the growth of computational power
more transistors

predicted (1965)

actual (2010)
reversing a string, 1960-1990

**Algol 68**

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

**360 Assembly**

```
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
VREGS
END REVERSE
```

**Fortran**

```
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
```

**Haskell**

```
reverse = foldl (flip (:)) []
```
a benchmark example
todomvc.com showcase of MVC frameworks

300 loc for this?
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

User Application

Desktop Technologies
  - Deployment
  - Swing
  - AWT
  - Java 2D
  - Media APIs

Core Tools and Libraries

Java SE Desktop
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(), true);
                    out.println(new Date().toString());
                } finally {socket.close();}
            }
        } finally {listener.close();}
    }
}

package awt;
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("Hello World 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);
            }
        });
    }
}
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]
use a database backend

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

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

Rails raw SQL query
**route separated from function**

```javascript
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);
```

**templates separate view from control**

```html
<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>
```

```javascript
var counter = 0;
app.get('/', function (req, res) {
    res.render('index', {counter: counter});
});
```
`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);
});`

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
```
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)

and the answer is...
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?

**web server**

d = db.query();
respond(d);

**database**

db.query(
  function (d) {
    respond(d)
  })

server blocks waiting for DB

server returns immediately
“low code”
to the rescue
Though hard to describe in words, it 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)
“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

Microsoft Access (first release 1992)

user interface builders

Netbeans GUI Builder (Project Matisse, 2008)
ingredients
visual editing

Zoho’s form builder

direct manipulation

recognition, not recall
abstractions

activities

“form-tables”

rules
declarative code

1. Basic Details
   Rule Name: Urgency rule

2. Execute On
   Add
   While adding a new entry by this form

3. Criteria
   Selected Records
   Urgency equals "pretty urgent"

4. Associate Tasks
   Field Tasks
   Notifications
   Task Name: Send service call
   Choose Task: SMS Notification

Zoho’s rules & tasks
integration

Workato’s integrated backends

cloud recipes: Zapier, IFTTT, Workato

mobile apps: Appery.io, SkyGiraffe, Appian QuickApps
easy deployment

Mendix’s AWS-based deployment
not yet in paradise
an example: hackathon Q

My name is [name] and I need help with [something] in your area. Where are you?

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

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

Stock equals "low"

must recall field values
bad value, no warning
not uniformly expressive

### 4. Associate Tasks

<table>
<thead>
<tr>
<th>Field Tasks</th>
<th></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Notifications</th>
<th></th>
</tr>
</thead>
</table>

- **Task Name**: Milk reminder
- **Choose Task**: Email Notification
- **From**: ${zoho.adminuserid}
- **To**: ${Email},
- **Subject**: Subject
- **Message**:

  ![Message editor](image)

  Insert Fields: dropdown menu

- **can use variable here**
- **but not here**
welcome back, VBA?

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 *columns* to display specific columns.
- You have *filtering options* along the top-right side of the table, where you can filter the table by function name and category.
- For more detailed information about a particular function, click the function name to go to its page.

<table>
<thead>
<tr>
<th>Description</th>
<th>Syntax</th>
<th>Example</th>
<th>Result</th>
<th>Collapse All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td></td>
<td>Example</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Array</strong></td>
<td>Used within your expressions to manipulate, insert, and/or select values from arrays.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>append()</td>
<td>append({10, 20, 30}, 99)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>index()</td>
<td>index({10, 20, 30}, 2, 1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>insert()</td>
<td>insert({10, 20, 30, 40}, 99, 1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>joinarray()</td>
<td>joinarray({1, 2, 3, 4},</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>drop()</td>
<td>drop({10, 20, 30}, 1)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

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

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

- Custom HTML5 attributes
- Embedded formula
- Data becomes schema
- Implicit editing controls

The code

```html
<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>
```
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

<table>
<thead>
<tr>
<th>Room</th>
<th>Sq. footage</th>
<th>Occupant</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dungeon Five</td>
<td>480</td>
<td>Sirius</td>
<td>Grad. student</td>
</tr>
<tr>
<td></td>
<td></td>
<td>James</td>
<td>Post-doc</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wormtail</td>
<td>Grad. student</td>
</tr>
<tr>
<td>Greenhouse Two</td>
<td>561</td>
<td>Bellatrix</td>
<td>Visiting Prof.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lily</td>
<td>Post-doc</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Remus</td>
<td>Post-doc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Role</th>
<th>Allocated space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grad. student</td>
<td>12</td>
</tr>
<tr>
<td>Post-doc</td>
<td>20</td>
</tr>
<tr>
<td>Visiting Prof.</td>
<td>45</td>
</tr>
</tbody>
</table>
in google spreadsheet

<table>
<thead>
<tr>
<th></th>
<th>room</th>
<th>sq. footage</th>
<th>occupant</th>
<th>alloc. space</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dungeon Five</td>
<td>480</td>
<td>Sirius</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>James</td>
<td>Post-doc</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Wormtail</td>
<td>Grad. student</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Harry</td>
<td>Grad. student</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Bellatrix</td>
<td>Visiting Prof.</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Lily</td>
<td>Post-doc</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Remus</td>
<td>Post-doc</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Grad. student</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Post-doc</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Visiting Prof.</td>
<td>45</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

formulas are complex

formulas are unstable

but nesting is only visual, not computational
### in object sheets

- **Formulas over sets (now stable):**

  \[ \text{sqFoot} - \sum_{o: \text{Occupant}} (o\.role\.allocSpace) \]

<table>
<thead>
<tr>
<th>Room</th>
<th>sqFoot</th>
<th>Occupant</th>
<th>Role</th>
<th>allocSpace</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dungeon Five</td>
<td>480</td>
<td>Sirius</td>
<td>Grad. student</td>
<td>436</td>
</tr>
<tr>
<td></td>
<td></td>
<td>James</td>
<td>Post-doc</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wormtail</td>
<td>Grad. student</td>
<td></td>
</tr>
<tr>
<td>Greenhouse Two</td>
<td>561</td>
<td>Bellatrix</td>
<td>Visiting Prof.</td>
<td>476</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lily</td>
<td>Post-doc</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Remus</td>
<td>Post-doc</td>
<td></td>
</tr>
</tbody>
</table>

- **Nesting is now semantic, not just visual**

- **First-class object references**
a parent-teacher app

Parent view for Molly

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

Teacher view for Flitwick

<table>
<thead>
<tr>
<th>Slot time</th>
<th>Scheduled meeting</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014-12-17 13:00</td>
<td>Ginevra in 6.005</td>
</tr>
<tr>
<td>2014-12-17 14:00</td>
<td></td>
</tr>
</tbody>
</table>

what parent sees

what teacher sees

what principal sees
```excel
<table>
<thead>
<tr>
<th>Person</th>
<th>Slot</th>
<th>Enrollments</th>
<th>valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snape</td>
<td>scheduledEnrollment: Section: teacher = Person</td>
<td>Party in 6.820</td>
<td>true</td>
</tr>
<tr>
<td>Flitwick</td>
<td>scheduledEnrollment: Section: teacher = Person</td>
<td>Snape @ 2014-12-16 13:45</td>
<td>true</td>
</tr>
<tr>
<td>Ronald</td>
<td>scheduledEnrollment: Section: teacher = Person</td>
<td>Snape in 6.170</td>
<td>true</td>
</tr>
<tr>
<td>Ginevra</td>
<td>scheduledEnrollment: Section: teacher = Person</td>
<td>Snape @ 2014-12-16 13:00</td>
<td>true</td>
</tr>
<tr>
<td>Seamus</td>
<td>scheduledEnrollment: Section: teacher = Person</td>
<td>Snape @ 2014-12-16 13:00</td>
<td>true</td>
</tr>
</tbody>
</table>
```

"the teacher of the section of this enrollment is the person for this slot"
code reductions for some apps

<table>
<thead>
<tr>
<th>Application</th>
<th>Language/Framework</th>
<th>Original Implementation</th>
<th></th>
<th></th>
<th></th>
<th>Objshests+Mavo</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Code (LoC)</td>
<td></td>
<td>Code (LoC)</td>
<td></td>
<td>Formulas (count)</td>
<td></td>
<td>Macros (LoC)</td>
<td></td>
</tr>
<tr>
<td>PTC</td>
<td>(unavailable to us)</td>
<td>113</td>
<td></td>
<td>75</td>
<td></td>
<td>20</td>
<td></td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>TodoMVC</td>
<td>JavaScript/Angular</td>
<td>694</td>
<td></td>
<td>399</td>
<td></td>
<td>30</td>
<td></td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>Conf</td>
<td>Python/Django</td>
<td>158</td>
<td></td>
<td>142</td>
<td></td>
<td>2</td>
<td></td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>HackQ</td>
<td>JavaScript/Meteor</td>
<td>188</td>
<td></td>
<td>40</td>
<td></td>
<td>2</td>
<td></td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Got Milk</td>
<td>Perl/CGI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
conclusions
pros and cons

**Pros:**
- Bringing together visual GUI building, model-driven development, and declarative queries.
- Mature platforms especially for larger players.
- Integration of web service APIs and enterprise backends.
- Easy ramp-up with simple things being simple but getting harder fast.

**Cons:**
- Technology limitations and ad hoc limitations.
- Query language is SQL, not non-declarative scripts.
- Sidesteps standard tools like automated testing, version control, and even collaboration.
- Talent shortage may be hard to hire with few resources online.
### Stack Overflow Tags

<table>
<thead>
<tr>
<th>Tag</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>rails</td>
<td>272674</td>
</tr>
<tr>
<td>spring</td>
<td>108989</td>
</tr>
<tr>
<td>excel-vba</td>
<td>60557</td>
</tr>
<tr>
<td>meteor</td>
<td>25549</td>
</tr>
<tr>
<td>sap</td>
<td>3356</td>
</tr>
<tr>
<td>abap</td>
<td>1316</td>
</tr>
<tr>
<td>servicenow</td>
<td>309</td>
</tr>
<tr>
<td>nintex</td>
<td>125</td>
</tr>
<tr>
<td>k2</td>
<td>115</td>
</tr>
<tr>
<td>outsystems</td>
<td>80</td>
</tr>
<tr>
<td>quickbase</td>
<td>58</td>
</tr>
<tr>
<td>mendix</td>
<td>8</td>
</tr>
<tr>
<td>bizagi</td>
<td>8</td>
</tr>
<tr>
<td>appian</td>
<td>4</td>
</tr>
<tr>
<td>salesforce app cloud</td>
<td>4</td>
</tr>
<tr>
<td>agilepoint</td>
<td>2</td>
</tr>
<tr>
<td>caspio</td>
<td>0</td>
</tr>
<tr>
<td>matssoft</td>
<td>0</td>
</tr>
<tr>
<td>micropact</td>
<td>0</td>
</tr>
<tr>
<td>miosoft</td>
<td>0</td>
</tr>
</tbody>
</table>
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