

Листинг Программного кода сайта <https://go.spark.kz>

(First 20 pages of Listing of Source Code, Arial, Font 10)

```
<template>
  <div class="FileLoader__wrap">
    <label
      class="FileLoader"
      :class="{ 'FileLoader--notempty': value.length > 1, disabled, isSmall, isInvalid }"
      tabindex="0"
    >
      <input
        type="file"
        hidden
        :disabled="disabled"
        :accept="accept"
        :multiple="isMultiSelect"
        @change="emitUpdate"
      />

      <div v-if="isSmall" class="FileLoader__area">
        <div class="area__clip">
          <template v-if="loading">
            <s-loader />
          </template>

          <template v-else-if="value.length">
            <div class="clip__flex">
              {{ loadedFileTitle }}

              <i-close
                v-if="!disabled && value.length > 0"
                class="FileLoader__reset"
                @click.prevent="resetFiles"
              />
            </div>
          </template>

          <template v-else>
            <i-clip />

            <p class="clip__placeholder">
              {{ placeholder }}
            </p>
          </template>
        </div>
      </div>

      <div v-else class="FileLoader__area">
        <template v-if="loading">
          <s-loader />
        </template>
```

```

        <template v-else>
            {{ placeholder }}
        </template>
    </div>
</label>

<div class="FileLoader__list" v-if="value.length > 1">
    <div class="list__item" v-for="(file, i) in value" :key="file">
        <div class="item__name">{{ file.name }}</div>

        <i-close v-if="!disabled" class="item__remove" @click.prevent="removeFile(i)" />
    </div>
</div>
</div>
</template>

<script lang="ts">
import { computed, defineComponent, PropType } from 'vue'

import SLoader from '@common/components/SLoader/index.vue'

// icons
import iClip from '@assets/icons/Clip.svg'
import iClose from '@assets/icons/Close.svg'

export default defineComponent({
  components: { 's-loader': SLoader, iClip, iClose },
  props: {
    value: {
      type: Array as PropType<File[]>,
      default: () => [],
    },
    placeholder: {
      type: String,
      default: 'Прикрепите файл',
    },
    accept: {
      type: String,
      default: "",
    },
    loading: {
      type: Boolean,
      default: false,
    },
    disabled: {
      type: Boolean,
      default: false,
    },
    isSmall: {
      type: Boolean,
      default: false,
    },
  },
})

```

```

    },
    isMultiSelect: {
      type: Boolean,
      default: false,
    },
    isInvalid: {
      type: Boolean,
      default: false,
    },
    },
    loadedFilesPlaceholder: {
      type: String,
      default: null,
    },
    },
  },
  emits: ['update:value', 'file-loaded'],
  setup(props, { emit }) {
    const emitUpdate = (event: InputEvent) => {
      const target = event.target as HTMLInputElement
      const files = target.files

      const array_of_files: File[] = []

      if (!files || !files.length) {
        emit('update:value', array_of_files)
        return
      }

      for (let index = 0; index < files.length; index++) {
        const file = files.item(index)
        if (!file) continue
        array_of_files.push(file)
      }

      if (props.isMultiSelect) {
        const value_files = [...props.value]
        emit('update:value', value_files.concat(array_of_files))
      } else {
        emit('update:value', array_of_files)
      }

      target.value = ""
    }

    const resetFiles = () => {
      emit('update:value', [])
    }

    const removeFile = (file_index: number) => {
      const files = [...props.value]
      files.splice(file_index, 1)

      emit('update:value', files)
    }
  }

```

```

    }

    //

    const loadedFileTitle = computed(() => {
      if (props.value.length) {
        const placeholder = props.loadedFilesPlaceholder || 'Прикреплено'

        if (props.value.length > 1) {
          return `${placeholder}: ${props.value.length}`
        }

        return `${placeholder}: ${props.value[0].name}`
      }

      return props.placeholder
    })

    //

    return { emitUpdate, loadedFileTitle, resetFiles, removeFile }
  },
})
</script>

```

```

<style lang="scss" scoped src="./index.scss" />

```

```

<template>
  <div class="FTLOrderStatus">
    <template v-if="Boolean(statusCode) && Boolean(statusText)">
      <icon-status-waiting
        class="FTLOrderStatus__icon"
        v-if="statusCode === FTL_STATUS_SEARCHING_TOKEN"
      />
      <icon-status-in-progress
        class="FTLOrderStatus__icon spin"
        v-if="statusCode === FTL_STATUS_WAIT_CONFIRM_TOKEN"
      />
      <icon-status-completed
        class="FTLOrderStatus__icon"
        v-if="statusCode === FTL_STATUS_ACCEPTED_TOKEN"
      />
      <icon-status-canceled
        class="FTLOrderStatus__icon"
        v-if="statusCode === FTL_STATUS_CANCELED_TOKEN"
      />

      <span class="FTLOrderStatus__text">
        {{ statusText }}
      </span>
    </template>
  </div>

```

```

        <template v-else> Hem cmamyca </template>
    </div>
</template>

```

```

<script lang="ts">

```

```

import { defineComponent } from 'vue'

```

```

import iStatusWaiting from '@assets/icons/statuses/StatusWaiting.svg'

```

```

import iStatusInProgress from '@assets/icons/statuses/StatusInProgress.svg'

```

```

import iStatusCompleted from '@assets/icons/statuses/StatusCompleted.svg'

```

```

import iStatusCanceled from '@assets/icons/statuses/StatusCanceled.svg'

```

```

import {

```

```

    FTL_STATUS_SEARCHING_TOKEN,

```

```

    FTL_STATUS_WAIT_CONFIRM_TOKEN,

```

```

    FTL_STATUS_ACCEPTED_TOKEN,

```

```

    FTL_STATUS_CANCELED_TOKEN,

```

```

} from '@core/constants/FTL.constants'

```

```

export default defineComponent({

```

```

    components: {

```

```

        'icon-status-waiting': iStatusWaiting,

```

```

        'icon-status-in-progress': iStatusInProgress,

```

```

        'icon-status-completed': iStatusCompleted,

```

```

        'icon-status-canceled': iStatusCanceled,

```

```

    },

```

```

    props: {

```

```

        statusCode: {

```

```

            type: String,

```

```

            default: null,

```

```

        },

```

```

        statusText: {

```

```

            type: String,

```

```

            default: null,

```

```

        },

```

```

    },

```

```

    setup() {

```

```

        return {

```

```

            FTL_STATUS_SEARCHING_TOKEN,

```

```

            FTL_STATUS_WAIT_CONFIRM_TOKEN,

```

```

            FTL_STATUS_ACCEPTED_TOKEN,

```

```

            FTL_STATUS_CANCELED_TOKEN,

```

```

        }

```

```

    },

```

```

})

```

```

</script>

```

```

<style lang="scss" scoped src="./index.scss" />

```

```

<template>

```

```

    <div class="SAdditionalService">

```

```

        <span class="SAdditionalService__label"> {{ label }} </span>

```

```

<div class="SAdditionalService__counter">
  <span class="counter__currency">{{ currency }}</span>
  <div class="counter__controls">
    <icon-vector
      class="controls__control"
      :class="{ disabled: isDecreaseDisabled }"
      @click="decreaseValue"
    />

    <span class="controls__value" :class="{ disabled: isDecreaseDisabled }">
      {{ value }}
    </span>

    <icon-vector
      class="controls__control right"
      :class="{ disabled: isIncreaseDisabled }"
      @click="increaseValue"
    />
  </div>
</div>
</div>
</template>

```

```

<script lang="ts">
import { computed, defineComponent } from 'vue'

```

```

import iVector from '@assets/icons/Vector.svg'

```

```

export default defineComponent({
  components: {
    'icon-vector': iVector,
  },
  props: {
    label: {
      type: String,
      default: 'label',
    },
    currency: {
      type: String,
      default: 'currency',
    },
    value: {
      type: Number,
      default: 0,
    },
    minValue: {
      type: Number,
      default: 0,
    },
    maxValue: {
      type: Number,

```

```

      default: 999,
    },
    step: {
      type: Number,
      default: 1,
    },
  },
  emits: ['change'],
  setup(props, { emit }) {
    const isIncreaseDisabled = computed(() => props.value >= props.maxValue)
    const isDecreaseDisabled = computed(() => props.value <= props.minValue)

    const increaseValue = () => {
      if (isIncreaseDisabled.value) return
      emit('change', props.value + props.step)
    }

    const decreaseValue = () => {
      if (isDecreaseDisabled.value) return
      emit('change', props.value - props.step)
    }

    return { isIncreaseDisabled, isDecreaseDisabled, increaseValue, decreaseValue }
  },
})
</script>

```

```
<style lang="scss" src="./index.scss" />
```

```

<template>
  <div class="AuthFormWrapper">
    <icon-logo class="AuthFormWrapper__logo" />

    <div class="AuthFormWrapper__slot">
      <slot />
    </div>
  </div>
</template>

```

```

<script lang="ts">
import { defineComponent } from 'vue'

import iLogo from '@assets/icons/Logo.svg'

export default defineComponent({
  components: {
    'icon-logo': iLogo,
  },
})
</script>

```

```
<style lang="scss" src="./index.scss" />
```

```

<template>
  <form class="LoginForm" @submit.prevent="emitSignin">
    <h2 class="title">Автоматизация</h2>

    <s-input placeholder="Логин" v-model:value="credentialsState.login">
      <template v-slot:icon>
        <icon-email class="icon-email" />
      </template>
    </s-input>

    <s-password placeholder="Пароль" v-model:value="credentialsState.password">
      <template v-slot:icon>
        <icon-password class="icon-password" />
      </template>
    </s-password>

    <span class="restore-text" @click="emitRestore"> Забыли пароль? </span>

    <div class="LoginForm__actions">
      <router-link to="/register">
        <s-button class="green" type="button"> Зарегистрироваться как Юр. лицо </s-button>
      </router-link>

      <s-button class="green" type="submit" :loading="loading"> Войти </s-button>
    </div>
  </form>
</template>

```

```

<script lang="ts">
import { defineComponent, reactive, toRaw } from 'vue'

import SButton from '@common/components/SButton/index.vue'
import SInput from '@common/components/SInput/index.vue'
import SPasswоrd from '@common/components/SPasswоrd/index.vue'

import iEmail from '@assets/icons/Email.svg'
import iPassword from '@assets/icons/Password.svg'

export default defineComponent({
  components: {
    's-button': SButton,
    's-input': SInput,
    's-password': SPasswоrd,
    'icon-email': iEmail,
    'icon-password': iPassword,
  },
  props: {
    loading: {
      type: Boolean,
      default: false,
    },
  },
})

```



```

},
emits: ['restore', 'signin'],
setup(props, { emit }) {
  const credentialsState: { login: string | null; password: string | null } = reactive({
    login: null,
    password: null,
  })

  const emitSignin = () => {
    if (!credentialsState.login || !credentialsState.password || props.loading) return
    emit('signin', toRaw(credentialsState))
  }

  const emitRestore = () => {
    if (props.loading) return
    emit('restore')
  }

  return { credentialsState, emitSignin, emitRestore }
},
})
</script>

<style lang="scss" scoped src="./index.scss" />

<template>
  <form class="RestoreForm" @submit.prevent="emitSendCode">
    <h2 class="title">Восстановление пароля</h2>

    <p class="form-text">
      Введите свой адрес электронной почты и мы отправим вам письмо с кодом для сброса пароля
    </p>

    <s-input placeholder="E-mail" v-model:value="email">
      <template v-slot:icon>
        <icon-email class="icon-email" />
      </template>
    </s-input>

    <s-button class="green" :loading="loading"> Отправить код </s-button>
    <span class="back-text" @click="emitBack"> Назад </span>
  </form>
</template>

<script lang="ts">
import { defineComponent, ref, unref } from 'vue'

import SInput from '@common/components/SInput/index.vue'
import SButton from '@common/components/SButton/index.vue'

import iEmail from '@assets/icons/Email.svg'

```

```

export default defineComponent({
  components: {
    's-input': SInput,
    's-button': SButton,
    'icon-email': iEmail,
  },
  props: {
    loading: {
      type: Boolean,
      default: false,
    },
  },
  emits: ['send-code', 'back'],
  setup(_, { emit }) {
    const email = ref(null)

    const emitBack = () => {
      emit('back')
    }

    const emitSendCode = () => {
      emit('send-code', unref(email))
    }

    return { emitBack, emitSendCode, email }
  },
})
</script>

```

```

<style lang="scss" scoped src="./index.scss" />

```

```

<script lang="ts">
import { defineComponent, reactive, watch } from 'vue'

```

```

// @types
import { CalculationAdditionalServicePayload } from '@new-services/types/billing'

```

```

// @components
import SCheckbox from '@common/components/SCheckbox/index.vue'
import NewButton from '@common/new-components/NewButton/index.vue'

```

```

export default defineComponent({
  components: {
    SCheckbox,
    NewButton,
  },
  props: {
    setted: {
      type: Boolean,
      default: false,
    },
  },
})

```

```

    disabled: {
      type: Boolean,
      default: false,
    },
  },
  emits: ['reset', 'submit'],
  setup(props, { emit }) {
    const state = reactive({
      hasManipulator: false,
      hasCrane: false,
      hasHydraulicTrolley: false,
      hasLoaders: false,
    })

    const onReset = () => {
      if (props.disabled) return

      state.hasManipulator = false
      state.hasCrane = false
      state.hasHydraulicTrolley = false
      state.hasLoaders = false

      emit('reset')
    }

    const getPayloadService = (code: string): CalculationAdditionalServicePayload => {
      return {
        code,
        value: 0,
        duration: 0,
        costPerHour: 0,
        costTotal: 0,
        paidPricePerHour: 0,
        paidPriceTotal: 0,
      }
    }

    const onSubmit = () => {
      if (props.disabled) return

      const services = [] as CalculationAdditionalServicePayload[]
      if (state.hasManipulator) {
        services.push(getPayloadService('manipulator'))
      }
      if (state.hasCrane) {
        services.push(getPayloadService('crane'))
      }
      if (state.hasHydraulicTrolley) {
        services.push(getPayloadService('hydraulic_trolley'))
      }
      if (state.hasLoaders) {
        services.push(getPayloadService('loader'))
      }
    }
  }
}

```

```

    }

    emit('submit', services)
  }

  watch(
    () => props.setted,
    (is_setted: boolean) => {
      if (!is_setted) {
        state.hasManipulator = false
        state.hasCrane = false
        state.hasHydraulicTrolley = false
        state.hasLoaders = false
      }
    }
  )

  return {
    state,
    onReset,
    onSubmit,
  }
},
})
</script>

```

```

<template>
  <div class="AdditionalServicesOnDelivery">
    <div class="AdditionalServicesOnDelivery__title">Дополнительные услуги при
    доставке</div>
    <div class="AdditionalServicesOnDelivery__hint">(не обязательно)</div>

    <div class="AdditionalServicesOnDelivery__content">
      <s-checkbox
        :disabled="disabled || setted"
        v-model:value="state.hasManipulator"
        theme="primary"
        label="Манипулятор"
      />

      <s-checkbox
        :disabled="disabled || setted"
        v-model:value="state.hasCrane"
        theme="primary"
        label="Кран"
      />

      <s-checkbox
        :disabled="disabled || setted"
        v-model:value="state.hasHydraulicTrolley"
        theme="primary"
        label="Рохля"
      />
    </div>
  </div>

```

```

    />

    <s-checkbox
      :disabled="disabled || setted"
      v-model: value="state.hasLoaders"
      theme="primary"
      label="Грузчику"
    />
  </div>

  <div class="AdditionalServicesOnDelivery__actions">
    <new-button
      v-if="setted"
      :disabled="disabled"
      class="button--primary button--small button--outline"
      @click="onReset"
    >
      Сбросить
    </new-button>

    <new-button
      v-else
      :disabled="disabled"
      class="button--primary button--small"
      @click="onSubmit"
    >
      Подтвердить
    </new-button>
  </div>
</div>
</template>

<style lang="scss" scoped src="./index.scss" />

<script lang="ts">
import { defineComponent, reactive, watch } from 'vue'

// @types
import { CalculationAdditionalServicePayload } from '@new-services/types/billing'

// @components
import SCheckbox from '@common/components/SCheckbox/index.vue'
import NewButton from '@common/new-components/NewButton/index.vue'

export default defineComponent({
  components: {
    SCheckbox,
    NewButton
  },
  props: {
    setted: {
      type: Boolean,

```

```

        default: false,
    },
    disabled: {
        type: Boolean,
        default: false,
    }
},
emits: ['reset', 'submit'],
setup(props, { emit }) {
    const state = reactive({
        hasManipulator: false,
        hasCrane: false,
        hasHydraulicTrolley: false,
        hasLoaders: false,
    })

    const onReset = () => {
        if (props.disabled) return

        state.hasManipulator = false
        state.hasCrane = false
        state.hasHydraulicTrolley = false
        state.hasLoaders = false

        emit('reset')
    }

    const getPayloadService = (code: string): CalculationAdditionalServicePayload => {
        return { code, value: 0, duration: 0, costPerHour: 0, costTotal: 0, paidPricePerHour: 0,
paidPriceTotal: 0 }
    }

    const onSubmit = () => {
        if (props.disabled) return

        const services = [] as CalculationAdditionalServicePayload[]
        if (state.hasManipulator) { services.push(getPayloadService('manipulator')) }
        if (state.hasCrane) { services.push(getPayloadService('crane')) }
        if (state.hasHydraulicTrolley) { services.push(getPayloadService('hydraulic_trolley')) }
        if (state.hasLoaders) { services.push(getPayloadService('loader')) }

        emit('submit', services)
    }

    watch(() => props.setted, (is_setted: boolean) => {
        if (!is_setted) {
            state.hasManipulator = false
            state.hasCrane = false
            state.hasHydraulicTrolley = false
            state.hasLoaders = false
        }
    })
})

```

```

    return {
      state,
      onReset,
      onSubmit
    }
  }
})
</script>

<template>
  <div class="AdditionalServicesOnTake">
    <div class="AdditionalServicesOnTake__title">Дополнительные услуги при заборе</div>
    <div class="AdditionalServicesOnTake__hint">(не обязательно)</div>

    <div class="AdditionalServicesOnTake__content">
      <s-checkbox
        :disabled="disabled || setted"
        v-model:value="state.hasManipulator"
        theme="primary"
        label="Манипулятор"
      />

      <s-checkbox
        :disabled="disabled || setted"
        v-model:value="state.hasCrane"
        theme="primary"
        label="Кран"
      />

      <s-checkbox
        :disabled="disabled || setted"
        v-model:value="state.hasHydraulicTrolley"
        theme="primary"
        label="Рохля"
      />

      <s-checkbox
        :disabled="disabled || setted"
        v-model:value="state.hasLoaders"
        theme="primary"
        label="Грузчику"
      />
    </div>

    <div class="AdditionalServicesOnTake__actions">
      <new-button
        v-if="setted"
        :disabled="disabled"
        class="button--primary button--small button--outline"
        @click="onReset"
      >

```

```

        Сбросить
    </new-button>

    <new-button
      v-else
      :disabled="disabled"
      class="button--primary button--small"
      @click="onSubmit"
    >
      Подтвердить
    </new-button>
  </div>
</div>
</template>

<style lang="scss" scoped src="./index.scss" />
<script lang="ts">
import { defineComponent, PropType } from 'vue'

import { PAYMENT_METHOD_OPTIONS } from '@core/constants/common.constants'
import { OptionT } from '@core/types/common.types'

export default defineComponent({
  props: {
    value: {
      type: [Number, null] as PropType<number | null>,
      default: null
    },
    disabled: {
      type: Boolean,
      default: false,
    },
    disabledOptionsValues: {
      type: Array as PropType<number[]>,
      default: () => []
    }
  },
  emits: ['update:value', 'change'],
  setup(props, { emit }) {
    const isOptionSelected = (option: OptionT<number>) => {
      return option.value === props.value
    }

    const onOptionSelect = (option: OptionT<number>) => {
      if (props.disabled) return
      if (isOptionSelected(option)) return

      emit('update:value', option.value)
      emit('change', option.value)
    }

    const isOptionDisabled = (option: OptionT<number>) => {

```



```

        return props.disabledOptionsValues.includes(option.value)
    }

    return {
        isOptionSelected,
        isOptionDisabled,
        onOptionSelect,
        PAYMENT_METHOD_OPTIONS
    }
}
})
</script>

```

```

<template>
  <div class="PaymentMethod" :class="{ disabled }">
    <div class="PaymentMethod__title">Выберите способ оплаты</div>

    <div class="PaymentMethod__options">
      <div
        v-for="payment_method in PAYMENT_METHOD_OPTIONS"
        :key="payment_method.id"
        class="options__option"
        :class="{
          'option--selected': isOptionSelected(payment_method),
          'option--disabled': isOptionDisabled(payment_method),
        }"
        @click="onOptionSelect(payment_method)"
      >
        <svg width="14" height="14" viewBox="0 0 14 14" fill="none"
xmlns="http://www.w3.org/2000/svg">
          <circle class="option__circle" cx="7" cy="7" r="5" fill="currentColor"/>
          <circle class="option__border" cx="7" cy="7" r="6.5" stroke="currentColor"/>
        </svg>

        <div class="option__name">
          {{ payment_method.name }}
        </div>
      </div>
    </div>
  </div>
</template>

```

```

<style lang="scss" scoped src="./index.scss" />
<script lang="ts">
import { defineComponent, PropType } from 'vue'

import { PAYMENT_TYPE_OPTIONS } from '@core/constants/common.constants'
import { OptionT } from '@core/types/common.types'

export default defineComponent({
  props: {

```

```

    value: {
      type: [Number, null] as PropType<number | null>,
      default: null
    },
    disabled: {
      type: Boolean,
      default: false,
    },
    payersLength: {
      type: Number,
      default: 0
    },
    disabledOptionsValues: {
      type: Array as PropType<number[]>,
      default: () => []
    },
    senderCompanyName: {
      type: String,
      default: null
    }
  },
  emits: ['update:value', 'change'],
  setup(props, { emit }) {
    const isOptionSelected = (option: OptionT<number>) => {
      return option.value === props.value
    }

    const onOptionSelect = (option: OptionT<number>) => {
      if (props.disabled) return
      if (isOptionSelected(option)) return

      emit('update:value', option.value)
      emit('change', option.value)
    }

    const isOptionDisabled = (option: OptionT<number>) => {
      return props.disabledOptionsValues.includes(option.value)
    }

    return {
      isOptionSelected,
      onOptionSelect,
      isOptionDisabled,
      PAYMENT_TYPE_OPTIONS
    }
  }
})
</script>

```

```

<template>
  <div class="PaymentType" :class="{ disabled }">

```

```

<div class="PaymentType__title">Кем будем производиться оплата</div>

<div class="PaymentType__options">
  <div
    v-for="payment_type in PAYMENT_TYPE_OPTIONS"
    :key="payment_type.id"
    class="options__option"
    :class="{
      'option--selected': isOptionSelected(payment_type),
      'option--disabled': isOptionDisabled(payment_type),
    }"
    @click="onOptionSelect(payment_type)"
  >
    <svg width="14" height="14" viewBox="0 0 14 14" fill="none"
xmlns="http://www.w3.org/2000/svg">
      <circle class="option__circle" cx="7" cy="7" r="5" fill="currentColor"/>
      <circle class="option__border" cx="7" cy="7" r="6.5" stroke="currentColor"/>
    </svg>

    <div class="option__name">
      <template v-if="payment_type.value === 1 && Boolean(senderCompanyName)">
        {{ payment_type.name }} {{ senderCompanyName }}
      </template>

      <template v-else>
        {{ payment_type.name }}
      </template>
    </div>
  </div>
</div>

<div v-if="!payersLength" class="PaymentType__alert">
  Для оплаты третьим лицом или плательщиком обратитесь к вашему менеджеру
</div>
</div>
</template>

<style lang="scss" scoped src="./index.scss" />
<template>
  <div class="AdditionalService">
    <label class="AdditionalService__label">
      {{ label }}
    </label>

    <div class="AdditionalService__counter">
      <p class="counter__currency">
        {{ currency }}<sup v-if="sup > 1">{{ sup }}</sup>
      </p>
      <div class="counter__controls">
        <icon-vector
          class="controls__control"
          :class="{ disabled: isDecreaseDisabled }"

```

```

        @click="decreaseValue"
      />
      <span class="controls__value" :class="{ disabled: isDecreaseDisabled }">
        {{ value }}
      </span>
      <icon-vector
        class="controls__control right"
        :class="{ disabled: isIncreaseDisabled }"
        @click="increaseValue"
      />
    </div>
  </div>
</div>
</template>

```

```

<script lang="ts">
import { computed, defineComponent } from 'vue'

```

```

import iVector from '@assets/icons/Vector.svg'

```

```

export default defineComponent({
  components: {
    'icon-vector': iVector,
  },
  props: {
    label: {
      type: String,
      default: 'label',
    },
    currency: {
      type: String,
      default: 'currency',
    },
    value: {
      type: Number,
      default: 0,
    },
    maxValue: {
      type: Number,
      default: 999,
    },
    minValue: {
      type: Number,
      default: 0,
    },
    step: {
      type: Number,
      default: 1,
    },
    sup: {
      type: Number,
      default: 0,
    },
  },
})

```

```

    },
  },
  emits: ['update:value'],
  setup(props, { emit }) {
    const isIncreaseDisabled = computed(() => props.value >= props.maxValue)
    const isDecreaseDisabled = computed(() => props.value <= props.minValue)

    const increaseValue = () => {
      if (isIncreaseDisabled.value) return
      emit('update:value', props.value + props.step)
    }

    const decreaseValue = () => {
      if (isDecreaseDisabled.value) return
      emit('update:value', props.value - props.step)
    }

    return { isIncreaseDisabled, isDecreaseDisabled, increaseValue, decreaseValue }
  },
})
</script>

<style lang="scss" src="./index.scss" />

```