

Web und Mobile Apps Programmieren mit Dart

Marco Jakob

Workshop INFOS 2013 in Kiel
28.09.2013.

ZIEL

attraktiv

aktuell

Programmier-
unterricht

wenig
Hürden

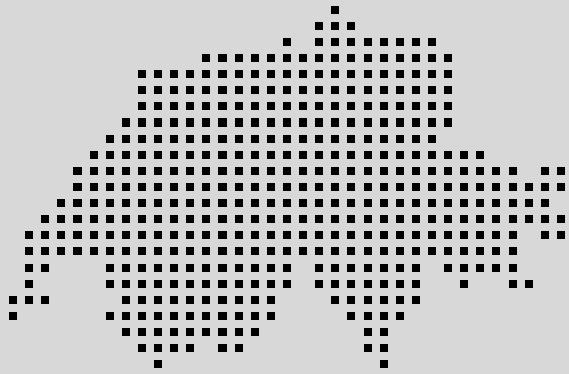
Inhalt

- Weshalb **Web und Mobile** im Unterricht?
- aktuelle **Möglichkeiten** für Web und Mobile
- die Sprache **Dart**
- Einsatzmöglichkeiten für den **Unterricht**

- **Praktischer** Teil

- Diskussion

ZU MIR...



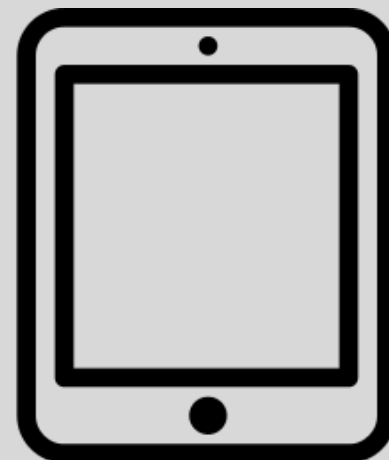
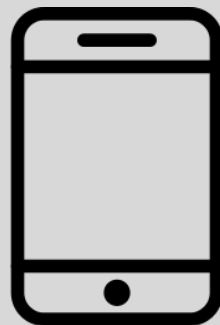
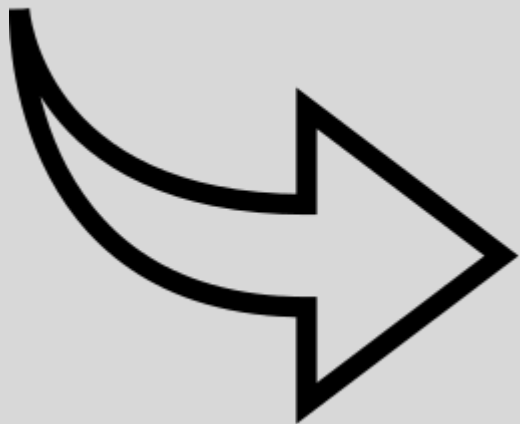
hat sich bewährt

ABER...

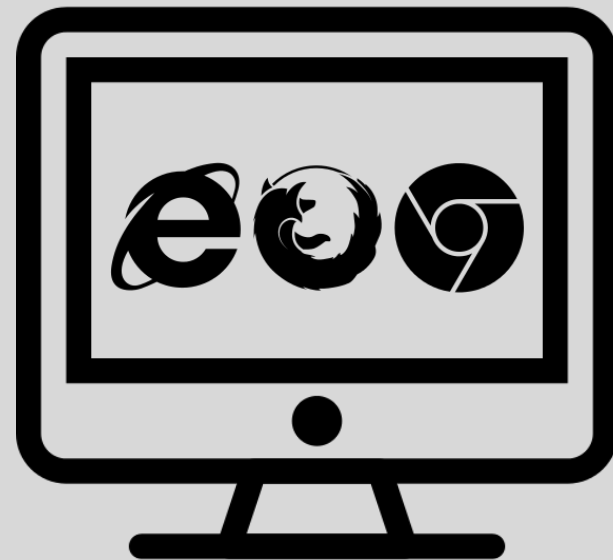
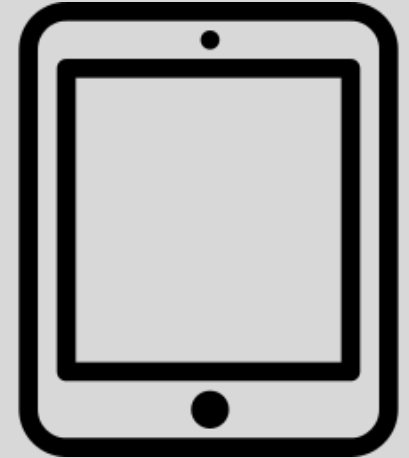
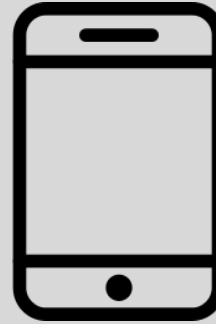
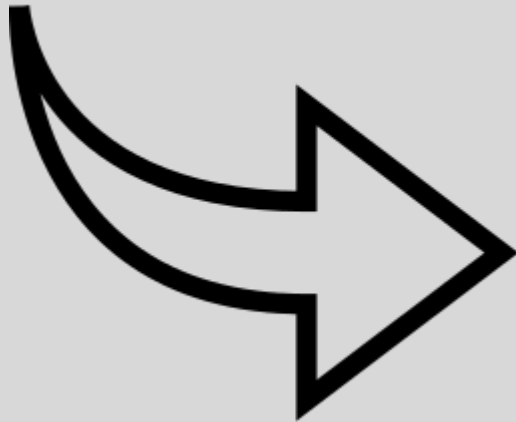
"Läuft das auch auf
meinem Handy?"



"Kann ich damit eine
Webseite programmieren?"

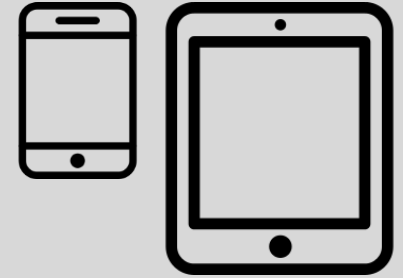


Weshalb **Web** und **Mobile** im Unterricht?



AKTUELLE **MÖGLICHKEITEN** FÜR WEB UND MOBILE

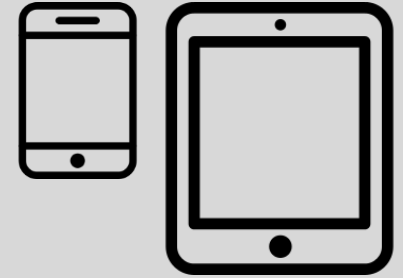
Problematik mobiler Anwendungen



iOS-Apps

- Sprache: Objective-C
- Laufen nur auf iPhones und iPads
- Entwicklung bevorzugt auf Mac-Betriebssystem
- Installation über App Store schwierig

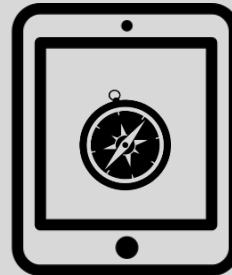
Problematik mobiler Anwendungen



Android-Apps

- Sprache: Java (eigene Philosophie)
- Laufen nur auf Android-Geräten

Eine Möglichkeit – Alles im Webbrowser



Problematik **webbasierter** Anwendungen

PHP, ASP.net, ...

- 👍 Häufig verwendet für dynamische Webanwendungen
- ✗ Komplexe Kommunikation mit Server
- ✗ Weniger geeignet für moderne, objektorientierte Programmierkonzepte



Problematik **webbasierter** Anwendungen

Python, Java, C++, Visual Basic, ...

- 👍 Gut für objektorientierte Konzepte
- ✘ Langer und steiniger Weg bis zu einer Webapplikation
- ✘ Komplexe Kommunikation mit Server
- ✘ Sprachen ursprünglich nicht für Webprogrammierung entwickelt



Problematik **webbasierter** Anwendungen

JavaScript

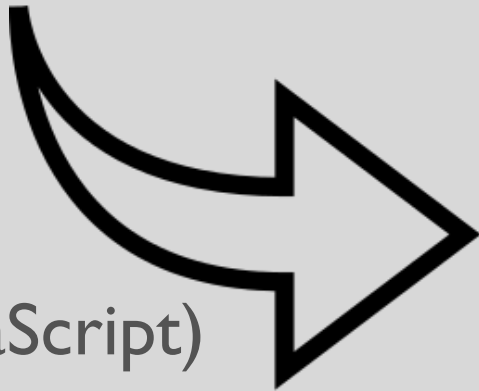
- 👍 Wird als einzige Sprache von allen gängigen Browsern unterstützt
- 👍 Webapplikationen für Mobile und Desktop sind möglich
- ✘ Schwierige Programmierkonzepte
- ✘ Viele Ausnahmefälle und Überraschungen



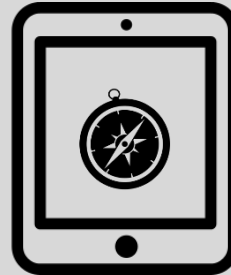
DIE SPRACHE DART



DART



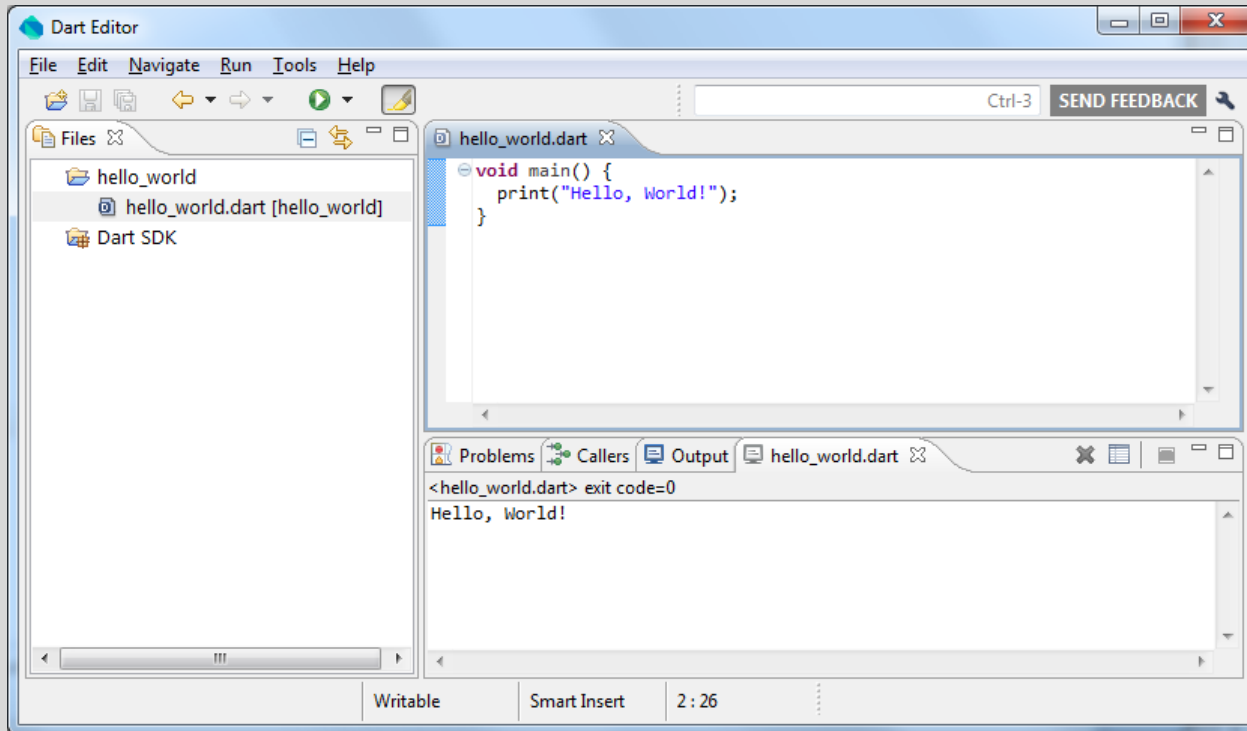
(JavaScript)





- 👍 Objektorientierte Sprache
- 👍 Stark an Java, C++, Smalltalk, ... angelehnt
- 👍 Einfacher zu lernen
- 👍 Dart Ökosystem
 - Dart Editor
 - Viele nützliche Bibliotheken: Math, HTML, Kryptografie, Datenbanken, ...
 - Hilfsbereite Open Source Community

Dart Editor



Code completion

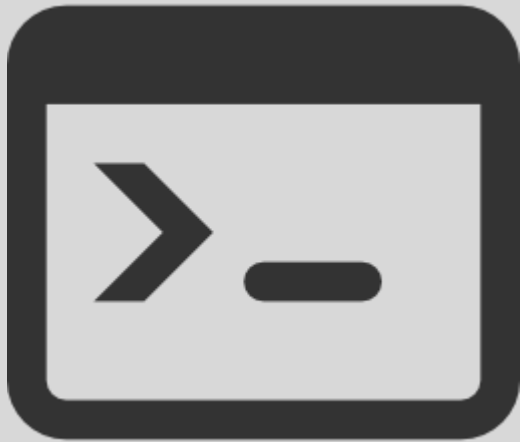
Debugger

Refactoring

Warnings

Demo

3 Möglichkeiten ein **Dart-Programm** zu **starten**



Hello World

Dart

```
main() {  
  print("Hello, World!");  
}
```

Java

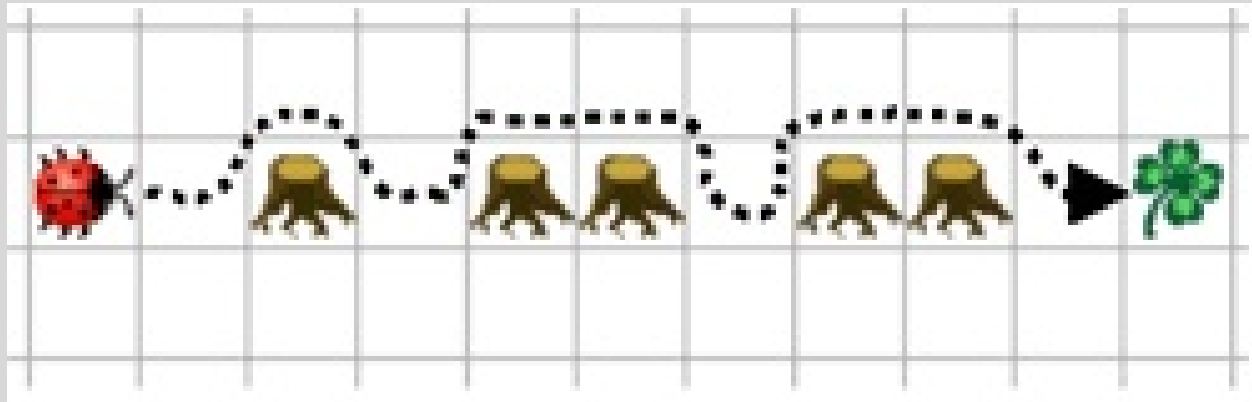
```
public class HelloWorld {  
  public static void main(String[] args) {  
    System.out.println("Hello, World!");  
  }  
}
```

PRAKTISCHER TEIL

Möglichkeiten für den Unterricht

- Konsole (Hello World, Math)
- Lernszenarien (Miniwelt Kara)
- Dynamische Webseiten (Todo-Liste)
- Spieleprogrammierung (Canvas)
- Client-Server Programmierung (z.B. Chat)

Die Welt von Kara



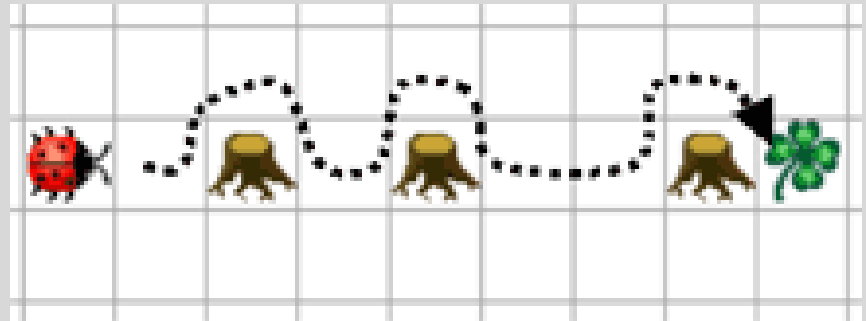
Unterlagen

- <http://edu.makery.ch/>
 - Unter **Projects, Learn Dart**

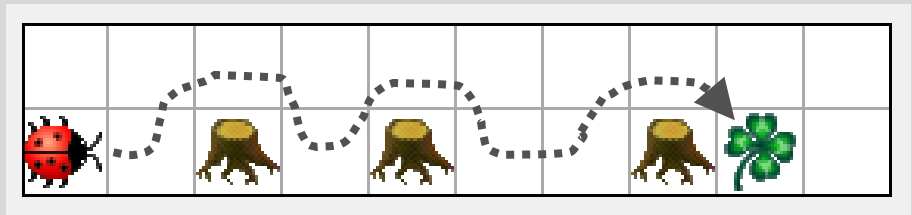
Aufgabe 2 – Um Baum herum (S. 5)

Schreiben Sie ein Programm, welches Kara auf dem angegebenen Weg zum Kleeblatt führt. Er muss dabei um die Bäume herumlaufen. Beim Kleeblatt angekommen soll er es aufheben.

```
void act() {  
    move();  
  
    turnLeft();  
    move();  
    turnRight();  
    move();  
    move();  
    turnRight();  
    move();  
    turnLeft();  
  
    ...  
}
```



Aufgabe 2



```
public class MyKara extends Kara
{
    public void act()
    {
        move();

        turnLeft();
        move();
        turnRight();
        move();
        move();
        turnRight();
        move();
        turnLeft();
    }
}
```



```
turnLeft();
move();
turnRight();
move();
move();
turnRight();
move();
turnLeft();
```



```
move();
```

```
turnLeft();
move();
turnRight();
move();
move();
turnRight();
move();
turnLeft();
```



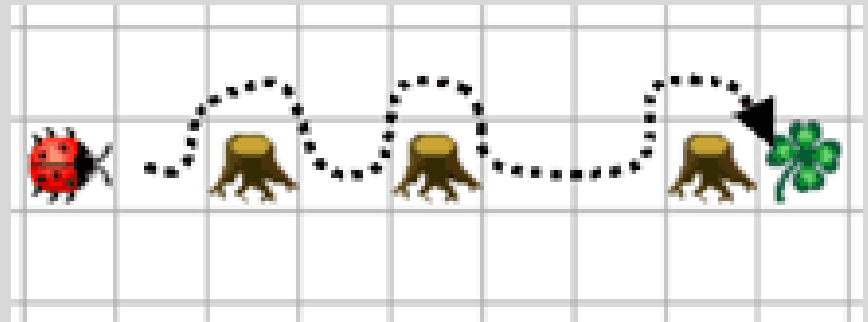
```
removeLeaf();
```



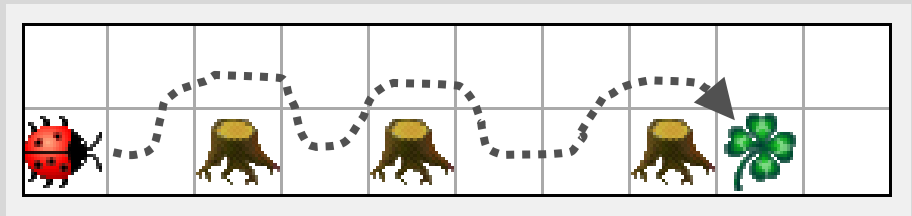
Aufgabe 3 – Um Baum Methode

Schreiben Sie zwischen die geschweiften Klammern der Methode `goAroundTree()` die Befehle, die es braucht, um um den Baum zu kommen.

```
void goAroundTree() {  
    turnLeft();  
    move();  
    turnRight();  
    move();  
    move();  
    turnRight();  
    move();  
    turnLeft();  
}
```




Aufgabe 3



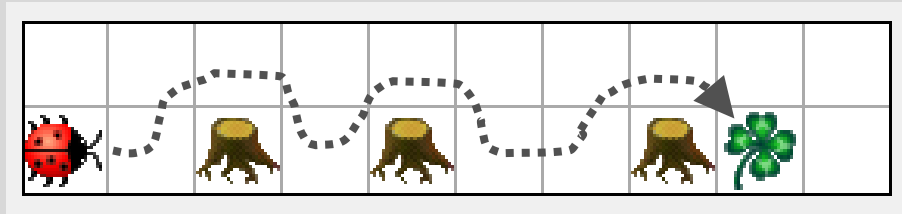
```
void act() {  
    move();  
    goAroundTree();  
    goAroundTree();  
    move();  
    goAroundTree();  
    removeLeaf();  
}
```

```
void goAroundTree() {  
    turnLeft();  
    move();  
    turnRight();  
    move();  
    move();  
    turnRight();  
    move();  
    turnLeft();  
}
```

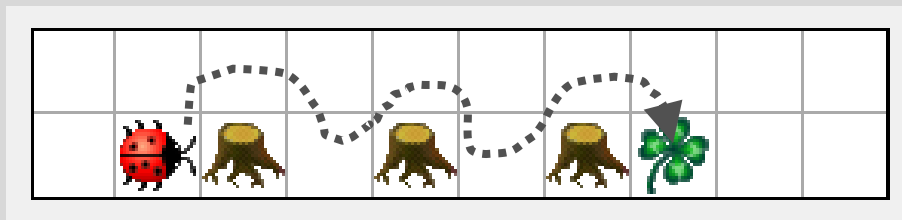


Was, wenn sich die Wiese ändert?

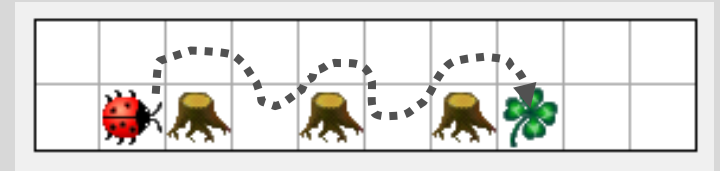
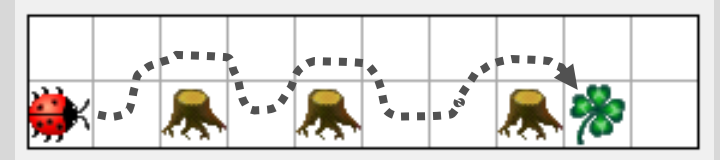
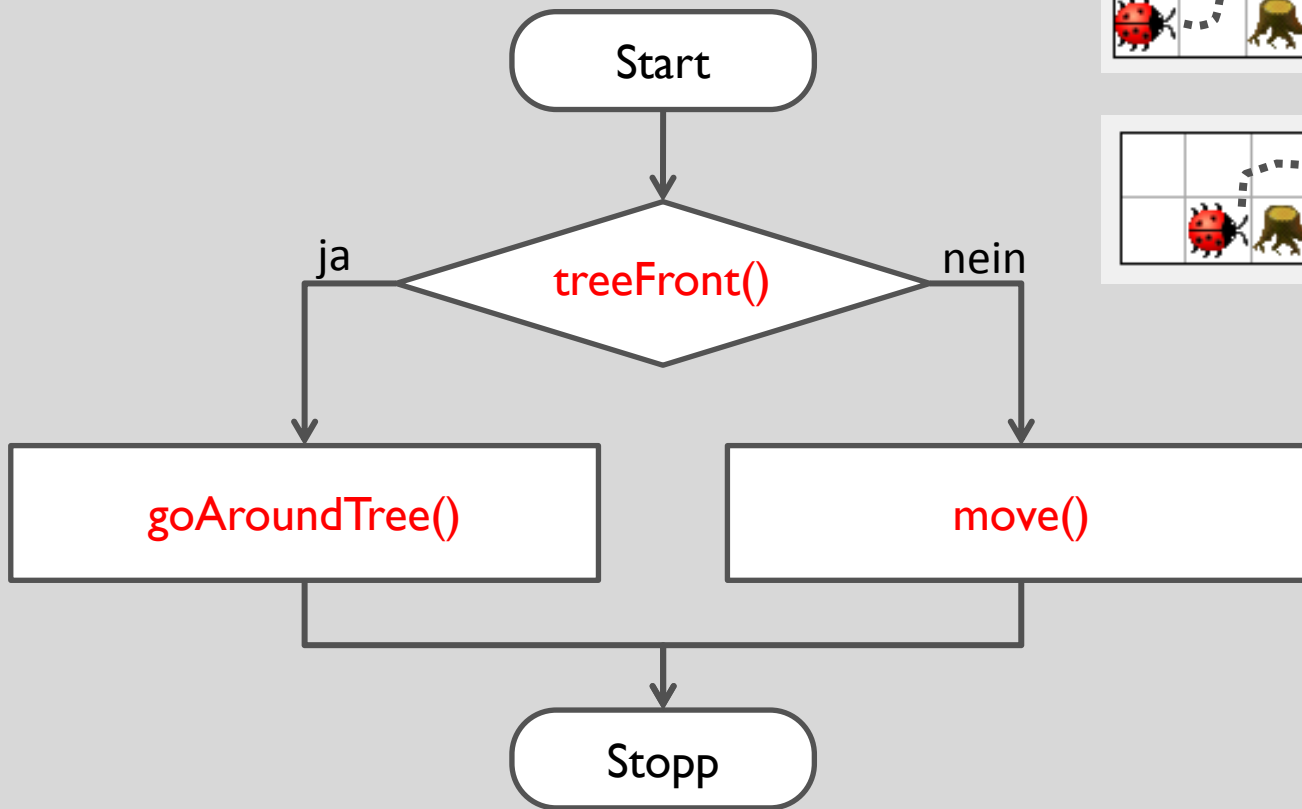
```
void act() {  
  move();  
  goAroundTree();  
  goAroundTree();  
  move();  
  goAroundTree();  
  removeLeaf();  
}
```



Was passiert auf dieser Wiese?



Ausblick Kapitel 2 Programmfluss



Möglicher Aufbau mit Kara

Tag 1

- Dart Editor kennen lernen
- Kara Szenario verstehen
- Ausführen und schreiben von Programmen

Tag 2

- Bedingungen (evtl. mit Flussdiagrammen)
- Logische Operatoren

Tag 3

- Schleifen
- Variablen

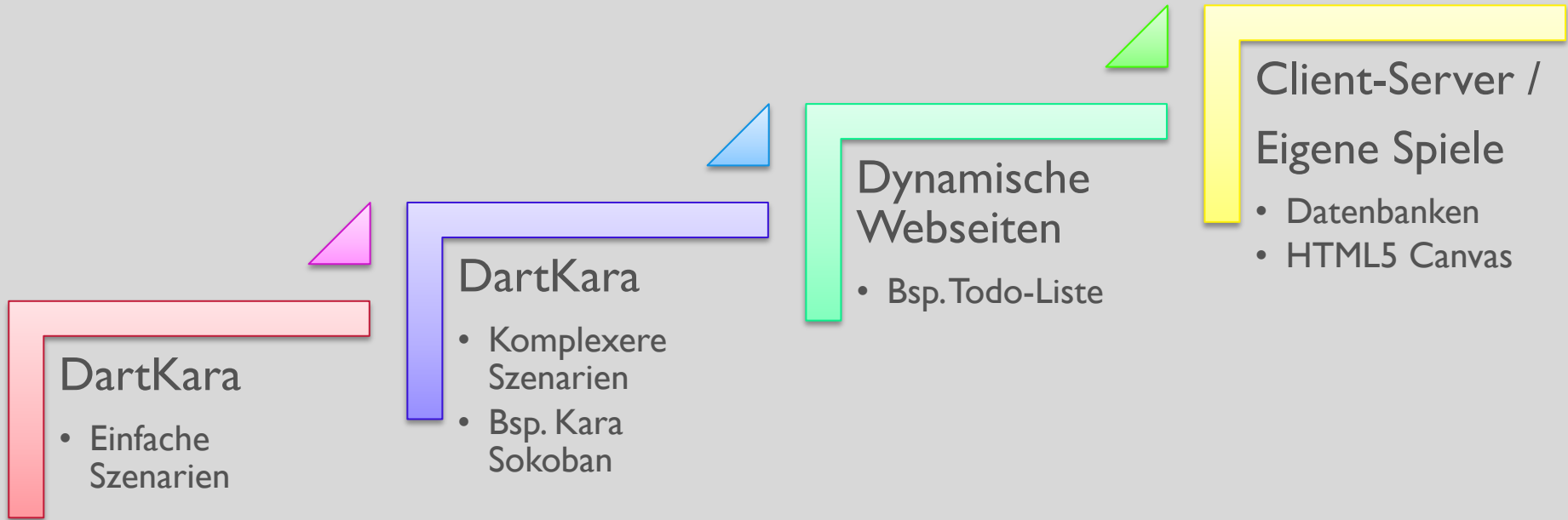
Tag 4 – Kara Sokoban

- Interaktive Steuerung
- Eigene Levels und Bilder
- Szenario veröffentlichen

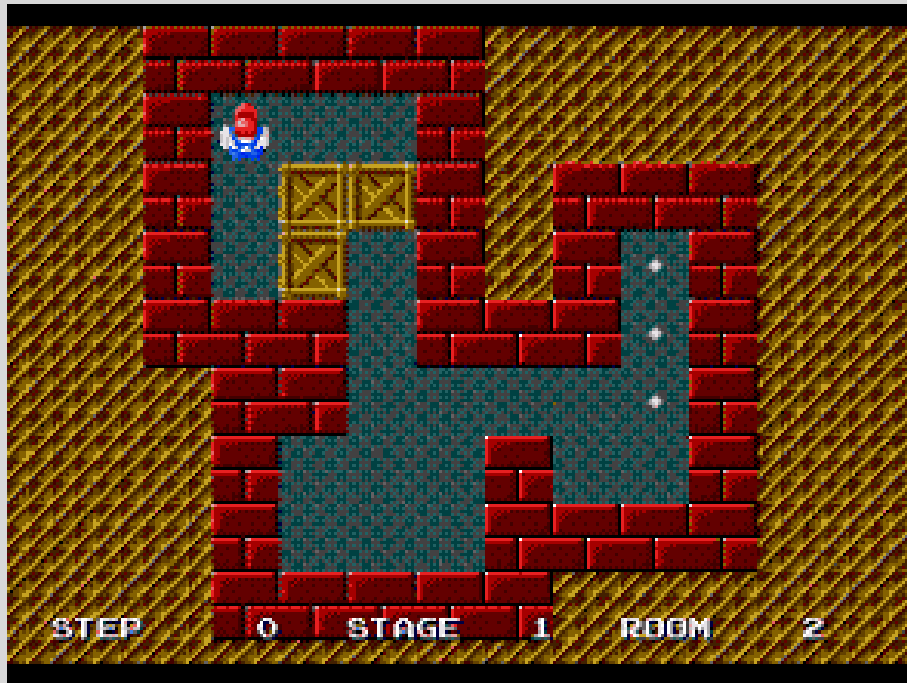
Tag 5

- Methoden mit Parametern und Rückgabewerten

Wie weiter...?



Beispiel Kara Sokoban



Beispiel Kara Sokoban

- Pfeil-Steuerung
 - Üben mit Variablen und Bedingungen
 - Dokumentation lesen
- Vor Baum schützen, Pilze schieben
 - Methode schreiben
- Schritte zählen
 - Variablen
- Eigene Levels
 - Kreativität mit ASCII
 - Levels mit Anderen teilen
- Highscore-Liste (für Fortgeschrittene)



DART CODE-BEISPIELE

Funktionen und Variablen

```
// Funktion definieren.  
printNumber(int number) {  
    // Nummer auf Konsole ausgeben.  
}  
  
main() {  
    // Variable deklarieren und initialisieren.  
    var number = 42;  
    // Funktion aufrufen.  
    printNumber(number);  
}
```

Alles sind Objekte

```
// String in einen int umwandeln.  
int i = int.parse("5");
```

```
// int in einen String umwandeln  
String s = 22.toString();
```

Klassen

```
import 'dart:math';

class Point {
  num x;
  num y;

  // Konstruktor (kurz)
  Point(this.x, this.y);
}

main() {
  var p = new Point(2, 3);
}
```

```
// Konstruktor (lang)
Point(num x, num y) {
  this.x = x;
  this.y = y;
}
```


String Interpolation

```
print('Hallo $name');
```

```
print('Distanz ${p.distanceTo(q)}');
```

Interaktion mit dem Browser

```
// HTML Knopf erstellen.  
var button = new ButtonElement()  
..text = 'Bestellen'  
..classes.add('wichtig')  
  
// Knopf in HTML einfügen.  
query('#bestellung').children.add(button);  
  
// Beim Klicken die Funktion handleClick aufrufen.  
button.onClick.listen(handleOnClick);  
  
void handleClick(MouseEvent event) {  
    window.alert('Danke!');  
}
```

Fazit?

attraktiv

aktuell

Programmier-
unterricht

wenig
Hürden

attraktiv

aktuell



DART

👍 Objektorientierte Sprache

👍 Client-Server (in einer Sprache)

👍 Professionelle Entwicklungsumgebung



wenig
Hürden



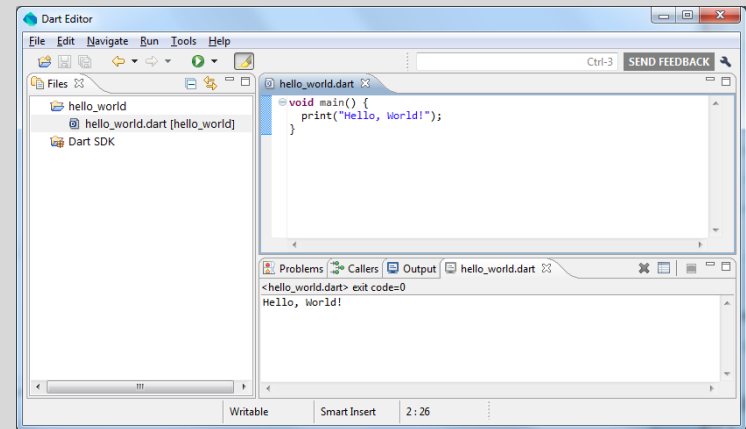
DART

👍 Elegante Sprache

👍 Grosse Ähnlichkeiten mit
Java, C#, etc.

👍 Einfacher Editor

👍 Viele nützliche Bibliotheken



Diskussion

- Meinung zu **Dart im Unterricht**
- Meinung zu **Einsatzmöglichkeiten**
 - Konsole (Hello World, Math)
 - Lernszenarien (Kara)
 - Dynamische Webseiten (Todo-Liste)
 - Spieleprogrammierung (Canvas)
 - Client-Server Programmierung (z.B. Chat)

Umfrage



- Ihre Meinung!!
 - Soll das Material weiterentwickelt werden?
- Umfrage:

<http://goo.gl/sYwnG1>