div>
</template>

<script>
export default {…
}
</script>
```

PostsPanel.vue components U

src > components > PostsPanel.vue > { } "PostsPa

```vue
<template>
  <post-card
    v-for="card in postsList"
    :key="card.id"
    :card-data="card"
    @like="onLike"
    @dislike="onDislike"
  />
</template>

<script>
import PostCard from './PostCard.vue'
export default {
  components: { PostCard },
  name: 'PostsPanel',

  props: {…
  },

  methods: {
    onLike(cardId) {…
    },
    onDislike(cardId) {…
    },
  }
}
</script>
```

2.На дочірній компонент призначаємо обробника

3. Викликається обробник події

PostCard.vue components U

src > components > PostCard.vue > { } "PostCard.vue"

```vue
<template>
  <div class="container">
    <div class="logo-container">
      <img v-if="cardData.logo" :src="cardData.logo"
      class="logo" />
      <div v-else class="logo">{{ firstLetter }}</div>
    </div>
    <div class="content">
      <div>{{ cardData.userName }}</div>
      <div>{{ cardData.message }}</div>
      <div>
        <button @click="$emit('like',cardData.id)">
          Like
        </button>
        <span>{{ cardData.likes }}</span>
        <button @click="$emit('dislike',cardData.id)">
          DisLike
        </button>
      </div>
    </div>
  </div>
</template>

<script>
export default {…
}
</script>
```

PostsPanel.vue components U

src > components > PostsPanel.vue > { } "PostsPa

```vue
<template>
  <post-card
    v-for="card in postsList"
    :key="card.id"
    :card-data="card"
    @like="onLike"
    @dislike="onDislike"
  />
</template>

<script>
import PostCard from './PostCard.vue'
export default {
  components: { PostCard },
  name: 'PostsPanel',

  props: {…
  },

  methods: {
    onLike(cardId) {…
    },
    onDislike(cardId) {…
    },
  }
}
</script>
```

2.На дочірній компонент призначаємо обробника

3. Викликається обробник події

4. Обробник події отримує передані дані

v-model на нативному елементі

Як ми використовуємо ***v-model***
для стандартних компонентів ?

```
<input v-model="searchText" />
```

```
data() {
  return {
    searchText: null
  }
},
```

v-model на нативному елементі

Як ми використовуємо ***v-model***
для стандартних компонентів ?

```
<input v-model="searchText" />
```

```
data() {
  return {
    searchText: null
  }
},
```

Насправді тут
1)визначачається значення властивості ***:value***
2)аналізується подія ***@input***

```
<input
  :value="searchText"
  @input="searchText = $event.target.value"
/>
```

```
data() {
  return {
    searchText: null
  }
},
```

v-model для власних компонентів у Vue 2

Власний компонент

```
<template>
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”,
    props:{
    ......
      value:{
        type: … ,
        default: …
      },
    ......
    },
    methods: {
      метод () {
        ......
        this.$emit(‘input’, дані )
        ......
      }
    },
    ......
 }
</script>
<style>  ... </style>
```

1)описуємо власт. *value*
2)при зміні генеруємо подію *input*

Батьківський компонент

```
<template>
  . . . . .
  <дочірній_компонент
     :value = “ модель_даних ”
     @input = “ модель_даних = $event ”
   />
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”
    data: {
      ......
      модель_даних : почткове_значення ;
        ......
      },
      ......
    },
    ......
 }
</script>
<style>  ... </style>
```

При зміні *моделі даних* дані передаються у властивість *value компонента*

v-model для власних компонентів у Vue 2

Власний компонент

```
<template>
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”,
    props:{
    ......
      value:{
        type: … ,
        default: …
      },
    ......
    },
    methods: {
      метод () {
        ......
        this.$emit(‘input’, дані )
      }
    },
    ...
}
</script>
<style>  ... </style>
```

При необідності зміни значення властивості *value компонент* генерує подію *input*

Батьківський компонент

```
<template>
  . . . . .
  <дочірній_компонент
     :value = “ модель_даних ”
     @input = “ модель_даних = $event ”
   />
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”
    data: {
      ......
      модель_даних : почткове_значення ;
        ......
      },
      ......
    },
    ......
 }
</script>
<style>  ... </style>
```

v-model для власних компонентів у Vue 2

Власний компонент

```
<template>
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”,
    props:{
    ......
      value:{
        type: … ,
        default: …
      },
    ......
    },
    methods: {
      метод () {
        .......
        this.$emit(‘input’, дані )
        .......
      }
    },
    ......
 }
</script>
<style>  ... </style>
```

Батьківський компонент

```
<template>
  . . . . .
  <дочірній_компонент v-model = “ модель_даних ”/>
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”
    data: {
      .......
      модель_даних : початкове_значення ;
        .......
      },
      .......
    },
    ......
 }
</script>
<style>  ... </style>
```

v-model – це спрощений запис для:

```
:value = “ модель_даних ”
@input = “ модель_даних = $event ”
```

Модифікатор .sync (аналог v-model) для власних компонентів у Vue 2

Власний компонент

```
<template>
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”,
    props:{
    ......
      назва_властивості:{
        type: … ,
        default: …
      },
    ......
    },
    methods: {
      метод () {
        ......
        this.$emit(
         ‘update: назва_властивості’,
         дані
        )
        ......
      }
    },
    ......
 }
</script>
<style>  ... </style>
```

Батьківський компонент

```
<template>
  . . . . .
  <дочірній_компонент
     :назва_властивості.sync = “ модель_даних ”
  />
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”
    data: {
      ......
      модель_даних : початкове_значення ;
        ......
      },
      ......
    },
    ......
 }
</script>
<style>  ... </style>
```

У Vue 3 вже немає !!!

v-model для власних компонентів у Vue 3

Власний компонент у Vue 2

```
<template>
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”,
    props:{
    ......
      value:{
        type: … ,
        default: …
      },
    ......
    },
    methods: {
      метод () {
        ......
        this.$emit(‘input’, дані )
        ......
      }
    },
    ......
 }
</script>
<style>  ... </style>
```

Власний компонент у Vue 3

```
<template>
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”,
    props:{
    ......
      modelValue:{
        type: … ,
        default: …
      },
    ......
    },
    methods: {
      метод () {
        ......
        this.$emit(‘update:modelValue’, дані )
        ......
      }
    },
    ......
 }
</script>
<style>  ... </style>
```

При переході до Vue 3 змінились назви властвості і події

- назва властивості:
  value → modelValue
- подія:
  input → update:modelValue

v-model для власних компонентів у Vue 3

Власний компонент

```
<template>
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”,
    props:{
    ......
      modelValue:{
        type: … ,
        default: …
      },
    ......
    },
    methods: {
      метод () {
        .......
        this.$emit(‘update:modelValue’, дані )
        .......
      }
    },
    ......
 }
</script>
<style>  ... </style>
```

Батьківський компонент

```
<template>
  . . . . .
  <дочірній_компонент v-model = “ модель_даних ”/>
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”
    data: {
      .......
      модель_даних : почткове_значення ;
        .......
      },
      .......
    },
    ......
 }
</script>
<style>  ... </style>
```

CustomInput.vue components U

src > components > CustomInput.vue > {} "CustomInput.vue" > sc

```vue
<template>
  <label>
    {{ title }}
    <input
      type="text"
      :value="modelValue"
      @input="$emit('update:modelValue',
      $event.target.value)"
    />
  </label>
</template>

<script>
export default {
  name: 'CustomInput',

  props: {
    title: {
      type: String,
      default: '',
    },
    modelValue: {
      type: String,
    },
  },
}
</script>
```

App.vue src M

src > App.vue > {} "App.vue" > style

You, 28 seconds ago | 1 author (You)

```vue
<template>
  <custom-input title="User name" v-model="myValue" />
  <div>User name: {{ myValue }}</div>
</template>

<script>
import CustomInput from './components/CustomInput.vue'

export default {
  name: 'App',

  components: {
    CustomInput,
  },

  data() {
    return {
      myValue: null,
    }
  },
}
</script>

<style> ···
```

User name Ivan

User name: Ivan

v-model для власних компонентів у Vue 3

Власний компонент

```
<template>
  . . . . .
  <input type="text" v-model="currentValue" />
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”,
    props:{
    ......
      modelValue:{
        type: … ,
        default: …
      },
    ......
    },
    computed: {
      currentValue      :     {
       get(){
         return this.modelValue
       },
       set(newValue){
         this.$emit(‘update:modelValue’, newValue )
       }
     }
    },
    ......
 }
</script>
<style>  ... </style>
```

Спрощений запис з використанням ***computed***

Батьківський компонент

```
<template>
  . . . . .
  <дочірній_компонент v-model = “ модель_даних ”/>
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”
    data: {
      ......
      модель_даних : початкове_значення ;
        ......
      },
      ......
    },
    ......
 }
</script>
<style>  ... </style>
```

v-model для власних компонентів у Vue 3

Власний компонент

```
<template>
  . . . . .
  <input type="text" v-model="currentValue" />
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”,
    props:{
    ......
      modelValue:{
        type: … ,
        default: …
      },
    ......
    },
    computed: {
      currentValue        :      {
        get(){
          return this.modelValue
        },
        set(newValue){
          this.$emit(‘update:modelValue’, newValue )
        }
      }
    },
    ......
 }
</script>
<style>  ... </style>
```

Спрощений запис з використанням ***computed***

Батьківський компонент

```
<template>
  . . . . .
  <дочірній_компонент v-model = “ модель_даних ”/>
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”
    data: {
      ......
      модель_даних : початкове_значення ;
        ......
      },
      ......
    },
    ......
 }
</script>
<style>  ... </style>
```

Передача даних дочірньому компоненту

v-model для власних компонентів у Vue 3

Власний компонент

```
<template>
  . . . . .
  <input type="text" v-model="currentValue" />
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”,
    props:{
    ......
      modelValue:{
        type: … ,
        default: …
      },
    ......
    },
    computed: {
      currentValue      :     {
       get(){
         return this.modelValue
       },
       set(newValue){
         this.$emit(‘update:modelValue’, newValue )
       }
     }
    },
    ......
 }
</script>
<style>  ... </style>
```

Спрощений запис з використанням ***computed***

Батьківський компонент

```
<template>
  . . . . .
  <дочірній_компонент v-model = “ модель_даних ”/>
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”
    data: {
      ......
      модель_даних : початкове_значення ;
        ......
      },
      ......
    },
    ......
 }
</script>
<style>  ... </style>
```

Передача даних батьківському компоненту

CustomInput.vue components U

src > components > CustomInput.vue > {} "CustomInput.vue" > template

```vue
<template>
  <label>
    {{ title }}
    <input type="text" v-model="currentValue" />
  </label>
</template>

<script>
export default {
  name: 'CustomInput',

  props: {
    title: {
      type: String,
      default: '',
    },
    modelValue: {
      type: String,
    },
  },

  computed: {
    currentValue: {
      get() {
        return this.modelValue
      },
      set(newVal) {
        this.$emit('update:modelValue', newVal)
      },
    },
  },
}
</script>

<style lang="css" scoped></style>
```

App.vue src M

src > App.vue > {} "App.vue"

You, 5 seconds ago | 1 author (You)

```vue
<template>
  <custom-input title="User name" v-model="myValue" />
  <div>User name: {{ myValue }}</div>
</template>

<script>
import CustomInput from './components/CustomInput.vue'

export default {
  name: 'App',

  components: {
    CustomInput,
  },

  data() {
    return {
      myValue: null,
    }
  },
}
</script>

<style>…
```

CustomInput.vue components U

src > components > CustomInput.vue > {} "CustomInput.vue" > template

```vue
<template>
  <label>
    {{ title }}
    <input type="text" v-model="currentValue" />
  </label>
</template>

<script>
export default {
  name: 'CustomInput',

  props: {
    title: {
      type: String,
      default: '',
    },
    modelValue: {
      type: String,
    },
  },

  computed: {
    currentValue: {
      get() {
        return this.modelValue
      },
      set(newVal) {
        this.$emit('update:modelValue', newVal)
      },
    },
  },
}
</script>

<style lang="css" scoped></style>
```

App.vue src M

src > App.vue > {} "App.vue"

You, 5 seconds ago | 1 author (You)

```vue
<template>
  <custom-input title="User name" v-model="myValue" />
  <div>User name: {{ myValue }}</div>
</template>

<script>
import CustomInput from './components/CustomInput.vue'

export default {
  name: 'App',

  components: {
    CustomInput,
  },

  data() {
    return {
      myValue: null,
    }
  },
}
</script>

<style>…
```

CustomInput.vue components U

src > components > CustomInput.vue > {} "CustomInput.vue" > template

```vue
<template>
  <label>
    {{ title }}
    <input type="text" v-model="currentValue" />
  </label>
</template>

<script>
export default {
  name: 'CustomInput',

  props: {
    title: {
      type: String,
      default: '',
    },
    modelValue: {
      type: String,
    },
  },

  computed: {
    currentValue: {
      get() {
        return this.modelValue
      },
      set(newVal) {
        this.$emit('update:modelValue', newVal)
      },
    },
  },
}
</script>

<style lang="css" scoped></style>
```

App.vue src M

src > App.vue > {} "App.vue"

You, 5 seconds ago | 1 author (You)

```vue
<template>
  <custom-input title="User name" v-model="myValue" />
  <div>User name: {{ myValue }}</div>
</template>

<script>
import CustomInput from './components/CustomInput.vue'

export default {
  name: 'App',

  components: {
    CustomInput,
  },

  data() {
    return {
      myValue: null,
    }
  },
}
</script>

<style>…
```

CustomInput.vue components U

src > components > CustomInput.vue > {} "CustomInput.vue" > template

```vue
<template>
  <label>
    {{ title }}
    <input type="text" v-model="currentValue" />
  </label>
</template>

<script>
export default {
  name: 'CustomInput',

  props: {
    title: {
      type: String,
      default: '',
    },
    modelValue: {
      type: String,
    },
  },

  computed: {
    currentValue: {
      get() {
        return this.modelValue
      },
      set(newVal) {
        this.$emit('update:modelValue', newVal)
      },
    },
  },
}
</script>

<style lang="css" scoped></style>
```

App.vue src M

src > App.vue > {} "App.vue"

You, 5 seconds ago | 1 author (You)

```vue
<template>
  <custom-input title="User name" v-model="myValue" />
  <div>User name: {{ myValue }}</div>
</template>

<script>
import CustomInput from './components/CustomInput.vue'

export default {
  name: 'App',

  components: {
    CustomInput,
  },

  data() {
    return {
      myValue: null,
    }
  },
}
</script>

<style>…
```

CustomInput.vue components U

src > components > CustomInput.vue > {} "CustomInput.vue" > template

```vue
<template>
  <label>
    {{ title }}
    <input type="text" v-model="currentValue" />
  </label>
</template>

<script>
export default {
  name: 'CustomInput',

  props: {
    title: {
      type: String,
      default: '',
    },
    modelValue: {
      type: String,
    },
  },

  computed: {
    currentValue: {
      get() {
        return this.modelValue
      },
      set(newVal) {
        this.$emit('update:modelValue', newVal)
      },
    },
  },
}
</script>

<style lang="css" scoped></style>
```

App.vue src M

src > App.vue > {} "App.vue"

You, 5 seconds ago | 1 author (You)

```vue
<template>
  <custom-input title="User name" v-model="myValue" />
  <div>User name: {{ myValue }}</div>
</template>

<script>
import CustomInput from './components/CustomInput.vue'

export default {
  name: 'App',

  components: {
    CustomInput,
  },

  data() {
    return {
      myValue: null,
    }
  },
}
</script>

<style>…
```

CustomInput.vue components U

src > components > CustomInput.vue > {} "CustomInput.vue" > template

```vue
<template>
  <label>
    {{ title }}
    <input type="text" v-model="currentValue" />
  </label>
</template>

<script>
export default {
  name: 'CustomInput',

  props: {
    title: {
      type: String,
      default: '',
    },
    modelValue: {
      type: String,
    },
  },

  computed: {
    currentValue: {
      get() {
        return this.modelValue
      },
      set(newVal) {
        this.$emit('update:modelValue', newVal)
      },
    },
  },
}
</script>

<style lang="css" scoped></style>
```

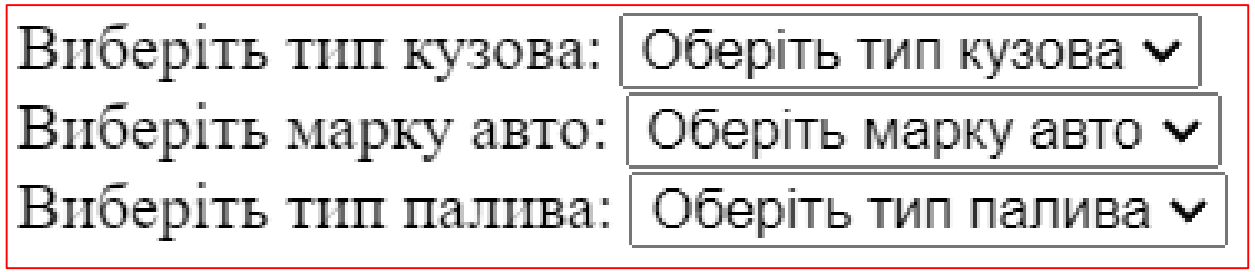
App.vue src M

src > App.vue > {} "App.vue"

You, 5 seconds ago | 1 author (You)

```vue
<template>
  <custom-input title="User name" v-model="myValue" />
  <div>User name: {{ myValue }}</div>
</template>

<script>
import CustomInput from './components/CustomInput.vue'

export default {
  name: 'App',

  components: {
    CustomInput,
  },

  data() {
    return {
      myValue: null,
    }
  },
}
</script>

<style>…
```

Приклад. Розробити компонент у який передається selectorTtitle та список можливих значень для вибору. Компонент повертає вибране значення з використанням v-model. З його використанням організувати ввід марки авто та кузова




```
const carBrands = [
  { id: 1, title: "Toyota" },
  { id: 2, title: "Ford" },
  { id: 3, title: "Honda" },
  { id: 4, title: "Chevrolet" },
  { id: 5, title: "BMW" },
  { id: 6, title: "Audi" },
  { id: 7, title: "Mercedes-Benz" }
];
```

```
const carBodyTypes = [
  { id: 1, title: "Седан" },
  { id: 2, title: "Хетчбек" },
  { id: 3, title: "Купе" },
  { id: 4, title: "Універсал" },
  { id: 5, title: "Спорткар" },
  { id: 6, title: "Кросовер" },
  { id: 7, title: "Мінівен" }
];
```

```
const fuelTypes = [
  { id: 1, title: "Бензин" },
  { id: 2, title: "Дизель" },
  { id: 3, title: "Гібрид" },
  { id: 4, title: "Електрика" },
  { id: 5, title: "Газ" },
];
```

Декілька v-model для власних компонентів у Vue 3. **v-model** з аргументами

Власний компонент

```
<template>
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”,
    props:{
    ......
      властивість_1:{
        type: … ,
        default: …
      },

    ......
    },
    methods: {
      метод_1 () {
        ......
       this.$emit(‘update:властивість_1’, дані1 )
        ......
      },

    },
    ......
 }
</script>
<style>  ... </style>
```

Батьківський компонент

```
<template>
  . . . . .
  <дочірній_компонент
     v-model:властивість_1 = “ модель_даних_1 ”


  />
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”
    data: {
      ......
      модель_даних_1 : почткове_значення ;

        ......
      },
      ......
    },
    ......
 }
</script>
<style>  ... </style>
```

Декілька v-model для власних компонентів у Vue 3. **v-model** з аргументами

Власний компонент

```
<template>
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”,
    props:{
    ......
      властивість_1:{
        type: … ,
        default: …
       },

    ......
    },
    methods: {
      метод_1 () {
       ......
      this.$emit(‘update:властивість_1’, дані1 )
       ......
     },

    },
    ......
 }
</script>
<style>  ... </style>
```

Батьківський компонент

```
<template>
  . . . . .
  <дочірній_компонент
     v-model:властивість_1 = “ модель_даних_1 ”


  />
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”
    data: {
      ......
      модель_даних_1 : почткове_значення
        ......
      },
      ......
    },
    ......
 }
</script>
<style>  ... </style>
```

Декілька v-model для власних компонентів у Vue 3. ***v-model*** з аргументами

Власний компонент

```
<template>
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”,
    props:{
    ......
      властивість_1:{
        type: … ,
        default: …
      },
      властивість_2:{
        type: … ,
        default: …
      },

    ......
    },
    methods: {
      метод_1 () {
        ......
       this.$emit(‘update:властивість_1’, дані1 )
        ......
     },
     метод_2 () {
        ......
       this.$emit(‘update:властивість_2’, дані2 )
        ......
     }
    },
    ......
 }
</script>
<style>  ... </style>
```

Батьківський компонент

```
<template>
  . . . . .
  <дочірній_компонент
     v-model:властивість_1 = “ модель_даних_1 ”

     v-model:властивість_2 = “ модель_даних_2 ”

  />
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”
    data: {
      ......
      модель_даних_1 : початкове_значення ;
      модель_даних_2 : початкове_значення ;
        ......
      },
      ......
    },
    ......
 }
</script>
<style>  ... </style>
```

LoginForm.vue components U

src > components > LoginForm.vue > {} "LoginForm.vue" > template > label

```vue
<template>
  <label>
    Login
    <input
      type="text" :value="login"
      @input="$emit('update:login', $event.target.value)"
    />
  </label>
</template>

<script>
export default {
  name: 'LoginForm',

  props: {
    login: {
      type: String,
      default: '',
    },
  }
}
</script>
```

App.vue src M

src > App.vue > {} "App.vue" > template > login-form

```vue
<template>
  <login-form
    v-model:login="userLogin"
  />
  <hr />
  <div>User data: {{ userLogin }} - {{ userPassword }}</div>
</template>

<script>
import LoginForm from './components/LoginForm.vue'

export default {
  name: 'App',

  components: {
    LoginForm,
  },

  data() {
    return {
      userLogin: null,
    }
  },
}
</script>

<style>...
```

Login Ivan Password 123

User data: Ivan - 123

LoginForm.vue components U

src > components > LoginForm.vue > {} "LoginForm.vue" > template > label

```vue
<template>
  <label>
    Login
    <input
      type="text" :value="login"
      @input="$emit('update:login', $event.target.value)"
    />
  </label>
  <label>
    Password
    <input
      type="text" :value="password"
      @input="$emit('update:password', $event.target.value)
      "
    />
  </label>
</template>

<script>
export default {
  name: 'LoginForm',

  props: {
    login: {
      type: String,
      default: '',
    },
    password: {
      type: String,
      default: '',
    },
  },
}
</script>
```

App.vue src M

src > App.vue > {} "App.vue" > template > login-form

```vue
...
<template>
  <login-form
    v-model:login="userLogin"
    v-model:password="userPassword"
  />
  You, 1 second ago • Uncommitted changes
  <hr />
  <div>User data: {{ userLogin }} - {{ userPassword }}</div>
</template>

<script>
import LoginForm from './components/LoginForm.vue'

export default {
  name: 'App',

  components: {
    LoginForm,
  },

  data() {
    return {
      userLogin: null,
      userPassword: null,
    }
  },
}
</script>

<style>...
```

Login Ivan Password 123

User data: Ivan - 123

LoginForm.vue components U

src > components > LoginForm.vue > {} "LoginForm.vue" > script > default > computed >

```vue
<template>
  <label>
    Login
    <input type="text" v-model="loginValue" />
  </label>
  <label>
    Password
    <input type="text" v-model="passwordValue" />
  </label>
</template>
<script>
export default {
  name: 'LoginForm',
  props: {
    login: { type: String },
    password: { type: String },
  },
  computed: {
    loginValue: {
      get() {
        return this.login
      },
      set(newVal) {
        this.$emit('update:login', newVal)
      },
    },
    passwordValue: {
      get() {
        return this.password
      },
      set(newVal) {
        this.$emit('update:password', newVal)
      },
    },
  },
}
</script>
```

З використанням обчислювальних властивостей

App.vue src M

You, 47 seconds ago | 1 author (You)

```vue
<template>
  <login-form v-model:login="userLogin"
   v-model:password="userPassword" />
  <hr />
  <div>User data: {{ userLogin }} - {{ userPassword }}</div>
</template>

<script>
import LoginForm from './components/LoginForm.vue'

export default {
  name: 'App',

  components: {
    LoginForm,
  },

  data() {
    return {
      userLogin: null,
      userPassword: null,
    }
  },
}
</script>

<style>…
```

LoginForm.vue components U

src > components > LoginForm.vue > {} "LoginForm.vue" > script > default > computed >

```vue
<template>
  <label>
    Login
    <input type="text" v-model="loginValue" />
  </label>
  <label>
    Password
    <input type="text" v-model="passwordValue" />
  </label>
</template>
<script>
export default {
  name: 'LoginForm',
  props: {
    login: { type: String },
    password: { type: String },
  },
  computed: {
    loginValue: {
      get() {
        return this.login
      },
      set(newVal) {
        this.$emit('update:login', newVal)
      },
    },
    passwordValue: {
      get() {
        return this.password
      },
      set(newVal) {
        this.$emit('update:password', newVal)
      },
    },
  },
}
</script>
```

З використанням обчислювальних властивостей

App.vue src M

You, 47 seconds ago | 1 author (You)

```vue
<template>
  <login-form v-model:login="userLogin"
   v-model:password="userPassword" />
  <hr />
  <div>User data: {{ userLogin }} - {{ userPassword }}</div>
</template>

<script>
import LoginForm from './components/LoginForm.vue'

export default {
  name: 'App',

  components: {
    LoginForm,
  },

  data() {
    return {
      userLogin: null,
      userPassword: null,
    }
  },
}
</script>

<style>…
```

LoginForm.vue components U

src > components > LoginForm.vue > {} "LoginForm.vue" > script > default > computed >

```vue
<template>
  <label>
    Login
    <input type="text" v-model="loginValue" />
  </label>
  <label>
    Password
    <input type="text" v-model="passwordValue" />
  </label>
</template>
<script>
export default {
  name: 'LoginForm',
  props: {
    login: { type: String },
    password: { type: String },
  },
  computed: {
    loginValue: {
      get() {
        return this.login
      },
      set(newVal) {
        this.$emit('update:login', newVal)
      },
    },
    passwordValue: {
      get() {
        return this.password
      },
      set(newVal) {
        this.$emit('update:password', newVal)
      },
    },
  },
}
</script>
```

З використанням обчислювальних властивостей

App.vue src M

You, 47 seconds ago | 1 author (You)

```vue
<template>
  <login-form v-model:login="userLogin"
   v-model:password="userPassword" />
  <hr />
  <div>User data: {{ userLogin }} - {{ userPassword }}</div>
</template>

<script>
import LoginForm from './components/LoginForm.vue'

export default {
  name: 'App',

  components: {
    LoginForm,
  },

  data() {
    return {
      userLogin: null,
      userPassword: null,
    }
  },
}
</script>

<style>…
```

LoginForm.vue components U

src > components > LoginForm.vue > {} "LoginForm.vue" > script > default > computed >

```vue
<template>
  <label>
    Login
    <input type="text" v-model="loginValue" />
  </label>
  <label>
    Password
    <input type="text" v-model="passwordValue" />
  </label>
</template>
<script>
export default {
  name: 'LoginForm',
  props: {
    login: { type: String },
    password: { type: String },
  },
  computed: {
    loginValue: {
      get() {
        return this.login
      },
      set(newVal) {
        this.$emit('update:login', newVal)
      },
    },
    passwordValue: {
      get() {
        return this.password
      },
      set(newVal) {
        this.$emit('update:password', newVal)
      },
    },
  },
}
</script>
```

З використанням обчислювальних властивостей

App.vue src M

src > App.vue > {} "App.vue"

You, 47 seconds ago | 1 author (You)

```vue
<template>
  <login-form v-model:login="userLogin"
   v-model:password="userPassword" />
  <hr />
  <div>User data: {{ userLogin }} - {{ userPassword }}</div>
</template>

<script>
import LoginForm from './components/LoginForm.vue'

export default {
  name: 'App',

  components: {
    LoginForm,
  },

  data() {
    return {
      userLogin: null,
      userPassword: null,
    }
  },
}
</script>

<style>…
```

LoginForm.vue components U

src > components > LoginForm.vue > {} "LoginForm.vue" > script > default > computed >

```vue
<template>
  <label>
    Login
    <input type="text" v-model="loginValue" />
  </label>
  <label>
    Password
    <input type="text" v-model="passwordValue" />
  </label>
</template>
<script>
export default {
  name: 'LoginForm',
  props: {
    login: { type: String },
    password: { type: String },
  },
  computed: {
    loginValue: {
      get() {
        return this.login
      },
      set(newVal) {
        this.$emit('update:login', newVal)
      },
    },
    passwordValue: {
      get() {
        return this.password
      },
      set(newVal) {
        this.$emit('update:password', newVal)
      },
    },
  },
}
</script>
```

З використанням обчислювальних властивостей

App.vue src M

src > App.vue > {} "App.vue"

You, 47 seconds ago | 1 author (You)

```vue
<template>
  <login-form v-model:login="userLogin"
   v-model:password="userPassword" />
  <hr />
  <div>User data: {{ userLogin }} - {{ userPassword }}</div>
</template>

<script>
import LoginForm from './components/LoginForm.vue'

export default {
  name: 'App',

  components: {
    LoginForm,
  },

  data() {
    return {
      userLogin: null,
      userPassword: null,
    }
  },
}
</script>

<style>…
```

Приклад. Компонент фільтр дозволяє вводити параметри фільтрації і повертає їх з використанням v-model

Фільтри : Назва [ ] **Рік видання** [ ] **Автор** [ ]

Список книг

Назва книги	Рік видання	Автор
"Тисяча чудес"	2003	Марк Левін
"Шлях до багатства"	1937	Наполеон Хілл
"Гаррі Поттер і філософський камінь"	1997	Джоан Роулінг
"Таємниця старого мосту"	1982	Ерл Стенлі Гарднер
"Маленький принц"	1943	Антуан де Сент-Экзюпери

Використання модифікаторів

Модифікатор	Призначення	Приклад
.lazy	для синхронізації після подій change	`<input v-model.lazy="msg" />`
.number	дані автоматично перетворювалися на числові	`<input v-model.number="age" />`
.trim	видаляються початкові і кінцеві пробіли	`<input v-model.trim="msg" />`

Власний компонент

```
<template>
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”,
    props:{
    ......
      modelValue:{
        type: … ,
        default: …
      },
      modelModifiers: {
        default: () => ({})
      }
    ......
    },
    methods: {
      метод () {
        ......
        . . . . використання . . .
        this.modelModifiers.назва_модифіктора
        ......
        this.$emit(‘update:modelValue’, дані )
        ......
     }
    },
    ......
 }
</script>
<style>  ... </style>
```

Батьківський компонент

```
<template>
  . . . . .
  <дочірній_компонент
     v-model.модифікатор1.модифікатор2.… = “ модель_даних ”
   />
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”
    data: {
      ......
      модель_даних : початкове_значення ;
        ......
      },
      ......
    },
    ......
 }
</script>
<style>  ... </style>
```

Власний компонент

```
<template>
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”,
    props:{
    ......
      modelValue:{
        type: … ,
        default: …
      },
      modelModifiers: {
        default: () => ({})
      }
    ......
    },
    methods: {
      метод () {
        ......
        . . . . використання . . .
        this.modelModifiers.назва_модифіктора
        ......
        this.$emit(‘update:modelValue’, дані )
        ......
      }
    },
    ......
 }
</script>
<style>  ... </style>
```

```
{
  модифікатор1 : true,
  модифікатор2 : true,
 ......
}
```

Батьківський компонент

```
<template>
  . . . . .
  <дочірній_компонент
     v-model.модифікатор1.модифікатор2.… = “ модель_даних ”
   />
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”
    data: {
      ......
      модель_даних : почткове_значення ;
        ......
      },
      ......
    },
    ......
 }
</script>
<style>  ... </style>
```

CustomInput.vue components U

src > components > CustomInput.vue > {} "CustomInput.vue" > script > default > props > title > de

```vue
<template>
  <label>{{ title }}
    <input type="text" v-model="currentValue" />
  </label>
</template>
<script>
export default {
  name: 'CustomInput',
  props: {
    title: { type: String, default: '' },
    modelValue: {
      type: String,
    },
    modelModifiers: {
      default: () => ({}),
    },
  },
  computed: {
    currentValue: {
      get() {
        return this.modelValue
      },
      set(newVal) {
        if (this.modelModifiers.trimStart) newVal = newVal.trimStart()
        if (this.modelModifiers.makeFirstUppercase && newVal) {
          newVal = `${newVal[0].toUpperCase()}${newVal.slice(1)}`
        }
        this.$emit('update:modelValue', newVal)
      },
    },
  },
}
</script>

<style lang="css" scoped></style>
```

```
{
  trimStart: true,
  makeFirstUppercase: true
}
```

App.vue src M

src > App.vue > {} "App.vue" > script > default

You, 1 second ago | 1 author (You)

```vue
<template>
  <custom-input
    title="User name"
    v-model.trimStart.makeFirstUppercase="myValue"
  />
  <div>User name: {{ myValue }}</div>
</template>

<script>
import CustomInput from './components/CustomInput.vue'

export default {
  name: 'App',

  components: {
    CustomInput,
  },

  data() {
    return {
      myValue: null,
    }
  },
}
</script>

<style>…
```

Приклад. Розробити компонент для введення балансу користувача. При введенні кожні 3 розряди потрібно відділяти пробілом

“1234567” => "1 234 567"

Модифікатори для v-model з аргументами

Власний компонент

```
<template>  . . . . .</template>
<script>
export default {
    name: “. . . . . .”,
    props:{
      властивість_1:{
        type: … ,
        default: …
      },
      властивість_1Modifiers: {
        default: () => ({})
      }

    ......
    },
    methods: {
      метод_1 () {
        . . . використовуємо . . .
       this.властивість_1Modifiers.назва_модифіктора
        ......
       this.$emit(‘update:властивість_1’, дані1 )
     },

    ......
 }
</script>
<style>  ... </style>
```

Додаємо до назви властивості закінчення Modifiers

Батьківський компонент

```
<template>
  . . . . .
  <дочірній_компонент
   v-model:властивість_1.модифіктор1_1.модифікатор1_2. …= “ модель_даних_1 ”

  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”
    data: {
      ......
      модель_даних_1 : почткове_значення ;

      ......
    },
    ......
 }
</script>
<style>  ... </style>
```

Модифікатори для v-model з аргументами

Власний компонент

```
<template>  . . . . .</template>
<script>
export default {
    name: “. . . . . .”,
    props:{
      властивість_1:{
        type: … ,
        default: …
      },
      властивість_1Modifiers: {
        default: () => ({})
      }

      властивість_2:{
        type: … ,
        default: …
      },
      властивість_2Modifiers: {
        default: () => ({})
      }
    ......
    },
    methods: {
      метод_1 () {
        . . . використовуємо . . .
       this.властивість_1Modifiers.назва_модифіктора
        ......
       this.$emit(‘update:властивість_1’, дані1 )
     },
      метод_2 () {
        . . . використовуємо . . .
       this.властивість_2Modifiers.назва_модифіктора
        ......
       this.$emit(‘update:властивість_2’, дані2 )
     }
    },
    ......
 }
</script>
<style>  ... </style>
```

Додаємо до назв властивостей закінчення Modifiers

Батьківський компонент

```
<template>
  . . . . .
  <дочірній_компонент
   v-model:властивість_1.модифіктор1_1.модифікатор1_2. …= “ модель_даних_1 ”

   v-model:властивість_2.модифіктор1_1.модифікатор1_2. …= “ модель_даних_2 ”

  />
  . . . . .
</template>
<script>
export default {
    name: “. . . . . .”
    data: {
      ......
      модель_даних_1 : почткове_значення ;

      модель_даних_2 : почткове_значення ;
        ......
      },
      ......
    },
    ......
 }
</script>
<style>  ... </style>
```

LoginForm.vue components U

src > components > LoginForm.vue > {} "LoginForm.vue" > script > default > computed > loginValue

```vue
<template>
  <label> Login <input type="text" v-model="loginValue" /> </label>
  <label> Password <input type="text" v-model="cardValue" /> </label>
</template>
<script>
export default {
  name: 'LoginForm',
  props: {
    login: { type: String },
    loginModifiers: { default: () => ({}) },
    cardNumber: { type: String },
    cardNumberModifiers: { default: () => ({}) },
  },
  computed: {
    loginValue: {
      get() { return this.login},
      set(newVal) {
        if (this.loginModifiers.uppercase) newVal = newVal.toUpperCase()
        this.$emit('update:login', newVal)
      },
    },
    cardValue: {
      get() { return this.cardNumber },
      set(newVal) {
        if (this.cardNumberModifiers.separate4Digits) {
          newVal = newVal.replace(/(\d{4}(?=\S+))/g, '$1 ')
        }
        this.$emit('update:cardNumber', newVal)
      },
    },
  },
}
</script>

<style lang="css" scoped></style>
```

```
{
  uppercase :true
}
```

App.vue src M

src > App.vue > {} "App.vue" > template > login-form

You, 27 minutes ago | 1 author (You)

```vue
<template>
  <login-form
    v-model:login.uppercase="userLogin"
    v-model:cardNumber.separate4Digits="userCreditCard"
  />
  <hr />
  <div>User data: {{ userLogin }} : {{ userCreditCard }}</div>
</template>

<script>
import LoginForm from './components/LoginForm.vue'

export default {
  name: 'App',

  components: {
    LoginForm,
  },

  data() {
    return {
      userLogin: null,
      userCreditCard: null,
    }
  },
}
</script>

<style>…
```

LoginForm.vue components U

src > components > LoginForm.vue > {} "LoginForm.vue" > script > default > computed > loginValue

```vue
<template>
  <label> Login <input type="text" v-model="loginValue" /> </label>
  <label> Password <input type="text" v-model="cardValue" /> </label>
</template>
<script>
export default {
  name: 'LoginForm',
  props: {
    login: { type: String },
    loginModifiers: { default: () => ({}) },
    cardNumber: { type: String },
    cardNumberModifiers: { default: () => ({}) },
  },
  computed: {
    loginValue: {
      get() { return this.login},
      set(newVal) {
        if (this.loginModifiers.uppercase) newVal = newVal.toUpperCase()
        this.$emit('update:login', newVal)
      },
    },
    cardValue: {
      get() { return this.cardNumber },
      set(newVal) {
        if (this.cardNumberModifiers.separate4Digits) {
          newVal = newVal.replace(/(\d{4}(?=\S+))/g, '$1 ')
        }
        this.$emit('update:cardNumber', newVal)
      },
    },
  }
}
</script>

<style lang="css" scoped></style>
```

```
{
  separate4Digits :true
}
```

App.vue src M

src > App.vue > {} "App.vue" > template > login-form

You, 27 minutes ago | 1 author (You)

```vue
<template>
  <login-form
    v-model:login.uppercase="userLogin"
    v-model:cardNumber.separate4Digits="userCreditCard"
  />
  <hr />
  <div>User data: {{ userLogin }} : {{ userCreditCard }}</div>
</template>

<script>
import LoginForm from './components/LoginForm.vue'

export default {
  name: 'App',

  components: {
    LoginForm,
  },

  data() {
    return {
      userLogin: null,
      userCreditCard: null,
    }
  },
}
</script>

<style>…
```

Приклад. Розробити компонент для вводу віку користувача та імені. Робити форматування відповідних input у випадку потожнього поля (червони фон) і вік (вік менший за 18 – колір фону червоний, інакше – зелений).