Bck2Brwsr

From APIDesign

Bck2Brwsr VM is a Java virtual machine which is capable to take bytecode and convert it (either ahead-of-time e - e.g. during compilation - or just-in-time, e.g. in a browser) into appropriate JavaScript code which does the same thing. As a result one can write in Java, compile with JavaC, process it with **Bck2Brwsr** (usually via its Maven plugin, but possibly also directly via programmatic APIs) into a JavaScript file(s) that can be executed in any modern browser (IE 10 at least, please).

The easiest way to start is to play with a DEW - the Development Environment for Web. Just visit http://dew.apidesign.org/dew and edit, compile, run in a browser without installing anything or using (now banned) Java plugins.

The **Bck2Brwsr** project supports the generic and portable HTML/Java APIs as defined by NetBeans. Use them to run your application on desktop as a standalone application, as a NetBeans or Eclipse plugin or in a browser, via **Bck2Brwsr** VM.

Contents

Motivation

During my duties at JavaOne2012 my feeling (primed by observing various RSS feeds full of posts about something which ends **.js**) that Java is no longer as cool as it used to be straightened. It always starts by loosing interest of newcomers, targeting just a piece of the technology segment and soon once Good Technology becomes new COBOL. Can that happen to Java? Just read Is JavaScript the Future of Programming? (http://mashable.com/2012/11/12/javascript/) and (if you like Java), you'll get scared. I decided to do something about that. I'd like to return my favorite programming language **Bck2Brwsr**.

Licensing

The code of **Bck2Brwsr** is heavily derived from OpenJDK source code (especially the libraries part, not the HotSpot part). **Bck2Brwsr** thus needs to match the OpenJDK licensing and be compatible with it. The licensing goal is however the same as in case of OpenJDK:

- if you write your application and execute it against standard JDK libraries - use the classpath exception
- if you hack the **Bck2Brwsr** VM itself - follow the GPLv2 license

This kind of setup should give Java developers freedom to build their applications and distribute them under conditions of their choose, while making sure modifications, enhancements and improvements of the **Bck2Brwsr** VM code itself remain publicly available.

Goals

Create **small** Java (check the Bck2BrwsrJavadoc) capable to boot fast and run in 100% of modern browsers including those that have no special support for Java.

Demonstrate that Java has benefits over JavaScript when creating larger HTML5 applications.

Unlike other similar efforts, the goal of this project is **not** to execute any existing Java library. It is expected that libraries for the new, limited environment need to be specially designed. For example one cannot start new threads (as of Bck2Brwsr 0.13 even there is a proposal to do something with it).

Demo

The current flagship demo is MineSweeper running in the browser: http://xelfi.cz/minesweeper/bck2brwsr/

As a memento it makes sense to keep in mind the little calculator demo (http://xelfi.cz/calc) showing the first application that I managed to get running in the browser. The application is packaged in a JAR file and the **Bck2Brwsr** VM loads necessary classes and transforms them into JavaScript on the fly.

There is/was (relatively) enhanced Twitter demo showing various technologies including JSONP communication with the server. However as Twitter abandoned its authorization free API in May 2013, the communication with the server is broken.

For JavaOne2013 I developed a chess application (http://xelfi.cz/JavaOneChess) showing use of knockout.js from a Java code. See Knockout4Java.

To demonstrate my geolocation API, I created another Where are you? (http://xelfi.cz/kde/) demo.

We also managed to port JavaFX to **Bck2Brwsr** and here is a proof of that (http://xelfi.cz/fishsim) . There is an overhead to load JavaFX scene graph API, so excuse slight delay when downloading and booting the app. When finally running, the performance should be OK.

Try It Yourself!

It is simple to try **Bck2Brwsr**! The project skeleton is provided as Maven archetype. Follow steps described at Bck2BrwsrViaCLI or Knockout4Java tutorial.

What is in New Releases?

Next Version 0.23 Will ...

After few months the newest version of **Bck2Brwsr** VM is here. It contains:

- Fast emulation of **long** numbers based on scala.js one (https://github.com/jtulach/bck2brwsr/pull/7) - thanks Sébastien!
- Gradle tasks - see howto (https://github.com/jtulach/bck2brwsr/blob/master/docs/GRADLE.md)
- New maven tutorial (https://github.com/jtulach/bck2brwsr/blob/master/docs/MAVEN.md)

It was quite a fun to write Gradle tasks together with Maven and build them by Maven.

Get started with README (https://github.com/jtulach/bck2brwsr/#readme) . Enjoy.

Version 0.22

- Support for Apache HTML/Java API version 1.5.1
- Exceptions capture stacktrace via **Error.stack** when created (if the **stack** field is supported)

Version 0.21

Contains necessary bugfixes to run Dew with most recent versions of the used projects:

- Internationalization issues fixed
- Proxies now implement even super interfaces
- Caching a miss when running tests
- Use of shl64 fixed.
- Update to JDK8_144

List of changes is here (https://github.com/jtulach/bck2brwsr/compare/release-0.20...release-0.21)

Version 0.20

This version of **Bck2Brwsr** VM is called **Radtouren 2017** version, as it has been prepared and released while our gang was bicycling in Korutany. Sleeping in a tent in camps, bicycling whole day, coding in a morning. What can be more fun?

- Support for Html/Java API (http://bits.netbeans.org/html+java/) version 1.4
- Build & tests succeed on GraalVM
- Few bugfixes
  - Support for **NOP** instruction
  - Can execute some Kotlin code
- Implementation of ClassValue (https://docs.oracle.com/javase/8/docs/api/java/lang/ClassValue.html)

Wanna transpile Java to JavaScript. Give **Bck2Brwsr** a try!

Version 0.19

Bck2Brwsr 0.19 comes with many little improvements and one new feature: it can execute JUnit in the browser!

- Better support for Annotation (https://docs.oracle.com/javase/8/docs/api/java/lang//Annotation.html) - so good that bck2brwsr can now execute JUnit tests
- Support for all classes from *java.util.concurrent* package, so JUnit runs without problems
- Switching to version *2.1.0* for *retrolamda*
- More robust generated JavaScript file - doesn't override already defined classes
- Uses version 1.3 of netbeans:Html4Java API.
- Record name of an OSGi bundle to be used when Maven coordinates are missing like in JUnit case
- Don't convert Date (https://docs.oracle.com/javase/8/docs/api/java/util/Date.html) objects to current time millis
- Use **boolean**.valueOf() to simplify mixing of boxed and unboxed booleans
- Perform more conversions before returning Java value to JavaScript from a Java callback
- **Launcher.createBrowser** can specify its own page
- Make sure class cast exception contains the same message as produced by JDK
- Convert undefined value read from arrays into **null**
- Convert content of array properly before entering JavaScript
- Turning the bck2brwsr VM into an OSGi bundle
- Introducing **vmtest.precompiled**=<regexp> property to verify that bck2brwsr generated resources are really used
- Don't include VM in **VMTest** initialization, let it be loaded on demand
- **System.exit** terminates associated launcher
- eval(JavaScriptResource (http://bits.netbeans.org/html+java/1.1/net/java/html/js/JavaScriptResource.html)) as string to prevent double obfuscation
- Don't report warnings when generating minified version of bck2brwsr libraries

Get the bits from the Maven central repository (http://repo1.maven.org/maven2/org/apidesign/bck2brwsr/rt/0.19/) !

Version 0.17

Bck2Brwsr version 0.17 is faster. Ten years ago nobody would imagine dynamic languages could get as good performance as they have now. The feeling that JavaScript just can't be fast is presumably present in many of our souls. The truth is, it can be relatively fast - not as fast as Java as my experiment with Sieve of Eratosthenes (https://github.com/jtulach/sieve) shows, but pretty damn fast. Certainly not an excuse to be ten times slower than HotSpot (which was the previous state of **Bck2Brwsr**).

The daily work on Truffle compiler team and the time I got when traveling from *Snowcamp* at Grenoble gave me a chance to speed **Bck2Brwsr** up. The sieve (https://github.com/jtulach/sieve) being a nice - e.g. small and focused - benchmark. Originally the algorithm couldn't be finished in a reasonable time when running on old version of **Bck2Brwsr**, but knowing what optimizing compilers seek for, it was relatively easy to speed it up ten times.

With great pleasure I announce that **Bck2Brwsr**, the most complete Java VM in browser (that can run Javac as shown by Dew project) has been sped up many times being at most three times slower than HotSpot. Given the primary goal of **Bck2Brwsr** is modularity and not speed, I consider it a good sped up even knowing there is a room to make it even faster.

Enjoy the Bck2Brwsr 0.17's speed (http://xelfi.cz/minesweeper/bck2brwsr/) !

Version 0.16

Bck2Brwsr now bundles precompiled Bck2BrwsrLibraries for HTML/Java API version 1.2.3. Better handling of fully qualified names that contain an underscore. Reflection to non-static methods passes parameters the *right way*.

Release 0.14 and 0.15

Bugfix 5930 (https://java.net/bugzilla/show_bug.cgi?id=5930) . Implemented StrictMath (https://docs.oracle.com/javase/8/docs/api/java/lang/StrictMath.html) to delegate to Math (https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html) . Locale (https://docs.oracle.com/javase/8/docs/api/java/util/Locale.html) reads real user locale (http://source.apidesign.org/hg/bck2brwsr/rev/681e61714a1d) from surrounding browser. A bit more optimized for execution on Chrome. Fixes a VM bug (http://source.apidesign.org/hg/bck2brwsr/rev/3c23f0cebd32) so JBox2D Physics Engine (http://www.jbox2d.org/) executes without errors. Support for signed JARs.

The original plan to run RxJava in **Bck2Brwsr** VM (demo available at http://xelfi.cz/RxJava/0.1/) has not materialized. I got upset as the RxJava guys decided to not accept my pull request (https://github.com/ReactiveX/RxJava/pull/2643) for artificial reasons. Looks like they don't have much clue about modularity.

Release 0.13

Release 0.13 of **Bck2Brwsr** VM is mostly a bugfix release. It fixes problem in Collections (https://docs.oracle.com/javase/8/docs/api/java/util/Collections.html) .shuffle caused by wrong **shr64** implementation (http://source.apidesign.org/hg/bck2brwsr/rev/d2a5a7a0e167) (which was basically a single character fix, but it took me an hour to find it) and library generating problem (http://source.apidesign.org/hg/bck2brwsr/rev/6e50103c0f1c) on computers using different encoding than UTF-8.

The **Bck2Brwsr** VM 0.13 is good enough to power another classical game: fifteen (http://dukescript.com/game/fifteen.html) .

The released bits are @ maven.java.net (https://maven.java.net/content/repositories/releases/) repository as of Jan 12, 2015.

Release 0.12

Libraries can be pre-compiled and published as Maven artefacts (see Bck2BrwsrLibraries how to). The Knockout4Java Maven archetype has been modified to use the precompiled version of **Bck2Brwsr** rt. jar emulation library and HTML/Java APIs:

```
$ mvn archetype:generate \
  -DarchetypeGroupId=org.apidesign.html \
  -DarchetypeArtifactId=knockout4j-archetype \
  -DarchetypeVersion=1.1.2 \
  -Dbck2brwsr=true
# answer few questions...
$ cd nameofyourproject
# run on desktop
$ mvn process-classes exec:java
# run in a browser
$ mvn -Pbck2brwsr clean package bck2brwsr:show
```

Supporting Bck2BrwsrBlobURLs so one can display images available as in JAR resources.

Release 0.11

The new version fixes problems with obfuscation mode (a regression in Bck2Brwsr 0.10). Now the final application can be **FULL**y obfuscated and the sample *org.apidesign.html:knockout4j-archetype:1.0* seems to work with version 0.11. System has *nanoTime* method (per requests of Toni Epple). Iterating through JavaScript or Java array

```
var array = [1, 3, 5]
for (var i in array) {
  console.log(i);
}
```

shows only *0, 1, 2*. All additional **Bck2Brwsr** functions are added as non-enumerable.

Release 0.10

The ahead-of-time mode has support for JDK8's Lambdas (thanks to RetroLambda project). Following example properly returns "XXXXXXXXXX" in Bck2Brwsr 0.10 when **compound** methods is called:

```
private static void fewTimes(Runnable r, int cnt) {
    while (cnt-- > 0) {
        r.run();
    }
}

public static String compound() {
    StringBuilder sb = new StringBuilder();
    fewTimes(() -> sb.append('X'), 10);
    return sb.toString();
}
```

My experience from implementing lambdas in **Bck2Brwsr** VM was so horrible that I had to express my hate of invokeDynamic in a dedicated essay. Such instruction should have never be added into the JVM specification! See the *Lambdas Go **Bck2Brwsr*** video:

Support for JDK8 defender and interface static methods. In the following example the method **defaultValue** properly returns *42* in Bck2Brwsr 0.10:

```
public interface Value {
    public static int staticValue(Value v) {
        return v.value();
    }

    public default int value() {
        return 42;
    }

    public static int defaultValue() {
        return staticValue(new Value() {});
    }
}
```

The support for lamdas does not mean Bck2Brwsr 0.10 supports JDK8 APIs. It does not. The libraries are still subset of JDK7 - one can use lamdas only in own code so far. Technically it should not be a problem to backport JDK8, libraries - it just has not been done yet.

Using Object.defineProperty to make sure the JavaScript Object has all the methods of Object (https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html) , while those methods do not show up during iteration

```
for (var p in anObject) {
  console.log('A prop found ' + p);
}
```

Release 0.9

Version 0.9 eliminates useless stack assignments. Instead of doing

```
var stI0 = lcI0;
var stI1 = lcI1;
var stI0 = stI0 + stI1;
return stI0;
```

the now generated code is

```
return lcI0 + lcI1;
```

which is shorter and more human readable. However I doubt the V8 virtual machine sees any benefits - I think the final native code remains the same. But at least the debugging of the generated JavaScript code is now easier - there is less *Step Over* invocations and it mimics more closely the original Java source.

Optimized the ahead-of-time compilation, so now the http://xelfi.cz/minesweeper/bck2brwsr/ demo starts up instantly. I had to do it, because it was so embarrassing to see TeaVM to boot the same application so quickly: The initial delay is gone, and moreover it downloads necessary libraries in parallel and on background. Now we are ready for next step: share the libraries between different applications.

Can ObfuscatePerLibrary - e.g. each JAR gets compiled ahead-of-time into its own JavaScript file, which can be shared between many applications.

Release 0.8.1

- Supports latest netbeans:Html4Java revision 0.7.5
- Now we can run JavaC in the browser.
- Many new JDK classes implemented - see at github (https://github.com/jtulach/bck2brwsr/compare/release-0.8...release-0.8.1)

Release 0.8

- this is the *JavaOne2013 version*
- comes with nice NetBeans IDE integration, see plugin portal page (http://plugins.netbeans.org/plugin/50521/) .
- sound API
- geolocation API
- Support for smooth communication over WebSocket protocol

Release 0.7.2

- Donated our JSON4Jersey mappings to Jersey project (pull request (https://github.com/jersey/jersey/pull/16) has been accepted) for Jersey 2.1
- Support for different HTTP methods (like *PUT*, *PULL*, etc.)
- Knockout4Java Archetype for dual profile project (FXBrwsr as well as **Bck2Brwsr**)
- Better support for Boolean (https://docs.oracle.com/javase/8/docs/api/java/lang/Boolean.html) mapping in **Bck2Brwsr** VM
- FXBrwsr launcher stops the HTTP server when the FXBrwsr window is closed

Release 0.7

- FXBrwsr with full debugging support and a demo
- Dual Twitter demo - single source code, dual deployment (watch the same demo)
- Lightweight, generic JSON <-> Java mapping: javadoc (http://bck2brwsr.apidesign.org/javadoc/net.java.html.json/)
- ProviderAPI and a TCK to bind to other technologies than **Bck2Brwsr**, Knockout.js and FXBrwsr: javadoc (http://hudson.apidesign.org/hudson/job/bck2brwsr-net.java.html.json/lastSuccessfulBuild/artifact/json-tck/target/site/apidocs/index.html)
- Follow naming convention -- now *bck2brwsr-maven-plugin* -- thanks to Miloš Kleint

Release 0.6

- Full featured demo Twttr demo (http://xelfi.cz/twttr)
- **Bck2Brwsr** provides better binding of complex classes (defined by a special **@Model**) annotation
- The **@Model** classes can be obtained from a server via JSON and JSONP. Use **@OnReceive** annotation
- Browser testing harness has nicer output with UL and expandable LI

Release 0.5

- **Bck2Brwsr** 0.5 has better support for MVVC via Knockout.js - see the calculator demo (http://xelfi.cz/bck2brwsr-0.5/) version 0.5
  - Binds String and primitive types
  - Bind array types (exposed as List (https://docs.oracle.com/javase/8/docs/api/java/util/List.html)
  - Basic binding of complex classes
- Separate module for Maven archetype called *org.apidesign.bck2brwsr:bck2brwsr-archetype-html-sample* (and thus instructions for getting Bck2BrwsrViaCLI has changed)
- Improved speed of **Bck2Brwsr** virtual machine via better control flow
- Can use Closure compiler to generate more compact code
  - FULL mode: For batch compilation of everything for now (example pom.xml

    - (http://source.apidesign.org/hg/bck2brwsr/file/151f4ccd7673/javaquery/demo-calculator/pom.xml) that uses the **j2js** goal)
    - MINIMAL mode: Strips spaces. Works in dynamic mode (part of the default Maven archetype)
- One incompatible change: AnnotationProcessor for the **@Page** annotation no longer capitalizes field names found in the HTML page. This was meaningful when the fields were static constants. Now (when they are plain instance fields) it makes little sense.

Release 0.4

0.4 is the first release we managed to upload to java.net Maven repository. Heuréka! See Bck2BrwsrViaCLI for simple three steps towards using this version of **Bck2Brwsr**.

What you can expect? It works. It is sometimes not fast. It is sometimes broken (what would you expect after four months of development?). Errata:

- When you create new project and want to use it in NetBeans, you need to update nbactions.xml file to refer to *0.4* version, rather than *0.3-SNAPSHOT*.

Here is list of achievements for the *0.4* version:

- Throwing and catching exceptions by *Tomáš Z.*, finally block by me.
- Support for converting ByteCode in the browser
- Speed via register based system - *Ľubomír* finished first version of his register based rewrite (http://hudson.apidesign.org/hudson/job/bck2brwsr.registers/) on Dec 14, 2012.
- Speed benchmark and infrastructure to measure it in various environments - *Martin Š.*
    - Run with *-Dvmtest.brwsrs=firefox,chromium-browser* (or any other browsers you want to test)
- Maven archetype for creating the calculator like demos
- Int32, Int16, Int8 arithmetics done by Martin Š.
- API for drawing on the canvas: Thanks Toni! Read about his experience (http://jayskills.com/blog/2013/01/22/canvas-for-bck2brwsr/) using **Bck2Brwsr**.
- More precise int64 support - *Martin Š.* working on
- Convertor from GWT's native code to **Bck2Brwsr**'s *@JavaScriptBody* - is sort of there, but not really functional.
- Fields of same name in subclasses. Thanks to Bck2BrwsrMangling.
- Compatibility tests can be written with help of **@Compare** annotation
- Basic reflection support (e.g. **Bck2Brwsr** throws SecurityException (https://docs.oracle.com/javase/8/docs/api/java/lang/SecurityException.html) when allowed),
    - **Done**: Class.newInstance() works.
    - **Done**: Class.getMethods() works (returns only public methods)
    - **Done**: Annotations of classes and methods
- Support for MVVC like Knockout.js that binds String and primitive types
- Packages into a static website via JAR files (which then take long time to inflate)
- Implements *java.util.zip* APIs

TODO

Although the system is capable to run and execute trivial applications, there remains tons of things to improve and fix. Any help is welcomed. Just let me know if something interests you:

- Access to multipage via *sammy.js* or crossroads.js (http://millermedeiros.github.io/crossroads.js/)
- Method and field overriding with various modifiers
- More reflection support (e.g. don't throw SecurityException (https://docs.oracle.com/javase/8/docs/api/java/lang/SecurityException.html) when allowed),
    - No private method/field/constructor/class access
    - Probably no field access
    - May need constructor access
- Debugger of Java (and not JavaScript would be good)
- Performance benchmark Sci2000
- Investigate generating asm.js friendly code
- Generate Java wrappers for all HTML5 elements dynamically (Honza)

Resources

- Project page (http://github.com/jtulach/bck2brwsr) at GitHub
- Mailing list (mailto:bck2brwsr@apidesign.org) and its archive (https://groups.google.com/d/forum/bck2brwsr)
- As of January 2017 I am moving the development to GitHub (https://github.com/jtulach/bck2brwsr) - (the original repository (http://source.apidesign.org/hg/bck2brwsr) remains read-only).
- Hudson jobs (http://hudson.apidesign.org/hudson/view/bck2brwsr/)
- Maven Repository at java.net (https://maven.java.net/content/repositories/releases/org/apidesign/bck2brwsr/)
- Javadoc

Categories: Video | Bck2Brwsr | OpenSourceContribution
