typst-bake Quick Start

Step 1: Configure Cargo.toml

Add typst-bake to your dependencies and configure the template and font directories:

```toml
[dependencies]
typst-bake = "0.1"

[package.metadata.typst-bake]
template-dir = "./templates"
fonts-dir = "./fonts"
```

> **Tip**
>
> You can also use `TYPST_BAKE_TEMPLATE_DIR` and `TYPST_BAKE_FONTS_DIR` environment variables. If both are set, environment variables take priority.

Step 2: Prepare Files

Place all required files and fonts in the configured directories.

Template directory (`template-dir`): All files are embedded and can be accessed from `.typ` files.

Fonts directory (`fonts-dir`): Supports TTF, OTF, and TTC formats. At least one font is required. If your document includes equations or code blocks, add dedicated fonts for them. See the `font-guide` example for details.

Step 3: Define Inputs (Rust)

Define structs with derive macros to pass data to templates:

```rust
use typst_bake::{IntoValue, IntoDict};

#[derive(IntoValue, IntoDict)]
struct Inputs {
    title: String,
    discount: f64,
    products: Vec<Product>,
}

#[derive(IntoValue)]
struct Product {
    name: String,
    price: f64,
}
```

> **Info**
>
> **Top-level struct:**
> `IntoValue`, `IntoDict`
>
> **Nested structs:**
> `IntoValue`

Create input data:

```rust
let inputs = Inputs {
    title: "Sale".to_string(),
    discount: 20.0,
    products: vec![
        Product { name: "Apple".to_string(), price: 2.0 },
```

```
        Product { name: "Banana".to_string(), price: 1.0 },
    ],
};
```

Step 4: Write Typst Template

Access inputs via `sys.inputs` and use embedded files:

```typst
#import sys: inputs

#image("images/logo.png", width: 50%)

= #inputs.title (#inputs.discount% off)

#for product in inputs.products [
  - #product.name: $#product.price
]
```

> **Info**
>
> Using packages requires no manual setup. Just use `#import "@preview/...":` as you normally would in Typst, and `typst-bake` handles the rest automatically.

Step 5: Generate PDF

Generate PDF with `document!` macro:

```rust
let pdf = typst_bake::document!("main.typ")
    .with_inputs(inputs)
    .to_pdf()?;

std::fs::write("output.pdf", &pdf)?;
```

> **Tip**
>
> When no inputs are needed, call `.to_pdf()` directly without `.with_inputs()`. For SVG/PNG output, see the `output-formats` example.

Embedded Resources

`typst-bake` embeds all resources (templates, fonts, and packages) directly into the binary at compile time. To minimize binary size, resources are compressed using zstd and identical contents are deduplicated. Decompression is performed lazily at runtime—only when each resource is actually accessed.

The table below shows the resources embedded to generate this document.

Category	Original	Compressed	Ratio
Templates (7 files)	5.7 KB	2.9 KB	49.6%
Fonts (4 files)	1.57 MB	847.7 KB	47.1%
Packages (5)	2.83 MB	535.6 KB	81.5%

Compressed: 4.4 MB → 1.35 MB (69.2% reduced)
Deduplicated: 149 unique, 4 removed (−2.2 KB)
Total: 4.4 MB → 1.35 MB (69.3% reduced)

For more examples, see the `examples/` directory in the repository.