Dart

Eine kurze Einführung

David Stutz

Lehrstuhl für Datenmanagement und -exploration
RWTH Aachen

Proseminar SS 2012

Überblick

Einführung

1

- Klassen-basierte Programmiersprache für Webanwendungen
- Entwickelt von Google

[1] Bild: `http://www.dartlang.org/`

Einführung

1

- Klassen-basierte Programmiersprache für Webanwendungen
- Entwickelt von Google
- Vorgestellt auf der GOTO Konferenz in Aarhus am 12. Oktober 2011

[1] Bild: `http://www.dartlang.org/`

Einführung

1

- Klassen-basierte Programmiersprache für Webanwendungen
- Entwickelt von Google
- Vorgestellt auf der GOTO Konferenz in Aarhus am 12. Oktober 2011
- Zurzeit: “technical preview”

[1] Bild: `http://www.dartlang.org/`

Einführung

1

- Klassen-basierte Programmiersprache für Webanwendungen
- Entwickelt von Google
- Vorgestellt auf der GOTO Konferenz in Aarhus am 12. Oktober 2011
- Zurzeit: “technical preview”
- Diese Präsentation: Dart für clientseitige Programmierung

[1] Bild: `http://www.dartlang.org/`

Motivation

- "Building delightful applications on the web today is far too difficult."[2] – Mark S. Miller

[2] https://gist.github.com/1208618

Motivation

- "Building delightful applications on the web today is far too difficult."[2] – Mark S. Miller
- Komplexe Anwendungen benötigen modularen und strukturierten Code

[2] https://gist.github.com/1208618

Motivation

- "Building delightful applications on the web today is far too difficult."[2] – Mark S. Miller
- Komplexe Anwendungen benötigen modularen und strukturierten Code
- Kaum Möglichkeiten homogene Systeme zu entwickeln

[2] https://gist.github.com/1208618

Motivation

- "Building delightful applications on the web today is far too difficult."[2] – Mark S. Miller
- Komplexe Anwendungen benötigen modularen und strukturierten Code
- Kaum Möglichkeiten homogene Systeme zu entwickeln
  - Google Web Toolkit
    `https://developers.google.com/web-toolkit/`
  - Node.js
    `http://nodejs.org/`

[2] `https://gist.github.com/1208618`

Motivation

- "Building delightful applications on the web today is far too difficult."[2] – Mark S. Miller
- Komplexe Anwendungen benötigen modularen und strukturierten Code
- Kaum Möglichkeiten homogene Systeme zu entwickeln
  - Google Web Toolkit
    `https://developers.google.com/web-toolkit/`
  - Node.js
    `http://nodejs.org/`
- Clientseitig vorallem Schwierigkeiten mit Javascript

[2] `https://gist.github.com/1208618`

Werkzeuge

Google stellt hilfreiche Werkzeuge zur Entwicklung bereit:

- Dart–zu–Javascript Übersetzer

Werkzeuge

Google stellt hilfreiche Werkzeuge zur Entwicklung bereit:

- Dart–zu–Javascript Übersetzer
- Dart Editor – auf Eclipse basierter Editor

Werkzeuge

Google stellt hilfreiche Werkzeuge zur Entwicklung bereit:

- Dart–zu–Javascript Übersetzer
- Dart Editor – auf Eclipse basierter Editor
- “Dartium” – auf Chromium basierter Browser mit integrierter Dart VM

Werkzeuge

Google stellt hilfreiche Werkzeuge zur Entwicklung bereit:

- Dart–zu–Javascript Übersetzer
- Dart Editor – auf Eclipse basierter Editor
- "Dartium" – auf Chromium basierter Browser mit integrierter Dart VM

Downloads unter `http://dartlang.org`.

Demonstration

Klassen

Dart unterstützt:

- Klassen-basierte Objektorientierung

Klassen

Dart unterstützt:

- Klassen-basierte Objektorientierung
- Einfache Vererbung

Klassen

Dart unterstützt:

- Klassen-basierte Objektorientierung
- Einfache Vererbung
- Abstrakte Klassen

Klassen

Dart unterstützt:

- Klassen-basierte Objektorientierung
- Einfache Vererbung
- Abstrakte Klassen
- Interfaces

Klassen

Dart unterstützt:

- Klassen-basierte Objektorientierung
- Einfache Vererbung
- Abstrakte Klassen
- Interfaces

Außerdem:

- Verzicht auf public / private
- Getter/Setter mittels get/set deklarieren
- Benannte Konstruktoren

Optionale Typen

Eingebaute Typen:

- Zahlen – num
- Zeichenketten – String
- “Collections” – Listen, Assoziative Felder...

Optionale Typen

Eingebaute Typen:

- Zahlen – num
- Zeichenketten – String
- “Collections” – Listen, Assoziative Felder...

Aber:

- Typen nicht verpflichtend
- Variablen alternativ mit var deklarierbar

DOM – Document Object Model

DOM Manipulation – Zugriff auf HTML:

- Neue DOM Elemente erstellen:
  - Element.tag('tag')
  - Element.html('<html>...</html>')

DOM – Document Object Model

DOM Manipulation – Zugriff auf HTML:

- Neue DOM Elemente erstellen:
  - Element.tag('tag')
  - Element.html('<html>...</html>')
- Auf Attribute direkt mittels element. attributes zugreifen

DOM – Document Object Model

DOM Manipulation – Zugriff auf HTML:

- Neue DOM Elemente erstellen:
  - Element.tag('tag')
  - Element.html('<html>...</html>')
- Auf Attribute direkt mittels element. attributes zugreifen
- Knoten von DOM Elementen mittels element.nodes ansprechen

Modularität

Anwendungen lassen sich modular aufbauen:

- Eigene Bibliotheken erstellen:

```
1 // In View.dart eine Bibliothek definieren :
2 #library('view');
```

Modularität

Anwendungen lassen sich modular aufbauen:

- Eigene Bibliotheken erstellen:

```
1 // In View.dart eine Bibliothek definieren:
2 #library('view');
```

- Bibliotheken einbinden:

```
1 // Dart's HTML Bibliothek einbinden:
2 #import('dart:html');
3 // Eigene Bibliothek einbinden:
4 #import('View.dart');
```

Evaluation

Dart	Javascript
Klassen-basiert, Interfaces, Vererbung	Prototypen-basiert, komplizierte Vererbung
Optionale Typen	erschwerte Typüberprüfung
	▶ “undefined” ▶ “falsify”
Bibliotheken	Keine Bibliotheken – Globaler Namespace
DOM Manipulation mittels CSS Selektoren	externe Bibliotheken (z.B. JQuery[3])

[3] `http://jquery.com/`

Evaluation (II)

Probleme für den clientseitigen Einsatz:

- Übersetzung zu Javascript vor Ausführung

[4] http://en.wikipedia.org/wiki/Dart_(programming_language)

Evaluation (II)

Probleme für den clientseitigen Einsatz:

- Übersetzung zu Javascript vor Ausführung
- Nativer Browser-Support – "as of March 2012, Microsoft Internet Explorer, Mozilla Firefox, Opera Software's Opera browser, and Apple Safari do not have plans to implement support for Dart"[4]

[4] http://en.wikipedia.org/wiki/Dart_(programming_language)

Quellen

- Dartlang.org
  http://dartlang.org
- Dart (programming language)
  http://en.wikipedia.org/wiki/Dart_(programming_language)
- Dart: 10 Punkte, in denen es JavaScript übertrifft
  http://t3n.de/news/dart-10-punkte-denen-javascript-358345/

Ende

Vielen Dank für die Aufmerksamkeit!

Fragen?

Typ-Überprüfung

```dart
var nullVal = null;
if (!nullVal) {
  // 'null' wird als 'false'
       behandelt ...
}
```

```dart
var nullVal = null;
if (nullVal == null) {
  // Dart kennt lediglich das
       'wahre' false!
}
```

Typ-Überprüfung

```
1 var nullVal = null;
2 if (!nullVal) {
3   // 'null' wird als 'false'
        behandelt ...
4 }
```

```
1 var nullVal = null;
2 if (nullVal == null) {
3   // Dart kennt lediglich das
        'wahre' false!
4 }
```

```
1 var emptyString = '';
2 if (!emptyString) {
3   // leere Zeichenketten werden
        als 'false' behandelt ...
4 }
```

```
1 var emptyString = '';
2 if (emptyString.isEmpty()) {
3   // ...
4 }
```

Typ-Überprüfung

```
var nullVal = null;
if (!nullVal) {
  // 'null' wird als 'false'
       behandelt ...
}
```

```
var nullVal = null;
if (nullVal == null) {
  // Dart kennt lediglich das
       'wahre' false!
}
```

```
var emptyString = '';
if (!emptyString) {
  // leere Zeichenketten werden
       als 'false' behandelt ...
}
```

```
var emptyString = '';
if (emptyString.isEmpty()) {
  // ...
}
```

```
var undefinedVal;
if (!undefinedVal) {
  // 'undefined' wird auch als
       'false' behandelt ...
}
```

```
// Dart kennt kein 'undefined'!
```

Optionale Parameter

```
1 function foo(x, y, z) {
2   return z;
3 }
4
5 foo(1); // Wird 'undefined'
      zurueckgeben...
```

```
1 // Rueckgabewert in der
      Deklaration anzugeben ist
      optional:
2 foo(x, y, z) {
3   return z;
4 }
5
6 foo(1); // NoSuchMethodException!
7
8 // y und z sind optionale
      Parameter:
9 bar(x, [y, z]) {
10  return z;
11 }
12
13 bar(1); // Wird 'null'
      zurueckgeben!
```

Klassen

```
1  // Ein Objekt erstellen :
2  function Car(brand) {
3    this.brand = brand;
4  }
5
6  // Dem Prototypen eine Methode
       hinzufuegen:
7  Car.prototype.alertBrand =
       function() {
8    alert(this.brand);
9  }
```

```
1  class Car {
2
3    // Durch '_'-Praefix automatisch
         privat :
4    var _brand;
5
6    // Konsturktor wird _brand
         automatisch zuweisen:
7    Car(this._brand);
8
9    alertBrand() {
10     window.alert(this._brand);
11   }
12 }
```