

Programación Orientada a Componentes Web

Principios de Diseño en Componentes Web

Javier Vélez Reyes

@javiervelezreye

Javier.velez.reyes@gmail.com

Marzo 2015



Principios de Diseño en Componentes Web

Autores

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



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 son los Componente Web
- Por Qué los Componentes Web
- Por Qué Principios de Diseño para Componentes Web
- El Paradigma de Componentes Web

Principios de Diseño en Componentes Web

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.

”

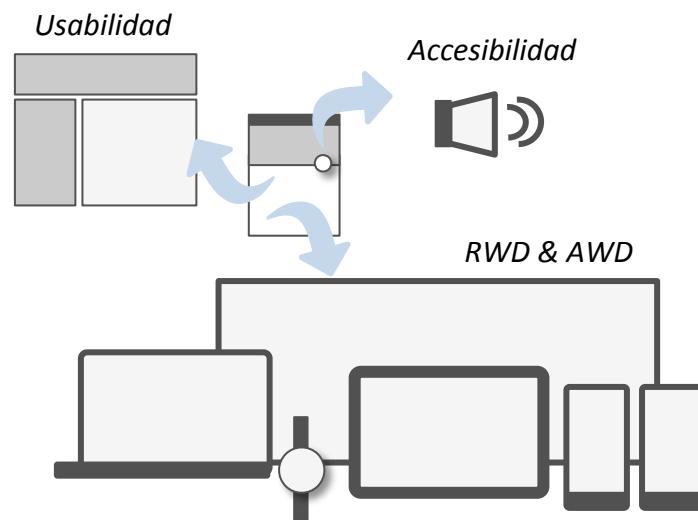
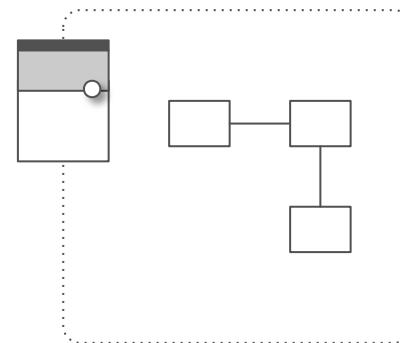
Principios de Diseño en Componentes Web

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.

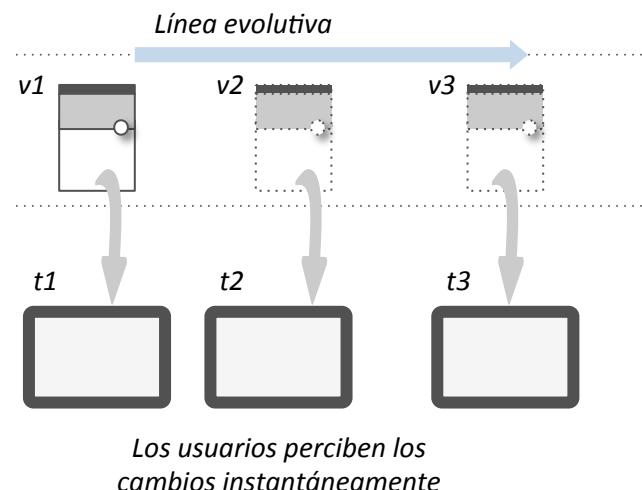
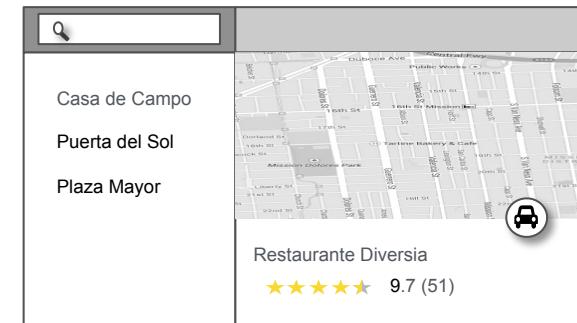
Principios de Diseño en Componentes Web

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.

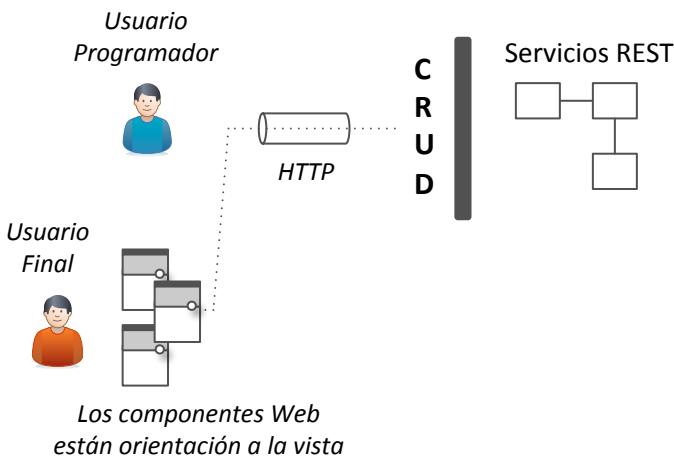
Principios de Diseño en Componentes Web

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.

Principios de Diseño en Componentes Web

Introducción

Por Qué Principios de Diseño Para Componentes Web

“

WebComponents.org is where pioneers and community-members of the Web Components ecosystem (like Polymer, X-tags, and other interested parties) document web components best practices so that others can follow the same path instead of needlessly striking out on their own.



- Zeno Rocha

”

¿Cómo se que mi componente es suficientemente reutilizable?

¿Cómo se cuándo debo aplicar encapsulación o simple agregación?

¿Cómo se que mi componente va a perdurar en el tiempo?

¿Cómo se si mi componente se integrará bien con los demás?

¿Cómo se que mi componente está preparado para utilizarse en muchos contextos?

?



Principios de Diseño en Componentes Web

Introducción

Por Qué Principios de Diseño Para Componentes Web

Podemos exponer gráficamente todos los objetivos primarios y secundarios vinculados a la tecnología de componentes Web. Los principios de diseño nos ayudarán a alcanzar dichos objetivos.



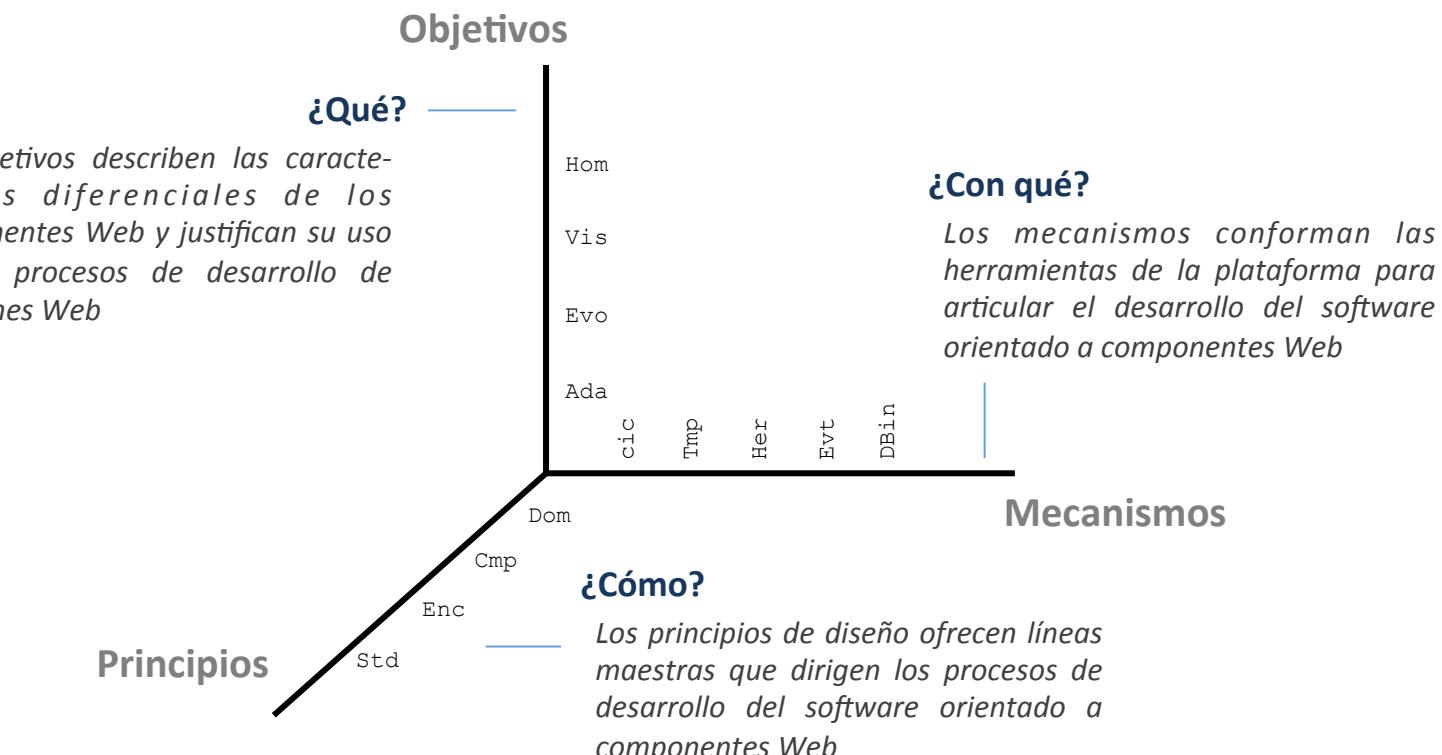
Principios de Diseño en Componentes Web

Introducción

El Paradigma de Componentes Web

Ejes Dimensionales

En torno al concepto de componente Web se pueden describir tres ejes dimensionales. Los objetivos determinan los propósitos perseguidos, los mecanismos caracterizan las capacidades nativas de plataforma y los principios de diseño ofrecen directrices generales de actuación.



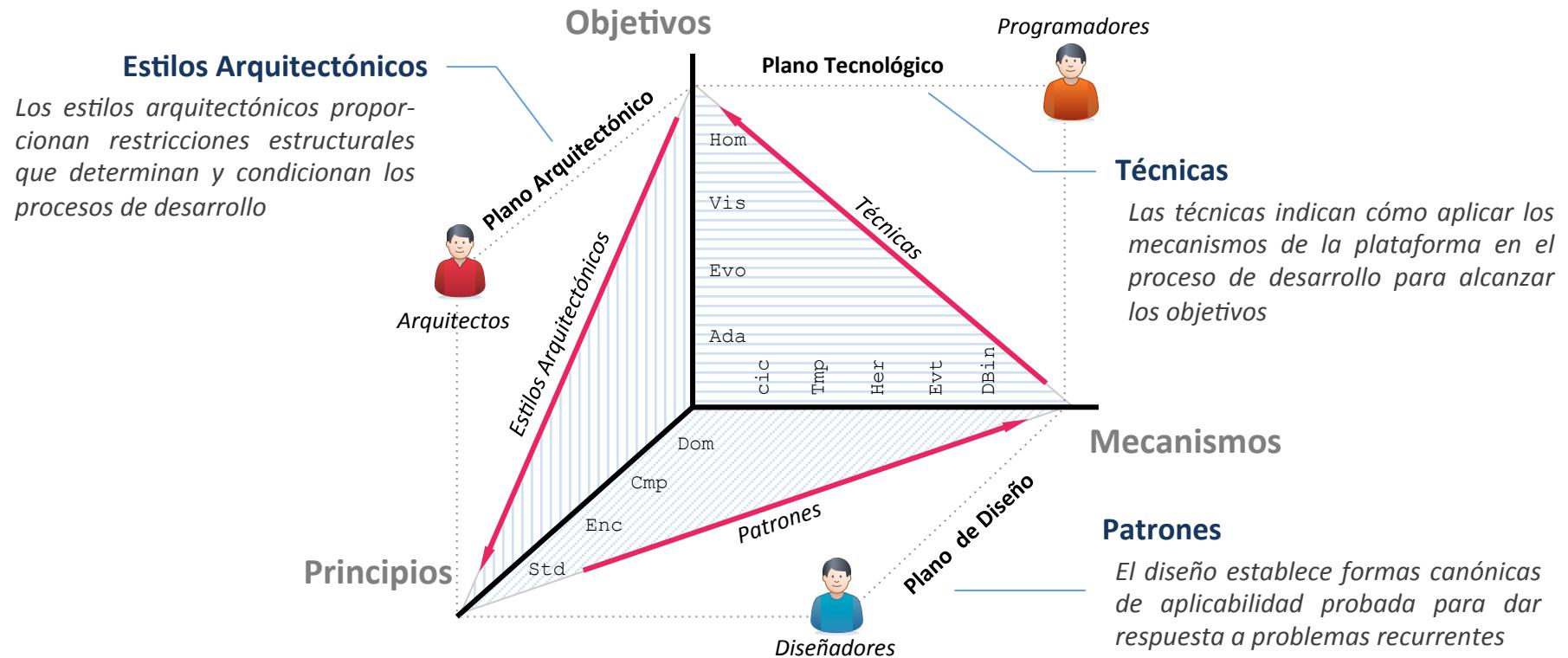
Principios de Diseño en Componentes Web

Introducción

El Paradigma de Componentes Web

Planos de Actividad

La intersección de las dimensiones por pares configura tres planos de actividad. En el plano arquitectónico se definen los estilos arquitectónicos o modelos de programación soportados por el paradigma. En el plano de diseño se establecen los patrones fundamentales. Y en el plano tecnológico se describen las técnicas utilizadas.



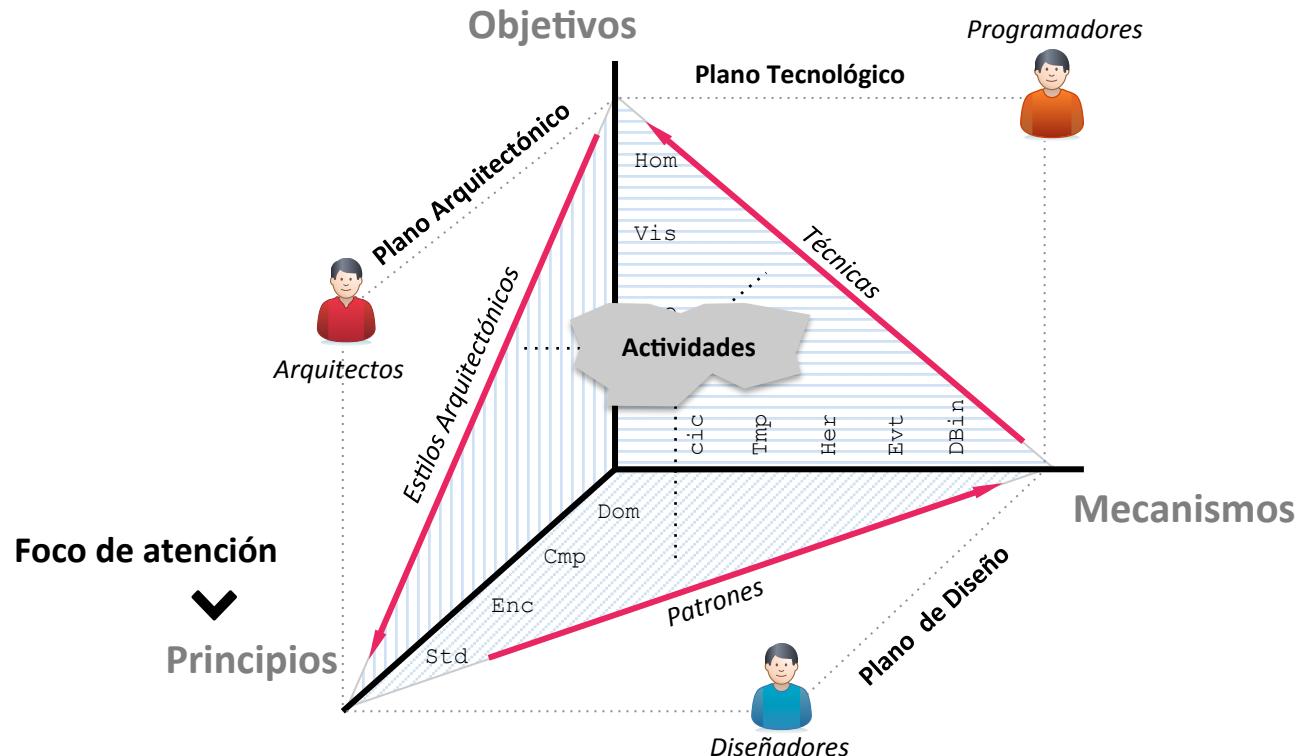
Principios de Diseño en Componentes Web

Introducción

El Paradigma de Componentes Web

Actividades

Sobre este espacio dimensional pueden ubicarse las actividades que conforman los procesos de desarrollo de software orientado a componentes Web. En lo que resta de este trabajo nos centraremos en describir los principios de diseño fundamentales. Otros elementos serán cubiertos en próximos trabajos.



Javier Vélez Reyes

@javiervelezreye

Javier.velez.reyes@gmail.com

Principios de Diseño

- Introducción
- Principios de Diseño en el Stream OOP
- Principios de Diseño en el Stream COP
- Principios de Diseño en el Stream SOC

Principios de Diseño en Componentes Web

Introducción

Principios de Diseño en Componentes Web

Principios de Diseño

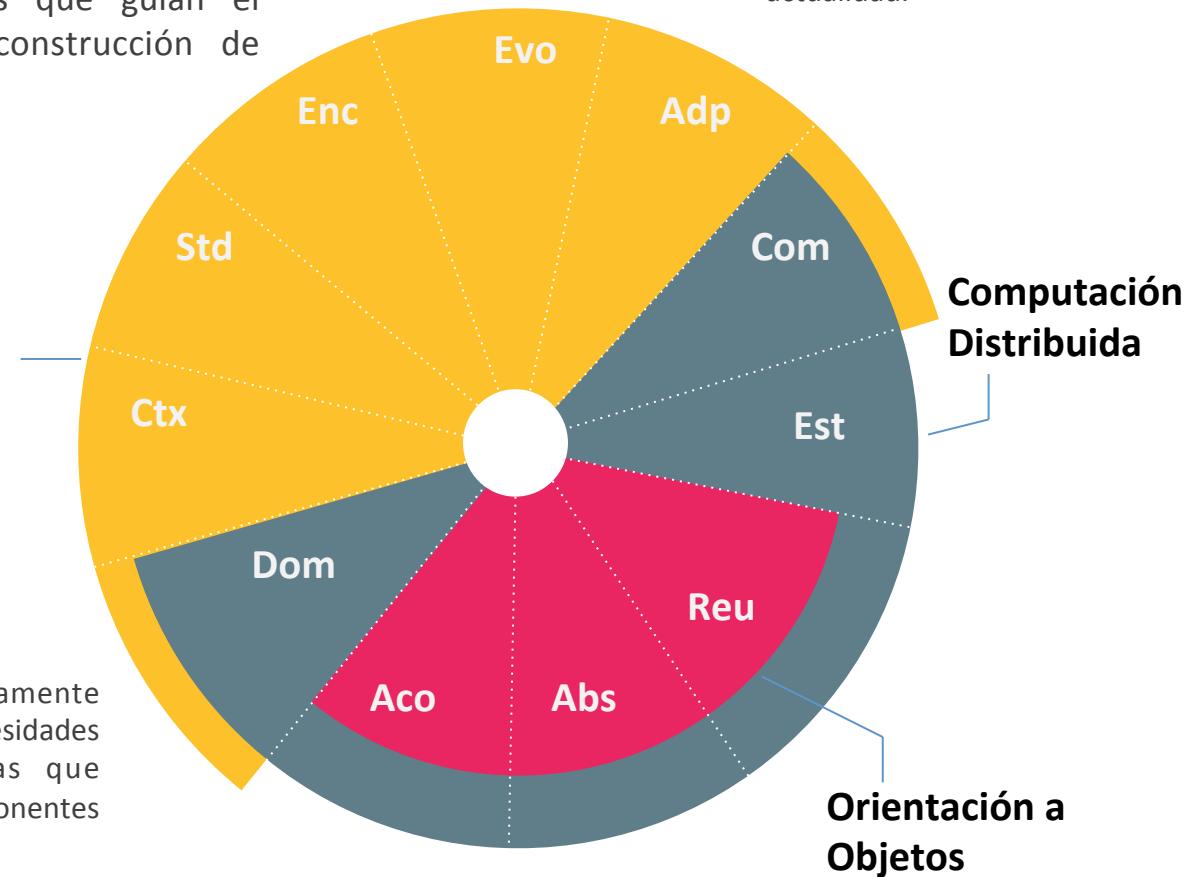
Introducción

A lo largo de este capítulo revisaremos los principios fundamentales que guían el proceso de diseño y construcción de componentes Web.

Este material entraña con los streams fundamentales de desarrollo vigentes en la actualidad.

Orientación a Componentes

Su contenido ha sido cuidadosamente adaptado para encajarlos en las necesidades y peculiaridades arquitectónicas que demanda la tecnología de componentes Web.



Principios de Diseño en Componentes Web

Principios de Diseño

Introducción

La descripción detallada de cada uno de los principios que se enumeran en este trabajo se realizará en torno a un esquema de características que recogen de manera sumaria las preocupaciones esenciales que dichos principios abordan. A continuación presentamos cuáles son tales características.

Nombre	Descripción	Actores
Describe el nombre del principio	<p>Principio de Diseño Centrado en el Dominio</p> <p>Nombre Diseño Centrado en el Dominio</p> <p>“ Comienza siempre realizando una adecuada división de responsabilidades en el ámbito de tu dominio y garantizando la ausencia de huecos y solapamientos entre componentes.”</p>	<p>Actores</p> <p>Analista de Dominio Arquitecto de Componentes</p>

Principios de Diseño en Componentes Web

Principios de Diseño

Introducción

La descripción detallada de cada uno de los principios que se enumeran en este trabajo se realizará en torno a un esquema de características que recogen de manera sumaria las preocupaciones esenciales que dichos principios abordan. A continuación presentamos cuáles son tales características.

Facetas
Cada principio se analiza desde una colección de facetas complementarias

Explicación
Las facetas se acompañan de explicaciones que ayudan a entender su relevancia

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Diseño Centrado en el Dominio

Nunca Defines Componentes Aislados

Por norma general, cuando trabajas construyes familias de componentes cohesionados que incluye en una biblioteca de etiquetas personalizadas.

Un componente aislado puede ser puntualmente útil pero frecuentemente forma parte de una colección de otros componentes.

Es frecuente construir componentes por familias que operan de manera cooperativa

Los componentes aislados no suelen ser frecuentes en el paradigma de componentes

</> En este ejemplo se ve como los componentes trabajan frecuentemente de manera conjunta.

```
<google-map latitude="37.77493" longitude="-122.41942" fitToBounds>
  <google-map-marker latitude="37.779" longitude="-122.3892" draggable="true" title="Go Giants!">
  </google-map-marker>
</google-map>
```

El componente google-map-marker opera siempre en el ámbito light DOM de un componente google-map.

18 @javiervelezreye @nmizar

Ejemplo
Los ejemplos ilustran la aplicación de cada faceta en la práctica y en su mayoría son extraídos del código y documentación de Polymer

Principios de Diseño en Componentes Web

Principios de Diseño

Introducción

La descripción detallada de cada uno de los principios que se enumeran en este trabajo se realizará en torno a un esquema de características que recogen de manera sumaria las preocupaciones esenciales que dichos principios abordan. A continuación presentamos cuáles son tales características.

Objetivos
Se describe la colección de objetivos que se persiguen con la aplicación del principio

Región
Se ilustran las regiones en el marco de la arquitectura donde impacta la aplicación del principio



Motivación
Se ofrece el fundamento que justifica y motiva el principio

Fases
Se describen las fases de aplicación del principio dentro de los proyectos de construcción o uso de componentes

Principios de Diseño en Componentes Web

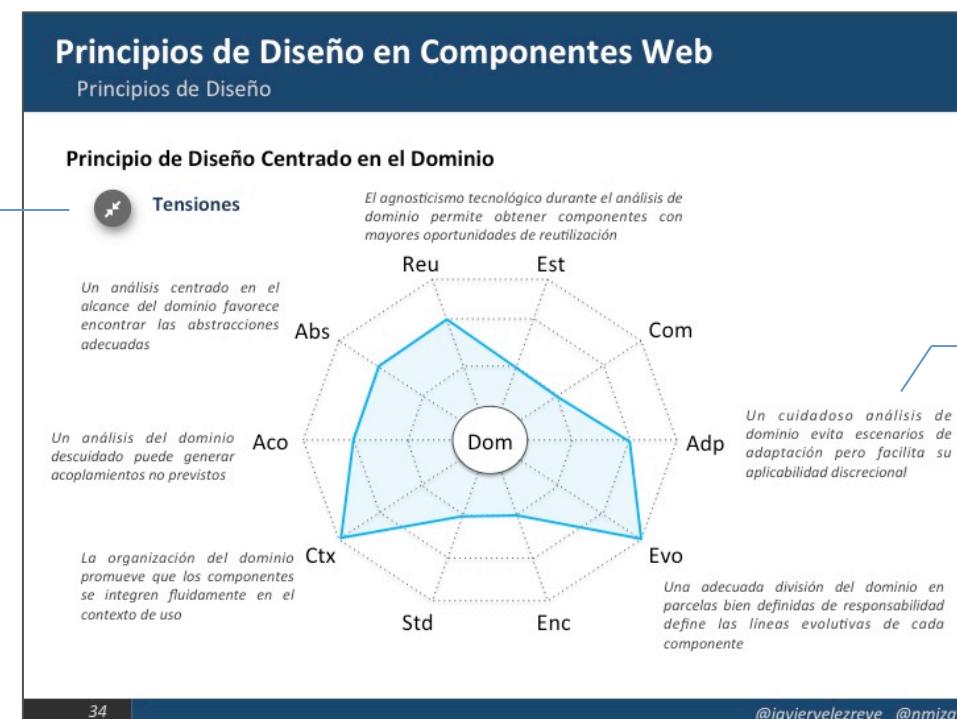
Principios de Diseño

Introducción

La descripción detallada de cada uno de los principios que se enumeran en este trabajo se realizará en torno a un esquema de características que recogen de manera sumarial las preocupaciones esenciales que dichos principios abordan. A continuación presentamos cuáles son tales características.

Tensiones

Se analiza la repercusión de la aplicación del principio en relación con los demás principios



Argumentos

Se argumenta la razón del impacto en cada principio para aquellos casos de mayor correlación

Principios de Diseño en Componentes Web

Principios de Diseño

Introducción

La descripción detallada de cada uno de los principios que se enumeran en este trabajo se realizará en torno a un esquema de características que recogen de manera sumaria las preocupaciones esenciales que dichos principios abordan. A continuación presentamos cuáles son tales características.

Ejemplo
Desde una perspectiva visual se presentan ejemplos que aplican claramente el principio

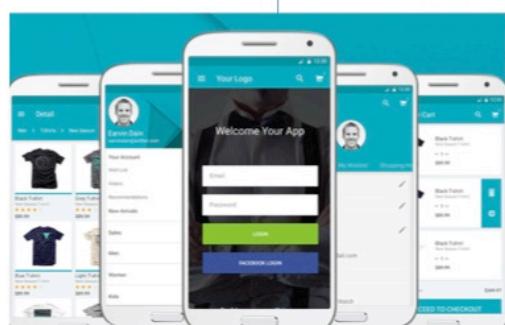
Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Diseño Centrado en el Dominio

Ejemplo
E-commerce App

La tecnología de componentes Web permite definir familias de elementos de interfaz para dar respuesta a las necesidades de negocio específicas vinculadas a un dominio



http://goo.gl/3MH5kg

35 @javiervelezreye @nmizar

Comentarios
Se ofrecen explicaciones en relación a por qué el componente de ejemplo cumple el principio

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Diseño Centrado en el Dominio



Nombre

Diseño Centrado en el Dominio



Comienza siempre realizando una adecuada división de responsabilidades en el ámbito de tu dominio y garantizando la ausencia de huecos y solapamientos entre componentes.



Actores



Analista de Dominio

Arquitecto de Componentes

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Diseño Centrado en el Dominio



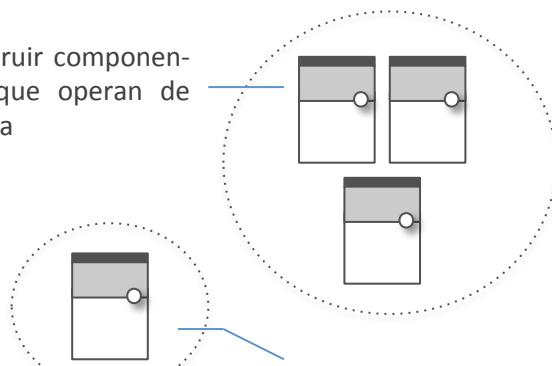
Nunca Defines Componentes Aislados

Por norma general, cuando trabajas construyes familias de componentes que incluyes en una librería de etiquetas personalizadas.

🎓 Un componente aislado puede ser puntualmente útil pero frecuentemente forma parte de una colección de otros componentes.

</> En este ejemplo se ve como los componentes trabajan frecuentemente de manera conjunta.

Es frecuente construir componentes por familias que operan de manera cooperativa



Los componentes aislados no suelen ser frecuentes en el paradigma de componentes

```
<google-map latitude="37.77493"  
longitude="-122.41942"  
fitToMarkers>  
  <google-map-marker  
    latitude="37.779"  
    longitude="-122.3892"  
    draggable="true"  
    title="Go Giants!">  
  </google-map-marker>  
</google-map>
```

El componente google-map-marker opera siempre en el ámbito light DOM de un componente google-map

Principios de Diseño en Componentes Web

Principios de Diseño

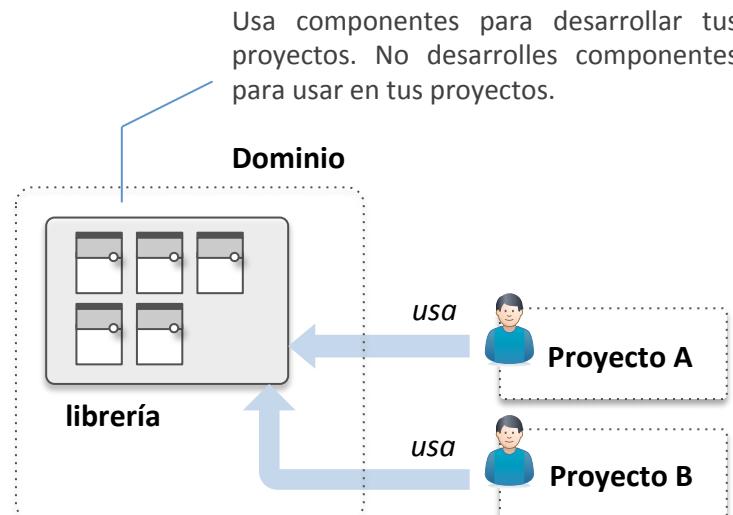
Principio de Diseño Centrado en el Dominio



Estás Contribuyendo A Dominio

Siempre que construyes componentes Web estás elaborando un dominio cuyos componentes se aplicarán en multitud de proyectos.

🎓 Esto significa que tus clientes de librería no son clientes finales sino desarrolladores de proyecto.



</> En este ejemplo se ve como Google ha adaptado su API de análisis creando una librería de componentes Web.

```
<google-analytics-dashboard>
  <google-analytics-view-selector>
  </google-analytics-view-selector>
  <google-analytics-chart>
    metrics="ga:sessions"
    dimensions="ga:date">
  </google-analytics-chart>
</google-analytics-dashboard>
```

Los desarrolladores utilizan discrecionalmente estos componentes en sus proyectos

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Diseño Centrado en el Dominio



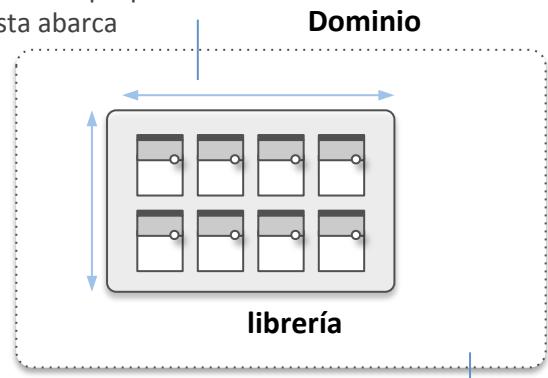
Define El Alcance

Antes de empezar la construcción de una librería de componentes determina el ámbito preciso de responsabilidades que quieras cubrir con ella.

🎓 Lo prioritario es tener claro qué capacidades quieras cubrir con tu librería y qué otras dejarás conscientemente fuera.

</> Un buen ejemplo de alcance de librería lo encontramos en la orientación a componentes que Google ha hecho de sus APIs.

El alcance de librería se puede definir en términos de la proporción de dominio que ésta abarca



Existen ámbitos dentro del dominio que estratégicamente se decide dejar no cubiertos

google-analytics-base
google-analytics-chart
google-analytics-dashboard
google-analytics-date-selector
google-analytics-query
google-analytics-view-selector

google-map
google-map-directions
google-map-marker
google-map-search

Cada una de estas dos librerías se centra en dar cobertura a las necesidades concretas dentro de un dominio específico

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Diseño Centrado en el Dominio

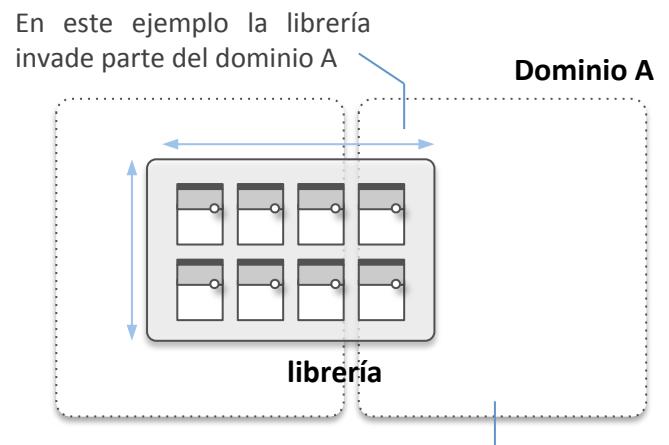


Centra El Dominio

Asegúrate de que la construcción de tu librería de componentes Web sólo aborda competencias propias del dominio al que pertenece.

🎓 Incluir componentes en tu librería fuera del alcance de su dominio provoca colisiones potenciales con otras librerías.

</> La librería core-* de Polymer es un claro contraejemplo de diseño centrado en el dominio.



Es importante asegurarte que tu librería pertenece enteramente a un único dominio

Helpers

core-a11y-keys
core-ajax
core-xhr
core-localstorage
core-media-query
core-meta
core-shared-lib
core-signals
core-style

Layout

core-animated-pages
core-drag-drop
core-drawer-panel
core-header-panel
core-overlay
core-scaffold
core-splitter

Control

core-menu
core-pages
core-scroll-header-panel
core-scroll-threshold
core-selection
core-selector

Dentro de la librería se encuentran categorías funcionales muy dispares

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Diseño Centrado en el Dominio



Fomenta La Cohesión De Librería

La aplicación de las dos facetas anteriores puede reformularse en términos de la cohesión. Se dice que existe cohesión alta en una librería si sus componentes están altamente relacionados.

- 🎓 Existen distintos tipos de cohesión que pueden ser establecidos sobre una librería de componentes. Por orden de impacto de negativo a positivo éstos son los siguientes.

Dos componentes se incluyen en — **Coincidencial**
la misma librería por criterios arbitrarios

Dos componentes se incluyen en la — **Temporal**
misma librería porque se usan siempre a la vez

Dos componentes se incluyen en — **Informacional**
la misma librería porque operan sobre el mismo tipo de datos

Dos componentes contribuyen uniformemente a un conjunto de responsabilidades relacionadas en el dominio — **Funcional**

—
Dos componentes se incluyen en la misma librería por pertenecer a una categoría lógica común

Lógica —
Dos componentes se incluyen en la misma librería porque siempre se usan de acuerdo a un determinado orden

Procedimental —
Dos componentes se incluyen en la misma librería porque siempre se conectan compositivamente en los diferentes contextos de uso

Secuencial —
Dos componentes se incluyen en la misma librería porque siempre se conectan compositivamente en los diferentes contextos de uso

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Diseño Centrado en el Dominio



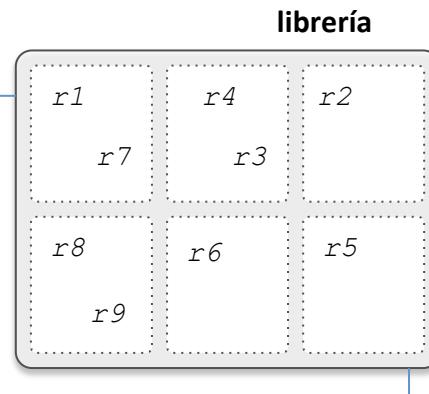
Define Espacios De Responsabilidad

Para definir el alcance de tu librería de componentes distribuye las competencias que deseas cubrir entre espacios de responsabilidad disjuntos.

🎓 La división en espacios de responsabilidad organiza el alcance ya que explicita y aclara la descomposición que se hace del dominio en el marco de la librería.

</> Nuevamente la librería de componentes de Google que encapsula las capacidades de Google Maps es un ejemplo de buena segmentación de responsabilidades.

Un espacio de responsabilidad es un clasificador de competencias dentro del alcance de la biblioteca



El alcance es una característica invariante que se establece en el tiempo de diseño de la librería. La cobertura puntual de requisitos que den los componentes no guarda relación con esta característica

google-map
google-map-directions
google-map-marker
google-map-search

Cada una de las cuatro etiquetas constituyentes se dirigen a cubrir una competencia ortogonal en el marco del sistema de mapas de Google

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Diseño Centrado en el Dominio

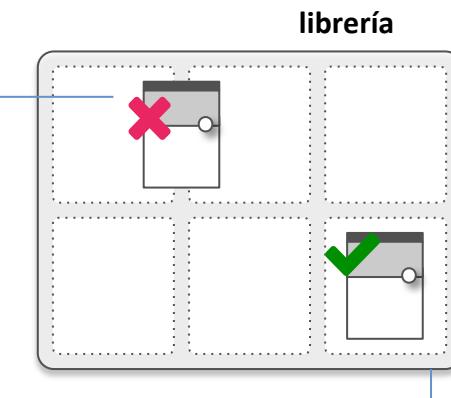


Evita Solapamientos Funcionales

Durante el proceso de análisis, diseño y construcción de componentes asegúrate de que no produces solapamientos funcionales entre los mismos.

🎓 Cada componente será ahora responsable de implementar las competencias definidas en el marco de uno y sólo uno de los espacios de responsabilidad.

Lo que en cualquier caso hay que garantizar es que ningún componente se posiciona entre dos espacios de responsabilidad



La segmentación completa y disjunta en espacios de responsabilidad ayuda a garantizar ausencias de solapamientos funcionales entre componentes

</> Como contraejemplo de segmentación de responsabilidades podemos volver a traer la librería core-*. Incluso dentro de los componentes de control las competencias entre etiquetas están solapadas.

Control

core-menu
core-pages
core-scroll-header-panel
core-scroll-threshold
core-selection
core-selector

Pese a que atienden a contextos arquitectónicos de uso diferentes, las etiquetas **core-selection** y **core-selector** presentan solapamientos funcionales en relación al control de la interacción

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Diseño Centrado en el Dominio



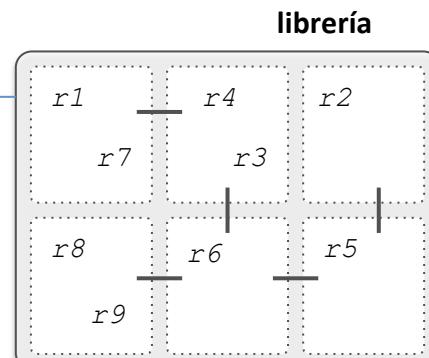
Descubre Las Relaciones De Composición Emergentes

Las dependencias en el dominio son una consecuencia del proceso de división de responsabilidades. Una vez establecidos los componentes analiza las dependencias emergentes.

🎓 Distintos criterios de división en espacios de responsabilidad conducen potencialmente a distintos conjuntos de dependencias entre los componentes.

</> En este ejemplo la relación de dependencia entre componentes es una clara consecuencia de los criterios de división de responsabilidades establecidos

Una vez identificado cada componente estudia las relaciones que guarda con los componentes de su vecindad



Es importante descubrir las dependencias de dominio y distinguirlas claramente de aquellas motivadas por una decisión de diseño o implementación. El principio de acoplamiento se ocupará posteriormente de éstas últimas

```
<google-map latitude="37.77493"
            longitude="-122.41942"
            fitToMarkers>
  <google-map-marker
    latitude="37.779"
    longitude="-122.3892"
    draggable="true"
    title="Go Giants!">
  </google-map-marker>
</google-map>
```

El componente google-map-marker se debe evaluar en el contexto DOM de un componente google-map

Principios de Diseño en Componentes Web

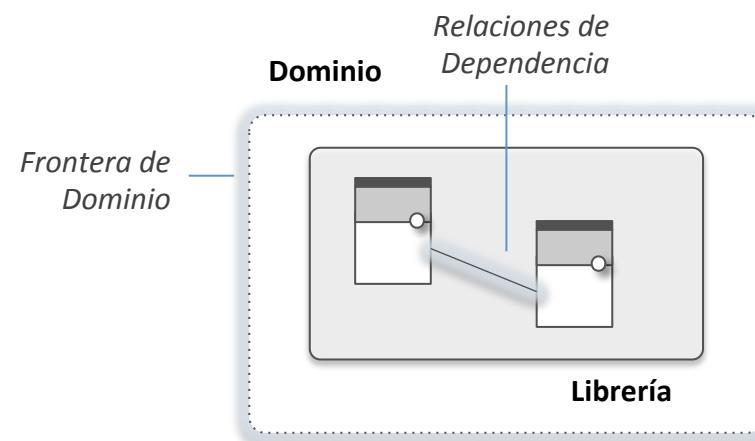
Principios de Diseño

Principio de Diseño Centrado en el Dominio



Objetivos

- Organizar el proceso de desarrollo
- Garantizar una buena cobertura
- Evitar el versionado
- Mantener acotado el alcance
- Facilitar la evolución
- Planificar el trabajo en equipo
- Promover dialectos de dominio



Motivación



El principio de diseño centrado en el dominio persigue orientar el proceso de desarrollo de una librería para asegurar su aplicabilidad máxima.

Fases



Análisis & Diseño de Dominio

Región



Fronteras de Dominio
Relaciones de Dependencia

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Diseño Centrado en el Dominio



Tensiones

Un análisis centrado en el alcance del dominio favorece encontrar las abstracciones adecuadas

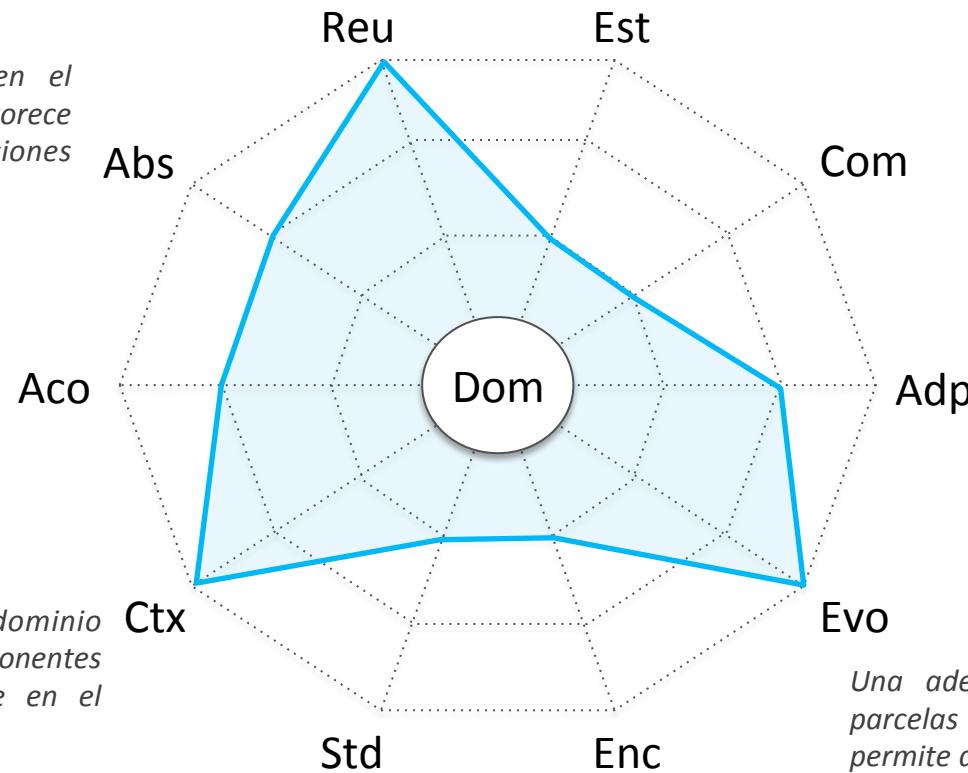
Un análisis del dominio descuidado puede generar acoplamientos no previstos

La organización del dominio promueve que los componentes se integren fluidamente en el contexto de uso

Una adecuada división de responsabilidades permite obtener componentes con mayores oportunidades de reutilización

Un cuidadoso análisis de dominio evita escenarios de adaptación pero facilita su aplicabilidad discrecional

Una adecuada división del dominio en parcelas bien definidas de responsabilidad permite definir las líneas evolutivas de cada componente



Principios de Diseño en Componentes Web

Principios de Diseño

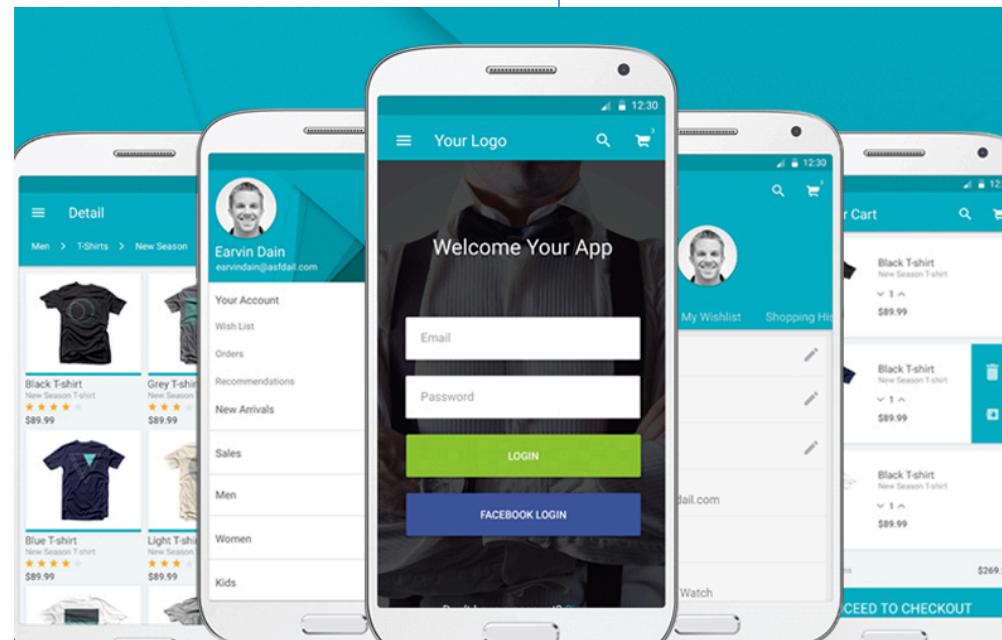
Principio de Diseño Centrado en el Dominio



Ejemplo

E-commerce App

La tecnología de componentes Web permite definir familias de elementos de interfaz para dar respuesta a las necesidades de negocio específicas vinculadas a un dominio



<http://goo.gl/3MHSkg>

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Acoplamiento Débil



Nombre

Acoplamiento Débil



Cuando construyas un componente asegúrate de minimizar el número de dependencias existentes y de mantener un diseño agnóstico de las características de las mismas.



Actores



Arquitecto & Desarrollador de Componentes

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Acoplamiento Débil

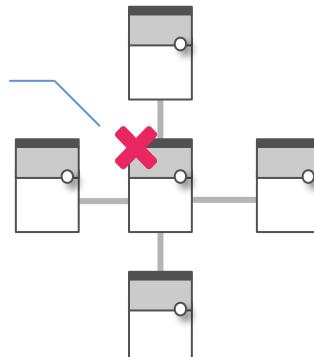


Minimiza Las Dependencias

Durante las fases de diseño y desarrollo conviene analizar la prescindibilidad de las dependencias entre componentes. En la medida de lo posible reduce al máximo su número.

🎓 Las dependencias que un componente establece con su vecindad limitan su autonomía y los escenarios en los que éste puede operar.

Cada dependencia supone un riesgo de propagación de errores entre componentes y una traba para la evolución



Frecuentemente el exceso de dependencias está motivado por una mala estrategia de diseño que puede ser solventada con el uso de patrones que fomenten el desacoplamiento

</> Un componente sin dependencias o sólo con dependencias opcionales es ideal. Uno de tantos ejemplos es el google-signin.

```
<google-signin  
  clientid="..."  
  scopes="profile"  
  role="button"  
  tabindex="0">  
</google-signin>
```

Google-signin es un componente para poder autenticarse en un tercero a través de las credenciales de Google. Su carácter autónomo sirve de ejemplo en esta faceta

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Acoplamiento Débil

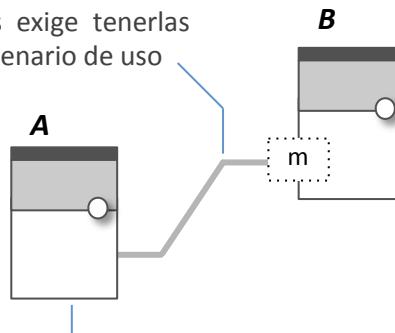


Haz Explícitas Las Dependencias Obligatorias

Procura que dentro de tu arquitectura se mantenga un conocimiento explícito de todas las dependencias obligatorias que cada componente tiene con la vecindad.

🎓 Las dependencias obligatorias entre componentes son aquellas necesarias para garantizar su correcto funcionamiento.

La explicitación arquitectónica de las dependencias obligatorias exige tenerlas en cuenta en cualquier escenario de uso



La explicitación arquitectónica se hace patente al observar el código interno del componente. La mejor forma de conseguirlo es mediante el uso del modelo de paso de mensajes (llamadas a métodos). A necesitará una referencia explícita a B dentro de su código

</> Por ejemplo considere un componente asistente que requiere obligatoriamente hacer uso de una ventana modal donde mostrar la ayuda.

```
<core-overlay id="A">
  <h2>Help</h2>
  ...
  <button toggle-btn>OK</button>
</core-overlay>

<wc-help-assistant window="A">
</wc-help-assistant>
```

El core-overlay expone métodos para abrir y cerrar explícitamente la ventana modal

El asistente de ayuda usa la ventana modal y la abre o cierra explícitamente por invocación de sus métodos. La referencia explícita a A en el atributo window ayuda a destacar la existencia de una dependencia obligatoria

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Acoplamiento Débil

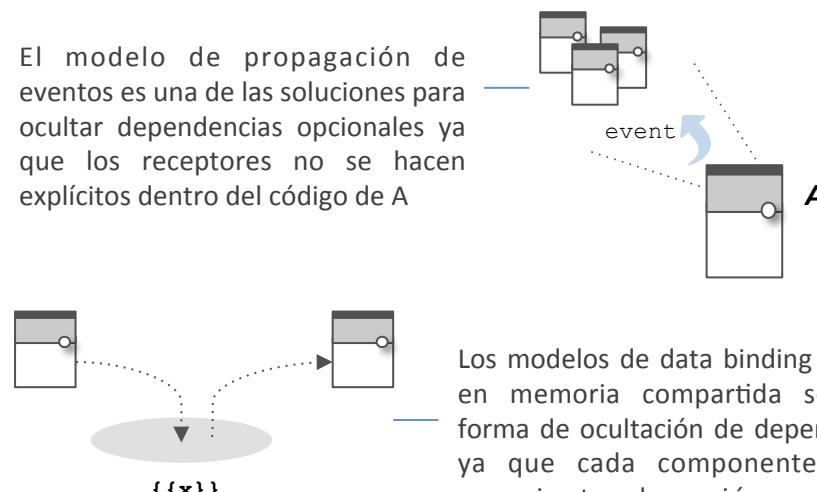


Haz Implícitas Las Dependencias Opcionales

Procura que dentro de tu arquitectura se mantengan ocultas todas las dependencias opcionales que cada componente tiene con la vecindad.

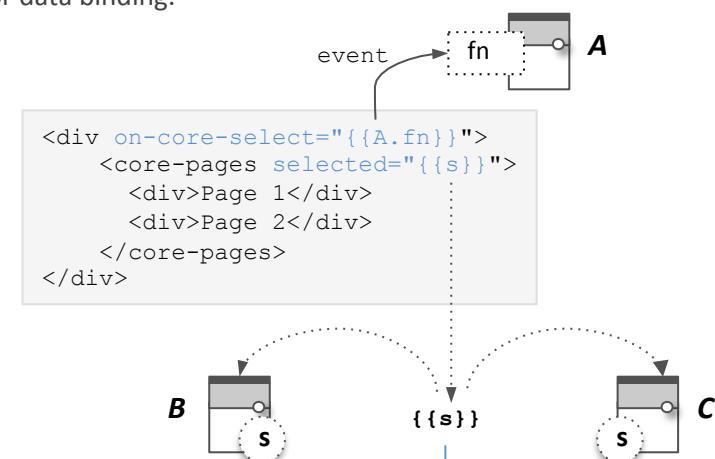
🎓 Las dependenciasopcionales sirven para comunicar información a los componentes de la vecindad que no son necesariamente requeridos para el correcto funcionamiento del componente.

El modelo de propagación de eventos es una de las soluciones para ocultar dependenciasopcionales ya que los receptores no se hacen explícitos dentro del código de A



Los modelos de data binding basados en memoria compartida son otra forma de ocultación de dependencias ya que cada componente no es consciente de quién recibe la información compartida

</> La comunicación desacoplada puede realizarse en Polymer mediante arquitecturas de eventos o memoria compartida por data binding.



Cada vez que en core-pages se selecciona una página se dispara un evento core-select que escucha A. Asimismo los cambios son propagados por data binding en la variable s lo que permite informar a B y C

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Acoplamiento Débil

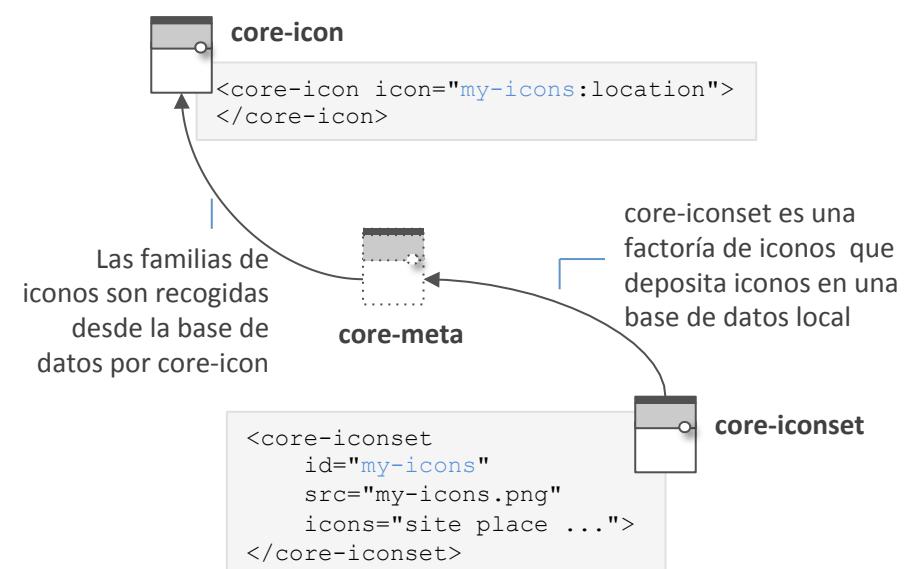
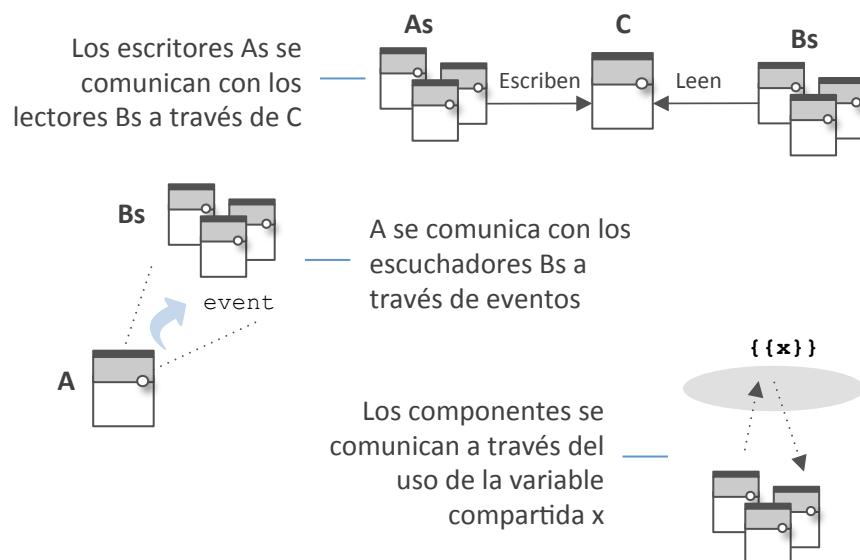


Busca La Indirección

En términos generales procura que en tus diseños de componentes la comunicación entre los mismos se realice de manera no directa.

🎓 Los esquema de indirección siempre fomentan el desacoplamiento ya que permiten segmentar la comunicación.

</> Un buen ejemplo de comunicación indirecta por paso de mensajes se da entre el componente core-icon y el core-iconset.



Principios de Diseño en Componentes Web

Principios de Diseño

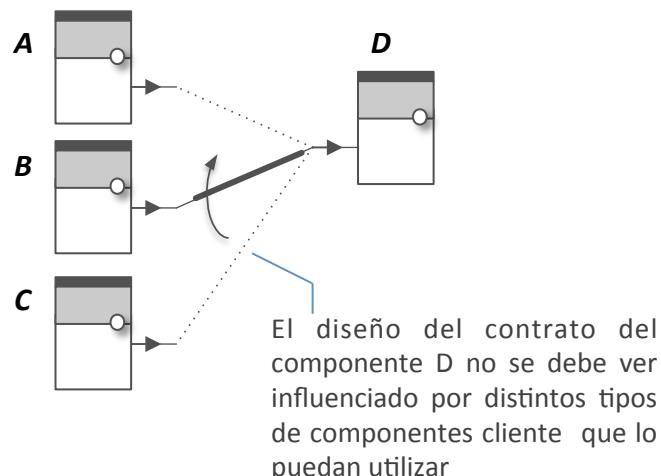
Principio de Acoplamiento Débil



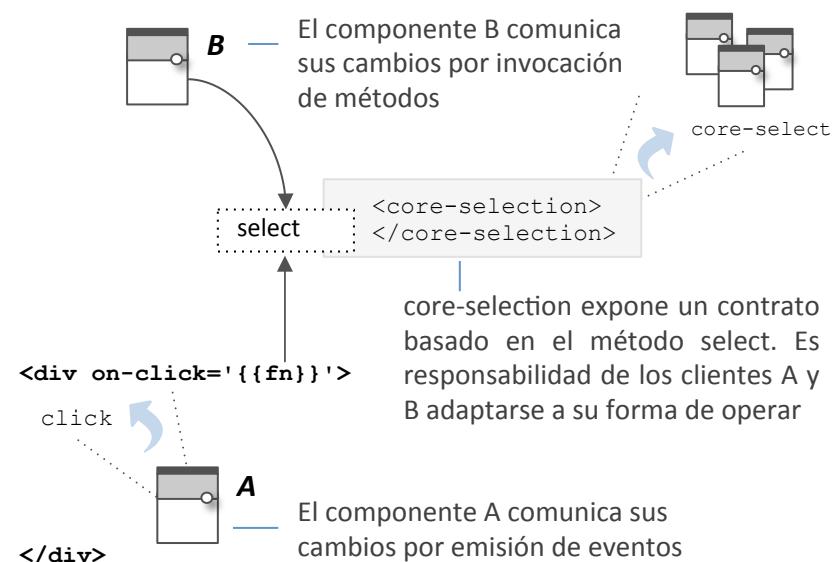
No Importa Quién

En el marco de una comunicación, el diseño de un componente no debería verse influenciado por quién o quienes son los demás componentes implicados en la misma.

- Al diseñar un componente no debería importar el tipo, comportamiento, propósito o identidad del resto de componentes implicados en una comunicación.



- </> El componente core-selection es un buen ejemplo de diseño que mantiene el desacoplamiento de cliente.



Principios de Diseño en Componentes Web

Principios de Diseño

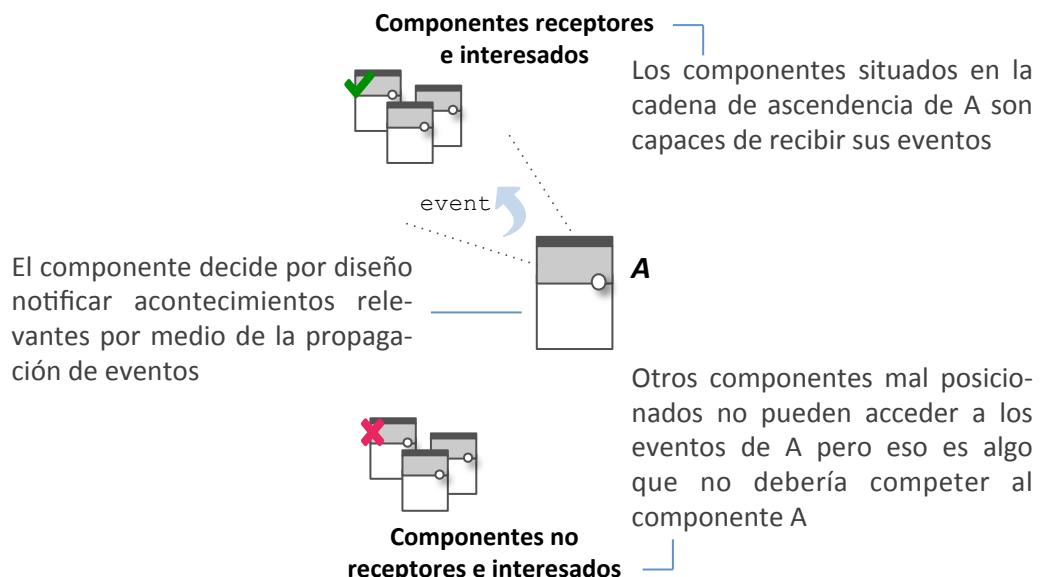
Principio de Acoplamiento Débil



No Importa Dónde

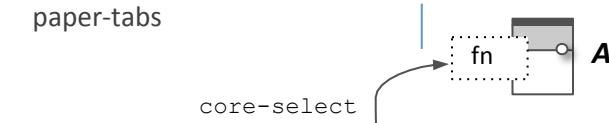
En el marco de una comunicación, el diseño de un componente no debería verse influenciado por dónde se encuentran los demás componentes implicados en la misma.

- 🎓 Al diseñar un componente no debería importar la posición relativa en el DOM donde residen el resto de componentes implicados en una comunicación.



- </> En este caso el componente paper-tabs nos sirve de contraejemplo para ilustrar el desacoplamiento a contexto.

Los componentes escuchadores deben registrarse necesariamente en algún nivel dentro de la cadena de ascendencia de paper-tabs



```
<div on-core-select="{}{A.fn}">
  <paper-tabs>
    <paper-tab>1</paper-tab>
    <paper-tab>2</paper-tab>
  </paper-tabs>
  <div>...</div>
</div>
```

Otros componentes mal posicionados no podrán establecer una comunicación con paper-tabs

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Acoplamiento Débil

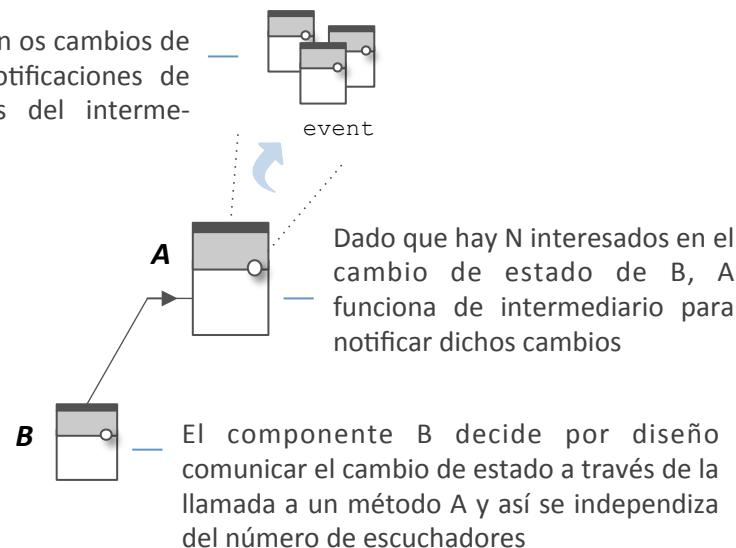


No Importa Cuántos

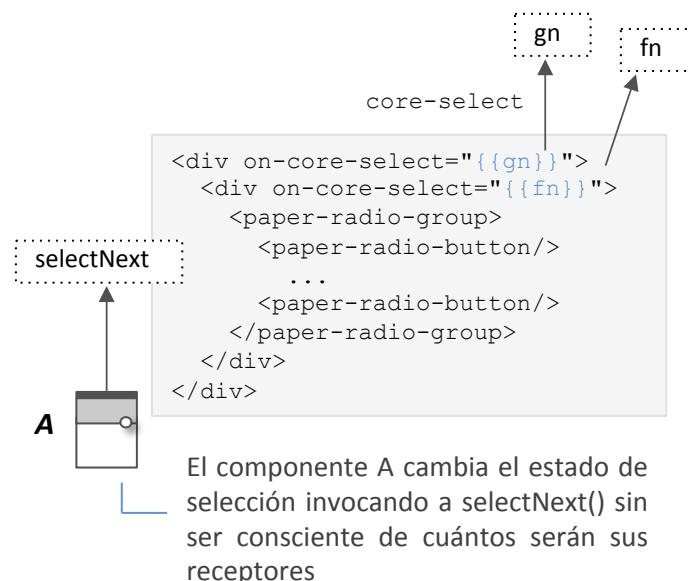
En el marco de una comunicación, el diseño de un componente no debería verse influenciado por cuántos son los demás componentes implicados en la misma.

Al diseñar un componente no debería importar la cantidad de componentes que participan potencialmente en una comunicación.

Los interesados en los cambios de B reciben las notificaciones de cambio a través del intermediario A



</> El componente paper-radio-group es un ejemplo de cómo una notificación de selección se puede desacoplar de la cantidad de escuchadores.



Principios de Diseño en Componentes Web

Principios de Diseño

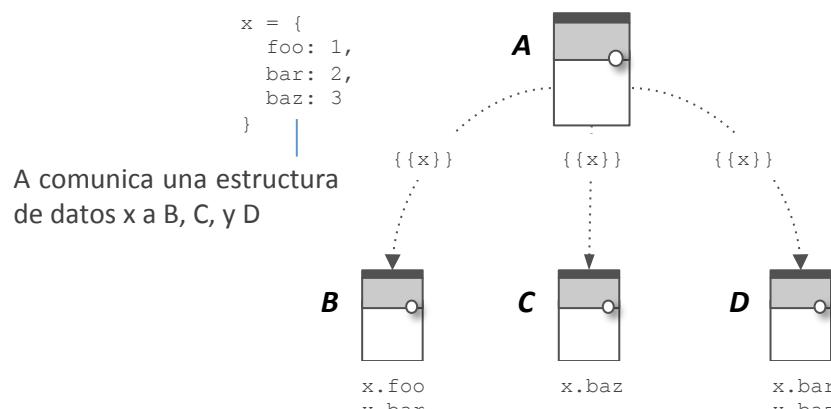
Principio de Acoplamiento Débil



No Importa Cuánto

En el marco de una comunicación, el diseño de un componente no debería verse influenciado por la cantidad de información adicional que se comparte dentro de la misma.

🎓 Al diseñar un componente no deberían importar los cambios en la cantidad de información que se comparta a lo largo del tiempo.



Cada componente B, C, D hace sólo uso de parte de los datos en x sin importar el resto

</> El siguiente ejemplo ilustra bien la transferencia de un volumen de datos potencialmente variante.

A lo largo del tiempo el componente google-map podría ir incluyendo nueva información no dirigida a google-map-directions dentro del objeto map

```
<google-map map="{{map}}"  
latitude="37.779"  
longitude="-122.3892">  
</google-map>
```

```
<google-map-directions map="{{map}}"  
startAddress="San Francisco"  
endAddress="Mountain View">  
</google-map-directions>
```

Los cambios aditivos sobre el objeto map no deberían impactar en el funcionamiento de google-map-directions

Principios de Diseño en Componentes Web

Principios de Diseño

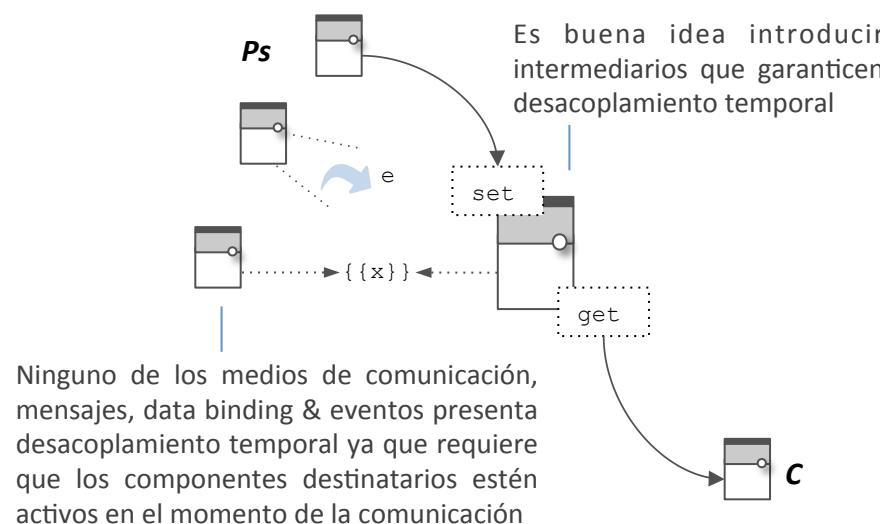
Principio de Acoplamiento Débil



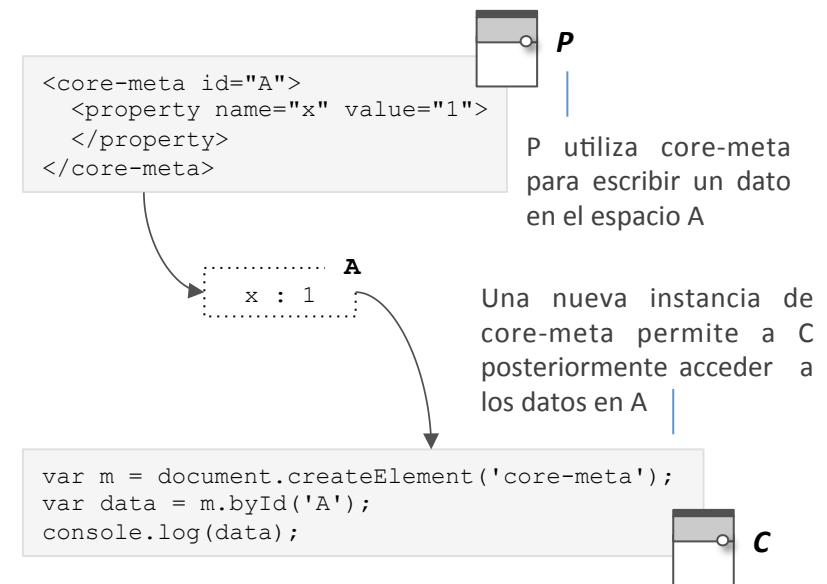
No Importa Cuándo

En el marco de una comunicación, el diseño de un componente no debería verse influenciado por cuándo se produce la información compartida dentro de la misma.

- 🎓 Al diseñar un componente no debería importar con qué desfase llega a los destinatarios la información compartida en una comunicación.



- </> Muchos componentes de Polymen en las librerías core-* y paper-* utilizan la etiqueta core-meta para ofrecer desacoplamiento temporal.



Principios de Diseño en Componentes Web

Principios de Diseño

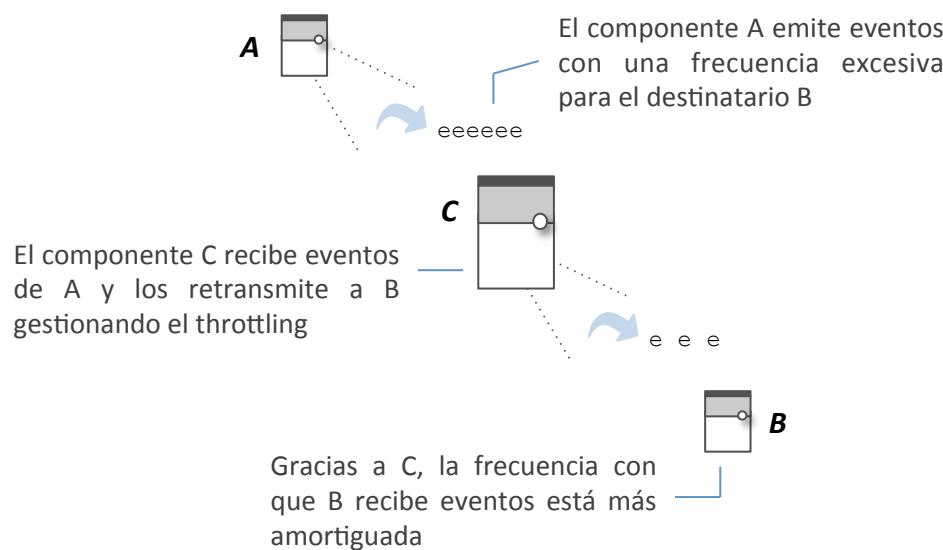
Principio de Acoplamiento Débil



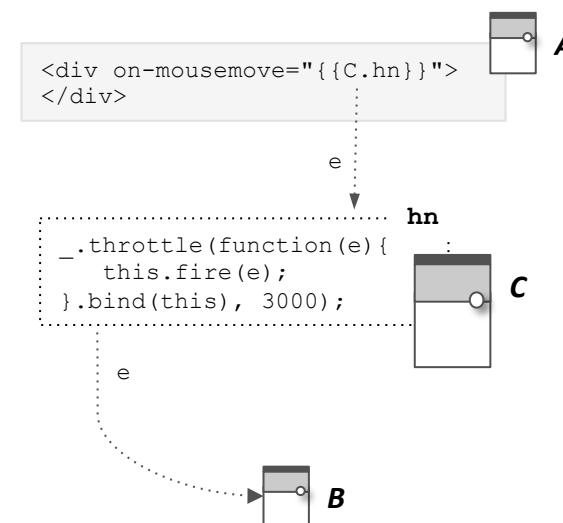
No Importa Cada Cuánto

En el marco de una comunicación, el diseño de un componente no debería verse influenciado por la frecuencia con la que otros componentes interactúan dentro de la misma.

- 🎓 Al diseñar un componente no debería importar con qué frecuencia llegan solicitudes al mismo desde los demás participantes en una comunicación.



- </> Existen muchas librerías como underscore.js que permiten implementar componentes C que articulen políticas de throttling.



Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Acoplamiento Débil



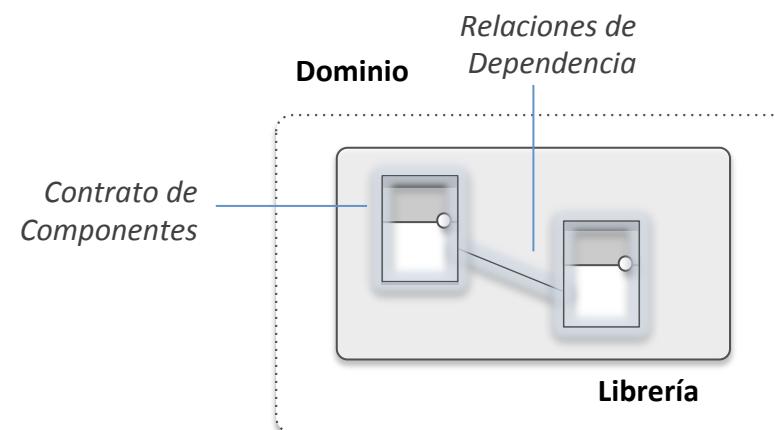
Objetivos

- Aumentar la mantenibilidad
- Simplificar el uso
- Simplificar el desarrollo
- Facilitar la sustitución
- Fomentar la autonomía
- Mejorar la evolutividad

Motivación



El principio de acoplamiento débil persigue obtener componentes idealmente autónomos que sean fácilmente reutilizables en distintos contextos de uso.



Fases



- Diseño de Dominio
- Desarrollo de Componentes

Región



- Contrato de Componentes
- Relaciones de Dependencia

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Acoplamiento Débil

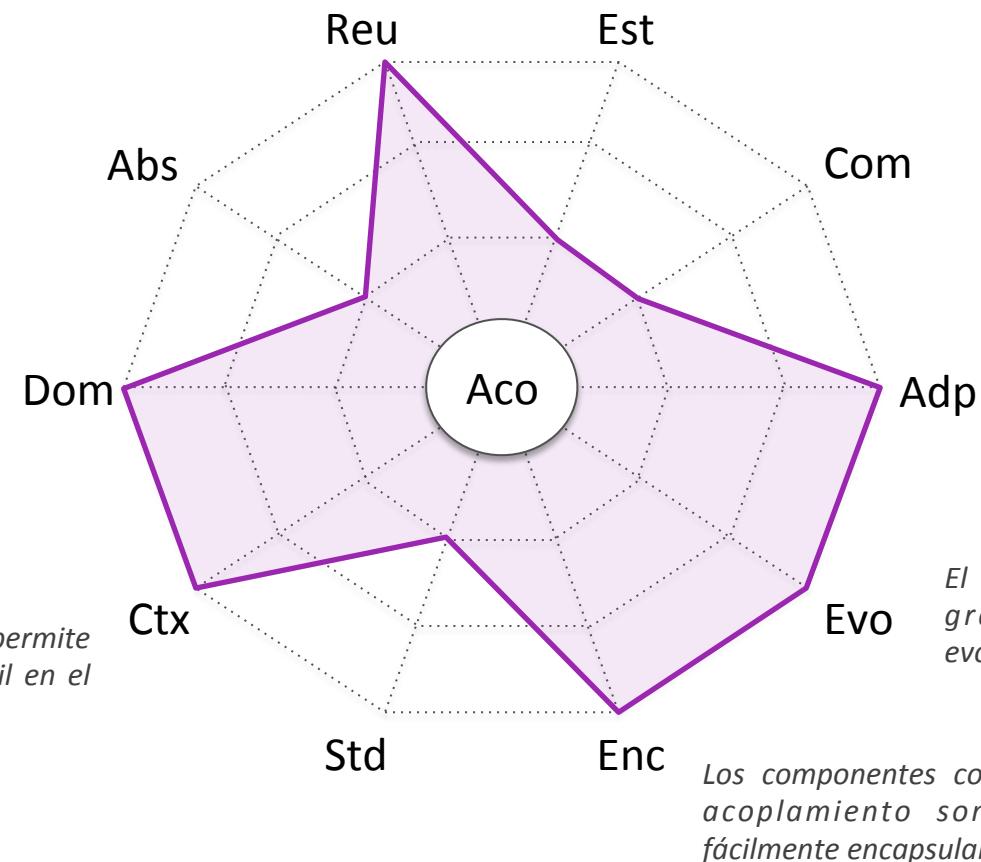


Tensiones

Cuantas menos dependencias presenta un componente mayor libertad de reutilización ofrece

Articular mecanismos de desacoplamiento conduce a generar confusión en el dominio

El desacoplamiento permite una integración grácil en el contexto



El bajo acoplamiento permite articular técnicas de adaptación en los componentes

El desacoplamiento ofrece más grados de libertad en la evolución del componente

Los componentes con bajo acoplamiento son más fácilmente encapsulables

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Acoplamiento Débil

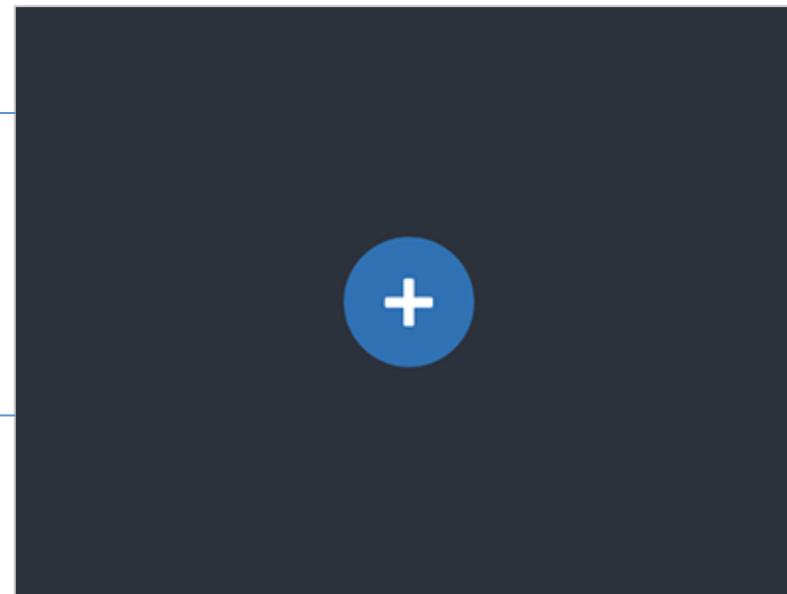


Ejemplo

Collection

Este componente no presenta dependencias con ningún otro elemento

Cuando pinchas en un ítem de la colección se le comunica al padre por eventos



Cada avatar es un componente que es responsable de su propio rendering con lo que el volumen de información intercambiada es nulo

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Abstracción



Nombre

Abstracción



Asegúrate siempre de que tus componentes operan al nivel más elevado de abstracción posible dentro de los contextos de colaboración en que participen.



Actores



Analista de Dominio

Arquitecto & Desarrollador de Componentes

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Abstracción



Abstacta Los Detalles

Durante la definición del contrato de un componente asegúrate de que no expones detalles internos de implementación y de que revelas la mínima cantidad de información posible sobre él.

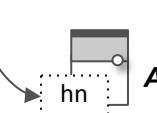
- 🎓 Incluye la parte pública en el prototipo. Usa los mecanismos de declaración de atributos de Polymer para definir el estado. Encapsula mediante clausuras las partes privadas.

	Privada	Pública
Estado	Variables implementación	Atributos Propiedades
Funciones	Helpers Adaptadores ...	Capacidades Controladores Manejadores

</> core-ajax es un buen ejemplo de componente que permite realizar una prolífica operación abstractando de los detalles de implementación.

```
<core-ajax
  auto
  url="..."
  params="..."
  handleAs="json"{{A.hn}}>
</core-ajax>
```

De una manera declarativa se exponen los datos de una solicitud AJAX incluyendo la función que gestionará la respuesta cuando ésta se haya obtenido



Principios de Diseño en Componentes Web

Principios de Diseño

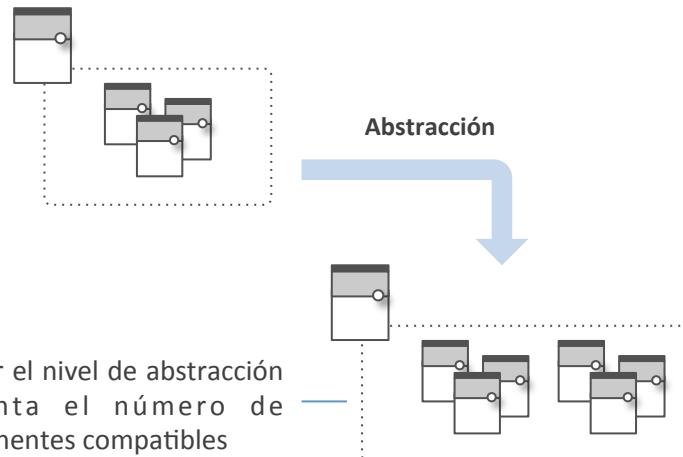
Principio de Abstracción



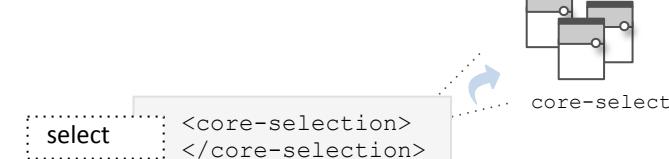
Opera Al Mayor Nivel De Abstracción Posible

Expresa las colaboraciones entre componentes al mayor nivel de abstracción posible. De esta manera se dará cobertura participativa a un mayor número de variantes.

- 🎓 Al definir un componente en un nivel de abstracción dado se condiciona su número de potenciales colaboradores. Si el nivel aumenta dicho número también.



- </> Como ejemplo seleccionaremos de nuevo el componente core-selection y lo compararemos con otro hipotético definido a un nivel más abstracto.



Al subir el nivel de abstracción en el que se define el contrato del componente, éste no queda restringido a escenarios de selección sino a cualquier situación de cambio de estado

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Abstracción

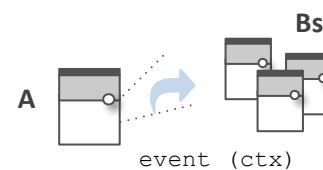
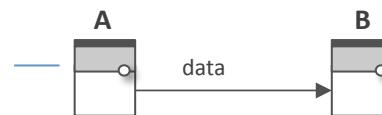


Construye Abstracciones

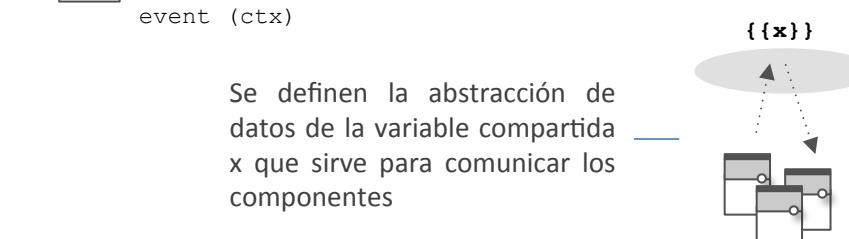
Durante la fases de diseño y análisis establece las abstracciones de datos pertinentes que serán elementos de intercambio en las comunicaciones entre componentes.

🎓 Además de definir el entramado de componentes de dominio es importante establecer las abstracciones de datos que permiten a éstos comunicarse.

Se define la abstracción de datos data que se utiliza para comunicar A con B en los parámetros de una llamada



Se definen la abstracción de datos del contexto de evento que permite a A comunicarse con los escuchadores Bs



Se definen la abstracción de datos de la variable compartida x que sirve para comunicar los componentes

</> La familia de componentes google-map-* es un ejemplo de librería que utiliza un objeto map para comunicar unos componentes con otros.

```
<google-map-directions map="{{map}}"  
startAddress="San Francisco"  
endAddress="Mountain View">  
</google-map-directions>
```

```
<google-map map="{{map}}>  
</google-map>
```

```
<google-map-search map="{{map}}"  
query="Pizza"  
result="{{result}}>  
</google-map-search>
```

El objeto {{map}} es una abstracción de datos que representa un mapa de Google Maps

Principios de Diseño en Componentes Web

Principios de Diseño

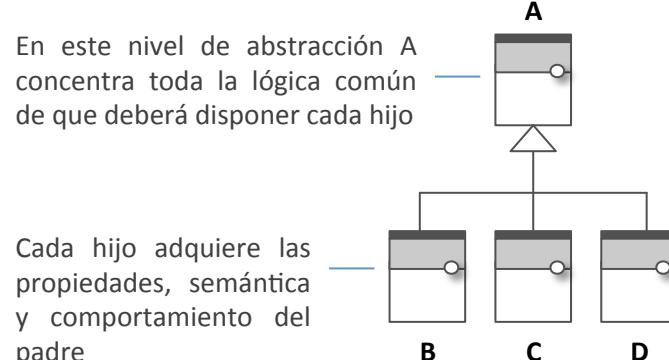
Principio de Abstracción



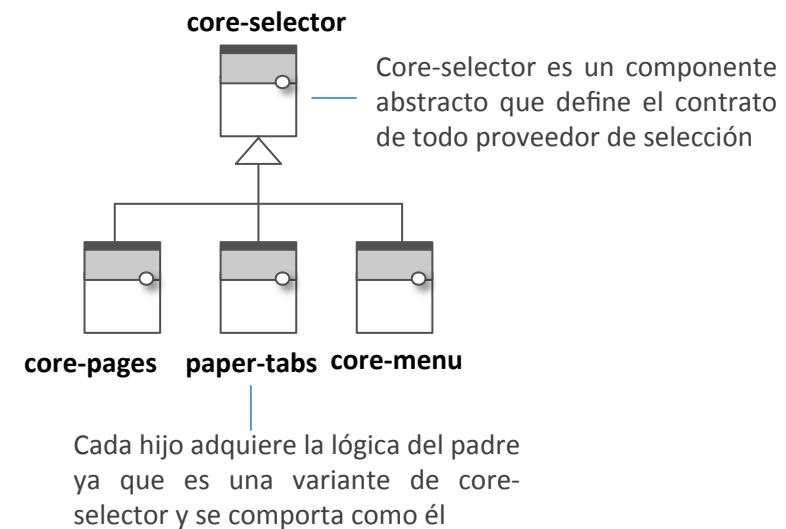
Abstacta Por Prototipos

Define componentes abstractos que identifiquen un modelo de capacidades y comportamiento prototípico para que otros componentes puedan adquirirlo por herencia.

🎓 La definición de un componente abstracto que es extendido por herencia permite centralizar cierta lógica común que debe ser conferida a cada hijo.



</> Un buen ejemplo de abstracción por herencia que persigue este objetivo lo encontramos en componentes que heredan de core-selector.



Principios de Diseño en Componentes Web

Principios de Diseño

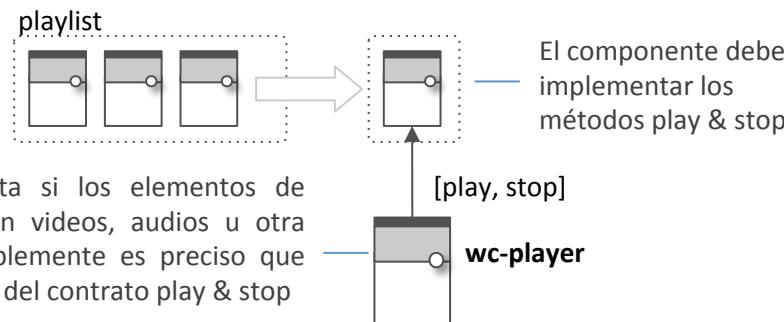
Principio de Abstracción



Abstacta Por Roles

Cuando diseñas un componente define los requisitos sobre los contratos de los componentes de la vecindad que éstos deben satisfacer para que puedan operar con él.

- Para operar con un componente generalmente basta con exigir que los colaboradores implementen cierto contrato parcial.



```
<wc-player>
  <video> ... </video>
  <audio> ... </audio>
  <audio> ... </audio>
  <video> ... </video>
</wc-player>
```

- Un ejemplo de abstracción por roles lo encontramos en el componente core-pages.

```
<core-pages selected="{{s}}>
  <div>Page 1</div>
  <div>Page 2</div>
  <div>Page 3</div>
</core-pages>
```

Las etiquetas anidadas representan las páginas gestionadas. Pueden ser cualquier elemento. Sólo deben responder a eventos tap

Principios de Diseño en Componentes Web

Principios de Diseño

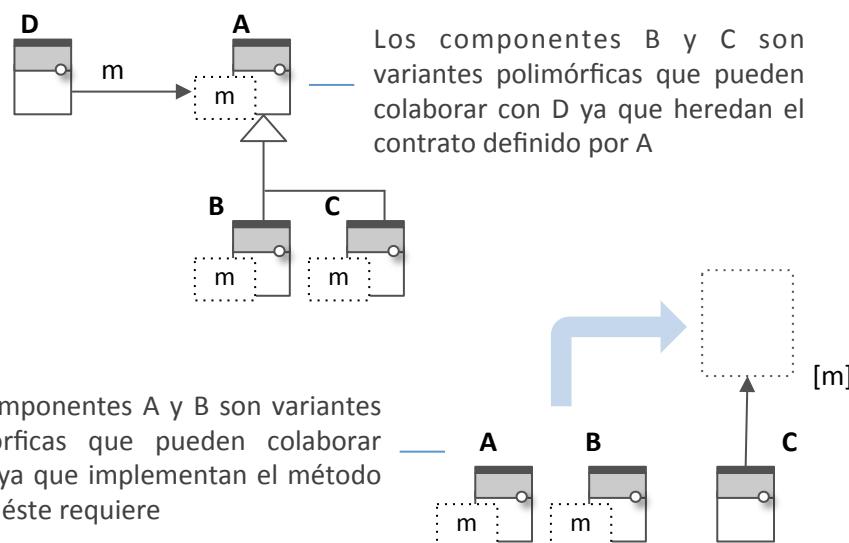
Principio de Abstracción



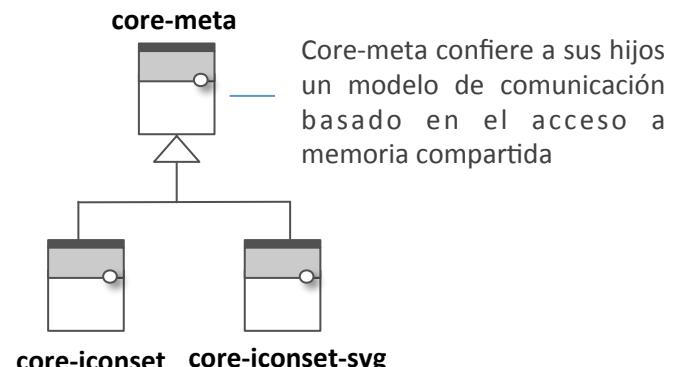
Define Variantes Polimórficas

Cada vez que definas una abstracción considera qué variantes polimórficas específicas existen dentro del dominio para cumplir con ella.

🎓 La abstracción permite definir puntos dentro de la arquitectura donde diversos componentes pueden ser intercambiados para jugar un determinado rol.



</> Un contrajejemplo de abstracción por variantes lo encontramos en las etiquetas core de gestión de iconos.



Aquí se utiliza la herencia exclusivamente como método de factorización de código y no como soporte a la definición de variantes intercambiables. No se puede decir que ninguno de los componentes hijo mantengan una relación semántica es-un con el padre

Principios de Diseño en Componentes Web

Principios de Diseño

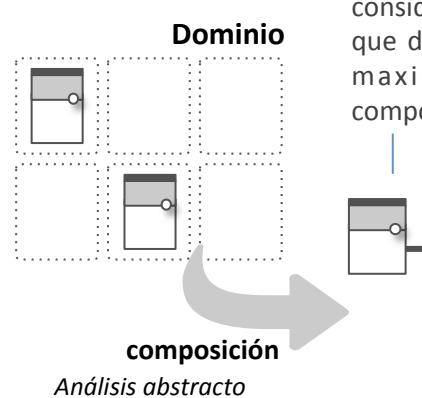
Principio de Abstracción



Abstacta Para Componer

En el ejercicio de división de responsabilidades orienta los procesos de abstracción para maximizar las posibilidades compositivas con otros componentes del entorno.

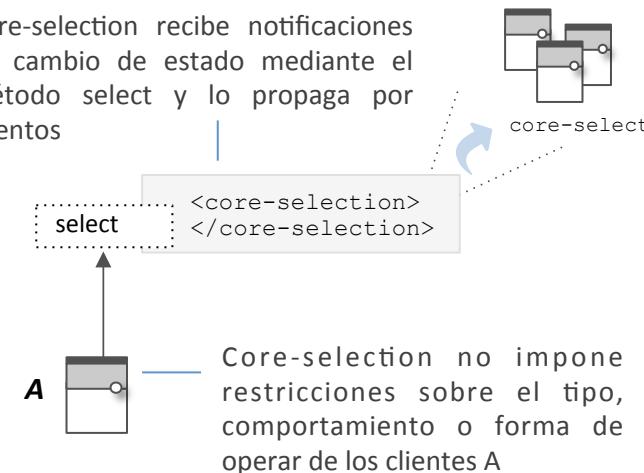
- 🎓 Diseñar al nivel adecuado de abstracción permite hacer una división de responsabilidades sobre el dominio que fomenta las oportunidades de composición.



Cuando definas un componente considera el nivel de abstracción al que debes definir su contrato para maximizar sus posibilidades compositivas

- </> El componente core-selection está definido a un nivel de abstracción elevado para dar soporte a muchos escenarios de uso.

Core-selection recibe notificaciones de cambio de estado mediante el método select y lo propaga por eventos



Principios de Diseño en Componentes Web

Principios de Diseño

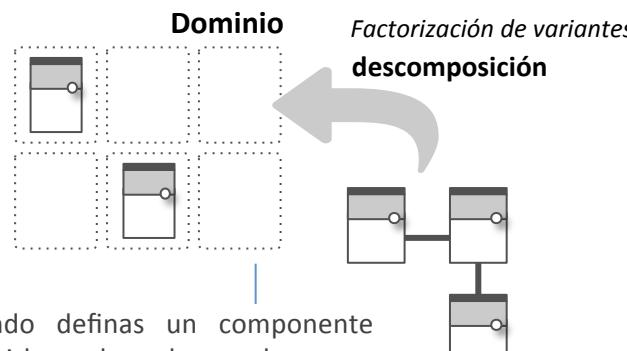
Principio de Abstracción



Abstacta Para Descomponer

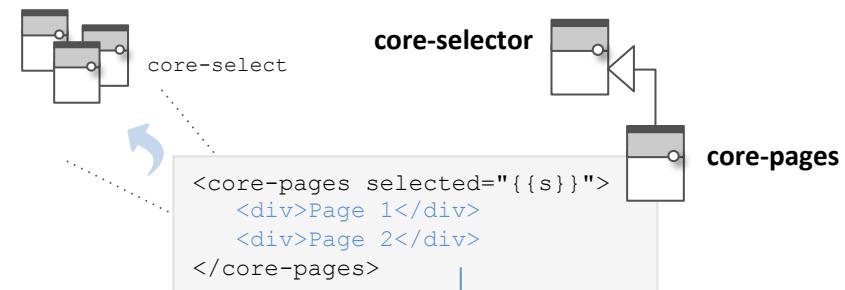
En el ejercicio de división de responsabilidades orienta los procesos de abstracción teniendo en cuenta las necesidades prototípicas que demanden otros componentes en el dominio.

🎓 Pese a que es buena práctica perseguir el agnosticismo entre componentes resulta pragmático atender en la medida de lo posible las necesidades de la vecindad.



Cuando definas un componente considera las demandas que imponen los escenarios prototípicos de uso para establecer el nivel de abstracción del mismo

</> Core-selector es un componente similar a core-selection preparado para aquellos escenarios en los que los proveedores de selección se anidan dentro del componente.



Core-pages hereda de core-selector. Este componente funciona como core-selection pero está preparado para aquellos casos donde los componentes que cambian el estado son los elementos agregados en el light DOM

Principios de Diseño en Componentes Web

Principios de Diseño

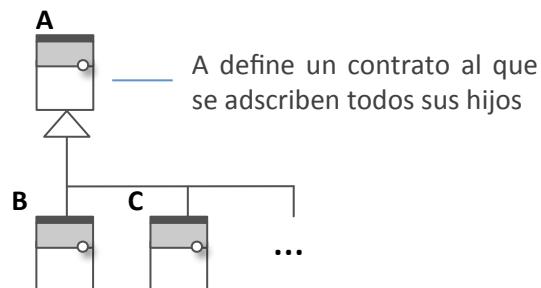
Principio de Abstracción



AbstRAE Para Extender

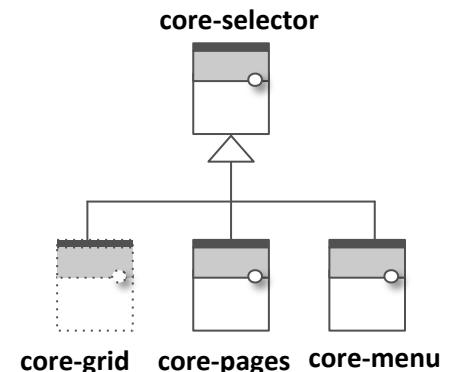
Durante las fases de diseño establece distintos niveles de abstracción para fomentar la libertad evolutiva de los componentes en el marco de tu dominio.

- 🎓 La definición de abstracciones permite articular puntos de extensión variante donde nuevos componentes puedan incluirse como sustitutos.



Cada herencia polimórfica supone un punto de extensión potencial para añadir futuras variantes que se adscriban al contrato definido por A

- </> En la librería core-* encontramos varios componentes que heredan de core-selector. Nuevos componentes similares deberían también heredar de él.



Si algún día se crea un componente para hacer grid de contenidos con un comportamiento similar a core-pages o core-menu tendría sentido hacerlo heredando de core-selector

Principios de Diseño en Componentes Web

Principios de Diseño

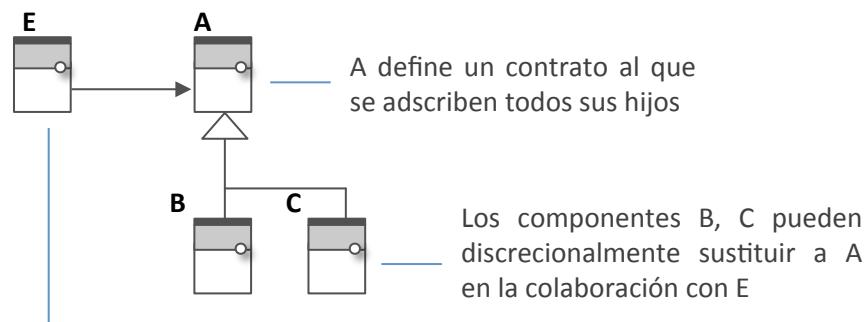
Principio de Abstracción



Abstera Para Desacoplar

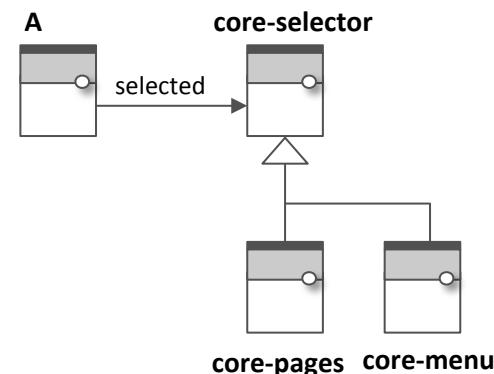
Diseña tus componentes al nivel de abstracción adecuado para permitir que éstos puedan operar entre sí con suficiente desacoplamiento.

🎓 Cuando definimos puntos de extensión por abstracción estamos garantizando un desacoplamiento entre los participantes.



Dado que E no es consciente con qué variante de A está operando en cada momento, se ha establecido un desacoplamiento entre E y sus colaboradores

</> Un ejemplo de abstracción que fomenta el desacoplamiento lo encontramos en la jerarquía de herencia de core-selector dentro de la librería core-*.



Dado que ambos componentes implementan el contrato definido en core-selector, A puede asumir siempre la existencia de un atributo selected

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Abstracción



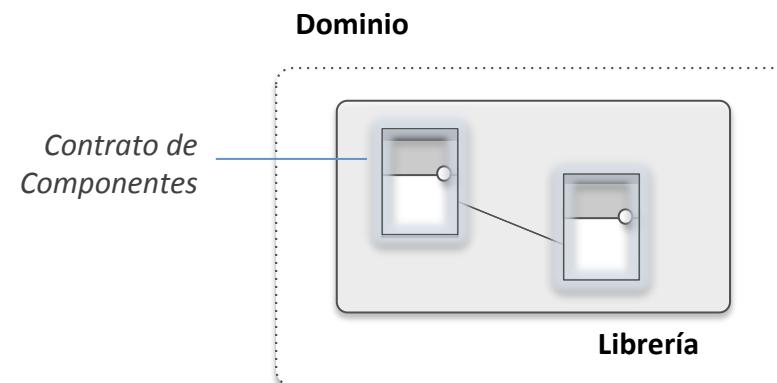
Objetivos

- Facilitar el desarrollo evolutivo
- Aumentar la versatilidad compositiva
- Favorecer la organización del dominio
- Promover la economía de Código
- Fomentar la factorización de Código
- Simplificar el uso de componentes
- Fomentar la declaratividad

Motivación



Al definir los componentes al nivel de abstracción adecuado se consigue que éstos puedan operar en el mayor número posible de contextos de uso.



Fases



Análisis & Diseño de Dominio
Diseño de Componentes

Región



Contrato de Componentes

Principios de Diseño en Componentes Web

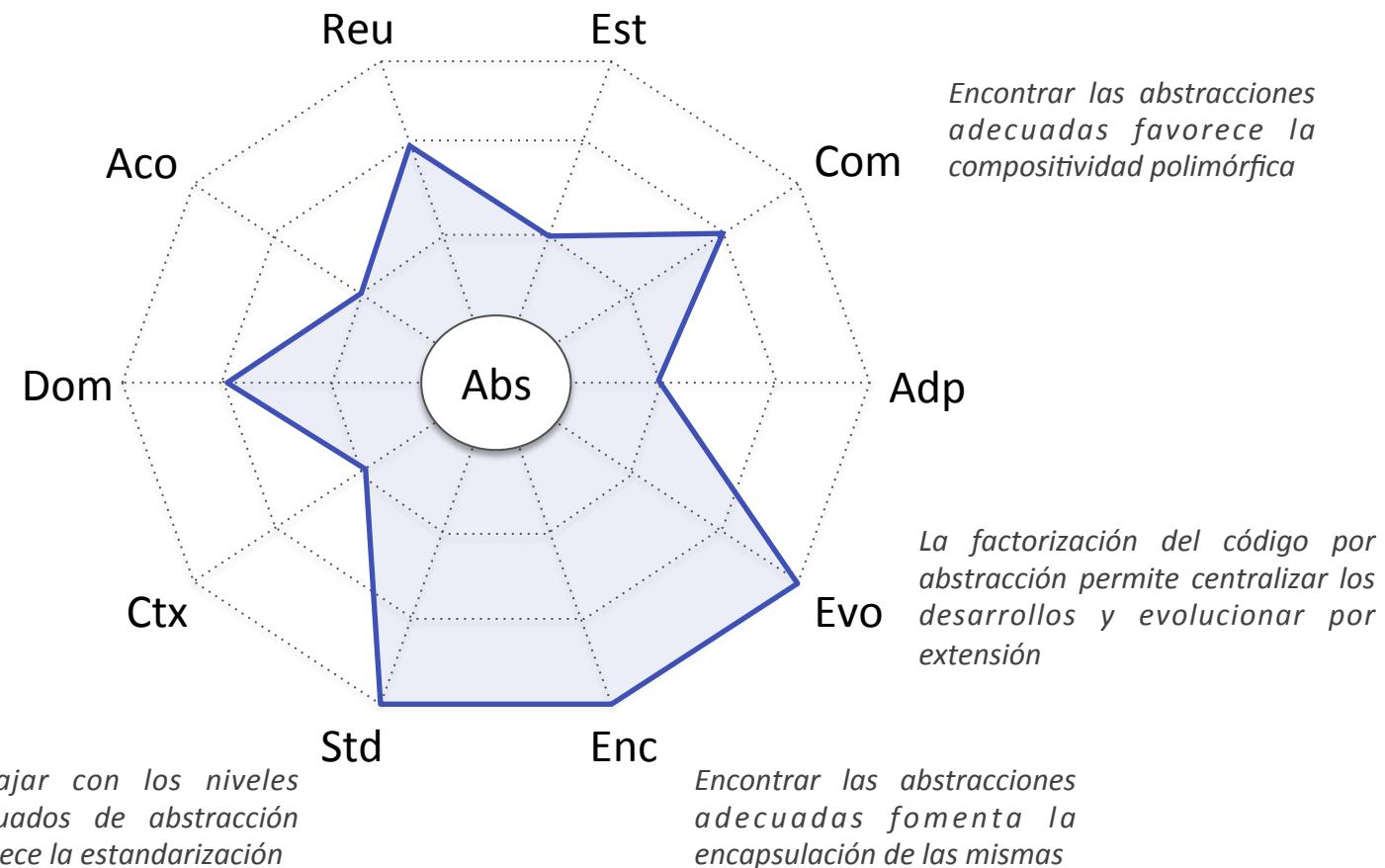
Principios de Diseño

Principio de Abstracción



Tensiones

Los componentes más abstractos presentan mayor grado de reutilización potencial



Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Abstracción

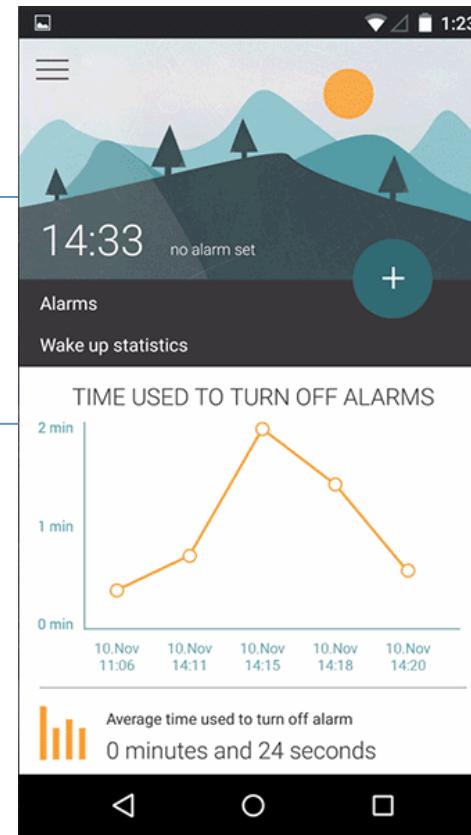


Ejemplo

Add Alarm animation

Este componente captura una experiencia de usuario que se abstracta de los **detalles** de contenido

El componente de animación es un presentador de información independientemente de que en este **contexto** se use para configurar una alarma



Todos los componentes de esta interfaz operan en un **nivel** de abstracción común

Esta **composición** particular de contenidos se ha encontrado a partir de una **descomposición** inicial del dominio

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Reutilización



Nombre

Reutilización



Asegúrate de maximizar siempre las posibilidades de reutilización de tus componentes dentro del dominio al que pertenecen.



Actores



Arquitecto & Desarrollador de Componentes

Principios de Diseño en Componentes Web

Principios de Diseño

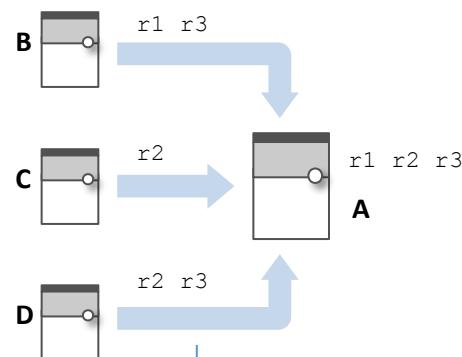
Principio de Reutilización



Entiende A Tus Clientes

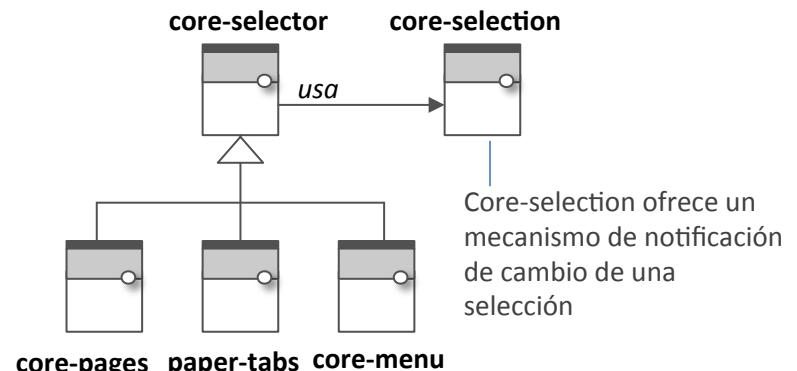
Antes de empezar a desarrollar un componente estudia todos los requisitos que te exigen los componentes de la vecindad en los diferentes contextos de uso potencial.

- 🎓 Habrá muchos componentes que aún no existan cuando diseñes. Tu componente será más reutilizable cuanta mayor cobertura a posibles requisitos ofrezca.



Los componentes de la vecindad B, C y D condicionan las capacidades que debe ofrecer el componente A

- </> Como ejemplo de esta faceta podemos considerar el modelo arquitectónico que permite construir etiquetas como core-pages, paper-tabs y core-menu.



Core-selector es una adaptación arquitectónica de core-selection dirigida a dar respuesta a las necesidades de notificación de cambio que requieren core-pages, core-tabs y core-menu entre otros

Principios de Diseño en Componentes Web

Principios de Diseño

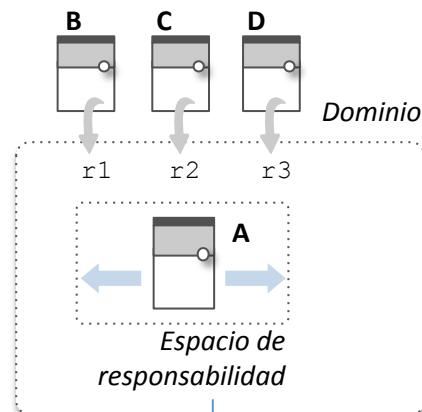
Principio de Reutilización



Respetá El Dominio

Modula la demanda de requerimientos de tus clientes con las restricciones impuestas por la división de responsabilidades que has hecho en el marco de tu dominio.

- 🎓 La creación de nuevos componentes deberá estar siempre en claro alineamiento con las estrategias de división de responsabilidades establecidas en el dominio.



Los requerimientos de los componentes cliente B, C y D no deben desestabilizar al componente A dentro del dominio

- </> Un claro ejemplo de esta faceta lo encontramos en la etiqueta core-header-panel.

El componente proporciona un layout de contenidos usado por otros componentes especialmente en contextos de móvil

```
<core-header-panel>
  <div class="core-header">...</div>
  <div class="content" fit>...</div>
</core-header-panel>
```

Sin embargo su comportamiento no se ve influenciado por el comportamiento ni las necesidades particularidades de los componentes clientes que se agregan dentro de él

Principios de Diseño en Componentes Web

Principios de Diseño

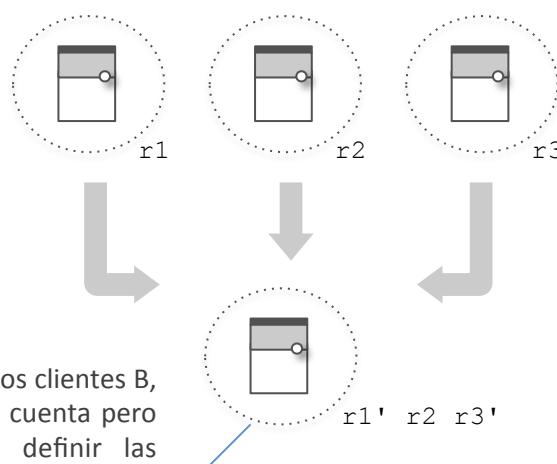
Principio de Reutilización



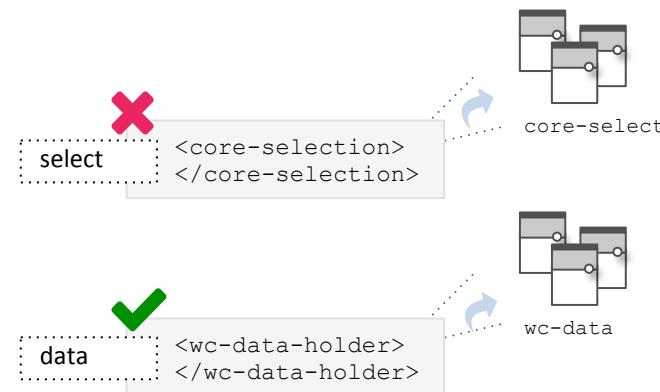
Mantén La Equidistancia

Durante las actividades de diseño y desarrollo de componentes no cedas a las demandas que los demás componentes cliente puedan hacer puntualmente en cada contexto de uso.

- 🎓 Cualquier extensión del contrato debe ser sometida a un riguroso análisis. El contrato debe responder a criterios de dominio y no a necesidades de cliente.



</> Para ilustrar la equidistancia de cliente podemos retomar el ejemplo de abstracción de core-selection que presentamos antes.



Al expresar el contrato del componente a mayor nivel de abstracción se mantiene una mayor distancia de los contextos de uso lo que fomenta la reutilización

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Reutilización

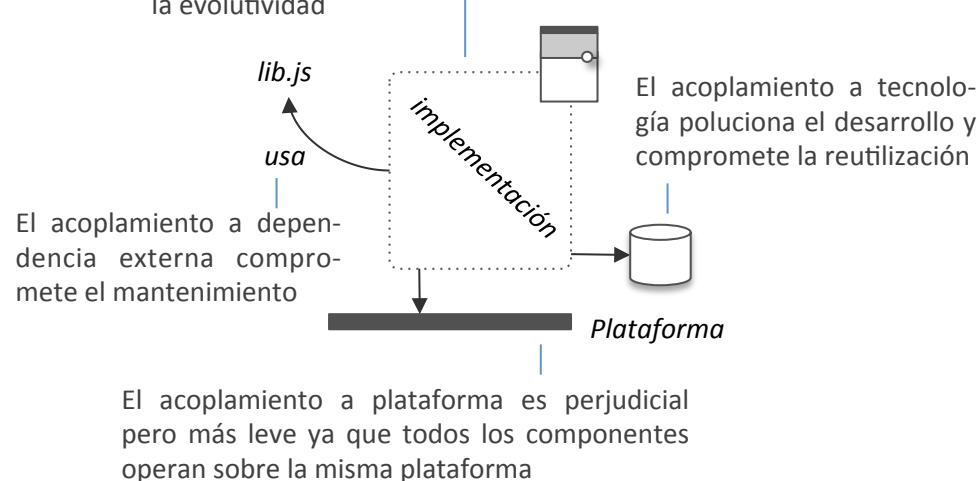


Desacopla Tu Contrato

Cuando diseñas componentes centra tu atención en definir un contrato desacoplado de la tecnología y decisiones de implementación subyacentes.

🎓 Todo tipo de acoplamiento de contrato es un síntoma de mal diseño que compromete la reutilización del componente.

El acoplamiento de contrato a implementación pone en peligro la evolutividad



</> Las etiquetas core-animation-* son un buen ejemplo de componentes que presenta un alto acoplamiento a tecnología.

```
<core-animation id="fadeout" duration="500">
  <core-animation-keyframe>
    <core-animation-prop name="opacity" value="1">
    </core-animation-prop>
  </core-animation-keyframe>
  <core-animation-keyframe>
    <core-animation-prop name="opacity" value="1">
    </core-animation-prop>
  </core-animation-keyframe>
</core-animation>
```

Este conjunto de etiquetas expone todos los detalles del modelo de animación CSS sin realizar ninguna abstracción más ni aportar mayor valor

Principios de Diseño en Componentes Web

Principios de Diseño

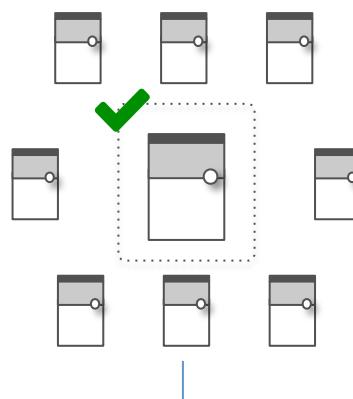
Principio de Reutilización



Persigue La Autonomía

Una buena forma de conseguir un alto grado de reutilización en el ámbito de tu dominio es asegurar el comportamiento autónomo de los componentes.

🎓 El acoplamiento es el gran enemigo de la reutilización. Idealmente etiquetas que no presentan relaciones de dependencia ofrecen alta reutilización.



Si bien la autonomía permite alcanzar cotas altas de reutilización no hay que olvidar que la construcción basada en componentes se fundamenta en relaciones de composición

</> Ya presentamos en el principio de acoplamiento débil algún ejemplo de autonomía. No obstante en Polymer hay muchas opciones para ilustrar esta faceta.

```
<google-hangout-button>  
</google-hangout-button>
```

Se utiliza para incluir un botón que arranca Google hangouts

```
<google-streetview-pano  
panoid="VsCKIVGfpEAAAQJKfdWlw"  
heading="330"  
pitch="-2"  
zoom="0.8"  
disableDefaultUI>  
</google-streetview-pano>
```

Una bonita panorámica de las montañas Machu Pichu en Perú

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Reutilización

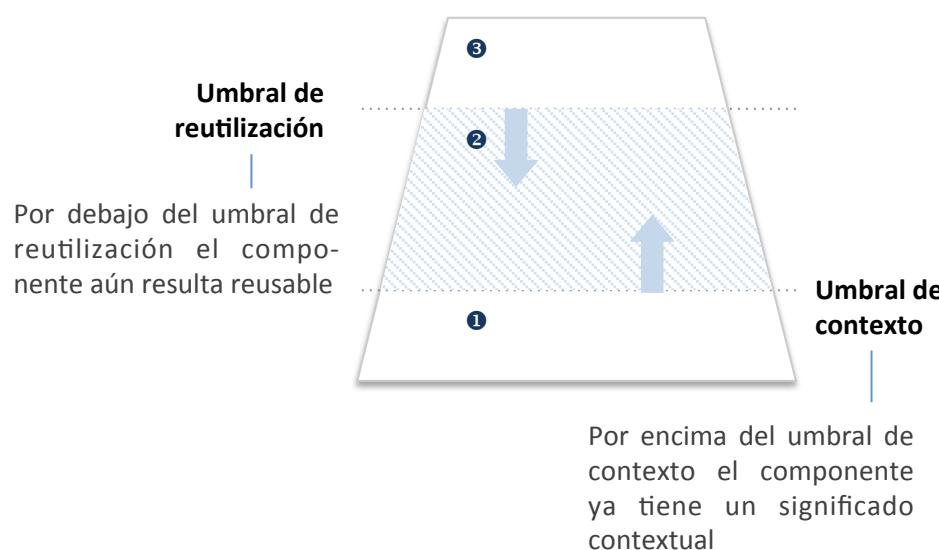


Vigila La Granularidad

Durante el desarrollo de soluciones compositivas asegúrate de que el nivel de granularidad no resulta demasiado grande como para comprometer su valor reutilizable.

🎓 El nivel de granularidad que alcanza un componente por sucesivas encapsulaciones puede llegar a poner en riesgo su capacidad de reutilización.

</> Para ilustrar el significado de los umbrales de contexto y reutilización seleccionaremos algunos ejemplos de componentes en cada región.



①

core-image
core-icon, core-iconset
core-ajax, core-meta

Los componentes presentan una altísima reutilización pero no tienen gran significado contextual

②

core-pages, core-menu
paper-tabs, paper-fab
core-drawer-panel

Los componentes presentan alta reutilización y además tienen un contexto de aplicación claro

③

google-map
google-chart
google-youtube

Los componentes están dedicados a un dominio tan específico que su capacidad de reutilización es menor

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Reutilización



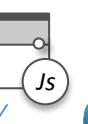
Persigue La Declaratividad

En la medida de lo posible, construye componentes que puedan operar exclusivamente a partir de la configuración semántica que reciben desde el marcado HTML.

- 🎓 Las etiquetas que deben ser configuradas, compuestas, adaptadas o explotadas mediante la inclusión de código presentan un nivel de reutilización menor.



Los componentes que se instancian declarativamente son cómodos y fáciles de usar



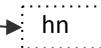
Desarrollador
Web

Los componentes que requieren configuración o código pegamento hacen más improbable su uso

- </> Un contrajeemplo del uso del diseño declarativo lo encontramos en la etiqueta core-a11y-keys.

La etiqueta core-a11y-keys proporciona un mecanismo para procesar eventos de teclado

```
<core-a11y-keys  
target="{}"  
keys="up down left right"  
on-keys-pressed="{{hn}}>  
</core-a11y-keys>
```



Este componente puede resultar de gran utilidad práctica. Sin embargo el hecho de requerir definir una función manejadora en JavaScript compromete su reutilización ya que exige de competencias programáticas al usuario

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Reutilización

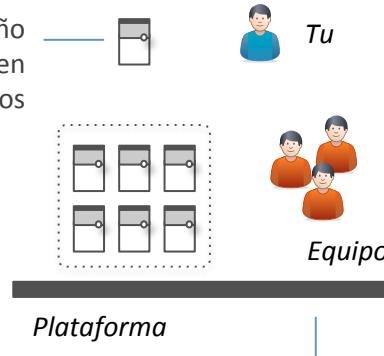


Reduce Los Convenios

En la medida de lo posible evita establecer convenios de desarrollador porque limitan su aplicabilidad al ámbito personal y reducen las posibilidades de reutilización.

🎓 Cuando los convenios no tienen una amplitud de aceptación considerable pueden suponer un problema para el cliente.

Tus convenciones de diseño y desarrollo se convierten en restricciones para otros usuarios



Al menos deberías alinear tus criterios de estandarización con tu organización o equipo de trabajo

</> Presentaremos ejemplos típicos de convenios usados por desarrolladores. El propósito es meramente ilustrativo y no prescriptivo.

my_variable MyMethod init()	myVariable myMethod constructor()
<myWebComponent> myColor = "blue" isVisible = "true"	<my-web-component> my-color = "blue" visible

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Reutilización

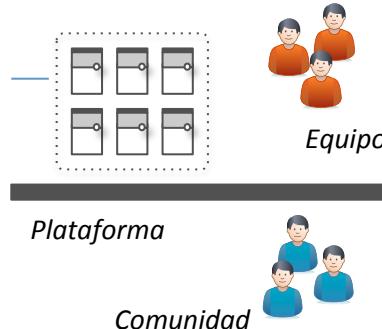


Abraza Los Estándares

Sigue los estándares establecidos en el ámbito de la comunidad, la tecnología y la organización para promover una mayor potencialidad de reutilización.

🎓 Los estándares de comunidad, los impuestos por la tecnología y los de ámbito organizacional tienen alcance suficiente como para poder aplicarlos.

El uso de estándares y convenciones fomenta la reutilización



La aplicabilidad de los estándares crece cuando hay una gran comunidad que los abraza

</> Existen multitud de estándares establecidos dentro de Polymer ya que se trata de un framework que sigue el principio de convención frente a configuración.

HTML

```
on-event = "{{hn}}"  
attributes = "x y z"  
<dom-module>
```

JS

```
this.fire (event)  
attrChanged (before, after)  
properties:{ ... }  
Polymer.dom
```

Estos son sólo algunos ejemplos de convenios establecidos en Polymer. El objetivo es sólo ilustrativo y no tiene ánimo de completitud

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Reutilización



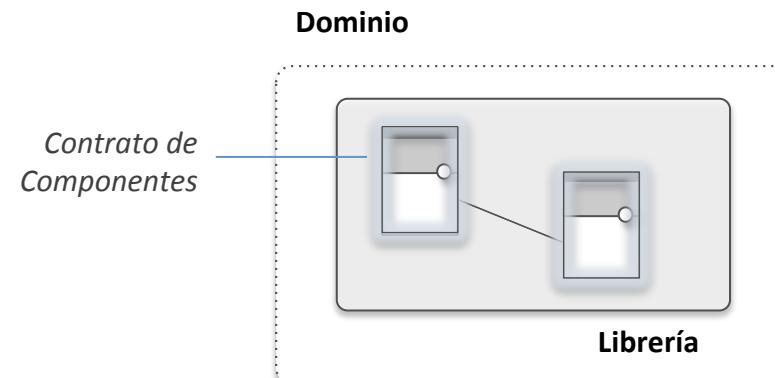
Objetivos

- Fomentar la productividad
- Aumentar la estabilidad de uso
- Mantener el alineamiento a dominio
- Reducir el código pegamento
- Acelerar el time to market
- Prolongar el tiempo de vida

Motivación



Cuanto mayor es el grado de reutilización de un componente mayor resulta su utilidad neta y mejor ayuda en el desarrollo de soluciones basadas en componentes.



Fases



Diseño de Componentes
Mantenimiento Evolutivo

Región



Contrato de Componentes

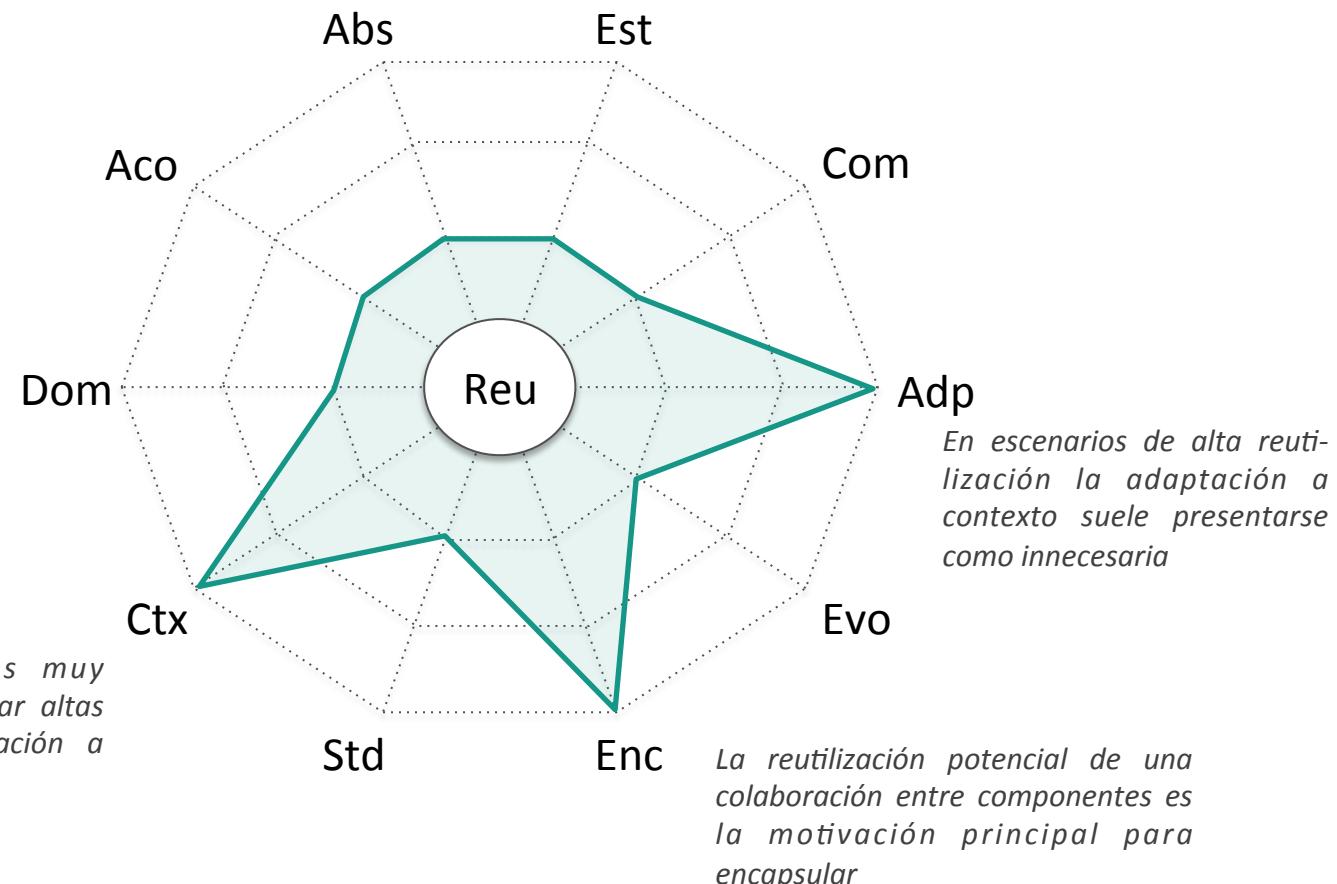
Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Reutilización



Tensiones



Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Reutilización



Ejemplo

Material Design

Cada componente funciona **desde cero** sin demasiada configuración como cualquier etiqueta estándar

En general cada componente es independiente y **autónomo** de los demás



La descomposición en componentes que se ha hecho en material responde a un **entendimiento** claro de las necesidades comunes de los **clientes**

Cada componente responde a una necesidad **común**

Se establecen unos **estándares** visuales y de UX en el marco de material

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Diferimiento de Estado



Nombre

Diferimiento de Estado



Ningún componente debería ser responsable del mantenimiento de su información de estado entre sesiones. Si existe, esta lógica debería diferirse a un componente externo.



Actores



Arquitecto & Desarrollador de Componentes

Maquetador & Desarrollador Web

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Diferimiento de Estado

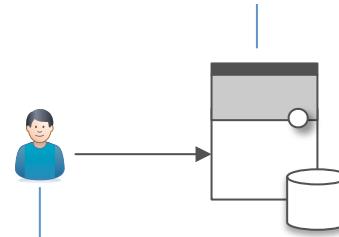


Los Componentes Encapsulan Estado

En términos generales, todos los elementos de una solución orientada a componentes pueden poseer un estado interno que mantienen durante la sesión activa.

🎓 El principio de diferimiento de estado no niega la existencia de estado en los componentes sino que predica técnicas de diferimiento en su gestión.

Todo componente dispone de un estado interno que mantiene mientras se encuentra activo



El usuario es consciente de que el comportamiento del componente depende del estado en curso

</> Es raro encontrar componentes que no dispongan de estado interno. Mostraremos a modo de ejemplo categorías según el estado principal que encapsulan.

selected

core-pages, core-menu
paper-tabs, paper-fab
core-drawer-panel

map

google-map
google-map-directions
Google-map-search

icon

core-icon, core-iconset
core-iconset-svg
paper-icon-button

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Diferimiento de Estado

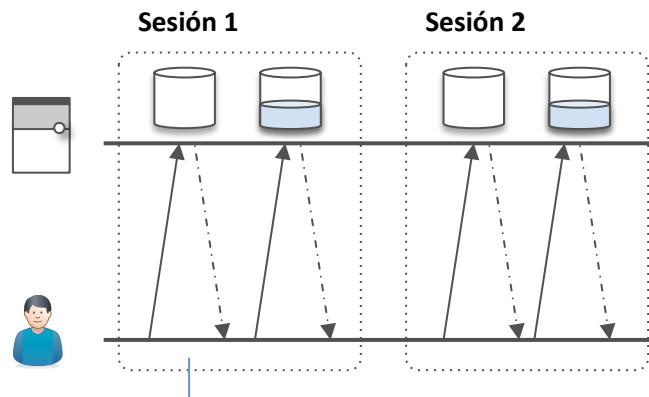


Valora La Gestión De La Información De Estado

Antes de complicar la arquitectura valora si merece la pena hacer que el componente disponga de estado persistente entre sesiones.

🎓 En la mayoría de las ocasiones, el usuario puede asumir que la información de estado no se mantenga más allá de la sesión de trabajo en curso

</> Los componentes de la familia de google-map son un buen ejemplo donde tal vez no es tan importante mantener el estado entre sesiones.



Para el usuario es asumible que cada vez que se carga el componente se empiece sin información de estado

Google-map



¿De verdad que es tan importante que cada vez que arranques se recuerde el lugar exacto donde dejaste posicionado el mapa o es posible arrancar siempre desde una localización por defecto?

Principios de Diseño en Componentes Web

Principios de Diseño

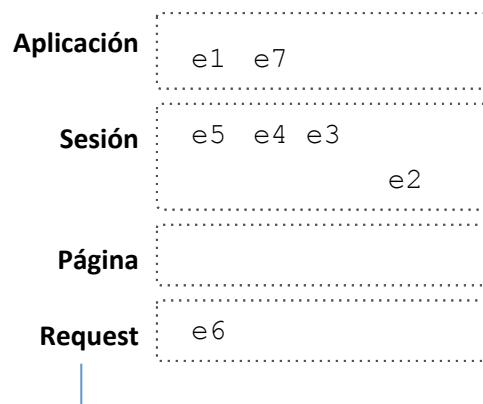
Principio de Diferimiento de Estado



Organiza La Información De Estado

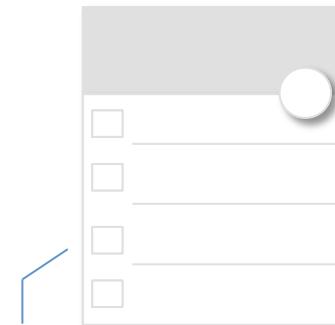
Organiza la información de estado para determinar qué información persistir y en qué ámbito de perdurabilidad colocar cada dato.

🎓 Para determinar la complejidad de tu problema en relación al diferimiento de estado determina cuál es el tiempo de vida que tiene cada dato.



Los niveles de tiempo de vida suelen estar vinculados con el ámbito donde operan los datos. No obstante, también es posible organizarlos temporalmente como se hace al establecer la caducidad de las cookies

</> Un ejemplo de componente con varios niveles de información de estado podemos encontrarlo en la etiqueta core-list.



Cada ítem de la lista debe mantenerse entre sesiones

El estado de los checkbox de la lista no es susceptible de persistirse entre sesiones

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Diferimiento de Estado

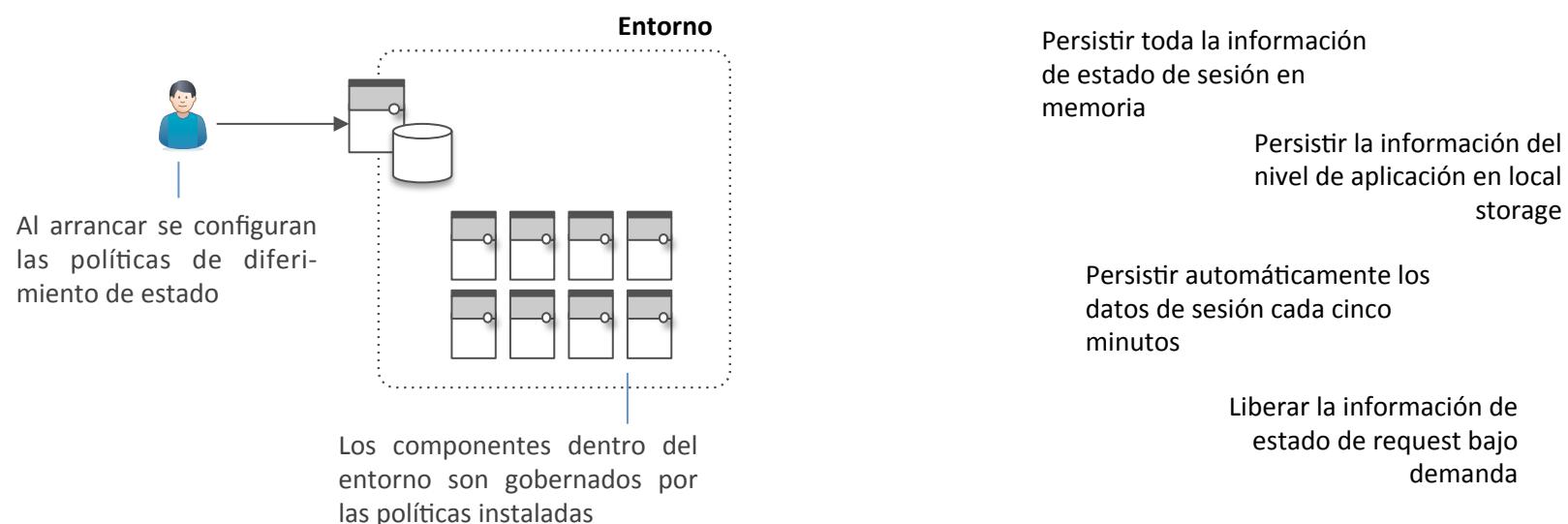


Define Políticas De Diferimiento

Las políticas de diferimiento determinan cómo, cuándo y dónde debe diferirse la persistencia de cada tipo de información así como qué técnica utilizar si es que existe varias disponibles.

🎓 Idealmente, las políticas deberían ser configurables y dinámicamente cambiantes a lo largo del tiempo en función de condiciones ambientales.

</> Simplemente a modo ilustrativo pondremos algunos ejemplos de políticas prototípicas que se vinculan a este tipo de escenarios.



Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Diferimiento de Estado

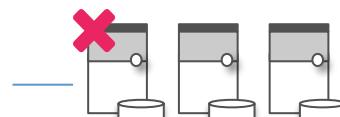


Centraliza La Gestión De Estado

Una buena arquitectura debería utilizar un único componente para articular la gestión de persistencia de estado entre sesiones liberando de tal responsabilidad al resto de componentes.

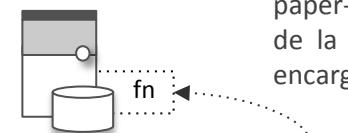
- 🎓 La centralización desacopla de la tecnología a cada componente lo cual redundá en mayores cotas de reutilización.

No es buena idea delegar la persistencia de estado localmente en cada componente



- </> Pongamos como ejemplo dos componentes de selección que necesitan persistir su estado de selección.

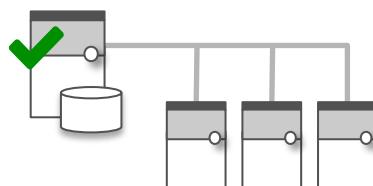
Gestor de Estado



Los componentes core-pages y paper-tabs no se preocupan de la persistencia. De ello se encarga el escuchador fn

```
<div on-core-select="{{fn}}>
  <core-pages>
    <div>Page 1</div>
    <div>Page 2</div>
  </core-pages>
  ...
  <paper-tabs>
    <paper-tab>1</paper-tab>
    <paper-tab>2</paper-tab>
  </paper-tabs>
</div>
```

Gestor de Estado



El principio de diferimiento de estado promueve en su lugar la idea de centralizar esta problemática

Componentes agnósticos de persistencia de estado

Principios de Diseño en Componentes Web

Principios de Diseño

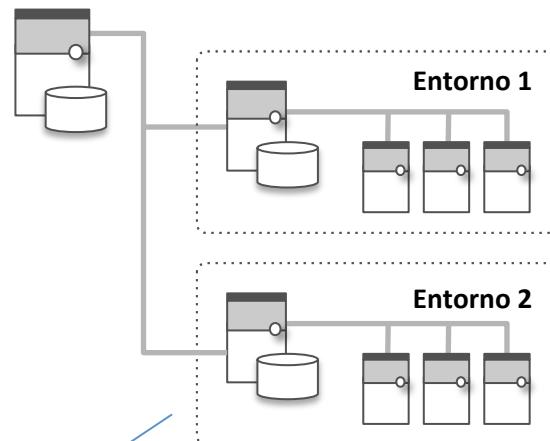
Principio de Diferimiento de Estado



Federa La Gestión De Estado

Para fomentar la escalabilidad considera federar la arquitectura de diferimiento de estado utilizando varios gestores de estado locales conectados entre sí.

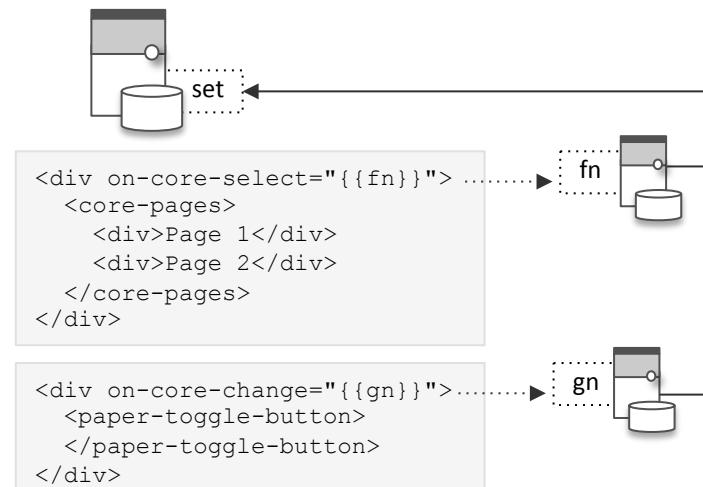
🎓 La encapsulación como actividad nuclear de los componentes Web promueve el uso de arquitecturas federadas para el diferimiento de estado.



Cada entorno se vincula generalmente a un ámbito de encapsulación de componentes

</> En este ejemplo mostramos dos componentes que deben aplicar políticas de gestión diferentes.

Gestor de Estado



Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Diferimiento de Estado

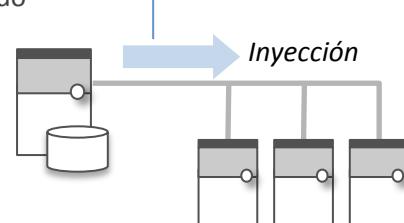


Facilita La Inyección

En la medida de lo posible construye componentes cuyos esquemas de configuración e inicialización permitan injectar la información de estado fácilmente.

- 🎓 Utiliza atributos semánticos de marcado HTML y adscríbete a algún convenio de código para poder realizar la inyección de estado de forma transparente.

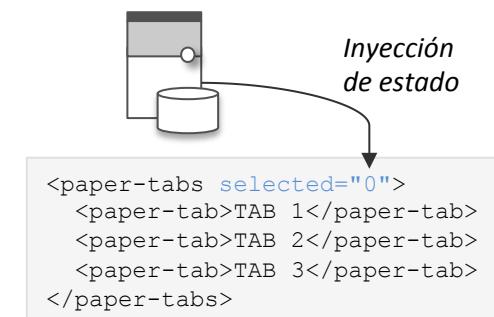
Los convenios y estándares de configuración confieren la homogeneidad contractual necesaria para centralizar la inyección de estado



Para los componentes la inyección de estado es un mero ejercicio de configuración

- </> Un buen ejemplo de inyección de estado de forma sencilla es mediante el uso de atributos HTML. Esta manera permite además aplicar técnicas de data binding.

Gestor de Estado



Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Diferimiento de Estado

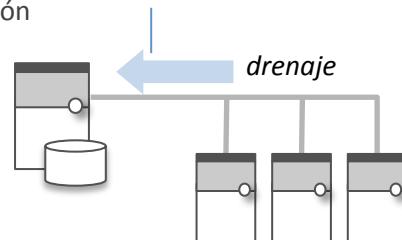


Facilita El Drenaje

Recíprocamente, procura que los componentes provean mecanismos sencillos de lectura de la información de estado para facilitar el drenaje.

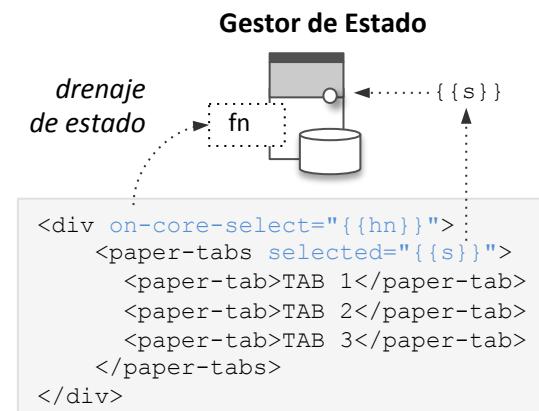
🎓 Nuevamente aplica estándares sobre el contrato y estrategias declarativas para permitir que otros componentes capturen el estado.

Al establecer mecanismos estándares para leer la información de estado se facilita el drenaje y se permite la centralización de la gestión



Para los componentes el drenaje es una mera operación de lectura sobre su estado interno

</> El mismo ejemplo de paper-tabs es bueno para ilustrar cómo muchas veces se pueden proporcionar de forma simultánea varias formas de drenaje de estado.



Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Diferimiento de Estado

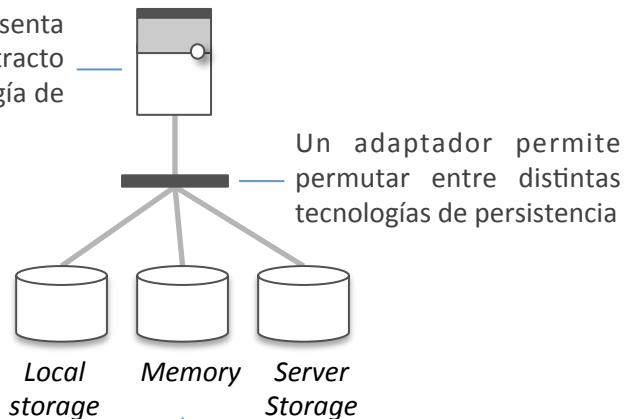


Desacopla La Tecnología

El componente responsable de llevar a cabo la gestión de la información de estado no debería estar acoplado con ninguna tecnología de persistencia en particular.

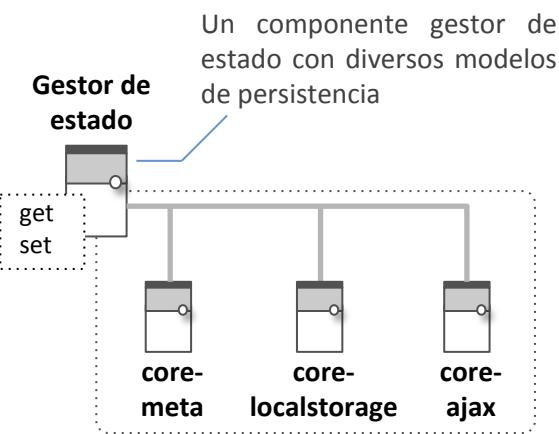
🎓 Mediante adaptadores y proxies debería ser posible cambiar la tecnología subyacente de persistencia de manera transparente.

El gestor de estado se presenta como un componente abstracto desacoplado de la tecnología de persistencia



Pueden utilizarse diferentes tecnologías de persistencia para el diferimiento de estado o vincularse a diferentes políticas

</> Polymer ya provee los medios necesarios para dar soporte a distintos modelos de persistencia de la información de estado.



Core-meta implementa una pequeña base de datos en memoria. Core-localstorage da acceso a local storage y con core-ajax se puede persistir en servidor

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Diferimiento de Estado



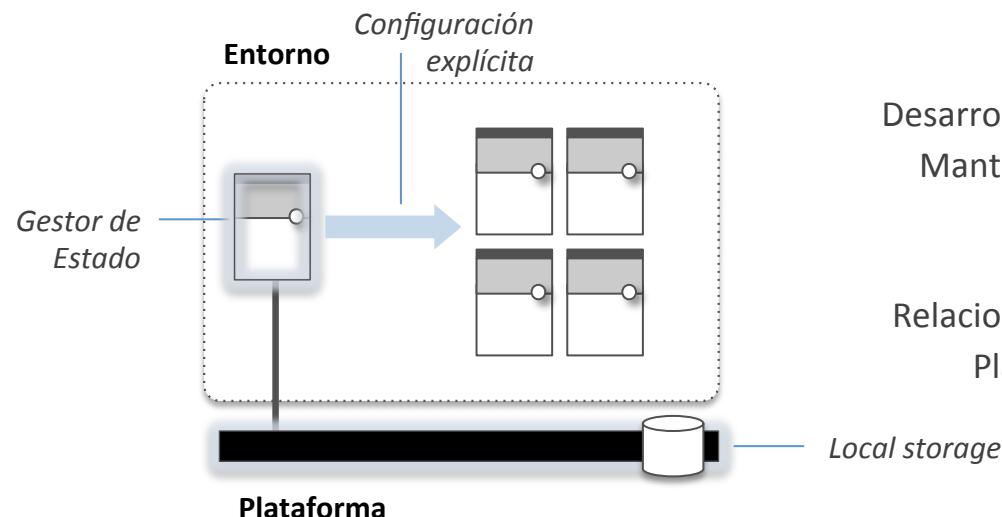
Objetivos

- Disminuir el acoplamiento tecnológico
- Agilizar los procesos de inicialización
- Facilitar el desarrollo
- Simplificar el uso
- Centralizar la lógica específica común
- Fomentar la estandarización

Motivación



El diferimiento de estado aligera y simplifica los procesos de construcción de componentes, responde a un equilibrio en la división de responsabilidades y fomenta la reutilización.



Fases



Desarrollo de Componentes
Mantenimiento Evolutivo

Región



Relaciones de dependencia
Plataforma & Entorno

Principios de Diseño en Componentes Web

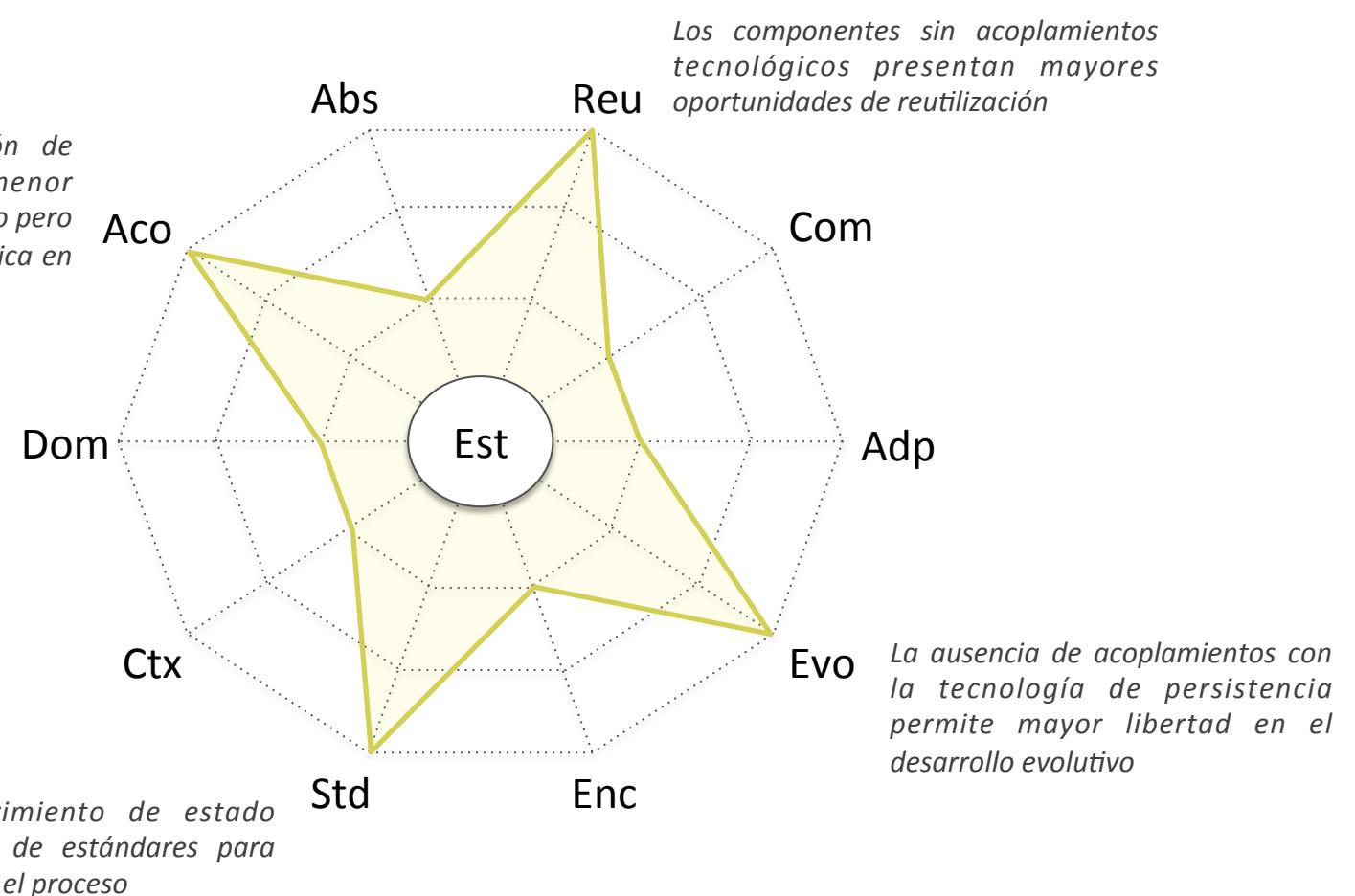
Principios de Diseño

Principio de Diferimiento de Estado



Tensiones

La ausencia de gestión de estado supone un menor acoplamiento tecnológico pero obliga a diferir dicha lógica en un componente externo



Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Diferimiento de Estado

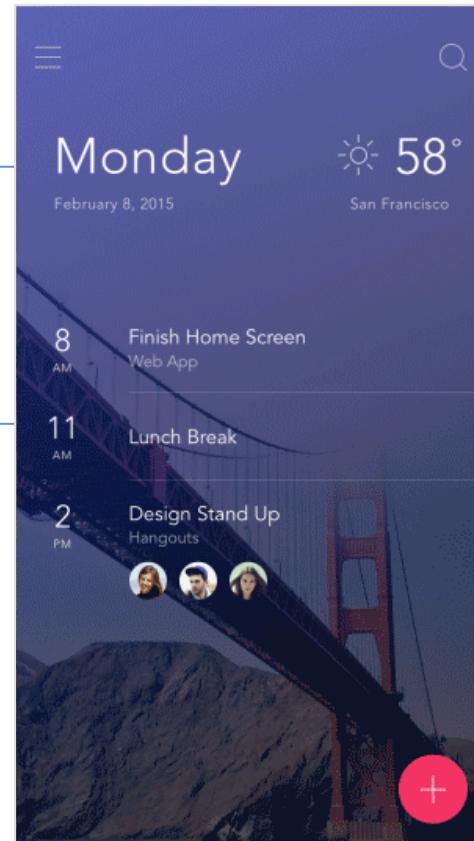


Ejemplo

Secret Project

Estos datos son de configuración y se alimentan del Hw del dispositivo

La agenda y los detalles de cada cita son un típico ejemplo de información de **estado** con caducidad temporal



Es de esperar que la información de estado sea inyectada por **configuración** y **drenada** al finalizar la sesión en un componente gestor de estado

<http://goo.gl/F3VUpw>

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Compositividad



Nombre

Compositividad



Diseña tus componentes de manera que se maximicen las oportunidades de composición con otros componentes presentes en el contexto de uso.



Actores



Arquitecto & Desarrollador de Componentes

Maquetador & Desarrollador Web

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Compositividad



Todo Son Componentes

En la construcción de soluciones orientadas a componentes Web todas las responsabilidades tanto del dominio del problema como de la solución están acomodadas en componentes.

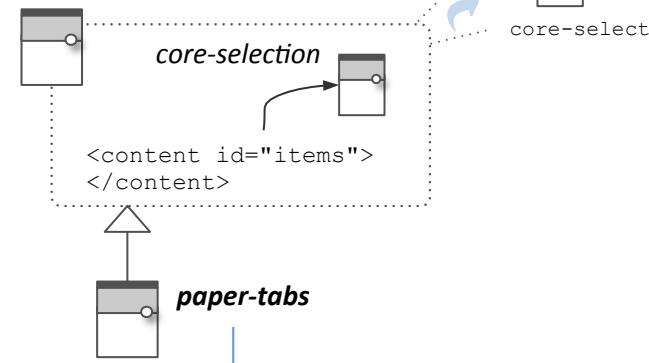
🎓 Pese a que el paradigma está centrado en la vista, incluso las responsabilidades más transversales sin vista se articulan a través de componentes.



En las arquitecturas de orientación a componentes las responsabilidades con frecuencia se organizan por capas. Dentro de ellas, todo son componentes

</> Para ilustrar el hecho de que en Polymer todo está basado en componentes analicemos la anatomía interna de core-selector y paper-tabs.

core-selector



Paper-tabs hereda de core-selector que a su vez introduce por inserción a los agregados y delega en un core-selection. Esto ilustra una compleja colaboración en la que todo son componentes

Principios de Diseño en Componentes Web

Principios de Diseño

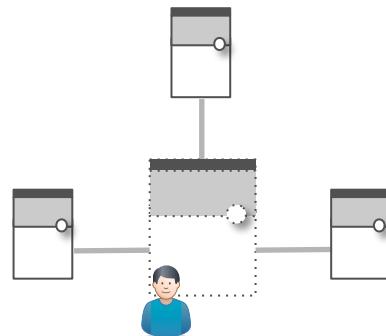
Principio de Compositividad



Diseña Para Componer

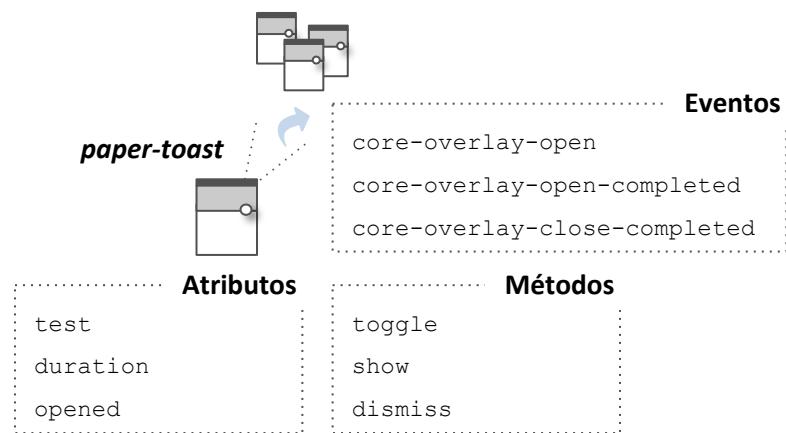
Cada vez que diseñas un componente ten presente que lo haces para ofrecer un elemento de composición que conecta con otros componentes en el marco de un contexto de uso.

- 🎓 A diferencia de lo que ocurre con las etiquetas estándar que funcionan con alta autonomía, los componentes Web se centran en capturar esquemas de composición.



Un componente se diseña mirando las oportunidades de composición que ofrece en potenciales contextos de colaboración

- </> Las posibilidades de composición están en relación directa con el número de medios por los que un componente ofrece acceso. Estudiemos el caso de paper-toast.



Paper-toast permite añadir mensajes momentáneos al pie. Dado que este componente ofrece acceso a través de métodos, atributos y eventos presenta un alto nivel de compositividad potencial

Principios de Diseño en Componentes Web

Principios de Diseño

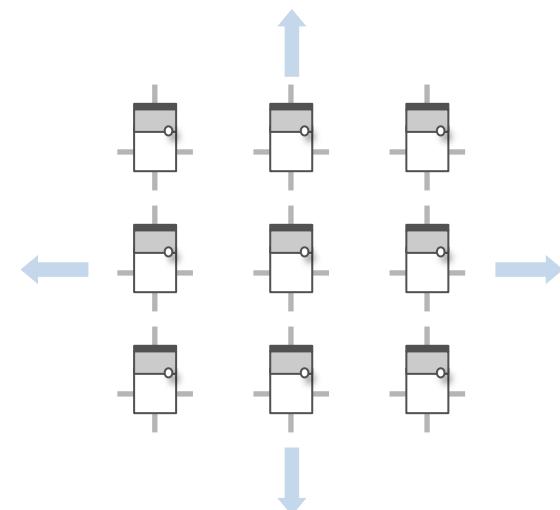
Principio de Compositividad



Asegura La Escalabilidad

Cualquier componente debe al menos mantener el grado de compositividad que exponen sus vecinos y no levantar fronteras que cierren el sistema monolíticamente.

🎓 Los componentes deben ofrecer una experiencia funcional de usuario cerrada pero a la vez mantener la receptividad a nuevas composiciones.



</> Donde más patente queda el mantenimiento de la escalabilidad es en relación a las arquitecturas de eventos. Veámoslo en paper-tabs.

```
<div on-core-select="{{hn}}">
  <paper-tabs>
    <paper-tab>1</paper-tab>
    <paper-tab>2</paper-tab>
  </paper-tabs>
</div>
```

Un cuadro que muestra un fragmento de código HTML para un componente 'paper-tabs'. Debajo del cuadro, se comparan dos fragmentos de código JavaScript que manejan el evento 'e.stopPropagation()'. El primer fragmento, marcado con un 'X', tiene la línea ': e.stopPropagation ()'. El segundo fragmento, marcado con un checkmark verde, tiene la línea ': e.stopPropagation ()' seguida de 'this.fire (e');

Para asegurar la escalabilidad compositiva evita en todo momento parar la propagación de eventos y si lo haces que sea para subir el nivel de abstracción del mismo

Principios de Diseño en Componentes Web

Principios de Diseño

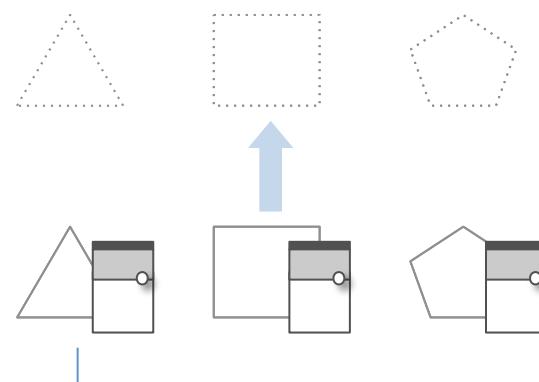
Principio de Compositividad



Reduce El Código Pegamento

En la medida de lo posible, cuando diseñes componentes, reduce la necesidad de incluir código específico para cohesionar las diferentes partes de la arquitectura de cliente.

- 🎓 Los componentes son soluciones de validez probada a problemas recurrentes. El código específico debería ser mínimo en este tipo de aproximaciones.



Los componentes se construyen por diseño para encajar en el marco arquitectónico al que se orientan. Por tanto los desarrolladores no deberían necesitar implementar código específico sino simplemente operar incluyendo componentes en su solución HTML

</> Un buen ejemplo de composición sin uso de código pegamento lo encontramos en los componentes de google-map-*.

El componente google-map y google-map-directions operan compositivamente en una colaboración sin más que compartir una variable por data-binding

```
<google-map map="{map}" .....> {map} </google-map>
```

```
<google-map-directions map="{map}" .....> {map} </google-map-directions>
```

Principios de Diseño en Componentes Web

Principios de Diseño

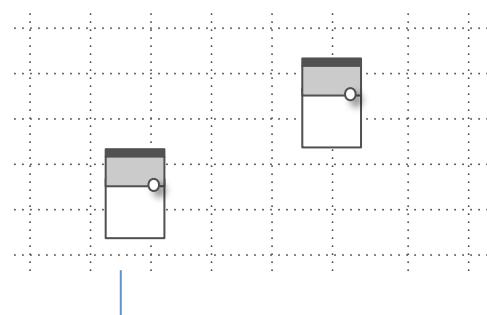
Principio de Compositividad



Compón Sobre El Layout

Cuando diseñas una composición considera los condicionantes impuestos por el layout de la solución buscada y de los componentes implicados en la misma.

- 🎓 El layout de la solución condiciona las posibilidades compositivas de que dispone un componente para conectarse con la vecindad.



La posición relativa dentro del contexto DOM de los elementos puede complicar las oportunidades compositivas. En el principio de contextualización estudiaremos como mitigar estos problemas

- </> Un claro ejemplo en el que la disposición de los elementos en relación al contexto DOM dificulta la composición es en la comunicación por eventos.

Se impone que los escuchadores de eventos se posicionen en la cadena de ascendentes

```
<div on-core-select="{{ hn }}>
  <paper-tabs>
    <paper-tab>1</paper-tab>
    <paper-tab>2</paper-tab>
  </paper-tabs>
  <core-pages>
    <div>1</div>
    <div>2</div>
  </core-pages>
</div>
```

Core-pages y paper-tabs se ven forzados a colocarse bajo el escuchador que define el comportamiento reactivo

Principios de Diseño en Componentes Web

Principios de Diseño

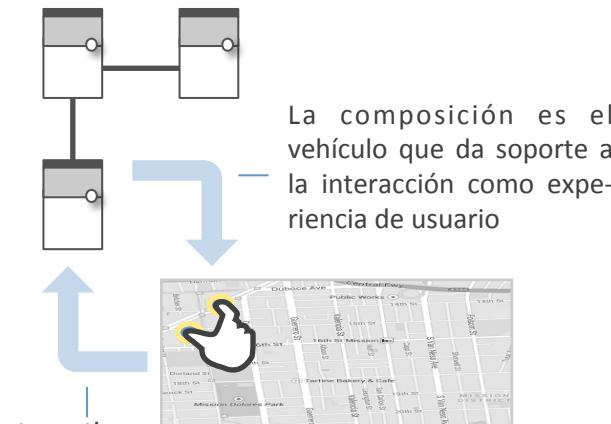
Principio de Compositividad



Compón A Través De La Interacción

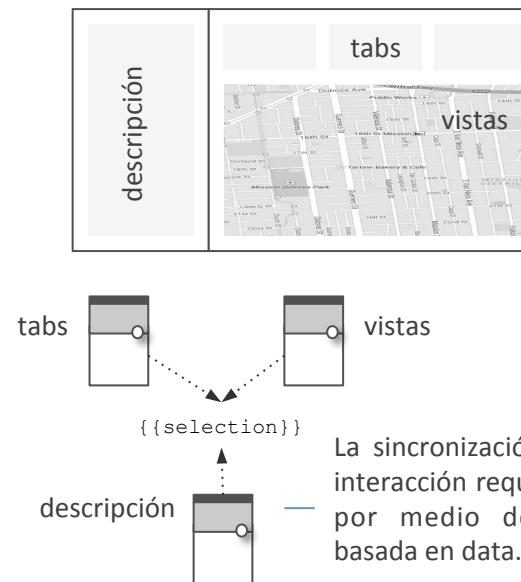
Cuando diseñas una composición guíate por los objetivos perseguidos en el marco de la experiencia interactiva de usuario a la que pretendes dar soporte con la misma.

- 🎓 Cada vez que componemos buscamos articular una experiencia de usuario que se materializa en una interacción sobre los componentes involucrados.



La experiencia interactiva debe ser el driver fundamental para determinar qué composición articular

- </> Considera un típico problema de coordinación entre componentes que deben mantenerse sincronizados para capturar la interacción de los usuarios.



Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Compositividad



Busca La Continuidad

Cuando diseñas una composición asegúrate de que no dificultas la continuidad de la experiencia de usuario ni en el plano visual ni en el interactivo.

- 🎓 Una composición correcta es aquella en la que no es posible discernir las partes implicadas ya que no presenta fisuras de continuidad visuales o interactivas.

Una experiencia visual unitaria sin fronteras ni disrupciones



Una experiencia interactiva cohesionada y homogénea

- </> Un ejemplo de composición armónica visual lo encontramos entre los componentes de google-map y google-map-search.

La continuidad visual permite integrar de forma fluida los elementos por yuxtaposición



Visualmente la experiencia de usuario es unitaria. La yuxtaposición visual asegura que el usuario no es consciente de que hay 2 componentes colaborando

Principios de Diseño en Componentes Web

Principios de Diseño

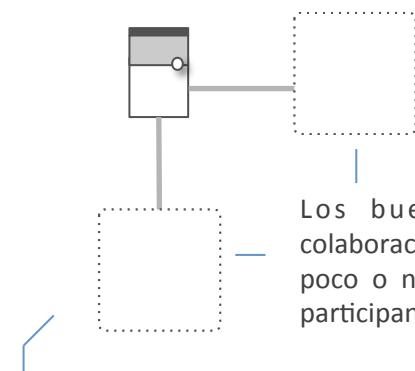
Principio de Compositividad



Abstacta La Composición

Cuando diseñas una composición asegúrate de que la defines conectando con componentes establecidos a alto nivel de abstracción.

- 🎓 Se deben imponer las menores restricciones sobre los elementos de una composición para aumentar el número de posibles candidatos a participantes.

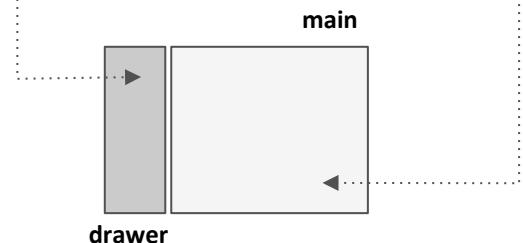


Existen 3 niveles de restricción compositiva creciente. Exigir que los colaboradores sean cualquier contenido HTML, exigir que sean componentes que implementen cierto contrato parcial o exigir que hereden de cierto prototipo

- </> El diseño abstracto es frecuente en muchos componentes de Polymer. A modo de ejemplo veamos el funcionamiento de core-drawer-panel.

Los elementos anidados dentro de core-drawer-panel pueden ser cualquier tipo de contenido

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



Principios de Diseño en Componentes Web

Principios de Diseño

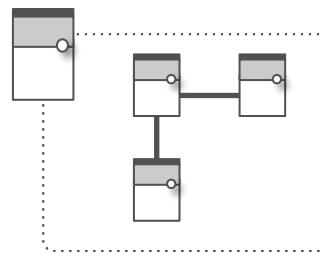
Principio de Compositividad



Encapsula La Composición

Guíate por las composiciones más recurrentemente utilizadas en el marco de tu dominio para identificar las encapsulaciones que debas establecer como elementos reutilizables.

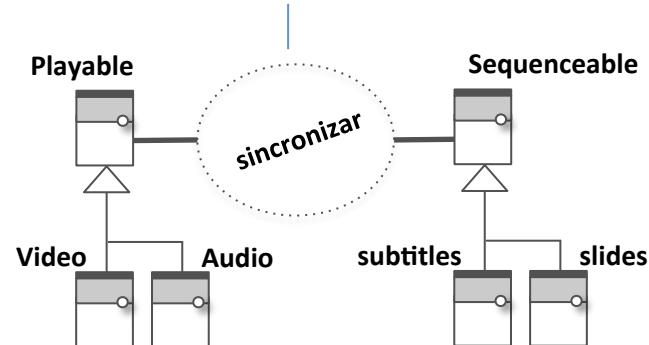
- 🎓 Cuando encapsulamos pretendemos capturar interacciones soportadas en base a las composiciones que conectan los componentes.



Contrariamente a lo que se suele pensar dentro de una encapsulación suelen ser más importante las conexiones entre componentes que los propios componentes en si mismos

- </> A modo de ejemplo pensemos en un componente video que debe sincronizarse con un componente que muestra los substitutos.

En este escenario es más importante capturar la sincronización que la naturaleza particular de los componentes implicados



El video también podría ser un audio

Los subtítulos también podrían ser unas transparencias

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Compositividad



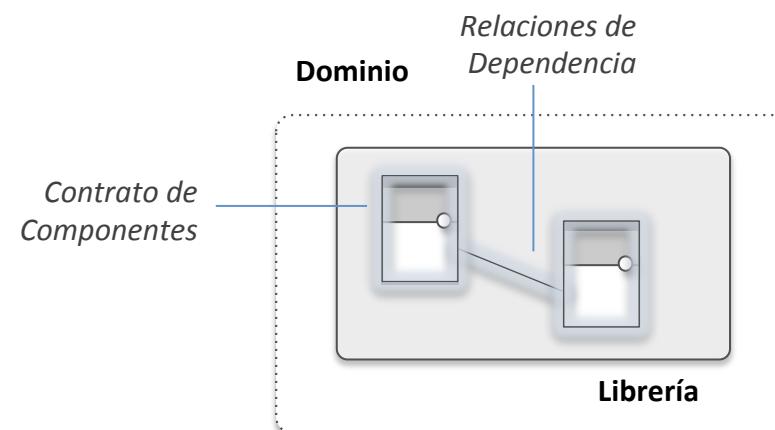
Objetivos

- Maximizar la reutilización
- Aumentar la versatilidad
- Fomentar la economía de código
- Aumentar la interoperabilidad
- Reducir la carga de trabajo

Motivación



Al fomentar la compositividad se consigue aumentar la reutilización de los componentes y mejorar la división de responsabilidades dentro del dominio lo que a su vez facilita la construcción de aplicativos.



Fases



- Desarrollo de Componentes
- Uso de Componentes

Región



- Contrato de Componentes
- Relaciones de Dependencia

Principios de Diseño en Componentes Web

Principios de Diseño

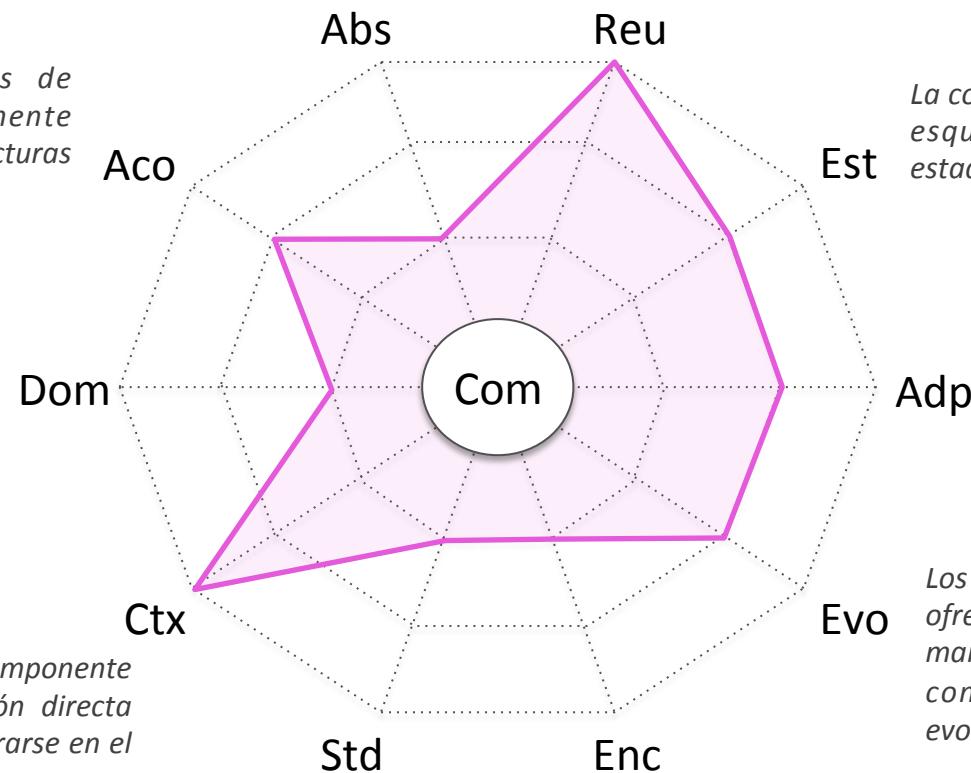
Principio de Compositividad



Tensiones

Mejorar los mecanismos de composición del componente puede conducir a arquitecturas más desacopladas

Las facilidades de composición fomentan el nivel de reutilización del componente



La compositividad de un componente guarda una clara correlación directa con su capacidad para integrarse en el contexto de uso

La compositividad permite articular esquemas de diferimiento de estado

Un alto nivel de compositividad en un componente mitiga la necesidad de aplicar adaptaciones al contexto de uso

Los componentes diseñados para ofrecer una buena compositividad mantienen mayor agnosticismo de contexto lo que fomenta su evolutividad

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Compositividad

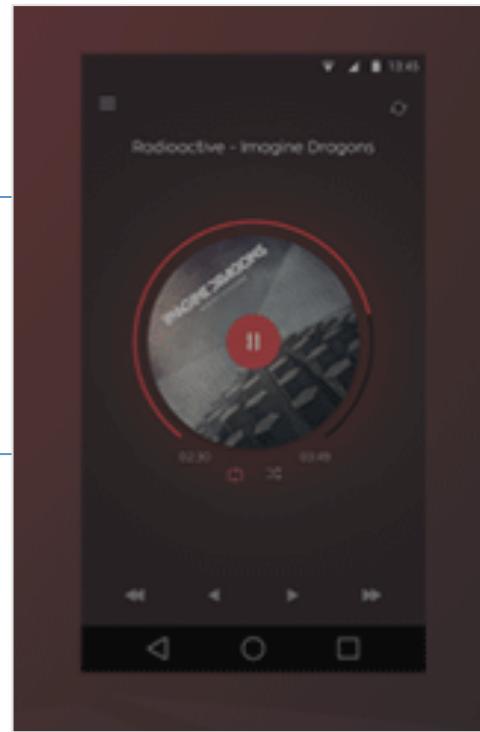


Ejemplo

Music Player App

Cada elemento de este reproductor de audio es un componente Web

Se ha hecho una descomposición estratégica para maximizar las oportunidades de composición



*En la superficie se podrían incluir **escalaramente** tantos más componentes como se quisiera*

*Cada componente opera visualmente sin imponer **fronteras** de contexto*

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Adaptabilidad



Nombre

Adaptabilidad



Asegúrate de que los componentes disponen de la plasticidad suficiente como para adaptarse fácilmente a cualquier contexto de uso.



Actores



Arquitecto & Desarrollador de Componentes

Desarrollador Web

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Adaptabilidad

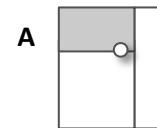


La Adaptación Es Modal

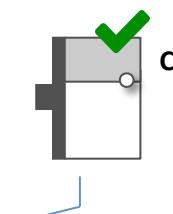
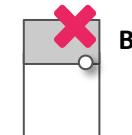
La adaptación de un componente es un ejercicio de transformación opcional que va dirigido a mejorar su acomodamiento al contexto de uso.

🎓 La adaptación hace uso de potentes técnicas meta-programáticas. Sin embargo su aplicación debe entenderse como un ejercicio de carácter opcional.

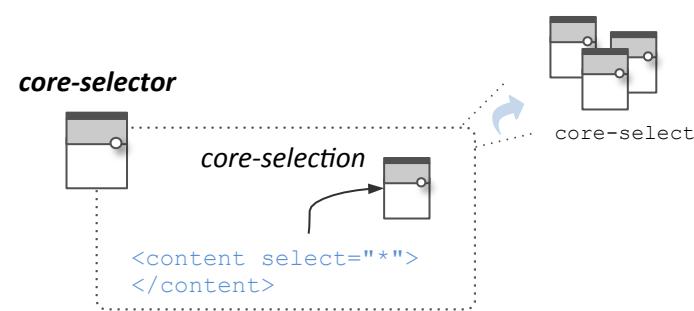
Las características contractuales del componente A imponen ciertas restricciones arquitectónicas que exigen una adaptación en B y C para poder colaborar



La adaptación ayuda a acomodar el componente C para que encaje con A mientras que B requerirá mayor esfuerzo arquitectónico al no tener adaptación



</> Un ejemplo de adaptación modal lo encontramos en core-selector y todos sus derivados dentro de las librerías core-* y paper-*.



Core-selector hace una adaptación de core-selection de manera que los proveedores de selección son los elementos agregados dentro de la etiqueta

Principios de Diseño en Componentes Web

Principios de Diseño

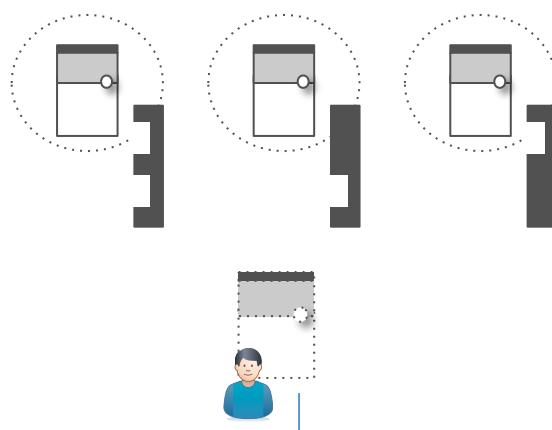
Principio de Adaptabilidad



Conoce El Problema

Se consciente de las adaptaciones potenciales que sufrirán tus componentes en el marco de los distintos contextos de uso para tomar decisiones de carácter arquitectónico sobre ellos.

- 🎓 Algunos componentes demandan adaptaciones canónicas para poder operar sobre ellos de manera compositiva.



A la hora de diseñar un nuevo componente estudia las adaptaciones que se aplican frecuentemente en el contexto donde éste se usará

- </> La adaptación de core-selection realizada en core-selector es fruto de la reflexión de que muchos componentes tienen a sus agregados por proveedores de selección.



Todos los componentes que heredan de core-selector necesitan adquirir el comportamiento de core-selection pero donde los proveedores de selección son los elementos agregados

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Adaptabilidad

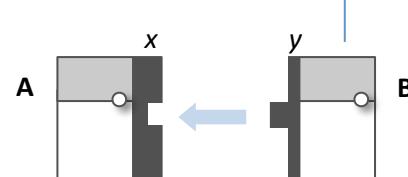


Diseña Para Adaptar

Durante las actividades de diseño ten en cuenta que los componentes requerirán adaptaciones puntuales a distintos contextos de uso.

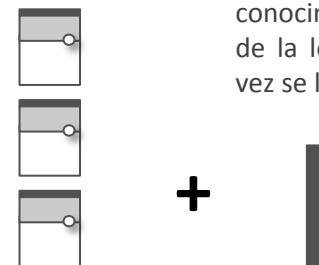
☞ No cierres arquitectónicamente el componente a la posibilidad de realizar adaptaciones dinámicas sobre él.

El componente B expone un contrato con una adaptación, y, que facilita su integración con A



El componente A se diseña para poder integrar una adaptación x complementaria con la adaptación y de B

</> La i18n es un problema transversal que se puede aplicar sobre cualquier componente. Sin embargo requiere ser tenido en cuenta durante el diseño.



La adaptación i18n no tiene conocimiento de dominio pero dispone de la lógica de gestión idiomática una vez se le proporcionan recursos locales

Los componentes de dominio no tienen lógica i18n pero están diseñados para poder adquirirla y operar con ella en cualquier momento. Por ejemplo incluyen recursos idiomáticos locales

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Adaptabilidad

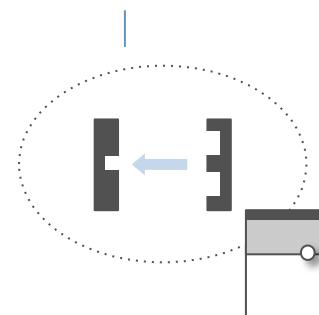


Abstacta Adaptaciones

Dado que las mismas adaptaciones se repiten recurrentemente en el ámbito del dominio conviene identificarlas como nuevos componentes dentro del mismo.

Al aplicar recurrentemente adaptaciones en el marco de un dominio se descubren patrones de adaptación que pueden materializarse como componentes .

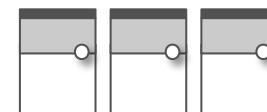
Al abstraer adaptaciones y encapsularlas como componentes éstas se pueden aplicar de manera declarativa



La encapsulación de las adaptaciones en forma de componentes permite homogenizar la arquitectura de manera que todo son componentes

</> El análisis del dominio debe permitirnos identificar qué adaptaciones son recurrentes y deben encapsularse en un componente nuevo.

Componentes de dominio



Se identifica que i18n, i10n y diferimiento de estado son tres lógicas transversales para adaptar componentes



Adaptaciones de dominio

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Adaptabilidad

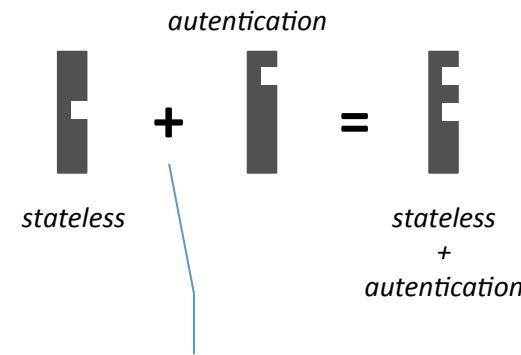
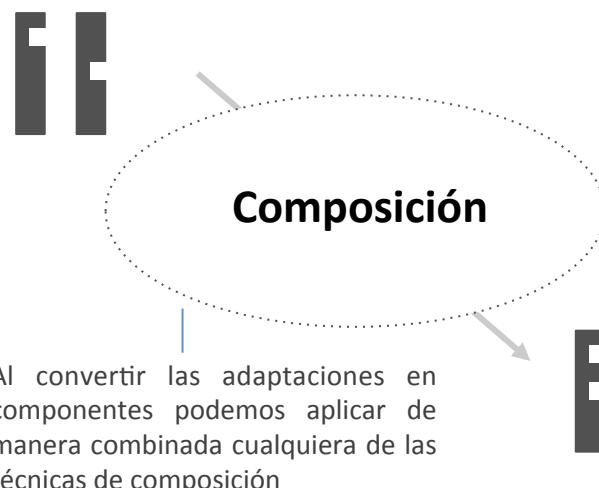


Adapta Abstracciones

Una vez que dispones de varias adaptaciones abstractas, y dado que se han encapsulado en forma de componentes, éstas pueden ser objeto de subsiguientes adaptaciones.

🎓 Desde el mismo momento en que una adaptación se convierte en componente se le pueden aplicar cualquiera de sus técnicas compositivas.

</> Un ejemplo típico lo encontramos al combinar dos lógicas transversales que suelen encontrarse siempre juntas. El diferimiento de estado y la autenticación.



Podemos aplicar, por ejemplo, técnicas aditivas sobre adaptaciones para crear adaptaciones compuestas

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Adaptabilidad

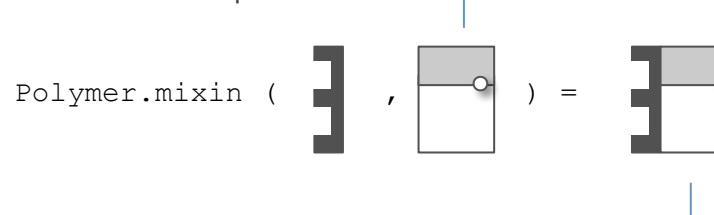


Automatiza La Adaptación

Similarmente, dado que las adaptaciones son componentes, es posible aplicarlas de manera dinámica y automática en tiempo de ejecución mediante las técnicas de composición.

- 🎓 Puede definirse una librería que aplique técnicas compositivas diversas sobre las adaptaciones de manera que el uso de adaptaciones se simplifique.

La aplicación del método mixin sobre una adaptación y un componente permite integrar la adaptación al componente de manera transparente



El resultado es una nueva versión del componente preparado adaptativamente para operar en el contexto de uso sobre el que trabaja la adaptación elegida

- </> Muchos de los componentes de las librerías core-* y paper-* se construyen por mixtura con otros componentes de base de manera automática. Veamos algunos ejemplos.

CoreResizer

core-animated-pages
core-overlay
core-splitter

CoreFocusable

core-tooltip
paper-button-base
paper-dropdown-menu

CoreResizable

core-list
core-scroll-header-panel
core-tooltip
paper-tabs

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Adaptabilidad

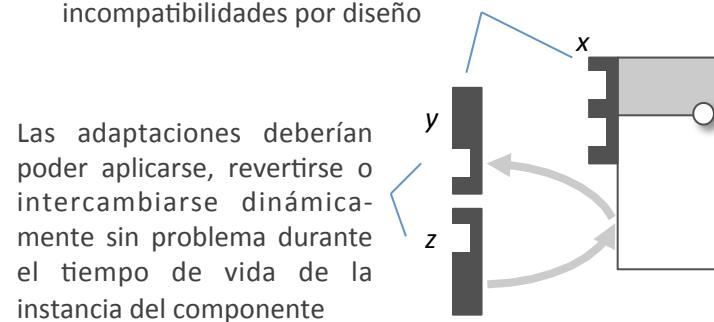


Fomenta La Reversibilidad

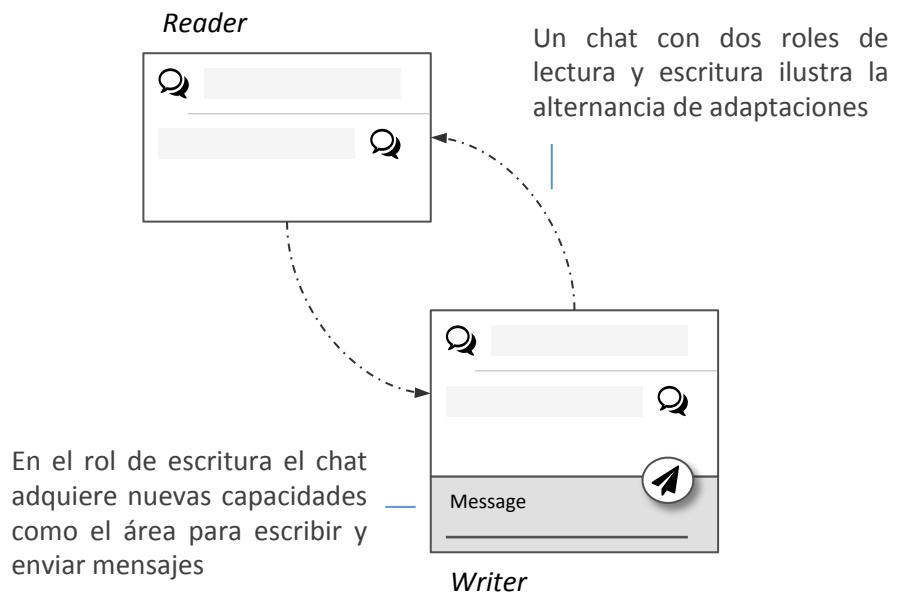
En términos ideales una adaptación debería tener una aplicabilidad reversible y sustitutiva por nuevas adaptaciones aplicables cuando se cambia dinámicamente de contexto.

- 🎓 La integración de adaptaciones no debería alterar indefinidamente la esencia funcional de un componente. En su lugar, éste debería poder abandonar la adaptación si fuera necesario.

Un componente puede incorporar varias adaptaciones de manera simultanea siempre que éstas no presenten incompatibilidades por diseño



- </> Un usuario cambiará potencialmente de rol multitud de veces durante una sesión de trabajo lo que le debería dar acceso a distintas vistas y capacidades.



Principios de Diseño en Componentes Web

Principios de Diseño

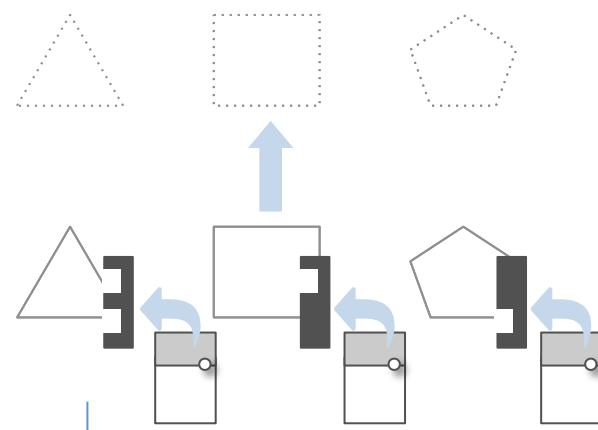
Principio de Adaptabilidad



Adapta Para Alinear

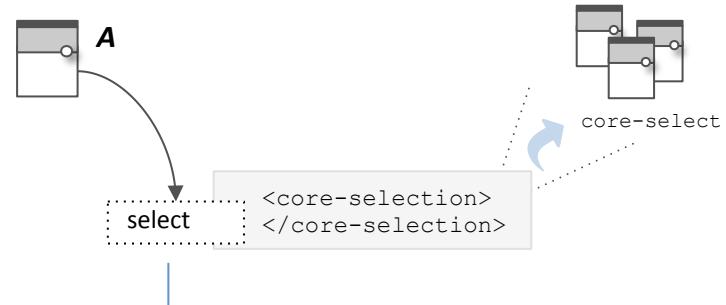
Define adaptaciones para aplicar sobre los componentes de forma automática y permitir que encajen sin problema en los contextos arquitectónicos de uso más frecuentes.

- 🎓 Acomoda siempre los componentes a la arquitectura. No adaptes la arquitectura para que encaje en las restricciones de contrato de los componentes.



Los componentes se adaptan para encajar en los contextos arquitectónicos de uso más frecuentes

- </> El componente core-selection es un buen ejemplo de adaptación dirigida a alcanzar alineamiento arquitectónico.



En aquellos casos donde la arquitectura de componentes lo requiere se utiliza un core-selection para pasar del modelo de comunicación basada en invocación de métodos – A invoca a select – al modelo de eventos

Principios de Diseño en Componentes Web

Principios de Diseño

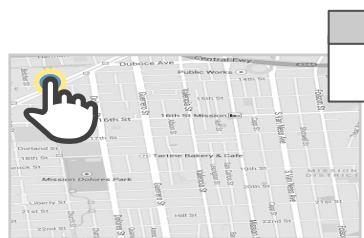
Principio de Adaptabilidad



Adapta Para Transformar

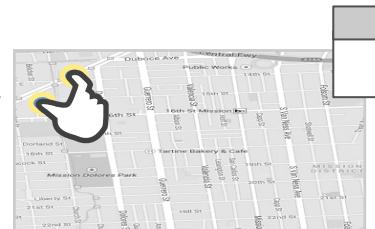
Define adaptaciones para aplicar sobre los componentes de forma automática y permitir que transformen de manera no disruptiva la experiencia de usuario.

- 🎓 Adapta para transformar la experiencia de usuario con un componente de acuerdo a unos objetivos claros y bien definidos.



El componente expone un determinado modelo de eventos que condiciona la experiencia de usuario

Si adaptamos el componente conseguimos que el modelo de eventos cambie y con él la experiencia de usuario



- </> Core-selector y cualquiera de sus hijos son ejemplos de transformación que reciben interacciones de usuario basadas en tap y las convierten en otro evento de negocio.

core-select



core-select

```
<paper-tabs> ..... on-tap="{{fn}}"  
..... <paper-tab>1</paper-tab>  
..... <paper-tab>2</paper-tab>  
..... .....  
</paper-tabs>
```

tap

El componente core-selector y sus descendientes puede percibirse como una transformación adaptativa desde eventos tap a eventos core-selection

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Adaptabilidad



Adapta Para Extender

Define adaptaciones para aplicar sobre los componentes de forma automática y permitir que se enriquezcan con nuevas capacidades o elementos.

- 🎓 Adapta los componentes para conferir más capacidades y elementos que se integren gráclimente en la interacción de usuario.

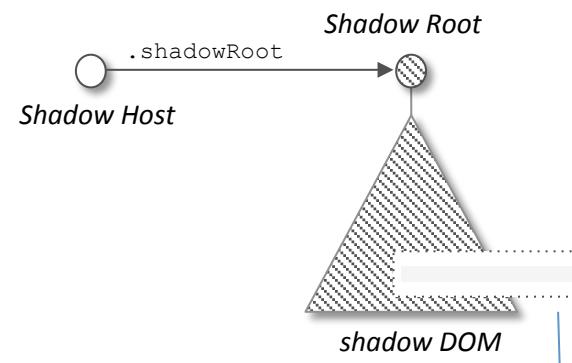


El componente expone contenido determinado sobre el que el usuario puede intervenir

Al adaptar el componente extendemos los contenidos posicionando nuevos elementos que enriquecen la experiencia



- </> Aunque parezca disruptivo con relación al modelo de encapsulación, una manera de hacer extensión adaptativa es por sobreescritura del shadow DOM.



Se inserta un contenido dentro del marcado HTML en la sombra una vez cargado el componente

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Adaptabilidad



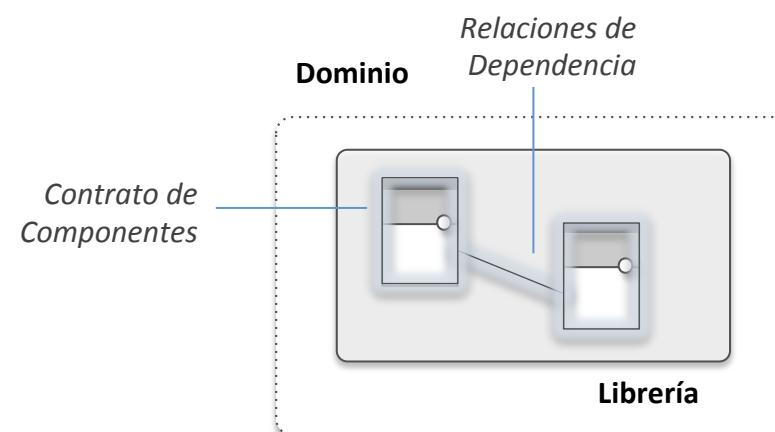
Objetivos

- Fomentar la reutilización
- Mejorar la contextualización
- Simplificar el desarrollo
- Ortogonalizar el dominio
- Abstraer la lógica específica
- Mejorar la evolutividad

Motivación



La adaptación permite transformar los componentes de una arquitectura de cliente para que se adapten a las necesidades particular de este y facilita la compositividad declarativa.



Fases



Análisis & Diseño de Dominio
Diseño de Componentes

Región



Contrato de Componentes
Relaciones de Dependencia

Principios de Diseño en Componentes Web

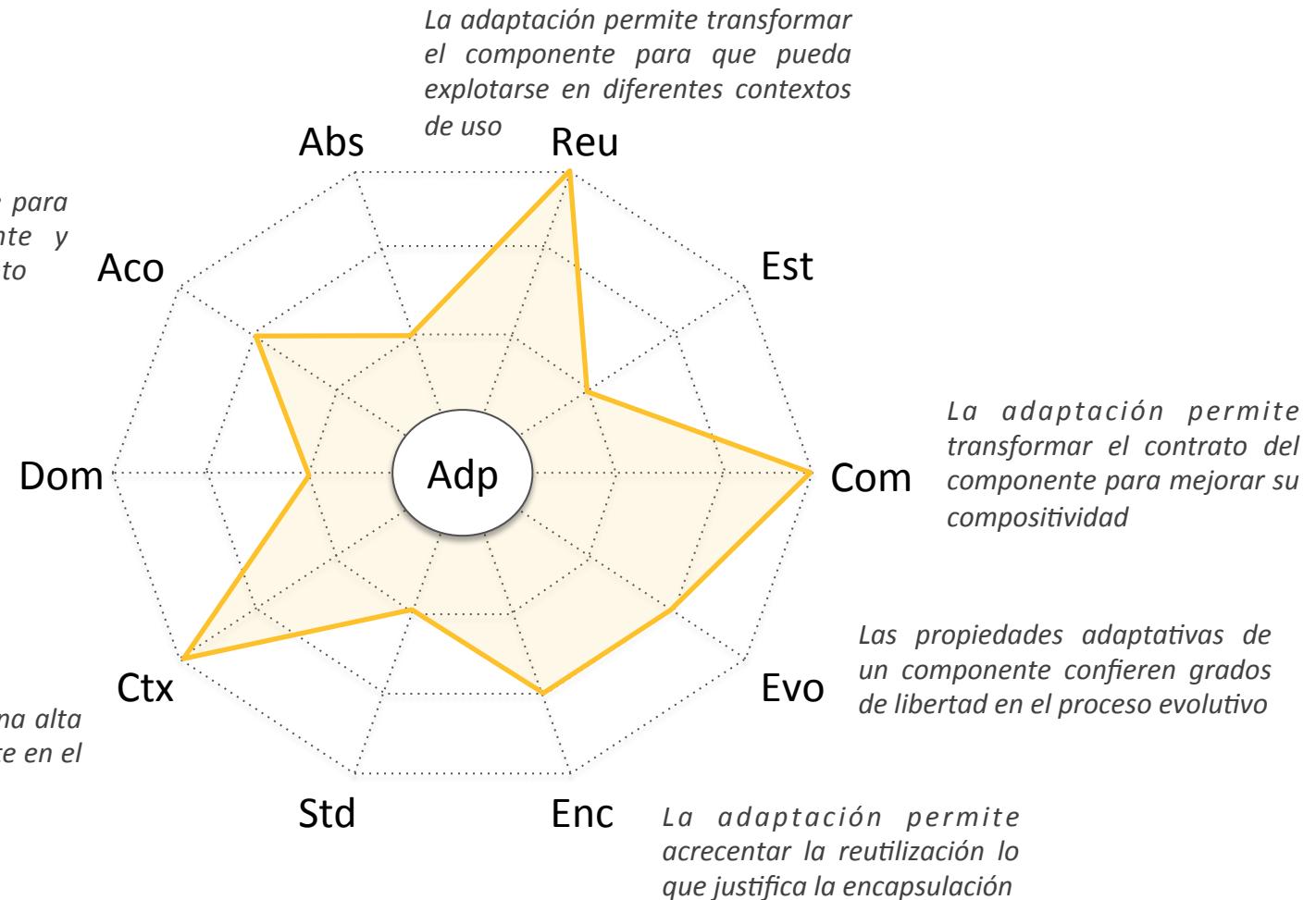
Principios de Diseño

Principio de Adaptabilidad



Tensiones

La adaptación puede usarse para transformar el componente y aliviar el nivel de acoplamiento



Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Adaptabilidad

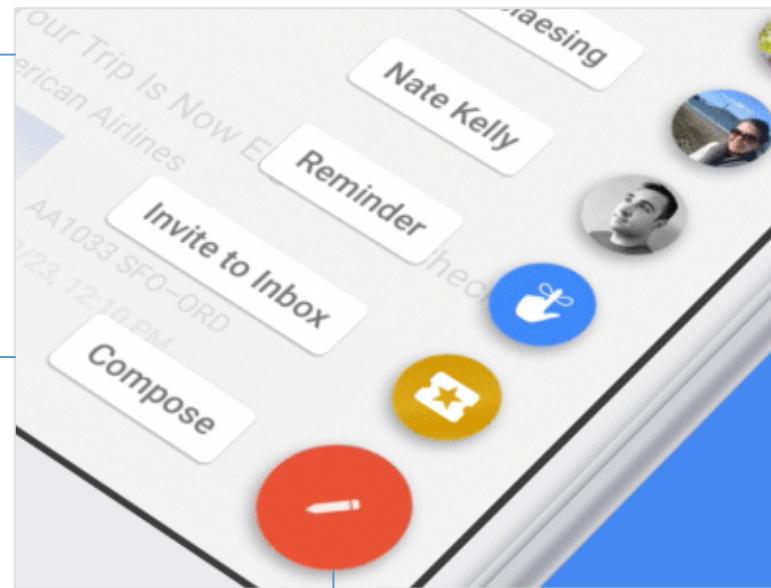


Ejemplo

Inbox Speed Dial Animation

Conocemos que en el contexto material muchos componentes incluyen un fab

En este contexto el fab-menu se *adapta* para incluir etiquetas



Automáticamente se sustituye el fab original por un fab-menu

La adaptación puede aplicarse en sentido inverso para *revertir* los cambios

Es posible *adaptar* un fab para cambiar su comportamiento

Se identifica como una *abstracción* la adaptación a fab-menu

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Evolutividad



Nombre

Evolutividad



Asegúrate de que tus decisiones de diseño y desarrollo mantienen siempre el mayor grado de libertad posible de cara a la evolución de los componentes.



Actores



Arquitecto & Desarrollador de Componentes

Principios de Diseño en Componentes Web

Principios de Diseño

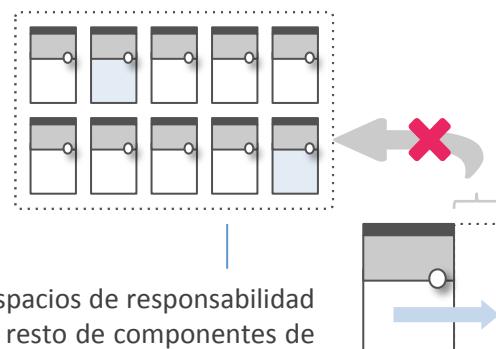
Principio de Evolutividad



Respeto El Dominio

Antes de llevar a cabo cualquier evolución analiza con detalle el dominio para comprobar que las extensiones propuestas no colisionen con otras funcionalidades definidas dentro de él.

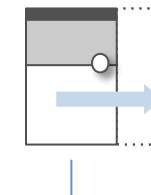
- 🎓 Hacer una adecuada división de responsabilidades del dominio ayuda a simplificar los procesos evolutivos y a garantizar la ausencia de solapamientos funcionales.



Al analizar los espacios de responsabilidad cubiertos por el resto de componentes de la librería se descubren solapamientos funcionales con la extensión que hacen replanificarla

- </> Cada componente constituye un espacio de responsabilidad que debe ser respetado en relación a la adición de nuevos componentes.

paper-tabs



Cualquier proceso de mejora evolutiva en relación al soporte de pestañas en la interfaz de usuario debería ser implementado sobre paper-tabs

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Evolutividad

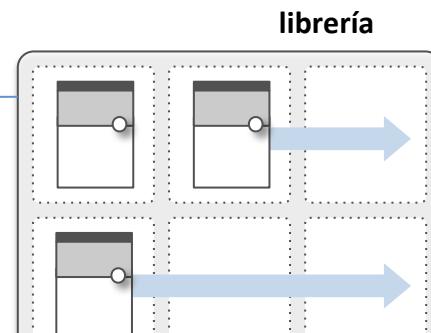


Evoluciona En Anchura

Evoluciona progresivamente el desarrollo de tu librería para dar cobertura en anchura a todas las responsabilidades identificadas durante la fase de análisis.

🎓 La cobertura de las responsabilidades de librería identificadas en la fase de análisis se va implementando de forma progresiva a lo largo del tiempo.

Algunos componentes han cubierto el espacio de responsabilidad al que van asociados



Otros espacios de responsabilidad, aún por implementar, se cubrirán con nuevos componentes que se desarrollarán en versiones futuras de la librería

</> Cualquier librería es susceptible de extenderse en anchura con nuevos componentes que respondan a responsabilidades emergentes.

*google-map-**

google-map
google-map-directions
google-map-marker
google-map-search

En versiones futuras aparecerán potencialmente nuevos componentes dentro de la librería pero éstos en ningún caso deberán solapar con los espacios de responsabilidad ya identificados dentro de la misma

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Evolutividad

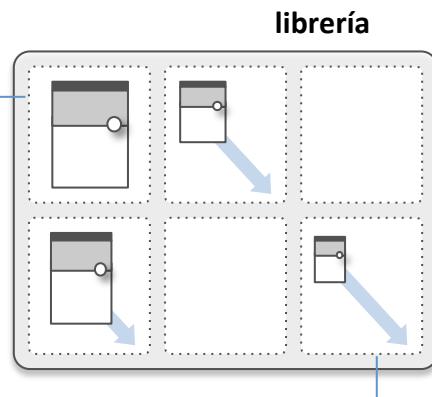


Evoluciona En Profundidad

Evoluciona progresivamente el desarrollo de tu librería para mejorar la cobertura en profundidad de las responsabilidades asignadas en la fase de análisis a cada componente.

- 🎓 La cobertura a las responsabilidades de librería asignadas a cada componente también evoluciona de manera progresiva a lo largo del tiempo.

Algunos componentes ya están cerrados y dan cobertura completa a sus responsabilidades



Cada componente va creciendo evolutivamente en profundidad al dar cobertura a más requisitos dentro de su espacio de responsabilidad

- </> Un ejemplo de desarrollo evolutivo en profundidad corresponde a añadir capacidades de internacionalización a un componente.

```
<acme-login-form>  
</acme-login-form>
```

```
<acme-login-form i18n="es">  
</acme-login-form>
```

La nueva versión del componente permite, opcionalmente, especificar el idioma en que se renderizará el mismo

Principios de Diseño en Componentes Web

Principios de Diseño

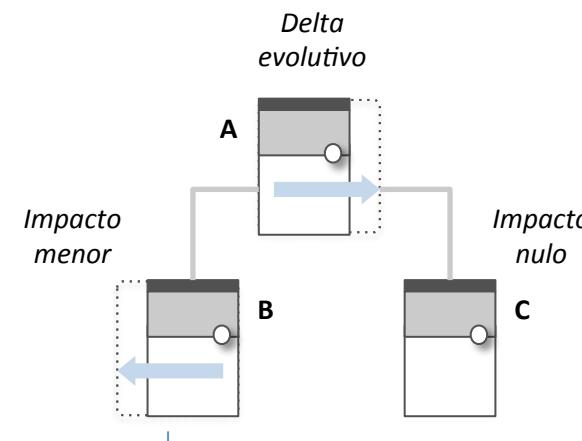
Principio de Evolutividad



Aísla la Evolución

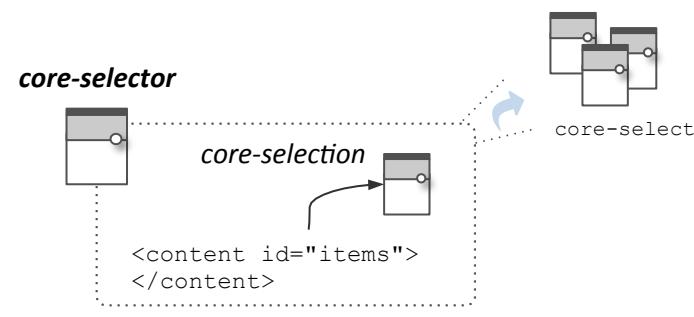
Un buen diseño de Componentes impone que un pequeño cambio en uno de ellos debe impactar poco o nada en los componentes de la vecindad.

🎓 El aislamiento evolutivo confiere ciertos grados de libertad a los desarrolladores de componentes para que sus decisiones no impacten en la arquitectura global.



Los cambios en el componente A impactan poco o nada en los componentes B y C respectivamente

</> Existen multitud de ejemplos en relación al aislamiento evolutivo dado que se trata de una directriz general. Ilustraremos uno de los casos más claros de dependencia.



Core-selector hace uso interno del componente core-selection. Sin embargo, los cambios evolutivos sobre core-selection no deberían impactar en el funcionamiento de core-selector

Principios de Diseño en Componentes Web

Principios de Diseño

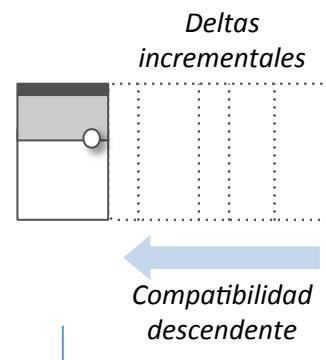
Principio de Evolutividad



Asegura La Compatibilidad Descendente

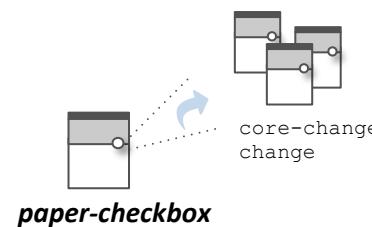
Un cambio evolutivo en un componente no debería ir en contra de la decisiones arquitectónicas tomadas local o globalmente con anterioridad.

- 🎓 La compatibilidad descendente garantiza el mantenimiento del código sobre versiones anteriores y permite construir sobre él nuevo código de forma operativa.



Cada mejora incremental de un componente respeta las decisiones de diseño y desarrollo fijadas en versiones anteriores

- </> Cada vez que desarrollamos componentes estamos tomando decisiones que condicionarán el desarrollo evolutivo de sucesivas versiones.



En paper-checkbox se emite eventos 'change' que resultan diferentes de los habituales 'core-select' de otros componentes. Si esto es algo que tendrá mucho o poco impacto en el desarrollo de componentes es pronto para decirlo pero en cualquier caso es una decisión que deberá respetarse por compatibilidad descendente

Principios de Diseño en Componentes Web

Principios de Diseño

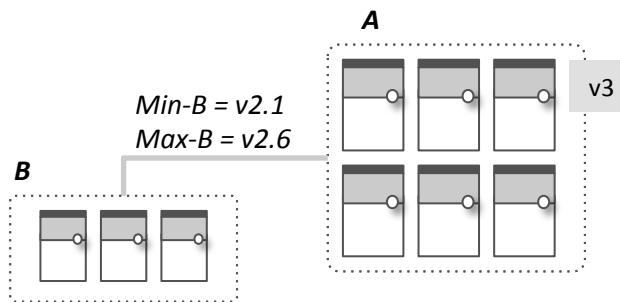
Principio de Evolutividad



Establece Políticas de Versionado

Para mantener la coherencia evolutiva de los componentes es preciso definir políticas de versionado a nivel de librería que orienten los criterios de compatibilidad con otras librerías.

🎓 Como parte de los metadatos que caracterizan a los componentes deberían incluirse rangos de versionado mínimo y máximo que garantizan la interoperabilidad.



La librería A mantiene una dependencia con la B pero dentro del rango de versiones de B entre 2.1 y 2.6

</> Los cambios sobre la propia plataforma y framework Polymer imponen restricciones sobre el versionado de los componentes en core-* y paper-*.

*core-**

core-selection
core-selector
core-pages
core-menu ...

0.5

*paper-**

paper-button
paper-fab
paper-input
Paper-label

0.5

Ambas librerías han sido desarrolladas y diseñadas bajo los parámetros de funcionamiento de la versión 0.5 de Polymer

Principios de Diseño en Componentes Web

Principios de Diseño

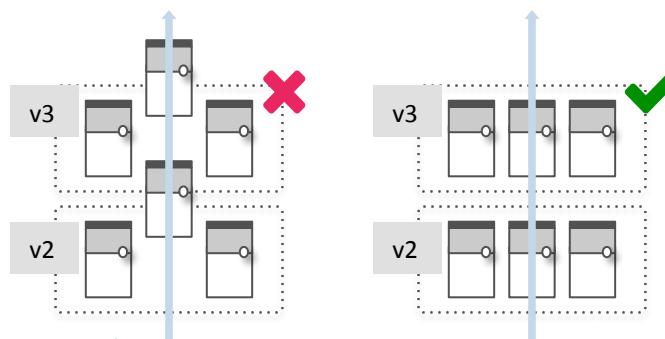
Principio de Evolutividad



Alinéate Con La Versión

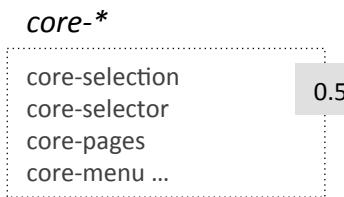
Los cambios en cualquier componente deberían estar en cualquier caso sometidos a los estándares evolutivos definidos en el marco de una versión de librería.

🎓 En los procesos de desarrollo evolutivo evita tomar decisiones de naturaleza disruptiva en relación a los estándares de librería.



Ningún componente debería encontrarse desalineado con las directrices de diseño y desarrollo establecidas en el marco de la versión de la librería a la que pertenece

</> Cada versión de librería expone implícita o explícitamente ciertas directrices y estándares de diseño y desarrollo para los componentes que contiene.



El uso del componente `<polymer-element>` o la construcción por herencia de componentes son dos convenios no soportados en la 0.8

Principios de Diseño en Componentes Web

Principios de Diseño

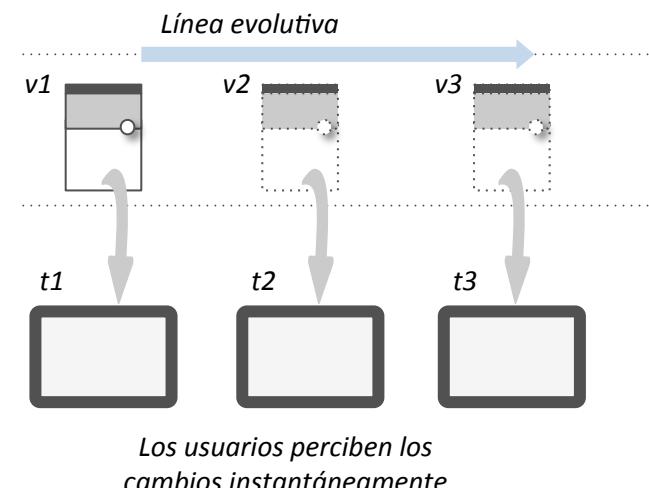
Principio de Evolutividad



Practica El Despliegue Automático

Los cambios evolutivos de cada componente Web deberían ser entregados y desplegados de manera automática y transparente al usuario final para mejorar su experiencia.

- 🎓 El usuario debería percibir de manera automática las mejoras evolutivas que se van desarrollando a lo largo del tiempo sin necesidad de realizar actualizaciones.



- </> Tomemos como ejemplo el componente de google-map. La responsabilidad de evolucionar este componente descansa en Google.

google-map



Si en nuestra aplicación utilizamos el componente google-map lo lógico es que lo importemos directamente de los servidores de google y no de una copia en nuestro propio servidor. De esta manera la gestión de cambios se pueden delegar en el propio Google

Principios de Diseño en Componentes Web

Principios de Diseño

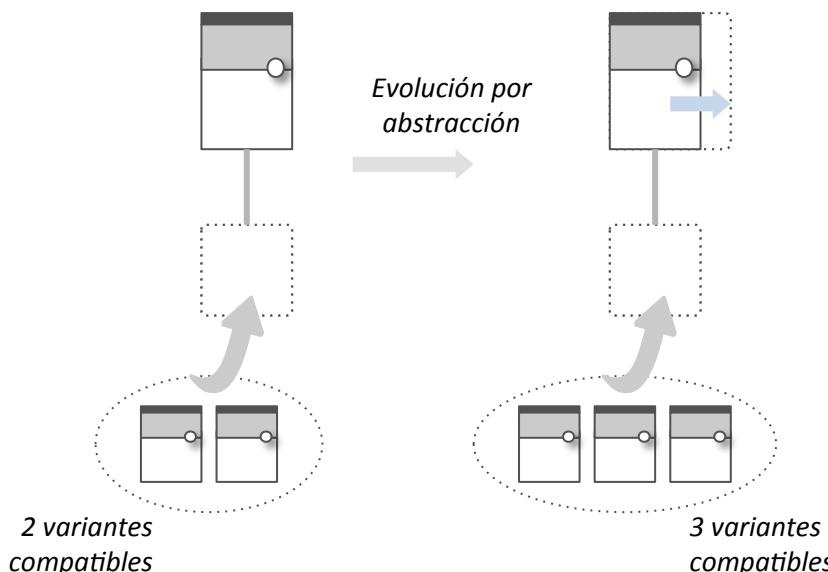
Principio de Evolutividad



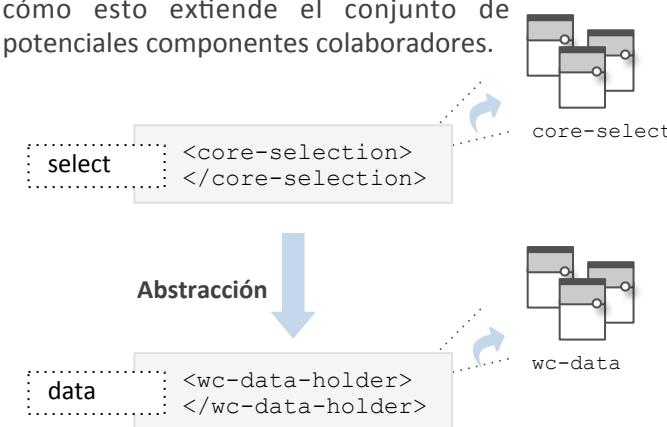
Evoluciona Por Abstracción

Evoluciona a través de la abstracción cuando adviertas una necesidad de refactorización que deba dar cobertura a nuevas variantes presentes en el dominio.

🎓 A veces una actividad evolutiva tiene por objeto permitir a nuevas variantes operar con un componente. Para ello se debe subir el nivel de abstracción.



</> Retomemos un ejemplo de abstracción que pusimos anteriormente para señalar cómo esto extiende el conjunto de potenciales componentes colaboradores.



Sobre el nivel de abstracción de data-holder es posible que un mayor número de componentes operen con él

Principios de Diseño en Componentes Web

Principios de Diseño

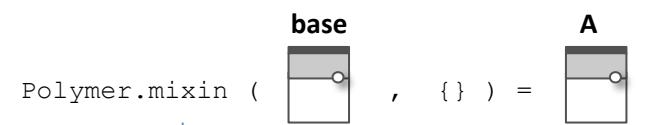
Principio de Evolutividad



Evoluciona Por Composición

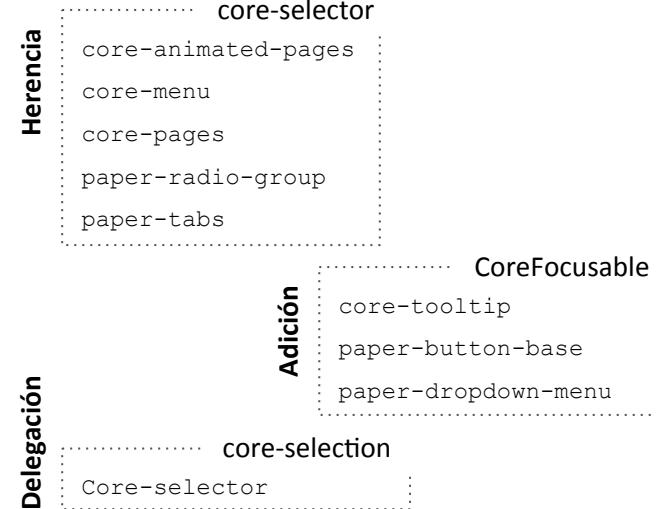
Para desarrollar actividades evolutivas de un componente utiliza técnicas compositivas como la adición, la delegación, la herencia o la agregación.

🎓 El objetivo de llevar a cabo la evolución por medio de la composición es garantizar que el código común permanece centralizado en otros componentes.



Distintas técnicas compositivas se aplican para articular actividades evolutivas en las que la base contiene el código común mientras que los cambios residen en el nuevo componente A

</> Los componentes de las librerías core-* y paper-* hacen uso frecuente de la adición y la herencia y en mucha menor medida de la delegación.



Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Evolutividad



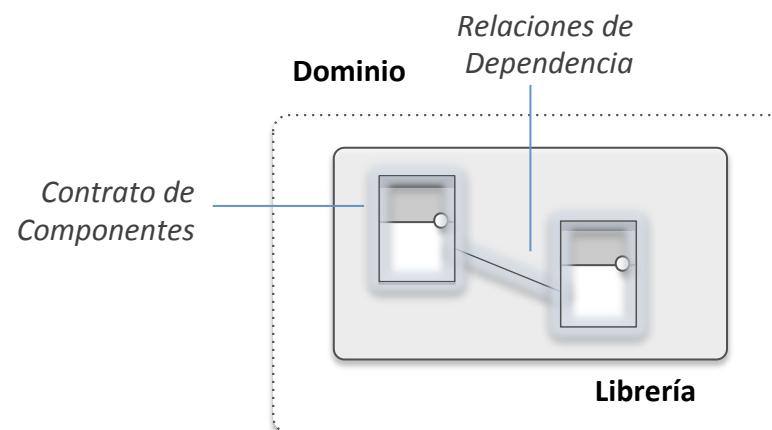
Objetivos

- Fomentar la autonomía
- Aumentar el tiempo de vida
- Evitar la replicación
- Evitar la deprecación
- Evitar el versionado
- Asegurar la compatibilidad
- Facilitar la mantenibilidad

Motivación



El principio de evolución dirige los procesos de mantenimiento de una librería de componentes para garantizar que éstos se mantengan como artefactos sólidos dentro del dominio.



Fases



Diseño Componentes
Mantenimiento Evolutivo

Región



Contrato de Componentes
Relaciones de Dependencia

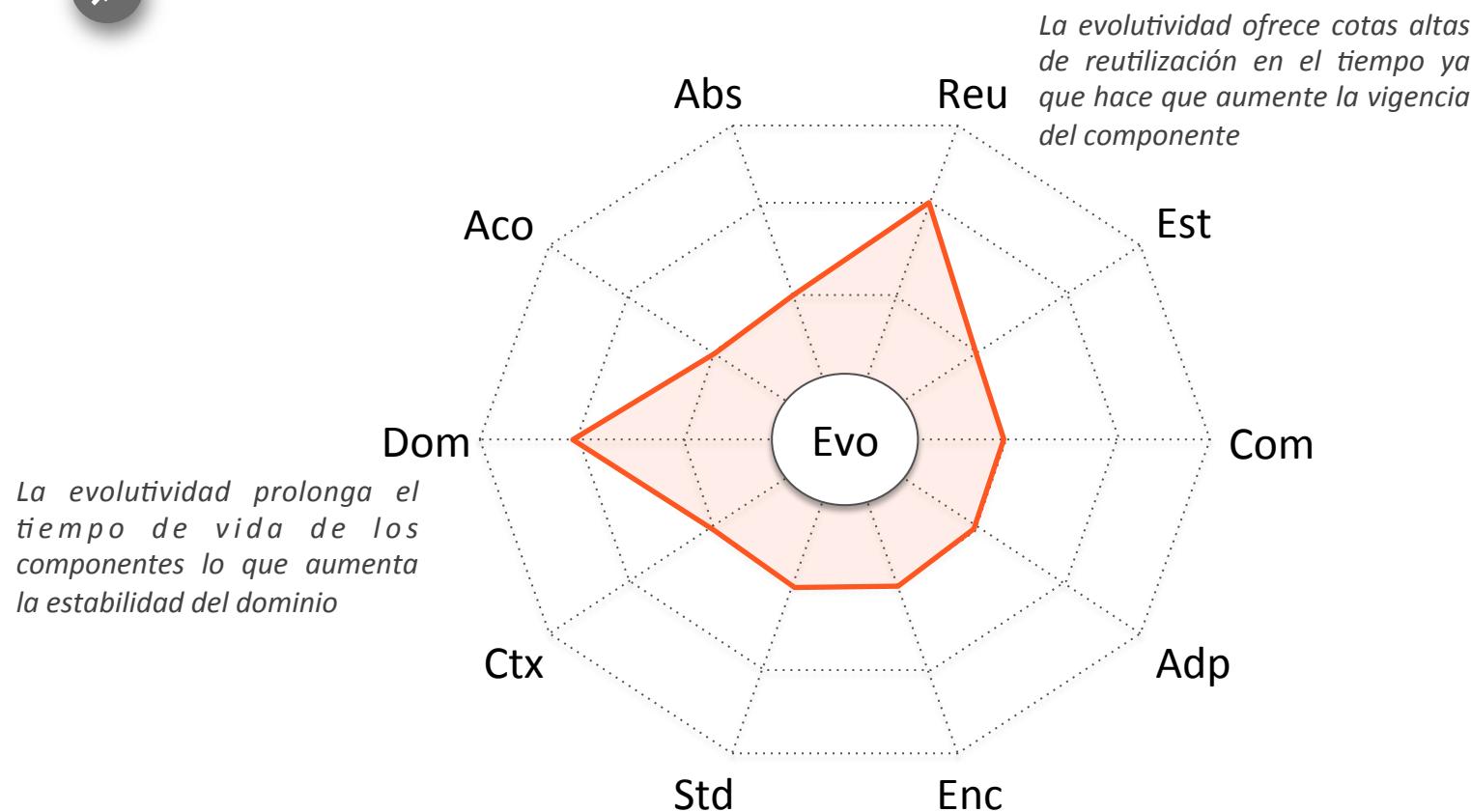
Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Evolutividad



Tensiones



Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Evolutividad

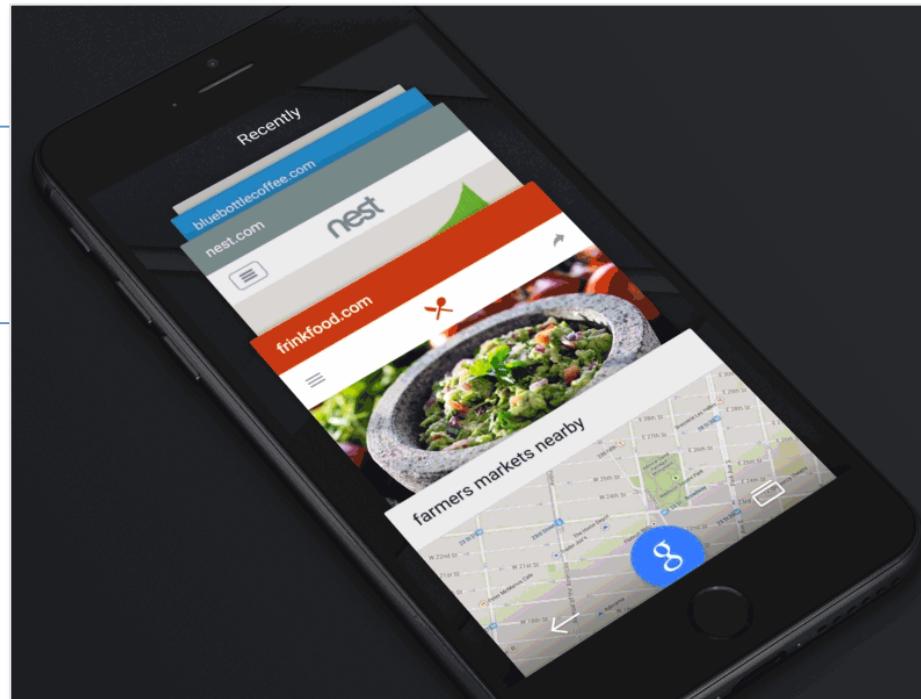


Ejemplo

Recents in the Google App

La autonomía de las tabs mantiene la continuidad

En su esencia funcional las tabs mantienen la compatibilidad descendente



La evolución mantiene el carácter compositivo lo que promueve la escalabilidad

Esta forma de organizar las pestañas corresponde a la versión Lollipop

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Encapsulación



Nombre

Encapsulación



Encapsula sólo cuando identifiques una colaboración entre componentes como un esquema de alta reutilización potencial en distintos contextos de uso.



Actores



Analista de Dominio

Arquitecto & Desarrollador de Componentes

Maquetador Web

Principios de Diseño en Componentes Web

Principios de Diseño

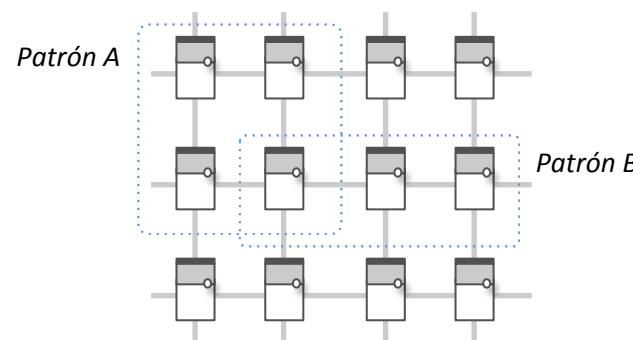
Principio de Encapsulación



Busca Patrones Recurrentes

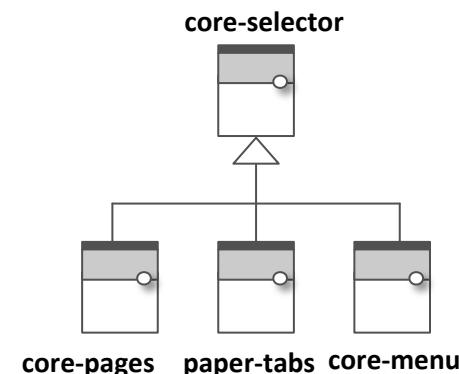
Recuerda que el objetivo de la encapsulación consiste en identificar nuevos componentes de dominio a partir de composiciones recurrentes en distintos contextos de uso.

🎓 Adquirir soltura con la identificación de patrones recurrentes es un reto para arquitectos y analistas que se mitiga con la experiencia de dominio.



El análisis de diversas arquitecturas de cliente basadas en componentes permite descubrir patrones de reutilización potencial

</> Como ya hemos demostrado en numerosos ejemplos a lo largo de este texto, el componente core-selector es uno de los patrones de mayor recurrencia.



Core-selector representa un patrón entre componentes que acomoda un evento de usuario a un evento de negocio

Principios de Diseño en Componentes Web

Principios de Diseño

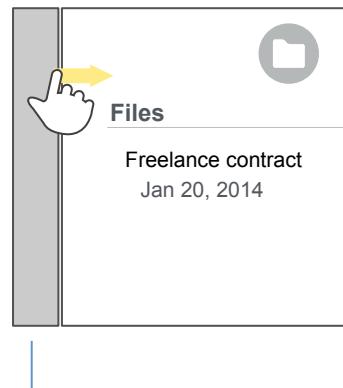
Principio de Encapsulación



Captura La Interacción

Se consciente de que el propósito de un componente Web frecuentemente es capturar modelos de interacción en relación con la experiencia de usuario.

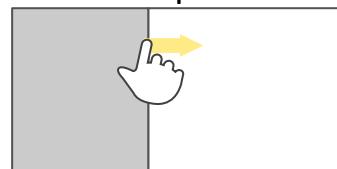
Los componentes Web ofrecen una oportunidad única de capturar en artefactos computables elementos de la experiencia de usuario.



No se trata tanto de encapsular una cierta combinación de componentes colaborando entre sí sino de capturar una experiencia prototípica y nuclear de usuario

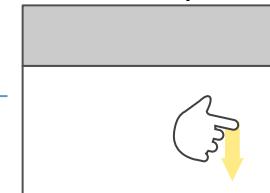
</> Dos ejemplos muy buenos de captura de la interacción los encontramos en los componentes de layout core-header-panel y core-drawer-panel.

core-drawer-panel



Core-drawer-panel gestiona un área lateral deslizable para incluir contenido vinculado al área principal

core-header-panel



Core-header-panel permite incluir una cabecera fija o móvil en relación a un área inferior que se gestiona por scrolling

En ambos casos se da soporte a un modelo de interacción con independencia de los contenidos específicos que completen el layout

Principios de Diseño en Componentes Web

Principios de Diseño

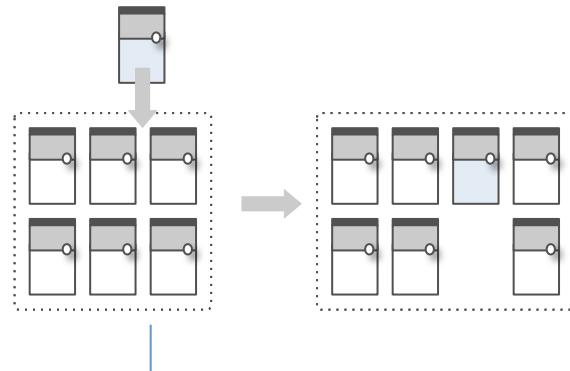
Principio de Encapsulación



Acomoda El Dominio

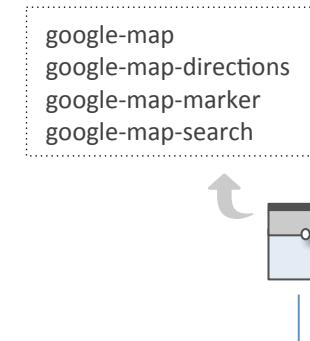
Cada nueva encapsulación debe encontrar su posición exacta dentro del dominio antes de materializarse como un nuevo componente. Esto puede requerir el acomodo del mismo.

🎓 Un análisis preliminar del alcance de librería junto a una adecuada división de responsabilidades mitiga este tipo de actividades.



Encontrar una nueva encapsulación susceptible de incorporarse a dominio puede requerir un reajuste en la división de responsabilidades

</> Pensemos en la librería de google-map dedicada a concentrar todas las capacidades de geolocalización y visualización cartográfica de Google.



Si se construye por encapsulación un nuevo componente relacionado con esta librería sería necesario acomodarla para garantizar la ausencia de solapamientos funcionales

Principios de Diseño en Componentes Web

Principios de Diseño

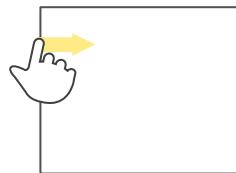
Principio de Encapsulación



Persigue La Cohesión

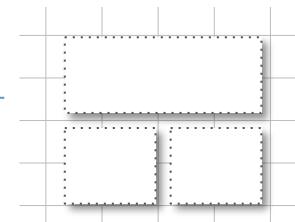
Al encapsular asegúrate que los elementos internos mantienen cierta cohesión interactiva y visual como para justificar la creación de un nuevo componente.

🎓 En UX, la cohesión se traduce en similitud visual e interactiva. Al encapsular comprueba que los componentes internos mantienen dicha similitud.



Garantiza durante la encapsulación que los agregados responderán a un mismo modelo de interacción

Garantiza igualmente homogeneidad en los criterios de visualización y gestión del espacio



</> En la representación adjunta mostramos un claro contraejemplo que va en contra de la cohesión visual.

La barra de búsqueda tal y como se presenta resulta visualmente disruptiva en relación a los demás componentes del entorno

Location



From

Fri, Feb 10 2014 ▾

4:30 PM ▾

To

Fri, Feb 10 2014 ▾

5:30 PM ▾

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Encapsulación

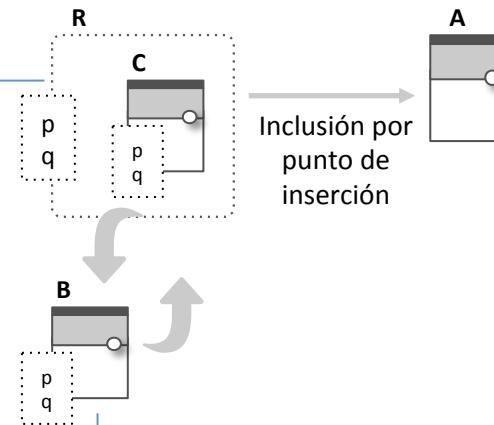


Persigue La Sustitutividad

Cada encapsulación debería definir el contrato con el mayor grado de abstracción posible sobre sus elementos de configuración para poder articular sustituciones polimórficas.

- Los elementos del Light DOM de cada componente deberían ser a su vez componentes Web que operan a nivel abstracto.

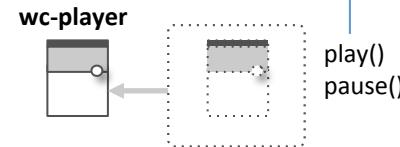
El componente A exige el cumplimiento de un contrato parcial con métodos p y q a todo aquel que ocupe el rol R



B y C cumplen ambos con el contrato parcial exigido por A en el rol R con lo que pueden ser sustitutos sintácticos el uno del otro

- Consideremos como ejemplo un componente reproductor que recibe como parámetro una lista de reproducción.

Los elementos de la lista de reproducción pueden ser de cualquier tipo pero deben implementar los métodos play y pause



Las etiquetas video y audio cumplen ambas con el contrato parcial exigido por wc-player

```
<wc-player>
  <video> ... </video>
  <audio> ... </audio>
  <audio> ... </audio>
  <video> ... </video>
</wc-player>
```

Principios de Diseño en Componentes Web

Principios de Diseño

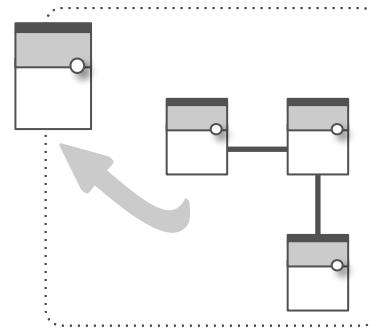
Principio de Encapsulación



Encapsula Por Abstracción

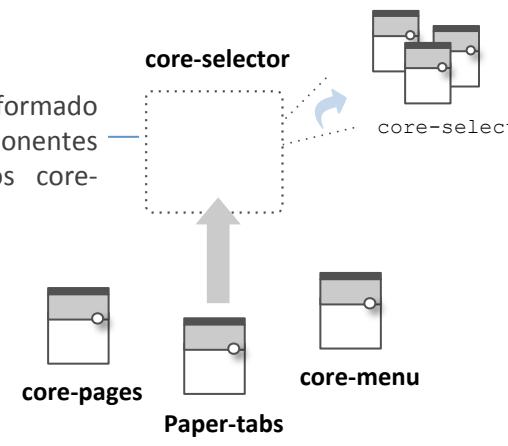
Todo ejercicio de encapsulación sucede como consecuencia de un proceso de abstracción en el que se ocultan los detalles de los componentes internos.

🎓 Cuando encapsulas encuentra el esquema de composición más abstracto para fomentar la oportunidades de reutilización potencial del nuevo componente.



El contrato externo del componente oculta los detalles de implementación interna ofreciendo al usuario un comportamiento nuevo

</> Al expresar el contrato con alto nivel de abstracción este resulta altamente reubicable en diferentes contextos de uso. Pensemos en core-selector.



El contexto de uso está formado por una colección de componentes que reaccionan a eventos core-select

Son varios los componentes que se ajustan al contrato definido en core-selector

Principios de Diseño en Componentes Web

Principios de Diseño

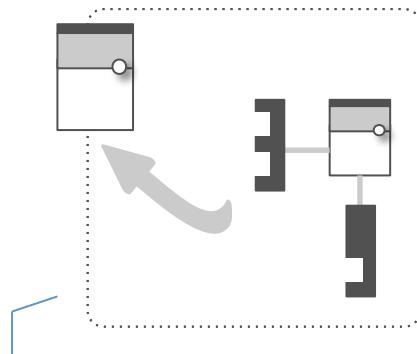
Principio de Encapsulación



Encapsula Por Adaptación

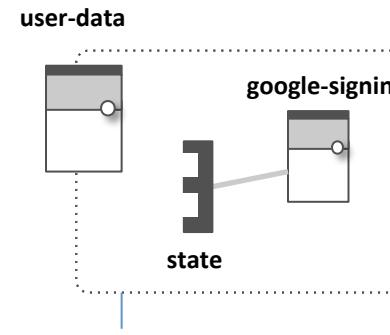
Muchas encapsulaciones están vinculadas a procesos específicos de adaptación de colaboraciones que se han convertido en esquemas de aplicabilidad recurrente.

🎓 La encapsulación adaptativa responde a composiciones entre componentes y adaptaciones abstractas para dar lugar a componentes más específicos.



El contrato externo del componente constituye una encapsulación de una colaboración adaptada entre componentes

</> Pensemos como ejemplo en el componente resultante de encapsular google-signin compuesto con un componente para gestionar los datos de usuario.



Los datos de usuario se gestionan en esta nueva encapsulación que cachea dichos datos y hace uso discrecional del servicio de autenticación de google a través del componente google-signin

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Encapsulación

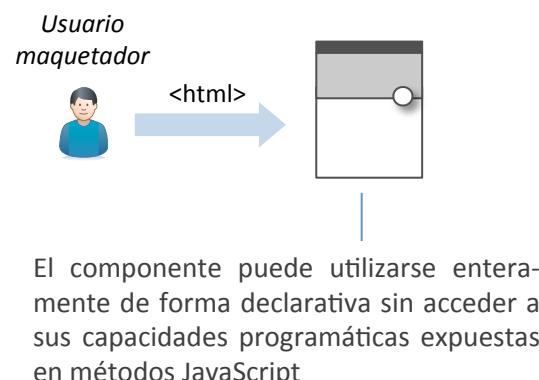


Define Contratos Declarativos

Diseña los contratos de cada componente fomentando su carácter declarativo. Ello significa poder utilizarlos como etiquetas de autor usando sólo marcado HTML.

🎓 No olvides que la tecnología de componentes Web va dirigida a crear nuevo léxico de autor que se incorpora al estándar HTML.

</> La etiqueta estándar `<video>` es en realidad un componente Web. Este es un buen ejemplo de uso declarativo de componentes.



La etiqueta `video` expone un reproductor de video sin necesidad de acceder a sus capacidades programáticas

```
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
  No existe soporte a video.
</video>
```

De una manera completamente declarativa se configura el reproductor de video para su correcto funcionamiento

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Encapsulación

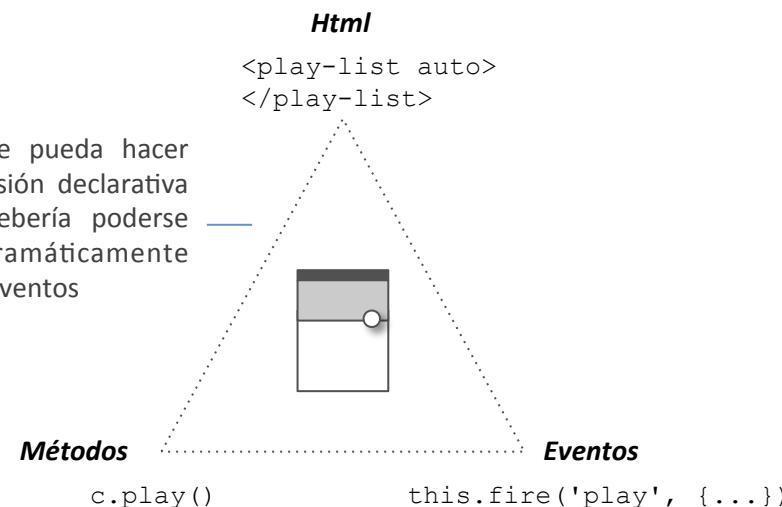


Alinea El Contrato

El contrato de un componente conjuga una vista, una capacidad funcional y una respuesta reactiva. Asegúrate de alinear estos tres perfiles.

- 🎓 El contrato del componente incluye un modelo de vista, eventos y capacidades. Asegúrate de que estas dimensiones se encuentran alineadas en prestaciones.

Todo lo que se pueda hacer desde la dimensión declarativa del contrato debería poderse articular programáticamente con métodos y eventos



- </> La etiqueta estándar <audio> también es un componente Web. Sus capacidades se exponen a través de los tres perfiles de contrato.



Principios de Diseño en Componentes Web

Principios de Diseño

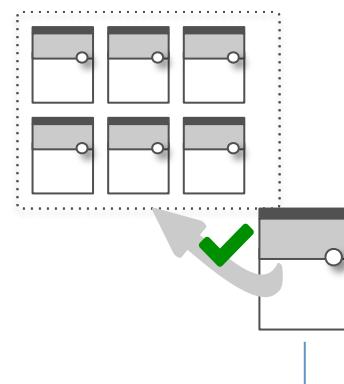
Principio de Encapsulación



Comprueba El Valor Diferencial

Antes de contribuir a dominio un nuevo componente asegúrate de que éste aporta un valor diferencial en el marco de dicho dominio.

- 🎓 La comprobación del valor diferencial garantiza el mantenimiento de la ausencia de solapamientos funcionales dentro del dominio.



El componente es una nueva contribución en el dominio que aporta un valor diferencial en relación al alcance y responsabilidades previstas en la librería

- </> Tomemos en consideración nuevamente como ejemplo la librería de google dedicada a la visualización cartográfica google-map-*.

google-map
google-map-directions
google-map-marker
google-map-search

Cada componente dentro de la librería aporta un valor diferencial con respecto al resto de componentes de la misma

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Encapsulación



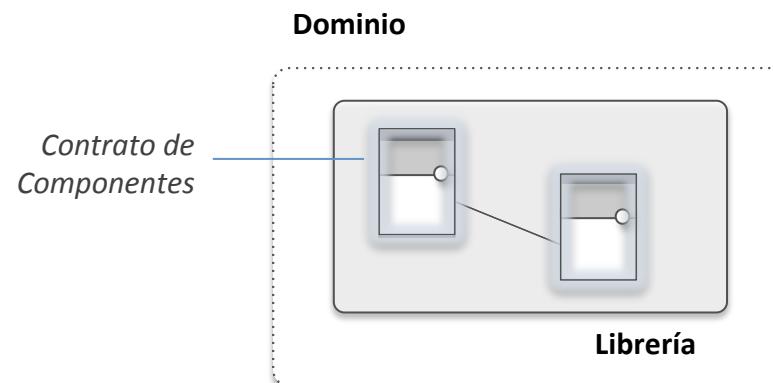
Objetivos

- Fomentar la reutilización
- Ocultar la implementación
- Simplificar el uso
- Disminuir el conocimiento técnico
- Ampliar el vocabulario de negocio
- Canonizar la adaptación
- Reacomodar y actualizar el dominio

Motivación



La encapsulación fomenta la creación por descubrimiento de nuevos componentes y permite garantizar que éstos respetan los criterios de diseño bajo los cuales se construyeron previamente los demás componentes.



Fases



Análisis de Dominio
Diseño de Componentes

Región



Contrato de Componentes

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Encapsulación



Tensiones

La encapsulación de componentes permite ocultar los desacoplamientos internos para que no sean percibidos desde el exterior

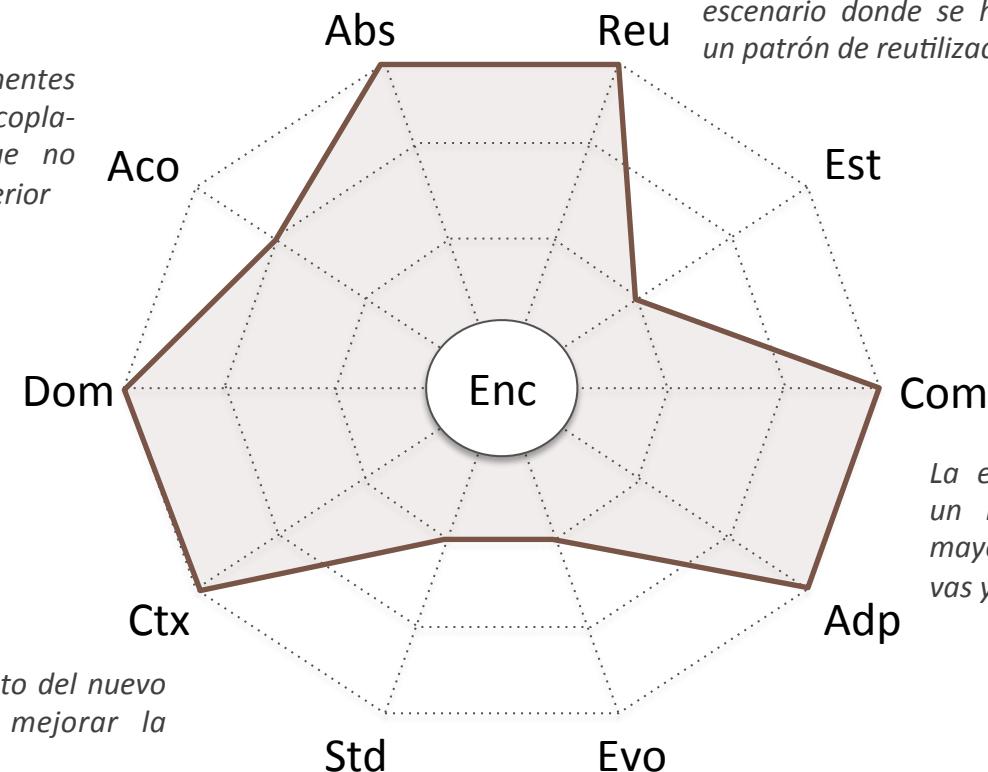
La encapsulación oculta los detalles del interior lo que se traduce en mayor abstracción

La encapsulación responde a un escenario donde se ha identificado un patrón de reutilización potencial

La inclusión de un nuevo componente en el dominio puede provocar disruptiones en la organización del mismo

La encapsulación permite ofrecer un nuevo contrato que ofrezca mayores posibilidades compositivas y adaptativas

La definición del contrato del nuevo componente permite mejorar la contextualización



Principios de Diseño en Componentes Web

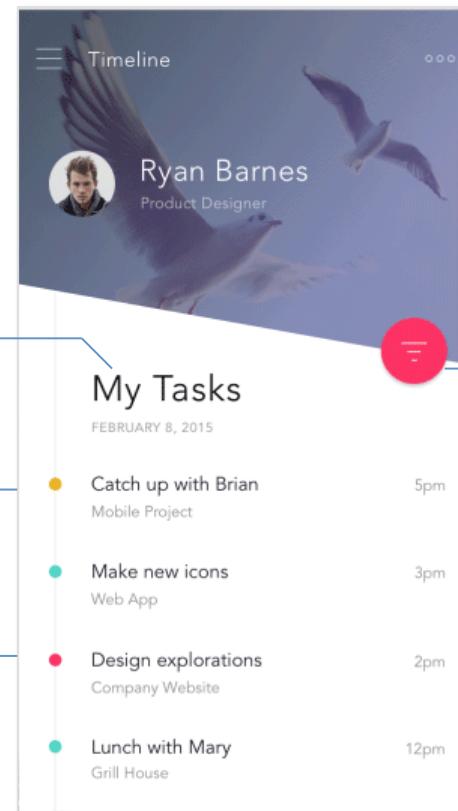
Principios de Diseño

Principio de Encapsulación



Ejemplo

Filter menu



Se encapsula por *abstracción* con
indecencia del tipo de elementos con
lo que se permite la *sustitutividad*

El timeline es un
patrón recurrente de
diseño

Los elementos de la colección
están homogéneamente
cohesionados

Se captura un modelo
de *interacción*

El componente se
configura y usa de
forma *declarativa*

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Estandarización



Nombre

Estandarización



Durante las actividades de diseño y desarrollo de componentes establece y sigue estándares y convenciones que ayuden a homogeneizar el proceso compositivo.



Actores



Arquitecto & Desarrollador de Componentes

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Estandarización

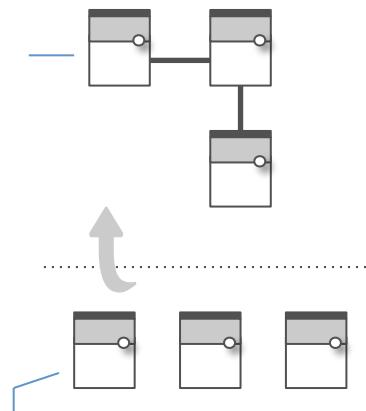


Usa Componentes Estándar

Durante el desarrollo aparece frecuentemente la necesidad de hacer uso de componentes prototípicos y estándar que resuelvan ciertos problemas de carácter transversal.

Por norma general, este tipo de componentes responden a patrones de diseño que dan respuesta a necesidades de aplicación recurrente.

Tu solución hace uso frecuente de componentes estándar de aplicación prototípica



Componentes estándar no son solamente los construidos por Polymer en core-* y paper-* sino todos aquellos que desarrolles para resolver problemas recurrentes

</> Un buen ejemplo de componente estándar por su gran aplicabilidad práctica en cualquier tipo de solución es la etiqueta core-ajax.

```
<core-ajax  
    auto  
    url="..."  
    params="..."  
    handleAs="json">{{hn}}>  
</core-ajax>
```

El componente core-ajax hace solicitudes en background al servidor de manera automática y entrega la respuesta para que la gestione la función hn

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Estandarización

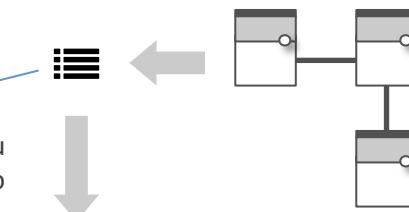


Diseña por Mimetismo

En la medida de lo posible, para construir tus componentes insírstate en las decisiones arquitectónicas y de diseño tomadas por terceros para construir otros componentes del entorno.

🎓 El objetivo del diseño por mimetismo es conseguir comportamientos y estilos arquitectónicos con un nivel de homogeneidad elevado.

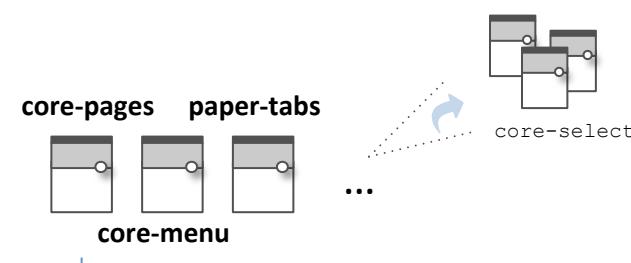
Los componentes de tu solución imponen un estilo arquitectónico de facto



Tus componentes siguen ese estilo como si fueran unos criterios de diseño establecidos explícitamente



</> Pensemos en los componentes de las librerías de Polymer core-* y paper-* que operan como proveedores de selección.



Por diseño cada proveedor de selección emite eventos core-select

Si tu componente también es un proveedor de selección entonces debería igualmente adscribirse a ese criterio arquitectónico en busca de la homogeneidad



Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Estandarización



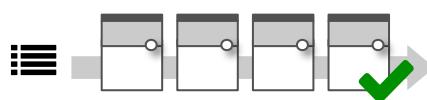
Evita Convenios De Componente

En la medida de lo posible, evita hacer uso de convenios que se apliquen a un único componente. Ello dificulta su uso y exigirá mayores esfuerzos de documentación.

- 🎓 Siempre es preferible diseñar de acuerdo a criterios y convenciones estándar de alta aplicación que a decisiones exclusivas que atañen a un único componente.



No es una buena idea definir criterios de diseño que se aplican a un único componente



Lo adecuado es que los criterios de diseño se aplican a más de un componente

- </> Como contrapunto podemos señalar las divergencias de diseño que presentan algunos componentes de la librería core-* que no responden a motivación alguna.

En core-drawer-panel los roles se identifican mediante atributos semánticos HTML

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

```
<core-header-panel>
  <div class="core-header">...</div>
  <div>...</div>
</core-header-panel>
```

Sin embargo, en core-header-panel lo mismo se hace mediante atributos de estilo class

Principios de Diseño en Componentes Web

Principios de Diseño

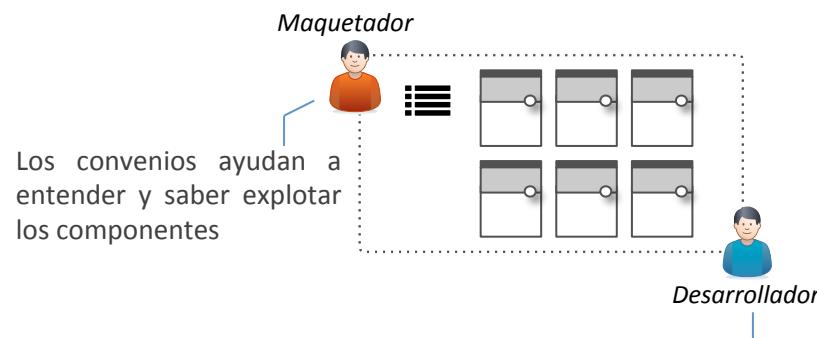
Principio de Estandarización



Fomenta Convenios De Librería

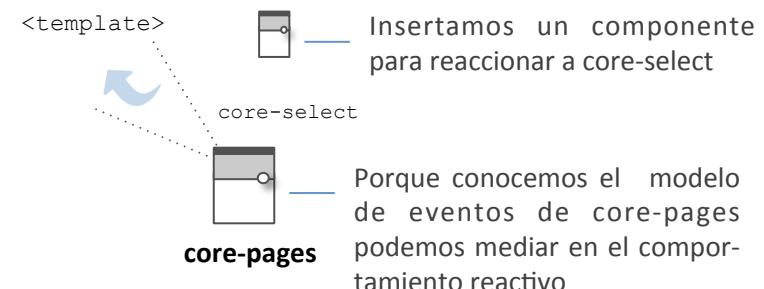
Cuando diseñas una librería establece los convenios necesarios y suficientes para dar respuesta a los problemas de interoperabilidad en el ámbito de tu dominio.

🎓 Los convenios de librería ofrecen directrices acerca de cómo se usan y comportan los componentes dentro de la misma.



Los convenios también ofrecen directrices para construir nuevos componentes a partir de los elementos de la librería

</> El conocimiento que tenemos de la operativa y comportamiento de un componente nos permite articular colaboraciones sobre él.



Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Estandarización

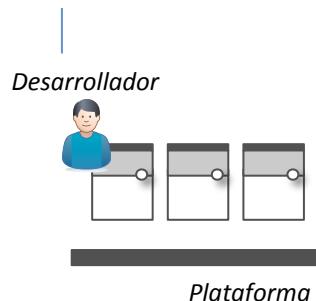


Alinéate Con La Tecnología

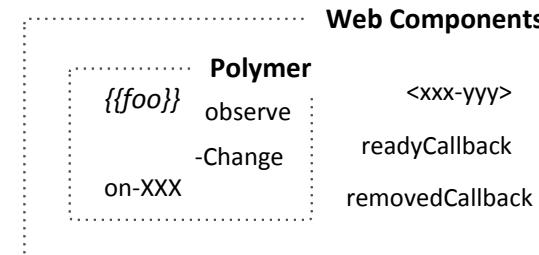
El desarrollo de componentes bajo un framework o plataforma determinado impone el uso de ciertas conceptualizaciones y convenios establecidos.

- 🎓 El desarrollador debe adscribirse a las convenciones impuestas por la tecnología bajo la que se desarrolla en aras a fomentar el código homogéneo.

La plataforma establece criterios y convenciones de diseño y nombrado que el desarrollador debe tener en cuenta y respetar



- </> Polymer es un framework basado en convención frente a configuración. Ponemos a continuación sólo algunos ejemplos de estas convenciones.



Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Estandarización

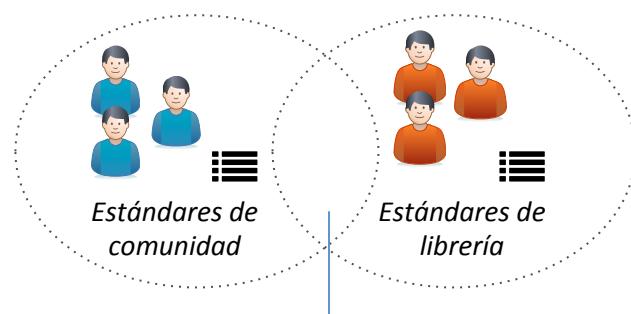


Alinéate Con La Comunidad

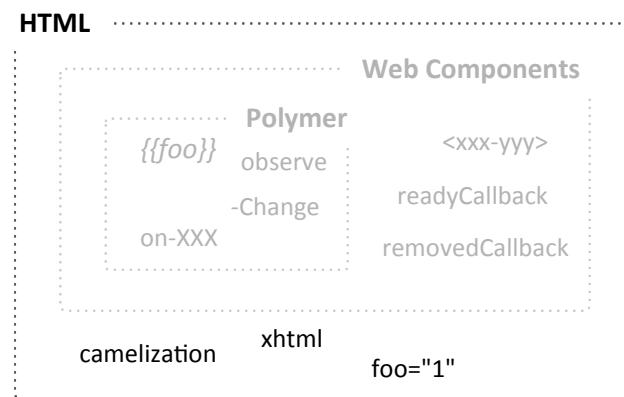
No olvides que el desarrollo de componentes Web se circumscribe a prácticas y convenciones establecidas en el ámbito de una comunidad tecnológica.

🎓 Las convenciones de desarrollo siempre se circunscriben a las decisiones tomadas por comités, grupos de trabajo, etc. que dan voz a la comunidad.

</> El desarrollo de componente Web está inmerso en la comunidad de HTML5 con lo que sus estándares son de aplicación en componentes Web.



En cualquier caso las convenciones y estándares de comunidad siempre deben prevalecer sobre aquellas que tengan un ámbito de alcance menor



Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Estandarización



Estandariza por Abstracción

Cuando definas convenios y estándares debes hacerlo manteniendo un adecuado nivel de abstracción como para que maximices sus posibilidades de aplicabilidad transversal.

🎓 La abstracción es un paso esencial en la definición de convenciones y estándares para conseguir un nivel de aplicabilidad transversal elevado.

Durante el desarrollo descubro nuevos criterios de diseño



Mediante un proceso de abstracción generalizo los criterios para convertirlos en convenios estándar



El carácter estándar de los criterios permite que éstos se apliquen a otros componentes



</> Tomemos como ejemplo los criterios de diseño y desarrollo utilizados en cualquier componente. Estos deberían aplicarse extensivamente al resto de componentes.

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



Los roles en la configuración de un componente deberán establecerse mediante atributos semánticos de HTML

Principios de Diseño en Componentes Web

Principios de Diseño

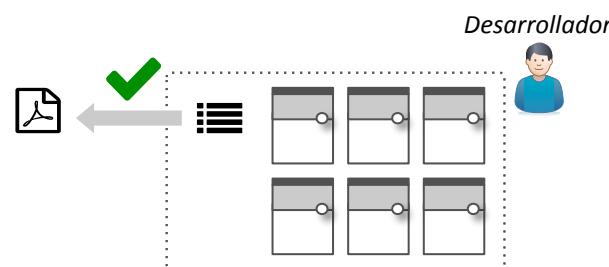
Principio de Estandarización



Documenta Los Convenios

Los convenios que establezcas con carácter personal en el ámbito de tu librería deben ser documentados adecuadamente para dar soporte a terceros contribuidores.

🎓 Esta es típica documentación susceptible de aparecer en la Wiki de Github y su lectura es importante antes de hacer cualquier contribución.

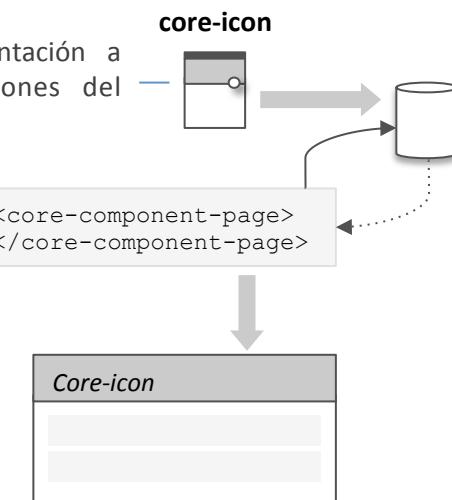


Tan importante es diseñar de acuerdo a estándares como saber transmitirlos y documentarlos apropiadamente

</> Polymer utiliza un conjunto de anotaciones en comentarios sobre el código de los componentes que permite desarrollar páginas de documentación.

Se genera la documentación a partir de las anotaciones del código

Este componente recupera la documentación del servidor y la presenta



Principios de Diseño en Componentes Web

Principios de Diseño

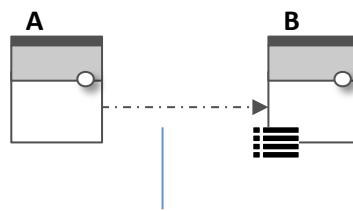
Principio de Estandarización



Persigue La Introspección

En la medida de lo posible aplica estándares y convenciones que permitan a los componentes descubrir dinámicamente las características y comportamiento de los demás componentes.

- 🎓 El uso de estándares y convenciones de alta aplicabilidad transversal permite descubrir de manera dinámica las características de un componente.



A asume que B se ha diseñado siguiendo convenios y estándares establecidos. Esto le permite ejecutar algoritmos de introspección sobre B para descubrir en tiempo de ejecución sus características y comportamiento

- ⟨/⟩ ¿Cómo harías para descubrir en tiempo de ejecución el conjunto de propiedades vinculadas a atributos HTML por medio de data-binding?

```
function find (c) {
  return [].slice.call (c.attributes)
    .filter (function (attr){
      return attr.value.indexOf('{{') > -1;
    })
    .map (function (attr){
      return attr.name
    });
}
```

Por convenio estándar todas las propiedades de c vinculadas por data-binding a algún atributo harán uso en su valor de la notación de mostachos

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Estandarización



Busca Los Estándares

Durante toda actividad con componentes Web estudia las particularidades de la tecnología para descubrir puntos susceptibles de incluir prescripciones estándar.

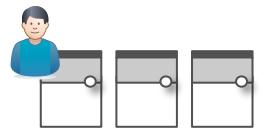
🎓 Es buena idea estudiar la tecnología para encontrar los puntos de las arquitecturas de componentes donde es susceptible definir algún estándar.

¿Qué partes de mi arquitectura de componentes requieren un convenio?

¿Dónde es posible prescribir estándares?

¿Dónde busco la homogeneidad arquitectónica?

Desarrollador



Plataforma

</> A modo de ejemplo describimos a continuación una colección de convenios de nombrado que aparecen dentro de las librerías core-* y paper-*.

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

```
<core-header-panel>
  <div class="core-header">...</div>
  <div>...</div>
</core-header-panel>
```

```
<core-icon icon="iconset:icon">
</core-icon>
```

```
<body unresolved fullbleed>
</body>
```

Principios de Diseño en Componentes Web

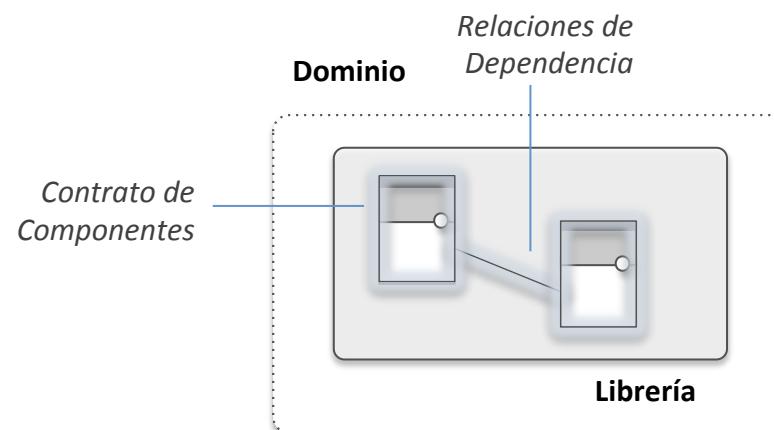
Principios de Diseño

Principio de Estandarización



Objetivos

- Homogeneizar la arquitectura
- Fomentar el desarrollo por terceros
- Mejorar las posibilidades compositivas
- Simplificar el uso de componentes
- Reducir la lógica adaptativa
- Mejora del time to market
- Mejorar la reutilización



Motivación



El uso de estándares y convenciones es sin duda la mejor manera que obtener arquitecturas homogéneas que facilitan el desarrollo y fomentan la reutilización.

Fases



Diseño de Componentes
Mantenimiento Evolutivo

Región



Contrato de Componentes
Relaciones de Dependencia

Principios de Diseño en Componentes Web

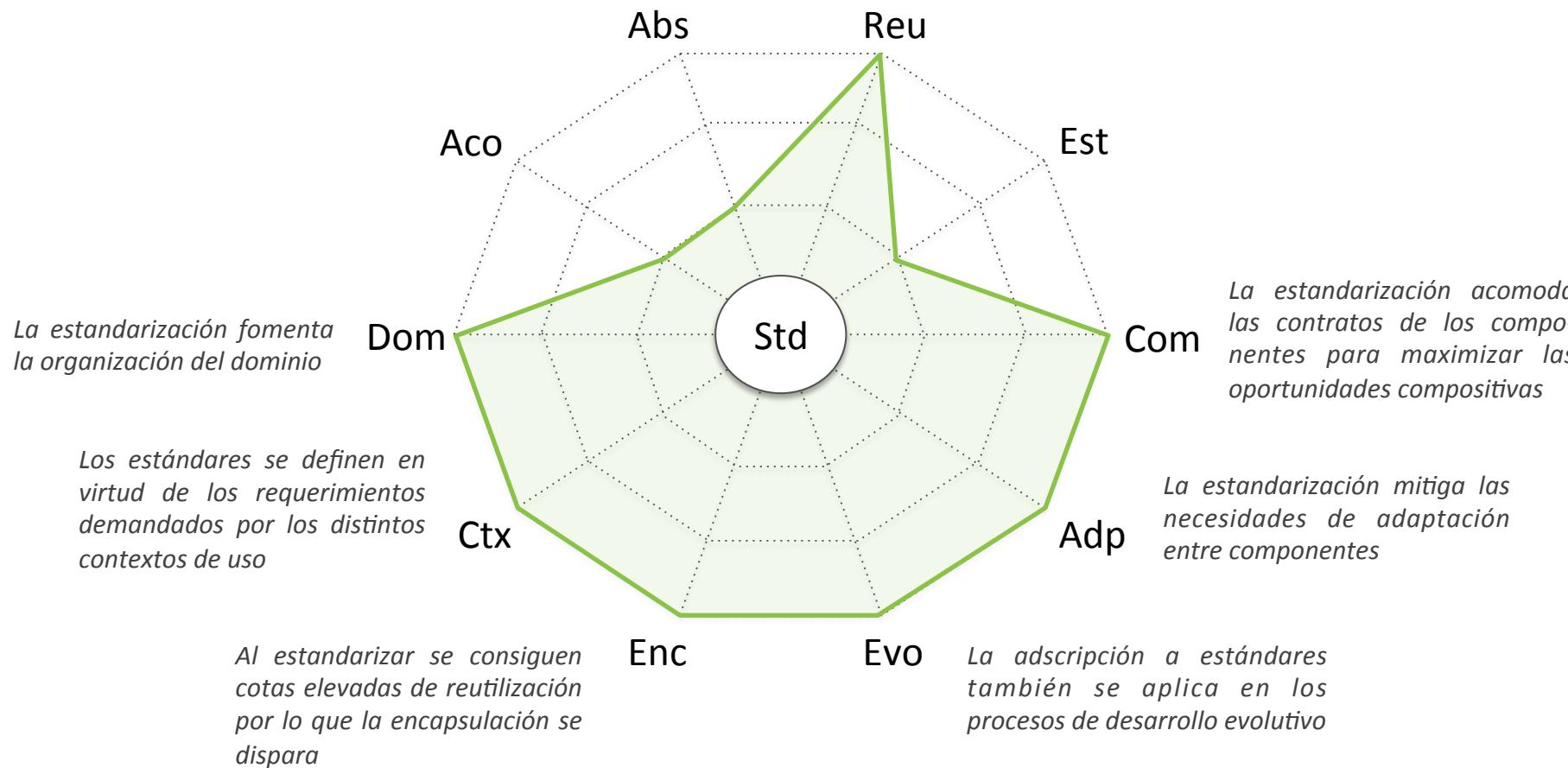
Principios de Diseño

Principio de Estandarización



Tensiones

La estandarización promueve la construcción de componentes con un alto índice de reutilización



Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Estandarización



Ejemplo

Facebook Material Design

Buena parte de los elementos de la interfaz siguen el estándar paper-element

A diferencia de lo que suele ocurrir en material las fotos de avatar con esta librería se presentan cuadradas

The screenshot shows the Facebook homepage using Material Design. The left side features a news feed with standard paper-elements: avatars, post cards with titles and descriptions, and a comment section. The right side has a sidebar with rounded cards for 'Trending' news items like 'The Simpsons' and 'NHL Hockey'. Below that is a 'Suggested Groups' section. At the bottom, there's a search bar and a footer with links.

La estética general de la interfaz se alinea con las directrices de la **comunidad material**

Toda la estética propia de Facebook debe estar recogida en **documentación**

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Contextualización



Nombre

Contextualización



Diseña tus componentes para que se acomoden automáticamente con fluidez máxima a cada contexto de uso en el que participen.



Actores



Arquitecto & Desarrollador de Componentes

Maquetador & Desarrollador Web

Principios de Diseño en Componentes Web

Principios de Diseño

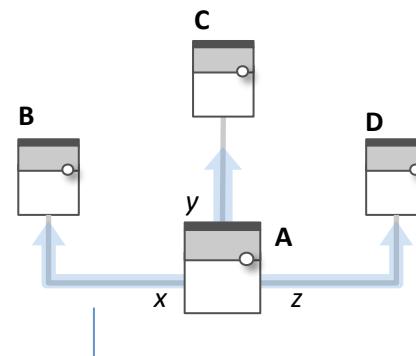
Principio de Contextualización



Explora y Descubre

Un componente debería ser responsable de explorar su vecindad autónomamente para descubrir posibles colaboraciones con otros componentes.

🎓 La colaboración entre componentes debería ser fruto de un descubrimiento implícito. Esto no sólo requiere descubrir participantes sino averiguar sus roles.



El componente A explora la vecindad y descubre a los componentes B, C y D que operan con roles x, y, y z respectivamente en relación a A

</> El componente core-drawer-panel opera con dos secciones de componentes independientemente de su tipo y comportamiento.

```
<core-drawer-panel>
  <div drawer>
    A
  </div>
  <div main>
    B
  </div>
</core-drawer-panel>
```

Core-drawer-panel opera con la vecindad A y B independientemente de quien sean, cómo sean y cómo operen estos componentes

Principios de Diseño en Componentes Web

Principios de Diseño

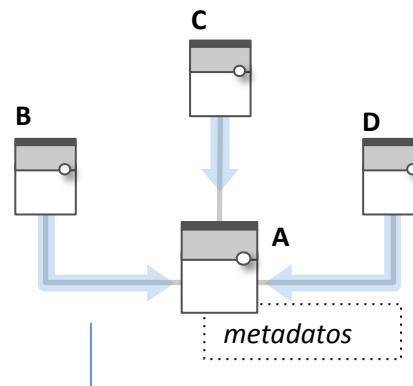
Principio de Contextualización



Déjate Encontrar

En aras a facilitar el descubrimiento, cada componente debería exponer la mayor cantidad de información posible sobre sí mismo en base a mecanismos de estandarización.

🎓 La meta-documentación exhaustiva es la clave para facilitar los procesos de descubrimiento basados en técnicas de localización e introspección.



El componente A se meta-documenta adecuadamente para permitir a los componentes B, C y D descubrir e interpretar el comportamiento de A

</> Core-drawer-panel exige, como parte de su contrato de uso, que los componentes identifiquen su rol con los atributos drawer y main respectivamente.

```
<core-drawer-panel>
  <div drawer>
    A
  </div>
  <div main>
    B
  </div>
</core-drawer-panel>
```

Core-drawer-panel ofrece un esquema de meta-documentación para identificar los roles que juegan los componentes A y B

Principios de Diseño en Componentes Web

Principios de Diseño

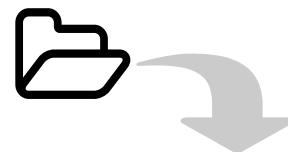
Principio de Contextualización



Opera Por Defecto

Idealmente, los componentes deberían presentar un comportamiento por defecto similarmente a como ocurre con las etiquetas estándar.

- 🎓 En alineamiento con el principio de racionalidad de la Web ninguna etiqueta debería exigir una configuración compleja para un uso por defecto.



Todo componente debe presentar un comportamiento out of the box que no exija grandes composiciones ni complejas configuraciones

- </> El componente google-map presenta siempre un comportamiento adecuado incluso aun cuando no tiene una información de configuración.

```
<google-map>  
</google-map>
```



Cuando el componente se despliega sin latitud y longitud, google map muestra una representación por defecto

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Contextualización

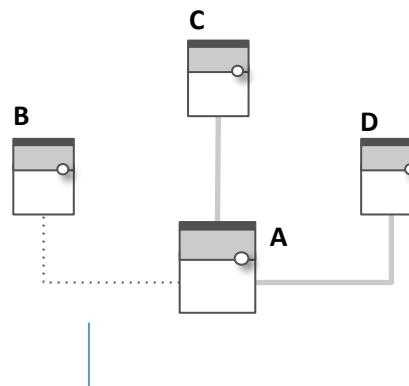


Opera Transparentemente

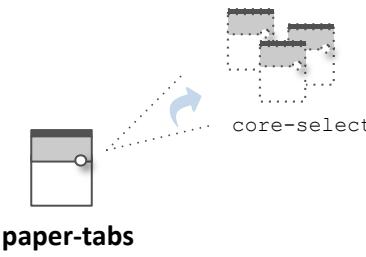
En caso de que un componente no encuentre un contexto de colaboración, éste debería seguir operando igualmente mostrando su comportamiento por defecto no colaborativo.

Todo componente debe presentar la suficiente autonomía como para poder operar funcionalmente en aquellos casos en los que el contexto resulte deficiente.

</> Por ejemplo si un componente emite eventos que nadie escucha el evento quedará sin atender pero el componente seguirá operando.



El componente A explora la vecindad y descubre a los componentes C y D. Aunque B no es compatible con A el componente debe seguir operando



Dado que paper-tabs no tiene componentes que responden a los eventos core-select no presentará comportamiento reactivo alguno pero eso no hará que el componente no funcione

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Contextualización

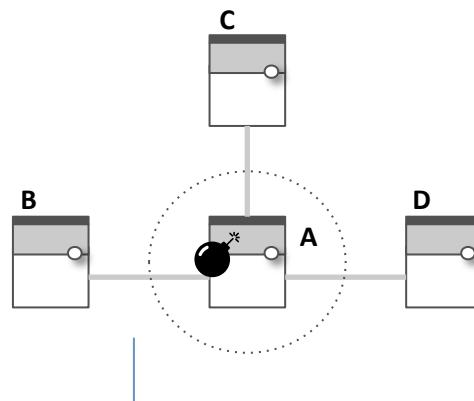


Falla Silenciosamente

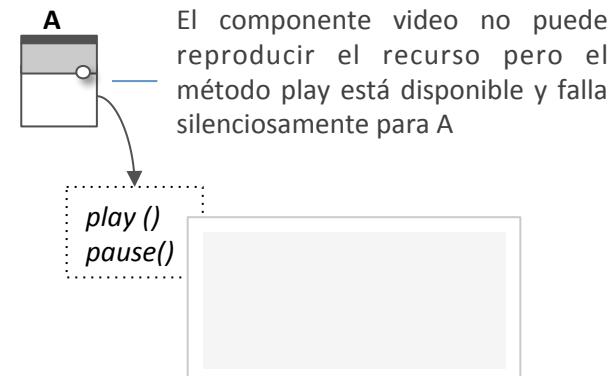
Si un componente se encuentra en una situación de fallo entonces el error no debería ser intrusivo con respecto a la experiencia de usuario.

🎓 Esto implica que los componentes en fallo no deberían propagar sus errores a otros componentes fuera de su ámbito de encapsulación.

</> Cuando la etiqueta video no encuentra los recursos necesarios falla en su reproducción pero este hecho no afecta a los componentes que colaboran con el.



El componente A falla pero ello no repercute en los componentes de la vecindad



El componente video no puede reproducir el recurso pero el método play está disponible y falla silenciosamente para A

Principios de Diseño en Componentes Web

Principios de Diseño

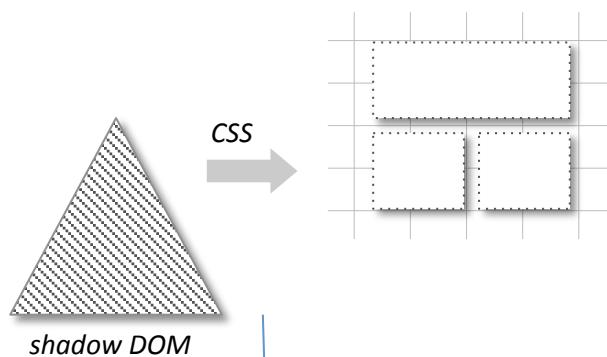
Principio de Contextualización



Desacopla El Layout

Idealmente el posicionamiento que ocupan los componentes en relación al espacio de pantalla no debería estar condicionado por la estructuración semántica dentro del árbol DOM.

🎓 Es conveniente no condicionar el layout por la estructuración semántica y sobre todo no condicionar la estructuración semántica por el layout.

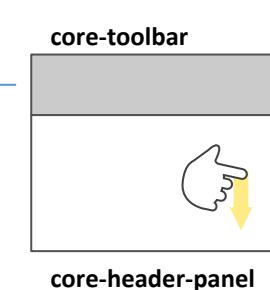


La estructura del DOM se centra en los aspectos semánticos. El layout debe resolverse con lógica CSS

</> Un buen ejemplo de este tipo de desacoplamiento lo encontramos en la relación que guarda core-header-panel con core-toolbar.

```
<core-header-panel>
  <div class="content"> ... </div>
  <core-toolbar>
    ...
  </core-toolbar>
</core-header-panel>
```

Independientemente de donde se incluya el componente core-toolbar dentro del core-header-panel, éste se posiciona en la cabecera



Principios de Diseño en Componentes Web

Principios de Diseño

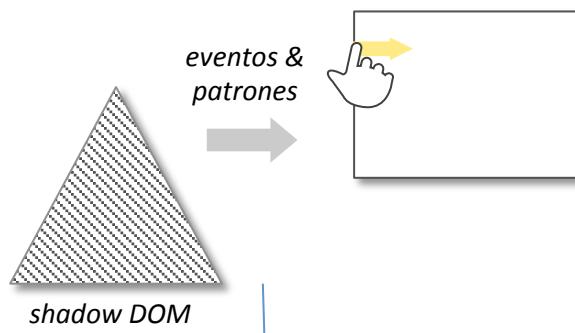
Principio de Contextualización



Desacopla La Interacción

Igualmente el modelo de interacción soportado por los componentes tampoco debería estar condicionado por la estructuración semántica dentro del árbol DOM.

- 🎓 Las oportunidades de soporte a la interacción en relación a la experiencia de usuario no debería verse condicionada por las restricciones en la estructura semántica.



Debería ser posible cambiar los modelos de interacción UX fácilmente y no verse condicionado por la posición relativa de los componentes en el DOM

- </> El componente core-selector define un atributo target para redirigir a otro nodo DOM dónde deben buscarse los proveedores de selección.

```
<core-selector target="{{$.target}}>
</core-selector>
...
<ul id="target">
  <li> Uno </li>
  <li> Dos </li>
  <li> Tres </li>
</ul>
```

En este caso los proveedores de selección no son los nodos anidados en core-selector como ocurre habitualmente sino los del nodo externo ul. Esta es una forma de desacoplamiento de la estructura DOM

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Contextualización

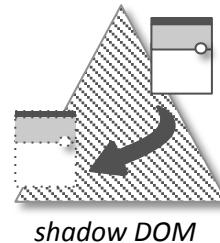


Fomenta La Reubicación

En la medida de lo posible asegúrate de que cada componente encuentra colaboración con la vecindad independientemente de donde se ubique.

🎓 Las reubicación contextual fomenta la versatilidad del componente en tanto que ofrece un comportamiento de mayor desacoplamiento del contexto DOM.

Una buena arquitectura de componentes debería ser flexible en relación a que posición relativa que ocupan los componentes dentro del DOM (light & shadow)



Esta es una clara manifestación del desacoplamiento que deben presentar los componentes en relación al contexto

</> El mismo ejemplo que la faceta anterior nos sirve para reflejar el hecho de que la composición debe fomentar la reubicación.

```
<ul id="target">
  <li> Uno </li>
  <li> Dos </li>
  <li> Tres </li>
</ul>
...
<core-selector target="{{$.target}}>
</core-selector>
```

Gracias al desacoplamiento la posición del nodo ul en relación al componente core-selector no es relevante

Principios de Diseño en Componentes Web

Principios de Diseño

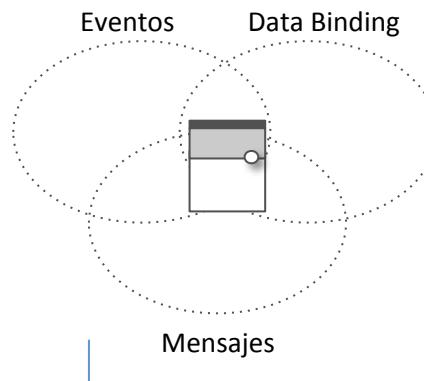
Principio de Contextualización



Respetá La Web

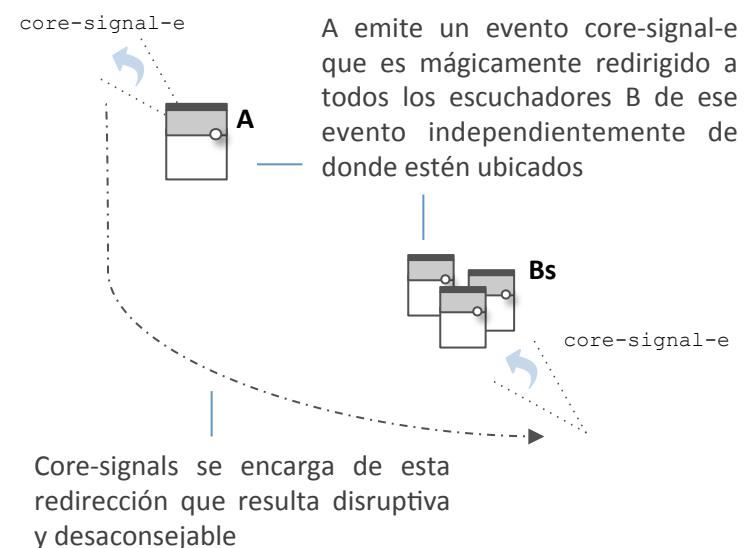
El comportamiento de un componente en un contexto de uso debe acomodarse a los mecanismos provistos por la Web como plataforma.

- En Polymer existen tres mecanismos esenciales de comunicación nativa. Todo componente debería acomodarse a alguno o varios de ellos.



Eventos, mensajes y data-binding son los tres mecanismos de comunicación entre componentes. Idealmente no deberían idearse mecanismos de comunicación alternativos sin una razón justificada

- Dado que en la Web existe un modelo de eventos por bubbling el componente core-signals que implementa el patrón pub-sub resulta disruptivo.



Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Contextualización

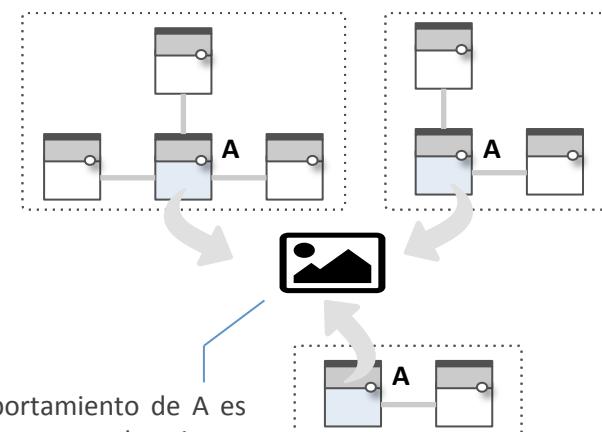


Se Predecible

El rango de comportamientos de un componente debería ser siempre el mismo independientemente del contexto de uso en el que se despliegue.

🎓 Evidentemente la vecindad condicionará parcialmente el comportamiento al articular diferentes composiciones pero éste debería ser básicamente el mismo.

</> Un buen ejemplo de predictibilidad lo encontramos en la relación que se da entre core-icon y core-iconset.



El comportamiento de A es básicamente el mismo independientemente de la vecindad de que disponga

```
<core-iconset id="icons-A" src="iconsA.png" ...>
<core-iconset id="icons-B" src="iconsB.png" ...>
<core-iconset id="icons-C" src="iconsC.png" ...>
```

```
<core-icon icon="icons-A:car">
</core-icon>
```

Si los componentes core-iconset no son cargados la base de datos de iconos será más reducida pero core-icon aún tendrá una colección de iconos por defecto que ofrecer

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Contextualización



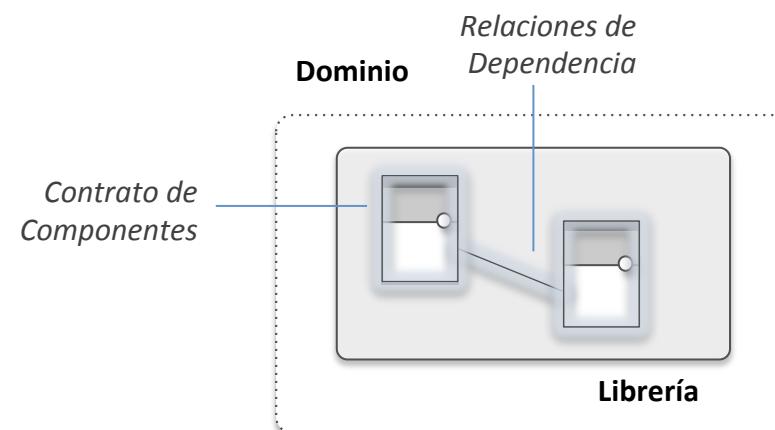
Objetivos

- Fomentar la autonomía
- Aumentar la reutilización
- Aumentar la interoperabilidad
- Reducción de la lógica específica
- Fomentar el descubrimiento
- Acomodar la UX al contexto

Motivación



La contextualización persigue el desacoplamiento de los componentes y de su comportamiento a la posición relativa dentro de la estructura semántica del DOM.



Fases



- Diseño de Componentes
- Desarrollo de soluciones

Región



- Contrato de Componentes
- Relaciones de Dependencia

Principios de Diseño en Componentes Web

Principios de Diseño

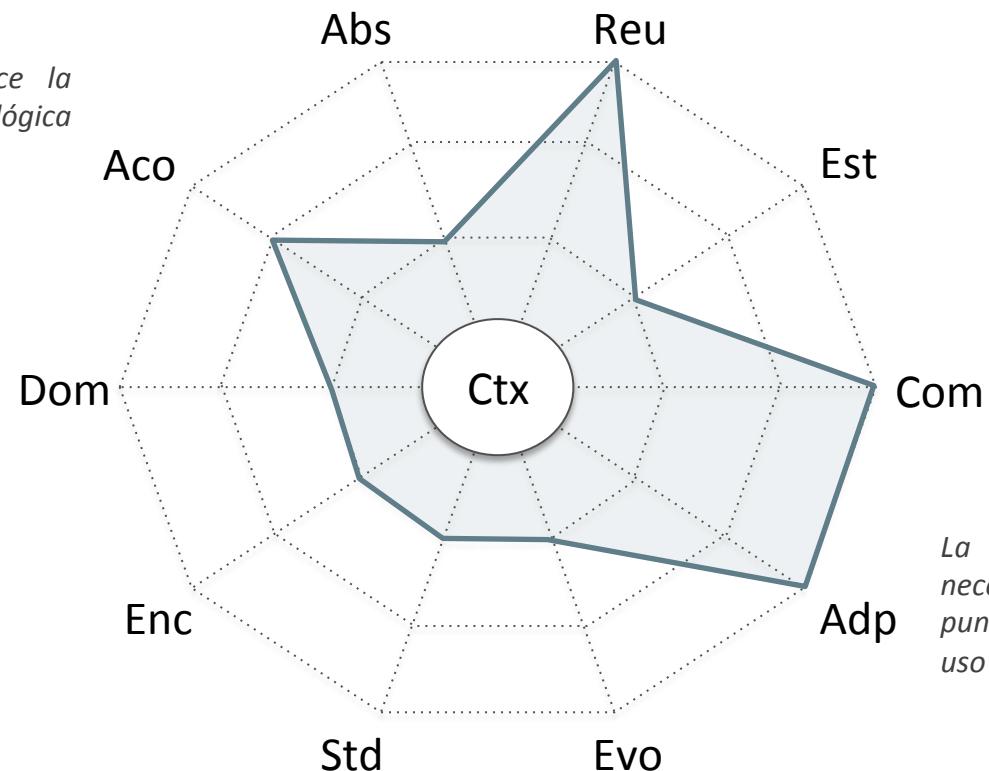
Principio de Contextualización



Tensiones

La contextualización reduce la necesidad de desarrollar lógica específica de contexto

Al aumentar la contextualización crecen las oportunidades de reutilización del componente



La contextualización permite mejorar las oportunidades compositivas en distintos contextos

La contextualización mitiga la necesidad de hacer adaptaciones puntuales para cada contexto de uso

Principios de Diseño en Componentes Web

Principios de Diseño

Principio de Contextualización



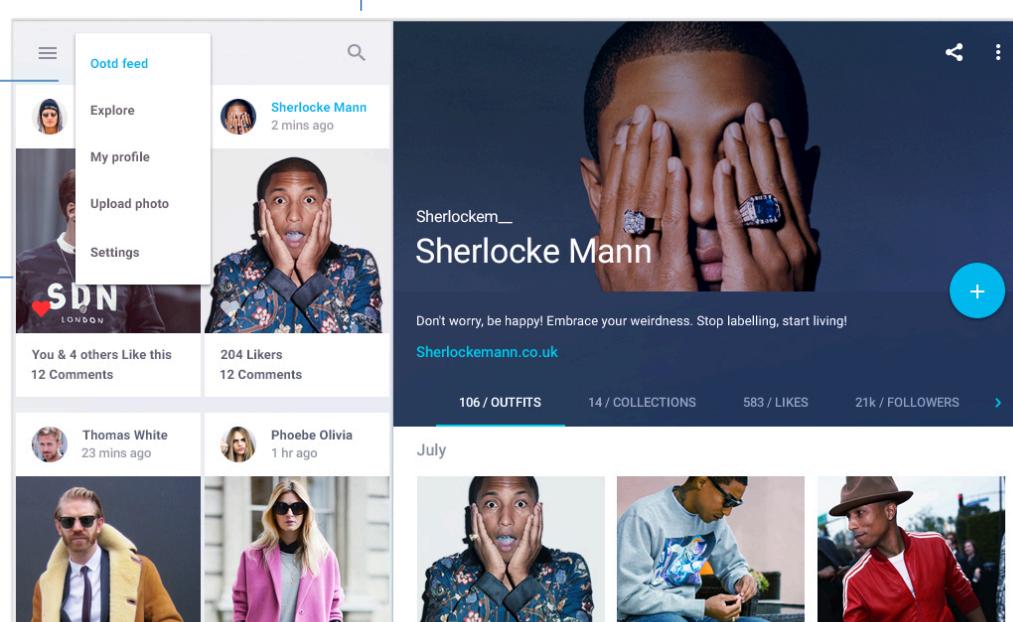
Ejemplo

Facebook Material Design

El mecanismo de exploración que hacen los elementos de menu con respecto al padre es utilizar la comunicación por eventos en bubbling

Si el item de menu no encuentra escuchador el evento se pierde y se falla silenciosamente

La lupa opera transparente y predeciblemente. Filtra los elementos del interfaz y si éstos no estuvieran presentes el resultado sería inocuo



El estilo de la Web impone una comunicación por eventos en bubbling

<http://goo.gl/xJYUel>

Principios de Diseño en Componentes Web

Preguntas

Reutilización

Acoplamiento

Abstracción

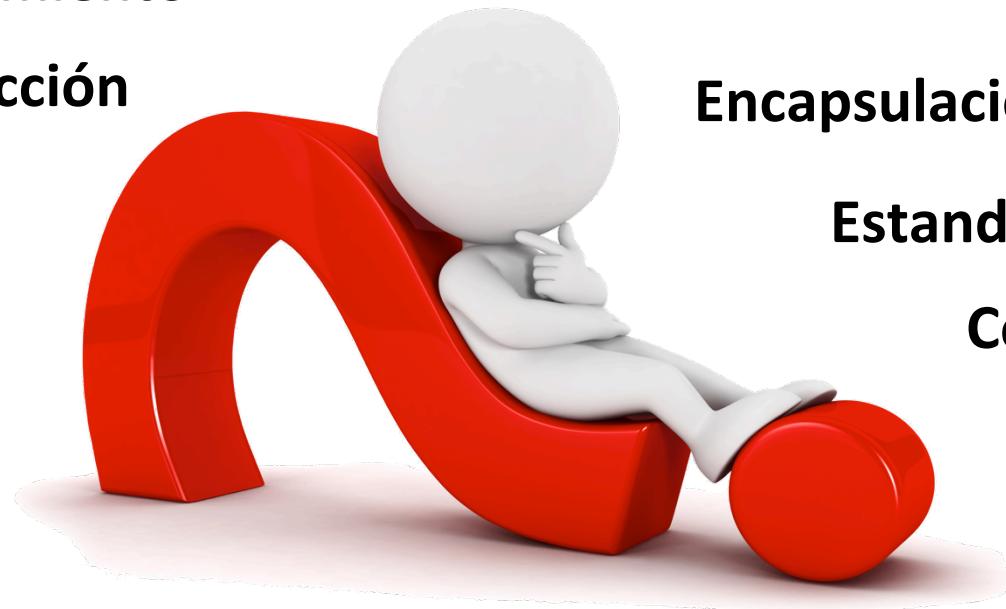
Dominio

**Estado Composición
Adaptación**

Encapsulación Evolución

Estandarización

Contextualización



Javier Vélez Reyes

@javiervelezreye

Javier.velez.reyes@gmail.com

Programación Orientada a Componentes Web

Principios de Diseño en Componentes Web

Javier Vélez Reyes

@javiervelezreye

Javier.velez.reyes@gmail.com

Marzo 2015

