SVELTE • LEGACY APIS

Reactive $: statements

ON THIS PAGE

In runes mode, reactions to state updates are handled with the `$derived` and `$effect` runes.

In legacy mode, any top-level statement (i.e. not inside a block or a function) can be made reactive by prefixing it with a `$:` label. These statements run after other code in the `<script>` and before the component markup is rendered, then whenever the values that they depend on change.

```
<script>
  let a = 1;
  let b = 2;

  // this is a 'reactive statement', and it will re-run
  // when `a`, `b` or `sum` change
  $: console.log(`${a} + ${b} = ${sum}`);

  // this is a 'reactive assignment' — `sum` will be
  // recalculated when `a` or `b` change. It is
  // not necessary to declare `sum` separately
  $: sum = a + b;
</script>
```

Statements are ordered *topologically* by their dependencies and their assignments: since the `console.log` statement depends on `sum`, `sum` is calculated first even though it appears later in the source.

Multiple statements can be combined by putting them in a block:

```
  for (const item of items) {
    total += item.value;
  }
}
```

The left-hand side of a reactive assignments can be an identifier, or it can be a destructuring assignment:

```
$: ({ larry, moe, curly } = stooges);
```

Understanding dependencies

The dependencies of a `$:` statement are determined at compile time — they are whichever variables are referenced (but not assigned to) inside the statement.

In other words, a statement like this will *not* re-run when `count` changes, because the compiler cannot 'see' the dependency:

```
let count = 0;
let double = () => count * 2;

$: doubled = double();
```

Similarly, topological ordering will fail if dependencies are referenced indirectly: `z` will never update, because `y` is not considered 'dirty' when the update occurs. Moving `$: z = y` below `$: setY(x)` will fix it:

```
<script>
  let x = 0;
  let y = 0;

  $: z = y;
  $: setY(x);
```

```
</script>
```

Browser-only code

Reactive statements run during server-side rendering as well as in the browser. This means that any code that should only run in the browser must be wrapped in an `if` block:

```
$: if (browser) {
  document.title = title;
}
```

Edit this page on GitHub

PREVIOUS
Reactive let/var declarations

NEXT
export let