The \TeX{} hyphenation applied to HTML

About Frank M. Liangs hyphenation algorithm and its port to Javascript

Mathias Nater
mathias@mnn.ch
http://mnn.ch/

Bacho\TeX{} 2010
Organisation

Motivation
Text layout without hyphenation
Text layout with hyphenation

The TeX hyphenation algorithm
The original TeX hyphenation algorithm (1977)
The current TeX hyphenation algorithm (1983)
Creating the patterns (patgen)
Using the patterns (hyphenation)

HTML and the soft hyphen

The Port to Javascript
Server side or Client side?
How it works
Differences and Improvements
Back to the Future
Motivation

Text layout without hyphenation
Text layout with hyphenation

The TeX hyphenation algorithm

The original TeX hyphenation algorithm (1977)
The current TeX hyphenation algorithm (1983)
Creating the patterns (patgen)
Using the patterns (hyphenation)

HTML and the soft hyphen

The Port to Javascript
Server side or Client side?
How it works
Differences and Improvements
Back to the Future
Organisation

Motivation

Text layout without hyphenation
Text layout with hyphenation

The \TeX\ hyphenation algorithm

The original \TeX\ hyphenation algorithm (1977)
The current \TeX\ hyphenation algorithm (1983)
Creating the patterns (patgen)
Using the patterns (hyphenation)

HTML and the soft hyphen

The Port to Javascript
Server side or Client side?
How it works
Differences and Improvements
Back to the Future
Organisation

Motivation
Text layout without hyphenation
Text layout with hyphenation

The \TeX{} hyphenation algorithm
The original \TeX{} hyphenation algorithm (1977)
The current \TeX{} hyphenation algorithm (1983)
Creating the patterns (patgen)
Using the patterns (hyphenation)

HTML and the soft hyphen

The Port to Javascript
Server side or Client side?
How it works
Differences and Improvements
Back to the Future
Hyphenation for HTML

Mathias Nater
mathias@mnn.ch
http://mnn.ch/

Motivation
layout w/o hyphenation
layout with hyphenation

The \TeX\ hyphenation algorithm
The original \TeX\ hyphenation algorithm (1977)
The current \TeX\ hyphenation algorithm (1983)
Creating the patterns
Using the patterns

HTML and the soft hyphen

The Port to Javascript
Server side or Client side?
How it works
Differences and Improvements
Back to the Future

Organisation

need

word list
patgen

hyphenation patterns

hyphenation algorithm

soft hyphen

hyphenator

creating the patterns (patgen)
using the patterns (hyphenation)
Text layout without hyphenation

Current Browsers
- MS IE 6/7/8 (∼ 44%)
- Firefox 3.5 (∼ 42%)
- Safari 4 (∼ 4%)
- Opera 10 (∼ 3%)

Do not hyphenate text automatically!
- align left: overfull boxes and unbalanced line endings
- justified: big word spaces and rivers

Motivation
Layout w/o hyphenation
Layout with hyphenation

The TeX hyphenation algorithm
The original TeX hyphenation algorithm (1977)
The current TeX hyphenation algorithm (1983)
Creating the patterns (patgen)
Using the patterns (hyphenation)

HTML and the soft hyphen

The Port to Javascript
Server side or Client side?
How it works
Differences and improvements
Back to the Future
Text layout without hyphenation

Current Browsers

- MS IE 6/7/8 (~ 44%)
- Firefox 3.5 (~ 42%)
- Safari 4 (~ 4%)
- Opera 10 (~ 3%)

do not hyphenate text automatically!

- align left: overfull boxes and unbalanced line endings
- justified: big word spaces and rivers
Text layout without hyphenation

Current Browsers

- MS IE 6/7/8 (∼ 44%)
- Firefox 3.5 (∼ 42%)
- Safari 4 (∼ 4%)
- Opera 10 (∼ 3%)

do not hyphenate text automatically!

This leads to poor typography:

- align left: overfull boxes and unbalanced line endings
- justified: big word spaces and rivers
Text layout without hyphenation

Current Browsers

- MS IE 6/7/8 (∼ 44%)
- Firefox 3.5 (∼ 42%)
- Safari 4 (∼ 4%)
- Opera 10 (∼ 3%)

do not hyphenate text automatically!

This leads to poor typography:

- align left: overfull boxes and unbalanced line endings
- justified: big word spaces and rivers
Text layout without hyphenation

Current Browsers
- MS IE 6/7/8 (∼ 44%)
- Firefox 3.5 (∼ 42%)
- Safari 4 (∼ 4%)
- Opera 10 (∼ 3%)

do not hyphenate text automatically!

This leads to poor typography:
- align left: overfull boxes and unbalanced line endings
- justified: big word spaces and rivers
Motivation
layout w/o hyphenation
layout with hyphenation

The TEX hyphenation algorithm
The original TEX hyphenation algorithm (1977)
The current TEX hyphenation algorithm (1983)
Creating the patterns (patgen)
Using the patterns (hyphenation)

HTML and the soft hyphen

The Port to Javascript
Server side or Client side?
How it works
Differences and Improvements
Back to the Future

Text layout without hyphenation
text-align: left;
Hyphenation for HTML

Mathias Nater
mathias@mnn.ch
http://mnn.ch/

Motivation
layout w/o hyphenation
layout with hyphenation

The TeX hyphenation algorithm
The original TeX hyphenation algorithm (1977)
The current TeX hyphenation algorithm (1983)
Creating the patterns (patgen)
Using the patterns (hyphenation)

HTML and the soft hypen

The Port to Javascript
Server side or Client side?
How it works
Differences and Improvements
Back to the Future

Text layout without hyphenation
text-align: justify;
Hyphenation for HTML

Mathias Nater
mathias@mnn.ch
http://mnn.ch/

Motivation

layout w/o hyphenation
layout with hyphenation

The \TeX\ hyphenation algorithm

The original \TeX\ hyphenation algorithm (1977)
The current \TeX\ hyphenation algorithm (1983)

Creating the patterns (patgen)
Using the patterns (hyphenation)

HTML and the soft hyphen

The Port to Javascript

Server side or Client side?
How it works
Differences and Improvements
Back to the Future
Text layout with hyphenation

text-align: left;

DER MÄDCHEN-EFFEKT
von Birgit Schmid, 1 Kommentar

WENN HELVETIA HADERT
von Peter Ziegler, 8 Kommentare
In diesen wirren Zeiten dominieren die Kopfjäger, Alleingänger und Quick-Fix-Technokraten die innenpolitische Diskussion. Doch was es wirklich braucht, ist eine Regierungsreform, eine gescheite Bankenregulierung, und, ja, den EU-Beitritt.

PROJEKT 80*81
von Das Magazin
Die Achtzigerjahre in Bildern

AUS DER REDAKTION
Projekt 80*81

JOURNAL DER GEGENWART
Verantwortungsscheue Volksvertreter
Glücklich werden #4
Plüsch, empirisch
KÜNG
Max Küng

MONATSGESPRÄCH
Alain de Botton Teil 4

HIER FRAGT DER CHEF
Finn Canonica auf DRS 4 Talk

Motivation
layout w/o hyphenation
layout with hyphenation

The TeX hyphenation algorithm
The original TeX hyphenation algorithm (1977)
The current TeX hyphenation algorithm (1983)
Creating the patterns (patgen)
Using the patterns (hyphenation)

HTML and the soft hyphen

The Port to Javascript
Server side or Client side?
How it works
Differences and Improvements
Back to the Future
Text layout with hyphenation

text-align: justify;

DER MÄDCHEN-EFFEKT
von Birgit Schmid, 1 Kommentar

WENN HELVETIA HADERT
von Peter Ziegler, 8 Kommentare
In diesen wirren Zeiten dominieren die Kopfjäger, Alleingänger und Quick-Fix-Technokraten die innenpolitische Diskussion. Doch was es wirklich braucht, ist eine Regierungsreform, eine gescheite Bankenregulierung, und ja, den EU-Beitritt.

PROJEKT 80*81
von Das Magazin
Die Achtzigerjahre in Bildern
Text layout with hyphenation

motivation

layout w/o hyphenation

layout with hyphenation

The \TeX\ hyphenation algorithm

The original \TeX\ hyphenation algorithm (1977)

The current \TeX\ hyphenation algorithm (1983)

Creating the patterns (patgen)

Using the patterns (hyphenation)

HTML and the soft hyphen

The Port to Javascript

Server side or Client side?

How it works

Differences and Improvements

Back to the Future
We need (automatic) hyphenation in HTML!
The original TeX hyphenation algorithm

1977 by Donald E. Knuth and Franklin M. Liang

- for english only
- suffix and prefix removal
- vowel-consonant-consonant-vowel breaking
- special case rules (e.g. “break after ck!”)
- small exception dictionary
The original TeX hyphenation algorithm

1977 by Donald E. Knuth and Franklin M. Liang

- for English only
- suffix and prefix removal
- vowel-consonant-consonant-vowel breaking
- special case rules (e.g. “break after ck!”)
- small exception dictionary
The original TeX hyphenation algorithm

1977 by Donald E. Knuth and Franklin M. Liang

- for english only
- suffix and prefix removal
- vowel-consonant-consonant-vowel breaking
- special case rules (e.g. “break after ck!”)
- small exception dictionary
The original TeX hyphenation algorithm

1977 by Donald E. Knuth and Franklin M. Liang

- for english only
- suffix and prefix removal
- vowel-consonant-consonant-vowel breaking
- special case rules (e.g. “break after ck!”)
- small exception dictionary
The original TeX hyphenation algorithm

1977 by Donald E. Knuth and Franklin M. Liang

- for english only
- suffix and prefix removal
- vowel-consonant-consonant-vowel breaking
- special case rules (e.g. “break after ck!”)
- small exception dictionary
1977 by Donald E. Knuth and Franklin M. Liang

- for English only
- suffix and prefix removal
- vowel-consonant-consonant-vowel breaking
- special case rules (e.g. “break after ck!”)
- small exception dictionary
The original \TeX\ hyphenation algorithm

The original hyphenation algorithm

1977 by Donald E. Knuth and Franklin M. Liang

- for english only
- suffix and prefix removal
- vowel-consonant-consonant-vowel breaking
- special case rules (e.g. “break after ck!”)
- small exception dictionary

Found \(\sim 40\%\) of the allowable hyphen points with \(1\%\) error
The current TeX hyphenation algorithm

1983 PhD thesis by Franklin M. Liang

▶ use of *hyphenation patterns*

▶ two algorithms:
  ▶ pattern creation (patgen)
  ▶ applying the patterns (TeX)

▶ support for a wide range of languages

▶ small, easy, fast
The current TeX hyphenation algorithm

1983 PhD thesis by Franklin M. Liang

- use of hyphenation patterns
- two algorithms:
  - pattern creation (patgen)
  - applying the patterns (TeX)
- support for a wide range of languages
- small, easy, fast
The current TeX hyphenation algorithm

The current hyphenation algorithm

1983 PhD thesis by Franklin M. Liang

- use of *hyphenation patterns*
- two algorithms:
  - pattern creation *(patgen)*
  - applying the patterns *(TeX)*
- support for a wide range of languages
- small, easy, fast
The current TeX hyphenation algorithm

1983 PhD thesis by Franklin M. Liang

- use of *hyphenation patterns*
- two algorithms:
  - pattern creation (patgen)
  - applying the patterns (TeX)
- support for a wide range of languages
- small, easy, fast
Creating patterns with patgen

- INPUT: a list of hyphenated words [, precomputed pattern, translate file]
- takes up to 9 runs (asking for many settings, adding a new level in each run)
- OUTPUT: pattern file, statistics (a lot!)
- old code
- no UTF-8
- refactored by David Antoš (OPatGen), but doesn’t compile
Creating patterns with patgen

- **INPUT**: a list of hyphenated words [, precomputed pattern, translate file]
- takes up to 9 runs (asking for many settings, adding a new level in each run)
- **OUTPUT**: pattern file, statistics (a lot!)

- old code
- no UTF-8
- refactored by David Antoš (OPatGen), but doesn’t compile
Creating patterns with patgen

- **INPUT:** a list of hyphenated words [, precomputed pattern, translate file]
- takes up to 9 runs (asking for many settings, adding a new level in each run)
- **OUTPUT:** pattern file, statistics (a lot!)

- old code
- no UTF-8
- refactored by David Antoš (OPatGen), but doesn’t compile
Creating patterns with patgen

- INPUT: a list of hyphenated words [, precomputed pattern, translate file]
- takes up to 9 runs (asking for many settings, adding a new level in each run)
- OUTPUT: pattern file, statistics (a lot!)

- old code
- no UTF-8
- refactored by David Antoš (OPatGen), but doesn’t compile
Creating patterns with patgen

- **INPUT**: a list of hyphenated words [, precomputed pattern, translate file]
- takes up to 9 runs (asking for many settings, adding a new level in each run)
- **OUTPUT**: pattern file, statistics (a lot!)

- old code
- no UTF-8

- refactored by David Antoš (**OPatGen**), but doesn’t compile
Creating patterns with patgen

- INPUT: a list of hyphenated words [, precomputed pattern, translate file]
- takes up to 9 runs (asking for many settings, adding a new level in each run)
- OUTPUT: pattern file, statistics (a lot!)

- old code
- no UTF-8
- refactored by David Antoš (OPatGen), but doesn’t compile
applying the patterns

- .in1 b2l2 4edi b4le.
- patterns: short strings with integer values
- odd values: valid breakpoints
- even values: forbidden breakpoints
- lower values are overwritten by higher values
- points mark begin/end of the word
Hyphenation for HTML
Mathias Nater
mathias@mnn.ch
http://mnn.ch/

Motivation
- layout w/o hyphenation
- layout with hyphenation

The \TeX\ hyphenation algorithm
- The original \TeX\ hyphenation algorithm (1977)
- The current \TeX\ hyphenation algorithm (1983)
  - Creating the patterns (patgen)
  - Using the patterns (hyphenation)

HTML and the soft hyphen

The Port to Javascript
- Server side or Client side?
- How it works
- Differences and Improvements
- Back to the Future

applying the patterns

- \texttt{.in1 b2l2 4edi b4le.}
- patterns: short strings with integer values
  - odd values: valid breakpoints
  - even values: forbidden breakpoints
  - lower values are overwritten by higher values
  - points mark begin/end of the word
applying the patterns

- .in1 b212 4edi b4le.
- patterns: short strings with integer values
- odd values: valid breakpoints
- even values: forbidden breakpoints
- lower values are overwritten by higher values
- points mark begin/end of the word
applying the patterns

- \texttt{.in1 b2l2 4edi b4le.}
- patterns: short strings with integer values
- odd values: valid breakpoints
- even values: forbidden breakpoints
- lower values are overwritten by higher values
- points mark begin/end of the word
applying the patterns

▶ .in1 b2l2 4edi b4le.
▶ patterns: short strings with integer values
▶ odd values: valid breakpoints
▶ even values: forbidden breakpoints
▶ lower values are overwritten by higher values
▶ points mark begin/end of the word
applying the patterns (example)

incredible

. i n c r e d i b l e .
. i n l

   b2l2
   4e d i
   i1b l

 n1c r
  b4l e .

 5c r e d
 e d3i b
 2r2e d

-------------

. i n5c2r4e d3i1b4l2e .
in-cred-i-ble
applying the patterns (example)

incredible

\text{incredible.}

\text{incredible.}

\text{incredible.}

\text{incredible.}

\text{incredible.}
applying the patterns (example)

incredible
\[
\text{. i n c r e d i b l e .}
\]
\[
\text{. i n l}
\]
\[
\text{b2l2}
\]
\[
\text{4e d i}
\]
\[
\text{i1b l}
\]
\[
\text{n1c r}
\]
\[
\text{b4l e .}
\]
\[
\text{5c r e d}
\]
\[
\text{e d3i b}
\]
\[
\text{2r2e d}
\]
\[
\text{-------------}
\]
\[
\text{. i n5c2r4e d3i1b4l2e .}
\]
in-cred-i-ble
applying the patterns (example)

incredible
    .i n c r e d i b l e .
    .i n 1
        b212
        4e d i
        i1b l
        n1c r
            b4l e .
    5c r e d
        e d3i b
    2r2e d
---------
    .i n 5c2r4e d3i1b4l2e .
in-cred-i-ble
applying the patterns (example)

incredible
.i n c r e d i b l e .
i n 1
  b2l2
  4e d i
    i1b l
  n1c r
    b4l e .
  5c r e d
    e d3i b
    2r2e d
--------------
  . i n5c2r4e d3i1b4l2e .
in-credd-i-ble
HTML and the Soft Hyphen

- limited control over text layout
  (text-align: left | right | justify)
- manual line breaks (<br>)
- manually inserted soft hyphens
  (&shy; – discretionary hyphen)
- some more controls are upcoming with CSS3

- laying out text is up to the browser
- developer has no control over how text is displayed
HTML and the Soft Hyphen

- limited control over textlayout
  (text-align: left | right | justify)
- manual line breaks (<br>)
- manually inserted soft hyphens (&shy; – discretionary hyphen)
- some more controls are upcoming with CSS3
- laying out text is up to the browser
- developer has no control over how text is displayed
HTML and the Soft Hyphen

- limited control over text layout
  (text-align: left | right | justify)
- manual line breaks (<br>)
- manually inserted soft hyphens
  (&shy; – discretionary hyphen)
- some more controls are upcoming with CSS3
- laying out text is up to the browser
- developer has no control over how text is displayed
HTML and the Soft Hyphen

- limited control over text layout
  - (text-align: left | right | justify)
- manual line breaks (<br>)
- manually inserted soft hyphens
  - (&shy; – discretionary hyphen)
- some more controls are upcoming with CSS3

- laying out text is up to the browser
- developer has no control over how text is displayed
HTML and the Soft Hyphen

- limited control over text layout
  (text-align: left | right | justify)
- manual line breaks (<br>)
- manually inserted soft hyphens (&shy; – discretionary hyphen)
- some more controls are upcoming with CSS3

- laying out text is up to the browser
- developer has no control over how text is displayed
Text layout with hyphenation

text-align: justify;

DER MÄDCHEN-EFFEKT
von Birgit Schmid, 1 Kommentar

WENN HELVETIA HADERT
von Peter Ziegler, 8 Kommentare
In diesen wirren Zeiten dominieren die Kopfjäger, Alleingänger und Quick-Fix-Technokraten die innenpolitische Diskussion. Doch was es wirklich braucht, ist eine Regierungsreform, eine gescheite Bankenregulierung, und ja, den EU-Beitritt.

PROJEKT 80*81
von Das Magazin
Die Achtzigerjahre in Bildern
Text layout with hyphenation

text-align: justify;

Hyphenation for HTML

Mathias Nater
mathias@mnn.ch
http://mnn.ch/

Motivation

layout w/o hyphenation
layout with hyphenation

The TEX hyphenation algorithm

The original TEX hyphenation algorithm (1977)
The current TEX hyphenation algorithm (1983)

Creating the patterns (patgen)
Using the patterns (hyphenation)

HTML and the soft hyphen

The Port to Javascript

Server side or Client side?
How it works
Differences and Improvements
Back to the Future
Putting all together

Motivation
- layout w/o hyphenation
- layout with hyphenation

The TeX hyphenation algorithm
- The original TeX hyphenation algorithm (1977)
- The current TeX hyphenation algorithm (1983)
- Creating the patterns (patgen)
- Using the patterns (hyphenation)

HTML and the soft hyphen

The Port to Javascript
- Server side or Client side?
- How it works
- Differences and Improvements
- Back to the Future

HTML and the soft hyphen

The Port to Javascript
- Server side or Client side?
- How it works
- Differences and Improvements
- Back to the Future

Hyphenation for HTML
Mathias Nater
mathias@mnn.ch
http://mnn.ch/
Server side or Client side hyphenation?

Pro server side:
- lower bandwidth usage
- faster
- only hyphenate once, store the result

Pro client side:
- cleaner HTML (search engines!)
- takes in count client oddities
- can be switched on/off
- hyphenation is part of CSS3, so even the W3C believes that hyphenation belongs to the client
- user generated text can be hyphenated on the fly
Server side or Client side hyphenation?

Pro server side:
- lower bandwidth usage
- faster
- only hyphenate once, store the result

Pro client side:
- cleaner HTML (search engines!)
- takes in count client oddities
- can be switched on/off
- hyphenation is part of CSS3, so even the W3C believes that hyphenation belongs to the client
- user generated text can be hyphenated on the fly
Server side or Client side hyphenation?

Pro server side:
- lower bandwidth usage
- faster
  - only hyphenate once, store the result

Pro client side:
- cleaner HTML (search engines!)
- takes in count client oddities
- can be switched on/off
- hyphenation is part of CSS3, so even the W3C believes that hyphenation belongs to the client
- user generated text can be hyphenated on the fly
Server side or Client side hyphenation?

Pro server side:
- lower bandwidth usage
- faster
- only hyphenate once, store the result

Pro client side:
- cleaner HTML (search engines!)
- takes in count client oddities
- can be switched on/off
- hyphenation is part of CSS3, so even the W3C believes that hyphenation belongs to the client
- user generated text can be hyphenated on the fly
Server side or Client side hyphenation?

Pro server side:
- lower bandwidth usage
- faster
- only hyphenate once, store the result

Pro client side:
- cleaner HTML (search engines!)
- takes in count client oddities
- can be switched on/off
- hyphenation is part of CSS3, so even the W3C believes that hyphenation belongs to the client
- user generated text can be hyphenated on the fly
Server side or Client side hyphenation?

Pro server side:
- lower bandwidth usage
- faster
- only hyphenate once, store the result

Pro client side:
- cleaner HTML (search engines!)
- takes in count client oddities
- can be switched on/off
- hyphenation is part of CSS3, so even the W3C believes that hyphenation belongs to the client
- user generated text can be hyphenated on the fly
Server side or Client side hyphenation?

Pro server side:
- lower bandwidth usage
- faster
- only hyphenate once, store the result

Pro client side:
- cleaner HTML (search engines!)
- takes in count client oddities
- can be switched on/off
- hyphenation is part of CSS3, so even the W3C believes that hyphenation belongs to the client
- user generated text can be hyphenated on the fly
Server side or Client side hyphenation?

Pro server side:
- lower bandwidth usage
- faster
- only hyphenate once, store the result

Pro client side:
- cleaner HTML (search engines!)
- takes in count client oddities
- can be switched on/off
- hyphenation is part of CSS3, so even the W3C believes that hyphenation belongs to the client
- user generated text can be hyphenated on the fly
Server side or Client side hyphenation?

Pro server side:
- lower bandwidth usage
- faster
- only hyphenate once, store the result

Pro client side:
- cleaner HTML (search engines!)
- takes in count client oddities
- can be switched on/off
- hyphenation is part of CSS3, so even the W3C believes that hyphenation belongs to the client
- user generated text can be hyphenated on the fly
My Decision

- server side solutions already existed: php, perl, java, python
- I believe that hyphenation has to be done in the client
- Javascript is a very interesting language
- the acceptance of Javascript is growing
- Firefox 2 didn’t support &shy;
- I like bookmarklets

- hyphenator.js: client-side hyphenation
- it’s proofing to be a good decision:
  - other – webkit based – programs are using hyphenator
  - it’s easy to use
  - there’s a big effort on making javascript faster
My Decision

- server side solutions already existed: php, perl, java, python
- I believe that hyphenation has to be done in the client
- Javascript is a very interesting language
- the acceptance of Javascript is growing
- Firefox 2 didn’t support &shy;
- I like bookmarklets

- hyphenator.js: client-side hyphenation
- it’s proofing to be a good decision:
  - other – webkit based – programs are using hyphenator
  - it’s easy to use
  - there’s a big effort on making javascript faster
My Decision

- server side solutions already existed: php, perl, java, python
- I believe that hyphenation has to be done in the client
- Javascript is a very interesting language
- the acceptance of Javascript is growing
- Firefox 2 didn’t support &shy;
- I like bookmarklets

- hyphenator.js: client-side hyphenation
- it’s proofing to be a good decision:
  - other – webkit based – programs are using hyphenator
  - it’s easy to use
  - there’s a big effort on making javascript faster
My Decision

- server side solutions already existed: php, perl, java, python
- I believe that hyphenation has to be done in the client
- Javascript is a very interesting language
- the acceptance of Javascript is growing
- Firefox 2 didn’t support &shy;
- I like bookmarklets

- hyphenator.js: client-side hyphenation
- it’s proofing to be a good decision:
  - other – webkit based – programs are using hyphenator
  - it’s easy to use
  - there’s a big effort on making javascript faster
My Decision

- server side solutions already existed: php, perl, java, python
- I believe that hyphenation has to be done in the client
- Javascript is a very interesting language
- the acceptance of Javascript is growing
- Firefox 2 didn’t support \&shy;
- I like bookmarklets

- hyphenator.js: client-side hyphenation
- it’s proofing to be a good decision:
  - other – webkit based – programs are using hyphenator
  - it’s easy to use
  - there’s a big effort on making javascript faster
My Decision

- server side solutions already existed: php, perl, java, python
- I believe that hyphenation has to be done in the client
- Javascript is a very interesting language
- the acceptance of Javascript is growing
- Firefox 2 didn’t support &shy;
- I like bookmarklets

- hyphenator.js: client-side hyphenation
- it’s proofing to be a good decision:
  - other – webkit based – programs are using hyphenator
  - it’s easy to use
  - there’s a big effort on making javascript faster
My Decision

- server side solutions already existed: php, perl, java, python
- I believe that hyphenation has to be done in the client
- Javascript is a very interesting language
- the acceptance of Javascript is growing
- Firefox 2 didn’t support &shy;
- I like bookmarklets

- hyphenator.js: client-side hyphenation
- it’s proofing to be a good decision:
  - other – webkit based – programs are using hyphenator
  - it’s easy to use
  - there’s a big effort on making javascript faster

Motivation
layout w/o hyphenation
layout with hyphenation

The TeX hyphenation algorithm
- The original TeX hyphenation algorithm (1977)
- The current TeX hyphenation algorithm (1983)
- Creating the patterns (patgen)
- Using the patterns (hyphenation)

HTML and the soft hyphen

The Port to Javascript
Server side or Client side?
- How it works
- Differences and Improvements
- Back to the Future

My Decision

- server side solutions already existed: php, perl, java, python
- I believe that hyphenation has to be done in the client
- Javascript is a very interesting language
- the acceptance of Javascript is growing
- Firefox 2 didn’t support &shy;
- I like bookmarklets

- hyphenator.js: client-side hyphenation
- it’s proofing to be a good decision:
  - other – webkit based – programs are using hyphenator
  - it’s easy to use
  - there’s a big effort on making javascript faster

Motivation
layout w/o hyphenation
layout with hyphenation

The TeX hyphenation algorithm
- The original TeX hyphenation algorithm (1977)
- The current TeX hyphenation algorithm (1983)
- Creating the patterns (patgen)
- Using the patterns (hyphenation)

HTML and the soft hyphen

The Port to Javascript
Server side or Client side?
- How it works
- Differences and Improvements
- Back to the Future

My Decision

- server side solutions already existed: php, perl, java, python
- I believe that hyphenation has to be done in the client
- Javascript is a very interesting language
- the acceptance of Javascript is growing
- Firefox 2 didn’t support &shy;
- I like bookmarklets

- hyphenator.js: client-side hyphenation
- it’s proofing to be a good decision:
  - other – webkit based – programs are using hyphenator
  - it’s easy to use
  - there’s a big effort on making javascript faster

Motivation
layout w/o hyphenation
layout with hyphenation

The TeX hyphenation algorithm
- The original TeX hyphenation algorithm (1977)
- The current TeX hyphenation algorithm (1983)
- Creating the patterns (patgen)
- Using the patterns (hyphenation)

HTML and the soft hyphen

The Port to Javascript
Server side or Client side?
- How it works
- Differences and Improvements
- Back to the Future
My Decision

- server side solutions already existed: php, perl, java, python
- I believe that hyphenation has to be done in the client
- Javascript is a very interesting language
- the acceptance of Javascript is growing
- Firefox 2 didn’t support &shy;
- I like bookmarklets

- hyphenator.js: client-side hyphenation

- it’s proofing to be a good decision:
  - other – webkit based – programs are using hyphenator
  - it’s easy to use
  - there’s a big effort on making javascript faster
How it works

1. register all elements that need hyphenation
2. if the language is not set, ask for it
3. download the patterns, if not already done
4. split the paragraphs in words (and URLs)
5. process each word, put &shy; at every valid breakpoint
6. The browser will re-render the text automatically, taking in account the soft hyphens.

▶ execution is fast
▶ downloading the script and the patterns takes time

script: 25 KB, en: 25 KB, pl: 37 KB, de: 74 KB
How it works

1. register all elements that need hyphenation
2. if the language is not set, ask for it
3. download the patterns, if not already done
4. split the paragraphs in words (and URLs)
5. process each word, put &shy; at every valid breakpoint
6. The browser will re-render the text automatically, taking in account the soft hyphens.

▶ execution is fast
▶ downloading the script and the patterns takes time

script: 25 KB, en: 25 KB, pl: 37 KB, de: 74 KB
How it works

1. register all elements that need hyphenation
2. if the language is not set, ask for it
3. download the patterns, if not already done
4. split the paragraphs in words (and URLs)
5. process each word, put &shy; at every valid breakpoint
6. The browser will re-render the text automatically, taking in account the soft hyphens.

▶ execution is fast
▶ downloading the script and the patterns takes time

script: 25 KB, en: 25 KB, pl: 37 KB, de: 74 KB
How it works

1. register all elements that need hyphenation
2. if the language is not set, ask for it
3. download the patterns, if not already done
4. split the paragraphs in words (and URLs)
5. process each word, put &shy; at every valid breakpoint
6. The browser will re-render the text automatically, taking in account the soft hyphens.

► execution is fast
► downloading the script and the patterns takes time

script: 25 KB, en: 25 KB, pl: 37 KB, de: 74 KB
How it works

1. register all elements that need hyphenation
2. if the language is not set, ask for it
3. download the patterns, if not already done
4. split the paragraphs in words (and URLs)
5. process each word, put &shy; at every valid breakpoint
6. The browser will re-render the text automatically, taking in account the soft hyphens.

▶ execution is fast
▶ downloading the script and the patterns takes time
script: 25 KB, en: 25 KB, pl: 37 KB, de: 74 KB
How it works

1. register all elements that need hyphenation
2. if the language is not set, ask for it
3. download the patterns, if not already done
4. split the paragraphs in words (and URLs)
5. process each word, put &shy; at every valid breakpoint
6. The browser will re-render the text automatically, taking in account the soft hyphens.

▶ execution is fast
▶ downloading the script and the patterns takes time
script: 25 KB, en: 25 KB, pl: 37 KB, de: 74 KB
How it works

1. register all elements that need hyphenation
2. if the language is not set, ask for it
3. download the patterns, if not already done
4. split the paragraphs in words (and URLs)
5. process each word, put &shy; at every valid breakpoint
6. The browser will re-render the text automatically, taking in account the soft hyphens.

▶ execution is fast
▶ downloading the script and the patterns takes time

script: 25 KB, en: 25 KB, pl: 37 KB, de: 74 KB
How it works

1. register all elements that need hyphenation
2. if the language is not set, ask for it
3. download the patterns, if not already done
4. split the paragraphs in words (and URLs)
5. process each word, put &shy; at every valid breakpoint
6. The browser will re-render the text automatically, taking in account the soft hyphens.

▶ execution is fast
▶ downloading the script and the patterns takes time

script: 25 KB, en: 25 KB, pl: 37 KB, de: 74 KB
How it works

1. register all elements that need hyphenation
2. if the language is not set, ask for it
3. download the patterns, if not already done
4. split the paragraphs in words (and URLs)
5. process each word, put &shy; at every valid breakpoint
6. The browser will re-render the text automatically, taking in account the soft hyphens.

▶ execution is fast
▶ downloading the script and the patterns takes time

script: 25 KB, en: 25 KB, pl: 37 KB, de: 74 KB
Main Differences

- don’t care about space in RAM, care about program size
- no Trie (retrieval tree)
  - no special data structures in Javascript
  - using a trie is faster in execution (10ms)
  - but: building the tree from the patterns takes time
  - but: for a tree extra code is used (uses bandwith)
  - but: transferring the hardcoded trie is no solution, either (overhead: 50%)
- using a hash table (Javascript: object) instead
- UTF-8 (Thanks to Arthur and Mojca)
Main Differences

- don’t care about space in RAM, care about program size
- no Trie (retrieval tree)
  - no special data structures in Javascript
  - using a trie is faster in execution (10ms)
  - but: building the tree from the patterns takes time
  - but: for a tree extra code is used (uses bandwith)
  - but: transferring the hardcoded trie is no solution, either (overhead: 50%)
- using a hash table (Javascript: object) instead
- UTF-8 (Thanks to Arthur and Mojca)
Main Differences

- don’t care about space in RAM, care about program size
- no Trie (retrieval tree)
  - no special data structures in Javascript
  - using a trie is faster in execution (10ms)
  - but: building the tree from the patterns takes time
  - but: for a tree extra code is used (uses bandwith)
  - but: transferring the hardcoded trie is no solution, either (overhead: 50%)
- using a hash table (Javascript: object) instead
- UTF-8 (Thanks to Arthur and Mojca)
Improvements I

Packing the patterns (helper: compressor):

- size of the pattern file does matter
- no whitespace (> 12% saved!)
  
a1 ą1 e1 ę1 i1 o1 ó1 u1 y1 _a1 _b8 _c8 _ć8 _d8
  
  2:'a1ą1e1ę1i1o1ó1u1y1',
  
  3:'_a1_b8_c8_ć8_d8_e1_f8

- http-requests take time
- merge the script and the necessary patterns (usually just one) in one file
- saves 2 requests per pattern-file
Improvements I

Packing the patterns (helper: compressor):

- size of the pattern file does matter
- no whitespace (> 12% saved!)

a1 ą1 e1 ę1 i1 o1 ó1 u1 y1 _a1 _b8 _c8 _ć8 _d8

2:’a1ą1e1ę1i1o1ó1u1y1’,
3:’_a1_b8_c8_ć8_d8_e1_f8

- http-requests take time
- merge the script and the necessary patterns (usually just one) in one file
- saves 2 requests per pattern-file
Improvements I

Packing the patterns (helper: compressor):

- size of the pattern file does matter
- no whitespace (> 12% saved!)

```plaintext
a1 a1 e1 e1 i1 i1 o1 o1 u1 y1 _a1 _b8 _c8 _ć8 _d8
2:'a1a1e1e1i1i1o1o1u1y1',
3:'_a1_b8_c8_ć8_d8_e1_f8
```

Merging script and patterns in one file (helper: merge+pack)

- http-requests take time
- merge the script and the necessary patterns (usually just one) in one file
- saves 2 requests per pattern-file
Improvements I

Packing the patterns (helper: compressor):

- size of the pattern file does matter
- no whitespace (> 12% saved!)
  
  \[a1 \, â1 \, e1 \, ė1 \, i1 \, o1 \, ó1 \, u1 \, y1 \, _a1 \, _b8 \, _c8 \, _ć8 \, _d8\]
  
  2: ’a1â1e1ë1i1o1ó1u1y1’,
  3: ’_a1_b8_c8_ć8_d8_e1_f8’

Merging script and patterns in one file (helper: merge+pack):

- http-requests take time
- merge the script and the necessary patterns (usually just one) in one file
- saves 2 requests per pattern-file
Improvements II

Using reduced pattern sets for static sites (helper: reducePatternSet)

- most patterns are not used
- if the text will not change, use a precomputed subset
- savings vary

- only take in account breakpoints of composite words: Zeilen-ende instead of Zei-len-en-de
- patterns are now 37 KB instead of 74 KB
- 265683 good, 22837 bad, 995752 missed
- 21.06 %, 1.81 %, 78.94 %
- or use different settings for patgen!
Improvements II

Using reduced pattern sets for static sites (helper: reducePatternSet)

- most patterns are not used
- if the text will not change, use a precomputed subset
- savings vary

- only take in account breakpoints of composite words: Zeilen-ende instead of Zei-len-en-de
- patterns are now 37 KB instead of 74 KB
- 265683 good, 22837 bad, 995752 missed
- 21.06 %, 1.81 %, 78.94 %

- or use different settings for patgen!
Improvements II

Using reduced pattern sets for static sites (helper: reducePatternSet)

- most patterns are not used
- if the text will not change, use a precomputed subset
- savings vary

(Recompute the patterns)

- only take in account breakpoints of composite words: Zeilen-ende instead of Zei-len-en-de
- patterns are now 37 KB instead of 74 KB
- 265683 good, 22837 bad, 995752 missed
- 21.06 %, 1.81 %, 78.94 %
- or use different settings for patgen!
Improvements II

Using reduced pattern sets for static sites (helper: reducePatternSet)

- most patterns are not used
- if the text will not change, use a precomputed subset
- savings vary

(Recompute the patterns)

- only take in account breakpoints of composite words:
  Zeilen-ende instead of Zei-len-en-de
  de patterns are now 37 KB instead of 74 KB
  265683 good, 22837 bad, 995752 missed
  21.06 %, 1.81 %, 78.94 %

- or use different settings for patgen!
Improvements II

Using reduced pattern sets for static sites (helper: reducePatternSet)

- most patterns are not used
- if the text will not change, use a precomputed subset
- savings vary

(Recompute the patterns)

- only take in account breakpoints of composite words: Zeilen-ende instead of Zei-len-en-de
- patterns are now 37 KB instead of 74 KB
- 265683 good, 22837 bad, 995752 missed
- 21.06 %, 1.81 %, 78.94 %

- or use different settings for patgen!
Hyphenator.js Problems and Oddities

- Problems upon copy/paste of hyphenated text
- Problems with loaded fonts (@font-face)
- Patterns very different in size
- Some rare misplaced hyphenation breaks may happen
Hyphenator.js Problems and Oddities

- Problems upon copy/paste of hyphenated text
- Problems with loaded fonts (@font-face)
- Patterns very different in size
- Some rare misplaced hyphenation breaks may happen
Hyphenation for HTML

Mathias Nater
mathias@mnn.ch
http://mnn.ch/

Motivation
layout w/o hyphenation
layout with hyphenation

The \TeX\ hyphenation algorithm

- The original \TeX\ hyphenation algorithm (1977)
- The current \TeX\ hyphenation algorithm (1983)
- Creating the patterns (patgen)
- Using the patterns (hyphenation)

HTML and the soft hyphen

The Port to Javascript

Server side or Client side?
How it works
Differences and Improvements
Back to the Future

Hyphenator.js Problems and Oddities

- Problems upon copy/paste of hyphenated text
- Problems with loaded fonts (@font-face)
- Patterns very different in size
- some rare misplaced hyphenation breaks may happen
Hyphenator.js Problems and Oddities

- Problems upon copy/paste of hyphenated text
- Problems with loaded fonts (@font-face)
- Patterns very different in size
- Some rare misplaced hyphenation breaks may happen
What the future shall/may bring

- CSS3: browsers do hyphenation (w/o hyphenator.js)
- TUG: maintained hyphenation patterns (beware of size!)
- Wish: better typography in web sites.
- Me: Try to rewrite PatGen for UTF-8 Support
What the future shall/may bring

- CSS3: browsers do hyphenation (w/o hyphenator.js)
- TUG: maintained hyphenation patterns (beware of size!)
- Wish: better typography in web sites.
- Me: Try to rewrite PatGen for UTF-8 Support
What the future shall/may bring

- CSS3: browsers do hyphenation (w/o hyphenator.js)
- TUG: maintained hyphenation patterns (beware of size!)
- Wish: better typography in web sites.
- Me: Try to rewrite PatGen for UTF-8 Support
What the future shall/may bring

- CSS3: browsers do hyphenation (w/o hyphenator.js)
- TUG: maintained hyphenation patterns (beware of size!)
- Wish: better typography in web sites.
- Me: Try to rewrite PatGen for UTF-8 Support
In chapter 9.3.3. the HTML 4.01 Specification tells us the following about hyphenation in HTML:

[...] The soft hyphen tells the user agent where a line break can occur.

[...] If a line is broken at a soft hyphen, a hyphen character must be displayed at the end of the first line. If a line is not broken at a soft hyphen, the user agent must not display a hyphen character. [...] The soft hyphen is represented by the character entity reference &shy; (&#173; or &#xAD;);
applying the patterns – example 2

hyphenation
  . hyphenation .
    2i o
      1n a
        o2n
    h e2n
      n2a t
        1t i o
  h e n a4
    h y3p h
      h e n5a t
  ------------
    . h y3p h e2n5a4t2i o2n .
hyphenation
For Further Reading I

David Antoš (2001):
PatLib, Pattern Manipulating Library – Master Thesis
Masaryk University Brno, Faculty of Informatics

Donald E. Knuth (1999):
Digital Typography.
Stanford, California: Center for the Study of Language
and Information
ISBN 1-57586-010-4

Franklin Mark Liang (1983):
Word Hyphenation by Computer. PhD thesis
Department of Computer Science, Stanford University:
Stanford, CA 94305.
For Further Reading II

Christine Römer, Herbert Voss (2008):
Deutsche Silbentrennmuster – aus linguistischer und TeXnischer Sicht. PDF, Jena 06. 03. 2008

Raggett Dave, Le Hors Arnaud, Jacobs Ian (1999):
http://www.w3.org/TR/html401/