DOM event handlers can have *modifiers* that alter their behaviour. For example, a handler with a `once` modifier will only run a single time:

```
<script>
    function handleClick() {
        alert('no more alerts')
    }
</script>

<button on:click|once={handleClick}>
    Click me
</button>
```

The full list of modifiers:

- `preventDefault` — calls `event.preventDefault()` before running the handler. Useful for client-side form handling, for example.
- `stopPropagation` — calls `event.stopPropagation()`, preventing the event reaching the next element
- `passive` — improves scrolling performance on touch/wheel events (Svelte will add it automatically where it's safe to do so)
- `nonpassive` — explicitly set `passive: false`
- `capture` — fires the handler during the *capture* phase instead of the *bubbling* phase (MDN docs)
- `once` — remove the handler after the first time it runs
- `self` — only trigger handler if event.target is the element itself
- `trusted` — only trigger handler if `event.isTrusted` is `true`. I.e. if the event is triggered by a user action.

You can chain modifiers together, e.g. `on:click|once|capture={...}`.