

TechTarget.com/searchdatacenter

<https://www.techtarget.com/searchdatacenter/definition/screen-scraping>

screen scraping

By Alexander S. Gillis

What is screen scraping?

Screen scraping is a data collection method used to gather information shown on a display to use for another purpose. Typically, screen scraping is used to collect [data](#) from one application to then translate it to another -- or, more controversially, to steal data.

Visual data is collected as plain text from onscreen information, such as text, images or charts that appear on the desktop, in an application or on a website.

By comparison, gathering data from a display through an automatic process like screen scraping is significantly faster and more efficient than collecting data manually.

Screen scraping can be performed automatically with a scraping program or manually by an individual extracting data. Screen scraping programs are designed to search through and recognize elements in a user interface ([UI](#)). Data from that display is extracted and transformed into text. When the displayed data contains images, a screen scraper utilizes optical character recognition ([OCR](#)) technology to gather information.

Screen scraping has a variety of uses, both ethical and unethical. An app for transferring data from a legacy application to a modern UI is an ethical use case. Stealing data from applications, on the other hand, is an unethical use for screen scraping. This might occur if a developer wants to steal code from another application to make the process of development faster and easier for their own project.

What is screen scraping used for?

Screen scrapers have been applied in a broad number of fields for a variety of use cases. Some potential uses include the following:

- Banking applications and financial transactions.
- Saving meaningful data for later use.
- Performing actions as a user would on a website.
- Translating data from a [legacy application](#) to a modern application.
- Aggregating data, such as for price comparison websites.
- Tracking user profiles to see online activities.
- Stealing data.

One of the largest use cases for screen scraping has been in banking. Lenders can use screen scraping to gather a customer's financial data. Screen scraping can also be used for mortgage provider applications. Financial service applications might use screen scraping to access multiple bank accounts from a user and aggregate all the information in one place. Users would need to explicitly trust the application, however, as they are entrusting it with their accounts, customer data and passwords.

An organization can use screen scraping to translate legacy application programs to a new UI so the logic and data associated with the legacy programs can continue to be accessed. This option is rarely used, however, and is

typically chosen only when other methods are impractical.

If an individual can gain access to the underlying code in an application, screen scraping can be used to steal the [code](#) for the user's own application. This would save the individual time and effort or enable them to learn how a feature in an application works without permission.

Screen scraping will sometimes involve a third-party system. An example of this is using screen scraping to enable a third-party organization to access data on financial transactions in a budgeting app.

Screen scraping has changed its main use cases over time. In 2019, screen scraping began to be phased out of banking, one of its larger use cases. This was done to ease security concerns surrounding the practice. Budgeting apps now must use a single open banking technology. To create more trust, some organizations route customer data through a secure application programming interface ([API](#)) [to avoid](#) the screen scraping process.

How does screen scraping work?

Screen scraping is done in several ways, depending on what the process is being used for. For example, through Java, individuals can copy and paste [source code](#) from one application into their own if they have direct access to it.

In general, screen scraping enables a user to extract screen display data from a specific UI element. With a desktop, web page or application, the screen scraper first spots onscreen elements that appear on the display to extract. When the displayed information includes an image, screen scrapers use OCR for extraction. The data is then transformed into text. Different methods can be used to obtain all the text on a page, which can be unformatted or formatted with exact positioning.

Screen scrapers can be based on an application such as Selenium or [PhantomJS](#), which enables users to obtain information from HTML in a browser. Unix tools, such as [shell scripts](#), can also be used as simple screen scrapers.

In banking, a third party will request users share their login information so it can access financial transaction data by logging into digital portals for the customers. A budgeting app can then retrieve the incoming and outgoing transactions across accounts.

While transferring data from a legacy program, a screen scraper must take the data coming from the legacy program, which is formatted for older terminals, and reformat it for newer operating systems or web browsers. The program must also reformat user inputs from the newer interface so the request can be handled as if it were the legacy application.

How to prevent screen scraping

Unfortunately, there is no definitive way to prevent unethical screen scraping from happening. However, there are ways to help deter it. An organization can detect screen scraping through a few given signatures or behaviors. For example, if a nonstandard user agent is detected, if [JavaScript](#) fails to run client-side or if several page request sequences are made, it could be a sign of screen scraping.

An organization can take these actions to help deter screen scraping:

- **Require a password login.** This won't stop screen scraping, but it can help identify who is performing it. If a page requires a login, then the scraper must send identifying information with each request, which helps identify who is doing the screen scraping.
- **Set a rate limit for individual IP addresses.** This will slow down requests from computers that make a large number of requests within a short amount of time, which can be a sign of screen scraping.

- **Use CAPTCHAs.** [CAPTCHAs](#) help separate real users from bots by presenting image-based information that computers have a difficult time understanding.
- **Use web application firewalls.** A [WAF](#) can help detect signature- or behavior-based actions.
- **Run fraud detection software.** This helps to catch screen scraping, potentially even while it is happening.
- **Set content to be shown as an image.** This won't stop screen scraping from happening, but it will stop programs that can't translate images.

All these methods can help deter screen scraping but won't stop it completely. In addition, organizations must make sure that their actions won't make the end-user experience worse. For example, setting a website's content to appear as an image can make it difficult for individuals to find the page, as it affects how search engines find the page as well.

Screen scraping benefits

Advantages of screen scraping include the following:

- **Speed.** Screen scraping lets users collect data that appears on a display at a much faster speed than if a user were to manually collect the data.
- **Scale.** Screen scraping can collect large volumes of data quickly.
- **Accuracy.** Manually collecting data is a repetitive task, leading a person to potentially make mistakes. Automating this process is a fast way to collect accurate data in a readable format.
- **Customization.** Screen scrapers can be customized to extract specific types of data.
- **Automatically structured data.** Scraped data is, by default, visible in a machine-readable format, enabling simple values to be used in other programs quickly.
- **Integration.** An automated screen scraping tool can be integrated with other applications, which aids in data processing and efficiency. For example, screen scraping software could be integrated with an organization's [customer relationship management](#) system to automatically update customer data.

Screen scraping drawbacks

Potential disadvantages of screen scraping include the following:

- **Maintenance.** If using screen scraping to collect data from web content, be mindful of when the website changes the page's structure or content, as this means the data would have to be re-scraped. The screen scraper might also need regular maintenance to keep running.
- **It does not analyze data.** Although screen scraping collects requested data and puts it into an acceptable format, the data still needs to be analyzed to pull any knowledge or insights. Likewise, complex data should be processed for further use in other programs.
- **Screen scraping can be used to steal content.** Although screen scraping can be used for legal purposes, others might use the data collection method to steal data for their own use.

Screen scraping tools

If individuals don't want to manually screen scrape, there are [several tools that can help automate the process](#):

- **UiPath.** The UiPath [robotic process automation](#) tool can be used for screen scraping by capturing bitmap data from a display and cross-checking it with the stored data for deciphering. UiPath supports full text, native and OCR screen scraping.
- **FMiner.** This screen scraping tool for Windows and macOS supports screen scraping, web scraping, web data extraction and [web crawling](#) methods for collecting data.
- **Macro Scheduler.** With Macro Scheduler, users can create macros and automate software processes for Windows applications. This app can be used to create a [script](#) to screen scrape data using methods such as

OCR.

- **ScreenScraper.** A tool for the development of apps and scripts, ScreenScraper Studio enables users to define what they want to be scraped, then generate code in languages such as C++, C#, Visual Basic 6.0 and JavaScript.
- **Existek.** Software developer Existek offers its Screen Scraping Software Automation for Desktop Apps, which includes OCR, system API interception, screen scraping plugins and browser extensions, and the ability to set standard APIs for scraping text.
- **Diffbot.** This data scraping tool enables users to automatically scrape text, videos and images. Scraped data can be specified and then processed in a [JSON](#) or CSV format.

Screen scraping vs. web scraping

Users have different scraping techniques to choose from when collecting data. While screen scraping is the process of extracting data shown on a screen, a web scraper extracts data from the web. The two concepts share many similarities to the point that web scraping can be seen as a specific type of screen scraping. The main differences lie in where the data is being taken from and what it is being used for.

Web scraping is used to extract data exclusively from the web -- unlike screen scraping, which can also scrape data from a user's desktop or applications. This form of data extraction can be used to compare prices for goods on an [e-commerce shop](#), for web indexing and for [data mining](#).

The process accesses the web through HTTP over a web browser and can either be done manually or automatically through a bot or web crawler.

There are numerous web scraping tools available, such as Octoparse, Apify and Bright Data.

Screen scraping vs. data scraping

Data scraping is a variant of screen scraping that is used to copy data from documents and web applications. Data scraping is a technique where structured, human-readable data is extracted. This method is mostly used for exchanging data with a legacy system and making it readable by modern applications.

Screen scraping and open banking

Open banking is the concept of sharing secured financial information that third-party developers can use for the creation of banking applications. This concept is based on the sharing of APIs, which lets an application use the same API to aggregate information from different accounts into one place. This is what enables a banking app to let users look at their multiple accounts from different banks in one place.

In the past, some banking apps would gather information using screen scraping. This process would require users to share their bank login credentials with the third-party app. The application would then log on to the user's accounts and screen scrape the needed data to show in the app.

By contrast, open banking now uses shared APIs, meaning the exact data needed is copied without requiring the user to share login credentials. The concept was introduced in 2018 and is now becoming a standard over the use of screen scraping.

Learn more about [how finance firms scrape data](#) from different sources.

03 May 2023

All Rights Reserved, [Copyright 2000 - 2025](#), TechTarget | [Read our Privacy Statement](#)