Fun with Python and Javascript

dis, ast and transcrypt
A Journey by Evan Carroll

Goal

- **Migrate a major Python project to Javascript! Yay!**
- **What project? `youtube-dl`**
- **Why? Rx.js, .retry, parallelization, DOM-readability, browser-integration, CSS-selectors, speed. That's not what this is about.**

Migration tactics

In order of desirability

- **Fully-automated migration**
- **Extractor and Test-suite migration, leaving common base classes to be re-implemented**
- **Test-suite migration**
- **Coverage**

What can we do?

- **Too early to tell.**
- **Don’t know**

Compilation Phases

1. Yacc or BISON (parsing)
2. AST construction
3. AST transformations
4. Byte-code generation
5. Virtual Machine execution
6. Machine Code
7. CPU execution

AST

```
a = 5;


def f():
    global a
    a = a+1
    print ("foo" + str(a))


f();
f();
```

```
Module(body=[Import(names=[al
ias(name='dis', asname=None)]),
Assign(targets=[Name(id='a',
ctx=Store())], value=Num(n=5)),
FunctionDef(name='f',
args=arguments(args=[],
vararg=None, kwarg=None,
defaults=[]),
body=[Assign(targets=[Name(id
='a', ctx=Store())],
value=BinOp(left=Name(id='a',
ctx=Load()), op=Add(),
right=Num(n=1))),
Print(dest=None,
values=[Str(s='foo')], nl=True)],
decorator_list=[])])
```

AST Pretty with `astpretty`

```
Module(
    body=[
        Assign( … a    ),
```

- Above sets a
- Right is function f

```
FunctionDef(
    lineno=3,
    col_offset=0,
    name='f',
    args=arguments(…),
    body=[
        Global( … stuff ),
        Print( … stuff ),
        Expr( …. stuff),
        Expr( …. stuff),
```

DIS (bytedcode dissassembly)

```
import dis;

a = 5;

def f():
    global a
    a = a+1
    print ("foo" + str(a))

print dis.dis(f);
```

```
5           0 LOAD_GLOBAL              0 (a)
            3 LOAD_CONST               1 (1)
            6 BINARY_ADD
            7 STORE_GLOBAL             0 (a)

6          10 LOAD_CONST               2 ('foo')
           13 LOAD_GLOBAL              1 (str)
           16 LOAD_GLOBAL              0 (a)
           19 CALL_FUNCTION            1
           22 BINARY_ADD
           23 PRINT_ITEM
           24 PRINT_NEWLINE
           25 LOAD_CONST               0 (None)
           28 RETURN_VALUE
None
```

Transcrypt

- **Compiles to JS**
- **Uses AST Transformations (and AST module)**
- **Arguments**
  - b = build
  - p = object that holds app
  - e = ecmascript version
  - n = minfication
- **Run as**

transcrypt -b -p .none -e6 -n ./test.py

Obstacles

`transcrypt` needs collections and _weakref

Have a lot to work out yet.