HTML Custom Elements

This is the last API of the HTML Web components. It allows you to define new HTML elements that extend the browser (the browser will render them in addition to HTML5 elements).

Basic usage:

```
document.registerElement('element-name', {
   prototype: proto
})
```

This is done using JavaScript and there are some constraints:

1. The element's new name should have a dash,
2. The prototype must inherit from HTMLElement or from any HTML existing element such as HTMLButton. It's even possible to inherit from another custom element (see external resources at the end of this page).

Here is an example that defines a new element named `<my-widget>`, that will render as an instance of a template with a shadow DOM:

HTML code for the use of the custom element:

```
<body>
   <my-widget>
     <span id="titleToInject">Title injected</span>
     <span id="partToInject">Paragraph injected</span>
   </my-widget>
</body>
```

Look at *lines 2 and 5*...

HTML code for the declaration of the template (the same as in one of the previous examples):

```
<template id="mytemplate">
 <style>
   h1 {
     color:white;
     background:red;
   }
 </style>
 <h1 part='heading'>
     <content select="#titleToInject"></content>
 </h1>
 <p part="paragraph">
    <content select="#partToInject">
 </content>
</p>
</template>
```

JavaScript code:

```
// Instanciate the template
var t =document.querySelector('#mytemplate');
// Populate the src at runtime.
var clone = document.importNode(t.content,true);

// Custom element code for creating an instance of
// the custom element
var widgetProto =Object.create(HTMLElement.prototype);

// called when the custom element is created
widgetProto.createdCallback = function() {
  // create and fill the shadow root with a cloned
  // template
  var root = this.createShadowRoot();
  root.appendChild(clone);
};
```

```
    // Register the new custom element, and set the
    // prototype for creating it
20. var Widget = document.registerElement("my-widget", {
        prototype : widgetProto
    });
```

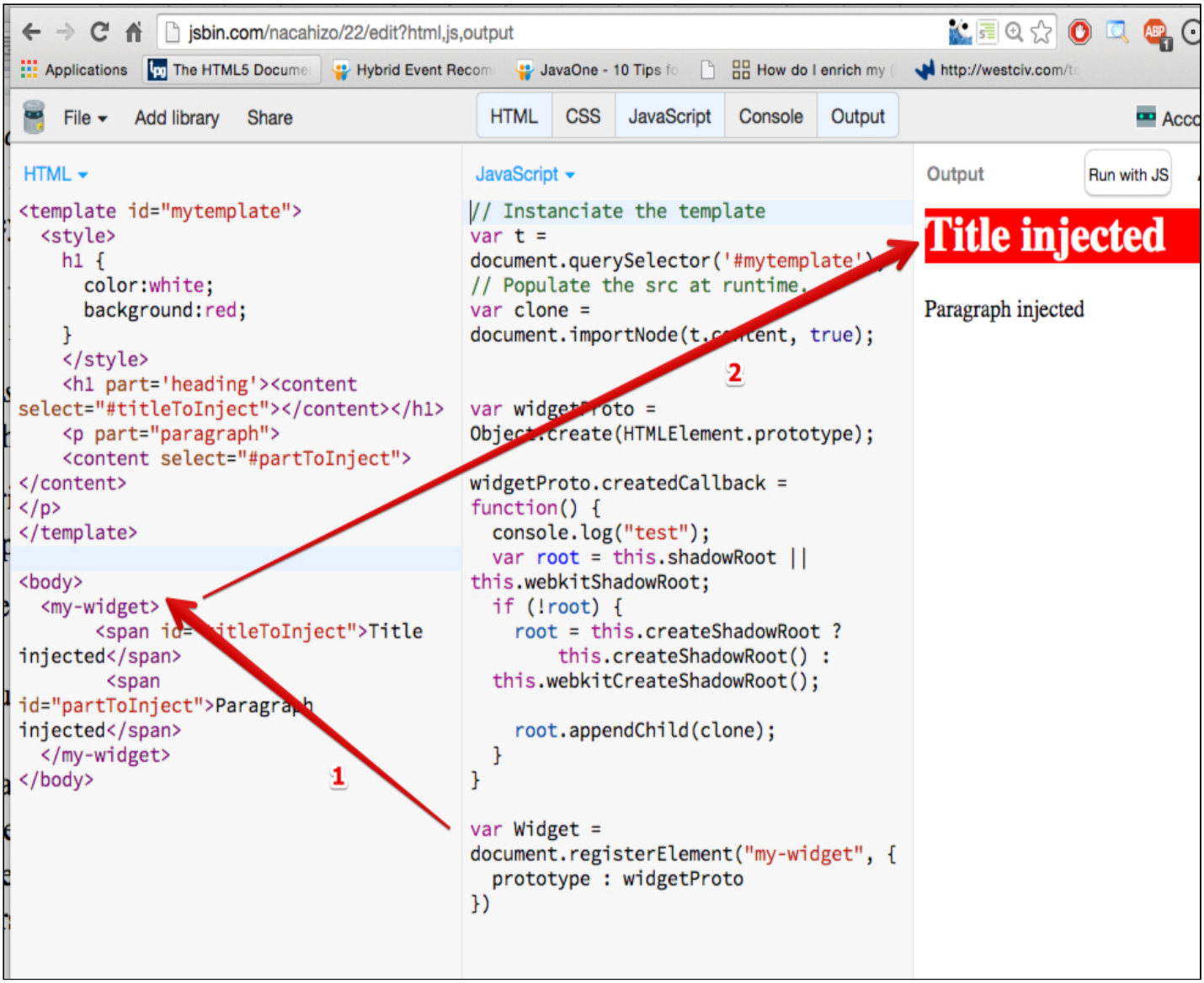
Explanations:

- *Lines 1-4*: instantiation of a template. We cloned the template content in the clone variable.
- *Lines 20-21*: registration of a new custom element named `<my-widget>`, when in the HTML document the browser will encounter `<my-widget>` it will look for the prototype function`widgetProto` to create it.
- *Line 11*: a `create` callback is defined. This function will be called by the browser when the custom element is created.
- *Lines 14-15*: these lines are inside the create callback. They will be executed when the custom element is created. They associate a shadow DOM with a shadow host referenced by the keyword `this`. In that case it's the custom element itself. This custom element is the shadow host.

To sum up, it's an example similar to a previous one we saw in the course, except that the shadow host is not the custom element itself. And the code that instantiate the template and put it in a shadow DOM, that associates this shadow DOM with the custom element, is located in a `createCallback` called when the browser needs to create the code for rendering the custom element.

COMPLETE EXAMPLE

Now, we can use the newly created element and inject content. The template used here is the last one we studied in a previous lesson about HTML templates. Check the complete online example at JSBin:



EXTERNAL RESOURCE

- This lesson is just a basic introduction to custom elements. More advanced users who would like to see how a custom element can inherit from another custom element, are welcome to read this article on HTML5Rocks.