Geek out with the smart language additions in ColdFusion 2018

Rakshith Naresh | Product Manager, Adobe Inc

Programming languages introduced in the 90s

Game Maker Language (GML)
UnrealScript
Borland Delphi
Lua
ISLISP
EuLisp
NewtonScript
E
Oberon-2
ColdFusion (CFML)
PHP
JavaScript
Amiga E
Squeak Smalltalk
Object Oberon
Borland Pascal
RAPID
Harbour
Z Shell (zsh)
Pikt
Java
LiveScript
Ruby
F-Script
GNU E
Pike
J
Lasso
K
AMOS BASIC
AppleScript
ECMAScript
Perl Data Language (PDL)
Pico
PureBasic
Python
Ada 95
ANS Forth
OCaml
Haskell
NetRexx
ZPL
AMPL
XSLT (+ XPath)
Q
M2001
Oz
Component Pascal
REBOL
Dylan
Visual Basic
Revolution Transcript
R
Claire
ANSI Common Lisp
Curl
Tea
Self
Mercury
Standard C++

Relevant and well known in 2019

Game Maker Language (GML)
UnrealScript
Borland Delphi
Lua
ISLISP
EuLisp
NewtonScript
E
Oberon-2
ColdFusion (CFML)
PHP
JavaScript
Amiga E
Squeak Smalltalk
Object Oberon
Borland Pascal
RAPID
Harbour
Z Shell (zsh)
Pikt
Java
LiveScript
Ruby
F-Script
GNU E
Pike
J
Lasso
K
AMOS BASIC
AppleScript
ECMAScript
Perl Data Language (PDL)
Pico
PureBasic
Python
Ada 95
ANS Forth
OCaml
Haskell
NetRexx
ZPL
AMPL
XSLT (+ XPath)
Q
M2001
Oz
Component Pascal
REBOL
Dylan
Visual Basic
Revolution Transcript
R
Claire
ANSI Common Lisp
Curl
Tea
Self
Mercury
Standard C++

Agenda for the day – ColdFusion 2018 language improvements

- You will love these changes!
- Object oriented adrenaline
- Asynchronous programming
- All about arrays
- Some additional goodness
- What's coming up?

You will love these changes!

- Null support
- Datatype preservation

Null support

- JavaScript to ColdFusion
  - A null variable is undefined
- Database to ColdFusion
  - A null value becomes “”

Datatype preservation

- CF used to interpret:
  - “1234” as 1234
  - “005678” as 5678
  - “true” or “yes” to Boolean true

Now, CF preserves the type. Let’s take a look at CFFiddle!

Object oriented adrenaline

- Abstract
- Default functions in interface
- Final

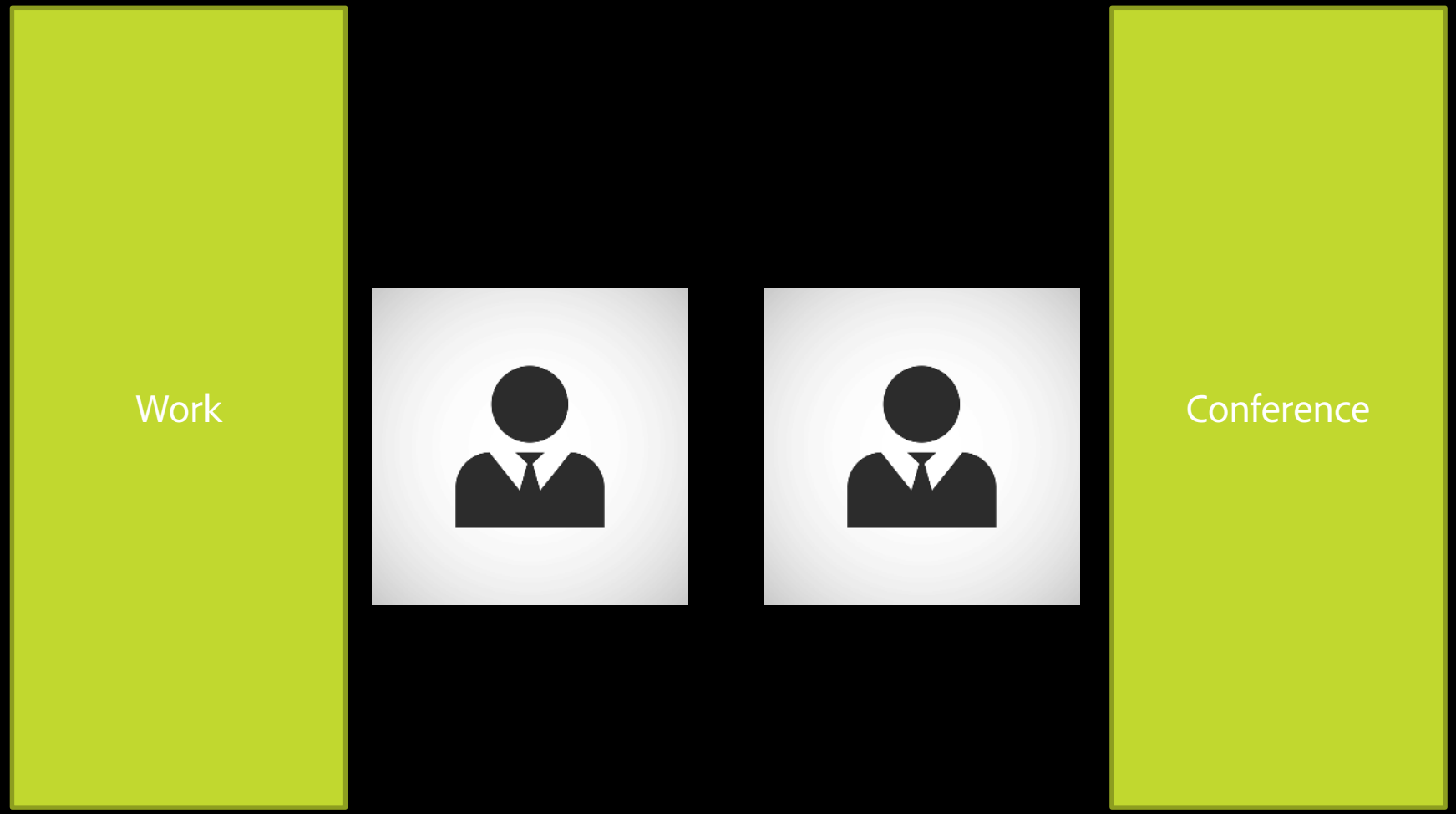
ColdFusion Developers

- Code
- Meet deadlines
- Respond to urgent emails
- Be available on calls

ColdFusion Developers at CFSummit East

- Learn best practices
- Network with CF Devs

Asynchronous programming - The dilemma of conference travel



Tasks associated with the dilemma

Work

- RespondToUrgentEmails
- MeetDeadlines
- BeAvailableOnCalls

Conference

- Promise to Boss to show ROI
  - Implement at least three best practices learnt at the conference
- Promise to self – have good time
  - Network with other ColdFusion developers

Pseudo code

```
while (your business trip does not end)
{
    checkUrgentEmails();
    workToMeetDeadlines();
    checkForImportantCalls();
}
```

The runasync magic

```
runAsync(learnBestPractices);

while (your business trip does not end)
{
    checkUrgentEmails();
    workToMeetDeadlines();
    checkForImportantCalls();
}
```

The runAsync Magic with a future

```
var future = runAsync(learnBestPractices);

while (your business trip does not end)
{
    checkUrgentEmails();
    workToMeetDeadlines();
    checkForImportantCalls();
}
```

The runAsync Magic - chaining

```
var future = runAsync(learnBestPractices).then(networkWithCFDevs);

while (your business trip does not end)
{
    checkUrgentEmails();
    workToMeetDeadlines();
    checkForImportantCalls();
}
```

The runAsync Magic - timeout

```
var future = runAsync(learnBestPractices, 0.5 day).then(networkWithCFDevs);

while (your business trip does not end)
{
    checkUrgentEmails();
    workToMeetDeadlines();
    checkForImportantCalls();
}
```

The runAsync Magic – error handler

```
var future = runAsync(learnBestPractices).then(networkWithCFDevs).error(getStuckInDCTraffic);

while (your business trip does not end)
{
    checkUrgentEmails();
    workToMeetDeadlines();
    checkForImportantCalls();
}
```

The runAsync Magic – time for result

```
var future = runAsync(learnBestPractices).then(networkwithCFDevs).error(getStuckInDCTraffic);

while (your business trip does not end)
{
    checkUrgentEmails();
    workToMeetDeadlines();
    checkForImportantCalls();
}
```

```
future.get();
```

Syntax

future = runAsync(func,1).then(thenFunc).error(errorHandler);

Set of APIs on future

get()
get(timeout)
get(timeout, timeUnit)
isDone()
cancel(mayInterruptIfRunning)
isCancelled()
then(UDFMethod)
then(UDFMethod, timeout)
error(UDFMethod)
error(UDFMethod, timeout)

All about Arrays

- Typed Array
- Negative index
- Slicing

Some additional goodness

- Optional semi colon

What's coming up?

- Support for Lambda
- Array functions – pop,push,shift,unshift,splice
- Query functions –
  - Updated querynew, queryappend, queryprepend, queryrowswap, queryslice, queryclear
- Some, Every – array, struct, query

Here is a trick question before we wrap up!

All code samples are on CFFiddle

Make sure to login to fiddle to view the code samples.

Here is the link for all the code


Adobe