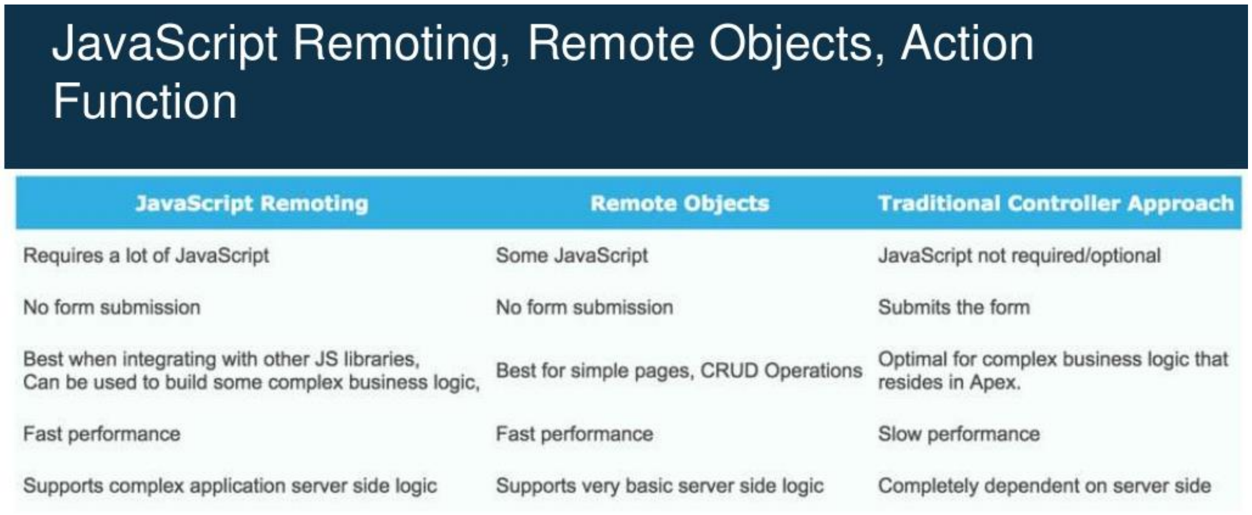
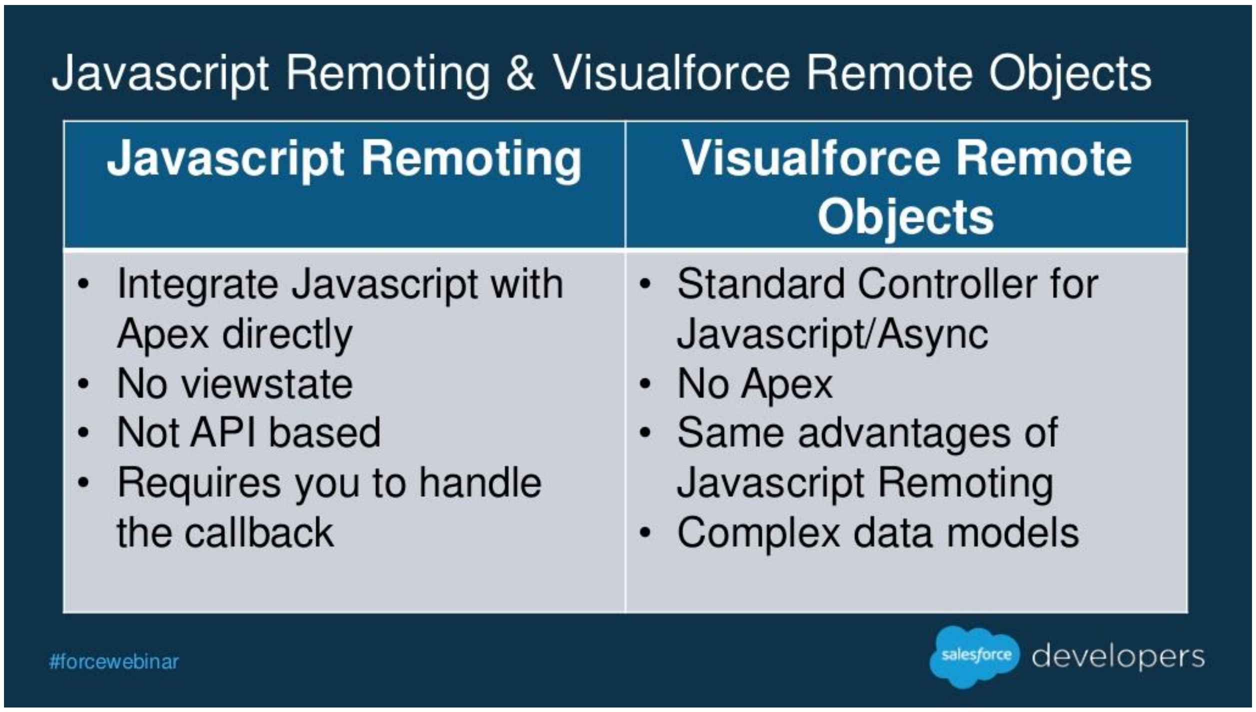


JavaScript Remoting, Remote Objects, Action Function

JavaScript Remoting	Remote Objects	Traditional Controller Approach
Requires a lot of JavaScript	Some JavaScript	JavaScript not required/optional
No form submission	No form submission	Submits the form
Best when integrating with other JS libraries, Can be used to build some complex business logic,	Best for simple pages, CRUD Operations	Optimal for complex business logic that resides in Apex.
Fast performance	Fast performance	Slow performance
Supports complex application server side logic	Supports very basic server side logic	Completely dependent on server side

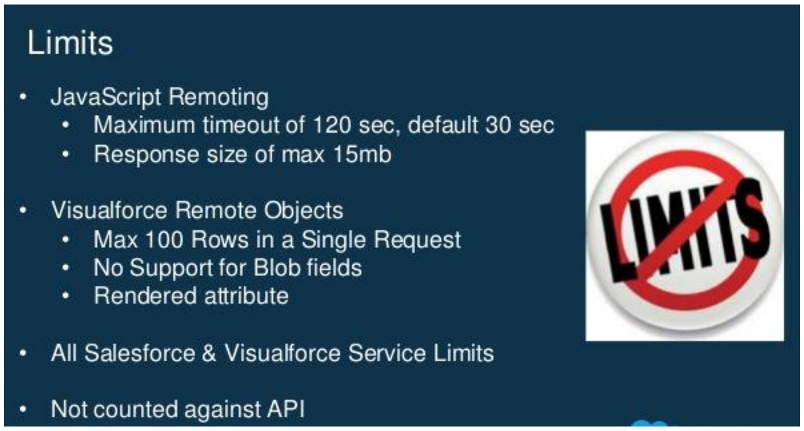


Javascript Remoting & Visualforce Remote Objects

Javascript Remoting	Visualforce Remote Objects
• Integrate Javascript with Apex directly • No viewstate • Not API based • Requires you to handle the callback	• Standard Controller for Javascript/Async • No Apex • Same advantages of Javascript Remoting • Complex data models

#forcewebinar

salesforce developers



Limits

- JavaScript Remoting
  - Maximum timeout of 120 sec, default 30 sec
  - Response size of max 15mb
- Visualforce Remote Objects
  - Max 100 Rows in a Single Request
  - No Support for Blob fields
  - Rendered attribute
- All Salesforce & Visualforce Service Limits
- Not counted against API

Below are the JavaScript Remoting Limits:

Remote call`s default response time is 30 seconds. If remote call taking longer time than 30 seconds then you will get a time out exception.
If your request requires more time to complete the transaction, then configure a longer timeout. You can setup it to 120 seconds.
Most of the interviewer asks the question, what is the maximum response size of your remote call ?
Answer is : maximum 15 MB.

Why do we need "Visualforce Remote Objects" when we already have "JavaScript Remoting" ?

Well, here are few advantages of "Visualforce Remote Objects" :

1. No need to write Controllers, Everything can be done in Visualforce only.
2. As *@RemoteAction* annotated methods needs to be static so you had to take special precaution as it didn't supported Viewstate. This hurdle is completely removed now.
3. No need to write Test Class now, as no Controller is involved.

Javascript remoting in salesforce can be useful in making a call to controller method without posting the whole form(<apex:form>) to server but whereas visualforce ajax functions(action function, action support, action poller) post the form.

Javascript remoting in salesforce let's you pass parameter and it provides a callback function in which its parameter contains result returned back from apex method. **JavaScript remoting** allows you to run asynchronous actions by decoupling the page from the controller and to perform tasks on the page without having to reload the entire page, Apex ***@RemoteAction*** methods must be static and either global or public and requires you to write some JavaScript

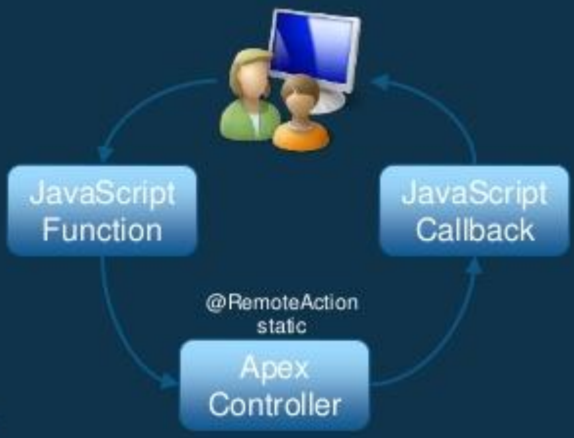
Javascript remoting can ensure that we are passing the needed data each time when we make a call and it is the most efficient way of calling and passing the data.

Javascript remoting is more interactive and dynamic than visualforce pages and it is optimized for use on mobile pages and pages that use third party javascript libraries.

As it is **asynchronous**, only needed data is loaded and displayed on the page and later we can do lazy loading to display additional data on the page which is not required to load immediately.

JavaScript Remoting

- A Stateless way to call Apex from JavaScript



#forcewebinar

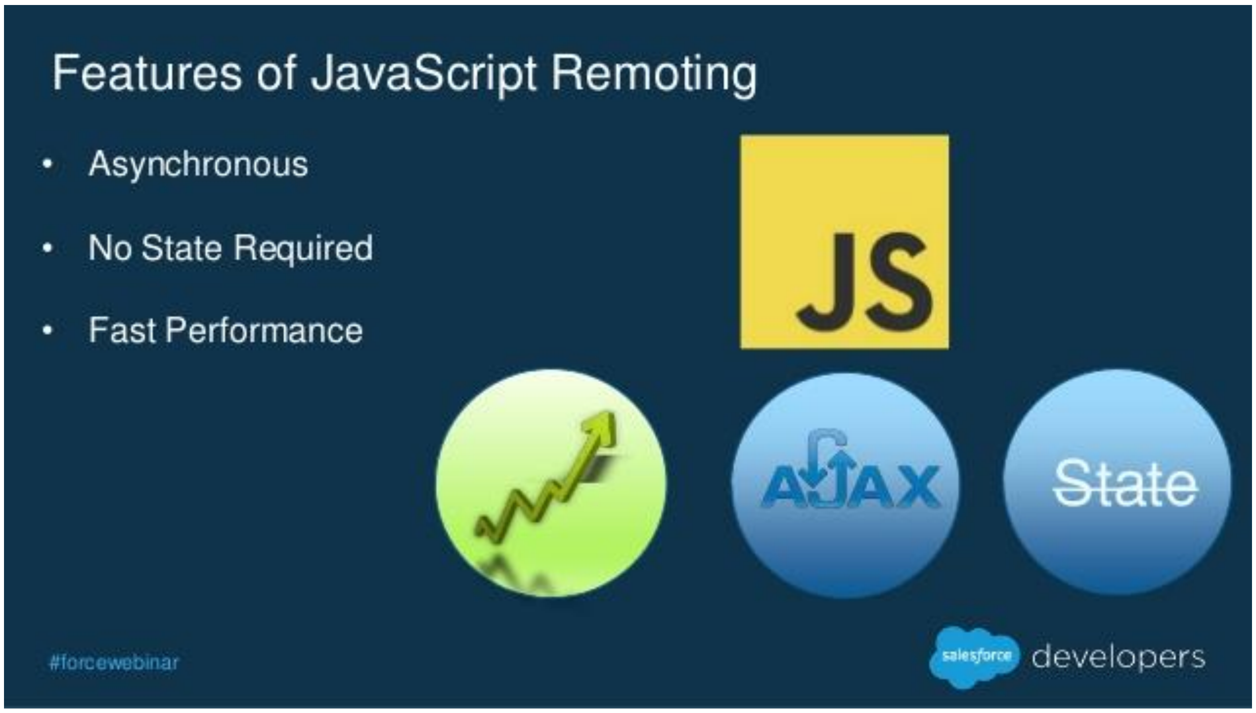
salesforce developers

JavaScript Remoting Use cases

- Slick & Fast User Interfaces
- Third party JavaScript Libraries or Frameworks
- Mobile Applications
- Single Page Applications
- JavaScript Guru

#forcewebinar

salesforce developers



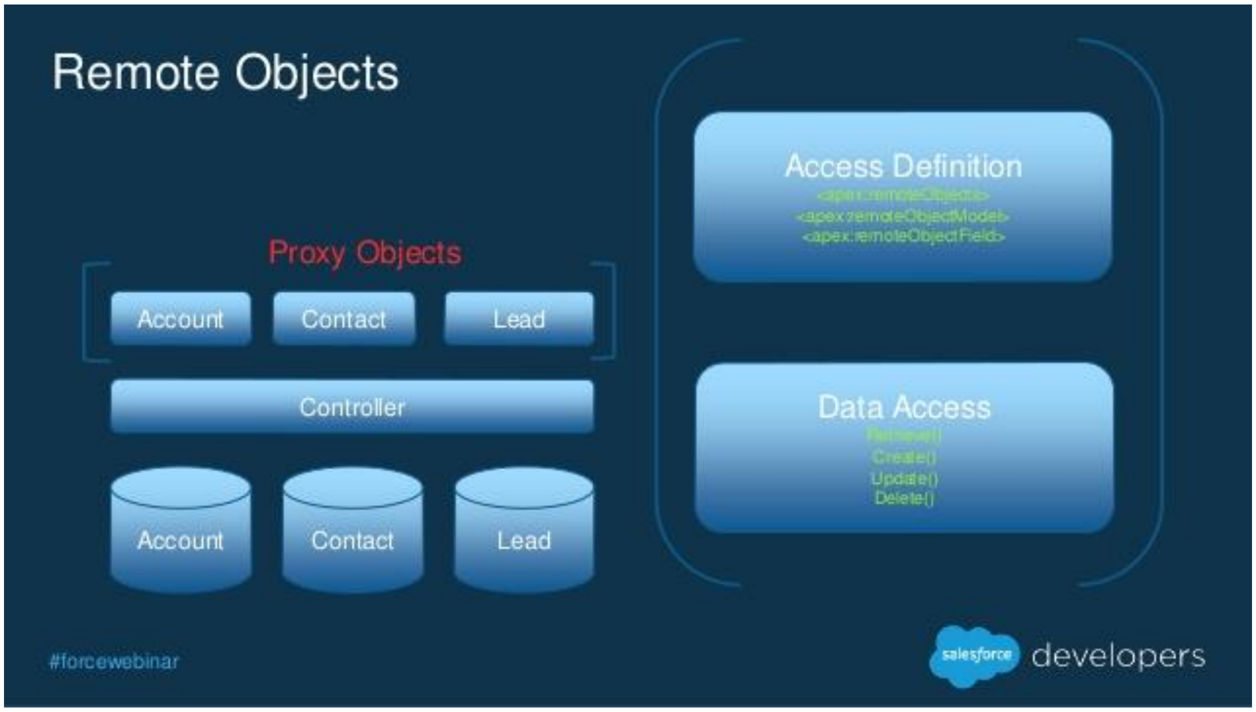
Features of JavaScript Remoting

- Asynchronous
- No State Required
- Fast Performance

JS
AJAX
~~State~~

#forcewebinar

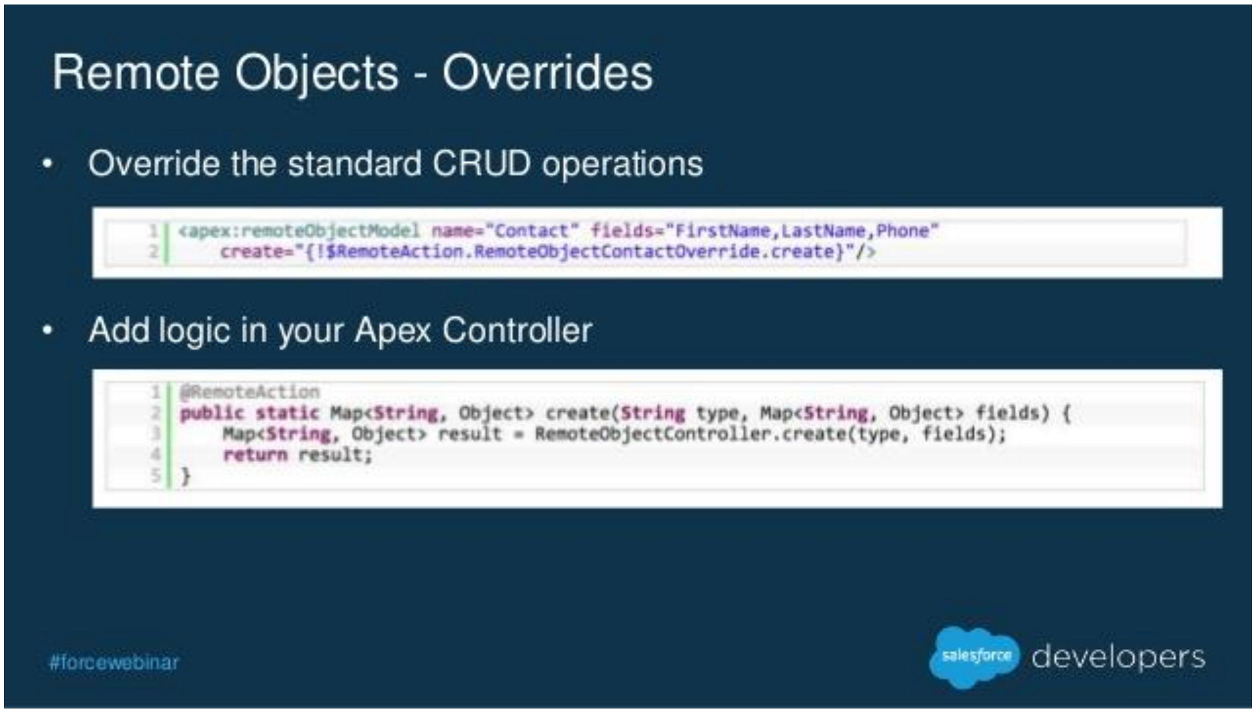
salesforce developers



Remote Objects

#forcewebinar

salesforce developers



Remote Objects - Overrides

- Override the standard CRUD operations

```
<apex:remoteObjectModel name="Contact" fields="FirstName,LastName,Phone"
    create="{!$RemoteAction.RemoteObjectContactOverride.create}"/>
```

- Add logic in your Apex Controller

```
@RemoteAction
public static Map<String, Object> create(String type, Map<String, Object> fields) {
    Map<String, Object> result = RemoteObjectController.create(type, fields);
    return result;
}
```

#forcewebinar

salesforce developers