

COMPUTO 1

Semana1

¿Qué es JavaScript?

JavaScript es un lenguaje de programación que los desarrolladores utilizan para hacer páginas web interactivas. Desde actualizar fuentes de redes sociales a mostrar animaciones y mapas interactivos, las funciones de JavaScript pueden mejorar la experiencia del usuario de un sitio web. Como lenguaje de scripting del lado del servidor, se trata de una de las principales tecnologías de la World Wide Web. Por ejemplo, al navegar por Internet, en cualquier momento en el que vea un carrusel de imágenes, un menú desplegable “click-to-show” (clic para mostrar), o cambien de manera dinámica los elementos de color en una página web, estará viendo los efectos de JavaScript.

Funcionamiento JavaScript

El código JavaScript es interpretado, es decir, que es directamente traducido a código de lenguaje de máquina mediante un motor de JavaScript.

JavaScript | Lado del Cliente

JavaScript del lado del cliente se refiere a la forma en que JavaScript funciona en el navegador. En este caso, el motor de JavaScript está dentro del código del navegador. Todos los principales navegadores web incluyen sus propios motores de JavaScript incorporados.

Los desarrolladores de aplicaciones web escriben código JavaScript con diferentes funciones asociadas a varios eventos, como hacer clic con el ratón o situar el ratón sobre un elemento. Estas funciones realizan cambios en HTML y CSS.

JavaScript | Lado del Servidor

JavaScript del lado del servidor hace referencia al uso del lenguaje de codificación en la lógica del servidor de Backend. El motor de JavaScript se encuentra directamente en el servidor. Una función de JavaScript del lado del servidor puede acceder a la base de datos, realizar diferentes operaciones lógicas y responder a varios eventos desencadenados por el sistema operativo del servidor. La ventaja principal del scripting del lado del servidor es que admite un alto nivel de personalización de la respuesta del sitio web según sus requisitos, sus derechos de acceso y las solicitudes de información provenientes del sitio web.

Semana2

¿Qué es Bootstrap?

Bootstrap es un marco de desarrollo web que ayuda a los desarrolladores a crear interfaces web. Concebido originalmente en Twitter en 2011 por Mark Otto y Jacob Thornton, el marco es ahora de código abierto y se ha convertido en uno de los programas de desarrollo web más populares frameworks hasta la fecha.

Diseño

- Contenedores responsivos
- Puntos responsivos para ajustar el diseño de la página en respuesta a una pantalla de diferentes tamaños
- Un diseño de cuadrícula de 12
- columnas para organizar de
- manera flexible los elementos

- Objetos multimedia que funcionan como bloques de construcción
- Clases de utilidad que le permiten manipular elementos de manera responsiva

Componentes

Los componentes permiten construir rápidamente las características más fundamentales de un sitio web. Los componentes de la interfaz de usuario de Bootstrap abarcan todos los bloques de construcción fundamentales que esperarías que un kit de herramientas de desarrollo web ofreciera: cuadros de diálogo modales, barras de progreso, navegación, barras, información sobre herramientas, ventanas emergentes, un carrusel, alertas, menús desplegables, grupos de entrada, pestañas, paginación y componentes para enfatizar ciertos contenidos.

Clases

Aparte de los componentes listos para usar, Bootstrap ofrece una gran selección de clases de utilidad que encapsular las reglas de estilo más comúnmente necesarias.

	Extra small devices Phones (<768px)	Small devices Tablets (≥768px)	Medium devices Desktops (≥992px)	Large devices Desktops (≥1200px)
<code>.visible-xs-*</code>	Visible	Hidden	Hidden	Hidden
<code>.visible-sm-*</code>	Hidden	Visible	Hidden	Hidden
<code>.visible-md-*</code>	Hidden	Hidden	Visible	Hidden
<code>.visible-lg-*</code>	Hidden	Hidden	Hidden	Visible
<code>.hidden-xs</code>	Hidden	Visible	Visible	Visible
<code>.hidden-sm</code>	Visible	Hidden	Visible	Visible
<code>.hidden-md</code>	Visible	Visible	Hidden	Visible
<code>.hidden-lg</code>	Visible	Visible	Visible	Hidden

¿Qué es VueJS?

Es un marco progresivo para construir interfaces de usuario. A diferencia de otros marcos monolíticos, Vue está diseñado desde cero para ser adoptable de forma incremental. La biblioteca principal se centra solo en la capa de vista y es fácil de recoger e integrar con otras bibliotecas o proyectos existentes. Por otro lado, Vue también es perfectamente capaz de impulsar aplicaciones sofisticadas de una sola página cuando se usa en combinación con herramientas modernas y bibliotecas de soporte .

VueJS

VueJS es una librería javascript pensada para tener un framework con el que desarrollar páginas web. Con Vue se pueden crear todas las vistas de una página web, se pueden hacer dinámicas, puede conectarse a un servidor para tener datos dinámicos de una base de datos, etc.

Características Vue.js

- Modularidad. Es completamente modular.
- Reactividad. Posee propiedades reactivas, lo que quiere decir que si cambia una variable en una parte

de la vista de la página, Vue actualizará el valor sin necesidad de hacerlo manualmente.

- Componentes Web. Es basado en componentes, un componente web es una parte que puede ser reutilizada.
- Virtual DOM. Si hay que hacer un cambio en la vista, en lugar de sustituir directamente los nuevos valores en la vista, Vue creará una especie de réplica del DOM, es decir, de los elementos de la página web para que a la hora de hacer cambios en la vista se hagan de forma óptima.
- Eventos y transiciones. Puede reacciones a eventos que realiza el usuario, por ejemplo, cuando un usuario hace clic en un elemento.
- Mixins. Los mixins son funciones y lógica de los componentes que se pueden reutilizar y reusar en otros componentes web.
- Lifecycle. Se puede controlar lo que ocurre antes de que se cargue el componente, lo que pasa justo al cargarse o al destruirse
- Aplicaciones SPA. Posee un paquete especial para tener un sistema de rutas.

Vue CLI

Vue tiene su propia interfaz de línea de comandos. Permite inicializar una aplicación Vue con la configuración que se desea.

Instalación con npm:

```
C:\Users\Gisela>npm install -g vue-cli
npm WARN deprecated vue-cli@2.9.6: This package has been deprecated in favour of @vue/cli
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
npm WARN deprecated coffee-script@1.12.7: CoffeeScript on NPM has moved to "coffeescript" (no hyphen)
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
C:\Users\Gisela\AppData\Roaming\npm\vue-list -> C:\Users\Gisela\AppData\Roaming\npm\node_modules\vue-cli\bin\vue-list
C:\Users\Gisela\AppData\Roaming\npm\vue -> C:\Users\Gisela\AppData\Roaming\npm\node_modules\vue-cli\bin\vue
C:\Users\Gisela\AppData\Roaming\npm\vue-init -> C:\Users\Gisela\AppData\Roaming\npm\node_modules\vue-cli\bin\vue-init
+ vue-cli@2.9.6
added 236 packages from 204 contributors in 36.084s
```

Existen diferentes formas de inicializar una aplicación:

- vue init webpack
- vue init webpack-simple
- vue init browserify
- vue init browserify-simple
- vue init simple

Vue CLI

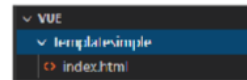
Para ver la diferencia, ejecutar vue init con una template simple y con webpack template, se podra notar la diferencia en la estructura

```
C:\Users\Gisela\Documents\vue>vue init simple templatesimple

? name templatesimple
? author Gisela

  vue-cli - Generated "templatesimple".

C:\Users\Gisela\Documents\vue>
```



Estructura

Vue CLI

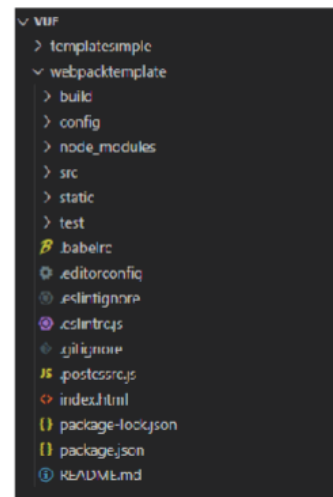
```
C:\Users\Gisela\Documents\vue>vue init webpack webpacktemplate

Project name webpacktemplate
Project description A Vue.js project
Author Gisela
Vue build standalone
Install vue-router? Yes
Use ESLint to lint your code? Yes
Pick an ESLint preset Standard
Set up unit tests? Yes
Pick a test runner jest
Set up e2e tests with Nightwatch? Yes
Should we run 'npm install' for you after the project has been created? (recommended) npm

  vue-cli - Generated "webpacktemplate".

Installing project dependencies ...
*****
npm WARN deprecated eslint-loader@1.9.0: This loader has been deprecated. Please use eslint-webpack-plugin
npm WARN deprecated extract-text-webpack-plugin@3.0.2: Deprecated. Please use https://github.com/webpack-contrib/mini-css-extract-plugin
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
npm WARN deprecated core-js@2.5.12: core-js@<3 is no longer maintained and not recommended for usage due to the number of issues. Please, upgrade your dependencies to the actual version of core-js@.
npm WARN deprecated browserslist@2.11.3: Browserslist 2 could fail on reading Browserslist >3.0 config used in other tools
[ ] / fetchMetadata: GET range manifest for resolve@1.3.3 fetched in 3494ms
```

Vue CLI
Fuente: Docente Autor de contenidos



Estructura

Semana3

Rendering declarativo

El rendering es una opción que propone VueJS para interactuar con el DOM, para lo cual se tendría:

- Vista: HTML
- Estado: Datos JS
- Usuario: Interactúa e introduce cambios en la vista.

La vista es la encargada de decirle al estado que hay cambios, a lo cual el estado responde enviando una nueva vista.

El estado se encarga de decirle a la vista como y cuando debe compilar y cuál será el resultado que debe lograr, esto se hace mediante una función llamada: render.

Render

Los sitios web dinámicos son creados cuando existe la posibilidad de representar datos; a continuación se crea un nuevo componente y una variable llamada mensaje, la cual será renderizada en la aplicación.

```
programacion.vue X
vuejs-app > src > components > programacion.vue
1 <template>
2   <div>
3     <h2>{{ mensaje }}</h2>
4   </div>
5 </template>
6
7
8 <script>
9 export default {
10   name: 'Programacion',
11   props: {
12     mensaje: String
13   }
14 }
15 </script>
```

```
programacion.vue App.vue X
vuejs-app > src > App.vue
1 <template>
2   <div id="app">
3     
4     <Programacion mensaje="Programacion Computacional IV"/>
5   </div>
6
7 </template>
8
9
10
11 <script>
12 import Programacion from './components/programacion.vue'
13
14 export default {
15   name: 'App',
16   components: {
17     Programacion
18   }
19 }
20 </script>
21
22
```

Directivas

Las directivas de Vue son atributos especiales que se colocan en las etiquetas HTML y están prefijados por v-, como por ejemplo, v-for, v-bind o v-on, entre muchas otras. Estas directivas permiten realizar acciones dinámicas potentes (bucles, condicionales, etc...) que no se pueden realizar en HTML por si solo, pero que Vue permite utilizar en sus etiquetas <template>.

Directivas básicas

Directiva vue	Valor	Descripción
v-text	Sí	Equivalente a <code>{{ texto }}</code> . Usa <code>.textContent</code> internamente.
v-html	Sí	Inserta HTML en un elemento sin procesarlo. Usa <code>.innerHTML</code> internamente.
v-pre	No	Mantiene las <code>{{ templates }}</code> del elemento intactas, sin renderizar.
v-once	No	Renderiza las <code>{{ templates }}</code> solo la primera vez, y no lo hace más.
v-cloak	No	Directiva que permanece hasta que la <code>{{ template }}</code> se renderiza con contenido.
v-model	Sí	Enlaza el valor de una variable con un <code><input></code> , <code><select></code> , <code><textarea></code> o un componente.

Directivas condicionales

Directiva vue	Descripción
v-show	Muestra/oculta el elemento, alternando con un display: none .
v-if	Equivalente a un if de Javascript. Acepta una expresión Javascript por parámetro.
v-else-if	Equivalente a un else if de Javascript. Acepta una expresión Javascript por parámetro.
v-else	Equivalente a un else de Javascript. No tiene parámetros.

Directivas personalizadas

Sintaxis:

- `Vue.directive(id, definition);`

Parámetros

- `Id`: cadena-> ID de la directiva que se utilizara
- `Definition`: objeto->un objeto de definición

Directivas personalizadas

Propiedades de la instancia directiva

- `el`: el elemento al que está dirigida la directiva.
- `vm`: el contexto ViewModel que posee esta directiva.
- `expresión`: la expresión del enlace, excluyendo argumentos y filtros.
- `arg`: el argumento, si está presente.
- `nombre`: el nombre de la directiva, sin el prefijo.
- `modificadores`: un objeto que contiene modificadores, si los hay.
- `descriptor`: un objeto que contiene el resultado del análisis de toda la directiva.

- params: un objeto que contiene atributos param. Explicado a continuación.

Representación de lista

Se puede representar una lista usando la directiva v-for . La sintaxis requiere que especifique la matriz de origen para iterar, y un alias que se usará para hacer referencia a cada elemento en la iteración. En el siguiente ejemplo, usamos items como la matriz de origen, y item como el alias para cada elemento.

Representación de lista

Se crea un modulo JS, en cual se genera una lista, y el método que servirá para agregar nuevos elementos, en el HTML se agrega una lista la cual mostrara los elementos del array.

```
<script>
var Vue({
  el: '#app',
  data: {
    lista: ['Web', 'Programacion IV', 'Ciclo 1-2021'],
  },
  elemento: '',
  methods: {
    agregarElementoLista: function(){
      this.lista.push(this.elemento); //agrega elemento a la lista
      this.elemento = '';
    }
  }
});
</script>
```

```
<div>{sample render de listas}</div>
<div id="app">
  <div>
    <li v-for="item in lista" v-text="item"></li>
  </div>
  <input type="text" v-model="elemento" v-on:keyup.enter="agregarElementoLista"/>
</div>
```

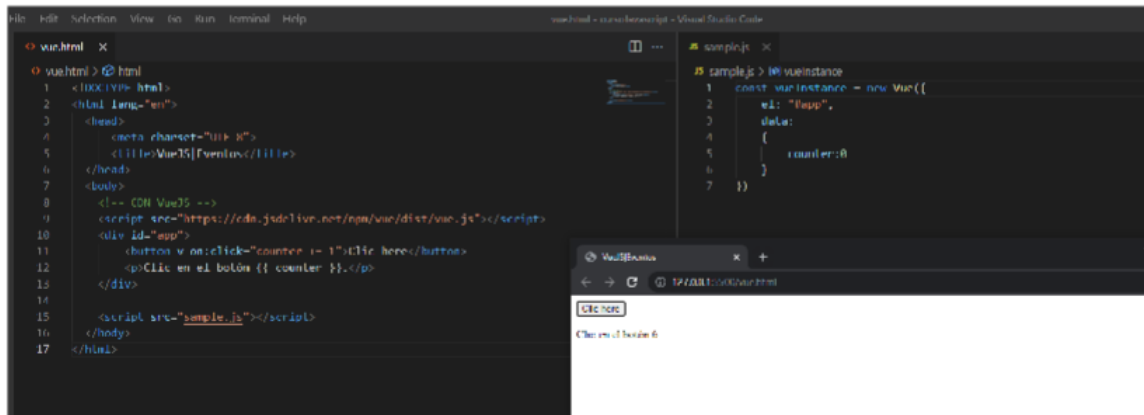
Semana4

¿Que son los eventos?

- En JavaScript, la interacción con el usuario se consigue mediante la captura de los eventos que éste produce. Un evento es una acción del usuario ante la cual puede realizarse algún proceso (por ejemplo, el cambio del valor de un formulario, o la pulsación de un enlace).
- Los eventos se capturan mediante los manejadores de eventos. El proceso a realizar se programa mediante funciones JavaScript llamadas por los manejadores de eventos.

Manejadores de eventos en vue.js

Se puede utilizar la directiva v-on para escuchar eventos del DOM

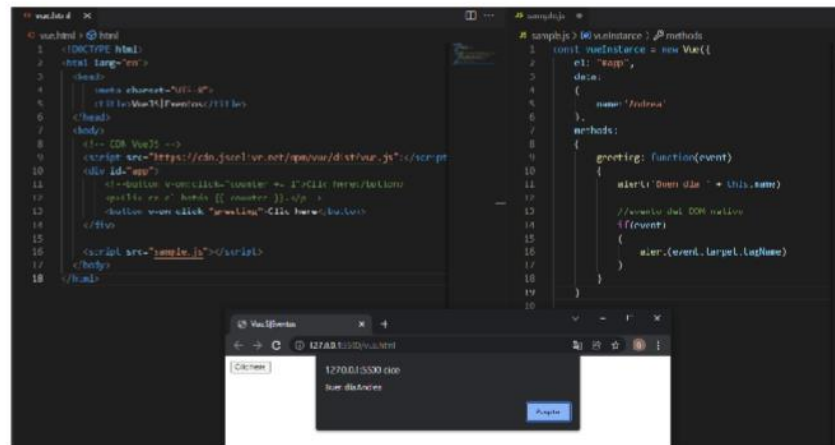


```
vue.html <?html>
1 <DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>VueJS [Eventos] </title>
6 </head>
7 <body>
8   <!-- CON VueJS -->
9   <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
10  <div id="app">
11    <button v-on:click="counter += 1">Click here</button>
12    <p>Click en el botón {{ counter }}.</p>
13  </div>
14
15  <script src="sample.js"></script>
16 </body>
17 </html>
```

```
sample.js <?vueinstance>
1 const vueInstance = new Vue({
2   el: "#app",
3   data:
4     {
5       counter: 0
6     }
7 })
```

Manejadores de eventos en vue.js

La lógica para muchos controladores puede resultar mas compleja por lo que no es posible mantener el valor del atributo v-on, en este caso v-on también puede aceptar el nombre del método al que quiere llamar.



```
vue.html <?html>
1 <DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>VueJS [Eventos] </title>
6 </head>
7 <body>
8   <!-- CON VueJS -->
9   <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
10  <div id="app">
11    <button v-on:click="greeting">Click here</button>
12    <p>Click en el botón {{ counter }}.</p>
13  </div>
14
15  <script src="sample.js"></script>
16 </body>
17 </html>
```

```
sample.js <?vueinstance> <?methods>
1 const vueInstance = new Vue({
2   el: "#app",
3   data:
4     {
5       name: 'Andres'
6     },
7   methods:
8     {
9       greeting: function(event)
10       {
11         alert('Hola día ' + this.name)
12       }
13     }
14   //evento del DOM native
15   {
16     //event
17     {
18       //event, target, tagName
19     }
20   }
21 })
```


Modificadores de eventos

Vue.js proporciona modificadores de eventos para v-on para manejar detalles de eventos DOM, como: `event.preventDefault()` `Oevent.stopPropagation()`

Vue.js invoca el modificador mediante el sufijo de instrucción indicado por un punto (.)

`.stop`

`.prevent`

`.capture`

`.self`

`.once`

Semana5

Componentes

Un componente en Vue.js es la forma de reutilizar una instancia de Vue tantas veces como sea necesario.

Mediante los componentes se pueden crear elementos propios html, cada uno con sus propiedades y métodos particulares que fácilmente se pueden reutilizar en toda la aplicación.

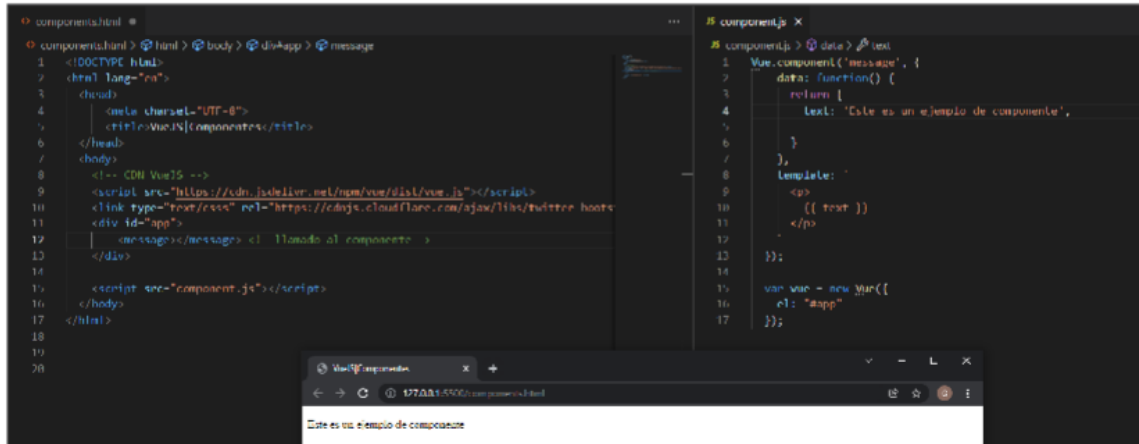
Sintaxis para creación de componentes

- name es el nombre del componente
- { options } son todas aquellas opciones que se pueden enviar a una instancia de Vue con dos pequeñas diferencias las cuales son las siguientes:
 - no es necesario especificar el elemento contexto { el: '#app' }
 - data ahora es una función que retorna un objeto.

`Vue.component("name", { options })`

Componentes

Ejemplo:



```
components.html
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <title>Vue.js[componentes]</title>
6 </head>
7 <body>
8   <!-- CDN VueJS -->
9   <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
10  <link type="text/css" rel="https://cdnjs.cloudflare.com/ajax/libs/twitter/boots
11  <div id="app">
12    <message></message> <!-- Llamado al componente -->
13  </div>
14
15  <script src="component.js"></script>
16 </body>
17 </html>
18
19
20

# component.js
1 Vue.component('message', {
2   data: function() {
3     return {
4       text: 'Este es un ejemplo de componente',
5     }
6   },
7   template: `
8     <p>
9       {{ text }}
10    </p>
11  `;
12 });
13
14
15 var vue = new Vue({
16   el: '#app'
17 });
```

Registro Global de Componentes

El registro global se realiza al llamar al método `component` de Vue. Que un componente se registre de forma global significa que estará disponible en cualquier componente de Vue no importa cuán anidado esté. Para ejemplo se crearán dos componentes, el primero contendrá una etiqueta `<p>` la cual se cargará en el segundo componente que contendrá una etiqueta `<div>`

Registro Local de Componentes

El registro global de componentes provee una gran ventaja también tiene una desventaja asociada. Resulta que con el tiempo puede parecer sencillo comenzar a acumular componentes de manera global aún cuando no se utilicen en todas las vistas de una aplicación. Para esto Vue aporta la flexibilidad de agregar los componentes que se utilizarán en un componente específico.

Envío de datos entre componentes

En Vue.js se envían datos desde un componente padre hacia el hijo mediante sus propiedades (`props`) y desde el componente hijo hacia el padre mediante eventos (`$emit`).

Para el ejemplo se crean dos componentes el primero captura los datos de los productos y el segundo componente mostrara el total de acuerdo con el precio y cantidad establecidos en el primer componente.

Envío de datos entre componentes

El primer componente será hermano de otro componente que capturará la información

```
Vue.component('productTotal',{
  props: ['total'],
  template: `
    <p>
      total: {{ total }}
    </p>
  `
});
```

Entonces en el HTML en donde estarán estos componentes será el siguiente:

```
<div id="app">
  <!--llamado al componente-->
  <product
    v-on:changeTotal="updateTotal"></product>

  <productTotal
    :total="totalP">
  </productTotal>
</div>
```

Envío de datos entre componentes

Paso de datos con el bus de eventos

Un bus de eventos no es más que otra instancia de Vue. Una instancia que puede tener las mismas propiedades que cualquier otra instancia. Se realizará un cambio en el ejemplo de la semana anterior se va a crear una instancia para generar en ella el evento changetotal.

Envío de datos entre componentes

Como el bus es una instancia diferente no tiene mucho sentido conservar el evento en el HTML, por lo cual se simplificaría de la siguiente manera:

```
var vue = new Vue({
  el: "#app",
  data: {
    totalP: 0
  },
  methods: {
    updateTotal: function(total) {
      this.totalP = total;
    }
  },
  created: function() {
    bus.$on('changetotal', this.updateTotal)
  }
});
```

Envío de datos entre componentes

Ejecución

