


Creative FP

Dave Gurnell, [@davegurnell](#)



\$0+

Creative Scala


underscore

Putting the FUN
into FUNCTIONAL!

Creative Scala (Early Access)

Creative Scala is aimed at developers who have no prior experience in Scala. It is designed to give you a fun introduction to functional programming. We assume you have some familiarity with another programming language but little or no experience with Scala or other functional languages.

I want this!

You'll get 4 files.

<http://underscore.io/training/courses/creative-scala>

Agenda

What is FP?

Creative Drawing with Doodle

<https://github.com/davegurnell/doodlejs>

Creative Music with Compose

<https://github.com/davegurnell/composejs>

FP Elsewhere

What is FP?

Functions as values

Higher order
functions

Haskell

Immutability

Composition and
transformation

Purity
(no side-effects)

Recursion

Types

Algebraic data
structures

Monads!
(Scala joke)

Functions as values

Higher order
functions

Haskell

Immutability

Composition and
transformation

Purity
(no side-effects)

Recursion

Types

Algebraic data
structures

Monads!
(Scala joke)

Functions as values

Higher order
functions

Haskell

Immutability

Interpreters

Purity

(no side-effects)

Composition and
transformation

Recursion

Types

Algebraic data
structures

Monads!
(Scala joke)

“Almost all designs fall into the
‘compiler’ or ‘interpreter’ pattern,
using a model of the data and
functions on that data ...”

Don Syme on Stack Overflow

[http://stackoverflow.com/questions/27852709/
enterprise-patterns-with-functional-programming](http://stackoverflow.com/questions/27852709/enterprise-patterns-with-functional-programming)

“You can implement most systems by
writing a compiler (or interpreter).
So learn to write compilers.”

Don Syme on Stack Overflow

[http://stackoverflow.com/questions/27852709/
enterprise-patterns-with-functional-programming](http://stackoverflow.com/questions/27852709/enterprise-patterns-with-functional-programming)

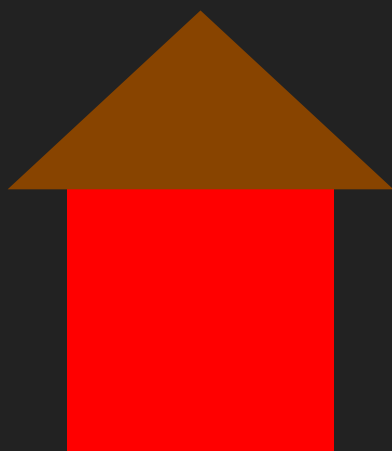
Motivation

What does this code draw...?

```
var canvas = document.getElementById('canvas');  
var context = canvas.getContext('2d');
```

```
context.fillStyle = 'brown';  
context.moveTo(50, 0);  
context.lineTo(100, 50);  
context.lineTo(0, 50);  
context.lineTo(50, 0);  
context.fill();
```

```
context.fillStyle = 'red';  
context.fillRect(10, 50, 80, 80);
```



Problems

Unclear

“Magic numbers”

Implementation detail

Not reusable

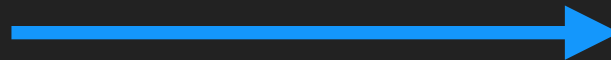
Doodle

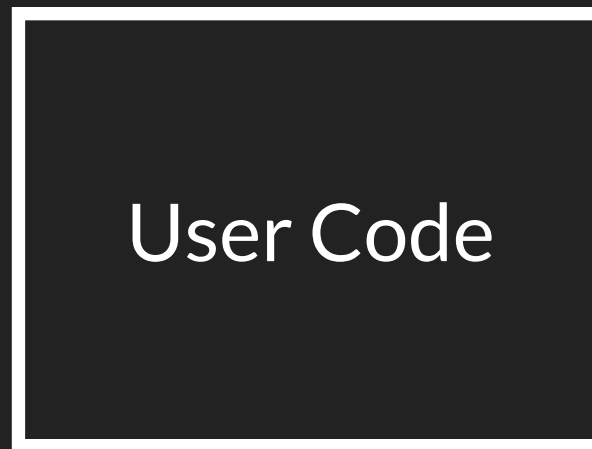
Graphics Interpreter

<https://github.com/davegurnell/doodlejs>

```
var house = triangle(100, 50).  
    above(rectangle(80, 80));  
  
draw(house);
```

Demo

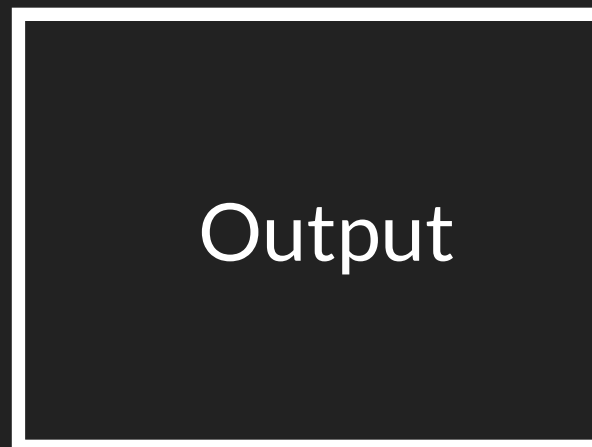




Representation



Imperative Commands



Recipe

Representation

Primitives

Combinators

Interpreter

Syntax

Primitives

Combinators

Interpreter

Syntax

Primitives

Images: Circle, Rectangle, Triangle

Combinators

Interpreter

Syntax

Primitives

Images: Circle, Rectangle, Triangle

Combinators

Interpreter

Syntax

Primitives

Images: Circle, Rectangle, Triangle

Combinators

Geometric: Above, Beside, Overlay

Interpreter

Syntax

Code

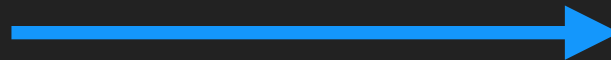


image-ast.js

Primitives

Images: Circle, Rectangle, Triangle

Combinators

Geometric: Above, Beside, Overlay

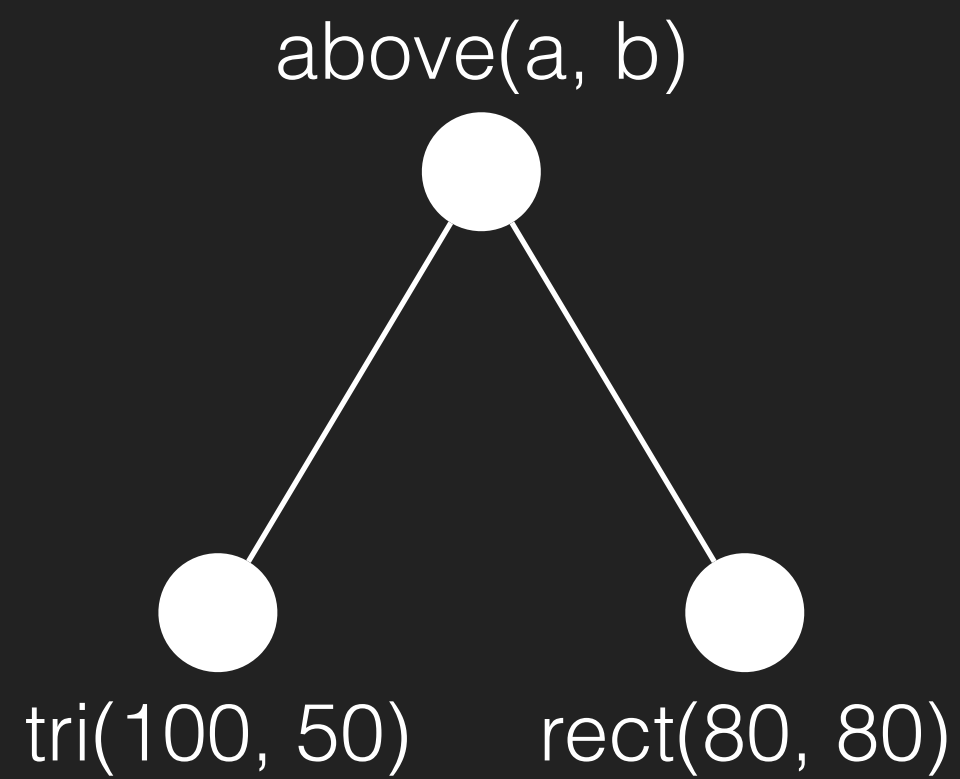
Interpreter

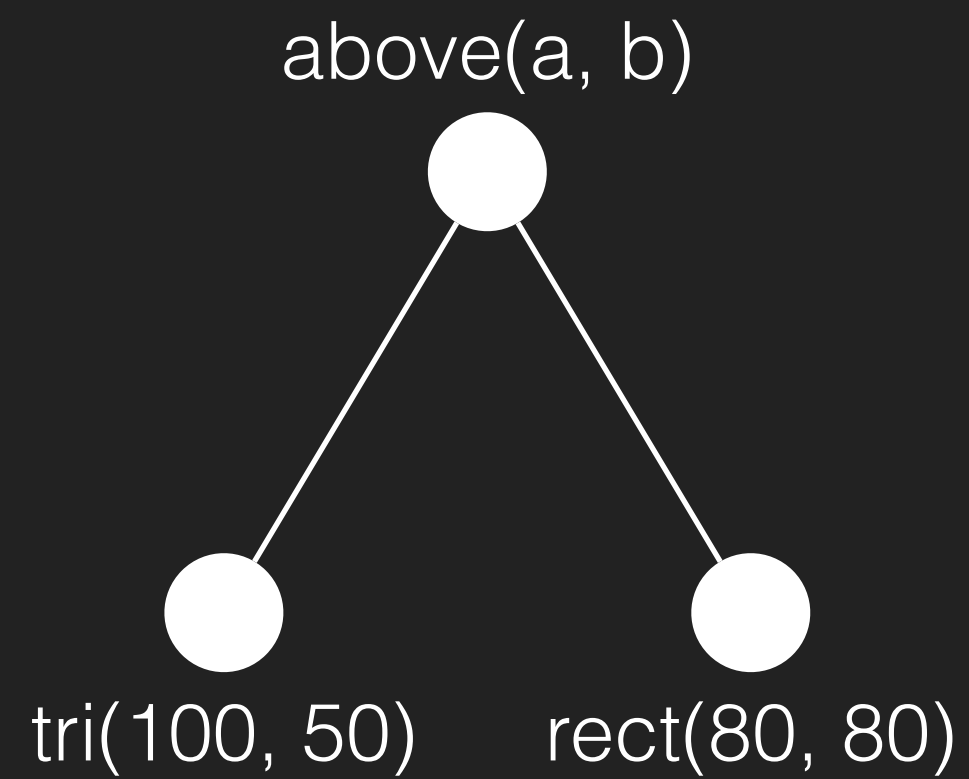
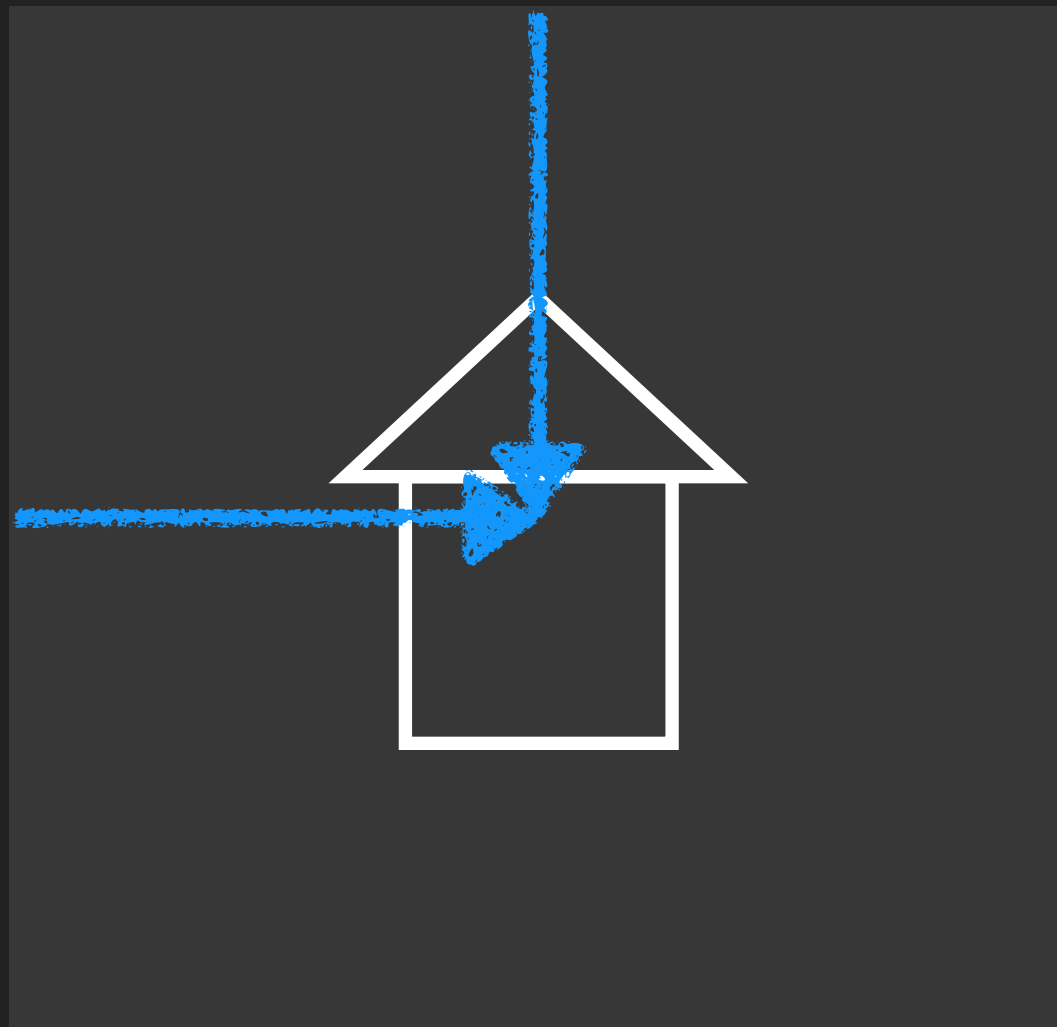
Syntax

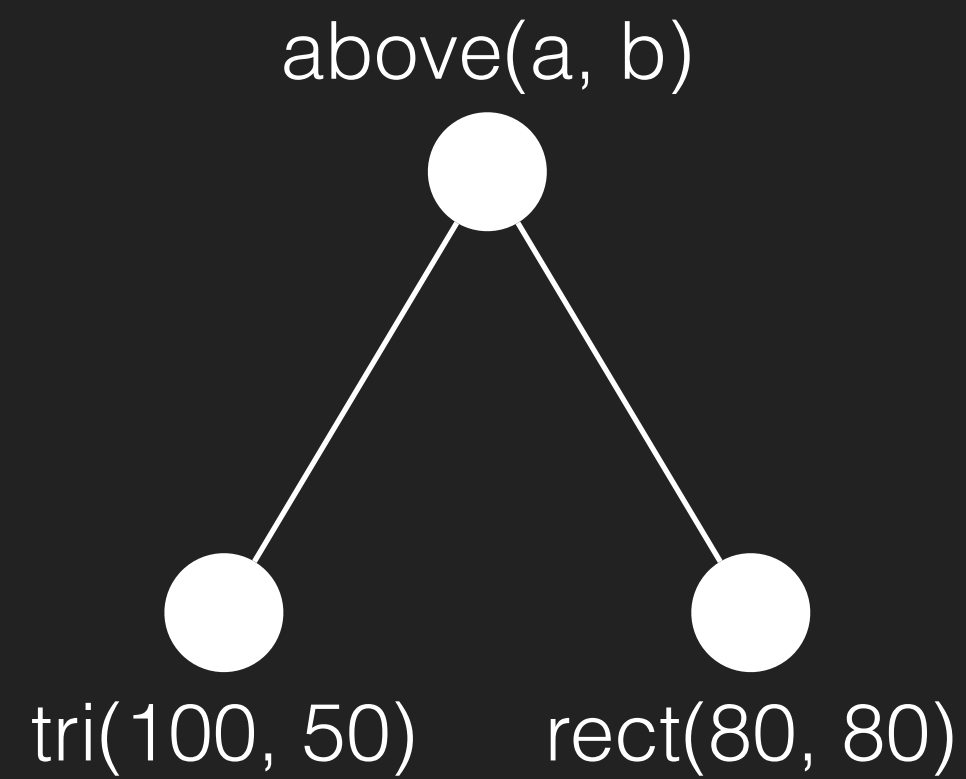
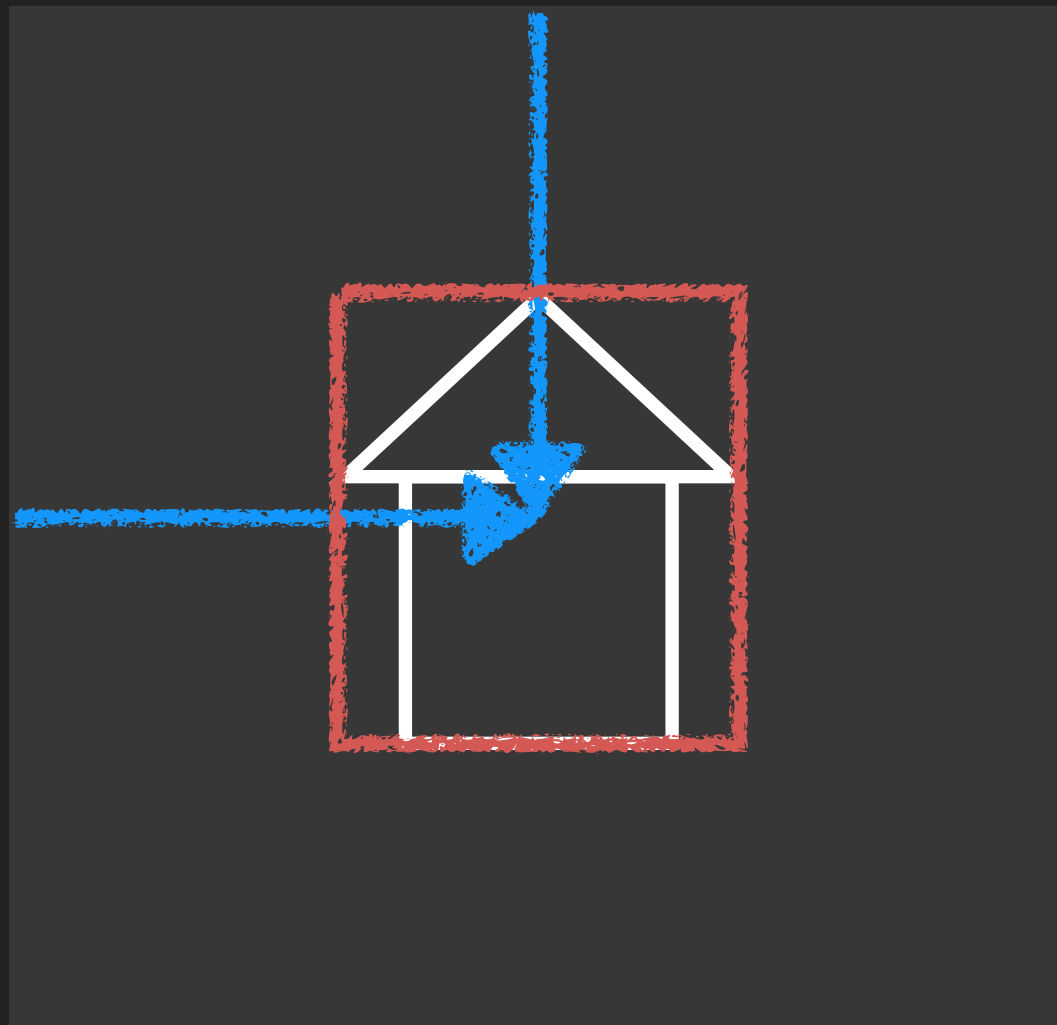

```
// CanvasContext Image -> Void  
function draw(ctx, image) {  
    // ...  
  
}
```

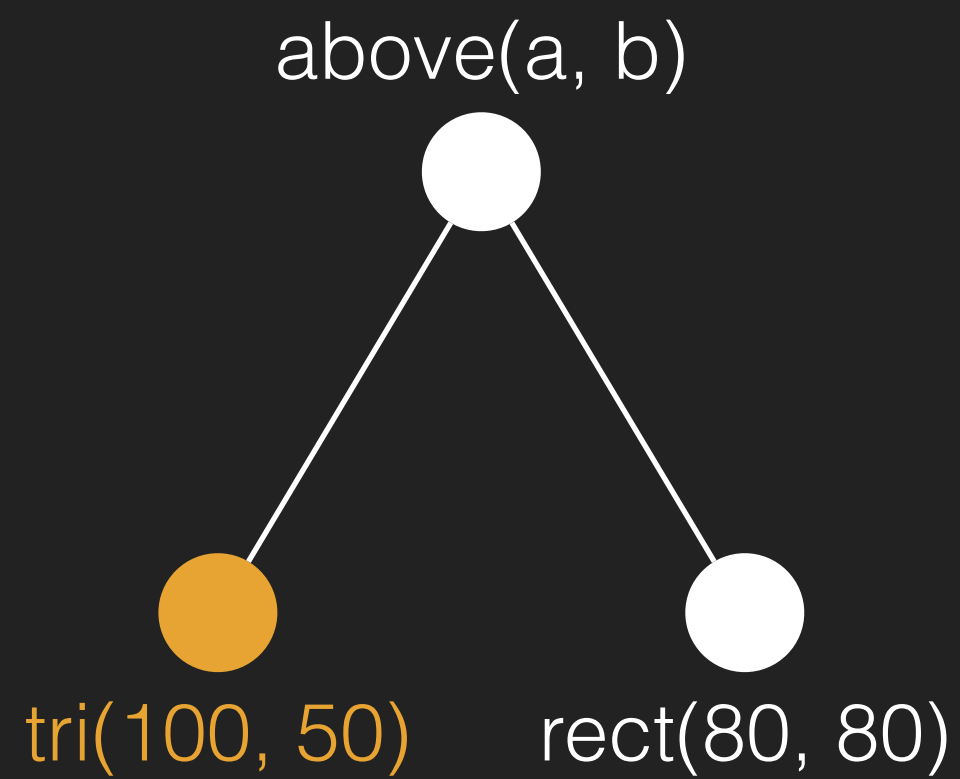
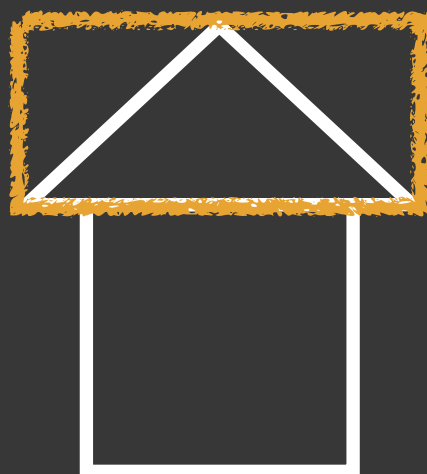
```
// CanvasContext Image -> Void
function draw(ctx, image) {
    var bounds = boundingBox(image);

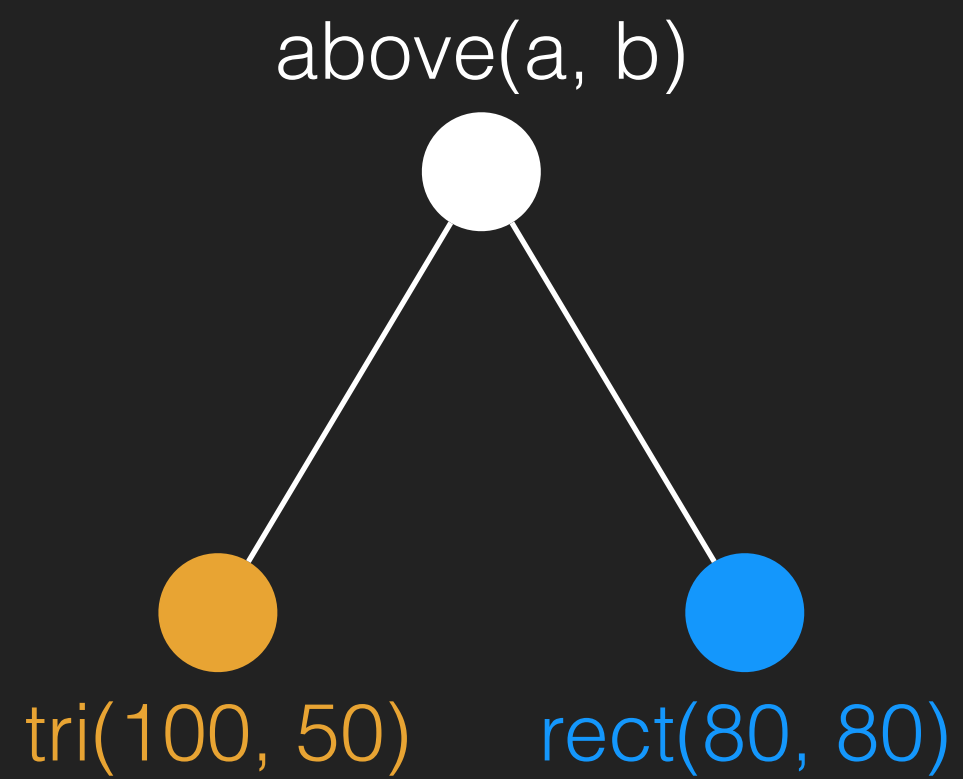
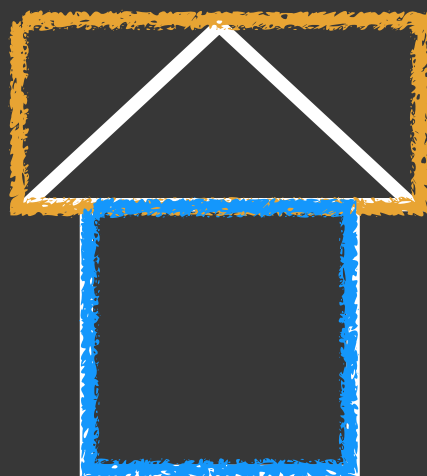
    drawImageAt(ctx, bounds, image);
}
```

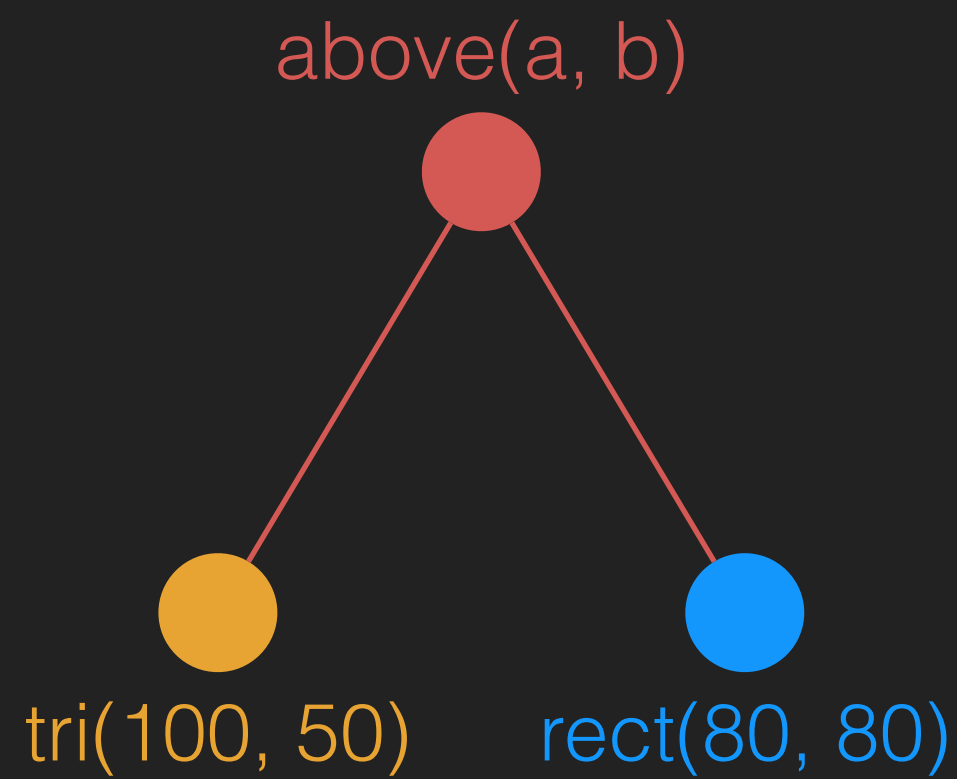
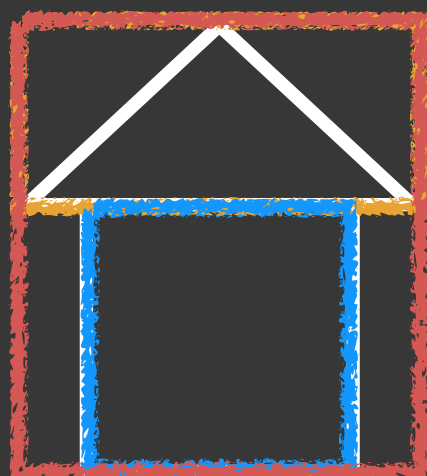




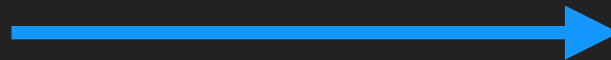








Code



bounds.js, canvas.js

Primitives

Images: Circle, Rectangle, Triangle

Combinators

Geometric: Above, Beside, Overlay

Interpreter

Syntax

```
image1.above(image2);  
// new Above(image1, image2)
```

```
beside(image1, image2, image3);  
// new Beside(  
//     new Beside(  
//         image1,  
//         image2),  
//     image3)
```

Code and Demo

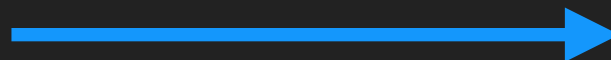


image-ast.js, image-helpers.js

Recap

Build a Representation

small set of primitives and combinators

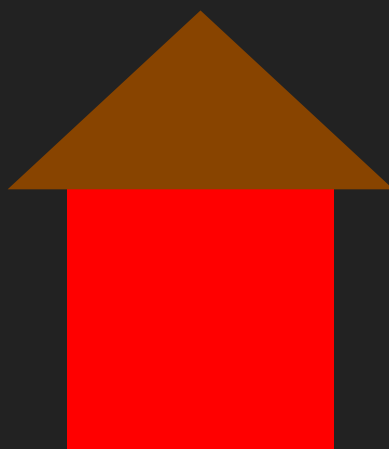
Write an Interpreter

simple functions, switch on type, structural recursion

Advantages

simple, dependency-free, composable, reusable,
can plug in different front/back-ends

Doodling with Style



Primitives

Images: Circle, Rectangle, Triangle

Combinators

Geometric: Above, Beside, Overlay

Interpreter

Syntax

Primitives

Images: Circle, Rectangle, Triangle
Styles

Combinators

Geometric: Above, Beside, Overlay
StyleTransform

Interpreter

Syntax

Code and Demo

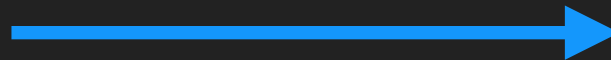


image-ast.js, canvas.js

Compose

Music Interpreter

<https://github.com/davegurnell/composejs>

Recipe

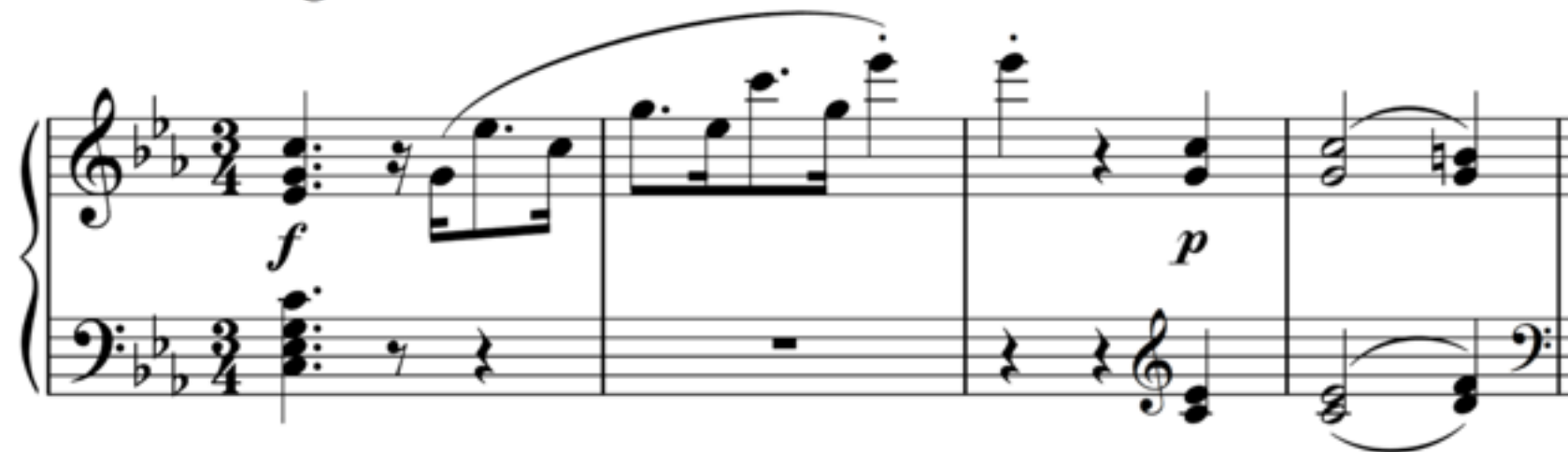
Primitives

Combinators

Interpreter

Syntax

Allegro molto e con brio



Recipe

Primitives

Pitches, Durations

Notes, Rests

Combinators

Interpreter

Syntax

Recipe

Primitives

Pitches, Durations

Notes, Rests

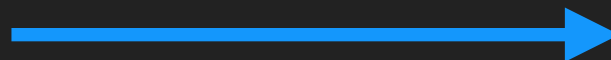
Combinators

Parallel, Sequential

Interpreter

Syntax

Code and Demo



score-ast.js

Recipe

Primitives

Pitches, Durations

Notes, Rests

Combinators

Parallel, Sequential

Interpreter

Syntax

```
// Score -> ???  
function play(score) {  
    // ...  
}
```

```
var ctx = new AudioContext();

var req = new XMLHttpRequest();
req.open('GET', 'samples/bell.wav', true);
req.responseType = 'arraybuffer';

req.onload = function() {
    ctx.decodeAudioData(req.response, function(buffer) {
        var source = ctx.createBufferSource();
        source.buffer = buffer;
        source.connect(ctx.destination);
        source.playbackRate.setValueAtTime(2.0, 0);
        source.start(0);
    });
};

req.send();
```

Promises!

```
// PromiseOf(String)
```

```
var stringPromise = Q.fncall(function() {  
    return "Some Value";  
});
```

```
// PromiseOf(Number)
```

```
var waitABit = Q.delay(300);
```

```
// PromiseOf(B)
```

```
var aThenB = promiseOfA.then(function(a) {  
    return promiseOfB;  
});
```

```
// PromiseOf([A, B])
```

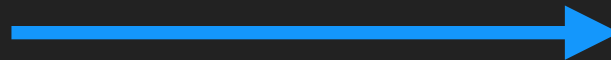
```
var aAndB = Q.all([ promiseOfA, promiseOfB ]);
```

```
// Score -> Promise  
function play(score) {  
    // ...  
}
```

```
// Score -> PromiseOf(Any)
function play(score) {
  return initialize().then(function(config) {
    return playScore(score, config);
  });
}
```

```
// Score SomeConfig -> PromiseOf(Any)
function playScore(score, config) {
  // ...
}
```

Code



player

Recipe

Primitives

Pitches, Durations

Notes, Rests

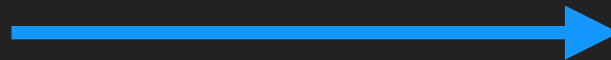
Combinators

Parallel, Sequential

Interpreter

Syntax

Code and Demo



score-helpers.js

Re-recap

Build a Representation

small set of primitives and combinators

Write an Interpreter

simple functions, switch on type, structural recursion

Advantages

simple, dependency-free, composable, reusable,
can plug in different front/back-ends

Re-recap

Build a Representation

small set of primitives and combinators

Write an Interpreter

simple functions switch on type, structural recursion

Advantages

simple, dependency-free, composable, reusable,
can plug in different front/back-ends

Interpreters Elsewhere

Promises

Primitives

Promises, Functions

Combinators

then, all, ...

Interpreter

???

Syntax

then, all, ...

Form Validation

Primitives

Combinators

Interpreter

Syntax

Form Validation

Primitives

Result, Pass, Fail, Rule (function)

Combinators

Interpreter

Syntax

Form Validation

Primitives

Result, Pass, Fail, Rule (function)

Combinators

Sequence, Parallel, Drill-Down

Interpreter

Syntax

Form Validation

Primitives

Result, Pass, Fail, Rule (function)

Combinators

Sequence, Parallel, Drill-Down

Interpreter

Run the rule!

Syntax

Form Validation

My talk at Scala Exchange

[https://skillsmatter.com/skillscasts/
5837-functional-data-validation](https://skillsmatter.com/skillscasts/5837-functional-data-validation)

```
// String -> ResultOf(String)
function nonEmpty(str) {
    return (str.length == 0) ? fail(...) : pass(str);
}

// String -> ResultOf(Int)
function stringToInt(str) { ... }

// Object -> ResultOf(Signup)
var validateSignup = validateAll(
    validateSeq(getField('name'), nonEmpty),
    validateSeq(getField('age'), nonEmpty, stringToInt)
).map(function(name, age) {
    return new Signup(name, age);
});
```

Batching API Calls

Haxl

<https://github.com/facebook/Haxl>

Stitch

<https://www.youtube.com/watch?v=VVpmMfT8aYw>

Thank You

Dave Gurnell, [@davegurnell](#)

<https://github.com/davegurnell/asyncjs-creative-fp>

