Runtime File Injection Example

Files that don't exist at compile time can be injected at runtime using the `add_file()` method. This is useful for dynamically generated content, downloaded resources, or any data that is only available when the program runs.

How It Works

The `Document` struct provides two methods for runtime file injection:

- `add_file(path, data)` — Inject a file at the given path. The file becomes available to the Typst template as if it were embedded at compile time.
- `has_file(path)` — Check whether a file exists (either embedded or injected at runtime). Useful for conditional rendering.

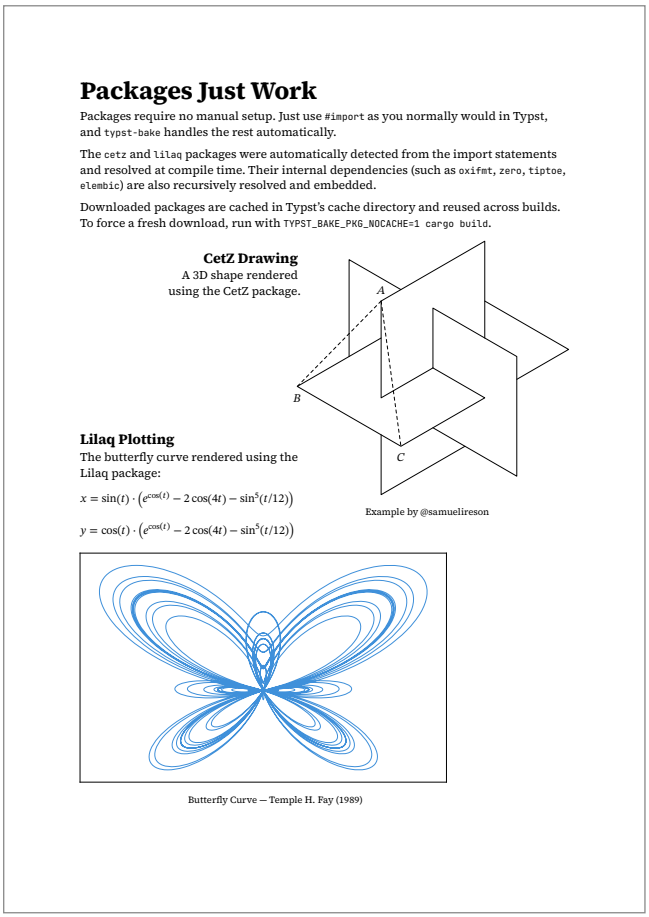
The following example downloads a PDF from a remote server and injects it into the document:

```rust
// Download a PDF at runtime
let url = "https://example.com/report.pdf";
let pdf_bytes = ureq::get(url).call()?.body_mut().read_to_vec()?;

// Inject it so the template can reference "downloaded.pdf"
let pdf = typst_bake::document!("main.typ")
    .add_file("downloaded.pdf", pdf_bytes)?
    .to_pdf()?;
```

Result

The PDF shown on the right was downloaded at runtime and injected into this document using `add_file()`.



Packages Just Work

Packages require no manual setup. Just use `#import` as you normally would in Typst, and `typst-bake` handles the rest automatically.

The `cetz` and `lilaq` packages were automatically detected from the import statements and resolved at compile time. Their internal dependencies (such as `oxifmt`, `zero`, `tiptoe`, `elembic`) are also recursively resolved and embedded.

Downloaded packages are cached in Typst's cache directory and reused across builds. To force a fresh download, run with `TYPST_BAKE_PKG_NOCACHE=1 cargo build`.

CetZ Drawing
A 3D shape rendered using the CetZ package.

Lilaq Plotting
The butterfly curve rendered using the Lilaq package:

$$x = \sin(t) \cdot \left(e^{\cos(t)} - 2\cos(4t) - \sin^5(t/12)\right)$$

$$y = \cos(t) \cdot \left(e^{\cos(t)} - 2\cos(4t) - \sin^5(t/12)\right)$$

Example by @samuelireson

Butterfly Curve — Temple H. Fay (1989)