Using Font Awesome in HTML and PDF documents

R (Chandra) Chandrasekhar

2022-01-20

Make me a résumé, bake me a résumé

My friend, the polymath Solus "Sol" Simkin, sauntered into my relaxation-cubicle one morning muttering cryptically, "Make me a résumé, bake me a résumé". Noticing that I was not on the same page, he quickly asked, "Do you know the difference between a CV (curriculum vitae) and a résumé?"

When I said that it was the same thing, named differently across the Atlantic divide, he grinned and said "Gotcha!" and remained tantalizingly silent. After discussing other matters, he left, and I was not to see him again for the better part of three days.

Meanwhile, I tried to buttress my case by mining the Web for the differences in nuance and usage between a CV and a résumé. I sought and pounced upon the charming book *Divided By A Common Language: A Guide to British and American English* by Christopher Davies. In it, I discovered that a CV in the UK is called a résumé in the US, but that a CV is typically longer than a résumé. *So, a condensed CV was indeed called a résumé in the US*. In search of more details, I happened upon this clear explanation of the distinction between the two.

Armed with this authoritative knowledge, I was prepared for Sol the next time we met. Somewhat piqued, I asked him why he sniggered at me the last time, when I had been correct—even if not complete—in my answer.

He apologized and said, "I enjoy seeing you stew sometimes. That's what friends are for: to stew each other on occasion. Since we are into culinary metaphors, let me share with you more details of that latest commission.

"A youngish tech-type wandered into my office recently and said, 'Make me a résumé, bake me a résumé. But make it the best. I want a résumé to outzip all others.' "

"I took him to mean that he wanted a short but piquant résumé, that so stood out from all the others, that it would bag him whatever job he fancied."

I then asked him what his non-negotiable requirements were. His reply was prompt, as though he had rehearsed for this very question. He said, 'I want dinky *Font Awesome* icons ⌑ for my contact details.'

"Anything else?"

"Only that the résumé must be generated from the *same* source document to give *identical* outputs in three formats: PDF, HTML, and `docx`. That will make maintenance a cinch. No other stipulations."

The yeast gets to work

Sol continued: "For the world of me, I didn't know what Font Awesome was, let alone its icons. I discovered that it was in its sixth incarnation by the time I got acquainted with it. In any case, I didn't think it would strain my brain to insert graphical icons into a résumé to satisfy a client-stipulation.

But how wrong I was! It cost me a week in time just to investigate and meet the single source document requirement, let alone the Font Awesome icons. In the end, though, slogging and slugging it out, I managed the feat. I am sure you don't want to hear about it."

He knew my weakness—*I crave the thing denied*—and used it to tease me. So, I had to coax and cajole Sol to narrate to me yet another of his scriptorial escapades, leaving out no essential clue so, that I could track his mental trajectory and experience for myself the thrill of the chase. Exuding the essence of amicability, Sol agreed to treat me to another of his tales of cerebral sleuthing.

Font Awesome and Unicode real-estate

Sol recounted that Font Awesome was an atypical font. Its glyphs did not fit into standard Unicode blocks reserved for the symbols of a natural language or academic discipline. The fonts did not represent a finite alphabet, but were rather an extensible set of hieroglyphs, not unlike emoticons. And while the latter have been recognized and given their own Emoticon Unicode block, the Font Awesome icons are yet to receive their due recognition through allocation of dedicated Unicode real estate.

Accordingly, Font Awesome started out using unallocated areas of Unicode, defined as the Private Use Area (PUA), which at present can accommodate a total of 137,468 private-use characters. Given that Font Awesome is open-ended at present, it remains to be seen whether this unallocated Unicode PUA will prove sufficient for its future growth.

Moreover, the PUA may legitimately be used by other fonts as well. So, the problem of collisions—two *different* fonts using the *same* Unicode location in the PUA—remains a possible and unresolved issue. Such ambiguity might give rise to unpredictable output due to glyph clashes.

Moreover, Font Awesome has in the past strayed away from the PUA, and encroached upon the domains reserved for other purposes. This has thankfully been corrected in more recent versions. Sol decided to avoid altogether the complications of unwitting encroachment by using the latest version of Font Awesome.

Glyphs: characters, numbers, or images?

When Sol downloaded the desktop and web versions of the latest Font Awesome free version, he noticed when rummaging into the downloaded files that each glyph had a name, not unlike the name we give to letters of the Latin alphabet. The downloaded fonts shared a triune property:

1. Each glyph had a *name*. For example, ⌓ was called, not unsurprisingly, *font-awesome* since the flag it embodies is the branding chosen by Font Awesome for itself.

2. As an OpenType (OTF) or TrueType (TTF) font, each glyph had a *Unicode code point*. For example the *font-awesome* icon had the hexadecimal location code `f2b4` or `F2B4`.

3. In addition to its expression as a glyph in an OTF/TTF font, each symbol was also available as a *scalable vector image* in SVG format. This is unsurprising as we are dealing with a character set of icons or images.

Sol told me that he was not at first aware of this triple correspondence, but that he accidentally stumbled upon it as he delved deeper into the downloaded Font Awesome files. He said that the triad of a named glyph, a Unicode point, and an SVG icon each came to his rescue for output to PDF, HTML, and `docx`.

Sol made a mule-headed decision after this discovery: he would not fall back upon the SVG versions for his assignment, for it would defeat the very idea of a font if he started using the SVG icons that could as well have been arbitrary images. He wanted a *true* challenge rather than a staged one.

Two additional layers of identity

Font Awesome is distributed in two versions: a free version available at no charge, and a paid version called `Font Awesome Pro`. Over and above that, the free version comes in three *flavours*:

- `Regular`,
- `Solid`, and
- `Brands`.

The first two contained icons for general use while the third was explicitly for the logos of brands. Sol confided that skill was needed to invoke the correct flavour of free fonts from the appropriate font file to get the desired output. "A miss is as good as a mile, mind you," he said.

With this convoluted introduction, Sol excused himself to rush away for an appointment. It would be a week before he took up the dangling thread of his narrative again.

Single source file to multiple targets

When next we met, Sol started explaining how he fulfilled the stipulation of a single source file that would generate three target files in three different formats: PDF, HTML, and `docx`.

Sol had relaxed in his *cogitation hammock* and decided that he would ruminate on this vital requirement. His first thought was to do it all in Microsoft Word and use its built-in conversion capabilities to generate the PDF and HTML outputs. But there was a mute dissension at the background of his mind that refused to be silenced. He decided to let the creative yeast do its work undisturbed, and left the problem to ferment.

A few days later, it struck him that if Microsoft Word changed its default format as a result of program development in the same way that PDF version 1.4 gave way to PDF 1.5, or as HTML 4 deferred to HTML 5, his work would not survive the march of time. The more he thought about it, the more he became convinced that the most granitic of formats would be *pure text encoded in UTF 8*.

He rued that—by giving up on using Microsoft Word as his base document—he was now further away from his goal than ever before. He now had to convert text into three formats, not two, and that too, unaided by any professional program. The responsibility for this was his alone.

There was a nagging temptation to revert in silent acquiescence to Microsoft Word and put paid to the issue. But each time, he had to contend with the disquiet in his heart. It was a heavy feeling that he was running away from a hard problem rather than facing it manfully. He felt that he would sooner or later be forced to convert a single plain text source to multiple formatted documents: a challenge better faced sooner rather than later.

Pandoc to the rescue

In his quest for a plain text input format, Sol first thought of LaTeX with which he was familiar. It had several paths to convert output to PDF and also HTML. While there was no guarantee of foolproof conversion, it seemed promising. But the dealbreaker was that there was no route to Microsoft Word's `.docx` format. Sadly, he let his firstborn inspiration take flight.

He next surfed the Web for parsing programs that wielded the power of computer science to guarantee that the conversion would be *almost* error-free if indeed not *totally* error-free. He needed to convert formatted text into a structure like an abstract syntax tree that could again be re-converted into formatted text. He thought it strange that the most robust or efficient route from *A* to *B* was sometimes through *C*, but granted that it could be so sometimes.

He spent the next day sweeping the Web for "format converters using parsing". But the results he got were about *data* parsing: extraction of possibly useful data from documents. He decided to set that right by specifying "document format converters using parsing". But again he got mired in results for *parsing of data extracted from documents*.

Somewhat vexed, he tried "document format converters" and was led to a promising cloud-based service called Convertio. Another was Online-Convert.com. But the more he thought about it, the less inclined was he to use such services. For one, he could have stuck to his original idea of using Microsoft Word as his input format, and using its built-in converters for PDF and HTML. No uploading to the cloud. No security hassles. No fuss. No muss. But that was a road he had already decided to avoid.

He stared at his non-negotiable specifications once again. It was then that he realized that two of the three stipulated formats, PDF and HTML, were publicly specified, standardized specifications. Only the `.docx` format was proprietary. Sol decided to search using "open source document format converter". The first result showed Pandoc.

He first checked whether Pandoc could take in plain text and put out the three desired formats. After confirming that it could, he realized that it would be prudent to start off with Markdown and use that almost-plain-text source to generate the other three. He decided to use Pandoc's Markdown, which was a sophisticated outgrowth from the original Markdown.

He next took a look at the somewhat intricate network diagram on the right of the Pandoc home page. The plethora of formats available convinced him that Pandoc would future-proof his work in case even more output formats were required by his client later on. Sol decided to pitch his format conversion tent squarely on the Pandoc field.

A skeleton in Markdown

"A résumé must have content: in this case, my client's", said Sol, stating the blindingly obvious. "But for our intellectual excursion, we shall skip the content and replace it by abstract placeholders. That way, I keep my client's details confidential, and we will not muddy the waters with needless detail. In fact, I propose to show you only *one* simple line of pseudo content, which is also *the single line* where Font Awesome makes its debut."

Sol then showed a snippet of a Pandoc Markdown text file with the pseudo content shown below:

```
:::::center
[Joe Bloggs]{.larger}\
[[mail icon] <joebloggs@mydomain.com> |
[mobile icon] +1-555-3456-7890 |
[LinkedIn icon] [LinkedIn] |
[Github icon] [Github]]{.smaller}
:::::
```

He explained that the four icons above were a sufficient complement of Awesome Font icons to demonstrate proof of concept, and that any additions would simply translate to 'more of the same'. "Now that the scaffolding is in place, let us get to the meat of the matter."

The atom icon

Schooled as he was Greek philosophy, Sol decided that the atom would be the fittest icon to begin his acquaintance with Font Awesome. He would experiment and build his knowledge atom by atom.

He went to the latest icons page, selected `Free` on the left checkbox and typed "atom". From the gallery of 32 icons, he selected the one that best met his expectations. The popup screen he saw is shown below:
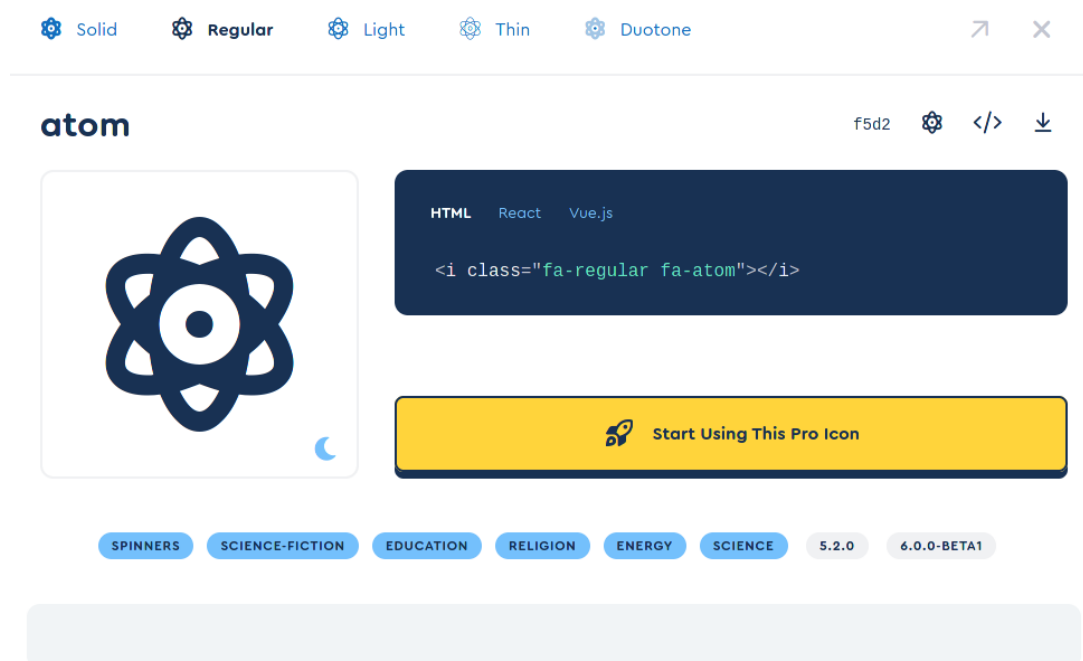
Its Unicode code point was `f5d2`, its name was `atom`, its SVG icon file was called `atom.svg`, and the glyph could be copied by clicking on the picture of the atom next to the Unicode value. He was seeing the `Solid` flavour of the icon; other variants were not included in the free version he had downloaded. Interestingly, the code snippet for HTML was also given with the icon as:

```
<i class="fa-solid fa-atom"></i>
```

where the `<i>` stood for "Idiomatic Text element".

Not one to take things at face value, Sol immediately tested the HTML invocation on his browser hoping against hope that the given incantation *alone* could automagically conjure up the atom icon. When it did not, Sol muttered to himself that more work was in store.

He wracked his brain for the reason for failure. After some effort, he decided to simulate his web browser in his head by attempting to 'act' like it. "I will assume that the given incantation is meaningful and correct. But it is only an HTML fragment. If I were a web browser, what would I need *in addition to the incantation* to display the atom icon? How does my browser *know* what that incantation means? It was not born with that knowledge pre-programmed into it somehow. *So who tells it what the incantation means, and where to look for that icon?*" With that thought uppermost in his mind, Sol retired for the night.

A second look at the atom icon

When Sol took a second look at the problem the next day, he was aghast that he had missed many clues strewn all over the Font Awesome website that would have guided him in the right direction. Not one to waste time on regret, he started to assemble the given sequence of instructions and follow them meticulously.

Sol loaded this page to be greeted by two large selection buttons:
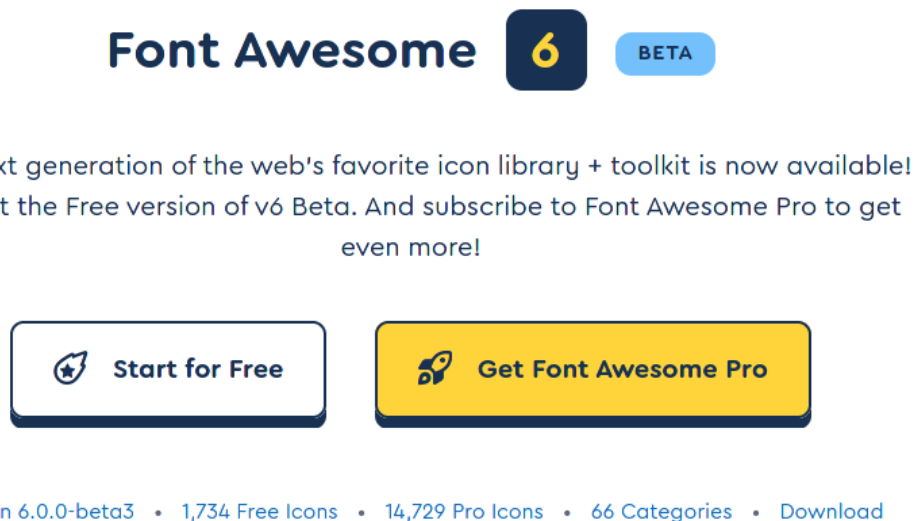
And when he selected `Start for Free`, he was led to a clear and detailed algorithm on how to set up and use Font Awesome. Central to everything was a simple one-liner *customized for him* that yielded an HTML script when he clicked `Set up a Kit`. The script he got looked like this:

```
<script src="https://kit.fontawesome.com/[some-hex-digits].js"
    crossorigin="anonymous"></script>
```

The script was supposed to be included in the `<head>` of the HTML document. This one line was the automagical incantation that Sol was looking for the previous night. He tried it out on the `atom` icon. The text

```
<i class="fa-solid fa-atom"></i>
```

yielded the icon ⚛Voila!

The maroon[1] colour arose from a little CSS fragment that set the colour of the icon to maroon with

```
.fa-solid {
    color: maroon;
}
```

This meant that *all* Font Awesome `Solid` icons will show up in maroon. If he wanted another `Solid` icon, say, `paper-plane`, to show up in a blue-green colour, he would need to override this setting with

```
.fa-paper-plane {
    color: #004369; /* a deep shade of blue-green */
}
```

to get a correctly coloured `paper-plane` icon from the `Solid` flavour: ✈. Indeed, even the version from the `Regular` flavour would be in the same colour: ✈

The four-icon alphabet

Inspired by his success and encouraged by the ease of use of the icons in HTML, Sol decided on the four-icon alphabet he would use for the placeholders in his single-line résumé template. He laid out his data so:

Unicode	Flavour	Name	Icon	Glyph
f0e0	Regular	envelope		✉
f3cd	Solid	mobile-screen-button		📱
f0e1	Brands	linkedin-in		in
f09b	Brands	github		ⓞ

The column headed `Icons` was generated by the `<i class="fa-... fa-..."></i>` code while that labelled `Glyphs` contained the relevant glyph directly copied from the popup.

Note that the mobile phone icon available in the free version was only from the `Solid` flavour, whereas the envelope icon was available in both the `Regular` and `Solid` flavours.

The icon colour was set by this CSS fragment, and applies only to the `Icons` column:

---

[1]The PDF version of this font on the PDF version of this blog will be shown in grey as explained later.

```css
.fa-envelope, .fa-mobile-screen-button, .fa-linkedin-in, .fa-github {
  color: #2e8bc0;
}
```

"Notice that the Glyph column[2] only shows tofu rectangles" continued Sol. "This is because the browser has not been instructed on which fonts to use for these specialized Unicode characters from the PUA."

"On my text editor, the glyphs I see in column 5 correspond exactly with those displayed on the web page in column 4. To get the browser to display the glyphs directly, without the `<i class=...></i>` invocation, requires more effort than I am willing to expend right now. So, I took the easy route here." ☺

Wrapping up the HTML version

"Wait a minute," I said. What about the PDF and the `.docx` versions?

"Patience! Patience," he replied. "Markdown was originally devised to play well with HTML. Pandoc has carried that ball far and wide into many other formats. All I have told you thus far applies only to HTML. Let me wrap it up first before moving on to PDF. In HTML, the one-line résumé template in Markdown looks like this:

```
Joe Bloggs
<i class="fa-regular fa-envelope"></i> <joebloggs@mydomain.com>\
<i class="fa-solid fa-mobile-screen-button"></i> +1-555-3456-7890\
<i class="fa-brands fa-linkedin-in"></i> [LinkedIn account of Joe Bloggs]()\
<i class="fa-brands fa-github"></i> [Github account of Joe Bloggs]()
```

which appears like this[3]:

Figure 1: The Font Awesome icons take on the blue colour above.

The styling, like font size and text-alignment, is accomplished using CSS." With a look of accomplishment, Sol ended his narration and asked me whether I honestly wanted to hear how the PDF version was generated. When I nodded in eager agreement, he said he would take up the thread on another day.

---

[2]In the PDF version of the blog, it is the `Icon` column that is blank, whereas the `Glyph` column, containing the glyph, shows up.

[3]This file could not be named as `opening-html.html` because of an operational quirk in the Pelican Static Site Generator being used on this site. Accordingly, the `.html` file extension has been left out. Save this file, rename it as `opening-html.html`, open it in a Web browser, and verify for yourself the claim made here.

The route to PDF

"The single most important difference between HTML on the one hand, and PDF and .docx on the other is that the latter two are bound to a physical paper size. The maximum width of text in these two formats are constrained by the type of paper used. I will assume either A4 or US letter sizes for the output.

"The second aspect of these latter two formats is that the requisite font must reside on the device generating the output. Moreover it is possible, with some effort, to embed the fonts within the document so that another person viewing the document on another device *without* the fonts on his or her system can still view the document with precisely the intended fonts."

With this preamble, Sol started on his story of how he got the PDF version of the one-line résumé template.

Does the font exist on my desktop?

He said he used a combination of Pandoc and the TeX/LaTeX typesetting eco-system to generate the PDF. A very important first step was to confirm that the desired glyphs were indeed provided by the font in question. For that, Sol used albatross. This utility may be invoked either with the Unicode glyph itself as argument, or with the Unicode code point prefixed with either 0x or U+.

Sol tried albatross U+f3cd (for the mobile-screen-button) and got Font Awesome 6 Free Solid as the only result. But when he typed albatross 0xf0e0 (for the envelope) he got a total of seven fonts, including Font Awesome 6 Free Regular and also Font Awesome 6 Free Solid. This was why one should specify the font to be used—to avoid the same glyphs from a different font, possibly with an incompatible style, or worse, of an entirely different glyph as well, because the glyph is in the PUA area of the font.

The results he got were sufficient for Sol to write the following fragments in LaTeX:

```
\newfontfamily{\regulariconfont}{Font Awesome 6 Free Regular}[Color=Grey]
\newfontfamily{\solidiconfont}{Font Awesome 6 Free Solid}[Color=Grey]
\newfontfamily{\brandsiconfont}{Font Awesome 6 Brands}[Color=Grey]
%
\newcommand{\faEnvelope}{\regulariconfont\ ^^^^f0e0\normalfont}
\newcommand{\faMobile}{\solidiconfont\ ^^^^f3cd\normalfont}
\newcommand{\faLinkedin}{\brandsiconfont\ ^^^^f0e1\normalfont}
\newcommand{\faGithub}{\brandsiconfont\ ^^^^f09b\normalfont}
...
\begin{center}
{\Large Joe Bloggs}\\
{\small \faEnvelope\
\href{mailto: joebloggs@mydomain.com}{joebloggs@mydomain.com} |
\faMobile\ +1-555-3456-7890 |
\faLinkedin\ \href{https://www.linkedin.com/}{LinkedIn} |
\faGithub\ \href{https://github.com/}{GitHub}}\normalsize
\end{center}
```

The resulting PDF content, imaged below, matches the HTML output shown earlier.

Joe Bloggs

✉ joebloggs@mydomain.com | 📱 +1-555-3456-7890 | in LinkedIn | ○ GitHub

Figure 2: Note that the Font Awesome icons are grey, following the settings above.

File manifest

Sol concluded by saying that the following files, in the order of their invocation in the `make-html-pdf` script, were used to generate the HTML and PDF output from Pandoc's Markdown source:

1. `html-defaults.yaml`
2. `html-header-includes`
3. `awesome.css`
4. `opening-html`
5. `opening-html.txt`
6. `latex-defaults.yaml`
7. `latex-font.yaml`
8. `latex-header-includes.tex`
9. `latex-before-body-includes.tex`
10. `opening-latex.pdf`
11. `opening-latex.txt`

It bears noting that the input files for the two formats, `opening-latex.txt` and `opening-html.txt`, are indeed non-identical. In that sense the challenge has not been met. But given that the first line of the résumé may be concatenated with the rest of the résumé, *which is unchanged for both formats*, the challenge has been successfully, in spirit if not in letter.

I was lost in admiration for Sol's tenacity, which bent fonts and typesetting systems to his will, to generate near-identical output in two different, ubiquitous formats. Even if the effort expended on this single line of text was disproportionate to the outcome, the experience earned was worth every ounce of effort.

The `.docx` output

It was some weeks before I met Sol again in an unhurried setting. I then reminded him to finish his saga with Font Awesome by explaining how he generated the `.docx` version of the résumé opening. Sol squinted his eyes and addressed me seriously: "Do you *really* want to know that? If you do, grab a chair and have some water handy, in case you feel dizzy. ☺ "

With that admonitory preamble, Sol launched into the third part of his exploit: to get a `.docx` fragment visually identical to the HTML and PDF versions from the same Markdown source. It was more a marathon than a sprint, and required fortitude of body and mind.

"I do not claim to be an expert on Microsoft Word or its `.docx` format. But I do have a secret weapon, or should I say, a secret agent: my paternal cousin, once removed, Hieronymus Septimus Simkin, who is affectionately known to me as Seven. He has been steeped and pickled in the brine of Microsoft software since he barely started walking, and he is my go-to resource on all matters relating to Windows applications.

"The road to `.docx` is a little convoluted and very tedious. Why don't I try to get Seven to brief you himself. Let us agree to hold that story in abeyance for another day and another blog.

Feedback

Please email me your comments and corrections.

A PDF version of this article is available for download here.[4]

---

[4]Note that some icons that show up in the HTML version are missing from the PDF version of the blog, and vice versa. It was too tedious to make them feature-identical. One workaround would have been to use scaled SVG icons for both output formats, instead of HTML snippets for HTML, and OTF fonts invoked by Unicode code points, for LaTeX, but Sol felt that was a cop out. The opening of the résumé is almost identical in the HTML and PDF versions, and that was the challenge that was to be fulfilled: so no rules were broken in playing the game. 😊