

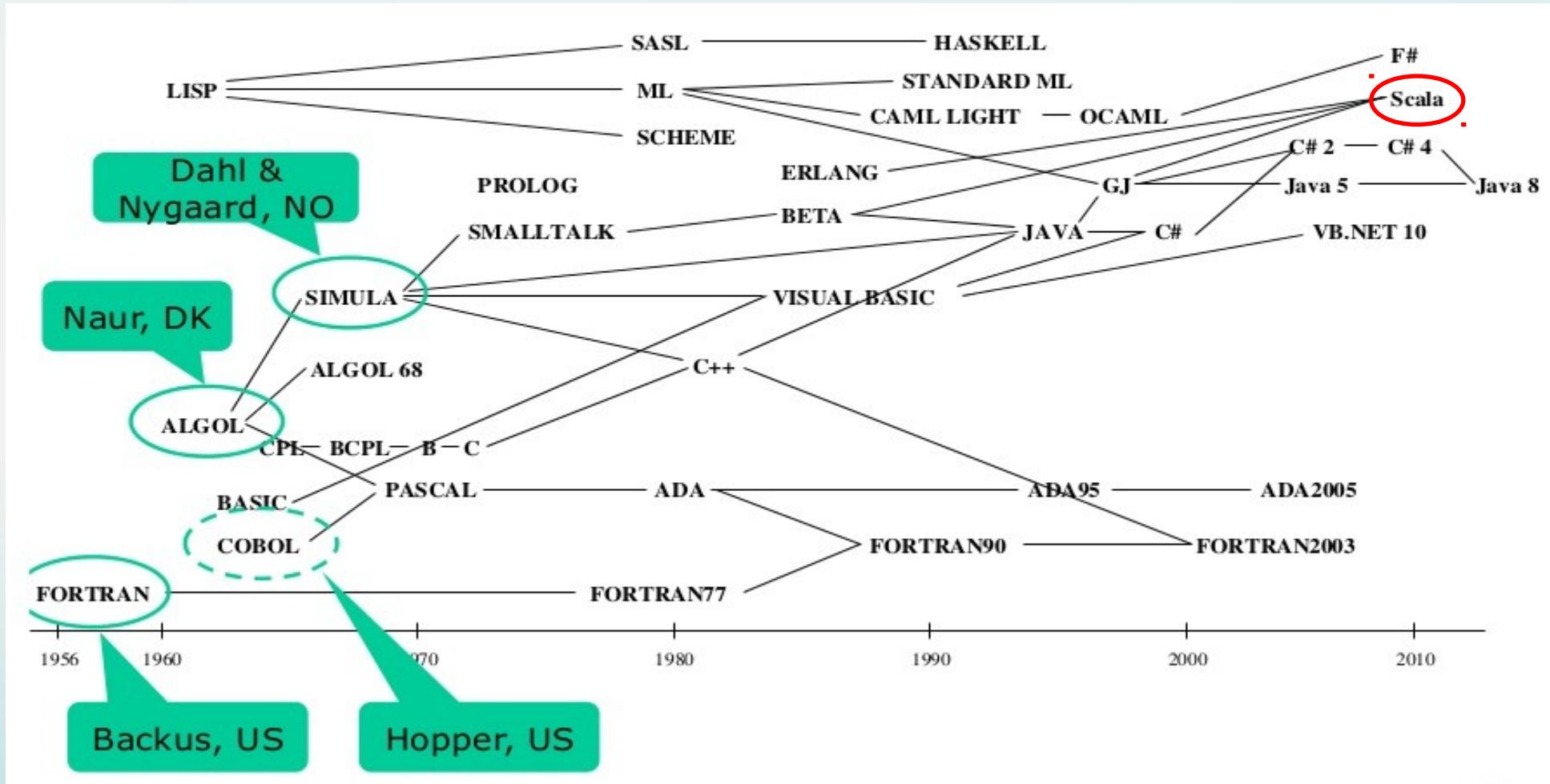
Funkcionalno programiranje

Uvod

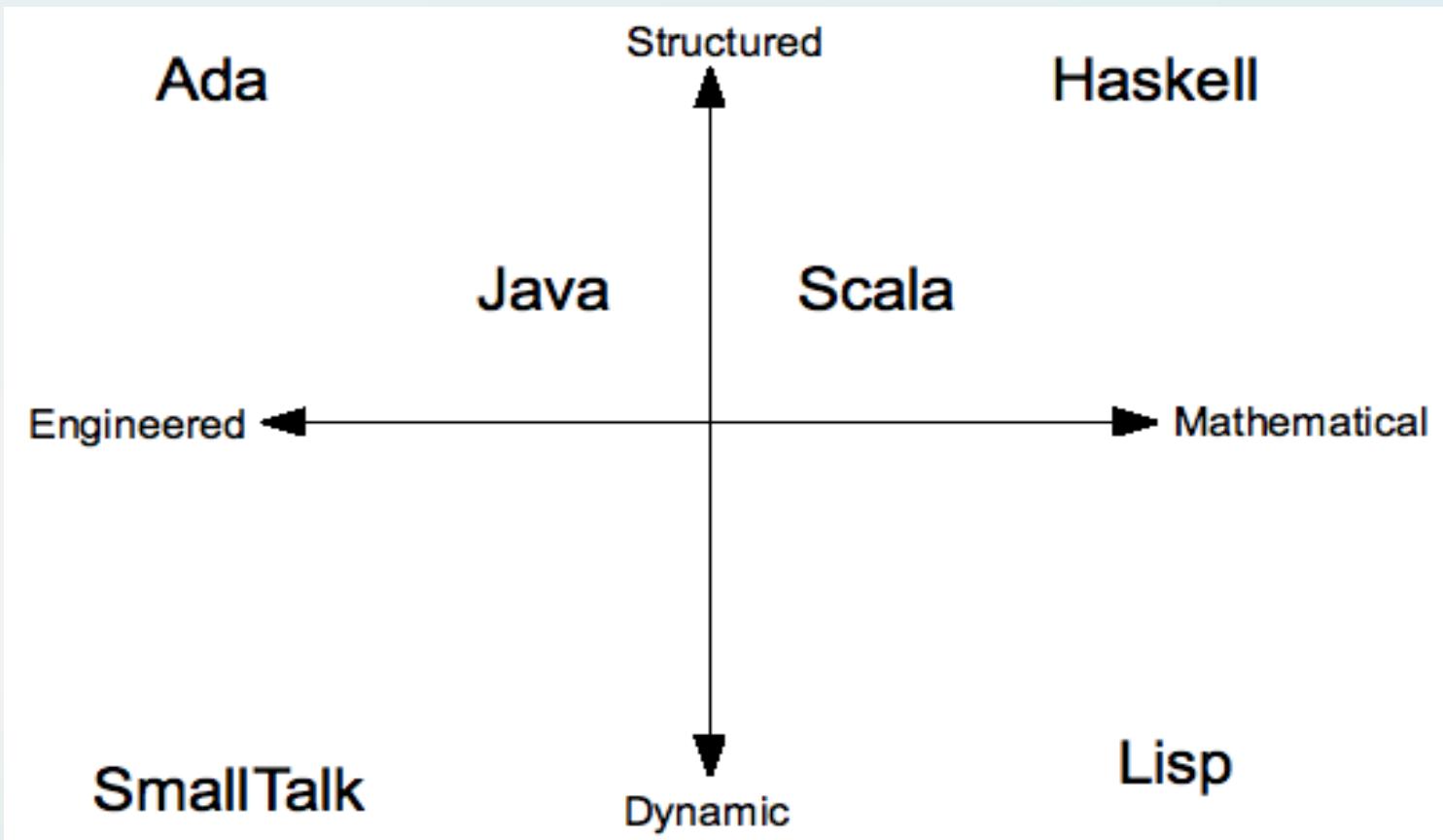
Funkcionalno programiranje - uvod

- Razlike u odnosu na druge paradigme u programiranju
 - nepromenljivi (nemutabilni) parametri funkcija
 - funkcije služe za **preslikavanje** ulaznih podataka u izlazne
 - funkcije su "građani prvog reda"
 - postoje funkcijski literali, promenljive
 - funkcije mogu biti prenete kao parametri drugih funkcija
 - funkcije mogu da vrate funkcije kao svoje rezultate
- Scala
 - objedinjuje koncepte OO, imperativ. i funkc. programiranja
 - "skalira" sa potrebama programera
 - može se koristiti za pisanje malih skripti
 - može se koristiti za pisanje složenih softverskih sistema

Genealoško stablo nekih programskih jezikov

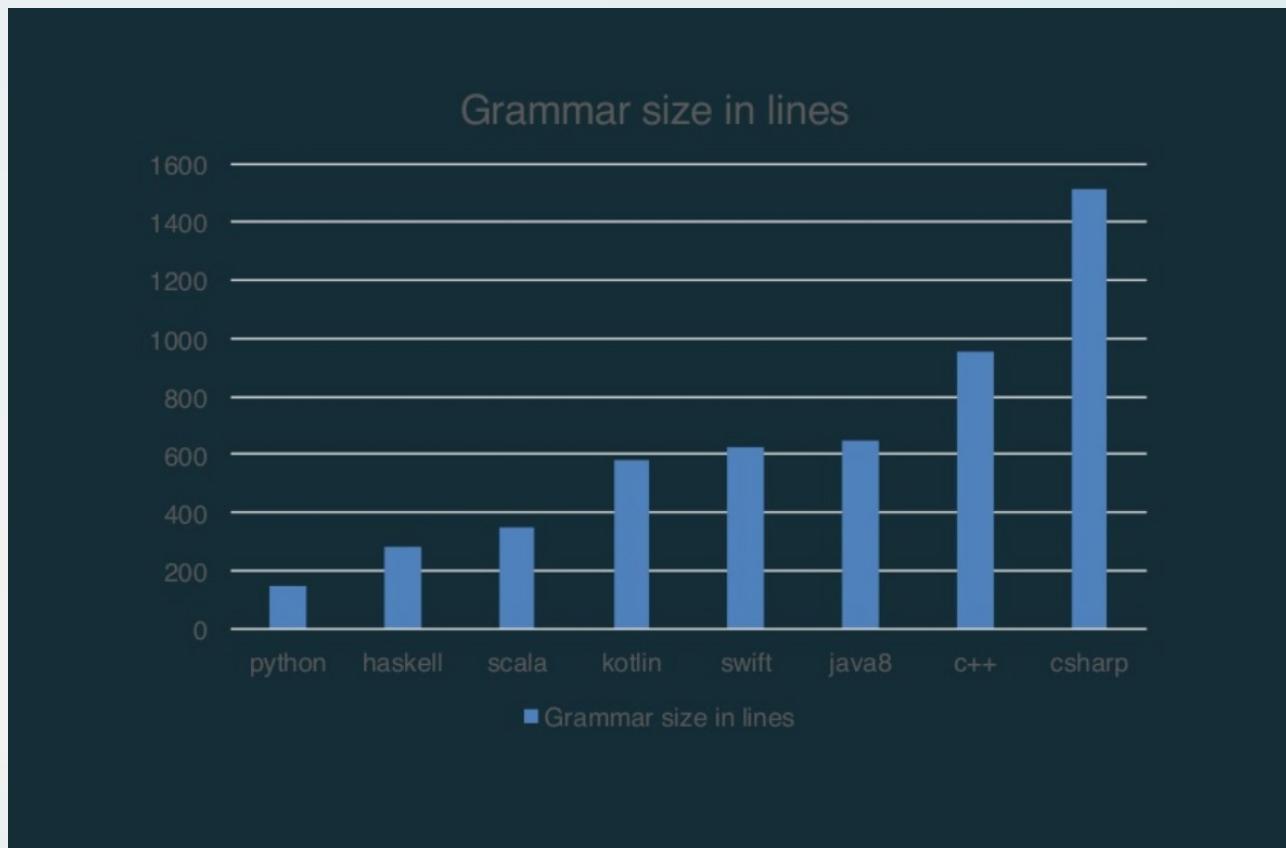


Strukturiranost i namena jezika



<http://erikengbrecht.blogspot.rs/2008/01/programming-language-continuum.html>

Kompleksnost jezika



www.slideshare.net/Odersky/preparing-for-scala-3

Paradigme u programiranju

- Paradigma:
 - ono što služi kao obrazac ili model
 - skup prepostavki, koncepata, vrednosti – način razmišljanja
 - ►obrazac koji služi kao pravac razmišljanja u programiranju
- Glavne paradigme
 - imperativno programiranje
 - funkcionalno programiranje
 - logičko programiranje
- Ortogonalno na glavne paradigme
 - objektno-orientisano programiranje

Imperativno programiranje

- Programira se upotrebom:
 - promenljivih (varijabli)
 - dodela
 - kontrolnih struktura (uslovna grananja, ciklusi, ...)

Funkcionalno programiranje

- U užem smislu:
programiranje bez promenljivih, dodela ili kontrolnih struktura
- U širem smislu:
programiranje sa akcentom na upotrebi funkcija

Funkcionalno programiranje

- Funkcije postaju vrednosti koje se
 - proizvode
 - konzumiraju
 - sastavljaju
- Funkcionalan jezik pojednostavljuje ovakav pristup
 - funkcije mogu biti definisane unutar drugih funkcija
 - funkcije mogu biti parametri i rezultati drugih funkcija
 - postoje osnovne operacije koje omogućavaju sastavljanje funkcija

Poređenje Java/Scala (1/4)

Java

```
public class Person {  
    public final String name;  
    public final int age;  
    Person(String name, int age) {  
        this.name = name;  
        this.age = age;  
    }  
}
```

Scala

```
class Person(val name: String,  
           val age: Int)
```

Poređenje Java/Scala (2/4)

```
Java    Person[] people;
        Person[] minors;
        Person[] adults;
        ...
        {
            ArrayList<Person> minorsList = new ArrayList<>();
            ArrayList<Person> adultsList = new ArrayList<>();
            for(int i = 0; i < people.length; i++)
                (people[i].age < 18 ? minorsList : adultsList)
                    .add(people[i]);
            minors = minorsList.toArray(people);
            adults = adultsList.toArray(people);
        }
```

```
Scala   val people: Array[Person]
        val (minors, adults) = people partition (_ .age < 18)
```

Poređenje Java/Scala (3/4)

Java

```
Person[] people;
Person[] minors;
Person[] adults;
...
{
    ArrayList<Person> minorsList = new ArrayList<>();
    ArrayList<Person> adultsList = new ArrayList<>();
    for(int i = 0; i < people.length; i++)
        (people[i].age < 18 ? minorsList : adultsList)
            .add(people[i]);
    minors = minorsList.toArray(people);
    adults = adultsList.toArray(people);
}
// Kako paralelizovati?
```

Scala

```
val people: Array[Person]
val (minors, adults) = people.par partition (_.age < 18)
```

Poređenje Java/Scala (4/4)

- Novije verzije Jave su donele koncepte iz funkcionalnog programiranja
 - lambda funkcije
 - tokove (stream) za podršku paralelnoj obradi
- To ukazuje na trendove povećanog interesa za FP
- Međutim
 - sam jezik Java je i dalje veoma eksplicitan
 - nije koncipiran oko ideje funkcionalnog programiranja
 - nije proširljiv tako da proširenja deluju kao deo osnovnog jezika

Scala

- Jezik razvijen na EPFL-u (Martin Odersky)
 - Poseduje sve konstrukte funkcionalnih jezika
 - Statičko tipiziranje
 - Objektno-orientisan jezik
 - Funkcioniše na JVM, interoperabilnost sa Javom
 - Adaptiran za simbolički račun
 - Adaptiran za paralelno izvršavanje
- Kao i svaki netrivijalni jezik, poseduje:
 - primitivne izraze – najjednostavnije entitete kojima se manipuliše
 - sredstvo kombinovanja – za sastavljanje složenih elemenata od jednostavnih
 - sredstvo apstrakcije – za imenovanje elemenata i njihovo upravljanje kao da su osnovni

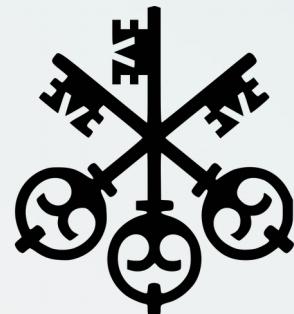
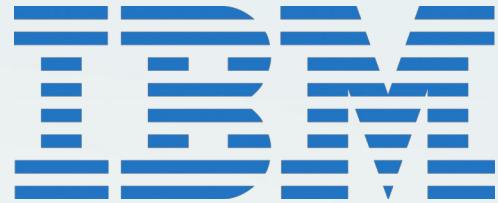
Prednosti FP

- Manje pisanja koda
 - jednostavnije za razumevanje (ali zahteva veliko predznanje)
 - jednostavnije za održavanje, povećava produktivnost
- Nema bočnih efekata
 - podaci "ulaze" u funkcije, ali ih funkcije ne menjaju
 - jednostavnije za razumevanje
 - reupotrebljivost koda
- Rekurzija
 - prirodna kontrolna struktura u funkcionalnim jezicima
 - alternativa za cikluse u imperativnim jezicima

Mane FP

- Ulaz i izlaz
 - prirodna je protočna obrada, ne odgovara stilu FP
- Složeniji razvoj interaktivnih aplikacija
 - zasnivaju se na ciklusima zahtev/odgovor
 - inicira korisnik
- Programi koji se izvršavaju u ciklusu su složeniji za razvoj
- Manja efikasnost na modernom hardveru
 - FP je dominantno bio zastavljen u akademiji, gde performanse nisu bile od velikog značaja
- Nije orijentisano ka podacima
 - dohvatanje, manipulisanja i vraćanje podataka (baze podataka)
 - OO pristup je prirodniji

Scala danas (1/2)



UBS

Linkedin

foursquare™



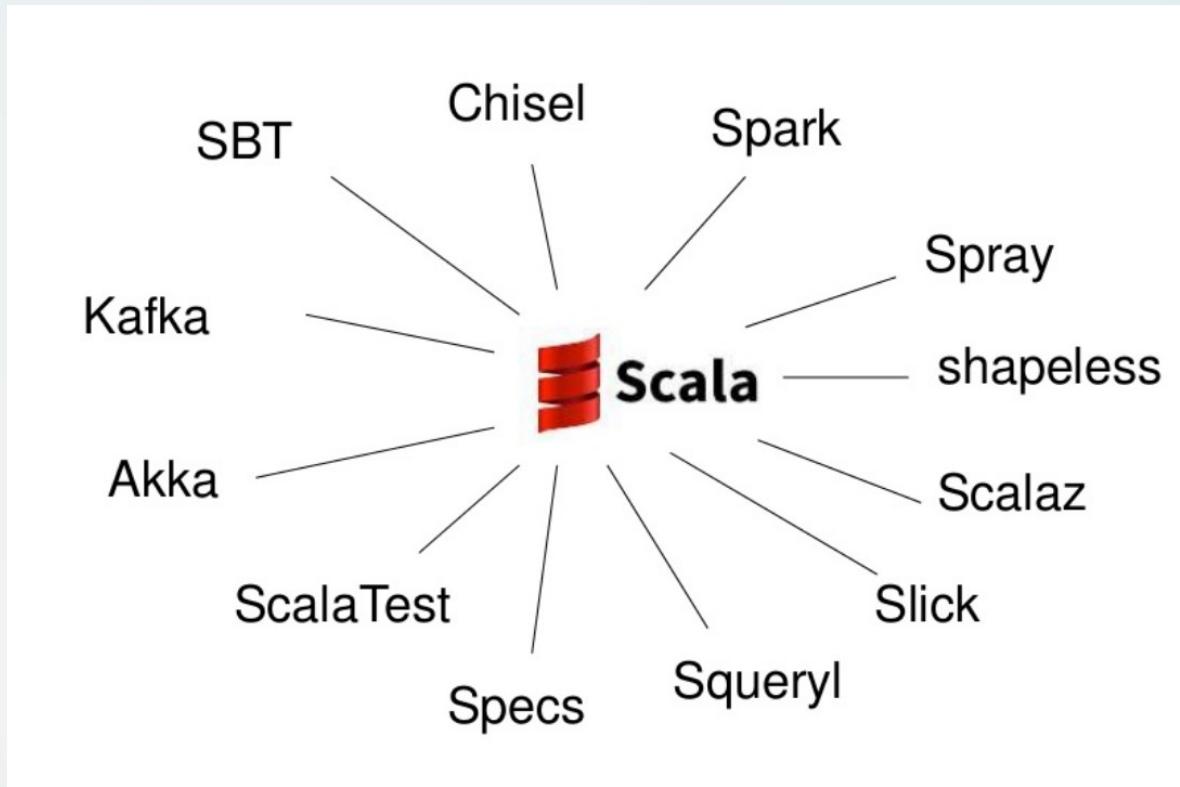
tumblr.

NETFLIX

the guardian

SONY

Scala danas (2/2)



Radno okruženje (1/3)

- REPL (Read-Eval-Print-Loop)

```
C:\Windows\system32\cmd.exe - scala
a: Int = 5

scala> var capital = Map("US"-">"Washington", "Japan"-">"Tokyo")
capital: scala.collection.immutable.Map[String,String] = Map(US -> Washington, J
apan -> Tokyo)

scala> capital += ("France"-">"Paris")

scala> capital
res8: scala.collection.immutable.Map[String,String] = Map(US -> Washington, Japa
n -> Tokyo, France -> Paris)

scala> println(capital("France"))
Paris

scala> -
```

```
File Edit View Search Terminal Help
Welcome to Scala 2.11.12 (OpenJDK 64-Bit Server VM, Java 10.0.2).
Type in expressions for evaluation. Or try :help.

scala> def pi = 3.141592653589793238462
pi: Double

scala> def radius = 10
radius: Int

scala> 2*pi*radius
res0: Double = 62.83185307179586

scala> []
```

Radno okruženje (2/3)

- IDE (ScalaIDE, Eclipse, NetBeans, IntelliJ)

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Scala - MaterijaliSaPredavanja/src/States.sc - Eclipse
- Menu Bar:** File, Edit, Refactor, Navigate, Search, Project, Scala, Run, Window, Help
- Toolbar:** Includes icons for file operations like New, Open, Save, Cut, Copy, Paste, Find, and Run.
- Package Explorer:** Shows the project structure: MaterijaliSaPredavanja > src > States.sc.
- Editor:** Displays the Scala code:

```
1 object States {  
2  
3     var capital = Map("US" -> "Washington", "France" -> "Paris")  
4  
5     capital += ("Japan" -> "Tokyo")  
6     println(capital("France"))  
7  
8 }  
//| on, France -> Paris)  
//> Paris
```
- Console:** Shows the output of the Scala Interpreter:

```
val Pi = 3.14159265358979323864246  
Pi: Double = 3.141592653589793
```

Radno okruženje (3/3)

- Scastie: scastie.scala-lang.org/

The screenshot shows the Scastie web interface. The top navigation bar includes the Scastie logo, Feedback, and Login links. The main toolbar has buttons for Editor (selected), Run, New, Format, Clear Annotations, Worksheet (with a green dot), and Save. On the left, a sidebar shows Build Settings with the code: `List("Hello", "World").mkString("", " ", " ", "!")`. The main workspace displays the result: `Hello, World!: java.lang.String`. Below the workspace is a dark console window showing the command-line interaction:

```
scastie: Sending task to the server.  
scastie: Connected. Waiting for sbt  
sbt: [info] Compiling 1 Scala source to /tmp/scastie8820588691387975860/target/scala-2.12/classes ...  
sbt: [info] Running Main  
scastie: Closed.
```

The bottom left sidebar includes Dark, Help, and Up buttons.

- “If I were to pick a language to use today other than Java, it would be Scala.”
- James Gosling (Java)
- “I can honestly say if someone had shown me the Programming in Scala book by Martin Odersky, Lex Spoon & Bill Venners back in 2003 I'd probably have never created Groovy.”
- James Strachan (Groovy)

Dodatno

- scala-lang.org
- courseware.epfl.ch/courses/course-v1:EPFL+progfun1+2018_T1/