oberonjs

Алексей Веселовский

сентябрь 2014

Кто? Зачем? Что?

Кто?

- Владислав Фолтс (aka vlad) - основная разработка компилятора.

- Алексей Веселовский (aka valexey) - тестирование, вдумчивое курение Oberon report и консультирование по языку, биндинги, демо-приложения.

- Сообщество oberspace.dyndns.org - обсуждение, тестирование, моральная поддержка. :-)

Зачем?

Зачем?

- Изучить оригинальный Оберон по спецификации а не реализации.

Зачем?

- Изучить оригинальный Оберон по спецификации, а не реализации.

- Попробовать силы в проектировании и написании компиляторов и библиотек с чистого листа.

Зачем?

- Изучить оригинальный Оберон по спецификации, а не реализации.
- Попробовать силы в проектировании и написании компиляторов и библиотек с чистого листа.
- В перспективе - получить опыт разработки ЯП.

Зачем?

- Изучить оригинальный Оберон по спецификации, а не реализации.
- Попробовать силы в проектировании и написании компиляторов и библиотек с чистого листа.
- В перспективе - получить опыт разработки ЯП
- Just for fun

Зачем?

- Изучить оригинальный Оберон по спецификации, а не реализации.
- Попробовать силы в проектировании и написании компиляторов и библиотек с чистого листа.
- В перспективе - получить опыт разработки ЯП
- Just for fun
- От js (aka ECMAScript) тошнит, но…

Пару слов о js

js как язык

- динамическая типизация
- динамическая **слабая** типизация
- абсолютно все mutable! В том числе предопределенные типы и объекты!
- проглатывает ошибки в рантайме
- в результате рефакторинг превращается в ад

Состояние js-приложения после рефакторинга

Да и до, тоже

Хорошее в js

- компактная запись
- наличие удобных структурных литералов (json - это javascript object notation – это просто часть языка)
- гибкий во все стороны - лепи что хочешь
- прекрасен для мелкого скриптинга страничек и DOM

js как платформа

- есть везде где есть браузер (Win, Lin, OS X, Android, iOS, FreeBSD and so on)

- много где можно писать “родные” приложения (есть системное API для js и рантайм) - Windows 8+ modern apps (ex Metro), WinPhone, Ubuntu Unity, Tizen, Samsung SmartTV, FirefoxOS.

- для сервера есть node.js

- богатый API

Ниша Оберона в мире js

Ниша Оберона в мире js

- Приложения, не скриптинг

Ниша Оберона в мире js

- Приложения, не скриптинг
- Долгий цикл жизни, предполагающий рефакторинг

Ниша Оберона в мире js

- Приложения, не скриптинг
- Долгий цикл жизни, предполагающий рефакторинг
- Внутренней логики больше чем работы со внешним API

Ниша Оберона в мире js

- Приложения, не скриптинг
- Долгий цикл жизни, предполагающий рефакторинг
- Внутренней логики больше чем работы со внешним API
- Придаем форму структуре приложения, даже если в нем есть части на js

Вместе с js, а не вместо

- мелкий скриптинг, и мир нетепизированного хаоса оставляем js

Вместе с js, а не вместо

- мелкий скриптинг, и мир нетепизированного хаоса оставляем js

- сложную логику, мир порядка – мир Оберона. Оберон сможет поддерживать порядок

Вместе с js, а не вместо

- мелкий скриптинг, и мир нетепизированного хаоса оставляем js

- сложную логику, мир порядка – мир Оберона. Оберон сможет поддерживать порядок

- проще и быстрее проверить один раз на стыке миров, чем постоянно проверять в js

Требования к реализации oberonjs

- генерация человекочитабельного кода, годного для правки и отладки руками
- полная реализация ЯП Оберон (ревизии 2013 года)
- легкая интеграция с js-кодом

Реализация

Кратко о компиляторе

- однопроходный (пока)
- написан частью на js, частью на eberon (диалект - Experimental oBERON), компилирует сам себя
- opensource: http://github.com/vladfolts/oberonjs
- реализует два языка - последнюю ревизию Oberon и eberon
- демо тут: http://oberspace.dyndns.org/oberonjs.html
- работает в браузере и оффлайн (node.js)

Пример генерации кода

```
MODULE A;
IMPORT B;
VAR
    r : B.R;
BEGIN
    r.b := FALSE;
    B.T(r);
END A.
```

node.js

```
var B = require(".//B.js");
var r = new B.R();
r.b = false;
B.T(r);
```

browser

```
var A = function (B){
    var r = new B.R();
    r.b = false;
    B.T(r);
}(B);
```

Интеграция с js туда

- Генерируются модули в формате понятном node.js, Оберон-код можно вызывать родным для js образом

- Для браузера генерируется в виде объектов-функций (стандартизированных модулей для js нет)

- Код вполне человекочитабельный

Интеграция с js обратно

- Псевдомодуль JS
- Волшебная псевдофункция JS.do(). js-вставки в Оберон-код. *Легко* писать биндинги

```
MODULE A;
IMPORT B, JS;
VAR
    r : B.R;
BEGIN
    r.b := FALSE;
    B.T(r);
    JS.do("console.log(r.b)")
END A.
```

```
var B = require(".//B.js");
var JS = GLOBAL;
var r = new B.R();
r.b = false;
B.T(r);
console.log(r.b);
```

Пример биндига – ProcessingJs

Demo

На реализацию demo ушло 45 минут (в том числе на доработку биндига)

Сборка Oberon Compiler by N. Wirth (2014)

Project Oberon Compiler

- Реально большой проект, самый большой и сложный из доступных (более 16000 слов, и 3200 строк кода)
- То самое, сложное приложение, не скриптинг
- Стиль Вирта сильно отличается от нашего стиля
- Отличный полигон для испытаний нашего компилятора

Project Oberon Compiler, особенности портирования

- Другой язык. Не тот что в репорте (отличий не очень много, но есть)
- Использование SYSTEM в компиляторе – делает генерацию для target-платформы зависимой от host-платформы
- Зависимость от OS Oberon (Texts, Files, Oberon)

Project Oberon Compiler, портирование

- Первая попытка в декабре 2013 года - январе 2014. Результат: формально собрать удалось (после правки ряда багов в компиляторе). Но тестирование, попыток запуска не проводилось.

- Второй подход: август-сентябрь 2014 года

Demo

http://oberspace.dyndns.org/PO_2013_compiler.html

Project Oberon Compiler, текущие результаты портирования

- Быстродействие oberonjs компилятора: проект полностью собирается за 0.4 секунды

- Объем генерируемого кода: из 3200 строк и 16000 слов кода Вирта получается 6100 строк и 19000 слов форматированного js-кода

- Найдено и исправлено несколько ошибок в компиляторе oberonjs

- Работоспособность порта частичная – есть еще не исправленные ошибки в oberonjs, и кое-что не реализовано из модулей.

oberonjs статус и планы, задачи

- Завершить портирование PO Compiler, путем исправления ошибок в oberonjs. Это будет переход проекта oberonjs в статус beta

- Добавить поддержку source map

- Добавить поддержку парсинга логов компиляции в Sublime Text (Sublime - основной редактор которым мы пользуемся, есть наш плагин для поддержки Оберона)

- Стандартная библитека

- Больше биндингов!

Спасибо!

Вопросы?

http://oberspace.dyndns.org/Oberonjs.pdf