

Programación Orientada a Componentes Web

El Proyecto Polymer

Javier Vélez Reyes

@javiervelezreye
Javier.velez.reyes@gmail.com

Febrero 2015



El Proyecto Polymer

Autor

I. ¿Quién Soy?



Licenciado en informática por la UPM desde el año 2001 y doctor en informática por la UNED desde el año 2009, Javier es investigador y su línea de trabajo actual se centra en la innovación y desarrollo de tecnologías de Componentes Web. Además realiza actividades de evangelización y divulgación en diversas comunidades IT, es Polymer Polytechnic Speaker y co-organizador del grupo Polymer Spain que conforma una comunidad de interés de ámbito nacional en relación al framework Polymer y a las tecnologías de Componentes Web.



javier.velez.reyes@gmail.com



[@javiervelezreye](https://twitter.com/javiervelezreye)



linkedin.com/in/javiervelezreyes



gplus.to/javiervelezreyes



[jvelez77](#)



[javiervelezreyes](#)



youtube.com/user/javiervelezreyes

II. ¿A Qué Me Dedico?

Polymer Polytechnic Speaker

Co-organizador de Polymer Spain

Evangelización Web

Desarrollador JS Full stack

Arquitectura Web

Formación & Consultoría IT

e-learning

Javier Vélez Reyes
@javiervelezreye
Javier.velez.reyes@gmail.com

1 *Introducción*

- Qué es un Componente Web
- Por Qué los Componentes Web
- La Web Como Plataforma de Componentes
- Un Nuevo Modelo de Roles

El Proyecto Polymer

Introducción

Qué Son Los Componentes Web



Componentes Web

Etiquetas de Autor

“

La tecnología de Componentes Web proporciona un mecanismo para construir nuevas etiquetas de autor personalizadas que incluyen una semántica, un comportamiento funcional y una lógica de presentación propia.

”

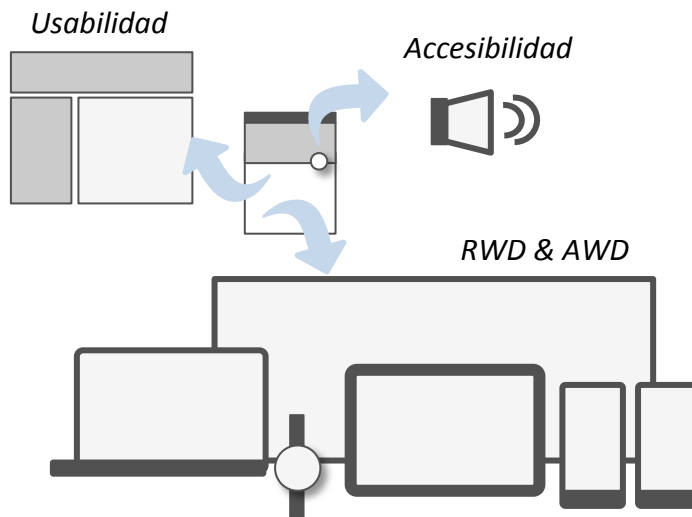
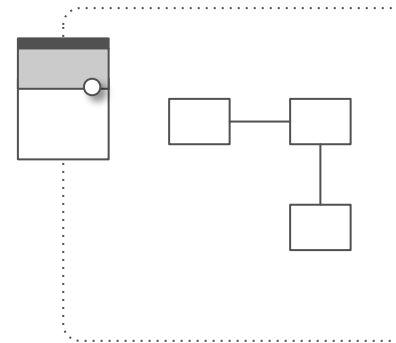
El Proyecto Polymer

Introducción

Por Qué Componentes Web

Encapsulación De La Lógica

Un Componente Web encapsula cierta lógica funcional y presentacional elaborada que se activa desde el navegador como una simple etiqueta estándar de HTML.



Comportamiento Adaptativo

Los componentes se desarrollan para hacerse responsables de la gestión del espacio en el área de presentación, de las necesidades de accesibilidad y las preocupaciones generales de usabilidad.

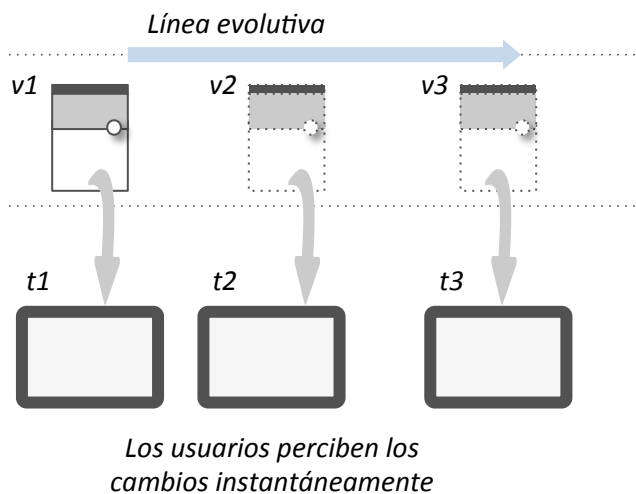
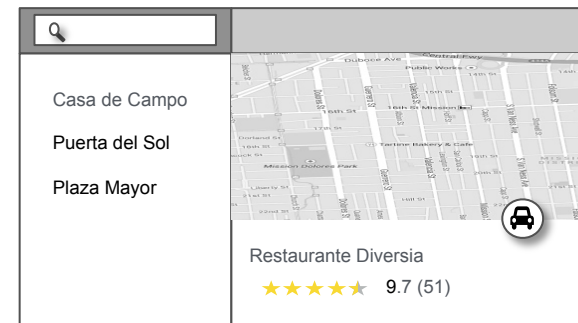
Principios de Diseño en Componentes Web

Introducción

Por Qué Componentes Web

Homogenización Del diseño

Los componentes Web también fomentan un proceso de diseño Web que tiende a la homogenización lo que desde un punto de vista corporativo resulta ventajoso.



Gestión De La Evolución Centralizada

La lógica de presentación de los componentes Web se puede adaptar evolutivamente bajo demanda de las necesidades empresariales sin impactar en el correcto funcionamiento de los aplicativos y sitios web que hacen uso de los mismos.

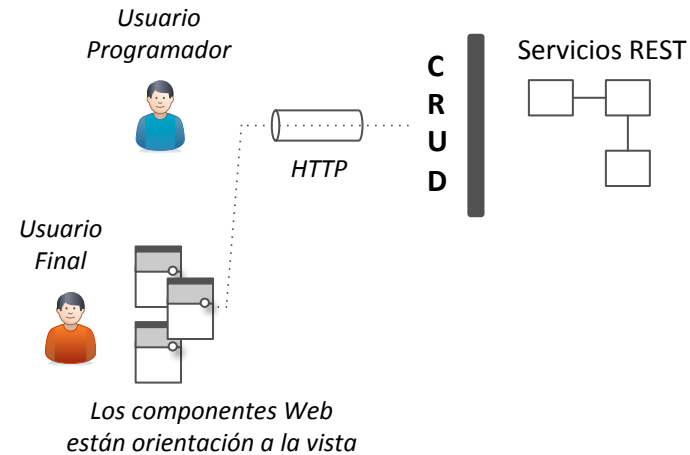
El Proyecto Polymer

Introducción

Por Qué Componentes Web

Orientación A La Vista

La tecnología de componentes Web ofrece la posibilidad de exponer los servicios de una organización en forma de un conjunto de etiquetas visuales que conectan con capacidades de negocio.



Construcción Compositiva

Como cualquier otra etiqueta estándar los componentes Web pueden conectarse de forma compositiva a través de las relaciones de anidamiento y propagación de eventos que soporta la Web.

El Proyecto Polymer

Introducción

La Web Como Plataforma De Componentes

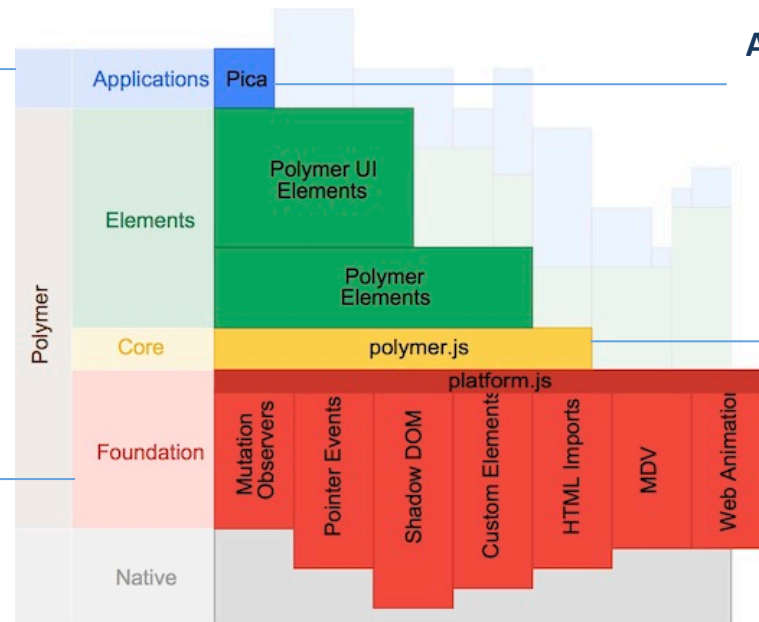
La Web se presenta como una plataforma de hospedaje para una colección de componentes con un comportamiento funcional subyacente que se comunican y coordinan entre si, de acuerdo a diferentes esquemas para ofrecer servicios de valor en el contexto de la página donde se inscriben.

Elementos

Componentes estándar dedicados a dar respuesta a necesidades recurrentes de control o UI que surgen en el desarrollo orientado a Componentes Web

Plataforma

Incluye todas las nuevas capacidades que extienden la Web como plataforma de orientación a componentes



Aplicaciones

Aplicaciones orientadas a componentes al servicio de los usuarios de la Web

Core

Capacidades funcionales sobre la plataforma que simplifican el proceso de desarrollo de Componentes Web

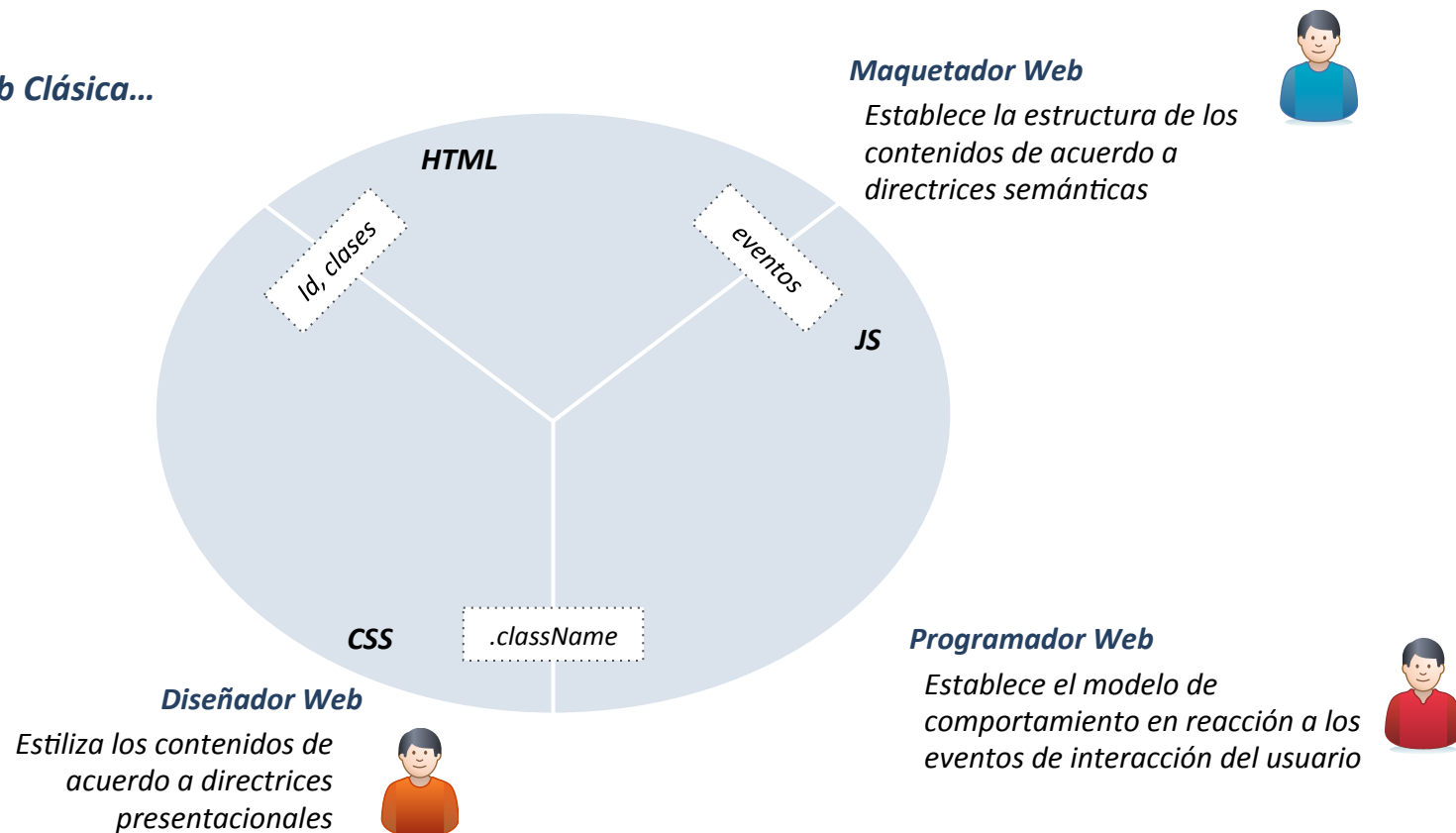
El Proyecto Polymer

Introducción

Un Nuevo Modelo De Roles

Conviene señalar que, en el marco de este nuevo proceso de construcción de soluciones para la Web basadas en componentes, se requieren nuevos roles de especialización técnica y flujos de trabajo establecidos. A continuación bosquejamos los mismos.

En la Web Clásica...



El Proyecto Polymer

Introducción

Un Nuevo Modelo De Roles

Conviene señalar que, en el marco de este nuevo proceso de construcción de soluciones para la Web basadas en componentes, se requieren nuevos roles de especialización técnica y flujos de trabajo establecidos. A continuación bosquejamos los mismos.

En la Web Orientada a Componentes...

Diseñador de Componentes

Define el modelo de rendering del componente de acuerdo a directrices de estilo



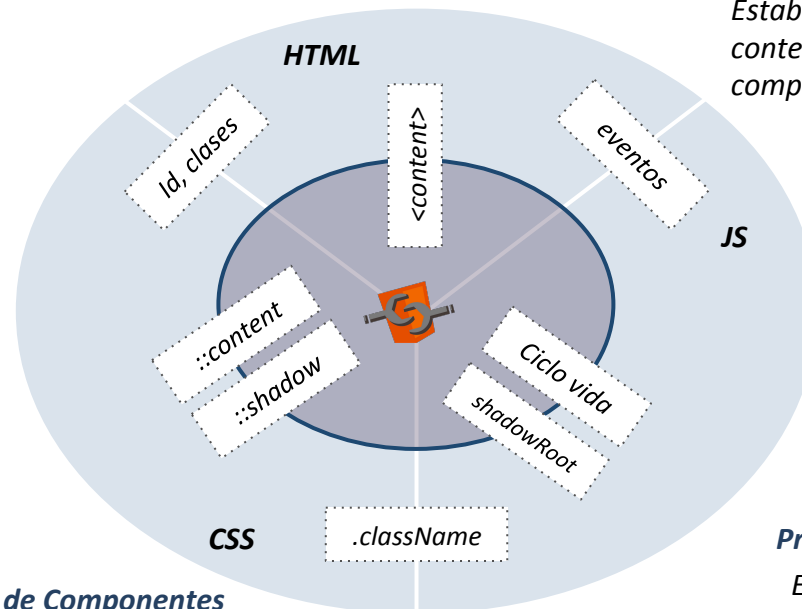
Maquetador de Componentes

Establece la estructura de los contenidos internos que presenta el componente cuando se renderiza



Programador de Componentes

Establece el modelo de comportamiento subyacente que se activa cuando se renderiza el componente



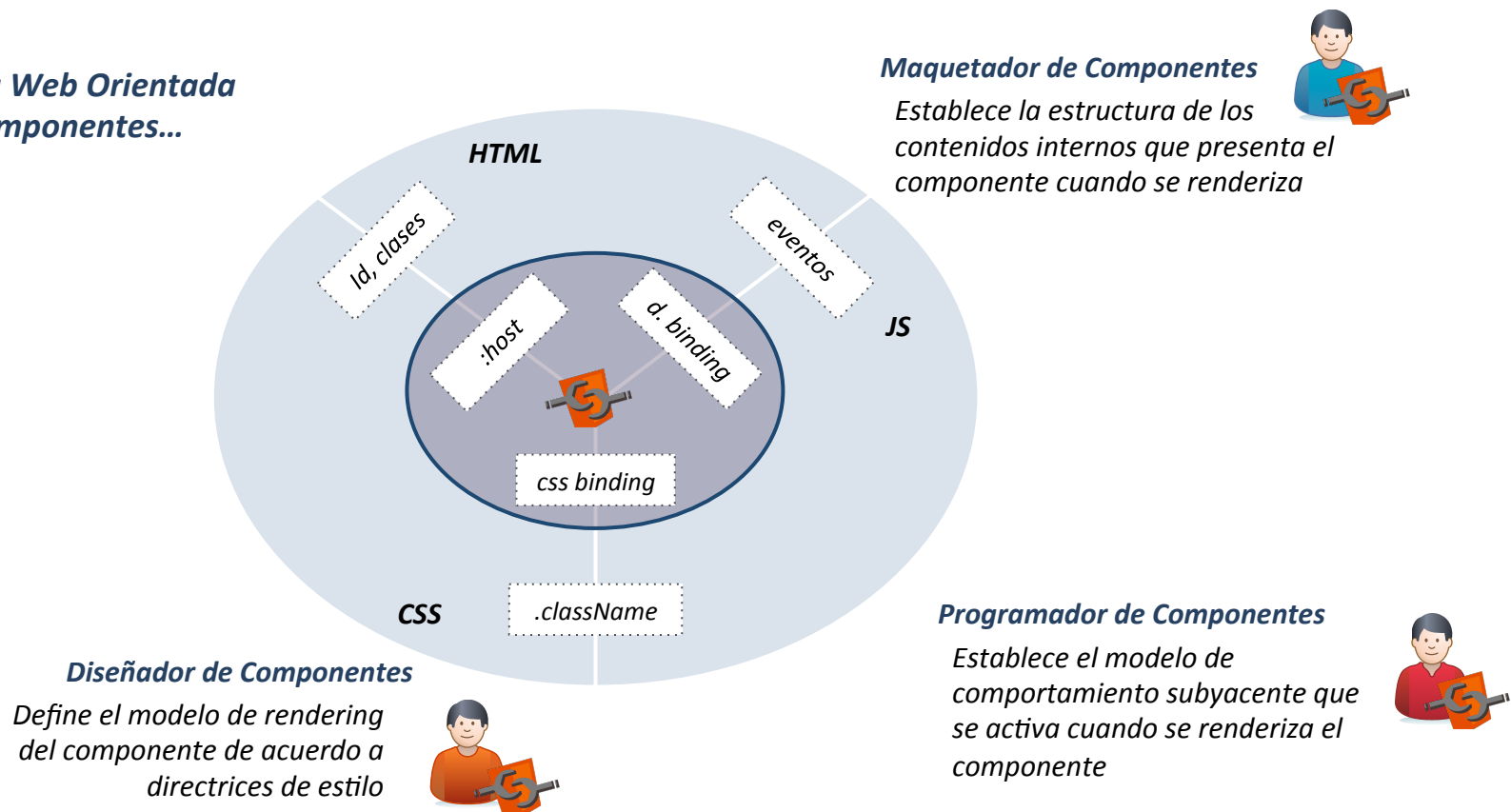
El Proyecto Polymer

Introducción

Un Nuevo Modelo De Roles

Conviene señalar que, en el marco de este nuevo proceso de construcción de soluciones para la Web basadas en componentes, se requieren nuevos roles de especialización técnica y flujos de trabajo establecidos. A continuación bosquejamos los mismos.

En la Web Orientada a Componentes...



Javier Vélez Reyes
@javiervelezreye
Javier.velez.reyes@gmail.com

2 *Estándares En Componentes Web*

- Modelo de Encapsulamiento. Shadow DOM
- Modelo de Renderizado. HTML Templates
- Modelo de Extensión. Custom Elements
- Modelo de Modularización. HTML Imports

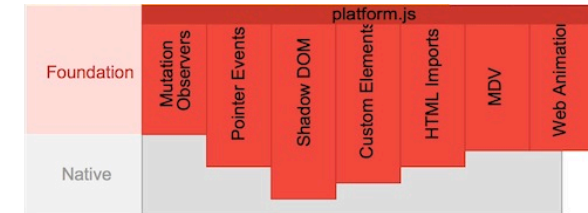
El Proyecto Polymer
Introducción

El Proyecto Polymer

Estándares En Componentes Web

4 Estándares Independientes

El modelo de componentes para la Web definido por la W3C consiste en cuatro especificaciones tecnológicamente independientes que cubren distintos aspectos del proceso constructivo. A lo largo de este capítulo revisaremos los Estándares En Componentes Web que se manejan en relación a tales especificaciones.



Shadow DOM

Ofrece un modelo de encapsulamiento que permite aislar el contenido interno del componente de aquél en la página donde éste es renderizado



HTML Templates

Ofrece un modelo de construcción basado en plantillas inertes de código HTML que sólo son activadas cuando se renderiza el componente



HTML Imports

Ofrece un modelo de modularización basado en la posibilidad de incluir ficheros de código HTML dentro de otros ficheros HTML



Custom Elements

Ofrece un modelo de extensibilidad que permite a los desarrolladores definir sus propias etiquetas o redefinir semánticamente las etiquetas del estándar DOM

El Proyecto Polymer

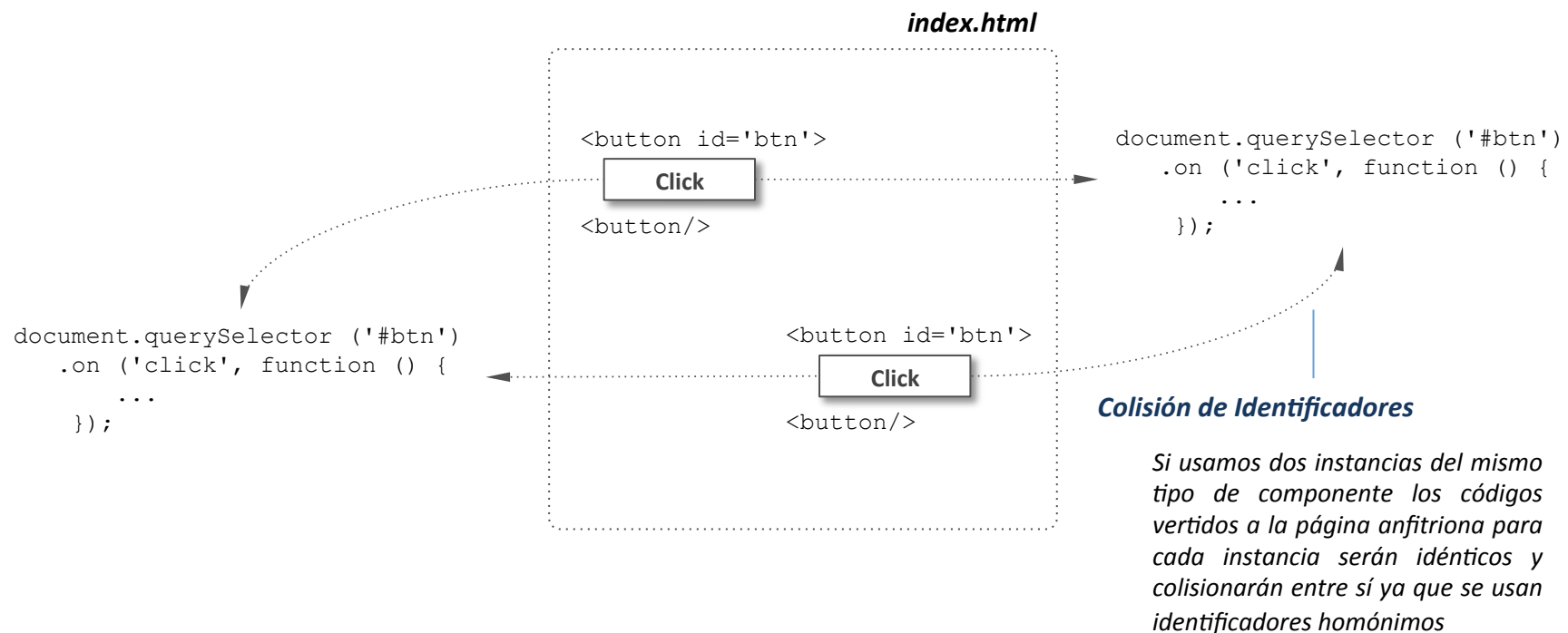
Estándares En Componentes Web



Modelo De Encapsulamiento. Shadow DOM

El Problema

Sin más infraestructura de soporte, los identificadores JS y CSS utilizados internamente dentro de cada componente podrían colisionar con los usados dentro de la página huésped o en otros componentes residentes. Es necesario proporcionar un mecanismo de encapsulamiento que aísle cada instancia de componente del resto de instancias y de la propia página.



El Proyecto Polymer

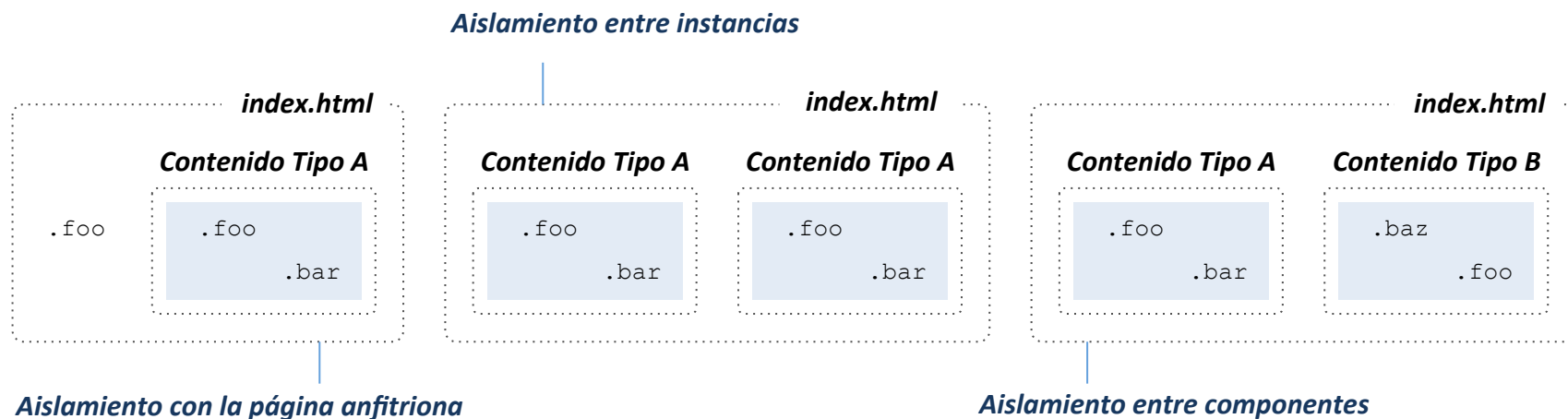
Estándares En Componentes Web



Modelo De Encapsulamiento. Shadow DOM

La Solución

Es necesario, por tanto, proporcionar un mecanismo de encapsulación que aísle cada instancia de componente del resto de instancias y de la propia página. desde un punto de vista conceptual, es posible clasificar los tipos de aislamiento necesarios en tres categorías diferentes de acuerdo a la relación de los elementos en potencial conflicto.



El Proyecto Polymer

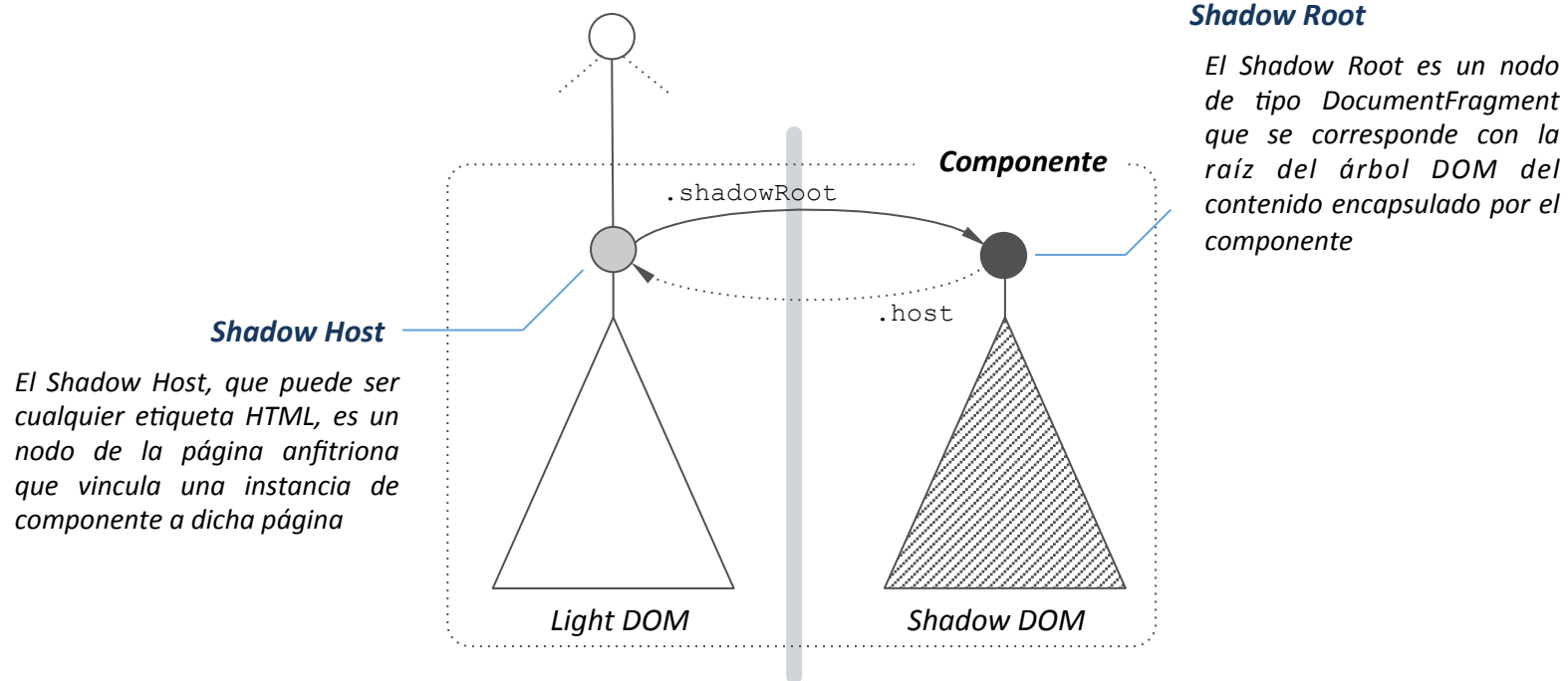
Estándares En Componentes Web



Modelo De Encapsulamiento. Shadow DOM

Shadow DOM & Shadow HOST

El proceso de encapsulación consiste, esencialmente, en generar nuevos árboles DOM para albergar los contenidos de cada instancia de componente Web y vincularlos a ciertos nodos de la página anfitriona. De esta manera se consigue que dichos contenidos residan en un espacio de nombres exclusivo y aislado del de los demás y del de la propia página.



El Proyecto Polymer

Estándares En Componentes Web



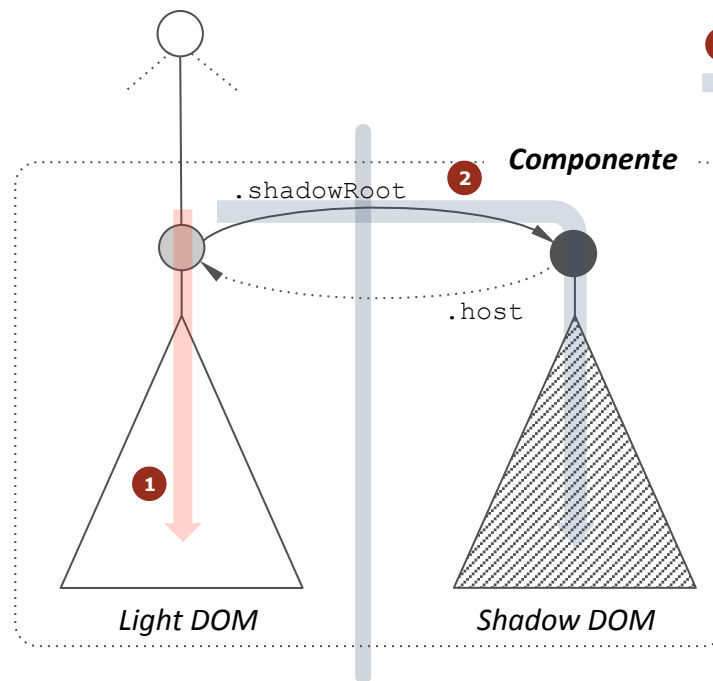
Modelo De Encapsulamiento. Shadow DOM

Shadow DOM & Shadow HOST

El proceso de encapsulación consiste, esencialmente, en generar nuevos árboles DOM para albergar los contenidos de cada instancia de componente Web y vincularlos a ciertos nodos de la página anfitriona. De esta manera se consigue que dichos contenidos residan en un espacio de nombres exclusivo y aislado del de los demás y del de la propia página.

1 **Renderizado normal**

En los navegadores sin soporte a componentes web se aplica el algoritmo de renderizado habitual. Como el shadow DOM no es accesible por este algoritmo, el componente no se renderiza y se continúa por el recorrido habitual



2 **Renderizado shadow**

Cuando el soporte a componentes web está activo, el algoritmo de renderizado discrimina entre nodos normales y nodos shadow host. En éstos últimos, deriva el proceso de renderizado para mostrar el contenido del shadow DOM mientras el Light DOM no se muestra

El Proyecto Polymer

Estándares En Componentes Web



Modelo De Encapsulamiento. Shadow DOM

Shadow DOM & Shadow HOST

Desde la especificación de Shadow DOM, la única manera de crear esta estructura dual de contenidos es a través de JavaScript. Para ello, se extiende el nodo Element con un nuevo método para la creación de nodos Shadow Root que se vinculan automáticamente al nodo sobre el que se aplican convirtiéndolo así en un Shadow Host.

```
<div id="myHost">
  contenido del light dom
</div>
```

Shadow Host

Cualquier tipo de nodo DOM puede ser convertido en Shadow Host. Para ello es necesario utilizar alguno de los mecanismos de referencia disponibles en HTML que permitan seleccionar el nodo desde JavaScript

Shadow Root

Primero se localiza el nodo que hará de anfitrión y se convierte en shadow host creando en él un nodo shadow root. Después sólo queda inyectar contenido HTML en este nuevo nodo por cualquier mecanismo estándar para crear el Shadow DOM

```
<script>
  var host = document.querySelector('#myHost');
  var root = host.createShadowRoot();
  root.innerHTML = "contenido del Shadow DOM";
</script>
```

El Proyecto Polymer

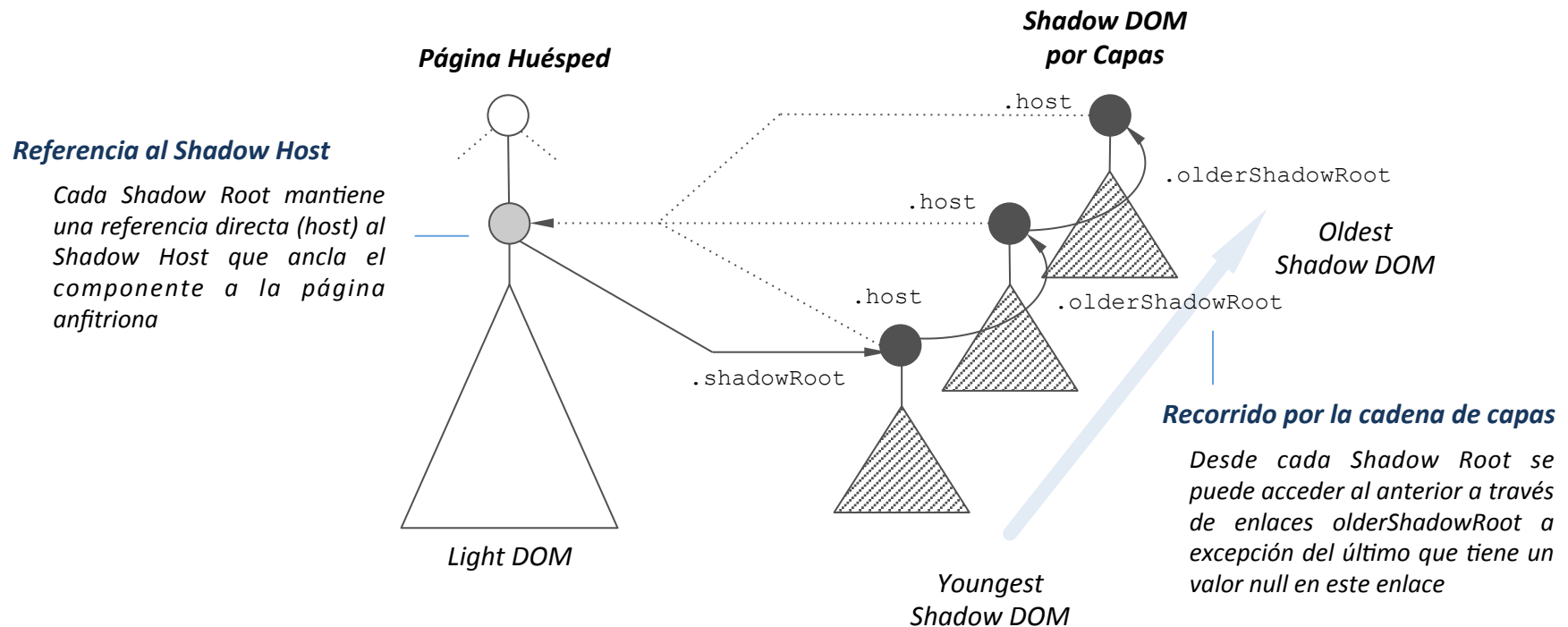
Estándares En Componentes Web



Modelo De Encapsulamiento. Shadow DOM

Shadow DOM Por Capas

Es posible definir múltiples árboles en la sombra organizados por capas. El Shadow Host sigue reteniendo un enlace al ShadowRoot más reciente. Desde éste se puede recorrer sucesivamente cada Shadow DOM a través de enlaces `olderShadowRoot`. Y por último, cada Shadow Root dispone de una referencia al Shadow Host.



El Proyecto Polymer

Estándares En Componentes Web



Modelo De Encapsulamiento. Shadow DOM

Shadow DOM Por Capas

Para crear una estructura multicapa simplemente es necesario invocar repetidamente el método `createShadowRoot` sobre el nodo Host. Cada vez que se llama a este método se crea un nuevo `ShadowRoot` que se establece como primera capa de acuerdo a la arquitectura de enlaces anterior. Una vez que se crea un enlace `Shadow Root` es imposible eliminarlo.

```
<div id="myHost">
  contenido del light dom
</div>
```

Shadow Host

Igual que antes es necesario hacer uso de alguno de los mecanismos de referencia de HTML para poder acceder al nodo DOM que hará de Shadow Host desde el código JavaScript

Shadow Roots

Cada vez que se invoca al método `createShadowRoot` se genera un nuevo Shadow DOM que se establece como primera capa dentro de la arquitectura de enlaces

```
<script>
  var host = document.querySelector('#myHost');
  var root1 = host.createShadowRoot();
  var root2 = host.createShadowRoot();
  var root3 = host.createShadowRoot();
  root1.innerHTML = ...
  root2.innerHTML = ...
  root3.innerHTML = ...
</script>
```

El Proyecto Polymer

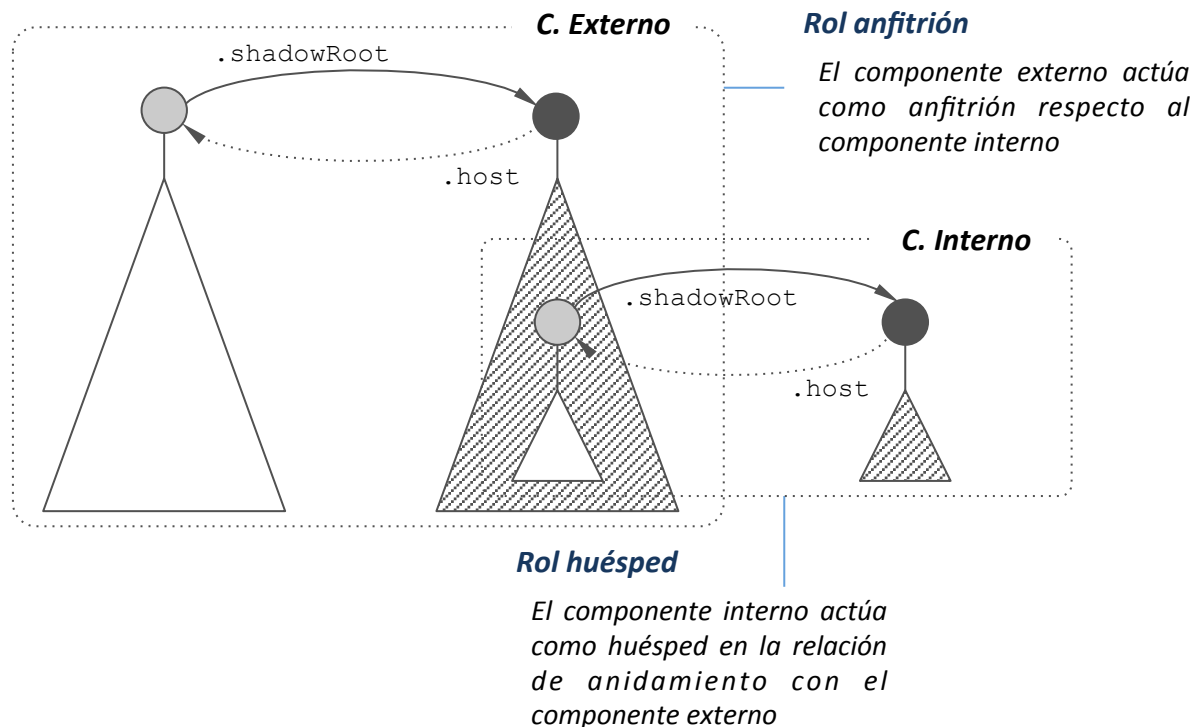
Estándares En Componentes Web



Modelo De Encapsulamiento. Shadow DOM

Shadow DOM Anidado

Los componentes Web pueden construirse a través de procesos de anidamiento de manera que un componente interno pueda hospedarse dentro del Shadow DOM de otro componente externo. El papel del modelo de encapsulación en este tipo de escenarios de anidamiento mantiene la garantía de la ausencia de colisión para cada componente participante.



El Proyecto Polymer

Estándares En Componentes Web



Modelo De Encapsulamiento. Shadow DOM

Shadow DOM Anidado

El proceso de creación de componentes anidados requiere identificar el nodo Host del componente externo y crear un Shadow Root donde se inyecta un contenido HTML. Este contenido incluye todo el cuerpo del Light DOM del componente interno. Finalmente es preciso crear un Shadow Root para el componente interno e inyectarle su contenido HTML.

```
antes del componente externo  
<div id="outerHost"> ... </div>  
después del componente externo
```

Componente externo

Se idéntica el nodo que servirá de anfitrión para hospedar el componente externo y poder accederlo desde JavaScript

Anidamiento de componente Interno

Primero se crea el Shadow Root del componente externo y se inyectan los contenidos de su Shadow DOM que encapsulan al Light DOM del componente interno. Después se crea el Shadow Root del componente interno y se le inyecta su contenido HTML

```
<script>  
  var outerHost = document.querySelector('#outerHost');  
  var outerRoot = outerHost.createShadowRoot();  
  outerRoot.innerHTML =  
    'antes del componente interno' +  
    ' <div id="innerHost"> Light DOM interno </div> ' +  
    'después del componente interno';  
  var innerHost = outerRoot.querySelector('#innerHost');  
  var innerRoot = innerHost.createShadowRoot();  
  innerRoot.innerHTML = 'en el componente interno';  
</script>
```

El Proyecto Polymer

Estándares En Componentes Web



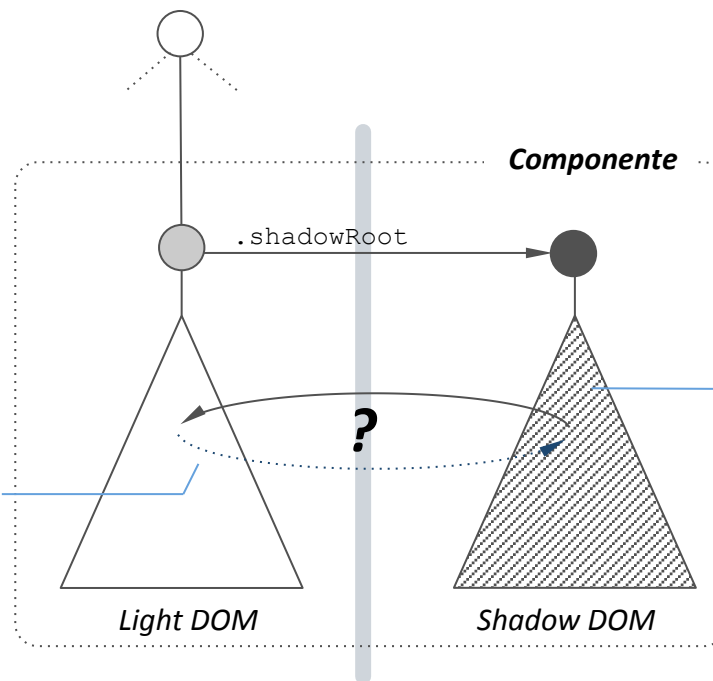
Modelo De Encapsulamiento. Shadow DOM

Distribución De Contenidos

En componentes simples el Shadow DOM puede requerir acceder de forma discrecional a la información de configuración contenida dentro del Light DOM para que forme parte del proceso de rendering del componente. Gracias a este mecanismo se consigue adaptar el aspecto y comportamiento de cada instancia de componente a su contexto de uso.

Información de configuración

El Light DOM contiene información semántica sobre la configuración del componente que permite adaptar su aspecto y comportamiento



Referencia de información

El shadow DOM podría necesitarse acceder de forma discrecional a partes de la información Light DOM para articular su renderizado. Necesitamos un mecanismo de referencia de información desde el Shadow DOM al Light DOM

El Proyecto Polymer

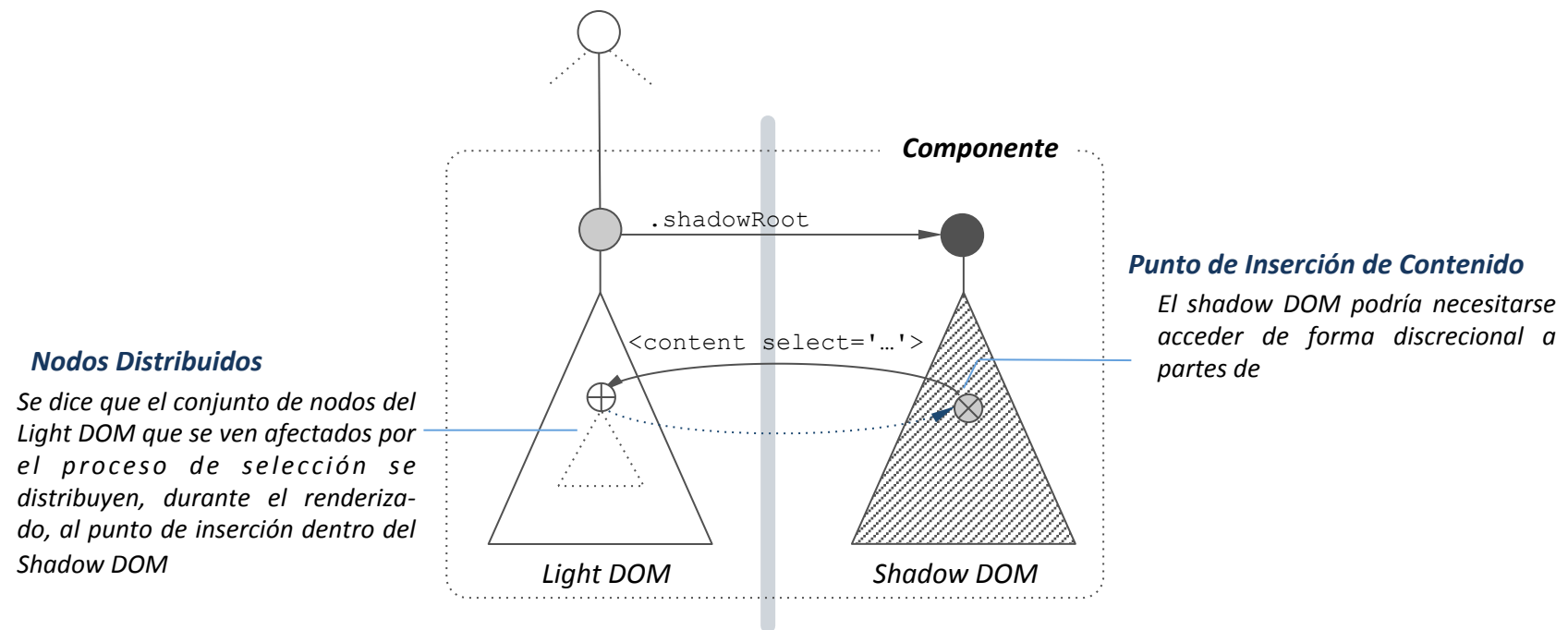
Estándares En Componentes Web



Modelo De Encapsulamiento. Shadow DOM

Distribución De Contenidos

En componentes simples el Shadow DOM puede requerir acceder de forma discrecional a la información de configuración contenida dentro del Light DOM para que forme parte del proceso de rendering del componente. Gracias a este mecanismo se consigue adaptar el aspecto y comportamiento de cada instancia de componente a su contexto de uso.



El Proyecto Polymer

Estándares En Componentes Web



Modelo De Encapsulamiento. Shadow DOM

Distribución De Contenidos

El diseño del Light DOM debe anotarse semánticamente de manera apropiada para que pueda referirse mediante expresiones de selección CSS a partir de las etiquetas content que figuran dentro del Shadow DOM. Hasta cierto punto puede considerarse que las etiquetas content son un mecanismo de inyección paramétrica de los valores mantenidos por el Light DOM.

```
<div class="product">
  <div class="type">Food</div>
  <div class="name">Pears</div>
</div>
```

Light DOM

El Light DOM contiene la información que va a ser referida desde los puntos de inserción de contenido presentes en el Shadow DOM

Shadow DOM & Content Select

El Shadow DOM contiene puntos de inserción para referir a la información del Light DOM

```
<script>
var host = document.querySelector('#myHost');
var root = host.createShadowRoot();
root.innerHTML =
  '<ul>' +
  '  <li>Type: <content select=".type"></content></li>' +
  '  <li>Name: <content select=".name"></content></li>' +
  '  <li>Amount: <content select=".amount">1</content></li>' +
  '</ul>'
</script>
```

El Proyecto Polymer

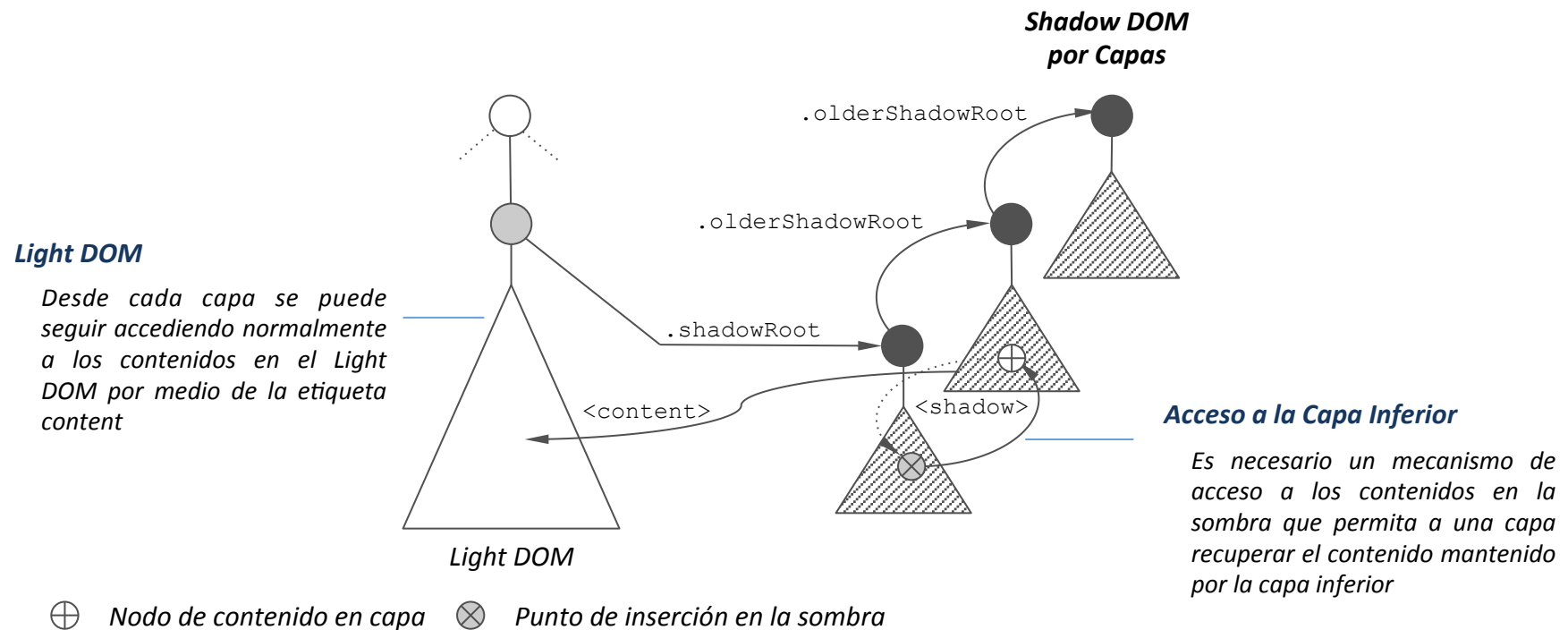
Estándares En Componentes Web



Modelo De Encapsulamiento. Shadow DOM

Distribución De Contenidos En La Sombra

En los escenarios de componentes contruidos por capas, es necesario, adicionalmente a lo anterior, disponer de un mecanismo que permita acceder desde cada capa a la información contenida en la inmediatamente inferior. De esta manera, cada capa se concibe como una decoración que contribuye a enriquecer el contenido de la que tiene por encima.



El Proyecto Polymer

Estándares En Componentes Web



Modelo De Encapsulamiento. Shadow DOM

Distribución De Contenidos En La Sombra

De manera similar a la distribución de contenidos existe una etiqueta shadow que permite a un shadow DOM capturar todo el contenido del del shadow DOM que se encuentra inmediatamente por debajo de él en la jerarquía de capas. Esta etiqueta no dispone sin embargo de filtro select.

Existen Capas Ocultas

Las capas que no tuvieran una referencia <shadow> desde capas superiores no serían incluidas en el proceso de renderizado

```
<script>
...
root1.innerHTML = 'Esta capa está oculta';
root2.innerHTML = 'D. <content select...';
root3.innerHTML =
  '<div class=".box">' +
  '  <shadow></shadow>' +
  '</div>';
</script>
```

El Proyecto Polymer

Estándares En Componentes Web



Modelo De Renderizado. HTML Templates

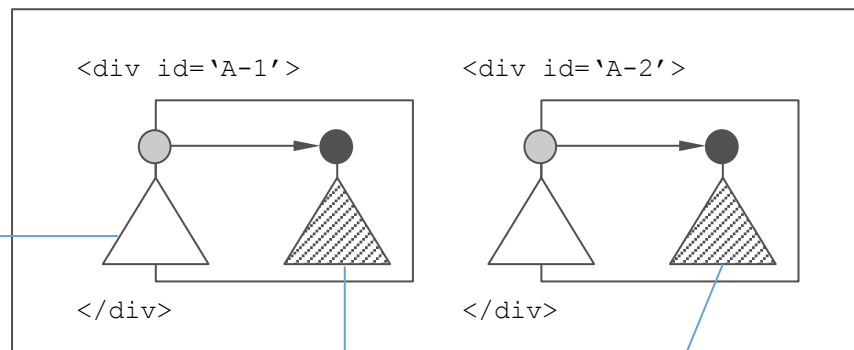
Plantillas Inertes Y Activación De Código Por Clonación

No es poco frecuente encontrar situaciones en las que es necesario instanciar dos ocurrencias diferentes de un mismo componente sobre la misma página, aunque sea con light DOM diferentes. ¿Pomo podemos evitar el trabajo de reescribir el shadow DOM entero para cada instancia de forma eficaz y segura?

Light DOM diferentes

Cada instancia sin embargo contiene unos contenidos semánticos distintos dentro del subárbol light que sirven para configurar la misma de forma particular

Página Huésped



Shadow DOM idénticos

El contenido de los dos subárboles shadow es el mismo puesto que ambas etiquetas responden a instancias distintas (-1 y -2) del mismo componente A

El Proyecto Polymer

Estándares En Componentes Web



Modelo De Renderizado. HTML Templates

Plantillas Inertes Y Activación De Código Por Clonación

No es poco frecuente encontrar situaciones en las que es necesario instanciar dos ocurrencias diferentes de un mismo componente sobre la misma página, aunque sea con light DOM diferentes. ¿Pomo podemos evitar el trabajo de reescribir el shadow DOM entero para cada instancia de forma eficaz y segura?

```
<template id="news">
  <img src="">
  <div class="body"></div>
</template>
<div id="new-1"></div> <div id="new-2"></div>
```

Plantilla Inerte

El código de la plantilla se procesa pero la carga de recursos no se produce hasta la activación

Activación

La activación de una plantilla consiste en un proceso de clonación del nodo DOM que mantiene los contenidos

```
<script>
function createNew(container, image, body) {
  var root = document.querySelector(container).createShadowRoot();
  var template = document.querySelector("#news").content;
  template.querySelector("img").src = image;
  template.querySelector(".body").textContent = body;
  root.appendChild(template.content.cloneNode(true));
}
createNew('#new-1', 'imagen1.png', 'noticia 1');
createNew('#new-2', 'imagen2.png', 'noticia 2');
</script>
```

El Proyecto Polymer

Estándares En Componentes Web



Modelo De Extensión. Custom Tags

Creación De Etiquetas Personalizadas. Custom Tags

Ahora que tenemos los mecanismos necesarios para definir y encapsular comportamiento y lógica de presentación personalizada necesitamos una manera de referirlo nominalmente dentro de nuestra página Web en forma de etiquetas personalizadas. Esto permite distinguir los componentes de las etiquetas estándar lo que aumenta la legibilidad de la página.

```
...  
<wc-calendar>  
  <wc-calendar-day>12</wc-calendar-day>  
  <wc-calendar-month>06</wc-calendar-month>  
  <wc-calendar-year>2014</wc-calendar-year>  
</wc-calendar>  
...
```

Regla 1. Nombres con al menos un guión

Los nombres de las etiquetas personalizadas deben tener al menos un guión en el nodeName para diferenciarlas de las etiquetas estándar.

```
<wc-calendar> ... </wc-calendar>
```

Regla 2. Nombres reservados

No pueden utilizarse como nombres de etiquetas personalizadas cualquiera de las cadenas con guión que se enumeran a continuación.

```
annotation-xml color-profile font-face  
font-face-src font-face-uri font-face-format font-face-  
name missing-glyph
```

El Proyecto Polymer

Estándares En Componentes Web



Modelo De Extensión. Custom Tags

Creación De Etiquetas Personalizadas. Custom Tags

Debemos preocuparnos, por un lado, de registrar cada componente construido como un nuevo tipo de etiqueta dentro del sistema de etiquetas del navegador y después de instanciar componentes tanto desde el lenguaje de marcado como a través de JS.

Registro de un Componente

```
var p = Object.create(HTMLElement.prototype);
p.foo = function () {...};
p.bar = 7;
var Foo = document.registerElement(
    'wc-foo',
    {prototype: p});
```

Creación

Primero se crea el prototipo del componente y se registra como una nueva etiqueta con la función registerElement

Instanciación de un Componente

```
<wc-foo id="a">
  ...
</wc-foo>
<wc-foo id="b">
  ...
</wc-foo>
```

```
var a = new Foo ();
var b = document.createElement('wc-foo');
document.body.appendChild (a);
document.body.appendChild (b);
```

Uso

Una vez creado el componente y registrado como una nueva etiqueta se puede usar como cualquier otra etiqueta estándar

El Proyecto Polymer

Estándares En Componentes Web



Modelo De Extensión. Custom Tags

Extensión de Etiquetas Estándar. Extended Element Types

Es posible extender las propias etiquetas del estándar HTML para generar variantes que incluyan nuevo comportamiento o lógica presentacional cuando se rendericen. Estas etiquetas mantienen, naturalmente, el nombre de la etiqueta a la que extienden pero incluyen un calificador `is` para indicar al navegador de que se trata de una variante distinta.

```
...  
<button id="btn" is="fancy-button">  
  Ok  
</button>  
  
<p is="lorem-ipsum"><p>  
...  
  
<script>  
  var btn = document.querySelector('#btn');  
  btn.addEventListener('click', function(e){  
    ...  
  });  
</script>
```

Extensión de Etiquetas Estándar

Las etiquetas del estándar se extienden con variantes cualificadas nominalmente dentro del atributo `is`. Estas etiquetas mantienen el nombre, métodos y atributos de la etiqueta de la que derivan pero pueden sobrescribir o agregar nuevas capacidades

Herencia de Capacidades

La nueva variante de botón incluye, por herencia todos los atributos, métodos y modelo de eventos que tiene la etiqueta base de la que deriva

El Proyecto Polymer

Estándares En Componentes Web



Modelo De Extensión. Custom Tags

Extensión de Etiquetas Estándar. Extended Element Types

De forma similar al caso anterior, comenzaremos por registrar cada componente construido como una variante que extiende una etiqueta del estándar y después nos ocuparemos de instanciar componentes tanto desde el lenguaje de marcado como a través de JS.

Registro de un Componente

```
var p = Object.create(HTMLButtonElement.prototype);
p.foo = function () {...};
p.bar = 7;
var FancyButton = document.registerElement(
    'fancy-button',
    {extends: 'button',
     prototype: p});
```

Creación

En este caso la creación requiere crear un prototipo que hereda del prototipo HTMLButtonElement

Instanciación de un Componente

```
<button is="fancy-button"
        id="a"> Ok </button>

<button is="fancy-button"
        id="b"> Ok </button>
```

```
var a = new FancyButton ();
var b = document.createElement
    ('fancy-button');
document.body.appendChild (a);
document.body.appendChild (b);
```

Uso

A diferencia de las etiquetas personalizadas, aquí se mantiene el nombre de la etiqueta y se cualifica semánticamente con el atributo is para favorecer el descubrimiento SEO

El Proyecto Polymer

Estándares En Componentes Web



Modelo De Modularización. HTML Imports

Importación de documentos HTML

La implementación de un componente incluye un código de plantilla inerte, especificaciones de estilo y no poca lógica de scripting para especificar el modelo de comportamiento. Parece necesario disponer de un mecanismo de modularización que permita incluir todo ese contenido en un fichero de índice a parte HTML que pueda importarte a la página huésped.

Index.html

```
<!doctype html>
<html lang="es">
  <head>
    <meta charset="utf-8">
    <script src="webcomponents.js"></script>
    <script src="polymer.js"></script>
    <link rel="import" href="my-lib.html">
  </head>

  <body>
    ...
  </body>
</html>
```

My-lib.html

```
<!doctype html>
<html lang="es">
  <head>
    <link rel="import" href="tag-1.html">
    <link rel="import" href="tag-2.html">
    <link rel="stylesheet" href="style-1.html">
    <link rel="stylesheet" href="style-2.html">
    <script src="script-1.js"></script>
    <script src="script-2.js"></script>
  </head>
  <body>
    <!-- codigo del componente -->
    ...
  </body>
</html>
```

style-1.css

script-1.js

tag-1.html

El Proyecto Polymer

Estándares En Componentes Web



Modelo De Modularización. HTML Imports

Acceso al Recurso Importado y a la Página Anfitriona

La implementación de un componente incluye un código de plantilla inerte, especificaciones de estilo y no poca lógica de scripting para especificar el modelo de comportamiento. Parece necesario disponer de un mecanismo de modularización que permita incluir todo ese contenido en un fichero de índice a parte HTML que pueda importarte a la página huésped.

Index.html

```
<!doctype html>
<html lang="es">
  <head>
    ...
    <link rel="import" href="file.html">
  </head>

  <body>
    ...
  </body>
</html>
```

file.html

```
<!doctype html>
<html lang="es">
  ...
  <body>
    <script>
      var f = document.currentScript.ownerDocument;
      var h = document;
      // f refiere a este documento HTML
      // h refiere a la página huésped
    </script>
  </body>
</html>
```

I. Contexto de Anfitrión

El contexto en el que se evalúan los script dentro de una página importada es el de la página anfitriona

II. Evaluación Secuencial

Las importaciones son no bloqueantes. Sin embargo los script dentro de la página importada se ejecutan en orden

III. Evaluación Única

Las importaciones desde una misma URL son procesadas sólo una vez con lo que sus script sólo se ejecutan una vez

El Proyecto Polymer

Estándares En Componentes Web



Modelo De Modularización. HTML Imports

Eventos de Carga y Error

Para controlar los procesos de carga de recursos de documentos HTML importados mediante la directiva import se pueden inyectar manejadores a eventos de éxito y error.

```
<script async>
  function handleLoad(e) {
    console.log('Loaded import:' + e.target.href);
  }
  function handleError(e) {
    console.log('Error loading import:' + e.target.href);
  }
</script>

<link rel="import" href="file.html"
      onload="handleLoad(event)" onerror="handleError(event)">
```

```
<head>
  <link id="foo-link" rel="import" href="wc-foo.html">
</head>
<body>
  ...
  <script>
    var link = document.querySelector('#foo-link');
    var content = link.import;
    var el = content.querySelector('...');
    ...
  </script>
</body>
```

Manipulación De Contenidos

El desarrollador tiene la oportunidad de capturar programáticamente el contenido importado mediante una directiva import y operar sobre él a conveniencia.

Javier Vélez Reyes
@javiervelezreye
Javier.velez.reyes@gmail.com

El Framework Polymer

- Modelo de Componentes Polymer
- Extensiones al Modelo de Plantillas
- Extensiones al Modelo de Componente

El Proyecto Polymer

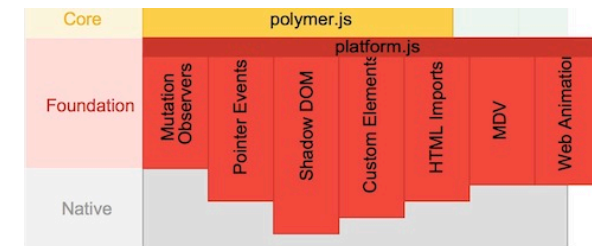
Introducción

El Proyecto Polymer

El Framework Polymer

El Framework Polymer

Google ha desarrollado, por encima de la implementación de los estándares de Componentes Web, un framework dirigido a simplificar los procesos de desarrollo con esta tecnología. A lo largo de este capítulo veremos cuales son sus ventajas particulares.



Modelo de Componente Polymer



Custom Elements

Ofrece un modelo de extensibilidad que permite a los desarrolladores definir sus propios elementos o redefinir los elementos del estándar

Atributos
Ciclo de Vida
Observers



HTML Templates

Ofrece un modelo de construcción basado en plantillas inertes de código HTML que se renderizan cuando se necesita el contenido.

Data Binding
Plantillas Dinámicas

El Proyecto Polymer

El Framework Polymer



Modelo De Componentes De Polymer

Componentes Web Sin Lógica

La definición de Componentes Web sin lógica funcional con Polymer es un ejercicio completamente declarativo mediante la etiqueta `polymer-element` y el modificador `noscript`.



Web Components



Polymer

```
<script>
  var host = document.querySelector('#myHost');
  var root = host.createShadowRoot();
  root.innerHTML = "contenido del Shadow DOM";
</script>
```

```
<polymer-element name="profile-card" noscript>
  <template>
    ...
  </template>
</polymer-element>
```

El atributo `noscript` incluye una lógica de configuración por defecto para los componentes sin lógica funcional

El Proyecto Polymer

El Framework Polymer



Modelo De Componentes De Polymer

Componentes Web Con Lógica

Cuando el componente tiene lógica funcional se añade un script que se vincula automáticamente a la plantilla HTML adjunta sin utilizar el atributo noscript.



Web Components



Polymer

```
<script>
  var host = document.querySelector('#myHost');
  var root = host.createShadowRoot();
  root.innerHTML = "contenido del Shadow DOM";
</script>
```

```
<polymer-element name="profile-card">
  <template>
    ...
  </template>
  <script>
    ...
  </script>
</polymer-element>
```

Los componentes con lógica funcional añaden una etiqueta script que opera sobre los contenidos de la plantilla adjunta

El Proyecto Polymer

El Framework Polymer



Modelo De Componentes De Polymer

Extensión De Componentes Web

Para definir un Componente Web como extensión de otro componente o de una etiqueta estándar basta con usar el par de atributos extends / is.



Web Components



Polymer

Después para su uso se utiliza la etiqueta del padre y se cualifica semánticamente al componente con el atributo is

```
var p = Object.create(HTMLButtonElement.prototype);  
var FancyButton = document.registerElement(  
    'fancy-button',  
    {extends: 'button', prototype: p});
```

```
<polymer-element name="fancy-button" extends="button">  
    ...  
</polymer-element>
```

```
<button is="fancy-button"> ... </button>
```

Para declarar un componente como extensión de otro se utiliza el atributo extends en la definición del mismo

El Proyecto Polymer

El Framework Polymer



Extensiones Al Modelo De Custom Elements

Atributos & Ciclo de Vida

La lógica de comportamiento del componente incluye un modelo de datos, un conjunto de métodos de ciclo de vida y un manejador de cambios de los atributos.

El modelo de datos se soporta a partir de la declaración de variables locales. Ojo los tipos primitivos operan a nivel de instancia mientras que los objetos a nivel de prototipo

```
<polymer-element name="wc-card" attributes="name age">
  ...
  <script>
    Polymer ('wc-card', {
      this.name,
      this.age: 18,

      created : function () {},
      ready   : function () {},
      attached : function () {},
      domReady : function () {},
      detached : function () {},

      ...
    });
  </script>
</polymer-element>
```

Los manejadores del ciclo de vida permiten incluir código que trata los eventos vinculados a la creación y destrucción de la instancia

El Proyecto Polymer

El Framework Polymer



Extensiones Al Modelo De Custom Elements

Métodos & Observadores

La lógica de comportamiento del componente incluye un modelo de datos, un conjunto de métodos de ciclo de vida y un manejador de cambios de los atributos.

Existen dos formas para gestionar el cambio de valor de los atributos. Se puede usar el método genérico `attributeChanged` o definir manejadores de cambio dedicados

```
<polymer-element name="wc-card" attributes="name age">
  ...
  <script>
    ...
    adult : function (){ return this.age > 18 },
    name  : function (name){ ... },
    age   : function (age){ ... },

    attributeChanged : function (att, ov, nv){},
    nameChanged      : function (ovalue, nvalue){},
    ageChanged       : function (ovalue, nvalue){}

  });
  </script>
</polymer-element>
```

Los métodos de negocio se incluyen tal cual al prototipo del componente definido

El Proyecto Polymer

El Framework Polymer



Extensiones Al Modelo De Plantillas

Data Binding Bidireccional

Polymer vincula cada referencia en el contenido de las plantillas HTML al modelo de datos subyacente de manera que los cambios se propagan automáticamente.

Los valores de los atributos del modelo de datos se propagan a la vista de manera automática

```
<polymer-element name="wc-card" attributes="name age">
  <template>
    Hola, mi nombre es {{ name }}
    y tengo {{ age }} años
  </template>

  <script>
    Polymer ('wc-card') {
      this.name,
      this.age: 18,
      ...
    }
  </script>
</polymer-element>
```

```
<polymer-element name="wc-card" attributes="name age">
  <template>
    Hola, mi nombre es {{ name }}
    y tengo {{ age }} años
  </template>

  <script>
    Polymer ('wc-card') {
      this.name,
      this.age: 18,
      ...
    }
  </script>
</polymer-element>
```

Asimismo, las referencias sobre elementos de edición HTML que causan cambios en los atributos provocan actualizaciones en el modelo subyacente.

Recíprocamente los cambios sobre los atributos que se producen sobre la vista actualizan el modelo subyacente

El Proyecto Polymer

El Framework Polymer



Extensiones Al Modelo De Plantillas

Plantillas Dinámicas

Polymer extiende la etiqueta template para dotarla de capacidades de renderizado dinámico. A continuación describimos las principales capacidades.

Enlace a Texto

```
<p>{{user.name}}</p>
```

Enlace a Atributo

```

```

Manejador de Evento Estándar

```
<button on-click="{{doClick}}">
  Ok
</button>
```

Enlace a Atributo Condicional

```
<div class="card" hidden?="{{show}}">
```

Manejador de Evento de Negocio

```
<div id="my-component"
  on-done="{{doAlarm}}">
  ...
</div>
```

Enlace con Operador Condicional

```
<template repeat="{{item, i in items}}">
  <div class="{{(i%2==0)? 'on' : 'off'}}">
    {{item}}
  </div>
</template>
```

El Proyecto Polymer

El Framework Polymer



Extensiones Al Modelo De Plantillas

Plantillas Dinámicas

Polymer extiende la etiqueta template para dotarla de capacidades de renderizado dinámico. A continuación describimos las principales capacidades.

Plantilla de Enlace de Datos

```
<template bind="{{person as me}}">
  {{me.name}} - {{me.twitter}}
</template>
```

Plantilla Condicional

```
<template bind="..." if="truthy">
  {{me.name}} - {{me.twitter}}
</template>
```

Plantilla Iterativa

```
<template repeat="{{item in items}}">
  <li>{{item}}</li>
</template>
```

Plantillas Referidas & Anidadas

```
<template repeat="i in items" id="t">
  <li>{{i}}
  <ul>
    <template repeat="{{i.hijos}}" ref="t">
      </template>
    </ul>
  </template>
  ...

<template bind ref="t">
</template>
```

Plantillas Anidada

```
<template repeat="{{u in users}}">
  <template repeat="{{i in u.items}}">
    <li>{{i}}</li>
  </template>
</template>
```

Javier Vélez Reyes
@javiervelezreye
Javier.velez.reyes@gmail.com

4 *Ecosistema de Componentes de Polymer*

- Componentes Web De Polymer
- Core Elements
- Paper Elements
- Desarrollo de Aplicativos Basados en Componentes

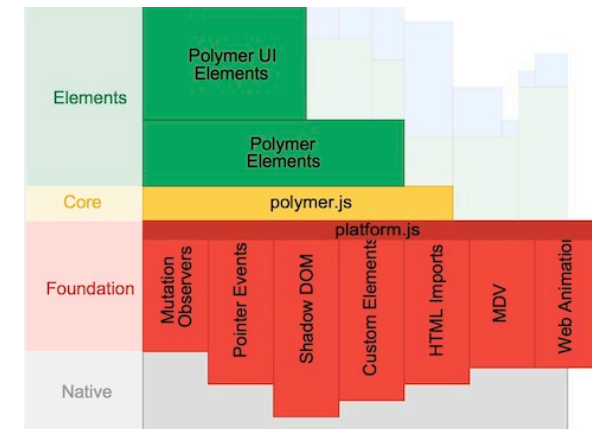
El Proyecto Polymer
Introducción

El Proyecto Polymer

Ecosistema de Componentes de Polymer

Componentes Web De Polymer

Google ha desarrollado, por encima de la implementación de los estándares de Componentes Web, un framework dirigido a simplificar los procesos de desarrollo con esta tecnología. A lo largo de este capítulo veremos cuales son sus ventajas particulares.



Core Elements

Los componentes core-element de Polymer dan soporte a capacidades nucleares relacionadas con la gestión de la lógica de presentación de componentes Web

Paper Elements

Los componentes paper-element de Polymer se encargan de articular el modelo de presentación basado en Material Design propio de Google

El Proyecto Polymer

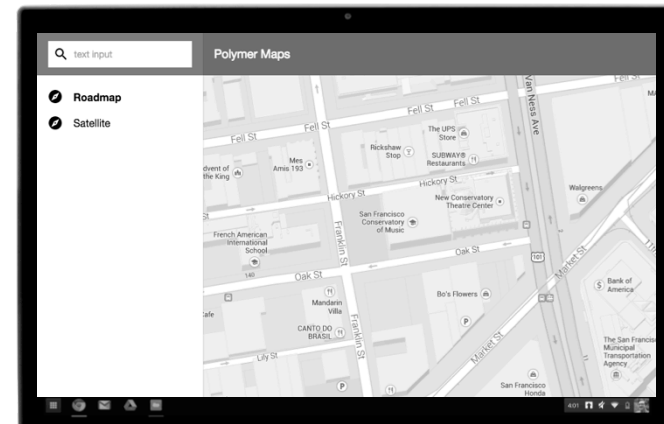
Ecosistema de Componentes de Polymer

Core Elements

Core Header Panel

Con Polymer es posible crear aplicaciones por composición de componentes Web. Core header panel es un contenedor simple con una sección de cabecera y otra de contenido.

```
<core-header-panel mode="scroll" flex>
  <core-toolbar>
    <core-icon-button icon="menu">
    </core-icon-button>
    <div>MY APP</div>
  </core-toolbar>
  <div class="content">...</div>
</core-header-panel>
```



Core Drawer Panel

El componente core-header-panel es un contenedor responsive que combina un menú lateral deslizante con un área de contenido principal.

```
<core-drawer-panel>
  <div drawer> Drawer panel... </div>
  <div main> Main panel... </div>
</core-drawer-panel>
```

El Proyecto Polymer

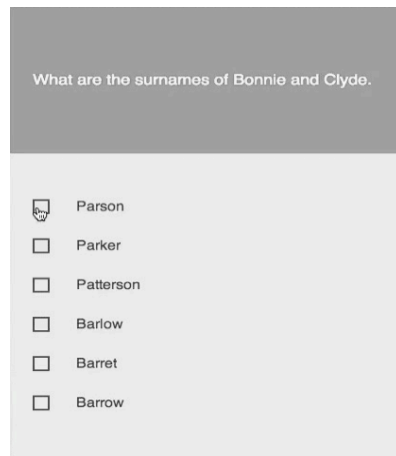

Ecosistema de Componentes de Polymer

Paper Elements

Paper Checkbox

Polymer también ofrece una amplia librería de componentes que implementan las normas de estilo de Material Design. Paper-checkbox es un ejemplo de ello.

```
<paper-checkbox>
...
</paper-checkbox>
```

A screenshot of a web form. At the top, a grey header bar contains the text "What are the surnames of Bonnie and Clyde." Below this, a list of six checkboxes is displayed, each followed by a surname: Parson, Parker, Patterson, Barlow, Barret, and Barrow. The first checkbox, for "Parson", is checked and has a mouse cursor hovering over it.A screenshot of a web form titled "Text fields and dialogs". It features two text input fields. The first field is labeled "Type something..." and the second is labeled "Type only numbers... (floating)". Below the second field is a dark grey button with the text "DIALOG" in white capital letters.

Paper Input

Paper-input es un componente que sustituye a la etiqueta estándar input de HTML ofreciendo un comportamiento más dinámico y mayor riqueza en la experiencia de usuario.

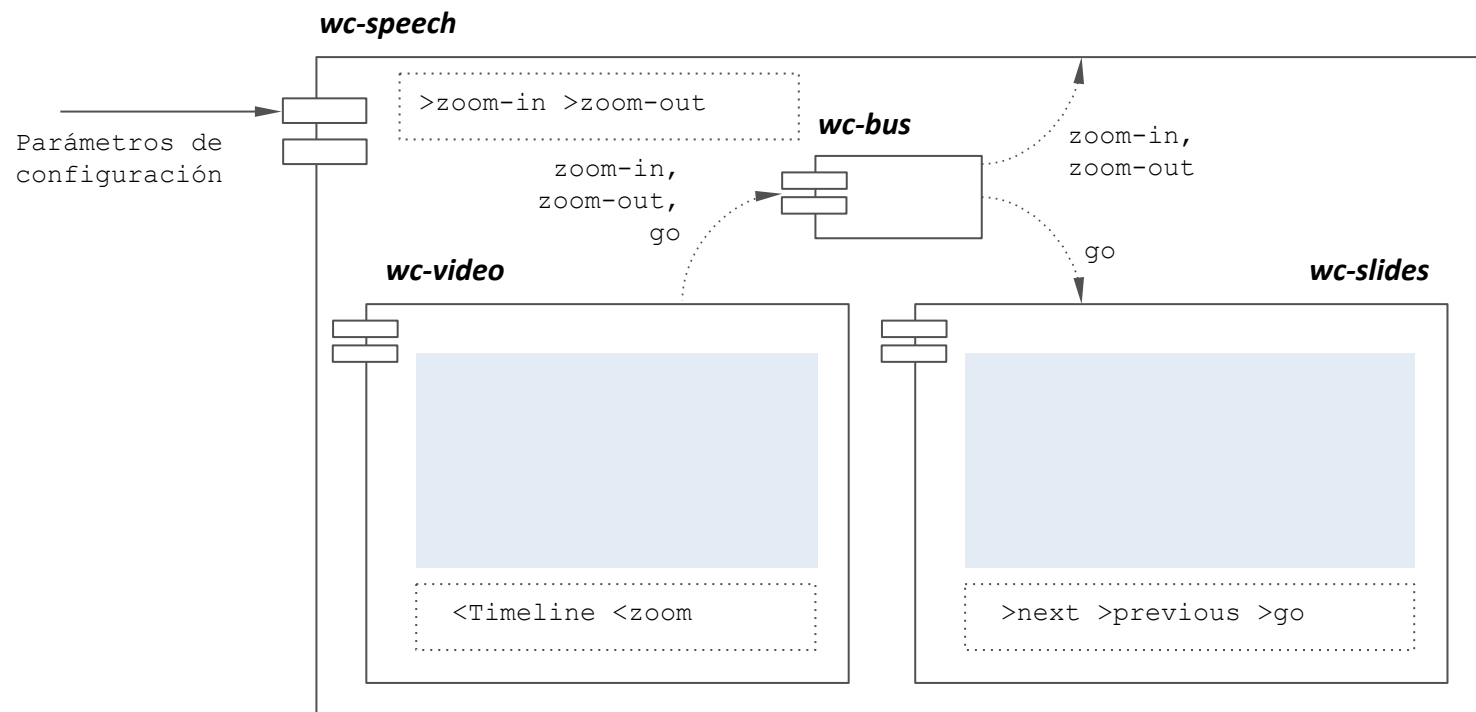
```
<paper-input label="Nombre">
...
</paper-input>
```

El Proyecto Polymer

Ecosistema de Componentes de Polymer

Desarrollo de Aplicativos Basados en Componentes

El desarrollo de aplicaciones orientadas a Componentes Web requiere de cierta infraestructura de control dedicada a articular procesos de coordinación y orquestación entre los componentes constituyentes. Esta lógica puede encapsularse en componentes de control reutilizables. Veamos un ejemplo.



El Proyecto Polymer

Preguntas

**Estándares de
Componentes Web**

**El Framework
Polymer**

**El Ecosistema de
Componentes**



Javier Vélez Reyes

@javiervelezreye

Javier.velez.reyes@gmail.com

Programación Orientada a Componentes Web

El Proyecto Polymer

Javier Vélez Reyes

@javiervelezreye
Javier.velez.reyes@gmail.com

Febrero 2015

