Subject: Re: JavaScriptCore
Posted by cbpporter on Mon, 19 Jul 2010 07:10:20 GMT
View Forum Message <> Reply to Message

Well, I have had a lot of contact with Unicode both in U++ and outside. From a pure completionist and compatibility point of view, when compared to the Unicode standard, U++ is around version 1.1, with some features from latter version but fewer characters and lacking some key features. This would seem very bad, but in practice this is not that bad because everything has very poor Unicode support. As you noticed, there are only 2048 characters in that table, but I think it is safe to assume that most of our users have not hit this barrier. If you live in the United States you are covered. Also, support for most of Europe is very good. Also, a lot of business software has very poor Unicode support. Delphi versions have only recently started using Unicode if I'm not mistaken.

The problem of Unicode is very simple to mask because if you do not want to process the text, only show if, Windows, especially Vista and probably 7 also, is very good at showing it. So your application may not know a thing about Unicode, but you will probably be able to render all text if you have fonts for it. Windows' support is not perfect either. Character composition and ligatures are generally sub par.

In the past I tried to improve Unicode compatibility, but I had little support in my efforts and rightly so because I was having a very specific Unicode issue and I realized that my need was not general.

The problem is that Unicode is very complex. It is not about the character tables. Even implementing a proper ToLower for only the first 2048 characters is a lot more complicated that the current implementation. In Linux it is exponentially more complicated because of the need for very complex character escaping because font support is very poor.

So at this point I believe we have two choices:
1. Continues as we have in the past, having good practical support for common need of Latin alphabet (and variations) using languages, but very poor theoretical compliance to the standard.
2. Go full monty, and add very good support, but the rest the ecosystem being as it is, we will probably be the only ones. Right now in 2010, KDE4, Windows (in general only at rendering) and a few specific scholarly tools have "good" support of Unicode. Standard C++ libs have abyssal support for it. std::wstring is not even Unicode. KDE4 was very good at displaying Unicode in my limited, good at allowing the user to input it, and probably has some potent methods for processing in Qt. Gnome is OK, but not great or as great as KDE. If you live in Europe your software has poor support for anything exotic, and might choke on a few common Unicode strings in various states of normalization, especially when using Mac strings. Probably very few people notice or need better.