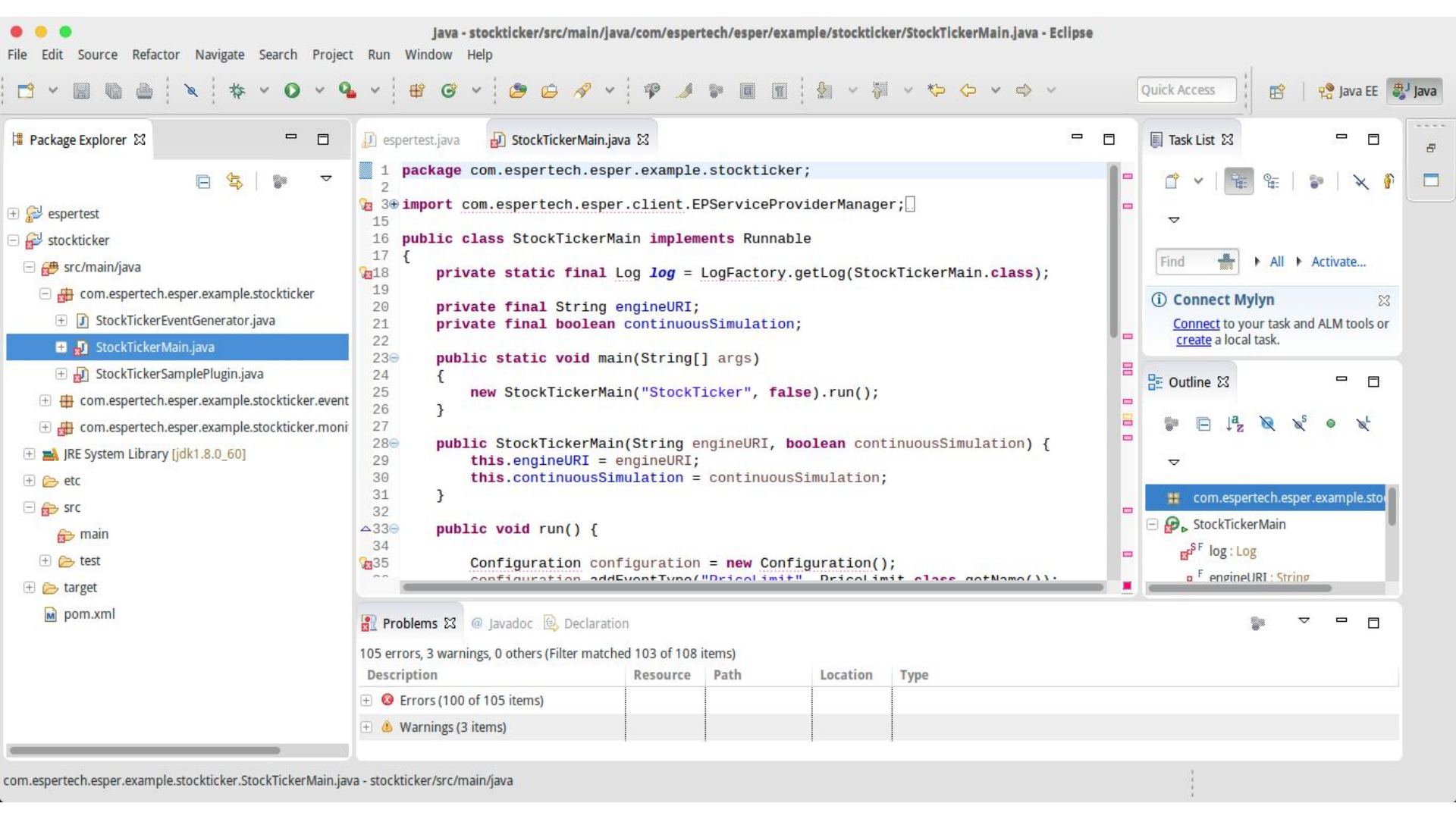

Why Static Typing Came Back
@rtfeldman


Java - stockticker/src/main/java/com/espertech/esper/example/stockticker/StockTickerMain.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
espertest
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.moni
JRE System Library [jdk1.8.0_60]
etc
src
main
test
target
pom.xml
espertest.java
StockTickerMain.java
1 package com.espertech.esper.example.stockticker;
2
3 import com.espertech.esper.client.EPServiceProviderManager;
15
16 public class StockTickerMain implements Runnable
17 {
18 private static final Log log = LogFactory.getLog(StockTickerMain.class);
19
20 private final String engineURI;
21 private final boolean continuousSimulation;
22
23 public static void main(String[] args)
24 {
25 new StockTickerMain("StockTicker", false).run();
26 }
27
28 public StockTickerMain(String engineURI, boolean continuousSimulation) {
29 this.engineURI = engineURI;
30 this.continuousSimulation = continuousSimulation;
31 }
32
33 public void run() {
34
35 Configuration configuration = new Configuration();
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
com.espertech.esper.example.sto
StockTickerMain
log : Log
engineURI : String
Problems
Javadoc
Declaration
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)
com.espertech.esper.example.stockticker.StockTickerMain.java - stockticker/src/main/java

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

```
    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 {
```

RedMonk

the developer-focused industry analyst firm

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

Static Typing

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

No Static Typing

JavaScript
Python
PHP
Ruby
R

RedMonk

Programming Language Rankings: January 2022

the developer-focused industry analyst firm

Static Typing			No Static Typing
C++	TypeScript	←	JavaScript
Rust	Dart	mypy ←	Python
Go	C#	Hack ←	PHP
C	Java	Sorbet ←	Ruby
Objective-C	Scala		R
Swift	Kotlin		

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

(mainstream at some point)

JS, Ruby, PHP

Lua, Perl

SHIP FAST

Visual Ba

STATIC TYPING

NO STATIC TYPING

(mainstream at some point)

JS, Ruby, PHP

Lua, R

Perl Bash Python

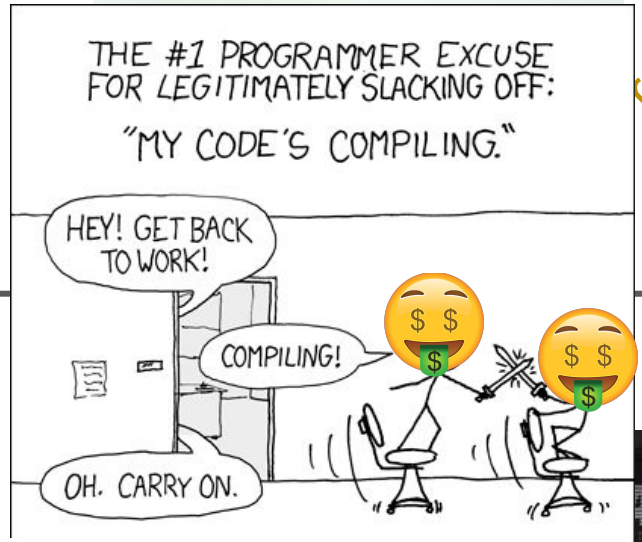
MATLAB

rolog

fast feedback

concise syntax

relevant sugar



xkcd.com/303

SHIP FAST

runs faster

IDE features

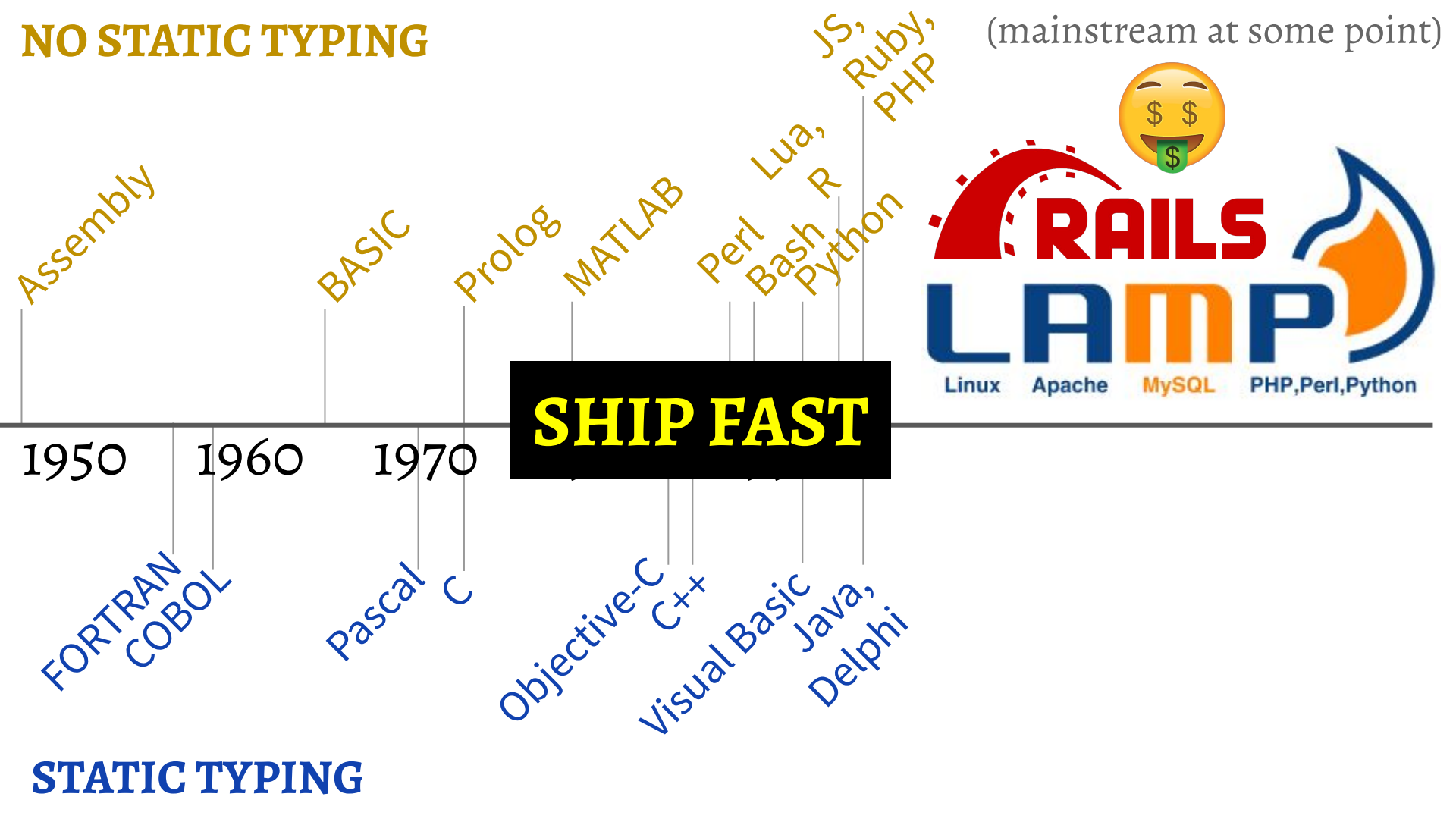
surfaces errors

Obje

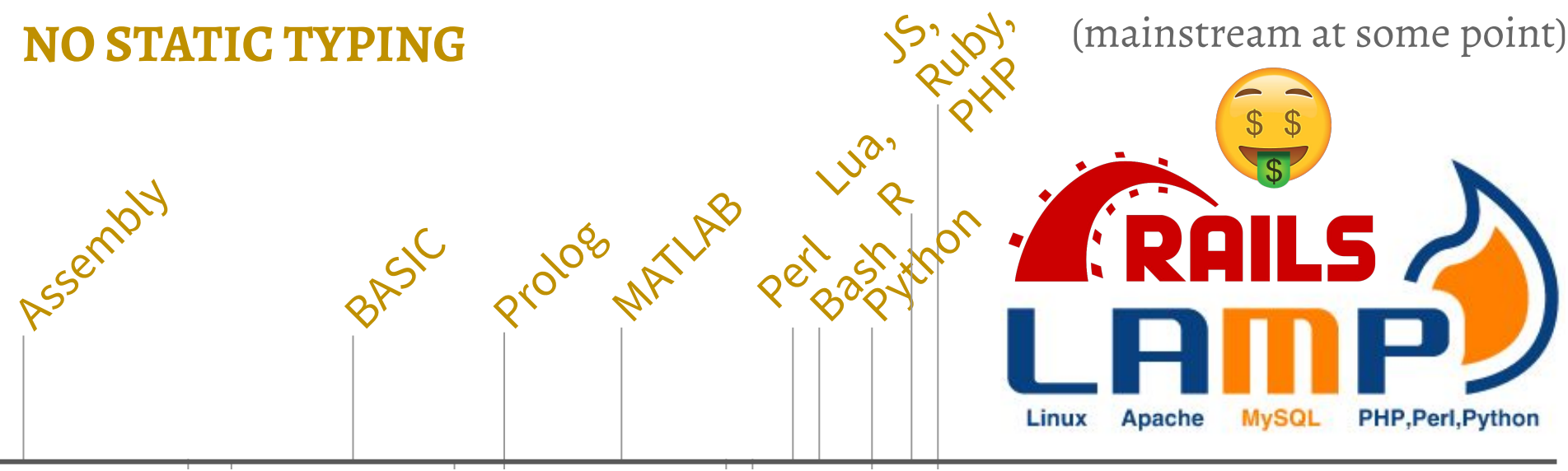
Visua

Del

STATIC TYPING


NO STATIC TYPING
(mainstream at some point)
JS,
Ruby,
PHP
Lua,
R
Assembly
BASIC
Prolog
MATLAB
Perl
Bash
Python
RAILS
LAMP
Linux
Apache
MySQL
PHP,Perl,Python
SHIP FAST
1950
1960
1970
FORTRAN
COBOL
Pascal
C
Objective-C
C++
Visual Basic
Java,
Delphi
STATIC TYPING

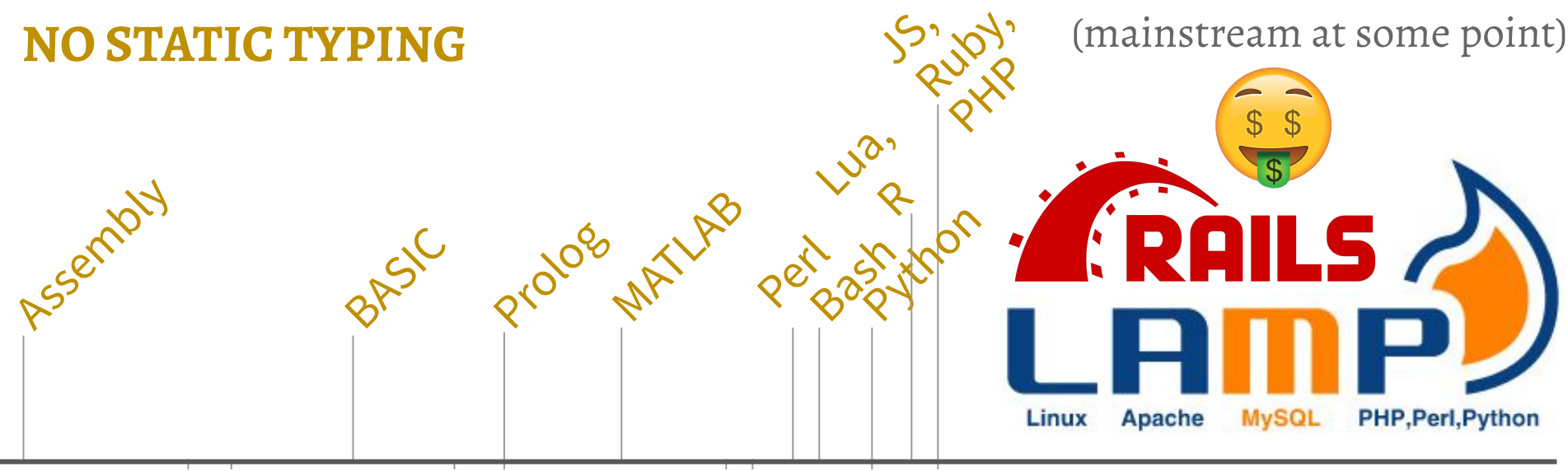
NO STATIC TYPING



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

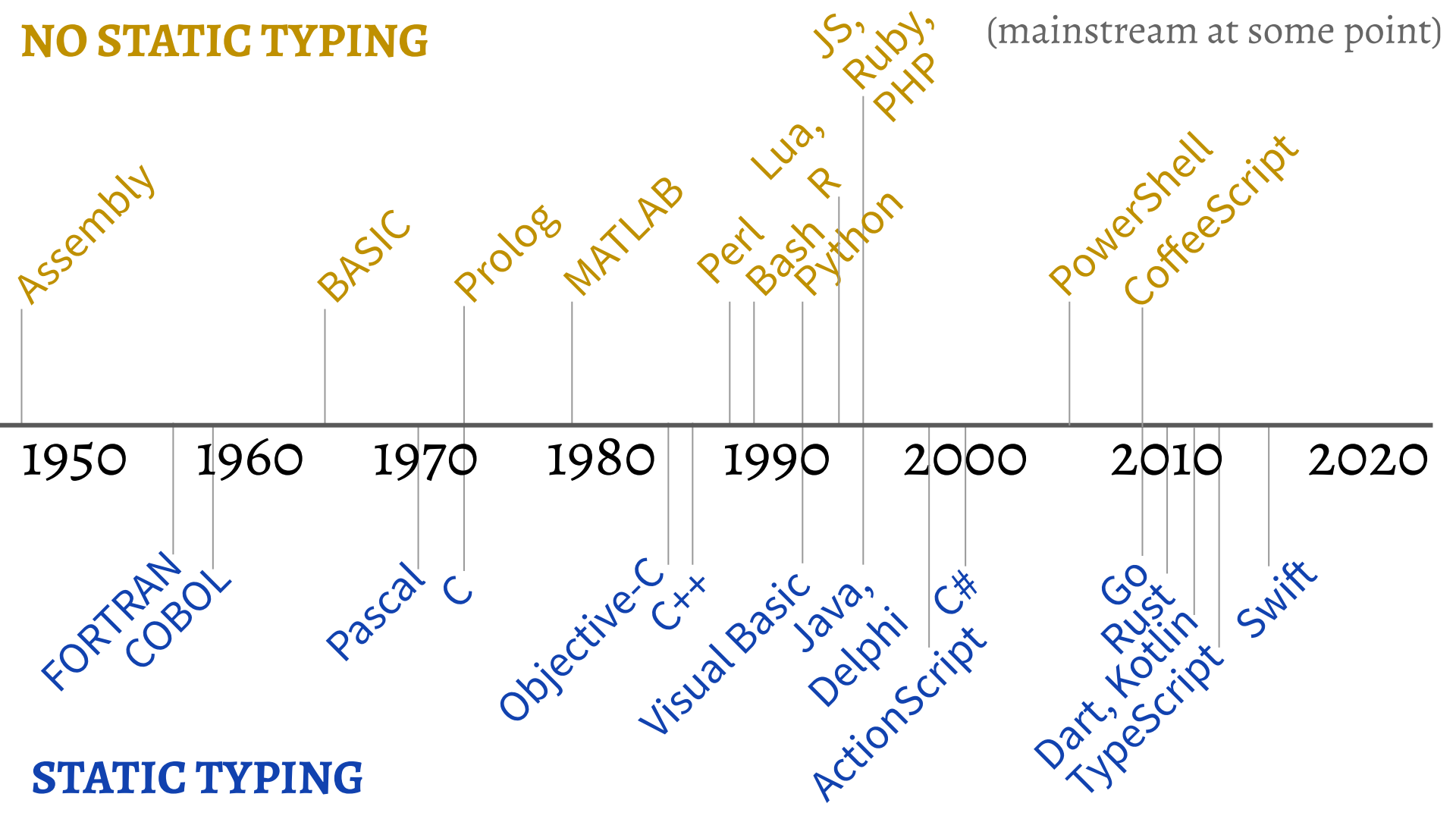
STATIC TYPING

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
Perl
Bash
Lua,
R
Python
JS,
Ruby,
PHP
PowerShell
CoffeeScript
1950
1960
1970
1980
1990
2000
2010
2020
FORTRAN
COBOL
Pascal
C
Objective-C
C++
Visual Basic
Java,
Delphi
C#
ActionScript
Go
Rust
Dart, Kotlin
TypeScript
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)",
            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 getWasEmpty() {
        return this.wasEmpty;
    }

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

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

    public void setDisplayName(String displayNa
        this.displayName = displayName;
    }

    @Override
    public boolean equals(final Obj
        if(obj instanceof Answer)
            final Answer other
            return new Eq
                .append
                .appe
                .i
        } else
            r
        }
    }

    pu int hashC
        ew Has
        d(was
        trimmed)
    }
}
```

object with 2 fields

constructor

getter

setter

getter

set

equals

hashCode

```
@JsonIgnoreProperties(ignoreUnk
public static class Json {
    @JsonProperty("name")
    public String name;

    @JsonProperty("
    public bool
}


publi
    Map
try
    J
    if (
        re
    } else {
        return Answer(json.name.isEmpty(), json.name);
    } catch (JsonProcessingException e) {
        return null;
    } catch (JsonMappingException e) {
        return null;
    }
}
```

schema

rawJson) {
ectMapper();
on = (Json)mapper.readValue(rawJson, Json.class);
admin) {
Answer(json.name.isEmpty(), json.name + " (Admin)"));

actual business logic

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

    if (typeof answer !==
        return "JSON de ailed!"
    } else if (answer
        return {
            disp Nam : answer.name + " (Admin)",
            was : answer.name.length === 0
        };
    } els {
        return {
            displayName: answer.name,
            wasEmpty: answer.name.length === 0
        };
}
```

STATIC TYPING

DYNAMIC TYPING

JavaScript™

Ruby on Rails demo

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/Gzj723LkRJY

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

```
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
        };
    }
}
```

```
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
        };
    }
}
```

```ts
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 getWasEmpty() {
        return this.wasEmpty;
    }

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

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

    public void setDisplayName(String displayNa
        this.displayName = displayName;
    }

    @Override
    public boolean equals(final Obj
        if(obj instanceof Answer)
            final Answer other
            return new Equ
                .append
                .appe
                .isEqu
        } else
            false
        }
    }
    e
    int hashC
        w Has
        (was
        trimmed)
    }
}
```

```
@JsonIgnoreProperties(ignoreUnk
public static class Json {
    @JsonProperty("name")
    public String name;

    @JsonProperty("
    public bool
}


publi
    Map
    try
        J
        if (
            re
        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(
    if (typeof answer !==
        return "JSON de
    } else if (answer
        return {
            disp
            was
        };
    } els
        {
            displayName: answer.name,
            wasEmpty: answer.name.length === 0
        };
}
```

STATIC TYPING

DYNAMIC 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 displayNa
        this.displayName = displayName;
    }

    @Override
    public boolean equals(final Obj
        if(obj instanceof Answer)
            final Answer other
            return new Eq
                .append
                .appe
                .
        } else
            r
        }
    }
    e
    int hashC
        w Has
        (was
        trimmed)
    }
}
```

```
@JsonIgnoreProperties(ignoreUnk
public static class Json {
    @JsonProperty("name")
    public String name;

    @JsonProperty("
    public bool
}


publi
    Map
    try
        J
        if (
            re
        else {
            return Answer(json.name.isEmpty(), json.name);
        } catch (JsonProcessingException e) {
            return null;
        } catch (JsonMappingException e) {
            return null;
        }
    }
```

```
g rawJson) {
ectMapper();
on = (Json)mapper.readValue(rawJson, Json.class);
admin) {
Answer(json.name.isEmpty(), json.name + " (Admin)"));
```

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

    if (typeof answer !== "object") {
        return "JSON de ailed!"
    } else if (answer. === true) {
        return {
            displ : `${answer.name} (Admin)`,
            wasEm answer.name.length === 0
        };
    } el {
        return {
            displayName: answer.name,
            wasEmpty: answer.name.length === 0
        };
    }
}
```

STATIC TYPING

STATIC TYPING

```
class Answer {
    private boolean
    private String


    public Answer(b
        this.wasEmp
        this.displa
    }

    public boolean
        return this
    }

    public void set
        this.wasEmp
    }

    public String g
        return this
    }

    public void set
        this.displa
    }

    @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

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

// false in JavaScript/TypeScript
```

```
@JsonIgnoreProperties(ignoreUnknown = true)
```

```
}
```


```
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
        };
    }
}
```

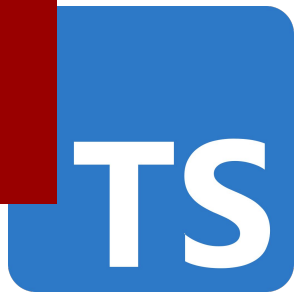
Advantages

nice `equals` and `hashCode` implementations

early JSON validation instead of crashing later

```
answer.name.length
```

```
Uncaught TypeError:
Cannot read properties of undefined
(reading 'length')
```




```
class Answer {
    private boolean
    private String

    public Answer(b
        this.wasEmp
        this.displa
    }

    public boolean
        return this
    }

    public void set
        this.wasEmp
    }

    public String g
        return this
    }

    public void set
        this.displa
    }

    @Override
    public boolean equals(fin
        if(obj instanceof Ans
            final Answer othe

            return new Equals
                .append(wasEm
                .append(displ
                .isEquals();
        } else {
            return false;
        }
    }

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

```
@JsonIgnoreProperties(ignoreUnknown = true)
}

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

```
class Answer {
    private boolean
    private String

    public Answer(b
        this.wasEmp
        this.displa
    }

    public boolean
        return this
    }

    public void set
        this.wasEmp
    }

    public String g
        return this
    }

    public void set
        this.displa
    }

    @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~~ 😂

```
@JsonIgnoreProperties(ignoreUnknown = true)
```

```
}
```

```
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
    private String

    public Answer(b
        this.wasEmp
        this.displa
    }

    public boolean
        return this
    }

    public void set
        this.wasEmp
    }

    public String g
        return this
    }

    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)
    }
}
```

Advantages

nice `equals` and `hashCode` implementations

early JSON validation instead of crashing later


```
@JsonIgnoreProperties(ignoreUnknown = true)
                }
            } 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
        };
    }
}
```

✅ 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
        };
    }
}
```

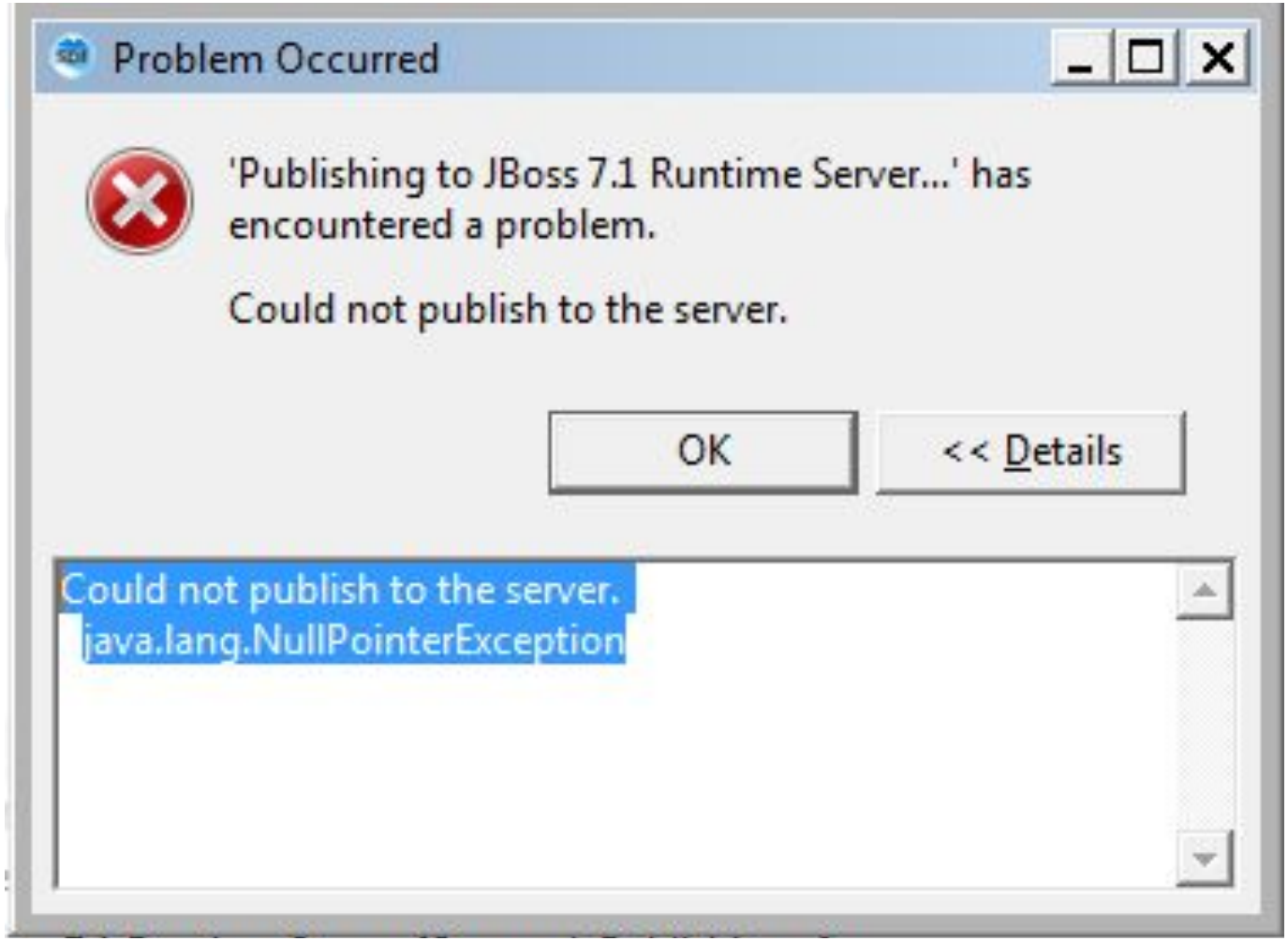
My view:

~~"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

noredink

`nil`/`null`/`undefined` errors in production

thousands reached production,

after thousands of **tests passed**

100% **caught at build time**,

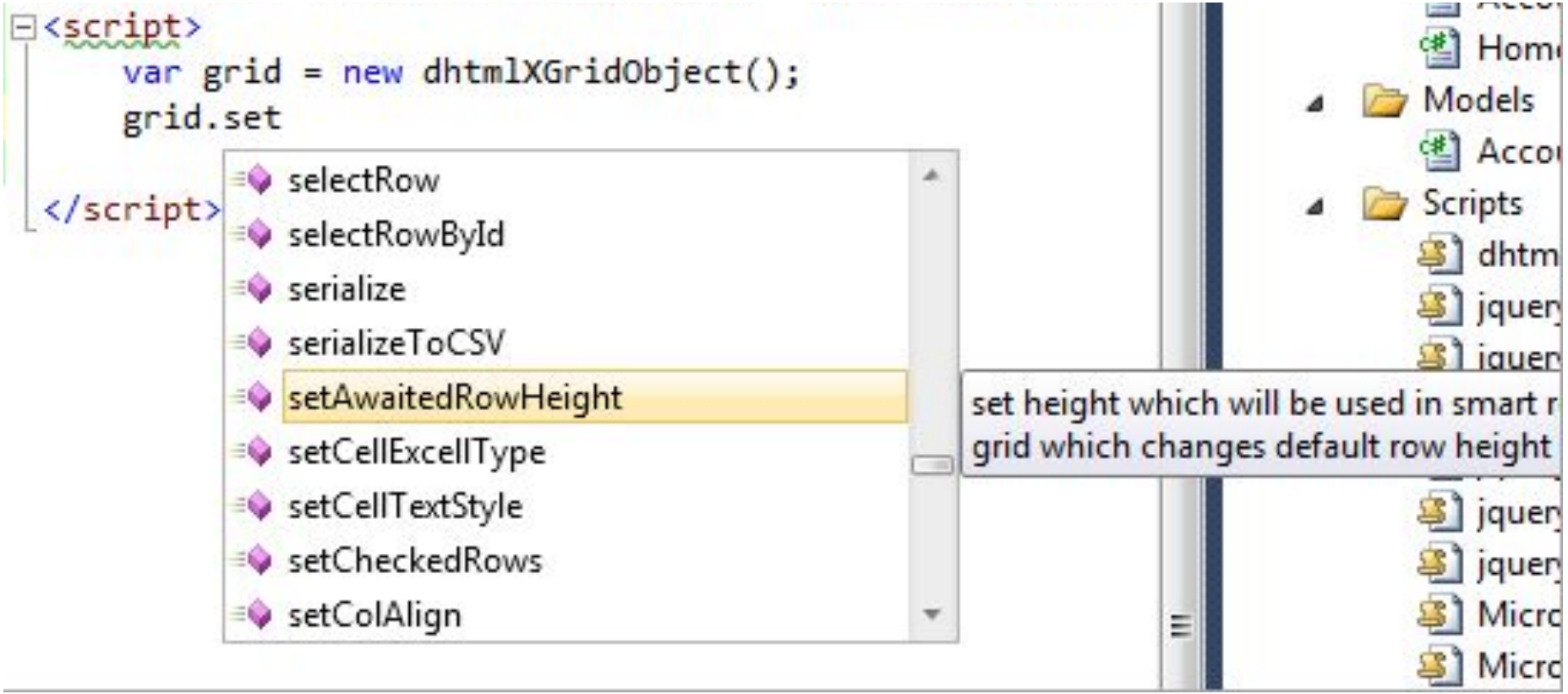
never reached production

mcc
@mcclure111

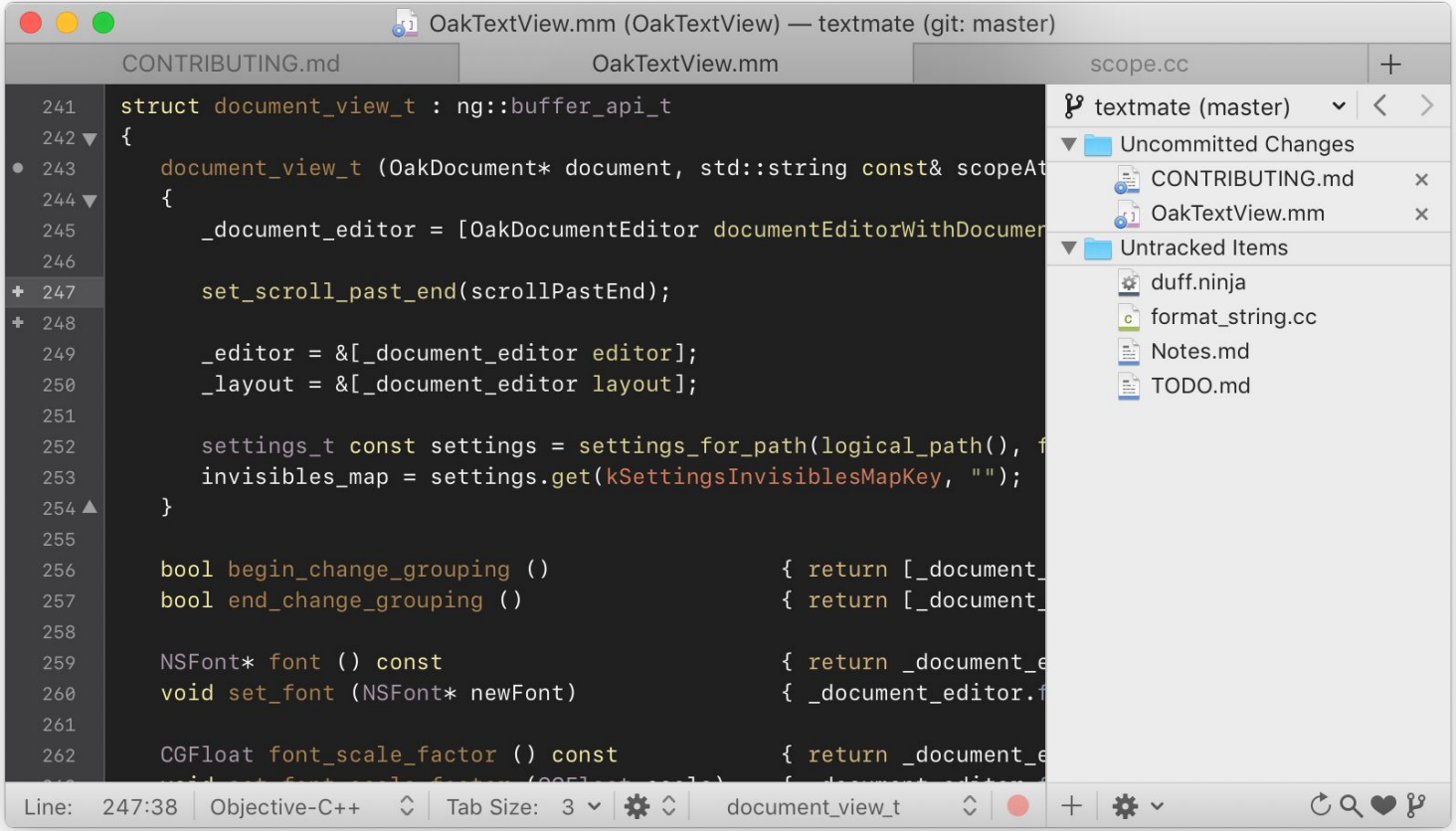
In C++ we don't say "Missing asterisk"

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
```



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



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>
```

REPL

vs.

C++, Java, etc…

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","
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

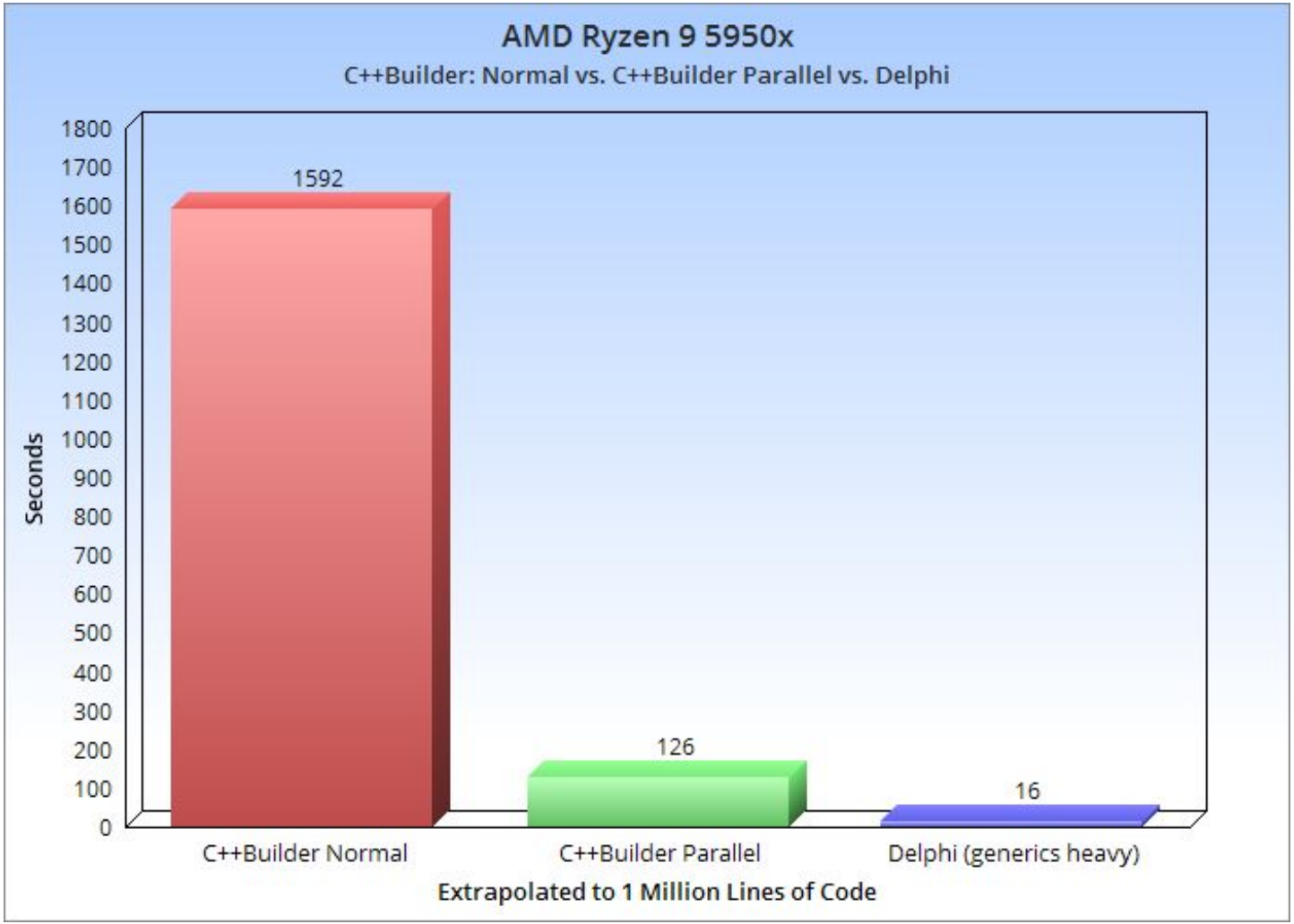
REPL

Compiler performance improvements

multicore compilation

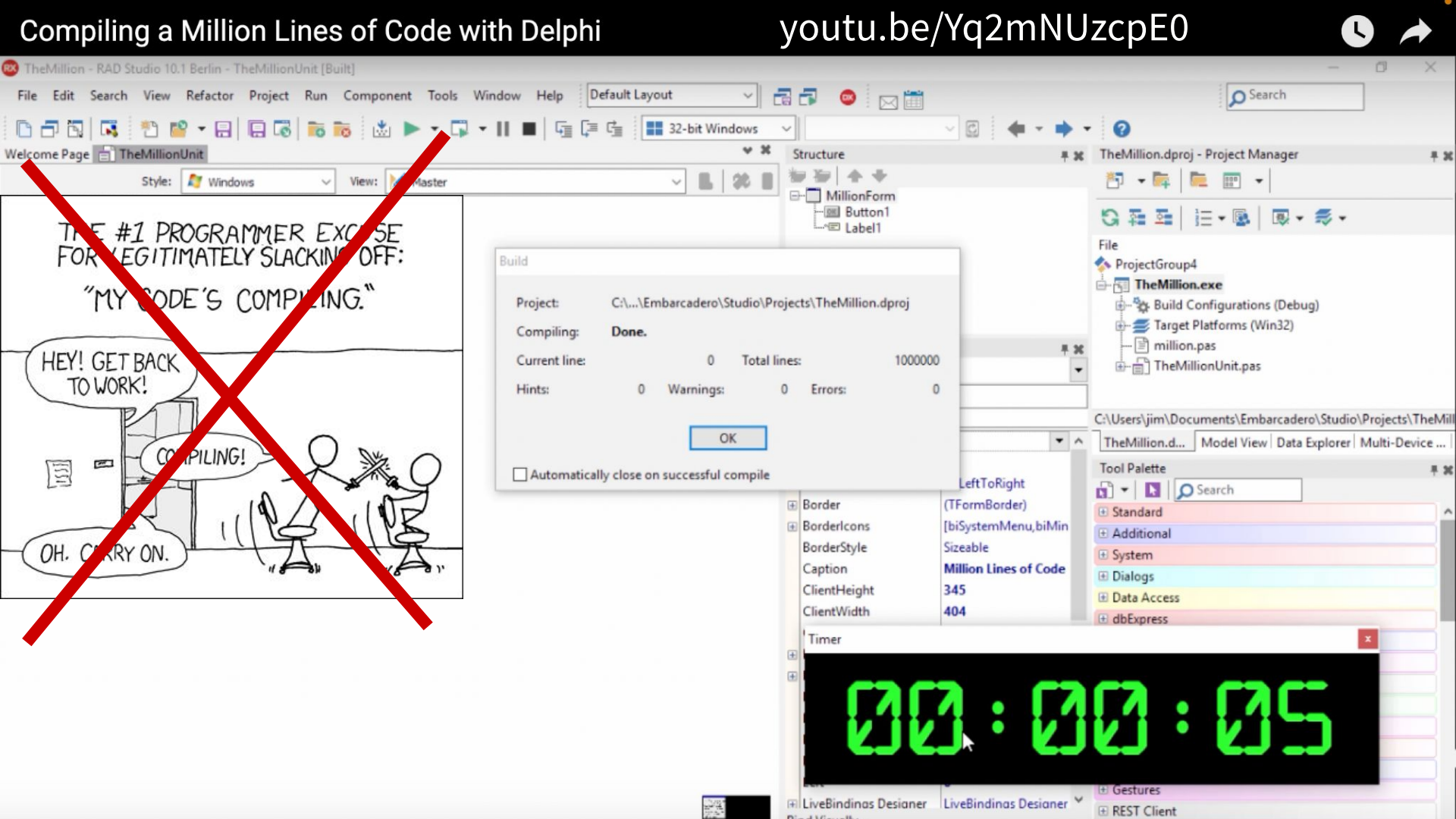
incremental recompiles

considering build speed in language design


AMD Ryzen 9 5950x
C++Builder: Normal vs. C++Builder Parallel vs. Delphi
Seconds
1800
1700
1600
1500
1400
1300
1200
1100
1000
900
800
700
600
500
400
300
200
100
0
1592
126
16
C++Builder Normal
C++Builder Parallel
Delphi (generics heavy)
Extrapolated to 1 Million Lines of Code

Compiling a Million Lines of Code with Delphi

youtu.be/Yq2mNUzcpE0



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

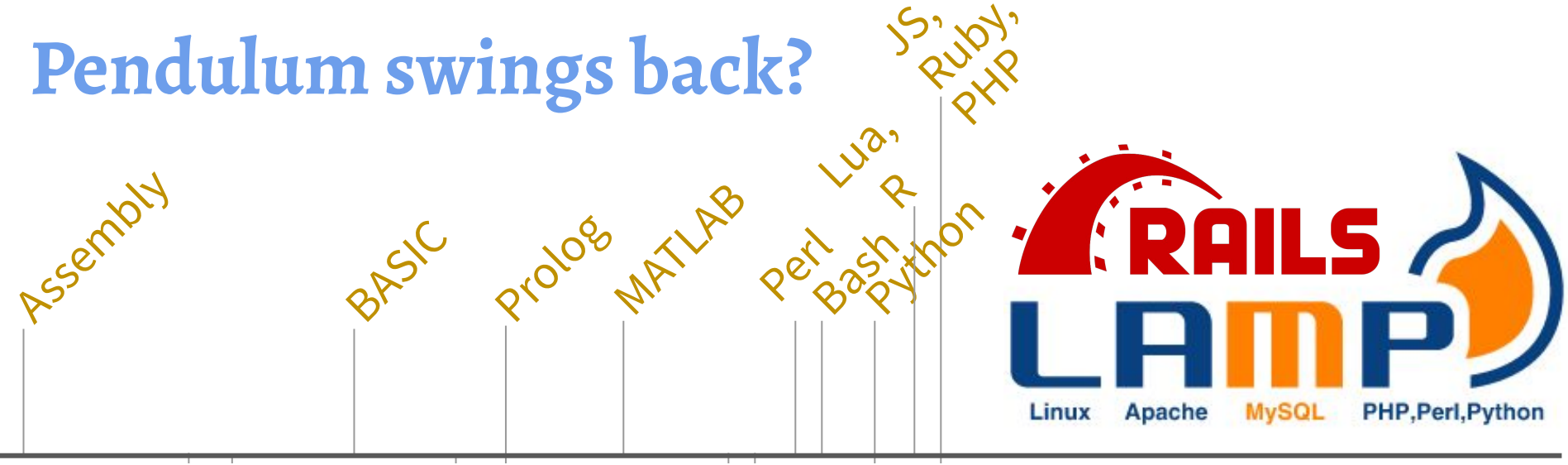
1. What made dynamic typing get big?
2. **What changed?**
3. What does this mean for the future?

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

3 hypothetical futures

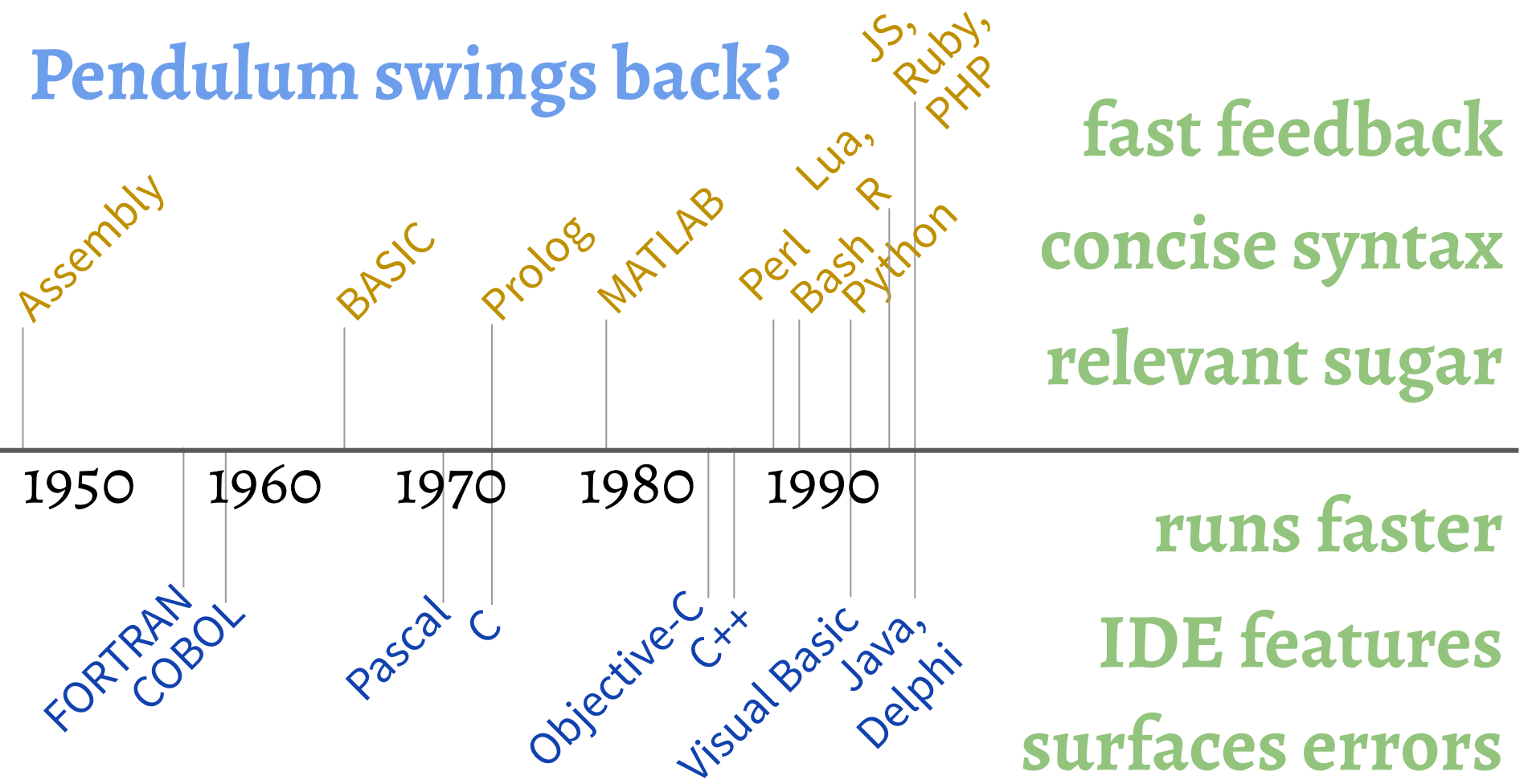
1. The pendulum swings back to dynamic
2. Most languages embrace gradual typing
3. Static without gradual gets more popular

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?



fast feedback

concise syntax

relevant sugar

runs faster

IDE features

surfaces errors

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.

fast feedback

concise syntax

relevant sugar

Dynamic typing **requires** some runtime overhead.

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

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

runs faster

IDE features

surfaces errors

Pendulum swings back?

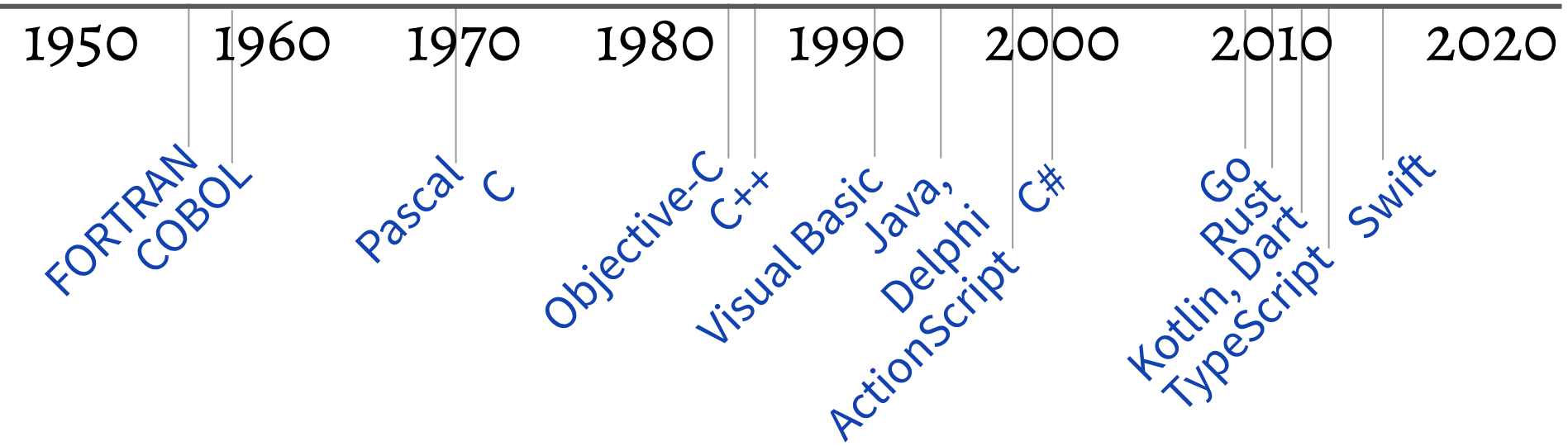
Statically typed languages can **learn from dynamic ones**

and incorporate their **most popular features**...

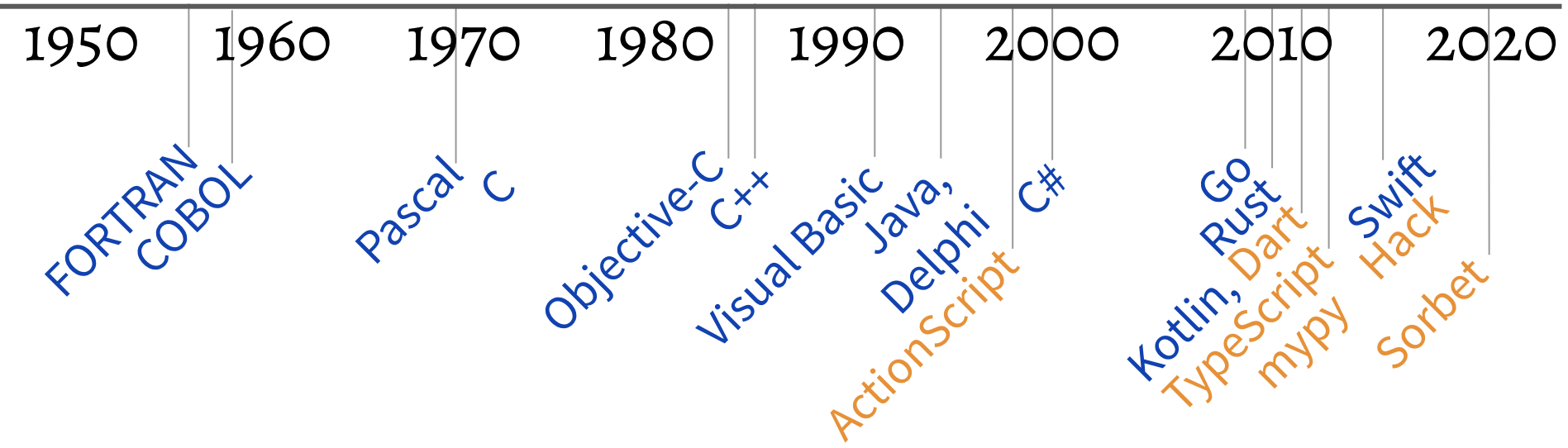
but **the reverse is not true**.

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

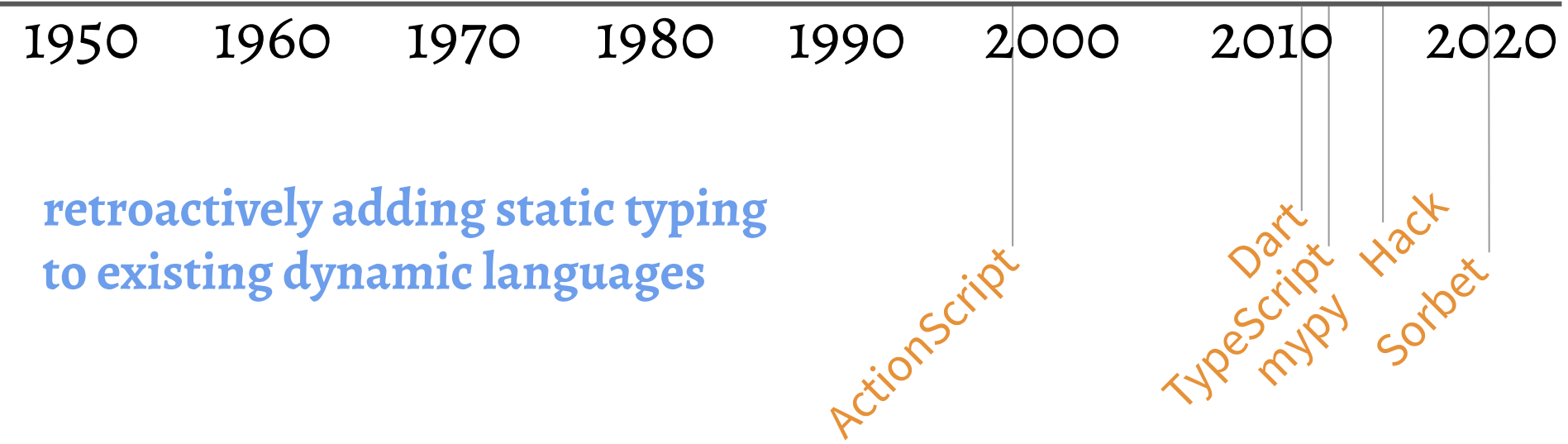
Most languages embrace gradual typing?



Most languages embrace gradual typing?

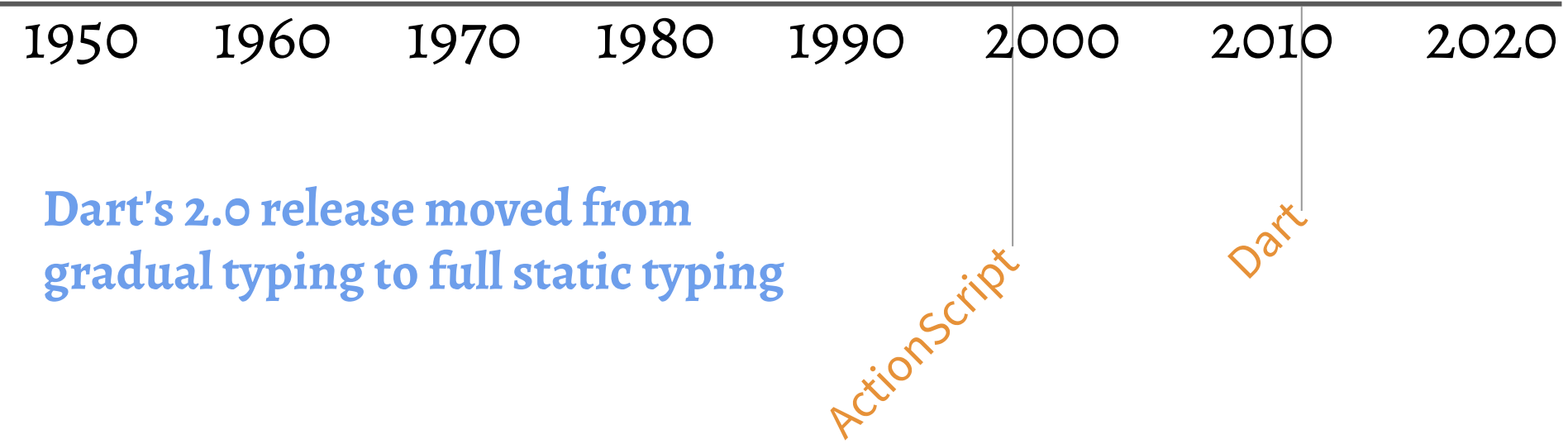


Most languages embrace gradual typing?



retroactively adding static typing
to existing dynamic languages

Most languages embrace gradual typing?



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

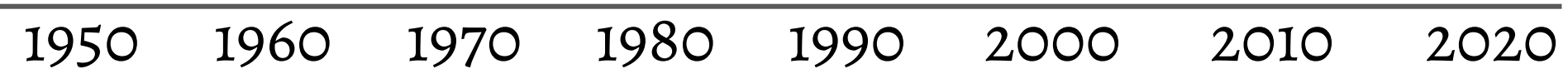
Most languages embrace gradual typing?

1950 1960 1970 1980 1990 2000 2010 2020

ADOBE
FLASH PLAYER

ActionScript

Most languages embrace gradual typing?

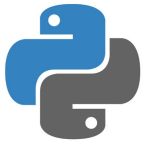


Historical mainstream use of gradual typing:

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

roc-lang.org

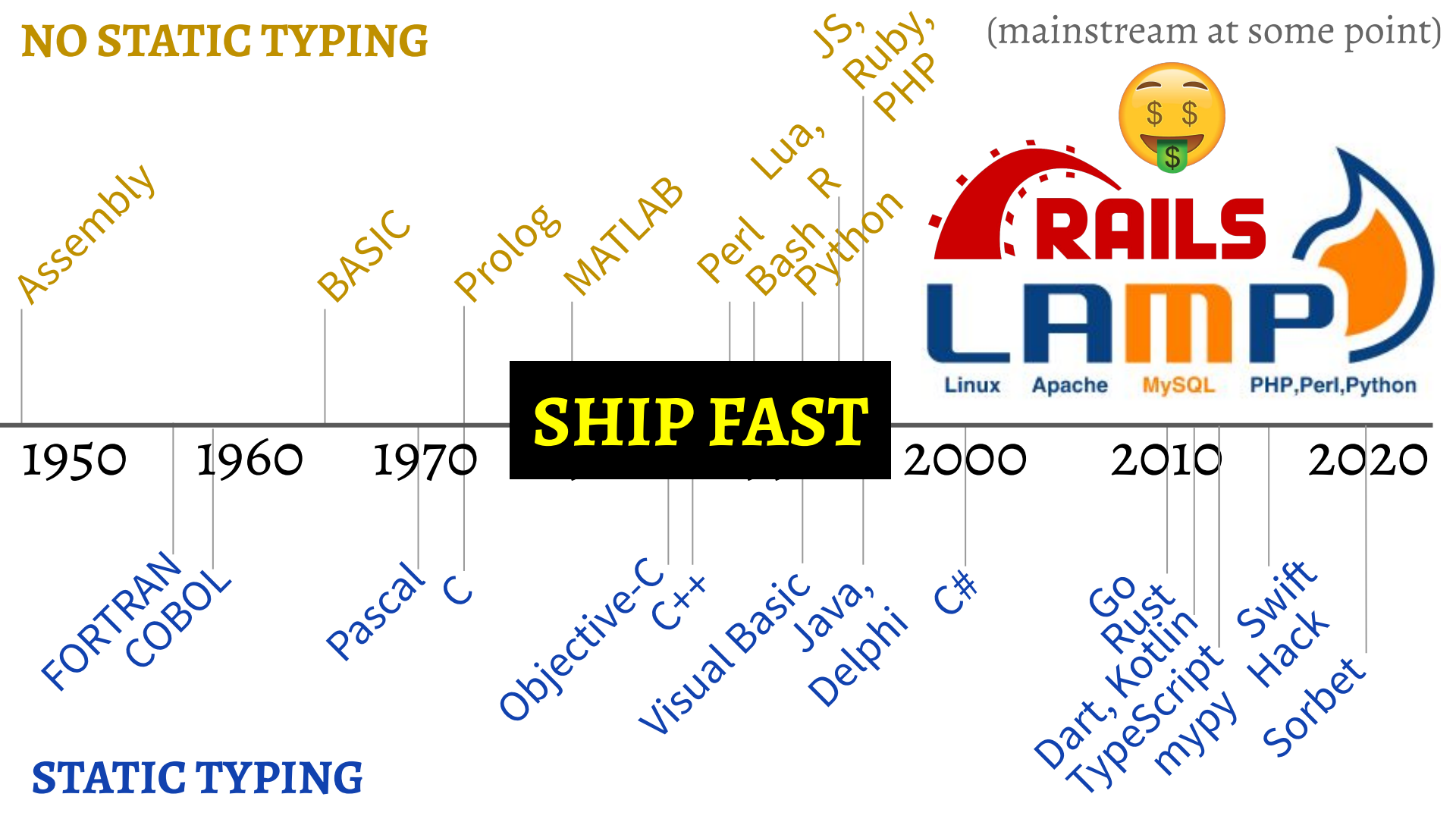
Prediction:

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
(mainstream at some point)
JS, Ruby, PHP
Assembly
BASIC
Prolog
MATLAB
Lua, R
Perl
Bash
Python
RAILS
LAMP
Linux
Apache
MySQL
PHP,Perl,Python
SHIP FAST
1950
1960
1970
2000
2010
2020
FORTRAN
COBOL
Pascal
C
Objective-C
C++
Visual Basic
Java, Delphi
C#
Go
Rust
Dart, Kotlin
TypeScript
mypy
Swift
Hack
Sorbet
STATIC TYPING

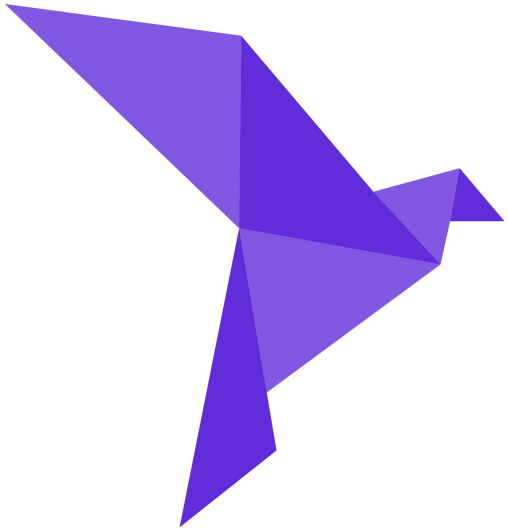
What changed?

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


Why Static Typing Came Back



roc-lang.org

I host a podcast!

software-unscripted.com