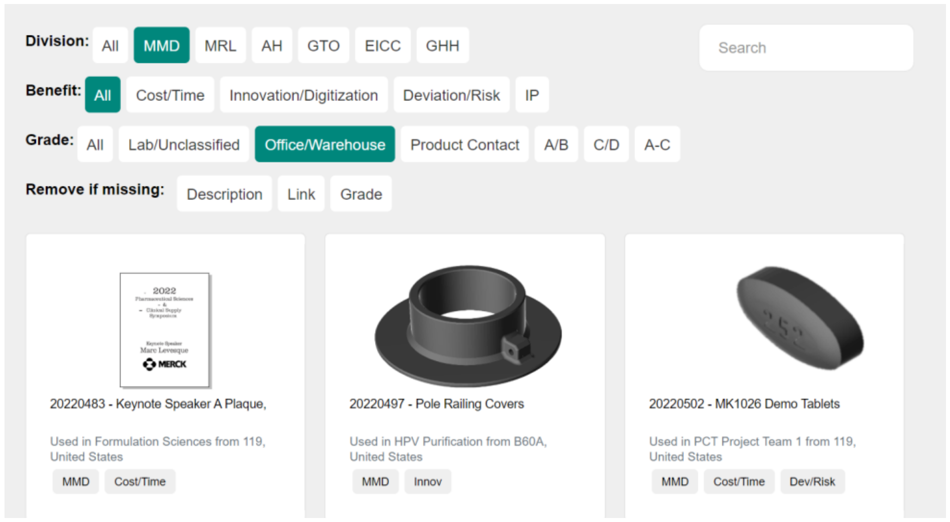
About

The 3DT catalog was designed and programmed by Zuting Chen, an IT Intern for Merck in the summer of 2022. This document was made to explain some of the programming and how to update certain things.

Combination Isotope Filtering:



Division: All [✓ MMD] MRL AH GTO EICC GHH

Search

Benefit: [✓ All] Cost/Time Innovation/Digitization Deviation/Risk IP

Grade: All Lab/Unclassified [✓ Office/Warehouse] Product Contact A/B C/D A-C

Remove if missing: Description Link Grade

20220483 - Keynote Speaker A Plaque,

Used in Formulation Sciences from 119, United States

MMD Cost/Time

20220497 - Pole Railing Covers

Used in HPV Purification from B60A, United States

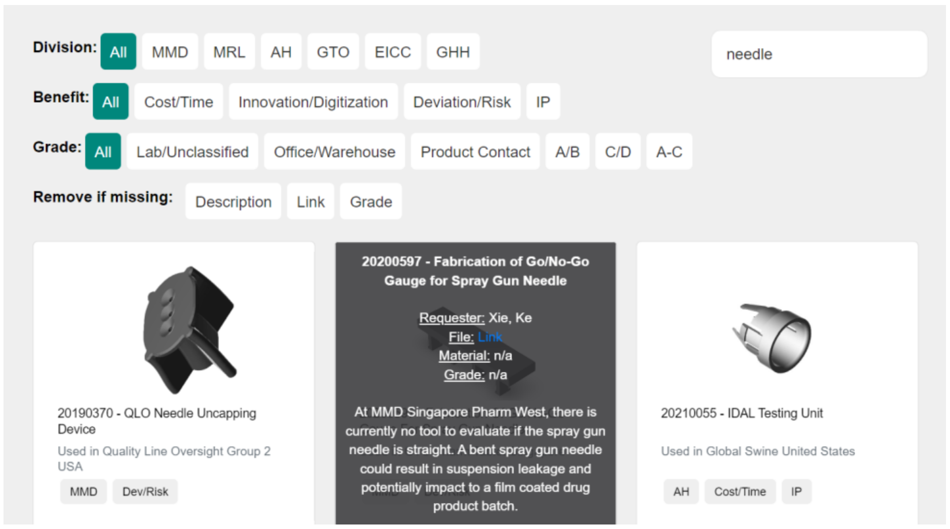
MMD Innov

20220502 - MK1026 Demo Tablets

Used in PCT Project Team 1 from 119, United States

MMD Cost/Time Dev/Risk

Searching & Information (hover or tap on card):



Division: [✓ All] MMD MRL AH GTO EICC GHH

needle

Benefit: [✓ All] Cost/Time Innovation/Digitization Deviation/Risk IP

Grade: [✓ All] Lab/Unclassified Office/Warehouse Product Contact A/B C/D A-C

Remove if missing: Description Link Grade

20190370 - QLO Needle Uncapping Device

Used in Quality Line Oversight Group 2 USA

MMD Dev/Risk

20200597 - Fabrication of Go/No-Go Gauge for Spray Gun Needle

Requester: Xie, Ke
File: Link
Material: n/a
Grade: n/a

At MMD Singapore Pharm West, there is currently no tool to evaluate if the spray gun needle is straight. A bent spray gun needle could result in suspension leakage and potentially impact to a film coated drug product batch.

20210055 - IDAL Testing Unit

Used in Global Swine United States

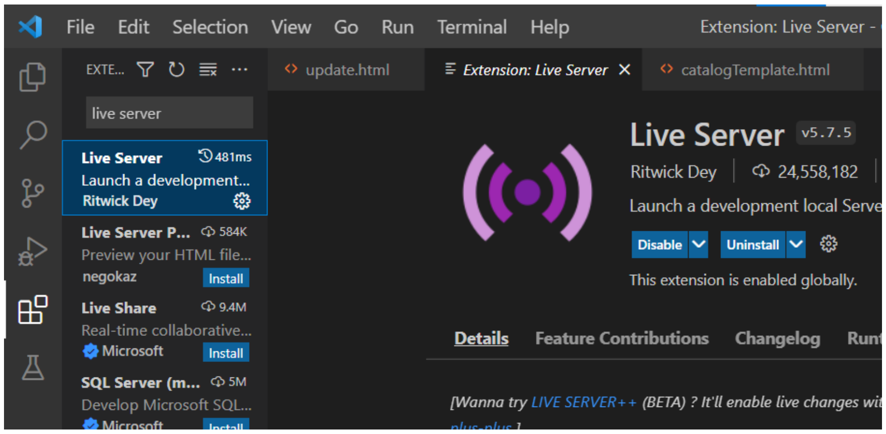
AH Cost/Time IP

Live Server Setup

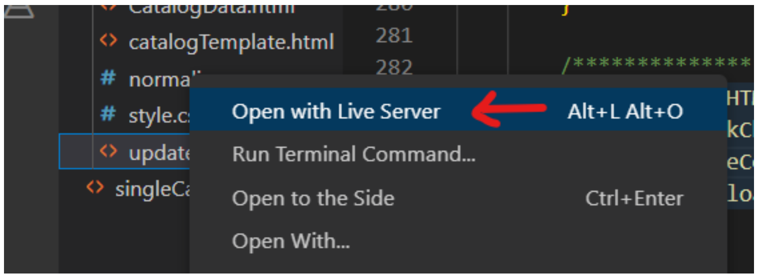
! Important: This update.html needs to be opened as a "Live Server" in Visual Studio Code.

Download Live Server from VS Code Extensions library. Right click on the .html file and "Open with Live Server"
To stop a live server (Ctrl + Shift + P -> Live Server: Stop Live Server)

Installing extension:

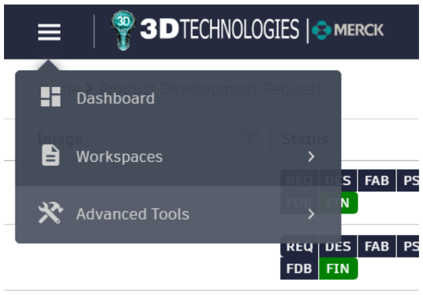


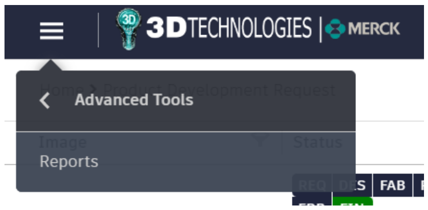
Opening Live Server:



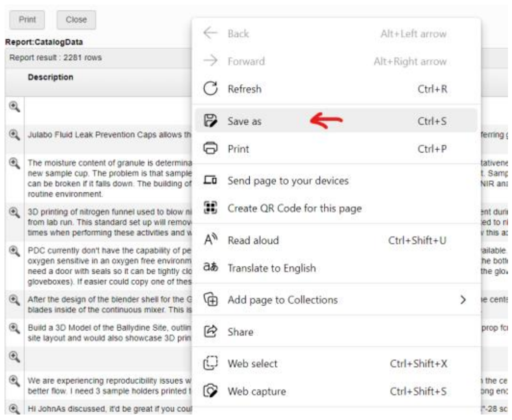
How to update catalog data

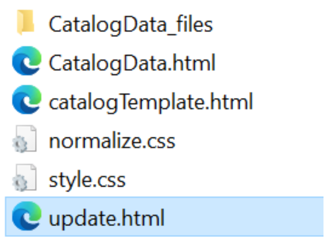
1. Go to PLM Reports in Advanced Tools on top left of PLM.





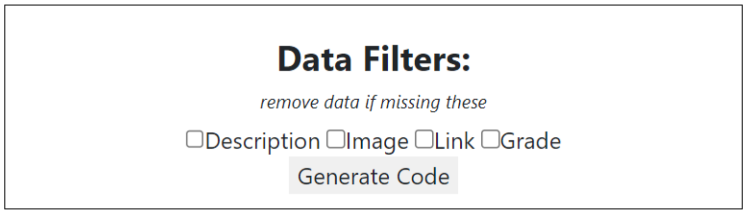
2. After the page loads, save the page HTML in the folder with this file using "Save As" (right click -> Save As, OR Ctrl + S).





^ Save in same folder as update.html

3. The saved file should be named "CatalogData.html" and folder "CatalogData_files".
4. Once set, select your data filters and hit "Generate Code" (box on this page).

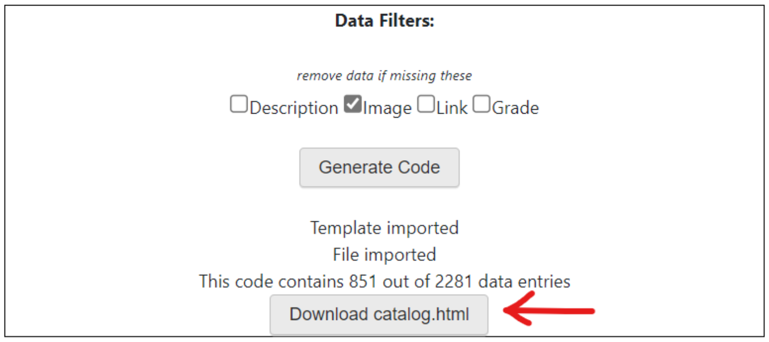

Data Filters:

remove data if missing these

☐Description ☐Image ☐Link ☐Grade

Generate Code

5. After download button shows up, download the file "catalog.html".



Data Filters:

remove data if missing these

Template imported
File imported
This code contains 851 out of 2281 data entries

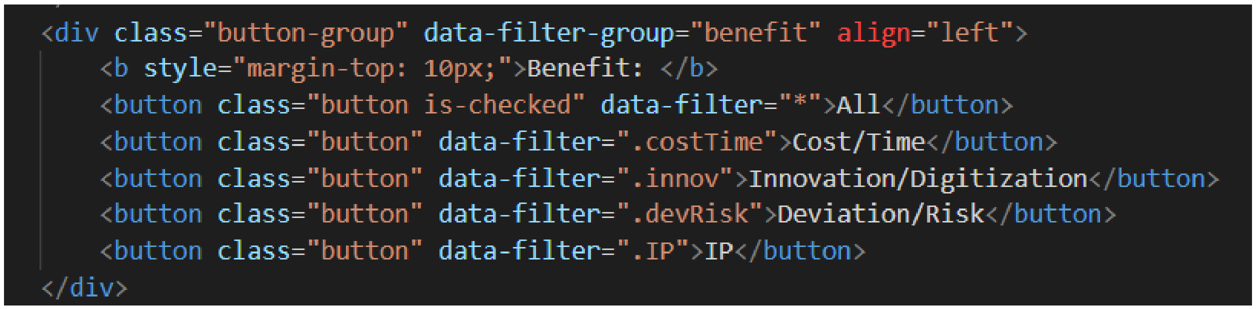
^ Code with 1000+ data entries are already high load. Best to keep under 1500

6. All done! Make sure you have catalog.html, normalize.css, style.css, CatalogData.html, and the CatalogData_files folder in the same folder if moved.

How to update filtering buttons

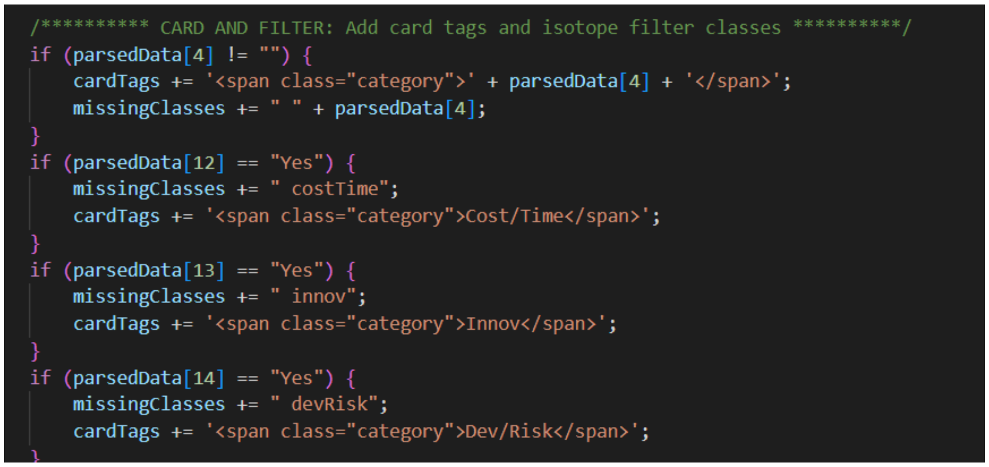
For if you would like to add additional filtering options for isotope filtering at the top. The filtering JS used in “catalog.html” is based on this isotope filtering example.

Buttons for filtering by sections



```
<div class="button-group" data-filter-group="benefit" align="left">
    <b style="margin-top: 10px;">Benefit: </b>
    <button class="button is-checked" data-filter="*">All</button>
    <button class="button" data-filter=".costTime">Cost/Time</button>
    <button class="button" data-filter=".innov">Innovation/Digitization</button>
    <button class="button" data-filter=".devRisk">Deviation/Risk</button>
    <button class="button" data-filter=".IP">IP</button>
</div>
```

1. Go to “catalogTemplate.html” in VS Code.
2. Search for the section with <button>s, as shown above.
3. Copy from the <div> with class=“button-group” to the end of div </div>
4. Change data-filter-group to be a unique value (if making a new row of buttons)
5. Change data-filter for each button to a corresponding class name with . in front of it
    1. This class name then needs to be changed in codePiece() in “update.html”. Make a condition for if the data should be selected. If that condition is true, then add to missingClasses the aforementioned class with a space in front of the class name

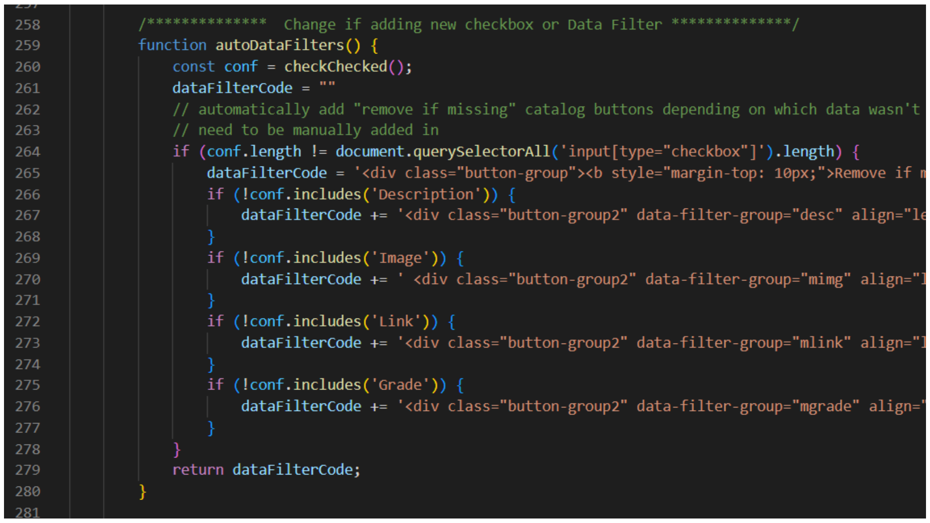


```
/********** CARD AND FILTER: Add card tags and isotope filter classes **********/
if (parsedData[4] != "") {
    cardTags += '<span class="category">' + parsedData[4] + '</span>';
    missingClasses += " " + parsedData[4];
}
if (parsedData[12] == "Yes") {
    missingClasses += " costTime";
    cardTags += '<span class="category">Cost/Time</span>';
}
if (parsedData[13] == "Yes") {
    missingClasses += " innov";
    cardTags += '<span class="category">Innov</span>';
}
if (parsedData[14] == "Yes") {
    missingClasses += " devRisk";
    cardTags += '<span class="category">Dev/Risk</span>';
```

Buttons for filtering by “Remove if missing”

- The “Remove if missing” buttons are automatically generated for catalog.html based on the checkboxes in update.html

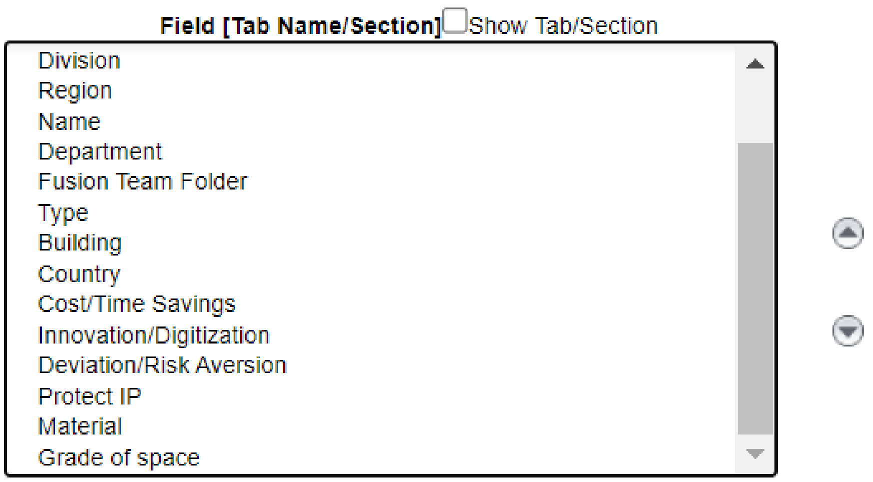
- ○ But you will need to go into autoDataFilters() in “update.html” to add the button details



```
/*************  Change if adding new checkbox or Data Filter *************/
function autoDataFilters() {
    const conf = checkChecked();
    dataFilterCode = ""
    // automatically add "remove if missing" catalog buttons depending on which data wasn't
    // need to be manually added in
    if (conf.length != document.querySelectorAll('input[type="checkbox"]').length) {
        dataFilterCode = '<div class="button-group"><b style="margin-top: 10px;">Remove if m
        if (!conf.includes('Description')) {
            dataFilterCode += '<div class="button-group2" data-filter-group="desc" align="le
        }
        if (!conf.includes('Image')) {
            dataFilterCode += ' <div class="button-group2" data-filter-group="mimg" align="l
        }
        if (!conf.includes('Link')) {
            dataFilterCode += '<div class="button-group2" data-filter-group="mlink" align="l
        }
        if (!conf.includes('Grade')) {
            dataFilterCode += '<div class="button-group2" data-filter-group="mgrade" align="
        }
    }
    return dataFilterCode;
}
```

Updating PLM Report Fields

This change can require the most changes in the code. In order to prevent extra work, please add additional fields below the ones prepared in the CatalogData report



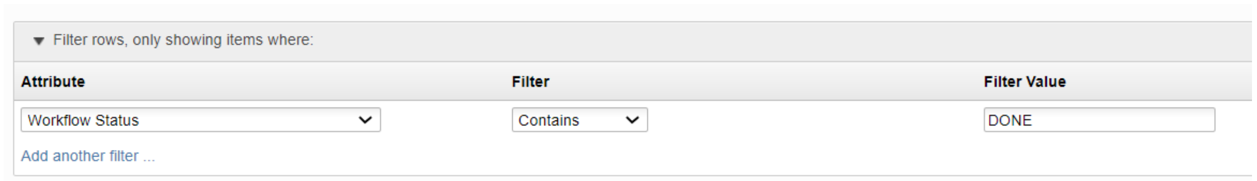
Field [Tab Name/Section] ☐ Show Tab/Section

Division
Region
Name
Department
Fusion Team Folder
Type
Building
Country
Cost/Time Savings
Innovation/Digitization
Deviation/Risk Aversion
Protect IP
Material
Grade of space

- Each of these fields correspond to an array value as outlined in parsedData. The code is hard coded to correspond to those specific fields.

Adding a Field to the PLM

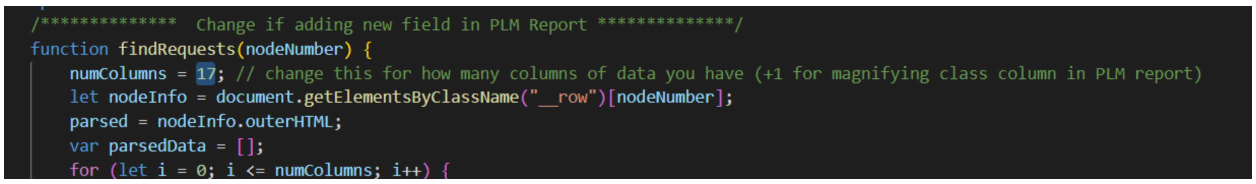
1. Go to PLM Reports in Advanced Tools on top left of PLM.
2. Find a report named CatalogData or make one as such with fields in the order as specified in parsedData
   a. If making a report, also filter by "Workflow Status" attribute to remove unfinished requests



▾ Filter rows, only showing items where:

Attribute	Filter	Filter Value
Workflow Status	Contains	DONE

3. Generating catalog.html and updating the button filters is the same process as above.
4. Change numColumns in findRequests() in "update.html" to match the number of columns in PLM report (or number of fields + 1)



```
/**************  Change if adding new field in PLM Report **************/
function findRequests(nodeNumber) {
    numColumns = 17; // change this for how many columns of data you have (+1 for magnifying class column in PLM report)
    let nodeInfo = document.getElementsByClassName("__row")[nodeNumber];
    parsed = nodeInfo.outerHTML;
    var parsedData = [];
    for (let i = 0; i <= numColumns; i++) {
```

5. Update codePiece() in "update.html" according to how you want the extra information to be displayed in each card. More about the card code in cardCode.

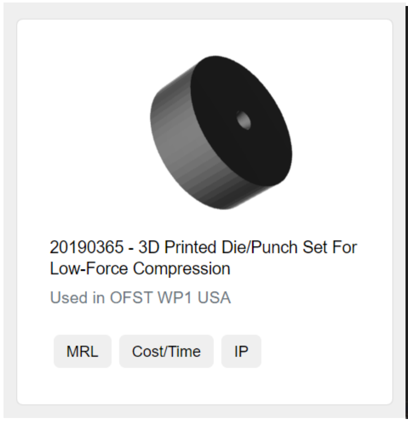
Understanding the Code

Files

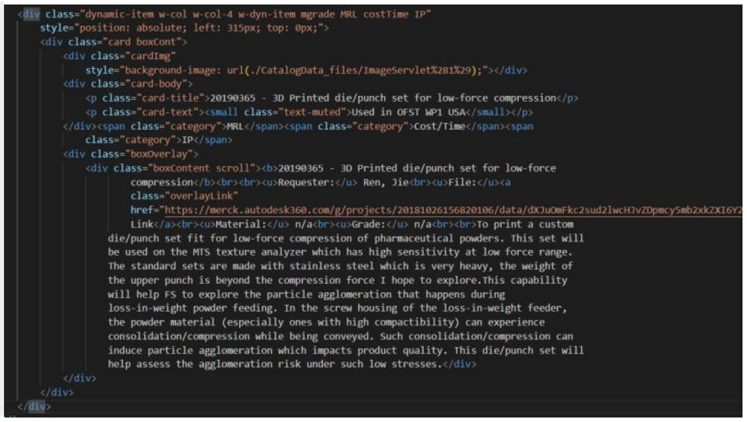
- CatalogData_files folder (imported)
    - Images from PLM report
- CatalogData.html (imported)
- catalog.html
    - MAIN webpage file
    - Generated in How to update catalog data
- catalogTemplate.html
    - Catalog format without the cards
    - Update this if you need to update filtering buttons
- normalize.css
- style.css
- update.html
    - Run this file using a Live Server to get “catalog.html”

Common Terms, Variables, and Functions

- **catalogTemplate.html**
    - data-filter-group: differentiates one group of isotope filtering selections from another. Name of this doesn’t matter as long as they are distinct
    - data-filter: what the isotope filtering looks for in the card classes. If selected, then only cards classed with those data-filter values will be shown
- **update.html**
    - cardCode: gathers up the HTML code (shown below) for a single card in codePiece()

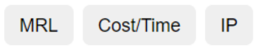


20190365 - 3D Printed Die/Punch Set For Low-Force Compression
Used in OFST WP1 USA
MRL
Cost/Time
IP



```
<div class="dynamic-item w-col w-col-4 w-dyn-item mgrade MRL costTime IP"
    style="position: absolute; left: 315px; top: 0px;">
    <div class="card boxCont">
        <div class="cardImg"
            style="background-image: url(./CatalogData_files/ImageServlet%281%29);"></div>
        <div class="card-body">
            <p class="card-title">20190365 - 3D Printed die/punch set for low-force compression</p>
            <p class="card-text"><small class="text-muted">Used in OFST WP1 USA</small></p>
        </div><span class="category">MRL</span><span class="category">Cost/Time</span><span
            class="category">IP</span>
        <div class="boxOverlay">
            <div class="boxContent scroll"><b>20190365 - 3D Printed die/punch set for low-force
                    compression</b><br><br><u>Requester:</u> Ren, Jie<br><u>File:</u><a
                    class="overlayLink"
                    href="https://merck.autodesk360.com/g/projects/20181026156820106/data/dXJuOmFkc2sud2lwcHJvZDpmcy5mb2xkZXI6Y2
                    Link</a><br><u>Material:</u> n/a<br><u>Grade:</u> n/a<br><br>To print a custom
                die/punch set fit for low-force compression of pharmaceutical powders. This set will
                be used on the MTS texture analyzer which has high sensitivity at low force range.
                The standard sets are made with stainless steel which is very heavy, the weight of
                the upper punch is beyond the compression force I hope to explore.This capability
                will help FS to explore the particle agglomeration that happens during
                loss-in-weight powder feeding. In the screw housing of the loss-in-weight feeder,
                the powder material (especially ones with high compactibility) can experience
                consolidation/compression while being conveyed. Such consolidation/compression can
                induce particle agglomeration which impacts product quality. This die/punch set will
                help assess the agglomeration risk under such low stresses.</div>
        </div>
    </div>
</div>
```

- cardTags: gathers up the HTML code for the tags for a single card in codePiece()



- dataFilterCode: used in autoDataFilters to store generated HTML code for “Remove if missing” buttons

  - missingClasses: gathers up the classes for isotope filtering for a single card in codePiece()
  - numColumns: in findRequests(). The highest value of parsedData (below) or the amount of columns in the PLM report (including magnifier)
  - parsedData: an array used to store a single row of data in the PLM report at any given moment
    - parsedData[0] - Magnifier, links to request on PLM
    - parsedData[1] - Description
    - parsedData[2] - Image
    - parsedData[3] - Request title
    - parsedData[4] - Division
    - parsedData[5] - Region
    - parsedData[6] - Name
    - parsedData[7] - Department
    - parsedData[8] - Fusion team folder
    - parsedData[9] - Type
    - parsedData[10] - Building
    - parsedData[11] - Country
    - parsedData[12] - Cost/Time Savings
    - parsedData[13] - Innovation/Digitization
    - parsedData[14] - Deviation/Risk Aversion
    - parsedData[15] - Protect IP
    - parsedData[16] - Material
    - parsedData[17] - Grade of space
- **Other Functions**
  - Broadly speaking, these are used to generate code in “update.html”. The contents of the generated code aren’t affected. These are functions that put together operations.
  - loadHTML()
    - Initiate data update process when user clicks “Generate Code”
  - checkChecked()
    - Retrieves checked checkboxes
  - writeCode()
    - Compiles code pieces (cardCode) into code for the pge
  - downloadFile()
    - Download function for downloading “catalog.html”

Future Work

- Add 3DT navbar and footer for integration into 3DT website

- More flexible filter and filter buttons, possibly. More flexibility in code generation to not require manual input of data configurations
    - Currently, you may need to update multiple sections of code to change things like:
        - PLM Report contents
        - Filtering options/buttons
        - Tags on cards
- Reduce load on website through caching and etc.
    - Current website has notable load times for over 1000+ requests
    - Maybe a different type of search or filtering
- Easier data retrieval and automated update process
    - Tinh Nguyen-Demary suggested that the Forge or Fusion 360 API connects the PLM data to other interfaces
- PLM Request details
    - Start adding material info and grade of space
    - Keep objects in images centered
    - Utilize images and descriptions