SVELTE • LEGACY APIS

# export let

ON THIS PAGE

In runes mode, component props are declared with the `$props` rune, allowing parent components to pass in data.

In legacy mode, props are marked with the `export` keyword, and can have a default value:

```
<script>
  export let foo;
  export let bar = 'default value';

  // Values that are passed in as props
  // are immediately available
  console.log({ foo });
</script>
```

The default value is used if it would otherwise be `undefined` when the component is created.

> Unlike in runes mode, if the parent component changes a prop from a defined value to `undefined`, it does not revert to the initial value.

Props without default values are considered *required*, and Svelte will print a warning during development if no value is provided, which you can squelch by specifying `undefined` as the default value:

```
export let foo = undefined;
```

An exported `const`, `class` or `function` declaration is *not* considered a prop — instead, it becomes part of the component's API:

Greeter.svelte

```
<script>
  export function greet(name) {
    alert(`hello ${name}!`);
  }
</script>
```

App.svelte

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

  let greeter;
</script>

<Greeter bind:this={greeter} />

<button on:click={() => greeter.greet('world')}>
  greet
</button>
```

# Renaming props

The `export` keyword can appear separately from the declaration. This is useful for renaming props, for example in the case of a reserved word:

App.svelte JS TS

```
<script lang="ts">
  let className: string;

  // creates a `class` property, even
  // though it is a reserved word
```

PREVIOUS
Reactive $: statements

NEXT
$$props and $$restProps