DART

Dart and Web Components - Scalable, Structured Web Apps

Seth Ladd, Developer Advocate, Dart

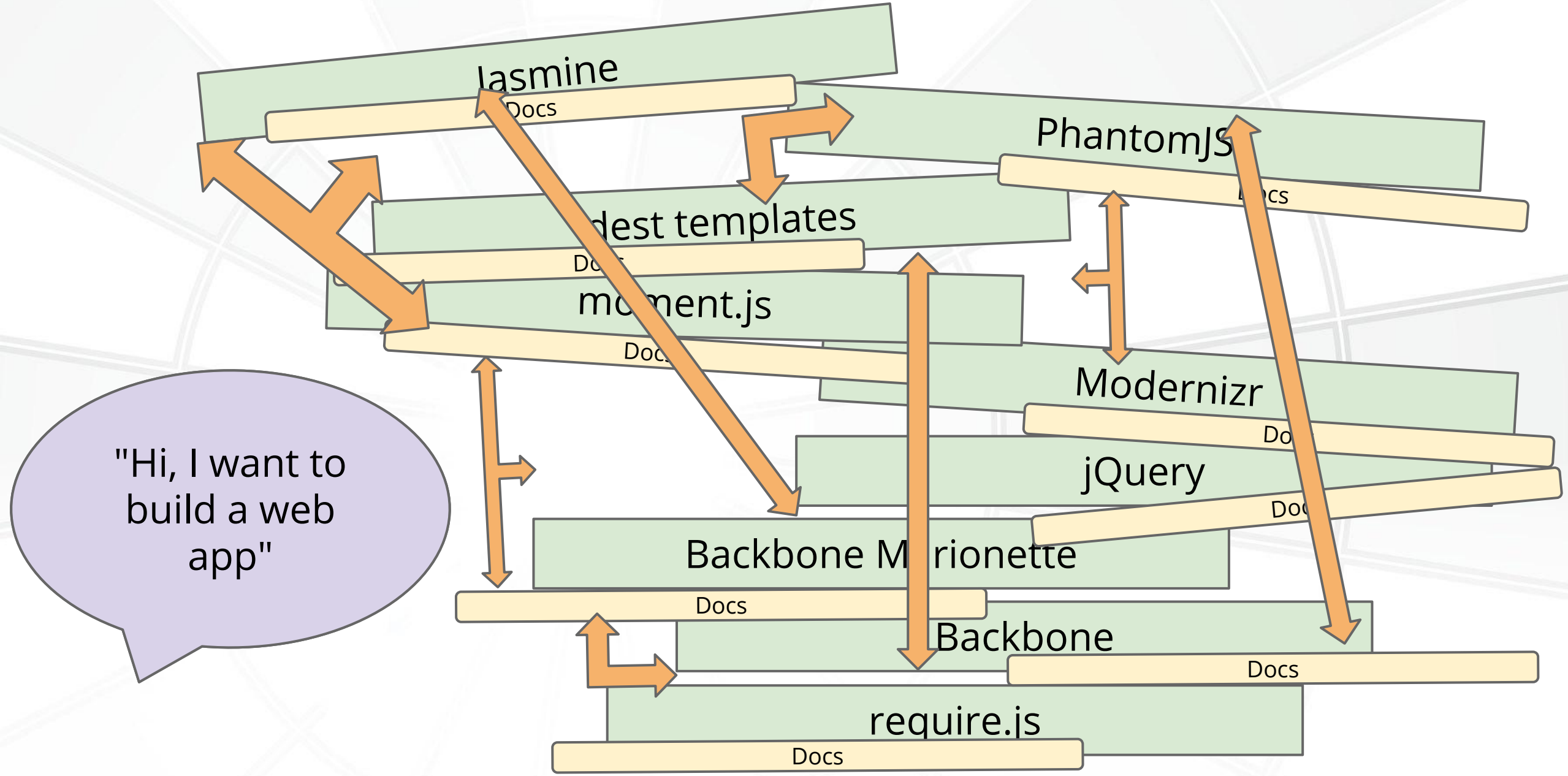
JavaZone 2013

#dartlang

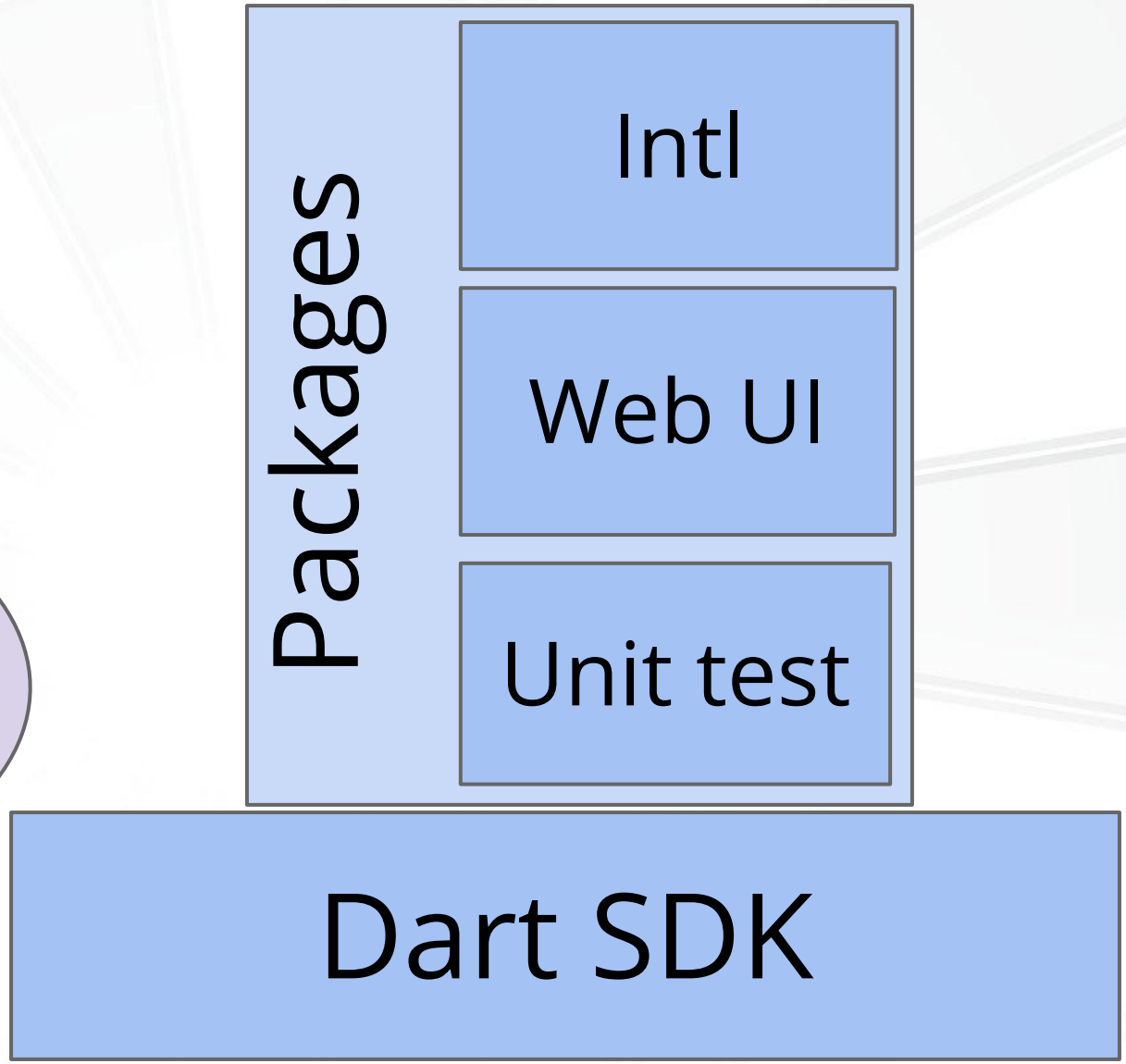
DART

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

#dartlang


Jasmine
Docs
PhantomJS
dest templates
moment.js
Modernizr
jQuery
Backbone Marionette
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

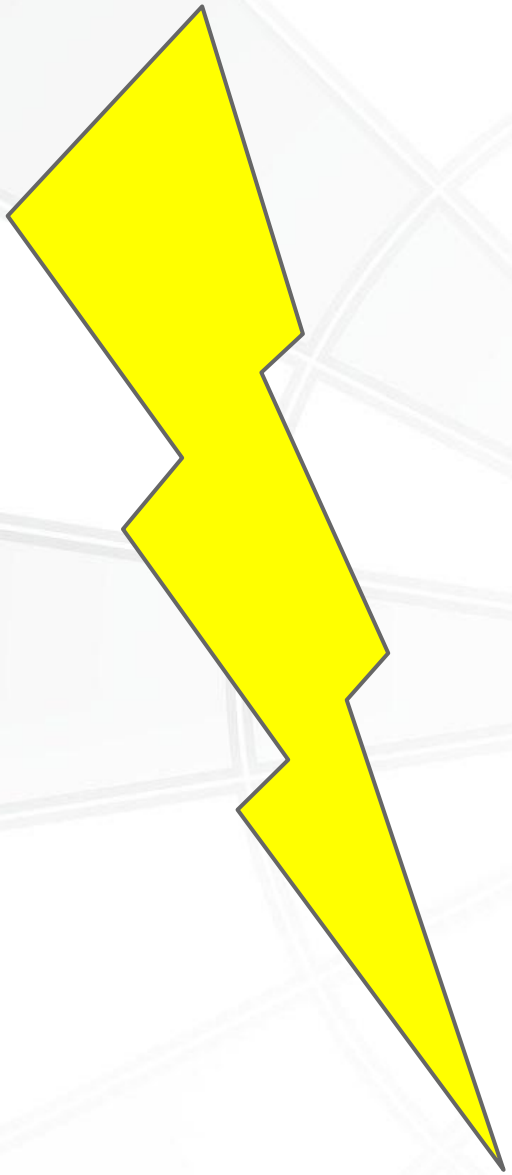
Low Productivity

- 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";
}
```

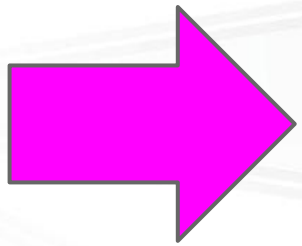
One-line function

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

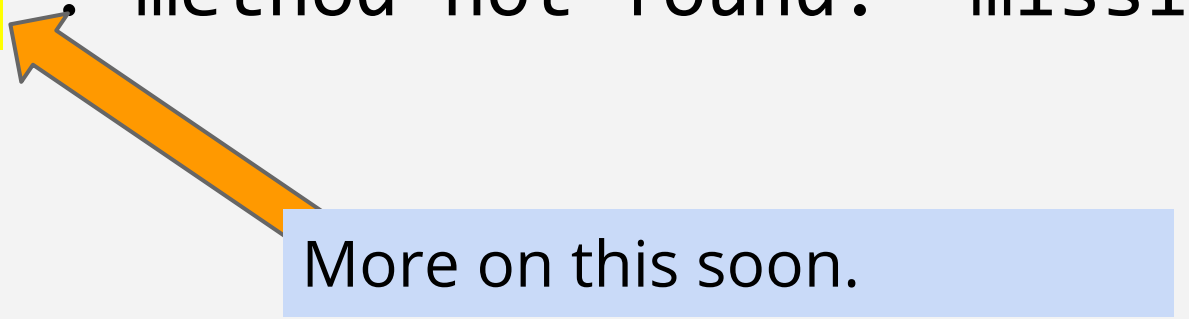
Examples:

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

Missing getter?

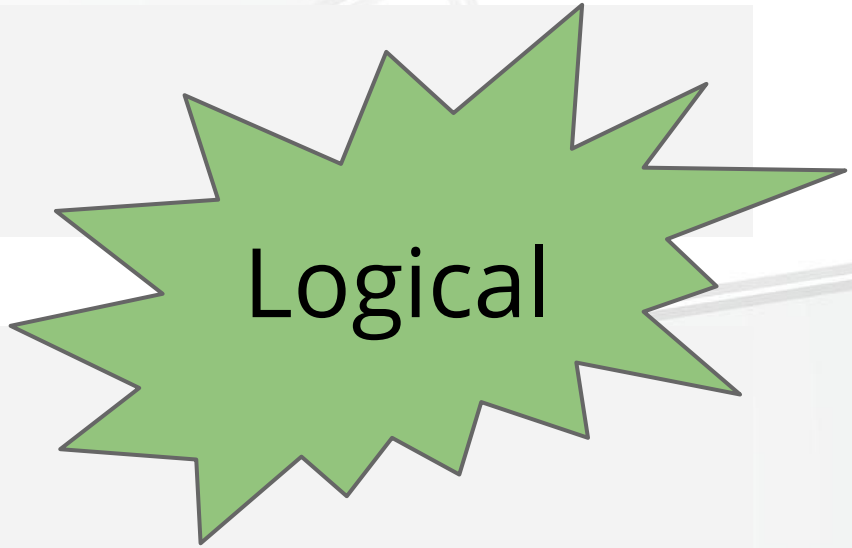
```
"hello".missing  // ??
```


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

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



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?
```

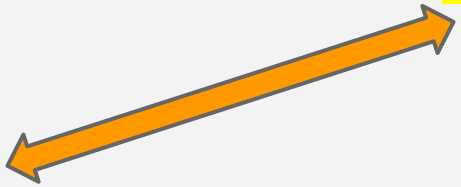
```
top-level
```

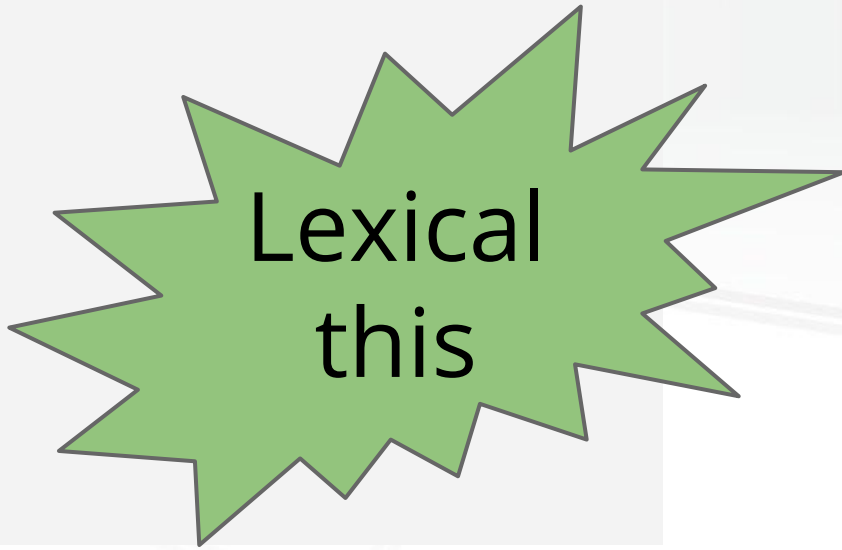
No hoisting

Logical

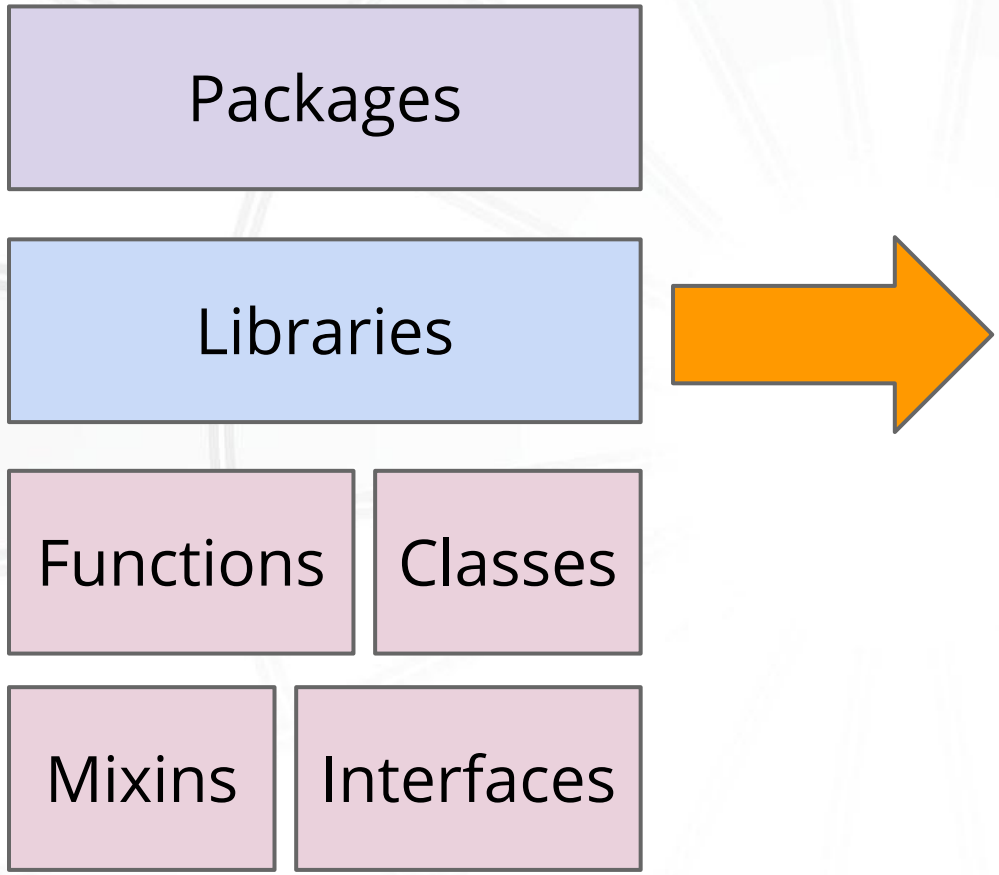
Scope of `this`?

```
class AwesomeButton {

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

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

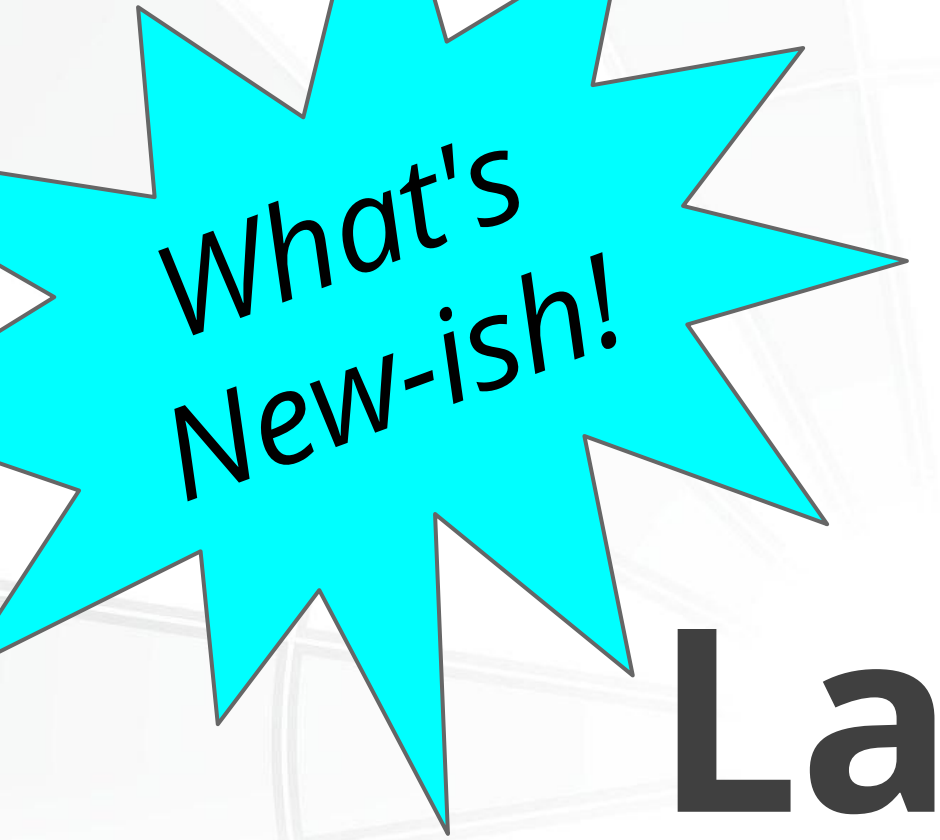




Scalable structure




```
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!

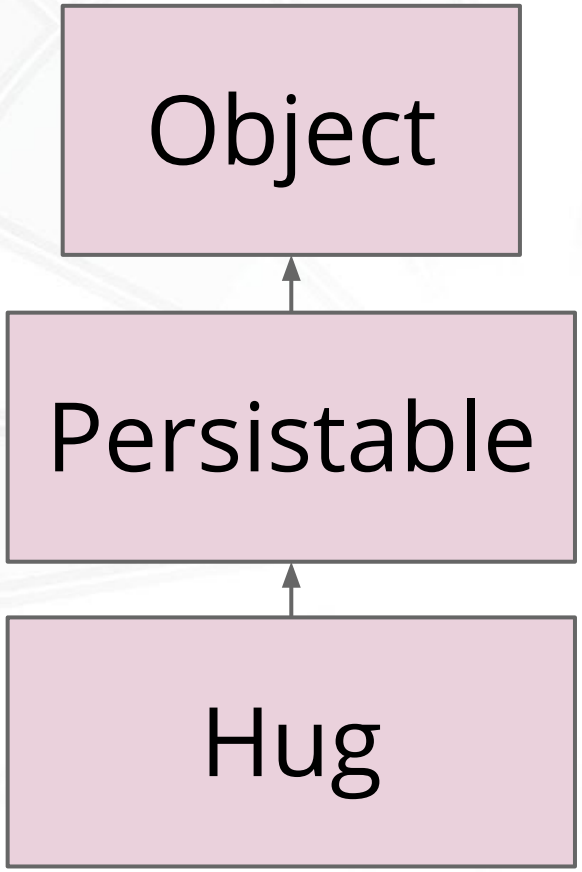
Method cascades

```
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()));
```

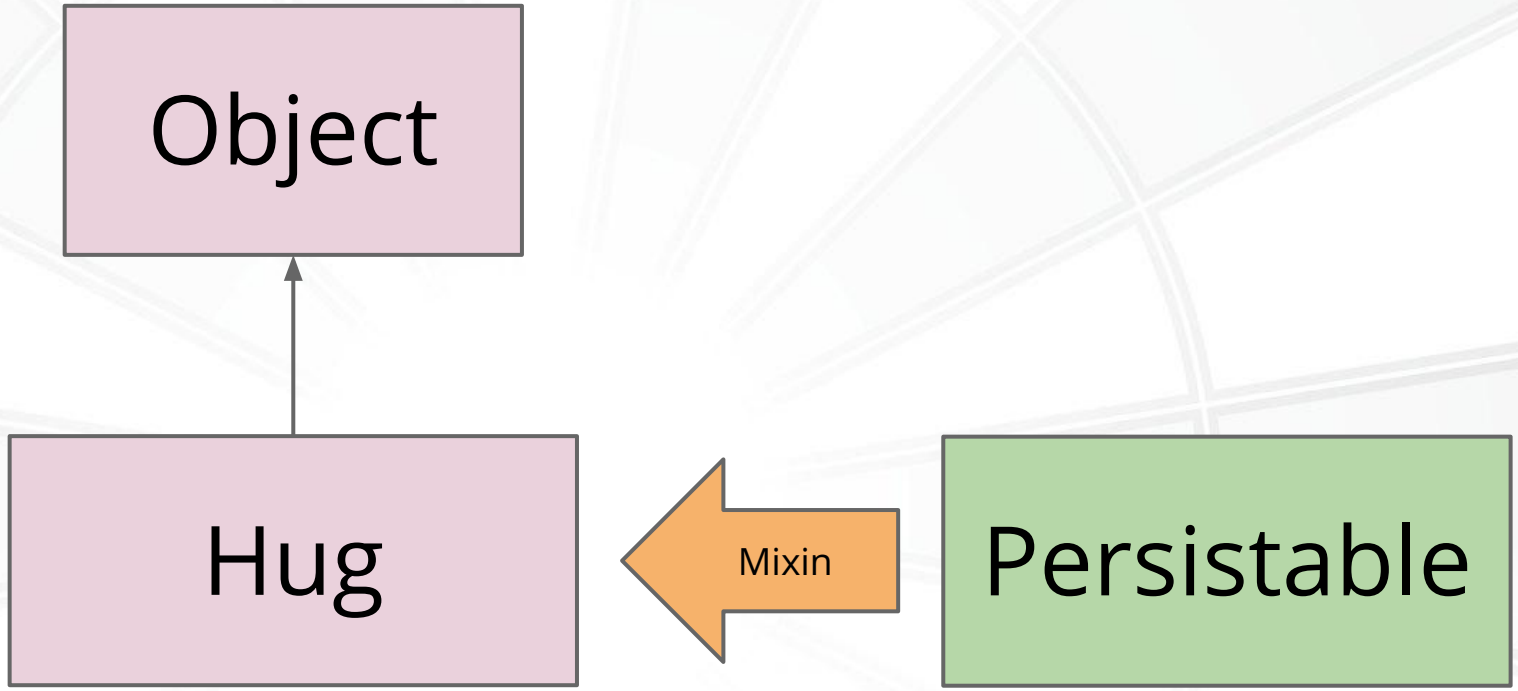
One of these things is not like the other



Hug is *not*
is-a Persistable

Don't pollute your inheritance tree!

Don't inherit, mixin!



Mixins

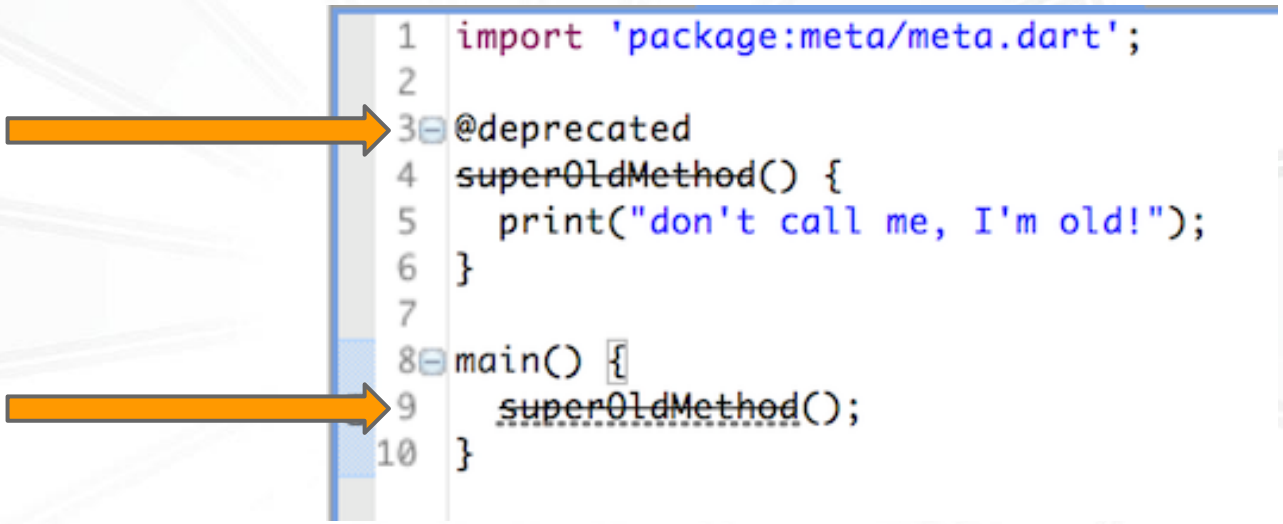
```
abstract class Persistable {
  save() { ... }
  load() { ... }
  toJson();
}

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();
}
```

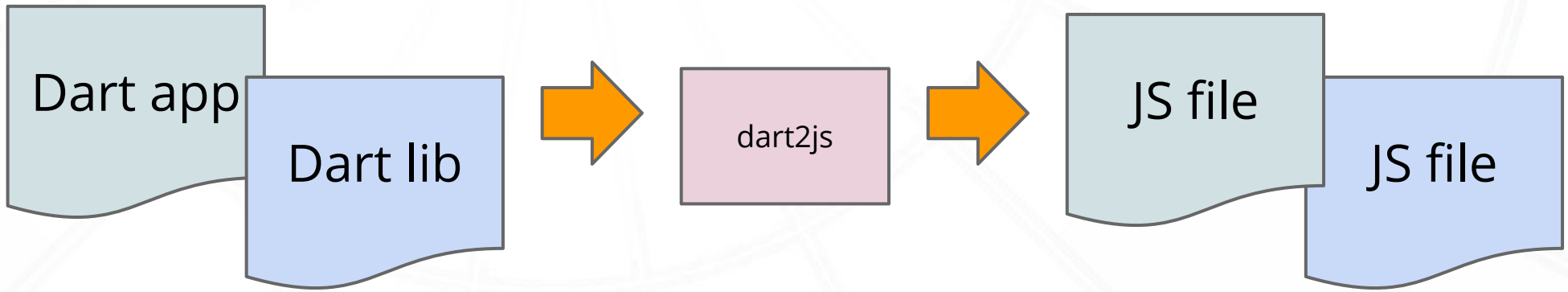
Lazy-load libraries

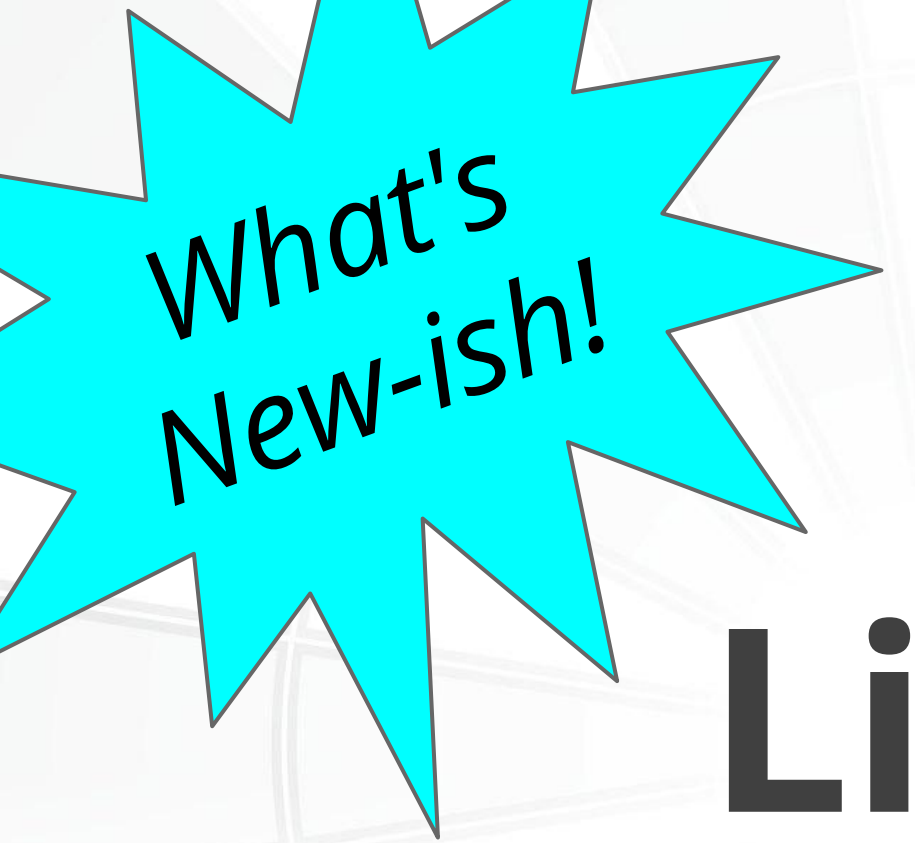
```
const lazy = const DeferredLibrary('my_lib');

@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.



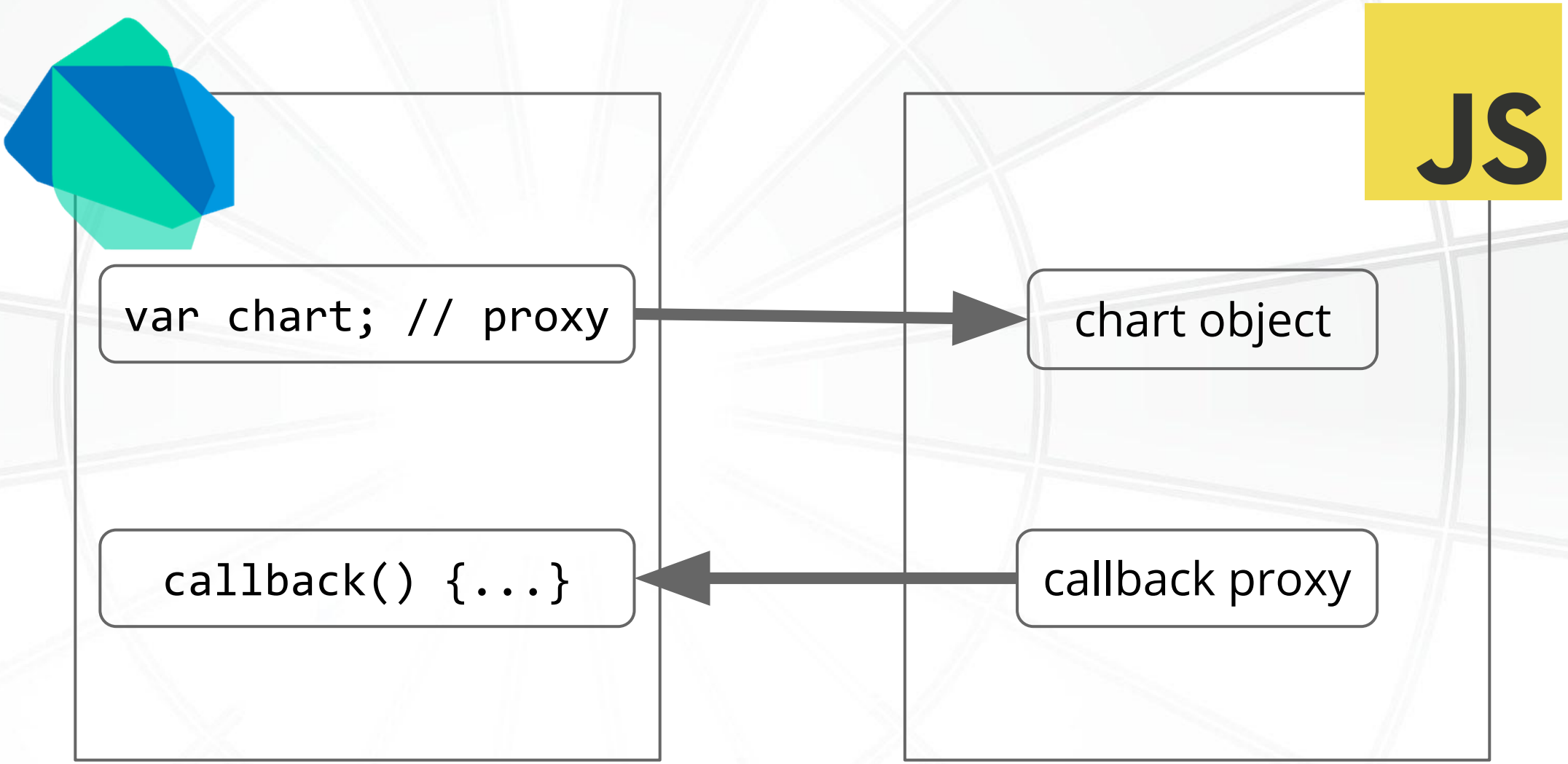


Libraries

JS-Interop

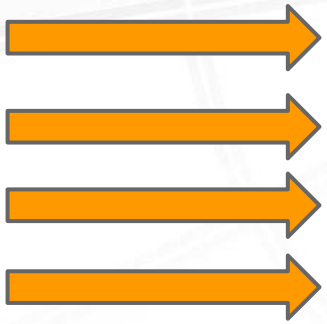

56
votes
4
answers
5k views
Will Dart support the use of existing JavaScript libraries?
I understand Dart compiles to JavaScript, and I read the Dart Language Spec on Libraries, although I didn't see an answer there. Also a search on their discussion form for the word 'existing' t s ...
javascript
libraries
dart
ct 1 11 at 16:44
3 24
YES!

Proxies: the abstraction



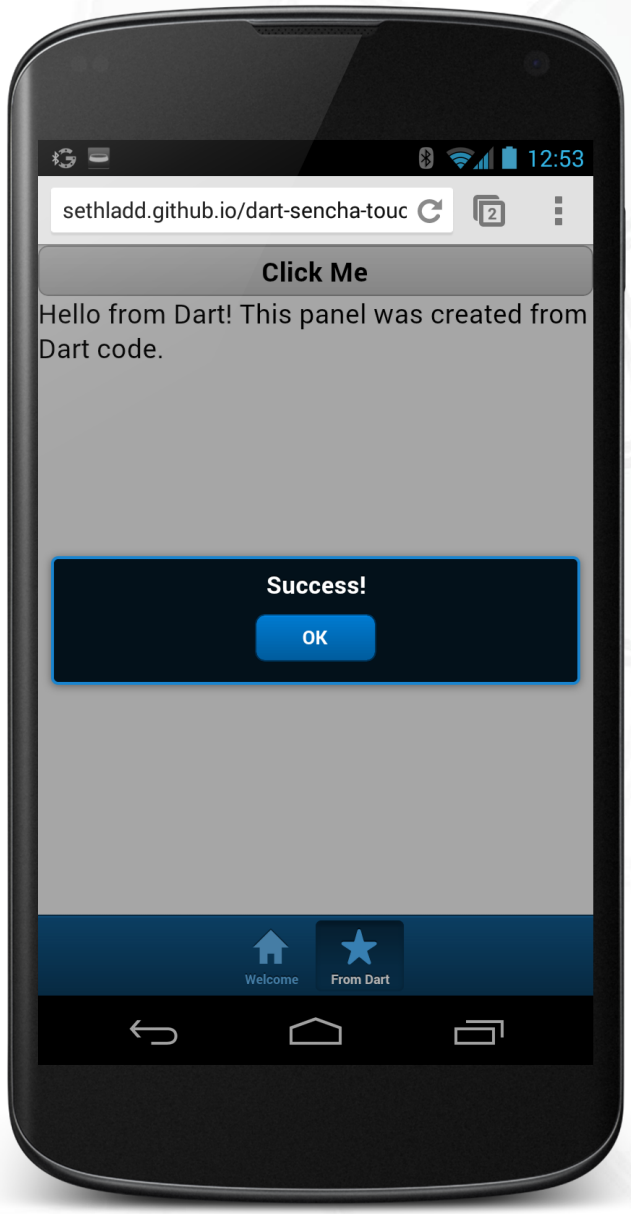
JS-Interop example

```dart
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);
```



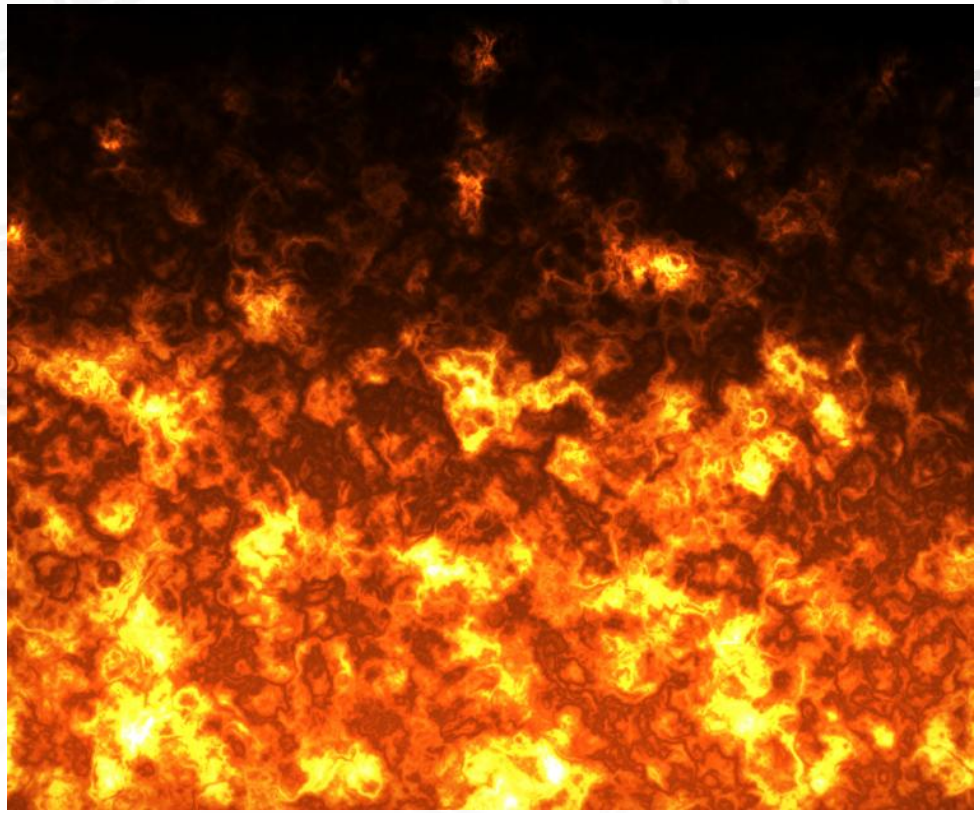
```dart
var api = chartsApi;
var data = [1,3,3,7];
var chart = new api.BubbleChart(querySelector('#chart'));
chart.draw(data);
```

Dart and Sencha Touch Demo



Async with callbacks

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



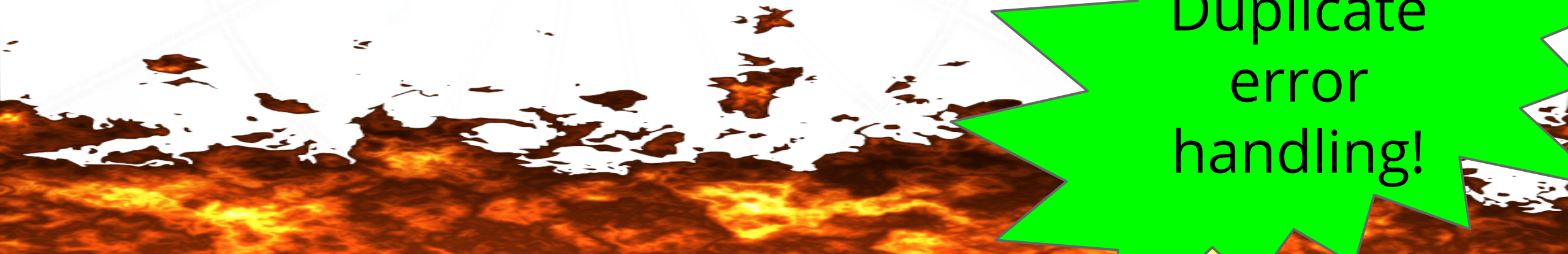
Async with Futures

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); });
```



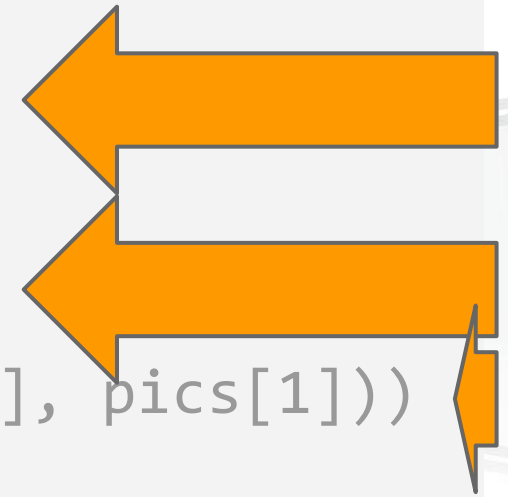
Duplicate error handling!

The Future looks bright

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

Composing futures

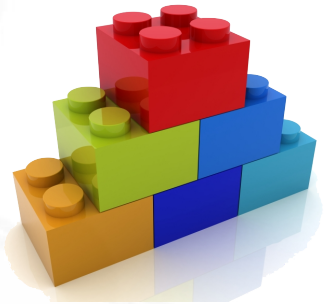
```
Future cute = catService.getPic("cute");
Future nyan = catService.getPic("nyan");

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.



Streams - *series* of async events

- Bytes from a file
- Messages from a web socket
- Clicks from a button
- Events from the DOM
- Incoming connections from a web server

```
stream.listen((event) => doSomething());
```

api.dartlang.org

HTML 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  // returns a Future
   .then((char) => print('First char=$char'));
```

HTML

Web programming with Dart

```
window.navigator.getUserMedia(audio:true, video: true)
  .then((mediaStream) {
    var video = new VideoElement()
      ..autoplay = true
      ..src = Url.createObjectUrl(mediaStream)
      ..onLoadedMetadata.listen((e) => /* ... */);
    document.body.append(video);
  })
  .catchError(reportIssue);
```

XHR with Dart

```
HttpRequest.request('/cats',
    method: 'POST',
    sendData: json.stringify({'name':'Mr. Jingles'}),
    requestHeaders: {'Content-Type': 'application/json'})
  .then((req) {
    catNames.add(json.parse(req.responseText));
  })
  .catchError((e) => print(e));
```

Event source mapping

```
document.body.onClick.matches('.thread').listen((event) {
  print('Super cool!');
});
```

```
<body>
  <p class="thread">Hello world from Dart!</p>
</body>
```

Automatic Sanitization

```
String userInput = "<script>I CAN HAZ XSS</script>" +
                   "<p>I am safe</p>";
query('#contents').innerHtml = userInput;
```


```
// Result:


<div id="contents">
  <p>I am safe</p>                    <!-- No <script> tag -->
</div>
```

Web Components

with Polymer.dart

#dartlang

Old 'n busted

New hotness

```
<messages>
  <message>
    <subject>
      Please fill out the TPS report
    </subject>
    <sent>2012-10-03</sent>
    <summary>
      I'm going to have to ask you to come in...
    </summary>
  </message>
  <message>
    <subject>
      Reminder: fill out that TPS report!
    </subject>
    <sent>2012-10-04</sent>
    <summary>
      It's been 24 hours...
    </summary>
  </message>
  ...
</messages>
```

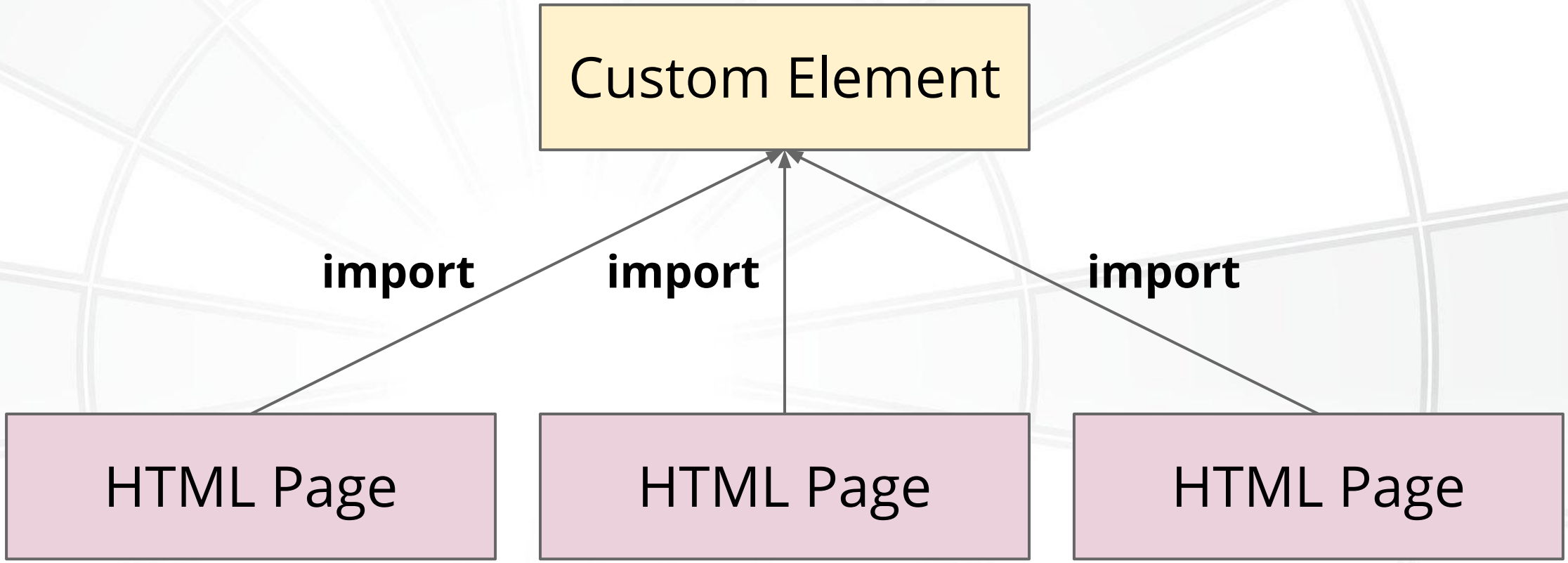
Encapsulation

<custom-element>

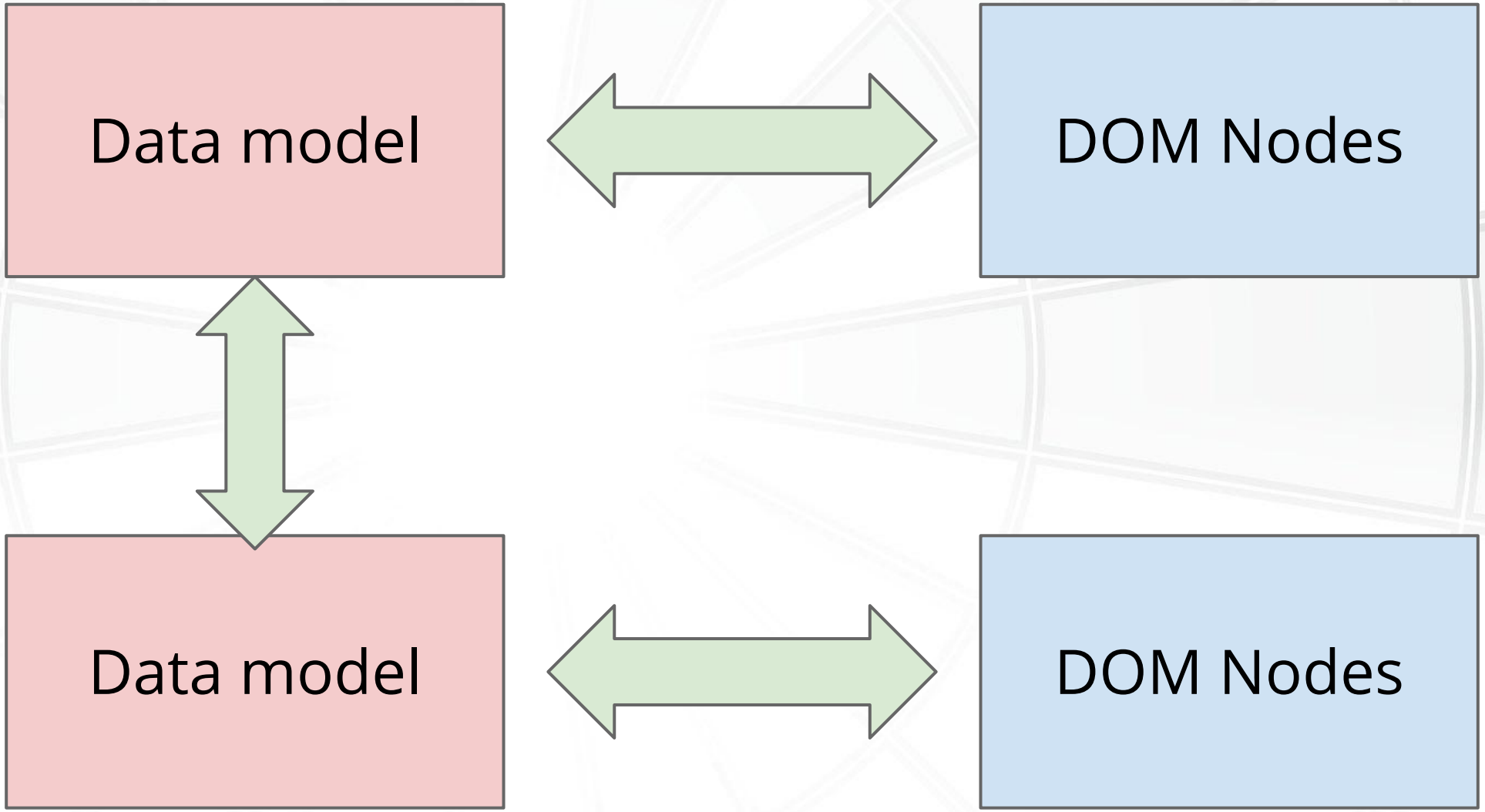
Structure	Behavior	Styles

`<div>`
`  <input>`
`  <p>`
`  <span></span>`
`  </p>`
`</div>`
	`tag.verifyAccount();`	`<style>`
		`  p { color: red; }`
		`</style>`

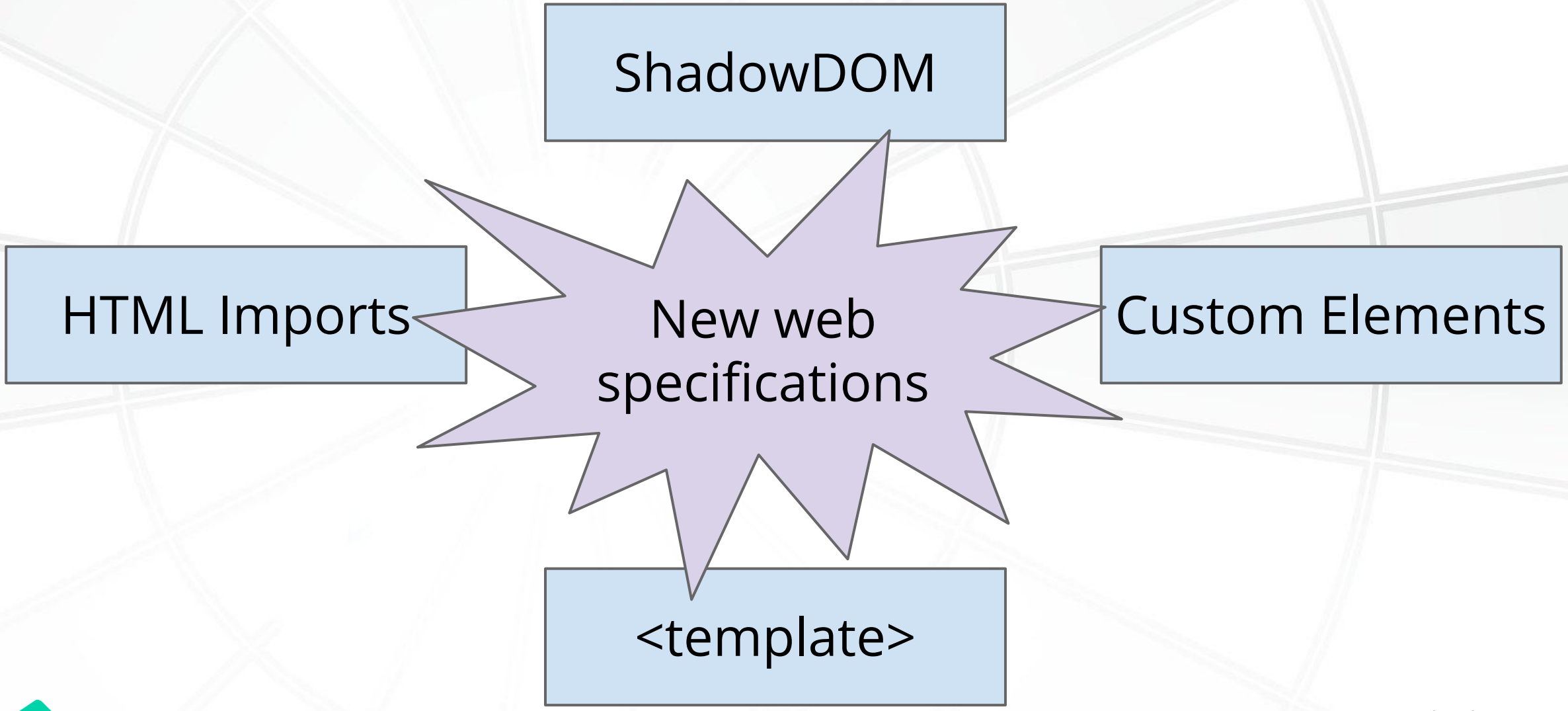
Reusability



Data binding



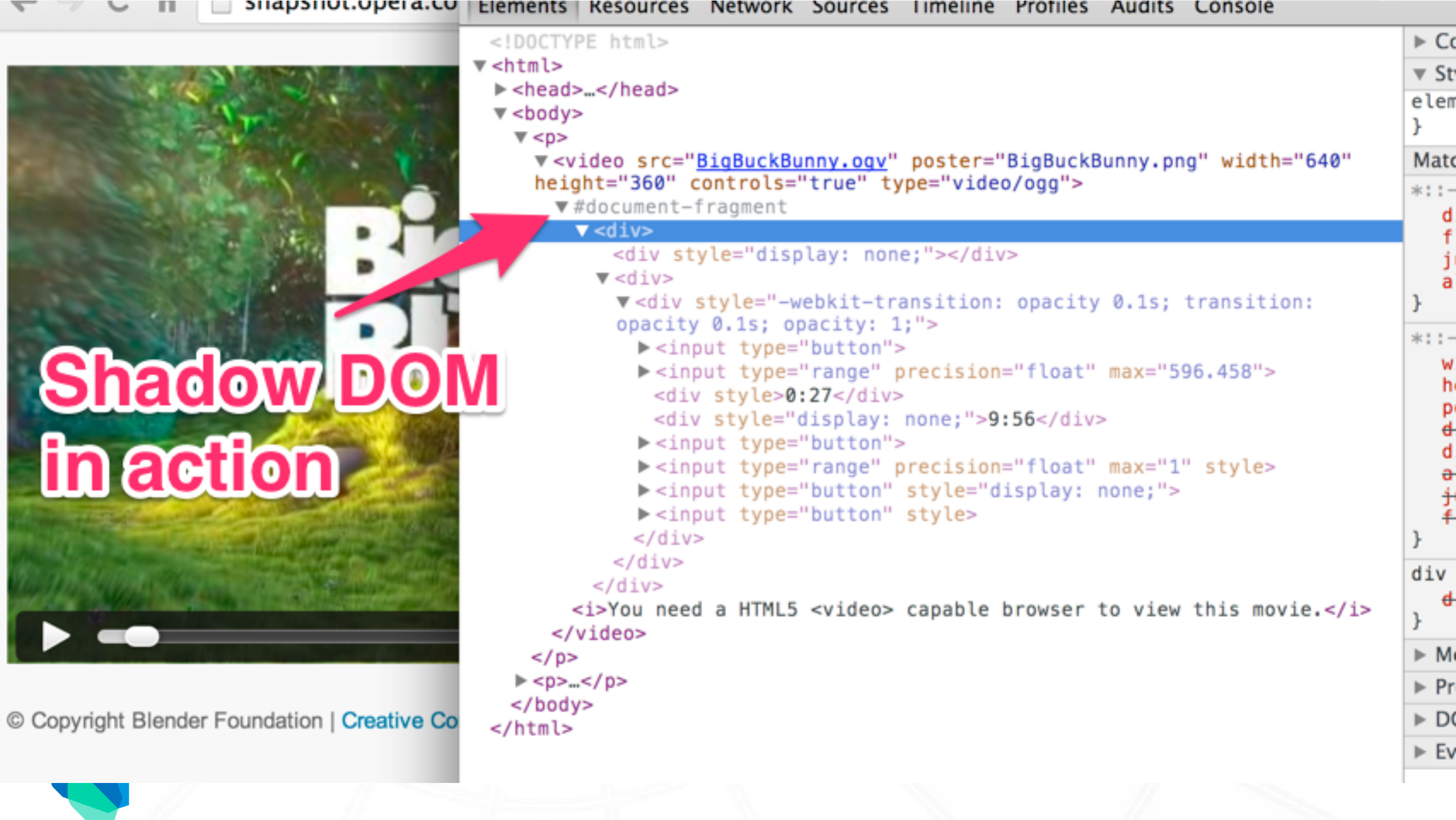
Future-proof



<template> - inert DOM trees

```
<template id="screenshot-tmpl">
  <img src="">
  <div id="comment"></div>
</template>
```

```
TemplateElement t = query('#screenshot-tmpl');
DocumentFragment tree = t.content.clone(true);
(tree.query('img') as ImageElement).src = 'sunrise.png';
tree.query('#comment').text = 'Beautiful';
query('#container').nodes.add(tree);
```

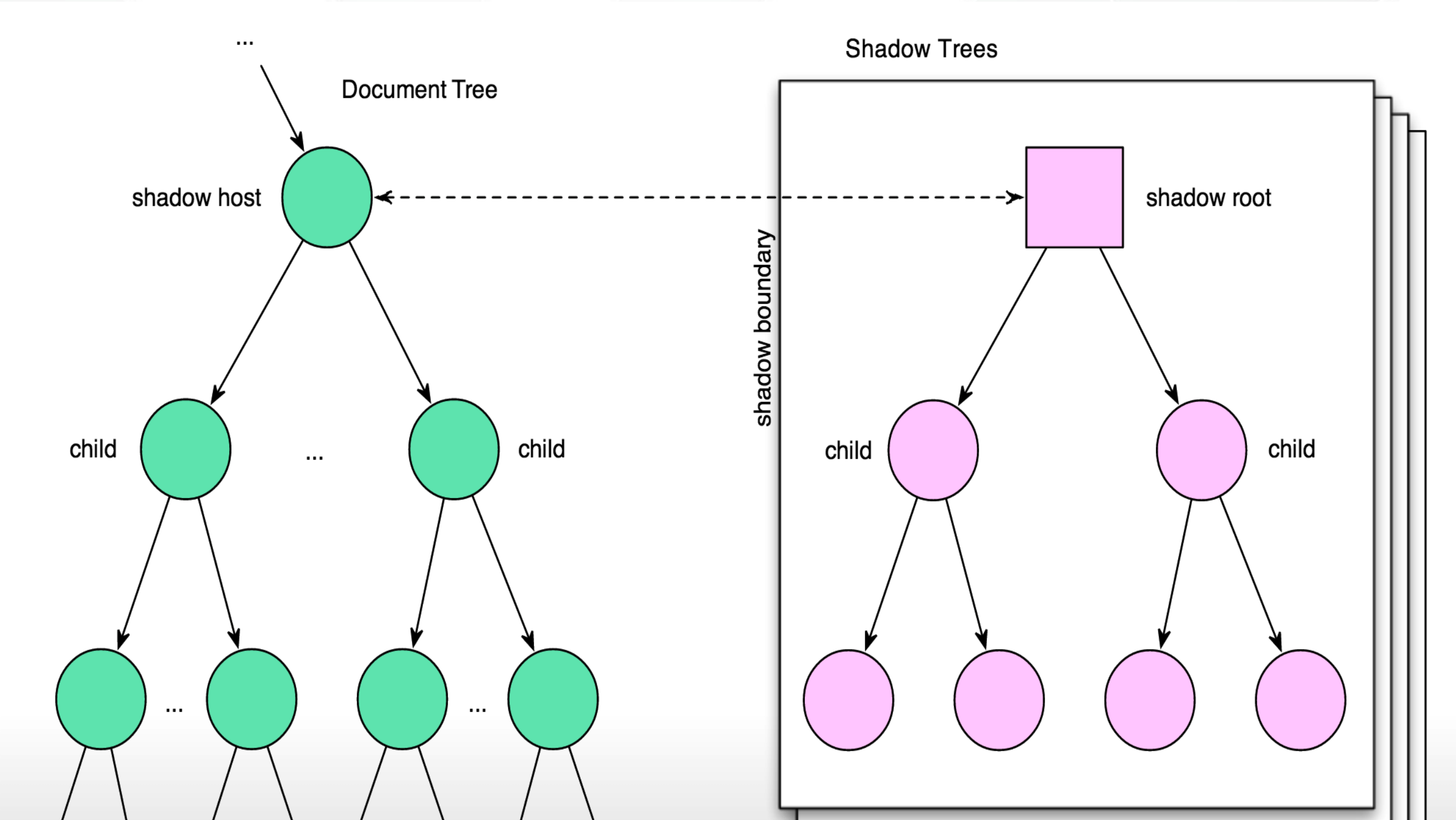

Shadow DOM
in action
Elements Resources Network Sources Timeline Profiles Audits Console
<!DOCTYPE html>
<html>
<head>…</head>
<body>

<video src="BigBuckBunny.ogv" poster="BigBuckBunny.png" width="640" height="360" controls="true" type="video/ogg">
#document-fragment

<input type="button">
<input type="range" precision="float" max="596.458">
<div style>0:27
9:56
<input type="button">
<input type="range" precision="float" max="1" style>
<input type="button" style="display: none;">
<input type="button" style>

You need a HTML5 <video> capable browser to view this movie.
</video>

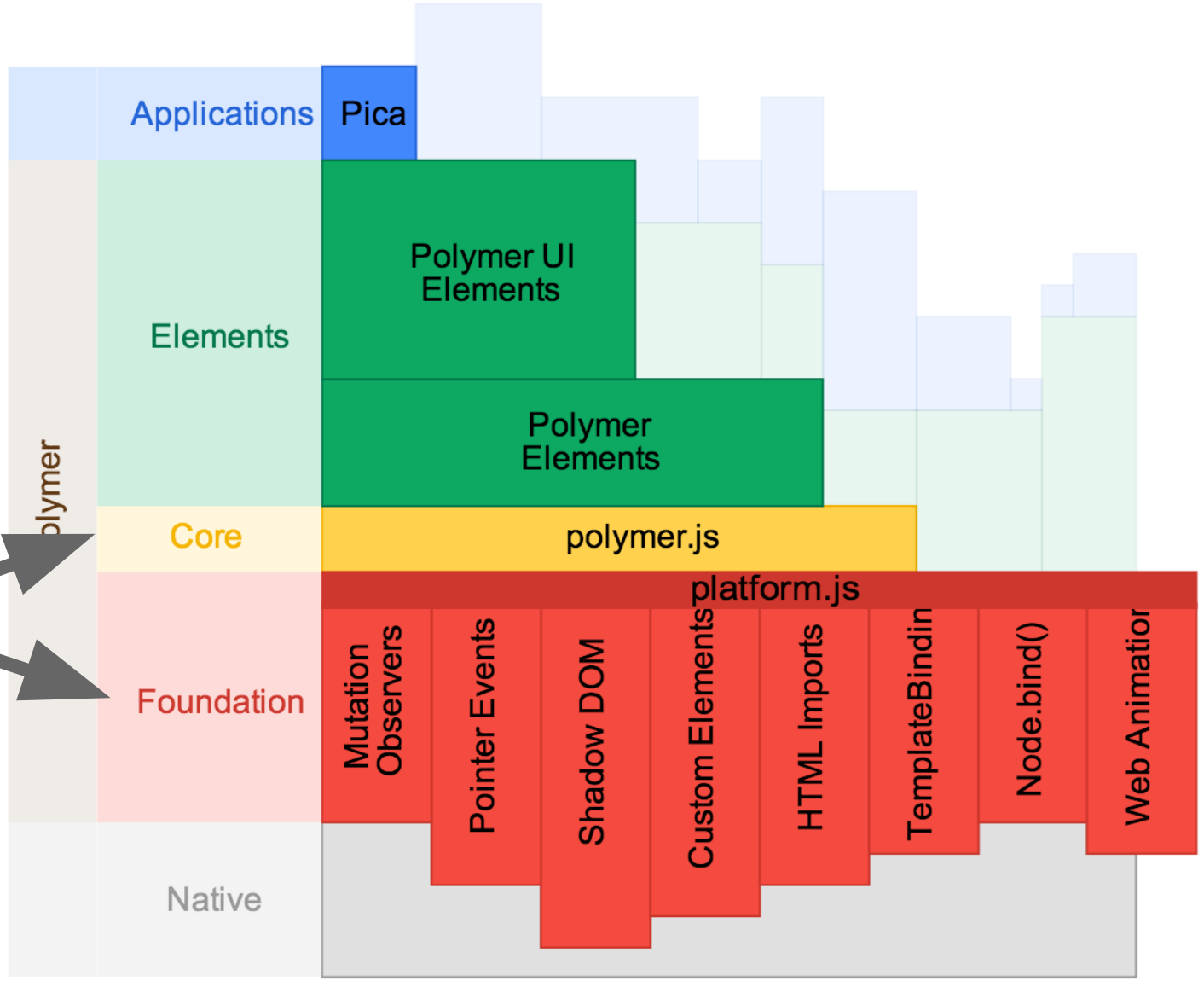
…
</body>
</html>
© Copyright Blender Foundation | Creative Co


...
Document Tree
Shadow Trees
shadow host
shadow root
shadow boundary
child
...
child
child
child
...
...
...

Polymer - use web components *today*

Polymer.dart

(today)



Custom Elements

```
<polymer-element name="click-counter">




</polymer-element>
```

Templates

```
<polymer-element name="click-counter">
  <template>


  </template>

</polymer-element>
```

Declarative event handling

```
<polymer-element name="click-counter">
  <template>
    <button on-click="increment">Click Me</button>

  </template>

</polymer-element>
```

Data binding

```
<polymer-element name="click-counter">
  <template>
    <button on-click="increment">Click Me</button>
    <p>You clicked the button {{count}} times.</p>
  </template>

</polymer-element>
```

Custom Elements

HTML

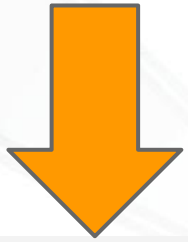
```
<polymer-element name="click-counter">
  <template>
    <button on-click="increment">Click Me</button>
    <p>You clicked the button {{count}} times.</p>
  </template>
  <script src="click_counter.dart" type="application/dart"></script>
</polymer-element>
```

Dart

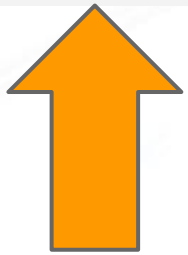
```
@CustomTag('click-counter')
class ClickCounterElement extends PolymerElement with ObservableMixin {
  @observable int count = 0;
  void increment(Event e, var detail, Node target) {
    count += 1;
  }
}
```

Custom Elements

Import the custom element code.



```
<link rel="import" href="clickcounter.html">
...
<click-counter></click-counter>
```



Use the custom element.

Conditionals

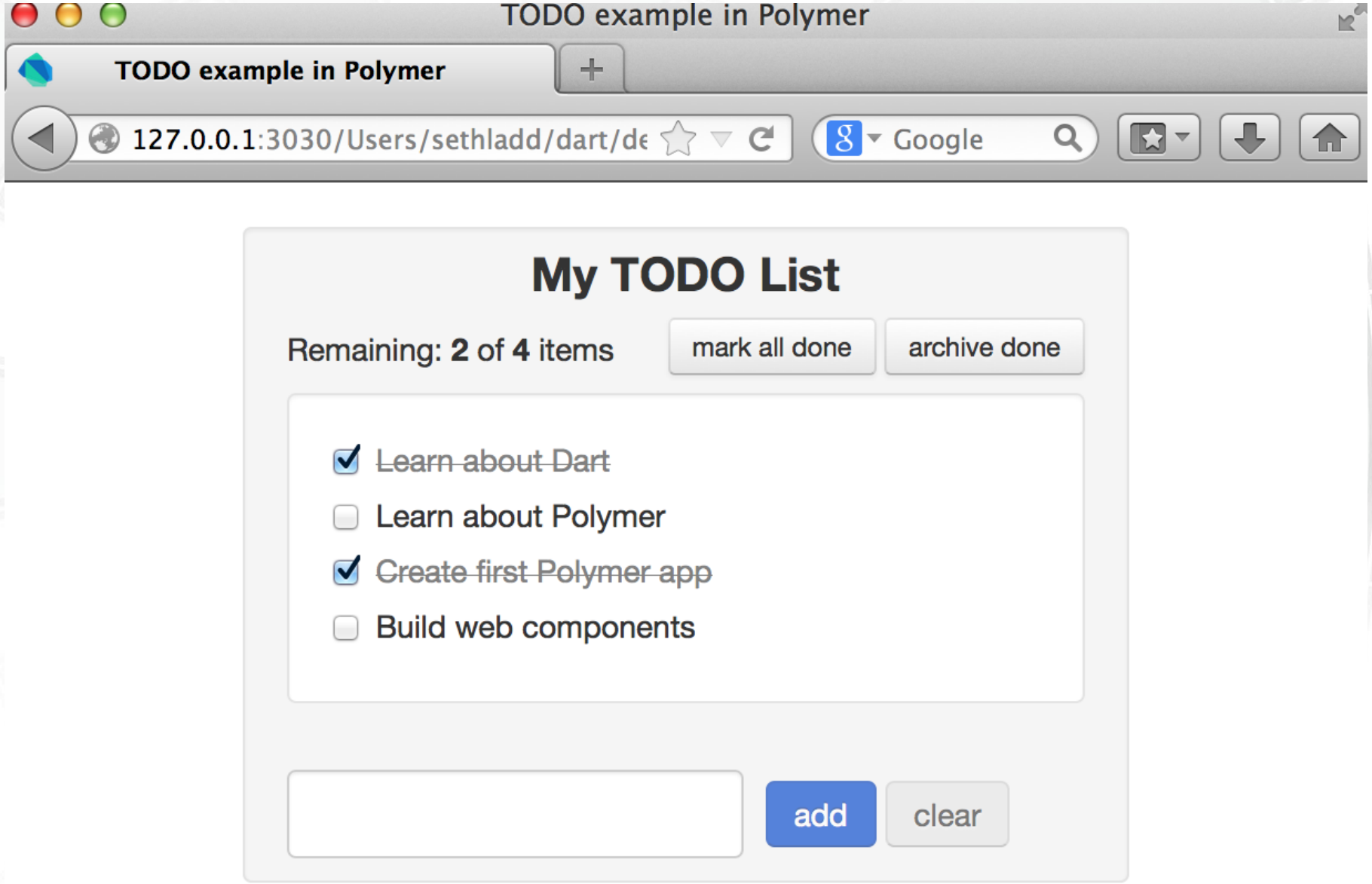
```
<polymer-element name="click-counter">
  <template>
    <button on-click="increment">Click Me</button>
    <template if="{{count <= 0}}">
      <p>Click the button. It's fun!</p>
    </template>
    <template if="{{count > 0}}">
      <p>You clicked the button {{count}} times.</p>
    </template>
  </template>
  <script type="application/dart" src="click_counter.dart"></script>
</polymer-element>
```

Loops

```
<polymer-element name="fav-fruits">
  <template>
    <ul>
      <template repeat="{{fruit in fruits}}">
        <li>
          I like {{ fruit }}.
        </li>
      </template>
    </ul>
  </template>
  <script type="application/dart" src="fav_fruits.dart"></script>
</polymer-element>
```

```
@CustomTag('fav-fruits')
class FavFruitsElement extends PolymerElement with ObservableMixin {
  final List fruits = toObservable(['apples', 'pears', 'bananas']);
}
```

TODO Demo

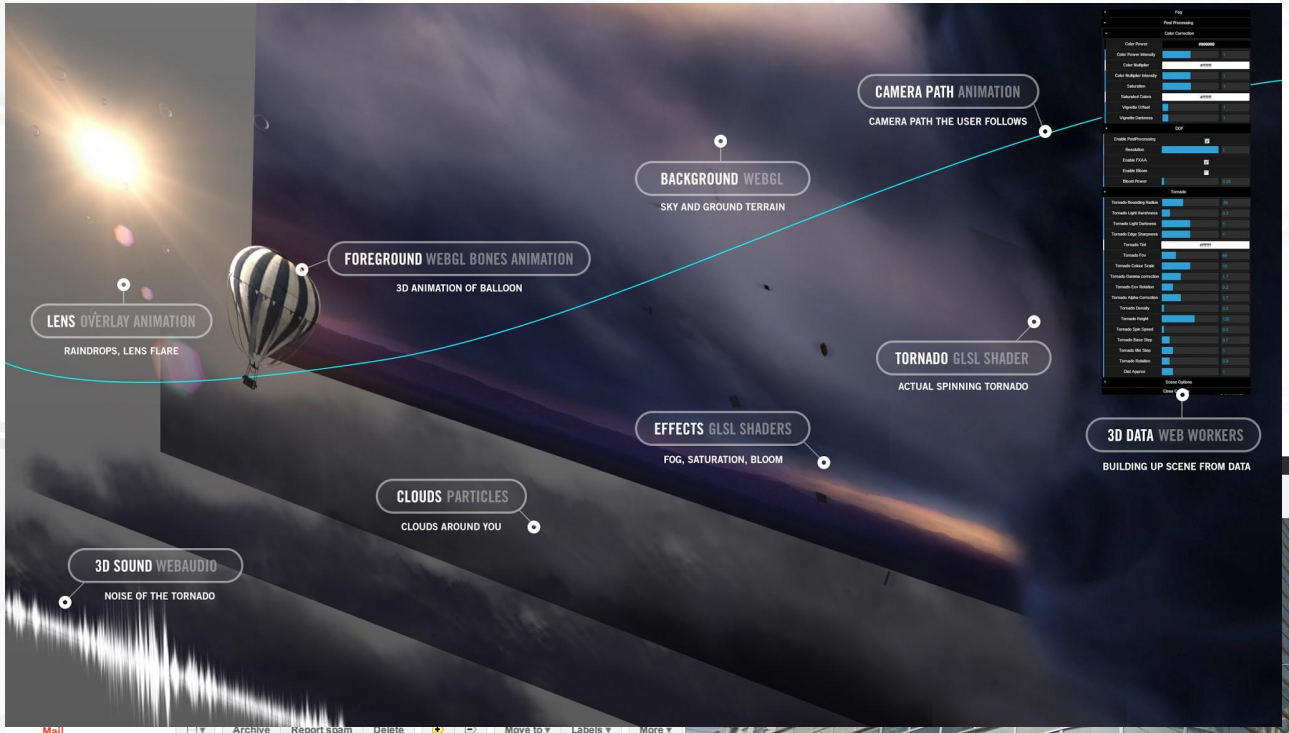

TODO example in Polymer
TODO example in Polymer
127.0.0.1:3030/Users/sethladd/dart/de
Google
My TODO List
Remaining: 2 of 4 items
mark all done
archive done
Learn about Dart
Learn about Polymer
Create first Polymer app
Build web components
add
clear

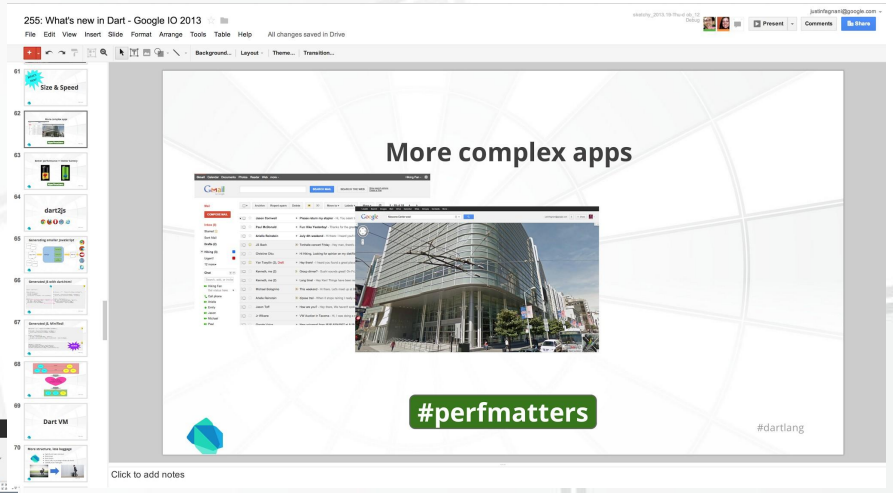
Polymer.dart

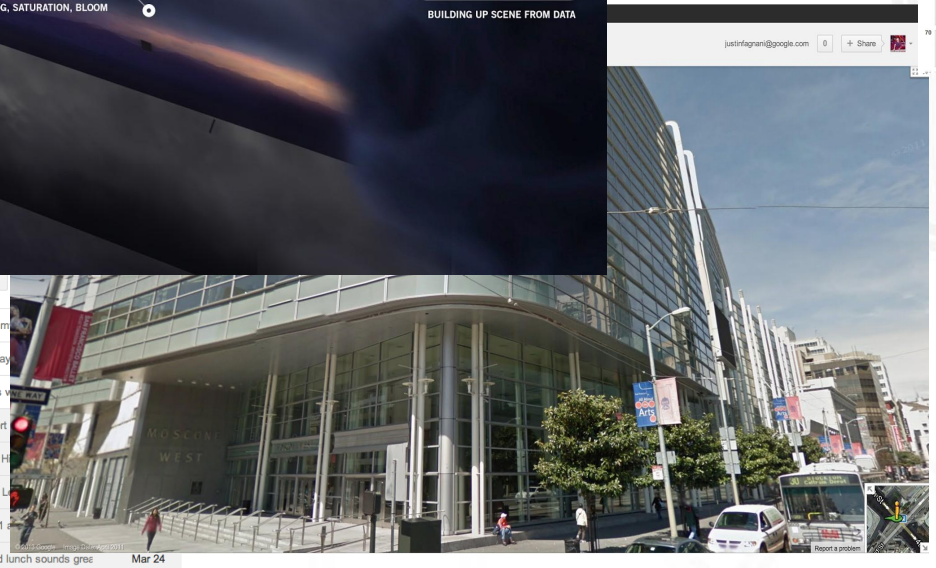
Bringing web components to Dart developers
Encapsulation, reusability
Built on emerging web standards
Learn more at dartlang.org/polymer-dart/

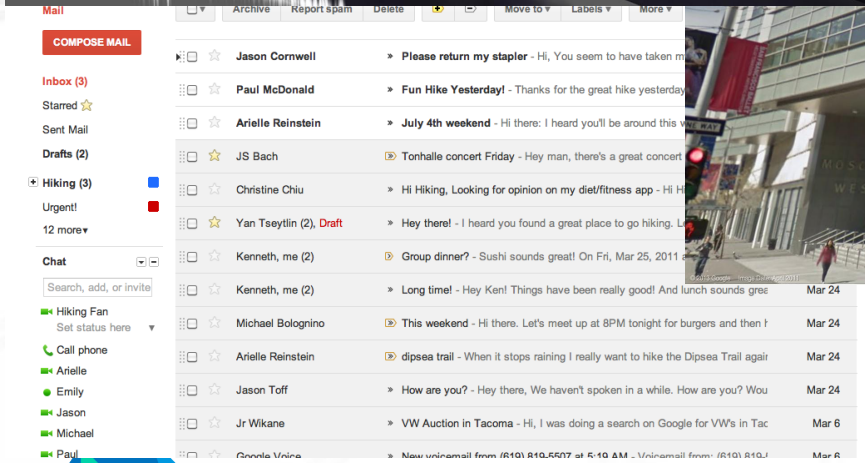
Size & Speed

More complex apps



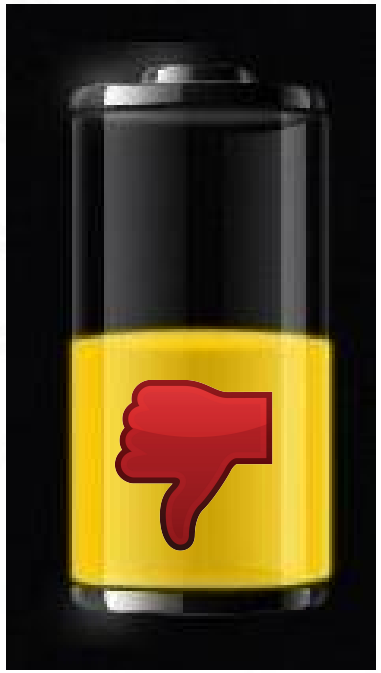






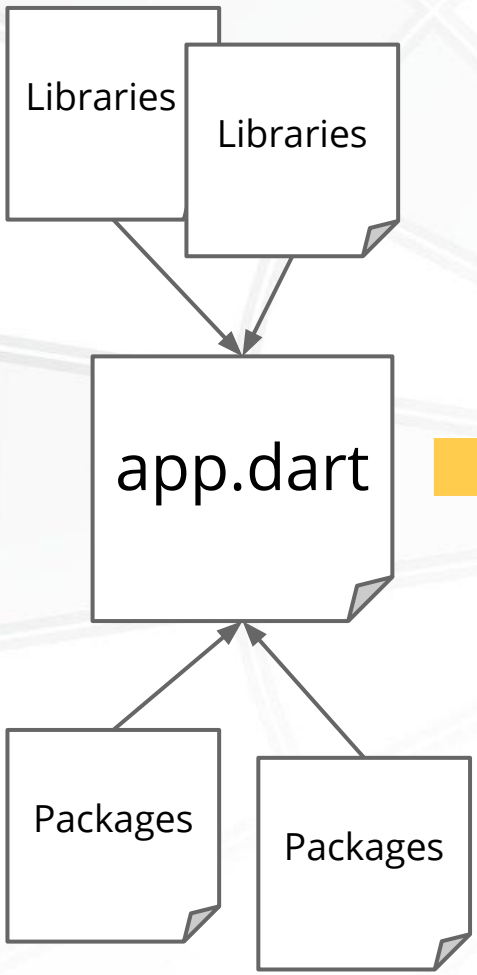
#perfmatters

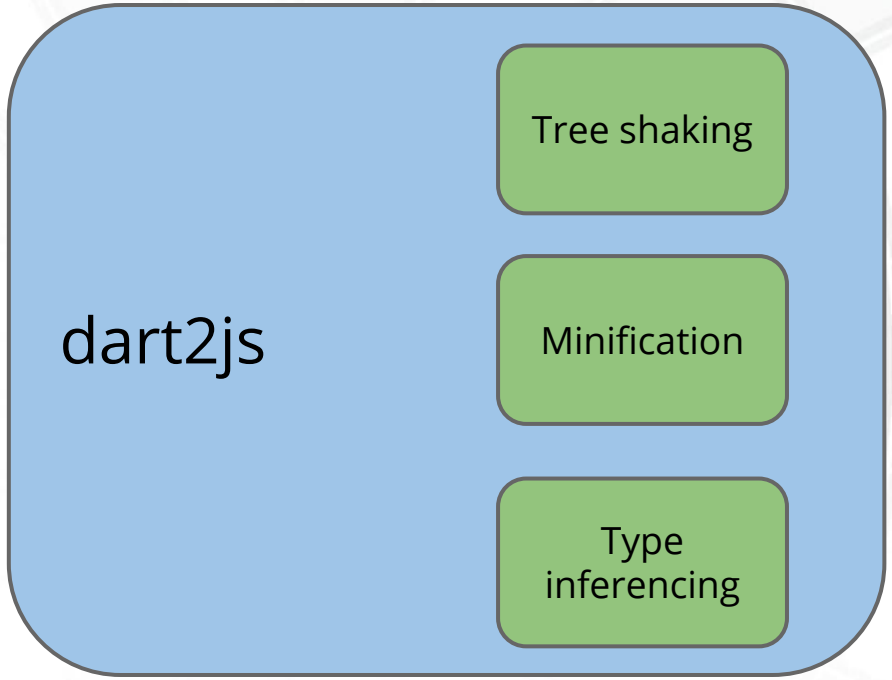
Better performance == Better battery

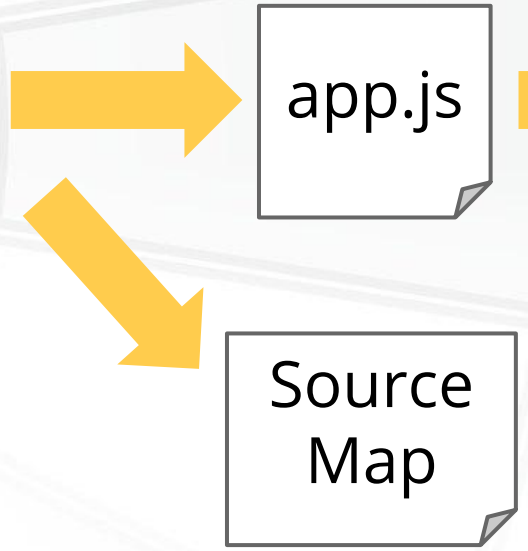


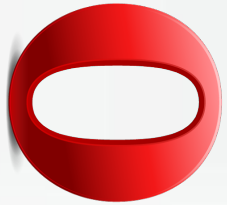
#perfmatters

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;
};
```

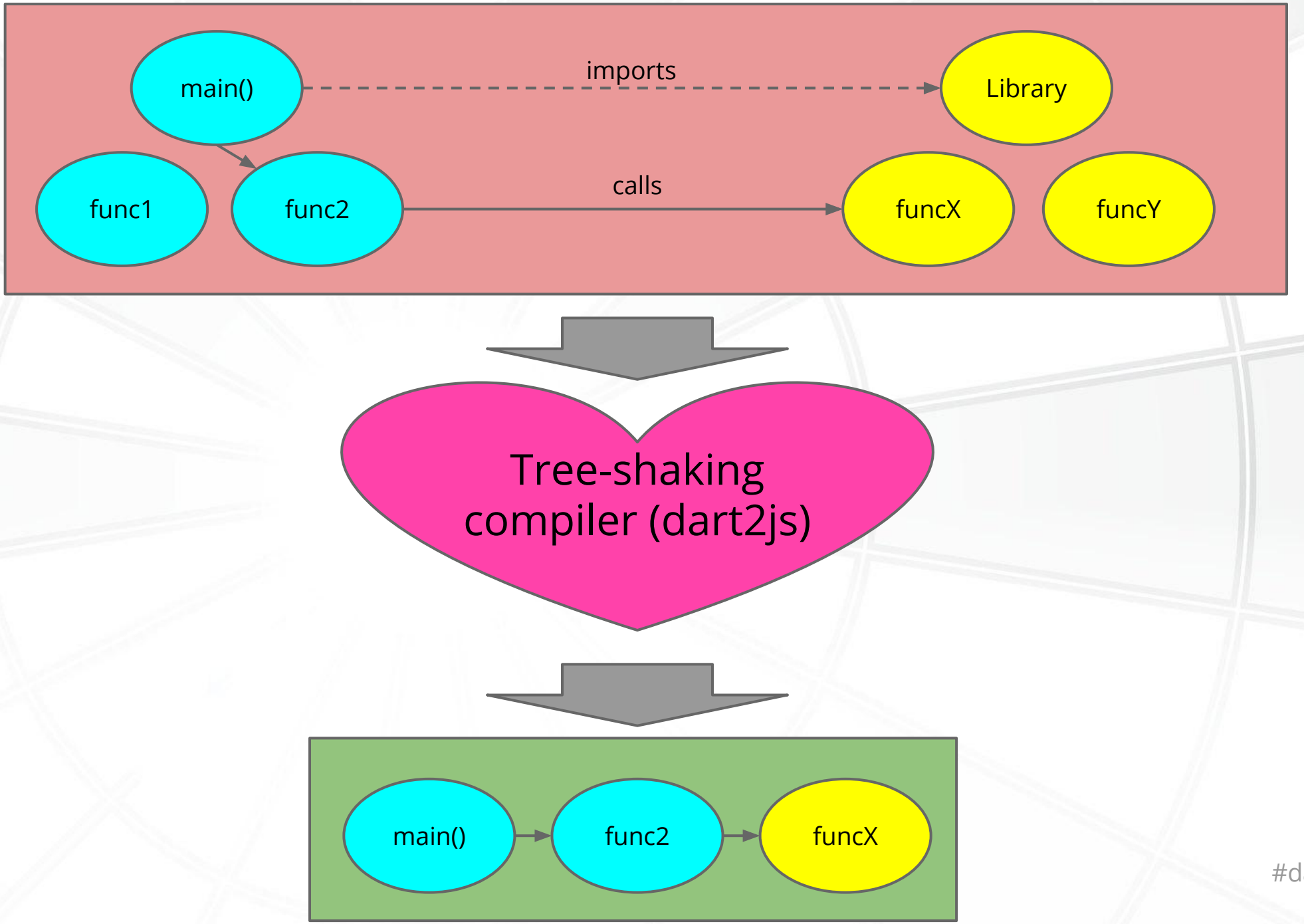
Generated JS, minified!

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

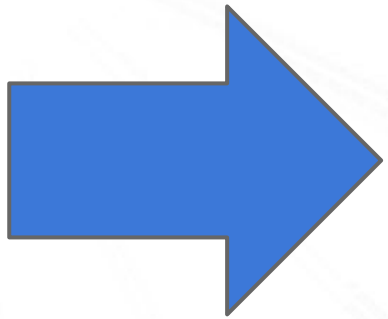
```
$$.mM={"":"a;Sz,dq"}
$.PH=function(a,b){return new $.mM(a,b)}
$.E2=function(){var z=$.PH("Bob","Smith")
document.querySelector("#msg").textContent=z.Sz}
```


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

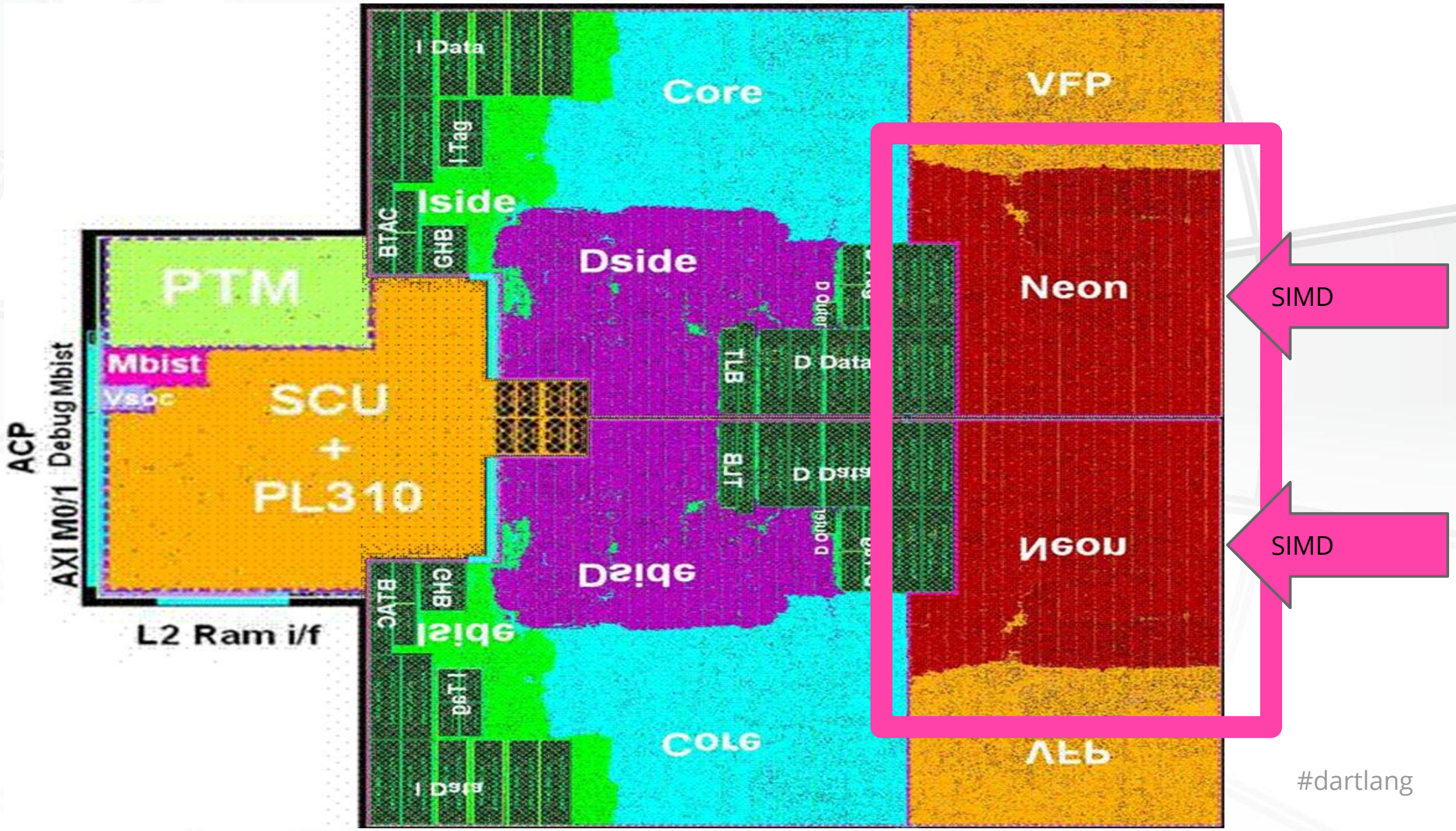
Dart VM

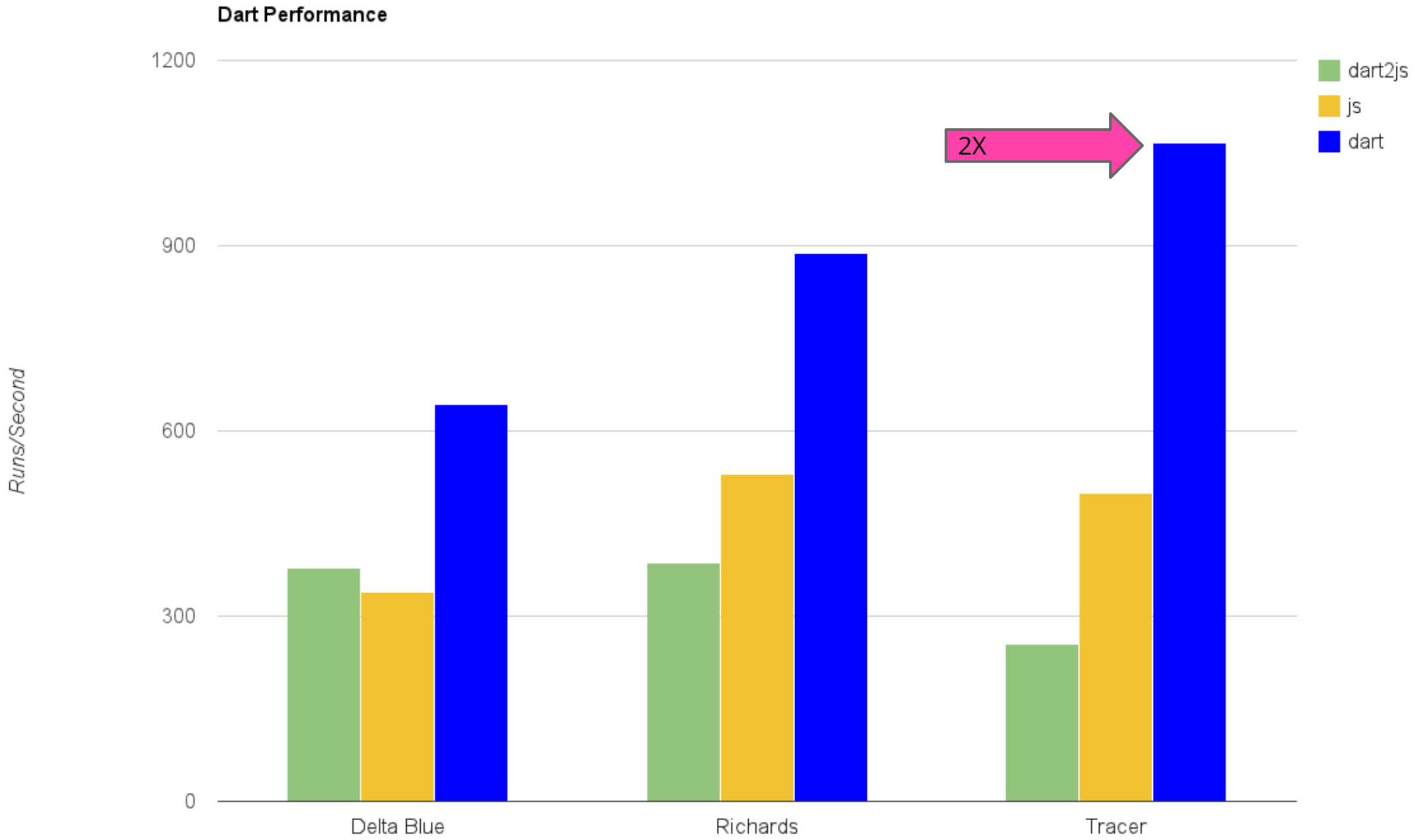
More structure, less baggage

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



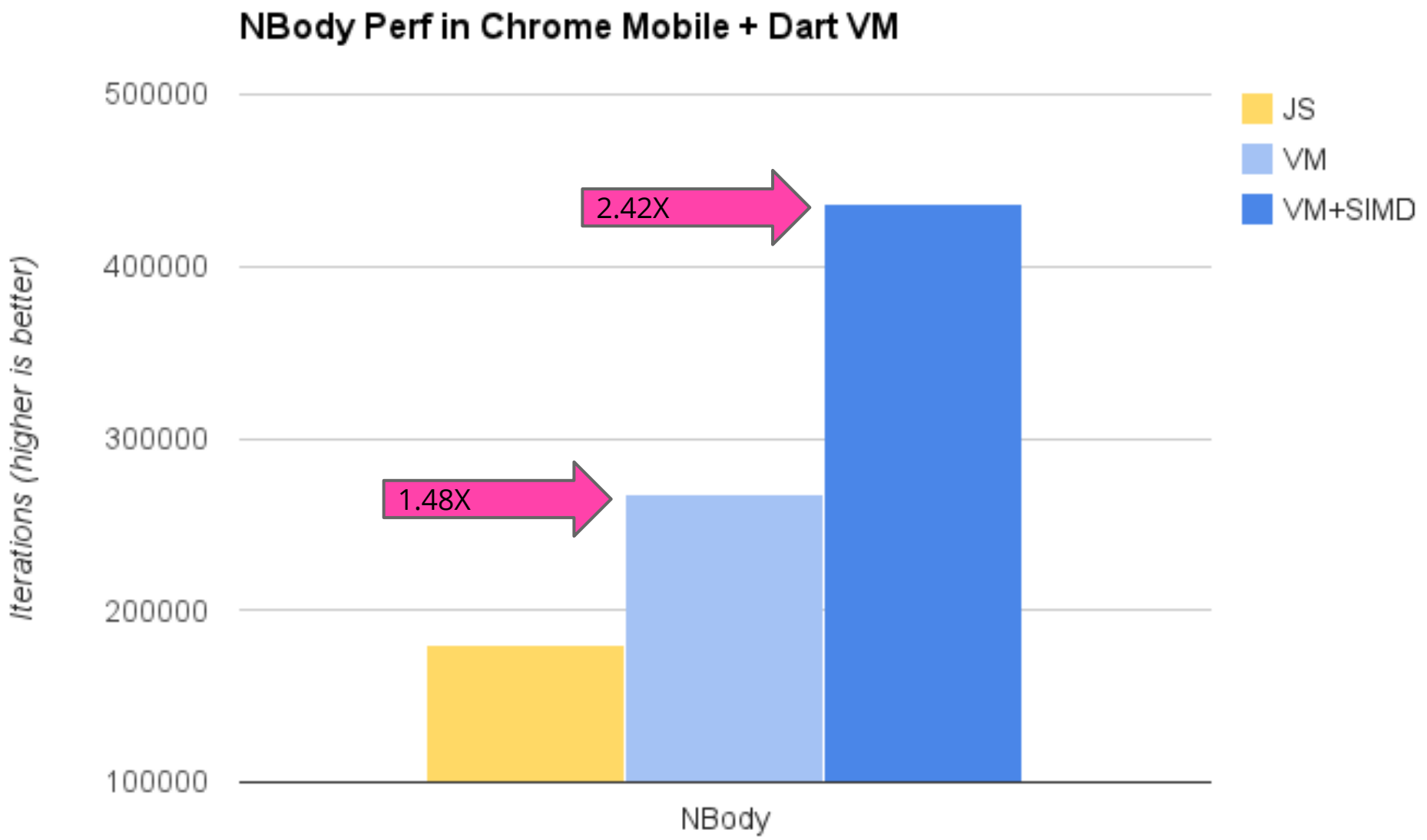
Unlock more of your CPU





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!

- Reflection
- Server-side
- Pub package manager
- Editor support
- 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

DART

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

Thank You!

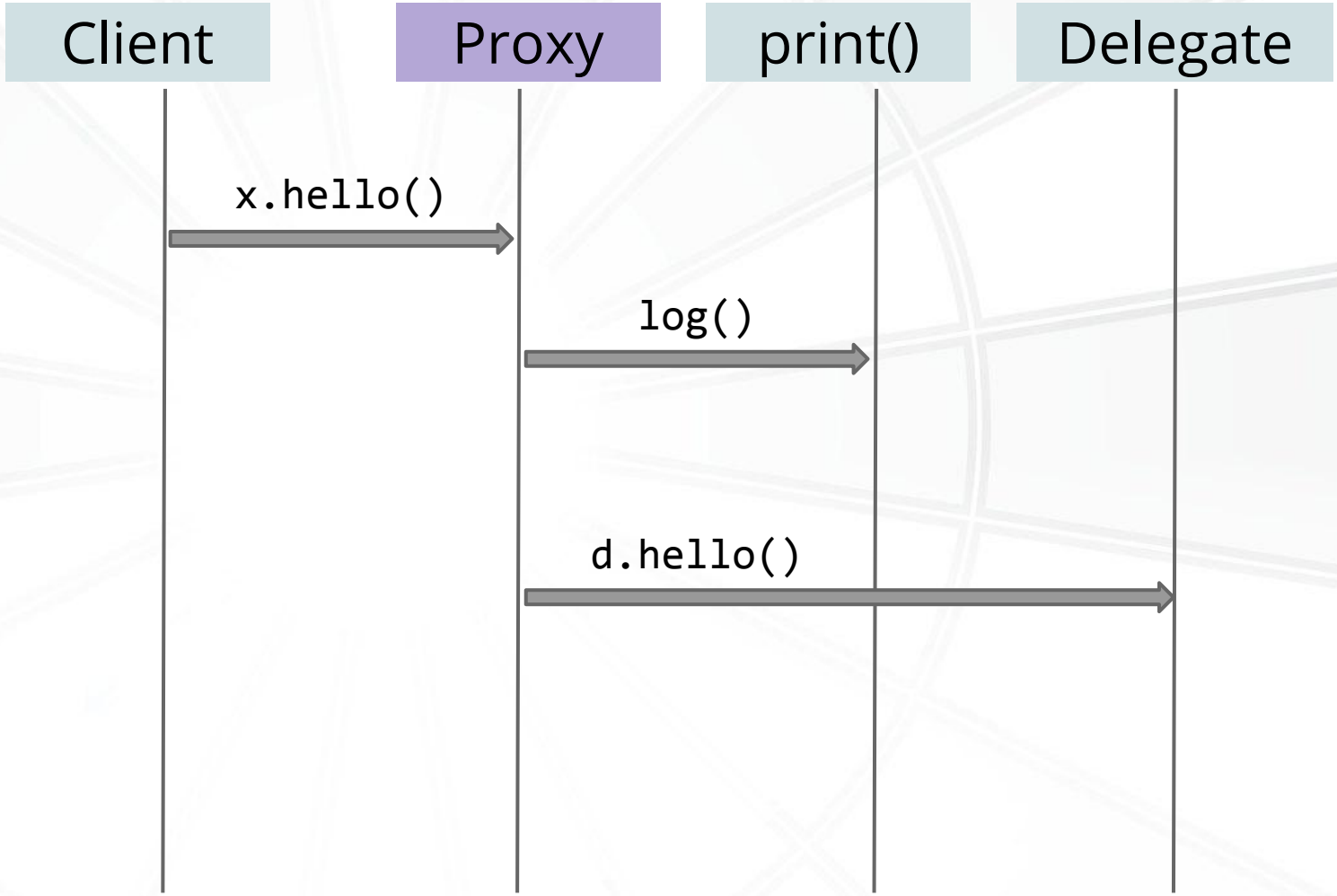
Find us at ***dartlang.org***

#dartlang

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');
    return 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');
    return 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');
    return 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');
    return 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