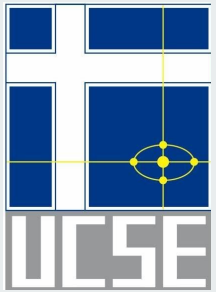
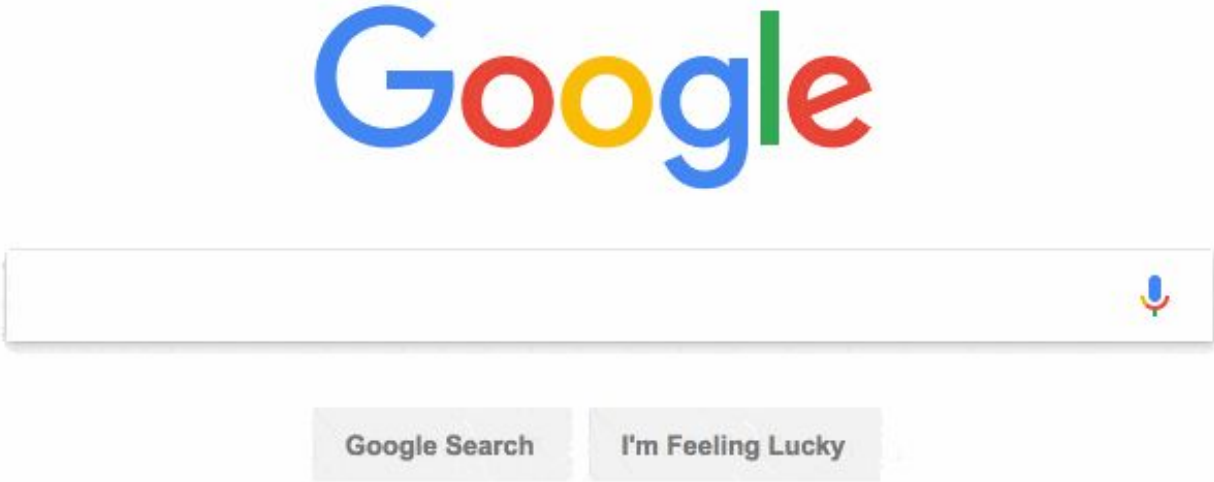
AJAX &
MVVM pattern - Vue.js
React

UCSE - SEIA



AJAX (Asynchronous Javascript And Xml)

UCSE - SEIA


Google
Google Search
I'm Feeling Lucky

AJAX (Asynchronous Javascript And Xml)

- “AJAX is a developer's dream”
- Permite enviar request's HTTP sin necesidad de actualizar el navegador!
- Utiliza funcionalidades que todo navegador implementa, como XMLHttpRequest, JavaScript y HTML DOM
- No es un lenguaje de programación ni una función de jQuery !
- Ejemplo

AJAX (Asynchronous Javascript And Xml)

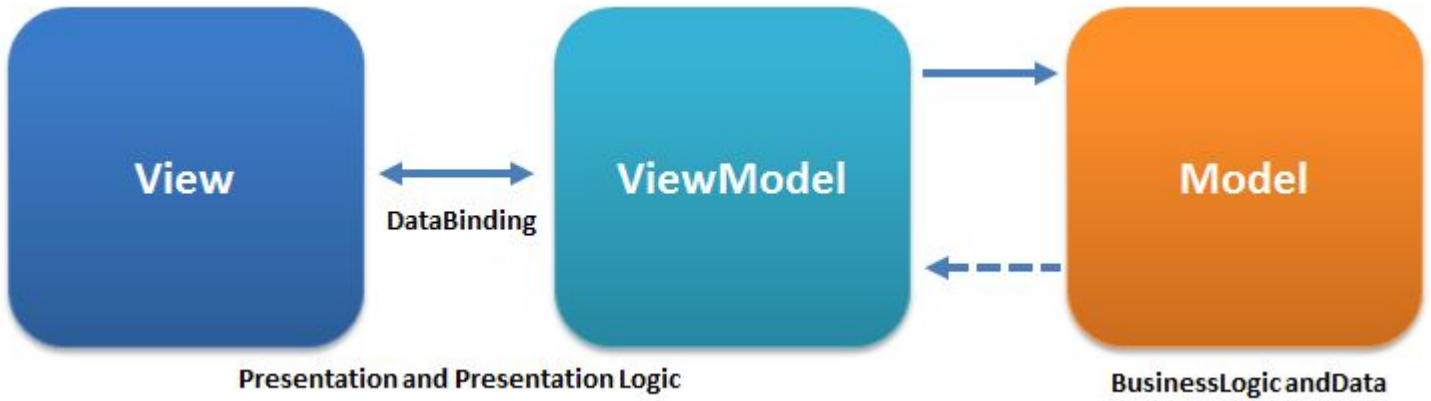
- Usos comunes:
  - Actualizar secciones de la página de forma "parcial"
  - Enviar información al servidor para realizar algún tipo de control
  - Hacer submit de un formulario en “background”
  - etc.
- Como todo, siempre es más fácil con jQuery: ejemplo

MVVM pattern - Vue.js

UCSE - SEIA

MVVM pattern (Model-View-ViewModel)

Patrón de diseño que busca separar lógica de aplicación de la capa de presentación.

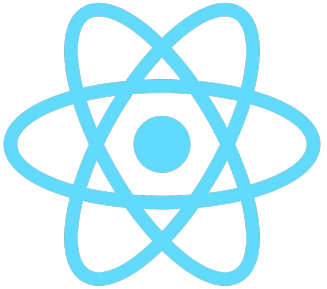


MVVM pattern (Model-View-ViewModel)

Aplicaciones + interactivas, + dinámicas, + complejas ...

MVVM pattern (Model-View-ViewModel)

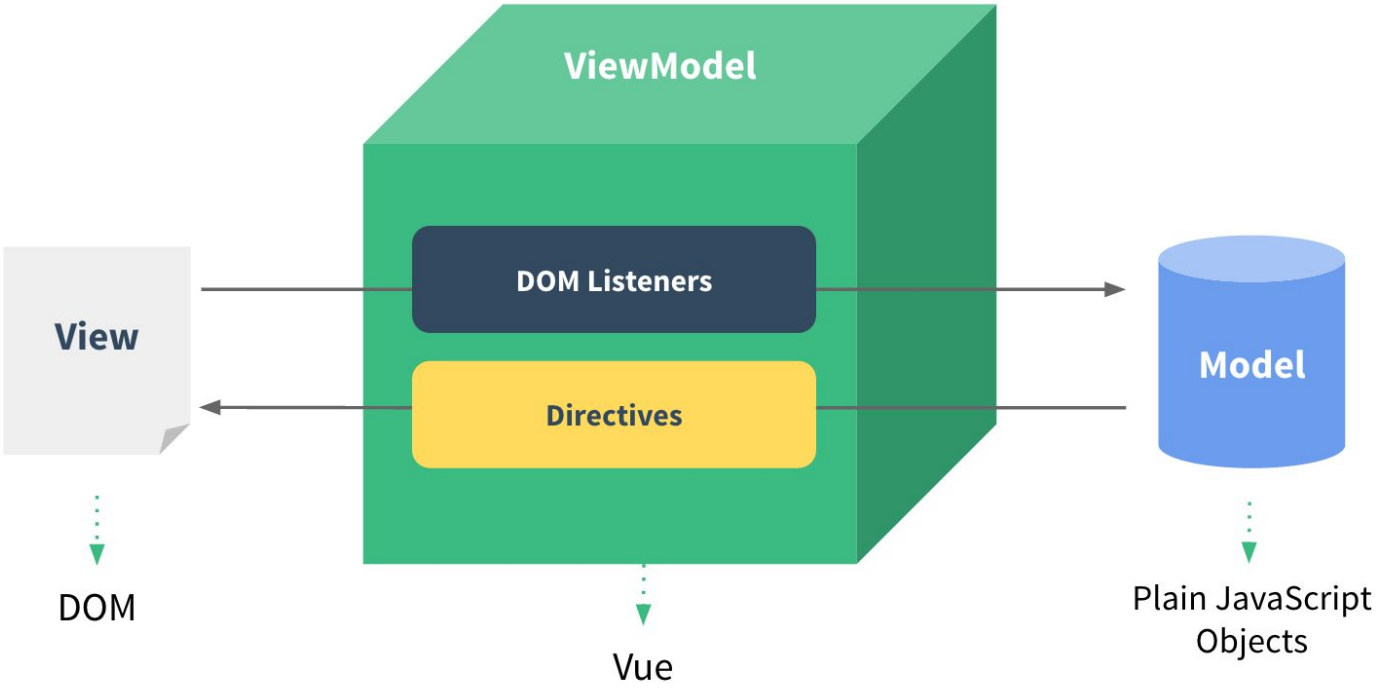
Surgen distintos frameworks/herramientas



Vue.js

- Es una librería para crear aplicaciones web interactivas
- Se concentra específicamente en el componente VM del patrón de diseño
- Se encarga de conectar la vista y el modelo de forma sencilla

Vue.js


ViewModel
DOM Listeners
Directives
View
Model
DOM
Vue
Plain JavaScript
Objects

Vue.js > ViewModel

- Es el encargado de sincronizar la vista y el modelo.
- Cada instancia de Vue es un ViewModel
- Se definen de la siguiente manera:

```JS
var vm = new Vue({ /* options */ })
```

Vue.js > View

- Es la parte o elemento del DOM que va a manejar el ViewModel.
- Dentro de dicho elemento, cuando se cree la instancia del VM, Vue recorre todos los elementos para realizar los “data bindings”.
- Se define como:

```js
vm.$el // The View
```

Vue.js > Model

- Es un objeto de JavaScript.
- Dicho objeto asignado a un VM se vuelve “reactivo”, y cada cambio que se produzca se ve reflejado “automáticamente” en la vista.
- Se definen como:

```JS
vm.$data // The Model
```

Vue.js > directives

- Atributos “custom” que se utilizan para notificar a Vue.js de algún comportamiento o lógica necesaria en relación a algún elemento del DOM.
- Ejemplo de sincronización entre la propiedad ***message*** del modelo y el texto contenido dentro de un elemento div:

HTML

```
1    <div v-text="message"></div>
```

Vue.js > Mustache Bindings

- Se pueden utilizar otro tipo de “bindings” más explícitos en los atributos de elementos HTML o en su interior (que se muestran como texto directamente).
- Ejemplo:

```HTML
<div id="person-{{id}}">Hello {{name}}!</div>
```

Vue.js > Filtros

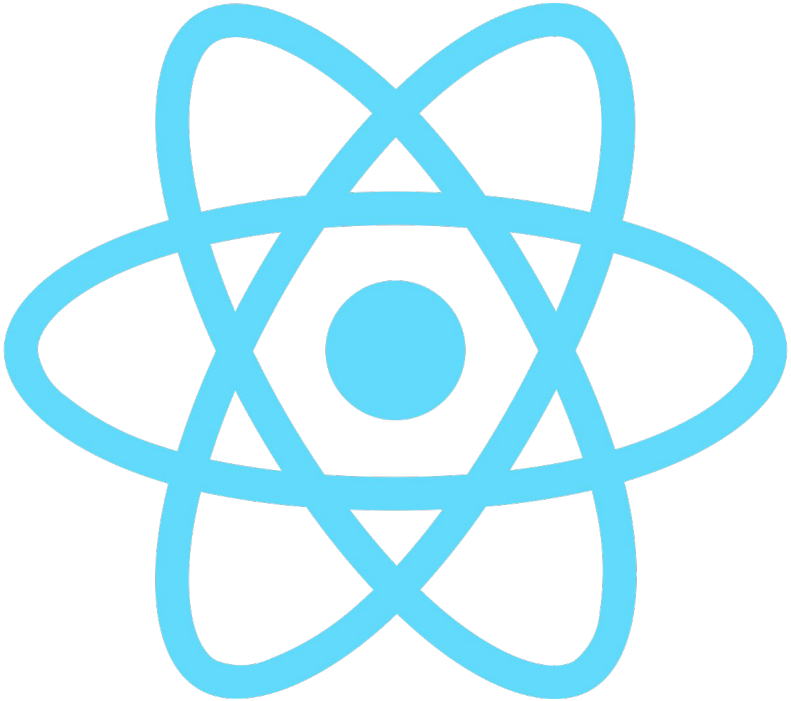
- Son funciones que se anteponen a la actualización de una vista en base a un valor del modelo.
- Ejemplo:

```HTML
<div>{{message | capitalize}}</div>
```

EJEMPLO !

React

UCSE - SEIA



React

- Una biblioteca de JavaScript para construir interfaces de usuario (mismo objetivo que Vue!)
- Desarrollada por Facebook (MIT licence)
- Declarativa y basada en componentes
- Trata de integrar la separación natural que existe entre HTML y JS, para facilitarle la vida a los desarrolladores

React > componentes

- Son elementos que van a ser mostrados al usuario.

```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Hola {this.props.name}
      </div>
    );
  }
}

ReactDOM.render(
  <HelloMessage name="gente de IW !" />,
  document.getElementById('hello-example')
);
```

Hola gente de IW !

React > JSX

- Es una extensión de la sintaxis de JS para ser utilizada con React.
- No es necesario utilizarla, pero suele ser más simple.
- Permite combinar HTML y JS para definir componentes.

```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Hola {this.props.name}
      </div>
    );
  }
}
```

React > componentes + estado

- Cada componente puede manejar un estado interno, que le permite guardar info, accesibles mediante **`this.state`**.

Se invoca cada vez que se produzca un cambio en el estado del componente.

```
class HelloMessage extends React.Component {
  constructor(props) {
    super(props);
    this.state = { message: props.m };
  }

  render() {
    return (<div>Hola {this.state.message }</div>);
  }
}

ReactDOM.render(
  <HelloMessage m="gente de IW !" />,
  document.getElementById('hello-example')
);
```

Hola gente de IW !

React

EJEMPLO !

AJAX &
MVVM pattern - Vue.js
React

UCSE - SEIA