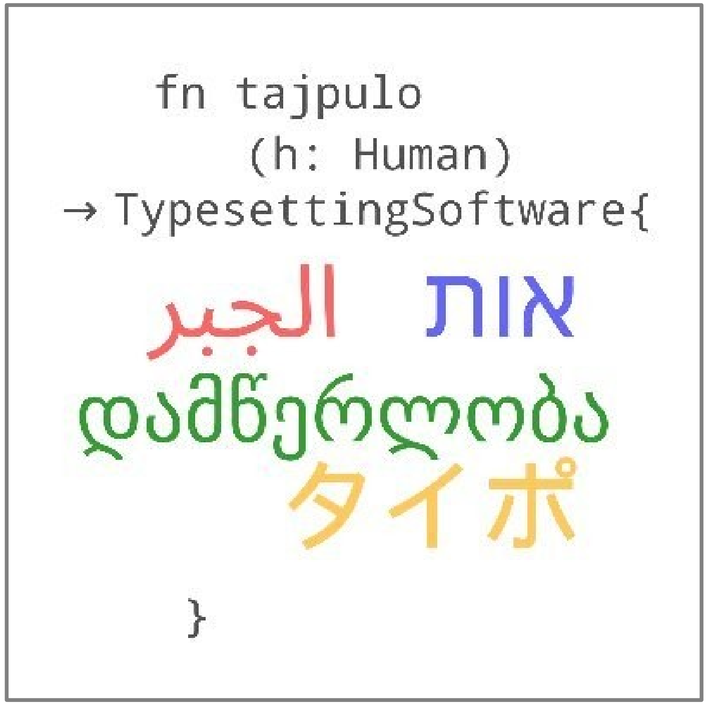
Are we digital typesetting yet?

2025-05-26
Grazer Linuxtage
https://lukas-prokop.at/talks/glt25-awdty

In memoriam

spel (1992 – 2025)

tajpulo



```
    fn tajpulo
         (h: Human)
→ TypesettingSoftware{
    الجبر   אות
  დამწერლობა
        タイポ

  }
```

my life goal:
provide you tools for digital typesetting.

@tajpulo@typo.social
github.com/tajpulo

Talk content

1) What is digital typesetting?
   Requirements and usecases

2) Getting started with …

3) Summary and categorization

Are we digital typesetting yet?

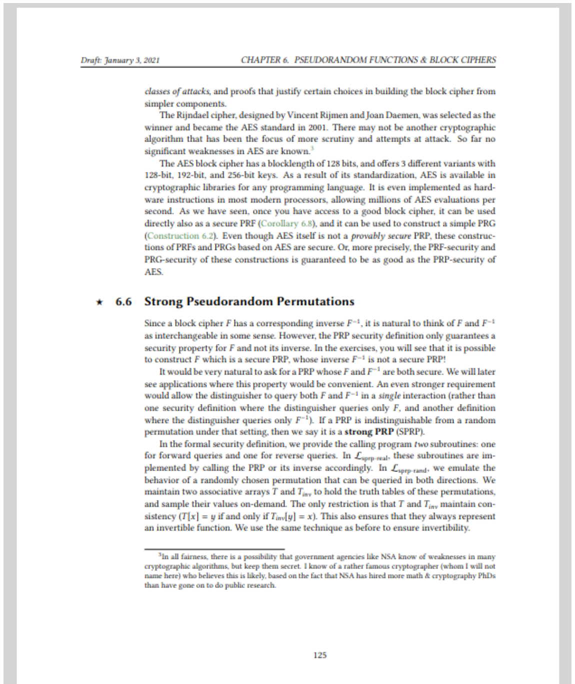
by digital means

setting type

Are we digital typesetting yet?

by digital means

setting type



classes of attacks, and proofs that justify certain choices in building the block cipher from simpler components.

The Rijndael cipher, designed by Vincent Rijmen and Joan Daemen, was selected as the winner and became the AES standard in 2001. There may not be another cryptographic algorithm that has been the focus of more scrutiny and attempts at attack. So far no significant weaknesses in AES are known.[3]

The AES block cipher has a blocklength of 128 bits, and offers 3 different variants with 128-bit, 192-bit, and 256-bit keys. As a result of its standardization, AES is available in cryptographic libraries for any programming language. It is even implemented as hardware instructions in most modern processors, allowing millions of AES evaluations per second. As we have seen, once you have access to a good block cipher, it can be used directly also as a secure PRF (Corollary 6.8), and it can be used to construct a simple PRG (Construction 6.2). Even though AES itself is not a *provably secure* PRP, these constructions of PRFs and PRGs based on AES are secure. Or, more precisely, the PRF-security and PRG-security of these constructions is guaranteed to be as good as the PRP-security of AES.

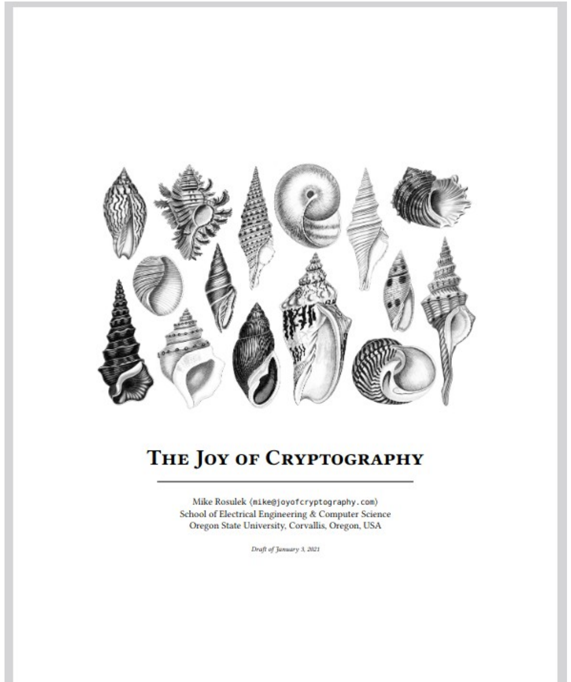
★ 6.6 Strong Pseudorandom Permutations

Since a block cipher F has a corresponding inverse F^{-1}, it is natural to think of F and F^{-1} as interchangeable in some sense. However, the PRP security definition only guarantees a security property for F and not its inverse. In the exercises, you will see that it is possible to construct F which is a secure PRP, whose inverse F^{-1} is not a secure PRP!

It would be very natural to ask for a PRP whose F and F^{-1} are both secure. We will later see applications where this property would be convenient. An even stronger requirement would allow the distinguisher to query both F and F^{-1} in a *single* interaction (rather than one security definition where the distinguisher queries only F, and another definition where the distinguisher queries only F^{-1}). If a PRP is indistinguishable from a random permutation under that setting, then we say it is a **strong PRP** (SPRP).

In the formal security definition, we provide the calling program *two* subroutines: one for forward queries and one for reverse queries. In $\mathcal{L}_{\text{sprp-real}}$, these subroutines are implemented by calling the PRP or its inverse accordingly. In $\mathcal{L}_{\text{sprp-rand}}$, we emulate the behavior of a randomly chosen permutation that can be queried in both directions. We maintain two associative arrays T and T_{inv} to hold the truth tables of these permutations, and sample their values on-demand. The only restriction is that T and T_{inv} maintain consistency ($T[x] = y$ if and only if $T_{\text{inv}}[y] = x$). This also ensures that they always represent an invertible function. We use the same technique as before to ensure invertibility.

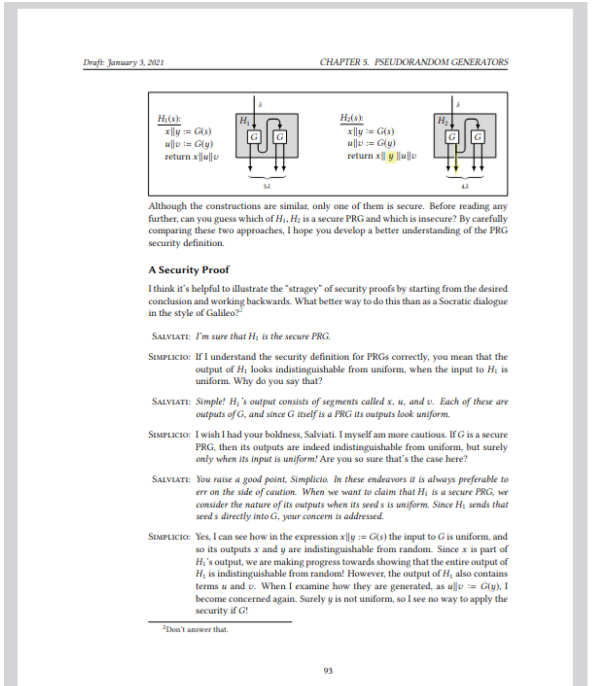
[3]In all fairness, there is a possibility that government agencies like NSA know of weaknesses in many cryptographic algorithms, but keep them secret. I know of a rather famous cryptographer (whom I will not name here) who believes this is likely, based on the fact that NSA has hired more math & cryptography PhDs than have gone on to do public research.



The Joy of Cryptography

Mike Rosulek ⟨mike@joyofcryptography.com⟩
School of Electrical Engineering & Computer Science
Oregon State University, Corvallis, Oregon, USA

Draft of January 3, 2021



```
H_1(s):
  x‖y := G(s)
  u‖v := G(y)
  return x‖u‖v

H_2(s):
  x‖y := G(s)
  u‖v := G(y)
  return x‖y‖u‖v
```

Although the constructions are similar, only one of them is secure. Before reading any further, can you guess which of H_1, H_2 is a secure PRG and which is insecure? By carefully comparing these two approaches, I hope you develop a better understanding of the PRG security definition.

A Security Proof

I think it's helpful to illustrate the "stragey" of security proofs by starting from the desired conclusion and working backwards. What better way to do this than as a Socratic dialogue in the style of Galileo?[2]

SALVIATI: *I'm sure that H_1 is the secure PRG.*

SIMPLICIO: If I understand the security definition for PRGs correctly, you mean that the output of H_1 looks indistinguishable from uniform, when the input to H_1 is uniform. Why do you say that?

SALVIATI: *Simple! H_1's output consists of segments called x, u, and v. Each of these are outputs of G, and since G itself is a PRG its outputs look uniform.*

SIMPLICIO: I wish I had your boldness, Salviati. I myself am more cautious. If G is a secure PRG, then its outputs are indeed indistinguishable from uniform, but surely *only when its input is uniform!* Are you so sure that's the case here?

SALVIATI: *You raise a good point, Simplicio. In these endeavors it is always preferable to err on the side of caution. When we want to claim that H_1 is a secure PRG, we consider the nature of its outputs when its seed s is uniform. Since H_1 sends that seed s directly into G, your concern is addressed.*

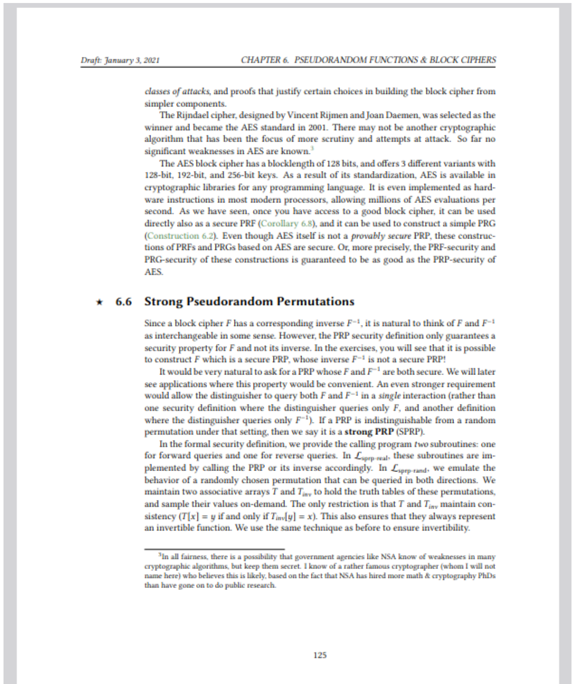
SIMPLICIO: Yes, I can see how in the expression $x \| y := G(s)$ the input to G is uniform, and so its outputs x and y are indistinguishable from random. Since x is part of H_1's output, we are making progress towards showing that the entire output of H_1 is indistinguishable from random! However, the output of H_1 also contains terms u and v. When I examine how they are generated, as $u \| v := G(y)$, I become concerned again. Surely y is not uniform, so I see no way to apply the security if G!

[2]Don't answer that.

Are we digital typesetting yet?

by digital means

setting type



Draft: January 3, 2021 CHAPTER 6. PSEUDORANDOM FUNCTIONS & BLOCK CIPHERS

classes of attacks, and proofs that justify certain choices in building the block cipher from simpler components.

The Rijndael cipher, designed by Vincent Rijmen and Joan Daemen, was selected as the winner and became the AES standard in 2001. There may not be another cryptographic algorithm that has been the focus of more scrutiny and attempts at attack. So far no significant weaknesses in AES are known.[3]

The AES block cipher has a blocklength of 128 bits, and offers 3 different variants with 128-bit, 192-bit, and 256-bit keys. As a result of its standardization, AES is available in cryptographic libraries for any programming language. It is even implemented as hardware instructions in most modern processors, allowing millions of AES evaluations per second. As we have seen, once you have access to a good block cipher, it can be used directly also as a secure PRF (Corollary 6.8), and it can be used to construct a simple PRG (Construction 6.2). Even though AES itself is not a *provably secure* PRP, these constructions of PRFs and PRGs based on AES are secure. Or, more precisely, the PRF-security and PRG-security of these constructions is guaranteed to be as good as the PRP-security of AES.

★ 6.6 Strong Pseudorandom Permutations

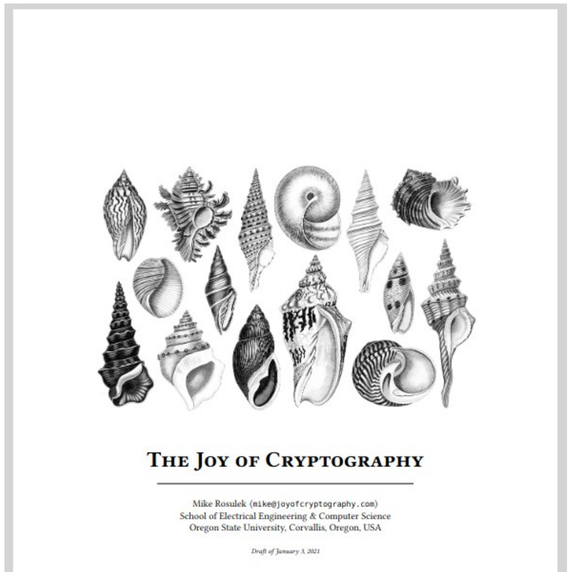
Since a block cipher F has a corresponding inverse F^{-1}, it is natural to think of F and F^{-1} as interchangeable in some sense. However, the PRP security definition only guarantees a security property for F and not its inverse. In the exercises, you will see that it is possible to construct F which is a secure PRP, whose inverse F^{-1} is not a secure PRP!

It would be very natural to ask for a PRP whose F and F^{-1} are both secure. We will later see applications where this property would be convenient. An even stronger requirement would allow the distinguisher to query both F and F^{-1} in a *single* interaction (rather than one security definition where the distinguisher queries only F, and another definition where the distinguisher queries only F^{-1}). If a PRP is indistinguishable from a random permutation under that setting, then we say it is a **strong PRP** (SPRP).

In the formal security definition, we provide the calling program *two* subroutines: one for forward queries and one for reverse queries. In $\mathcal{L}_{\text{sprp-real}}$, these subroutines are implemented by calling the PRP or its inverse accordingly. In $\mathcal{L}_{\text{sprp-rand}}$, we emulate the behavior of a randomly chosen permutation that can be queried in both directions. We maintain two associative arrays T and T_{inv} to hold the truth tables of these permutations, and sample their values on-demand. The only restriction is that T and T_{inv} maintain consistency ($T[x] = y$ if and only if $T_{\text{inv}}[y] = x$). This also ensures that they always represent an invertible function. We use the same technique as before to ensure invertibility.

[3]In all fairness, there is a possibility that government agencies like NSA know of weaknesses in many cryptographic algorithms, but keep them secret. I know of a rather famous cryptographer (whom I will not name here) who believes this is likely, based on the fact that NSA has hired more math & cryptography PhDs than have gone on to do public research.

125

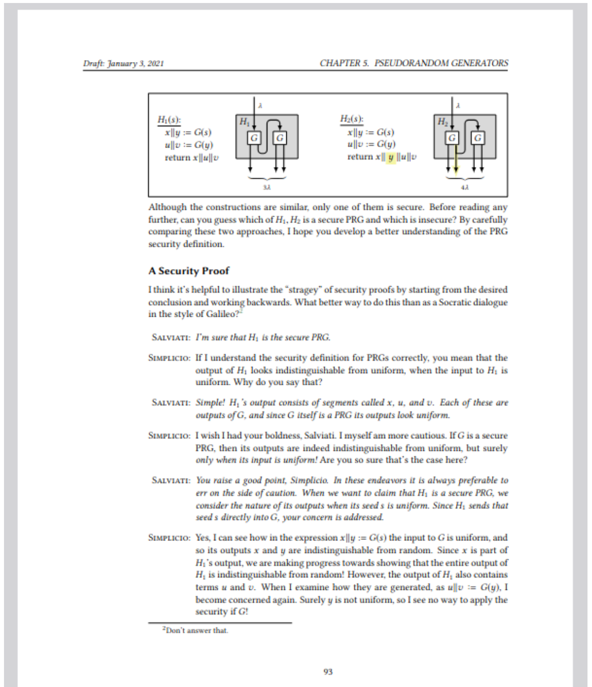


THE JOY OF CRYPTOGRAPHY

Mike Rosulek ⟨mike@joyofcryptography.com⟩
School of Electrical Engineering & Computer Science
Oregon State University, Corvallis, Oregon, USA

Draft of January 3, 2021

"The Joy of Cryptography"
by Mike Rosulek



Draft: January 3, 2021 CHAPTER 5. PSEUDORANDOM GENERATORS

```
H_1(s):
    x‖y := G(s)
    u‖v := G(y)
    return x‖u‖v

H_2(s):
    x‖y := G(s)
    u‖v := G(y)
    return x‖y‖u‖v
```

Although the constructions are similar, only one of them is secure. Before reading any further, can you guess which of H_1, H_2 is a secure PRG and which is insecure? By carefully comparing these two approaches, I hope you develop a better understanding of the PRG security definition.

A Security Proof

I think it's helpful to illustrate the "stragey" of security proofs by starting from the desired conclusion and working backwards. What better way to do this than as a Socratic dialogue in the style of Galileo?[2]

SALVIATI: *I'm sure that H_1 is the secure PRG.*

SIMPLICIO: If I understand the security definition for PRGs correctly, you mean that the output of H_1 looks indistinguishable from uniform, when the input to H_1 is uniform. Why do you say that?

SALVIATI: *Simple! H_1's output consists of segments called x, u, and v. Each of these are outputs of G, and since G itself is a PRG its outputs look uniform.*

SIMPLICIO: I wish I had your boldness, Salviati. I myself am more cautious. If G is a secure PRG, then its outputs are indeed indistinguishable from uniform, but surely *only when its input is uniform!* Are you so sure that's the case here?

SALVIATI: *You raise a good point, Simplicio. In these endeavors it is always preferable to err on the side of caution. When we want to claim that H_1 is a secure PRG, we consider the nature of its outputs when its seed s is uniform. Since H_1 sends that seed s directly into G, your concern is addressed.*

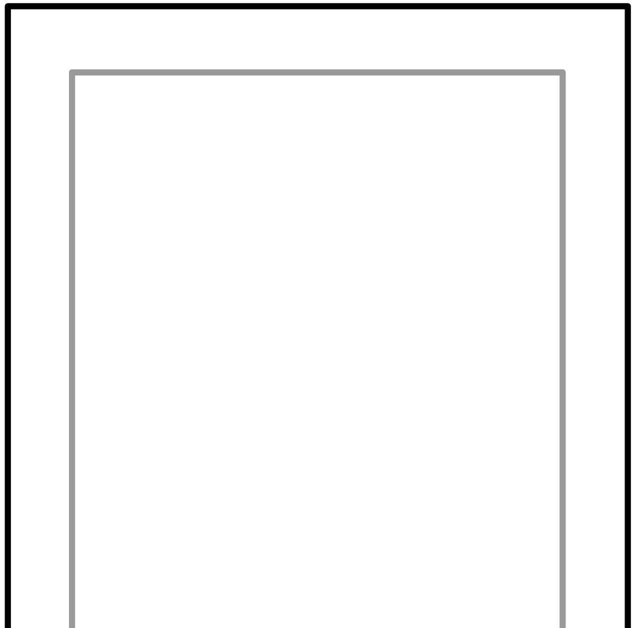
SIMPLICIO: Yes, I can see how in the expression $x\|y := G(s)$ the input to G is uniform, and so its outputs x and y are indistinguishable from random. Since x is part of H_1's output, we are making progress towards showing that the entire output of H_1 is indistinguishable from random! However, the output of H_1 also contains terms u and v. When I examine how they are generated, as $u\|v := G(y)$, I become concerned again. Surely y is not uniform, so I see no way to apply the security if G!

[2]Don't answer that.

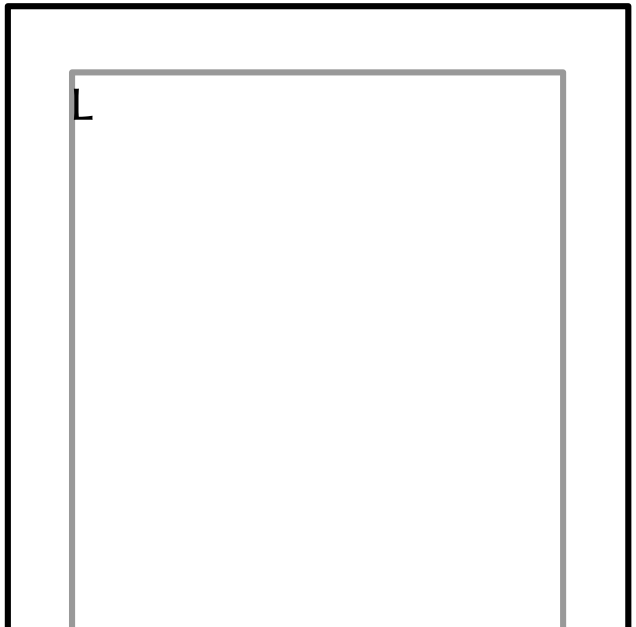
93

The Art of Digital Typesetting

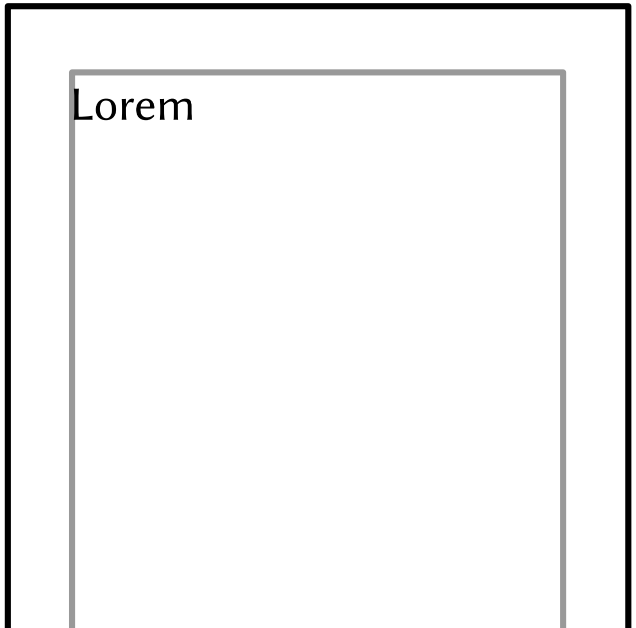
The Art of Digital Typesetting



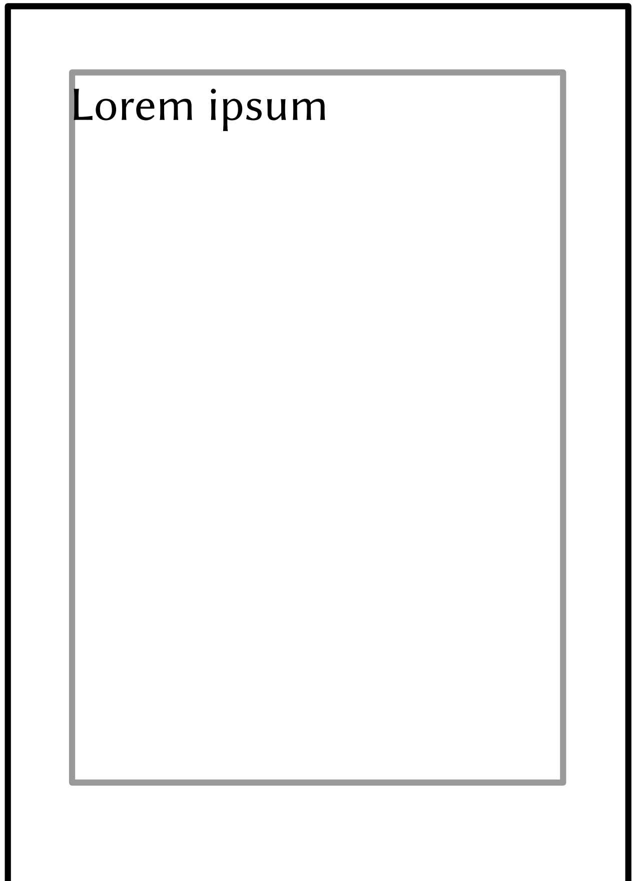
The Art of Digital Typesetting



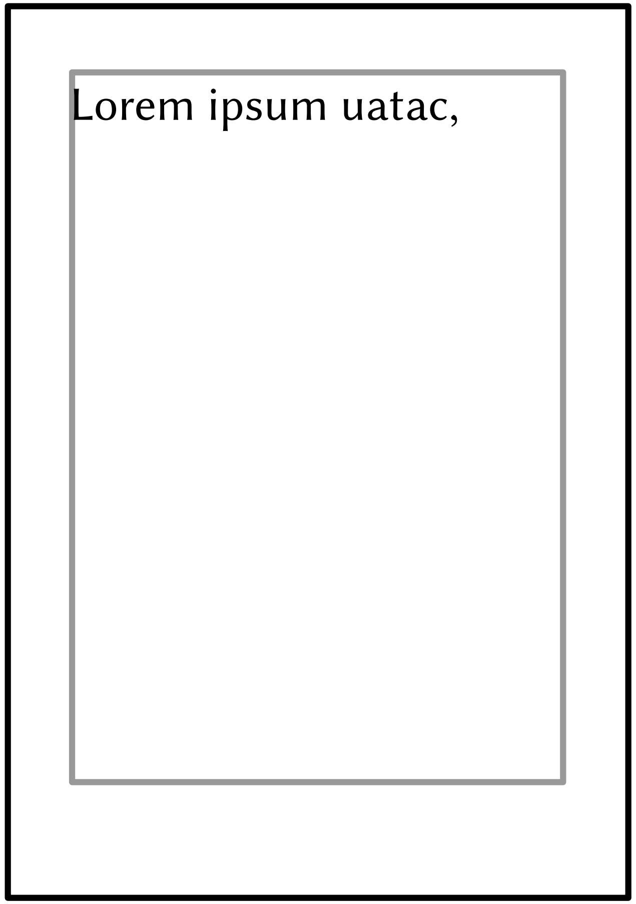
The Art of Digital Typesetting



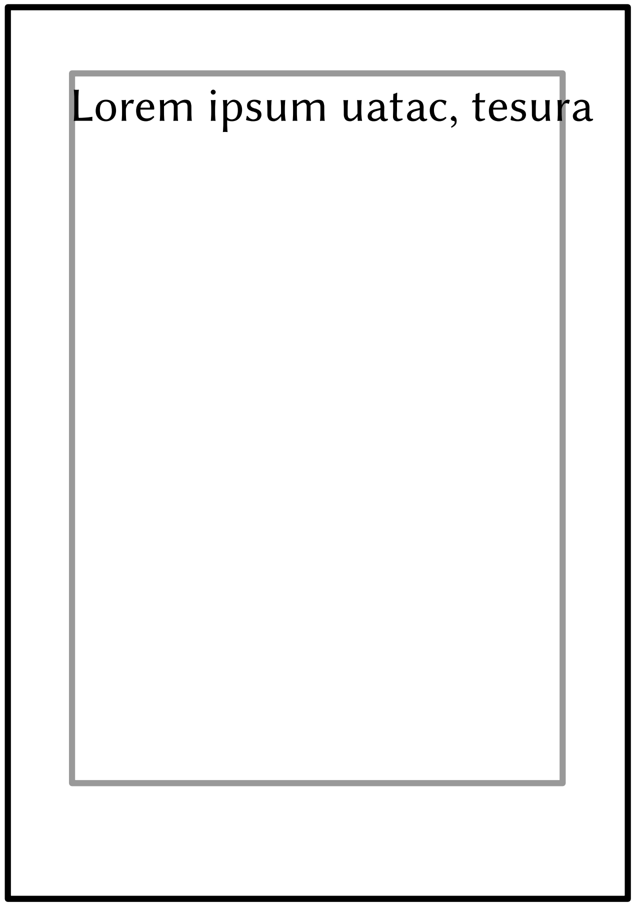
The Art of Digital Typesetting



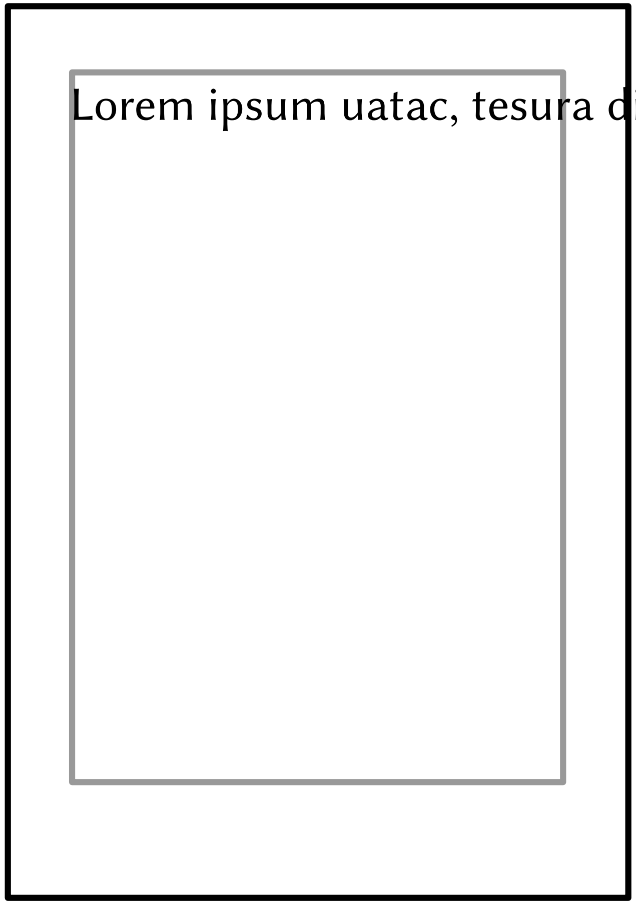
The Art of Digital Typesetting



The Art of Digital Typesetting

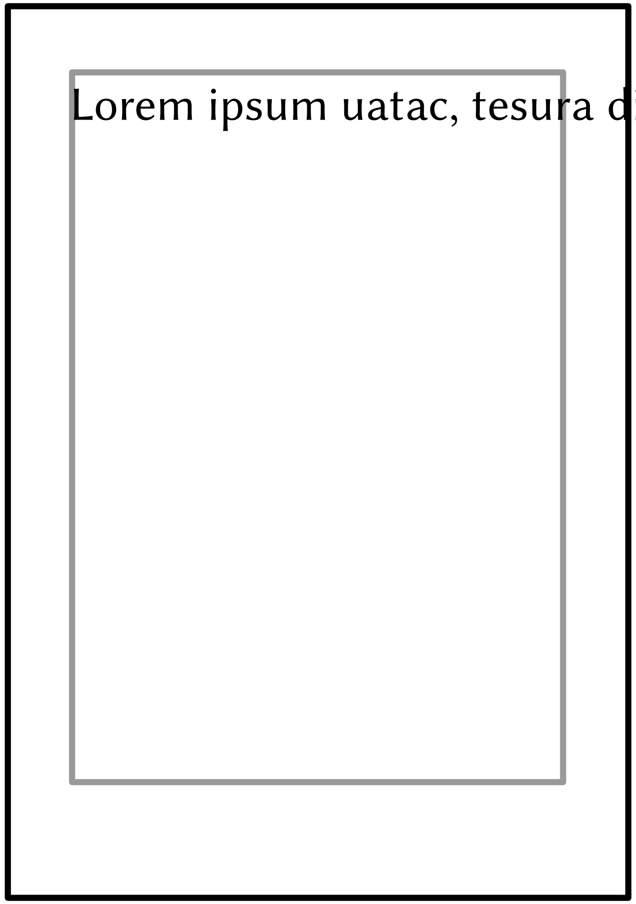


The Art of Digital Typesetting



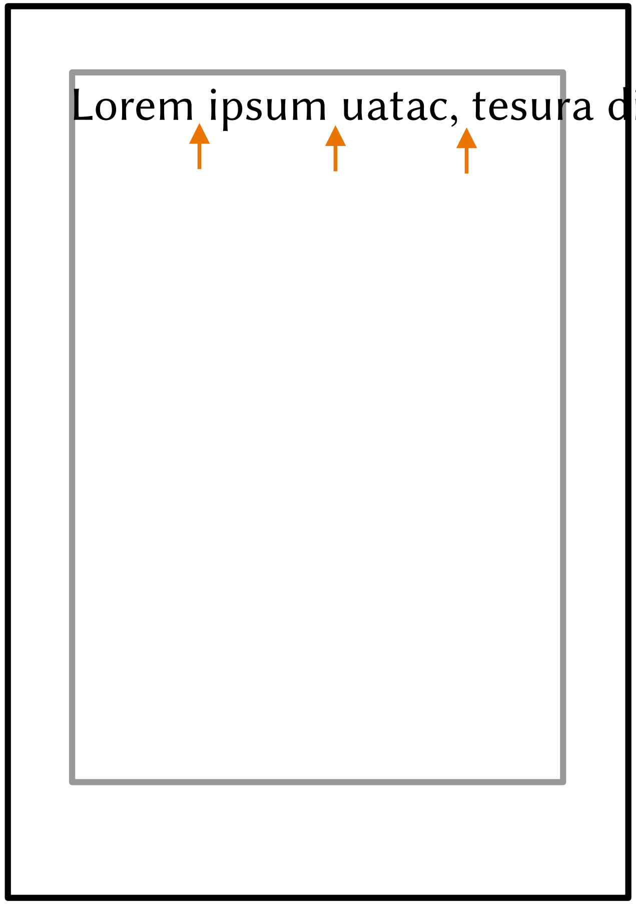
Lorem ipsum uatac, tesura dius atitac, ediafam isag umoli. Tare tanu bora mpere sitvi ncias ae

The Art of Digital Typesetting



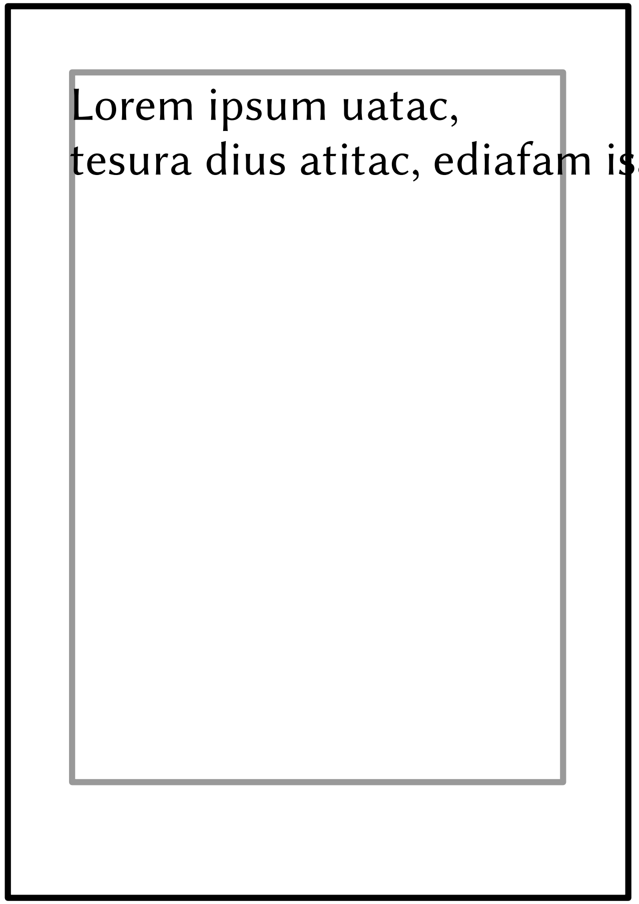
Lorem ipsum uatac, tesura dius atitac, ediafam isag umoli. Tare tanu bora mpere sitvi ncias ae

The Art of Digital Typesetting


Lorem ipsum uatac, tesura dius atitac, ediafam isag umoli. Tare tanu bora mpere sitvi ncias ae

UAX #14: "Unicode Line Breaking Algorithm"
https://www.unicode.org/reports/tr14/

The Art of Digital Typesetting

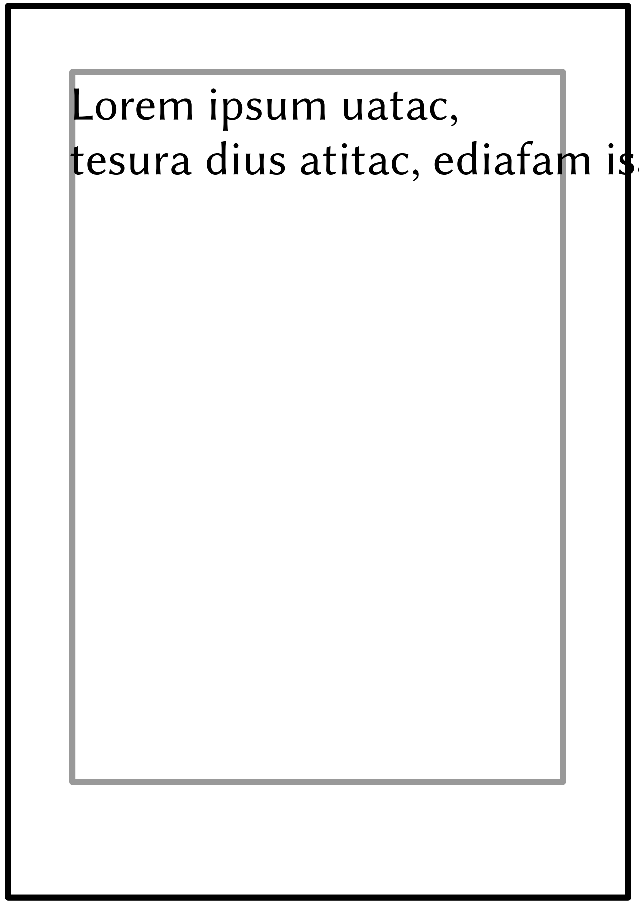


Lorem ipsum uatac,
tesura dius atitac, ediafam isag umoli. Tare tanu bora mpere sitvi ncias aetc.

UAX #14: "Unicode Line Breaking Algorithm"
https://www.unicode.org/reports/tr14/

naïve algorithm
or
"Breaking Paragraphs into Lines"
by Don Knuth and Michael Plass (1981)

The Art of Digital Typesetting



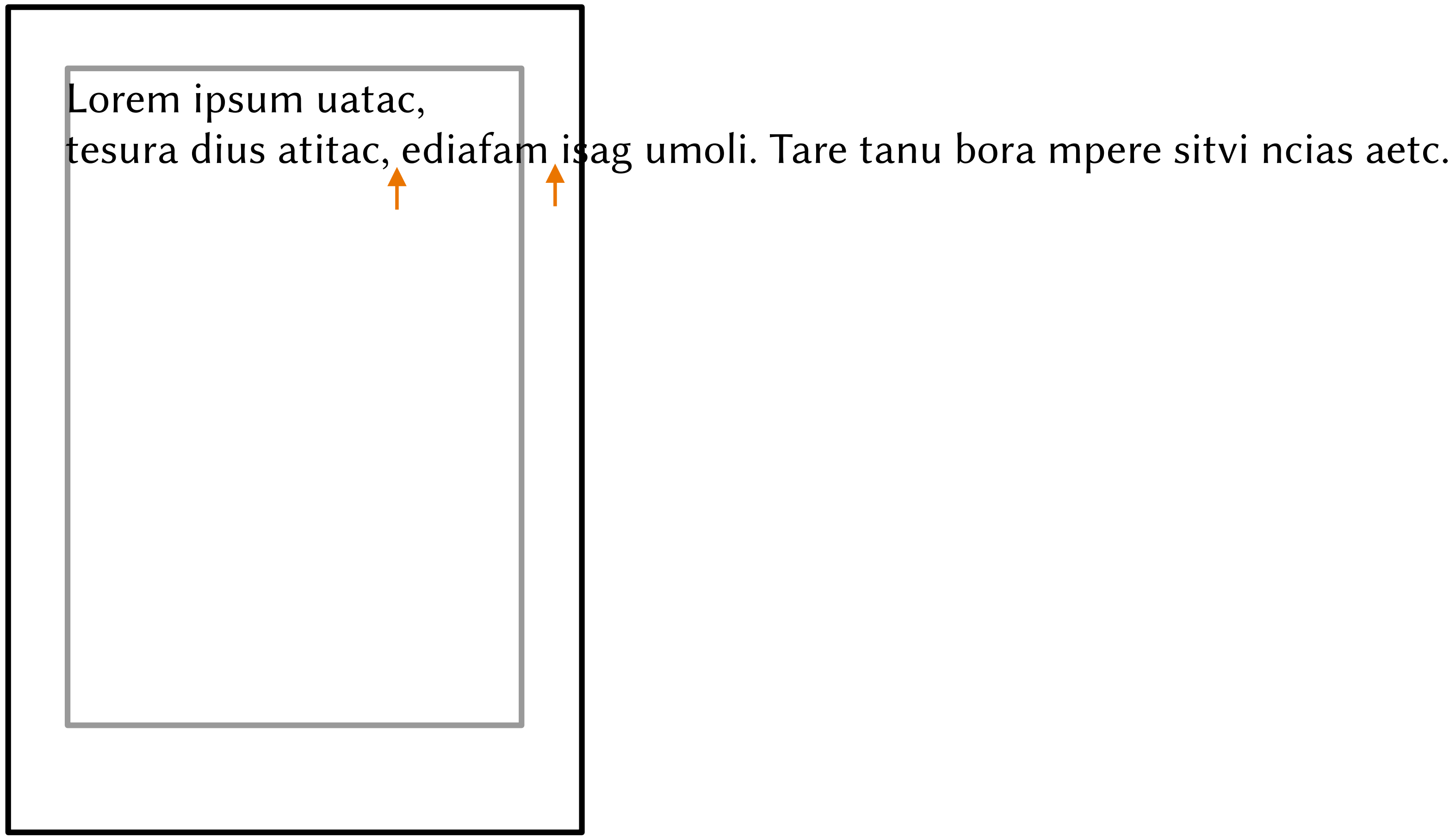
Lorem ipsum uatac,
tesura dius atitac, ediafam isag umoli. Tare tanu bora mpere sitvi ncias aetc.

UAX #14: "Unicode Line Breaking Algorithm"
https://www.unicode.org/reports/tr14/

naïve algorithm
or
"Breaking Paragraphs into Lines"
by Don Knuth and Michael Plass (1981)

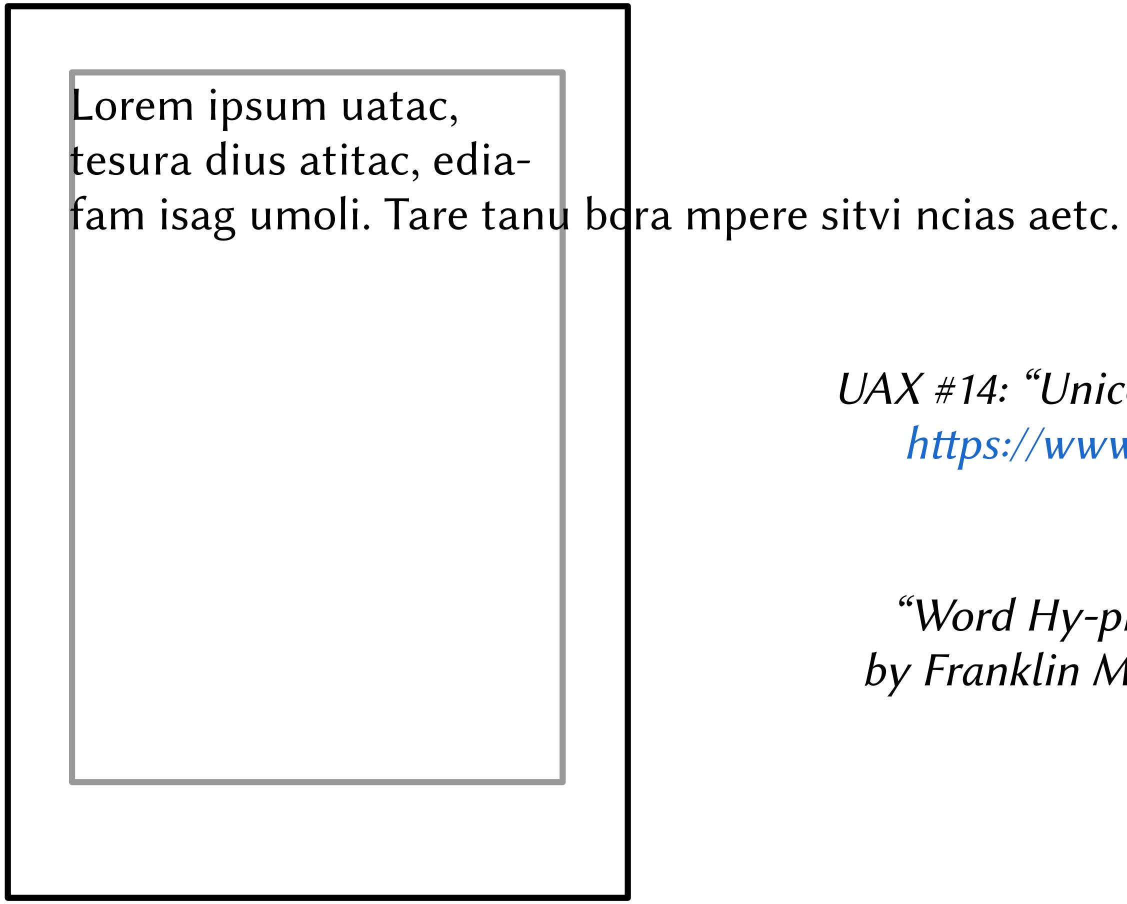
In CSS:
text-wrap: pretty [webkit.org, demo]

The Art of Digital Typesetting



Lorem ipsum uatac,
tesura dius atitac, ediafam isag umoli. Tare tanu bora mpere sitvi ncias aetc.

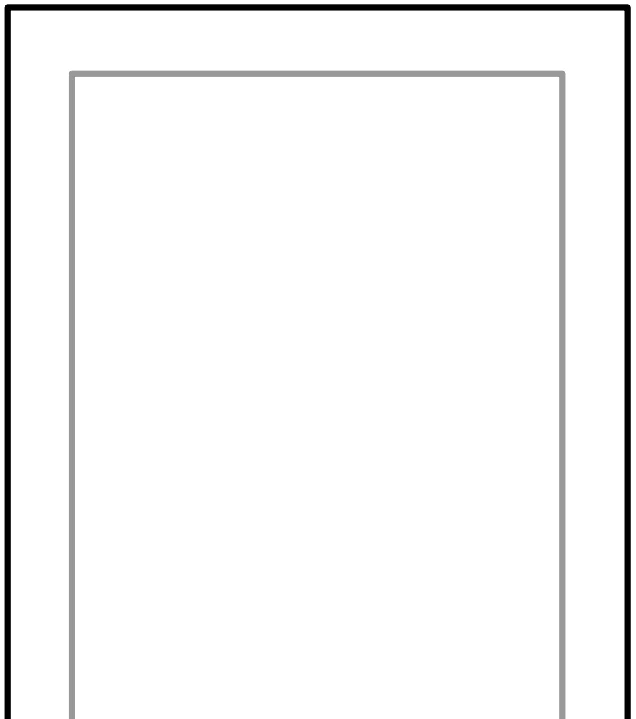
The Art of Digital Typesetting


Lorem ipsum uatac,
tesura dius atitac, edia-
fam isag umoli. Tare tanu bora mpere sitvi ncias aetc.

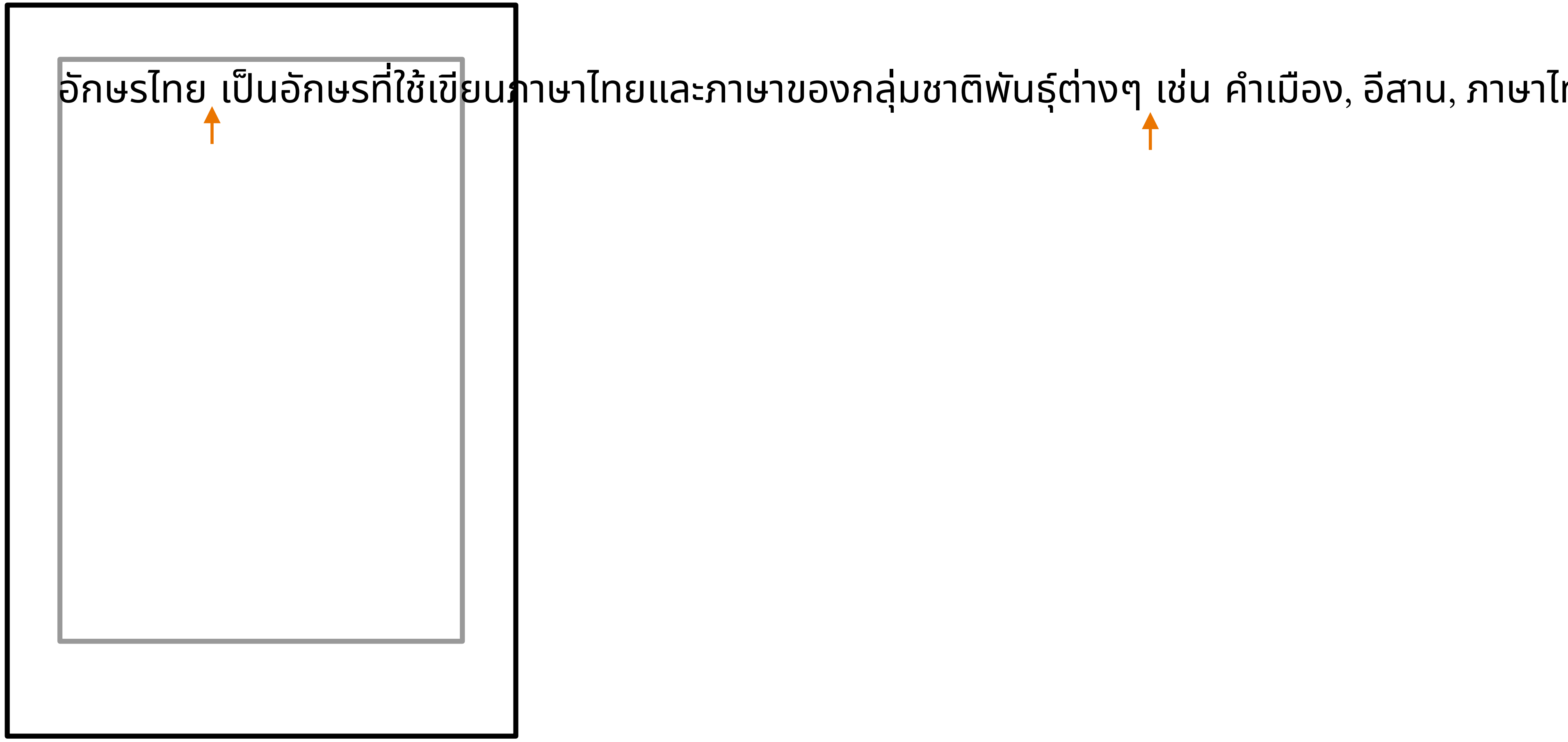
UAX #14: "Unicode Line Breaking Algorithm"
https://www.unicode.org/reports/tr14/

"Word Hy-phen-a-tion by Com-put-er"
by Franklin Mark Liang (1983), PhD thesis

The Art of Digital Typesetting

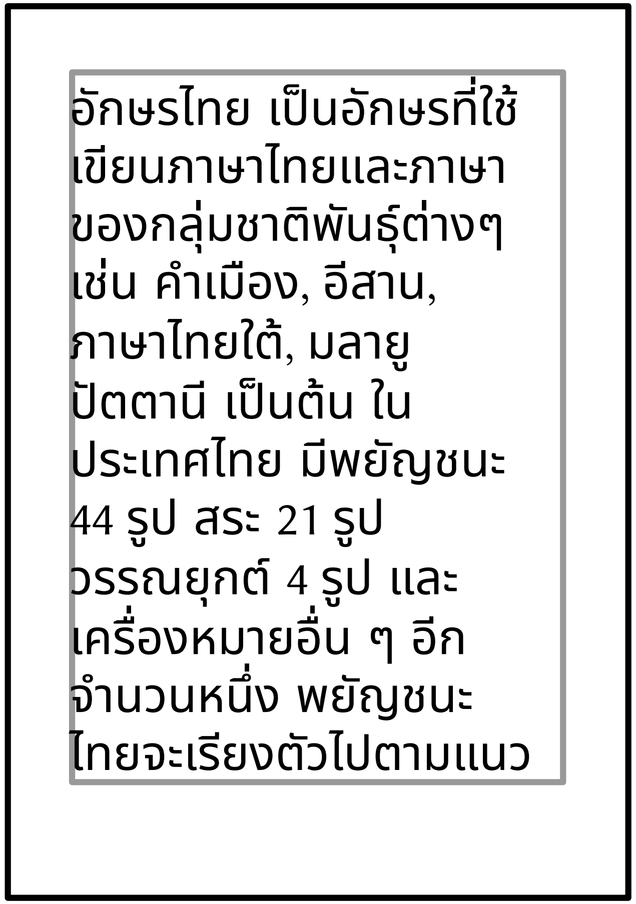


The Art of Digital Typesetting



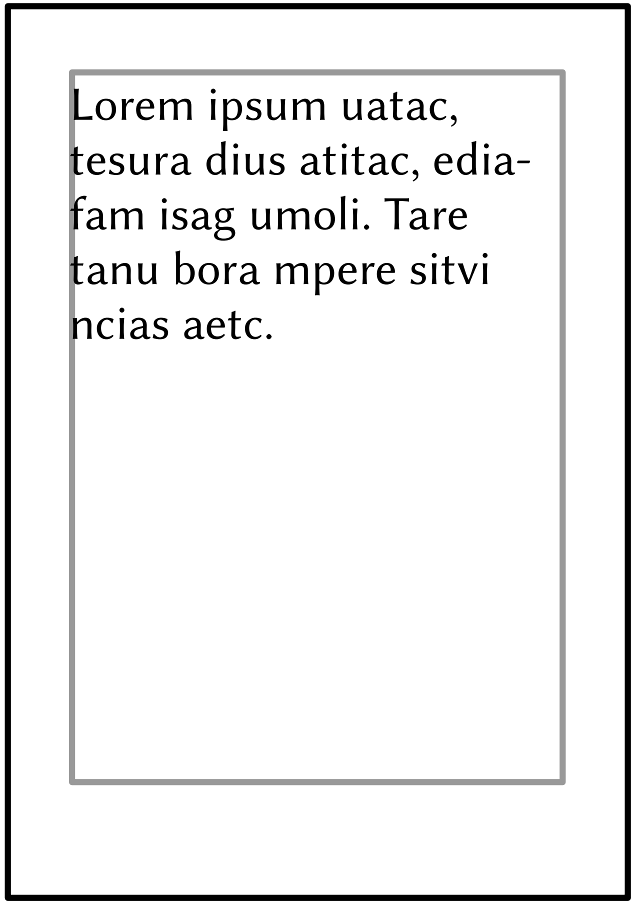
อักษรไทย เป็นอักษรที่ใช้เขียนภาษาไทยและภาษาของกลุ่มชาติพันธุ์ต่างๆ เช่น คำเมือง, อีสาน, ภาษาไ

The Art of Digital Typesetting



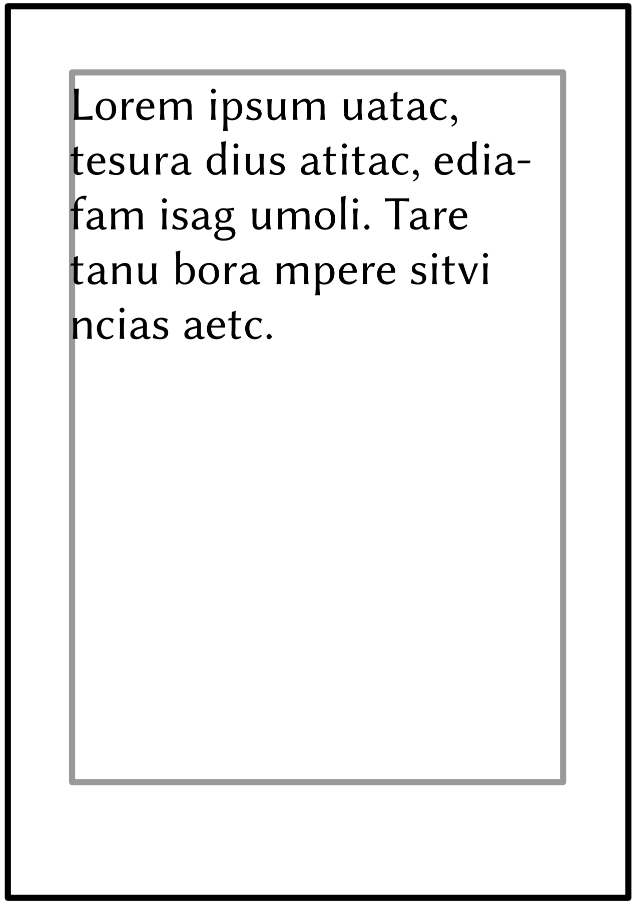
อักษรไทย เป็นอักษรที่ใช้
เขียนภาษาไทยและภาษา
ของกลุ่มชาติพันธุ์ต่างๆ
เช่น คำเมือง, อีสาน,
ภาษาไทยใต้, มลายู
ปัตตานี เป็นต้น ใน
ประเทศไทย มีพยัญชนะ
44 รูป สระ 21 รูป
วรรณยุกต์ 4 รูป และ
เครื่องหมายอื่น ๆ อีก
จำนวนหนึ่ง พยัญชนะ
ไทยจะเรียงตัวไปตามแนว

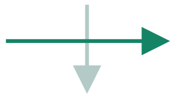
The Art of Digital Typesetting



Lorem ipsum uatac, tesura dius atitac, ediafam isag umoli. Tare tanu bora mpere sitvi ncias aetc.

The Art of Digital Typesetting

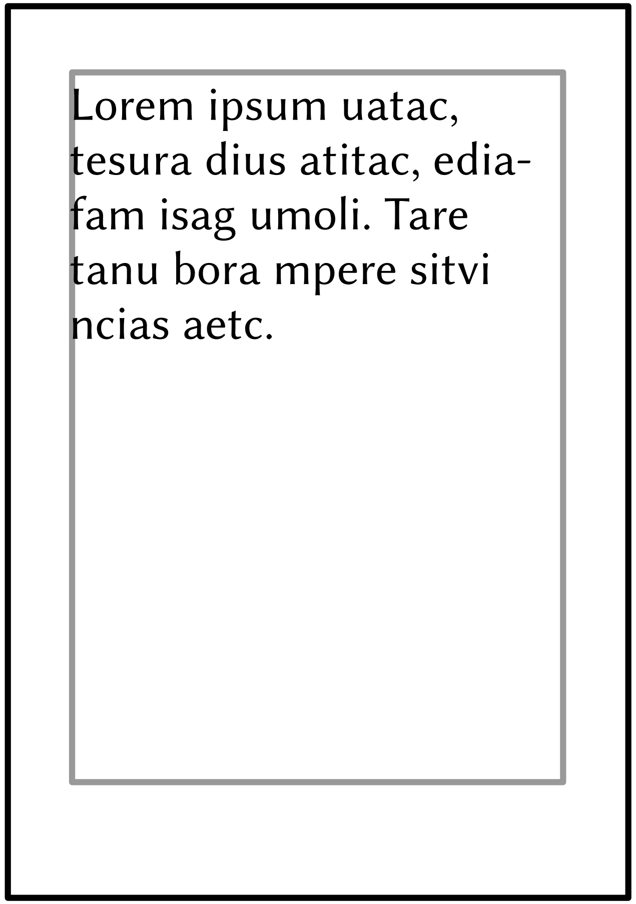

Lorem ipsum uatac, tesura dius atitac, edia-fam isag umoli. Tare tanu bora mpere sitvi ncias aetc.

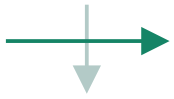


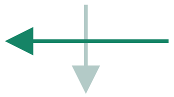
Latin script, Cyrillic, Burmese, Thai, …

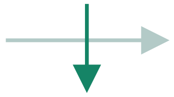
Source: omniglot.com/writing/direction.htm

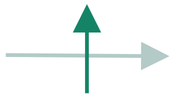
The Art of Digital Typesetting

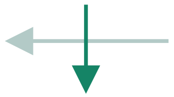

Lorem ipsum uatac, tesura dius atitac, edia-fam isag umoli. Tare tanu bora mpere sitvi ncias aetc.

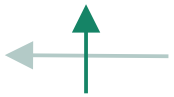

Latin script, Cyrillic, Burmese, Thai, …


Hebrew, Arabic, Etruscan, Aramaic, …


Old Elamite, Mongolian, Uyghur, …

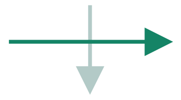

Batak, Hanuno'o, Tagbanwa

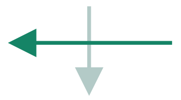

Chinese, Japanese, Korean, Nushu, …

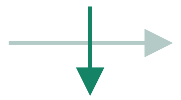

Ancient Berber

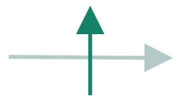
Source: omniglot.com/writing/direction.htm

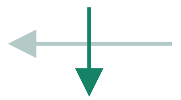
The Art of Digital Typesetting

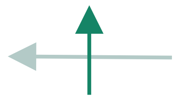

Latin script, Cyrillic, Burmese, Thai, …


Hebrew, Arabic, Etruscan, Aramaic, …

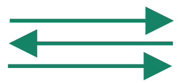

Old Elamite, Mongolian, Uyghur, …


Batak, Hanuno'o, Tagbanwa


Chinese, Japanese, Korean, Nushu, …

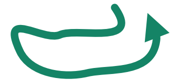

Ancient Berber

boustrophedon



Székely-Hungarian Rovás, Linear B,
Rongo Rongo, Sabaean

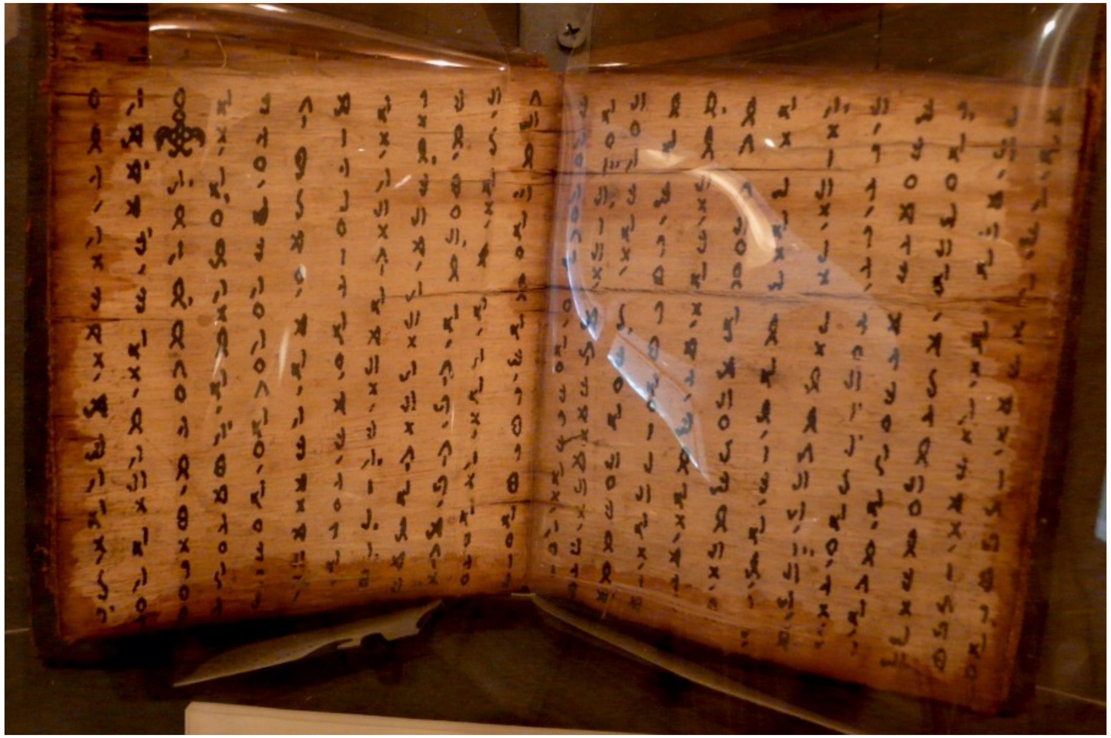
Hieroglyphic Egyptian



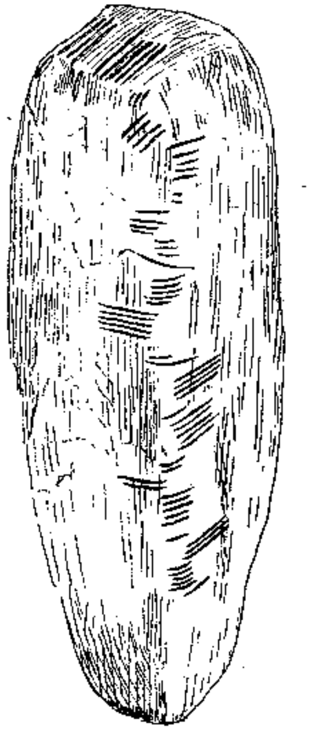
Hieroglyphic Egyptian, Ogham, …

Source: omniglot.com/writing/direction.htm

The Art of Digital Typesetting

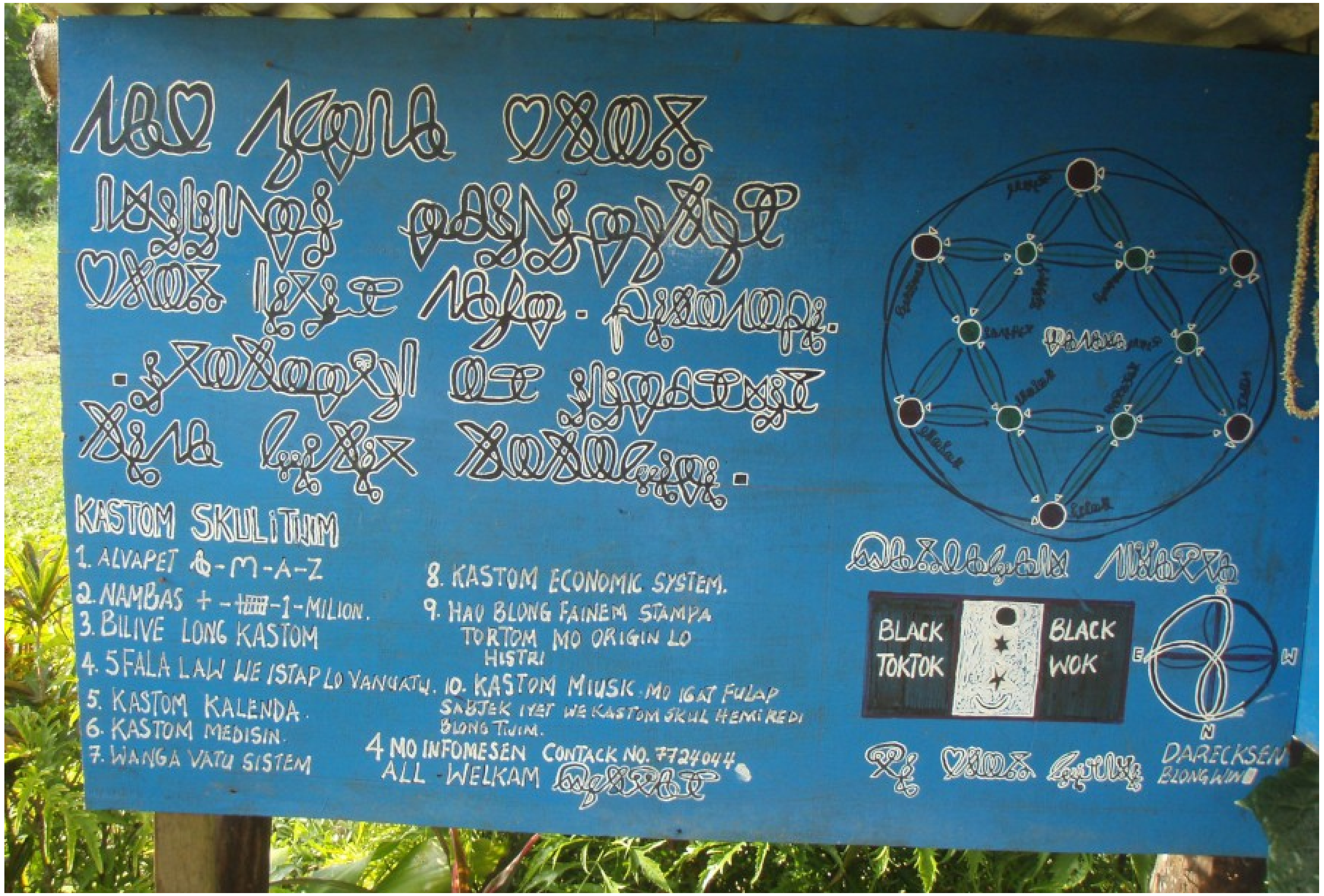


Batak script, source: Wiki Commons / Piotrus / CC-BY



Ogham text, source: EN Wikipedia

The Art of Digital Typesetting


KASTOM SKULI TIJIM
1. ALVAPET -M-A-Z
2. NAMBAS + - -1-MILION.
3. BILIVE LONG KASTOM
4. 5FALA LAW WE ISTAP LO VANUATU.
5. KASTOM KALENDA.
6. KASTOM MEDISIN.
7. WANGA VATU SISTEM
8. KASTOM ECONOMIC SYSTEM.
9. HAU BLONG FAINEM STAMPA TORTOM MO ORIGIN LO HISTRI
10. KASTOM MIUSK. MO IGAT FULAP SABJEK IYET WE KASTOM SKUL HEMI REDI BLONG TIJIM.
4 MO INFOMESEN CONTACK NO. 7724044.
ALL WELKAM
BLACK TOKTOK
BLACK WOK
N
E
W
DARECKSEN BLONG WIN

Avoiuli script example
source: EN Wikipedia /
Tabisini / CC BY-SA

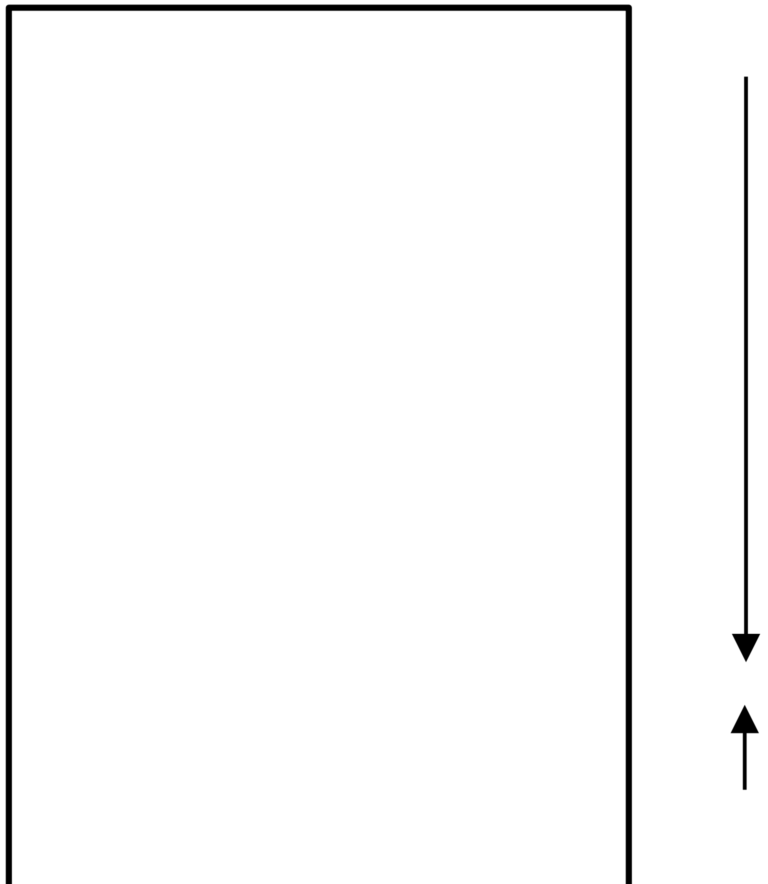
The Art of Digital Typesetting

KASTOM SKULİTUIM
1. ALVAPET -A-Z
2. NAMBAS + - -1-MILION.
3. BILIVE LONG KASTOM
4. 5FALA LAW WE ISTAP LO VANUATU.
5. KASTOM KALENDA.
6. KASTOM MEDISIN.
7. WANGA VATU SISTEM
8. KASTOM ECONOMIC SYSTEM.
9. HAU BLONG FAINEM STAMPA TORTOM MO ORIGIN LO HISTRI
10. KASTOM MIUSIK MO IGAT FULAP SABJEK IYET WE KASTOM SKUL HEMI REDI BLONG TIJIM.
4 MO INFOMESEN CONTACK NO. 7724044
ALL WELKAM
BLACK TOKTOK
BLACK WOK
DARECKSEN BLONG WIN
E
W
N

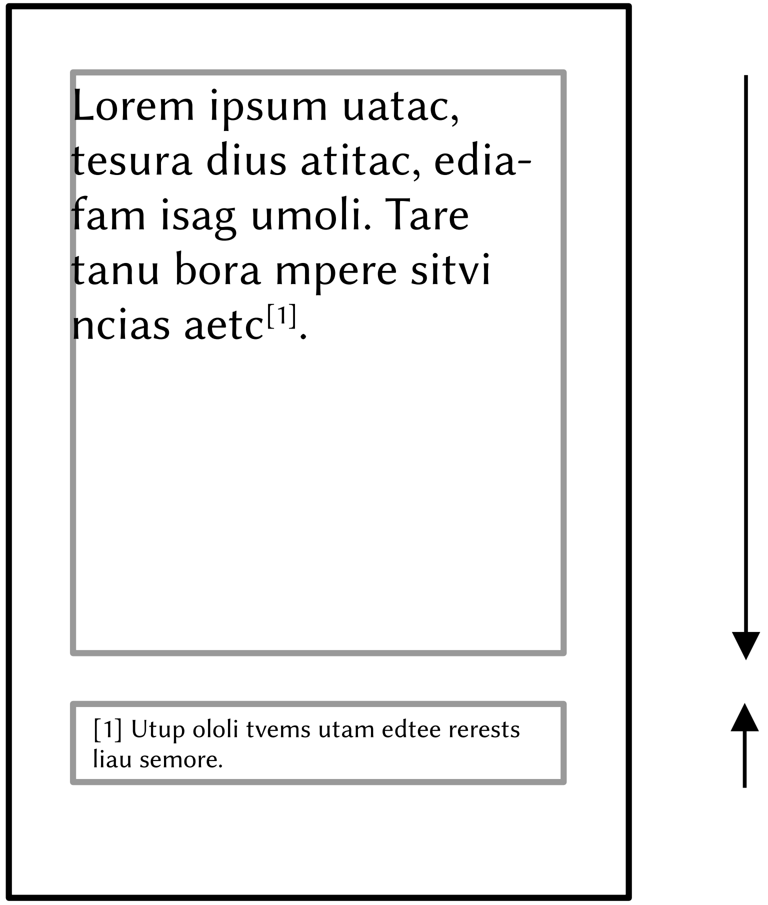
In this talk I am
going to talk
about The Art of
Digital
Typesetting
including the
boustrophedon
writing direction.

Avoiuli script example source: EN Wikipedia / Tabisini / CC BY-SA

The Art of Digital Typesetting

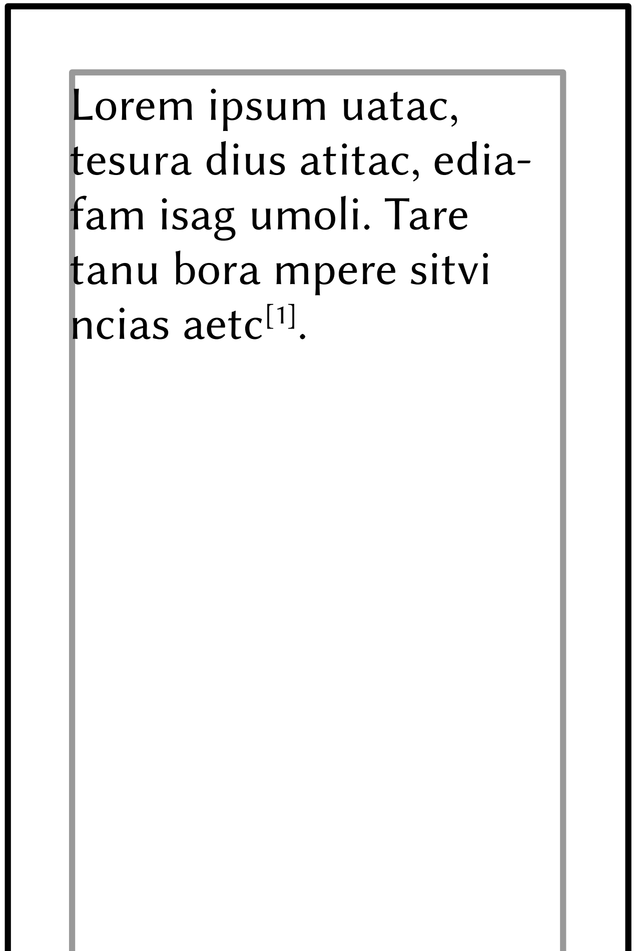


The Art of Digital Typesetting



Lorem ipsum uatac, tesura dius atitac, ediafam isag umoli. Tare tanu bora mpere sitvi ncias aetc[1].
[1] Utup ololi tvems utam edtee rerests liau semore.

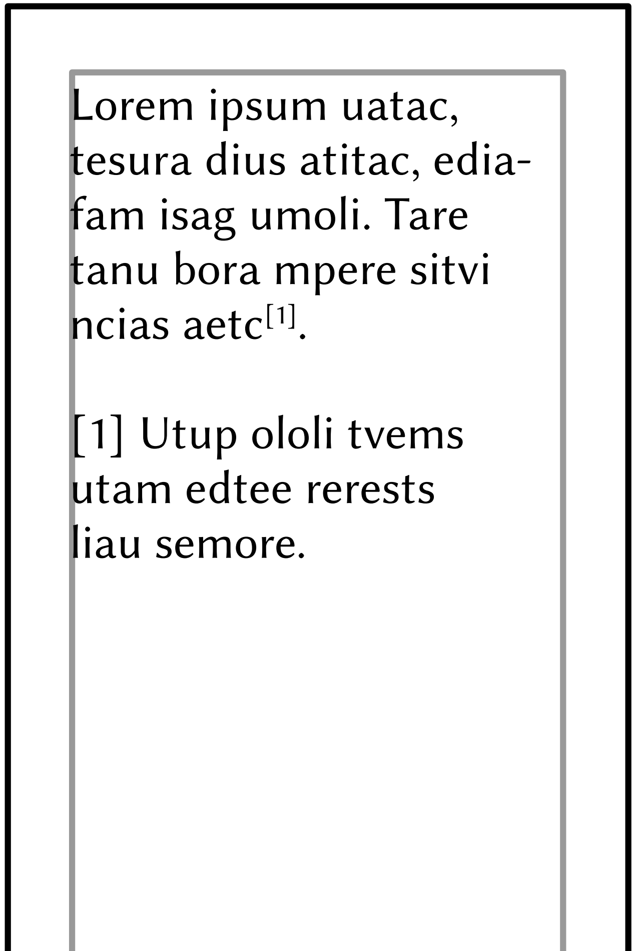
The Art of Digital Typesetting



Lorem ipsum uatac, tesura dius atitac, edia-fam isag umoli. Tare tanu bora mpere sitvi ncias aetc[1].

paged layouts
versus
reflowable layouts

The Art of Digital Typesetting

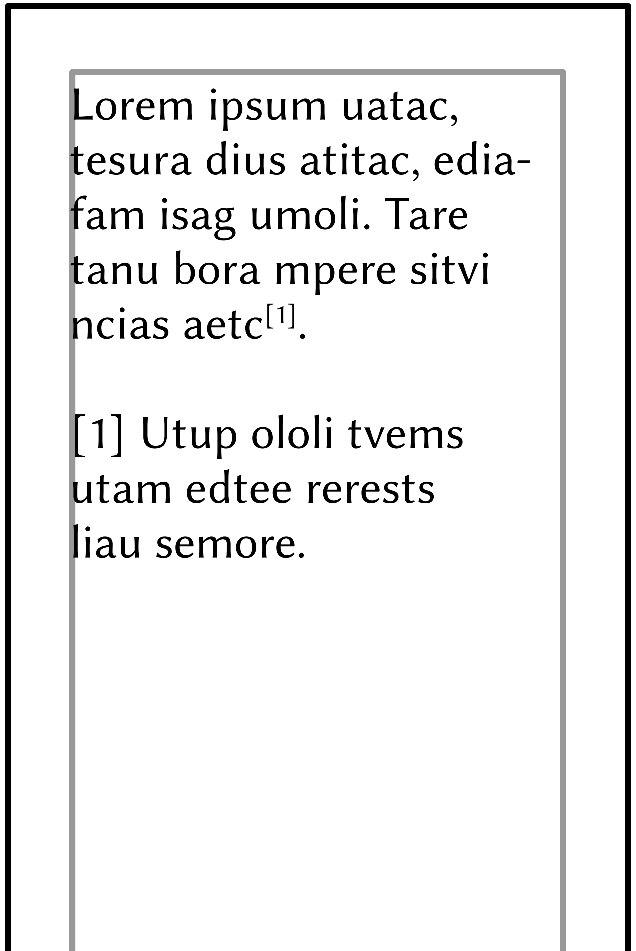


Lorem ipsum uatac, tesura dius atitac, ediafam isag umoli. Tare tanu bora mpere sitvi ncias aetc[1].

[1] Utup ololi tvems utam edtee rerests liau semore.

paged layouts
versus
reflowable layouts

The Art of Digital Typesetting



Lorem ipsum uatac, tesura dius atitac, ediafam isag umoli. Tare tanu bora mpere sitvi ncias aetc[1].

[1] Utup ololi tvems utam edtee rerests liau semore.

PDF, PostScript

↑

paged layouts
versus
reflowable layouts

↓

HTML5, EPUB

HarfBuzz

```
use harfbuzz_rs::*;

let path = "/home/tajpulo/.fonts/GentiumPlus-R.otf";
let face = Face::from_file(path, index)?;
let mut font = Font::new(face);
let buffer = UnicodeBuffer::new().add_str("Hello World!");
let output = shape(&font, buffer, &[]);
let positions = output.get_glyph_positions();
let infos = output.get_glyph_infos();

for (position, info) in positions.iter().zip(infos) {

    println!("gid{:?}=c{:?} X{:?}Y{:?} →{:?}↑{:?}",
        info.codepoint, info.cluster,
        position.x_advance, position.x_offset, position.y_offset
    );

}
```

```
gid43=c0 X0Y0 →1339↑0
gid72=c1 X0Y0 →946↑0
gid79=c2 X0Y0 →555↑0
gid79=c3 X0Y0 →555↑0
gid82=c4 X0Y0 →1030↑0
gid3=c5 X0Y0 →451↑0
gid58=c6 X0Y0 →1741↑0
gid82=c7 X0Y0 →1030↑0
gid85=c8 X0Y0 →811↑0
gid79=c9 X0Y0 →555↑0
gid71=c10 X0Y0 →1065↑0
gid4=c11 X0Y0 →557↑0
```

Are we digital typesetting yet?

Are we digital typesetting yet?

Usecases:

- books for novels
- school books
- academic papers
- conference proceedings
- product catalogue
- presentation and slides
- generate exercises for student sheets
- invoices
- financial statement
- technical specifications
- mathematical proof visualization
- API documentation
- restaurant menu
- recipes list
- …

Are we digital typesetting yet?

Criteria (today):

- FOSS
- paged output
- command line interface

Nice to have:

- Global scripts
- Domain-specific notations
- Single Source Publishing
- cross-platform
- easy to install
- separation of concerns
- modern fonts
- automation
- a11n, l10n, i18n
- performance
- web and print

Are we digital typesetting yet?

Criteria (today):

- FOSS
- paged output
- command line interface

Options?

Nice to have:

- Global scripts
- Domain-specific notations
- Single Source Publishing
- cross-platform
- easy to install
- separation of concerns
- modern fonts
- automation
- a11n, l10n, i18n
- performance
- web and print

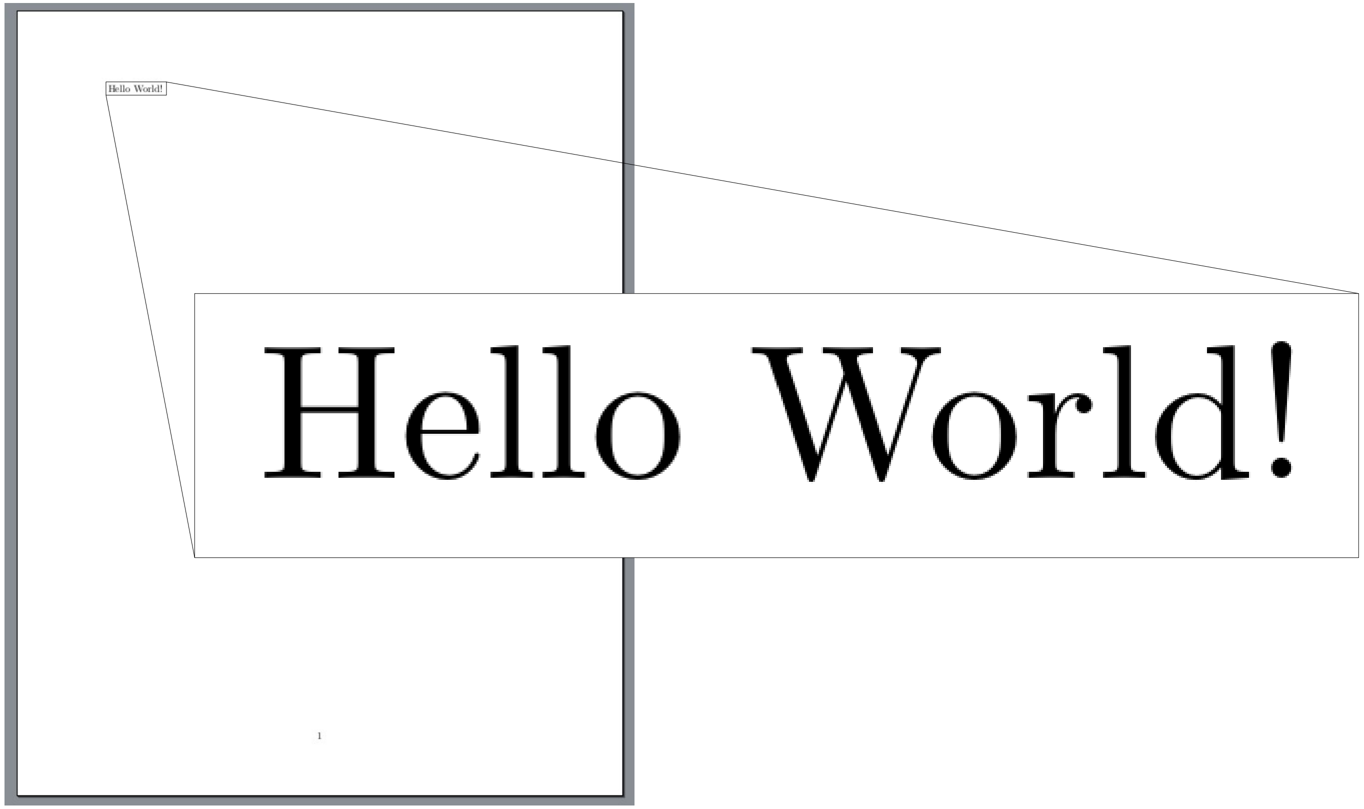
Teχ

- Since 1978, latest version 3.141592653 (Teχ78 and Teχ82)
- by Donald Knuth
- website https://tug.org/
- source code repo: https://www.tug.org/svn/texlive/
- license: rename-required (permissive free software)
- cross-platform: TeX Live (Linux), MacTeX (macOS), MiKTeX (Windows)
- written in WEB/Pascal/C

Teχ

```
$ cat example.tex
Hello World!
\bye
$ tex example.tex
This is TeX, Version 3.141592653 (TeX Live 2026/dev/Arch
Linux) (preloaded format=tex)
(./example.tex [1] )
Output written on example.dvi (1 page, 228 bytes).
Transcript written on example.log.
$
```

Teχ


Hello World!

1

Hello World!

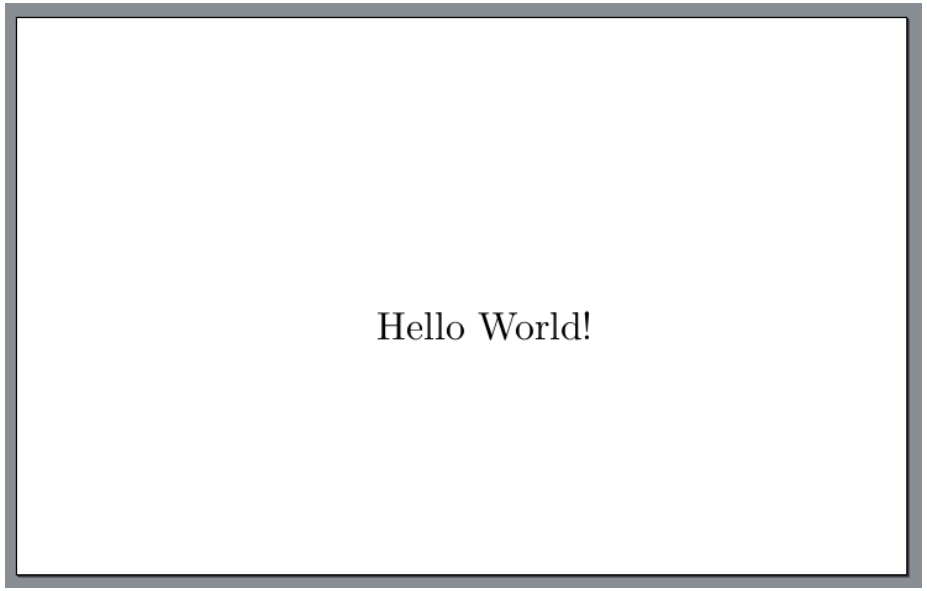
Teχ

```
$ tex
This is TeX, Version 3.141592653 (TeX Live 2026/dev/Arch
Linux) (preloaded format=tex)
**\show\bye
> \bye=\outer macro:
->\par \vfill \supereject \end .
<*> \show\bye


?
$
```

$$\begin{tikzcd}
 & \text{.tex file} \arrow[d] & \\
 & \text{primitives} \arrow[dl] & \\
\text{DVI file} & & \text{font data} \arrow[ul]
\end{tikzcd}$$

Teχ

```
$ cat example.tex
\special{papersize=8cm,1cm}
Hello World!
\bye
$
```



Hello World!

Teχ

```
$ cat example.tex
\def\event #1 organized by #2 at #3.{Hello #2!
Thank you for organizing #1 at #3.}
\event GLT25 organized by Grazer Linuxtage at TU Graz.
\bye
$ tex example.tex
This is TeX, Version 3.141592653 (TeX Live 2026/dev/Arch
Linux) (preloaded format=tex)
(./test.tex [1] )
Output written on test.dvi (1 page, 288 bytes).
Transcript written on test.log.
$
```

Teχ

```
$ cat example.tex
\def\event #1 organized by #2 at #3.{Hello #2!
Thank you for organizing #1 at #3.}

\event GLT25 organized by Grazer Linuxtage at TU Graz.

\bye
```

Hello Grazer Linuxtage! Thank you for organizing GLT25 at TU Graz.

Teχ

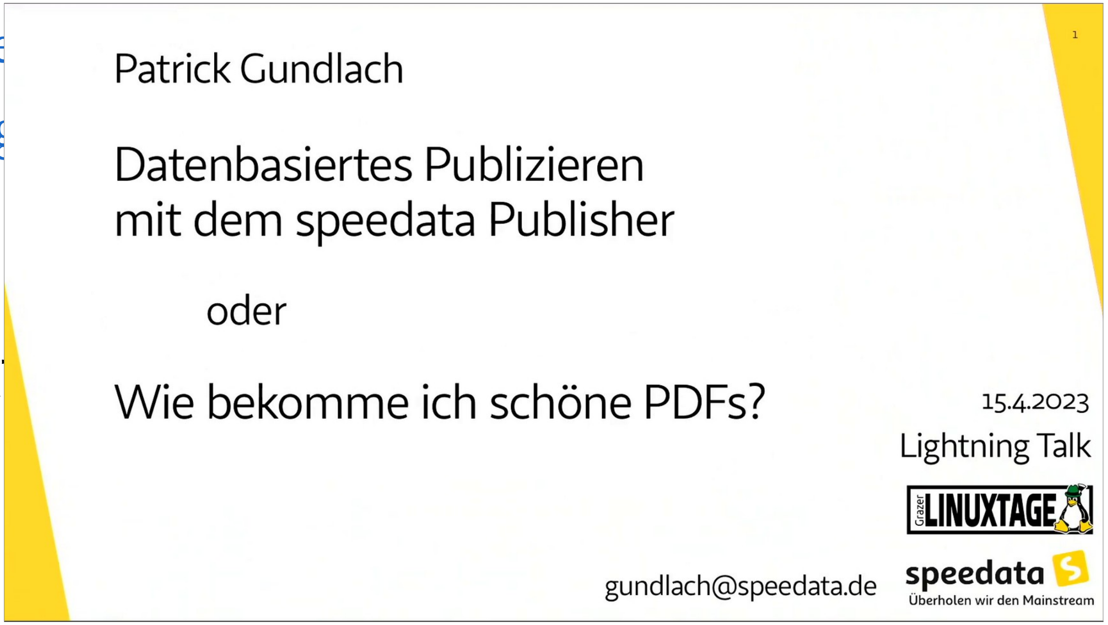
- Lead to the development of …
  - LaTeχ2ε (1994) by Leslie Lamport
  - ConTeχt (1995) by Hans Hagen, et al.
  - pdftex (1996) by Hàn Thế Thành
  - χeTeχ (2005) by Jonathan Kew
  - LuaTeχ (2007)
  - OpTeχ (2020) by Petr Olšák

speedata Publisher

- Since 2011-07-10, latest release 5.0.2 (March 12, 2025)
- by Patrick Gundlach
- website: https://www.speedata.de/
- source code repo: https://github.com/speedata/publisher
- license: AGPL-3.0
- cross-platform
- written in Go, uses LuaLaTeX
- .xml files into PDF

speedata Publisher

- Since 2011-07-10, latest release 5.0.2 (March 12, 2025)
- by Patrick Gundlach
- website: https://www.spee
- source code repo: https://g
- license: AGPL-3.0
- cross-platform
- written in Go, uses LuaLa
- .xml files into PDF


1

Patrick Gundlach

Datenbasiertes Publizieren
mit dem speedata Publisher

oder

Wie bekomme ich schöne PDFs?

15.4.2023
Lightning Talk

Grazer LINUXTAGE

gundlach@speedata.de

speedata
Überholen wir den Mainstream

speedata Publisher

- Since 2011-07-10, latest release 5.0.2 (March 12, 2025)
- by Patrick Gundlach
- website: https://www.speedata.de/
- source code repo: https://github.com/speedata/publisher
- license: AGPL-3.0
- cross-platform
- written in Go, uses LuaLaTeχ
- .xml files into PDF

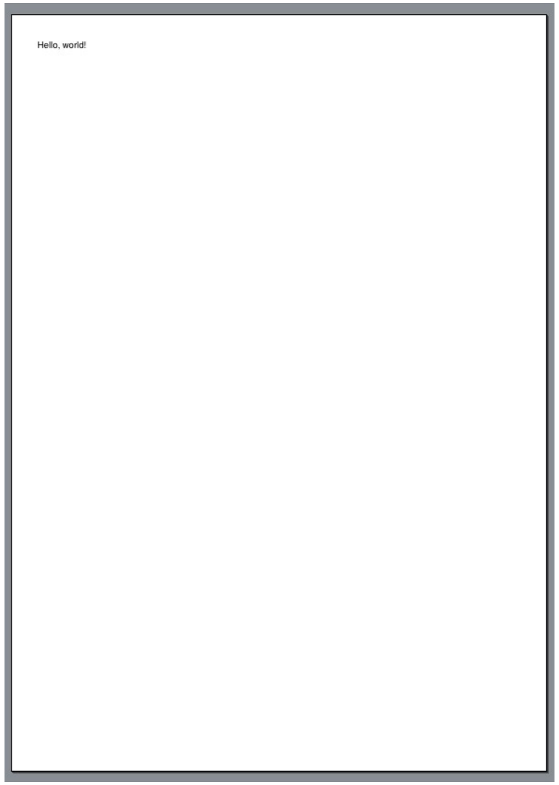
speedata Publisher

```
$ cat data.xml

<data>Hello, world!</data>

$ cat layout.xml

<Layout
  xmlns="urn:speedata.de:2009/publisher/en"
  xmlns:sd="urn:speedata:2009/publisher/functions/en">
  <Record element="data">
    <PlaceObject>
      <Textblock>
        <Paragraph>
          <Value select="."/>
        </Paragraph>
      </Textblock>
    </PlaceObject>
  </Record>
</Layout>

$
```

speedata Publisher

```
$ sp

Run speedata publisher 5.0.2

Finished with 0 errors and 0 warnings

Output written on publisher.pdf (1 pages, 3070 bytes)

Transcript written to publisher-protocol.xml

Total run time: 131ms

$ ls -1

data.xml
layout.xml
publisher-aux.xml
publisher.finished
publisher.pdf
publisher-protocol.xml
publisher.status
publisher.vars

$
```

speedata Publisher


Hello, world!

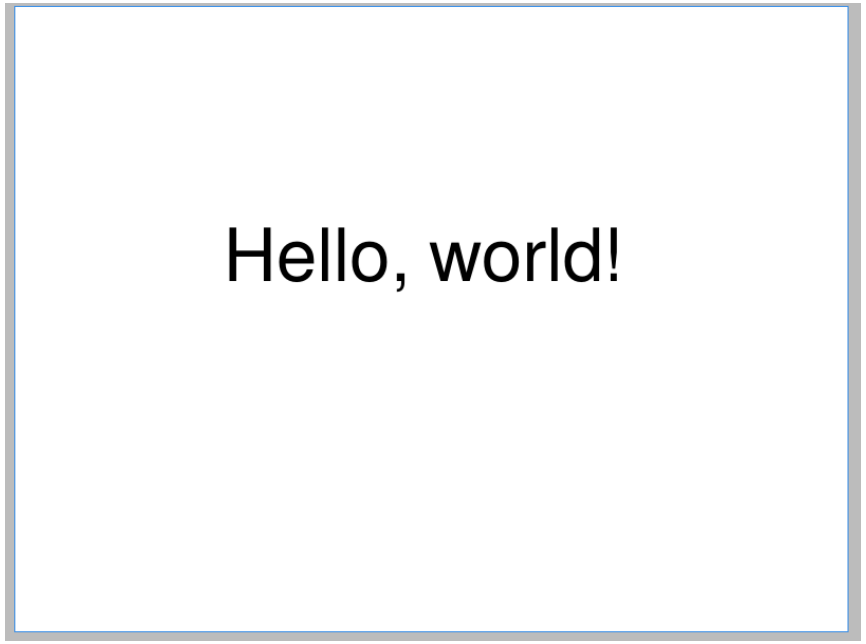
Hello, world!

speedata Publisher

```
$ cat data.xml

<data text="Hello, world!"/>

$ cat layout.xml

<Layout
  xmlns="urn:speedata.de:2009/publisher/en"
  xmlns:sd="urn:speedata:2009/publisher/functions/en">
  <Record element="data">
    <PlaceObject>
      <Textblock>
        <Paragraph>
          <Value select="./@text"/>
        </Paragraph>
      </Textblock>
    </PlaceObject>
  </Record>
</Layout>

$
```

speedata Publisher

```
$ cat data.xml

<data text=”Hello, world!”/>

$ cat layout.xml

<Layout
  xmlns="urn:speedata.de:2009/publisher/en"
  xmlns:sd="urn:speedata:2009/publisher/functions/en">
  <Pageformat width="4cm" height="3cm"/>
  <Record element="data">
    <PlaceObject>
      <Textblock>
        <Paragraph>
          <Value select="./@text"/>
        </Paragraph>
      </Textblock>
    </PlaceObject>
  </Record>
</Layout>

$
```



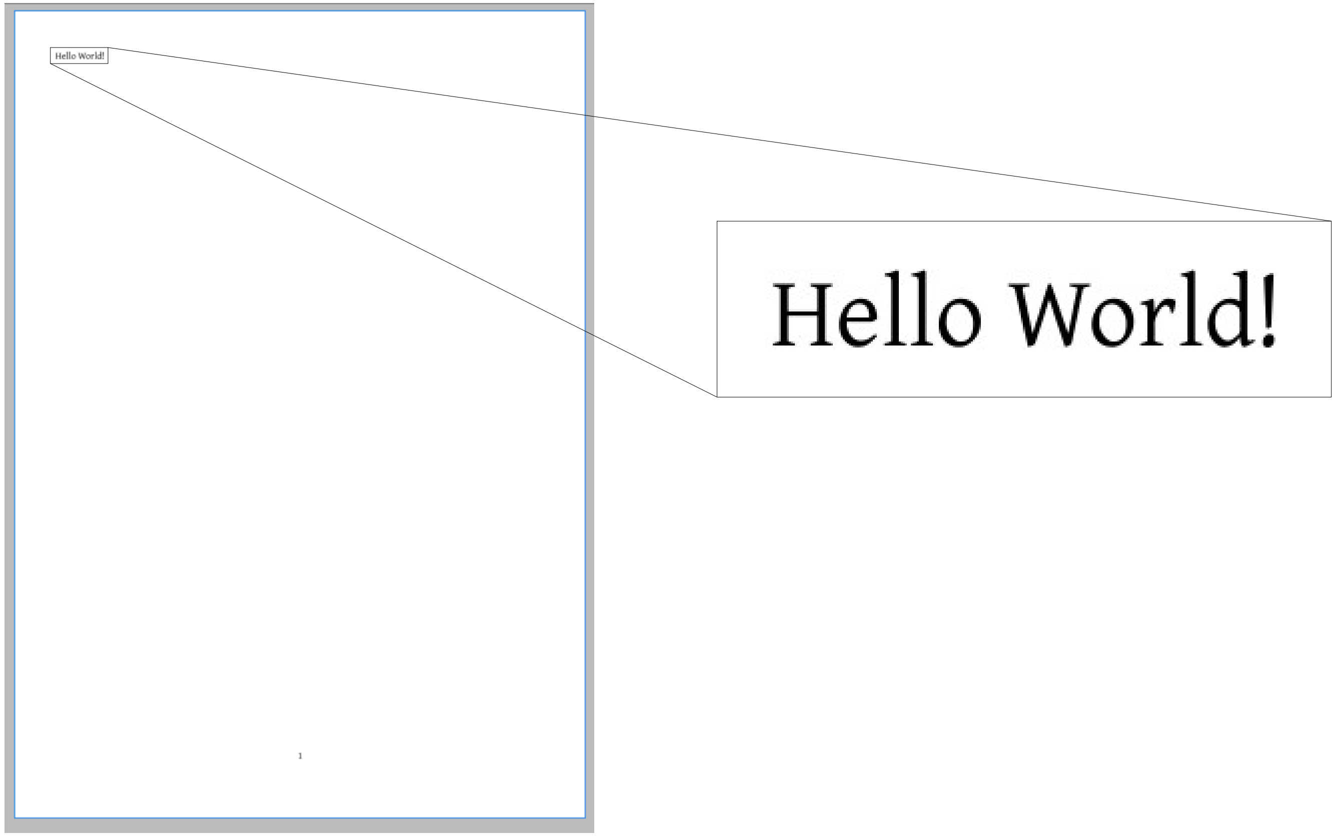
SILE

- Since 2012-07-29, latest release 0.15.12 (11 Apr 2025)
- by Simon Cozens, Caleb Maclennan
- website: https://sile-typesetter.org/
- source code repo: https://github.com/sile-typesetter/sile
- license: MIT
- (no Windows support)
- written in rust and Lua
- .sil files into .pdf files

SILE

```
$ cat example.sil
\begin{document}
Hello World!
\end{document}
$ sile example.sil
SILE v0.15.12 (LuaJIT 2.1.1703358377) [Rust]
<hello.sil> as sil
[1]
$
```

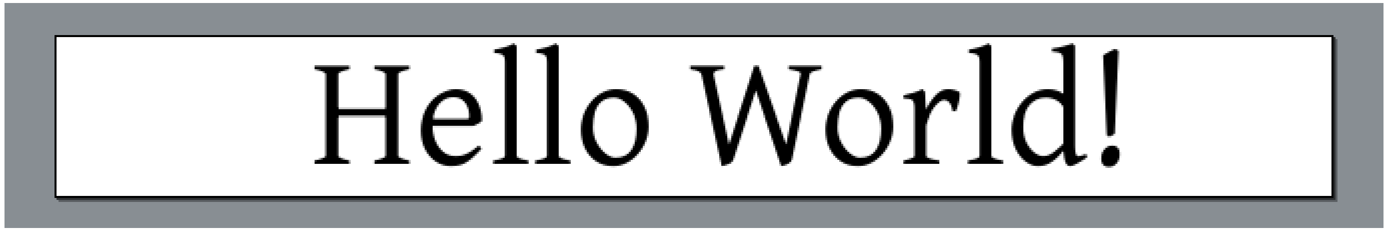
SILE


Hello World!

1

SILE

```
$ cat example.sil
\begin[papersize=10pt x 80pt,landscape=true]{document}
\nofolios
Hello World!
\end{document}
$ sile example.sil
SILE v0.15.12 (LuaJIT 2.1.1703358377) [Rust]
<hello.sil> as sil
[1]
$
```

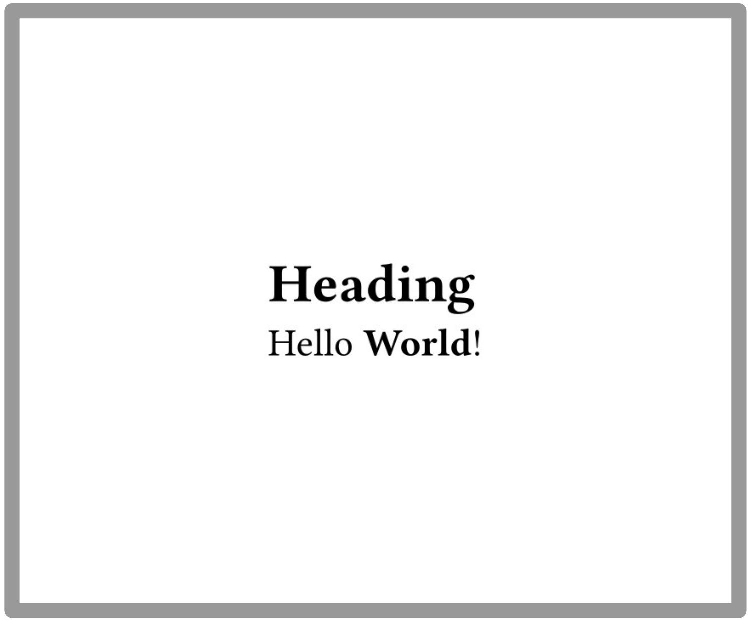

Hello World!

typst

- Since 2019-02-10, latest version: 0.13.1 (March 7, 2025)
- by Martin Haug, Laurenz Mädje, Ana Gelez
- website: https://typst.app/
- source code repo: https://github.com/typst/
- license: Apache v2
- cross-platform
- written in rust
- .typ files into PDF/PNG/SVG/HTML5 files

typst

```
$ cat example.typ
= Heading


Hello *World*!
$ typst compile example.typ
$
```

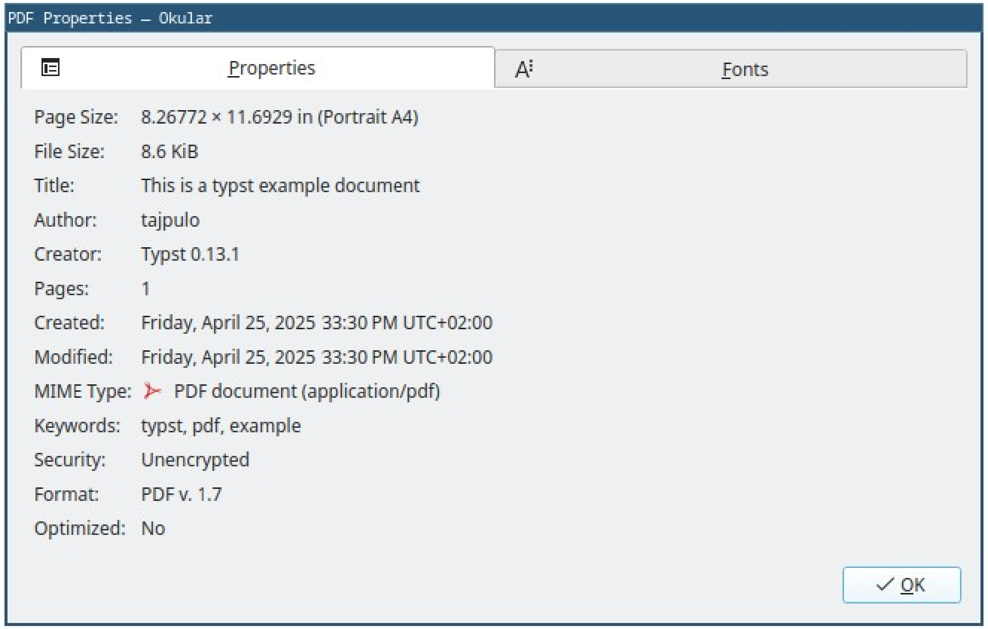


Heading

Hello **World**!

typst

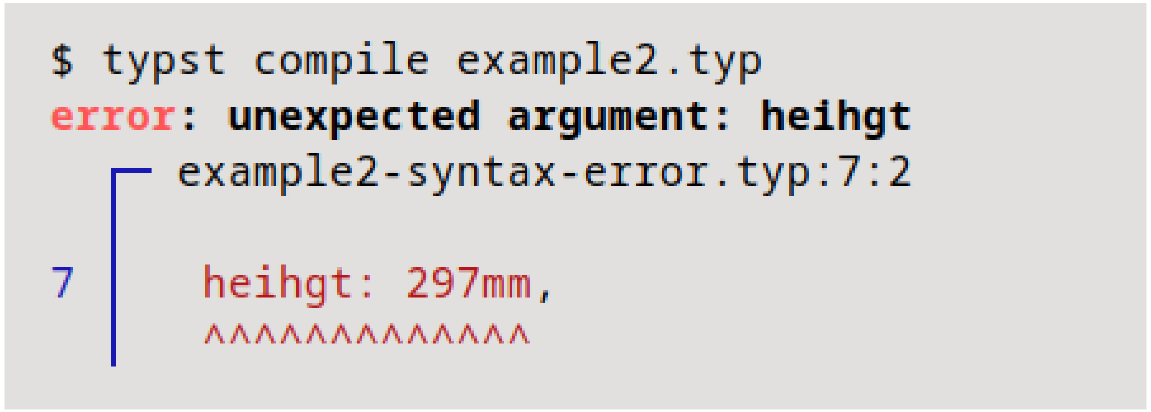
```
$ cat example.typ

#set document(
  title: "This is a typst example document",
  keywords: ("typst", "pdf", "example"),
  author: ("tajpulo"),
)

#set page(
  height: 297mm,
  width: 210mm,
)

= Heading

Hello *World*!

$ typst compile example.typ

$
```



Page Size:	8.26772 × 11.6929 in (Portrait A4)
File Size:	8.6 KiB
Title:	This is a typst example document
Author:	tajpulo
Creator:	Typst 0.13.1
Pages:	1
Created:	Friday, April 25, 2025 33:30 PM UTC+02:00
Modified:	Friday, April 25, 2025 33:30 PM UTC+02:00
MIME Type:	PDF document (application/pdf)
Keywords:	typst, pdf, example
Security:	Unencrypted
Format:	PDF v. 1.7
Optimized:	No

typst

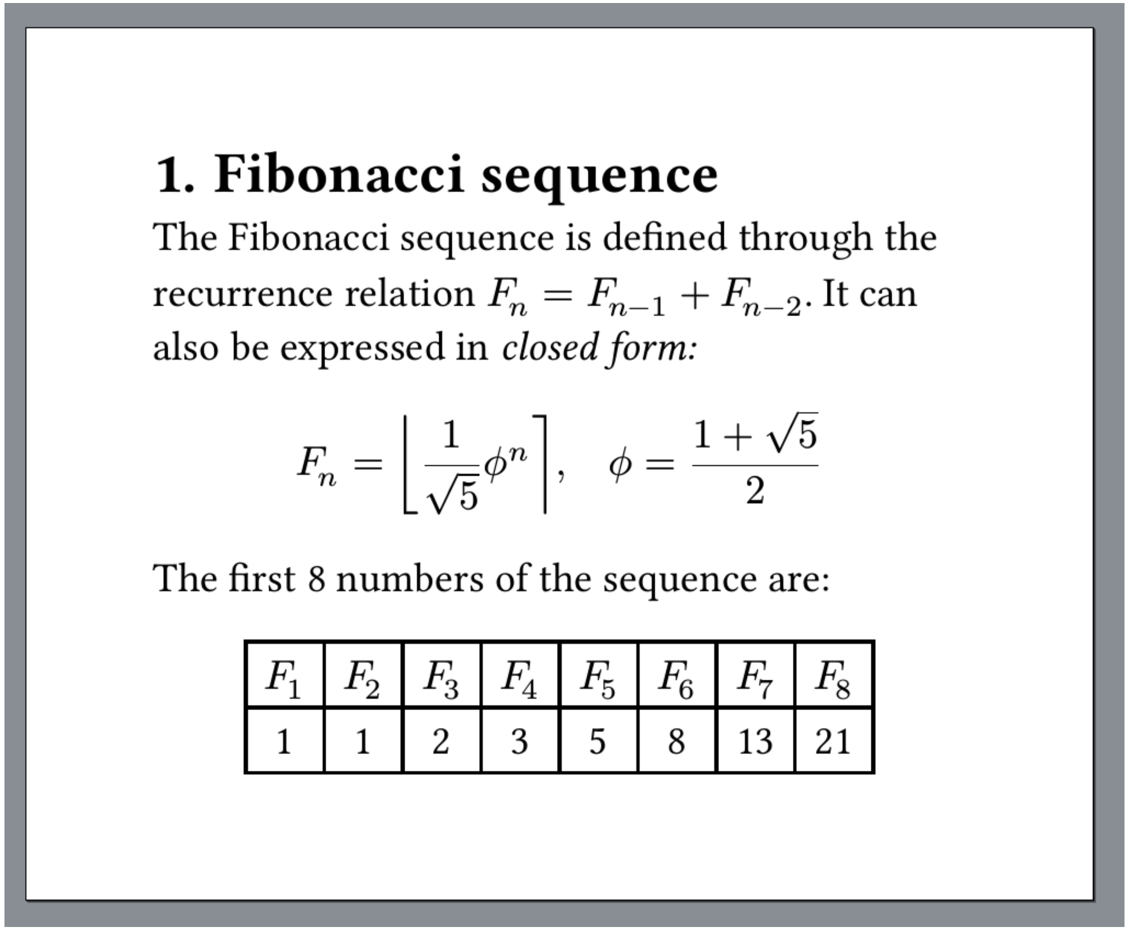
```
$ cat example.typ

#set document(
  title: "This is a typst example document",
  keywords: ("typst", "pdf", "example"),
  author: ("tajpulo"),
)

#set page(
  heihgt: 297mm,
  width: 210mm,
)

= Heading

Hello *World*!

$ typst compile example.typ

$
```



```
$ typst compile example2.typ
error: unexpected argument: heihgt
   ┌─ example2-syntax-error.typ:7:2
   │
 7 │   heihgt: 297mm,
   │   ^^^^^^^^^^^^^^
```

typst

```
$ cat official.typ

#set page(width: 10cm, height: auto)
#set heading(numbering: "1.")

= Fibonacci sequence
The Fibonacci sequence is defined through the
recurrence relation $F_n = F_(n-1) + F_(n-2)$.
It can also be expressed in _closed form:_

$ F_n = round(1 / sqrt(5) phi.alt^n), quad
  phi.alt = (1 + sqrt(5)) / 2 $

#let count = 8
#let nums = range(1, count + 1)
#let fib(n) = (
  if n <= 2 { 1 }
  else { fib(n - 1) + fib(n - 2) }
)

The first #count numbers of the sequence are:

#align(center, table(
  columns: count,
  ..nums.map(n => $F_#n$),
  ..nums.map(n => str(fib(n))),
))

$ typst compile official.typ
$
```



1. Fibonacci sequence

The Fibonacci sequence is defined through the recurrence relation $F_n = F_{n-1} + F_{n-2}$. It can also be expressed in *closed form:*

$$F_n = \left\lfloor \frac{1}{\sqrt{5}} \phi^n \right\rceil, \quad \phi = \frac{1+\sqrt{5}}{2}$$

The first 8 numbers of the sequence are:

F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
1	1	2	3	5	8	13	21

Are we digital typesetting yet?

summary:

Are we digital typesetting yet?

summary:

- Teχ
- speedata publisher
- SILE
- typst

criteria (today):

- FOSS
- paged output
- command line interface

Are we digital typesetting yet?

	input	*output*	*math support*	*recom. usecase*
*Teχ	custom lang	PDF	great!	domain-specific output style
sp	XML	PDF	no	catalogues & brochures
SILE	custom lang & adjustable	PDF	rudimentary	self-published books
typst	custom lang	PDF & HTML	good	academic papers

Are we digital typesetting yet?

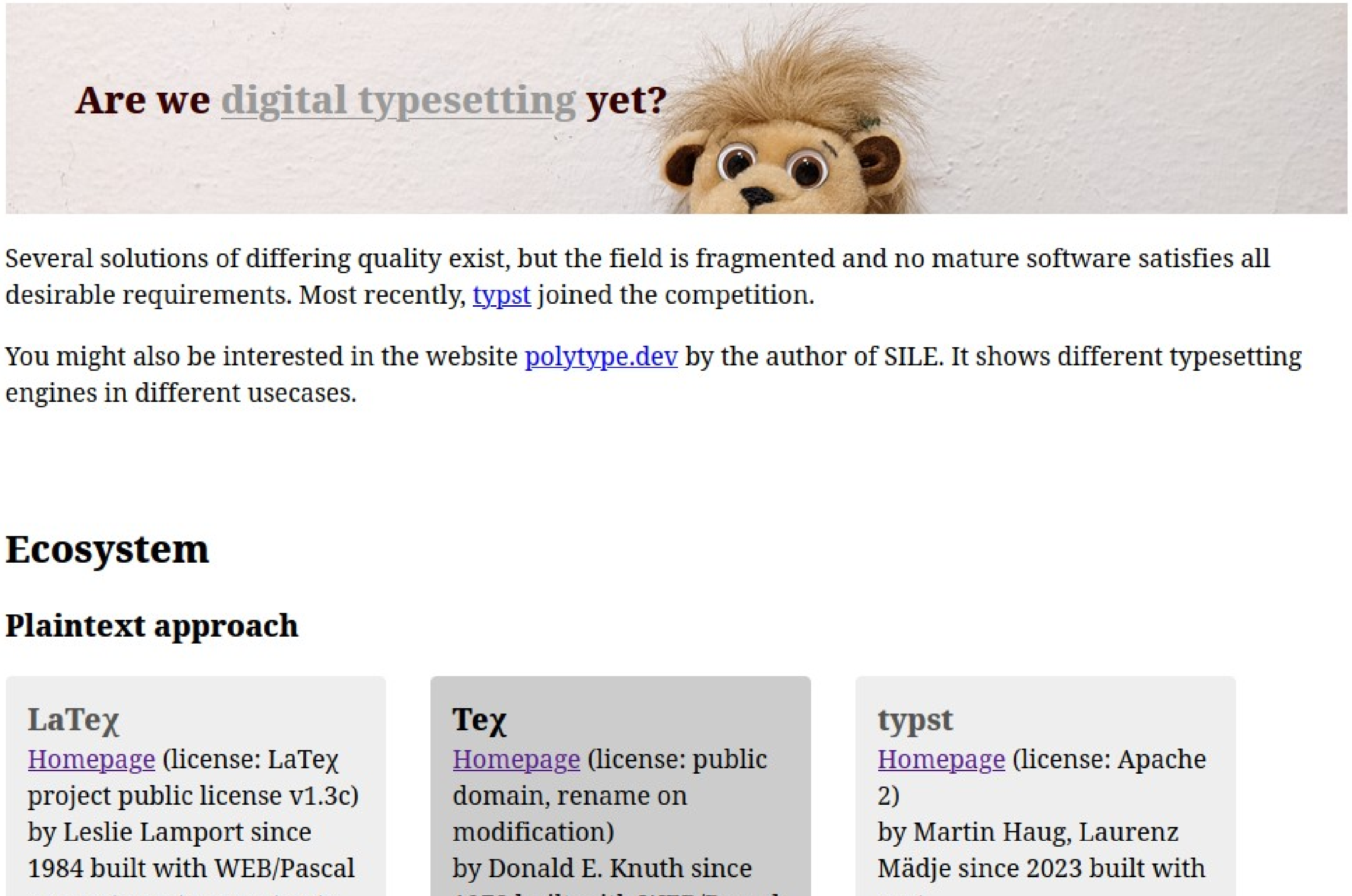
wiki.mozilla.org/Areweyet

areweguiyet
arewewebyet
arewequantumyet
…

Are we digital typesetting yet?

The roots are shallow and historic,
but progress can be observed.

https://Arewedigitaltypesettingyet.com/



Are we digital typesetting yet?

Several solutions of differing quality exist, but the field is fragmented and no mature software satisfies all desirable requirements. Most recently, typst joined the competition.

You might also be interested in the website polytype.dev by the author of SILE. It shows different typesetting engines in different usecases.

Ecosystem

Plaintext approach

LaTeχ
Homepage (license: LaTeχ project public license v1.3c)
by Leslie Lamport since
1984 built with WEB/Pascal

Teχ
Homepage (license: public domain, rename on modification)
by Donald E. Knuth since

typst
Homepage (license: Apache 2)
by Martin Haug, Laurenz Mädje since 2023 built with

https://Arewedigitaltypesettingyet.com/

- Run by the club “Verein zur Förderung von digitalem Textsatz”
- Do you like my work? Sponsor me on github!
- Thank you!