multilingual.js:
다국어 웹 타이포그래피를 위한 섞어쓰기 라이브러리

강이룬
Math Practice, 미국

소원영
매사추세츠 공과 대학교, 미국

주제어: 섞어쓰기, 웹 브라우저, 오픈 소스,
마이크로 타이포그래피, 타이포그래피

투고: 2016년 4월 12일
심사: 2016년 5월 2일–6일
게재 확정: 2016년 5월 16일

---

multilingual.js:
A javascript library for multilingual typesetting

Kang E Roon
Math Practice, USA

So Wonyoung
Massachusetts Institute of Technology, USA

Keyword: Multilingual Typesetting, Web browser,
Open Source, Microtypography, Typography

Received: 12 April 2016
Reviewed: 2–6 May 2016
Accepted: 16 May 2016

초록

복수의 언어로 쓰인 글을 조판할 때는 두 가지 서로 다른 언어 체계를 다루는 각각의 글자체가 문단에서 시각적으로 어떻게 어우러지는가가 중요한 문제가 된다. 다양한 글자체의 속성을 다루는 사용자 인터페이스(UI)의 한계와 인쇄물 디자인 소프트웨어에서 이를 개선하려는 사례들을 살피고, 그와 유사한 기능을 웹 브라우저에서 적용할 수 있는 방법을 제안한다. multilingual.js는 자바스크립트 라이브러리로 제4회 ‹타이포잔치›의 웹사이트를 제작하는 과정에서 개발했으나, 동아시아권의 디자이너 모두가 손쉽게 사용할 수 있도록 http://multilingualjs.github.io에서 오픈 소스로 배포한다.

Abstract

An important issue when typesetting a multilingual article is the visual harmony of multiple typefaces displaying different language systems in the same paragraph. This paper looks at the limitations of user interfaces when dealing with properties of diverse typefaces; provides a survey of existing efforts to overcome such limitations within publishing software; and suggests a method to apply similar functionalities in web browsers. multilingual.js is an open-source JavaScript library first developed during the process of creating the website for 4th ‹Typojanchi›. The library is distributed at http://multilingualjs.github.io, so that it can easily be used by designers working with East Asian languages.

배경

한글로 쓰인 문장이라도 문장의 뜻을 올바르게 전달하기 위해 다른 언어와 함께 표시해야 할 때가 있다. 많은 한국어 어휘가 한자어로 이루어져 있어, 동음이의어를 뚜렷이 구분하기 위해 한자를 병기하는 경우도 있다. 또한 학문적 글쓰기나 기술적 글쓰기의 경우 그 뜻을 명확히 하기 위해 영문의 병기가 불가피한 경우도 있다. 그리고 현재의 한글 맞춤법은 아라비아 숫자나 문장 부호 등 다른 언어 체계에서 오래전에 빌려온 요소들을 다수 포함하고 있다. 이렇게 복수의 언어로 쓰인 글을 조판할 때는 복수의 언어 체계를 다루는 각각의 글자체가 한 문단에서 어떻게 어우러지는가가 중요한 문제가 된다. 이 어울림은 역사적일 수도 있고 시각적일 수도 있다. 다만 고유한 역사와 체계를 가지고 발전해온 서로 다른 글자체를 단일한 서사로 엮어내는 데에 있어 기술적인 관점에서 가장 중요한 일은 이 어울림을 효과적으로 제어하는 일이다.

섞어쓰기의 현재

일상생활에서 한글과 영문 그리고 한문을 섞어 쓸 일이 많기 때문에, 대부분의 한글 글자체는 한글과 숫자, 영문, 그리고 많은 경우 한문도 표시할 수 있는 글리프를 포함하고 있다. 따라서 한글 글자체 한 가지만으로도 비교적 정돈된 섞어쓰기를 할 수 있다. 다만 한글 글자체에 포함되어 있지 않은 일본어 등을 섞어서 조판해야 하는 경우, 또는 영문에만 선택적으로 다른 글자체를 적용하고 싶은 경우 등에는 한 문단에 다수의 글자체를 적용해야 하는 상황에 직면하게 되는데, 여기서는 두 가지 현실적인 문제가 따른다.

Background

Korean sentences written in Hangeul sometimes require notation in additional languages to convey the accurate meaning. Because a large part of Korean vocabulary consists of Sino-Korean words, there are times when Chinese characters are written alongside to distinguish homonyms. Moreover, it is sometimes inevitable to use English alongside Korean in order to clarify the meaning in academic or technical writing. The current Korean orthography also contains a number of elements adopted long ago from other languages, such as Arabic numerals and punctuations. An important issue when typesetting a multilingual article is the harmony of two or more typefaces displaying different language systems within a single paragraph. This harmony can be defined with regards to historical context or visual elements. In any case, in a technical point of view, the most important thing in weaving different typefaces each of which has evolved with its own unique history and system is to effectively control the harmony.

The Current State of Multilingual Typesetting

Because Hangeul, Roman Alphabets and Chinese characters are often used together in our daily lives, most Korean fonts include glyphs of Hangeul, numerals, Roman Alphabets and often Chinese characters too. Therefore, a relatively organized multilingual typesetting is possible simply by using one Hangeul typeface. However, sometimes one needs to use multiple typefaces in a single paragraph, such as when typesetting characters not included in Hangeul fonts such as Japanese characters or

첫 번째는 글자체의 다양한 속성 때문에 발생하는 문단의 비일관성이다. 글자체의 기본적인 형태뿐 아니라 글자의 폭이나 두께, 간격, 글줄의 높이 등 다수의 글자체를 한 문단에 적용할 때 고려해야 할 변수는 많다. 하지만 이것이 서로 미려하게 조정된 글자체를 찾는 일은 매우 어렵다. 서로 다른 디자이너가 서로 다른 철학과 기준으로 개발한 글자체를 임의의 원칙으로 묶어내야 하기 때문이다.

두 번째는 이런 섞어쓰기를 지원하는 효율적인 사용자 인터페이스(UI)의 부재이다. 통상적인 그래픽 프로그램의 사용자 인터페이스에서 디자이너는 입력한 텍스트의 영역을 클릭 또는 드래그해 '선택'하고 시스템의 글자체 목록에서 특정 글자체를 지정해 선택 영역에 해당 글자체를 '적용'한다. 이런 인터페이스 모델은 한 영역에 한 가지 글자체를 적용하기 위해 설계된 것으로, 두 가지 이상의 글자체를 한 문장에 적용해야 할 때는 무리가 따른다. 따라서 이런 사용자 인터페이스를 통해 서로 다른 글자체를 사용하는 섞어쓰기를 시도할 때에는, 글리프가 풍부한 특정 언어의 글자체를 먼저 적용하고 글리프가 제한적인 또 다른 글자체를 중복 적용하는 등의 편법에 의존할 수밖에 없게 된다. 이런 편법은 소프트웨어 제작사가 공식적으로 제공한 작업 방식이 아니므로 소프트웨어 업데이트를 통해 그 가능성이 언제든지 사라져 버릴 수도 있다는 점 이외에도 몇 가지 중대한 디자인적 결함이 있다. 이 논문에서는 이런 결함을 보완하는 기존의 UI 사례들을 살피고, 이를 참조해 웹 브라우저에서 제어 가능한 섞어쓰기를 구현하는 방법을 제안하고자 한다.

when applying a separate font to Roman alphabets only. Such situations raise two practical issues. The first issue is the inconsistency of a paragraph caused by the varying attributes of typefaces. While there are a lot of variables to consider when applying multiple typefaces in a single paragraph, such as the basic form of the typeface, the width, weight, spacing, and height, it is very difficult to find typefaces within which these aspects elegantly relate. These typefaces are grouped arbitrarily, regardless of their varying philosophy and standards.

The second issue is the absence of an effective user interface (UI) that supports this kind of multilingual typesetting. In a typical graphic user interface, designers 'select' the area of entered text by clicking or dragging then 'apply' a typeface onto the selected area by designating a specific font from the font list of the system. This interface model is designated to apply one typeface for each area, hence is inconvenient when applying more than two typefaces in a single sentence. Therefore, in order to attempt multilingual typesetting of different typefaces using such a user interface, one needs to rely on workarounds; first apply a typeface of a particular language that has wide variety of glyphs, then apply another typeface with limited glyphs. However, such a workaround is not only a method officially supported by the software manufacturer—contains the risk of becoming obsolete after any software update—but also involves some serious design flaws. The following part of this paper will look at existing cases of UI complementing these flaws, and then propose a method for mixed writing in a way that is controllable within the web browser.

웹 브라우저에서의 섞어쓰기

소프트웨어의 사용자 인터페이스를 통해서 글자체의 사용을 제어하는 출판물 디자인의 작업 방식과는 달리, 웹 브라우저에서의 타이포그래피는 문서의 내용을 표현하는 HTML과 그 내용의 표현 방식을 지정하는 CSS를 통해 관리한다. 이것은 해당 문자열을 '선택'하고 글자체를 '적용'하는 과정을 일련의 코드로 작성하는 것인데, 그것의 일반적인 형태는 아래와 같다.

```
<html>
  <body>
    <p>CCL은 전세계적(worldwide)인 라이선스 시스템입니다.</p>
  </body>
</html>
```

```
/* apply Helvetica to the entire body text */
p {
  font-family: Helvetica, Arial, sans-serif;
}
```

온라인에서의 문서는 인쇄물처럼 완결된 형태로 물리적으로 구현되지 않는다. HTML과 CSS를 통해 기술된 타이포그래피는 클라이언트 컴퓨터에 설치되어있는 글자체를 이용해서만 재현될 수 있고, 따라서 지정한 글자체가 설치되어 있지 않을 경우를 대비해 몇 가지 대비책을 설정하는 것이 일반적이다. 위의 경우에는 `font-family:` 이후에 선언된 글자체의 목록(글자체 스택)에

Multilingual Typesetting in Web Browsers

Unlike publication design, where the usage of typefaces is solely controlled through software's user interface, web typography is managed by HTML, which expresses the content of the document, and CSS, which designates the expression method of the content. This is done by writing a series of code for the process of 'selecting' a particular string of characters and 'applying' a typeface. Below is a typical example of such code:

```
<html>
  <body>
    <p>CCL은 전세계적(worldwide)인 라이선스 시스템입니다.</p>
  </body>
</html>
```

```
/* apply Helvetica to the entire body text */
p {
  font-family: Helvetica, Arial, sans-serif;
}
```

Online documents are not physically materialized in a complete form like printed matter. Typography described with HTML and CSS is reproduced only through the fonts that are installed in the viewer's computer. Thus it is common to set some fallbacks in case the selected typeface is not installed. In the example above, the list of fonts (CSS font stack)

따라, 사용자의 컴퓨터에서 OS X의 기본 글자체 중 하나인 헬베티카를 가장 우선으로 찾고, 이것이 없을 경우에는 윈도우즈의 기본 글자체 중 하나인 에리얼을, 그리고 이것마저 없을 경우에는 브라우저의 기본 설정에 의해 산세리프로 설정된 글자체를 찾아서 해당 HTML 문서를 표시하게 된다.

이런 글자체의 적용 방식은 출판물 저작 소프트웨어에서 한 선택 영역에 서로 다른 글리프를 가진 글자체를 중복 적용하는 편법처럼, 한 영역에 다수의 글자체를 적용할 수 있게 한다. 위의 예시에서 최우선 순위인 헬베티카나 에리얼 모두 한글 글리프를 포함하고 있지 않으므로, 문서를 표시하는 OS에 따라서 설치되어 있는 글자체 중 산세리프로 분류된 한글 글자체—OS X의 경우에는 애플산돌네오고딕, 윈도우즈의 경우에는 맑은고딕 혹은 굴림—을 이용해 한글을 표시하게 된다. 따라서 결과적으로 위의 예시 문서는 대부분의 컴퓨터에서 헬베티카/애플산돌네오고딕(OS X) 또는 에리얼/맑은고딕 혹은 굴림체(윈도우즈)의 세트로 표시된다.

온라인에서의 통상적인 섞어쓰기는 이렇게 CSS 글자체 스택을 활용해 이루어진다. 일례로, 강이룬과 소원영이 함께 개발한 제4회 ‹타이포잔치› 웹사이트에서 사용한 CSS 글자체 스택은 아래와 같다. 이에 따르면 영문은 파운더스 그로테스크 텍스트[1]로, 한글은 노토 산스[2]로 표시된다.

```
font-family: 'FoundersGroteskTextWeb', 'NotoSansCJKkr-Regular-2350',
Arial, sans-serif;
```

1 뉴질랜드 소재 클림 타입 파운드리에서 제목용으로 제작한 클래식 그로테스크 리바이벌 글자체인 파운더스 그로테스크의 본문용 글자체이다.

2 구글과 어도비가 커미션하고 전 세계 여러 글자체 회사와 공동 제작한 오픈 소스 글자체로, 한글은 산돌커뮤니케이션에서 제작했다. 구글에서는 노토 산스 글자체 가족의 일부인 노토 산스 CJK 코리안으로, 어도비에서는 소스 산스의 일부인 소스 한 산스(한국명 본고딕)로 공개했다.

is declared after `font-family`. The instruction here is to first look for Helvetica, one of the OS X default typefaces; if Helvetica is absent, look for Arial, one of the Windows default typefaces; if both are absent, look for the sans-serif typeface set as default in the browser settings and use it to display the HTML document.

Such a method for applying typefaces allows the application of multiple typefaces in one area like the aforementioned workaround in publication design softwares where typefaces with different glyphs are applied multiple times in one selected area. In the above example, since neither Helvetica nor Arial include Hangeul glyphs, Hangeul is displayed using one of the sans-serif Hangeul typefaces installed in the OS where the document is displayed: Apple Sandol Neo Gothic in the case of OS X, and Malgun Gothic or Gulim in the case of Windows. Therefore, the above example document is most likely be represented by the set of Helvetica/Apple Sandol Neo Gothic (OS X) or Arial/Malgun Gothic or Gulim (Windows) depending on the viewer's computer.

The usual online multilingual typesetting is done this way, by utilizing CSS font stack. For example, the ‹Typojanchi› official website, developed by Kang E Roon and So Wonyoung, used the following CSS font stack. According to this, Roman Alphabets are displayed in Founders Grotesk Text[1], and Hangeul in Noto Sans[2].

```
font-family: 'FoundersGroteskTextWeb', 'NotoSansCJKkr-Regular-2350',
Arial, sans-serif;
```

1 The Founders Grotesk Text is the body text typeface of Founders Grotesk, a revival font of the Classic Grotesque made for titles by Klim Type Foundry in New Zealand.

2 Noto Sans is an open source font commissioned by Google and Adobe and co-produced with font companies worldwide. Sandol Communication produced the Hangeul version. Google released the font as Noto Sans CJK Korean, within the Noto Sans font family; Adobe released it as Source Han Sans (Bon Gothic in Korean), a member of the Source Sans family.

강제적 섞어쓰기의 문제

온라인에서 CSS 글자체 스택을 활용하거나, 디자인 소프트웨어에서 글자체를 중복 지정하는 방식은 글자체 자체가 가지고 있는 글리프의 갯수가 섞어쓰기의 결과를 좌우하기 때문에 사용자가 제어할 수 없다는 점에서 강제적이다.
이 단점을 극단적으로 재현하기 위해서, 한글은 노토 산스로, 영문은 고정폭 글자체인 소스 코드 프로[3]로 설정된 경우를 상정할 수 있다.

```
font-family: SourceCodePro, NotoSans, sans-serif;
```

소스 코드 프로는 보통의 영문 글자체처럼 영문 알파벳과 숫자, 문장 부호, 각종 심볼 및 빈칸(띄어쓰기) 등의 글리프로 이루어져 있다. 따라서 글자체 스택의 첫 번째 글자체부터 글리프를 검색해 무조건 표시하는 CSS의 원칙에 따라, 소스 코드 프로에 포함된 띄어쓰기나 각종 문장 부호까지도 영문 고정폭으로 표시되면서 문단의 가독성을 해치게 된다.

3 어도비의 소스 산스의 패밀리의 일부로 개발 코딩용 오픈 소스 고정폭 글자체이다.

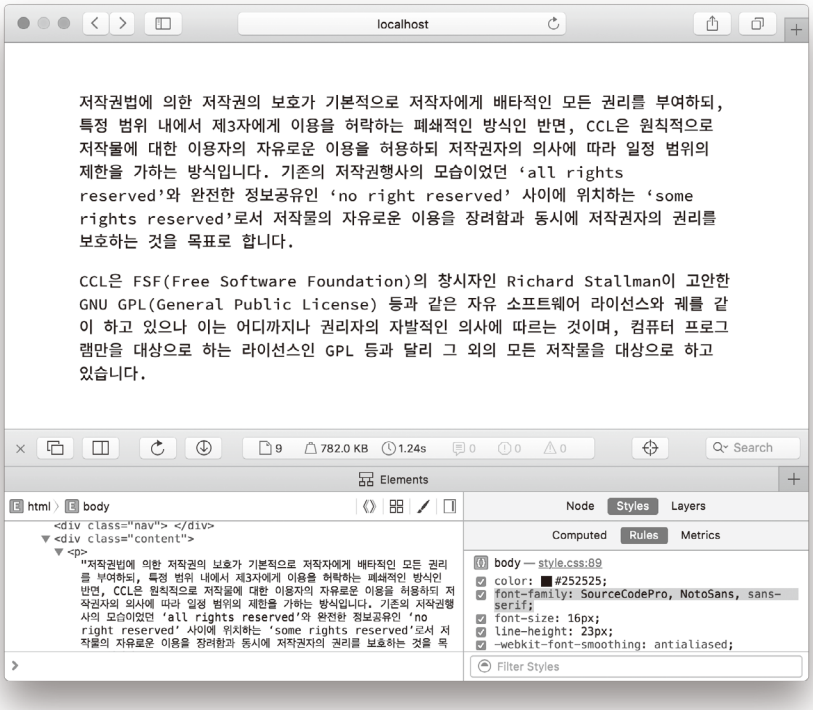
The Problem of a Forced Multilingual Typesetting

The use of CSS font stack or assigning typefaces multiple times to a paragraph using a software are forced methods, in that they allow the number of glyphs in a font to determine the result,instead of the user having control. For an extreme case of this type of disadvantage, one can imagine a situation where Hangeul is set in Noto Sans and Roman Alphabets are set in Source Code Pro[3], a fixed-width typeface.

```
font-family: SourceCodePro, NotoSans, sans-serif;
```

Source Code Pro is composed of glyphs including Roman Alphabets, numerals, punctuations, various symbols and blank spaces as English based typefaces usually are. Thus, due to CSS rules that show glyphs in whatever first typeface comes first in the font stack, all elements available in Source Code Pro will be displayed in Source Code Pro, including spaces and punctuations. This will harm the readability of the paragraph as seen in [1].
Even if one is not mixing a fixed-width typeface with a proportional-width Hangeul typeface, the fact that the typeface declared later is dependent on the attributes of the typeface declared earlier poses a structural problem. This problem relates not only with letter spacing but also with diverse attributes such as the font size, line length and more. It is possible to improve this problem by controlling the attributes

3 Source Code Pro is an open source set-width font for coding, which was developed as a part of Adobe's Source Sans family.



[1] 단어 사이의 간격과 문장 부호들 모두 고정폭 글자체인 소스 코드 프로로 표시된다.

Word space and punctuations are displayed in Source Code Pro, a fixed width typeface.

[1]에서처럼 고정폭 글자체와 가변폭인 국문을 섞어 쓰는 극단적인 경우가 아니더라도, 이후에 선언된 글자체가 이전에 선언된 글자체의 속성에 종속된다는 것은 구조적인 문제임에 틀림없다. 이 문제는 글자의 간격뿐 아니라 글자체의 크기, 글줄 등 글자체의 다양한 속성들과 관계하는데, 이것은 강제적 섞어쓰기를 위한 별도의 사용자 인터페이스를 개발해 이 속성들을 보다 섬세하게 제어함으로써 개선이 가능하다.

발단: 제4회 ‹타이포잔치› 웹사이트

2015년, 강이룬과 소원영은 제4회 ‹타이포잔치› 웹사이트를 디자인하고 개발하면서 웹 브라우저에서 강제적 섞어쓰기의 문제를 접하게 되었다.[2] 참여작가들의 위치와 전시장의 위치를 그려내는 지도와 목록으로 구성된 제4회 ‹타이포잔치› 웹사이트를 위해 우리는 가독성이 높고 용량이 가벼우면서도 우리가 제공하는 기계적인 인터페이스의 성격을 드러낼 수 있는 글자체를 찾고자 했다. 최종적으로 영문은 파운더스 그로테스크 텍스트, 한글은 노토 산스를 선택하게 되었는데, 문제는 이 두 글자체가 한 문단에서 어우러지면서 글자 크기의 차이가 심하게 발생한다는 점이었다.

많은 수의 외국 작가가 참여한 국제 타이포그래피 비엔날레인 만큼 영문과 한글을 함께 표기해야 하는 일은 빈번했고 제4회 ‹타이포잔치› 웹사이트를 위해서는 CSS 글자체 스택보다 섬세한 섞어쓰기 관리 도구가 필요했다.

with granularity by developing a separate user interface for forced multilingual typesetting.

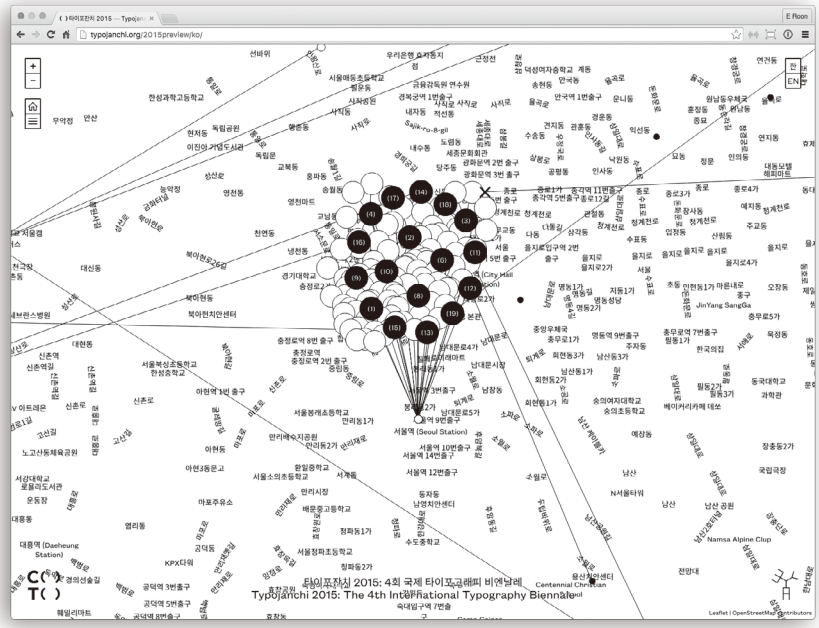
Prologue: The 4th ‹Typojanchi› Website

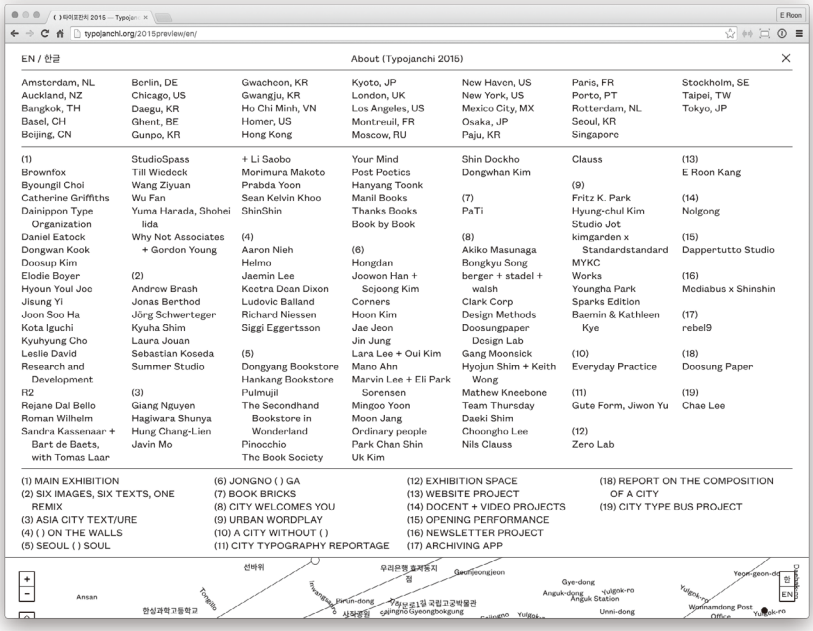
We came across the problem of forced multilingual typesetting within a web browser while designing and developing the ‹Typojanchi›. We looked for typefaces with good readability, light weight, and which reveals the characteristics of the technical interface that we provided in the website, consisting of lists and maps that draws the locations of the exhibition and participating artists. We decided to work with Founders Grotesk Text for Roman Alphabets and Noto Sans for Hangeul.

The problem was that there was a severe difference in the font sizes when the two were used together in a single paragraph. As ‹Typojanchi› was an international typography biennale with many foreign participating artists, the task of writing Roman Alphabets and Korean together occurred frequently. Therefore, we needed a management tool for a more delicate multilingual typesetting than the CSS font stack provided.

Case 1: Precedent of a Controlled Multilingual Typesetting

Adobe InDesign has improved the situation of a forced multilingual typesetting in publication design by introducing the function called Composite Font. Composite Font is a method of setting size, baseline, horizontal/vertical ratio, etc, independently for each element of Hangeul, Roman alphabet, numerals, symbols and producing a virtual typeface that





[2] 제4회 ‹타이포잔치› 웹사이트. (typojanchi.org/2015)
The 4th ‹Typojanchi› website. (typojanchi.org/2015)

multilingual 다국어

[3] 파운더스 그로테스크와 노토 산스의 글자 크기 비교.
Font size comparison between Founders Grotesk and Noto Sans.

참고 사례 1: 관리형 섞어쓰기의 선례

어도비 인디자인은 합성 글꼴이라는 기능을 통해 인쇄물을 디자인하는 데 있어서 강제적 섞어쓰기의 문제를 개선해왔다. 합성 글꼴은 한글, 로마자, 번호, 기호 등 각종 요소마다 각각 크기, 기준선, 가로와 세로 비율 등을 독립적으로 설정하고 이 모든 설정값을 가진 가상의 글자체를 생성하는 방식이다. 이렇게 생성된 가상의 글자체의 속성은 언제든지 합성 글꼴 편집기를 통해 다시 설정할 수 있고, 이렇게 생성된 가상 글자체는 여러 글자체를 섞어 만든 한 개의 글자체이므로 기존의 '선택' 및 '적용' 인터페이스를 통해 쉽게 사용할 수 있다.

이렇게 섞어쓰기를 관리하는 방식은 섞어쓰기의 첫 번째 문제인 다양한 글자체의 속성 때문에 발생하는 문단의 비일관성을 획기적으로 제어할 수 있도록 한다.

참고 사례 2: 웹 브라우저에서의 마이크로 타이포그래피

웹에서의 타이포그래피를 향상시키기 위한 잘 알려진 오픈 소스 자바스크립트 라이브러리[4] 중 하나인 lettering.js는 제이쿼리[5] 프레임 워크의 플러그인 형태로 동작하며, HTML 문서에서 낱자 단위의 스타일링을 제어할 수 있도록 도와준다. 마우스로 한 글자씩 선택해서 속성을 각각 적용할 수 있는 일반적인 그래픽 사용자 인터페이스 환경과는 달리, 온라인 문서에서는 HTML의 요소들을 CSS에서 각각 스타일링 하게 되는데, 이런 구조에서 낱자 단위의 스타일을 제어하려면 매 글자마다 고유한 클래스를 지정해주어야 하므로 불필요하게 복잡한 HTML 문서를 작성할 수밖에 없다. lettering.js는 이 과정을 자동화해,

4 자주 사용하는 기능을 모듈화해 쓰기 편하게 만들어둔 것으로 자바스크립트로 작성되어 있고, 그 소스 코드가 누구에게나 공개되어 있는 형태를 뜻한다.

5 자바스크립트를 편하게 개발할 수 있도록 지원하는 라이브러리로, 다양한 플러그인을 통해 기능의 확장이 가능하다.

has individual settings all of these properties. The attribute of the virtual typeface that was generated can always be managed through Composite Font editor, and since this virtual typeface is a single typeface that combines various typefaces, it can be easily used through the existing 'selecting' and 'applying' interface.

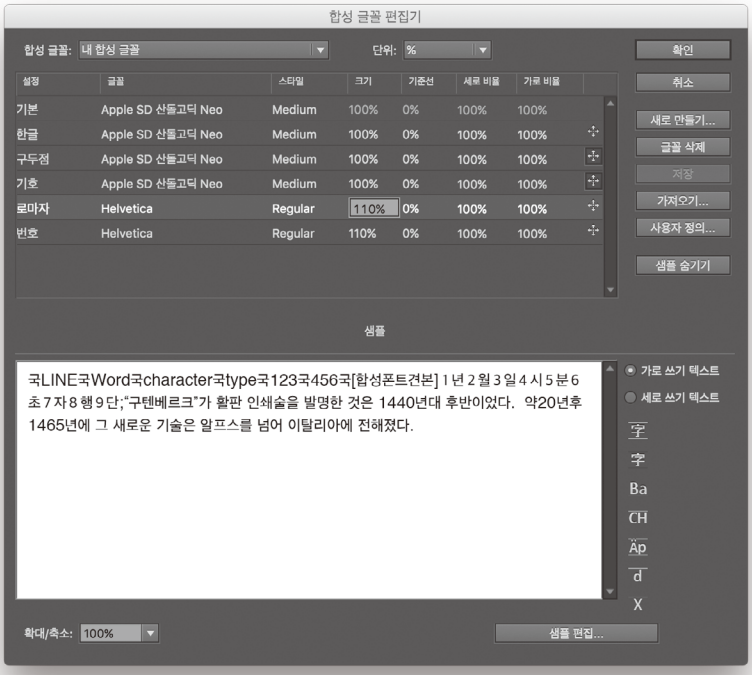
Such method of managing multilingual typesetting offers significant control over the paragraph inconsistency, caused by the conflicting attributes of various typefaces, which is the initial problem of multilingual typesetting.

Case 2: Microtypography in Web Browsers

lettering.js is a well-known open-source JavaScript library[4] for improved typography on the web. It exists in the form of a plug-in for the jQuery[5] framework, and helps to control the styling of individual characters in an HTML document. Unlike the typical graphic user interface environment where one can apply particular attributes by clicking each letter with the mouse, elements in HTML documents are styled through CSS, and in order to control the styling of individual characters, each character needs to be assigned with a unique class, leading to an unnecessarily complex HTML document. lettering.js automates this process by wrapping all the letters within the designated area with the `<span>` tag and specifying the class name.

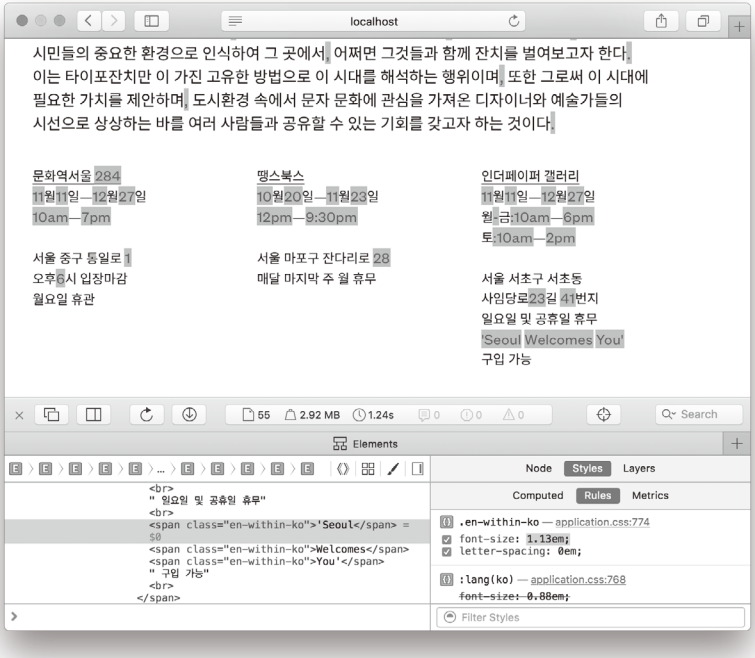
4 A JavaScript library is a modularized, pre-written JavaScript code for popular features that can be used by the public for easier development.

5 jQuery is a popular JavaScript framework, which can be extended by various plug-ins.



[4] 어도비 인디자인 CC 2015의 합성 글꼴 편집기.

Composite font editor of Adobe InDesign CC 2015.



[5] 태그를 이용한 파운더스 그로테스크와 노토 산스의 글자 크기 차이의 조절.

Adjusting the font size of Founders Grotesque and Noto Sans which using the tag.

지정된 영역 안의 모든 글자를 각각의 태그로 감싸고 클래스 이름을 자동으로 지정해준다.

```
<script>
  $(document).ready(function() {
    $(".fancy_title").lettering();
  });
</script>
```

```
<h1 class="fancy_title">
  <span class="char1">S</span>
  <span class="char2">o</span>
  <span class="char3">m</span>
  <span class="char4">e</span>
  <span class="char5">></span>
  <span class="char6">T</span>
  <span class="char7">i</span>
  <span class="char8">t</span>
  <span class="char9">l</span>
  <span class="char10">e</span>
</h1>
```

제안: 웹 브라우저에서 섬세한 섞어쓰기를 가능하게 하는 라이브러리의 개발
lettering.js가 필요한 요소들에 자동으로 클래스 이름을 부여할 수 있도록 자동으로 태그를 삽입하는 구조를 활용하면 어도비 인디자인의


```
<script>
  $(document).ready(function() {
    $(".fancy_title").lettering();
  });
</script>
```

```
<h1 class="fancy_title">
  <span class="char1">S</span>
  <span class="char2">o</span>
  <span class="char3">m</span>
  <span class="char4">e</span>
  <span class="char5">></span>
  <span class="char6">T</span>
  <span class="char7">i</span>
  <span class="char8">t</span>
  <span class="char9">l</span>
  <span class="char10">e</span>
</h1>
```


Proposition: Developing a Library that Enables Controlled Multilingual Typesetting in Web Browsers
Utilizing the structure that automatically inserts tag in order to automatically grant a class name for elements, it is possible to gain control through CSS in a way that resembles the functionality of Adobe InDesign's Composite Font. We implemented this through JavaScript while developing The ‹Typojanchi› website. This simple script filters

6 버전 관리 소프트웨어인 깃(Git)에 기반한 소셜 코딩 플랫폼으로, 다양한 오픈 소스 프로젝트들을 찾아보고 개발에 함께 참여할 수 있다.

합성 글꼴 기능에 근접하는 수준의 제어가 CSS를 통해서 가능해진다. 제4회 ‹타이포잔치› 웹사이트를 개발하면서 이것을 자바스크립트로 구현했는데, 이 간단한 스크립트는 HTML 문서 안의 내용 중 특정 언어를 정규식을 통해 걸러내어 이들을 태그로 감싸 지정된 클래스 이름을 부여한다. 이렇게 한 문단 내에서 영문과 한글에 각각의 클래스 이름이 지정되고 나면, CSS를 통해 영문과 한글 지정 글자체 사이의 글자 크기 차이를 제어할 수 있다.

이 스크립트는 본래 제4회 ‹타이포잔치› 웹사이트의 섞어쓰기 문제를 해결하기 위해 개발했으나 약간의 추가 개발을 통해 범용으로 사용될 수 있는 구현체를 만들 수 있고, 이 작업은 현재 진행 중이다. 2016년 4월 현재 오픈 소스 버전은 현재 버전 0.1.0은 GitHub[6]에 소스가 공개되어 있고 한글과 영어뿐 아니라 일본어나 중국어 등 웹 브라우저에서 섞어쓰기를 제어하고자 하는 누구나 활용할 수 있다.

multilingual.js

multilingual.js는 어도비 인디자인의 합성 글꼴 기능처럼, HTML/CSS 환경에서 보다 섬세하게 다국어 섞어쓰기를 제어하기 위한 오픈 소스 자바스크립트 라이브러리이다. 현재는 jQuery 플러그인의 형태로 GitHub(https://github.com/multilingualjs/multilingual.js)에 소스가 공개되어 있다.

이 플러그인은 HTML 문서 안에서 특정 문자 세트로 표기된 단어들을 정규식으로 골라내어 그 단어들을 태그로 감싸고, 언어 및 부호에 따라

6 This is a social coding platform that is based on Git, a version management software. Searching for various open source projects and participating in development is possible.

a specific language within an HTML document using regular expressions and assigns a designated class name through wrapping these with the tag. After the class names of Roman Alphabet and Hangeul in a paragraph are separately assigned, one can control the font size for English and Korean separately using CSS easily.

This script was originally developed to solve the multilingual typesetting issue of The ‹Typojanchi› website, but is in active development to become more general-purpose. At the time of writing (April 2016), the version 0.1.0 is open to the public in GitHub.[6] The script can be used to control multilingual typesetting in web browsers with not only Hangeul and Roman Alphabets but also Japanese and Chinese characters.[5]

multilingual.js

multilingual.js is an open-source JavaScript library that allows for detailed control over multilingual typesetting in HTML/CSS settings, in the manner of Adobe InDesign's Composite Font functionality. Currently the source is open to public in GitHub (https://github.com/multilingualjs/multilingual.js) in the form of a jQuery plug-in.

This plug-in selects words displayed in specific character sets in an HTML document using regular expressions. Then, it wraps those words with tag and assigns a particular class name according to the language and symbol. The character sets supported by default include English (en), Hangeul (ko), Chinese characters (cn), Japanese

별도의 클래스 이름을 부여한다. 기본으로 지원되는 문자 세트는 영어(en), 한글(ko), 중국어(cn), 일본어(jp), 숫자(num), 문장 부호(punct)가 있고, 별개의 낱자들을 골라내어 별도의 클래스 이름을 지정하는 것도 가능하다.

설치하기
최신 버전은 https://github.com/multilingualjs/multilingual.js/releases에서 다운로드하여 설치 가능하다.

HTML의 <head> 태그 안에 스타일 시트 파일을 삽입한다.

```
<link href="multilingual.css" rel="stylesheet" />
```

HTML의 <body> 태그가 끝나기 전에 multilingual.js 파일을 삽입한다.

```
<script type="text/javascript" src="jquery.multilingual.js">
</script>
```


characters (jp), numerals (num), and punctuations (punct). It is also possible to specify a separate class name for specific letters.

Installing
Install by downloading the latest version at:
https://github.com/multilingualjs/multilingual.js/releases

Insert the stylesheet file inside the HTML's <head> tag.

```
<link href="multilingual.css" rel="stylesheet" />
```

Insert multilingual.js file before closing HTML's <body> tag.

```
<script type="text/javascript" src="jquery.multilingual.js">
</script>
```

사용하기

설치가 완료되면, 자바스크립트에서 아래와 같이 설정하고 초기화할 수 있다.

```
<script>
  $(document).ready(function(e){
    $(".content").multilingual([
      "en", "num"
    ]);
  });
</script>
```

위의 예시에서는 페이지가 로딩될 때마다 `content` 클래스를 가진 요소 안의 모든 내용을 검색해, 영문(`en`)과 숫자(`num`)을 골라내어 각각의 단어/글자에 `ml-en` 또는 `ml-num` 클래스 이름을 할당한다. 이렇게 처리된 문서의 HTML 구조는 다음과 같다.

원본

```
<p>모든 CCL의 메타데이터에는 최소한 license 값을 기술하는 1개의 RDF 트리플이 반드시 포함됩니다.</p>
```

처리후

```
<p>모든 <span class="ml-en">CCL</span>의 메타데이터에는 최소한 <span class="ml-en">license</span> 값을 기술하는 <span class="ml-num">1</span>개의 <span class="ml-en">RDF</span> 트리플이 반드시 포함됩니다.</p>
```


In Use

After installation, one can initialize and configure the plug-in within JavaScript as seen below:

```
<script>
  $(document).ready(function(e){
    $(".content").multilingual([
      "en", "num"
    ]);
  });
</script>
```

In the example above, when the page loads, the script looks at all elements within the class name content in order to find Roman Alphabets(`en`) and numerals(`num`), and assign each word or letter to either the `ml-en` class or the `ml-num` class. The resulting HTML structure of the so treated document is as follows;

처리 후에는 각각의 문자 세트가 독자적인 클래스 이름으로 구별되어 있기 때문에, 각 클래스 이름에 해당하는 CSS 스타일을 적용해주는 것으로 섞어쓰기의 세부적인 사항들을 제어할 수 있다.

```
/* example css for multilingual.js */
p {
  font-family: NotoSans, Helvetica, Arial, sans-serif;
  font-size: 16px;
  line-height: 23px;
}
.ml-en, .ml-num {
  font-family: LiberationMono, Courier, monospace;
  letter-spacing: -0.02em;
  position: relative;
  top: -0.05em;
}
.ml-num {
  color: gray;
}
```


Original Text	`<p>모든 CCL의 메타데이터에는 최소한 license 값을 기술하는 1개의 RDF 트리플이 반드시 포함됩니다.</p>`
After Processing	`<p>모든 <span class="ml-en">CCL</span>의 메타데이터에는 최소한 <span class="ml-en">license</span> 값을 기술하는 <span class="ml-num">1</span>개의 <span class="ml-en">RDF</span> 트리플이 반드시 포함됩니다.</p>`

After processing, because each character set is assigned with individual class names, detailed control of multilingual typesetting can be achieved by simply applying CSS styles relevant to each class name.

```
/* example css for multilingual.js */
p {
  font-family: NotoSans, Helvetica, Arial, sans-serif;
  font-size: 16px;
  line-height: 23px;
}
.ml-en, .ml-num {
  font-family: LiberationMono, Courier, monospace;
  letter-spacing: -0.02em;
  position: relative;
  top: -0.05em;
}
.ml-num {
  color: gray;
}
```

옵션

1 기본 문자 세트: multilingual.js가 지원하는 문자 세트는 다음과 같다.[6]

2 커스텀 문자 세트: 기본 문자 세트 이외에도 특정 글자들을 선택해 클래스 이름을 지정할 수 있다. 이를테면 영문 글자체와 별개로 괄호만 스타일링하고 싶을 때는 다음과 같이 초기화 배열 안에 오브젝트로 옵션을 지정해주고, 지정한 클래스 이름(`className`)을 CSS에서 선언하면 된다.

```
 $(".content").multilingual([
  "en", {
    className: "ml-parenthesis", /* 클래스 이름은 어떤 것이든 가능하다. Class
name can be anything */
    charset: '()' /* ml-parenthesis 클래스 안에 포함될 문자 세트를 지정해준다.
characters to be selected, within ' ' */
  }
]);
```


Options

1 Basic: The letter sets supported by multilingual.js are shown in [6].

2 Custom character set: In addition to the basic character set, it is possible to select specific characters and designate a class name. For example, if a user wants to style parentheses separately from Roman Alphabets, it can be done by specifying the characters and their className in the initial array as an object:

```
 $(".content").multilingual([
  "en", {
    className: "ml-parenthesis", /* 클래스 이름은 어떤 것이든 가능하다. Class
name can be anything */
    charset: '()' /* ml-parenthesis 클래스 안에 포함될 문자 세트를 지정해준다.
characters to be selected, within ' ' */
  }
]);
```

Usage Examples

multilingual.js recommends to declare the basic attributes of typography to the basic elements of HTML such as `body` and selectively declare the additional attributes to the necessary class names such as `ml-en` or `ml-num` to override.

문자의 범위	문자 세트의 이름	클래스 이름
영어 [a-zA-Z]+	'en'	ml-en
한글 [가-힣]+	'ko'	ml-ko
일본어 [\u3040-\u309F\u30A0-\u30FF]+	'jp'	ml-jp
중국어 [\u4E00-\u9FBF]+	'cn'	ml-cn
숫자 [0-9]+	'num'	ml-num
문장 부호 [\(\).,“”‘’\-]|&|<|>| &emdash;|&endash;+	'punct'	ml-punct

[6]

Range of letters	Name of letter sets	Class name
English [a-zA-Z]+	'en'	ml-en
Korean [가-힣]+	'ko'	ml-ko
Japanese [\u3040-\u309F\u30A0-\u30FF]+	'jp'	ml-jp
Chinese [\u4E00-\u9FBF]+	'cn'	ml-cn
Numeric [0-9]+	'num'	ml-num
Punctuations [\(\).,“”‘’\-]|&|<|>| &emdash;|&endash;+	'punct'	ml-punct

[6]

사용 예시

multilingual.js는 아래의 예시처럼 `body` 등 HTML의 기본 요소에 타이포그래피의 기본이 되는 속성들을 선언하고, `ml-en` 또는 `ml-num` 등 필요한 클래스 이름에 선택적으로 추가 속성을 선언해 오버라이드할 것을 권한다.

```
body {
  font-family: NotoSans, Helvetica, Arial, sans-serif;
  font-size: 16px;
  line-height: 23px;
}
.ml-en, .ml-punct, .ml-parenthesis {
  /* shared styles for 'en', 'punct', and parentheses */
  font-family: SourceCodePro, Courier, monospace;
  font-size: 1.1em;
}
.ml-parenthesis {
  /* specific style for parentheses */
  /* shifting baseline */
  position: relative;
  top: -0.05em;
  /* adjust spacing before and after character */
  letter-spacing: -0.1em;
   margin-left: -0.1em;
}
```



```
body {
  font-family: NotoSans, Helvetica, Arial, sans-serif;
  font-size: 16px;
  line-height: 23px;
}
.ml-en, .ml-punct, .ml-parenthesis {
  /* shared styles for 'en', 'punct', and parentheses */
  font-family: SourceCodePro, Courier, monospace;
  font-size: 1.1em;
}
.ml-parenthesis {
  /* specific style for parentheses */
  /* shifting baseline */
  position: relative;
  top: -0.05em;
  /* adjust spacing before and after character */
  letter-spacing: -0.1em;
   margin-left: -0.1em;
}
```


The Control of Text Size and OtherAttributes

The size of the text is written in a relative value (`em` or `%`, etc.) based on the text size of the default font. `font-size: 1.1em;` is the same statement as `font-size: 110%;` and therefore the text size for the above example above is `17.6px`.

The basic attributes of typography for CSS3—`font size`, `line-height`, `letter-spacing`, `font-weight`, etc—are all usable. We recommend using relative units to override the values inherited, since

글자 크기 및 그 외 속성의 조절
글자의 크기는 기본으로 설정된 글자체 크기를 기준으로 상댓값으로(em 또는 % 등) 작성한다. font-size: 1.1em;은 font-size: 110%;와 동일한 선언으로, 위의 예시에서는 글자 크기가 17.6px로 표시된다.

이외에도 CSS3의 기본적인 타이포그래피 속성—font-size, line-height, letter-spacing, font-weight 등—은 모두 사용할 수 있다. 다만 px이나 pt 등 절대 단위로 속성을 정의하게 되면 매번 상속받는 글자 크기의 변화에 대응하지 못하므로 언제나 상댓값으로 정의하는 것이 바람직하다.

글줄(베이스라인)의 조절
CSS3 표준에서 글줄을 손쉽게 조절할 수 있는 방법은 없으므로, 편법에 의존할 수밖에 없다.

```
.ml-parenthesis {
  /* shifting baseline */
  position: relative;
  top: -0.05em;
}
```

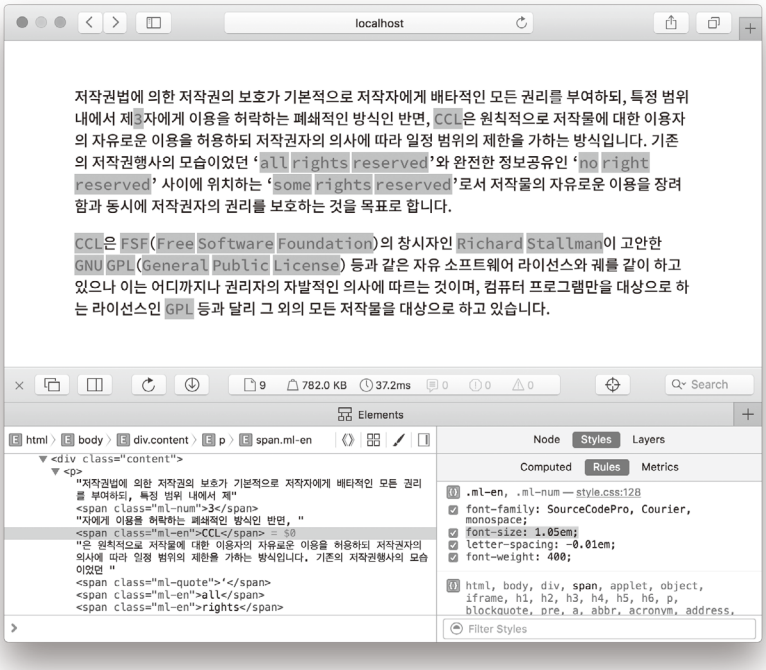

if the properties are defined in absolute terms such as px or pt, the character cannot adjust itself to the changing size of the text.

Baseline Shift
Since there is not an easy way for baseline shift in standard CSS3, there is no other choice but to rely on workarounds.


```
.ml-parenthesis {
  /* shifting baseline */
  position: relative;
  top: -0.05em;
}
```

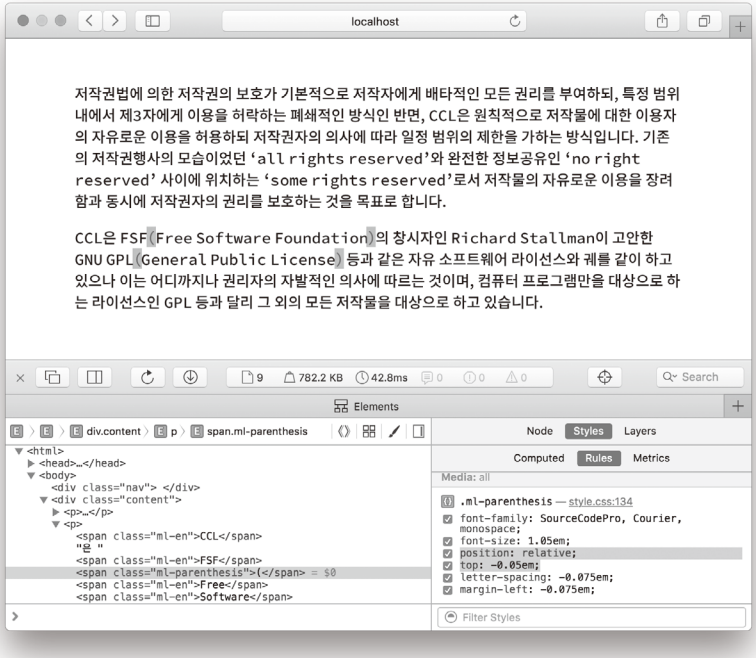

Demo
The complete demonstration of multilingual.js is available at http://multilingualjs.github.io.

Further improvements
Currently, dynamically adding content will result in content already wrapped ‹span› tags being wrapped again with the same tag. The regular expressions need to be improved in order to solve this problem.



[7] `font-size`를 이용한 글자 크기 조절.

Adjusting the text size by using `font-size`.



[8] `position: relative;`와 `top:`을 이용한 글줄의 조절.

Baseline shift using `position: relative;` and `top:`.

데모

multilingual.js의 온전한 데모는 http://multilingualjs.github.io에서 볼 수 있다.

추후 개선 사항

동적으로 콘텐츠가 추가될 경우에 이미 태그로 감싸져 있는 콘텐츠들에 한 번 더 을 적용하게 되는 문제가 있어 정규식의 개선이 필요하다.

결론

전통적인 인쇄의 영역에서 타이포그래퍼는 완벽한 상태의 타이포그래피를 만들어내기 위한 엔지니어링에 적극적으로 개입해왔고, 그에 따라 인쇄 기술 또한 발달해왔다. 마찬가지로 디지털 화면에서도 이와 같은 완전함을 추구하기 위해 운영 체제와 브라우저 단계에서는 글자체 렌더링 기술이 발달하고 있고, W3C와 같은 웹 표준 기관에서는 다양한 타이포그래피 장치를 쉽게 구현할 수 있도록 표준을 만들어오고 있다. 스크린 환경은 본질적으로 소프트웨어 엔지니어링이 함께 결부되는 영역으로, 디자이너 입장에서 이 모든 것을 조율하기에는 명백한 한계점이 존재하지만, 타이포그래퍼들은 언제나 관련 기술을 이해하고 때로는 직접 대면할 필요가 있다. multilingual.js는 브라우저에서 그런 한계점을 인식하고 스크린에서의 마이크로 타이포그래피 영역에서 더욱 완벽함을 추구하려는 시도이다. 이와 같은 활동이 인쇄 매체에서 구현했던 것과 근사한

Conclusion

Within the field of traditional printing, typographers have actively engaged in engineering in order to perfect typography, and printing technologies have developed accordingly. In the same vein, underlying technologies are being developed in the pursuit of such perfection within the digital display: such efforts include type rendering technology on the level of the operating system and web browser and the standardization of various typographical devices by web standard organizations such as W3C. The screen as environment is inherently associated with software engineering, and there exist apparent limitations to coordinate all those aspects from the designer's perspective; however, it is necessary that typographers understand and sometimes directly engage in the technologies that are involved. multilingual.js is an attempt to recognize such limitations within the web browser and pursue perfection in the field of microtypography. We hope this attempt serves as example for designers who wish to get on the screen a similar level of achievement to what has been realized in print media. It is also our hope that such accomplishments are made public through open activities including open-source development.

수준의 결과를 스크린에서 얻고자 하는 디자이너들에게 좋은 예시가 되어,
비슷한 성취가 오픈 소스와 같은 공개적 활동을 통해 드러나기를 기대한다.

웹사이트

Adobe Systems Incorporated. ‹InCopy Help | Using Fonts›. Accessed April 11, 2016. https://helpx.adobe.com/incopy/using/using-fonts.html#compositefonts

Clarke, Andy. ‹Improve your web typography with baseline shift›. Accessed May, 2009. https://stuffandnonsense.co.uk/blog/about/improve_your_web_typography_with_baseline_shift

Daggett, John. ‹CSS Fonts Module Level 3, W3C Candidate Recommendation 3›. Accessed October, 2013. https://www.w3.org/TR/css-fonts-3

Information Architect inc. ‹Responsive Typography: The Basics›. Accessed July, 2012. https://ia.net/know-how/responsive-typography-the-basics

Rauschmayer, Axel. ‹Speaking JavaScript: An In-Depth Guide for Programmers›. Accessed February, 2014.

Walton, Trent, Dave Rupert and Reagan Ray. «Lettering JS: A jQuery plugin for radical web typography». Accessed April 11, 2016. http://letteringjs.com

Website

Adobe Systems Incorporated. ‹InCopy Help | Using Fonts›. Accessed April 11, 2016. https://helpx.adobe.com/incopy/using/using-fonts.html#compositefonts

Clarke, Andy. ‹Improve your web typography with baseline shift›. Accessed May, 2009. https://stuffandnonsense.co.uk/blog/about/improve_your_web_typography_with_baseline_shift

Daggett, John. ‹CSS Fonts Module Level 3, W3C Candidate Recommendation 3›. Accessed October, 2013. https://www.w3.org/TR/css-fonts-3

Information Architect inc. ‹Responsive Typography: The Basics›. Accessed July, 2012. https://ia.net/know-how/responsive-typography-the-basics

Rauschmayer, Axel. ‹Speaking JavaScript: An In-Depth Guide for Programmers›. Accessed February, 2014.

Walton, Trent, Dave Rupert and Reagan Ray. «Lettering JS: A jQuery plugin for radical web typography». Accessed April 11, 2016. http://letteringjs.com

Translation. Kim Hansol
English supervision. Ku jaeun, Koh Achim

번역. 김한솔
감수. 구자은, 고아침