# 286.https://stackoverflow.com/questions/71790267/hashlips-art-engine-1-1-2-patch-v5-the-path-argument-must-be-of-type-string

**T:**(hashlips\_art\_engine-1.1.2\_patch\_v5) The "path" argument must be of type string. Received undefined

**Q:**I'm trying to update some metadata for 10 JSON files, some of them get updated properly and some others give me the following error.  
  
Below you can see 6 files give me errors and 4 files are correctly updated.  
  
error:TypeError [ERR\_INVALID\_ARG\_TYPE]: The "path" argument must be of type string. Received undefined  
  
error:TypeError [ERR\_INVALID\_ARG\_TYPE]: The "path" argument must be of type string. Received undefined  
  
error:TypeError [ERR\_INVALID\_ARG\_TYPE]: The "path" argument must be of type string. Received undefined  
  
error:TypeError [ERR\_INVALID\_ARG\_TYPE]: The "path" argument must be of type string. Received undefined  
  
error:TypeError [ERR\_INVALID\_ARG\_TYPE]: The "path" argument must be of type string. Received undefined  
  
error:TypeError [ERR\_INVALID\_ARG\_TYPE]: The "path" argument must be of type string. Received undefined  
  
5.png uploaded & 5.json updated!  
  
4.png uploaded & 4.json updated!  
  
1.png uploaded & 1.json updated!  
  
7.png uploaded & 7.json updated!  
  
This is the node.js code im using:  
  
onst FormData = require('form-data');const fetch = require('node-fetch');const path = require("path");const basePath = process.cwd();const fs = require("fs");fs.readdirSync(`${basePath}/build/images`).forEach(file => { const formData = new FormData(); const fileStream = fs.createReadStream(`${basePath}/build/images/${file}`); formData.append("file", fileStream); let url = 'https://api.nftport.xyz/v0/files'; let options = { method: 'POST', headers: { Authorization: 'SecretApiCode', }, body: formData }; fetch(url, options) .then(res => res.json()) .then((json) => { const fileName = path.parse(json.file\_name).name; let rawdata = fs.readFileSync(`${basePath}/build/json/${fileName}.json`); let metaData = JSON.parse(rawdata); metaData.file\_url = json.ipfs\_url; fs.writeFileSync(`${basePath}/build/json/${fileName}.json`, JSON.stringify(metaData, null, 2)); console.log(`${json.file\_name} uploaded & ${fileName}.json updated!`); }) .catch(err => console.error('error:' + err));});  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
I have 10png files:  
  
1.png2.png3.png4.png5.png6.png7.png8.png9.png10.png  
  
And 10 JSON files  
  
1.json2.json3.json4.json5.json6.json7.json8.json9.json10.json

2 **Answer**

**A1:**The issue in your code seems to be with fetching the URL.  
  
This is proven as sometimes, your code fails with an error, and sometimes, it works successfully.  
  
What might be happening is that sometimes, when you are fetching the data from the URL, it might not have the file\_name property (because of an error, etc.)  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
When you are trying to parse the path with the below method:  
  
path.parse(json.file\_name);  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
The file\_name might be undefined as there might not be such a key. And, as path.parse() can't parse undefined, it returns an error, saying that in only accepts strings (shown below).  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
TypeError [ERR\_INVALID\_ARG\_TYPE]: The "path" argument must be of type string. Received undefined  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
There are many possible solutions, so they are listed below.  
  
 ● Add a check for if file\_name is available. If it isn't, then don't try parsing it with path.parse(). You can use the code below to check it (in the second .then() method).  
if (!json.file\_name) {  
 console.error("Error fetching file name.");  
 return;  
}  
  
  
  
 ● If your API supports it, check if there is an error in the response body. Assuming that the error will be something like .error, you can use the code below for guidance (add it in the second .then() method).  
if (!json.file\_name && json.error) {  
 console.log(json.error.message);  
 return;  
}  
  
  
  
 ● There may be many other (better) solutions then these, but these solutions are just to give you a starting point.  
  
  
Solution 1 and 2 are very similar, but they work under different circumstances.

**C1:**Hi Arnav, I got a bit stuck tbh. On the fetch section I checked as you said if (!json.file\_name) and indeed it seems I can't fetch the file\_name as I got the errors in console for this checks , but not sure why the file\_name does not get fetched for some and it does for others, and how to fix it.

**C2:**fetch(url, options) .then(res => res.json()) .then((json) => { if (!json.file\_name) { console.error("Error fetching file name."); return; }

**C3:**And get following logs: Error fetching file name. Error fetching file name. Error fetching file name. Error fetching file name. Error fetching file name. Error fetching file name. Error fetching file name. 10.png uploaded & 10.json updated! 5.png uploaded & 5.json updated! 4.png uploaded & 4.json updated!

**A2:**Try this. You can also find it in codeSTACKr/create-10k-nft-collection" git repository if you change tag to v1.2.0.  
  
This is in uploadfiles.js  
  
const FormData = require("form-data");const fetch = require("node-fetch");const path = require("path");const basePath = process.cwd();const fs = require("fs");const AUTH = 'Your API KEY GOES HERE';const TIMEOUT = 1000; // Milliseconds. Extend this if needed to wait for each upload. 1000 = 1 second.const allMetadata = [];async function main() { const files = fs.readdirSync(`${basePath}/build/images`); files.sort(function(a, b){ return a.split(".")[0] - b.split(".")[0]; }); for (const file of files) { const fileName = path.parse(file).name; let jsonFile = fs.readFileSync(`${basePath}/build/json/${fileName}.json`); let metaData = JSON.parse(jsonFile); if(!metaData.file\_url.includes('https://')) { const response = await fetchWithRetry(file); metaData.file\_url = response.ipfs\_url; fs.writeFileSync( `${basePath}/build/json/${fileName}.json`, JSON.stringify(metaData, null, 2) ); console.log(`${response.file\_name} uploaded & ${fileName}.json updated!`); } else { console.log(`${fileName} already uploaded.`); } allMetadata.push(metaData); } fs.writeFileSync( `${basePath}/build/json/\_metadata.json`, JSON.stringify(allMetadata, null, 2) );}main();function timer(ms) { return new Promise(res => setTimeout(res, ms));}async function fetchWithRetry(file) { await timer(TIMEOUT) return new Promise((resolve, reject) => { const fetch\_retry = (\_file) => { const formData = new FormData(); const fileStream = fs.createReadStream(`${basePath}/build/images/${\_file}`); formData.append("file", fileStream); let url = "https://api.nftport.xyz/v0/files"; let options = { method: "POST", headers: { Authorization: AUTH, }, body: formData, }; return fetch(url, options).then(async (res) => { const status = res.status; if(status === 200) { return res.json(); } else { console.error(`ERROR STATUS: ${status}`) console.log('Retrying') await timer(TIMEOUT) fetch\_retry(\_file) } }) .then(async (json) => { if(json.response === "OK"){ return resolve(json); } else { console.error(`NOK: ${json.error}`) console.log('Retrying') await timer(TIMEOUT) fetch\_retry(\_file) } }) .catch(async (error) => { console.error(`CATCH ERROR: ${error}`) console.log('Retrying') await timer(TIMEOUT) fetch\_retry(\_file) }); } return fetch\_retry(file); });}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]