Hachioji.pm #18

Inline::JSX

2012-06-23 まかまか般若波羅蜜

makamaka.donzoko@gmail.com

Twitter: @maka2_donzoko

Inlineモジュール

Perlプログラムソース内に他の言語のコードをかけるようにする
-- C, Java, Ruby, Basic, etc.

https://github.com/makamaka/Inline-JSX

```
use Inline 'JSX';

say Addition->add(1, 2);          # => 3
say Addition->add('foo', 'bar'); # => foobar
say Addition->add('foo', 2)      # => error: type mismatch!

__END__
__JSX__
class Addition {
    static function add(x:int, y:int) : int {
            return x + y;
    }

    static function add(x:string, y:string) : string {
            return x + y;
    }
}
```

```
use Inline ‘JSX’;

my $p = Point->new( Point->new(1,2) );
say $p->x, “:”, $p->y;

__END__
__JSX__

class Point {
    var x = 0;
    var y = 0;

    function constructor() {}

    function constructor(x : number, y : number) {
        this.set(x, y);
    }

    function constructor(other : Point) {
        this.set(other);
    }

    function set(x : number, y : number) : void {
        this.x = x;
        this.y = y;
    }

    function set(other : Point) : void {
        this.x = other.x;
        this.y = other.y;
    }
}
```

やってること

JavaScriptのPerl実装 JE を利用。

JSXコードをjsxでコンパイル
→JavaScriptコードをJEで評価
→JE objectからPerlコード生成
→evalして実行！

このコードは……

```
class Addition {
    static function add(x:int, y:int) : int {
            return x + y;
    }

    static function add(x:string, y:string) : string {
            return x + y;
    }
}
```

=>このように変換される（抜粋）

```
sub Addition::add {
    my $self = shift;
    my $argstypes = encode_argstype( @_ );
    if ( check_types( ['I','I'], $argstypes ) ) {
        return convert_je_to_perl(
            $scope->{'Addition$add$II'}->(
                JE::Number->new($je,$_[0]),
                JE::Number->new($je,$_[1])
            )
        );
    }
    if ( check_types( ['S','S'], $argstypes ) ) {
        return convert_je_to_perl(
            $scope->{'Addition$add$SS'}->(
                JE::String->new($je,$_[0]),
                JE::String->new($je,$_[1])
            )
        );
    }
    Carp::croak('argument type mismatch');
}
```

このコードは……

```
class Point {
    var x = 0;
    var y = 0;

    function constructor() {}

    function constructor(x : number, y : number) {
        this.set(x, y);
    }

    function constructor(other : Point) {
        this.set(other);
    }

    function set(x : number, y : number) : void {
        this.x = x;
        this.y = y;
    }

    function set(other : Point) : void {
        this.x = other.x;
        this.y = other.y;
    }
}
```

=>このように変換される（抜粋）

```
{ package Point; our @ISA = ('Object'); }
{
    package Point;
    our $AUTOLOAD;
    sub AUTOLOAD {
        my $method = $AUTOLOAD;;
        $method =~ s{.*::}{};;
        return if $method eq 'DESTROY';
        no strict 'refs';
        Carp::croak("No property '$method'") unless exists $${$_[0]}->{props}->{ $method };
        *{"$method"} = sub {
            my $o = shift;
            if ( @_ ) {
                $$o->prop( $method => Inline::JSX::convert_je_for_x($$o->global, $_[0]) );
            }
            else {
                Inline::JSX::convert_je_to_perl( $$o->prop( $method ) );
            }
        };
        goto &$AUTOLOAD;
    }
}
```

```
sub Point::new {
    my $self = shift;
    my $m = $jsx->method('require', '-');
    my $argstypes = encode_argstype( @_ );
    if ( check_types( [], $argstypes ) ) {
        return bless ¥do{$m->prop('Point$')->construct()}, 'Point';
    }
    if ( check_types( ['N','N'], $argstypes ) ) {
        return bless ¥do{
                    $m->prop('Point$NN')->construct(
                        JE::Number->new($je,$_[0]), JE::Number->new($je,$_[1])
                    )
                }, 'Point';
    }
    if ( check_types( [['L','Point']], $argstypes ) ) {
        return bless ¥do{
                    $m->prop('Point$LPoint$')->construct(
                         Inline::JSX::convert_nullable_to_je($je,$_[0] => "L")
                    )
                }, 'Point';
    }
    Carp::croak('argument type mismatch');
}
```

```
sub Point::set {
    my $o = shift;
    my $argstypes = encode_argstype( @_ );
    if ( check_types( ['N','N'], $argstypes ) ) {
        return convert_je_to_perl(
            $$o->method('set$NN' => JE::Number->new($je,$_[0]), JE::Number->new($je,$_[1]))
        );
    }
    if ( check_types( [['L','Point']], $argstypes ) ) {
        return convert_je_to_perl(
            $$o->method('set$LPoint$' =>
                Inline::JSX::convert_nullable_to_je($je,$_[0] => "L")
            )
        );
    }
    Carp::croak('argument type mismatch');
}
```

一度コンパイルが済めば生成されたコードはキャッシュされる。

→それを使えばjsxがない環境でもJSXコードを実行できるぜ！