LOW CODE, CITIZEN DEVELOPMENT & NEW PROGRAMMING PARADIGMS

DANIEL JACKSON
the growth of computational power
more transistors

predicted (1965)
more transistors

predicted (1965)

actual (2010)
more calculations
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
    YREGS
    END    REVERSE
```
reversing a string, 1960-1990

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

360 Assembly
reversing a string, 1960-1990

**Algol 68**

```algol
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;
```

**360 Assembly**

```asm
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 LOOP
    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
```

**Fortran**

```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
```
### Reverse a String

**Algol 68**

```algol68
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;
```

### 360 Assembly

```assembly
REVERSE CSECT
    USING REVERSE,R13
    B 72(R15)
    DC 17F'0'
    STM R14,R12,12(R13)
    ST R13,4(R15)
    ST R15,0(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
```

### Fortran

```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

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

### 360 Assembly

#### 360 Assembly Code
- **Purpose**: Reversing a string
- **Language**: 360 Assembly
- **Description**: The code snippet shows the assembly language implementation of reversing a string.

#### 360 Assembly Code (Source)
- **Source Location**: 360 Assembly
- **Function**: `REVERSE`
- **Usage**: `USING REVERSE,R13`
- **Operands**: `B 72(R15)`
- **Memory Stacks**: `STM R14,R12,12(R13)`
- **Loop Control**: `DO i = 1, length/2`
- **Character Handling**: `MVC TMP(L'C),C`
- **String Manipulation**: `str(i:i) = str(length+1-i:length+1-i)`

### Fortran

#### Fortran Code
- **Purpose**: Reversing a string
- **Language**: Fortran
- **Description**: The code snippet shows the Fortran implementation of reversing a string.

#### Fortran Code (Source)
- **Source Location**: Fortran
- **Function**: `reverse`
- **Variables**: `str` of type `CHARACTER(80)`
- **Function Call**: `WRITE (*,*) str`
- **Loop**: `DO i = 1, length/2`
- **String Manipulation**: `temp = str(i:i)`
- **Output**: `WRITE(*,*) str`

### Haskell

#### Haskell Code
- **Purpose**: Reversing a string
- **Language**: Haskell
- **Description**: The code snippet shows the Haskell implementation of reversing a string.

#### Haskell Code (Source)
- **Source Location**: Haskell
- **Function**: `reverse`
- **Operation**: `foldl (flip (:)) []`

---

**Reverse a String, 1960-1990**

The document provides examples of reversing a string in various programming languages from 1960-1990, including 360 Assembly, Fortran, Algol 68, and Haskell. Each section includes the code snippet for reversing a string, demonstrating the syntax and structure of the respective languages.
a benchmark example
todomvc.com showcase of MVC frameworks
a benchmark example

todomvc.com
showcase of MVC frameworks
a benchmark example
todomvc.com showcase of MVC frameworks

300 loc for this?
pain points
consumer price index

from bls.gov
consumer price index

from bls.gov
consumer price index

from bls.gov
consumer price index

from bls.gov
costs of standard IT
costs of standard IT

buying off-the-shelf may not fit your needs paying for unused features
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
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

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.
costs of shadow IT

why people do it
storage & backup
sharing and sending files
hosting small websites

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
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
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();}
    }
}

using network to respond
to date requests
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();}
    }
}

using network to respond to date requests

using AWT to display a message

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);
            }
        });
    }
}
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);
            }
        });
    }
}

monolithic apps

uniform, explicit & simple dependences

using network to respond to date requests

using AWT to display a message
**monolithic apps**

- Uniform, explicit & simple dependences
- Statically typed interfaces

```java
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();}
    }
}
```

- Using network to respond to date requests

**using AWT to display a message**

```java
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);
                }
            });
    }
}
```
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();}
    }
}

using network to respond to date requests

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);
                }
            });
    }
}
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();
        }
    }
}

using network to respond to date requests

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);
            }
        });
    }
}

using AWT to display a message

monolithic apps

- uniform, explicit & simple dependences
- statically typed interfaces
- low level details
- just one language
```java
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();}
    }
}
```

```java
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);
            }
        });
    }
}
```
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();
        }
    }
}

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);
            }
        });
    }
}
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();}
    }
}

using network to respond to date requests

monolithic apps

uniform, explicit & simple dependences

statically typed interfaces

no separation of concerns

using AWT to display a message

low level details

just one language

spurious ordering

no separation of concerns
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();}
    }
}

using network to respond to date requests

monolithic apps

uniform, explicit & simple dependences

statically typed interfaces

verbose language

no separation of concerns

low level details

just one language

spurious ordering

no separation of concerns

using AWT to display a message

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
what the web wanted

a wish list
manipulate small data
use a database backend
interact with a client UI
separate 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
**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]
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

```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);
            }
        }
    }
}
```
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]

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);
            }
        }
    }
}

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);
            }
        }
    }
}

Javascript

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

Java

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

Javascript

no need to figure out types
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
sql = "Select * from Users where" +
      "name = '#{params[:name]}'}'" +
      "AND password = '#{params[:password]}'}'"

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

handlers for routes in separate files
separate concerns

```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);
```

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

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

<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

```js
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
```
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
interact with a client UI

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

save name in session

return name from 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);
});

a server that remembers your name

save name in session

return name from session

session structure hides cookies + db
so are we happy now?
JavaScript

“the duct tape of the Internet”
JavaScript

“the duct tape of the Internet”

> 1 + "hello"
JavaScript

"the duct tape of the Internet"

> 1 + "hello"
"1hello"
JavaScript
“the duct tape of the Internet”

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

> 1 / "hello"
```

JavaScript

“the duct tape of the Internet”

```javascript
> 1 + "hello"
"1hello"
> 1 / "hello"
NaN
```
JavaScript

“the duct tape of the Internet”

```javascript
> 1 + "hello"
"1hello"
> 1 / "hello"
NaN
> x = 1 / "hello"
```

JavaScript
“the duct tape of the Internet”

```javascript
> 1 + "hello"
"1hello"
> 1 / "hello"
NaN
> x = 1 / "hello"
> (x == NaN) ? "bad": "good"
```
JavaScript

“the duct tape of the Internet”

```javascript
> 1 + "hello"
"1hello"
> 1 / "hello"
NaN
> x = 1 / "hello"
> (x == NaN) ? "bad": "good"
"good"
```
JavaScript

“the duct tape of the Internet”

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

> 1 / "hello"
NaN

> x = 1 / "hello"

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

> typeof(NaN)
```

JavaScript is a popular scripting language used primarily in web development.
JavaScript

“the duct tape of the Internet”

```javascript
> 1 + "hello"
"1hello"
> 1 / "hello"
NaN
> x = 1 / "hello"
> (x == NaN) ? "bad": "good"
"good"
> typeof(NaN)
"number"
```
JavaScript
“the duct tape of the Internet”

```javascript
> 1 + "hello"
"1hello"
> 1 / "hello"
NaN
> x = 1 / "hello"
> (x == NaN) ? "bad": "good"
"good"
> typeof(NaN)
"number"
> NaN === NaN
false
```
JavaScript
“the duct tape of the Internet”

```javascript
> 1 + "hello"
"1hello"
> 1 / "hello"
NaN
> x = 1 / "hello"
> (x == NaN) ? "bad": "good"
"good"
> typeof(NaN)
"number"
> NaN === NaN
false
```
JavaScript

“the duct tape of the Internet”

```javascript
> 1 + "hello"
"1hello"
> 1 / "hello"
NaN
> x = 1 / "hello"
> (x == NaN) ? "bad": "good"
"good"
> typeof(NaN)
"number"
> NaN === NaN
false
> NaN !== NaN
```
JavaScript

“the duct tape of the Internet”

```javascript
> 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)
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)

and the answer is...
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...
database complexities

embedded SQL: not a good match for objects

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

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

embedded SQL: not a good match for objects

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

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

using an “object relational mapper”

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

embedded SQL: not a good match for objects

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

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

using an “object relational mapper”

```ruby
user = User.where("name = '#{params[:name]}\nAND 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

```javascript
  do_a();
  do_b();
  do_c();
```
callback hell

what we used to write

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

what we write now

```javascript
do_a (function () {
do_b (function () {
do_c(...)})})
})
```
callback hell

what we used to write

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

what we write now

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

what inspired this madness?
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  database
callback hell

what we used to write

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

what we write now

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

what inspired this madness?

**web server**
```
d = db.query();
respond (d);
```

**database**
callback hell

what we used to write

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

what we write now

do_a (function () {
do_b (function () {
doc_c(...)
})
})

what inspired this madness?

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

database

server blocks waiting for DB
callback hell

what we used to write

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

what we write now

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

what inspired this madness?

---

**web server**

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

server blocks waiting for DB

**database**

```javascript
db.query(
    function (d) {
        respond(d)
    }
)`
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);
db.query(
    function (d) {
    respond(d)
})

server returns immediately
“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)
Though hard to describe in words, Visicalc 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
technology origins

model-driven development

UML class diagram, Visual Studio (OMT, 1991)
technology origins

model-driven development

UML class diagram, Visual Studio (OMT, 1991)

user interface builders

Netbeans GUI Builder (Project Matisse, 2008)
technology origins

model-driven development

UML class diagram, Visual Studio (OMT, 1991)

user interface builders

Microsoft Access (first release 1992)

Netbeans GUI Builder (Project Matisse, 2008)

groupware databases
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
visual editing

Zoho’s form builder

recognition, not recall
visual editing

Zoho’s form builder

direct manipulation

recognition, not recall
abstractions

"form-tables"
abstractions

“form-tables”

activities
abstractions

“form-tables”

activities

rules
declare 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**
   - **Task Name**: Send service call
   - **Choose Task**: SMS Notification

**Zoho’s rules & tasks**
integration

Workato’s integrated backends
integration

Workato’s integrated backends

cloud recipes: Zapier, IFTTT, Workato
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
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

an example: hackathon Q
an example: hackathon Q

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
an example: hackathon Q

Mentors register skills
Participants request a skill
an example: hackathon Q

mentors register skills
participants request a skill
mentor assigns calls
An example: Hackathon Q

005q

There are 3 people in the queue!

My name is ___________ and I need help with ___________

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
Quickbase

tricky: had to provide reverse mapping from skills to mentors
tricky: had to provide reverse mapping from skills to mentors
not possible: allow requests for more than one skill
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?
what’s going on?

form-table approach
ad hoc query language
no underlying calculus
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
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
encountering limitations

let's make a rule
if user reports out of milk
then send message
not uniformly visual
not uniformly visual

<table>
<thead>
<tr>
<th>1. Basic Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule Name</td>
</tr>
</tbody>
</table>


1. Basic Details

Rule Name: Milk low

3. Criteria

- All Records
- Selected Records

Stock equals ""
not uniformly visual

1. Basic Details

Rule Name: Milk low

3. Criteria

- Selected Records
- Stock: equals

must recall field values
1. Basic Details

Rule Name: Milk low

3. Criteria

- All Records
- Selected Records

- Stock
- equals
- "low"
not uniformly visual

1. Basic Details

Rule Name: Milk low

3. Criteria

- All Records
- Selected Records

Stock: equals: "low"
not uniformly expressive
not uniformly expressive

4. Associate Tasks

Field Tasks

Notifications

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

can use variable here
not uniformly expressive

4. Associate Tasks

- **Field Tasks**
- **Notifications**

Task Name: Milk reminder
Choose Task: Email Notification
From: ${zoho.adminuserid}
To: ${Email},
Subject: Subject
Message: [Insert Fields]
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 columns to show 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>Name</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>Used within your expressions to manipulate, insert, and/or select values from arrays.</td>
</tr>
<tr>
<td>append()</td>
<td>append({10, 20, 30}, 99)</td>
</tr>
<tr>
<td>index()</td>
<td>index({10, 20, 30}, 2, 1)</td>
</tr>
<tr>
<td>insert()</td>
<td>insert({10, 20, 30, 40}, 99, 1)</td>
</tr>
<tr>
<td>joinarray()</td>
<td>joinarray({1, 2, 3, 4}, &quot;</td>
</tr>
<tr>
<td>ld()</td>
<td>ldrop({10, 20, 30}, 1)</td>
</tr>
</tbody>
</table>

Appian function API
some new research
two research projects
two research projects

aim
new approach to low-code
two research projects

aim
new approach to low-code

target
“community applications”
too complex for Drupal
too simple for full stack
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
a new layout language

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

HTML

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

CSS

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

by Lea Verou & David Karger

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

the app
to-do in Mavo

the app

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>
```
to-do in Mavo

the app

custom HTML5 attributes

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>
```
to-do in Mavo

the app

custom HTML5 attributes

data becomes schema

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>
```
to-do in Mavo

the app

custom HTML5 attributes

data becomes schema

the code

implicit editing controls
to-do in Mavo

the app

custom HTML5 attributes

the code

embedded formula

data becomes schema

implicit editing controls
a new data store model
a new data store model

spreadsheet
a new data store model

spreadsheet

intuitive visual layout
schema evolves with data
can see all the data
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
a new data store model

- **Spreadsheet**
  - Intuitive visual layout
  - Schema evolves with data
  - Can see all the data

- **Relational Database**
  - Numeric queries only
  - Can't handle nested data
  - Risky to insert/delete rows
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
### A New Data Store Model

<table>
<thead>
<tr>
<th>Spreadsheet</th>
<th>Relational Database</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Intuitive visual layout</strong></td>
<td><strong>Rich query language</strong></td>
</tr>
<tr>
<td>Schema evolves with data</td>
<td>Can encode structured data</td>
</tr>
<tr>
<td>Can see all the data</td>
<td>Easy to insert/delete tuples</td>
</tr>
<tr>
<td>Numeric queries only</td>
<td>Not intuitive to end users</td>
</tr>
<tr>
<td>Can’t handle nested data</td>
<td>Hard to evolve schema</td>
</tr>
<tr>
<td>Risky to insert/delete rows</td>
<td>Seeing data needs queries</td>
</tr>
</tbody>
</table>
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
A new data store model can encode structured data, is easy to insert/delete tuples, but not intuitive to end users and hard to evolve schema. Seeing data needs queries.

**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
### 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>
### 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>
### 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>

### Allocated Space

<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>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>room</td>
<td>sq. footage</td>
</tr>
<tr>
<td>Dungeon Five</td>
<td>480</td>
</tr>
<tr>
<td>James</td>
<td>Post-doc</td>
</tr>
<tr>
<td>Harry</td>
<td>Grad. student</td>
</tr>
<tr>
<td>Greenhouse Two</td>
<td>561</td>
</tr>
<tr>
<td>Lily</td>
<td>Post-doc</td>
</tr>
<tr>
<td>role</td>
<td>alloc. space</td>
</tr>
<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>

- cell F2: =VLOOKUP(D2, Sheet1!A10:B12, 1, FALSE)
- cell F10: =B2 - SUM(E2:E4)
- cell F12: =B6 - SUM(E6:E8)
In Google Spreadsheet:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>room</td>
<td>sq. footage</td>
<td>occupant</td>
<td>room</td>
<td>12</td>
<td>436</td>
</tr>
<tr>
<td>Dungeon Five</td>
<td>480</td>
<td>Sirius</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>James</td>
<td></td>
<td>Post-doc</td>
<td></td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Wormtail</td>
<td></td>
<td>Grad. student</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Harry</td>
<td></td>
<td>Grad. student</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Greenhouse Two</td>
<td>561</td>
<td>Bellatrix</td>
<td>Visiting Prof.</td>
<td>45</td>
<td>476</td>
</tr>
<tr>
<td>Lily</td>
<td></td>
<td>Post-doc</td>
<td></td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Remus</td>
<td></td>
<td>Post-doc</td>
<td></td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

Role: 

- Grad. student: 12
- Post-doc: 20
- Visiting Prof.: 45

Formulas:

- =VLOOKUP(D2, Sheet1!A10:B12, 3, FALSE)
- =B2 - SUM(E2:E4)
- =B6 - SUM(E6:E8)

but nesting is only visual, not computational
In Google spreadsheet, nesting formulas is only visual, not computational. The formulas are complex and include functions like `VLOOKUP` and `SUM` to calculate the alloc. space.
In Google spreadsheet, nesting is only visual, not computational. Formulas can be complex and unstable.
in object sheets

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

<table>
<thead>
<tr>
<th>Role</th>
<th>title</th>
<th>allocSpace</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Grad. student</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Post-doc</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Visiting Prof.</td>
<td>45</td>
</tr>
</tbody>
</table>
In object sheets, nesting is now semantic, not just visual.
in object sheets

<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
in object sheets

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

<table>
<thead>
<tr>
<th>Room</th>
<th>sqFoot</th>
<th>Occupant</th>
<th>Role</th>
<th>free</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
in object sheets

formulas over sets (now stable)

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

<table>
<thead>
<tr>
<th>Room</th>
<th>sqFoot</th>
<th>Occupant</th>
<th>free number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dungeon Five</td>
<td>480</td>
<td>Sirius Grad. student</td>
<td>436</td>
</tr>
<tr>
<td></td>
<td></td>
<td>James Post-doc</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wormtail Grad. student</td>
<td></td>
</tr>
<tr>
<td>Greenhouse Two</td>
<td>561</td>
<td>Bellatrix Visiting Prof.</td>
<td>476</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lily Post-doc</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Remus 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: 
  - Ginevra
    - Potions with Snape: 
    - Charms with Flitwick: 2014-12-17 13:00

Teacher view for Flitwick
- Slot time
  - 2014-12-17 13:00: Ginevra in 6.005
  - 2014-12-17 14:00

what parent sees
what teacher sees
what principal sees
<table>
<thead>
<tr>
<th>Person</th>
<th>Slot</th>
<th>Enrollment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>name</td>
<td>Class code</td>
</tr>
<tr>
<td></td>
<td>roles</td>
<td>name</td>
</tr>
<tr>
<td>Snape</td>
<td>teacher</td>
<td>6.170</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.820</td>
</tr>
<tr>
<td>Flitwick</td>
<td>teacher</td>
<td>6.005</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ronald</td>
<td>student</td>
<td>Molly</td>
</tr>
<tr>
<td>Ginevra</td>
<td>student</td>
<td>Molly</td>
</tr>
<tr>
<td>Seamus</td>
<td>student</td>
<td>Augustus</td>
</tr>
<tr>
<td>Molly</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Augustus</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
the backend

scheduledEnrollment.Section.teacher = Person

"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>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Code (LoC)</td>
<td>HTML (LoC)</td>
<td>Formulas (count)</td>
<td>Macros (LoC)</td>
</tr>
<tr>
<td><em>PTC</em></td>
<td>(unavailable to us)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>TodoMVC</em></td>
<td>JavaScript/Angular</td>
<td>113</td>
<td>75</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td><em>Conf</em></td>
<td>Python/Django</td>
<td>694</td>
<td>399</td>
<td>30</td>
<td>62</td>
</tr>
<tr>
<td><em>HackQ</em></td>
<td>JavaScript/Meteor</td>
<td>158</td>
<td>142</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td><em>Got Milk</em></td>
<td>Perl/CGI</td>
<td>188</td>
<td>40</td>
<td>2</td>
<td>15</td>
</tr>
</tbody>
</table>
conclusions
pros and cons
pros and cons

**brining 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
**pros and cons**

**pros**
- 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

**cons**
- technology limitations
- ad hoc limitations
- query language ≪ SQL non-declarative scripts
- sidesteps standard tools automated testing version control even collaboration
- talent shortage may be hard to hire few resources online
<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
questions for discussion

diagrams better than text?
what about sharing?
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?
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?
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