Introduction

COMP 524: Programming Language Concepts
Björn B. Brandenburg

The University of North Carolina at Chapel Hill

Based in part on slides and notes by S. Olivier, A. Block, N. Fisher, F. Hernandez-Campos, and D. Stotts.

About this Class

Programming Language Concepts

Goals.

- Study **concepts** and **abstractions** used in programming language design.
- Working knowledge of **parsing** and **grammars**.
- Gain **overview of major paradigms**.

Prerequisites.

- COMP 410: Data Structures.
- Proficient in Java.
- Comfortable with programming.

Alternatives.

- COMP 520: Compilers.
- COMP 523: Software Engineering.

Tell me about yourself.

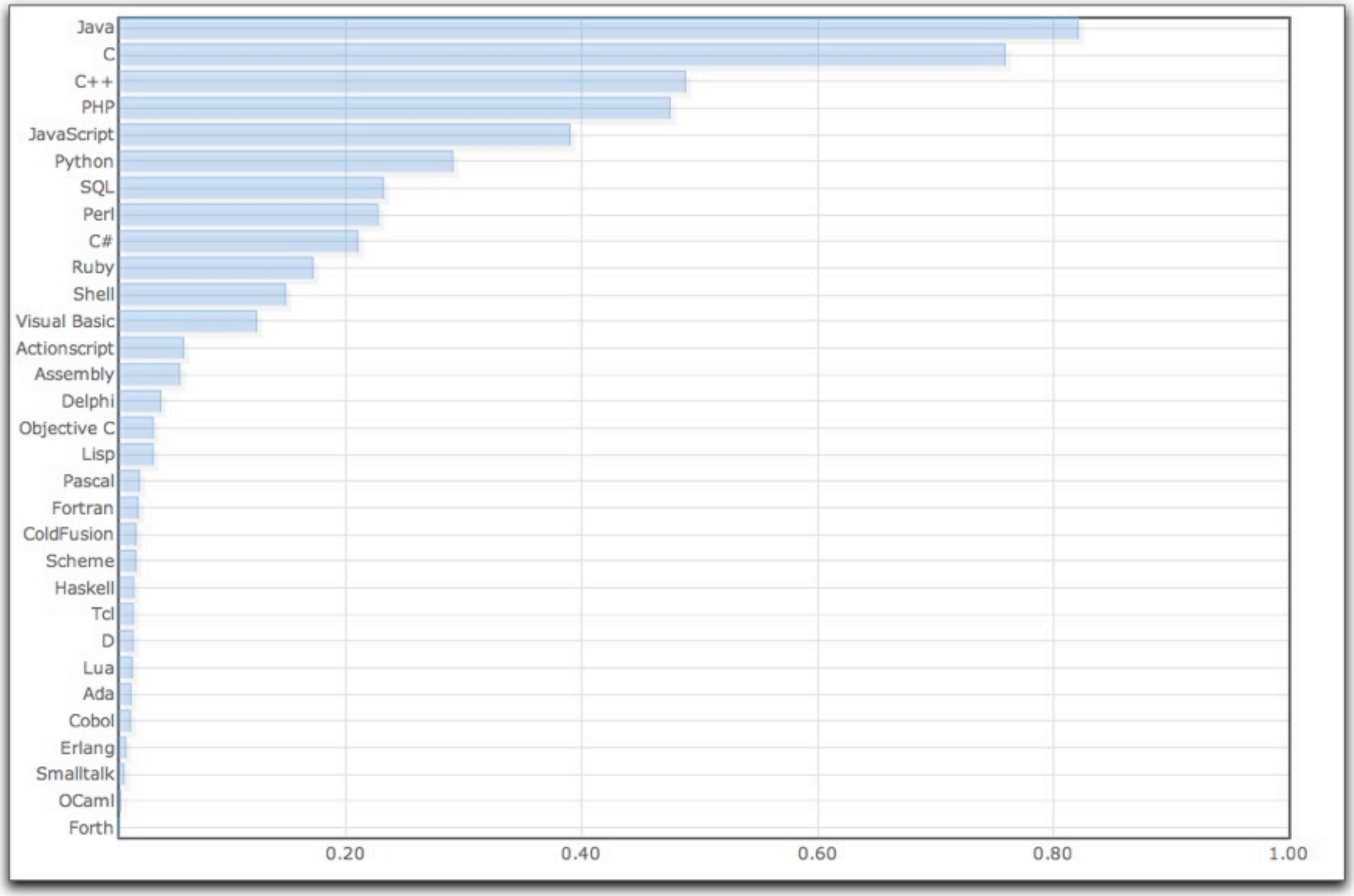
- What programming languages do you know?
- What do you expect to learn in this class?
- Any topic that you would like to see covered in particular?
- What's the coolest program that you've written/ worked on?
- Do you plan to go to grad school?

Motivation

Why study programming languages (PLs)?

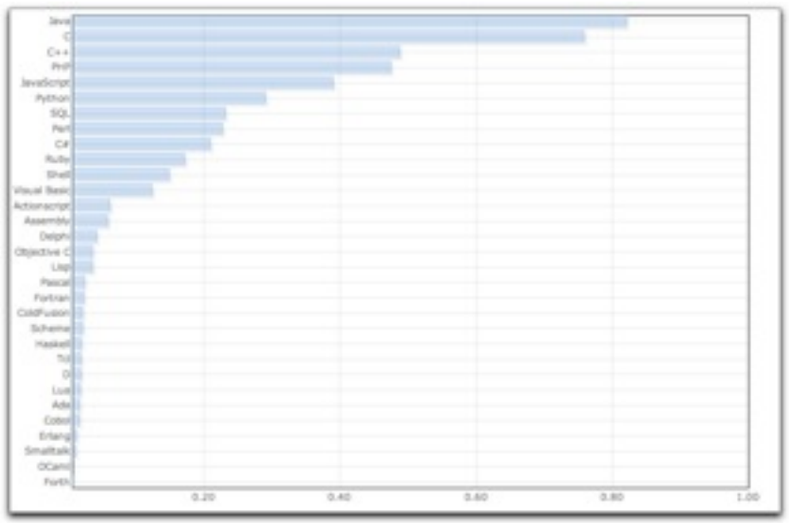
- It's **fun**.
- It's part of the very **core of computer science**.
- To **avoid re-inventing** the wheel.
- To **better apply PLs** you already know.
  - Understand the underlying design decisions.
- To be able to **effectively communicate** your ideas and questions about PLs.
- To **survive job interviews**.

Approximate Programming Language Popularity



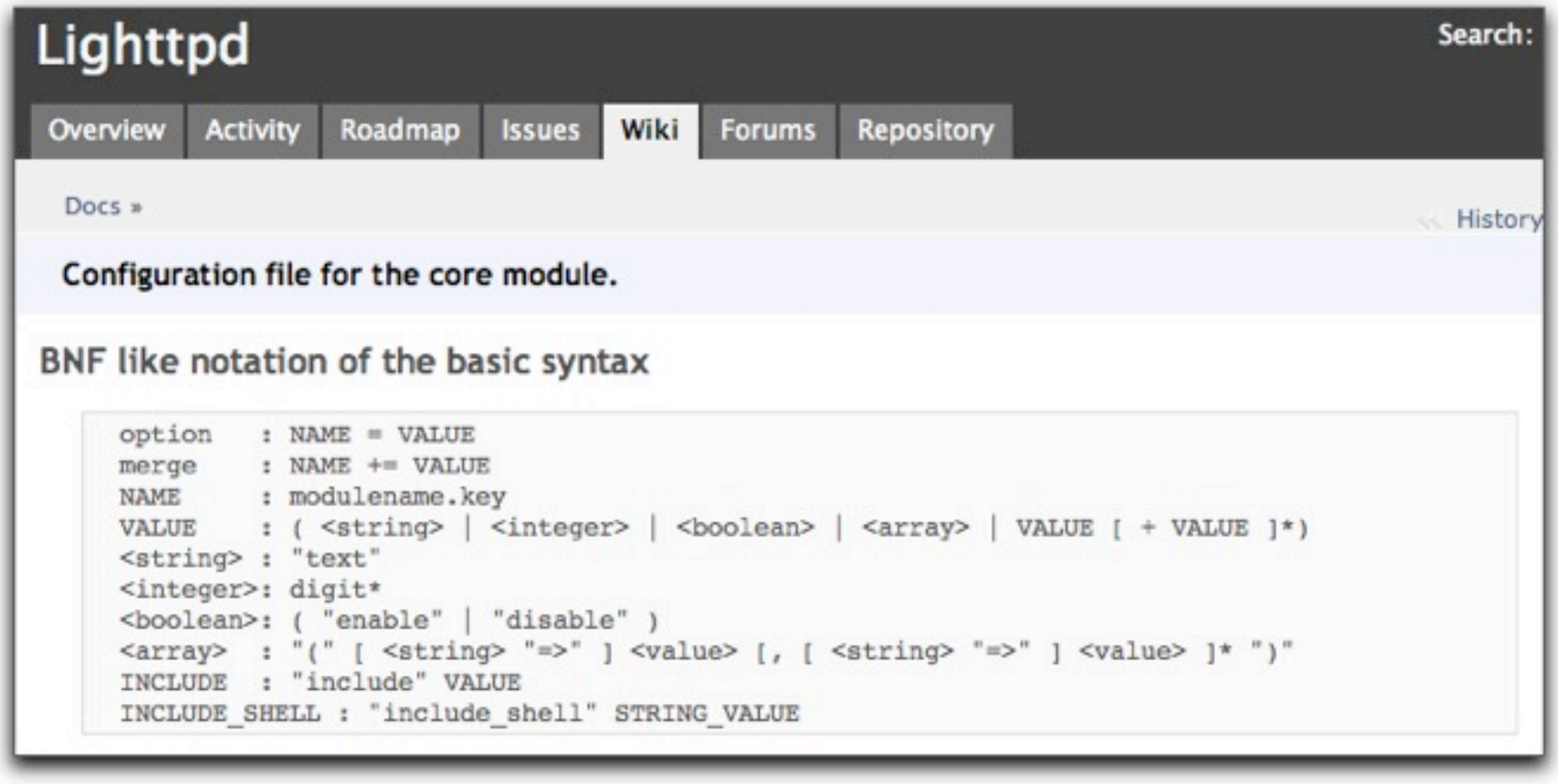
Source: http://www.langpop.com/

Useful Job Skills



- Make **educated decisions** when choosing PLs for a project.
- Which **features** is your chosen PL missing?
  - Can they be **emulated** in a library?
- **Learn** new PLs more quickly.

Useful Job Skills (II)



Web server configuration. *Source: http://redmine.lighttpd.net/projects/lighttpd/wiki/Docs:Configuration*

- Know how to create/describe/parse a (mini) language.
  - For example, **configuration files**.
- Ready to study and apply more **advanced texts**.

Useful Job Skills (III)

- Ability to **understand** and implement **specifications**.

Example: XML and Javascript.
(Javascript is officially named ECMAScript.)

W3C Recommendation

[4] NameStartChar ::= ":" | [A-Z] | "_" | [a-z] | [#xC0-#xD6] | [#xD8-#xF6] | [#xF8-#x2FF] | [#x370-#x37D] | [#x37F-#x1FFF] | [#x200C-#x200D] | [#x2070-#x218F] | [#x2C00-#x2FEF] | [#x3001-#xD7FF] | [#xF900-#xFDCF] | [#xFDF0-#xFFFD] | [#x10000-#xEFFFF]

[4a] NameChar ::= NameStartChar | "-" | "." | [0-9] | #xB7 | [#x0300-#x036F] | [#x203F-#x2040]

[5] Name ::= NameStartChar (NameChar)*

[6] Names ::= Name (#x20 Name)*

[7] Nmtoken ::= (NameChar)+

[8] Nmtokens ::= Nmtoken (#x20 Nmtoken)*

Note:

The Names and Nmtokens productions are used to define the validity of tokenized attribute values after normalization (see 3.3.1 Attribute Types).

XML specification. *Source: http://www.w3.org/TR/REC-xml/*

RegularExpressionBody ::
RegularExpressionFirstChar RegularExpressionChars

RegularExpressionChars ::
[empty]
RegularExpressionChars RegularExpressionChar

RegularExpressionFirstChar ::
RegularExpressionNonTerminator **but not** * **or** \ **or** / **or** [
RegularExpressionBackslashSequence
RegularExpressionClass

RegularExpressionChar ::
RegularExpressionNonTerminator **but not** \ **or** / **or** [
RegularExpressionBackslashSequence
RegularExpressionClass

RegularExpressionBackslashSequence ::
\ *RegularExpressionNonTerminator*

RegularExpressionNonTerminator ::
SourceCharacter **but not** *LineTerminator*

RegularExpressionClass ::
[*RegularExpressionClassChars*]

RegularExpressionClassChars ::
[empty]
RegularExpressionClassChars RegularExpressionClassChar

RegularExpressionClassChar ::
RegularExpressionNonTerminator **but not**] **or** \
RegularExpressionBackslashSequence

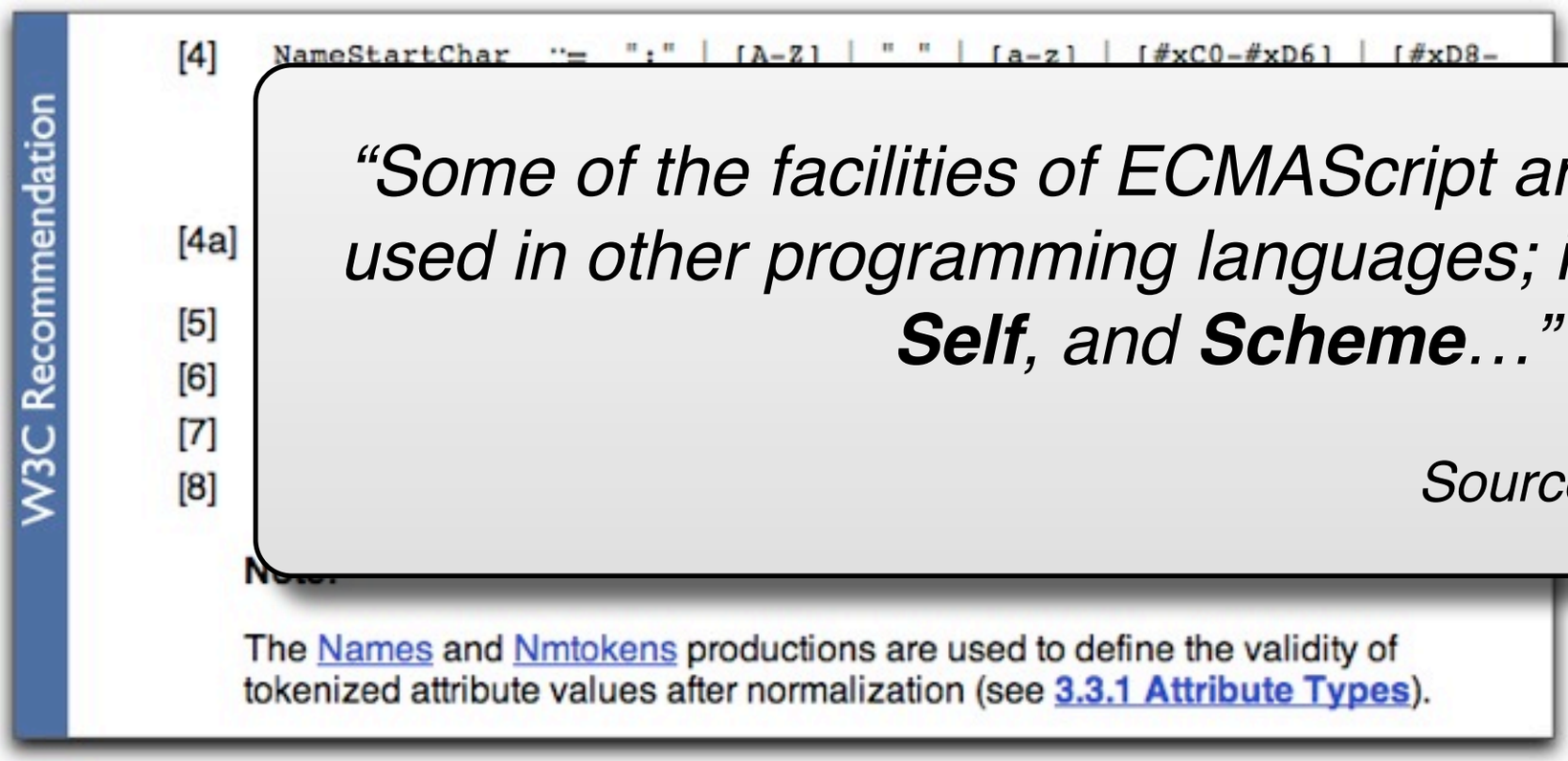
RegularExpressionFlags ::
[empty]
RegularExpressionFlags IdentifierPart

Javascript syntax. *Source: ECMA 262 standard.*

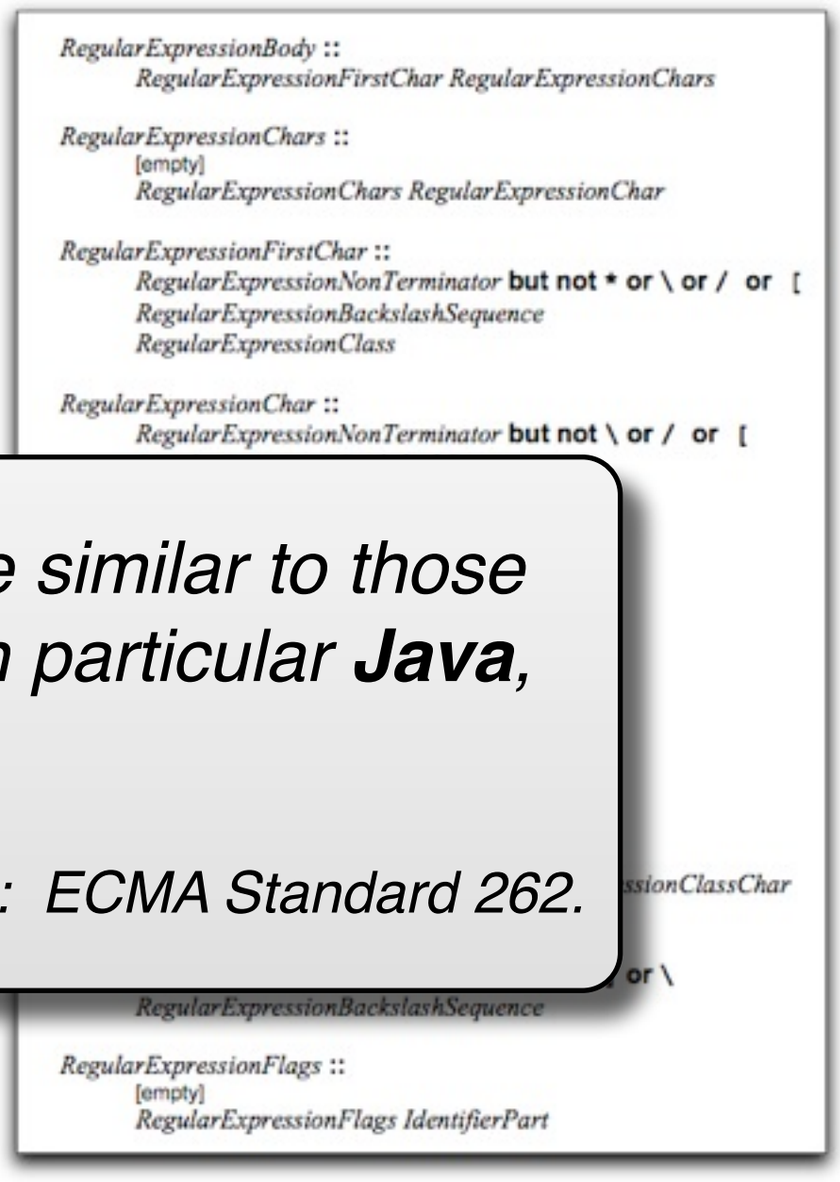
Useful Job Skills (III)

- Ability to **understand** and implement **specifications**.

Example: XML and Javascript.
(Javascript is officially named ECMAScript.)



XML specification. *Source: http://www.w3.org/TR/REC-xml/*



Javascript syntax. *Source: ECMA 262 standard.*

> *"Some of the facilities of ECMAScript are similar to those used in other programming languages; in particular **Java**, **Self**, and **Scheme**…"*
>
> *Source: ECMA Standard 262.*

Topics/Scope

Foundations.
- Syntax and syntactical analysis.
- Binding, scope, and storage.
- Semantic analysis.

Core language design.
- Control flow and subroutines.
- Evaluation strategies.

Paradigms.
- Object orientation.
- Functional programming.
- Logic programming.
- Scripting languages.

Select high-impact topics.
- Concurrency.
- Security concerns.
- Runtime systems.

Programming languages.
- Java.
- Prolog.
- Haskell.
- Python.

Topics/Scope

Foundations.
- Syntax and syntactical analysis.
- Binding
- Semant

Paradigms
- Object
- Functio
- Logic pr
- Scriptin

Core language design.
- Control flow and subroutines.
- Evaluation strategies.

pics.

Notable omissions.
- Formal background (see COMP 455).
- Target architectures (see COMP 411).
- Code generation and optimization (covered in COMP 520).
- Formal treatment of semantics (advanced grad level topic of little practical relevance).

Programming languages.
- Java.
- Haskell.
- Prolog.
- Python.

Class Rules

Let's have a look at the syllabus…