SVELTE • RUNES

$derived

ON THIS PAGE

Derived state is declared with the `$derived` rune:

```
<script>
  let count = $state(0);
  let doubled = $derived(count * 2);
</script>

<button onclick={() => count++}>
  {doubled}
</button>

<p>{count} doubled is {doubled}</p>
```

The expression inside `$derived(...)` should be free of side-effects. Svelte will disallow state changes (e.g. `count++`) inside derived expressions.

As with `$state` , you can mark class fields as `$derived` .

> Code in Svelte components is only executed once at creation. Without the `$derived` rune, `doubled` would maintain its original value even when `count` changes.

$derived.by

Sometimes you need to create complex derivations that don't fit inside a short expression. In these cases, you can use `$derived.by` which accepts a function as its argument.

```
    for (const n of numbers) {
      total += n;
    }
    return total;
  });
</script>

<button onclick={() => numbers.push(numbers.length + 1)}>
  {numbers.join(' + ')} = {total}
</button>
```

In essence, `$derived(expression)` is equivalent to `$derived.by(() => expression)`.

Understanding dependencies

Anything read synchronously inside the `$derived` expression (or `$derived.by` function body) is considered a *dependency* of the derived state. When the state changes, the derived will be marked as *dirty* and recalculated when it is next read.

To exempt a piece of state from being treated as a dependency, use `untrack`.

Edit this page on GitHub

PREVIOUS
$state

NEXT
$effect