

A wide-angle photograph of a canyon landscape. On the left, a massive, layered red rock cliff rises steeply. To the right, another large cliff is visible across the water. The river in the foreground is very still, creating a perfect mirror reflection of the surrounding rock formations and the sky above.

Why Static Typing Came Back

@rtfeldman

Quick Access | Java EE | Java

Package Explorer

- + esptest
- + stockticker
 - src/main/java
 - com.espertech.esper.example.stockticker
 - + StockTickerEventGenerator.java
 - + StockTickerMain.java
 - StockTickerSamplePlugin.java
 - com.espertech.esper.example.stockticker.event
 - com.espertech.esper.example.stockticker.monit
- + JRE System Library [jdk1.8.0_60]
- + etc
- + src
 - main
 - test
- + target
- pom.xml

StockTickerMain.java

```

1 package com.espertech.esper.example.stockticker;
2
3 import com.espertech.esper.client.EPServiceProviderManager;
4
5 public class StockTickerMain implements Runnable
6 {
7     private static final Log log = LoggerFactory.getLog(StockTickerMain.class);
8
9     private final String engineURI;
10    private final boolean continuousSimulation;
11
12    public static void main(String[] args)
13    {
14        new StockTickerMain("StockTicker", false).run();
15    }
16
17    public StockTickerMain(String engineURI, boolean continuousSimulation) {
18        this.engineURI = engineURI;
19        this.continuousSimulation = continuousSimulation;
20    }
21
22    public void run() {
23
24        Configuration configuration = new Configuration();
25        configuration.addEventType("PriceLimit", PriceLimit.class.getName());
26
27        EPServiceProviderManager manager = EPServiceProviderManager.getInstance();
28        manager.setConfiguration(configuration);
29
30        manager.start();
31
32        while (true) {
33            Thread.sleep(1000);
34
35            if (continuousSimulation) {
36                log.info("Continuous simulation mode enabled. Ticker will run indefinitely.");
37            } else {
38                log.info("Simulation mode disabled. Ticker will run until interrupted.");
39            }
40
41            manager.getEPService().process();
42
43            if (!continuousSimulation) {
44                break;
45            }
46        }
47
48        manager.stop();
49    }
50
51    public static void main(String[] args) throws Exception {
52        new StockTickerMain("StockTicker", false).run();
53    }
54
55    public static void main(String[] args, Properties properties) throws Exception {
56        new StockTickerMain("StockTicker", false).run();
57    }
58
59    public static void main(String[] args, Properties properties, Configuration configuration) throws Exception {
60        new StockTickerMain("StockTicker", false).run();
61    }
62
63    public static void main(String[] args, Properties properties, Configuration configuration, EPServiceProviderManager manager) throws Exception {
64        new StockTickerMain("StockTicker", false).run();
65    }
66
67    public static void main(String[] args, Properties properties, Configuration configuration, EPServiceProviderManager manager, boolean continuousSimulation) throws Exception {
68        new StockTickerMain("StockTicker", continuousSimulation).run();
69    }
70
71    public static void main(String[] args, Properties properties, Configuration configuration, EPServiceProviderManager manager, boolean continuousSimulation, boolean logToFile) throws Exception {
72        new StockTickerMain("StockTicker", continuousSimulation).run();
73    }
74
75    public static void main(String[] args, Properties properties, Configuration configuration, EPServiceProviderManager manager, boolean continuousSimulation, boolean logToFile, String logFile) throws Exception {
76        new StockTickerMain("StockTicker", continuousSimulation).run();
77    }
78
79    public static void main(String[] args, Properties properties, Configuration configuration, EPServiceProviderManager manager, boolean continuousSimulation, boolean logToFile, String logFile, String engineURI) throws Exception {
80        new StockTickerMain("StockTicker", continuousSimulation).run();
81    }
82
83    public static void main(String[] args, Properties properties, Configuration configuration, EPServiceProviderManager manager, boolean continuousSimulation, boolean logToFile, String logFile, String engineURI) throws Exception {
84        new StockTickerMain("StockTicker", continuousSimulation).run();
85    }
86
87    public static void main(String[] args, Properties properties, Configuration configuration, EPServiceProviderManager manager, boolean continuousSimulation, boolean logToFile, String logFile, String engineURI, String logLevel) throws Exception {
88        new StockTickerMain("StockTicker", continuousSimulation).run();
89    }
90
91    public static void main(String[] args, Properties properties, Configuration configuration, EPServiceProviderManager manager, boolean continuousSimulation, boolean logToFile, String logFile, String engineURI, String logLevel) throws Exception {
92        new StockTickerMain("StockTicker", continuousSimulation).run();
93    }
94
95    public static void main(String[] args, Properties properties, Configuration configuration, EPServiceProviderManager manager, boolean continuousSimulation, boolean logToFile, String logFile, String engineURI, String logLevel, String logFormat) throws Exception {
96        new StockTickerMain("StockTicker", continuousSimulation).run();
97    }
98
99    public static void main(String[] args, Properties properties, Configuration configuration, EPServiceProviderManager manager, boolean continuousSimulation, boolean logToFile, String logFile, String engineURI, String logLevel, String logFormat) throws Exception {
100       new StockTickerMain("StockTicker", continuousSimulation).run();
101   }
102 }
```

Task List

Find All Activate...

Outline

com.espertech.esper.example.stockticker StockTickerMain

- + log : Log
- + engineURI : String

Problems

105 errors, 3 warnings, 0 others (Filter matched 103 of 108 items)

Description	Resource	Path	Location	Type
+ Errors (100 of 105 items)				
+ Warnings (3 items)				

FileInfo.com Project < >

css

- controls.css
- custom.css
- frame.css
- widgets.css

embedding.html

empty.html

img

index.html

js

- custom.js
- menu.js
- widgets.js

LICENSE

menu.html

README.md

vid

widgets.html

widgets.js — js

```
        ,
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282

    if (typeof settings["parent"] === "undefined") {
      dialog_settings["position"] = {
        my: "center",
        at: "center",
        of: window
      };
    } else {
      dialog_settings["appendTo"] = settings["parent"];
      dialog_settings["position"] = {
        my: "center",
        at: "center",
        of: settings["parent"]
      };
    }
    var $dialog = $selector.dialog(dialog_settings);
    $selector_container = $selector.closest(".ui-dialog");
    $selector_container.find(".ui-dialog-titlebar-close").empty().append("<i class='fa fa-times fa-lg'></i>");
    if (!show_close_button) {
      $dialog.on("dialogopen", function () {
        $(this).parent().find(".ui-dialog-titlebar-close").hide();
      });
    }
    return $dialog;
}
this.createCustomDialog = createCustomDialog;

function setCustomDropdown($ui, settings) {
  var items = settings["items"]; // the text that will appear for each item
  var init_index = settings["init_index"];
  var init_text = settings["init_text"];
  var on_item_click_callback = settings["on_item_click_callback"];
  var on_item_create_callback = settings["on_item_create_callback"];
  var $menu = $ui.find("div").empty();
  var $button_text = $ui.find("a > span").text("");
  var $selected_item;
  // Set initial button text
  if (typeof init_text !== "undefined") {
    $button_text.text(init_text);
  } else {
```





Programming Language Rankings for January 2022 (omitting CSS and Shell)

the developer-focused industry analyst firm

Static Typing

C++	TypeScript
Rust	Dart
Go	C#
C	Java
Objective-C	Scala
Swift	Kotlin

No Static Typing

JavaScript
Python
PHP
Ruby
R



Programming Language Rankings: January 2022

the developer-focused industry analyst firm

Static Typing

C++

Rust

Go

C

Objective-C

Swift

TypeScript

Dart

C#

Java

Scala

Kotlin

No Static Typing

JavaScript

Python

PHP

Ruby

R





WHAT HAPPENED?

OUTLINE

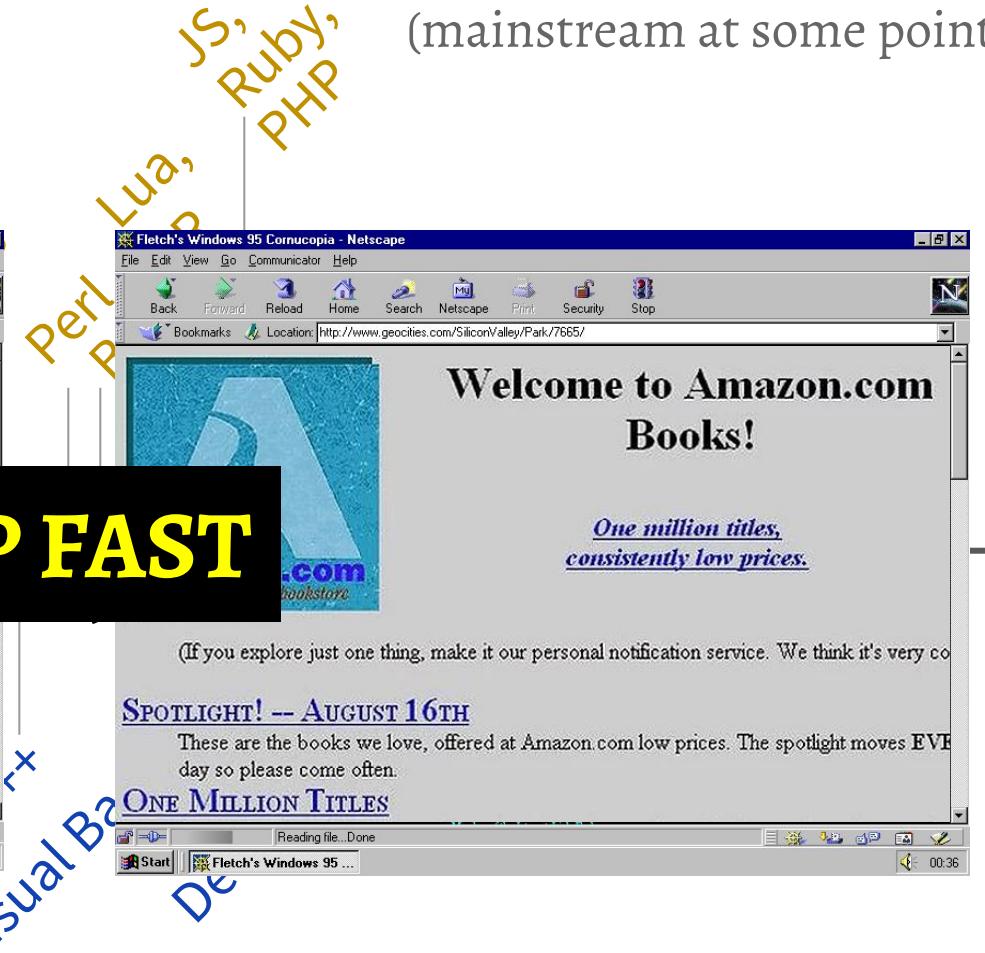
1. What made dynamic typing get big?
2. What changed?
3. What does this mean for the future?

1. What made dynamic typing get big?
2. What changed?
3. What does this mean for the future?

(mainstream at some point)

1950 1960 1970 1980 1990

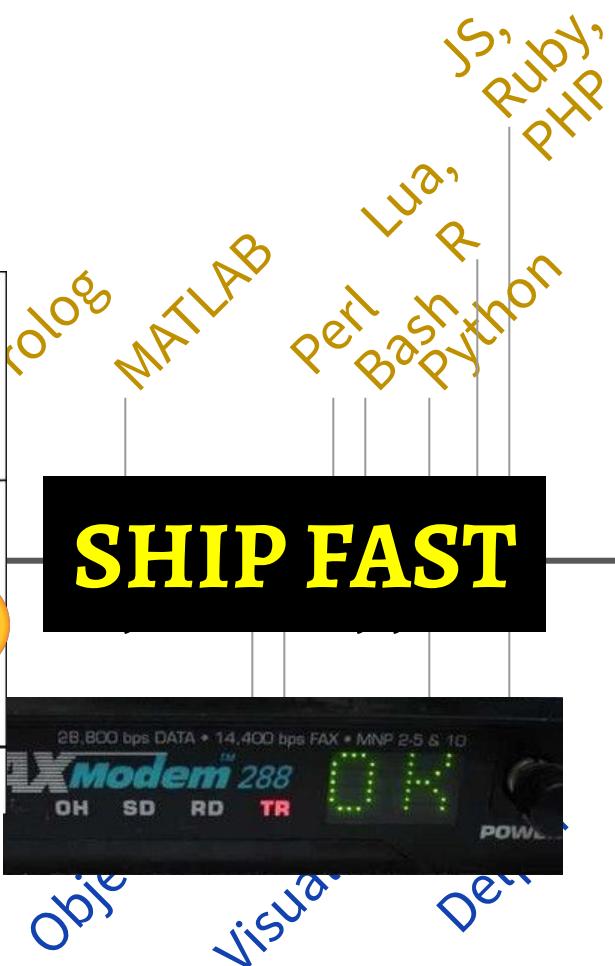
NO STATIC TYPING



STATIC TYPING

NO STATIC TYPING

(mainstream at some point)



xkcd.com/303

STATIC TYPING

fast feedback
concise syntax
relevant sugar

runs faster
IDE features
surfaces errors

NO STATIC TYPING

Assembly

BASIC

Prolog

MATLAB

Lua,
Perl
Bash
R
Python

JS,
Ruby,
PHP

(mainstream at some point)



SHIP FAST

1950

1960

1970

FORTRAN
COBOL

Pascal
C

Objective-C
C++
Visual Basic
Java,
Delphi

STATIC TYPING

NO STATIC TYPING

Assembly

BASIC

Prolog

MATLAB

Perl
Lua,
Bash
R
Python

JS,
Ruby,
PHP

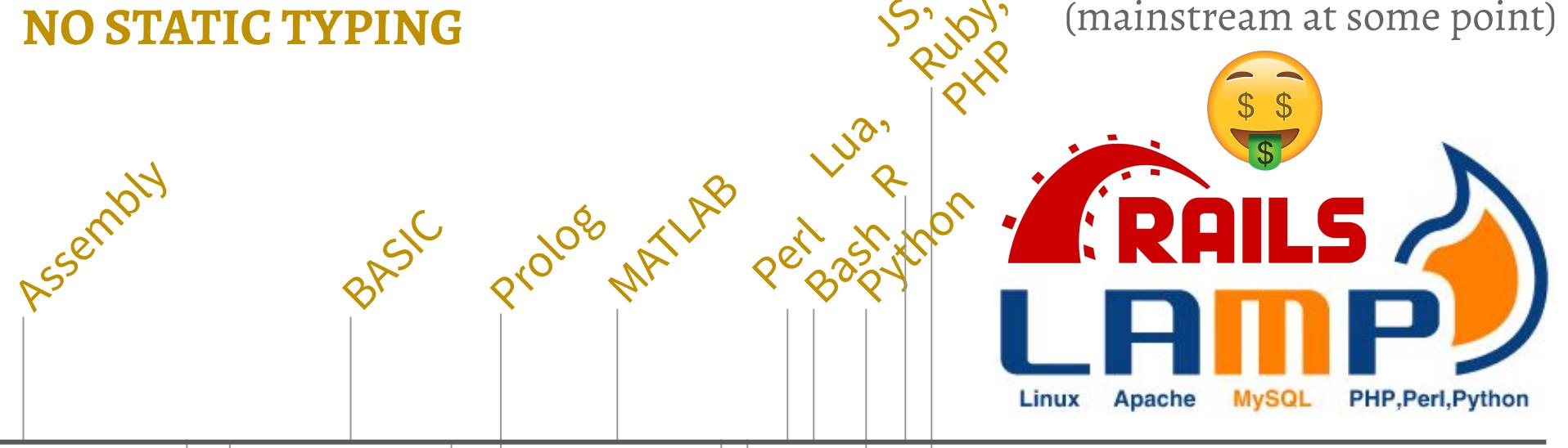
(mainstream at some point)



The popular **1990s-era dynamic languages**
had **better tradeoffs** for early Web use cases
than **that era's** popular statically-typed languages.

STATIC TYPING

NO STATIC TYPING



The popular **1990s-era dynamic languages** had **better tradeoffs** for early Web use cases than **that era's** popular statically-typed languages. The Web exploded in popularity, so they did too.

NO STATIC TYPING

(mainstream at some point)

Assembly

BASIC

Prolog

MATLAB

Lua,
Perl
Bash

JS,
Ruby,
PHP

R
Python

PowerShell
CoffeeScript

1950

1960

1970

1980

1990

2000

2010

2020

FORTRAN
COBOL

Pascal
C

Objective-C
C++

Visual Basic
Java,

Delphi
ActionScript
C#

Dart,
TypeScript
Go
Rust
Kotlin
Swift

STATIC TYPING

1. What made dynamic typing get big?
2. What changed?
3. What does this mean for the future?

1. What made dynamic typing get big?
2. **What changed?**
3. What does this mean for the future?

My view of static typing

~10 years ago:

1. lots of ceremony
2. not all that helpful
3. slow feedback loops

How have these changed in the past 10 years?

1. lots of ceremony
2. not all that helpful
3. slow feedback loops

My view:

“Static typing has lots of ceremony.”

```
function decode(rawJson) {
    var answer = JSON.parse(rawJson)

    if (typeof answer !== "object") {
        return "JSON decoding failed!"
    } else if (answer.admin === true) {
        return {
            displayName: answer.name + " (Admin)",
            isEmpty: answer.name.length === 0
        };
    } else {
        return {
            displayName: answer.name,
            isEmpty: answer.name.length === 0
        };
    }
}
```



JavaScript™

object with 2 fields

```

class Answer {
    private boolean wasEmpty;
    private String displayName;

    public Answer(boolean wasEmpty, String displayName) {
        this.wasEmpty = wasEmpty;
        this.displayName = displayName; constructor
    }

    public boolean getWasEmpty() {
        return this.wasEmpty;
    }

    public void setWasEmpty(boolean wasEmpty) {
        this.wasEmpty = wasEmpty;
    }

    public String getDisplayName() {
        return this.displayName;
    }

    public void setDisplayName(String displayName) {
        this.displayName = displayName; setter
    }

    @Override
    public boolean equals(final Object obj) {
        if(obj instanceof Answer) {
            final Answer other = (Answer) obj;
            return new EqualsBuilder()
                .append(wasEmpty, other.wasEmpty)
                .append(displayName, other.displayName)
                .isEquals();
        } else {
            return false;
        }
    }

    public int hashCode() {
        final HashCodeBuilder builder =
            new HashCodeBuilder()
                .append(wasEmpty)
                .append(trimmed());
        return builder.toHashCode();
    }
}

```

getter

setter

getter

setter

getters

hashCode

schema

```

@JsonIgnoreProperties(ignoreUnknown = true)
public static class Json {
    @JsonProperty("name")
    public String name;

    @JsonProperty("wasEmpty")
    public boolean isEmpty();
}

public Answer decode(final String rawJson) {
    ObjectMapper mapper = new ObjectMapper();
    try {
        Json json = (Json) mapper.readValue(rawJson, Json.class);

        if (json.isEmpty()) {
            return Answer(json.name.isEmpty(), json.name + " (Admin)");
        } else {
            return Answer(json.name.isEmpty(), json.name);
        }
    } catch (JsonProcessingException e) {
        return null;
    } catch (JsonMappingException e) {
        return null;
    }
}

```

actual business logic

DYNAMIC TYPING

```

function decode(rawJson) {
    var answer = JSON.parse(rawJson)

    if (typeof answer !== "object") {
        return "JSON decode failed!";
    } else if (answer.name === true) {
        return {
            displayName: answer.name + " (Admin)",
            wasEmpty: answer.name.length === 0
        };
    } else {
        return {
            displayName: answer.name,
            wasEmpty: answer.name.length === 0
        };
    }
}

```





How to build a blog engine in 15 minutes with Ruby on Rails

<http://www.rubyonrails.org>

By David Heinemeier Hansson,
originally prepared for the FISL 6.0 conference in Brazil 2005

youtu.be/Gzi723IkRJY

```
function decode(rawJson) {
    var answer = JSON.parse(rawJson)

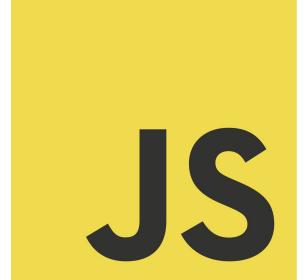
    if (typeof answer !== "object") {
        return "JSON decoding failed!"
    } else if (answer.admin === true) {
        return {
            displayName: answer.name + " (Admin)",
            isEmpty: answer.name.length === 0
        };
    } else {
        return {
            displayName: answer.name,
            isEmpty: answer.name.length === 0
        };
    }
}
```



JavaScript™

```
const decode = (rawJson) => {
  const answer = JSON.parse(rawJson)

  if (typeof answer !== "object") {
    return "JSON decoding failed!"
  } else if (answer.admin === true) {
    return {
      displayName: `${answer.name} (Admin)`,
      wasEmpty: answer.name.length === 0
    };
  } else {
    return {
      displayName: answer.name,
      wasEmpty: answer.name.length === 0
    };
  }
}
```



JS

```
const decode = (rawJson: string) => {
  const answer = JSON.parse(rawJson)

  if (typeof answer !== "object") {
    return "JSON decoding failed!"
  } else if (answer.admin === true) {
    return {
      displayName: `${answer.name} (Admin)`,
      wasEmpty: answer.name.length === 0
    };
  } else {
    return {
      displayName: answer.name,
      wasEmpty: answer.name.length === 0
    };
  }
}
```



```
const decode = (rawJson: string):  
    string | { displayName: string, wasEmpty: boolean } =>  
{  
    const answer = JSON.parse(rawJson)  
  
    if (typeof answer !== "object") {  
        return "JSON decoding failed!"  
    } else if (answer.admin === true) {  
        return {  
            displayName: `${answer.name} (Admin)`,  
            wasEmpty: answer.name.length === 0  
        };  
    } else {  
        return {  
            displayName: answer.name,  
            wasEmpty: answer.name.length === 0  
        };  
    }  
}
```



STATIC TYPING

```
class Answer {  
    private boolean wasEmpty;  
    private String displayName;  
  
    public Answer(boolean wasEmpty, String displayName) {  
        this.wasEmpty = wasEmpty;  
        this.displayName = displayName;  
    }  
  
    public boolean getWasEmpty() {  
        return this.wasEmpty;  
    }  
  
    public void setWasEmpty(boolean wasEmpty) {  
        this.wasEmpty = wasEmpty;  
    }  
  
    public String getDisplayName() {  
        return this.displayName;  
    }  
  
    public void setDisplayName(String displayName) {  
        this.displayName = displayName;  
    }  
  
    @Override  
    public boolean equals(final Object obj) {  
        if(obj instanceof Answer){  
            final Answer other = (Answer) obj;  
  
            return new EqualsBuilder()  
                .append(wasEmpty, other.wasEmpty)  
                .append(displayName, other.displayName)  
                .isEquals();  
        } else {  
            return false;  
        }  
    }  
  
    @Override  
    public int hashCode() {  
        return new HashCodeBuilder()  
            .append(wasEmpty)  
            .append(displayName)  
            .toHashCode();  
    }  
}
```

DYNAMIC TYPING

```
@JsonIgnoreProperties(ignoreUnknown = true)  
public static class Json {  
    @JsonProperty("name")  
    public String name;  
  
    @JsonProperty("wasEmpty")  
    public boolean isEmpty;  
}  
  
public Answer answerDecoder(String rawJson) {  
    ObjectMapper mapper = new ObjectMapper();  
  
    try {  
        Json json = (Json) mapper.readValue(rawJson, Json.class);  
  
        if (json.isEmpty) {  
            return Answer(json.name.isEmpty(), json.name + " (Admin)");  
        } else {  
            return Answer(json.name.isEmpty(), json.name);  
        }  
    } catch (JsonProcessingException e) {  
        return null;  
    } catch (JsonMappingException e) {  
        return null;  
    }  
}
```

```
const decode = (rawJson) =>  
{  
    const answer = JSON.parse(rawJson)  
  
    if (typeof answer !== 'object') {  
        return "JSON decode failed!"  
    } else if (answer.name === true) {  
        return {  
            displayName: `${answer.name} (Admin)`,  
            wasEmpty: answer.name.length === 0  
        };  
    } else {  
        return {  
            displayName: answer.name,  
            wasEmpty: answer.name.length === 0  
        };  
    }  
}
```



STATIC TYPING

```
class Answer {  
    private boolean wasEmpty;  
    private String displayName;  
  
    public Answer(boolean wasEmpty, String displayName) {  
        this.wasEmpty = wasEmpty;  
        this.displayName = displayName;  
    }  
  
    public boolean getWasEmpty() {  
        return this.wasEmpty;  
    }  
  
    public void setWasEmpty(boolean wasEmpty) {  
        this.wasEmpty = wasEmpty;  
    }  
  
    public String getDisplayName() {  
        return this.displayName;  
    }  
  
    public void setDisplayName(String displayName) {  
        this.displayName = displayName;  
    }  
  
    @Override  
    public boolean equals(final Object obj) {  
        if(obj instanceof Answer){  
            final Answer other = (Answer) obj;  
  
            return new EqualsBuilder()  
                .append(wasEmpty, other.wasEmpty)  
                .append(displayName, other.displayName)  
                .isEquals();  
        } else {  
            return false;  
        }  
    }  
  
    @Override  
    public int hashCode() {  
        return new HashCodeBuilder()  
            .append(wasEmpty)  
            .append(displayName)  
            .toHashCode();  
    }  
}
```

```
@JsonIgnoreProperties(ignoreUnknown = true)  
public static class Json {  
    @JsonProperty("name")  
    public String name;  
  
    @JsonProperty("admin")  
    public boolean admin;  
}  
  
public Answer answerFromJson(String rawJson) {  
    ObjectMapper mapper = new ObjectMapper();  
  
    try {  
        Json json = (Json) mapper.readValue(rawJson, Json.class);  
  
        if (json.admin) {  
            return Answer(json.name.isEmpty(), json.name + " (Admin)");  
        } else {  
            return Answer(json.name.isEmpty(), json.name);  
        }  
    } catch (JsonProcessingException e) {  
        return null;  
    } catch (JsonMappingException e) {  
        return null;  
    }  
}
```

```
const decode = (rawJson: string):  
    string | { displayName: string, wasEmpty: boolean } =>  
{  
    const answer = JSON.parse(rawJson)  
  
    if (typeof answer !== "object") {  
        return "JSON decode failed!"  
    } else if (answer.admin === true) {  
        return {  
            displayName: `${answer.name} (Admin)`,  
            wasEmpty: answer.name.length === 0  
        };  
    } else {  
        return {  
            displayName: answer.name,  
            wasEmpty: answer.name.length === 0  
        };  
    }  
}
```



```

class Answer {
    private boolean wasEmpty;
    private String displayName;

    public Answer(boolean wasEmpty, String displayName) {
        this.wasEmpty = wasEmpty;
        this.displayName = displayName;
    }

    @JsonIgnoreProperties(ignoreUnknown = true)
    public boolean isEmpty() {
        return wasEmpty;
    }

    public void setEmpty(boolean isEmpty) {
        this.wasEmpty = isEmpty;
    }

    public String getDisplayName() {
        return displayName;
    }

    public void setDisplayName(String displayName) {
        this.displayName = displayName;
    }

    @Override
    public boolean equals(final Object obj) {
        if(obj instanceof Answer){
            final Answer other = (Answer) obj;

            return new EqualsBuilder()
                .append(wasEmpty, other.wasEmpty)
                .append(displayName, other.displayName)
                .isEquals();
        } else {
            return false;
        }
    }

    @Override
    public int hashCode() {
        return newHashCodeBuilder()
            .append(wasEmpty)
            .append(trimmed);
    }
}

```

Advantages

nice equals and hashCode implementations

`{ x: 1, y: 2 } == { x: 1, y: 2 }`

// **false** in JavaScript/TypeScript

```

const decode = (rawJson: string):
    string | { displayName: string, wasEmpty: boolean } =>
{
    const answer = JSON.parse(rawJson)

    if (typeof answer !== "object") {
        return "JSON decoding failed!"
    } else if (answer.admin === true) {
        return {
            displayName: `${answer.name} (Admin)`,
            wasEmpty: answer.name.length === 0
        };
    } else {
        return {
            displayName: answer.name,
            wasEmpty: answer.name.length === 0
        };
    }
}

```



```
class Answer {  
    private boolean wasEmpty;  
    private String displayName;  
  
    public Answer(boolean wasEmpty, String displayName) {  
        this.wasEmpty = wasEmpty;  
        this.displayName = displayName;  
    }  
  
    @JsonIgnoreProperties(ignoreUnknown = true)  
    public boolean isEmpty() {  
        return wasEmpty;  
    }  
  
    public void setEmpty(boolean wasEmpty) {  
        this.wasEmpty = wasEmpty;  
    }  
  
    public String getDisplayName() {  
        return displayName;  
    }  
  
    public void setDisplayName(String displayName) {  
        this.displayName = displayName;  
    }  
  
    @Override  
    public boolean equals(final Object obj) {  
        if(obj instanceof Answer) {  
            final Answer other = (Answer) obj;  
  
            return new EqualsBuilder()  
                .append(wasEmpty, other.wasEmpty)  
                .append(displayName, other.displayName)  
                .isEquals();  
        } else {  
            return false;  
        }  
    }  
  
    @Override  
    public int hashCode() {  
        return new HashCodeBuilder()  
            .append(wasEmpty)  
            .append(trimmed())  
            .append(displayName)  
            .isHashCode();  
    }  
}
```

Advantages

nice equals and hashCode implementations

early JSON validation instead of crashing later

answer.name.length

Uncaught TypeError:

Cannot read properties of undefined
(reading 'length')

```
} else {  
    return {  
        displayName: answer.name,  
        wasEmpty: answer.name.length === 0  
    };  
}  
}
```



TS

```

class Answer {
    private boolean wasEmpty;
    private String displayName;

    public Answer(boolean wasEmpty, String displayName) {
        this.wasEmpty = wasEmpty;
        this.displayName = displayName;
    }

    @JsonIgnoreProperties(ignoreUnknown = true)
    public boolean isEmpty() {
        return wasEmpty;
    }

    public void setEmpty(boolean empty) {
        this.wasEmpty = empty;
    }

    public String getDisplayName() {
        return displayName;
    }

    public void setDisplayName(String name) {
        this.displayName = name;
    }

    @Override
    public boolean equals(final Object obj) {
        if(obj instanceof Answer){
            final Answer other = (Answer) obj;

            return new EqualsBuilder()
                .append(wasEmpty, other.wasEmpty)
                .append(displayName, other.displayName)
                .isEquals();
        } else {
            return false;
        }
    }

    @Override
    public int hashCode() {
        return new HashCodeBuilder()
            .append(wasEmpty)
            .append(trimmed);
    }
}

```

Advantages

nice equals and hashCode implementations

early JSON validation instead of crashing later

~~getters and setters have been helpful to me~~ 😂



```

const decode = (rawJson: string):
  string | { displayName: string, wasEmpty: boolean } =>
{
  const answer = JSON.parse(rawJson)

  if (typeof answer !== "object") {
    return "JSON decoding failed!"
  } else if (answer.admin === true) {
    return {
      displayName: `${answer.name} (Admin)`,
      wasEmpty: answer.name.length === 0
    };
  } else {
    return {
      displayName: answer.name,
      wasEmpty: answer.name.length === 0
    };
  }
}

```



```

class Answer {
    private boolean wasEmpty;
    private String displayName;

    public Answer(boolean wasEmpty, String displayName) {
        this.wasEmpty = wasEmpty;
        this.displayName = displayName;
    }

    public boolean isEmpty() {
        return wasEmpty;
    }

    public void setEmpty(boolean isEmpty) {
        this.wasEmpty = isEmpty;
    }

    public String getDisplayName() {
        return displayName;
    }

    public void setDisplayName(String displayName) {
        this.displayName = displayName;
    }

    @Override
    public boolean equals(final Object obj) {
        if(obj instanceof Answer){
            final Answer other = (Answer) obj;

            return new EqualsBuilder()
                .append(wasEmpty, other.wasEmpty)
                .append(displayName, other.displayName)
                .isEquals();
        } else {
            return false;
        }
    }

    @Override
    public int hashCode() {
        return newHashCodeBuilder()
            .append(wasEmpty)
            .append(trimmed);
    }
}

    @JsonIgnoreProperties(ignoreUnknown = true)

```

Advantages

nice equals and hashCode implementations

early JSON validation instead of crashing later



```

        }
        catch (JsonProcessingException e) {
            return null;
        }
        catch (JsonMappingException e) {
            return null;
        }
    }

const decode = (rawJson: string):
    string | { displayName: string, wasEmpty: boolean } =>
{
    const answer = JSON.parse(rawJson)

    if (typeof answer !== "object") {
        return "JSON decoding failed!";
    } else if (answer.admin === true) {
        return {
            displayName: `${answer.name} (Admin)`,
            wasEmpty: answer.name.length === 0
        };
    } else {
        return {
            displayName: answer.name,
            wasEmpty: answer.name.length === 0
        };
    }
}

```





early JSON validation instead of crashing later



nice equals and hashCode implementations



roc-lang.org

```
decode = \rawJson ->
  when Decode.fromBytes rawJson Json.fromUtf8 is
    Ok { name, admin: True } ->
      Ok {
        wasEmpty: Str.isEmpty name,
        displayName: `"(name) Admin",
      }

    Ok { name, admin: False } ->
      Ok {
        wasEmpty: Str.isEmpty name,
        displayName: name,
      }

  Err _ -> Err "JSON decoding failed!"
```

```
const decode = (rawJson: string):
  string | { displayName: string, wasEmpty: boolean } =>
{
  const answer = JSON.parse(rawJson)

  if (typeof answer !== "object") {
    return "JSON decoding failed!"
  } else if (answer.admin === true) {
    return {
      displayName: `${answer.name} (Admin)`,
      wasEmpty: answer.name.length === 0
    };
  } else {
    return {
      displayName: answer.name,
      wasEmpty: answer.name.length === 0
    };
  }
}
```

TS



early JSON validation instead of crashing later



nice equals and hashCode implementations



roc-lang.org

```
decode : Str -> { displayName : Str, wasEmpty : Bool }
decode = \rawJson ->
  when Decode.fromBytes rawJson Json.fromUtf8 is
    Ok { name, admin: True } ->
      Ok {
        wasEmpty: Str.isEmpty name,
        displayName: `"(name) Admin",
      }

    Ok { name, admin: False } ->
      Ok {
        wasEmpty: Str.isEmpty name,
        displayName: name,
      }

  Err _ -> Err "JSON decoding failed!"
```

```
const decode = (rawJson: string):
  string | { displayName: string, wasEmpty: boolean } =>
{
  const answer = JSON.parse(rawJson)

  if (typeof answer !== "object") {
    return "JSON decoding failed!"
  } else if (answer.admin === true) {
    return {
      displayName: `${answer.name} (Admin)`,
      wasEmpty: answer.name.length === 0
    };
  } else {
    return {
      displayName: answer.name,
      wasEmpty: answer.name.length === 0
    };
  }
}
```

TS

My view:

“Static typing has lots of ceremony.”

“Some languages and community norms
encourage ceremony. **They don’t have to.**”

My view:

“Static typing is not all that helpful.”



Problem Occurred



'Publishing to JBoss 7.1 Runtime Server...' has
encountered a problem.

Could not publish to the server.

OK

<< Details

Could not publish to the server.

java.lang.NullPointerException

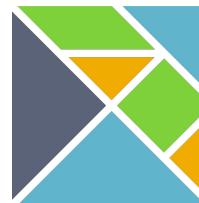
NullPointerException

null-aware static type checker

language doesn't have null at all



nil/null/undefined errors in production



thousands reached production,
after thousands of **tests passed**

100% **caught at build time**,
never reached production



mcc

@mcclure111

In C++ we don't say "Missing asterisk"



elm-lang.org

```
-- TYPE MISMATCH ----- tmp.elm
```

The 1st argument to function 'join' is causing a mismatch.

```
5| String.join 4 ["Alice", "Bob"]  
   ^
```

Function 'join' is expecting the 1st argument to be:

String

But it is:

number

```
<script>
    var grid = new dhtmlXGridObject();
    grid.set
</script>
```

- ≡ ♦ selectRow
- ≡ ♦ selectRowById
- ≡ ♦ serialize
- ≡ ♦ serializeToCSV
- ≡ ♦ setAwaitedRowHeight
- ≡ ♦ setCellExcelType
- ≡ ♦ setCellStyle
- ≡ ♦ setCheckedRows
- ≡ ♦ setColAlign

set height which will be used in smart r
grid which changes default row height

helpful...but feels **painfully laggy** to use

The screenshot shows a Textmate interface with three tabs open: 'CONTRIBUTING.md', 'OakTextView.mm', and 'scope.cc'. The 'OakTextView.mm' tab is active, displaying Objective-C++ code for a text view. The code includes methods like `begin_change_grouping`, `end_change_grouping`, `font`, `set_font`, and `font_scale_factor`. The sidebar on the right shows a file tree with 'Uncommitted Changes' containing 'CONTRIBUTING.md' and 'OakTextView.mm', and 'Untracked Items' containing 'duff.ninja', 'format_string.cc', 'Notes.md', and 'TODO.md'. The bottom status bar shows the line number (247:38), language (Objective-C++), tab size (3), and other editor settings.

```
CONTRIBUTING.md          OakTextView.mm          scope.cc
241 struct document_view_t : ng::buffer_api_t
242 {
243     document_view_t (OakDocument* document, std::string const& scopeAt
244     {
245         _document_editor = [OakDocumentEditor documentEditorWithDocument
246
247         set_scroll_past_end(scrollPastEnd);
248
249         _editor = &[_document_editor editor];
250         _layout = &[_document_editor layout];
251
252         settings_t const settings = settings_for_path(logical_path(), f
253         invisibles_map = settings.get(kSettingsInvisiblesMapKey, "");
254     }
255
256     bool begin_change_grouping () { return [_document_
257     bool end_change_grouping () { return [_document_
258
259     NSFont* font () const { return _document_e
260     void set_font (NSFont* newFont) { _document_editor.f
261
262     CGFloat font_scale_factor () const { return _document_e
263 }
```

less helpful...but feels nice and snappy!

My view:

“Static typing is not all that helpful.”

“Some type checkers can be helpful and friendly.
IDEs can be helpful and snappy. **Neither is a given.**”

My view:

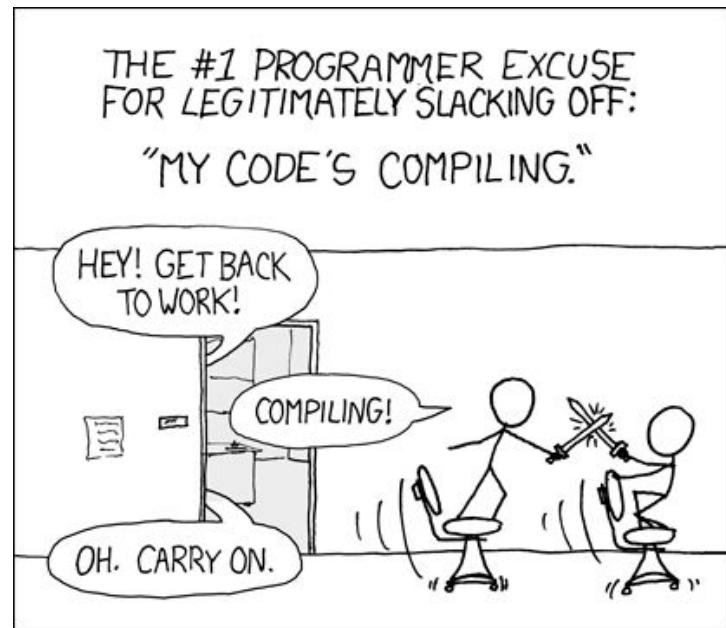
“Static typing has slow feedback loops.”

Ruby

```
irb(main):001:0> n = 5
=> 5
irb(main):002:0> def fact(n)
irb(main):003:1>   if n <= 1
irb(main):004:2>     1
irb(main):005:2>   else
irb(main):006:2*>     n * fact(n - 1)
irb(main):007:2>   end
irb(main):008:1> end
=> nil
irb(main):009:0> fact(n)
=> 120
irb(main):010:0> Dir.entries '/'
=> [".", "..", "sbin", "proc", "bin", "tmp", "med
trd.img.old", "lib", "root", "mnt", "selinux", "v
d", "var", "lib64", "initrd", "boot", ".Trash-0",
sys", "lost+found", "opt", "dev", "lib32", "home"
"cdrom", "srv", "usr"]
irb(main):011:0> []
```

VS.

C++, Java, etc...



REPL

xkcd.com/303

Ruby

```
irb(main):001:0> n = 5
=> 5
irb(main):002:0> def fact(n)
irb(main):003:1>   if n <= 1
irb(main):004:2>     1
irb(main):005:2>   else
irb(main):006:2*>     n * fact(n - 1)
irb(main):007:2>   end
irb(main):008:1> end
=> nil
irb(main):009:0> fact(n)
=> 120
irb(main):010:0> Dir.entries '/'
=> [".", "..", "sbin", "proc", "bin", "tmp", "med
trd.img.old", "lib", "root", "mnt", "selinux", "v
d", "var", "lib64", "initrd", "boot", ".Trash-0",
sys", "lost+found", "opt", "dev", "lib32", "home"
"cdrom", "srv", "usr"]
irb(main):011:0> []
```

REPL

Haskell

```
Prelude > let fb n = if rem n 15 == 0 then "fizzbuzz" else if rem n 3 == 0 then
"fizz" else if rem n 5 == 0 then "buzz" else show n
Prelude > let fb take 100 [ fb x | x <- [1..] ]
Prelude > ["1", "2", "fizz", "4", "buzz", "fizz", "7", "8", "fizz", "buzz", "11", "fizz", "1
3", "14", "fizzbuzz", "16", "17", "fizz", "19", "buzz", "fizz", "22", "23", "fizz", "buzz", "f
26", "fizz", "28", "29", "fizzbuzz", "31", "32", "fizz", "34", "buzz", "fizz", "37", "38", "f
izz", "buzz", "41", "fizz", "43", "44", "fizzbuzz", "46", "47", "fizz", "49", "buzz", "fizz",
"52", "53", "fizz", "buzz", "56", "fizz", "58", "59", "fizzbuzz", "61", "62", "fizz", "64",
"buzz", "fizz", "67", "68", "fizz", "buzz", "71", "fizz", "73", "74", "fizzbuzz", "76", "77",
"fizz", "79", "buzz", "fizz", "82", "83", "fizz", "buzz", "86", "fizz", "88", "89", "fizzbu
zz", "91", "92", "fizz", "94", "buzz", "fizz", "97", "98", "fizz", "buzz"]
```

100% statically typed,
no dynamic types

REPL

Compiler performance improvements

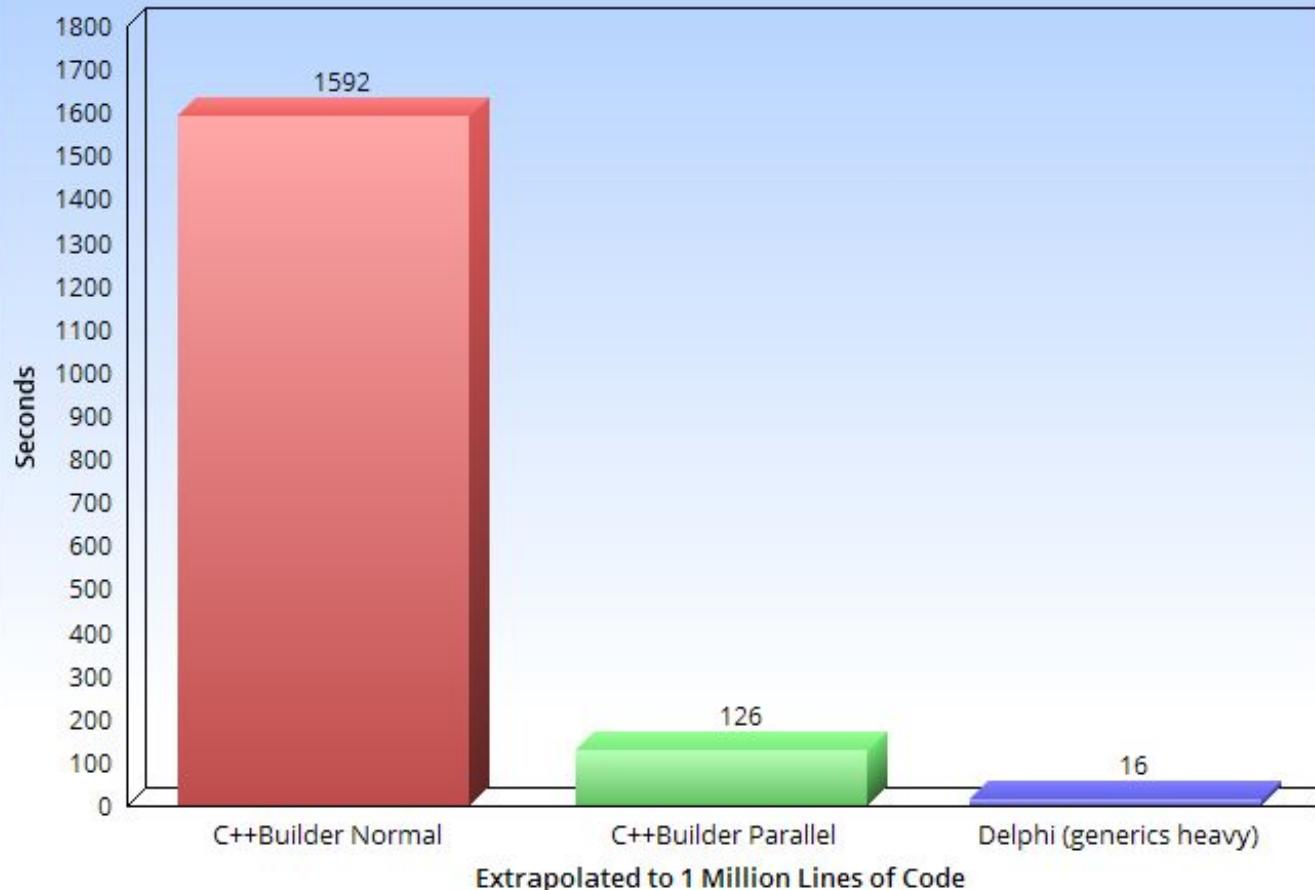
multicore compilation

incremental recompiles

considering build speed in language design

AMD Ryzen 9 5950x

C++Builder: Normal vs. C++Builder Parallel vs. Delphi



TheMillion - RAD Studio 10.1 Berlin - TheMillionUnit [Built]

File Edit Search View Refactor Project Run Component Tools Window Help Default Layout 32-bit Windows Structure TheMillion.dproj - Project Manager

Search

Welcome Page TheMillionUnit

Style: Windows View: Master

THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY CODE'S COMPILING."

HEY! GET BACK TO WORK!

COMPILED!
OH. CARRY ON.

A large red X is drawn across the cartoon panel.

Build

Project: C:\...\Embarcadero\Studio\Projects\TheMillion.dproj
Compiling: Done.
Current line: 0 Total lines: 1000000
Hints: 0 Warnings: 0 Errors: 0

OK

Automatically close on successful compile

LeftToRight (TFormBorder)
[biSystemMenu, biMin]
Border
BorderStyle
Caption
ClientHeight
ClientWidth

Million Lines of Code
345
404

Sizeable
Timer

00 : 00 : 05

C:\Users\jim\Documents\Embarcadero\Studio\Projects\TheMillion.d... Model View Data Explorer Multi-Device ...

Tool Palette

Border
BorderIcons
BorderStyle
Caption
ClientHeight
ClientWidth

Million Lines of Code
345
404

Sizeable
Timer

LeftToRight (TFormBorder)
[biSystemMenu, biMin]

Standard
Additional
System
Dialogs
Data Access
dbExpress

LiveBindinas Designer LiveBindinas Designer Gestures REST Client

JavaScript conferences around 2017

“What do you think of TypeScript?”

“What do you like about it?”

Property 'fullYear' does not exist on type 'Date'

date.fullYear



fast feedback loop

~~blocking compile errors~~

nonblocking error reports

Property 'fullYear' does not exist on type 'Date'

date.fullYear



faster feedback loop

My view:

~~“Static typing has slow feedback loops.”~~

“Compile times can be so fast they **feel instant**,
and IDEs can offer **actually faster** feedback loops.”

How have these changed in the past 10 years?

1. lots of ceremony
2. not all that helpful
3. slow feedback loops

How have these **changed** in the past 10 years?

1. lots of ceremony → **no ceremony** required
2. not all that helpful → way **more helpful**
3. slow feedback loops → **actually faster** feedback loops

1. What made dynamic typing get big?
2. **What changed?**
3. What does this mean for the future?

1. What made dynamic typing get big?
2. What changed?
3. **What does this mean for the future?**

3 hypothetical futures

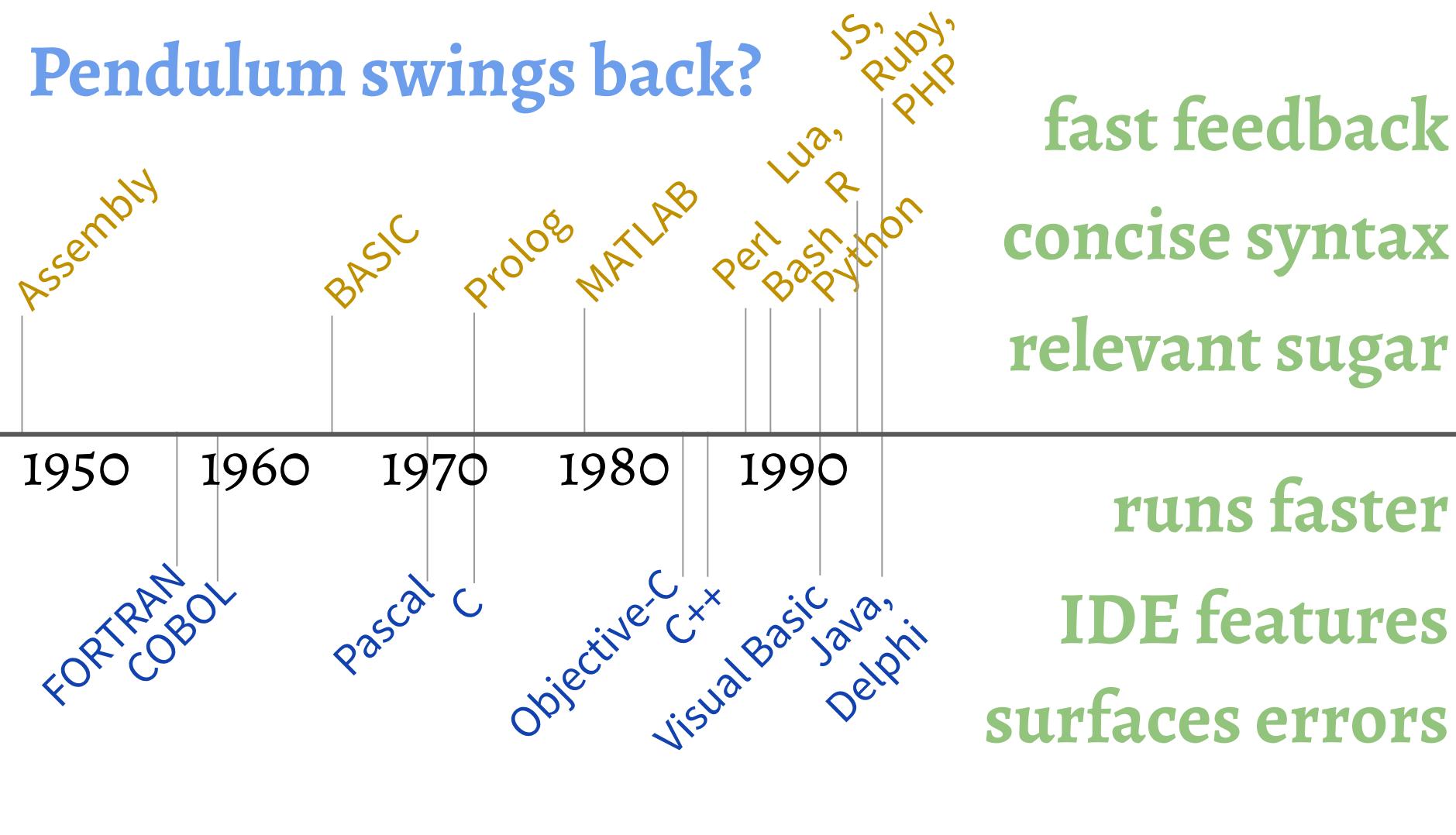
1. The pendulum swings back to dynamic
2. Most languages embrace gradual typing
3. Static without gradual gets more popular

Pendulum swings back?



The popular **1990s-era dynamic languages**
had **better tradeoffs** for early Web use cases
than **that era's** popular statically-typed languages.

Pendulum swings back?



Pendulum swings back?

These benefits don't **require** dynamic typing,

though they **first appeared** in dynamic languages.

A language can have **all of these** and static typing.

Dynamic typing **requires** some runtime overhead.

Some IDE features **require** build-time type info.

Build-time type errors **require** static type analysis.

fast feedback

concise syntax

relevant sugar

runs faster

IDE features

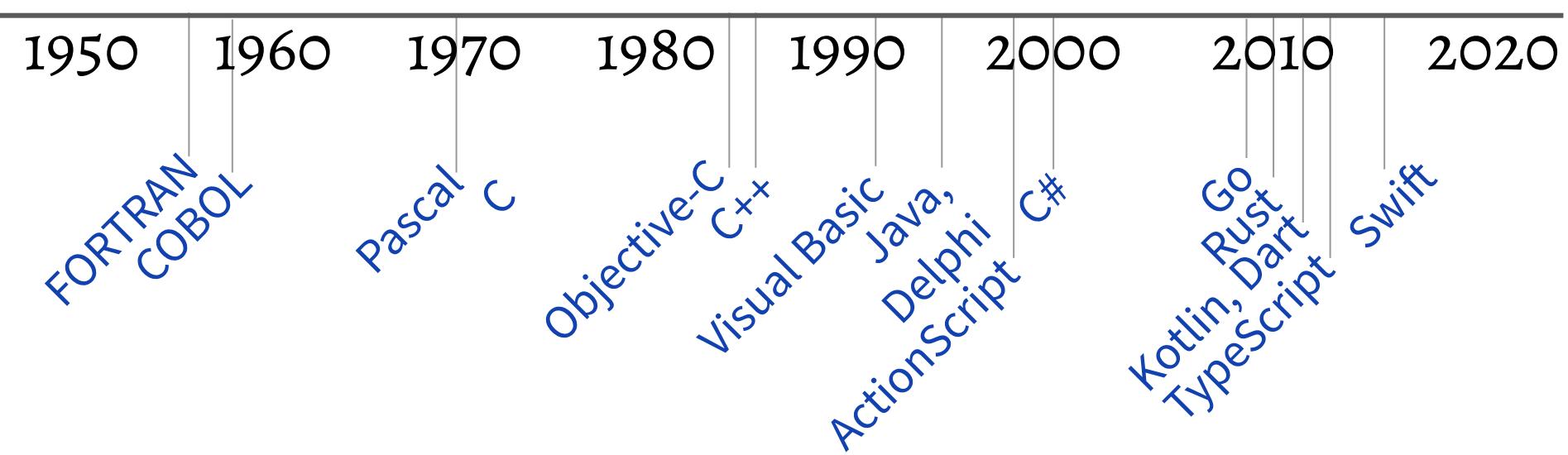
surfaces errors

Pendulum swings back?

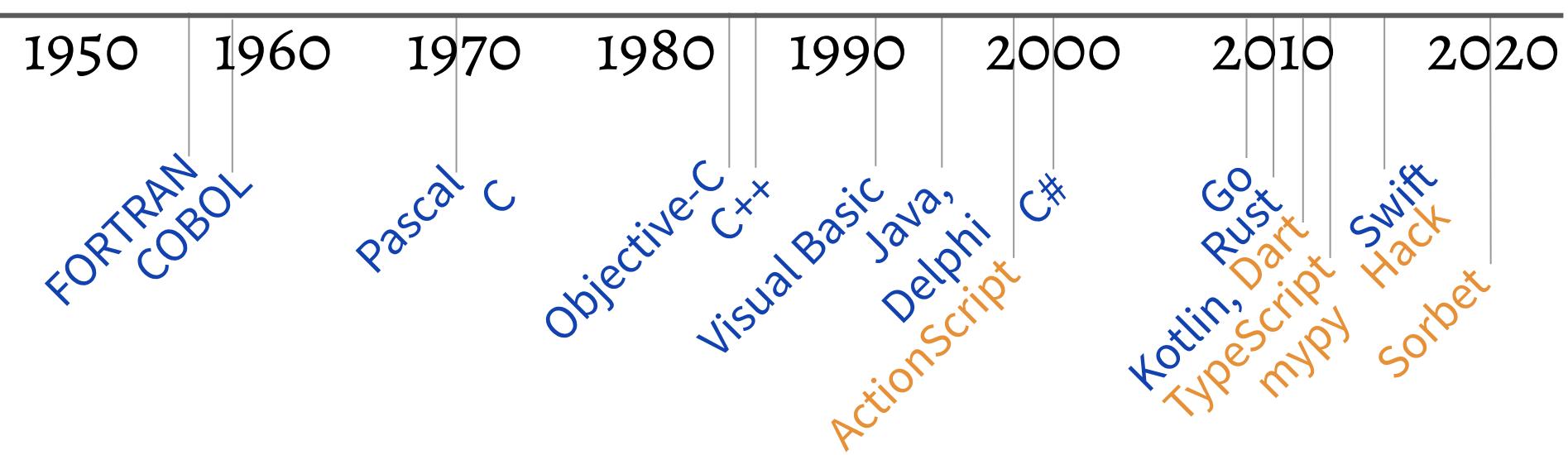
Statically typed languages can **learn from dynamic ones**
and incorporate their **most popular features...**
but **the reverse is not true.**

Will people **stop wanting red squiggles** for type errors?

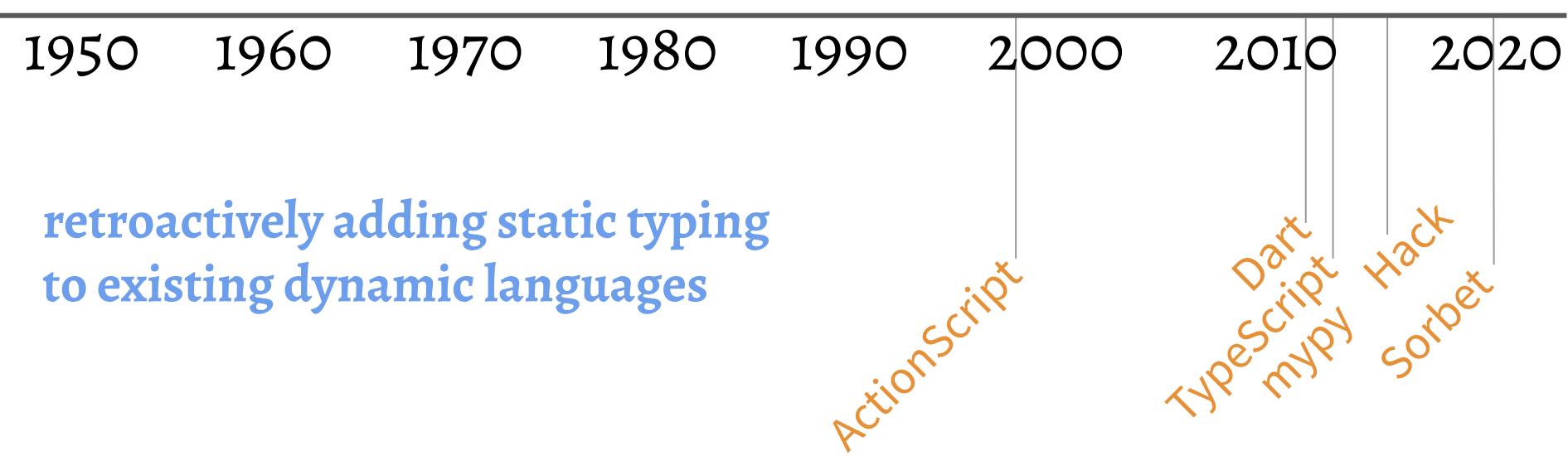
Most languages embrace gradual typing?



Most languages embrace gradual typing?



Most languages embrace gradual typing?



Most languages embrace gradual typing?

1950 1960 1970 1980 1990 2000 2010 2020

Dart's 2.0 release moved from
gradual typing to full static typing

ActionScript

Dart

Most languages embrace gradual typing?

1950 1960 1970 1980 1990 2000 2010 2020



ADOBE
FLASH PLAYER

ActionScript

Most languages embrace gradual typing?

1950 1960 1970 1980 1990 2000 2010 2020

Historical mainstream use of gradual typing:
retroactively adding static typing
to existing dynamic languages

Static without gradual gets more popular?

type system complexity



VS.

runtime overhead



VS.

Static without gradual gets more popular?

simple type system

no dynamic overhead

no mandatory type declarations

type error reports don't block running

validating deserialization via type inference



Prediction:

Among the next **5 languages** to enter the **top 20**,
most or all will be **statically but not gradually typed**.

SUMMARY

1. What made dynamic typing get big?
2. What changed?
3. What does this mean for the future?

NO STATIC TYPING

Assembly

BASIC

Prolog

MATLAB

Lua,
Perl
Bash
Python

JS,
Ruby,
PHP

(mainstream at some point)



SHIP FAST

1950 1960 1970 2000 2010 2020

FORTRAN
COBOL

Pascal
C

Objective-C
C++
Visual Basic
Java,
Delphi

C#

Dart,
Go
Rust
Kotlin
TypeScript
mypy
Swift
Hack
Sorbet

STATIC TYPING

What changed?

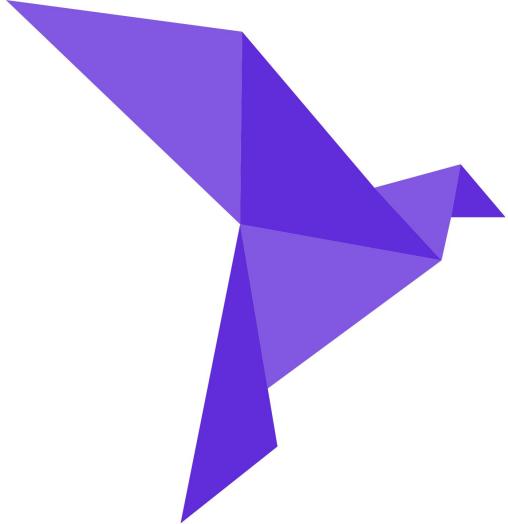
1. lots of ceremony → **no ceremony** required
2. not all that helpful → way **more helpful**
3. slow feedback loops → **actually faster** feedback loops

What does this mean for the future?

1. ~~The pendulum swings back to dynamic~~
2. ~~Most languages embrace gradual typing~~
3. Static without gradual gets more popular

A wide-angle photograph of a desert landscape. On the left, a massive, layered red rock cliff rises steeply from a lush green riverbank. To the right, another large red rock formation stands prominently. The river in the foreground is very still, creating a perfect mirror image of the surrounding rock faces and the sky above. The sky is a vibrant blue, dotted with scattered white and grey clouds.

Why Static Typing Came Back



roc-lang.org

I host a podcast!



software-unscripted.com