Many programming languages provide foreign function interfaces (FFIs) to allow them to use code written in other languages. Unlike traditional programming languages, Newspeak does not have specialized language constructs such as extern functions, native methods etc. for accessing foreign code. Instead, Newspeak uses aliens. Aliens are objects that provide access to foreign code. They are object-capabilities for accessing non-Newspeak code.

The exact same discipline of modularity and security used throughout the Newspeak system applies to foreign function calls. One cannot access foreign code unless one has access to alien objects. Newspeak's design makes it impossible to do otherwise.

The exact formulation of aliens depends on the host environment Newspeak is running in. In a native implementation, one typically needs to call out to C in order to interact with the host system. The current Newspeak implementation runs in the web browser, and one needs to run Javascript to access the host system and interact with the outside world. Hence the need for Javascript aliens.

Javascript aliens are implemented by class `JSForPrimordialSoup`. It defines a nested class `Alien` to represent aliens. `JSForPrimordialSoup` also provides a method, `global`, that returns an alien on the Javascript global namespace. Through `global`, we can reach any Javascript object. The conventions for accessing a Javascript object via an alien are as follows:

1. To get the value of a property named `p`, send the alien the message `at: #p`. Example: the equivalent of `oldVisual.firstChild` in Javascript would be `oldVisual at: '#firstChild'`. To set a property's value to `v`, use the message `at: #p put: v`. Example: `node at: #contentEditable put: 'false'` would be like writing `node.contentEditable = "false"`.

2. To invoke a method named `m` with no arguments, send the alien the unary message `m`. Example: `window.getSelection()` in Javascript turns into `window getSelection` using aliens. If the method takes a single argument, you use the message `m: e`, where `e` is the desired argument. Example: where in Javascript you would write `document.createTextNode(label)`, using aliens you'd write `document createTextNode: label`. If the method uses `n > 1` arguments, you use a keyword message of the form `m: e1 kw2: e2 ... kwn: en`, where `kw2 .. kwn` are arbitrarily chosen keywords. Example: you would write `oldVisual.replaceChild(newNode, oldNode)` as `oldVisual replaceChild : newNode oldChild: oldNode`. The second and later keywords don't matter; choose names that are clear and useful to you. We could have written `oldVisual replaceChild : newNode insteadOf: oldNode` with the same effect. As a matter of style, one should always use the same name for multi-argument Javascript method.

3. The rules for invoking constructors are similar, except that one uses `m = new`. So, to invoke a constructor with no arguments, send the message `new`. Example: `JSObject new`. To invoke a constructor of one argument, use `new:`. Example: `JSArray new: fs size`. For more than one argument, use `new: e1 kw2: e2 ... kwn: en`, where `kw2 .. kwn` are arbitrarily chosen keywords. Example: `Blob new: data options: iterableOptions`.

You may pass Newspeak objects as arguments when sending messages to aliens. The system will represent them to Javascript as Javascript objects; these are known as expats. In general, any Javascript object coming into Newspeak is converted into an alien, and any Newspeak object passed into Javascript is converted to an expat. When expats return to Newspeak they revert to their original Newspeak representation. Likewise, when aliens are passed back into Javascript, they will also be transformed back into the underlying Javascript object.

Some datatypes are handled specially by the alien system. Numbers get converted to floats when passed to Javascript. Strings are converted back and forth between their native Newspeak and Javascript representations. Arrays Newspeak arrays may not be passed into Javascript at the moment. Javascript arrays support the messages `at:` and `at:put:` for indexing, as well as their normal Javascript methods and properties. Closures Newspeak closures sent to Javascript are represented via wrapper closures that convert their arguments and results recursively as needed. Closures are a common and important case, because of their use as callbacks.