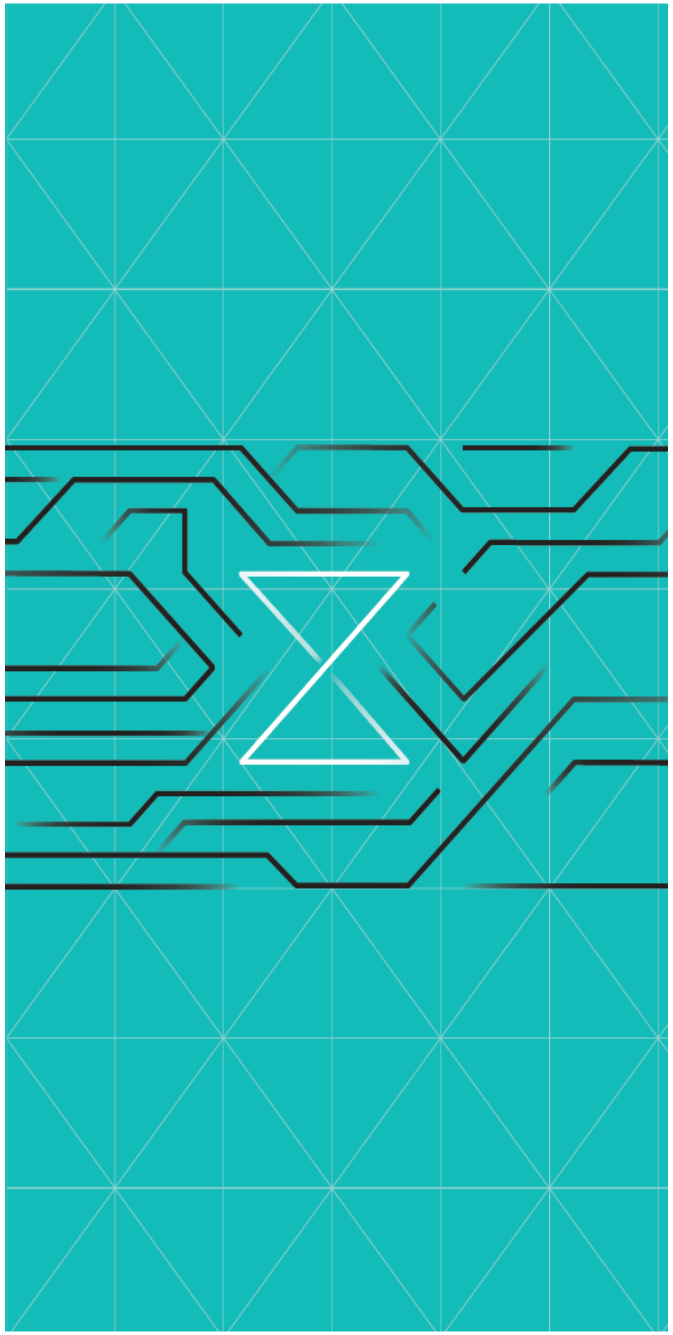


Advanced *250218-0001*

CBLH

Here for the long haul

Z Xplore

COBOL’S FUTURE IS BRIGHT

The Challenge

COBOL may be one of the oldest programming language, but it still runs the world!

In this challenge, you will crack open a COBOL program that takes a JSON file as input and prints a grocery store flyer.

You’ll learn the basics of COBOL in the process of completing the challenges. Being able to add COBOL to your own personal list of talents will definitely get you attention.

Before You Begin

Nothing besides the fundamentals (VSCode, datasets, JCL) is required.

Investment

Steps	Duration
16	150 minutes

1 CONFIGURE IBM Z OPEN EDITOR

This VSCode extension supplies COBOL language support, including syntax highlighting.

If you don't already have the IBM Z Open Editor installed, let's get that done.

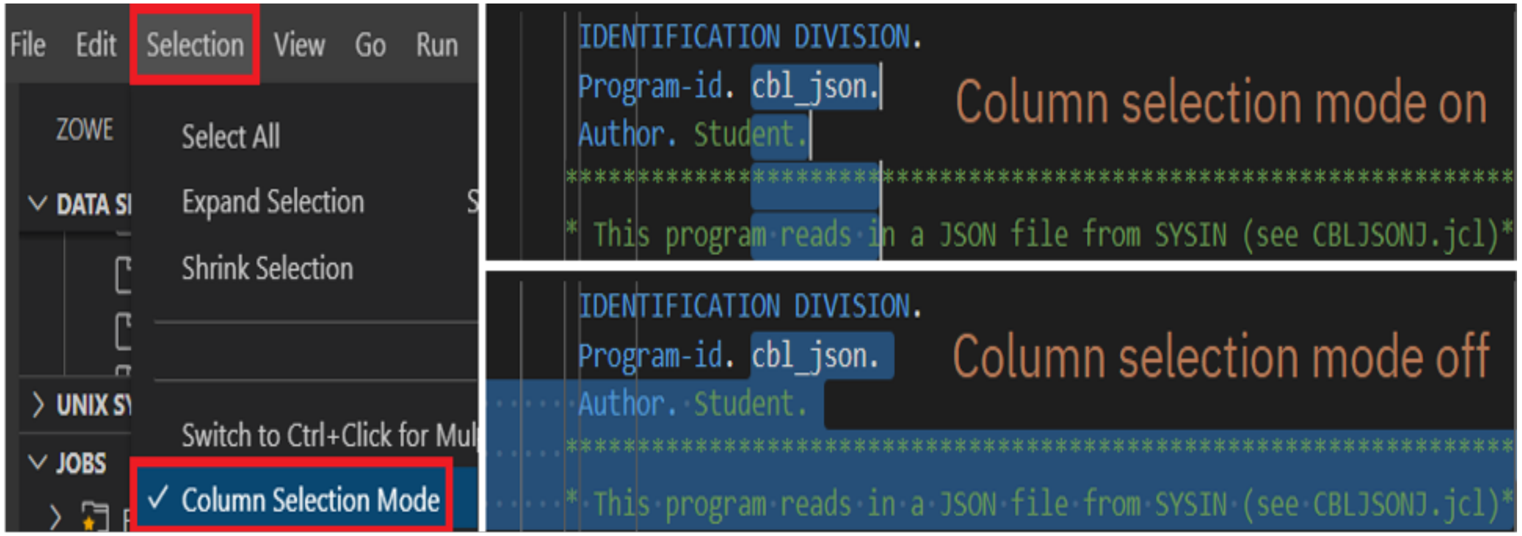
You should already have it, of course - or you would have found the Fundamentals challenges *very* difficult!

1.1 INSTALL IF MISSING

If not installed, open VSCode and in the left-side tool menu, select "Extensions".

From there, in the "Search Extensions in Marketplace" search field, type "IBM Z Open Editor". Search results will begin populating, select "IBM Z Open Editor" and click install.

1.2 SET UP COLUMN SELECTION MODE



```
       IDENTIFICATION DIVISION.
       Program-id. cbl_json.
       Author. Student.
      ***********************************************************
      * This program reads in a JSON file from SYSIN (see CBLJSONJ.jcl)*
```

By default, when you select lines of code, they will be highlighted in columns, which can be handy in some situations but probably not in this challenge.

If you'd like to turn off this feature, in your top menu bar, go to "Selection -> Column Selection Mode" and uncheck.

2 COBOL LIBRARY

You will need a dataset to keep your COBOL source code safe - check whether you already have one:

```
zowe files list data-set  Zxxxxx.CBL -a
```

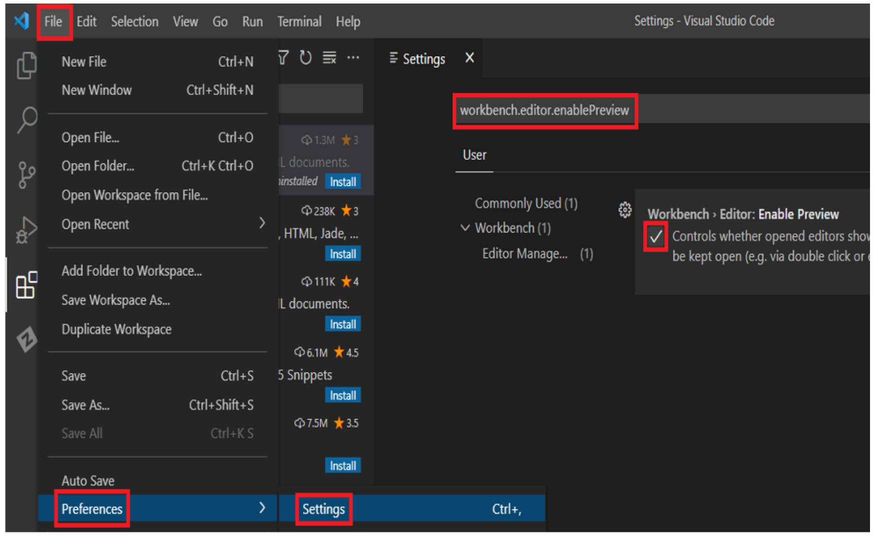
Only if you get an empty response, you need to create a CBL dataset.

- Create yourself a COBOL source “library” – from your VSCode terminal, allocate a new partitioned dataset using the zowe CLI:

```
zowe files create pds  Zxxxxx.CBL --data-class spds
```

Open the source dataset **ZXP.PUBLIC.SOURCE** and copy the **JSONCBL** and **JSONJCL** members into your own CBL and JCL datasets, as **CBLJSON** and **JSONJCL** respectively.

SOURCE MEMBER	YOUR COPY
ZXP.PUBLIC.SOURCE(JSONCBL)	Zxxxxx.CBL(CBLJSON)
ZXP.PUBLIC.SOURCE(JSONJCL)	Zxxxxx.JCL(JSONJCL)



Settings

workbench.editor.enablePreview

Workbench › Editor: **Enable Preview**

✓ Controls whether opened editors show
be kept open (e.g. via double click or

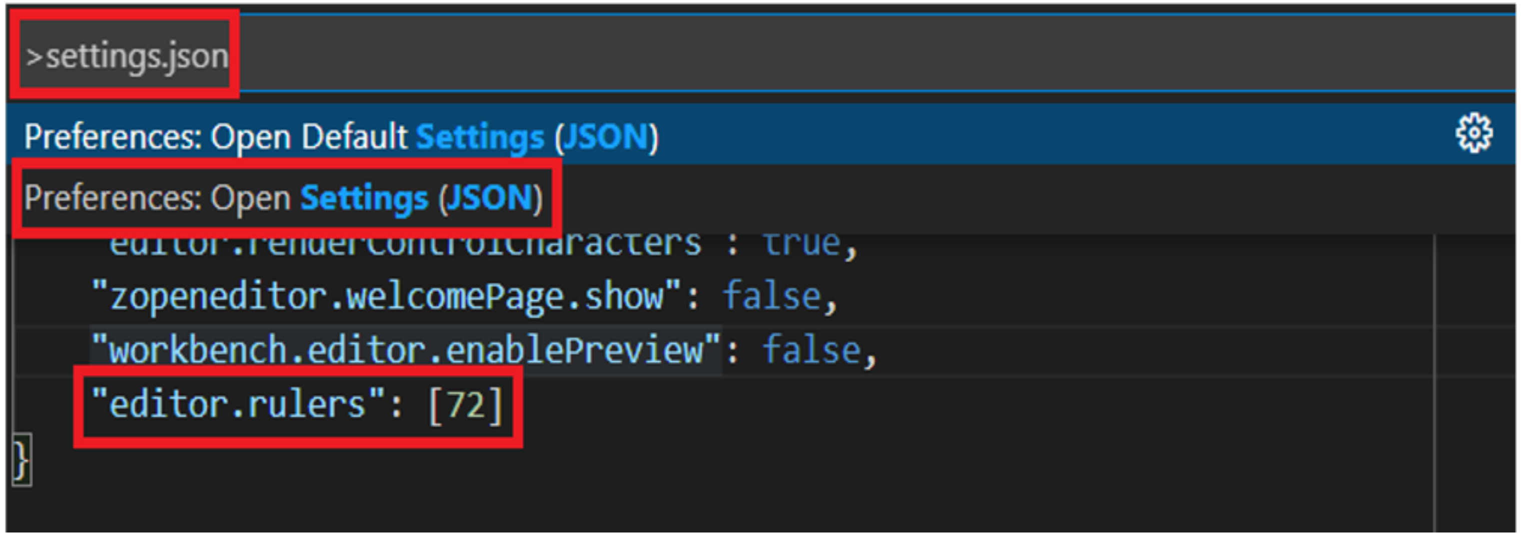
There is a default feature of VSCode that can be helpful in this challenge, to save you from having to close files you only need to preview; to get out of preview mode, double-click the italicized tab name. Once set, italics are removed.

3 COLUMNS RULES

IBM Z Open Editor already sets up the vertical rulers you'll need for COBOL source files.

JCL statements can be up to 80 columns wide, **but columns 73-80 are ignored**.

Use the following instructions to set up a column ruler after column 72 to help you avoid typing past there.


>settings.json
Preferences: Open Default Settings (JSON)
Preferences: Open Settings (JSON)

```
    "editor.renderControlCharacters": true,
    "zopeneditor.welcomePage.show": false,
    "workbench.editor.enablePreview": false,
    "editor.rulers": [72]
}
```

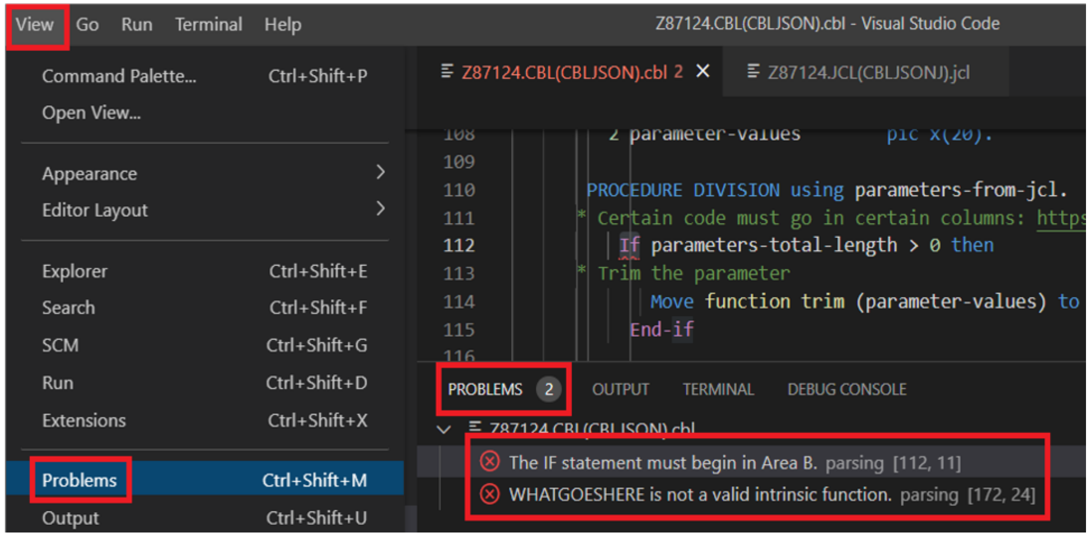
Open Command Palette (Ctrl+Shift+P on Windows, Command+Shift+P on MacOS) and type "settings json" (Select "Preferences: Open User Settings (JSON)"), search for "editor.rulers; if it is not there, search for "editor. and add this line: editor.rulers":[], .

Set the editor.rulers[] value to 72 – "editor.rulers":[72], Make sure to include the comma at the end of the line before.

If you are unable to make this change for some reason, don't worry! The vertical rule is not essential - even with it active, you may still need to handle JCL errors that result from data in column 73 and beyond. This can happen as a result of symbol substitution when the JCL is processed.

4 CHECK COBOL SYNTAX ERRORS

Open your COBOL source file **CBLJSON** by doing **Select View -> Problems** in the top menu.



```
108       2 parameter-values         pic x(20).
109
110      PROCEDURE DIVISION using parameters-from-jcl.
111     * Certain code must go in certain columns: https
112       If parameters-total-length > 0 then
113     * Trim the parameter
114            Move function trim (parameter-values) to
115         End-if
116
```

```
Z87124.CBL(CBLJSON).cbl
  The IF statement must begin in Area B. parsing [112, 11]
  WHATGOESHERE is not a valid intrinsic function. parsing [172, 24]
```

A list of two COBOL syntax errors is displayed in the "Problems" tab at the bottom. Double-click each problem and see if you can resolve the errors.

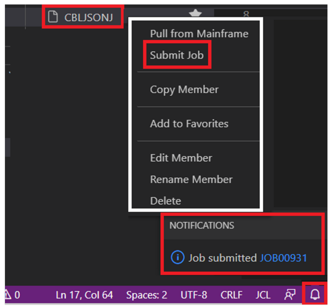
Once they are resolved, the "Problems" tab should show 0 errors and there should be no red underlines in your COBOL program.

5 SAVE, COMPILE, RUN

Save using **File -> Save**

Right-click your JCL member **JSONJCL** and select “Submit Job”.

The JCL script compiles and runs the COBOL program. A notification will appear that the job has been submitted. If it disappears, click the bell icon to see recent notifications.



Job submitted JOB00931

Save using **File->Save**.

Right click your JCL member JSONJCL and **Submit Job.**

The JCL script compiles and runs the COBOL program. A notification will appear that the job has been submitted. If it disappears, click the bell icon to see recent notifications.

Click the job number to open the job results.

Click the job number to open the job results.

WHY ARE WE LEARNING COBOL IN THIS DECADE?

Simple answer: COBOL is everywhere.

Have you, or someone you know, ever gone to a bank or withdrawn cash from an ATM?

Most likely, you were running a COBOL transaction.

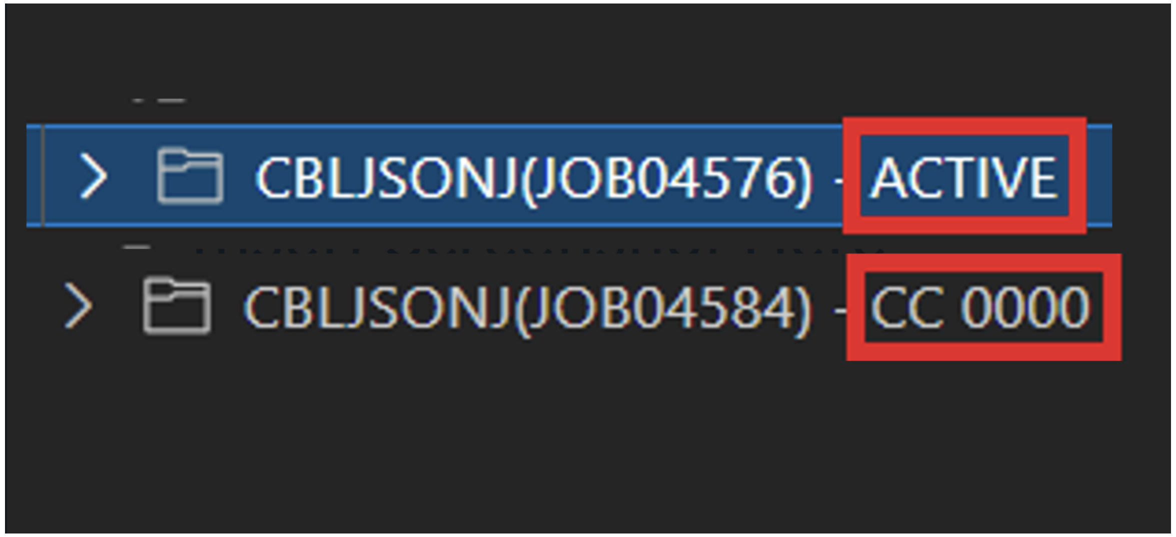
More than one million COBOL programs run every second of every day on mainframes. In fact, most IBM Fortune 1000 customers run core application using COBOL.

COBOL is self-documenting, a natural string-processing language and does fixed-point arithmetic.

It was designed for business processing, so it is very efficient, adaptable and maintainable.

To learn more about COBOL and its relevancy please look at:
https://community.ibm.com/community/user/ibmz-and-linuxone/blogs/dan-zhang1/2020/04/02/should-we-still-use-cobol

6 CHECK THE COMPLETION CODE

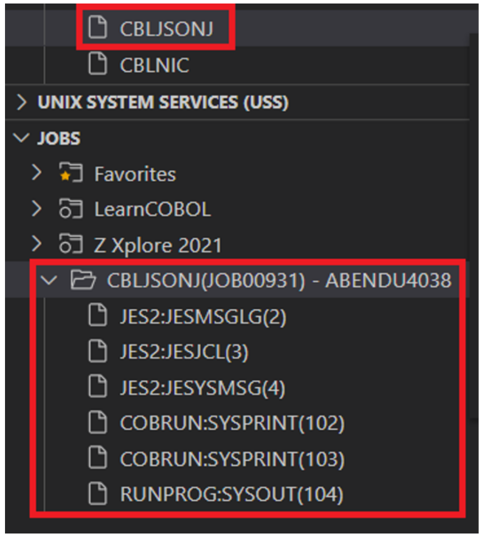

CBLJSONJ(JOB04576) - ACTIVE
CBLJSONJ(JOB04584) - CC 0000

If the program shows **ACTIVE**, collapse and re-expand the job results (or press the “Refresh” icon beside JOBS) until you see that the program ended (“ACTIVE” changes to a completion code).

A completion code (CC) of 0000 means success and anything else means there is an error (it may be severe, or just a warning).

Even if you successfully fixed the problems from the previous steps, you will still get an **ABEND** error message, which is up next.

7 OH NO! AN ABEND ...



ABEND stands for **AB**normal **END** and is a common error return code on z/OS.

Expand the joglog results to see the list of output files.

ABENDU4038 - does that look familiar? The last time you saw that error, there was a mismatch between the COBOL file name, and the DD names in the JCL ...

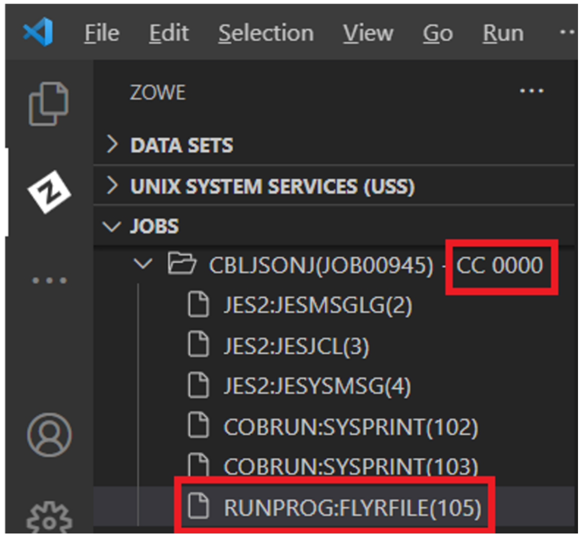
Examine the output files to locate the one with the error message.

The error message refers to **FLYYFILE**.

So, what is wrong? Examine both the COBOL and JCL files.

8 FIXED AND READY TO GO

Once you have fixed the cause of the ABEND, submit the JSONJCL job.



```
CBLJSONJ(JOB00945) - CC 0000
  JES2:JESMSGLG(2)
  JES2:JESJCL(3)
  JES2:JESYSMSG(4)
  COBRUN:SYSPRINT(102)
  COBRUN:SYSPRINT(103)
  RUNPROG:FLYRFILE(105)
```

You should now see completion code of 0000.

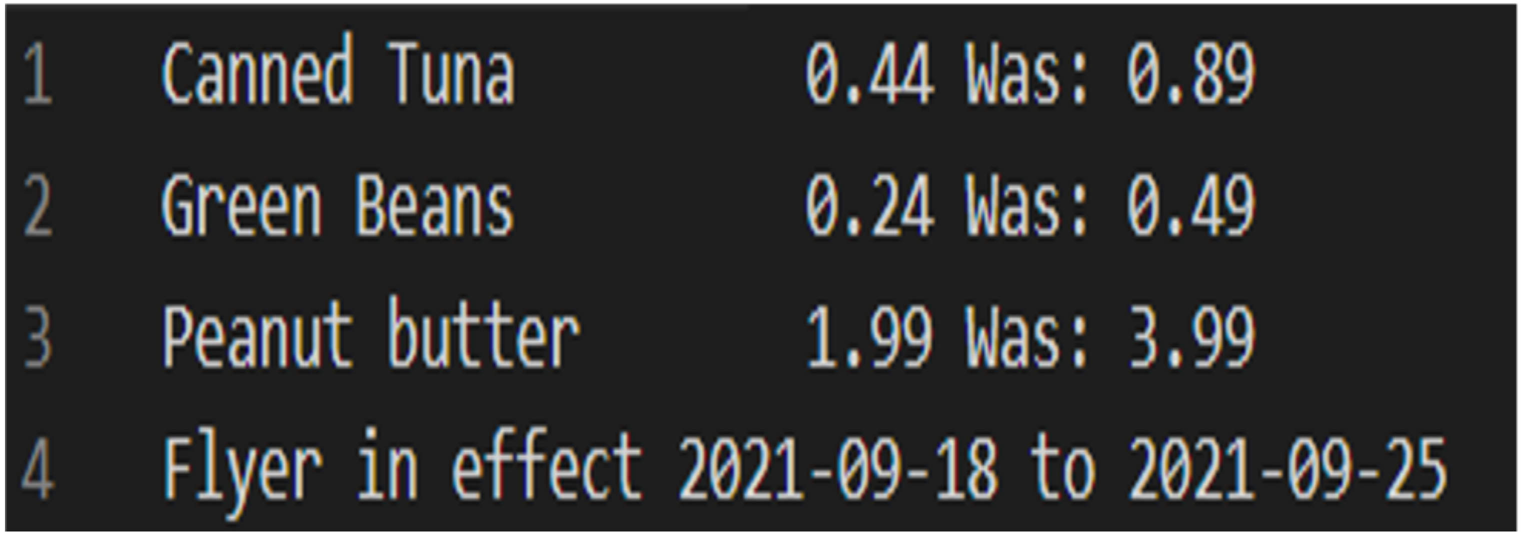
You might have seen that this program should output a grocery store flyer. You can examine the flyer in output file **RUNPROG:FLYRFILE**.

It shows that products that are about to expire have a 50% discount applied.

(The flyer is in plain text for now, but you will be fixing that soon!)

9 MISSING TITLE

The grocery store name “Corner Grocery Store” is missing from the plain text flyer.



```
Canned Tuna           0.44 Was: 0.89
Green Beans           0.24 Was: 0.49
Peanut butter         1.99 Was: 3.99
Flyer in effect 2021-09-18 to 2021-09-25
```

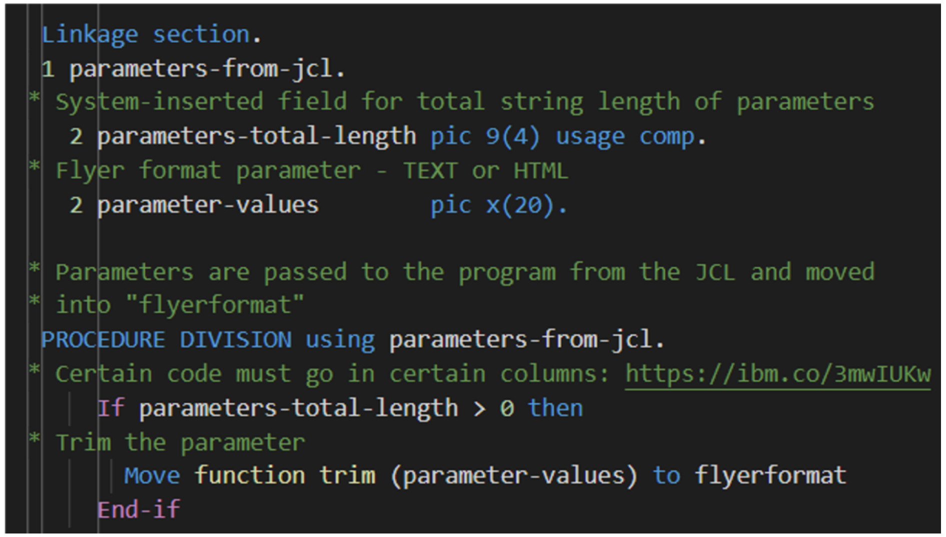
This one is tricky!

Locate and fix the problem, then submit JSONJCL job again to get sure the title added.

10 PLAIN JANE WON'T DO

You fixed the title, but you can't show the customers the flyer in plain text like this!

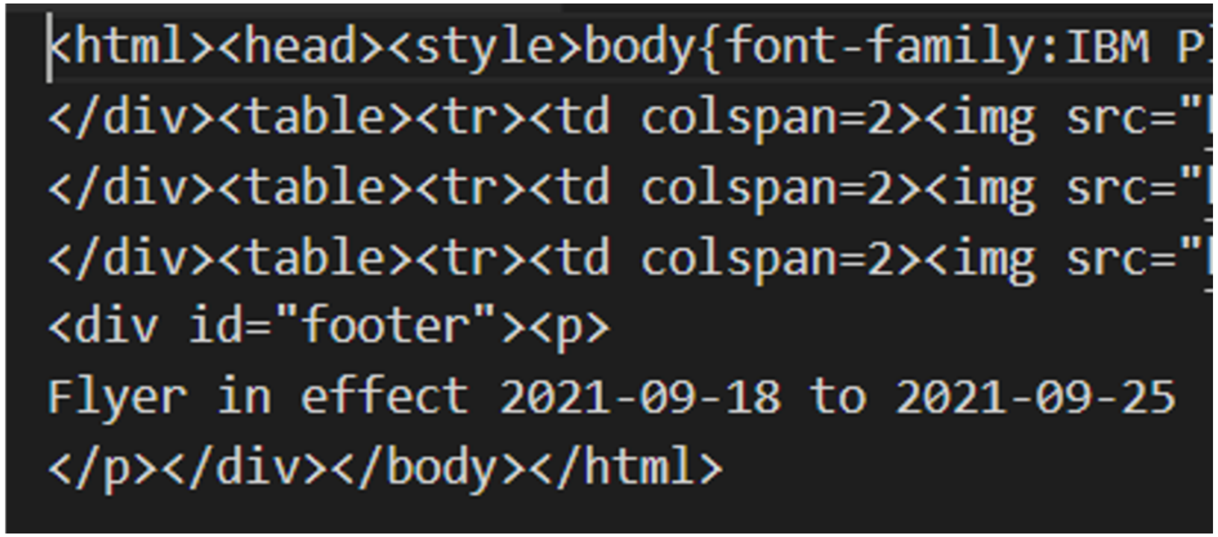
Perhaps there is a parameter to pass to the program to add HTML formatting?



```
 Linkage section.
 1 parameters-from-jcl.
* System-inserted field for total string length of parameters
   2 parameters-total-length pic 9(4) usage comp.
* Flyer format parameter - TEXT or HTML
   2 parameter-values        pic x(20).

* Parameters are passed to the program from the JCL and moved
* into "flyerformat"
 PROCEDURE DIVISION using parameters-from-jcl.
* Certain code must go in certain columns: https://ibm.co/3mwIUKw
     If parameters-total-length > 0 then
* Trim the parameter
       Move function trim (parameter-values) to flyerformat
     End-if
```

Once you think you've figured it out, submit JSONJCL again and check the flyer once more.

11 RUN THE CODE



```
<html><head><style>body{font-family:IBM P
</div><table><tr><td colspan=2><img src="
</div><table><tr><td colspan=2><img src="
</div><table><tr><td colspan=2><img src="
<div id="footer"><p>
Flyer in effect 2021-09-18 to 2021-09-25
</p></div></body></html>
```

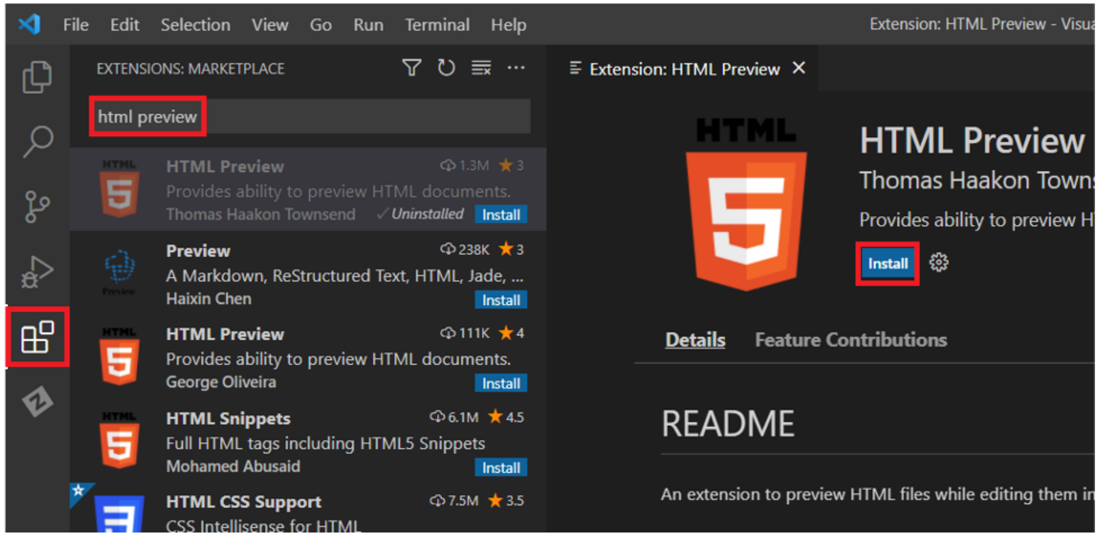
Open the job dataset **RUNPROG:FLYRFILE** again

If you see HTML tags starting with `<html>`, you did it!

But how can you preview it to see the formatting? VSCode can help ...

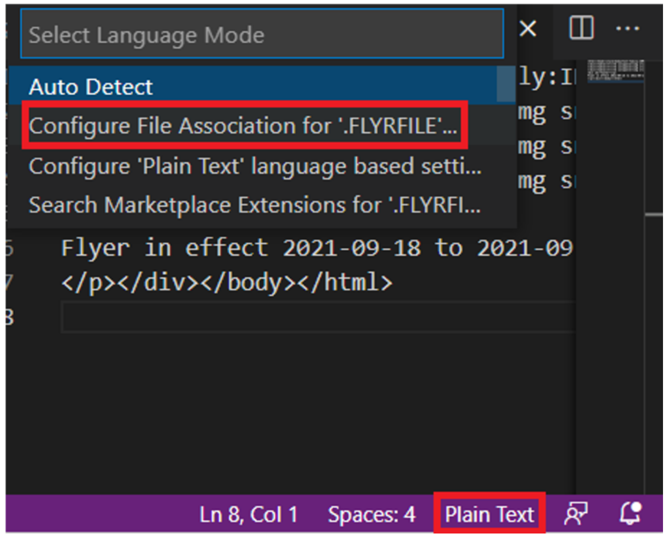
12 INSTALL HTML PREVIEWER

Select “Extensions” in the VSCode left-side navigation menu.



Search for “HTML Preview” and install one of the options, such as the one by Thomas Haakon Townsend

13 SET LANGUAGE ASSOCIATION

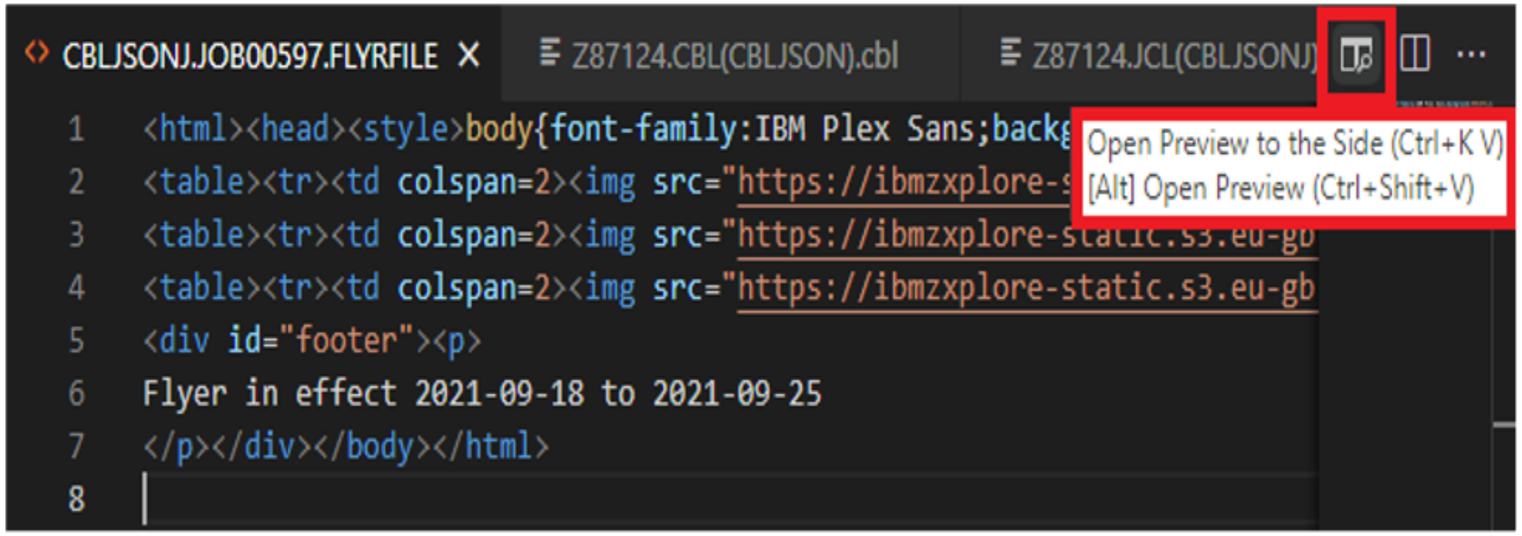

Select Language Mode
Auto Detect
Configure File Association for '.FLYRFILE'...
Configure 'Plain Text' language based setti...
Search Marketplace Extensions for '.FLYRFI...
Ln 8, Col 1
Spaces: 4
Plain Text

```
ly:I
mg s
mg s
mg s
Flyer in effect 2021-09-18 to 2021-09
</p></div></body></html>
```

With your job **FLYRFILE** open, in the bottom right, click “Plain Text”, then “Configure File Association for ‘.FLYRFILE’…” and enter HTML

Once done, you should see HTML syntax highlighting of the contents.

14 OPEN SESAME



CBLJSONJ.JOB00597.FLYRFILE
Z87124.CBL(CBLJSON).cbl
Z87124.JCL(CBLJSONJ)
Open Preview to the Side (Ctrl+K V)
[Alt] Open Preview (Ctrl+Shift+V)

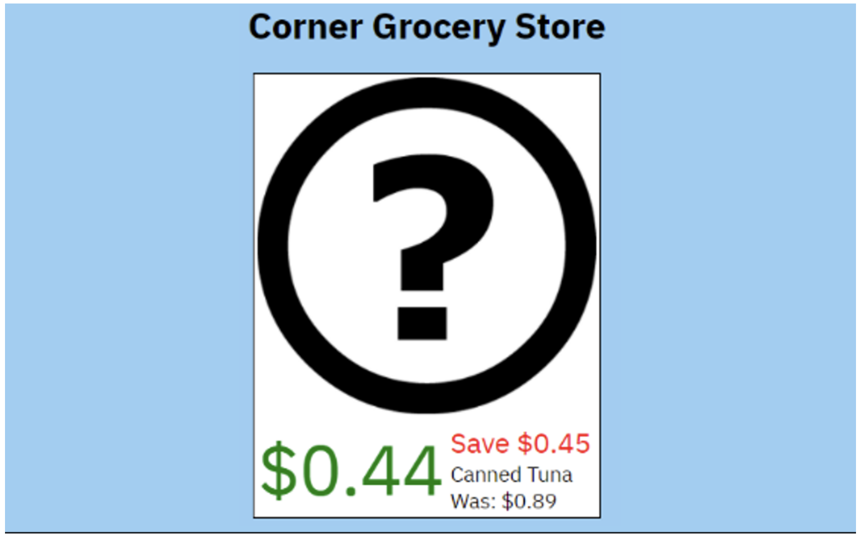
```
<html><head><style>body{font-family:IBM Plex Sans;backg
<table><tr><td colspan=2><img src="https://ibmzxplore-s
<table><tr><td colspan=2><img src="https://ibmzxplore-static.s3.eu-gb
<table><tr><td colspan=2><img src="https://ibmzxplore-static.s3.eu-gb
<div id="footer"><p>
Flyer in effect 2021-09-18 to 2021-09-25
</p></div></body></html>

```

At the top right, click the icon to “Open Preview to the Side”.

The flyer is better formatted now, but there are still some problems …

15 ADD CURRENCY SYMBOLS

The currency should be in US dollars ($), but the dollar sign is missing – what's wrong? Take a look at the code and identify the issue.



Once you have the issue fixed, submit JSONJCL again, then open the flyer HTML preview again to check your success.

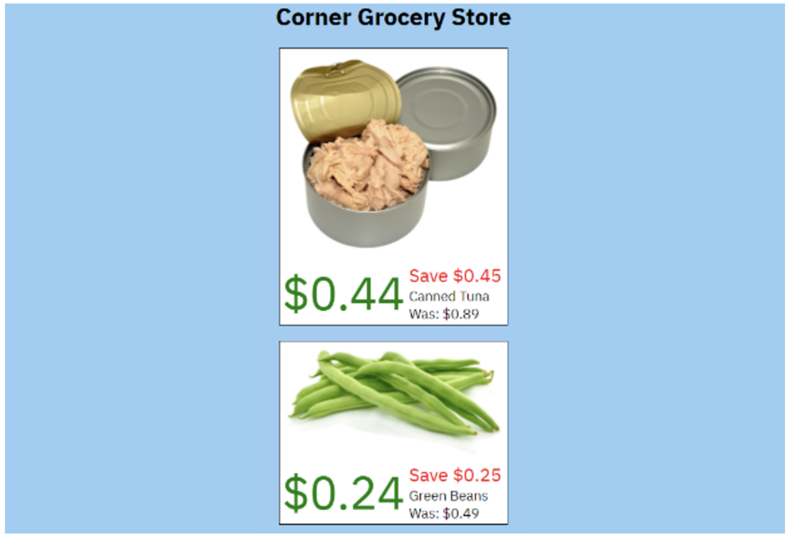
16 ADD THE PRODUCT IMAGES

Didn’t the input JSON (in the JCL file) contain links to real product images?

Why aren’t they showing up?

Did the code forget to map these images when declaring the JSON structure in COBOL? (yes, that is exactly what happened)

This is the final stage of this challenge. Once you figure it out, submit JSONJCL job and open the flyer HTML preview to make sure it looks nice.



To complete the challenge, create a copy of the flyer HTML in a **cobol.html** file in your USS home directory, and finally, submit the **CHKACBLH** job from ZXP.PUBLIC.JCL

17 EXPAND YOUR KNOWLEDGE

Learning COBOL Programming wi
Welcome to your introduction to COBOL!

Do you want to learn more? Take the free COBOL course from IBM.

It includes hands-on labs in VSCode:

- Video version: https://ibm.biz/learn-cobol-vscode
- Text version: https://ibm.biz/github-cobol

Nice job - let’s recap	Next up …
COBOL is one of the most used languages on IBM Z today, and runs the world’s economy, but those knowledgeable in the language are retiring. Your help is needed to fill that gap. You’ve demonstrated a solid understanding of some of the most important aspects of COBOL and have some good examples to share with prospective employers. You will especially impress them by demonstrating that you know about COBOL’s latest support for JSON, which they might not even know about yet!	Nice work! You’ve been able to expand your knowledge on COBOL using HTML. Continue your learning experience by choosing more Advanced or Extended challenges.

CBLHI250218-0001