Softtek®

POLYMER 2
CLASE 1

Value THROUGH TECHNOLOGY

Restricciones

Nombre del documento:
Clasificación de la Información: Interno
Restricciones

› Los contenidos de este documento son propiedad de Softtek y son internos. Queda estrictamente prohibido cualquier reproducción total o parcial sin la autorización escrita por parte de Softtek.

› Este documento está sujeto a cambios. Los comentarios, correcciones o dudas deberán ser enviados al autor.

Audiencia	Propósito
Desarroladores	Capacitación

Tabla de Revisión

› La siguiente tabla enlista las revisiones realizadas a este documento. Debe utilizarse para describir los cambios y adiciones cada vez que este documento vuelva a ser publicado. La descripción debe ser detallada e incluir el nombre de quien solicita los cambios.

Núm. de versión	Fecha de versión	Tipo de cambios	Dueño / Autor	Fecha de revisión / Expiración
1.1	5/10/2018	Creación del documento	Carlos Montero	12/10/2018

POLYMER 2

Softtek®

En Polymer 2 pasamos a trabajar con ES6 y html, por lo tanto, podemos definir nuestros componentes de forma mucho más ordenada y limpia haciendo uso de clases que extiendan de Polymer.Element.

La verdad que seguimos teniendo todo lo que teníamos antes pero de otra forma, por ejemplo, los ciclos de vida en Polymer 2 siguen existiendo y trabajan exactamente de la misma forma, propiedades, mixins etc, así que si ya conocías Polymer en sus versiones 1.x no tendrás problemas con Polymer 2.

POLYMER 2

Softtek®

web components:

Elementos reutilizables, que se declaran en el código de la web mediante sus propias etiquetas, y que son independientes.
Con independientes queremos decir que tienen su propia estructura y su contenido está aislado del resto de la web. El CSS de fuera no puede acceder aquí, al menos de la forma normal.

Las desventajas es que ya no tenemos un CSS general para toda la web, sino que ahora vamos a tener diferentes web components con los que vamos a construir nuestra aplicación, cada uno con sus propios estilos y su propia estructura.

Una de las cosas nuevas que también nos traen los web components, (algo que lleva un montón de tiempo en pre procesadores de CSS como SASS o LESS) son las variables y los mixins, algo que nos va a venir bastante bien tanto para modificar los estilos desde nuestros componentes desde fuera como para reutilizarlos.

POLYMER 2

Softtek®

Shadow DOM

Se refiere a la habilidad de un navegador de incluir un subárbol de elementos en un documento, pero no así en el árbol DOM de este documento.

Otra forma (más simple) de explicarlo con palabras: el Shadow DOM es un DOM encapsulado que vive dentro del DOM principal. Al leer esto, algunos estarán pensando que es lo mismo que un iframe, pero la diferencia está en que: el Shadow DOM es parte del documento, no un documento diferente.

La forma más fácil de entender qué es el Shadow DOM es, como comentaba más arriba, ver **como lo están utilizando los que desarrollan los navegadores**.

En el caso de Chrome, esto es muy fácil de ver:

1. Primero tenemos que ir a algún sitio donde haya un elemento *video*, *audio*, *range*, etc. En mi caso elegí un video.

2. Si inspeccionamos el elemento, veremos que el código es el simple de un video con el elemento source dentro como muestra la siguiente imagen:

POLYMER 2

```
▼<video controls autoplay name="media">
    <source src="http://ak6.picdn.net/shutterstock/videos/30
  </video>
```

3. Para poder ver el Shadow DOM, debemos activar la opción en la Configuración de las Herramientas de Desarrollador (Chrome Developer Tools):

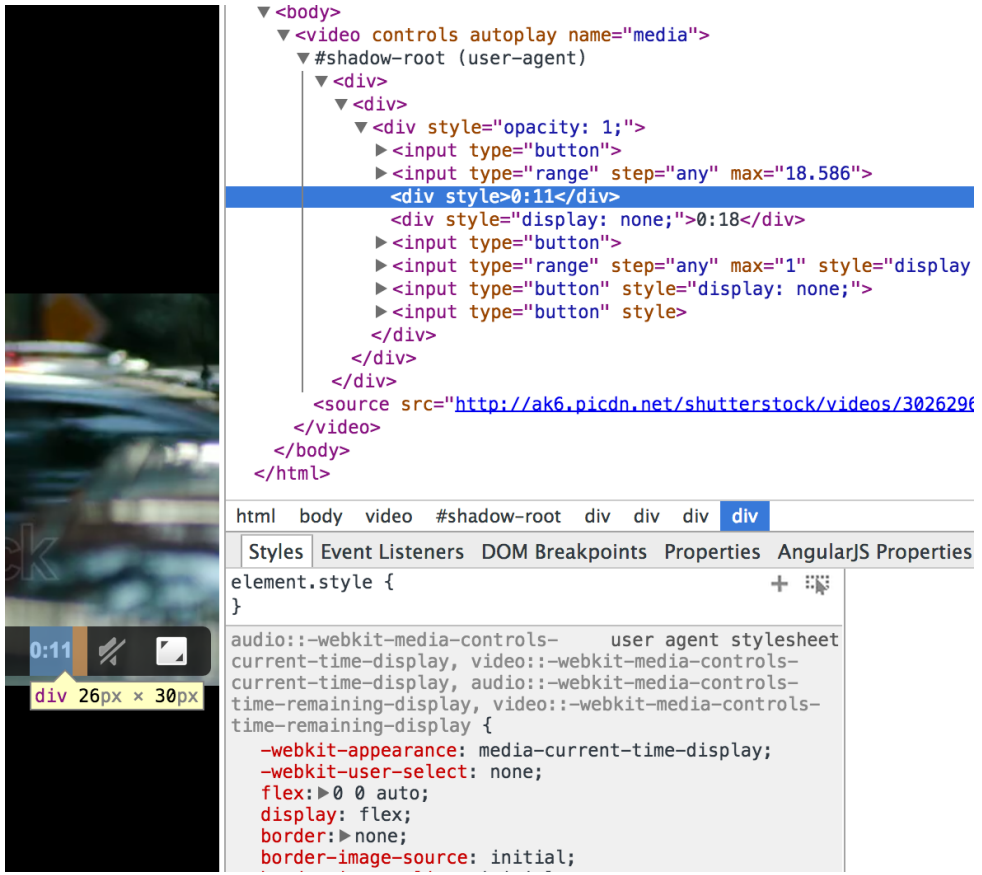
4. Ahora inspeccionamos nuevamente el elemento video y nos encontramos que, además de source, aparece el Shadow DOM:

```
▼<video controls autoplay name="media">
  ▶#shadow-root (user-agent)
    <source src="http://ak6.picdn.net/shutterstock/videos/30
  </video>
```


POLYMER 2

Softtek®

Al expandir Shadow DOM, vemos el sub-árbol que conforma el elemento video. Inspeccionando cada uno de sus elementos, podemos ver que son elementos comunes como divs:



Conclusión

El Shadow DOM nos permite definir elementos complejos para poder reutilizarlos. Su característica, de ser un DOM encapsulado que coexiste dentro del DOM padre, nos brinda un forma de tener código aislado del resto tanto en estilos como en funcionalidad.

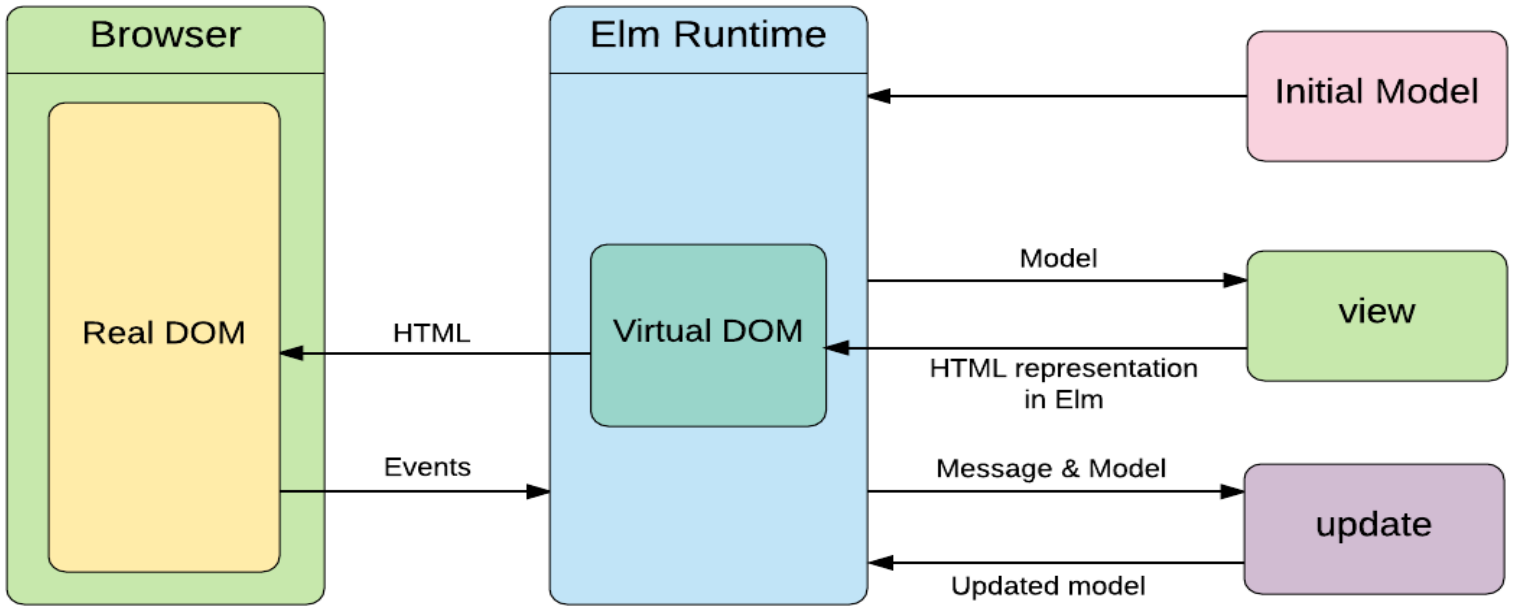
POLYMER 2

¿Qué es el DOM?

El modelo de objeto de documento (DOM) es una interfaz de programación para los documentos HTML y XML. Facilita una representación estructurada del documento y define de qué manera los programas pueden acceder, al fin de modificar, tanto su estructura, estilo y contenido. El DOM da una representación del documento como un grupo de nodos y objetos estructurados que tienen propiedades y métodos. Esencialmente, conecta las páginas web a scripts o lenguajes de programación.

Una página web es un documento. Éste documento puede exhibirse en la ventana de un navegador o también como código fuente HTML. Pero, en los dos casos, es el mismo documento. El modelo de objeto de documento (DOM) proporciona otras formas de presentar, guardar y manipular este mismo documento. El DOM es una representación completamente orientada al objeto de la página web y puede ser modificado con un lenguaje de script como JavaScript.

POLYMER 2



DOM (document Object Model), es la manera en que los navegadores le permiten a los desarrolladores manipular paginas web con JS, cada vez que un navegador carga una pagina web analiza todo el código HTML y CSS y convierte el documento en un DOM, ese DOM en realidad es un gran objeto de JS, que contiene todo lo que quisiéramos saber o manipular sobre una pagina (etiqueta, atributos, estilos para cada Etiqueta …), Para acceder al DOM de una pagina WEB desde JS usamos la variable global **DOCUMENT** y entonces podemos usar propiedades y llamar funciones ligadas a este objeto.

DOM DOCUMENT

Softtek®

Finding HTML Elements (Elementos de Busqueda HTML).

Method	Description
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

Changing HTML Elements (Elementos de Cambio HTML).

Method	Description
element.innerHTML = *new html content*	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.setAttribute*(attribute, value)*	Change the attribute value of an HTML element
element.style.*property = new style*	Change the style of an HTML element

DOM DOCUMENT

Adding and Deleting Elements (Agregar y Eliminar Elementos).

Method	Description
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element
document.replaceChild(*element*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

Adding Events Handlers (Agregar eventos manipulados).

Method	Description
document.getElementById(*id*).onclick = function() {*code*}	Adding event handler code to an onclick event

DOM DOCUMENT

Finding HTML Objects (Busca Objetos HTML)

Property	Description	DOM
document.anchors	Returns all <a> elements that have a name attribute	1
document.applets	Returns all <applet> elements (Deprecated in HTML5)	1
document.baseURI	Returns the absolute base URI of the document	3
document.body	Returns the <body> element	1
document.cookie	Returns the document's cookie	1
document.doctype	Returns the document's doctype	3
document.documentElement	Returns the <html> element	3
document.documentMode	Returns the mode used by the browser	3
document.documentURI	Returns the URI of the document	3
document.domain	Returns the domain name of the document server	1
document.domConfig	Obsolete. Returns the DOM configuration	3
document.embeds	Returns all <embed> elements	3

DOM DOCUMENT

Finding HTML Objects (Busca Objetos HTML)

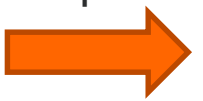
document.forms	Returns all <form> elements	1
document.head	Returns the <head> element	3
document.images	Returns all <img> elements	1
document.implementation	Returns the DOM implementation	3
document.inputEncoding	Returns the document's encoding (character set)	3
document.lastModified	Returns the date and time the document was updated	3
document.links	Returns all <area> and <a> elements that have a href attribute	1
document.readyState	Returns the (loading) status of the document	3
document.referrer	Returns the URI of the referrer (the linking document)	1
document.scripts	Returns all <script> elements	3
document.strictErrorChecking	Returns if error checking is enforced	3
document.title	Returns the <title> element	1
document.URL	Returns the complete URL of the document	1

Polymer 1 → Polymer 2

Softtek®

```
Polymer({
  is: 'ejemplo-ocho-hijocuatro',
```

Script



```
class RanaOk extends Polymer.Element {
  static get is() { return 'rana-ok'; }
```

```
static get properties() {
  return {
    propvelocidad: {
      type: Number,
      value: 0
    },
```

Propiedades

```
static get properties() {
  return {
    tamarr: {
      type: Number,
    }
  };
```

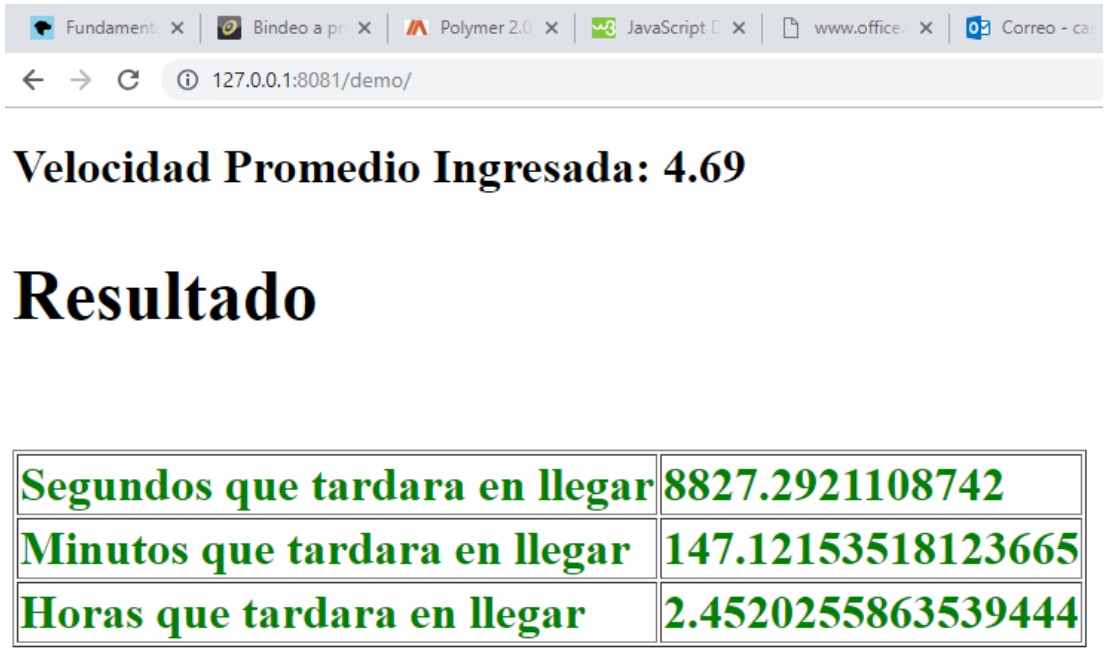
```
show_valor_metodo: function (){
  /*document.write(this.get('recibe'));*/
  console.log(this.get('recibe'));
}
```

Métodos

```
geanarreglo(lenarr){
  var line = ['frog'];
  for (let i = 0; i < lenarr; i++){
    if (line[i] != 'frog') {
      line[i] = 'Null'
    }
  }
```


Polymer 1 → Polymer 2

Genera una nueva practica con polymer 2, en la cual en tu pantalla inicial vas a solicitar la velocidad promedio para la ruta Softtek Toreo a Metro Toreo, y un botón que va a llamar a tu componente polymer en su versión 2 el cual va a generar un texto con la velocidad promedio y una tabla indicando el tiempo en segundos, minutos y Horas.
La tabla de tener un estilo a tu gusto, te muestro gráficamente lo solicitado:



propiedades computadas (computed)

Softtek®

Las propiedades computadas son aquellas que toman su valor por medio de un cómputo, al que se llega mediante los valores de otras propiedades del componente. Son especialmente útiles y se recurre a ellas en gran cantidad de ocasiones.

Polymer 2, y su predecesor 1.x, hacen muy sencillo el trabajar con propiedades computadas. Además realizan todo el trabajo pesado de manera automática, para que nosotros no tengamos que preocuparnos porque los cómputos estén siempre correctamente actualizados. Para ello, cada vez que cambian los valores de las propiedades usadas para realizar el cómputo, Polymer se encarga de actualizar el valor de las propiedades computadas.

Se entenderán sin duda explicando casos de uso sencillos. Por ejemplo, podemos tener en un componente dos propiedades: nombre y apellidos. Además podemos necesitar el "nombre completo" para otras cosas. Esa propiedad nombre completo podría ser una propiedad computada, pues para obtener su valor se consigue mediante la concatenación de las dos propiedades del elemento nombre y apellidos.

propiedades computadas (computed)

Softtek®

Otro buen candidato a propiedad computada. Si tengo una factura y el total se calcula por medio de la suma de todos los valores de cada concepto facturado, esa propiedad "total" podría ser una propiedad computada. La fuerza de una clave, que se calcula en función de la cadena de la contraseña escrita por el usuario, también podría ser una computada.

Vamos a realizar un primer ejemplo de un custom element en el que incluimos una propiedad computada. Vamos a trabajar sobre el sencillo caso del "nombre completo" que es la contactenación del nombre y apellidos. Verás que trabajar con computed properties en Polymer 2 es algo realmente fácil.

Como decimos, en nuestro componente tenemos tres propiedades:

- **nombre**
- **apellidos**
- **nombreCompleto, que es la computada a partir de las propiedades anteriores.**

Para crear una propiedad computada tenemos que declararla como cualquier otra. Dentro de todas las configuraciones que podemos realizar sobre las propiedades declaradas hay una llamada "computed". En ella debemos de colocar el nombre del método que se usará para realizar el cálculo, a fin de hallar el valor de la propiedad. Además hay que señalar qué propiedad o propiedades se usan para para calcular el valor resultante.

propiedades computadas (computed)

En base al ejemplo anterior, genera la practica 2 y realiza la propiedad computada para el Nombre Completo, considera los fragmentos de código que a continuación se muestran para que sea mas fácil realizar la practica.

```
static get properties() {
  return {
    clave: {
      type: String,
      value: ''
    },
    fuerza: {
      type: Number,
      value: 0,
      computed: 'calcularFuerzaClave(clave)'
    }
  };
}
```

```
calcularFuerzaClave(clave) {
  if(clave.length > 10) {
    return 10;
  }
  return clave.length;
}
```


Introducción a los eventos en Polymer 2

El mundo de los eventos en Polymer es bastante rico y además ha cambiado en algunas cosas en lo que respecta a su versión predecesora, por lo que tendremos que prestar bastante atención incluso si venimos de Polymer 1.x. La novedad principal es que en Polymer 2 se usa más el propio Javascript nativo para cosas en las que antes existían mecanismos proveídos por la propia librería (Recuerda: "UseThePlatform").

Asociar un evento a un elemento en línea

A la hora de escribir un template podemos asociar una función manejadora de evento en cualquier elemento del shadow dom del componente. Para ello simplemente usaremos un atributo "on-", seguido del nombre del evento que queramos definir su manejador. Ese atributo tendrá como valor el nombre del método manejador del evento que se desea asociar.

Por ejemplo, encontraremos un código como este para definir un manejador de evento frente a un clic sobre un botón.

```
<button on-click="incrementaClic">Clic para contar</button>
```


Introducción a los eventos en Polymer 2

Softtek®

Declaración de un manejador de evento

El manejador del evento en un componente de Polymer corresponde con un método de la propia clase del componente. El método lo podemos definir como cualquier otro método dentro de la clase ES6.

```
incrementaClic() {
  // código a ejecutar como manejador del evento
}
```

El manejador recibe el objeto evento, del que podemos obtener información diversa sobre el tipo de evento que se acaba de producir. Ese objeto evento será siempre el primer parámetro del manejador.

```
incrementaClic(e) {
  console.log('Objeto evento', e)
}
```


Introducción a los eventos en Polymer 2

Softtek®

Usar addEventListener de Javascript

En Polymer se recomienda usar el propio Javascript para crear tus eventos. Por ejemplo, si quieres crear un evento que afecte a todo el host (el componente completo, la etiqueta que engloba todo el Shadow DOM que podamos tener dentro del elemento personalizado) podrías crearlo directamente con this.addEventListener().

En este código vemos cómo definir el evento sobre el host, asociando el manejador en el método callback ready.

```
ready() {
  super.ready();
  this.addEventListener('click', () => {
    console.log('yeahhh este evento se asocia al componente completo');
  })
}
```


Introducción a los eventos en Polymer 2

Softtek®

Eventos de gestos

Los eventos de gestos ahora no están incluidos "de casa", por eso no te funcionará el evento on-tap en un componente básico. Hay un mixin llamado Polymer.GestureEventListeners que da soporte a los eventos de gestos y que tendremos que heredar en el componente donde los queramos usar.

Nota: Los mixins sirven para hacer lo que sería algo parecido a la herencia múltiple. Básicamente sirven para heredar código sin necesidad de que el mixin esté en el extends. Son necesarios porque Javascript no tiene herencia múltiple. Pero lo más importante que ya te adelantamos también, los mixins en Polymer 2 son los sustitutos a lo que conocíamos por "behaviors" en Polymer 1.x.

Introducción a los eventos en Polymer 2

Softtek®

Eventos de gestos Ejemplo con código.

Index.html

```
<link rel="import" href="polymer/lib/mixins/gesture-event-listeners.html">

<script>
   class TestEvent extends
Polymer.GestureEventListeners(Polymer.Element) {
      ...
</script>

<div id="dragme" on-track="handleTrack">Drag me!</div>
```


Introducción a los eventos en Polymer 2

Softtek®

```
<link rel="import" href="polymer/polymer-element.html">
<link rel="import" href="polymer/lib/mixins/gesture-event-listeners.html">

<dom-module id="drag-me">
  <template>
    <style>
      #dragme {
        width: 500px;
        height: 500px;
        background: gray;
      }
    </style>

    <div id="dragme" on-track="handleTrack">[[message]]</div>
  </template>
```

```
  <script>
    class DragMe extends Polymer.GestureEventListeners(Polymer.Element) {

      static get is() {return 'drag-me'}

      handleTrack(e) {
        switch(e.detail.state) {
          case 'start':
            this.message = 'Tracking started!';
            break;
          case 'track':
            this.message = 'Tracking in progress... ' +
              e.detail.x + ', ' + e.detail.y;
            break;
          case 'end':
            this.message = 'Tracking ended!';
            break;
        }
      }

    }
    customElements.define(DragMe.is, DragMe);
  </script>
</dom-module>
```


Funcionalidades incorporadas en las propiedades de Polymer 2

Softtek®

Comenzamos con un listado de las funcionalidades configurables en las propiedades de los componentes.

type: Sirve para indicar el tipo (Boolean, Number, String, Array, Object y la menos conocida Date)

value: Sirve para definir el valor predeterminado de una propiedad. Si se indica, se usará para inicializar esa propiedad, aunque a la hora de usar el componente mandará la posible inicialización que se realice en el atributo HTML del componente asociado con esa propiedad. El valor también se puede calcular por medio de una función, que se invocará si es necesario en la inicialización del componente. Se usará el valor de devolución de la función para inicializar la propiedad. Además, para propiedades de tipo Array y Object, si queremos que exista una copia del array o el objeto para cada elemento generado, tenemos que producir ese array u objeto en una función, devolviendo el valor a inicializar con el correspondiente return.

Funcionalidades incorporadas en las propiedades de Polymer 2

reflectToAttribute: Sirve para que el valor de la propiedad actualice el valor de la propiedad en el elemento. Por defecto esto no ocurre, por lo que, si se cambia el valor de una propiedad, no se refleja en el HTML del componente host. Si queremos que se actualice el atributo en la etiqueta HTML se debe marcar con true reflectToAttribute.

readOnly: Si lo configuramos como true, el valor de esa propiedad no se puede alterar desde fuera, vía binding.

notify: esta configuración permite que el componente notifique a los padres mediante binding cualquier cambio de la propiedad. Es una configuración muy importante para que los elementos que crees soporten binding de dos direcciones. Por defecto está a false y en ese caso, aunque el componente modifique internamente una propiedad, su valor no viajará al padre por binding, independientemente que usemos las dobles llaves al declarar el bindeo en el template.

Funcionalidades incorporadas en las propiedades de Polymer 2

computed: es otra de las más importantes configuraciones, que permite hacer propiedades que tiene valores que serán calculados por una función. Se utilizan muchísimo, ya que muchas veces los valores de propiedades son el resultado de realizar un cálculo. Para definir una propiedad computada usamos una función, que recibe los valores necesarios para realizar el cálculo, que deben ser otras propiedades del componente. Polymer ya automatiza el procedimiento para que, cuando cambian los parámetros de entrada en el cómputo, se invoque de nuevo la función, de modo que siempre las propiedades computadas tengan el valor actualizado.

observer: También es extremadamente útil, ya que permite definir una función que se ejecutará cuando se produzcan cambios en el valor de una propiedad, facilitando enormemente la programación reactiva. Cada vez que una propiedad se altere el componente reacciona por los observers y realiza todas las operaciones que ese cambio deba producir.

Funcionalidades incorporadas en las propiedades de Polymer 2

Softtek®

Ahora vamos a practicar con una de las configuraciones de propiedad más simples, reflectToAttribute. Como hemos mencionado anteriormente, sirve para que la etiqueta host refleje en sus atributos los cambios que se produzcan en las propiedades del componente. Recuerda, el atributo es lo que escribimos en una etiqueta HTML y la propiedad es lo que controlamos desde Javascript.

Al usarlo no indicamos ningún atributo, pero quizás dentro del componente complejo tenemos decenas de propiedades para controlar su estado. Esas propiedades generalmente se gestionan dentro del componente y los valores actuales no se reflejan modificando los atributos en la etiqueta host.

Nota: Por etiqueta host puedes entender la etiqueta que escribes para usar un componente. Dentro del componente podemos tener un template con una serie de etiquetas en lo que llamamos el Shadow DOM, pero el componente en si se usa con una sola etiqueta, con el nombre del elemento. Esa etiqueta que se coloca en el HTML para usar el componente es la etiqueta host.

Funcionalidades incorporadas en las propiedades de Polymer 2

Softtek®

Ejemplo de componente con propiedades reflectToAttribute
Entendido esto, podemos pasar a ver el uso de la configuración reflectToAttribute, que modifica el comportamiento del componente para reflejar el estado de las propiedades que deseemos que se vean desde fuera.

```
<link rel="import" href="../../bower_components/polymer/polymer-element.html">

<dom-module id="uso-reflecttoattribute">
  <template>
    <style>
      :host {
        display: block
      }
    </style>

    <a>Este es el valor que debe verse en el atributto del host: [[reflejada]]<a>
    <button on-click="cambiarValor">Haz clic para cambiar el valor (que se reflejará en el atributo)</button>
  </template>
```


Funcionalidades incorporadas en las propiedades de Polymer 2

```
<script>

  class TipoPersonalizado {
    //aquí el código de un nuevo tipo inventado
  }

  class UsoReflecttoattribute extends Polymer.Element {

    static get is() {
      return 'uso-reflecttoattribute';
    }

    static get properties() {
      return {
        reflejada: {
          type: String,
          value: 'Me reflejo!!',
          reflectToAttribute: true
        }
      };
    }
```


Funcionalidades incorporadas en las propiedades de Polymer 2


```
constructor() {
      super();
    }

    cambiarValor() {
      this.reflejada = Date.now();
    }
  }

  window.customElements.define(UsoReflecttoattribute.is,
UsoReflecttoattribute);
 </script>
</dom-module>
```


Funcionalidades incorporadas en las propiedades de Polymer 2

Ahora, para usar el componente podríamos tener perfectamente este HTML:

<uso-reflecttoattribute></uso-reflecttoattribute>

PRACTICA 3 EJERCICIO FINAL

Softtek®

Genera un componente polymer que reciba por parámetros la ruta de 3 imágenes que se encuentran y Un texto por cada imagen, a si como el ancho y largo de cada imagen esto va en formato json.
El componente a recibir estas imágenes debe generar 3 cuadros con las imágenes y el texto y centrar El block en la pagina.

```
<footer-elem footer-data='[{
"horizontal":"200",
"large":"200",
"info_1":"Texto Referente a primera imagen a la izquierda",
"img_1":"../img/prac_2_img_1.jpg",
"info_2":"Texto Referente a segunda imagen a la izquierda",
"img_2":"../img/prac_2_img_2.jpg",
"info_3":"Texto Referente a tercera imagen a la izquierda",
"img_3":"../img/prac_2_img_3.jpeg"
}]'>
</footer-elem>
```


Softtek®

Polymer 2 Clase 1

Curso de capacitación para personal asignado a BBVA