Give Basic Needs NFT Engine

(The Phoebe Concept v0.8)

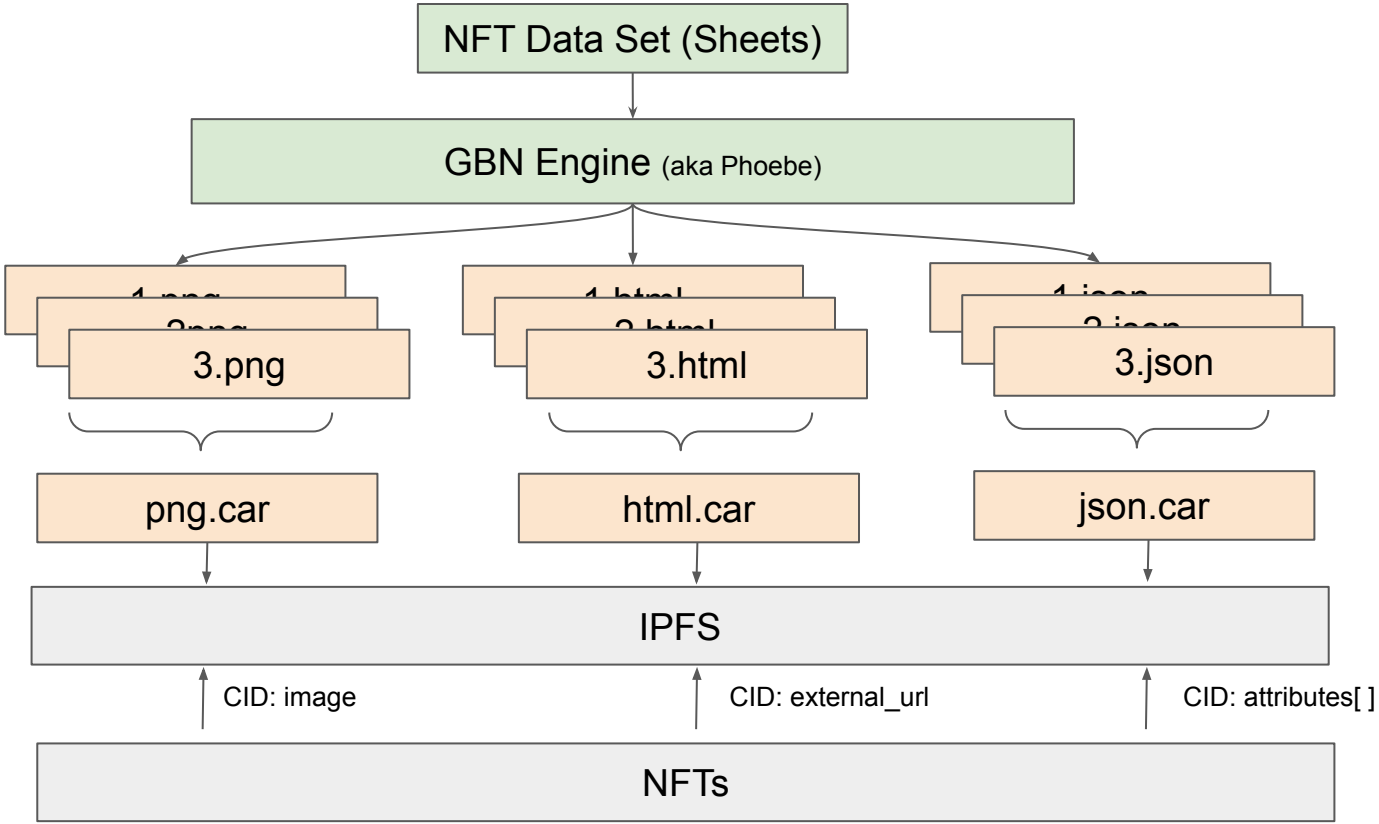
Why Use The Phoebe Concept?

Give Basic Needs
Food / Shelter / Medical

- Rapid development iterations, generating robust metadata and external sites consistently, because spreadsheet change and regens are simple for a fail fast mentality
- Consistent branding across all tokens in the collection automatically with your templates, you have total control of the user experience leveraging multi-file and directory support in IPFS deployment
- Standardized on robust token json metadata, promoting marketplace engagement thought collection filters resulting in more sales
- Unstoppable permanent images, animation, metadata and html all living on the fully distributed IPSF, pretty much for free
- No need to worry about how the marketplace is going to display your token metadata, because clickable external_url are widely supported and used by collectors
- Potentially to load less data on-chain, because of consistent external html generation from the same source spreadsheet stored in a single deployment file on IPFS
- Leverage the full power of a spreadsheet to track, sort, filter and calculate your metadata and html generation

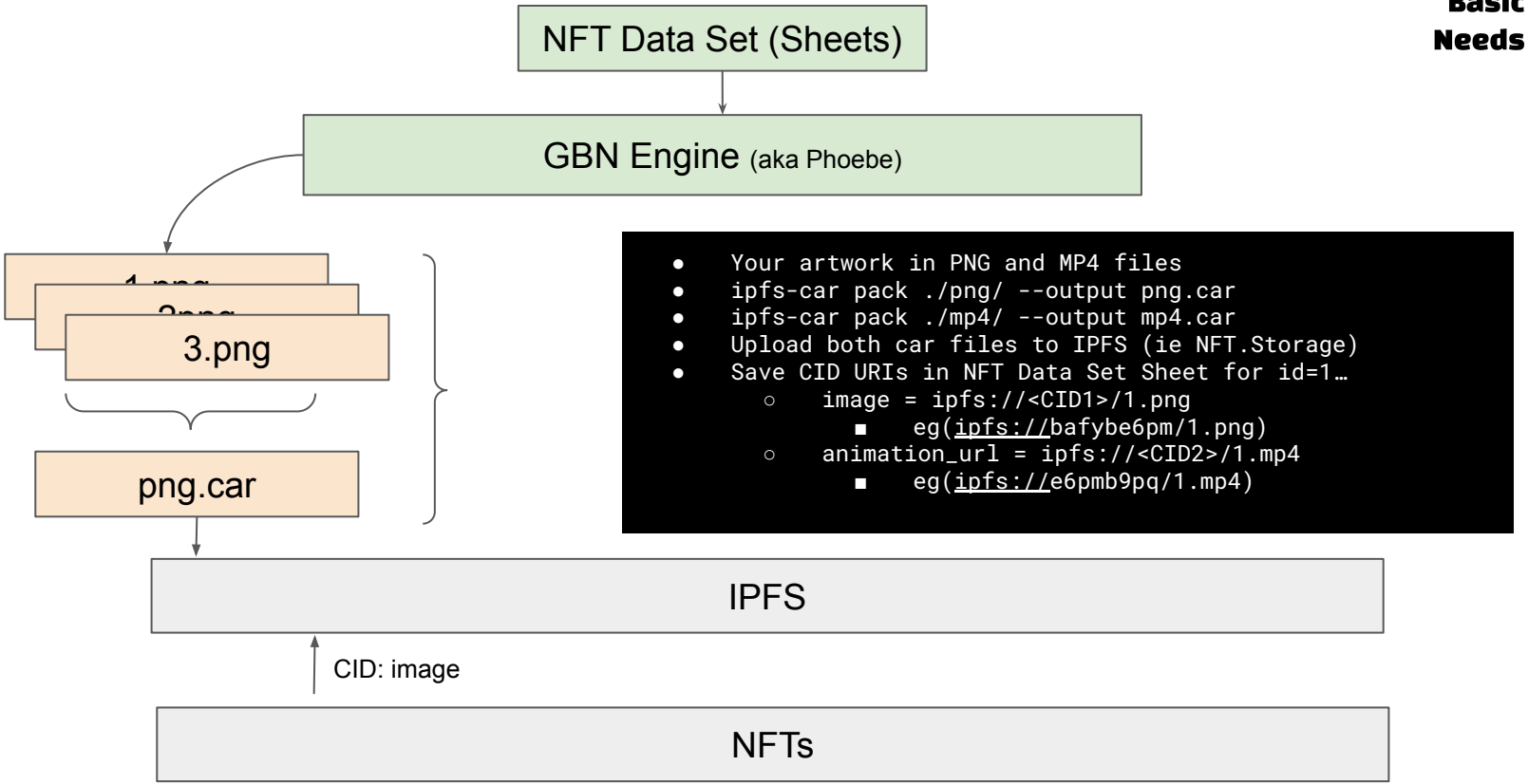
20k Foot View…

Give Basic Needs
Food / Shelter / Medical



1k Foot View PNGs…

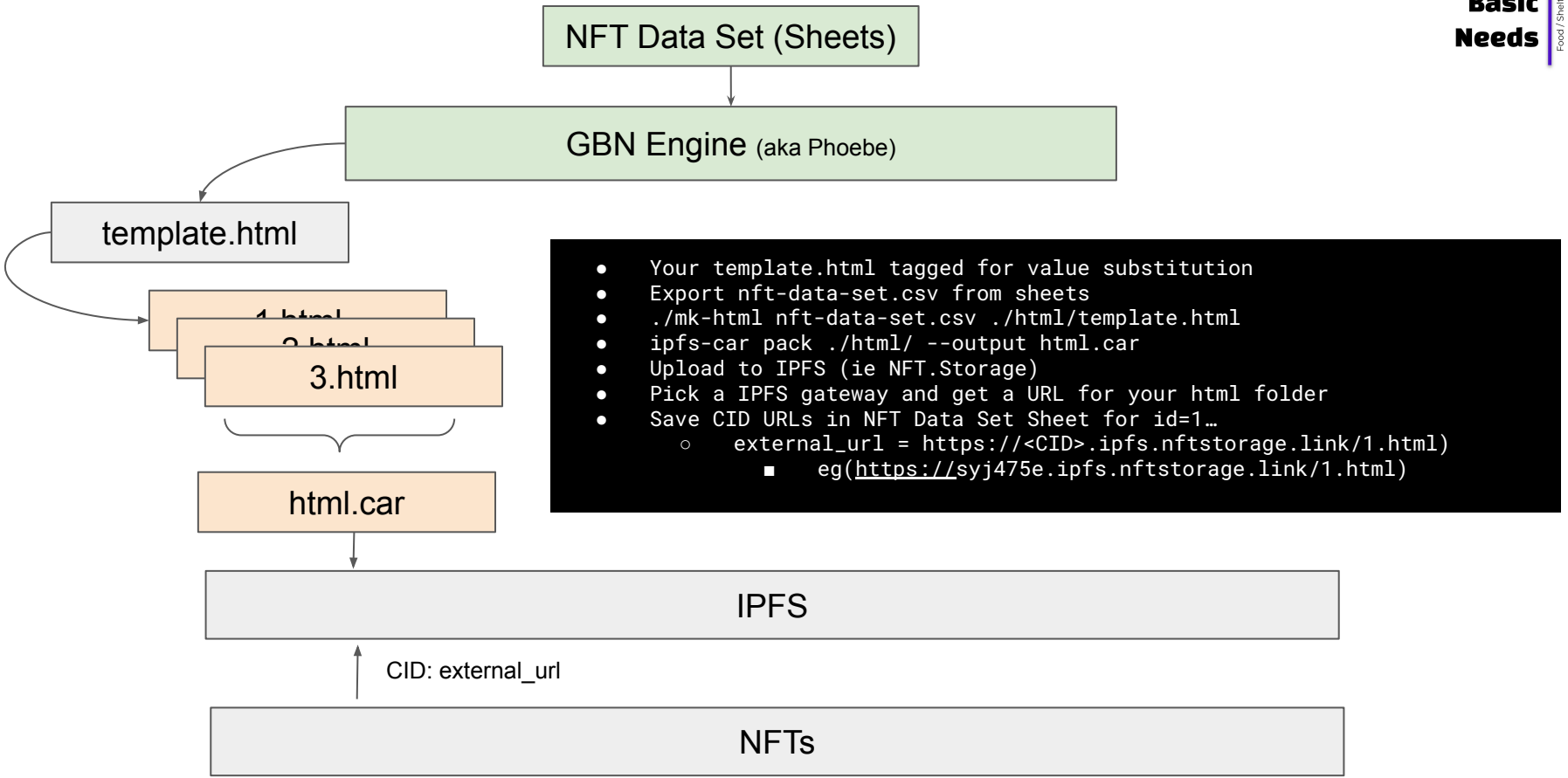
Give Basic Needs
Food / Shelter / Medical



- Your artwork in PNG and MP4 files
- ipfs-car pack ./png/ --output png.car
- ipfs-car pack ./mp4/ --output mp4.car
- Upload both car files to IPFS (ie NFT.Storage)
- Save CID URIs in NFT Data Set Sheet for id=1…
  - image = ipfs://<CID1>/1.png
    - eg(ipfs://bafybe6pm/1.png)
  - animation_url = ipfs://<CID2>/1.mp4
    - eg(ipfs://e6pmb9pq/1.mp4)

1k Foot View HTMLs…

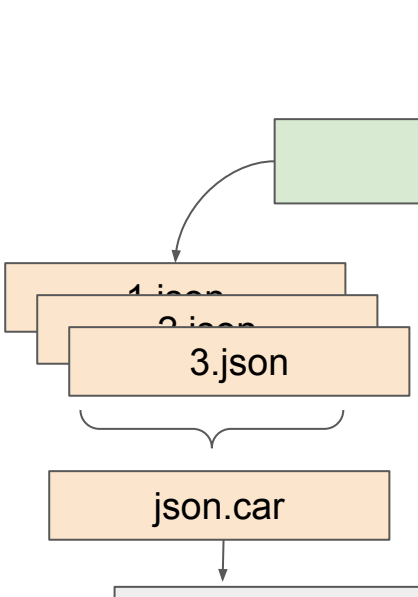
Give Basic Needs
Food / Shelter / Medical



- Your template.html tagged for value substitution
- Export nft-data-set.csv from sheets
- ./mk-html nft-data-set.csv ./html/template.html
- ipfs-car pack ./html/ --output html.car
- Upload to IPFS (ie NFT.Storage)
- Pick a IPFS gateway and get a URL for your html folder
- Save CID URLs in NFT Data Set Sheet for id=1…
  - external_url = https://<CID>.ipfs.nftstorage.link/1.html)
    - eg(https://syj475e.ipfs.nftstorage.link/1.html)

1k Foot View JSONs…

Give Basic Needs
Food / Shelter / Medical



- Config your mk-json value output for properties, attributes, traits and hidden_attributes
- Export nft-data-set.csv from sheets
- ./mk-json nft-data-set.csv
- ipfs-car pack ./json/ --output json.car
- Upload to IPFS (ie NFT.Storage)
- Save CID URI in the NFT Data Set Sheet for reference id=1…
  - json_uri = ipfs://<CID>/1.json
    - eg(ipfs://zadybe3pm/1.json)
- As you add tokens to your collection contract reference the json_uri to the metadata

Note: You may not need the .json file extension depending on how you are creating tokens

So Why A Concept?

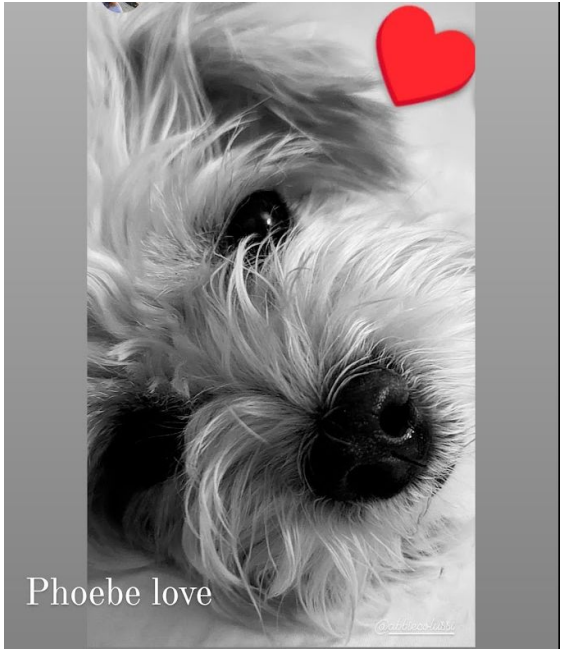
Give Basic Needs
Food / Shelter / Medical

- While we could create another code repository, it's not likely to be filled with much and I'm not going to maintain it, because I too am impermanent.
- Here are some helpful free tools that will get you 90% of the way right off the rip:
  - Google Sheets
  - Opensea Metadata Standard
  - Opensea Testnet
  - HeyMint Launchpad for ERC-1155 contract generation, deployment and management
  - LinuxOS
  - Emacs
  - ipfs-car
  - NFT.Storage (use a github login)
  - Cloudflare IPFS Gateway
    - https://cloudflare-ipfs.com/ipfs/QmNksJqvwHzNtAtYZVqFZFfdCVciY4ojTU2oFZQSFG9U7B
    - https://cloudflare-ipfs.com/ipfs/<CID>
  - Everything else – Google AI (Bard)
- Hope you find all this useful and enjoy Phoebe, its my tiny contribution to the community. It felt too good, not to share.

Credits / Inspiration:

Give Basic Needs

GiveBasicNeeds.org



If you would like to support our efforts please check out http://nft.givebasicneeds.org by, buying an NFT when they drop in Q1 2024, they are sure to make you and the world just a little happier.

Code Snips / Samples

Give Basic Needs
Food / Shelter / Medical

cvs-to-json.py

```
import argparse
import csv
import json
import os

# Define copywrite for each token
COPYWRITE = ["CC BY-NC 4.0 Attribution-NonCommercial 4.0 International"]

# Define the CSV columns that are attributes.
ATTRIBUTES = ["edition","card_group", "card_type", "rarity", "event_group", "location_name", "blog_post", "map", "travel_points", "happy_travel", "discord_invite"]

# Define the CSV columns that are hidden attributes.
HIDDEN_ATTRIBUTES = ["secret_extra_info", "secret_discord_invite_url", "secret_event_image", "secret_message"]

TRAIT_ORDER = ["edition","card_group", "card_type", "rarity", "event_group", "location_name", "blog_post", "map", "travel_points", "happy_travel", "discord_invite"]

def display_type(csv_header):
    """Determines the display type of a CSV column header.

    Args:
      csv_header: A string representing the CSV column header.

    Returns:
      A string representing the display type of the CSV column header.
    """
    case_statement = {
      "travel_points": "number",
      "discord_invite": "string",
      "blog_post": "url",
      "map": "address",
      "happy_travel": "address",
    }

    display_type = case_statement.get(csv_header, "string")

    return display_type

def generate_json_file(csv_row):
    """Generates a JSON file from a CSV row.

    Args:
      csv_row: A list of strings representing the CSV row.

    Returns:
      A Python dictionary representing the JSON object.
    """
    json_object = {}
    for i, csv_header in enumerate(csv_headers):
      if csv_header in ATTRIBUTES:
        if "attributes" not in json_object:
          json_object["attributes"] = []
        json_object["attributes"].append({
          "trait_type": csv_header,
          "display_type": display_type(csv_header),
          "value": csv_row[i]
        })
      elif csv_header in HIDDEN_ATTRIBUTES:
        if "hidden_attributes" not in json_object:
          json_object["hidden_attributes"] = {}
        json_object["hidden_attributes"][csv_header] = csv_row[i]
      else:
        json_object[csv_header] = csv_row[i]

    return json_object

def main():
    """Generates individual JSON files from the CSV file sample."""

    # Parse the command line arguments.
    parser = argparse.ArgumentParser()
    parser.add_argument("--input_file", type=str, required=True, help="The path to the input CSV file.")
    parser.add_argument("--output_directory", type=str, required=True, help="The path to the output directory.")
    parser.add_argument("--output_file_extension", type=str, default=".json", help="The output file extension.")
    args = parser.parse_args()

    # Get the input file path and output directory from the command line arguments.
    csv_input_file_path = args.input_file
    output_directory = args.output_directory
    output_file_extension = args.output_file_extension

    # Create the output directory if it does not exist.
    if not os.path.exists(output_directory):
      os.makedirs(output_directory)

    # Open the CSV input file.
    with open(csv_input_file_path, 'r') as csv_file:
      csv_reader = csv.reader(csv_file)

      # Get the CSV headers.
      global csv_headers
      csv_headers = next(csv_reader)

      # Iterate over the CSV rows and generate JSON files.
      for csv_row in csv_reader:
        json_object = generate_json_file(csv_row)
        json_object["trait_order"] = TRAIT_ORDER
        json_object["copywrite"] = COPYWRITE

        # Write the JSON object to a file.
        output_file_path = os.path.join(output_directory, f"{csv_row[0]}{output_file_extension}")
        with open(output_file_path, 'w') as json_file:
          json.dump(json_object, json_file, indent=4)

if __name__ == "__main__":
    main()
```

MD5 function Sheets

```
/**
* Compute MD5 digest
*
* @param {string} input Input for the MD5 digest algorithm
* @customfunction
*/
const MD5 = (input) => Utilities
  .computeDigest(Utilities.DigestAlgorithm.MD5, input)
  .map((hashByte) => ((hashByte + 256) % 256).toString(16).padStart(2, '0'))
  .join('')
  .toUpperCase();
```

CSV Header Line Sheets

```
id,name,description,image,animation_url,external_url,event_id,geocords,compiler,rarity_color_hex,background_color_hex,card_type_color_hex,est_human_impact,power_rating,created_date,edition,card_type,rarity,event_group,blog_post,location_name,map,travel_points,happy_travel,discord_invite,secret_extra_info,secret_discord_invite_url,secret_event_image,secret_message
```

mk-json.sh

```
#!/bin/bash

echo ""
# Prompt the user with a question
read -p "Have you updated all the URIs in the NFT-Plan '_lookup' sheet? Did you use the NFT-Plan macro to generate the 'NFT-Plan Export.csv' file and is the newly downloaded file in this scripts parent directory? (yes/no) " yn

# Check the user's answer
case $yn in
    [Yy]* )
                                    echo "" """ "
                                    ls "Export.csv
                                    echo """ "
                                    echo ""

                                    file_name=""Export.csv"
                                    echo "$file_name"

                                    if [ ! -f $file_name ]; then
                                        echo "The CSV file does not exist."
                                        exit 1
                                    fi

                                    echo "Removing export.csv"
                                    rm export.csv

                                    echo "Moving file to export.csv"
                                    mv "Export.csv export.csv

                                    echo ""
                                    echo """" Building json car files... """"
                                    echo "Using: export.csv"
                                    echo ""
        # Run the command if the user answers yes
                                    echo "Generating new json car files..."
                                    echo "Clean up files..."
                                    rm ./json/*
                                    rm ./json-ext/*

                                    echo "Gen JSON files..."
                                    python3 csv-to-json.py --input ./export.csv --output_directory ./json --output_file_extension ""
                                    python3 csv-to-json.py --input ./export.csv --output_directory ./json-ext --output_file_extension ".json"

                                    echo "Gen json CAR file..."
                                    ipfs-car pack ./json --output ./cars/gbn-json.car
                                    echo "Gen json-ext CAR file..."
                                    ipfs-car pack ./json-ext --output ./cars/gbn-json-ext.car
                                    echo ""
                                    echo " Note: You must get the ipfs URI after you upload each json car files via NFT.Storage and update the Google Sheet (NFT-Plan _lookup 'JSON Files w/ No Ext' & 'JSON Files w/ Ext' the new ipfs URI
address for completeness! Don't forget."
    [Nn]* )
        # Do nothing if the user answers no
                                    echo ""
        echo "You must update the URIs, run the macro, save the csv file to the parent directory before generating the json car files! Come back soon..."
        ;;
    * )
        # Exit the script if the user enters an invalid response
                                    echo "Invalid response."
        exit 1
        ;;
esac
echo ""
echo "Done."
```

Export / Download function Sheets

```
function CreateCleanExportTab() {
var ss = SpreadsheetApp.getActiveSpreadsheet();
var exportSheet = ss.getSheetByName("Export");
var metadataSheet = ss.getSheetByName("Metadata");

if (exportSheet) {
exportSheet.getDataRange().clearContent();
}

// Copy the values from the Metadata sheet to the Export sheet.
metadataSheet.getRange("A3:AC").copyTo(exportSheet.getRange("A1"), SpreadsheetApp.CopyPasteType.PASTE_VALUES, false);

// Copy the values from the Metadata sheet to the Export sheet.
// metadataSheet.getRange('AD3:AD').copyTo(exportSheet.getRange('AD1'), SpreadsheetApp.CopyPasteType.PASTE_NORMAL, false);
metadataSheet.getRange("AC3:AC").copyTo(exportSheet.getRange("AC1"), SpreadsheetApp.CopyPasteType.PASTE_NORMAL, false);

SpreadsheetApp.flush();
exportSheet.getRange("A1").activate();
downloadCSV();
};

function downloadCSV() {
var ss = SpreadsheetApp.getActive();
var sheets = ss.getSheets();
var urlList = "";
for (var i = 0; i < sheets.length; i++) {
urlList += '</br><a class="tablink mb-1 p-1"
href="https://docs.google.com/spreadsheets/d/${ss.getId().toString()}/export?gid=${sheets[i].getSheetId().toString()}&format=csv">${sheets[i].getName().toString()}</a>';
}
var html = HtmlService.createHtmlOutput('<html></head><link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
crossorigin="anonymous"></head><body><div class="d-flex flex-wrap m-2 h-75">${urlList}</div><div class="fixed-bottom p-2"><button class="btn m-25 btn-primary"
onclick="google.script.host.close()">Done</button></div></body></html>');
SpreadsheetApp.getUi().showModalDialog(html, 'CSV download interface');
}
```

Code Snips / Samples

Give Basic Needs
Food / Shelter / Medical

2.json


```
{
  "id": "2",
  "name": "#2  Super Rare Borehole (G247)",
  "description": "Owning a Give Basic Needs NFT is more than just a transaction. It's a heartfelt gesture that helps us make the world a better place, one NFT at a time. By sponsoring a completed project like this one in Pulo, Ghana (9.61999,-0.40608) you enable us to allocate funds to launch a new initiative, effectively creating a ripple effect of kindness. Your single action multiplies into a sequence of projects that touch lives worldwide! \ud83d\udc9d\ud83d\udc4f",
  "image": "ipfs://bafybeiel74ipxj7s5hn3nmqbkd5ak65yaqueqdtdvnvlueigshxiyla6pm2.png",
  "animation_url": "ipfs://bafybeiber4cxm2ld3cmwwg3dcarbcb2ihjnkl56dx5xeyrk3jlo4vkem2.mp4",
  "external_url": "https://nft.givebasicneeds.org/",
  "edition": "2",
  "geocoord": "9.61999,-0.40608",
  "compiler": "GBN NFT Engine (givebasicneeds.org)",
  "rarity_color_hex": "#f9f633",
  "background_color_hex": "#FFFFFF",
  "card_type_color_hex": "#b3dde1",
  "est_human_impact": "400",
  "power_rating": "50",
  "created_date": "1697932800",
  "attributes": [
    {
      "trait_type": "edition",
      "display_type": "string",
      "value": "G247"
    },
    {
      "trait_type": "card_type",
      "display_type": "string",
      "value": "Food"
    },
    {
      "trait_type": "rarity",
      "display_type": "string",
      "value": "Super Rare"
    },
    {
      "trait_type": "event_group",
      "display_type": "string",
      "value": "Borehole"
    },
    {
      "trait_type": "blog_post",
      "display_type": "url",
      "value": "https://givebasicneeds.org/events/borehole-project-4"
    },
    {
      "trait_type": "location_name",
      "display_type": "string",
      "value": "Pulo, Ghana"
    },
    {
      "trait_type": "map",
      "display_type": "address",
      "value": "https://www.google.com/maps/@9.61999,-0.40608,13z"
    },
    {
      "trait_type": "travel_points",
      "display_type": "number",
      "value": "2308"
    },
    {
      "trait_type": "trappy_travel",
      "display_type": "address",
      "value": "http://nft.travel.givebasicneeds.org"
    },
    {
      "trait_type": "discord_invite",
      "display_type": "string",
      "value": "http://discord.givebasicneeds.org"
    }
  ],
  "hidden_attributes": {
    "secret_extra_info": "http://metadata.nft.givebasicneeds.org",
    "secret_discord_invite_url": "http://2ef1461264.discord.givebasicneeds.org",
    "secret_event_image": "ipfs://bafybeibuzym56lih7vgrgbfcfl26jxcnzf33n7sf5zne2ivkeiefikiffa/F-4-34093.jpg",
    "secret_message": "2-507D47CBD12897"
  },
  "trait_order": [
    "edition",
    "card_group",
    "card_type",
    "rarity",
    "event_group",
    "location_name",
    "blog_post",
    "map",
    "travel_points",
    "trappy_travel",
    "discord_invite"
  ],
  "copyrights": [
    "CC BY-NC 4.0 Attribution-NonCommercial 4.0 International"
  ]
}
```