Subject: JavaScriptCore
Posted by dolik.rce on Sat, 10 Jul 2010 12:59:06 GMT
View Forum Message <> Reply to Message

Hello friends!

I happened to be stuck for a few days away from my computer, with just an old windows pc. So having nothing better to do, I installed theide and started to dig into WebKits JavaScriptCore.

After few days of head banging and hair pulling , I managed to get all the code to compile and link with mingw. And what is more important, it can be used within an U++ application.

As of now, I have a simple interactive console parser (think of calculator in theide, just with JS instead of Esc) based on jsc from WebKits tree (they use it for testing) rewritten in U++. The parser is small, about 100 lines of code, but JavaScriptCore is way too big post all the code here. If someone is interested I can post instructions how to get the code and necessary patches later. For now, you can download the binary files here (8MB). Also, there are still some obstacles, like dependence on ICU library (17MB of dlls ), but I believe it can be solved in near future (some WebKit ports work without it, so I guess we can too).

As soon as I get better feeling about how this thing work, I plan to write some nicer U++ style interface around the library, so that anyone can use it from within their apps. And of course, as soon as I get back to my computer, I try the same on linux.

I hope that having JavaScriptCore available in U++ will be useful for someone. But most importantly, it is the first step towards WebKit in U++

Best regards,
Honza

Subject: Re: JavaScriptCore
Posted by mirek on Sat, 10 Jul 2010 14:23:53 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Sat, 10 July 2010 08:59Hello friends!

I happened to be stuck for a few days away from my computer, with just an old windows pc. So having nothing better to do, I installed theide and started to dig into WebKits JavaScriptCore.

After few days of head banging and hair pulling , I managed to get all the code to compile and link with mingw. And what is more important, it can be used within an U++ application.

As of now, I have a simple interactive console parser (think of calculator in theide, just with JS instead of Esc) based on jsc from WebKits tree (they use it for testing) rewritten in U++. The parser is small, about 100 lines of code, but JavaScriptCore is way too big post all the code here. If someone is interested I can post instructions how to get the code and necessary patches later. For now, you can download the binary files here (8MB). Also, there are still some obstacles, like

dependence on ICU library (17MB of dlls ), but I believe it can be solved in near future (some WebKit ports work without it, so I guess we can too).

As soon as I get better feeling about how this thing work, I plan to write some nicer U++ style interface around the library, so that anyone can use it from within their apps. And of course, as soon as I get back to my computer, I try the same on linux.

I hope that having JavaScriptCore available in U++ will be useful for someone. But most importantly, it is the first step towards WebKit in U++

Best regards,
Honza

Sounds very very good.

Webkit itself, any chances? Anybody?

Mirek

---

Subject: Re: JavaScriptCore
Posted by dolik.rce on Sat, 10 Jul 2010 17:29:06 GMT
View Forum Message <> Reply to Message

luzr wrote on Sat, 10 July 2010 16:23Sounds very very good.

Webkit itself, any chances? Anybody?

Hi Mirek,

I was hoping this will drag someones attention so I don't have to do the WebKit port all by myself But in worst case, I will definitely at least try...

I cleaned the things up a bit, so here are the "How to do this at home" instructions

Get the WebKit sources from SVN:
cd C:\upp
svn checkout http://svn.webkit.org/repository/webkit/trunk/JavaScriptCore WebKit\JavaScriptCoreYou could also get the full trunk, but for now it is not necessary
Patch the sources:
Download the attachment of this message, inside is directory JavaScriptCore, which contains .upp file and three files that need to be patched. The directory structure is the same, so it should be save just to copy it over the original directory tree.
Copy API headers:
Because of the way how WebKit is normally built, it expects some of the headers in JavaScriptCore\JavaScriptCore, so we have to copy it in there.
cd C:\upp\WebKit\JavaScriptCore

copy API*.h JavaScriptCore
Create look-up tables:
Some parts of JavaScript core use look-up tables which are usually generated in makefile, but we have to do it ourselves for now.
python create_regex_tables > yarr/RegExpTables.h
perl create_hash_table parser/Keywords.table > parser/Lexer.lut.h
perl create_hash_table runtime/RegExpConstructor.cpp -i > runtime/RegExpConstructor.lut.h
perl create_hash_table runtime/MathObject.cpp -i > runtime/MathObject.lut.h
perl create_hash_table runtime/JSONObject.cpp -i > runtime/JSONObject.lut.h
perl create_hash_table runtime/DatePrototype.cpp -i > runtime/DatePrototype.lut.h
perl create_hash_table runtime/ArrayPrototype.cpp -i > runtime/ArrayPrototype.lut.h
perl create_hash_table runtime/StringPrototype.cpp -i > runtime/StringPrototype.lut.h
perl create_hash_table runtime/RegExpObject.cpp -i > runtime/RegExpObject.lut.h
perl create_hash_table runtime/NumberConstructor.cpp -i > runtime/NumberConstructor.lut.h
perl pcre/dftables pcre/chartables.c Sorry for the slashes, I did this in cygwin so they are unix style, if you have windows perl or python, you have to rewrite them. In case you don't have perl or python I added them into the attached file as well. They are in the lut directory, just copy them to correct places.
Download ICU:
Download and extract the library from http://download.icu-project.org/files/icu4c/4.4.1/icu4c-4_4_ 1-Win32-msvc9.zip. If you have 64b windows, you might need to download different file. I extracted this file into C:\upp\Webkit. If you put it somewhere else, you will have to change the following paths accordingly. Now it is necessary to make mingw aware of ICU, so we add C:\upp\WebKit\icu\include to include directories and C:\upp\WebKit\icu\lib in lib directories in build methods.
Setup assembly:
Create new assembly from the C:\upp\WebKit and C:\upp\uppsrc nests.
Get jsc package:
The last directory in the attachment file is jsc. That is the interpreter package. Just copy it into C:\upp\WebKit.

Doesn't look really difficult, right? The worst part was getting all the switches in wtf/Platform.h right
If I didn't forget something, you should be now able to compile, link and run jsc. If I did forget something, let me know

Honza

EDIT: Corrected paths in upp file and reuploaded the attachment.
EDIT2: I knew I forgot something One more lookup table, now added.
EDIT3: Hopefully last one... Forgot to mention ICU dlls (from C:\upp\WebKit\icu\bin) must be either on PATH or in the same directory as the final executable. But you all probably know that

## File Attachments

1) jsc_all.zip, downloaded 437 times

---

Subject: Re: JavaScriptCore

Posted by tojocky on Sat, 10 Jul 2010 22:05:54 GMT
View Forum Message <> Reply to Message

What About v8 from google ( code.google.com/p/v8 )?

Subject: Re: JavaScriptCore
Posted by dolik.rce on Sat, 10 Jul 2010 22:15:15 GMT
View Forum Message <> Reply to Message

tojocky wrote on Sun, 11 July 2010 00:05What About v8 from google ( code.google.com/p/v8 )?
Hi Ion,
It should be possible. When building Chromium port of WebKit, it is just one #define in config.h to switch between JavaScriptCore and V8. But I haven't got that far yet.

Honza

Subject: Re: JavaScriptCore
Posted by tojocky on Sun, 11 Jul 2010 12:35:26 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Sun, 11 July 2010 01:15tojocky wrote on Sun, 11 July 2010 00:05What About v8 from google ( code.google.com/p/v8 )?
Hi Ion,
It should be possible. When building Chromium port of WebKit, it is just one #define in config.h to switch between JavaScriptCore and V8. But I haven't got that far yet.

Honza

I was compiled v8 in u++. I tested only v8 stand alone simple code. I think it is not a problem to create a v8 wrap plug-in.
Goo wrapper is v8-juice (http://code.google.com/p/v8-juice/) We can try. I'm interested in this project!

Ion Lupascu (tojocky)

Subject: Re: JavaScriptCore
Posted by dolik.rce on Sun, 11 Jul 2010 14:24:16 GMT
View Forum Message <> Reply to Message

tojocky wrote on Sun, 11 July 2010 14:35dolik.rce wrote on Sun, 11 July 2010 01:15tojocky wrote on Sun, 11 July 2010 00:05What About v8 from google ( code.google.com/p/v8 )?
Hi Ion,
It should be possible. When building Chromium port of WebKit, it is just one #define in config.h to

switch between JavaScriptCore and V8. But I haven't got that far yet.

Honza

I was compiled v8 in u++. I tested only v8 stand alone simple code. I think it is not a problem to create a v8 wrap plug-in.
Goo wrapper is v8-juice (http://code.google.com/p/v8-juice/) We can try. I'm interested in this project!

Ion Lupascu (tojocky)

I guess we could make a common interface with two different backends, allowing user to switch between them. Each of those engines have it's advantages. If I'm not mistaken, V8 has better performance, while JSC supports more platforms. And probably even more key differences.

Honza

---

Subject: Re: JavaScriptCore
Posted by tojocky on Sun, 11 Jul 2010 18:32:45 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Sun, 11 July 2010 17:24
I guess we could make a common interface with two different backends, allowing user to switch between them. Each of those engines have it's advantages. If I'm not mistaken, V8 has better performance, while JSC supports more platforms. And probably even more key differences.

Honza

Please inform me about your actions.

Ion Lupascu (tojocky).

---

Subject: Re: JavaScriptCore
Posted by dolik.rce on Tue, 13 Jul 2010 18:39:01 GMT
View Forum Message <> Reply to Message

Small update: Above described process almost works on linux as well. The only differences are that step 5 can be skipped, since libicu is present in most distributions by default and that you must link with icui18n instead of icuin.

The example interpreter builds and works fine, but there is a small memory leak... Weird is that the code is the same as on windows. I'd be thankfull for any hints on why there can be difference between win32 and linux.

Honza

Subject: Re: JavaScriptCore
Posted by dolik.rce on Wed, 14 Jul 2010 15:01:31 GMT
View Forum Message <> Reply to Message

Hello all,

I put together an U++ style interface for JavaScriptCore, as I promised.

It's public part is super simple (well, actually the internals are very simple too ):
```
class JavaScript{
public:
 JavaScript();
 ~JavaScript();
 void AddNativeFunction(const char* name,NativeFunction function);
 bool Execute(const String& code,String& result);
};
```
Even though it is simple, it still should be usefull. I never used much more than this when scripting with Esc.

If anyone has any ideas how to improve or extend this interface, let me know. Maybe Execute(filename, result)?

Attachment contains new version of jsc package. If used as main package it compiles the demonstration interpreter. If included in other package, it just provides the interface.

Best regards,
Honza

EDIT: Removed the file in favor of the link below.

Subject: Re: JavaScriptCore
Posted by tojocky on Thu, 15 Jul 2010 06:25:17 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Wed, 14 July 2010 18:01Hello all,

I put together an U++ style interface for JavaScriptCore, as I promised.

It's public part is super simple (well, actually the internals are very simple too ):
```
class JavaScript{
public:
 JavaScript();
 ~JavaScript();
```

```
 void AddNativeFunction(const char* name,NativeFunction function);
 bool Execute(const String& code,String& result);
};
```
Even though it is simple, it still should be usefull. I never used much more than this when scripting with Esc.

If anyone has any ideas how to improve or extend this interface, let me know. Maybe Execute(filename, result)?

Attachment contains new version of jsc package. If used as main package it compiles the demonstration interpreter. If included in other package, it just provides the interface.

Best regards,
Honza

Can you upload all packages?

---

Subject: Re: JavaScriptCore
Posted by dolik.rce on Thu, 15 Jul 2010 17:00:04 GMT
View Forum Message <> Reply to Message

tojocky wrote on Thu, 15 July 2010 08:25Can you upload all packages?
Hi Ion,

Sure I can. It is just too big to put it on the forum. I've put a zip file up for download elsewhere (11MB). It contains JavaScriptCore and the interface package. Only additional thing needed is the ICU, which is linked above (only on windows,linux should have it).

Best regards,
Honza

---

Subject: Re: JavaScriptCore
Posted by mirek on Sat, 17 Jul 2010 07:52:57 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Thu, 15 July 2010 13:00tojocky wrote on Thu, 15 July 2010 08:25Can you upload all packages?
Hi Ion,

Sure I can. It is just too big to put it on the forum. I've put a zip file up for download elsewhere (11MB). It contains JavaScriptCore and the interface package. Only additional thing needed is the ICU, which is linked above (only on windows,linux should have it).

Best regards,

Honza

Ideally, we should have any such "plugin"  managed the same way, which means converted to source based package (like e.g. plugin/jpg).

Do you think anything like that would be possible for any of libraries mentioned here?

Mirek

---

Subject: Re: JavaScriptCore
Posted by dolik.rce on Sat, 17 Jul 2010 08:55:48 GMT
View Forum Message <> Reply to Message

luzr wrote on Sat, 17 July 2010 09:52Ideally, we should have any such "plugin"  managed the same way, which means converted to source based package (like e.g. plugin/jpg).

Do you think anything like that would be possible for any of libraries mentioned here?

Mirek
Hi Mirek,

I like the plugin philosophy for foreign code in U++, so yes, I will try to achieve that. Either I create a package for ICU or, which would be even better, try to get it to compile without it. I hope U++ knows enough unicode to substitute it. I will have a look at the Qt port, hopefully it will give me some hints how they do it.

Anyway, my goal is to keep the JSC code as vanilla as possible to let us upgrade the original sources easily. Ideally just with "svn up".

Honza

---

Subject: Re: JavaScriptCore
Posted by dolik.rce on Sat, 17 Jul 2010 21:24:41 GMT
View Forum Message <> Reply to Message

Good news: I managed to get rid of the ICU library. Bad news: I replaced it with gliband pango. We link with those two on Linux with GUI anyway, but I don't like this solution because it still means supplying those libs on windows and in console and NOGTK builds.

Another good news is, that I actually have even fully U++ solution, without external libs. The only problem is that I had to use some function stubs to get it working. To get it into production quality, I need to implement (in unicode safe manner):
case folding,
ToLower,

ToUpper,
comparation,
determining the character category (as described for example here).

I suspect that for win32 there should be some native functions to do this, but I am not familiar with windows at all... For Linux, I'm not really sure where to start if I don't wan't to read the whole unicode reference. So I'll be very thankfull for any help or hints where to look.

Honza

---

Subject: Re: JavaScriptCore
Posted by mirek on Sun, 18 Jul 2010 06:59:36 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Sat, 17 July 2010 17:24Good news: I managed to get rid of the ICU library. Bad news: I replaced it with gliband pango. We link with those two on Linux with GUI anyway, but I don't like this solution because it still means supplying those libs on windows and in console and NOGTK builds.

Another good news is, that I actually have even fully U++ solution, without external libs. The only problem is that I had to use some function stubs to get it working. To get it into production quality, I need to implement (in unicode safe manner):
case folding,
ToLower,
ToUpper,
comparation,
determining the character category (as described for example here).

U++ ToLower/Upper are unicode, at least they are intended to be.

We have some char properties too (actually, it is implemented with the same table:)

We can extend that.

Mirek

---

Subject: Re: JavaScriptCore
Posted by dolik.rce on Sun, 18 Jul 2010 21:04:30 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 18 July 2010 08:59U++ ToLower/Upper are unicode, at least they are intended to be.

We have some char properties too (actually, it is implemented with the same table:)

We can extend that.

Mirek

Hi Mirek,

I looked at the tables, and I think I got the idea. But it would save me a lot of time if you could point me what information are present in uni__info elements and also on how are they encoded. Is it some standard way, or is it U++ specific? It looks both space and speed efficient, so I guess it is U++ invention

Also, why is there only first 2048 characters? Do you think the next 20000 in the unicode standard are not really important so it is safe to save space in the resulting binaries by omitting them?

I also looked at the Unicode character database. It seems to contain all the information we need in quite friendly formats. It should not be too difficult to convert those files into something useful.

But the main question I have at this moment is: Should we even try to support all this? Everyone using U++ without all those Unicode characters and properties seemed to be quite happy so far. If it would just bloat the library without any added value, than it is useless, right? In such case, I would be more than happy to implement the functions in JavaScriptCore to the same extent as the rest of U++ and never talk about it again

Honza

---

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.

---

Subject: Re: JavaScriptCore
Posted by dolik.rce on Sat, 24 Jul 2010 13:26:43 GMT
View Forum Message <> Reply to Message

Hello all,

It took me some time, but I think I am getting closer to the goal. I think this could be called beta release

The attached archive should contain all necessary files for JavaScriptCore and three simple example apps (actually two, jsc_test was created just to be compatible with webkits test suite).

I used mozilla test suite included in webkits source tree to test the quality of this U++ port. It introduces 15 regressions, but hopefully nothing really bad. Most of the failed tests can be traced to the suboptimal unicode handling.

I didn't do any tests on windows lately, but I hope it should work there too (or with minor changes).

Known problems:
Heap leaks in webkits code. (Is there some way to disable them for part of the code?)
The regressions in unicode handling as mentioned.
The file jsc.h must be included before Core.h. (Might be possible to fix with some preprocessor magic)

I'm looking forward to your feedback.

To cbporter: Thanks for your summary on unicode. It helped me a lot. I decided to stick to the option 1 for now, supporting the current level of unicode implementation in U++. Getting full support would take too much time, maybe I will do it as a summer project next year...

Best regards,
Honza

File Attachments
1) jsc.zip, downloaded 416 times

---

Subject: Re: JavaScriptCore
Posted by cbpporter on Mon, 26 Jul 2010 09:46:37 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Sat, 24 July 2010 16:26
To cbporter: Thanks for your summary on unicode. It helped me a lot. I decided to stick to the option 1 for now, supporting the current level of unicode implementation in U++. Getting full support would take too much time, maybe I will do it as a summer project next year...

Best regards,
Honza
No problem. Please let me know if you need more information. And with regards of adding Unicode support, as far as I can tell, you only need it for the native function hooks in Java Script. Making a few functions fully Unicode aware (like ToUpper) coupled with input/output methods should do the trick. There is no need to improve U++'s Unicode support. Especially not the GUI part, which is the most time consuming. So basically doing the least amount of work in order for JS to not regress.

---

Subject: Re: JavaScriptCore
Posted by mirek on Thu, 05 Aug 2010 10:55:11 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Sat, 24 July 2010 09:26
Heap leaks in webkits code. (Is there some way to disable them for part of the code?)

Well, we cannot disable leaks, but we have means to disable the checking

```
MemoryIgnoreLeaksBlock __;
```

- any blocks allocated till the end of scope will not be considered leaks if still allocated at the program exit.

Mirek

---

Subject: Re: JavaScriptCore
Posted by dolik.rce on Thu, 05 Aug 2010 12:00:31 GMT
View Forum Message <> Reply to Message

luzr wrote on Thu, 05 August 2010 12:55dolik.rce wrote on Sat, 24 July 2010 09:26
Heap leaks in webkits code. (Is there some way to disable them for part of the code?)

Well, we cannot disable leaks, but we have means to disable the checking

MemoryIgnoreLeaksBlock __;

- any blocks allocated till the end of scope will not be considered leaks if still allocated at the program exit.

Mirek

Well, but wouldn't it be nice if U++ could disable leaks just by piece of code?

Anyway thank you for the hint. Placed in constructor of the wrapper works perfectly. I knew something like this existed, I just couldn't find it...

Honza

---

Subject: Re: JavaScriptCore
Posted by cbpporter on Thu, 12 Aug 2010 07:55:36 GMT
View Forum Message <> Reply to Message

I have investigated further the problem of ToUpper. After looking over the tables we have in U++, they have even better coverage than I thought. Those unit tests that failed must have been extremely thorough to have reported such a failure. So I figured that maybe not the limited coverage of the 2048 characters is the problem, rather maybe the table has some errors.

And indeed, there are some errors.

with stroke".

After some superficial testing, running the current ToUpper/ToLower on the whole 65536 range I have found 568/569 errors. This is really great news, seeing as the table only covers 2048 characters. This means that most of Unicode is case agnostic and we can get away with good support without using huge tables. Running it on only 2048 characters, I have found 50/43. I hope I used the correct testing method.

So if somebody can point me to the bit setup of uni__info, I could correct the values for the first 2048 characters. I can figure out most, but it would be better if I could get the real layout of that packed bitfield. If there are any free bits left, I have some data that I would like to store there, like if the character is punctuation, if it is Latin, etc.

---

Subject: Re: JavaScriptCore
Posted by dolik.rce on Thu, 12 Aug 2010 13:40:36 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Thu, 12 August 2010 09:55After some superficial testing, running the current ToUpper/ToLower on the whole 65536 range I have found 568/569 errors. This is really great news, seeing as the table only covers 2048 characters. This means that most of Unicode is case agnostic and we can get away with good support without using huge tables. Running it on only 2048 characters, I have found 50/43. I hope I used the correct testing method.

Wow. Just wow... I would expect much more errors too.

If I remember correctly, most of the troubles were with characters around 60k. You can run the webkit tests yourself, the jsc_test package (I think I uploaded it last time) is compatible. I will try to remember how I did it and send you the instructions if you want...

Having proper category info in the uni__info would be great too. Maybe even in separate field if it doesn't fit in this one. I just hope someone remembers what is the structure so we can find out.

Honza

---

Subject: Re: JavaScriptCore
Posted by mirek on Fri, 13 Aug 2010 07:09:04 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Thu, 12 August 2010 03:55
After some superficial testing, running the current ToUpper/ToLower on the whole 65536 range I have found 568/569 errors. This is really great news, seeing as the table only covers 2048 characters. This means that most of Unicode is case agnostic and we can get away with good support without using huge tables. Running it on only 2048 characters, I have found 50/43. I hope I used the correct testing method.

So if somebody can point me to the bit setup of uni__info, I could correct the values for the first 2048 characters. I can figure out most, but it would be better if I could get the real layout of that packed bitfield. If there are any free bits left, I have some data that I would like to store there, like if the character is punctuation, if it is Latin, etc.

I would say this defines it pretty well:

```
bool IsLetter(int c)       { return (dword)c < 2048 ? uni__info[c] & 0xc0000000 : 0; }
bool IsUpper(int c)         { return (dword)c < 2048 ? uni__info[c] & 0x40000000 : 0; }
bool IsLower(int c)         { return (dword)c < 2048 ? uni__info[c] & 0x80000000 : 0; }
int  ToUpper(int c)         { return (dword)c < 2048 ? (uni__info[c] >> 11) & 2047 : c; }
int  ToLower(int c)         { return (dword)c < 2048 ? uni__info[c] & 2047 : c; }
int  ToAscii(int c)        { return (dword)c < 2048 ? (uni__info[c] >> 22) & 0x7f : 0; }
```

If bits are 0..31, then (reading the code, please check me...:):

31 lower letter
30 upper letter
22-28 (7 bits) toascii
11-21 (11 bits) toupper
0-10 (11 bits) tolower

Looks like bit 29 is now free, if I am counting well..

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

---

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.

---

Subject: Re: JavaScriptCore
Posted by cbpporter on Fri, 13 Aug 2010 08:48:20 GMT
View Forum Message <> Reply to Message

PS: How did you get the table? I'm autogenerating it from Unicode Database.

---

Subject: Re: JavaScriptCore
Posted by mirek on Fri, 13 Aug 2010 09:30:07 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Fri, 13 August 2010 04:46luzr 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.

Just to be sure what toascii means

---

Subject: Re: JavaScriptCore
Posted by mirek on Fri, 13 Aug 2010 09:30:22 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Fri, 13 August 2010 04:48PS: How did you get the table? I'm autogenerating it from Unicode Database.

The same way.

---

Subject: Re: JavaScriptCore
Posted by cbpporter on Fri, 13 Aug 2010 09:36:55 GMT

View Forum Message <> Reply to Message

luzr wrote on Fri, 13 August 2010 12:30cbpporter wrote on Fri, 13 August 2010 04:46luzr 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.

Just to be sure what toascii means

I don't understand? It is not "E"?

Subject: Re: JavaScriptCore
Posted by cbpporter on Fri, 13 Aug 2010 13:28:55 GMT
View Forum Message <> Reply to Message

There are error with the IsUpper field also.

This is getting quite confusing.

I think that we need to change the meaning of these fields. Take for example the character dot ".". This is a punctuation mark, and as a code point IsLower is false and IsUpper is false. Yet, as a string, IsLower and IsUpper are both true. Unicode defines isLower(b): b == toLower(b). So the string "abc.=/123" is lowercase, and the string "ABC.=/123" is upper case.

If we create:

bool  IsSmthLower(int c)        { return (dword)c < 2048 ? (uni__info[c] & 2047)  == c: true; }

for testing the "logical"/"human interpretable" case of a character in string and keep the old IsLower to refer only to a code point's property of representing a particular cased letter, we can kill two flies with the same stone or how the expression goes. It will also work for the case of a "small a with a superscript capital H in the upper right corner" sequence (which is lowercase). The small a will have a code point property of lower as true, and it's lower case variant is the same. The capital H superscript has it's property as false, more precisely N/A, and it's lower case variant is again the same character.

Subject: Re: JavaScriptCore
Posted by cbpporter on Fri, 13 Aug 2010 14:30:20 GMT
View Forum Message <> Reply to Message

Well here is corrected table, but only with the IsUpper field. There should be exactly 38

differences. I'll recheck the table once more, but everything should be OK. Anyway, nobody noticed that something was wrong before, and I doubt that this correction will impact code for better of worse, unless you are doing an Unicode unit test or live in a very specific part of the world and are using U++ .

I don't have access to SVN right now and I'll be out of town during the weekend so I'm putting it here.

File Attachments
1) b.txt, downloaded 432 times

---

Subject: Re: JavaScriptCore
Posted by jeremy_c on Sat, 14 Aug 2010 19:19:46 GMT
View Forum Message <> Reply to Message

Is JavaScriptCore looking to replace ESC? I'm writing an app now that allows the user to perform certain tasks using ESC.

JavaScript is known by a lot more people than ESC, although it's very easy to pickup on. I would imagine, though, that JavaScript is faster and contains a much larger standard library.

Jeremy

---

Subject: Re: JavaScriptCore
Posted by dolik.rce on Sat, 14 Aug 2010 21:40:03 GMT
View Forum Message <> Reply to Message

jeremy_c wrote on Sat, 14 August 2010 21:19Is JavaScriptCore looking to replace ESC? I'm writing an app now that allows the user to perform certain tasks using ESC.

JavaScript is known by a lot more people than ESC, although it's very easy to pickup on. I would imagine, though, that JavaScript is faster and contains a much larger standard library.

Jeremy
Hi Jeremy,

I wouldn't say replace. I believe they will live together side by side  It is up to you which language you choose. I for example prefer ESC in most cases, especially for it's small size and because you can easily interact with the functions and variables from u++ code. On the other hand, as you say, JavaScript is generally known, and even though ESC syntax is easy, it takes few minutes to learn.

So it is up to you and your needs.

Honza

---

Subject: Re: JavaScriptCore
Posted by cbpporter on Tue, 17 Aug 2010 07:29:24 GMT
View Forum Message <> Reply to Message

IsLower codes fixed, 44 differences.

## File Attachments

1) c.txt, downloaded 397 times

Subject: Re: JavaScriptCore
Posted by cbpporter on Tue, 17 Aug 2010 08:05:40 GMT
View Forum Message <> Reply to Message

I have fixed the 50 ToUpper codes.

Yet, there are still 8 codes that are wrong. I can not fit the correct value in 11 bits. I need at least 14 bits for these 8 values, and maybe more for other codes.

I'll postpone the upload of fixes to ToLower until we see how we'll proceed with these 8 codes. I propose that we use two tables. One with 2048 1 byte values that encode only properties. Another with a dword for lower/upper variants. Maybe we can optimize the storage a little. Anyway, 2KiB extra RAM is not that much.

## File Attachments

1) d.txt, downloaded 478 times

Subject: Re: JavaScriptCore
Posted by dolik.rce on Tue, 17 Aug 2010 20:22:01 GMT
View Forum Message <> Reply to Message

Hi cbporter!

I downloaded your last fix and compiled the jsc test app. I think the results are much better now, but still there is few failures. The attachment contains the relevant results and links to the performed tests. Maybe it will give you some idea where to look. Most errors is in 65K+ region, but two of them are under 2048.

Honza

## File Attachments

1) results.html, downloaded 535 times