Note that the samples are most likely to work in a recent Chrome or Firefox browser.

6.2. Custom elements

Custom elements are a new standard allowing developers to create their own DOM elements, making something like `<ns-pony></ns-pony>` a perfectly valid HTML element. The specification defines how to declare such elements, how to make them extend existing elements, how to define your API, etc.

Declaring a custom element is done with a simple `document.registerElement('ns-pony')`:

```
// new element
var PonyComponent = document.registerElement('ns-pony');
// insert in current body
document.body.appendChild(new PonyComponent());
```

Note that the name must contain a dash, so that the browser knows it is a custom element. Of course, your custom element can have properties and methods, and it also has lifecycle callbacks, to be able to execute code when the component is inserted or removed, or when one of its attributes changes. It can also have a template of its own. Maybe the `ns-pony` displays an image of the pony or just its name:

```
// let's extend HTMLElement
var PonyComponentProto = Object.create(HTMLElement.prototype);
// and add some template using a lifecycle
PonyComponentProto.createdCallback = function() {
  this.innerHTML = '<h1>General Soda</h1>';
};

// new element
var PonyComponent = document.registerElement('ns-pony', {prototype:
PonyComponentProto});
// insert in current body
document.body.appendChild(new PonyComponent());
```

If you try to look at the DOM, you'll see `<ns-pony><h1>General Soda</h1></ns-pony>`. But that means the CSS and JavaScript logic of your app can have undesired effects on your component. So, usually, the template is hidden and encapsulated in something called Shadow DOM, and you'll only see `<ns-pony></ns-pony>` if you inspect the DOM, despite the fact that the browser displays the pony's name.

6.3. Shadow DOM

With a mysterious name like this, you expect something with great powers. And surely it is. The Shadow DOM is a way to encapsulate the DOM of our component. This encapsulation means that the stylesheet and JavaScript logic of your app will not apply on the component and ruin it inadvertently. It gives us the perfect tool to hide the internals of a component, and be sure that