How to refresh getRecord @wire adapter in Lightning Web Component

Hii there,
While developing in Lightning Web Component we often use @wire decorator with getRecord wire adapter to fetch the 'record' on which this component is being rendered. Once you wire getRecord with the current 'record' it detects any changes made on that record 'directly' (by directly I mean making changes using inline edit on record detail page).

The Loophole

Though the wire adapter can detect changes made on that record directly,it often fails to update itself if any change is made on Parent record which should reflect in Child record (Assuming that the Lightning Web Component is being rendered on child record). Especially when the field referenced in getRecord 'fields' parameter is a **Formula Field** which has to be updated whenever parent record is updated.

The Root Cause

So in a very simple terms, a formula field is more like a 'view' which gives you the result at the runtime hence you need to go back and refresh the child record every time,also this will only works if the formula field is present on the page layout.

How to refresh/update getRecord wire adaptor?

Though there is no standard way provided by Salesforce,unlike refreshApex,to refresh getRecord but I'll show you a work around by which we can refresh getRecord automatically.

Prerequisites

Should know about basic working of @wrie and getRecord.

The Solution

Scenario:
Lets say our Lightning Web Component is rendering on Opportunity record and there is a lookup 'AccountId' from Opportunity => Account.

Initial Setup:
We'll create a Lightning Web Component as below and an Apex class called 'Controller'.

```
force-app ▸ main ▸ default ▸ lwc ▸ eSF_PricingPortal ▸ JS Example.js ▸ ...
1  import {LightningElement,api,wire} from 'lwc';
2  import {getRecord,updateRecord} from 'lightning/uiRecordApi';
3  import GET_PARENT_RECORD from '@salesforce/apex/Controller.getParent';
4
5  export default class Example extends LightningElement {
6      @api recordId;
7      @api parentRecord;
8
9      connectedCallback()
10     {
11         GET_PARENT_RECORD({recordId:this.recordId}).then(result=>{
12             this.parentRecord=result;
13         });
14
15     }
16
17     @wire(getRecord,{recordId:'$recordId',fields:'Opportunity.Name'})
18     child({error,data})
19     {
20         if(data)
21         {
22             console.log('updated');
23         }
24     }
25 }
```

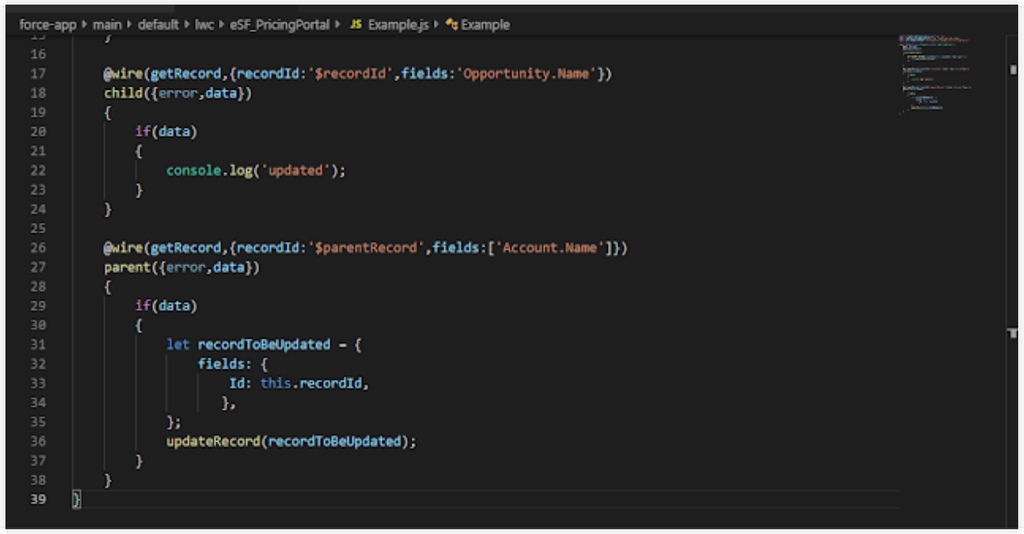
In the above component we have imported **getRecord** and **updateRecord** , apart from this we have imported **getParent** apex method as **GET_PARENT_RECORD** which will return us the parend record id. The code snippet is shown below

```
@AuraEnabled
    public static string getParent(string recordId)
    {
        Opportunity query=[SELECT AccountId From Opportunity WHERE id=:recordId];
        return query.AccountId;

    }
```

We will call this method in the **connectedCallback** lifecycle hook and update the parentRecord property with the parent id as shown in the Lightning Web Component.

Now we will wire getRecord with the current record on which it is being rendered. In our case this will be **Opportunity**record.This completes the initial setup,now we will add one more @wire(getRecord) for the parent Record as shown below.



With this we have wired Opportunity (child) record and Account (parent) record separately hence any changes made on any of them will be detected by their wired property. But to detect change in parent record and refresh **getRecord** of child notice that we are using **updateRecord** which we imported earlier.

In the recordToBeUpdated property,under fields parameter we just specify the recordId of Opportunity (child) and call updateRecord with this property as parameter.

Now whenever the parent record is updated, it will tend to update the child record which will refresh getRecord adapter of child with new data and further logic can be performed.

Conclusions:

1. This technique can be used in case when fields like Formula Fields or Roll-Up Summary fields have to be tracked inside getRecord.

2. With adding custom logic we can reduce the number of callouts to the apex methods to get updated data everytime.

3. This also works well in case the Formula Fields or Roll-Up Summary Fields are not to be shown on page layout.