Transforming Data

There are some cases where you may want to modify data as it's fetched from WPGraphQL but before it's stored in Gatsby. This source plugin does that quite a bit itself for html transformations and image processing. Because it's such a common need for implementing advanced use cases like automatic image optimization and building transformer plugins, we have a built in API for this (`options.type.[typename].beforeChangeNode`). This API is called before any node is created, updated, or deleted.

Modifying nodes when created or updated

A good example on how to use this API is by referencing internal uses of it. In `src/models/gatsby-api.ts` you will see the following code:

```
const defaultPluginOptions = {
  // ...
  type: {
    MediaItem: {
      beforeChangeNode: async ({ remoteNode, actionType, typeSettings }) => {
        // we fetch lazy nodes files in resolvers, no need to fetch them here.
        if (typeSettings.lazyNodes) {
          return {
            remoteNode,
          }
        }

        if (
          actionType === `CREATE_ALL` ||
          actionType === `CREATE` ||
          actionType === `UPDATE`
        ) {
          const createdMediaItem = await createLocalFileNode({
            mediaItemNode: remoteNode,
            parentName: `Node action ${actionType}`,
          })

          if (createdMediaItem) {
            remoteNode.localFile = {
              id: createdMediaItem.id,
            }

            return {
              remoteNode,
            }
          }
        }

        return {
          remoteNode,
        }
      },
```

```
    },
  },
}
```

As you can see, `beforeChangeNode` allows us to either leave a node as-is, modify it, or perform some related side effects before returning. The above code example fetches files from WPGraphQL and creates local Gatsby nodes on every `CREATE` or `UPDATE` action for the node being processed. After creating a local file node, we save the id of that node so that we can later link this node to the file node we created.

Performing side-effects before deleting a node

A good example of why you might want to perform some side effects before a node is deleted can be seen in our internal usage of this API for Menu nodes:

```
export const menuBeforeChangeNode = async api => {
  if (api.remoteNode && api.actionType === `DELETE`) {
    const {
      pluginOptions,
      helpers: { getNodesByType, actions },
    } = api.helpers

    // get all existing MenuItem nodes
    const allMenuItems = getNodesByType(
      `${pluginOptions.schema.typePrefix}MenuItem`
    )

    // find the nodes that are children of the current menu
    const allMenuItemsNodesWithThisMenuIdAsAParent = allMenuItems.filter(
      menuItemNode => menuItemNode.menu.node.id === api.remoteNode.id
    )

    // delete each child menu item
    allMenuItemsNodesWithThisMenuIdAsAParent?.forEach(menuItemNode =>
      actions.deleteNode(menuItemNode)
    )
  }
}
```

Here we're finding all child MenuItem's on the MenuItem being deleted and then deleting those items. We do this as a performance optimization. In WP if a Menu is deleted, the child MenuItem's are also deleted. Instead of sending an event for each item and needing Gatsby to fetch all of them, we know that when we receive an event to delete a Menu, we can safely delete its child MenuItem's.

Cancelling an update

There may be cases where you want to cancel a node create/update/delete before it happens. You can do that by adding `cancelUpdate: true` to the object you return to this API. A common use-case for doing this is when you have a multi-lingual site split across multiple domains where each language has a separate Gatsby build process.

```
const beforeChangeNodePage = ({ remoteNode, actionType }) => {
  if (
    [`CREATE`, `CREATE_ALL`, `UPDATE`].includes(actionType) &&
    remoteNode.language !== process.env.BUILD_LANGUAGE
  ) {
    return {
      cancelUpdate: true,
    }
  }
}
```

Now each build will only receive updates for the language specified in the `BUILD_LANGUAGE` env variable and updates for nodes of other languages will be cancelled.