Input组件

一.定义组件所需`Props`

> components/input/src/input.ts

```
import { ExtractPropTypes, PropType } from 'vue'
import { isString } from '@vue/shared'
export const inputProps = {
  modelValue: {
    // v-model绑定的属性
    type: [Number, String] as PropType<number | string>,
    default: ''
  },
  type: { type: String, default: 'text' }, // 输入框类型
  placeholder: { type: String }, // 输入提示
  disabled: { type: Boolean, default: false }, // 是否禁用
  readonly: { type: Boolean, default: false }, // 是否仅读
  clearable: { type: Boolean }, // 是否带有清空按钮
  showPassword: { type: Boolean, default: false }, // 密码框是否显示密码
  label: { type: String } // input配合的label属性
} as const
export type InputProps = ExtractPropTypes<typeof inputProps>

export const inputEmits = {
  'update:modelValue': (value: string) => isString(value),
  input: (value: string) => isString(value),
  change: (value: string) => isString(value),
  focus: (evt: FocusEvent) => evt instanceof FocusEvent,
  blur: (evt: FocusEvent) => evt instanceof FocusEvent,
  keydown: (evt: KeyboardEvent) => evt instanceof Event, // input相关事件
  clear: () => true // 清空事件
}
export type InputEmits = typeof inputEmits
```

二.`input.vue`结构实现

```
<template>
  <div :class="[bem.b(), bem.is('disabled', disabled)]">
```

```
    <div v-if="slots.prepend" :class="[bem.be('group', 'prepend')]">
      <slot name="prepend"></slot>
    </div>
    <div :class="[bem.e('wrapper')]">
      <span v-if="slots.prefixIcon" :class="bem.e('prefix')">
        <slot name="prefixIcon" />
      </span>
      <input type="text" v-bind="attrs" :class="bem.e('inner')" />
      <span v-if="slots.suffixIcon" :class="bem.e('suffix')">
        <slot name="suffixIcon" />
      </span>
    </div>
    <div v-if="slots.prepend" :class="[bem.be('group', 'append')]">
      <slot name="append"></slot>
    </div>
  </div>
</template>

<script lang="ts" setup>
import { createNamespace } from '@zi-shui/utils/create'
import { inputProps } from './input'
import { useSlots, useAttrs } from 'vue'

defineOptions({
  name: 'ZInput',
  inheritAttrs: false
})
// 获取属性,及事件
const bem = createNamespace('input')
const props = defineProps(inputProps)
const slots = useSlots() // 插槽
const attrs = useAttrs() // 属性
</script>
```

三.Input入口编写

```
import { withInstall } from '@zi-shui/utils/withInstall'
import _Input from './src/input.vue'

export const Input = withInstall(_Input)
export default Input

export * from './src/input'

declare module 'vue' {
  export interface GlobalComponents {
    ZInput: typeof Input
  }
}
```

四.Input组件试用

```
<ZInput>
  <template #prepend> 珠峰 </template>
  <template #append> 子水 </template>
  <template #prefixIcon>
    <ZIcon color="#ccc" size="20">
      <Heart></Heart>
    </ZIcon>
  </template>
  <template #suffixIcon>
```

```
    <ZIcon color="#ccc" size="20">
      <Heart></Heart>
    </ZIcon>
  </template>
</ZInput>
```

五.组件功能实现

1).样式编写

```
@use 'mixins/mixins' as *;
@use 'common/var' as *;
@include b(input) {
  // z-input
  position: relative;
  display: inline-flex;
  align-items: center;
  width: 100%;
  @include e(wrapper) {
    box-sizing: border-box;
    display: inline-flex;
    border: 1px solid #ccc;
    flex: 1;
    align-items: center;
    position: relative;
  }
```

```scss
  @include e(suffix) {
    display: inline-flex;
    margin-right: 5px;
  }
  @include e(prefix) {
    display: inline-flex;
    margin-left: 5px;
  }
  @include e(inner) {
    display: inline-flex;
    outline: none;
    border: none;
    background: none;
    appearance: none;
    padding: 0 10px;
    width: 100%;
    height: 30px;
  }
} // z-input__suffix

// 增加前后元素 变成网格  让前后元素 跑到一行去
@include b(input-group) {
  @include e(append) {
    display: inline-flex;
    padding: 0 10px;
    line-height: 32px;
    box-shadow: 0 0 0 1px #ccc inset, 0 1px 0
0#ccc inset, 0 -1px 0 0 #ccc inset;
  }
  @include e(prepend) {
    display: inline-flex;
    padding: 0 10px;
    line-height: 32px;
    box-shadow: 1px 0 0 0 #ccc inset, 0 1px 0
0#ccc inset, 0 -1px 0 0 #ccc inset;
  }
}
```

2).实现双向绑定

```
<input
  v-bind="attrs"
  ref="input"
  :class="bem.e('inner')"
  :type="showPassword ? (passwordVisible ?
'text' : 'password') : type"
  :placeholder="placeholder"
  :readonly="readonly"
  :disabled="disabled"
  @input="handleInput"
  @change="handleChange"
/>
```

```
watch( // 监控值的变化，设置输入框的值
  () => props.modelValue,
  () => {
    setNativeInputValue()
  }
)
const setNativeInputValue = () => {
  const inputEle = input.value
  if (!inputEle) return
  inputEle.value = String(props.modelValue); // 设置输入框的内容
}
const handleInput = async (event: Event) => { // 绑定input事件
  const value = (event.target as HTMLInputElement).value
  emit('input', value)
  emit('update:modelValue', value)
}
const handleChange = event => {
  // 处理事件变化
  emit('change', event.target.value)
}
onMounted(() => {
  setNativeInputValue()
})
```

3).Focus & Blur & Keydown

```
<input
  v-bind="attrs"
  ref="input"
  :class="bem.e('inner')"
  :type="showPassword ? (passwordVisible ?
'text' : 'password') : type"
  :placeholder="placeholder"
  :readonly="readonly"
  :disabled="disabled"
  @input="handleInput"
  @change="handleChange"
  @focus="handleFocus"
  @blur="handleBlur"
  @keydown="handleKeydown"
/>
```

```
const handleFocus = (event: FocusEvent) => {
  emit('focus', event)
}
const handleBlur = (event: FocusEvent) => {
  emit('blur', event)
}
const handleKeydown = (evt: KeyboardEvent) => {
  emit('keydown', evt)
}
```

4).切换密码显示

```
const passwordVisible = ref(false);
const focus = async function () {
  await nextTick()
  input.value?.focus()
}
const handlePasswordVisible = () => { // 切换输入框显示内容
  passwordVisible.value = !passwordVisible.value
  focus(); // 获取焦点
}
// 显示后icon，用户提供suffixIcon 需要切换密码显示，需要触发清除功能
const suffixVisible = computed(() => {
  return slots.suffixIcon || props.showPassword
|| props.clearable
})
const showPwdVisible = computed(
  // 显示密码按钮
  () => props.showPassword && !props.disabled &&
!props.readonly && props.modelValue
)
```

```
<span v-if="suffixVisible"
:class="bem.e('suffix')">
  <slot name="suffixIcon"> </slot>
  <ZIcon v-if="showPwdVisible"
@click="handlePasswordVisible">
    <svg
         v-if="passwordVisible"
         >...</svg>
    <svg
         v-else
         >...</svg>
  </ZIcon>
</span>
```

5).显示清除按钮

```
const showClear = computed(
  // 显示清除按钮
  () => props.clearable && !props.readonly &&
!props.disabled && props.modelValue
)
const clear = () => {
  // 清除实现
  emit('update:modelValue', '')
  emit('change', '')
  emit('clear')
}
```

```
<ZIcon v-if="showClear" @click="clear">
  <svg></svg>
</ZIcon>
```