What Programming Languages Do Developers Use?

A Theory of Static vs. Dynamic Language Choice

Aaron Pang, Craig Anslow, James Noble
Victoria University of Wellington
New Zealand
craig@ecs.vuw.ac.nz
Languages, Languages, Languages!!!

Java, Python, JavaScript, Ruby, Visual Basic, Visual Basic .NET, Assembly, PHP, Go, Delphi, SQL, PL/SQL, MATLAB, Swift, Objective-C
## Tiobe Index: Dec, 2018

<table>
<thead>
<tr>
<th>Dec 2018</th>
<th>Dec 2017</th>
<th>Change</th>
<th>Programming Language</th>
<th>Ratings</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>Java</td>
<td>15.932%</td>
<td>+2.66%</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td>C</td>
<td>14.282%</td>
<td>+4.12%</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>^</td>
<td>Python</td>
<td>8.376%</td>
<td>+4.60%</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>↓</td>
<td>C++</td>
<td>7.562%</td>
<td>+2.84%</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>^</td>
<td>Visual Basic .NET</td>
<td>7.127%</td>
<td>+4.66%</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>↓</td>
<td>C#</td>
<td>3.455%</td>
<td>+0.63%</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>↓</td>
<td>JavaScript</td>
<td>3.063%</td>
<td>+0.59%</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>^</td>
<td>PHP</td>
<td>2.442%</td>
<td>+0.85%</td>
</tr>
<tr>
<td>9</td>
<td>-</td>
<td>^</td>
<td>SQL</td>
<td>2.184%</td>
<td>+2.18%</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>^</td>
<td>Objective-C</td>
<td>1.477%</td>
<td>-0.02%</td>
</tr>
<tr>
<td>11</td>
<td>16</td>
<td>^</td>
<td>Delphi/Object Pascal</td>
<td>1.396%</td>
<td>+0.00%</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>^</td>
<td>Assembly language</td>
<td>1.371%</td>
<td>-0.10%</td>
</tr>
<tr>
<td>13</td>
<td>10</td>
<td>↓</td>
<td>MATLAB</td>
<td>1.283%</td>
<td>-0.29%</td>
</tr>
<tr>
<td>14</td>
<td>11</td>
<td>↓</td>
<td>Swift</td>
<td>1.220%</td>
<td>-0.35%</td>
</tr>
<tr>
<td>15</td>
<td>17</td>
<td>^</td>
<td>Go</td>
<td>1.189%</td>
<td>-0.20%</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>↓</td>
<td>R</td>
<td>1.111%</td>
<td>-0.80%</td>
</tr>
<tr>
<td>17</td>
<td>15</td>
<td>↓</td>
<td>Ruby</td>
<td>1.109%</td>
<td>-0.32%</td>
</tr>
<tr>
<td>18</td>
<td>14</td>
<td>↓</td>
<td>Perl</td>
<td>1.013%</td>
<td>-0.42%</td>
</tr>
<tr>
<td>19</td>
<td>20</td>
<td>^</td>
<td>Visual Basic</td>
<td>0.979%</td>
<td>-0.37%</td>
</tr>
<tr>
<td>20</td>
<td>19</td>
<td>↓</td>
<td>PL/SQL</td>
<td>0.844%</td>
<td>-0.52%</td>
</tr>
</tbody>
</table>

[https://www.tiobe.com/tiobe-index/](https://www.tiobe.com/tiobe-index/)
Tiobe Index: Dec, 2018

TIOBE Programming Community Index
Source: www.tiobe.com

https://www.tiobe.com/tiobe-index/
Limited research done on why developers do what they do

Understand why and how developers make programming language choices, and how these choices impact their work
Type checking increased productivity, reduced the number of interface defects, & reduced time defects remained throughout development. *Prechelt and Tichy, 1998.*

Increase in dynamic languages rather than static ones. *Paulson, 2007*

Study compared Purity typed and dynamically typed versions. Dynamic faster for both development time and error-fixing. *Hanenberg, 2010.*

Code completion had a small effect on programming speed, while there was a significant speed difference between TypeScript (in favour of) and JavaScript. *Fischer and Hanenberg, 2015.*

Found that Java was 50% faster than Groovy due to less time spent on fixing type errors. *Hanenberg et al., 2014 & 2016.*

Factors to adoption of programming languages. *Meyerovich & Rabkin 2013*

GitHub code quality - static typing is better than dynamic typing and strong typing better than weak. *Ray et al., 2014*

More programmers turning to dynamic languages. *Pypl.github.io, 2017*
Investigate why developers opt to use *dynamic* or *static* languages (vice versa) and how they make these choices
Grounded Theory

- Gather participants through industry and university contacts
- Interview participants
- Transcribing the interviews
- Analysing the data for codes
- Aggregating these codes into concepts and categories
- Form the theory from these concepts and categories

Hoda et al. 2011
## Participants

<table>
<thead>
<tr>
<th>ID</th>
<th>Role</th>
<th>Experience</th>
<th>Organization</th>
<th>Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Graduate</td>
<td>1</td>
<td>Government</td>
<td>Java, JavaScript</td>
</tr>
<tr>
<td>2</td>
<td>Graduate</td>
<td>1</td>
<td>Finance</td>
<td>C#</td>
</tr>
<tr>
<td>3</td>
<td>Graduate</td>
<td>1</td>
<td>Accounting</td>
<td>JavaScript, C#</td>
</tr>
<tr>
<td>4</td>
<td>Graduate</td>
<td>1</td>
<td>Development</td>
<td>Java, Python</td>
</tr>
<tr>
<td>5</td>
<td>PhD Student</td>
<td>4</td>
<td>Energy</td>
<td>Java, Coq</td>
</tr>
<tr>
<td>6</td>
<td>PhD Student</td>
<td>4</td>
<td>Education</td>
<td>JavaScript, Python</td>
</tr>
<tr>
<td>7</td>
<td>Intermediate</td>
<td>&gt;5</td>
<td>Consultancy</td>
<td>C#, JavaScript</td>
</tr>
<tr>
<td>8</td>
<td>PhD Student</td>
<td>1</td>
<td>Education</td>
<td>Python, C++</td>
</tr>
<tr>
<td>9</td>
<td>Senior</td>
<td>&gt;10</td>
<td>Self-Employed</td>
<td>Python</td>
</tr>
<tr>
<td>10</td>
<td>Senior</td>
<td>40</td>
<td>Consultancy</td>
<td>Python</td>
</tr>
<tr>
<td>11</td>
<td>Senior</td>
<td>10</td>
<td>Development</td>
<td>C++, Objective-C</td>
</tr>
<tr>
<td>12</td>
<td>Senior</td>
<td>&gt;10</td>
<td>Development</td>
<td>Java, TypeScript</td>
</tr>
<tr>
<td>13</td>
<td>Graduate</td>
<td>4</td>
<td>Development</td>
<td>JavaScript, TypeScript</td>
</tr>
<tr>
<td>14</td>
<td>Intermediate</td>
<td>&gt;5</td>
<td>Development</td>
<td>Clojure, JavaScript</td>
</tr>
<tr>
<td>15</td>
<td>Intermediate</td>
<td>&gt;5</td>
<td>Development</td>
<td>Clojure, JavaScript</td>
</tr>
</tbody>
</table>
Theory of *Static vs Dynamic* Programming Language Choice

**Attitudes**: preconceptions & biases developers have in regard to static or dynamic languages

**Choice**: thought process developers undergo when selecting a programming language

**Experience**: reflects past experiences a developer has with a language
Theory of *Static vs Dynamic* Language Choice

**Attitudes**

- Static language partisanship
- Developers with more experience prefer static languages
- Switching between static and dynamic languages is not an issue
Developers that primarily use static languages feel strongly about the advantages they offer.

“It gives you a better sense of security in the end that you’ve done something, you can leave it and it’s working. If you need to touch it, the compiler will tell you why. There’s a sense of security once you run the compiler and it tells you it’s ok. With JavaScript, you could have a typo and not notice it for 5 years.” P7
Developers with more experience programming tended to more strongly support the usage of static languages for personal and industry projects rather than dynamic languages.

“Where I have found serious problems is that say I made a typo, dynamic languages don’t tell me anything at all. I don’t find out until I eventually see that the code is not working and then I check that the spelling is wrong. If I had the ability to pre-declare and if I try to reference a member I didn’t declare, it’d immediately throw an exception and tell me to fix it.” P10
Differences between static and dynamic languages, using either for too long may cause developers to forget some of the quirks and idiosyncrasies of other languages and switching to another project maybe difficult.

“You’ll be programming in Java & then switch to Python, add a semi-colon, and think that is not right.” P4
Theory of *Static vs Dynamic* Language Choice

### Attitudes
- Static language partisanship
- Developers with more experience prefer static languages
- Switching between static and dynamic languages is not an issue

### Choice
- Project Languages often pre-selected
- Languages chosen based on familiarity
- Tooling, IDE, and Library support
Choice: Project languages often pre-selected

Choice of programming language was not their responsibility. Languages were pre-selected by lead developer or management.

“It was something that the founder learned and liked. They thought it was good for solving mathematical problems and we’ve used it since.” P14
Lead developers often selected programming languages that they were personally familiar with or felt the majority of workers within teams would be familiar with.

“I don’t think any decisions were made about Python because of syntactical reasons. I think they chose Python because everyone knew it.” P12
**Choice: Tooling, IDE, and Library support**

**Tool support** was a major benefit when selecting a language, due to the options it added. It simply allowed you to do more than an equivalent language without tool support.

“Static languages enables certain tool support that you can’t get otherwise or that requires type inference or runtime tracing or what have you.” P9

“We chose Java because there’s a library for whatever you need to do.” P5
Theory of *Static vs Dynamic* Language Choice

**Attitudes**
- Static language partisanship
- Developers with more experience prefer static languages
- Switching between static and dynamic languages is not an issue

**Choice**
- Project Languages often pre-selected
- Languages chosen based on familiarity
- Tooling, IDE, and Library support

**Experience**

### Static Languages
- **Speed**
  - Static build/compile times slow down large projects
- **Errors**
  - Compilers provide better error checking
  - No types means lower learning curve
- **Structure**
  - Static languages enforce more structure

### Dynamic Languages
- **Speed**
  - Dynamic languages good for quick starts
- **Errors**
  - No types means lower learning curve
- **Structure**
  - Dynamic Languages provide more flexibility
Experience, **Speed**: Static – build/compile times slow larger projects down

**Speed** of build and compile times when using static languages could become cumbersome and have a negative impact on development.

“There’s thousands of modules now in the project and TypeScript has to compile and it’s really slow. We’re often running out of memory in some cases, which is a real problem for us.” P12
Experience, **Speed**: Dynamic - good for smaller projects and quick starts

**Speed** of dynamic languages were suited for projects that were small in scale or needed a prototype running quickly, several mentioned Python and JavaScript as being two languages.

“Dynamic languages are great for small hacky things.” P9

“The setup was super fast. You just have the command line interface, the node package manager and it all just goes. The overall setup did contribute to the project.” P1
Experience, **Errors**: Static – provide better error checking

Type checking and the presence of a compiler generally meant that they provided **better error checking** for programs.

“A lot of errors don’t show up until they actually happen in JavaScript, C# is a lot clearer since the compiler will tell you if there is an error.” P2

“The times I’ve dealt with JavaScript, it hasn’t been good. It’s really not clear what types the inputs are and what the outputs are.” P10

“Looking through other people’s code to see what’s happening is a lot more difficult, especially when one person breaks one thing and find out where the break is being caused. It’s even worse there’s more people working on it. Using a static language might have reduced this.” P1
Dynamic languages tended to be easier to learn for: novice programmers, learning a new language, or who had just joined industry. Not declaring variables allowed more work with less effort, minimized overhead by having to think less.

“I program faster without types, which are obstructive to my thought process of continuing to design something. It may be because I design things as I go, rather than planning them out.” P11

… while new users of static languages believed that there was more of a gap before they could get something working.
Usage of static languages enforced structure throughout software development. Due to declaring the types of variables meant that forethought had to be put into envisaging how the code would look before entering it.

“Once we had it up and running and we could show them how everything was organised. In the end, code quality and organisation of code [using Java] was much higher than the JavaScript project we also had running.” P7
Dynamic languages allowed developers to have more flexibility in the development process, without being constrained by type declaration and other enforced structures that arise from static programming languages. Ignoring typing allowed more time thinking about how to solve problems rather than getting the structure and typing right.

“With JavaScript, you can do whatever you want. If you’re using Java, you have to adhere to the rules.” P1
Theory of *Static vs Dynamic* Language Choice

- **Attitudes**
  - **Informs**
  - **Shapes**
- **Experience**
  - **Provides**
- **Choice**
Represents the **choices** that a developer makes provides them with **experience** in the future.

- Project languages often being pre-selected
- Languages chosen based on familiarity
- Tooling, IDE, and library support
Experience Shapes Attitudes

Represents where a developer’s previous experience with a programming language then shapes their preconceptions and attitudes towards that language.

• Speed
• Errors
• Structure
Represents attitudes that developers have regarding certain languages and types of languages are significant in the choice of language.

- Static language partisanship
- Developers with more experience tend to prefer static languages
- Switching between static and dynamic languages was not an issue
Limitations:
• Small sample size
• Focused on one geographical region
• Subjective opinions

Future Work:
• Interview more developers
• Conduct online surveys
• Considering other languages aspects (beyond types systems)
Attitudes: preconceptions & biases developers have in regard to static or dynamic languages

Choice: thought process developers undergo when selecting a programming language

Experience: reflects past experiences a developer has with a language