SVELTE • RUNES

$state

ON THIS PAGE

The `$state` rune allows you to create *reactive state*, which means that your UI *reacts* when it changes.

```
<script>
  let count = $state(0);
</script>

<button onclick={() => count++}>
  clicks: {count}
</button>
```

Unlike other frameworks you may have encountered, there is no API for interacting with state — `count` is just a number, rather than an object or a function, and you can update it like you would update any other variable.

Deep state

If `$state` is used with an array or a simple object, the result is a deeply reactive *state proxy*. Proxies allow Svelte to run code when you read or write properties, including via methods like `array.push(...)`, triggering granular updates.

> Classes like `Set` and `Map` will not be proxied, but Svelte provides reactive implementations for various built-ins like these that can be imported from `svelte/reactivity`.

simple object. In a case like this...

```
let todos = $state([
  {
    done: false,
    text: 'add more todos'
  }
]);
```

...modifying an individual todo's property will trigger updates to anything in your UI that depends on that specific property:

```
todos[0].done = !todos[0].done;
```

If you push a new object to the array, it will also be proxified:

```
todos.push({
  done: false,
  text: 'eat lunch'
});
```

> When you update properties of proxies, the original object is *not* mutated.

Classes

You can also use `$state` in class fields (whether public or private):

```
class Todo {
  done = $state(false);
  text = $state();

  constructor(text) {
    this.text = text;
  }

  reset() {
```

> The compiler transforms `done` and `text` into `get / set` methods on the class prototype referencing private fields.

$state.raw

In cases where you don't want objects and arrays to be deeply reactive you can use `$state.raw`.

State declared with `$state.raw` cannot be mutated; it can only be *reassigned*. In other words, rather than assigning to a property of an object, or using an array method like `push`, replace the object or array altogether if you'd like to update it:

```
let person = $state.raw({
  name: 'Heraclitus',
  age: 49
});

// this will have no effect
person.age += 1;

// this will work, because we're creating a new person
person = {
  name: 'Heraclitus',
  age: 50
};
```

This can improve performance with large arrays and objects that you weren't planning to mutate anyway, since it avoids the cost of making them reactive. Note that raw state can *contain* reactive state (for example, a raw array of reactive objects).

To take a static snapshot of a deeply reactive `$state` proxy, use `$state.snapshot`:

```
<script>
  let counter = $state({ count: 0 });

  function onclick() {
    // Will log `{ count: ... }` rather than `Proxy { ... }`
    console.log($state.snapshot(counter));
  }
</script>
```

This is handy when you want to pass some state to an external library or API that doesn't expect a proxy, such as `structuredClone`.

Edit this page on GitHub

PREVIOUS
What are runes?

NEXT
$derived