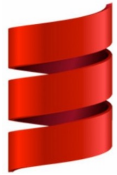
<epam>

Низкоуровневое программирование в браузере - готовимся использовать WebAssembly

May 19, 2016

Vyacheslav Lapin

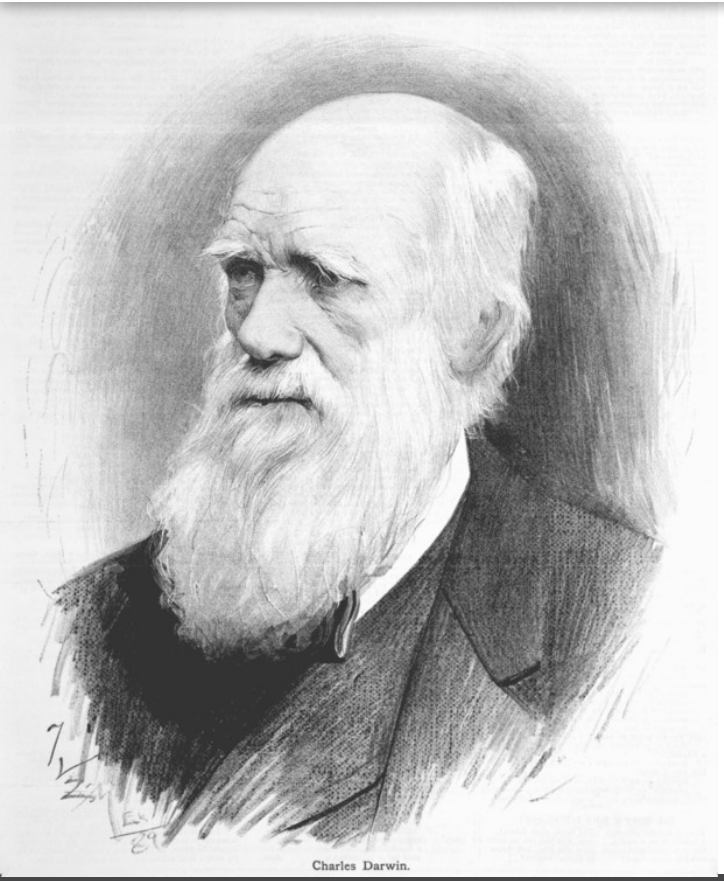
- 10+ years experience in IT
- 7+ Java-development experience
- 5+ trainer experience
- 3+ system analysis



Interests:

- Messaging (Nanomsg, Kafka)
- Functional programming (Clojure, Scala)

Проблема


Charles Darwin.

«Выживает не сильнейший из видов, но тот, что лучше прочих приспосабливается к изменяющейся среде»

Ч. Дарвин

Проблема

- *Скорость загрузки сайта учитывается при ранжировании сайта Google `ом*

Проблема

- *Скорость загрузки сайта учитывается при ранжировании сайта Google `ом*
- *Если взять доход Amazon за прошлый год, то он составил 74,5 долларов. Если верить Линдену, то задержка загрузки страницы на площадке буквально на долю секунда будет стоять компании $745`000`000*

Проблема

- *Скорость загрузки сайта учитывается при ранжировании сайта Google `ом*
- *Если взять доход Amazon за прошлый год, то он составил 74,5 долларов. Если верить Линдену, то задержка загрузки страницы на площадке буквально на долю секунда будет стоять компании $745`000`000*
- *В SaaS нет ни CounterStrike`а, ни 3D Max`а, да и Cloud9IDE сильно не дотягивает по функциональности до WebStorm`а…*

Проблема

- *Скорость загрузки сайта учитывается при ранжировании сайта Google `ом*
- *Если взять доход Amazon за прошлый год, то он составил 74,5 долларов. Если верить Линдену, то задержка загрузки страницы на площадке буквально на долю секунда будет стоять компании $745`000`000*
- *В SaaS нет ни CounterStrike`а, ни 3D Max`а, да и Cloud9IDE сильно не дотягивает по функциональности до WebStorm`а…*
- *IoT (Internet Of Things) устройства слишком просты для парсинга и исполнения JS-кода*

Проблема

- *Скорость загрузки сайта учитывается при ранжировании сайта Google `ом*
- *Если взять доход Amazon за прошлый год, то он составил 74,5 долларов. Если верить Линдену, то задержка загрузки страницы на площадке буквально на долю секунда будет стоять компании $745`000`000*
- *В SaaS нет ни CounterStrike`а, ни 3D Max`а, да и Cloud9IDE сильно не дотягивает по функциональности до WebStorm`а…*
- *IoT (Internet Of Things) устройства слишком просты для парсинга и исполнения JS-кода*
- *4G-сети обладают не достаточной пропускной способностью для сложных данных*

Что можно сделать в рамках JavaScript?

- Сервер должен обработать ответ не дольше, чем за 200 мс.
- Количество переадресаций нужно минимизировать.
- Необходимо уменьшить количество циклов, необходимых для первичной обработки.
- Не используйте внешнюю блокировку JavaScript и CSS для контента в верхней части страницы.
  - «defer»
- Оставьте браузеру время (200 мс) на анализ структуры страницы и ее отображение.
- Оптимизируйте код JavaScript, чтобы он обрабатывался быстрее.
  - http://webo.in

Что можно сделать в рамках JavaScript as compile-target?

- Статическая типизация – TypeScript, JSDoc (Google Closure Compiler)
- Gzip-компрессия

Проблема

Scott Hanselman: JavaScript is Assembly Language for the Web

1. *Sematic Markup is Dead! Clean vs. Machine-coded HTML (6 июля 2011)*
   - *http://www.hanselman.com/blog/JavaScriptIsAssemblyLanguageForTheWebSematicMarkupIsDeadCleanVsMachinecodedHTML.aspx*
2. *Madness or just Insanity? (19 июля 2011)*
   - *http://www.hanselman.com/blog/JavaScriptIsAssemblyLanguageForTheWebPart2MadnessOrJustInsanity.aspx*

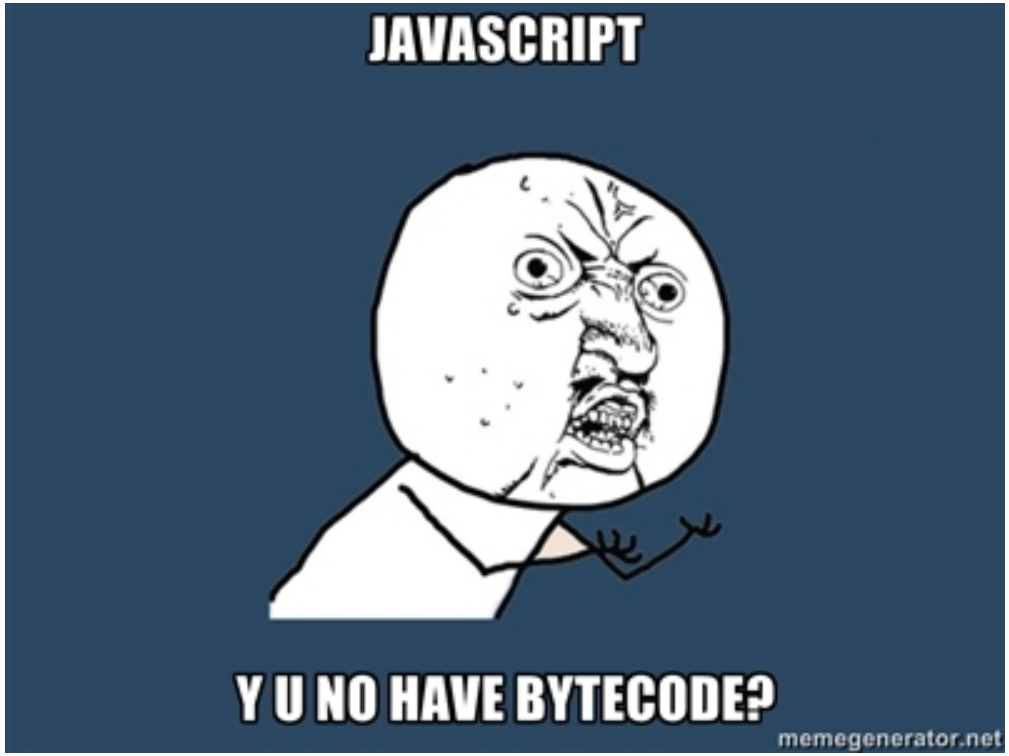
I was talking to Erik Meijer yesterday and he said:

JavaScript is an assembly language. The JavaScript + HTML generate is like a .NET assembly. The browser can execute it, but no human should really care what's there. - Erik Meijer

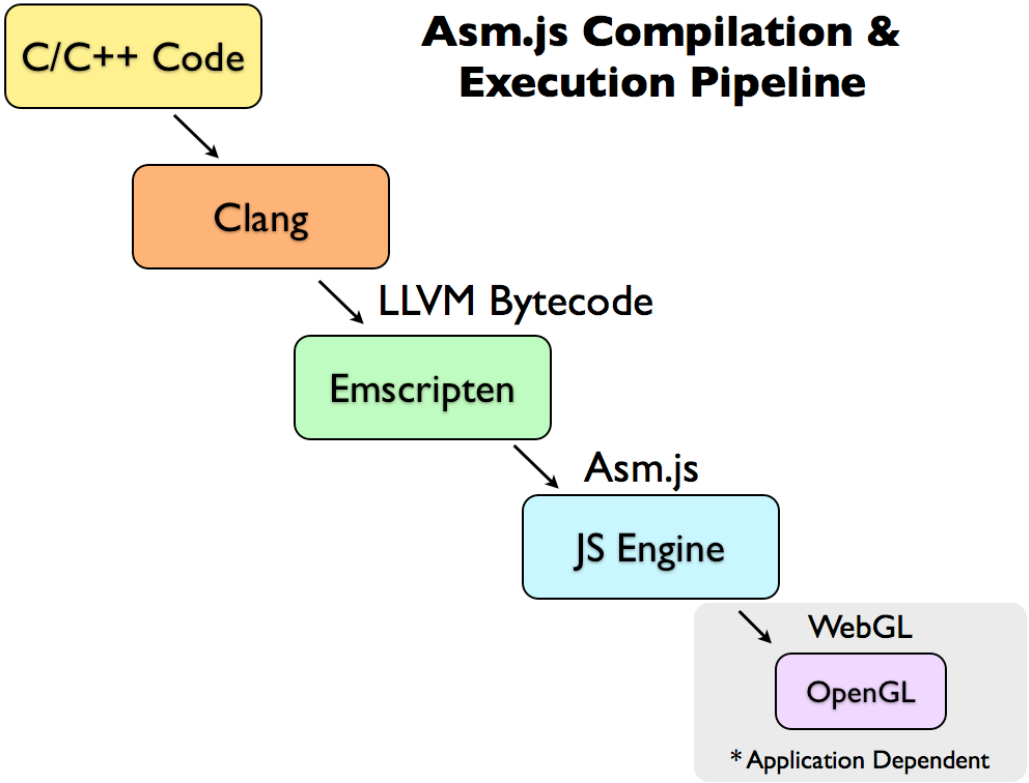
Проблема

```
<!DOCTYPE html><html lang="en" dir="ltr"><head><meta http-equiv="X-UA-Compatible" content="IE=9, IE=8, chrome=1"><base
href="https://plus.google.com/u/0/"><script>(function(){function a(c){this.t={};this.tick=function(c,e,b){b=b!=void 0?b:(new
Date).getTime();this.t[c]=[b,e]};this.tick("start",null,c)}var d=new a;window.jstiming={Timer:a,load:d};try{var
f=null;window.chrome&&window.chrome.csi&&(f=Math.floor(window.chrome.csi().pageT));f==null&&window.gtbExternal&&
(f=window.gtbExternal.pageT());f==null&&window.external&&(f=window.external.pageT);f&&(window.jstiming.pt=f)}catch(g){};})();
var loadTimer = window.jstiming.load; loadTimer.name = 'profload_streamv2_prod';var OZ_buildLabel = 'es_oz_b701.02_p6';var OZ_pathPrefix =
'\/u\/0'; var OZ_jsVersion = 'w6ZI0gslhvA.en.'; var OZ_windowName = 'oz'; var OZ_lang = 'en';var OZ_initWidget = 1;var e=this;var
f=RegExp("^(?:([^:/?#.]+):)?(?://(?:([^/?#]*)@)?([\\w\\d\\-\\u0100-\\uffff.%]*)(?::([0-9]+))?)?([^?#]+)?(?:\\?([^#]*))?(?:#(.*))?
$"),j=function(){for(var a=g,d=h,b=i;(b=a.indexOf("gpcaz",b))>=0&&b<d;){var c=a.charCodeAt(b-
1);if(c==38||c==63)if(c=a.charCodeAt(b+5),!c||c==61||c==38||c==35)return b;b+=6}return-1},k=/#|$/,l=/[?&](|$|#)/;var m,n=function(){var
a=window;return a.history&&a.history.pushState},o=function(){var a=top.location.href.match(f),d=a[7]||"";if(!d)return null;for(var b=
["","buzz","about","contact","sidewiki"],c=0;c<b.length;c++)if(d==b[c])return null;n()?(b=a[4]||"",c=a[6]||"",a=(a[1]||"")+"://"+
(a[3]||""),b&&(a+=":"+b),a+=e.OZ_pathPrefix+"/"+d,c&&(a+="?"+c),d=a):d=null;return d},p="OZ_prog".split("."),q=e;!(p[0]in
q)&&q.execScript&&q.execScript("var "+p[0]);
for(var r;p.length&&(r=p.shift());)!p.length?q[r]=0:q=q[r]?q[r]:q[r]={};var s=function(){var a=window;if(!a._jsl&&0<=m){var
d=document.getElementById("js").src;(new Image).src="/_/resourcefailure?
type=js&url="+encodeURIComponent(d)+"&hl="+encodeURIComponent(a.OZ_lang)}else m=0,a.setTimeout(s,3E4)};
if(n()){for(var t=window,u,g=t.location.toString(),h=g.search(k),i=0,v,w=[];
(v=j())>=0;)w.push(g.substring(i,v)),i=Math.min(g.indexOf("&",v)+1||h,h);w.push(g.substr(i));u=w.join("").replace(l,"$1");t.history.replaceSt
ate(null,t.document.title||"",u)}var x=(window!=top?"about:blank":null)||o();if(x)window.location=x;var
y=e.OZ_pathPrefix,z=top.location.href.match(f),A=z[5]||"",B=z[7]||"";
if(B&&(y&&A.indexOf(y)==0&&(A=A.substring(y.length)),A.indexOf("/")==0&&(A=A.substring(1)),A!=B)){var
C=document.createElement("style");C.type="text/css";C.styleSheet?C.styleSheet.cssText=".maybe-hide {visibility:hidden}":C.innerHTML=".maybe-
hide {visibility:hidden}";document.getElementsByTagName("head")[0].appendChild(C)}
window.setTimeout(function(){for(var a=!1,d=0;d<document.styleSheets.length;d++){var
b=document.styleSheets[d],c=b.href;if(c&&c.indexOf("/_/ss/")>-1)
{try{if(b.cssRules&&b.cssRules.length>0||b.rules&&b.rules.length>0)a=!0}catch(D){}break}}if(!a)a=document.getElementById("ss").href,d=window,
(new Image).src="/_/resourcefailure?type=css&url="+encodeURIComponent(a)+"&hl="+encodeURIComponent(d.OZ_lang);s()},3E4);
</script><title>Scott Hanselman - Google+</title><script>loadTimer.tick('vl');</script><style type="text/css">.esw {background-repeat: no-
repeat; border: 0; cursor: pointer; display: inline; height: 15px; overflow: hidden; width: 24px; border-style: inset; -webkit-box-align:
center; -webkit-appearance: button;}.eswd {background: url(//ssl.gstatic.com/s2/oz/images/stars/po/SRP/ploffhover2.gif);}.eswd:hover
{background: url(//ssl.gstatic.com/s2/oz/images/stars/po/SRP/ploffclick2.gif);}.eswd:active {background:
url(//ssl.gstatic.com/s2/oz/images/stars/po/SRP/plonclick2.gif);}.eswa {background:
url(//ssl.gstatic.com/s2/oz/images/stars/po/SRP/plon2.gif);}.eswa:active {background:
url(//ssl.gstatic.com/s2/oz/images/stars/po/SRP/plonclick2.gif);}.eswe {background:
url(//ssl.gstatic.com/s2/oz/images/stars/po/SRP/plerr2.gif);}.esww {background: url(//ssl.gstatic.com/s2/oz/images/stars/po/SRP/p1wkng2.gif);
cursor: default;}</style><style>
        body {
          visibility: hidden;
        }
      </style><link rel="stylesheet" id="ss" href="/_/apps-static/_/ss/home/ver=dtstw83raazl/am=!TR7zzHH9Ng5x8tLkLAvbue5MC-
oDoO39c8QK2HM/bf=BA/r=O"><style>#gb{font:13px/27px Arial,sans-serif;height:30px}#gbz,#gbg{position:absolute;white-
space:nowrap;top:0;height:30px;z-index:1000}#gbz{left:0;padding-left:4px}#gbg{right:0;padding-
```

Проблема



ASM.js


Asm.js Compilation & Execution Pipeline
C/C++ Code
Clang
LLVM Bytecode
Emscripten
Asm.js
JS Engine
WebGL
OpenGL
*Application Dependent

ASM.js

Подмножество JS со статической типизацией.

- *Поддерживается в FF 22+*
- *Edge*
- *V8 (частично)*

```
// C
int increment(int i) {
    return 1 + i;
}
```

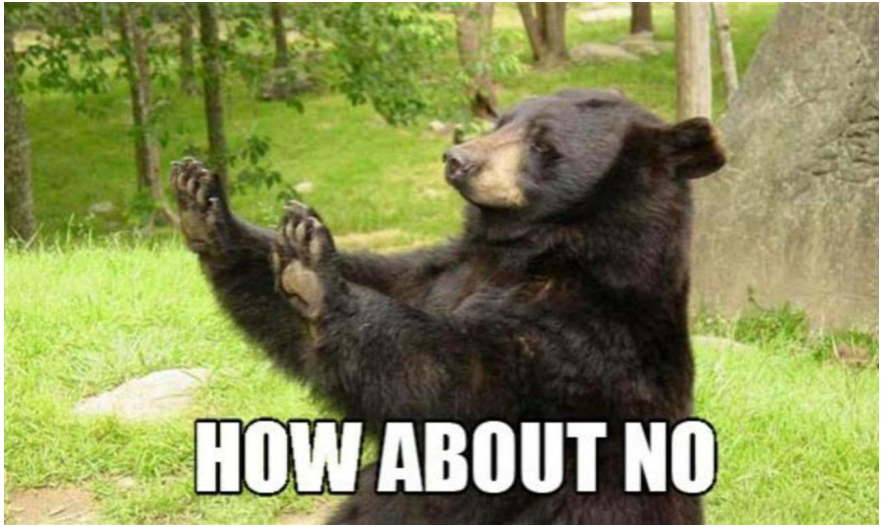
```
// asm.js
function increment(i) {
   i = i|0;
    return (1 + i)|0;
}
```

ASM.js


!function(window){"use strict";var ihex,NTP=25,buffer=ne
(8388608),series=function(stdlib,env,heap){"use asm";fur
+p,ak=+ak;var i=0,j=0,p1=0,pt=0,r=0;if(i=0|i,j=0|j,ntp=0
(tp1=1,tp[0]=1,i=1;(0|ntp)>(0|i);i=0|(0|i)+1)tp[i<<3>>3]
(1==ak)return 0;for(i=0;(0|ntp)>(0|i)&&!(+tp[i<<3>>3]>p)
[i-1<<3>>3],p1=+p,r=1,j=0;(0|i)>=(0|j);j=0|(0|j)+1)p1>=p
ak)*ak,p1-=pt),pt=.5*pt,pt>=1&&(r*=r,r-=+floor(r/ak)*ak)
series(m,id){m=0|m,id=0|id;var k=0,ak=0,eps=0,p=0,s=0,t=
id)>(0|k);k=0|(0|k)+1)ak=8*+(0|k)+ +(0|m),p=+(0|id)-+(0|
ak,s-=floor(s);for(k=0|id;(0|id+100)>=(0|k)&&(ak=8*+(0|k
id)-+(0|k))/+ak,!(eps>t));k=0|(0|k)+1)s+=t,s-=floor(s);r
floor=stdlib.Math.floor,pow=stdlib.Math.pow,tp1=0,tp=new
(heap),ntp=0|env.NTP;return series}
({Uint8Array:Uint8Array,Int8Array:Int8Array,Uint16Array:
{NTP:NTP},buffer);ihex=function(x,nhx,chx){var i,y,hx="0
(y=Math.abs(x),i=0;nhx>i;i++,y=16*(y-(0|y)),chx[i]=hx[0|
(id){var pid,s1,s2,s3,s4,hex=[];return s1=series(1,id),s
(5,id),s4=series(6,id),pid=4*s1-2*s2-s3-s4,pid=pid-(0|pi
{hex:hex.join("").substr(0,10),fraction:pid}}}(window);
{ASM.JS}

Минусы asm.js

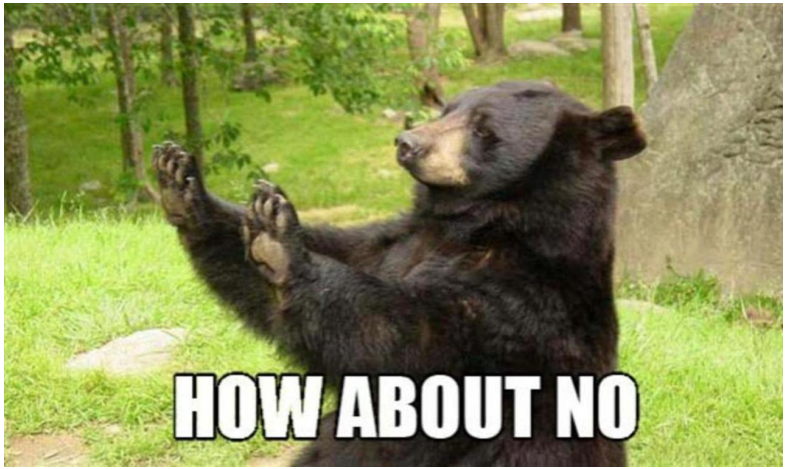
- Нет сборщика мусора
- Потоков
- Разделяемой памяти
- Долгий ресурсоёмкий парсинг
- Конечный код не оптимального размера
- Сложно писать вручную



PNaCL

Google предложил альтернативу на основе NaCL

- Наитвный код (LLVM)
  - получается при помощи AOT
- Исполняется в песочнице браузера
- Добавляет ещё один уровень абстракции



- Минус – изоляция в рамках собственной VM, отсутствие доступа к DOM и JS
  - (похож на worker`ов)

WASM - High-Level Goals

- Компактный бинарный формат для компьютеров, планшетов, мобильников и IoT
- Прозрачное преобразование в человеко-читаемый код и обратно.
- Обратная совместимость (polyfill-lib) для не поддерживающих браузеров
- Трассировки
- В перспективе – наоборот, реализация JS на wasm для обратной совместимости с JS-кодом (и не только)

WASM

- **Minimum Viable Product (**MVP)
  - Модули
  - AST
  - Бинарный формат
  - Текстовая проекция
  - «не только браузеры»
  - Polyfill в JavaScript

WASM

- Модули
  - Imports/exports
  - стартовый метод*
  - Линейная память
  - Код
  - Информация для дебага
  - *В будущем возможно добавление дополнительных секций*

**Опционально*

WASM – линейная память

- memory_size
- grow_memory
- load
- store

WASM – каноническое представление – AST (s-expressions)

```
(i32.add
    (set_local $x (i32.const 1))
     (set_local $x (i32.const 1)))
```

WASM – Типы

- 32 – 32-bit integer
- i64 – 64-bit integer
- f32 – 32-bit floating-point
- f64 – 64-bit floating-point (JS Number)

Принцип работы - как с Generics`ами: какими кладём, такими и вынимаем!

WASM – Управляющие конструкции

- block
- loop
- if, if_else
- br, br_if
- tableswitch, case
- return

WASM

- **PostMVP**
  - Thread’ы
  - Динамическое связывание
  - Fixed-width SIMD
  - Zero-cost Exception Handling
  - Feature Testing
    - has_feature

WASM

- **Post-Post-MVP**
  - GC
  - JIT
  - А-ля Unsafe

WASM

Статья “Experimental support for WebAssembly in V8” (15 марта 2016)
Статья “A WebAssembly Milestone: Experimental Support in Multiple Browsers” (14 марта 2016)
Статья “Previewing WebAssembly experiments in Microsoft Edge” (15 марта 2016)

Демо :

Demo	Binary	time to decode into asm.js
AngryBots (https://lukewagner.github.io/AngryBotsPacked)	6.3MiB	240ms
PlatformerGame (https://lukewagner.github.io/PlatformerGamePacked)	18MiB	550ms

WASM - HelloWorld

- Prerequisites:
  - ✓ Git
  - ✓ Make
  - ✓ Cmake
  - ✓ Python 2.x

WASM – HelloWorld (Linux)

- Download and install Emscripten (incoming-brunch):
  - ✓ https://s3.amazonaws.com/mozilla-games/emscripten/releases/emsdk-portable.tar.gz
  - ✓ Update, install and activate:

```
$ cd path/to/emsdk_portable
$ ./emsdk update
$ ./emsdk install sdk-incoming-64bit
$ ./emsdk activate sdk-incoming-64bit
```

WASM - HelloWorld

- Building Binaryen and sexpr-wasm:
  - ✓ binaryen:

```
$ git clone https://github.com/WebAssembly/binaryen
$ cd binaryen
$ cmake . && make
```

  - ✓ sexpr-wasm:

```
$ git clone https://github.com/WebAssembly/sexpr-wasm-prototype
$ cd sexpr-wasm-prototype
$ git submodule update --init
$ make
```

WASM - HelloWorld

- Write HelloWorld source in C++:
  - ✓ hello.cpp:

```
#include <iostream>

int main() {
  std::cout << "Hello World\n";
  return 0;
}
```

WASM - HelloWorld

- Building the app:

```
$ emcc -O2 hello.cpp -o hello.html -s BINARYEN=1

$ path/to/sexpr-wasm-prototype/out/sexpr-wasm hello.wast -o hello.wasm

$ cp hello.wast.mappedGlobals hello.wasm.mappedGlobals
```

Ссылки на материалы

- https://github.com/kripken/emscripten/wiki/WebAssembly
- https://github.com/WebAssembly/binaryen
- https://github.com/WebAssembly/sexpr-wasm-prototype
- https://github.com/kazuki/mediacodec.wasm
- http://webassembly.github.io/demo/

- WebAssembly "Hello World" in V8 - https://gist.github.com/s3ththompson/7b58b492b02703a0424b

- Binaryen - https://github.com/WebAssembly/binaryen

THANK YOU

CONTACT ME

selavy

Vyacheslav Lapin

Vseslavur