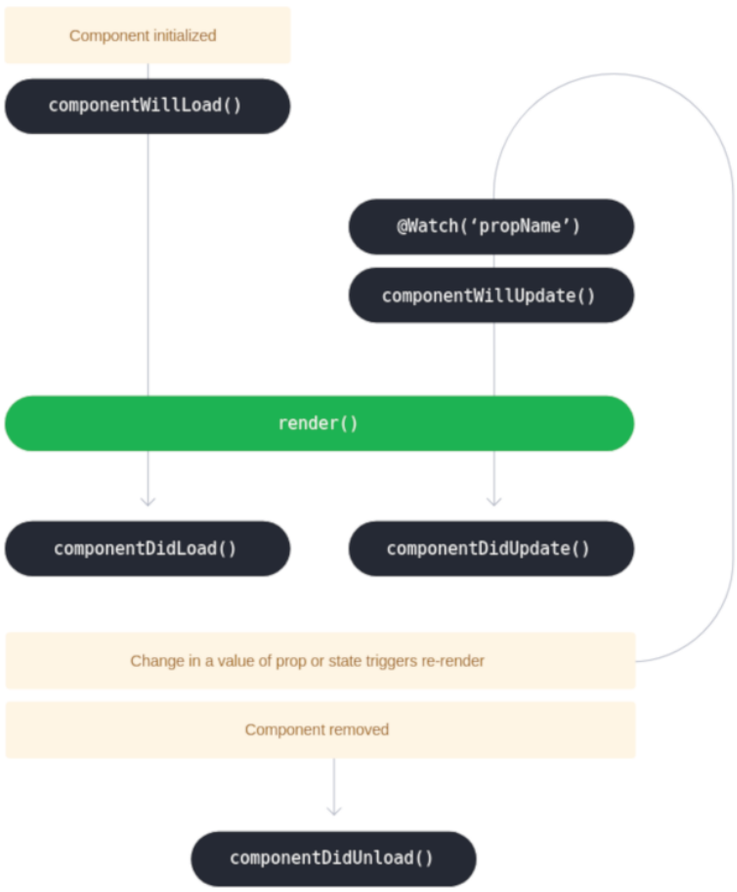
STENCIL CHEAT SHEET

A Compiler for Web Components.

COMPONENT LIFECYCLE



DECORATORS

@Component

```
import { Component } from '@stencil/core';

@Component({
  tag: 'todo-list',
  styleUrl: 'todo-list.css'
  /* styleUrls: [...] */,
  /* styles: '' */,
  shadow: true | false
})
export class TodoList {
  ...
}
```

@Prop

Declare an attribute for the outside, not directly mutable from the inside.

```
import { Prop } from '@stencil/core';

export class TodoList {
  @Prop() color: string;
  @Prop() favoriteNumber: number;
  @Prop() isSelected: boolean;
  @Prop({ mutable: true, reflectToAttr: true })
name: string = 'Stencil';

  componentDidLoad() {
    this.name = 'Stencil 0.7.0';
  }
}
```

```
/* in HTML */
<todo-list color="blue" favorite-number="24" is-
selected="true"></todo-list>
/* in JSX */
<todo-list color="blue" favoriteNumber={24}
isSelected="true"></todo-list>
```

@State

Declare an internal state only for the inside.

```
import { State } from '@stencil/core';

export class TodoList {

  @State() completedTodos: Todo[];

  completeTodo(todo: Todo) {
    /* This will cause our render function to be
called again */
    this.completedTodos = [...this.completedTodos,
todo];
  }
}
```

Update an array

this.names = [...this.names, 'Larry']

Update an object

this.options = { ...this.options, show: true }

STENCIL CHEAT SHEET

DECORATORS

@Watch()

Watch a property, useful for validating props or handling side effects.

```
@Prop() activated: boolean;

@Watch('activated')
watchHandler(newValue: boolean, oldValue:
boolean) {
  console.log('New value is: ', newValue);
}
```

@Method()

Used to expose methods on the public API. It should be async or returning a promise.

```
@Method()
showPrompt() {
  /* show a prompt */
}

todoListElement.showPrompt();
```

@Element()

Get access to the host element within the class instance. This returns an instance of an HTMLElement, so standard DOM methods/events can be used here.

```
@Element() todoListEl: HTMLElement;

addClass(){
  this.todoListEl.classList.add('active');
}
```

@Event()

Dispatch Custom DOM events for the outside world.

```
@Event() todoCompleted: EventEmitter;

todoCompletedHandler(todo: Todo) {
  this.todoCompleted.emit(todo);
}

todoListElement.addEventListener('todoCompleted',
() => {});
```

@Listen()

Listen to events dispatched from @Events.

```
@Listen('todoCompleted')
@Listen('window:scroll')
@Listen('keydown')
handler(event: CustomEvent) {

}

@Event() todoCompleted: EventEmitter;

<todo-list onTodoCompleted={ev =>
this.someMethod(ev)} />
```

STYLING

- style your component with Shadow DOM in mind
- use CSS variables for giving the opportunity for the outside world to customize your component design

INTEGRATION

```
<script src="https://unpkg.com/test-
components/latest/dist/test-components.js">
</script>
<test-component></test-component>
```

CLI

Serve

```
stencil build --dev --watch --serve
```

Build

```
stencil build
```

Build pre-rendered

```
stencil build --prerender
```

Test (Jest + Puppeteer inside)

```
stencil test --spec or --e2e
```