

Vue.js Tutorial

Sumario

- 1 Introducción a Vue.js
- 2 Fundamentos de Vue.js
 - ◆ 2.1 Trabajando con plantillas
 - ◆ 2.2 Métodos
 - ◆ 2.3 Introducción a directivas
 - ◇ 2.3.1 Expresiones y directivas
 - ◇ 2.3.2 Expresiones con directivas
 - ◆ 2.4 Introducción a eventos
 - ◇ 2.4.1 Paso de argumentos a eventos
 - ◇ 2.4.2 Modificadores de directivas de eventos
 - ◇ 2.4.3 Claves modificadoras
 - ◇ 2.4.4 Modificadores clave
 - ◆ 2.5 Binding en dos direcciones
 - ◆ 2.6 Seguridad mostrando y escapando HTML
 - ◆ 2.7 Renderizado elementos una sola vez
 - ◆ 2.8 Mostrar y ocultar elementos
 - ◆ 2.9 Ocultar elementos hasta que la instancia de Vue.js esté preparada
 - ◆ 2.10 Recorrido de arrays
 - ◆ 2.11 Recorrido de propiedades de un objeto
 - ◆ 2.12 Recorrido en rango numérico
 - ◆ 2.13 Explicación de actualizaciones usando v-for
 - ◆ 2.14 Detección de cambios en array
 - ◆ 2.15 Condicionales y bucles en binding bidireccional
 - ◆ 2.16 Optimización con propiedades
 - ◆ 2.17 Uso de getters y setters con propiedades
 - ◆ 2.18 Watchers
 - ◆ 2.19 Filtros
 - ◆ 2.20 Uso de estilos CSS en línea
 - ◆ 2.21 Accesos rápidos en bindings y eventos
- 3 Ejemplo de aplicación e-commerce con Vue.js
- 4 Profundizando en las instancias Vue.js
 - ◆ 4.1 Acceso a una instancia Vue.js desde afuera de su declaración
 - ◆ 4.2 Uso de múltiples instancias Vue.js en la misma página
 - ◆ 4.3 Proxy
 - ◆ 4.4 Entendiendo la reactividad
 - ◆ 4.5 Cola de actualizaciones asíncronas
 - ◆ 4.6 Entendiendo el DOM virtual
 - ◆ 4.7 Añadiendo watchers de forma dinámica
 - ◆ 4.8 Acceso al DOM con refs
 - ◆ 4.9 Montaje de plantillas dinámicamente
 - ◆ 4.10 Uso de plantillas inline
 - ◆ 4.11 Destrucción de una instancia Vue.js
 - ◆ 4.12 Más información sobre el ciclo de vida y hooks de una instancia Vue.js
- 5 Configuración de webpack con Vue.js
 - ◆ 5.1 Introducción
 - ◆ 5.2 Instalación de Chrome developer tools
 - ◆ 5.3 Introducción al CLI de Vue.js
 - ◆ 5.4 Creación de un proyecto Vue.js desde CLI
 - ◆ 5.5 Explicación de la estructura del proyecto Vue.js
 - ◆ 5.6 Componentes en un único fichero
 - ◆ 5.7 Compilación para producción del proyecto Vue.js
- 6 Componentes
 - ◆ 6.1 Introducción a componentes
 - ◆ 6.2 Por qué la propiedad data debe ser una función
 - ◆ 6.3 Componentes globales
 - ◆ 6.4 Creación de un componente

- ◆ 6.5 Organizando componentes
- ◆ 6.6 Estilos globales
- ◆ 6.7 Transfiriendo datos a los componentes
- ◆ 6.8 Validación de datos recibidos
- ◆ 6.9 Trabajando con eventos
- ◆ 6.10 Comunicación a través de un evento bus
- ◆ 6.11 Slots
- ◆ 6.12 Slots con nombre
- ◆ 6.13 Componentes dinámicos
- ◆ 6.14 Cómo mantener componentes dinámicos activos
- ◆ 6.15 Ciclo de vida de los componentes dinámicos
- 7 Ejemplo de aplicación mail con Vue.js
- 8 Filtros mixins
 - ◆ 8.1 Introducción a los mixins
 - ◆ 8.2 Utilización de mixins
 - ◆ 8.3 Cómo se mezclan los mixins
 - ◆ 8.4 Mixins globales
 - ◆ 8.5 Filtros globales
- 9 Formularios con Vue.js
 - ◆ 9.1 Input text y textarea
 - ◆ 9.2 Checkbox
 - ◆ 9.3 Radio
 - ◆ 9.4 Select
 - ◆ 9.5 Modificadores
 - ◆ 9.6 Cómo funciona la directiva v-model
 - ◆ 9.7 Añadiendo valores por defecto
 - ◆ 9.8 Envío de formularios
- 10 Animaciones y transiciones con Vue.js
 - ◆ 10.1 Introducción a animaciones y transiciones
 - ◆ 10.2 Explicación de transiciones en un único elemento
 - ◆ 10.3 Transiciones con clases CSS
 - ◆ 10.4 Implementando nuestra primera transición
 - ◆ 10.5 Especificando nombres a las transiciones
 - ◆ 10.6 Especificando clases de transiciones
 - ◆ 10.7 Implementación de una animación CSS específica
 - ◆ 10.8 Mezclando transiciones y animaciones
 - ◆ 10.9 Transiciones entre elementos
 - ◆ 10.10 Modos de transición
 - ◆ 10.11 Transición de elementos en la carga inicial de la página
 - ◆ 10.12 Transiciones con hooks Javascript
 - ◆ 10.13 Ignorando clases CSS
 - ◆ 10.14 Transiciones entre componentes dinámicos
 - ◆ 10.15 Transiciones entre múltiples elementos
 - ◆ 10.16 Transiciones entre elementos móviles
- 11 Enrutamiento en SPA (Single Page Applications)
 - ◆ 11.1 Introducción a SPA (Single Page Applications)
 - ◆ 11.2 Instalación de vue-router
 - ◆ 11.3 Activación del router
 - ◆ 11.4 Registro de rutas
 - ◆ 11.5 Renderización de componentes enrutados
 - ◆ 11.6 Modificación del modo de enrutamiento
 - ◆ 11.7 Ruta catch-all
 - ◆ 11.8 Moviendo rutas a un fichero
 - ◆ 11.9 Añadiendo enlaces de navegación
 - ◆ 11.10 Estilo del enlace activo en la navegación
 - ◆ 11.11 Rutas dinámicas matching y linking
 - ◆ 11.12 Rutas con nombre
 - ◆ 11.13 Recuperando parámetros en las rutas
 - ◆ 11.14 Uso de propiedades en las rutas
 - ◆ 11.15 Reaccionando a cambios en los parámetros
 - ◆ 11.16 Navegación programática

- ◆ 11.17 Navegando en el historial del navegador
- ◆ 11.18 Redireccionando
- ◆ 11.19 Alias
- ◆ 11.20 Rutas anidadas
- ◆ 11.21 Parámetros query
- ◆ 11.22 Fragmentos hash
- ◆ 11.23 Controlando el comportamiento del scroll
- ◆ 11.24 Vistas con nombre
- ◆ 11.25 Transición de rutas y demás
- 12 Conexión a servidores empleando HTTP
 - ◆ 12.1 Introducción al uso de HTTP en Vue
 - ◆ 12.2 Ajustes de vue-resource
 - ◆ 12.3 Obteniendo datos usando peticiones GET
 - ◆ 12.4 Plantillas URI
 - ◆ 12.5 Peticiones POST
 - ◆ 12.6 Uso de recursos
 - ◆ 12.7 Recursos específicos y acciones
 - ◆ 12.8 Configuración global
 - ◆ 12.9 Configuración con componentes
 - ◆ 12.10 Interceptores
- 13 Vuex
 - ◆ 13.1 Introducción a Vuex
 - ◆ 13.2 Por qué necesitamos Vuex
 - ◆ 13.3 Instalación de Vuex

Introducción a Vue.js



- Página oficial: <https://vuejs.org/>
- Vue (pronunciado /vju?/, como view) es un framework progresivo para construir interfaces de usuario. A diferencia de otros frameworks monolíticos, Vue está diseñado desde cero para ser utilizado incrementalmente.
- La librería central está enfocada solo en la capa de visualización, y es fácil de utilizar e integrar con otras librerías o proyectos existentes.
- Por otro lado, Vue también es perfectamente capaz de impulsar sofisticadas Single-Page Applications (SPA) cuando se utiliza en combinación con herramientas modernas y librerías de apoyo.
- La forma más fácil de probar Vue.js es usando un **ejemplo básico**:

```
<!DOCTYPE html>
<html>

<head>
  <title>Mi primera aplicación Vue</title>
  <script src="https://unpkg.com/vue"></script>
</head>

<body>
  <div id="app">
    {{ message }}
  </div>

  <script>
    var app = new Vue({
```

```

    el: '#app',
    data: {
      message: 'Hola Vue.js!'
    }
  })
</script>
</body>

</html>

```

- En el ejemplo básico hemos cargado Vue.js utilizando la librería remota, aunque también la podríamos tener alojada en nuestro propio servidor.
- Tenemos un div con el id app.
- Instanciamos la aplicación Vue, con **new Vue()** y en esa instancia creamos un objeto con las propiedades de la aplicación.
- En el momento de la instancia los datos y el DOM están vinculados, y todo es **reactivo**, es decir cualquier cambio en los datos se verán reflejados en el navegador en cualquier componente donde se estén utilizando.
 - ♦ Le indicamos el elemento (el) que va a contener la aplicación, en este caso #app.
 - ♦ Creamos una propiedad data que contendrá todas las variables de la aplicación.
- En el div #app, empleamos el **operador mustache {{ }}** para vincular datos en el div. Esta sintaxis básica se puede utilizar para vincular cualquier texto dentro de HTML (title, description, etc...)
- **La sintaxis {{ }} sin embargo no se puede utilizar en los atributos HTML tales como: href, id, src, etc.. Vue para ello proporciona el atributo nativo v-bind (conocido como directiva Vue).**

Fundamentos de Vue.js

Trabajando con plantillas

- En este ejemplo tenemos 2 propiedades nombre y edad.
- Se mostrará solamente el nombre, sin los apellidos.
- En la plantilla evaluamos la propiedad edad y según su valor se muestra Viejo o Joven.

```

<!DOCTYPE html>
<html lang="es">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Trabajando con plantillas en Vue.js</title>
  <script src="https://unpkg.com/vue"></script>
</head>

<body>
  <div id="app">
    <h1>{{ nombre.split(' ')[0] }}</h1>
    <h1>{{ edad > 60 ? 'Viejo' : 'Joven' }}</h1>

  </div>

  <script>
    var app = new Vue({
      el: '#app',
      data: {
        edad: 47,
        nombre: 'Rafa Veiga'
      }
    })
  </script>
</body>

</html>

```

Métodos

- Los métodos nos permitirán imprimir datos basados en ciertas reglas o lógica.
- Para ello son muy útiles las funciones.
- Lo usaremos en Vue.js con las propiedad **methods** de la instancia de Vue.js.

- La **propiedad methods** es un objeto cuyas **claves** son el **nombre de la función** y el **valor** es el **contenido de la función en si misma**.
- En el ejemplo tenemos el método **obtenerNombreCompleto** que podemos utilizarla directamente en `{{ }}`.

```
<!DOCTYPE html>
<html lang="es">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Trabajanco con métodos en Vue.js</title>
  <script src="https://unpkg.com/vue"></script>
</head>

<body>
  <div id="app">
    <h1>{{ obtenerNombreCompleto(nombre,apellidos) }}</h1>

  </div>

  <script>
    var app = new Vue({
      el: '#app',
      data: {
        nombre: 'Rafa',
        apellidos: 'Veiga'
      },
      methods: {
        obtenerNombreCompleto: function (primer, segundo) {
          // Podríamos acceder a las propiedades nombre y apellidos con
          // this.nombre o this.apellidos
          // si no quisiéramos pasarlas como parámetros.
          return primer + ' ' + segundo;

          // También se podría devolver un objeto con una nuevapropiedad
          /*
          return {
            nuevapropiedad: primer + ' ' + segundo
          }
          */

          // Atención no se pueden usar funciones arrow en los métodos,
          // ya que estas funciones están asociadas al contexto principal y no al contexto de Vue.js
          */
        }
      }
    });
  </script>
</body>

</html>
```

Introducción a directivas

Expresiones y directivas

Expresiones con directivas

Introducción a eventos

Paso de argumentos a eventos

Modificadores de directivas de eventos

Claves modificadoras

Modificadores clave

Binding en dos direcciones

Seguridad mostrando y escapando HTML

Renderizado elementos una sola vez

Mostrar y ocultar elementos

Ocultar elementos hasta que la instancia de Vue.js esté preparada

Recorrido de arrays

Recorrido de propiedades de un objeto

Recorrido en rango numérico

Explicación de actualizaciones usando v-for

Detección de cambios en array

Condicionales y bucles en binding bidireccional

Optimización con propiedades

Uso de getters y setters con propiedades

Watchers

Filtros

Uso de estilos CSS en línea

Accesos rápidos en bindings y eventos

Ejemplo de aplicación e-commerce con Vue.js

Profundizando en las instancias Vue.js

Acceso a una instancia Vue.js desde afuera de su declaración

Uso de múltiples instancias Vue.js en la misma página

Proxy

Entendiendo la reactividad

Cola de actualizaciones asíncronas

Entendiendo el DOM virtual

Añadiendo watchers de forma dinámica

Acceso al DOM con refs

Montaje de plantillas dinámicamente

Uso de plantillas inline

Destrucción de una instancia Vue.js

Más información sobre el ciclo de vida y hooks de una instancia Vue.js

Configuración de webpack con Vue.js

Introducción

Instalación de Chrome developer tools

Introducción al CLI de Vue.js

Creación de un proyecto Vue.js desde CLI

Explicación de la estructura del proyecto Vue.js

Componentes en un único fichero

Compilación para producción del proyecto Vue.js

Componentes

Introducción a componentes

Por qué la propiedad data debe ser una función

Componentes globales

Creación de un componente

Organizando componentes

Estilos globales

Transfiriendo datos a los componentes

Validación de datos recibidos

Trabajando con eventos

Comunicación a través de un evento bus

Slots

Slots con nombre

Componentes dinámicos

Cómo mantener componentes dinámicos activos

Ciclo de vida de los componentes dinámicos

Ejemplo de aplicación mail con Vue.js

Filtros mixins

Introducción a los mixins

Utilización de mixins

Cómo se mezclan los mixins

Mixins globales

Filtros globales

Formularios con Vue.js

Input text y textarea

Checkbox

Radio

Select

Modificadores

Cómo funciona la directiva v-model

Añadiendo valores por defecto

Envío de formularios

Animaciones y transiciones con Vue.js

Introducción a animaciones y transiciones

Explicación de transiciones en un único elemento

Transiciones con clases CSS

Implementando nuestra primera transición

Especificando nombres a las transiciones

Especificando clases de transiciones

Implementación de una animación CSS específica

Mezclando transiciones y animaciones

Transiciones entre elementos

Modos de transición

Transición de elementos en la carga inicial de la página

Transiciones con hooks Javascript

Ignorando clases CSS

Transiciones entre componentes dinámicos

Transiciones entre múltiples elementos

Transiciones entre elementos móviles

Enrutamiento en SPA (Single Page Applications)

Introducción a SPA (Single Page Applications)

Instalación de vue-router

Activación del router

Registro de rutas

Renderización de componentes enrutados

Modificación del modo de enrutamiento

Ruta catch-all

Moviendo rutas a un fichero

Añadiendo enlaces de navegación

Estilo del enlace activo en la navegación

Rutas dinámicas matching y linking

Rutas con nombre

Recuperando parámetros en las rutas

Uso de propiedades en las rutas

Reaccionando a cambios en los parámetros

Navegación programática

Navegando en el historial del navegador

Redireccionando

Alias

Rutas anidadas

Parámetros query

Fragmentos hash

Controlando el comportamiento del scroll

Vistas con nombre

Transición de rutas y demás

Conexión a servidores empleando HTTP

Introducción al uso de HTTP en Vue

Ajustes de vue-resource

Obteniendo datos usando peticiones GET

Plantillas URI

Peticiones POST

Uso de recursos

Recursos específicos y acciones

Configuración global

Configuración con componentes

Interceptores

Vuex

Introducción a Vuex

Por qué necesitamos Vuex

Instalación de Vuex

...