Click Here

Do you say, Saved in my folder or saved on my folder. Thanks for the help saved in my folder - it is saved inside a metaphorical folder, not on top of one. Exactly. However there are some cases where it's tough to decide on whether to use "in" or "on." A store can be located "on" or "at" the corner (of two streets). Since we don't think if it as being in the middle of the intersection, we don't say "in the corner." However a chair or a person can be situated "in the corner" (of a room). Best, cew I always thought it was “the documents are saved in my folder”. I was asking the question because some of my colleagues always say "saved on my folder" and they are English native speakers I am not. Thanks you for the clarification. I always thought it was “the documents are saved in my folder”. I was asking the question because some of my colleagues always say "saved on my folder" and they are English native speakers I am not. Thanks you for the clarification. You are right, however note that the terminology varies depending on where you are saving it: "They are saved on my computer." "They are saved on my hard drive." "They are saved on this disk." "They are saved in a folder." Thank you for the clarification We use "in" for books and folders. Thus if I were reading a book and did not want to converse, I might say, "Not now, I'm in the middle of a good book." I've never heard anyone who spoke English as a native say that something was saved "on" his/her folder. We say, "That is saved on my desktop" (of a computer) but of course it is "saved in" the "My Documents" folder. Best, cew Hi! I always thought it was “the documents are saved in my folder”. I was asking the question because some of my colleagues always say "saved on my folder" and they are English native speakers I am not. Thanks you for the clarification. I just realized that we can say that something (an address, or another record usually) "is on file" (which means that we have it either in our file folders that we maintain or in our computers files or both--and thus that we don't need to obtain it again or elaborate more about it perhaps). Thus, "on file" means "in the files we maintain," and it's used to describe the general location of data records (such as addresses). However, what it really says is that we have these data records and can access them/look up the data in them rather than specifying the location of the data records. I hope this is not confusing. Best, cew Yes, I thing I got it when we say “on file” is when we have recorded something and when “is saved in a file”, it’s referred to location. Thanks for your help Luisiher Yes, that's right. You understand it. "On file" indicates that a record is availabe. "In the/a [particular] file" specifies the location something is saved in. The etymology of the English prepositions "in" and "on" is interesting--apparently, to some degree, they share a root: (According to the resource above and also according to my offline dictionary, "on" used to be used where we use "in" and "on" was in Old English an unstressed variant of in.) Best, cew Last edited: Jul 22, 2009 These prepositions drive crazy. Now another similar question (I am not sure if I should open a new thread as this is related) To improve on or to improve in I think is to improve in For example I need to improve in my piano skills Thanks again Hi. Yes this is related to the 'on' and 'in' question, but improve on /in is a different idiom than save on/in and has a different meaning. So I'd start a new thread (the moderators will probably suggest that you do so too). Best, cew 将一个或多个 Docker 镜像保存到一个 tar 归档文件中，以便在其他环境中分发或备份。# 语法：docker save [OPTIONS] IMAGE [IMAGE...] # 保存单个镜像到文件 docker save -o myimage.tar myimage:latest # 保存多个镜像到同一个文件 docker save -o multiple_images.tar image1:latest image2:latest # 注意： # 1、保存镜像时，会包含镜像的所有层，因此生成的 tar 文件可能会很大。 # 2、如果保存多个镜像到同一个文件中，使用 docker load 命令时会加载所有包含的镜像。 # 3、为了减少文件大小，可以在保存前使用 docker image prune 命令清理未使用的镜像和层。load 命令用于从由 docker save 命令生成的 tar 文件中加载 Docker 镜像。# 语法：docker load [OPTIONS] # 从文件加载镜像 docker load -i myimage.tar # 从标准输入加载镜像 cat myimage.tar | docker load # 参数： # -i, --input: 指定输入文件的路径 # -q, --quiet: 安静模式，减少输出信息build 命令用于从 Dockerfile 构建 Docker 镜像。docker build 命令通过读取 Dockerfile 中定义的指令，逐步构建镜像，并将最终结果保存到本地镜像库中。# 语法：docker build [OPTIONS] PATH | URL | -常用参数：PATH: 包含 Dockerfile 的目录路径或 .（当前目录）URL: 指向包含 Dockerfile 的远程存储库地址（如 Git 仓库）-: 从标准输入读取 Dockerfile常用选项：-t, --tag: 为构建的镜像指定名称和标签-f, --file: 指定 Dockerfile 的路径（默认是 PATH 下的 Dockerfile）--build-arg: 设置构建参数--no-cache: 不使用缓存层构建镜像--rm: 构建成功后删除中间容器（默认开启）--force-rm: 无论构建成功与否，一律删除中间容器--pull: 始终尝试从注册表拉取最新的基础镜像更多选项：--build-arg=[]: 设置构建镜像时的变量--cpu-shares: 设置 CPU 使用权重-f: 指定 Dockerfile 的路径--force-rm: 强制在构建过程中删除中间容器--isolation: 使用指定的容器隔离技术--label=[]: 设置镜像的元数据-m: 设置内存的最大值--memory-swap: 设置交换空间的最大值（内存 + 交换空间），-1 表示不限制交换空间--no-cache: 构建镜像时不使用缓存--shm-size: 设置 /dev/shm 的大小，默认值为 64M--ulimit: 设置 Ulimit 配置--network: 在构建期间设置 RUN 指令的网络模式，默认值为 default其它命令docker images 命令用于列出本地的 Docker 镜像。2、容器生命周期管理run 命令docker run 命令用于创建并启动一个新的容器。# 语法：docker run [OPTIONS] IMAGE [COMMAND] [ARG...]常用参数：-d: 后台运行容器并返回容器 ID。-it: 交互式运行容器，分配一个伪终端。--name: 给容器指定一个名称。-p: 端口映射，格式为 host_port:container_port。-v: 挂载卷，格式为 host_dir:container_dir。--rm: 容器停止后自动删除容器。--env 或 -e: 设置环境变量。--network: 指定容器的网络模式。--restart: 容器的重启策略（如 no、on-failure、always、unless-stopped）。-u: 指定用户。简单示例# 1. 基本使用 docker run ubuntu # 拉取 ubuntu 镜像并在前台启动一个容器。 # 2. 后台运行容器 docker run -d ubuntu # 在后台运行 ubuntu 容器并返回容器 ID。 # 3. 交互式运行并分配终端 docker run -it ubuntu /bin/bash # 以交互模式运行 ubuntu 容器，并启动一个 Bash shell。 # 4. 指定容器名称 docker run --name my_container ubuntu # 运行一个 ubuntu 容器，并将其命名为 my_container。 # 5. 端口映射 docker run -p 8080:80 nginx # 将本地主机的 8080 端口映射到容器内的 80 端口，运行 nginx 容器。 # 6. 挂载卷 docker run -v /host/data:/container/data ubuntu # 将主机的 /host/data 目录挂载到容器内的 /container/data 目录。 # 7. 设置环境变量 docker run -e MY_ENV_VAR=my_value ubuntu # 设置环境变量 MY_ENV_VAR 的值为 my_value，运行 ubuntu 容器。 # 8. 使用网络模式 docker run --network host nginx # 使用主机的网络模式运行 nginx 容器。 # 9. 指定重启策略 docker run --restart always nginx # 设置容器的重启策略为 always，即使容器停止也会自动重启。 # 10. 指定用户 docker run -u user123 ubuntu # docker run -u user123 ubuntu # 11. 组合多个选项 docker run -d -p 8080:80 -v /host/data:/data --name webserver nginx # 后台运行一个命名为 webserver 的 nginx 容器，将主机的 8080 端口映射到容器的 80 端口，并将主机的 /host/data 目录挂载到容器的 /data 目录。exec 命令docker exec 命令用于在运行中的容器内执行一个新的命令。这对于调试、运行附加的进程或在容器内部进行管理操作非常有用。# 语法：docker exec [OPTIONS] CONTAINER COMMAND [ARG...]常用参数-d, --detach: 在后台运行命令。-i, --interactive: 保持标准输入打开。--workdir, -w: 指定命令的工作目录。-t, --tty: 分配一个伪终端。--privileged: 给这个命令额外的权限。-e, --env: 设置环境变量。--env-file: 从文件中读取环境变量。简单示例# 1、在容器内运行命令，在运行中的 my_container 容器内执行 ls /app 命令，列出 /app 目录的内容。docker exec my_container ls /app # 2、以交互模式运行命令，注意不一定是/bin/bash，可能是/bin/sh、sh、bash等等 docker exec -it my_container /bin/bash # 在运行中的 my_container 容器内启动一个交互式的 Bash shell。-i 保持标准输入打开，-t 分配一个伪终端。 # 后台运行命令，在运行中的 my_container 容器内后台执行 touch /app/newfile.txt 命令，创建一个新文件。docker exec -d my_container touch /app/newfile.txt # 3、设置环境变量，在运行中的 my_container 容器内执行 env 命令，并设置环境变量 MY_ENV_VAR 的值为 my_value。 docker exec -e MY_ENV_VAR=my_value my_container env # 4、以指定用户身份运行命令，在运行中的 my_container 容器内以 user123 用户身份执行 whoami 命令。 docker exec -u user123 my_container whoami # 5、指定工作目录，在运行中的 my_container 容器内以 /app 目录为工作目录执行 pwd 命令。 docker exec -w /app my_container pwd使用场景调试容器: 进入容器内部进行调试和排查问题。管理任务: 在容器内运行附加的管理任务或维护操作。监控和检查: 在容器内执行监控和检查命令，获取运行状态和日志。create 命令docker create 命令用于创建一个新的容器，但不会启动它。docker create 命令会根据指定的镜像和参数创建一个容器实例，但容器只会在创建时进行初始化，并不会执行任何进程。具体用法同 docker run 是一样的。其它命令docker start 命令用于启动一个或多个已经创建的容器。参数：-a: 附加到容器的标准输入输出流。-i: 附加并保持标准输入打开。docker stop 命令用于停止一个运行中的容器。参数：-t, --time: 停止容器之前等待的秒数，默认是 10 秒。docker restart 命令用于重启容器。docker kill 命令用于立即终止一个或多个正在运行的容器。docker rm 命令用于删除一个或多个已经停止的容器。docker pause - 暂停容器中所有的进程。docker unpause - 恢复容器中所有的进程。暂停的容器不会被终止，但其进程将被挂起，直到容器被恢复。这在需要临时暂停容器活动的情况下非常有用。使用场景临时暂停活动: 当需要临时暂停容器中的所有活动以进行系统维护或资源管理时，可以使用 docker pause。资源管理: 在需要重新分配系统资源时，暂停不必要的容器以释放资源。调试和故障排除: 在调试或故障排除过程中暂停容器以分析当前状态。docker rename 命令用于重命名已存在的容器。允许在不停止或删除容器的情况下，直接修改容器的名称。3、容器操作ps 命令docker ps 命令用于列出 Docker 容器。默认情况下，docker ps 命令只显示运行中的容器，但也可以通过指定选项来显示所有容器，包括停止的容器。# 语法：docker ps [OPTIONS]OPTIONS 说明：-a, --all: 显示所有容器，包括停止的容器。-q, --quiet: 只显示容器 ID。-l, --latest: 显示最近创建的一个容器，包括所有状态。-n: 显示最近创建的 n 个容器，包括所有状态。--no-trunc: 不截断输出。-s, --size: 显示容器的大小。--filter, -f: 根据条件过滤显示的容器。--format: 格式化输出。实例# 1、显示最近创建的一个容器 docker ps -l # 2、显示最近创建的 n 个容器 docker ps -n 3 # 3、显示容器的大小 docker ps -sinspect 命令docker inspect 命令用于获取 Docker 对象（容器、镜像、卷、网络等）的详细信息。docker inspect 命令返回 JSON 格式的详细信息，可以帮助用户了解对象的配置和状态。# 语法：docker inspect [OPTIONS] NAME|ID [NAME|ID...]OPTIONS 说明：-f, --format: 使用 Go 模板语法格式化输出。--type: 返回指定类型的对象信息（可选类型：container、image、network、volume）。使用场景调试容器: 获取容器的详细配置信息，以便进行调试和排查问题。查看网络配置: 查看容器的网络配置信息，了解其网络连接状态。监控资源: 获取容器的资源配置信息和使用情况，便于进行资源管理和监控。脚本自动化: 在自动化脚本中使用 docker inspect 获取对象的详细信息，以进行后续操作。top 命令docker top 命令用于显示指定容器中的正在运行的进程。docker top 命令类似于 Linux 中的 top 或 ps 命令，它帮助用户查看容器内的进程信息，便于监控和调试容器内的活动。# 语法：docker top [OPTIONS] CONTAINER [ps OPTIONS]使用自定义 ps 选项:docker top my_container -o pid,comm常用场景监控容器内部活动: 通过查看容器内的进程，用户可以监控容器内部正在运行的应用程序和服务。调试和排查问题: 当容器出现问题时，可以通过 docker top 命令查看容器内的进程，帮助排查问题。资源管理: 了解容器内的进程和资源使用情况，便于进行资源管理和优化。logs 命令docker logs 命令用于获取和查看容器的日志输出。docker logs 命令非常有用，可以帮助用户调试和监控运行中的容器。# 语法：docker logs [OPTIONS] CONTAINER常用选项：-f, --follow: 跟随日志输出（类似于 tail -f）。--since: 从指定时间开始显示日志。-t, --timestamps: 显示日志时间戳。--tail: 仅显示日志的最后部分，例如 --tail 10 显示最后 10 行。--details: 显示提供给日志的额外详细信息。--until: 显示直到指定时间的日志。实例# 1、显示带时间戳的日志 docker logs -t my_container # 2、从指定时间开始显示日志 docker logs --since="2023-07-22T15:00:00" my_container # 3、显示直到指定时间的日志 docker logs --until="2023-07-22T16:00:00" my_containercp 命令docker cp 命令用于在 Docker 容器和宿主机之间复制文件或目录。docker cp 命令支持从容器到宿主机，或从宿主机到容器的文件复制操作。# 语法 docker cp [OPTIONS] SRC_PATH CONTAINER:DEST_PATH docker cp [OPTIONS] CONTAINER:SRC_PATH DEST_PATHSRC_PATH: 源路径（可以是容器内的路径或宿主机的路径）。CONTAINER: 容器的名称或 ID。DEST_PATH: 目标路径（可以是容器内的路径或宿主机的路径）。# 1、从容器复制文件到宿主机 docker cp my_container:/path/in/container /path/on/host # 2、从宿主机复制文件到容器 docker cp /path/on/host my_container:/path/in/container注意在处理大文件或大目录时，复制操作可能需要一些时间。且宿主机或容器中应有足够的权限进行写入操作。其它命令docker attach 命令用于附加到正在运行的 Docker 容器的标准输入、输出和错误输出（stdin、stdout、stderr）。允许用户直接与容器交互，就像与正在运行的进程交互一样。docker events 命令用于实时获取 Docker 守护进程生成的事件。允许用户监控 Docker 容器、镜像、网络和卷的各种操作事件，例如创建、启动、停止、删除等。docker export 命令用于将 Docker 容器的文件系统导出为一个 tar 归档文件。用于备份或迁移容器的文件系统，而不包括 Docker 镜像的所有层和元数据。docker port 命令用于显示容器的端口映射信息，即容器内部的端口如何映射到宿主机的端口。docker stats 命令用于实时显示 Docker 容器的资源使用情况，包括 CPU、内存、网络 I/O 和块 I/O。docker update 命令用于更新 Docker 容器的资源限制，包括内存、CPU 等。docker commit 命令用于将容器的当前状态保存为一个新的 Docker 镜像。通常用于创建镜像来保存容器的状态，以便在将来可以重用或分发该镜像。4、网络命令docker network ls: 列出所有网络。docker network create : 创建一个新的网络。docker network rm : 删除指定的网络。docker network connect : 连接容器到网络。docker network disconnect : 断开容器与网络的连接。5、卷命令docker volume ls: 列出所有卷。docker volume create : 创建一个新的卷。docker volume rm : 删除指定的卷。docker volume inspect : 显示卷的详细信息。 最近在使用 Electron 做一个手写字体生成图片的工具。不可避免的，遇到了通过Electron 往本地存文件的问题。 在Electron 中，存取本地文件，有很多种办法。 通过 Electron 框架提供的能力，和 Node.js 的 fs 文件管理模块实现本地文件的存取。使用 app.getPath 和 fs 模块存储文件首先，我们可以通过 app.getPath 来获取当前用户的 data 存放目录。 app.getPath('userData') 返回一个字符串，该字符串表示应用程序的用户数据目录的路径。这个目录通常是用来存储用户相关的数据，例如配置文件、缓存等。具体路径会根据操作系统而变化。windows 系统中， 会返回类似（C:\Users\\AppData\Roaming\）这样的结果。Mac 系统中，会返回（/Users//Library/Application Support/）这种结果。linux 系统中 则会返回（/home//.config/）这种结果。我们通过 node.js 的path 模块， 使用 path.join(app.getPath('userData'), 'myFile.txt') 就能得到一个完整的文件路径。 接着使用 fs 模块的 来写入内容即可。示例代码如下：const { app } = require('electron'); const fs = require('fs'); const path = require('path'); const filePath = path.join(app.getPath('userData'),'/myfile.txt'); // Example file path try { fs.writeFileSync(filePath, 'hello world', 'utf-8'); } catch (e) { console.error('Failed to save the file!'); } 这种做法，有一个问题。那就是，用户不能在保存的时候，主动选择文件存放目录。使用 Dialog API 和 fs 模块配合Electron 提供了一个 Dialog API 来处理文件对话框。您可以专门调用一个保存对话框，让用户选择保存文件的位置。 下面是一个简单的，示例代码：const { dialog } = require('electron').remote; const fs = require('fs'); const options = { title: 'Save file', defaultPath: 'my_filename', buttonLabel: 'Save', filters: [{ name: 'txt', extensions: ['txt'] }, { name: 'All Files', extensions: ['*'] },], }; dialog.showSaveDialog(null, options).then(({ filePath }) => { if (filePath) { fs.writeFileSync(filePath, 'hello world', 'utf-8'); } });需要注意的点： 因为 fs 模块 和 Dialog 只能在，主进程中被调用。 但是我们的应用程序交互逻辑是在渲染进程，所以这段示例代码，只是演示了，如何去调用 Dialog 并手动选择文件存储位置。 并没有实际应用场景的参考意义。实际应用场景封装 调整对存取图片的封装想法跟之前的采集桌面思路基本一致（如果，没看过可以翻一下以前的文章）。 我们利用 Electron 的 ipcmain 模块在主进程中注册方法。 然后，在渲染进程去调用。 整理实现流程大概如下图所示。实例代码：// 主进程 -> main.js // .... ohter code // ... 在主进程注册我们封装后的 SaveFile 方法 const { app, BrowserWindow, ipcMain, dialog} = require("electron"); const path = require("path"); const fs = require('fs'); /** * @param options: { title: String, defaultPath: String, buttonLabel: String, filters: area} * @param content: String * @returns Promise */ ipcMain.handle('saveFile', async (event, content, options) => { let path; try { const { filePath } = await dialog.showSaveDialog(null, options); path = filePath; } catch(e) { return Promise.reject({error: e, success: false}); } if(path) { try { fs.writeFileSync(path, content, 'utf-8'); return Promise.resolve({error: null, success: true}); } catch(e) { return Promise.reject({error: e, success: false}); } } else { return Promise.reject({error: e, success: false, canceled: true}); } }); // 其他代码 ..../ Vue 文件、 // 在我们，的Vue 业务中。直接通过 ipc 调用 import {reactive, ref, onMounted} from "vue"; const textContent = ref('hello world'); const { ipcRenderer } = require('electron'); const exportImg = async () => { try { // 注意-> 第一个参数，为 主进程注册进来的方法名称。 // 其他参数为， 主进程注册的函数参数。 await ipcRenderer.invoke('saveFile', 'hello', { title: '导出图片', buttonLabel: '导出', defaultPath: 'text.txt', filters: [{ name: 'All Files', extensions: ['*'] }] }) } catch(e) { console.error('出错了！', e) } } 导出 总结在Electron 中，向本地存储和读取文件（文本或者是图片），都离不开 node.js 的 fs 模块 ，这个是文件系统处理的核心模块。 然后，我们搭配， path 模块。 dialog 模块，可以实现，用户主动选择存放位置。 或者 直接存到默认软件系统的 data 目录中。 当然，这些模块都只能在主进程中使用。 所以，我们不可避免的用到了， 之前的老朋友， ipc 进程间通讯方式。 本文，基于我在做一个手写照片 模拟器中遇到存储问题而来。 软件正在书写中。后续会开源，并放出使用的视频。 请关注我的后续文章。 I have a question related to ICT terminology. Which is the correct/preferred preposition in the following sentence: Save the file to your portfolio/Save the file in your portfolio. Thanks I'd use "for" in here but you should give us more context because I don't understand what you are really implying. Thanks for your reply. I'm sorry I wasn't able to express myself clearly. The sentence will be used in a task in which pupils will be asked to record a short text and save the sound file to a portfolio (a folder) on the digital learning management platform that we use at school. I instinctively wrote "Save the file to your portfolio", but then I wasn't so sure. In various computer software manuals I have seen the phrase "save to file", and I suppose that is why I wrote "to". Prepositions are difficult in the first place, and with all this new computer technology as well, it is not easy to write correctly! In reference to a computer portfolio, you could say either "save in" or "save to". Thank you, I appreciate your help Hi, there I would like to know if the correct is to say "save someone doing something" or "save someone from doing something"? Like: "You saved me from making a gaffe/being held up to ridicule/etc" or "You saved me making a gaffe/being held up to ridicule/etc"? Which one is correct? Thank you I think both forms are common but including "from" makes it a bit clearer. Semantically, there is no difference in that context. However: A: "Don't buy that camera here, buy it at the other shop, it is $100 cheaper." B(i): "Thanks! You have saved me $100" B(ii): "Thanks! You have saved me from $100" B(iii): "Thanks! You have saved me from wasting $100" A: "Don't move!" [A raises his gun and shoots a snake.] B: "You have saved me from the snake [killing/biting me]." B: "You have saved me the snake." B: "You have saved me from dying/being bitten." A: "I caught some parrots yesterday and I am going to sell them." B: "I want a parrot but I have no money at the moment, can you save one for me until I am paid?" Save (present) from an action, Save (cause to spend less on) an object. Save (to reserve) an object.