Web Components

by @tomascornelles

La web evoluciona continuamente

W3C ha aprendido de nosotros

Se adapta a nuestra forma de trabajar

Animaciones y efectos en CSS3, etiquetas semanticas...

Se adelanta a nuestras necesidades

Selectores CSS3, APIs en JS, nuevas etiquetas funcionales...

Pero cada vez exigimos más

Internet cada vez avanza más rápido

No podemos **esperar** nuevos elementos,
ni a que lo **implementen** los navegadores

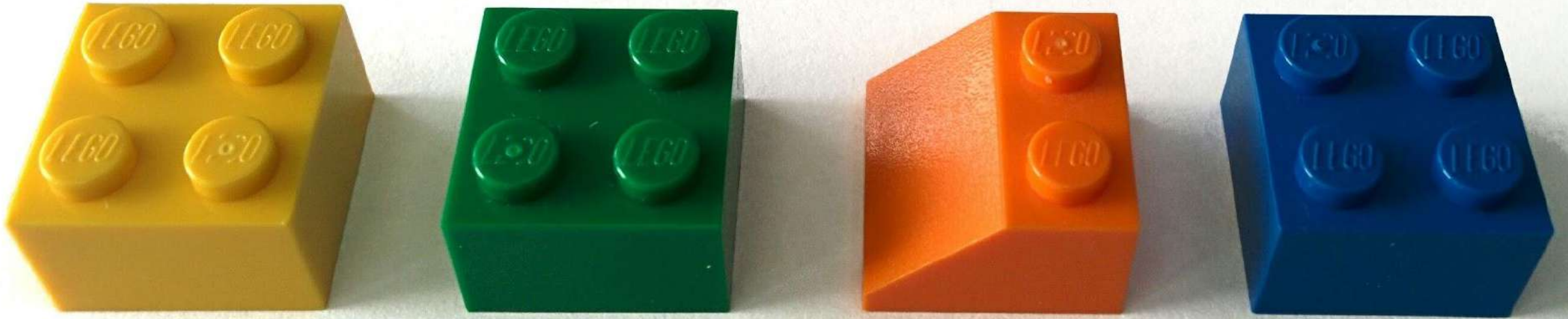
Queremos más y lo queremos **AHORA**

Imaginad crear un slider así de fácil

```
<mi-carousel>
    <img src="image1.jpg">
    <img src="image2.jpg">
    <img src="image3.jpg">
    <img src="image4.jpg">
</mi-carousel>
```

¿Como?

Web Components



¿Quién está hablando?

Tomás Cornelles

Frontend developer

Block builder

Veterano de la primera guerra de navegadores

Scientific

Photographer

Los web components nos ofrecen las herramientas necesarias

- Templates
- Shadow DOM
- Custom Elements
- HTML Import

Templates

Los templates nos permiten **definir** fragmentos de HTML para reutilizar.

El contenido del template NO se representa hasta ser usado

```
<table id="datos">
 <thead>
  <tr>
   <th>Nombre <th>Teléfono <th>Email
 <tbody>
  <template id="filas">
   <tr>
    <td><td><td>
  </template>
</table>
```

Templates

Mediante JavaScript **clonamos** el elemento

Y lo **insertamos** donde deseemos

```
var datos = document.querySelector('#datos');
var filas = document.querySelector('#filas');
var clone = document.importNode(filas.content);
datos.appendChild(clone);
```

Shadow DOM

Nos permite **encerrar** el código en un entorno aislado.

Por defecto, nada entra ni sale del Shadow boundary

Evita **conflictos** entre componentes

```
<div id="host">Visible</div>
var host = document.querySelector('#host');
var root = host.createShadowRoot();
var div = document.createElement('div');
div.textContent = 'Oculto';
root.appendChild(div);
```

Shadow DOM

```
<div id="host">Visible</div>
var host = document.querySelector('#host');
var root = host.createShadowRoot();
var div = document.createElement('div');
div.textContent = 'Oculto';
root.appendChild(div);
```

Los navegadores **compatibles** con el Shadow DOM mostrarán los textos "Visible" y "Oculto".

Los estilos o eventos encerrados no se aplican al exterior.

Custom Elements

Permite **definir** elementos personalizados

Va más allá de crear etiquetas semánticas

Son el eje **principal** de los Web Components

Custom Elements

Para crear un nuevo elemento personalizado lo **registramos** mediante javascript

```
var miElemento = document.registerElement('mi-elemento', {
  prototype: Object.create(HTMLElement.prototype)
});
```

¡Ya podemos **usar** el nuevo elemento!

```
<mi-elemento>
```

Custom Elements

También podemos **extender** los elementos existentes

```
var miElemento = document.registerElement('mi-elemento', {
  prototype: Object.create(HTMLElement.prototype),
  extends: 'h2'
});
```

Para usarlo añadimos el atributo is

```
<h2 is="mi-elemento">
```

HTML Import

Nos permite **incorporar** varios componentes externos
y usar sólo los que necesitemos en cada momento

Son diferentes a los includes

Ya existía para las hojas de estilos

```
<link rel="import" href="mi-elemento.html">
```

Browser Support

	Chrome	Opera	Firefox	Safari	IE
Templates					
Shadow DOM					
Custom elements					
HTML imports					

Para aumentar la compatibilidad, Polyfills

webcomponents.js

Polymer (Google)

X-Tags (Mozilla)

¿Cómo creo mi propio componente?

mi-elemento.html

```
<template id="mi-elemento">
     <style>h2 {color:green}</style>
     <content><h2>Mi contenido</h2></content>
</template>
<script>
var proto = Object.create(HTMLElement.prototype);
proto.createdCallback = function() {
  var root = this.createShadowRoot();
  var template = document.querySelector('#mi-elemento');
  var clone = document.importNode(template.content, true);
  root.appendChild(clone);
}
var miElemento = document.registerElement('mi-elemento', {
  prototype: proto
});
</script>
```

¿Cómo creo mi propio componente?

En tu archivo.html de destino, importa mi-elemento.html

```
<link rel="import" href="mi-elemento.html">
```

e invoca al elemento con su etiqueta

```
<mi-elemento>
```

Resultado

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Mi Elemento</title>
  <style>h2 {color: red}</style>
  <link rel="import" href="mi-elemento.html">
</head>
<body>
  <h2>Visible</h2>
  <mi-elemento></mi-elemento>
  </body>
</html>
```

Visible
Mi contenido

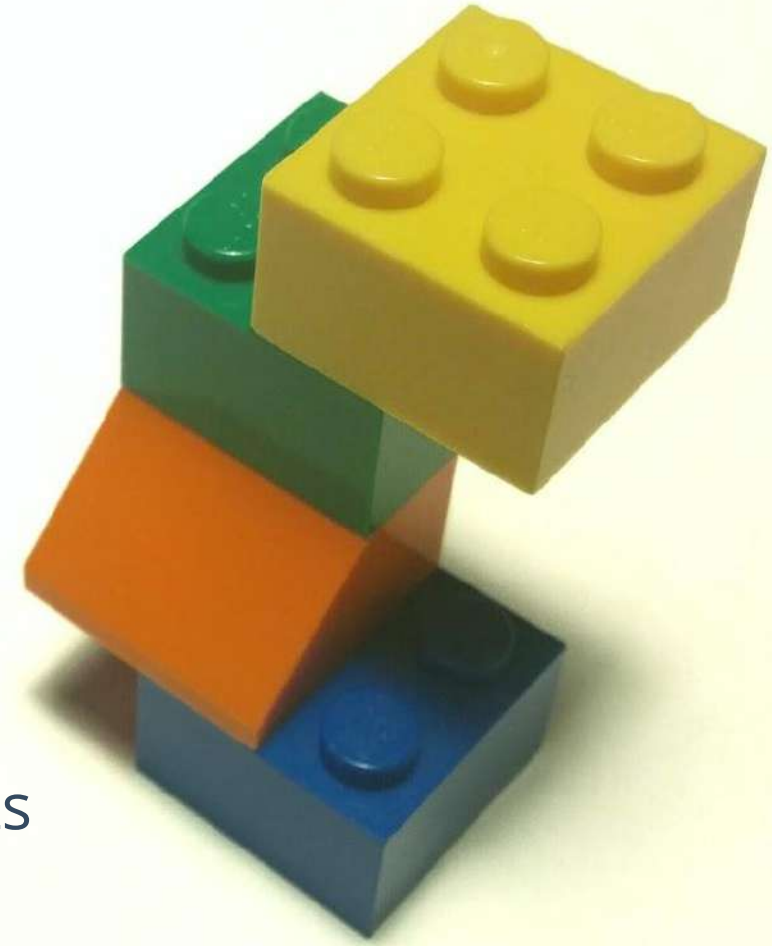
Es solo el primer día del futuro

La web debe ser **descriptiva**

Debe **unificar** esfuerzos entre tecnologías

El futuro del HTML está en **nosotros**

<muchas-gracias>



@tomascornelles

/tomascornelles

tomascornelles.com

Más información en

tomascornelles.com/webcomponents