DART

Your first-class upgrade to web development

Justin Fagnani, Software Engineer, Dart
Seth Ladd, Developer Advocate, Dart

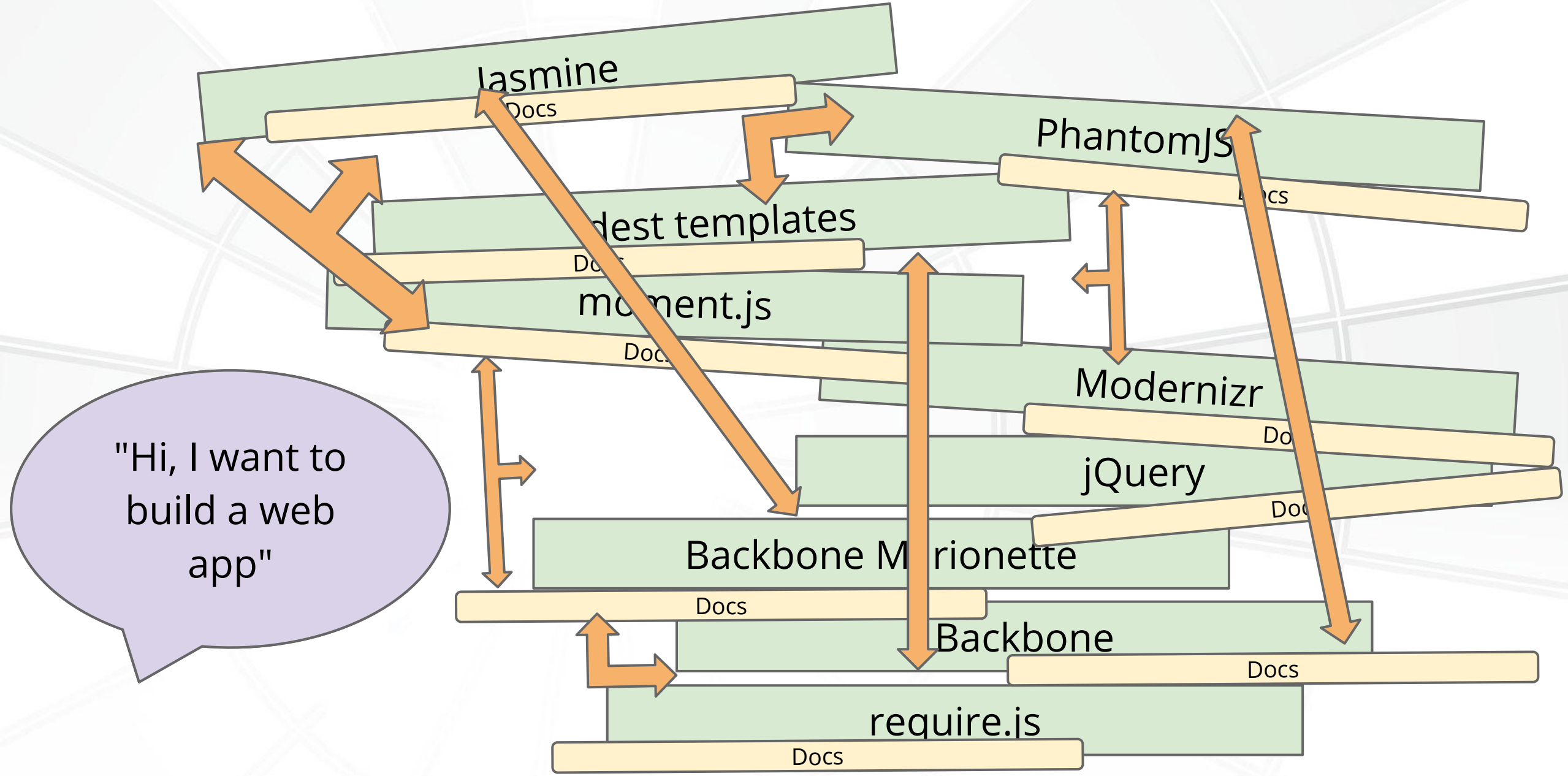
Google I/O 2013

#dartlang

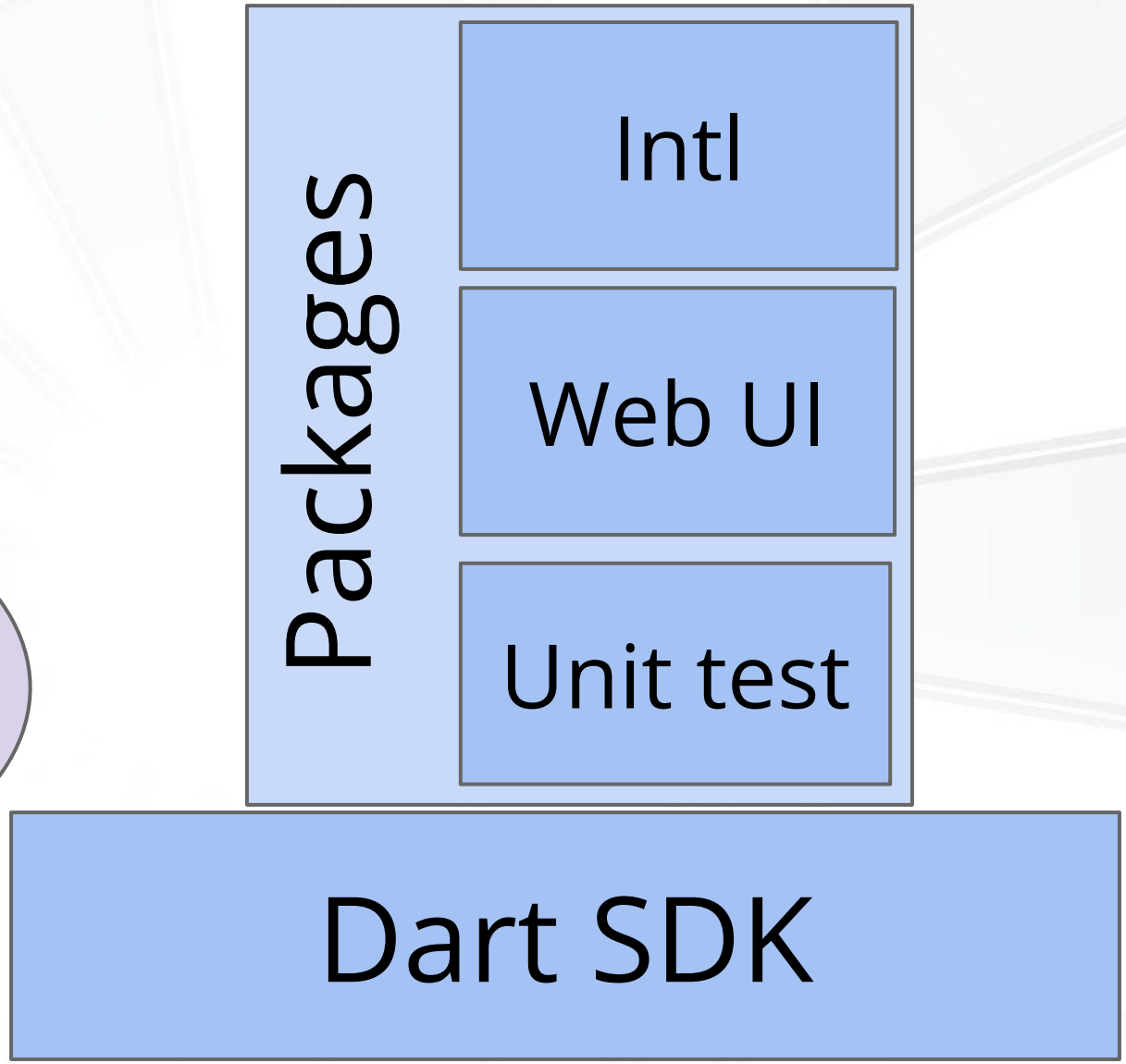
DART

- Language and libraries
- Tools
- VM
- Compiler to JavaScript

#dartlang


Jasmine
Docs
PhantomJS
dest templates
moment.js
Modernizr
jQuery
Backbone M rionette
Docs
Backbone
Docs
require.js
Docs
"Hi, I want to build a web app"

"Things are consistent and clear."



Inside Google

Big and Complex

- Dozens to Hundreds of Engineers
- Millions of Lines of Code

Lots of Layers

- GWT
- Closure
- Soy

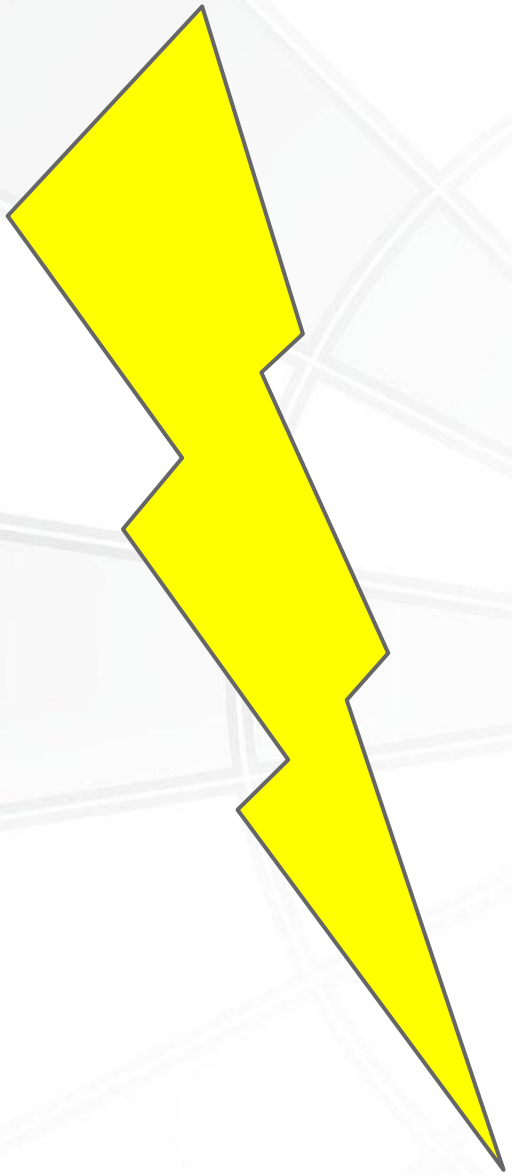
Low Productivity

- No edit/refresh
- *24 min to see a change!!*

Surely we can do better!

Improve all the things!

	Structure	Syntax	Semantics	Tools	Core Libs	Requires Compilation for Development	Performance
Vanilla JS	----	----	----	----	----	No	----
Dart						No	
Closure			----			Yes	----
CoffeeScript			----	----	----	Yes	----
TypeScript			----		----	Yes	----
GWT						Yes	----



Lightning Tour

- Syntax
- Semantics
- Structure

Simple syntax, ceremony free

```
class Hug {
```

Familiar

Simple syntax, ceremony free

```
class Hug {
  final num strength;
  Hug(this.strength);
```

Simple syntax, ceremony free

```
class Hug {
  final num strength;
  Hug(this.strength);
  Hug.bear() : strength = 100;
```

Named constructor

Simple syntax, ceremony free

```
class Hug {
  final num strength;
  Hug(this.strength);
  Hug.bear() : strength = 100;

  Hug operator +(Hug other) {
    return new Hug(strength + other.strength);
  }
```

Operator overriding

Simple syntax, ceremony free

```
class Hug {
  final num strength;
  Hug(this.strength);
  Hug.bear() : strength = 100;

  Hug operator +(Hug other) {
    return new Hug(strength + other.strength);
  }

  void patBack({int hands: 1}) {
    // ...
  }
```

Named, optional params w/ default value

Simple syntax, ceremony free

```
class Hug {
  final num strength;
  Hug(this.strength);
  Hug.bear() : strength = 100;

  Hug operator +(Hug other) {
    return new Hug(strength + other.strength);
  }

  void patBack({int hands: 1}) {
    // ...
  }

  String toString() => "Embraceometer reads $strength";
}
```

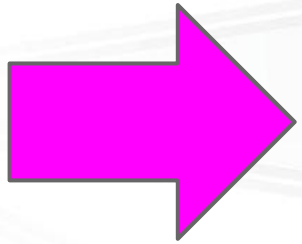
One-line function

Simple syntax, ceremony free

```
class Hug {
  final num strength;
  Hug(this.strength);
  Hug.bear() : strength = 100;

  Hug operator +(Hug other) {
    return new Hug(strength + other.strength);
  }

  void patBack({int hands: 1}) {
    // ...
  }

  String toString() => "Embraceometer reads $strength";
}
```

String Interpolation

Clean semantics and behavior



Clean semantics and behavior

Examples:

- Only *true* is truthy
- There is no *undefined*, only *null*
- No type coercion with ==, +

Missing getter?

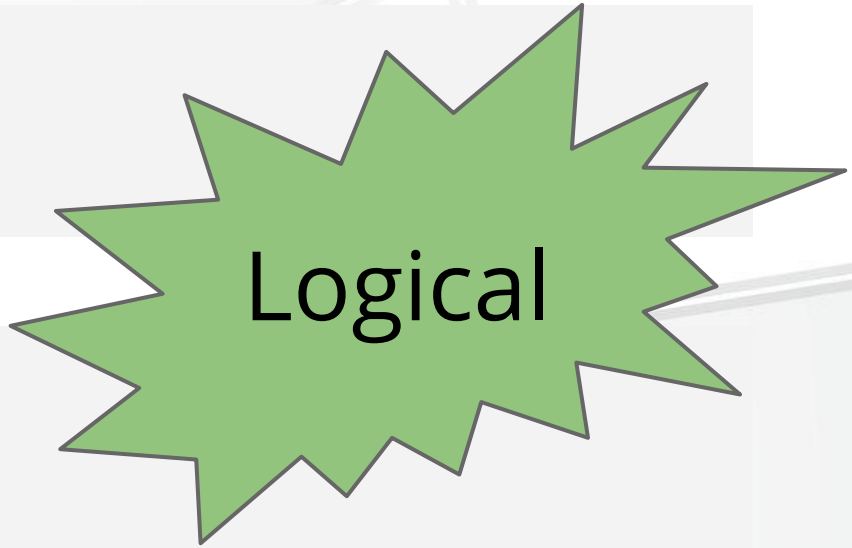
```
"hello".missing  // ??
```


```
Class 'String' has no instance getter 'missing'.

NoSuchMethodError : method not found: 'missing'
Receiver: "hello"
Arguments: []
```

More on this soon.

Index out of range?

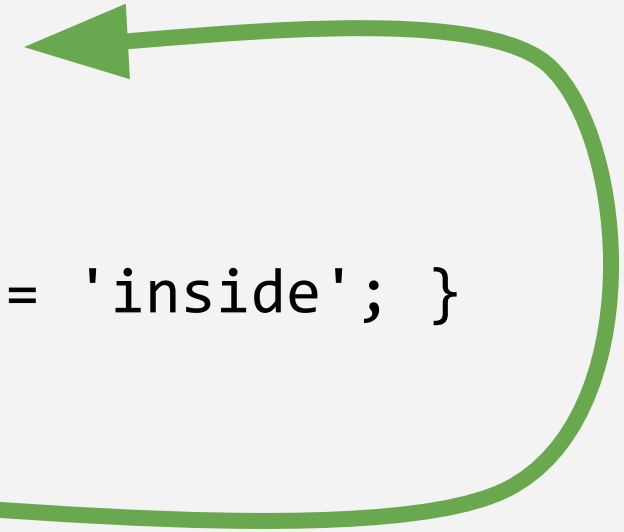
```
[][99]  // ??
```




```
RangeError: 99
```

Variable scope?

```
var foo = 'top-level';

void bar() {
  if (!true) { var foo = 'inside'; }

  print(foo);
}

main() { bar(); }  // ?? What will this print?
```



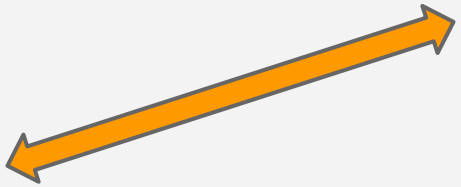
No hoisting

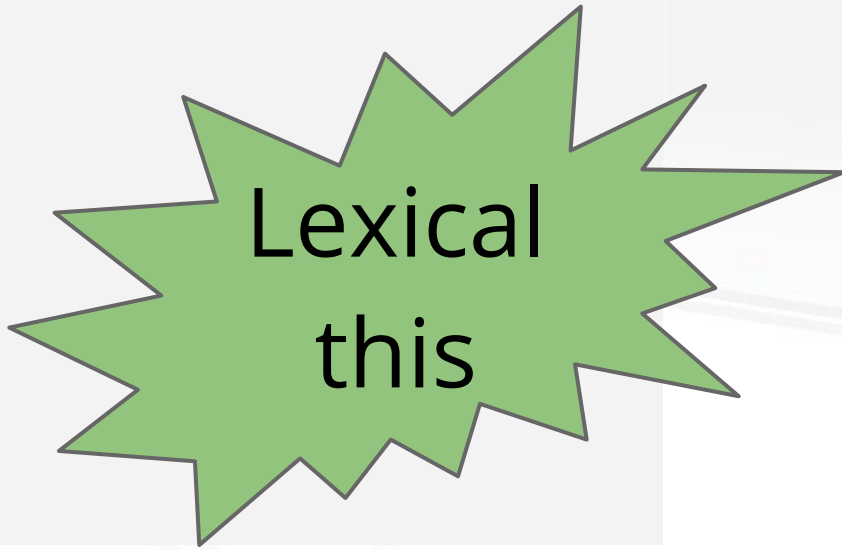
Logical

```
top-level
```

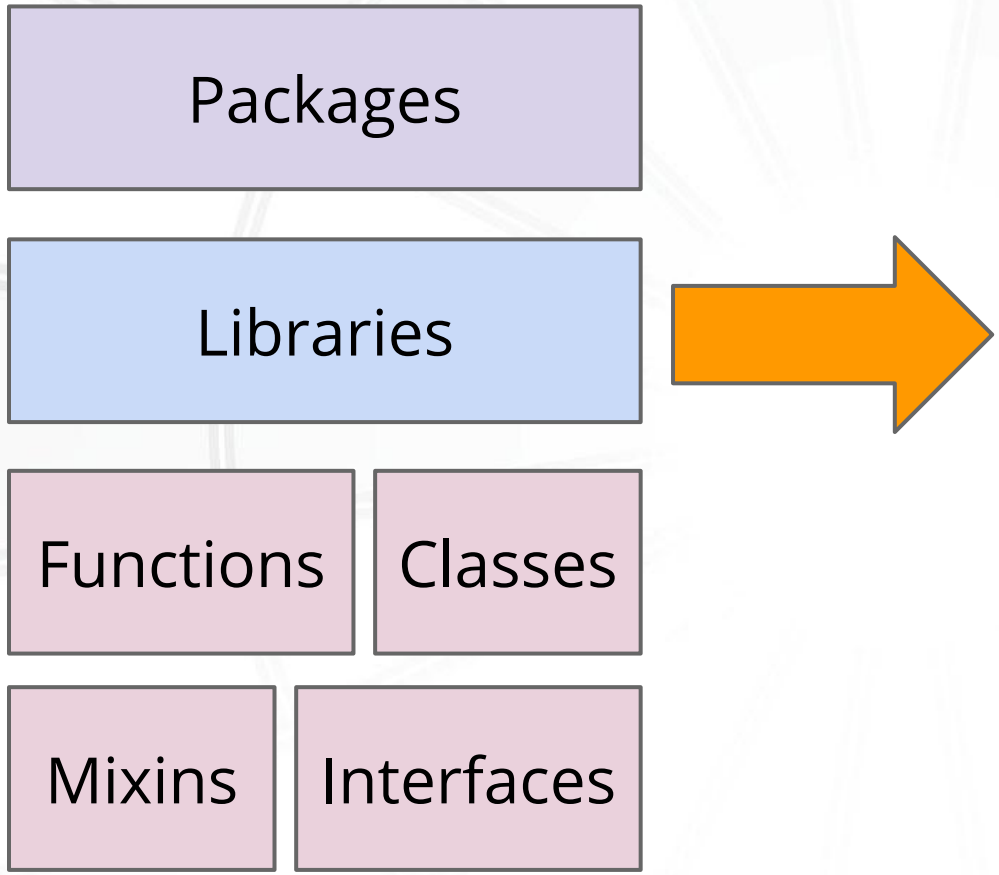
Scope of `this`?

```
class AwesomeButton {

  AwesomeButton(button) {
    button.onClick.listen((Event e) => this.atomicDinosaurRock());
  }

  atomicDinosaurRock() {
    /* ... */
  }
}
```





Scalable structure




```
library games;

import 'dart:math';
import 'players.dart';

class Darts {
  // ...
}

class Bowling {
  // ...
}

Player findOpponent(int skillLevel) {
  // ...
}
```

Language

Too many buttons

```
var button = new ButtonElement();
button.id = 'fancy';
button.text = 'Click Point';
button.classes.add('important');
button.onClick.listen((e) => addTopHat());

parentElement.children.add(button);
```

Yikes! Button is repeated 6 times!

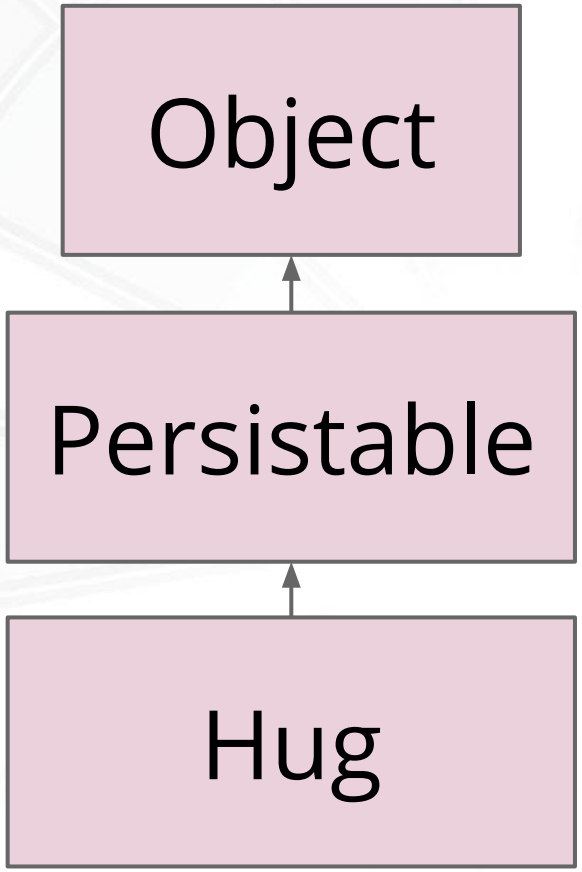
Method cascades

```
var button = new ButtonElement()
  ..id = 'fancy'
  ..text = 'Click Point'
  ..classes.add('important')
  ..onClick.listen((e) => addTopHat());

parentElement.children.add(button);
```

Inline initialization

```
parentElement.children.add(new ButtonElement()
  ..id = 'fancy'
  ..text = 'Click Point'
  ..classes.add('important')
  ..onClick.listen((e) => addTopHat()));
```

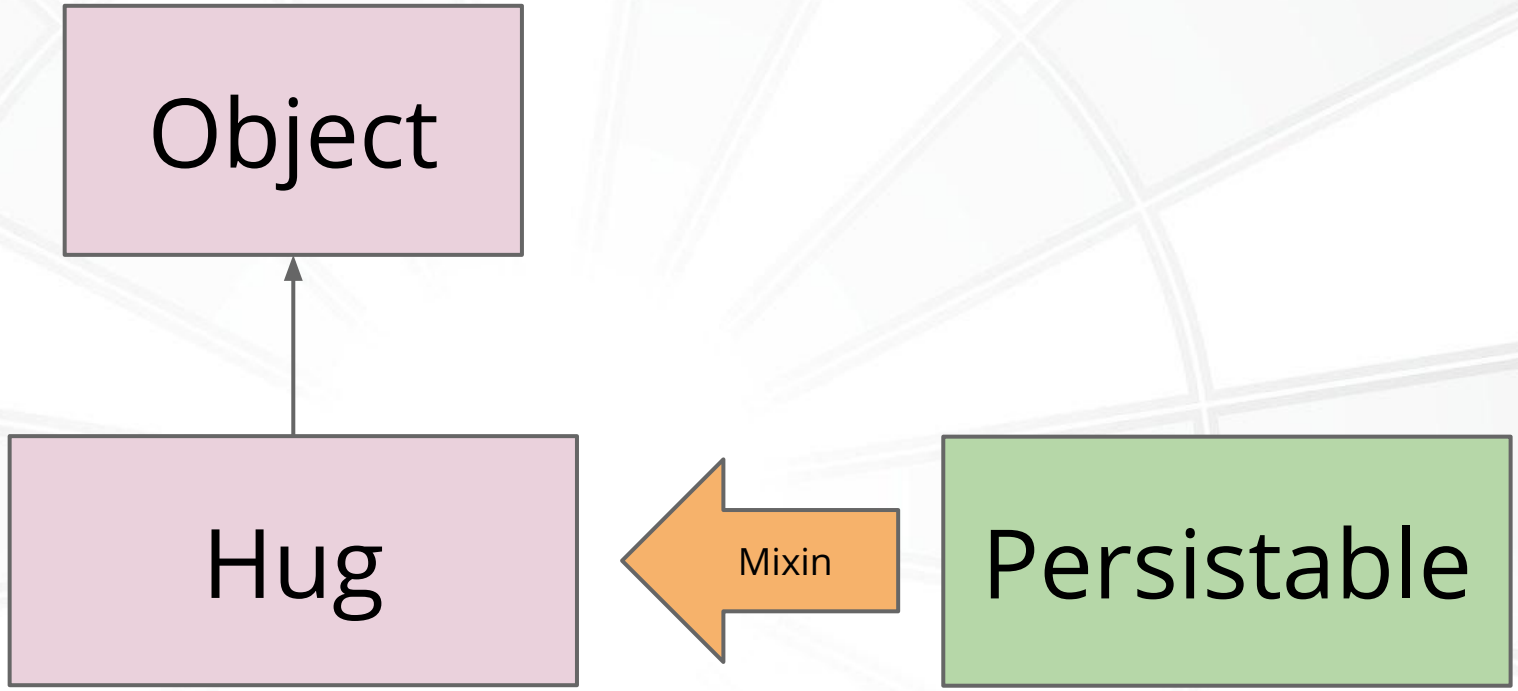
One of these things is not like the other



Hug is *not*
is-a Persistable

Don't pollute your inheritance tree!

Don't inherit, mixin!



Mixins

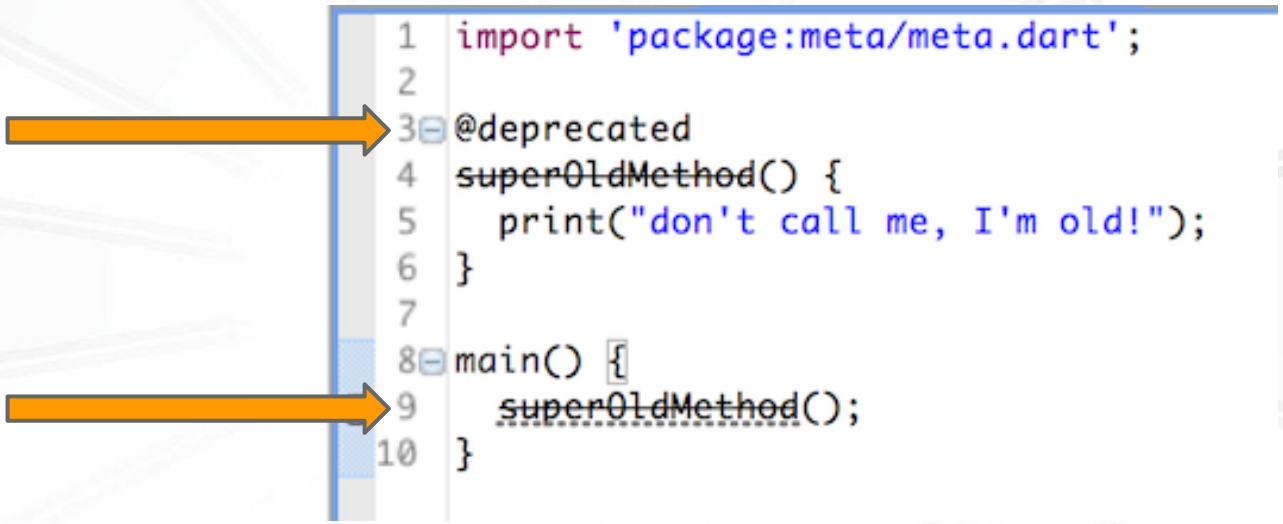
```
abstract class Persistable {
  save() { ... }
  load() { ... }
  toJson();
}

class Hug extends Object with Persistable {
  Map toJson() => {'strength':10};
}

main() {
  var embrace = new Hug();
  embrace.save();
}
```

Extend object & no constructors? You can be a mixin!

Apply the mixin.

Use methods from mixin.

Metadata



```dart
import 'package:meta/meta.dart';

@deprecated
superOldMethod() {
  print("don't call me, I'm old!");
}

main() {
  superOldMethod();
}
```

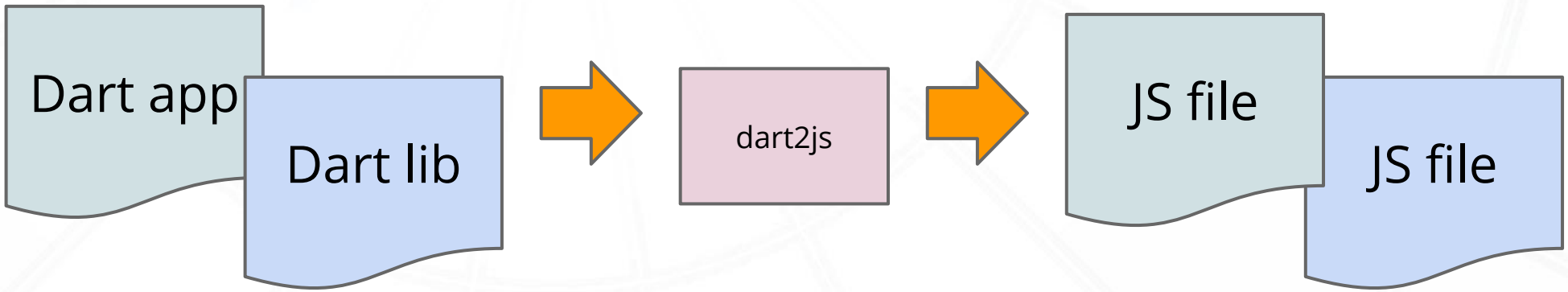
Lazy-load libraries

```
const lazy = const DeferredLibrary('my_lib');

@lazy
import 'my_lib.dart';

void main() {
  lazy.load().then((_) {
    print('library loaded');
    // use functions from my_lib
  });
}
```

Declare the library is deferred.

Mark the import

Use a Future to wait for library to load.

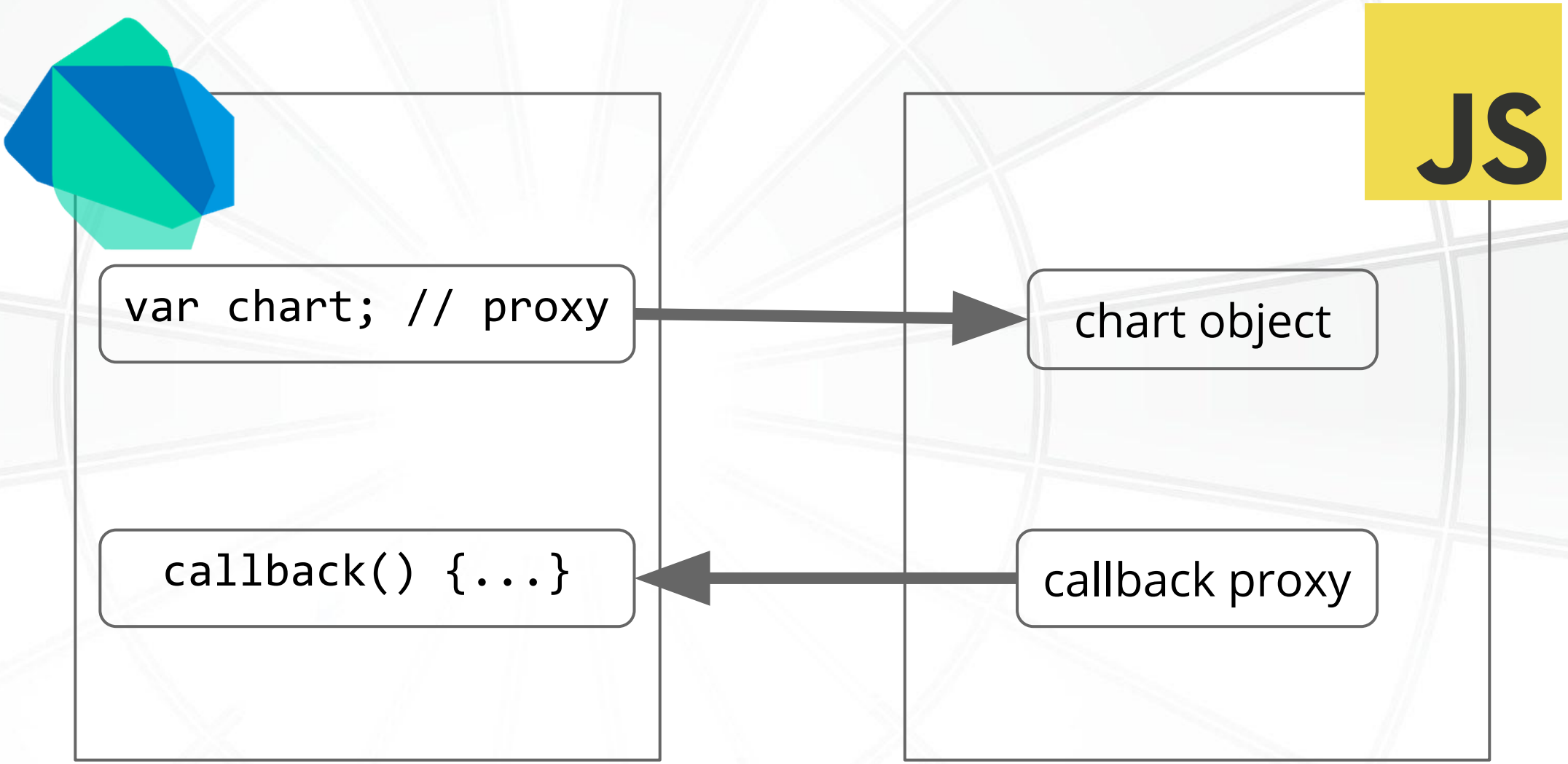


Libraries

JS-Interop

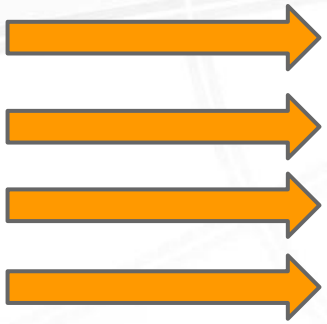
YES!

Proxies: the abstraction



JS-Interop example

```
var api = js.context.chartsApi;
var data = js.array([1,3,3,7]);
var chart = new js.Proxy(api.BubbleChart, query('#chart'));
chart.draw(data);
```

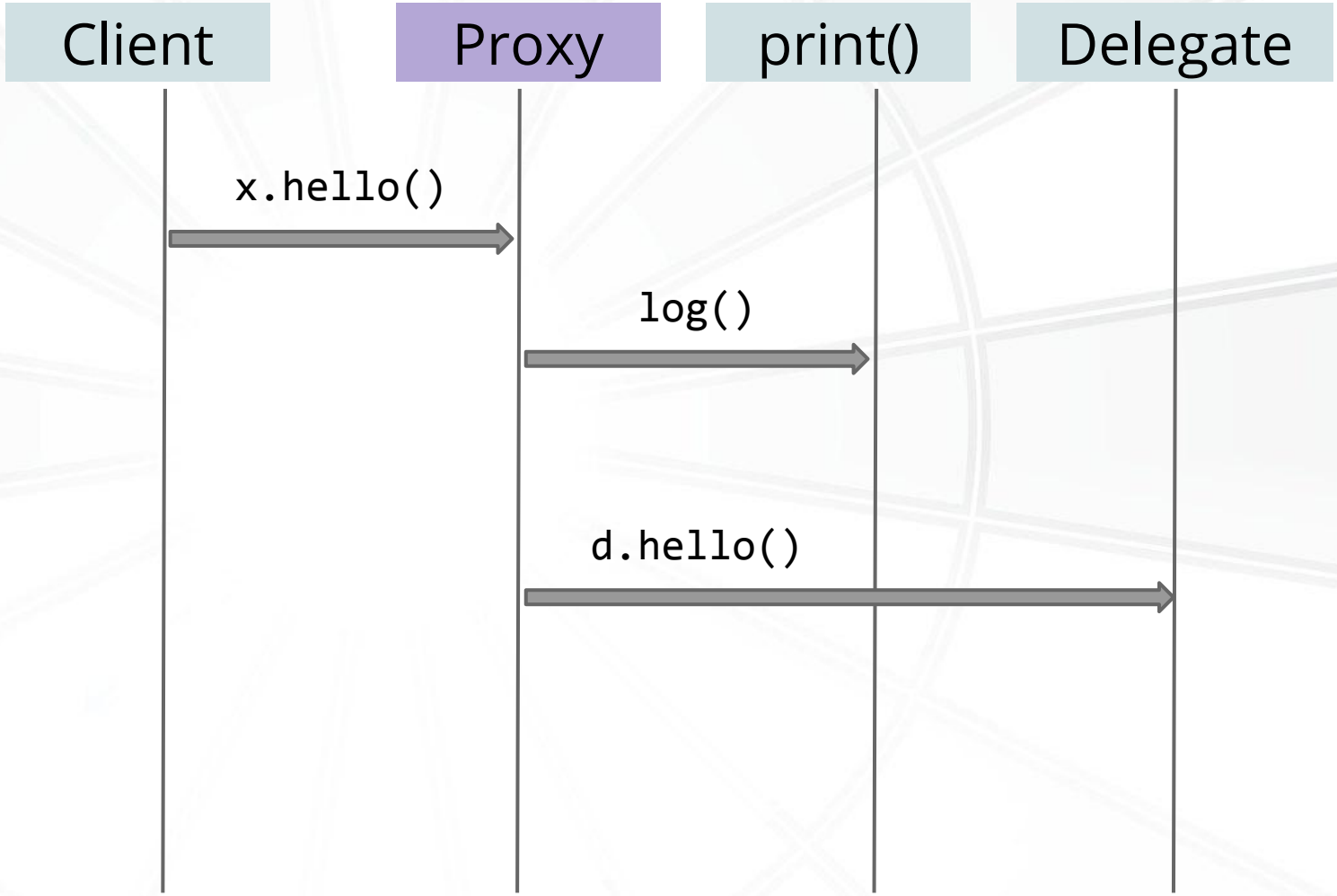


```
var api = chartsApi;
var data = [1,3,3,7];
var chart = new api.BubbleChart(querySelector('#chart'));
chart.draw(data);
```

Mirror-based reflection

- Source code *and* run-time
- Reflect on classes *and* instances
- Introspect *and* invoke

Using mirrors to build a logging proxy



Reflection and metaprogramming

```
import 'dart:mirrors';

class LoggingProxy {
  InstanceMirror mirror;
  LoggingProxy(delegate)
    : mirror = reflect(delegate);

  noSuchMethod(Invocation invocation) {
    var name = invocation.memberName;
    print('${name} was called');
    mirror.delegate(invocation);
  }
}
```

Import the mirrors library.

Reflection and metaprogramming

```
import 'dart:mirrors';

class LoggingProxy {
  InstanceMirror mirror;
  LoggingProxy(delegate)
    : mirror = reflect(delegate);

  noSuchMethod(Invocation invocation) {
    var name = invocation.memberName;
    print('${name} was called');
    mirror.delegate(invocation);
  }
}
```

Get a *mirror* of an object.

Reflection and metaprogramming

```
import 'dart:mirrors';

class LoggingProxy {
  InstanceMirror mirror;
  LoggingProxy(delegate)
    : mirror = reflect(delegate);

  noSuchMethod(Invocation invocation) {
    var name = invocation.memberName;
    print('${name} was called');
    mirror.delegate(invocation);
  }
}
```

Capture all calls to this proxy.

Reflection and metaprogramming

```
import 'dart:mirrors';

class LoggingProxy {
  InstanceMirror mirror;
  LoggingProxy(delegate)
    : mirror = reflect(delegate);

  noSuchMethod(Invocation invocation) {
    var name = invocation.memberName;
    print('${name} was called');
    mirror.delegate(invocation);
  }
}
```

Log the call.

Reflection and metaprogramming

```
import 'dart:mirrors';

class LoggingProxy {
  InstanceMirror mirror;
  LoggingProxy(delegate)
    : mirror = reflect(delegate);

  noSuchMethod(Invocation invocation) {
    var name = invocation.memberName;
    print('${name} was called');
    return mirror.delegate(invocation);
  }
}
```

Delegate the call through the mirror.

Reflection and metaprogramming

```
class Greeter {
  hello() => print("hello!");
}

void main() {
  var greeter = new LoggingProxy(new Greeter());
  greeter.hello();
}
```

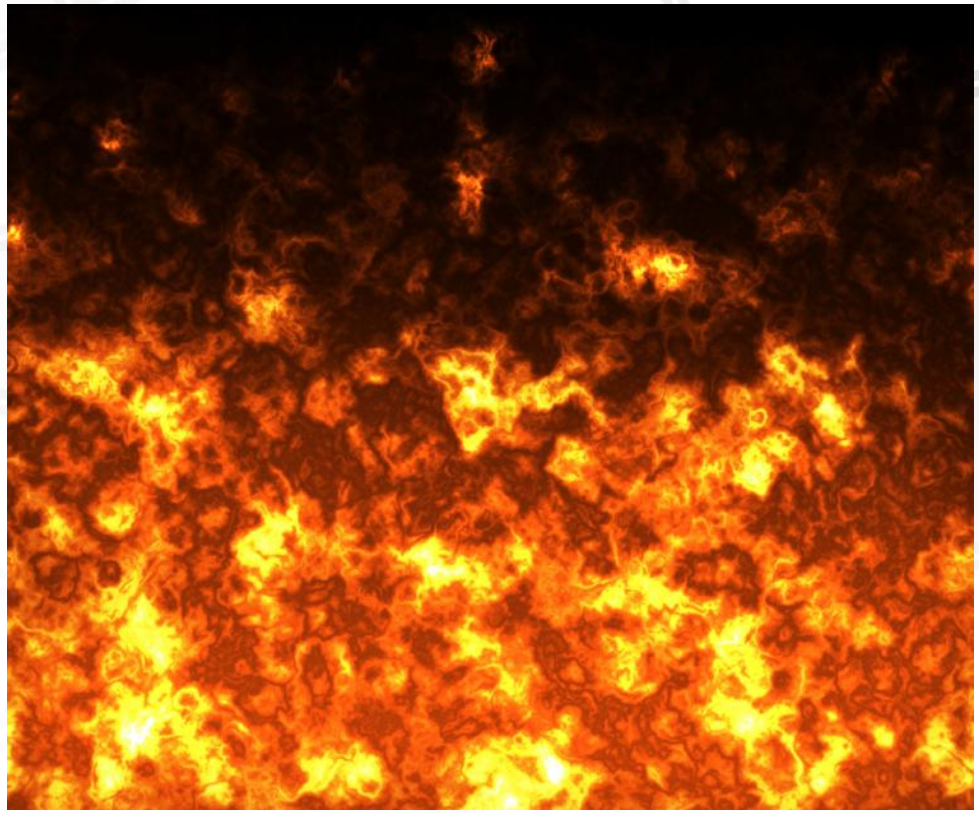
```
// Symbol("hello") was called
// hello!
```

From LoggingProxy

From Greeter

Async with callbacks

The web is an async world,
but *too many callbacks* leads to



Async with futures

Traditional callbacks

```
doStuff((results) {
  handle(results);
}, onError: (e) {
  handleError(e);
});
```

Futures

```
Future future = doStuff();
future.then(handle);
future.catchError(handleError);
```

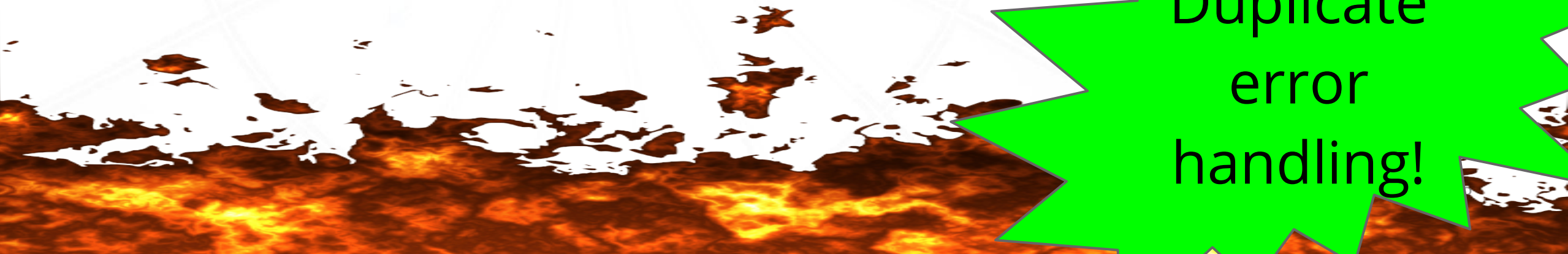
```
doStuff()
  .then(handle)
  .catchError(handleError);
```

Scary

```
catService.getCatData("cute", (cat) {
  catService.getCatPic(cat.imageId, (pic) {
    imageWorker.rotate(pic, 30, (rotated) {
      draw(rotated);
    });
  });
});
```

More scary

```
catService.getCatData("cute", (cat) {
  catService.getCatPic(cat.imageId, (pic) {
    imageWorker.rotate(pic, 30, (rotated) {
      draw(rotated, onError:(e) { draw(ohNoeImage); });
    }, onError: (e) { draw(ohNoeImage); });
  }, onError: (e) { draw(ohNoeImage); });
}, onError: (e) { draw(ohNoeImage); });
```



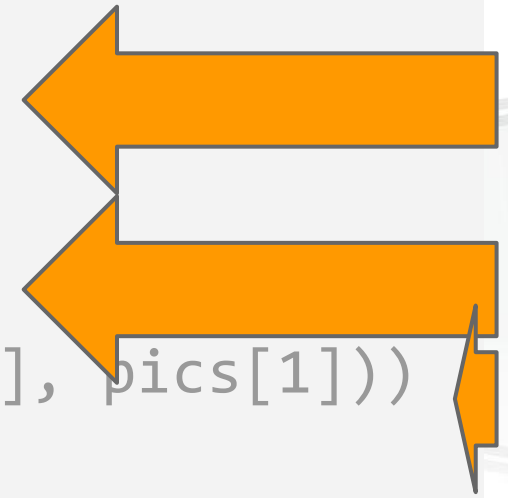
Duplicate error handling!

The Future looks bright

```
catService.getCat("cute")
  .then((cat) => catService.getCatPic(cat.imageId))
  .then((pic) => imageWorker.rotate(pic, 30))
  .then((rotated) => draw(rotated))
  .catchError((e) => print("Oh noes!"));
```

Composing futures

```
Future cute = catService.getPic("cute");
Future nyan = catService.getPic("nyan");

Future.wait([cute, nyan])
  .then((pics) => imageWorker.blend(pics[0], pics[1]))
  .then((cuteNyan) => draw(cuteNyan))
  .catchError((e) => print("Oh noes!"));
```



Request two pics

Wait for both

Work with both pics.

Futures fire only once...

What about recurring events?

Streams

Streams are the *repeating* analog to Futures.

Nearly all repeating events in Dart are Streams.

api.dartlang.org

Element abstract class

...

final Stream<KeyboardEvent> onKeyPress

```
query('textarea').onKeyPress.listen((event) {
  var char = new String.fromCharCode(e.charCode);
  print('char=$char');
});
```

api.dartlang.org

Element abstract class

...

final Stream<KeyboardEvent> onKeyPress

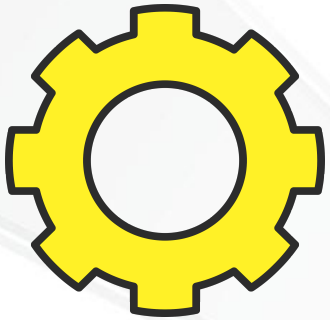
```
query('textarea').onKeyPress
  .where((e) => e.keyCode >= 32 && e.keyCode <= 122)
  .map((e) => new String.fromCharCode(e.charCode))
  .first
  .then((char) => print('First char=$char'));
```

HTML and Web Components

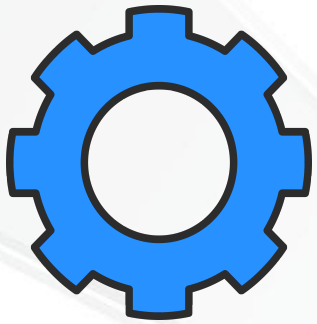
- More API Dartification of dart:html
  - Collections
  - Future, Stream
- Web UI: custom elements & templates
- Vendor prefix elimination

Find out more at 3:30/Room 6 in Dart: HTML of the Future, Today!

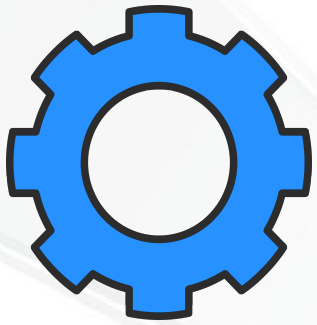
Tools & Ecosystem



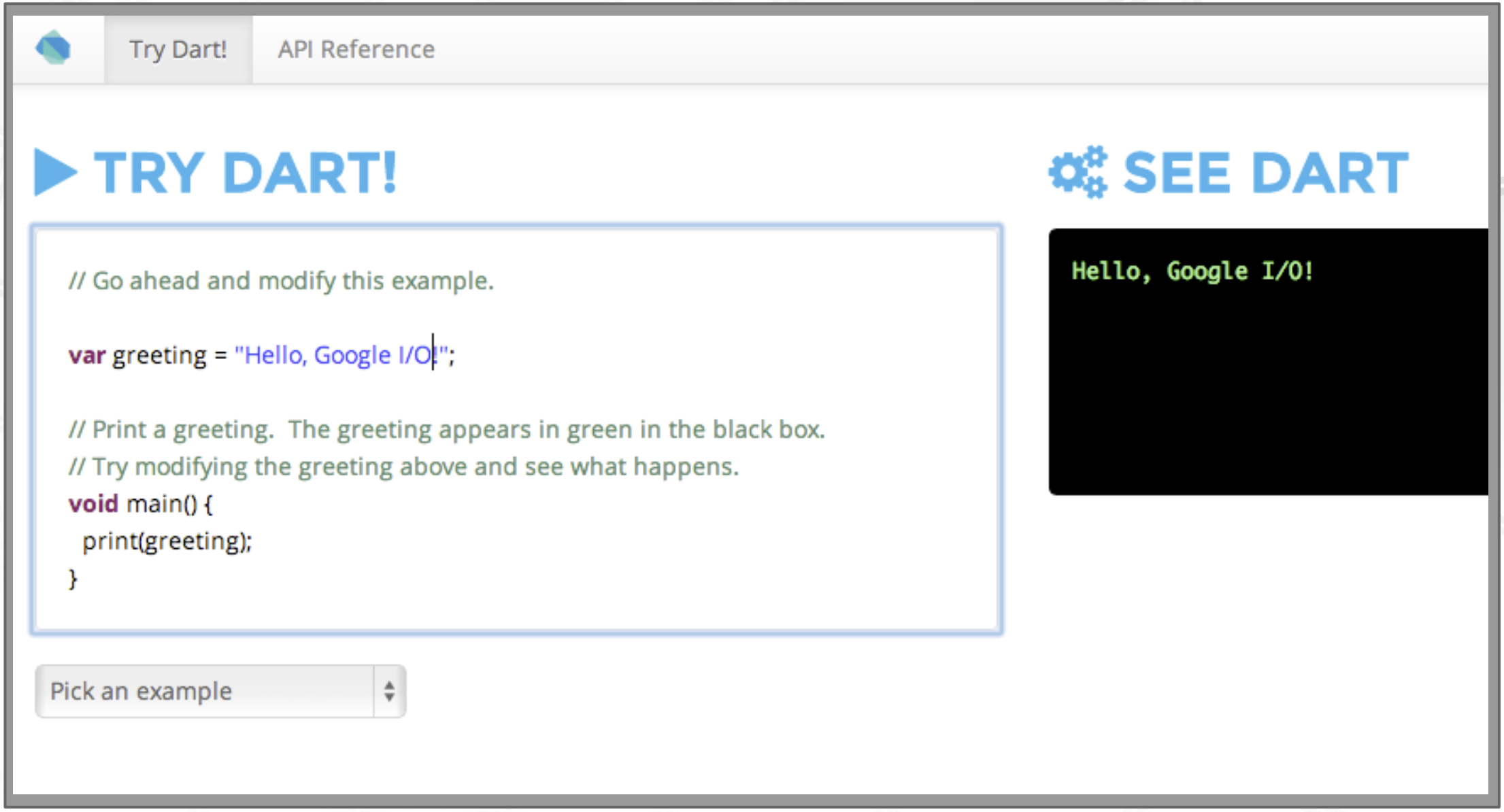
Fast development cycles



#dartlang



try.dartlang.org

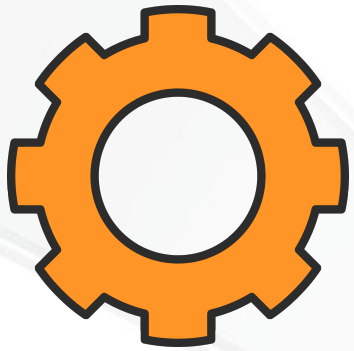


```
test('add', () {
  var answer = add(1, 2);
  expect(answer, equals(3));
});
```

Headless Chrome, command-line testing

drone.io

Continuous integration, native Dart support

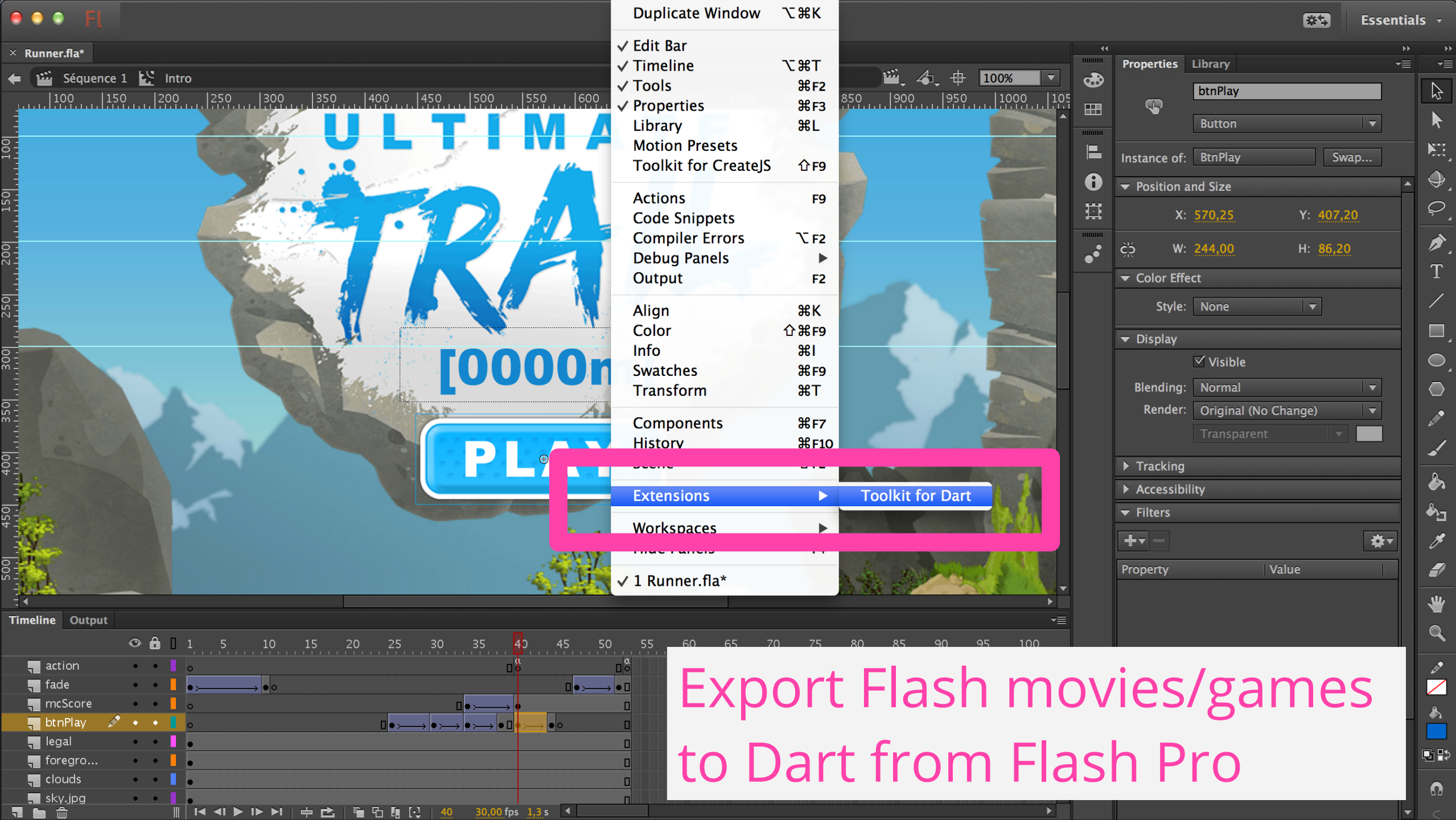


Pub, a package manager for Dart

- Download
- Manage
- Publish
- Browse

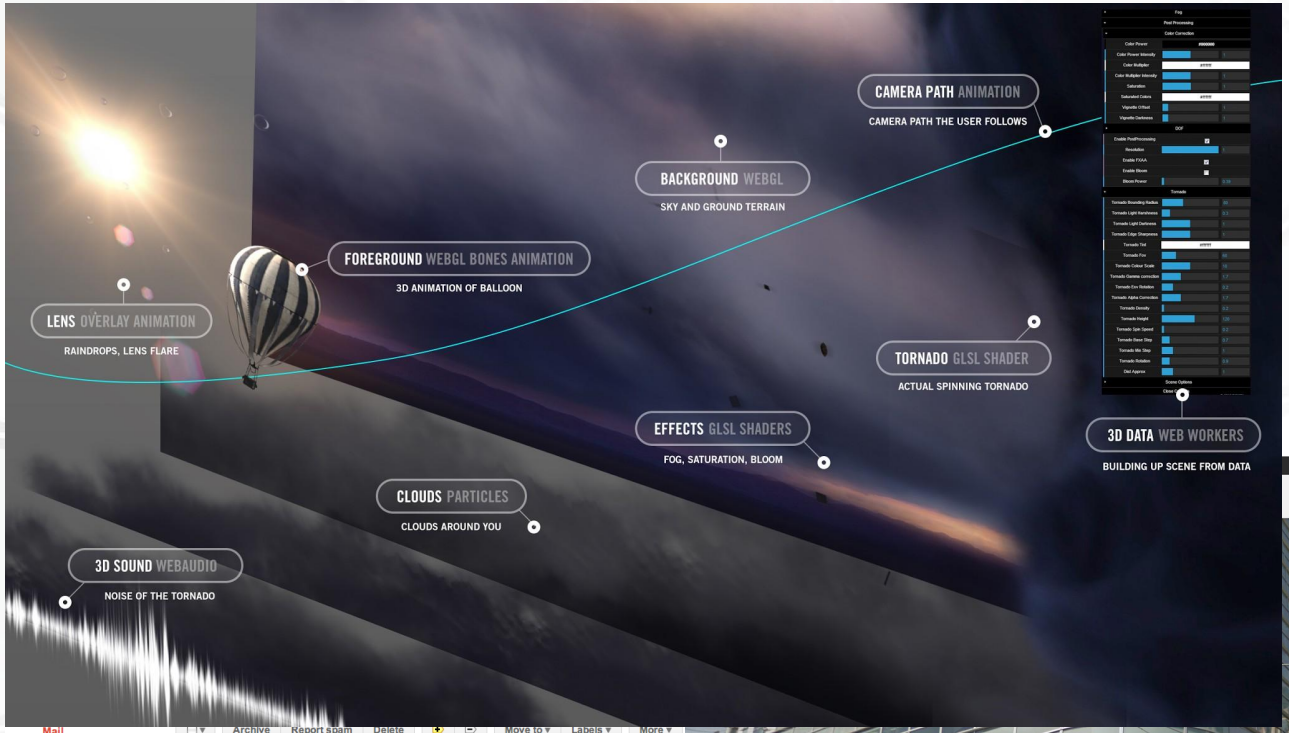
Available in **pub.dartlang.org**:

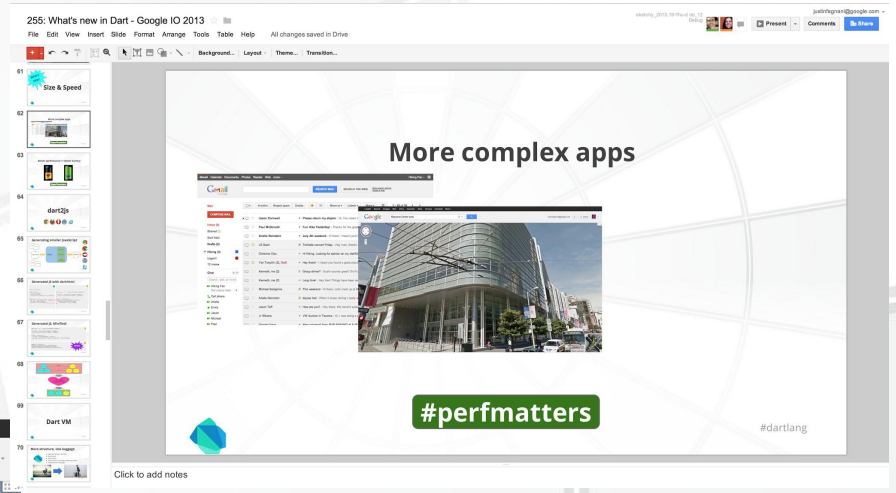
- MVC frameworks
- Template systems
- Google APIs
- Encryption
- Server-side frameworks
- DB drivers
- Parsers
- Game libraries
- Much, much more!

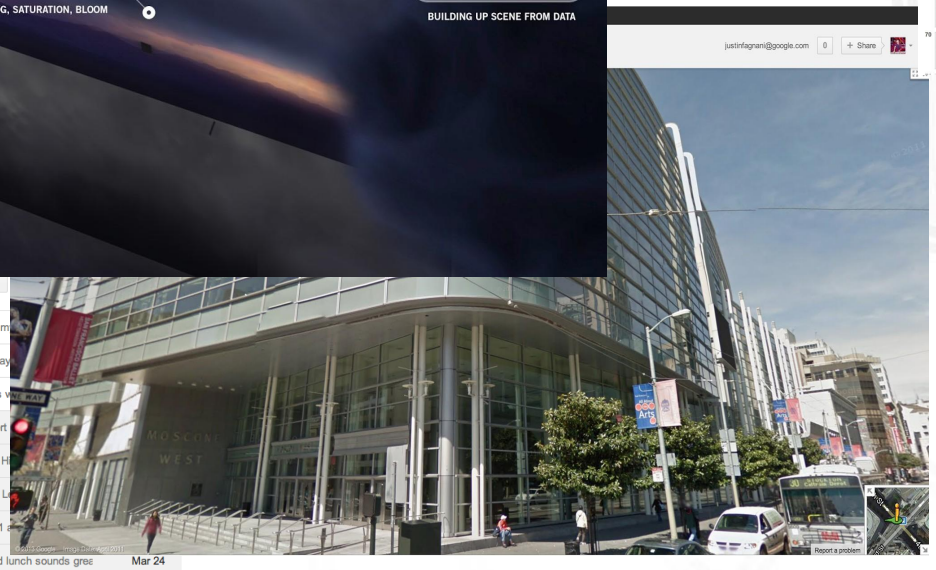

Essentials
Runner.fla*
Séquence 1
Intro
100%
Duplicate Window ⌥⌘K
Edit Bar
Timeline ⌥⌘T
Tools ⌘F2
Properties ⌘F3
Library ⌘L
Motion Presets
Toolkit for CreateJS ⇧F9
Actions F9
Code Snippets
Compiler Errors ⌥F2
Debug Panels
Output F2
Align ⌘K
Color ⇧⌘F9
Info ⌘I
Swatches ⌘F9
Transform ⌘T
Components ⌘F7
History ⌘F10
Extensions
Toolkit for Dart
Workspaces
1 Runner.fla*
Properties
Library
btnPlay
Button
Instance of: BtnPlay
Swap...
Position and Size
X: 570,25
Y: 407,20
W: 244,00
H: 86,20
Color Effect
Style: None
Display
Visible
Blending: Normal
Render: Original (No Change)
Transparent
Tracking
Accessibility
Filters
Property
Value
Timeline
Output
action
fade
mcScore
btnPlay
legal
foregro...
clouds
sky.jpg
30,00 fps
1,3 s
Export Flash movies/games to Dart from Flash Pro

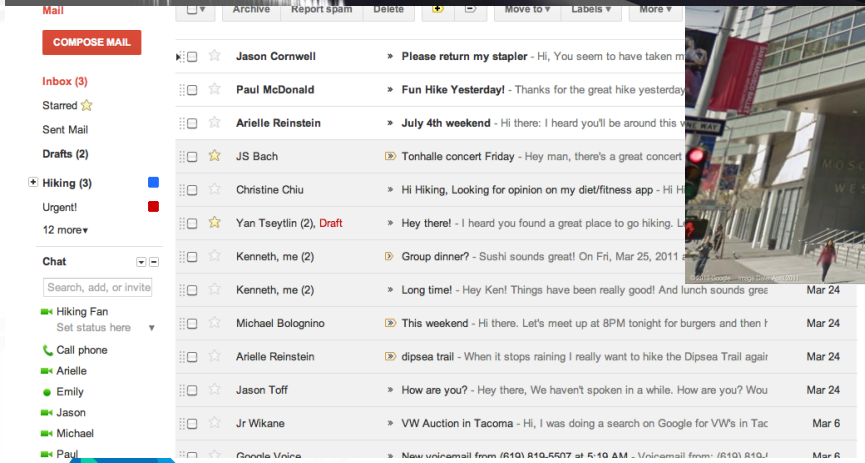
Size & Speed

More complex apps



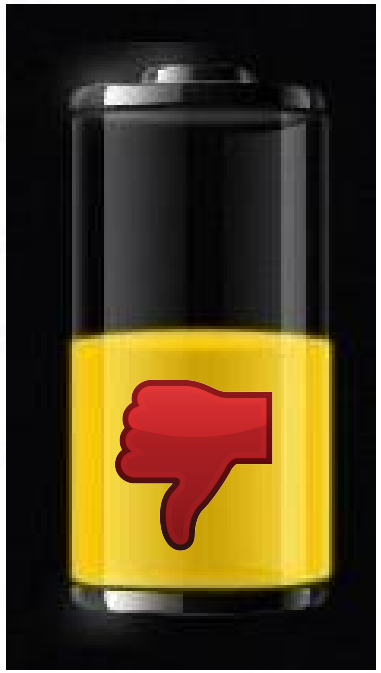






#perfmatters

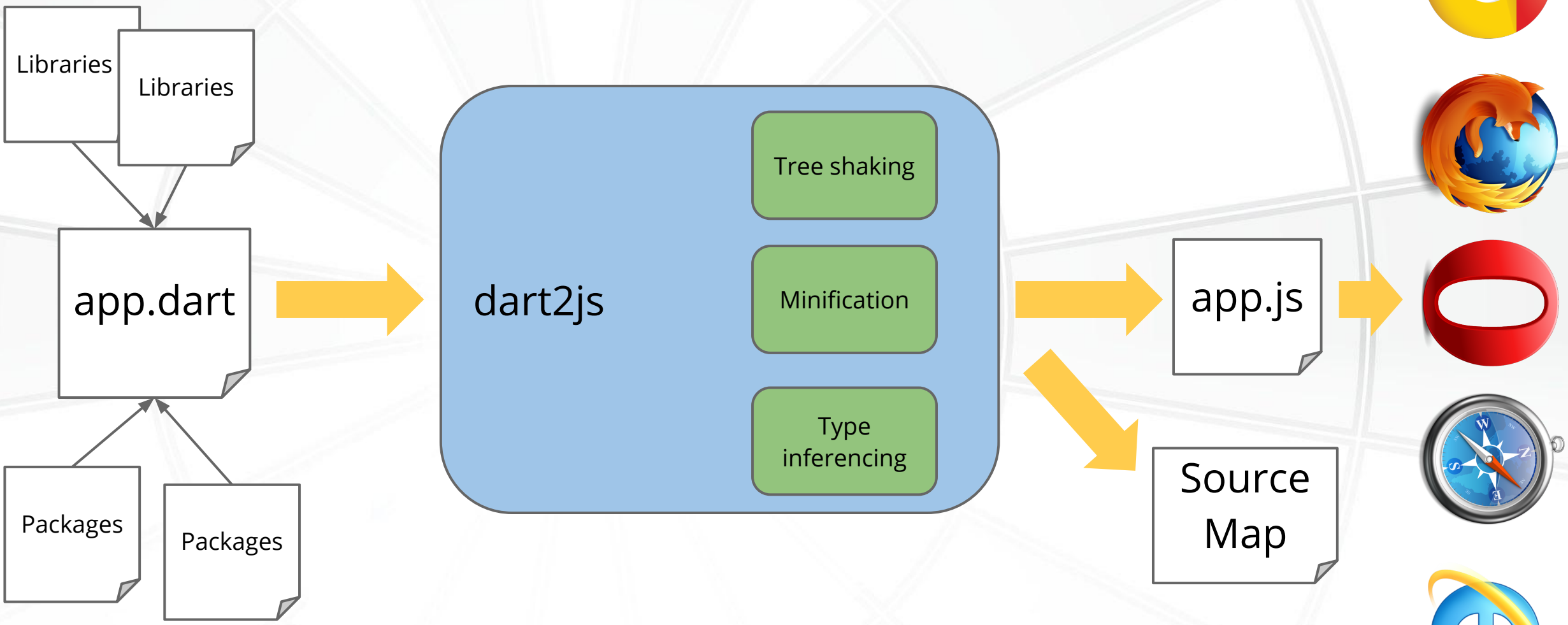
Better performance == Better battery



#perfmatters

dart2js

Generating smaller JavaScript



Generated JS with dart:html

```dart
import 'dart:html';

class Person {
  String firstName;
  String lastName;
  Person(this.firstName, this.lastName);
}

main() {
  var bob = new Person('Bob', 'Smith');
  var msg = query('#msg');
  msg.text = bob.firstName;
}
```

JS

```javascript
$$.Person = {"": "Object;firstName,lastName"};

$.Person$ = function(firstName, lastName) {
  return new $.Person(firstName, lastName);
};

$.main = function() {
  var bob = $.Person$("Bob", "Smith");
  document.querySelector("#msg")
    .textContent = bob.firstName;
};
```

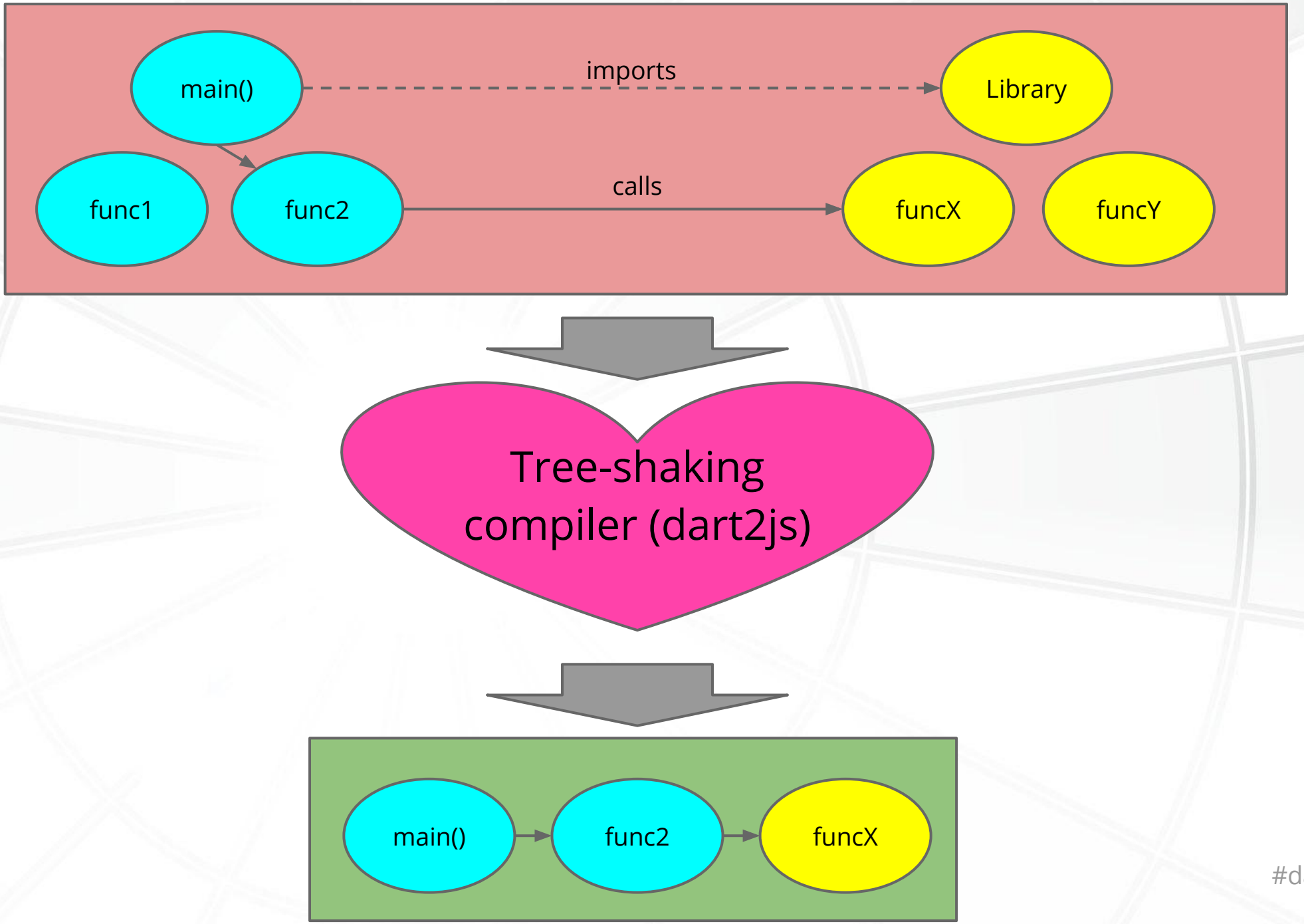
Generated JS, minified!

JS

```
$$.Person = {"": "Object;firstName,lastName"};

$.Person$ = function(firstName, lastName) {
  return new $.Person(firstName, lastName);
};

$.main = function() {
  var bob = $.Person$("Bob", "Smith");
  document.querySelector("#msg").textContent = bob.firstName;
};
```

JS

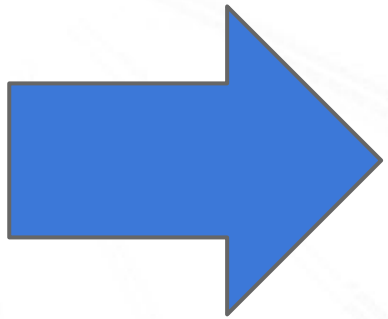
```
$$.mM={"":"a;Sz,dq"}
$.PH=function(a,b){return new $.mM(a,b)}
$.E2=function(){var z=$.PH("Bob","Smith")
document.querySelector("#msg").textContent=z.Sz}
```


main()
imports
Library
func1
func2
calls
funcX
funcY
Tree-shaking
compiler (dart2js)
main()
func2
funcX

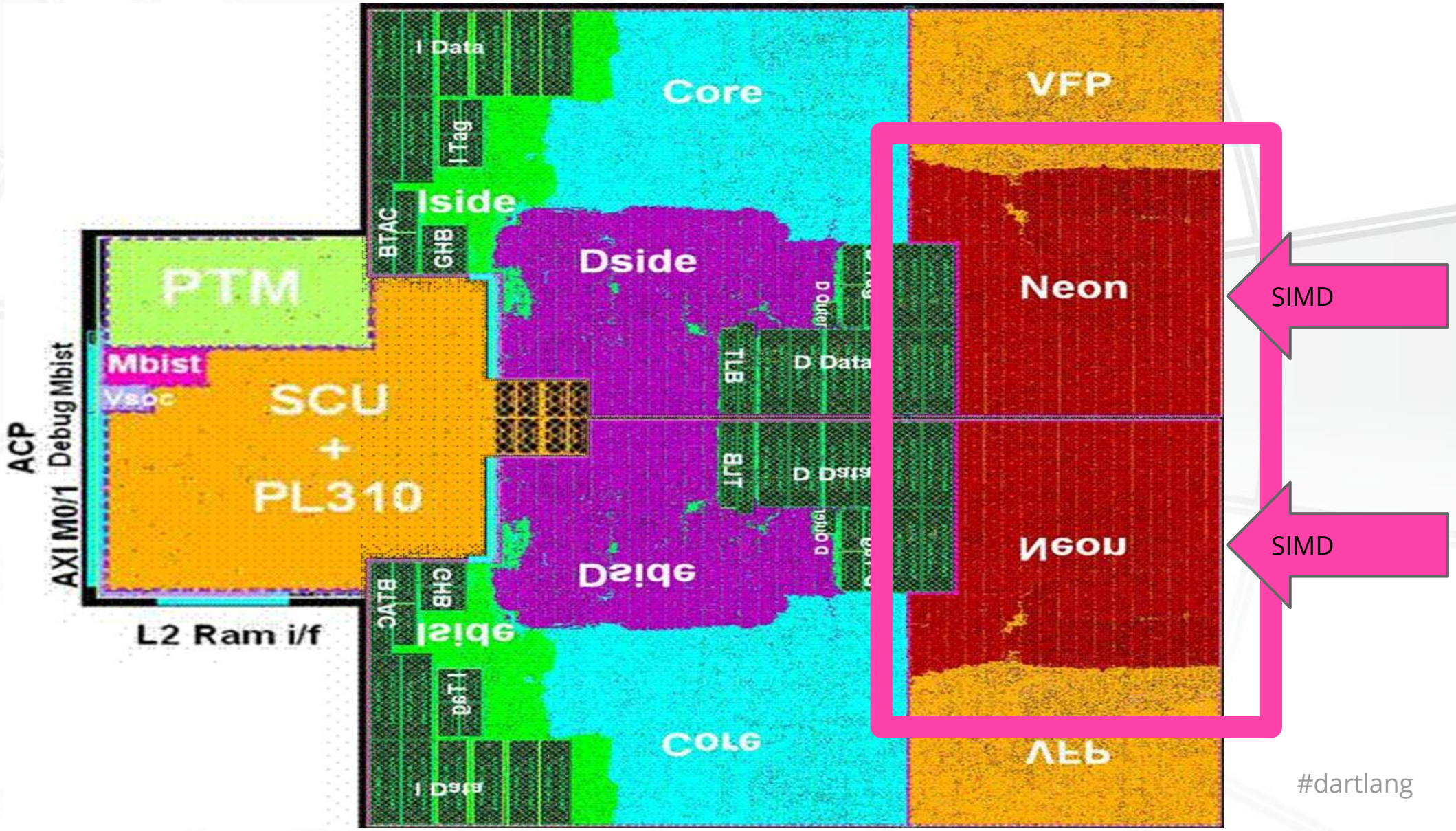
Dart VM

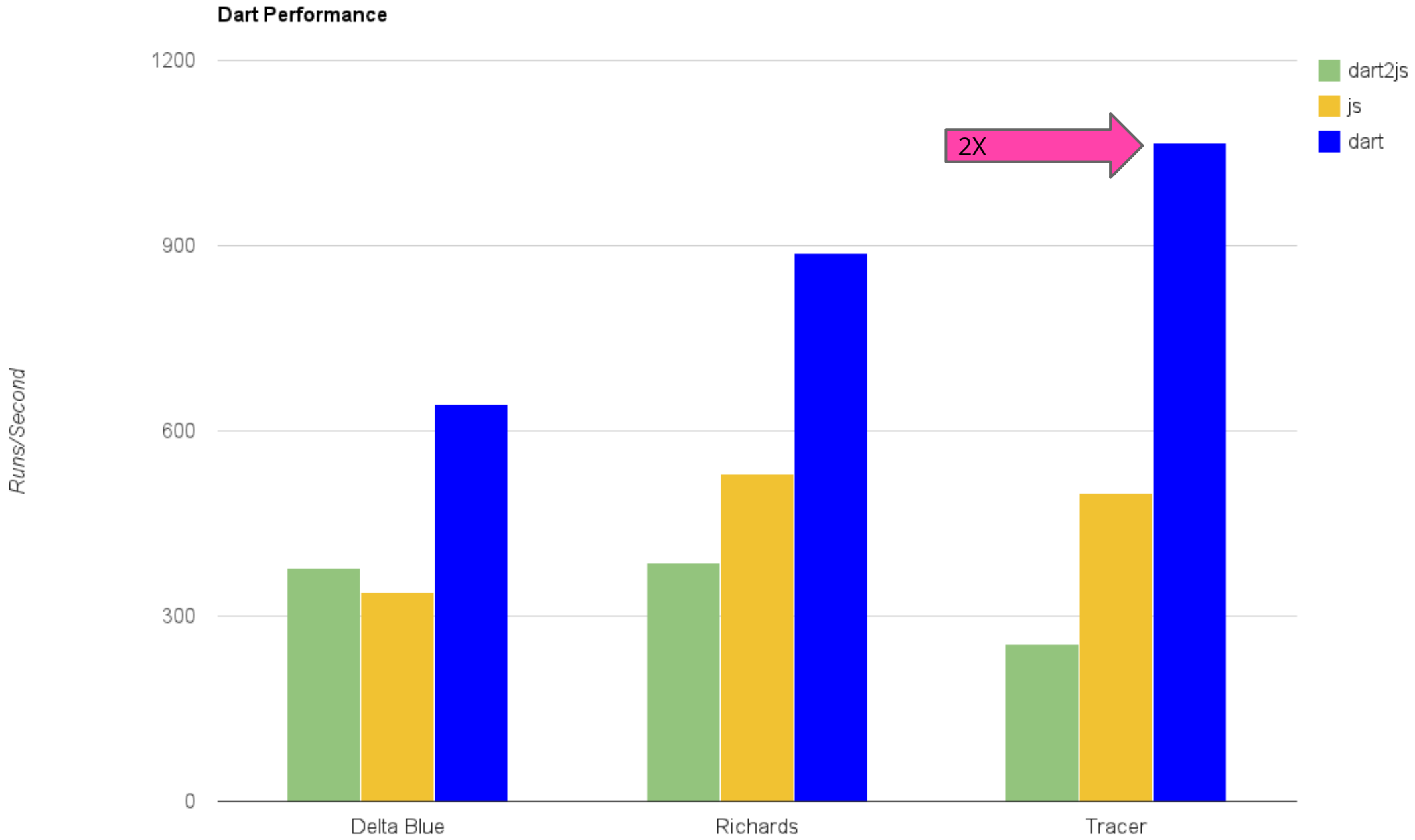
More structure, less baggage

- Explicit and static structure
- Real arrays
- Real classes
- Direct calls, no prototype chains to check
- Globally track field types



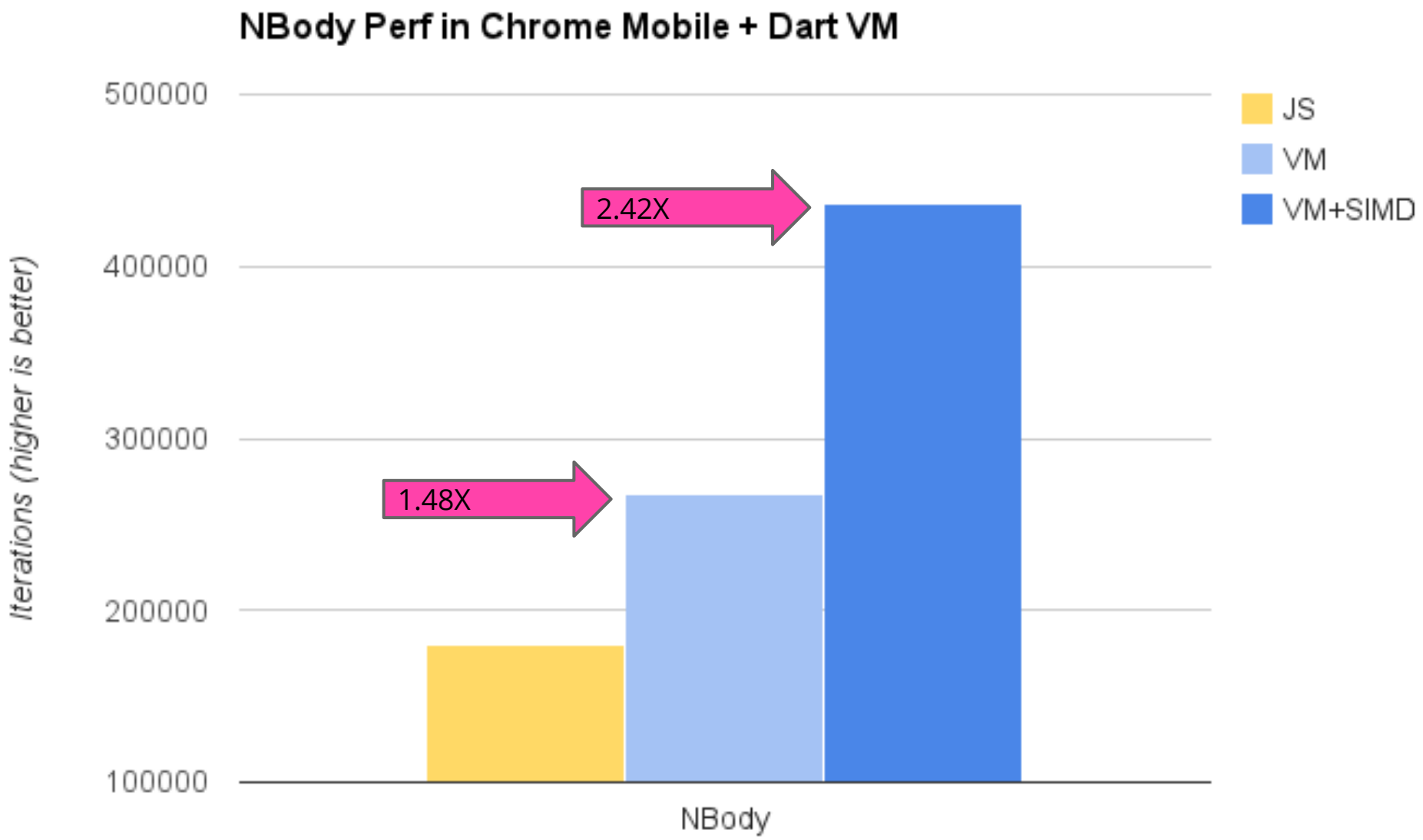
Unlock more of your CPU





dartlang.org/performance

Higher is better, as of 2013/05/12



x86, Chrome for Android

Higher is better, as of 2013/05/12

rtlang

There's more new stuff!

- Server-side
- Testing
- Isolates for concurrency
- *Lots more...*

Try Dart!

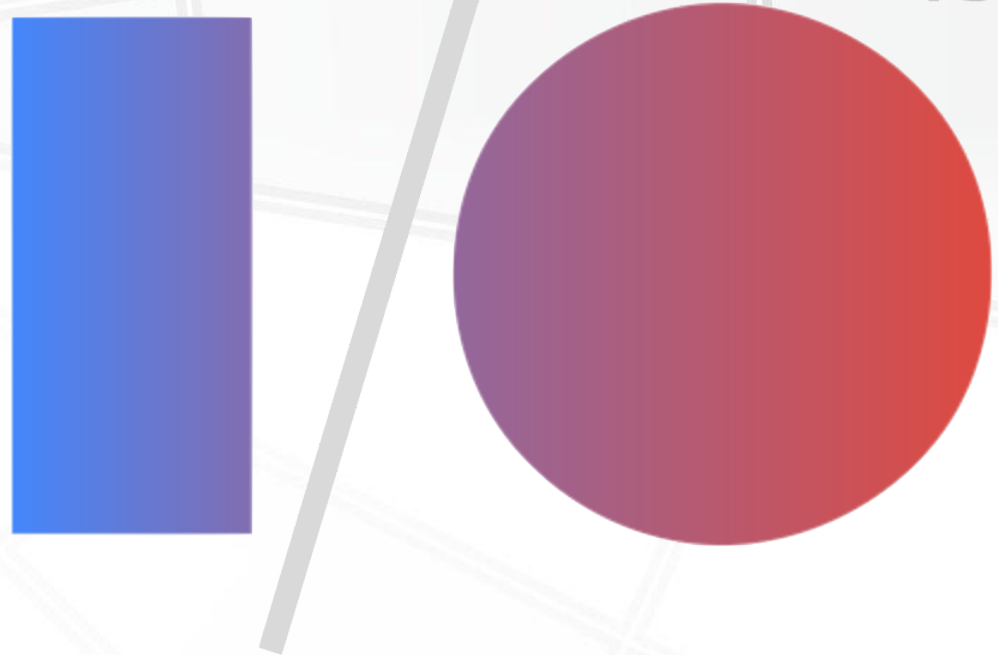
- **Download** from dartlang.org
- **Join** +Dartisans
- **Send pull requests** at Github
- **Ask** on Stack Overflow

open source initiative

#dartlang

Lots more Dart at I/O!

- Talk: Dart's HTML of the Future, today!
- Code lab (Friday)
- Office hours
- Demo booth



DART

- Stable language
- Stable core libs
- Compiles to JavaScript
- Evolved platform
- Commitment

=

Ready for your app!

#dartlang

Thank You!

Find us at ***dartlang.org***

#dartlang