# Taxonomy File Explorer

A file explorer like view with SPFx and Managed Metadata

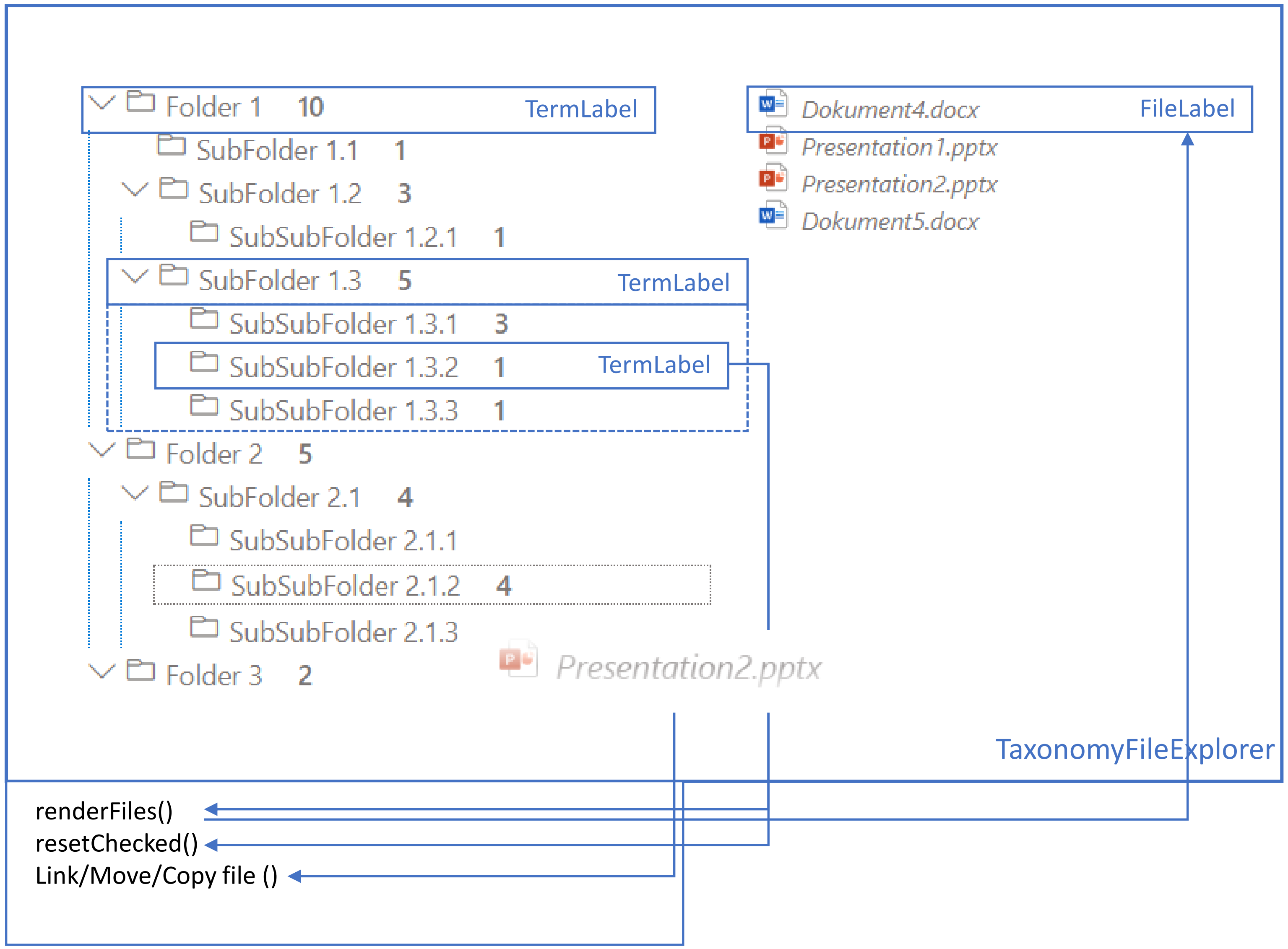
## About me

- Markus Moeller
- Microsoft 365 Developer Expert
- Microsoft MVP
- Avanade Germany
- @moeller2_0
- https://mmsharepoint.wordpress.com
- Proud dad of 1 (2yrs)

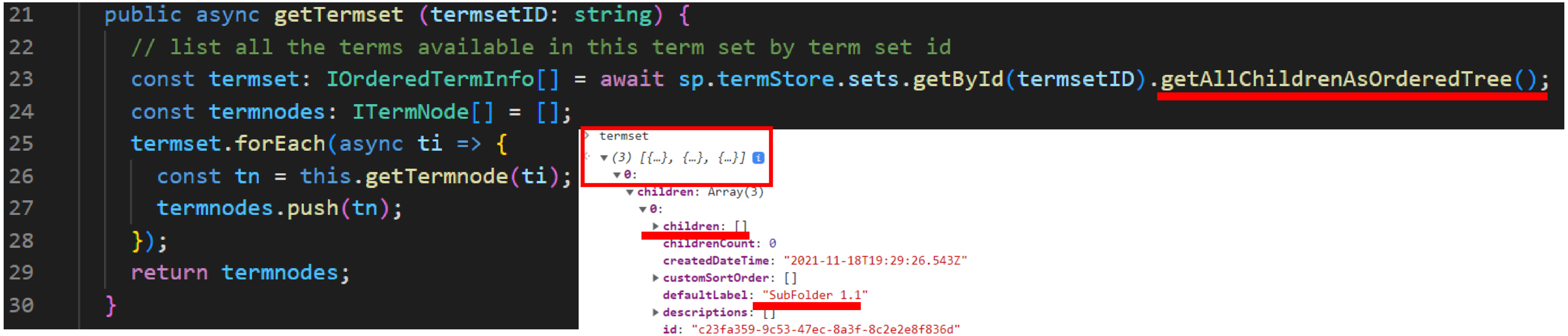
Render and manipulate files in a hierarchical view based on Managed Metadata and Drag&Drop

With SPFx

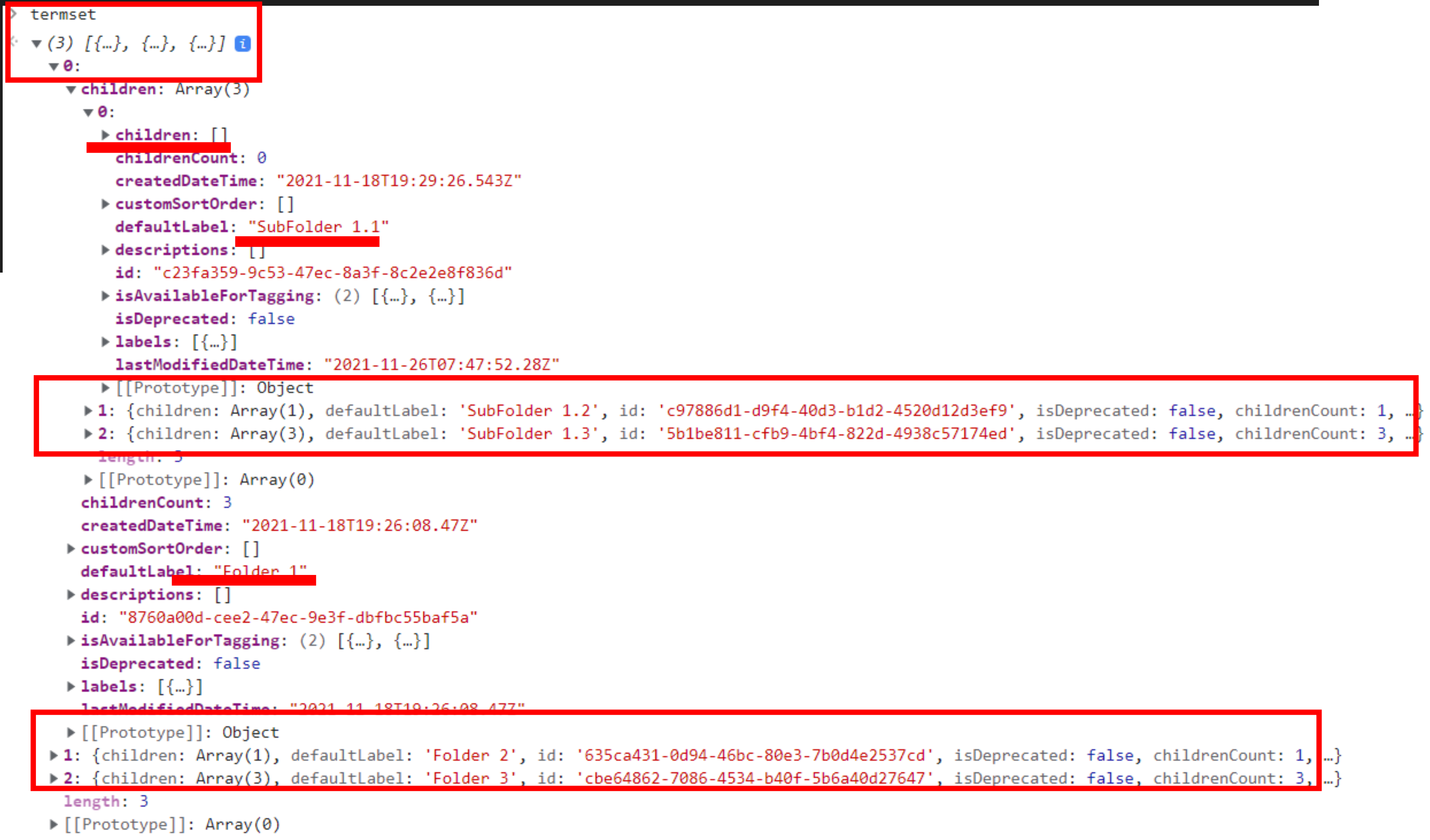
# React components [Structure]



# Retrieve Terms(et) by PnPJS (ordered!)

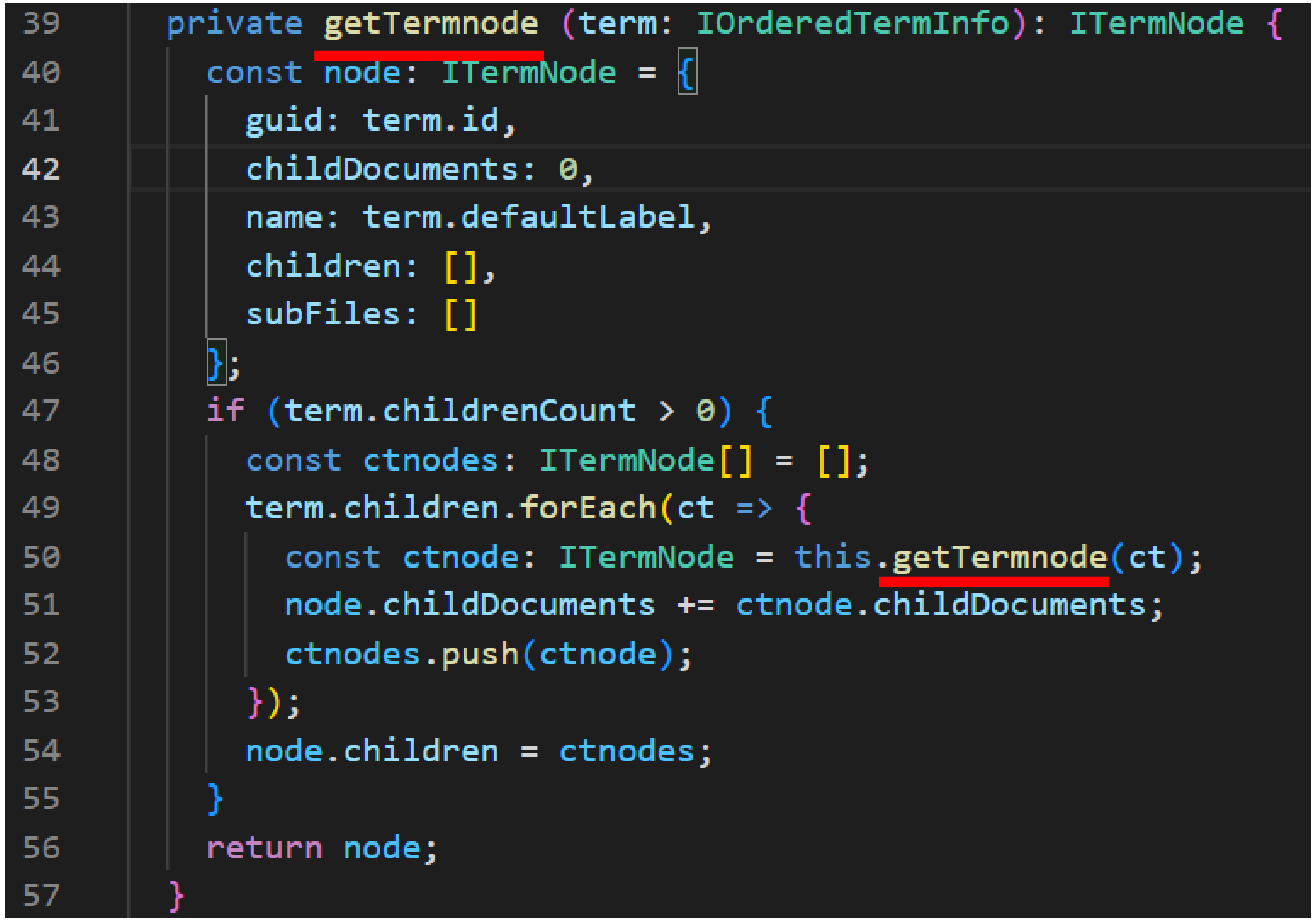


```
public async getTermset (termsetID: string) {
  // list all the terms available in this term set by term set id
  const termset: IOrderedTermInfo[] = await sp.termStore.sets.getById(termsetID).getAllChildrenAsOrderedTree();
  const termnodes: ITermNode[] = [];
  termset.forEach(async ti => {
    const tn = this.getTermnode(ti);
    termnodes.push(tn);
  });
  return termnodes;
}
```



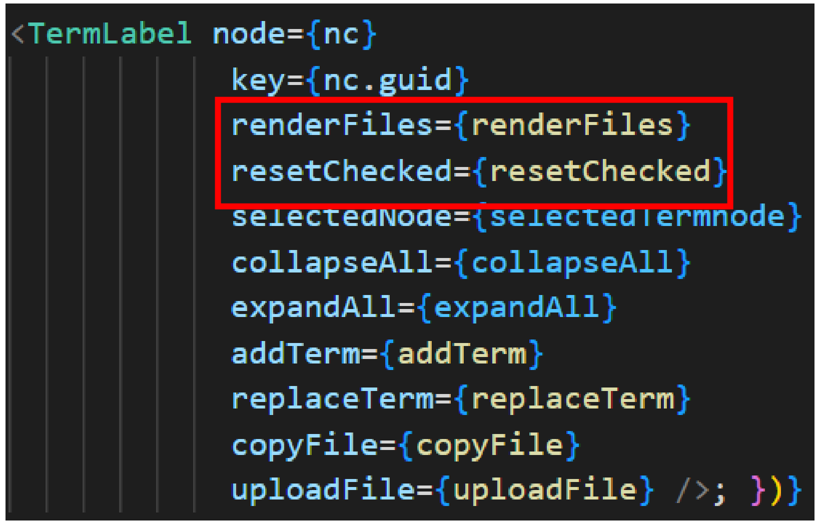
```
termset
▼(3) [{…}, {…}, {…}]
  ▼0:
    ▼children: Array(3)
      ▼0:
        ▶children: []
         childrenCount: 0
         createdDateTime: "2021-11-18T19:29:26.543Z"
        ▶customSortOrder: []
         defaultLabel: "SubFolder 1.1"
        ▶descriptions: []
         id: "c23fa359-9c53-47ec-8a3f-8c2e2e8f836d"
        ▶isAvailableForTagging: (2) [{…}, {…}]
         isDeprecated: false
        ▶labels: [{…}]
         lastModifiedDateTime: "2021-11-26T07:47:52.28Z"
        ▶[[Prototype]]: Object
      ▶1: {children: Array(1), defaultLabel: 'SubFolder 1.2', id: 'c97886d1-d9f4-40d3-b1d2-4520d12d3ef9', isDeprecated: false, childrenCount: 1, …}
      ▶2: {children: Array(3), defaultLabel: 'SubFolder 1.3', id: '5b1be811-cfb9-4bf4-822d-4938c57174ed', isDeprecated: false, childrenCount: 3, …}
       length: 3
      ▶[[Prototype]]: Array(0)
     childrenCount: 3
     createdDateTime: "2021-11-18T19:26:08.47Z"
    ▶customSortOrder: []
     defaultLabel: "Folder 1"
    ▶descriptions: []
     id: "8760a00d-cee2-47ec-9e3f-dbfbc55baf5a"
    ▶isAvailableForTagging: (2) [{…}, {…}]
     isDeprecated: false
    ▶labels: [{…}]
     lastModifiedDateTime: "2021-11-18T19:26:08.47Z"
    ▶[[Prototype]]: Object
  ▶1: {children: Array(1), defaultLabel: 'Folder 2', id: '635ca431-0d94-46bc-80e3-7b0d4e2537cd', isDeprecated: false, childrenCount: 1, …}
  ▶2: {children: Array(3), defaultLabel: 'Folder 3', id: 'cbe64862-7086-4534-b40f-5b6a40d27647', isDeprecated: false, childrenCount: 3, …}
   length: 3
  ▶[[Prototype]]: Array(0)
```

# Build tree of terms

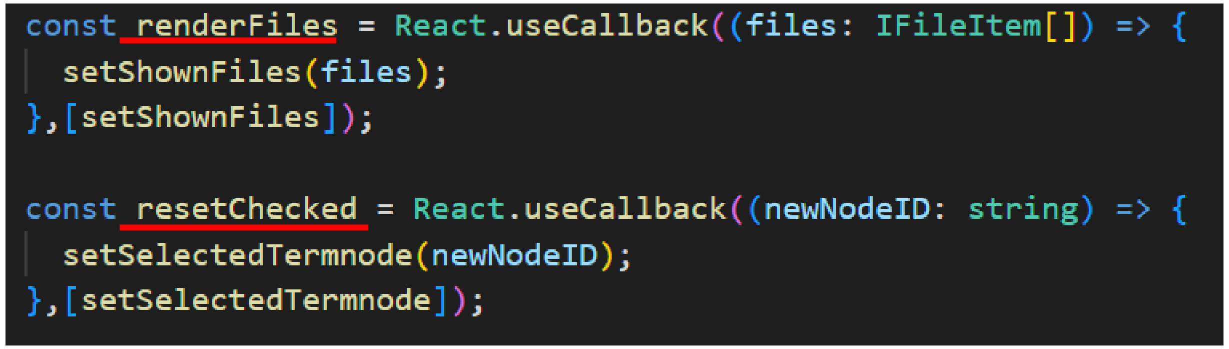


```typescript
private getTermnode (term: IOrderedTermInfo): ITermNode {
  const node: ITermNode = {
    guid: term.id,
    childDocuments: 0,
    name: term.defaultLabel,
    children: [],
    subFiles: []
  };
  if (term.childrenCount > 0) {
    const ctnodes: ITermNode[] = [];
    term.children.forEach(ct => {
      const ctnode: ITermNode = this.getTermnode(ct);
      node.childDocuments += ctnode.childDocuments;
      ctnodes.push(ctnode);
    });
    node.children = ctnodes;
  }
  return node;
}
```

# Parent Comp. Callbacks: Click a node

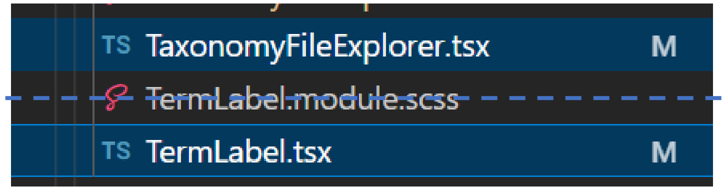


```
<TermLabel node={nc}
           key={nc.guid}
           renderFiles={renderFiles}
           resetChecked={resetChecked}
           selectedNode={selectedTermnode}
           collapseAll={collapseAll}
           expandAll={expandAll}
           addTerm={addTerm}
           replaceTerm={replaceTerm}
           copyFile={copyFile}
           uploadFile={uploadFile} />; })}
```

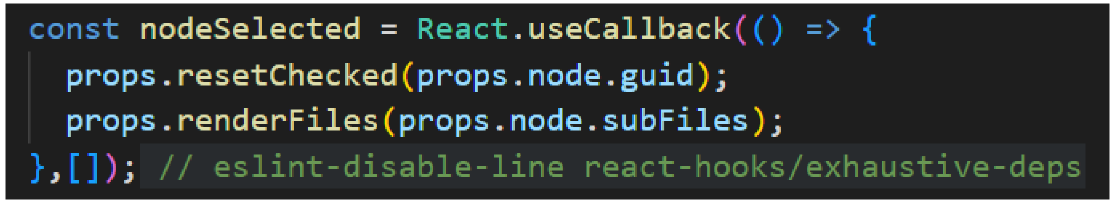


```
const renderFiles = React.useCallback((files: IFileItem[]) => {
  setShownFiles(files);
},[setShownFiles]);

const resetChecked = React.useCallback((newNodeID: string) => {
  setSelectedTermnode(newNodeID);
},[setSelectedTermnode]);
```

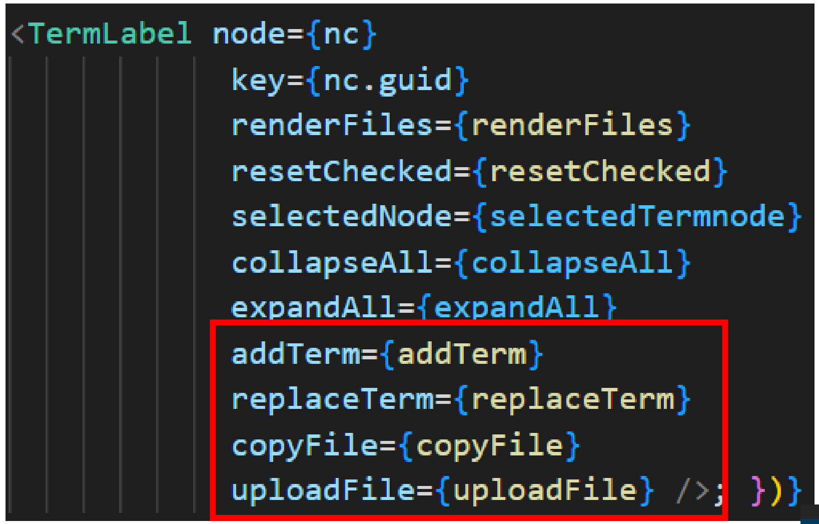


```
TS TaxonomyFileExplorer.tsx    M
   TermLabel.module.scss
TS TermLabel.tsx               M
```

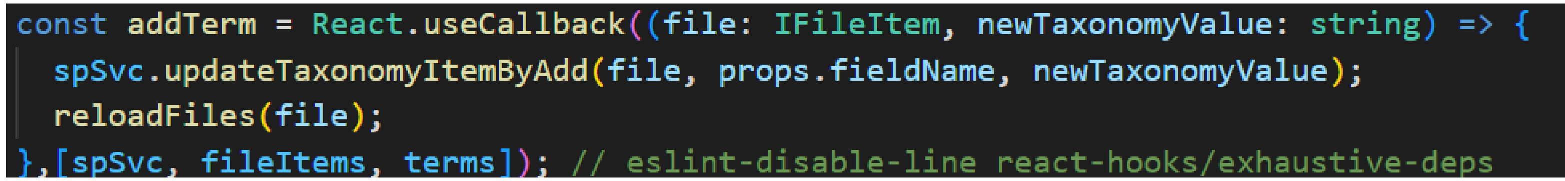


```
const nodeSelected = React.useCallback(() => {
  props.resetChecked(props.node.guid);
  props.renderFiles(props.node.subFiles);
},[]); // eslint-disable-line react-hooks/exhaustive-deps
```

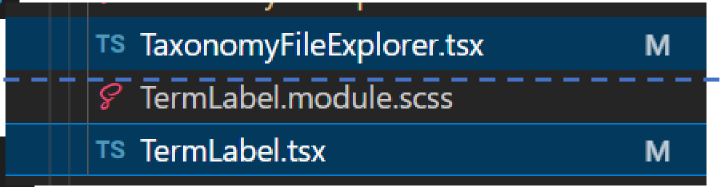
# Parent Comp. Callbacks: Term operation

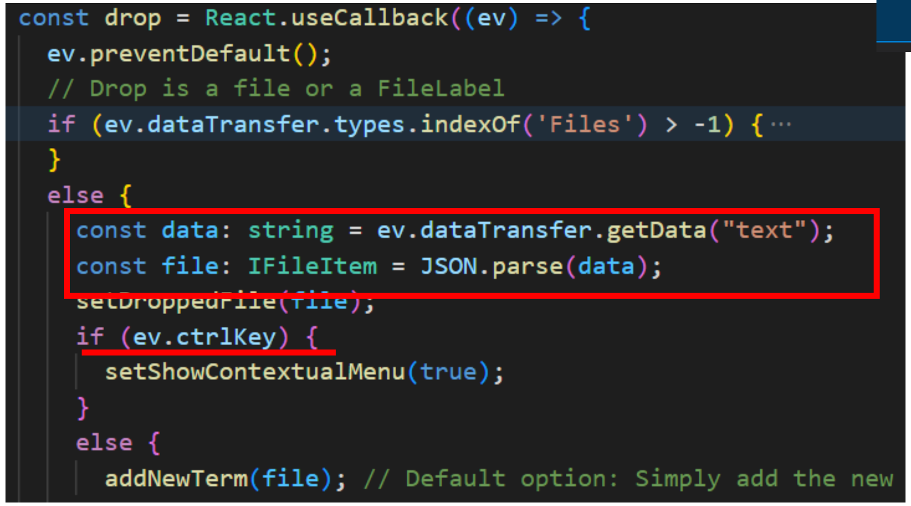


```
<TermLabel node={nc}
           key={nc.guid}
           renderFiles={renderFiles}
           resetChecked={resetChecked}
           selectedNode={selectedTermnode}
           collapseAll={collapseAll}
           expandAll={expandAll}
           addTerm={addTerm}
           replaceTerm={replaceTerm}
           copyFile={copyFile}
           uploadFile={uploadFile} />; })}
```

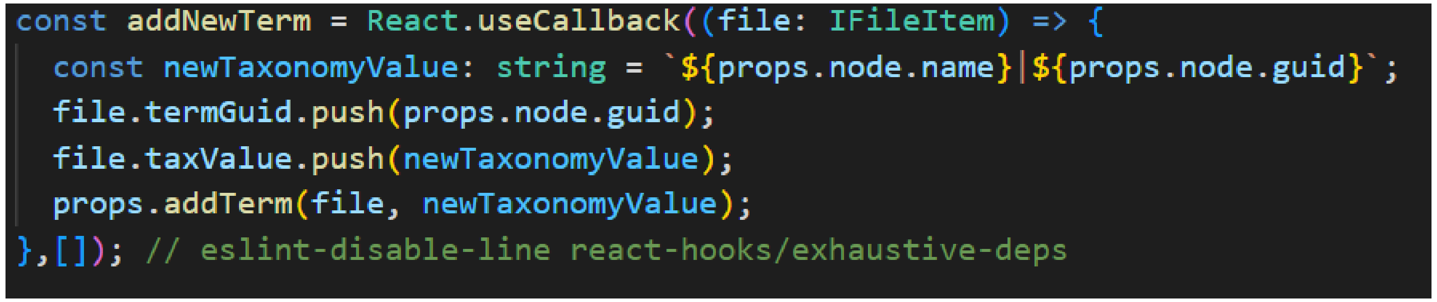


```
const addTerm = React.useCallback((file: IFileItem, newTaxonomyValue: string) => {
  spSvc.updateTaxonomyItemByAdd(file, props.fieldName, newTaxonomyValue);
  reloadFiles(file);
},[spSvc, fileItems, terms]); // eslint-disable-line react-hooks/exhaustive-deps
```



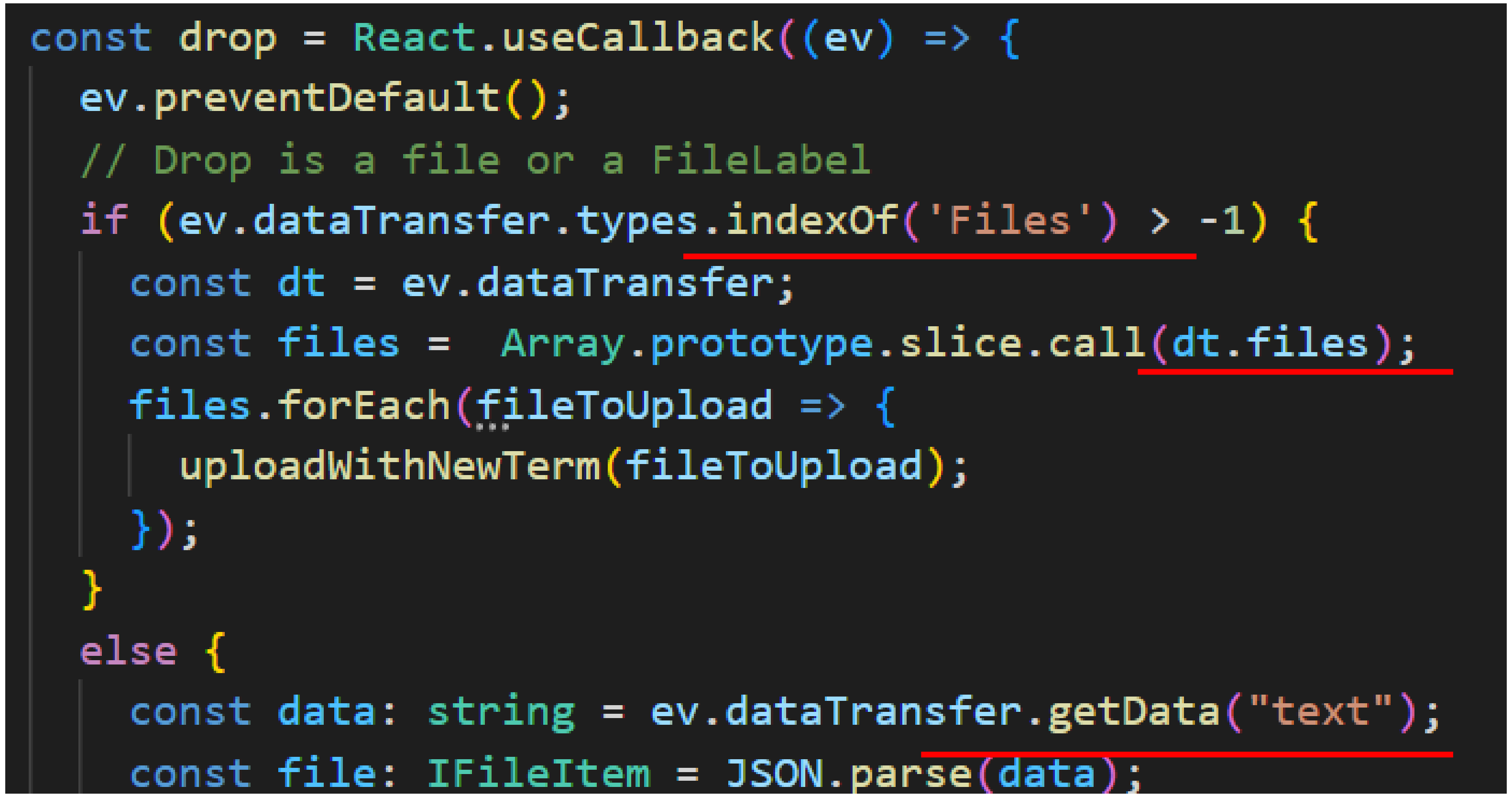


```
const drop = React.useCallback((ev) => {
  ev.preventDefault();
  // Drop is a file or a FileLabel
  if (ev.dataTransfer.types.indexOf('Files') > -1) {…
  }
  else {
    const data: string = ev.dataTransfer.getData("text");
    const file: IFileItem = JSON.parse(data);
    setDroppedFile(file);
    if (ev.ctrlKey) {
      setShowContextualMenu(true);
    }
    else {
      addNewTerm(file); // Default option: Simply add the new
```



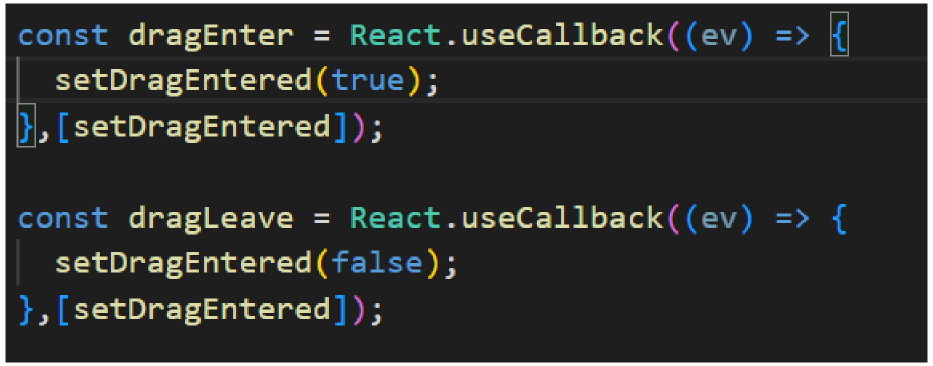
```
const addNewTerm = React.useCallback((file: IFileItem) => {
  const newTaxonomyValue: string = `${props.node.name}|${props.node.guid}`;
  file.termGuid.push(props.node.guid);
  file.taxValue.push(newTaxonomyValue);
  props.addTerm(file, newTaxonomyValue);
},[]); // eslint-disable-line react-hooks/exhaustive-deps
```

# Special: Drop ‚file‘ from outside



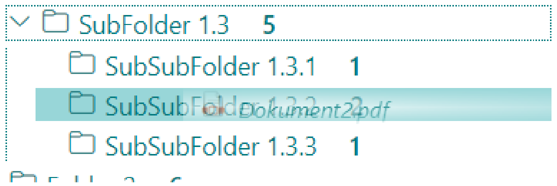
```
const drop = React.useCallback((ev) => {
  ev.preventDefault();
  // Drop is a file or a FileLabel
  if (ev.dataTransfer.types.indexOf('Files') > -1) {
    const dt = ev.dataTransfer;
    const files =  Array.prototype.slice.call(dt.files);
    files.forEach(fileToUpload => {
      uploadWithNewTerm(fileToUpload);
    });
  }
  else {
    const data: string = ev.dataTransfer.getData("text");
    const file: IFileItem = JSON.parse(data);
```

# TermLabel: Contextual styling

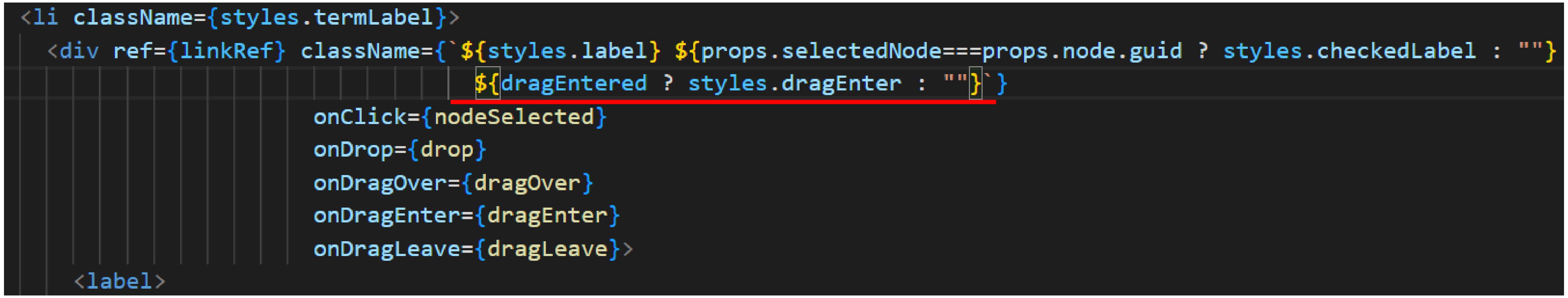


```
const dragEnter = React.useCallback((ev) => {
  setDragEntered(true);
},[setDragEntered]);

const dragLeave = React.useCallback((ev) => {
  setDragEntered(false);
},[setDragEntered]);
```

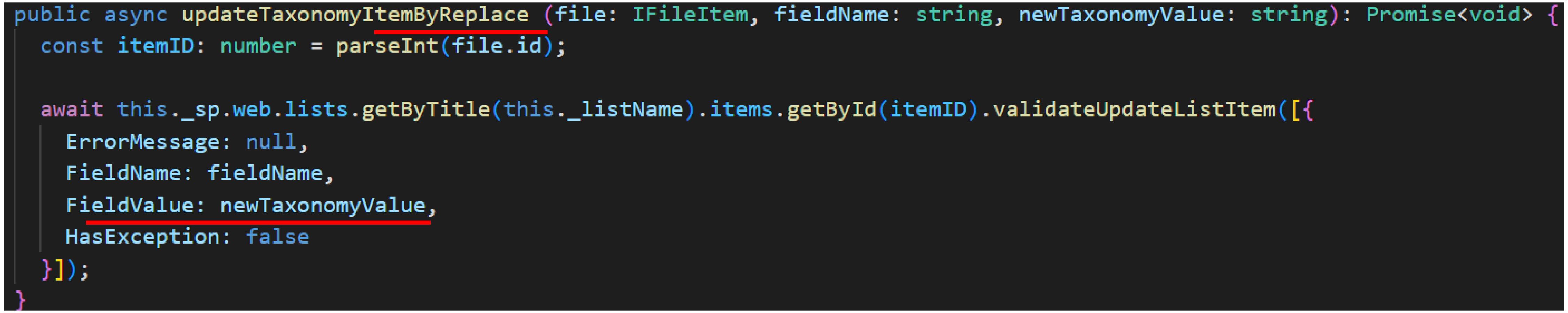


```
SubFolder 1.3   5
   SubSubFolder 1.3.1   1
   SubSubFolder 1.3.2   2   Dokument2.pdf
   SubSubFolder 1.3.3   1
```



```
<li className={styles.termLabel}>
  <div ref={linkRef} className={`${styles.label} ${props.selectedNode===props.node.guid ? styles.checkedLabel : ""}
                                 ${dragEntered ? styles.dragEnter : ""}`}
            onClick={nodeSelected}
            onDrop={drop}
            onDragOver={dragOver}
            onDragEnter={dragEnter}
            onDragLeave={dragLeave}>
    <label>
```

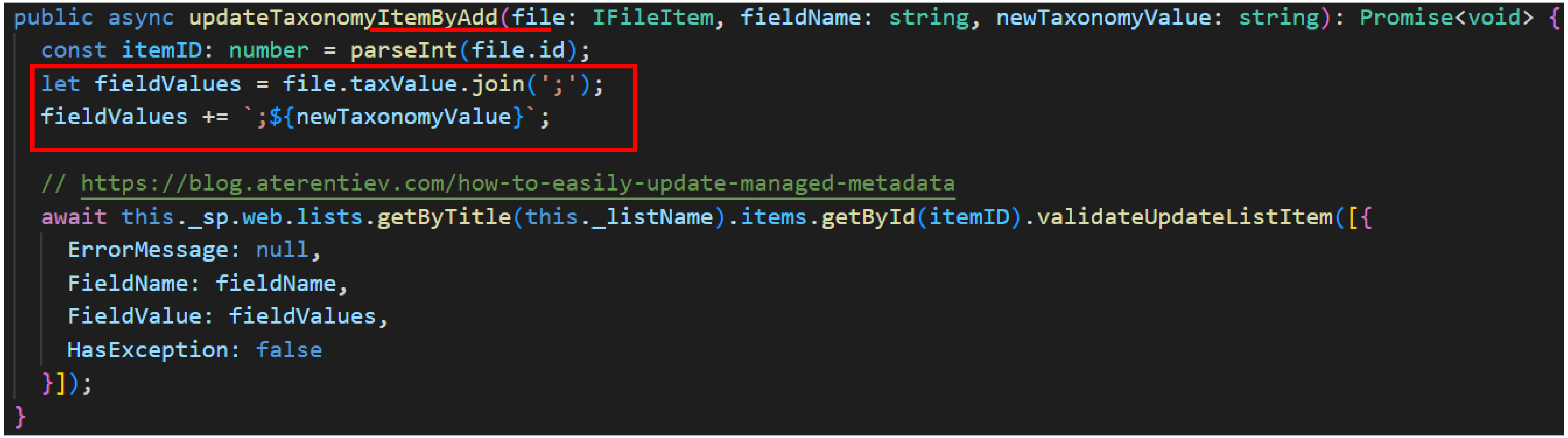
# SPService: Termset operations (1)



```
public async updateTaxonomyItemByReplace (file: IFileItem, fieldName: string, newTaxonomyValue: string): Promise<void> {
  const itemID: number = parseInt(file.id);

  await this._sp.web.lists.getByTitle(this._listName).items.getById(itemID).validateUpdateListItem([{
    ErrorMessage: null,
    FieldName: fieldName,
    FieldValue: newTaxonomyValue,
    HasException: false
  }]);
}
```

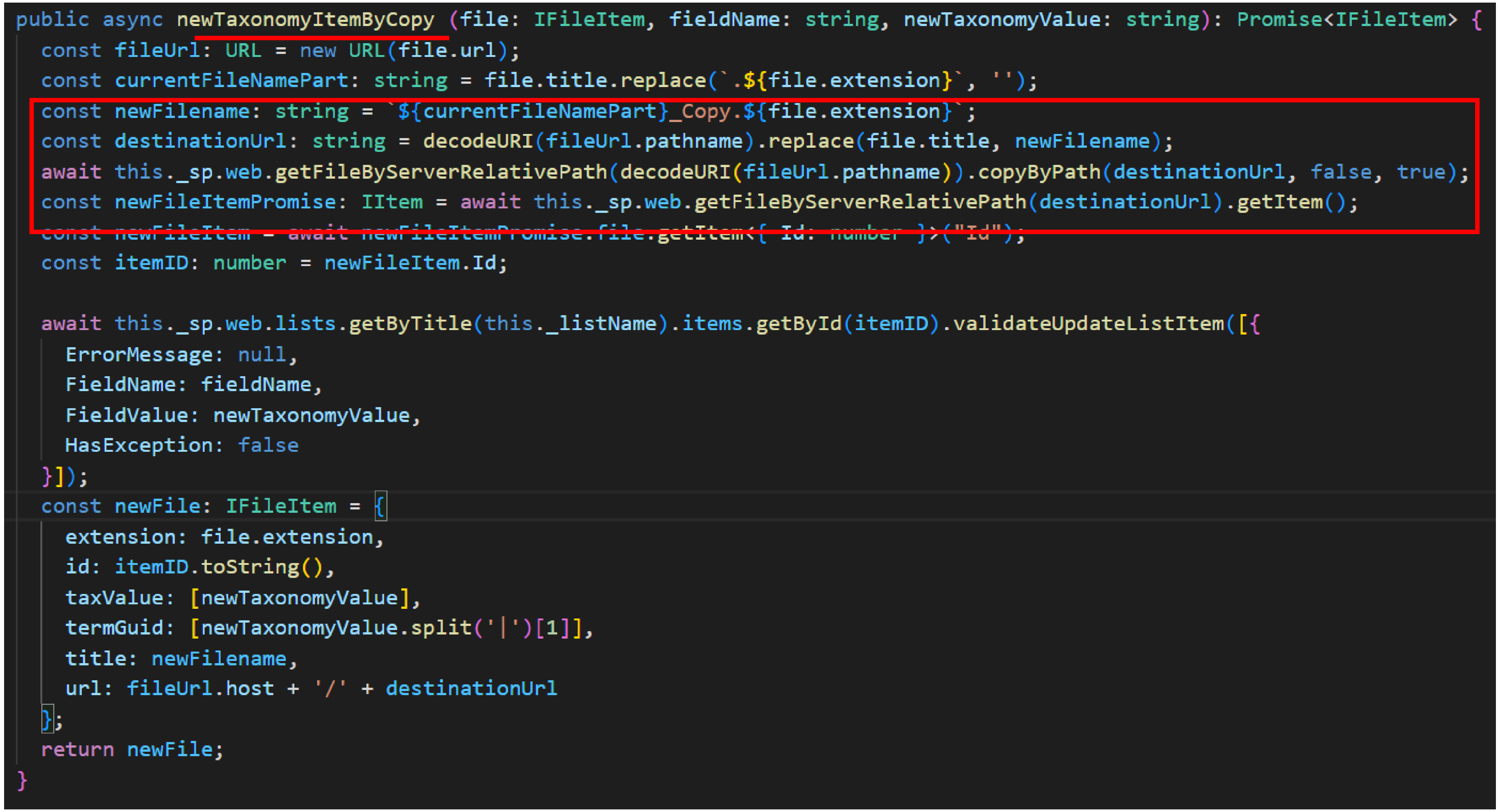
# SPService: Termset operations (2)



```
public async updateTaxonomyItemByAdd(file: IFileItem, fieldName: string, newTaxonomyValue: string): Promise<void> {
  const itemID: number = parseInt(file.id);
  let fieldValues = file.taxValue.join(';');
  fieldValues += `;${newTaxonomyValue}`;

  // https://blog.aterentiev.com/how-to-easily-update-managed-metadata
  await this._sp.web.lists.getByTitle(this._listName).items.getById(itemID).validateUpdateListItem([{
    ErrorMessage: null,
    FieldName: fieldName,
    FieldValue: fieldValues,
    HasException: false
  }]);
}
```

# SPService: Termset operations (3)



```
public async newTaxonomyItemByCopy (file: IFileItem, fieldName: string, newTaxonomyValue: string): Promise<IFileItem> {
  const fileUrl: URL = new URL(file.url);
  const currentFileNamePart: string = file.title.replace(`.${file.extension}`, '');
  const newFilename: string = `${currentFileNamePart}_Copy.${file.extension}`;
  const destinationUrl: string = decodeURI(fileUrl.pathname).replace(file.title, newFilename);
  await this._sp.web.getFileByServerRelativePath(decodeURI(fileUrl.pathname)).copyByPath(destinationUrl, false, true);
  const newFileItemPromise: IItem = await this._sp.web.getFileByServerRelativePath(destinationUrl).getItem();
  const newFileItem = await newFileItemPromise.file.getItem<{ Id: number }>("Id");
  const itemID: number = newFileItem.Id;

  await this._sp.web.lists.getByTitle(this._listName).items.getById(itemID).validateUpdateListItem([{
    ErrorMessage: null,
    FieldName: fieldName,
    FieldValue: newTaxonomyValue,
    HasException: false
  }]);
  const newFile: IFileItem = {
    extension: file.extension,
    id: itemID.toString(),
    taxValue: [newTaxonomyValue],
    termGuid: [newTaxonomyValue.split('|')[1]],
    title: newFilename,
    url: fileUrl.host + '/' + destinationUrl
  };
  return newFile;
}
```

## Resources

- https://mmsharepoint.wordpress.com/2021/12/23/a-sharepoint-file-explorer-based-on-managed-metadata-and-spfx/
  [BlogPost]
- https://github.com/pnp/sp-dev-fx-webparts/tree/main/samples/react-taxonomy-file-explorer
  [PnP Sample]
- https://pnp.github.io/pnpjs/sp/taxonomy/#getallchildrenasorderedtree
  [PnPJS Documentation]