SVELTE • RUNES

$bindable

Ordinarily, props go one way, from parent to child. This makes it easy to understand how data flows around your app.

In Svelte, component props can be *bound*, which means that data can also flow *up* from child to parent. This isn't something you should do often, but it can simplify your code if used sparingly and carefully.

It also means that a state proxy can be *mutated* in the child.

> Mutation is also possible with normal props, but is strongly discouraged — Svelte will warn you if it detects that a component is mutating state it does not 'own'.

To mark a prop as bindable, we use the `$bindable` rune:

FancyInput.svelte

```
<script>
  let { value = $bindable(), ...props } = $props();
</script>

<input bind:value={value} {...props} />

<style>
  input {
    font-family: 'Comic Sans MS';
    color: deeppink;
  }
</style>
```

Now, a component that uses `<FancyInput>` can add the `bind:` directive (demo):

```
<script>
  import FancyInput from './FancyInput.svelte';

  let message = $state('hello');
</script>

<FancyInput bind:value={message} />
<p>{message}</p>
```

The parent component doesn't *have* to use `bind:` — it can just pass a normal prop. Some parents don't want to listen to what their children have to say.

In this case, you can specify a fallback value for when no prop is passed at all:

FancyInput.svelte

```
let { value = $bindable('fallback'), ...props } = $props();
```

Edit this page on GitHub

PREVIOUS
$props

NEXT
$inspect