Subject: Re: JavaScriptCore
Posted by cbpporter on Fri, 13 Aug 2010 08:46:50 GMT
View Forum Message <> Reply to Message

luzr wrote on Fri, 13 August 2010 10:09
22-28 (7 bits) toascii

Great, now I have to check the ASCII codes too . But I could use this info in my Linux font escaper, where I have a separate table for it.

Quote:
It would be nice to know, after fixing <2048, what are codepoints of those more errors - perhaps we could handle them too...

That will not be a problem. Actually, Unicode 6.0 is very close, so I will be trying to get as many such unintrusive fixes as possible.

The hardest part is to figure out a very compact bit layout and get as much of the Unicode Database encoded. And I'm afraid I am going to need a little more than 1 bit. Also, the ratio of compactness/performance is important. Maybe latter we'll need something like (of the top of my head):

int ToUpper(int c)   { return (dword)c < 2048 ? (uni__info[c] >> 11) & 2047 : ((dword)c > 50000 && (dword) c < 50512] ? (uni__info[c - 50000] >> 11) & 2047: c); }
to handle those extra errors and we want to avoid an 64Ki table. Would such a performance penalty be acceptable? Or maybe we'll have a 64Ki 1 byte table with properties for characters, and extra 4 bytes for characters that need special case information. Even today, the last character that has meaningful lower/uppercase data is 1414. There are at least 1982 characters with case information, 938 in the first 2048.

Of course, this is just speculation for the future, right now I'm only concerned with keeping the current layout for uni__info, but fixing the 93 errored codes.