Props in Vue

Props (short for "properties") are how you pass data from a parent component to a child component.

Similer like function parameters, but for components.

Example : Want to send name, age, skills, etc., to a component that shows a user profile. You do that using props.

Parent component: App.vue

```
<template>
  <UserProfile
    name="John Doe"
    :age="30"
    :isActive="true"
    :skills="['Vue', 'React']"
    :location="{ city: 'Bangalore', country: 'India' }"
    :joinedDate="'2023-12-01'"
    status="active"
  />
</template>

<script setup>
import UserProfile from './components/UserProfile.vue'
</script>
```

Child component: UserProfile.vue

```
<script setup>
const props = defineProps({
  name: {
    type: String,
    required: true // This prop must be passed
  },
  age: {
    type: Number,
    default: 25 // If not passed, default is 25
  },
  isActive: {
    type: Boolean,
    default: false
  },
  skills: {
    type: Array,
    default: () => ['Vue', 'JS']
  },
  location: {
    type: Object,
    default: () => ({ city: 'Unknown', country: 'Unknown' })
  },
  joinedDate: {
    type: [String, Date], // Accepts string or Date
    default: () => new Date().toISOString()
  },
  status: {
    type: String,
    validator(value) {
      return ['active', 'inactive', 'suspended'].includes(value)
    },
    default: 'active'
  }
})
</script>

<template>
  <div>
    <h2>{{ props.name }}</h2>
    <p>Age: {{ props.age }}</p>
    <p>Location: {{ props.location.city }}, {{ props.location.country }}</p>
    <p>Status: {{ props.status }}</p>

    <ul>
      <li v-for="skill in props.skills" :key="skill">{{ skill }}</li>
    </ul>

    <p v-if="props.isActive">Active user</p>
    <p v-else>Inactive user</p>

    <p>Joined on: {{ props.joinedDate }}</p>
  </div>
</template>
```

props declaration types

props can be declared in diffrent ways

```
// simple way
defineProps(['name', 'age'])


// with type checking
defineProps({
  name: String,
  age: Number
})

// with full options : type, required, default, validator
defineProps({
  name: { type: String, required: true },
  age: { type: Number, default: 25 }
})
```

props with diffrent data types

1. String

The name prop should be a non-empty string with only letters and spaces. If no value is passed, it defaults to 'Anonymous'.

```
defineProps({
  name: {
    type: String,
    validator(value) {
      // Ensure name is a non-empty string and doesn't contain special characters
      return /^[A-Za-z ]+$/.test(value);
    },
    default: 'Anonymous'
  }
})
```

Type: String, Validator: Ensures the name only contains alphabetic characters and spaces and Default: 'Anonymous'

2. Number

The age prop should be a number between 0 and 120. If no value is passed, it defaults to 30.

```
defineProps({
  age: {
    type: Number,
    validator(value) {
      // Ensure the age is between 0 and 120
      return value >= 0 && value <= 120;
    },
    default: 30
  }
})
```

Type: Number, Validator: Ensures the number is between 0 and 120 and Default: 30

3. Boolean

The isActive prop should strictly be a boolean (true or false). If not provided, it defaults to false.

```
defineProps({
  isActive: {
    type: Boolean,
    validator(value) {
      // Ensure isActive is either true or false (explicit check)
      return typeof value === 'boolean';
    },
    default: false
  }
})
```

4. Array

Ensures every item in the array is a string.

The skills prop should be an array containing only strings. If no array is provided, the default is ['Vue', 'JavaScript'].

```
defineProps({
  skills: {
    type: Array,
    validator(value) {
      // Ensure all skills are strings
      return value.every(skill => typeof skill === 'string');
    },
    default: () => ['Vue', 'JavaScript']
  }
})
```

5. Object

```
defineProps({
  location: {
    type: Object,
    validator(value) {
      // Ensure the object has 'city' and 'country' properties, both strings
      return value && typeof value.city === 'string' && typeof value.country ===
'string';
    },
    default: () => ({ city: 'Unknown', country: 'Unknown' })
  }
})
```

Ensures the object has city and country properties, both of which should be strings.

Default: { city: 'Unknown', country: 'Unknown' }

The location prop should be an object with properties city and country that are both strings. If no object is provided, it defaults to { city: 'Unknown', country: 'Unknown' }.

6. Multiple Types (Union)

The joinedDate prop can either be a string or a Date object. If no value is provided, it defaults to the current date in ISO format.

```
defineProps({
  joinedDate: {
    type: [String, Date],
    validator(value) {
      // Ensure it's either a string or a valid Date
      return typeof value === 'string' || value instanceof Date;
    },
    default: () => new Date().toISOString()
  }
})
```

Type: [String, Date] (Union Type), Validator: Ensures the value is either a string or a Date and Default: Current date as ISO string.

7. Custom Validator for String Enum

The status prop must be a string from a defined set of values. If not passed, it defaults to 'active'.

```
defineProps({
  status: {
    type: String,
    validator(value) {
      // Ensure status is one of 'active', 'inactive', or 'banned'
      return ['active', 'inactive', 'banned'].includes(value);
    },
    default: 'active'
  }
})
```

Type: String, Validator: Ensures the value is one of the allowed options ('active', 'inactive', 'banned') and Default: 'active'

8. Default Value for Arrays or Objects

This is an array with default values ['dark', 'English']. Always use a function for default arrays/objects to avoid shared references between components.

```
defineProps({
  preferences: {
    type: Array,
    default: () => ['dark', 'English']
  }
})
```

Type: Array, Default: ['dark', 'English']