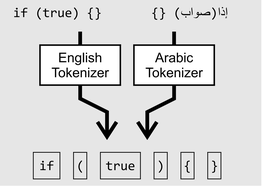
Programming in Babylscript

Different Language Modes for Different Languages

```
function hello()
{
  alert("hello world");
}

// Switch to French mode
---fr---
fonction allô()
{
  alerter(«Allô»);
}
```

Babylscript lets programmers write code in their own native languages. It does this by having different language modes for different languages. In the French language mode, for example, all keywords and function names are available in French. In the English language mode, all keywords and function names are available in English. In the English language mode, Babylscript behaves like normal JavaScript.



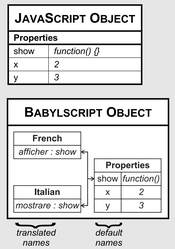
JavaScript syntax is heavily based on punctuation and mathematical symbols. Fortunately, the world's natural languages tend to have similar punctuation and mathematical symbols. As a result, Babylscript can use the same JavaScript language grammar and parser for different language modes. It uses different tokenizers to capture language differences.

```
// English code
var win = new Window();
win.setText("Hello", Color.RED);

// French code
---fr---
win.écrireTexte(«Allô», Couleur.ROUGE);
```

In different language modes, object and method names are translated into the corresponding language. The same object will expose an English API when used in English code while exposing a French API when used in French code.

Object Model Allows for Default Names and Translated Names



To support objects with different APIs in different languages, the JavaScript object model has been extended. JavaScript objects consist of a mapping of names to values known as properties. In Babylscript, the names given to properties are now called *default names*. In addition to default names, properties can have *translated names* for other languages.

```
// Create an object that has 3
// properties with the default
// names "show", "x", and "y"
var obj = {
   show: function() {},
   x: 2,
   y: 3};

// Provide translated names
// for the show property
obj['fr':'afficher'] = 'show';
obj['it':'mostrare'] = 'show';

---fr---
// In the French language mode,
// the property can be accessed
// with its default name "show"
// or its translated name "afficher"
assert(obj.show == obj.afficher);
```

By default, Babylscript properties only have default names. Translated names are defined by specifying a mapping from a name in a certain language to a default name. A property can be accessed by using either its translated name or its default name.

```
---fr---
fonction ajouterListeChaînée(val)
{
  var a = nouveau ListeChaînée();
  a.val = val;
  a.succ = tête;
  tête = a;
}
var tête = nul;

---en---
// Give an English name to the
// global function
this['en':'addLinkedList'] =
   'ajouterListeChaînée';

function removeLinkedList()
{
  if (tête != null) tête = tête.succ;
}
```

Because JavaScript global variables are stored as properties of a global "object," global variables automatically gain support for translated names. In practice though, most code will only have default names. Only the most common public APIs will typically be translated to other languages.

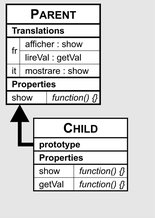
Support for Enumerations and Prototypes

```
var obj = new Object();
obj.cost = 3;
obj['fr':'coût'] = 'cost';

---fr---
// enumerate properties
pour (prop dans obj)
  alerter(prop);

// result: coût
```

Enumerations and prototypes have been modified to support translated names. When enumerating object properties, translated names are returned when they are available. During property lookups, translations in the prototype chain are used, meaning that translated names can be inherited and overridden.



Should a German subsidiary share code with its French parent company by writing it in English?

Why do students in non-English countries need to learn English before they can learn how to program?

Why do "non-programmers" like office workers need to learn English before they can write macros and scripts to automate their daily tasks?

Should a software firm with no native English speakers write their code in English, even if the resulting code is barely intelligible?

Babylscript

multilingual JavaScript

Ming-Yee lu
babylscript@babylscript.com

English-only World

Programmers can only write their code in English, but they can easily share it with others

Fragmented World

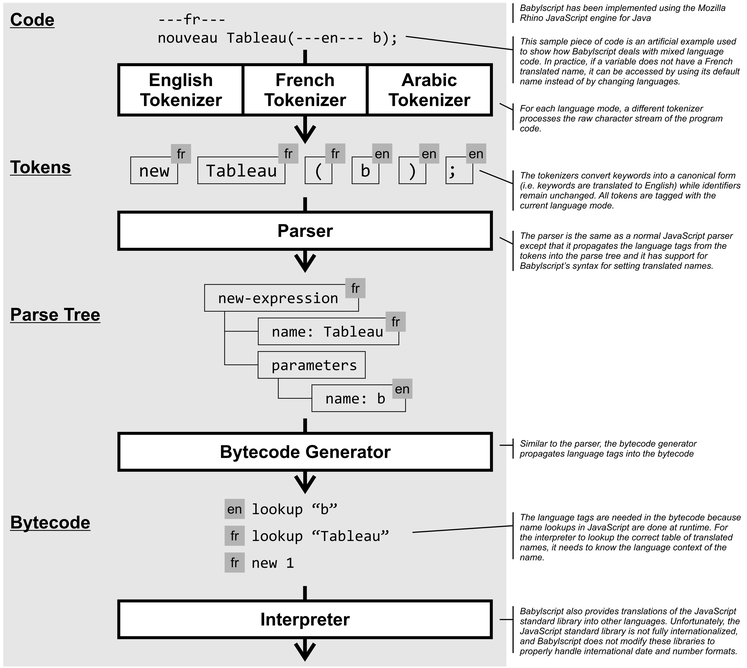
Programmers write their code in their own native languages, but they can only share it with programmers who speak the same language

Babylscript

A single multilingual programming language where programmers can write their code in their own native languages yet still easily share it with others

	JavaScript	Babylscript	Localized JavaScript
	Normal JavaScript is available in English only	*Babylscript allows code to be in a mix of different languages. French code can call a Chinese function, which in turn uses a Spanish library*	*JavaScript can be localized into different language versions such as a German JavaScript, a French JavaScript, etc.*
Can code in non-English languages?	✗ No	✓ Yes	✓ Yes
Easy to share libraries?	✓ Yes, everything is in English	✓ Yes, and APIs can be localized into French, Spanish, English, or other languages	✗ No. German JavaScript can only use German libraries. French JavaScript can only use French libraries. etc.
Easy to share code and knowledge online?	✓ Yes, everything is done in English	✓ Yes, code found online can be reused regardless of language	✗ No, programmers can only share with communities that use the same language
Anyone can extend and evolve the compiler?	✓ Yes, everything is in English	☐ New compiler code can be in any language, but the code can only be maintained by people who know that language	✗ Compiler is written in a single natural language, so programmers must know this language to extend the compiler

Implementation



Babylscript has been implemented using the Mozilla Rhino JavaScript engine for Java

This sample piece of code is an artificial example used to show how Babylscript deals with mixed language code. In practice, if a variable does not have a French translated name, it can be accessed by using its default name instead of by changing languages.

For each language mode, a different tokenizer processes the raw character stream of the program code.

The tokenizers convert keywords into a canonical form (i.e. keywords are translated to English) while identifiers remain unchanged. All tokens are tagged with the current language mode.

The parser is the same as a normal JavaScript parser except that it propagates the language tags from the tokens into the parse tree and it has support for Babylscript's syntax for setting translated names.

Similar to the parser, the bytecode generator propagates language tags into the bytecode

The language tags are needed in the bytecode because name lookups in JavaScript are done at runtime. For the interpreter to lookup the correct table of translated names, it needs to know the language context of the name.

Babylscript also provides translations of the JavaScript standard library into other languages. Unfortunately, the JavaScript standard library is not fully internationalized, and Babylscript does not modify these libraries to properly handle international date and number formats.

Discussion

Tokenizers Cannot Represent All Language Differences

- Some JavaScript symbols have multiple meanings
- In other languages, those symbols have different sets of multiple meanings
  - e.g. In English, the period is used as a decimal separator and for property access, but in French, the comma is used as a decimal separator and between function parameters
- Small parser tweaks were needed

Babylscript Has More Overhead than Normal JavaScript

- Accessing properties requires lookups of translated names
- Initializing all the translations imposes overhead
- Problems can be fixed with support for JIT compilation

Poorly Designed Keyboard Mappings

- Some keyboard mappings have no math symbols
  - More symbols are needed to represent Babylscript operations

Future Work

More Localization

- It's not enough to simply translate the language
- Other areas also need to be translated:
  - Error messages
  - Standard Library
  - Documentation
  - IDEs

Babylscript to JavaScript compiler

- Allow Babylscript to be used in browsers

Multilingual Java or C#

- Statically-typed languages have different characteristics
- Statically-typed languages can resolve translated names at compile-time
  - No performance overhead
  - More complicated language models
- Graceful ways to shoehorn multilingual support into existing Vms?
- Robust automated code translation?