



PROFESSIONAL  
EDUCATION

A large, stylized green arrow pointing to the right, with a cyan-to-green gradient, serving as a background for the main title.

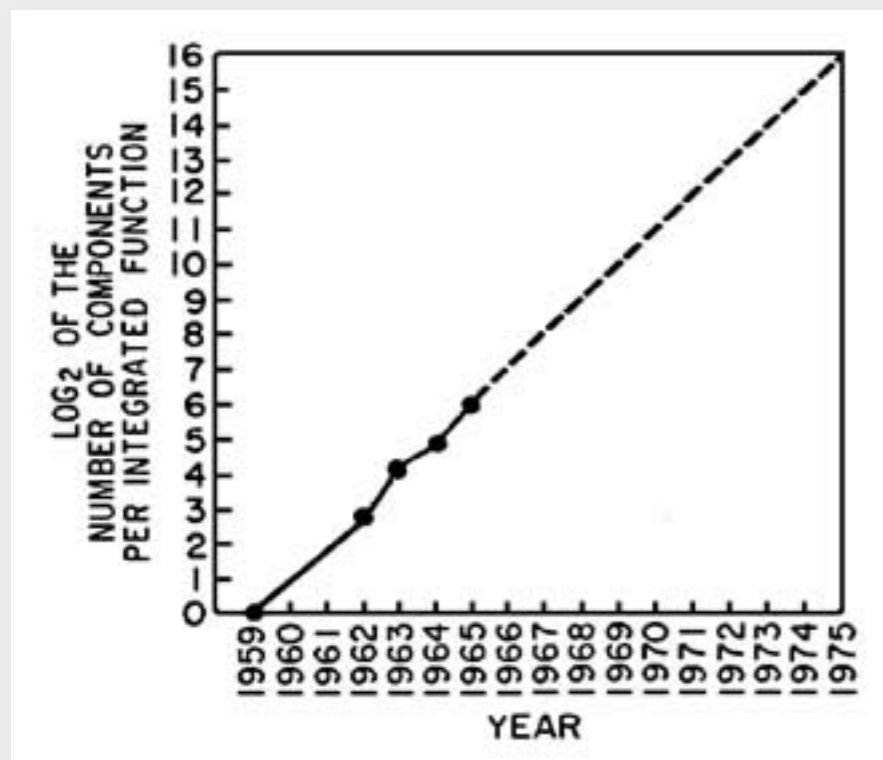
**ACCENTURE  
TECHNOLOGY  
ACADEMY  
2.0**

**LOW CODE, CITIZEN  
DEVELOPMENT & NEW  
PROGRAMMING  
PARADIGMS**

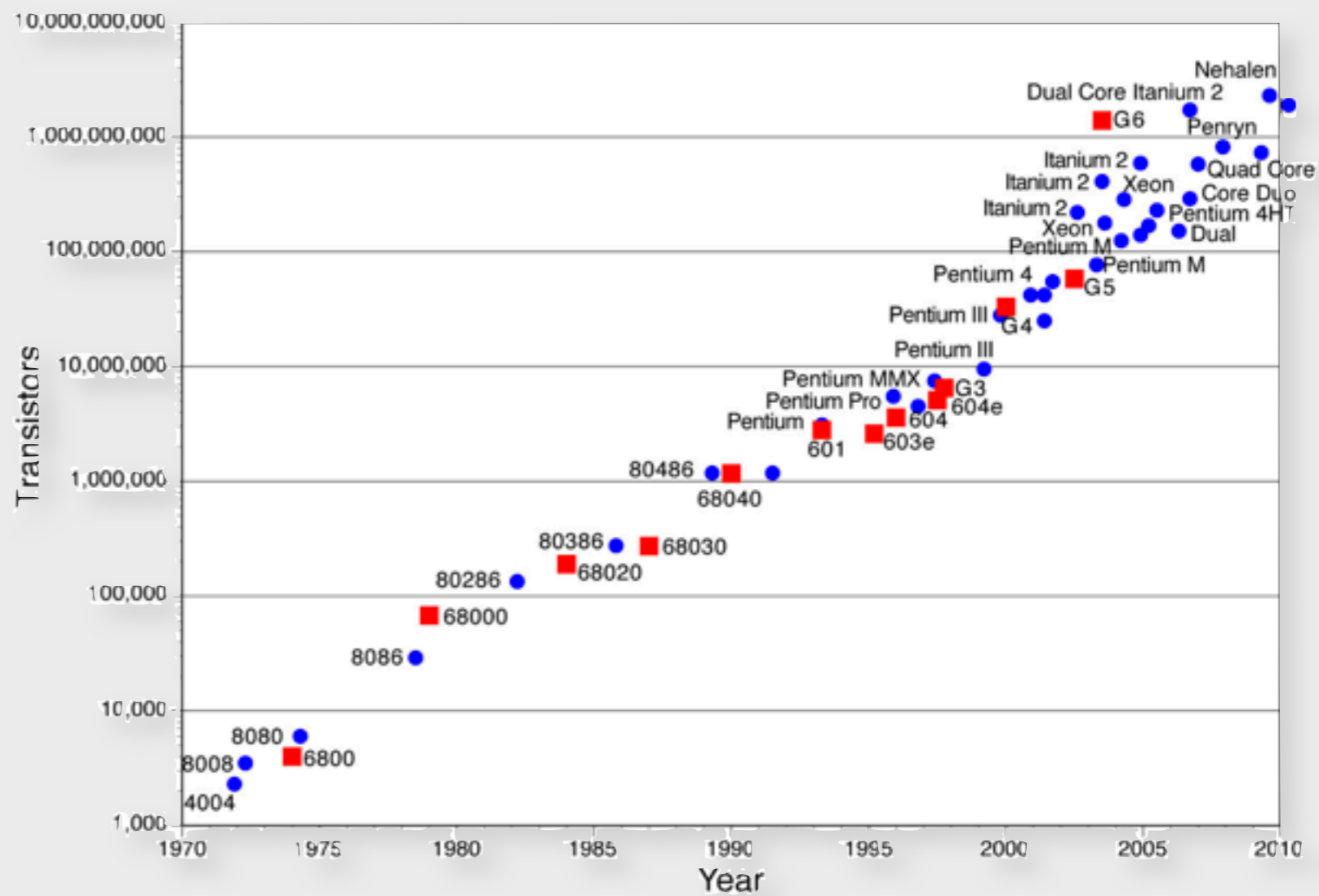
DANIEL JACKSON

the growth of  
computational  
power

# more transistors

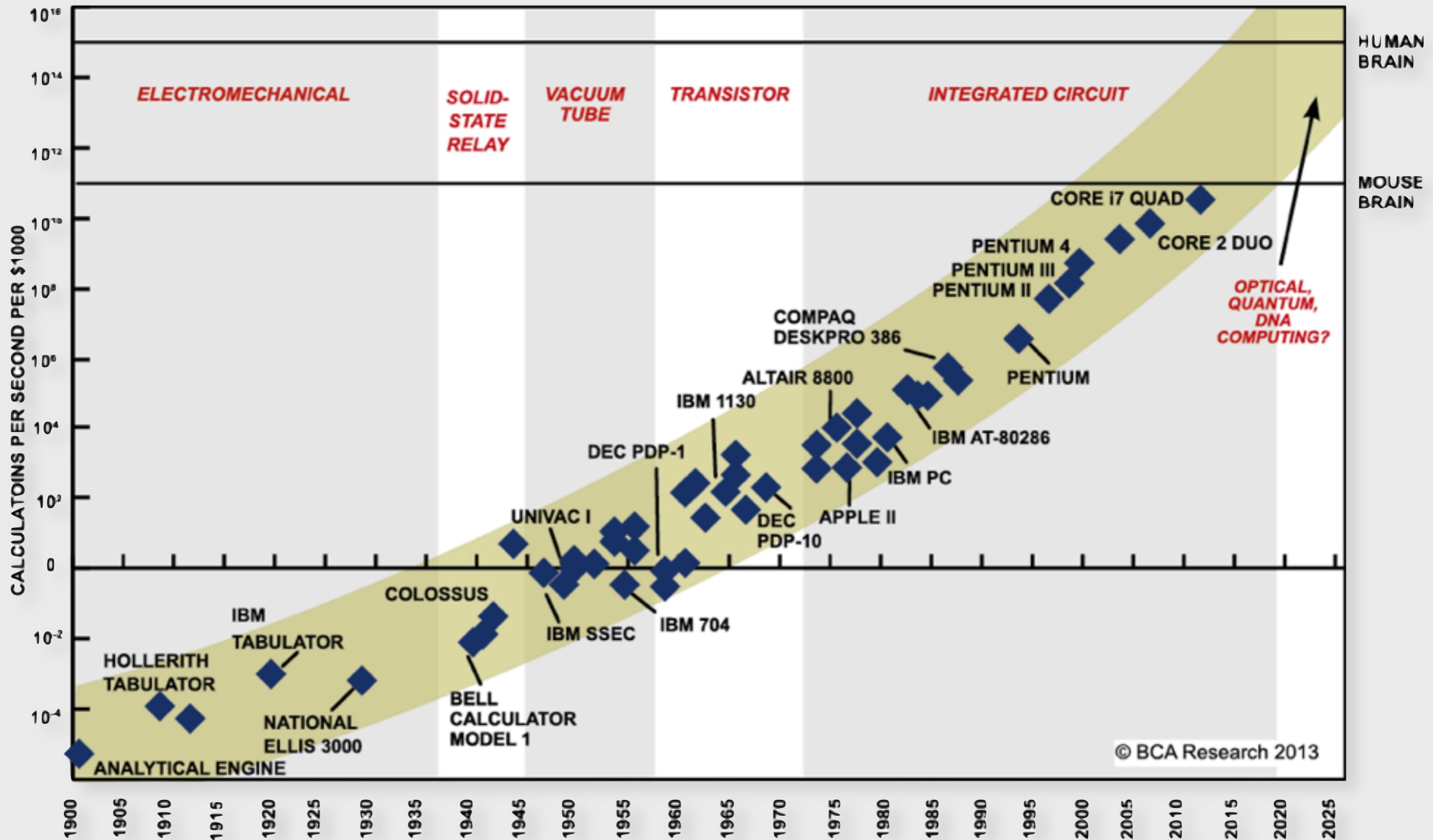


predicted (1965)



actual (2010)

# more calculations



SOURCE: RAY KURZWEIL, "THE SINGULARITY IS NEAR: WHEN HUMANS TRANSCEND BIOLOGY", P.67, THE VIKING PRESS, 2006. DATAPOINTS BETWEEN 2000 AND 2012 REPRESENT BCA ESTIMATES.

# reversing a string, 1960-1990

```
REVERSE CSECT
        USING REVERSE,R13
        B     72(R15)
        DC   17F'0'
        STM  R14,R12,12(R13)
        ST   R13,4(R15)
        ST   R15,8(R13)
        LR   R13,R15
        MVC  TMP(L'C),C
        LA   R8,C
        LA   R9,TMP+L'C-1
        LA   R6,1
        LA   R7,L'C
LOOPI   CR   R6,R7
        BH   ELOOPI
        MVC  0(1,R8),0(R9)
        LA   R8,1(R8)
        BCTR R9,0
        LA   R6,1(R6)
        B    LOOPI
ELOOPI  XPRNT C,L'C
        L    R13,4(0,R13)
        LM   R14,R12,12(R13)
        XR   R15,R15
        BR   R14
C       DC   CL12'edoC attesoR'
TMP     DS   CL12
        YREGS
        END  REVERSE
```

360 Assembly

```
PROGRAM Example

CHARACTER(80) :: str = "This is a string"
CHARACTER :: temp
INTEGER :: i, length

WRITE (*,*) str
length = LEN_TRIM(str)
DO i = 1, length/2
    temp = str(i:i)
    str(i:i) = str(length+1-i:length+1-i)
    str(length+1-i:length+1-i) = temp
END DO
WRITE(*,*) str

END PROGRAM Example
```

Fortran

```
PROC reverse = (REF STRING s)VOID:
  FOR i TO UPB s OVER 2 DO
    CHAR c = s[i];
    s[i] := s[UPB s - i + 1];
    s[UPB s - i + 1] := c
  OD;
```

Algol 68

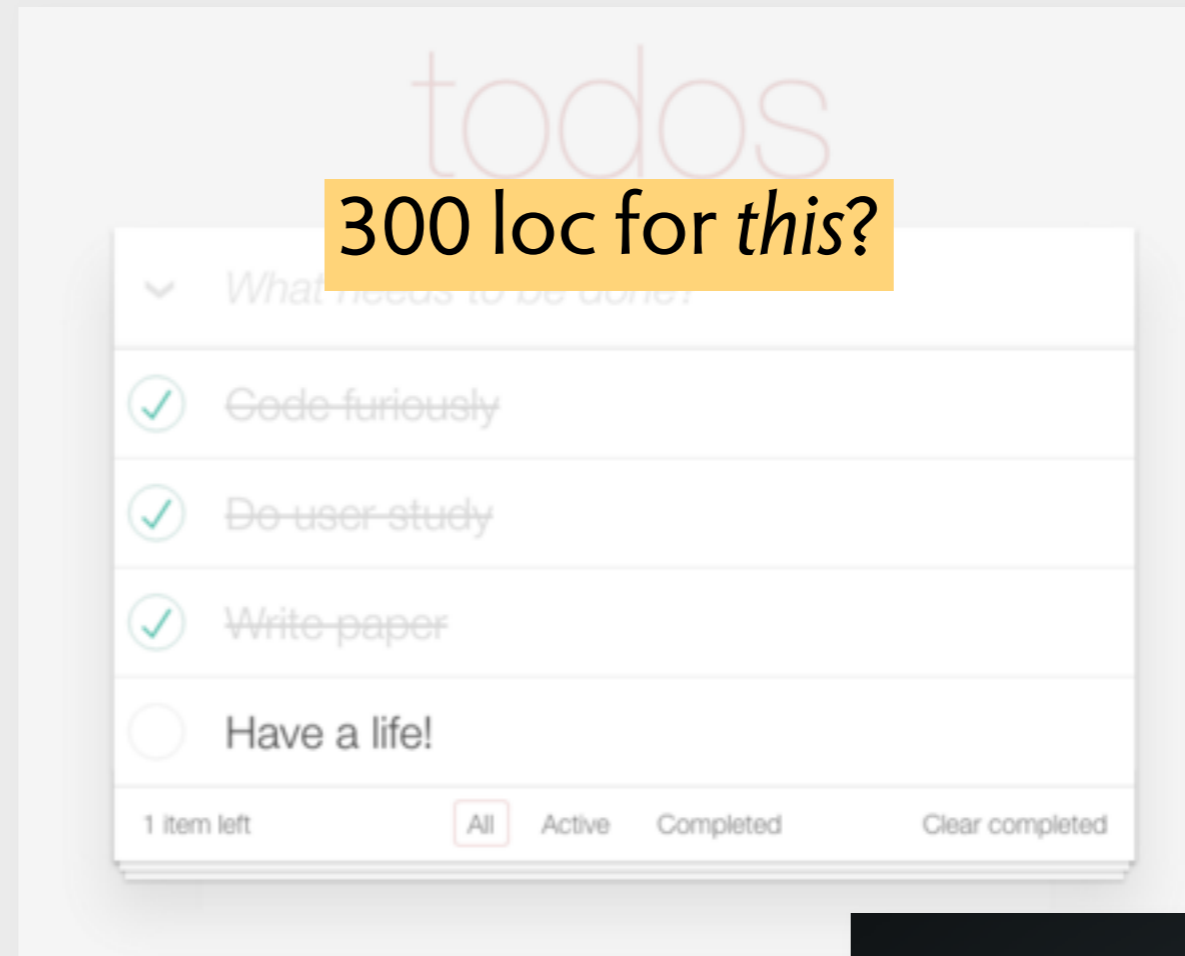
```
reverse = foldl (flip (:)) []
```

Haskell

# a benchmark example



**todomvc.com**  
showcase of MVC  
frameworks

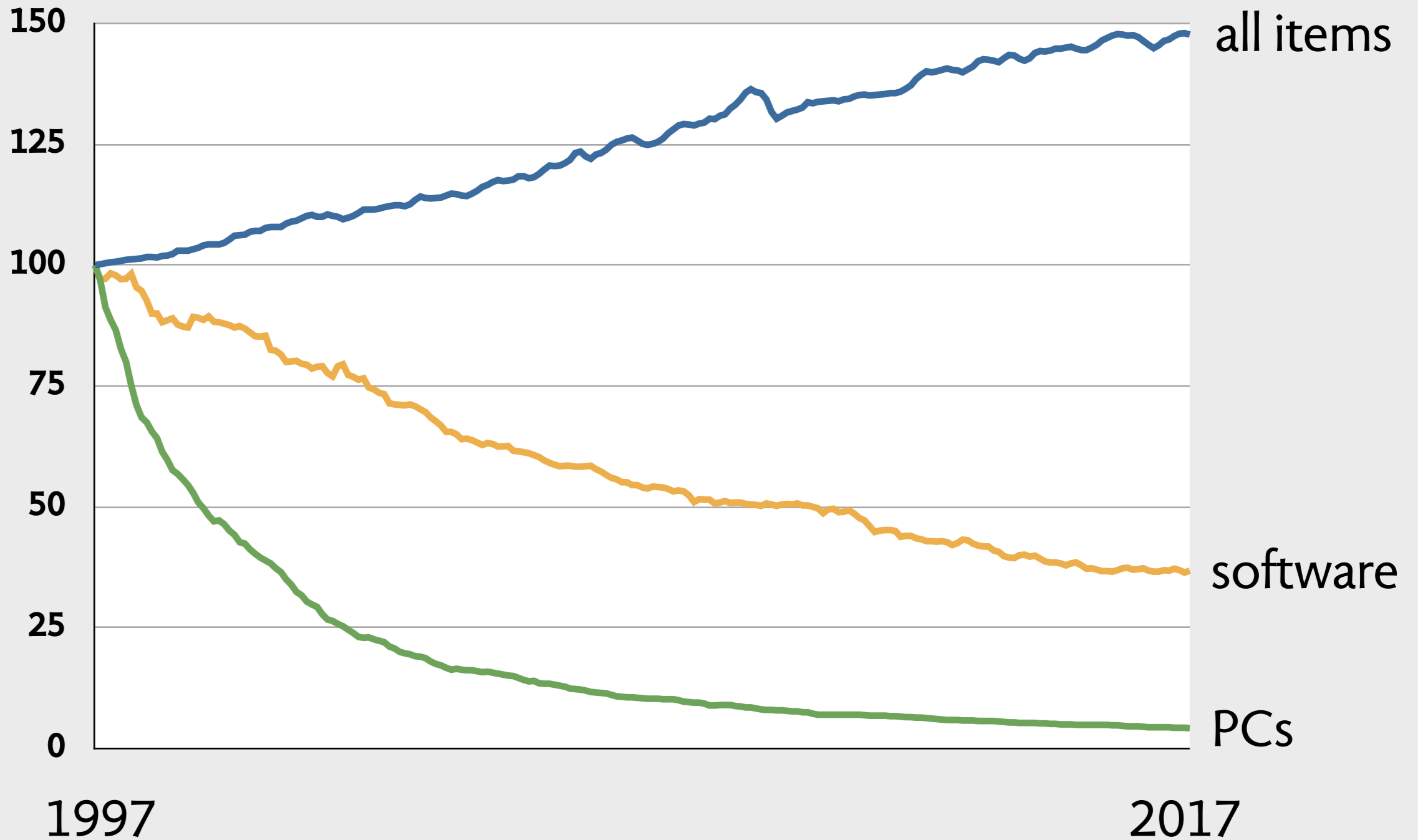


## todomvc.com

Framework	SLOC
Angular	294
Polymer	246
React	421
Backbone	297

pain points

# consumer price index



*from bls.gov*



# costs of standard IT

## **buying off-the-shelf**

may not fit your needs  
paying for unused features

## **hiring developers**

means waiting, maybe years  
costs \$1-\$100/line  
unaffordable for small orgs

## **tweaking the code**

is hard & dangerous  
only by developers

# costs of shadow IT

## why people do it

storage & backup  
sharing and sending files  
hosting small websites

## what goes wrong

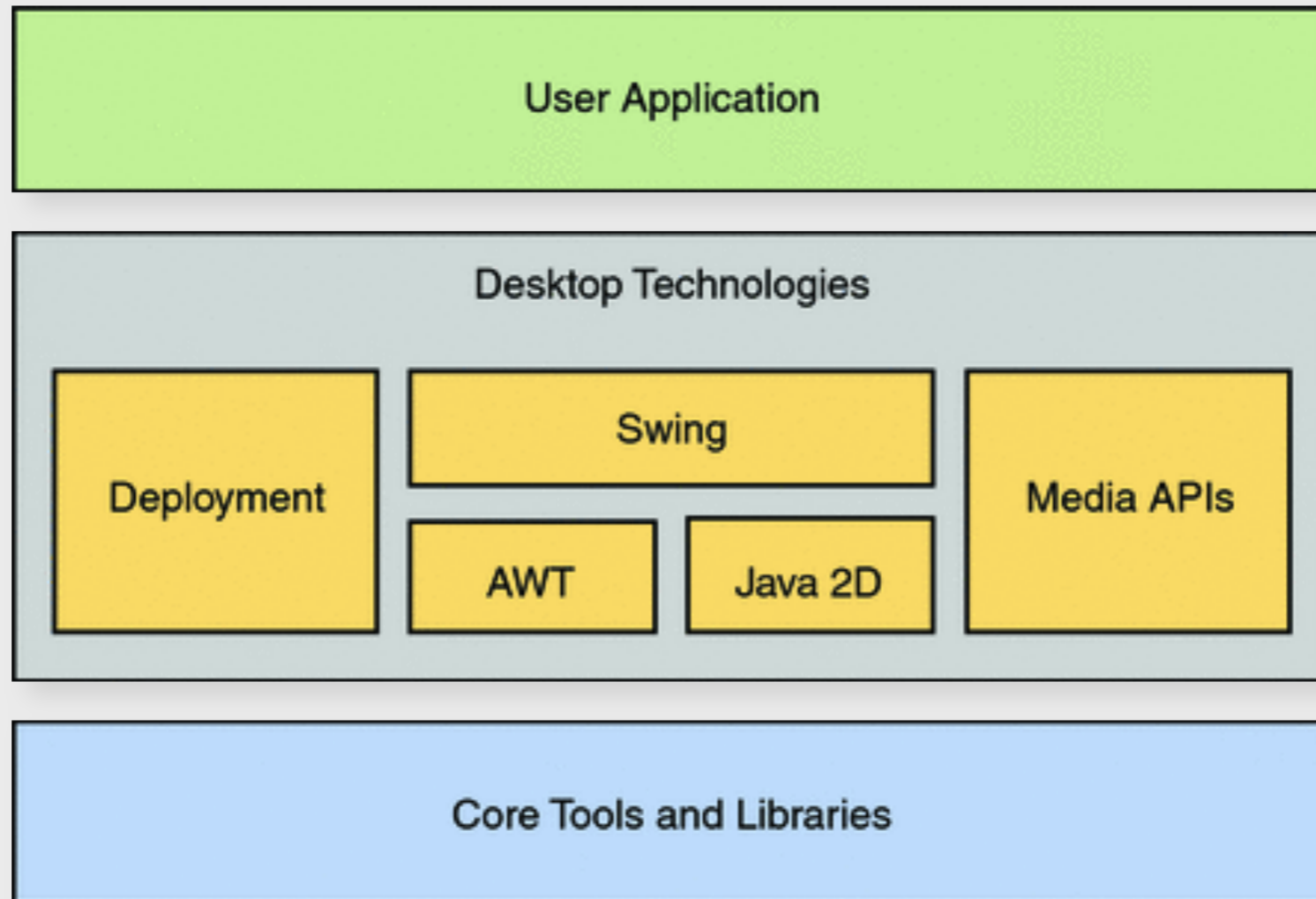
data loss and leaks  
inefficiency & wasted time  
creeping dependences  
non-compliance

2015: teenage hacker breaks into AOL account of CIA director John Brennan, obtaining many government materials including his 47 page application for top secret clearance



how we got here:  
before the web

# monolithic apps



Java SE Desktop

# monolithic apps

```
import java.io.IOException;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Date;
```

```
public class DateServer {

    public static void main(String[] args) throws IOException {
        ServerSocket listener = new ServerSocket(9090);
        try {
            while (true) {
                Socket socket = listener.accept();
                try {
                    PrintWriter out =
                        new PrintWriter(socket.getOutputStream());
                    out.println(new Date().toString());
                } finally {
                    socket.close();
                }
            }
        } finally {
            listener.close();
        }
    }
}
```

uniform,  
explicit & simple  
dependences

statically  
typed  
interfaces

verbose  
language

no  
separation of  
concerns

low level  
details

using network to respond  
to date requests

just one  
language

using AWT to  
display a message

```
package ...
import java.awt.Frame;
import java.awt.Label;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class Hello {

    public static void main(String[] args) {
        Frame f=new Frame("Example of awt application");
        Label label1=new Label("Hello World", Label.CENTER);
        f.add(label1);
        f.setSize(300,100);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        });
    }
}
```

spurious  
ordering

no  
separation of  
concerns

# what the web wanted

## **a wish list**

manipulate small data

use a database backend

interact with a client UI

separate concerns

**but none is easy in Java...**

the web arrives

# manipulate small data

“Java is to JavaScript as ham is to hamster” *Jeremy Keith*

example: flattening a list (from rosettacode.org)

from: `[[1], 2, [[3, 4], 5], [[]], [[[6]]], 7, 8, []]`

to: `[1, 2, 3, 4, 5, 6, 7, 8]`

Java

```
import java.util.LinkedList;
import java.util.List;

public final class FlattenUtil {

    public static List<Object> flatten(List<?> list) {
        List<Object> retVal = new LinkedList<Object>();
        flatten(list, retVal);
        return retVal;
    }

    public static void flatten(List<?> fromTreeList, List<Object> toFlatList) {
        for (Object item : fromTreeList) {
            if (item instanceof List<?>) {
                flatten((List<?>) item, toFlatList);
            } else {
                toFlatList.add(item);
            }
        }
    }
}
```

no need to figure out types

Javascript

```
function flatten(list) {
    return list.reduce(function (acc, val) {
        return acc.concat(val.constructor === Array ? flatten(val) : val);
    }, []);
}
```



# use a database backend

```
sql = "Select * from Users where" +  
      "name = '#{params[:name]}'" +  
      "AND password = '#{params[:password]}'"
```

```
user_array = ActiveRecord::Base.connection.execute(sql)
```

easy  
construction of  
query as a string

Rails raw SQL query

# separate concerns

route  
separated from  
function

```
var counter = 0;

app.get('/show', function (req, res) {
  res.send('Counter value is ' + counter);
});

app.get('/reset', function (req, res) {
  counter = 0;
  res.send('Counter value reset');
});

app.get('/inc', function (req, res) {
  counter++;
  res.send('Counter value incremented');
});
```

```
var users = require('./routes/users');
app.use('/users', users);
```

handlers for  
routes in  
separate files

templates  
separate view  
from control

```
<html><body>
Counter value is {{counter}}
<form action="/inc" method="post">
  <input type="submit" value="inc by">
  <input type="text" name="by"
value="1">
</form>
</body></html>
```

```
var counter = 0;

app.get('/', function (req, res) {
  res.render('index', {counter: counter});
});
```

# interact with a client UI

save name in session

```
app.use(session({ secret : 'foo', resave : true,
saveUninitialized : true }));

app.get('/:name', function (req, res) {
  req.session.name = req.params.name;
  res.send('Hello ' + req.params.name);
});

app.get('/', function (req, res) {
  res.send('Welcome back ' + req.session.name);
});
```

session structure hides cookies + db

return name from session

a server that remembers your name

so are we  
happy now?

# JavaScript

“the duct tape of the Internet”

```
> 1 + "hello"
"1hello"

> 1 / "hello"
NaN

> x = 1 / "hello"

> (x == NaN) ? "bad" : "good"
"good"

> typeof(NaN)
"number"

> NaN === NaN
false

> NaN !== NaN
true
```

# coming of age

If Javascript were a person, what life event would it be going through this year?

- a. Getting a driver's license (16)
- b. Voting in its first election (18)
- c. Buying its first legal drink (21)
- d. Getting a discount on car insurance (25)

# JS

*and the answer is...*

# database complexities

embedded SQL: not a good match for objects

```
sql = "Select * from Users where" +  
      "name = '#{params[:name]}'" +  
      "AND password = '#{params[:password]}'"  
user_array = ActiveRecord::Base.connection.execute(sql)
```

using an “object relational mapper”

```
user = User.where(  
  "name = '#{params[:name]}'" +  
  "AND password = '#{params[:password]}'" )
```

(and both still have an injection vulnerability)

# endless frameworks



“No JavaScript frameworks were created during the writing of this article”

from: *How it feels to learn JavaScript in 2016*



# callback hell

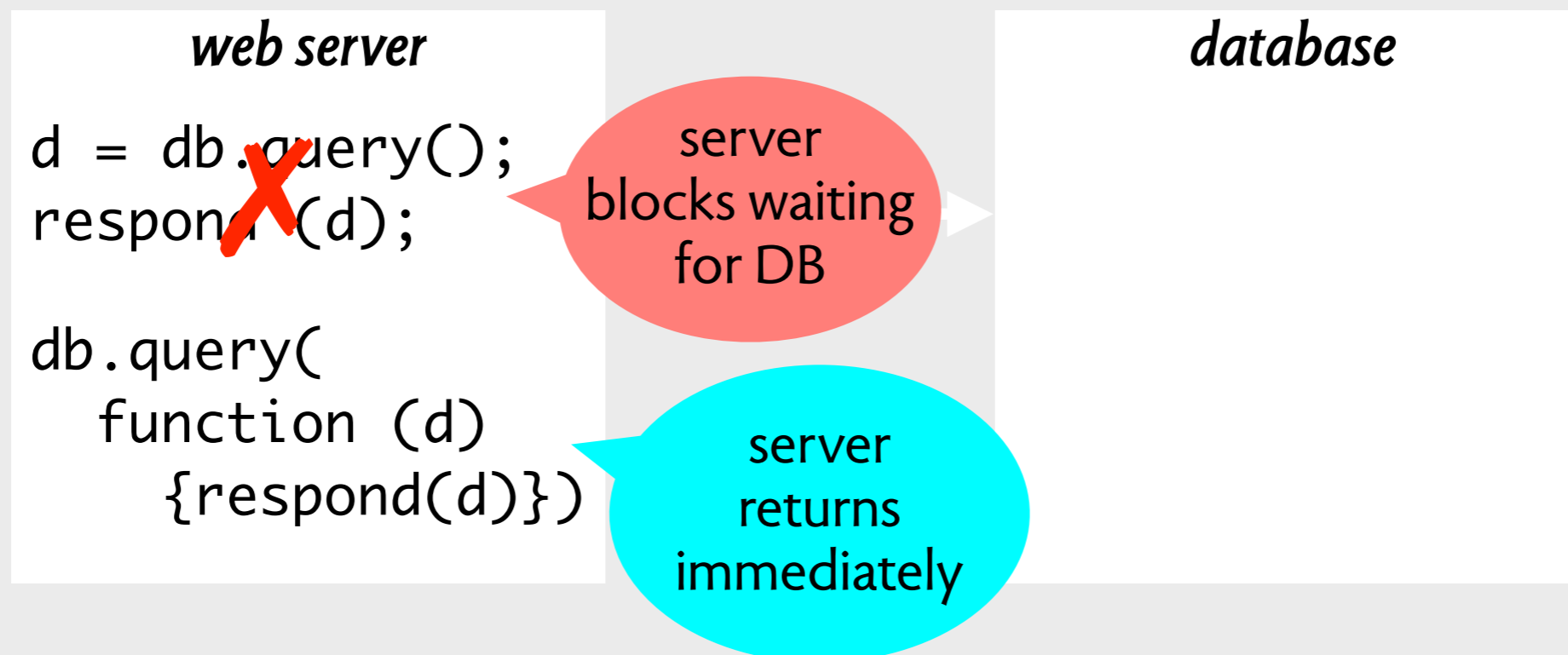
what we used to write

```
do_a();  
do_b();  
do_c();
```

what we write now

```
do_a (function () {  
  do_b (function () {  
    do_c(...)  
  })  
})
```

what inspired this madness?



“low code”  
to the rescue

# been there, done that?

Though hard to describe in words, **???** comes alive visually. In minutes, people who have never used a computer are writing and using programs. Although you are operating in plain English, the program is being executed in machine language. But as far as you're concerned, the entire procedure is software transparent. You simply write on this so-called electronic blackboard what you would like it to do -- and it does it.

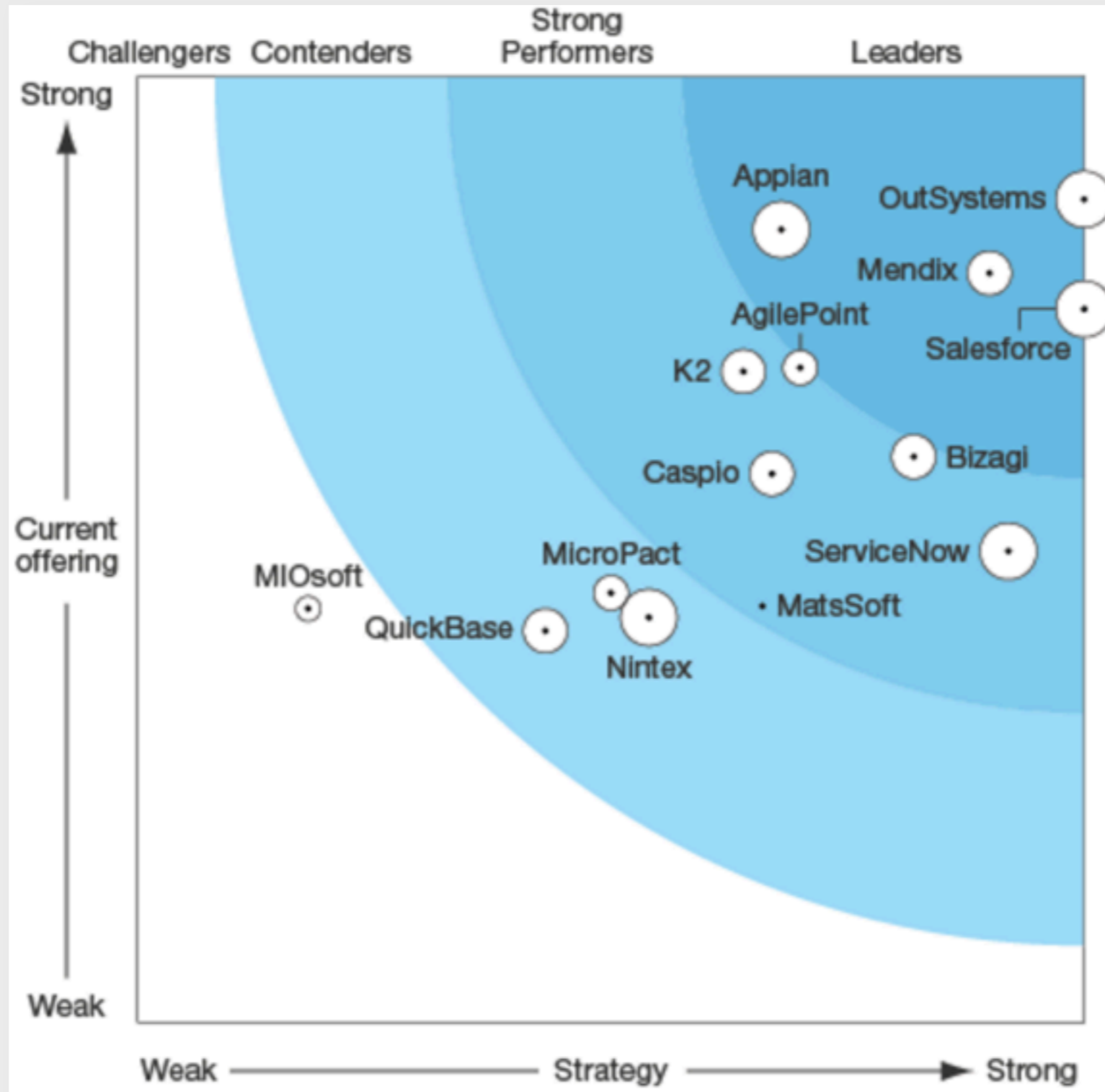
Ben Rosen, Morgan Stanley Electronics Letter (1979)

013 (V) +03-813+C13 CI  
27

	A	B	C	D
1	PAYEE	CHECKS	DEPOSITS	BALANCE
2				545.20
3				
4				
5	ELECTRIC	14.95		
6	OIL	102.15		
7	PHONE	36.80		
8	DENTIST	42.00		
9	SALARY		395.00	
10	RENT	350.00		
11	GAS CARD	12.93		
12				
13	TOTALS	558.83	395.00	381.37
14				
15				
16				
17				
18				
19				
20				

# “low code” platforms

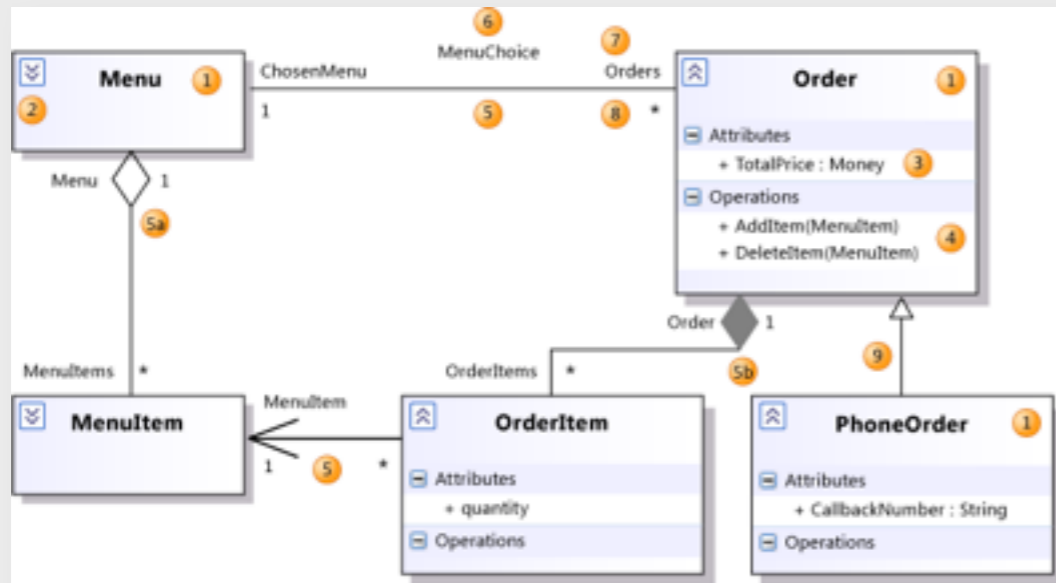
term coined by Forrester Research in 2014



Forrester:  
market for  
low-code dev  
will be \$15.5B  
by 2020

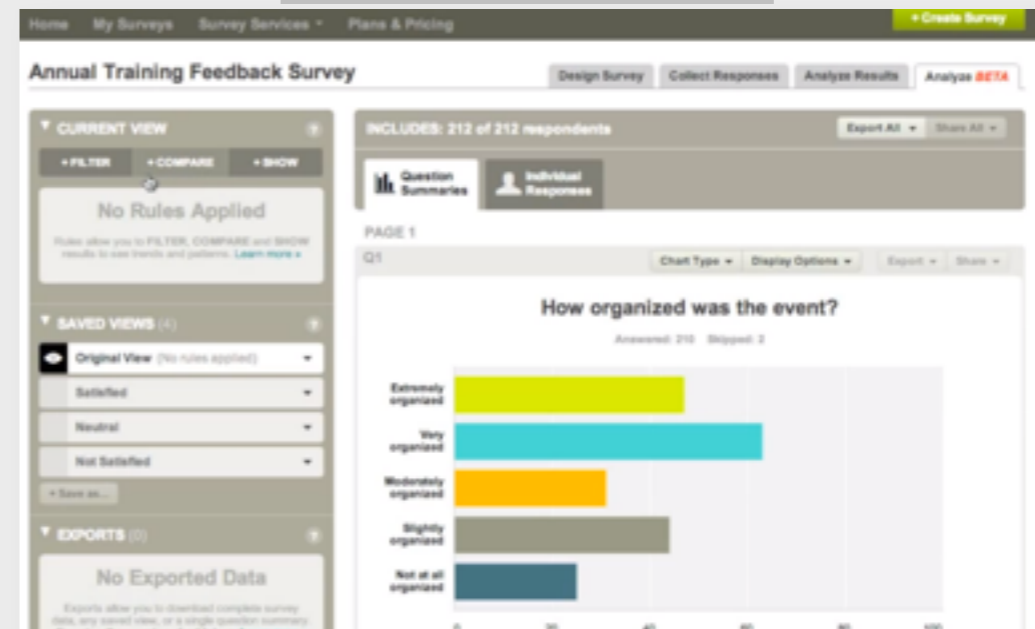
# technology origins

model-driven development



UML class diagram, Visual Studio (OMT, 1991)

form builders



Survey Monkey (founded 1999)

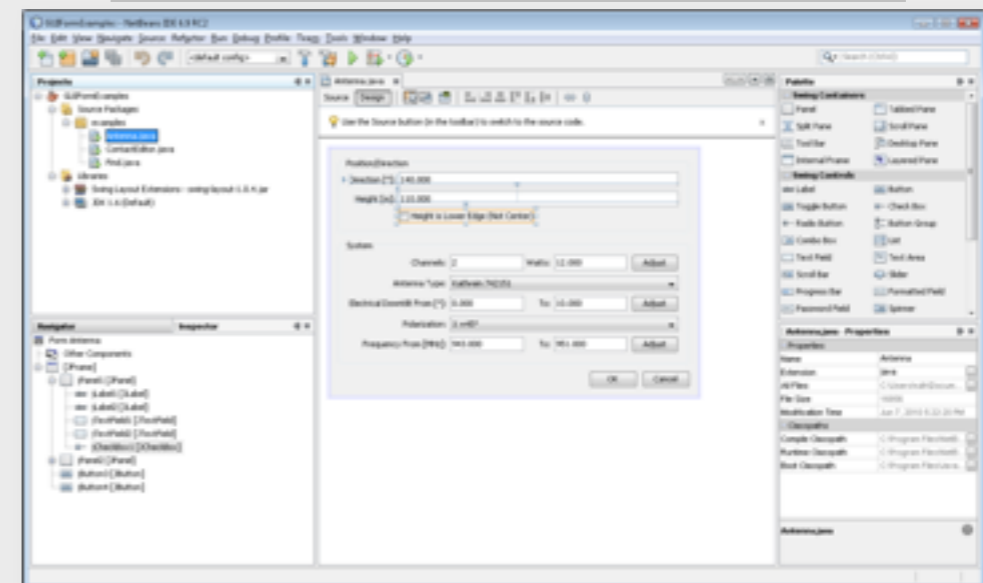
groupware databases

The screenshot shows the Microsoft Access interface with a table of album information. The table has columns for AlbumName, ReleaseDate, and ArtistName. The data is as follows:

AlbumName	ReleaseDate	ArtistName
PowerSlave	3/3/1984	Iron Maiden
Powerage	5/5/1978	AC/DC
Crimes of Passion	8/5/1980	Pat Benatar
Bitches Brew	3/30/1970	Miles Davis
Kind of Blue	8/17/1959	Miles Davis
Couldn't Stand the Weather	5/15/1984	Stevie Ray Vaughan
Somewhere in Time	9/29/1986	Iron Maiden
Piece of Mind	5/16/1983	Iron Maiden
Killers	2/2/1981	Iron Maiden
No Prayer for the Dying	10/1/1990	Iron Maiden
Texas Flood	6/13/1983	Stevie Ray Vaughan
Snoopified	9/28/2005	Snoop Dogg
The Doggfather	11/12/1996	Snoop Dogg
Hail to the King	8/23/2013	Avenged Sevenfold
Destiny Fulfilled	11/10/2004	Destiny's Child
Bush	5/12/2015	Snoop Dogg
The Book of Souls	5/4/2015	Iron Maiden
Coolaid	7/5/2016	Snoop Dogg
Black Ice	10/17/2008	AC/DC
Live Songs	1/29/2013	Destiny's Child
In a Silent Way	6/30/1969	Miles Davis
Sounding the Seventh Trumpet	1/21/2001	Avenged Sevenfold

Microsoft Access (first release 1992)

user interface builders



Netbeans GUI Builder (Project Matisse, 2008)

ingredients

# visual editing

The image shows a screenshot of Zoho's form builder interface. The top navigation bar includes 'Home', 'Builder', 'Rules', 'Settings', 'Themes', 'Share', and 'Integration'. The main area is titled 'Join Us' and contains a form with the following fields:

- Header: 'Join Us' (highlighted in yellow)
- Text: 'Join Us to change the world!' (highlighted in yellow)
- Form Fields:
  - First Name (input field)
  - Last Name (input field)
  - Single Line (input field, highlighted with a dashed box)
  - Email (input field)
  - Single Line (input field)
  - Address (input field, labeled 'Street Address')
  - Address Line 2 (input field)

The left sidebar, titled 'BASIC FIELDS', lists various field types:

- Single Line
- Multi Line
- 123 Number
- 00 Decimal
- Name
- Address
- Phone
- Email
- Time
- Radio
- Multiple Choice
- Checkbox
- Decision Box
- Date-Time
- Website
- Currency
- Section
- File Upload
- Payment

Below the sidebar is the 'ADVANCED FIELDS' section.

*direct manipulation*

*recognition, not recall*

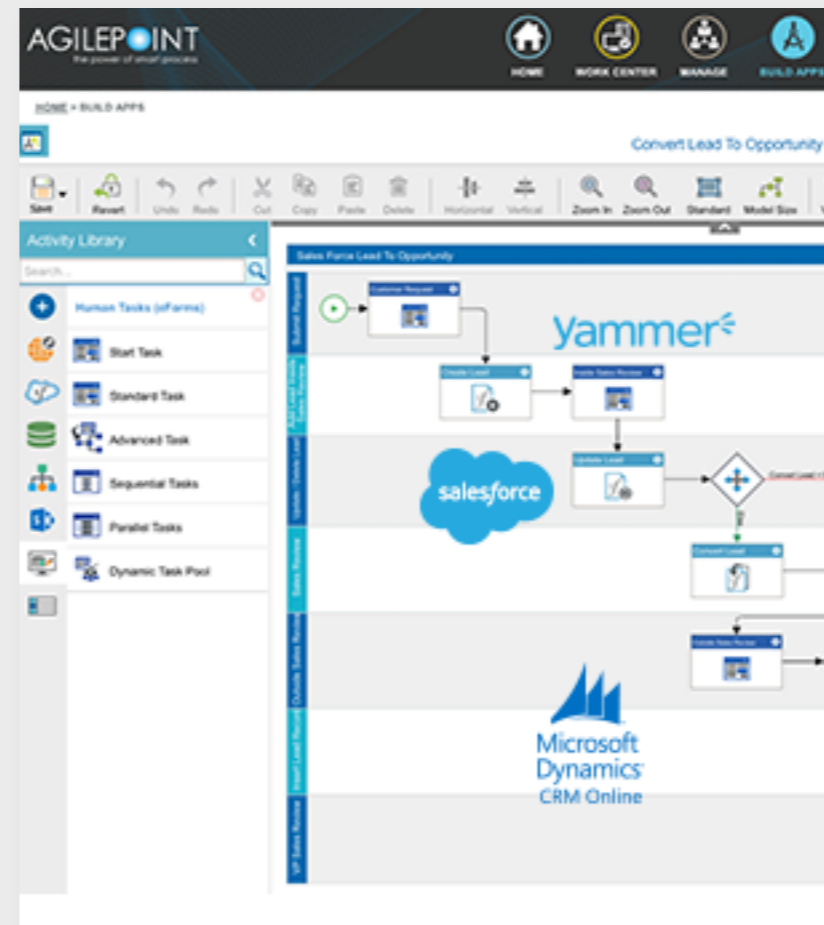
Zoho's form builder

# abstractions



	Date	Region	Product Category	Product	Customer Name	Sales	Cost
1	02 Apr, 2010	West	Grocery	Fruits and Vegetables	Vincent Herbst	\$1,682.39	\$250.05
2	04 Apr, 2010	East	Furniture	Chairs	John Brito	\$272.34	\$14.58
3	07 Apr, 2010	West	Grocery	Fruits and Vegetables	David Flushing	\$2,970.27	\$1,635.85
4	09 Apr, 2010	East	Stationery	File Labels	Maxwell Schwarz	\$190.05	\$90.85
5	12 Apr, 2010	West	Grocery	Fruits and Vegetables	Leta Donovan	\$5,342.57	\$1,829.65
6	14 Apr, 2010	East	Stationery	Art Supplies	Susan Juliet	\$45.31	\$12.93
7	16 Apr, 2010	East	Grocery	Fruits and Vegetables	Carl Lewis	\$2,974.81	\$986.08
8	18 Apr, 2010	East	Stationery	Specialty Envelopes	Pete Zachriah	\$455.08	\$195.66
9	19 Apr, 2010	West	Grocery	Fruits and Vegetables	Andy Roddick	\$3,328.38	\$1,386.98
10	21 Apr, 2010	West	Stationery	Copy Paper	Venus Powell	\$409.51	\$40.92
11	23 Apr, 2010	East	Stationery	Computer Paper	Pete Zachriah	\$27.89	\$8.51
12	24 Apr, 2010	East	Grocery	Fruits and Vegetables	Hilary Hudson	\$955.88	\$573.23
13	25 Apr, 2010	West	Stationery	Highlighters	Joseph Aaron	\$37.48	\$0.13
14	25 Apr, 2010	East	Stationery	Standard Labels	Patrick O'Sullivan	\$125.84	

“form-tables”



- ▼ Slack tab
  - Archive Channel activity
  - Create Channel activity
  - Create Child Group activity
  - Delete Chat activity
  - Get User's Presence activity
  - Invite To Channel activity
  - Join Channel activity
  - Kick Channel activity
  - Post Chat Message activity
  - Post Chat Me Message activity
  - Update Chat activity
  - UnArchive Channel activity
  - Users Info activity

activities

NAME	DESCRIPTION	EXECUTE ON
<a href="#">Usability Rule</a>	Assign UI related issues to Jasmine Frank	Field update <a href="#">Deactivate</a>
<a href="#">On Close Rule</a>	This rule will be invoked when the bug is clo:	Field update <a href="#">Activate</a>
<a href="#">Trigger Business rule</a>	This business rule will get triggered whenever	Bugs Creation <a href="#">Deactivate</a>

rules



# declarative code

### 1. Basic Details

---

Rule Name

### 2. Execute On

---

Add  
While adding a new entry by this form



### 3. Criteria

---

All Records  Selected Records

### 4. Associate Tasks

---

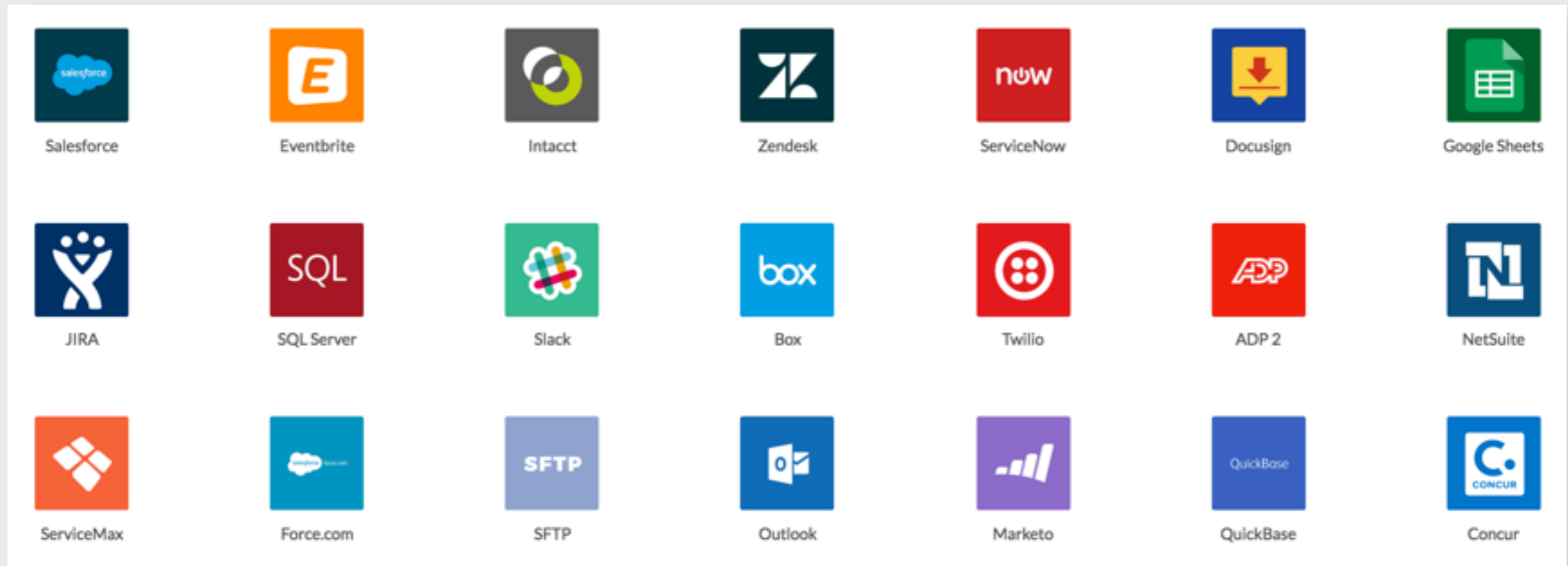
 Field Tasks	<input type="button" value="+"/> <input type="button" value="Folder"/>	>
 Notifications	<input type="button" value="+"/> <input type="button" value="Folder"/>	>

Task Name

Choose Task

Zoho's rules & tasks

# integration



Workato's integrated backends

cloud recipes: Zapier, IFTTT, Workato

mobile apps: Appery.io, SkyGiraffe, Appian QuickApps

# easy deployment

The screenshot displays the Mendix deployment interface. At the top, the Mendix logo is on the left, and a green button labeled "CREATE NEW APP +" is on the right. Below the logo, a navigation menu includes "Overview", "Capture", "Develop", "Deploy" (highlighted in blue), "Publish", "Monitor", and "Feedback".

The main content area is divided into three tabs: "Deploy" (selected), "Backup", and "Custom Domain Certificates". Under the "Deploy" tab, the following configuration details are shown:

Runtime version	7.0.0
Administrator user name	MxAdmin
Region	Mendix Cloud EU

Below the configuration details is the "Scaling" section, which includes two sliders and a circular progress indicator:

- Instances:** A slider set to 2 / 8.
- Memory (MB):** A slider set to 512 / 2048.
- Total allocated memory:** A circular progress indicator showing 1024 MB from 4096 MB.

A green "Scale now" button is located at the bottom of the scaling section. On the right side of the interface, there is a vertical button labeled "Feedback on Mendix".

Mendix's AWS-based deployment

not yet in  
paradise

# an example: hackathon Q

005q

There are 3 people in the queue!

My name is \_\_\_\_\_ name \_\_\_\_\_ and I need help with \_\_\_\_\_ something! \_\_\_\_\_  
\_\_\_\_\_ where are you? \_\_\_\_\_

HELP ME!

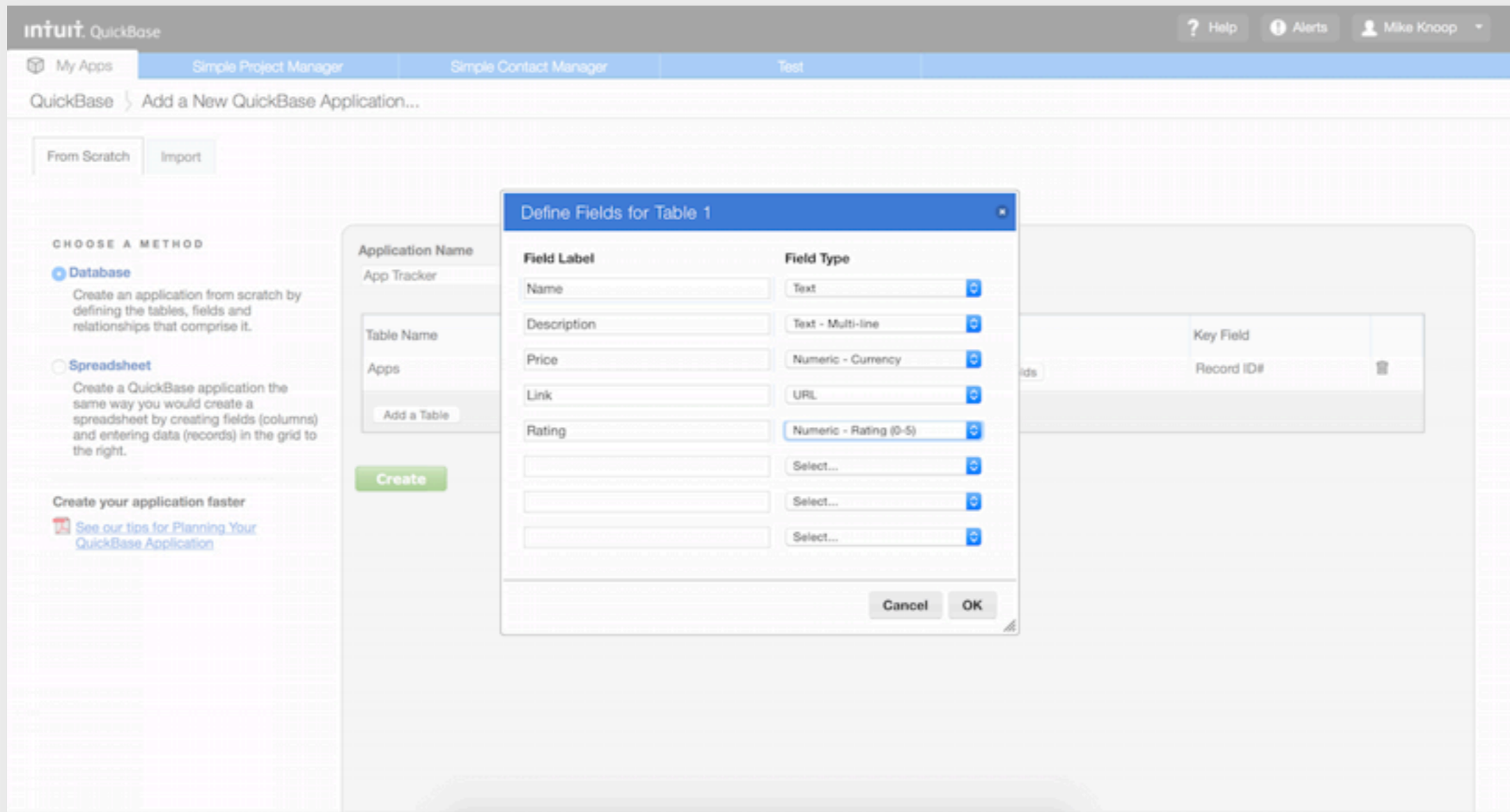
Currently in the queue:

- Angelina
- Myrtle
- Neville

mentors register skills  
participants request a skill  
mentor assigns calls

**tables:** skills, mentors, calls  
**report:** calls active/assigned  
**forms:** request, offer, assign

# Quickbase



**tricky:** had to provide reverse mapping from skills to mentors  
**not possible:** allow requests for more than one skill  
**needed custom code:** assign call to current user

# what's going on?

## **form-table approach**

ad hoc query language

no underlying calculus

## **some basic app features**

may require custom code

or not be expressible at all

## **biggest problem?**


hard to predict until you try

# encountering limitations

Milk Pool ▾ | | ▾    📱 ▾    ⚙️ ▾

Email

Stock



**let's make a rule**  
if user reports out of milk  
then send message



# not uniformly visual

## 1. Basic Details

Rule Name

Milk low

## 3. Criteria

All Records  Selected Records

must recall  
field values

bad value,  
no warning

Stock



equals

































"low"



# not uniformly expressive

## 4. Associate Tasks

Field Tasks	+ 	>
Notifications	+ 	∨
Task Name	<input type="text" value="Milk reminder"/>	
Choose Task	<input type="text" value="Email Notification"/> ∨	
From	<input type="text" value="{zoho.adminuserid}"/>	
To	<input type="text" value="{Email},"/>	
Subject	<input type="text" value="Subject"/>	
Message	<div><p><b>B</b> <i>I</i> <u>U</u> abc              </p><p>              <input type="text" value="Insert Fields"/> ∨</p></div>	

can use variable here

but not here

# welcome back, VBA?

## All Functions

The following table is a searchable listing of all Appian functions.

- Functions in this table are sorted by *category*, *sub-category*, then function *name*.
- By default, this table shows the function name, and an example where available. You can use the *column* specific columns.
- You have *filtering options* along the top-right side of the table, where you can filter the table by function filter on function name and category.
- For more detailed information about a particular function, click the function name to go to its page.

DESCRIPTION

SYNTAX

EXAMPLE

RESULT

COLLAPSE ALL

Name	Example
<b>▼ Array</b> Used within your expressions to manipulate, insert, and/or select values from arrays.	
<a href="#">append()</a>	<code>append({10, 20, 30}, 99)</code>
<a href="#">index()</a>	<code>index({10, 20, 30}, 2, 1)</code>
<a href="#">insert()</a>	<code>insert({10, 20, 30, 40}, 99, 1)</code>
<a href="#">joinarray()</a>	<code>joinarray({1, 2, 3, 4}, " ")</code>
<a href="#">ldrop()</a>	<code>ldrop({10, 20, 30}, 1)</code>

Appian function API

some new  
research

# two research projects

## **aim**

new approach to low-code

## **target**

“community applications”  
too complex for Drupal  
too simple for full stack

## **approach**

language first, wizards later  
declarative & expressive



# a new layout language

**HTML**



**CSS**



written by non-programmers  
flexible visual design  
cross-platform & responsive  
rich hierarchical data model

HTML only for data instances  
can't describe schemas  
need JavaScript to read/write

# mavo

by Lea Verou & David Karger

**HTML**



**CSS**

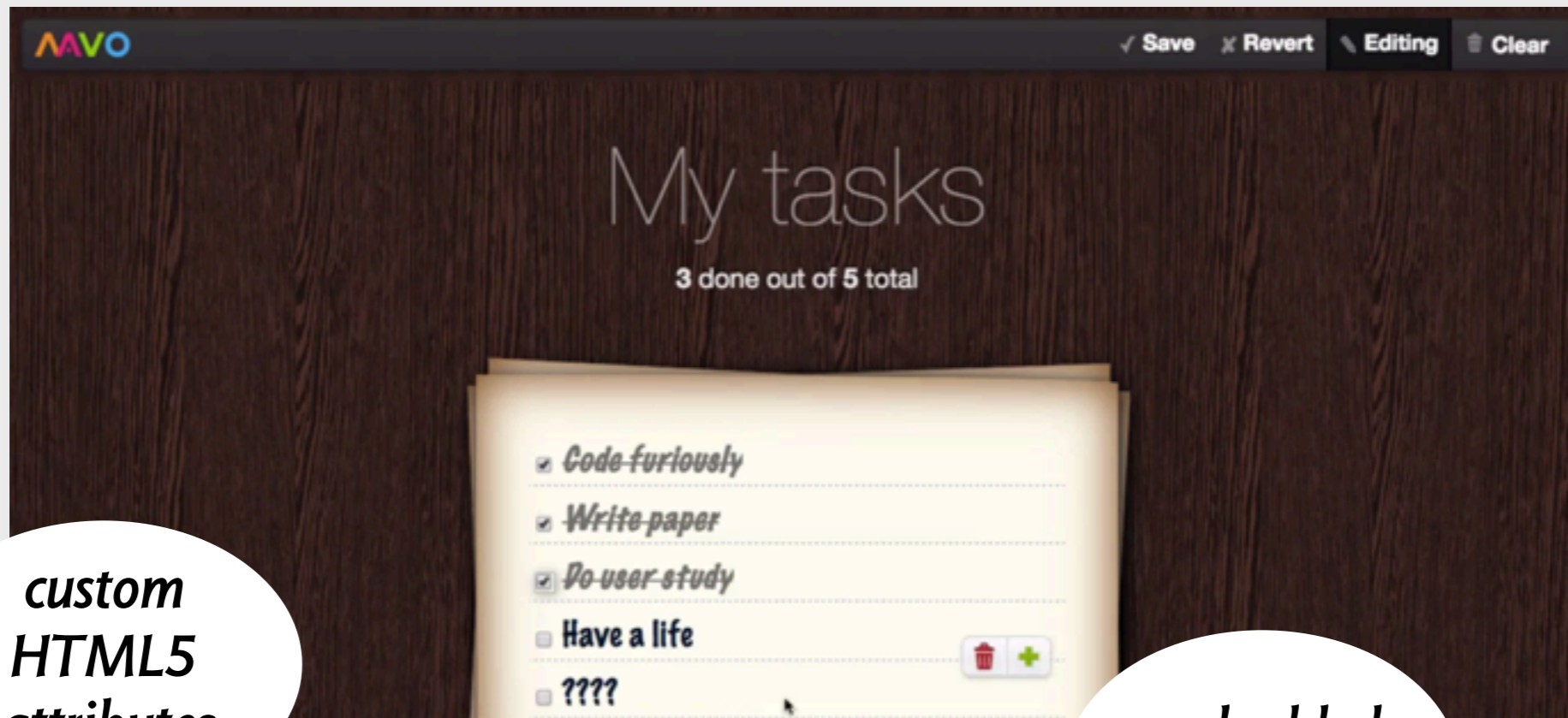


written by non-programmers  
flexible visual design  
cross-platform & responsive  
rich hierarchical data model

use HTML instance as schema  
make elements editable  
read/write to server for free

# to-do in Mavo

the  
app



*custom  
HTML5  
attributes*

*embedded  
formula*

the  
code

```
<body data-store="local">
  <h1>My tasks</h1>
  <p>[count(done)] done of [count(task)] total</p>
  <ul>
    <li property="task" data-multiple>
      <input property="done" type="checkbox" />
      <span property="taskTitle">Do stuff</span>
    </li>
  </ul>
</body>
```

*data  
becomes  
schema*

*implicit  
editing  
controls*



# a new data store model



## spreadsheet

intuitive visual layout  
schema evolves with data  
can see all the data

numeric queries only  
can't handle nested data  
risky to insert/delete rows

## relational database

rich query language  
can encode structured data  
easy to insert/delete tuples

not intuitive to end users  
hard to evolve schema  
seeing data needs queries



# object sheets

intuitive visual layout  
schema evolves with data  
can see all the data

rich query language  
can encode structured data  
easy to insert/delete tuples

## **challenges**

a new data model  
a new query language  
connection to clients

# example: allocating offices

Room	Sq. footage	Occupant	Role
Dungeon Five	480	Sirius	Grad. student
		James	Post-doc
		Wormtail	Grad. student
Greenhouse Two	561	Bellatrix	Visiting Prof.
		Lily	Post-doc
		Remus	Post-doc

**nested  
objects**

**object  
references**

Role	Allocated space
Grad. student	12
Post-doc	20
Visiting Prof.	45

# in google spreadsheet

<i>fx</i>	A	B	C	D	E	F
1	room	sq. footage	occupant	role	alloc. space	free
2	Dungeon Five	480	Sirius	Grad. student	12	436
3			James	Post-doc	20	
4			Wormtail	Grad. student	12	
5			Harry	Grad. student	12	
6	Greenhouse Two	561	Bellatrix	Visiting Prof.	45	476
7			Lily	Post-doc	20	
8			Remus	Post-doc	20	
9	role	alloc. space				
10	Grad. student	12			=VLOOKUP(D2, Sheet1!A10:B12,	
11	Post-doc	20			=B2 - SUM(E2:E4)	
12	Visiting Prof.	45			=B6 - SUM(E6:E8)	
13						

*but nesting is  
only visual, not  
computational*

*formulas are  
complex*

*formulas are  
unstable*

# in object sheets

*formulas over sets  
(now stable)*

```
sqFoot - sum[ o : Occupant ]( o.role.allocSpace )
```

Room			Occupant			Role			
	name	sqFoot		name	role	free	title	allocSpace	
•	text	number	•	text	Role	number	•	text	number
•	Dungeon Five	480	•	Sirius	<u>Grad. student</u>	436	•	Grad. student	12
			•	James	<u>Post-doc</u>		•	Post-doc	20
			•	Wormtail	<u>Grad. student</u>		•	Visiting Prof.	45
•	Greenhouse Two	561	•	Bellatrix	<u>Visiting Prof.</u>	476			
			•	Lily	<u>Post-doc</u>				
			•	Remus	<u>Post-doc</u>				

*nesting is now  
semantic, not just  
visual*

*first-class object  
references*

# a parent-teacher app

MVO ✓ Save ✕ Revert

[Back to login](#)

## Parent view for Molly

- Ronald
  - Potions with Snape: 2014-12-16 13:45 ▾
  - Defence with Snape: ▾
- Ginevra
  - Potions with Snape: ▾
  - Charms with Flitwick: 2014-12-17 13:00 ▾

what parent sees

MVO Editing ✓ Save ✕ Revert 🗑 Clear

[Back to login](#)

## Teacher view for Flitwick

Slot time	Scheduled meeting
2014-12-17 13:00	Ginevra in 6.005
2014-12-17 14:00	

Add slot

what teacher sees

MVO Edit ✓ Save ✕ Revert 🗑 Clear

	2014-12-16	2014-12-17
13:00	Augustus / Snape	Molly / Flitwick
13:15		
13:30		
13:45	Molly / Snape	
14:00		

what principal sees

# the backend

Person				Slot		
name	roles	parents	time	scheduledEnrollment	valid	
• text	text	Person	• text	Enrollment	bool	
• Snape	teacher		• 2014-12-16 13:00	<u>Seamus in 6.820</u>	true	
•			• 2014-12-16 13:30		true	
•			• 2014-12-16 13:45	<u>Ronald in 6.170</u>	true	
• Flitwick	teacher		• 2014-12-16 13:00	<u>Ginevra in 6.005</u>	true	
•			• 2014-12-16 14:00		true	
• Ronald	student	<u>Molly</u>				
• Ginevra	student	<u>Molly</u>				
• Seamus	student	<u>Augustus</u>				
• Molly						
• Augustus						

Class		Section	Enrollment			
code	name	teacher	student	scheduledSlot	valid	
• text	text	• Person	• Person	Slot	bool	
• 6.170	Potions	• <u>Snape</u>	• <u>Ronald</u>	<u>Snape @ 2014-12-16 13:45</u>	true	
•		•	• <u>Ginevra</u>		true	
• 6.820	Defence	• <u>Snape</u>	• <u>Ronald</u>		true	
•		•	• <u>Seamus</u>	<u>Snape @ 2014-12-16 13:00</u>	true	
• 6.005	Charms	• <u>Flitwick</u>	• <u>Ginevra</u>	<u>Flitwick @ 2014-12-16 13:00</u>	true	
•		•	• <u>Seamus</u>		true	

`scheduledEnrollment.Section.teacher = Person`

“the teacher of the section of this enrollment is the person for this slot”

# code reductions for some apps

Application	Original Implementation		Objsheets+Mavo			
	Language/Framework	Code (LoC)	HTML (LoC)	Formulas (count)	Macros (LoC)	HTML (LoC)
<i>PTC</i>	(unavailable to us)			20	29	73
<i>TodoMVC</i>	JavaScript/Angular	113	75	6	5	22
<i>Conf</i>	Python/Django	694	399	30	62	154
<i>HackQ</i>	JavaScript/Meteor	158	142	2	13	110
<i>Got Milk</i>	Perl/CGI	188	40	2	15	26



conclusions

# pros and cons

**bringing together**  
visual GUI building  
model-driven dev  
declarative queries

**mature platforms**  
especially larger players

**integration**  
web service APIs  
enterprise backends

**easy ramp-up**  
simple things are simple  
but gets harder fast

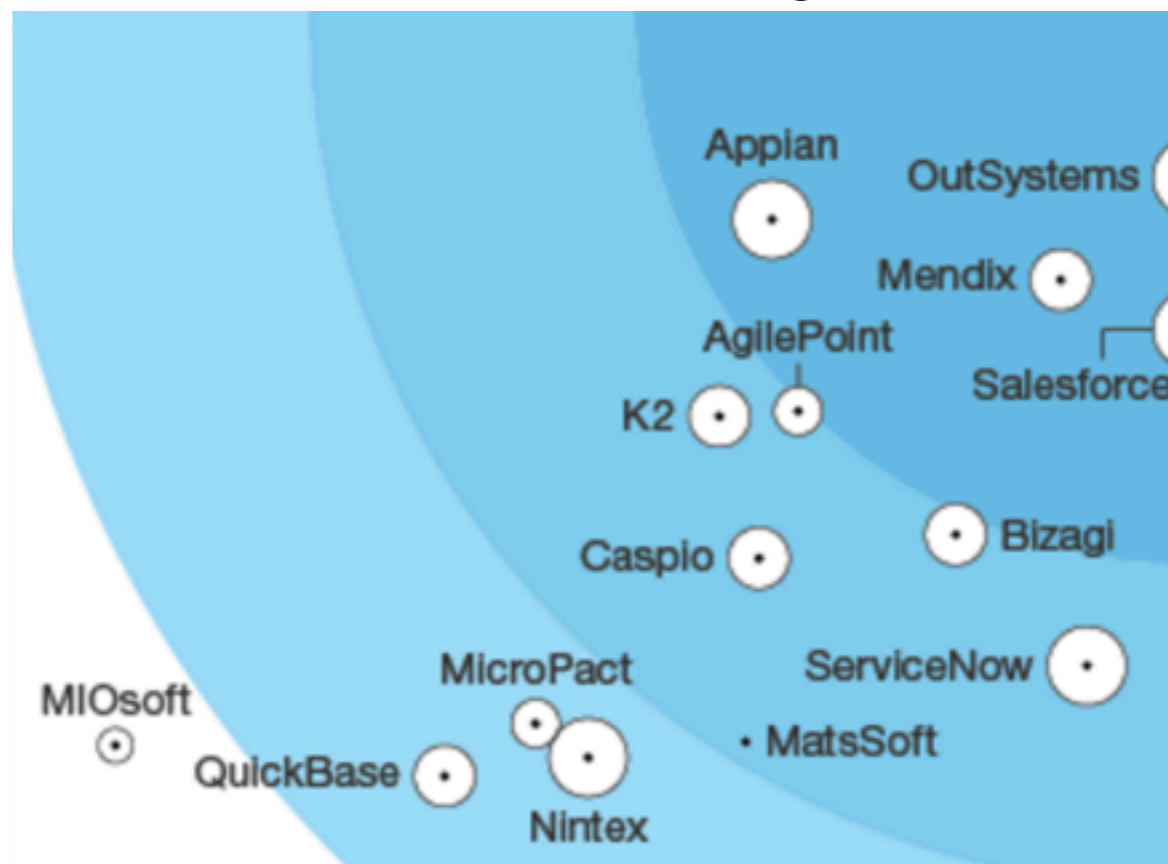
**technology limitations**  
ad hoc limitations  
query language  $\ll$  SQL  
non-declarative scripts

**sidesteps standard tools**  
automated testing  
version control  
even collaboration

**talent shortage**  
may be hard to hire  
few resources online

rails	272674
spring	108989
excel-vba	60557
meteor	25549
sap	3356
abap	1316
servicenow	309
nintex	125
k2	115
outsystems	80
quickbase	58
mendix	8
bizagi	8
appian	4
salesforce app cloud	4
agilepoint	2
caspio	0
matssoft	0
micropact	0
microsoft	0

stackoverflow tags for:



# questions for discussion

**diagrams better than text?**

what about sharing?

**what class of apps are low code platforms suited to?**

“community apps”? enterprise CRUD apps?

**what’s the impact on shadow IT?**

opportunity for coordinated citizen development?

slides, papers, links at: **[tiny.cc/lowcode](https://tiny.cc/lowcode)**