```
<template id="pony-tpl">
  <style>
    h1 { color: orange; }
  </style>
  <h1>General Soda</h1>
</template>

var PonyComponentProto = Object.create(HTMLElement.prototype);

// add some template using the template tag
PonyComponentProto.createdCallback = function() {
  var template = document.querySelector('#pony-tpl');
  var clone = document.importNode(template.content, true);
  this.createShadowRoot().appendChild(clone);
};

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

Maybe we could declare this in a single file, and we would have a perfectly encapsulated component… Let's do this with HTML imports!

6.5. HTML imports

This is the last specification. HTML imports allow to import HTML into HTML. Something like `<link rel="import" href="ns-pony.html">`. This file, `ns-pony.html`, would contain everything needed: the template, the scripts, the styles, etc.

If someone wants to use our wonderful component, they just have to use an HTML import and they are good to go!

6.6. Polymer and X-tag

All these things put together make the Web Components. I'm far from being an expert on this topic, and there are all sorts of twisted traps on this road.

As Web Components are not fully supported by every browser, there is a polyfill you can include in your app to make sure it will work. The polyfill is called web-component.js, and it's worth noting that it is a joint effort from Google, Mozilla and Microsoft among others.

On top of this polyfill, a few libraries have seen the light. All aim to facilitate working with Web Components, and often come with some ready-to-use Web Components.

Among the most notable initiatives, you find:

- Polymer from Google
- X-tag from Mozilla and Microsoft