Beyond Putting Lipstick on the Pig

Mark Reilly
@alien_resident

bit.ly/edu_pig
#edu_pig

Beyond Putting Lipstick on the Pig

Usability Testing and Designing New UIs for Open Source Projects

About This Presentation

I am using `reveal.js` for my slides. They're on github

You can follow along on bit.ly/edu_pig

What you'll Learn

Usability

- How to conduct usability tests
- How to communicate the results of the tests to generate excitement and commitment to fixing issues
- How to divide solutions into quick wins and long-term fixes

What you'll Learn *contd*

Designing Web Systems

- The best way to organize chaos and build a foundation via a style guide and a design pattern library for developers
- How to use modular design to create a sophisticated and highly customizable skin
- How you can contribute to an open source community and share your efforts

What you'll Learn *contd*

Selling your work but **not** your soul

- How to get buy-in from the multitude of stakeholders, committees, and service teams without going insane
- How to bring innovation to a risk-averse IT culture without compromising the design

Mark Reilly

User Experience ~ User Interface

UX~UI T-Shirt at UIStencils.com

Contact

@alien_resident - Twitter
alienresident - Github
Mark Reilly - LinkedIn

Background

A Common Scenario

A web designer or developer is tasked with adding the institutional branding and colors to a vendor’s software.

Background *contd*

A Uncommon Scenario

- Researching Users Needs
- Finding Users Problems
- Designing Solutions
- Working with Developers
- Testing to see If Users Achieve their Goals
- and Iterating

Background *contd*

How was the Product Selected?

- Project Requirements were Drawn Up
- RFPs were Solicited from a Small Pool of Candidates
- Selection is Based on Budget and their Sales Pitch

Background *contd*

The Result of this Broken Process?

A product:

- Pretty Interface Graphics
- Previously Installed by another Group
- Prioritized Integrations over Goals
- Poorly Implemented GUI

Background *contd*

Then you’re Tasked

To ‘tart’ up the product by adding your institution’s branding.

The Muppets Take Manhattan - 1984 *(TriStar Pictures)* Youtube

Aftermath: Unhappy users

- The system doesn’t meet their needs
- Users aren’t satisfied
- You’re frustrated
  - You’re unable to get the vendor to change things without additional funds
  - You’re not in control of what get’s prioritized

How Can We Change this Cycle?

Selecting Open Source products is a good start.

Why Open Source is Different

- *In theory* you can see the code and make changes
- *In practice* this is not always that straight forward
- A certain amount of technical expertise is need:
    - internal resource or a hired specialist

Why Open Source is Different

... "free software" is a matter of liberty, not price.
... you should think of "free" as in "free speech,"
not as in "free beer".

What is free software? gnu.org

Why Open Source is Different

“Free as in kittens”

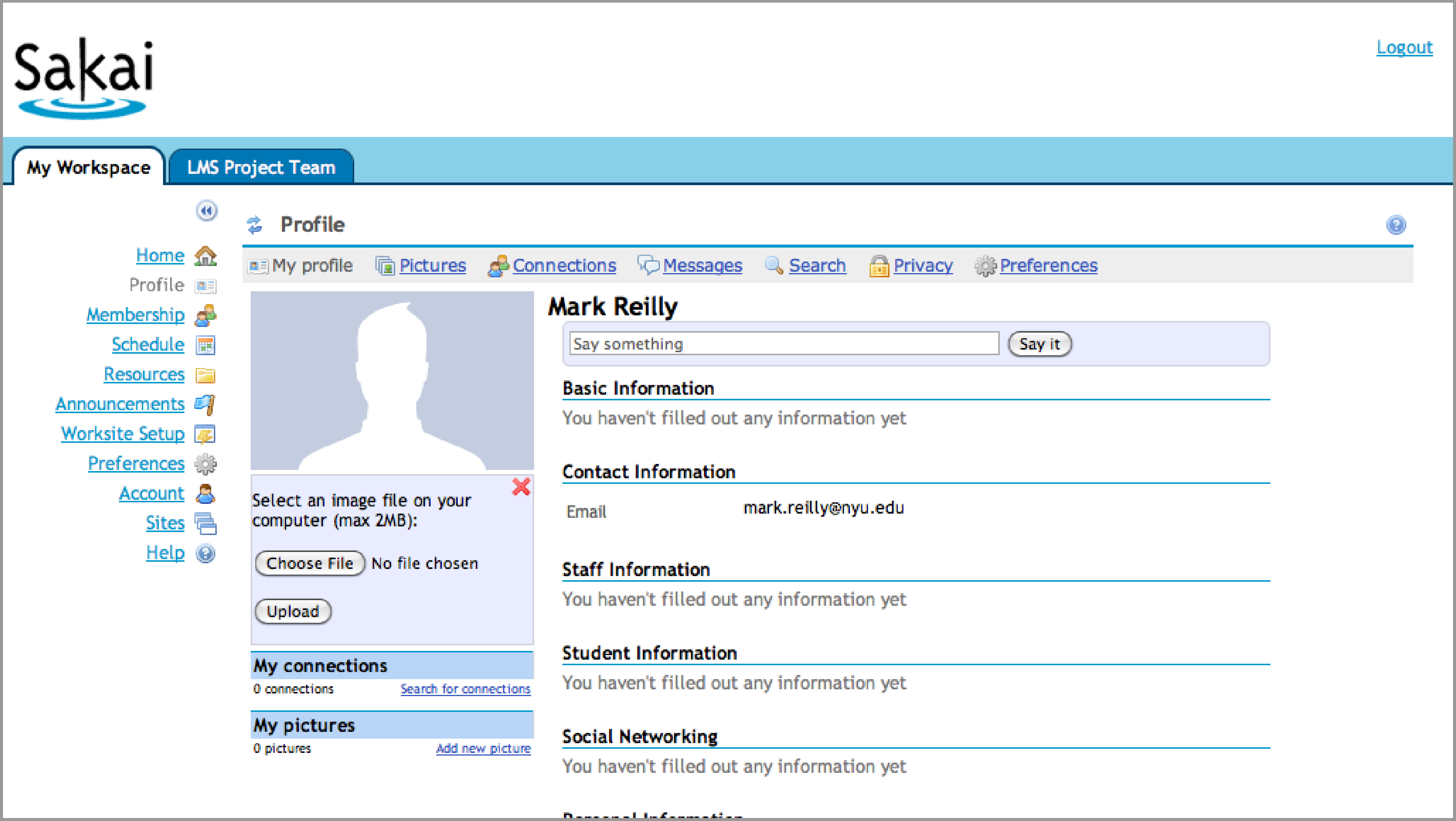
OMGSoCute.com

Sakai to Replace BlackBoard

NYU decided to move from BlackBoard to Sakai in the Fall 2011.

Sakai

NYUClasses



Logout

My Workspace | LMS Project Team

Home
Profile
Membership
Schedule
Resources
Announcements
Worksite Setup
Preferences
Account
Sites
Help

Profile

My profile | Pictures | Connections | Messages | Search | Privacy | Preferences

Mark Reilly

Say something | Say it

Select an image file on your computer (max 2MB):

Choose File No file chosen

Upload

My connections

0 connections Search for connections

My pictures

0 pictures Add new picture

Basic Information

You haven't filled out any information yet

Contact Information

Email	mark.reilly@nyu.edu

Staff Information

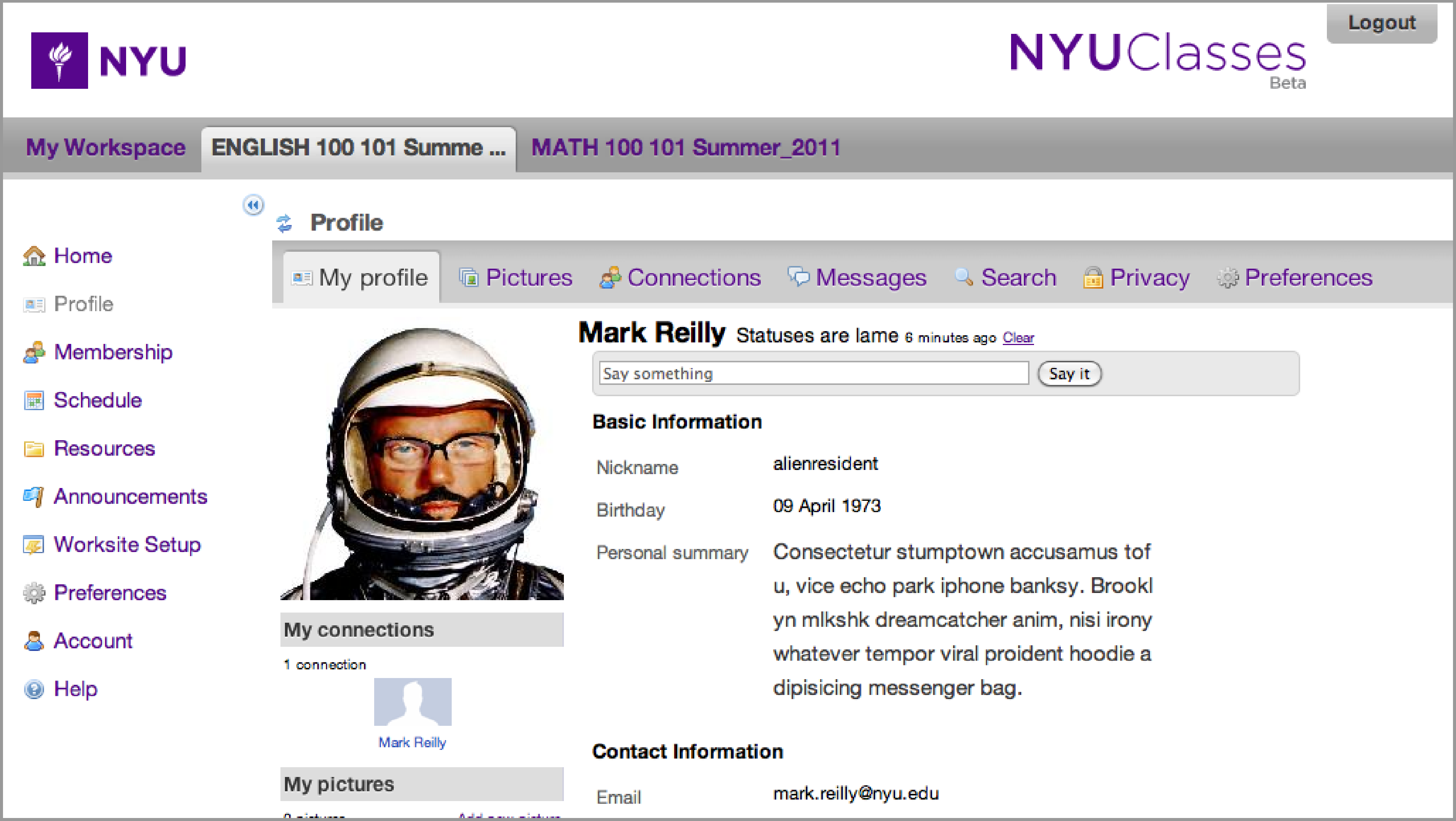
You haven't filled out any information yet

Student Information

You haven't filled out any information yet

Social Networking

You haven't filled out any information yet



NYU

NYUClasses Beta

Logout

My Workspace | ENGLISH 100 101 Summe ... | MATH 100 101 Summer_2011

- Home
- Profile
- Membership
- Schedule
- Resources
- Announcements
- Worksite Setup
- Preferences
- Account
- Help

Profile

My profile | Pictures | Connections | Messages | Search | Privacy | Preferences

Mark Reilly Statuses are lame 6 minutes ago Clear

Say something | Say it

Basic Information

Nickname	alienresident
Birthday	09 April 1973
Personal summary	Consectetur stumptown accusamus tofu, vice echo park iphone banksy. Brooklyn mlkshk dreamcatcher anim, nisi irony whatever tempor viral proident hoodie adipisicing messenger bag.

Contact Information

Email	mark.reilly@nyu.edu

My connections

1 connection

Mark Reilly

My pictures

Usability Testing

Typical Testing Schedule

- 3-5 users per test
- scheduled 1 test per hour
- allows time to tweak and reset test
- bathroom break
- connectivity issues

Usability Testing

Breakdown of a 45 minute test

- 5 mins introduction
- 10 mins of free exploration and general questions
- 20–25 mins script based test
- 5 mins feedback of the test

Usability Testing

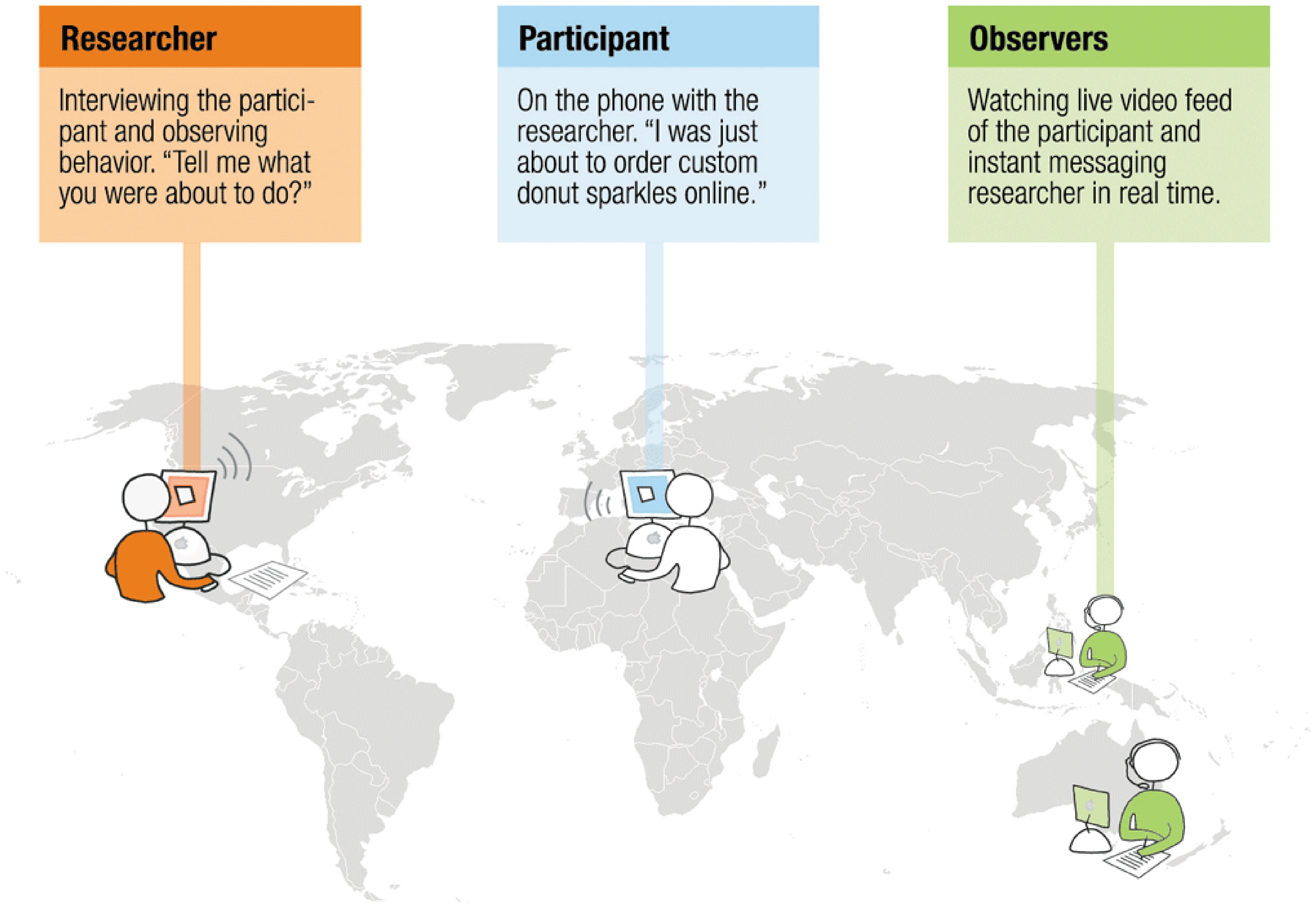
Breakdown of a 45 minute test

- 5 mins introduction
- 10 mins of free exploration and general questions
- 20–25 mins script based test
- 5 mins feedback of the test

Usability Testing

Our Criteria

- Never used NYU Classes/Sakai
- In their own environment — not in a usability lab
- Familiarity with other online class courseware



Researcher

Interviewing the participant and observing behavior. “Tell me what you were about to do?”

Participant

On the phone with the researcher. “I was just about to order custom donut sparkles online.”

Observers

Watching live video feed of the participant and instant messaging researcher in real time.

Remote Usability Testing

Things to Remember

- describe how the test works
- get verbal permission to record (voice and screen)
- remind Participant to close private documents
- passed presenter control to the Participant

Communicating the Results

- We done our usability tests!
- We found issues!
- We have solutions!
- Now what?

What to Fix?

“When fixing problems, always do the least you can.”

“Focus Ruthlessly on Fixing the Most Serious Problems First”

Steve Krug, *Rocket Surgery Made Easy*

Explaining the Results

Categorization

- Heuristics
  - Measuring
  - Weighting

How to Communicate the Results

- Not a dry report that no one reads
- A dynamic presentation
- Show don’t tell
- Present solutions

Quick Fixes

What can you do in the short term?

- Clarify the language?
- Draw attention to the solution using color?
- Add support documentation?
- file bug reports?

Beyond Quick Fixes

A Gut Renovation

Create a reusable, well organized, and easily extendable UI.

- Establish a new foundation
- Remove old CSS hacks
- Use a CSS preprocessor
- Organize and modularize the styles

Beyond Quick Fixes

Organizing Chaos

We're not designing pages, we're designing systems of components.

Stephen Hay

Beyond Quick Fixes

Designing Systems

- *Brad Frost* Atomic Design
- *Jonathan Snook* Scalable and Modular Architecture
- *Samantha Warren* Style Tiles
- *Dan Mall* Element Collages

Beyond Quick Fixes

Styleguide driven development

Living Styleguide

- Reusable
- Modular
- DRY
- A Manual (RTFM)

Beyond Quick Fixes

Using modular design practices

There are only two hard things in Computer Science: cache invalidation and naming things.

-- Phil Karlton

SASS

Syntactically Awesome StyleSheets

- Variables
- Mixins
- Extends
- Nesting
- @import & Partials

learn more at sass-lang.com/guide

SASS *contd*

Variables

```scss
$instution-color: #fc3;

a {
  color: $instution-color;
}

nav {
  background-color: $instution-color;
}
```

SASS *contd*

Mixins

```
@mixin default-type-size {
  font-size: 16px;
  line-height: 1.5em;
}
p {
  @include default-type-size;
}
footer {
  @include default-type-size;
}
```

```
p {
  font-size: 16px;
  line-height: 1.5em;
}
footer {
  font-size: 16px;
  line-height: 1.5em;
}
```

SASS *contd*

Mixins *(arguments)*

```
@mixin default-type-size($color) {
  font-size: 16px;
  color: $color;
}
p {
  @include default-type-size(#333);
}
footer {
  @include default-type-size(#eee);
}
```

```
p {
  font-size: 16px;
  color: #333333;
}
footer {
  font-size: 16px;
  color: #eeeeee;
}
```

SASS *contd*

Mixins *(more arguments)*

```scss
@mixin type-size($size, $color) {
  font-size: $size;
  color: $color;
}
p {
  @include type-size(16px, #333);
}
footer {
  @include type-size(14px, #eee);
}
```

```css
p {
  font-size: 16px;
  color: #333333;
}
footer {
  font-size: 14px;
  color: #eeeeee;
}
```

SASS *contd*

Extends

```
%default-type-size {
  font-size: 16px;
  line-height: 1.5em;
}
p {
  @extend %default-type-size;
}
footer {
  @extend %default-type-size;
}
```

```
p, footer {
  font-size: 16px;
  line-height: 1.5em;
}
```

SASS *contd*

Extends *contd*

```
%default-type-size {
  font-size: 16px;
  line-height: 1.5em;
}
p {
  @extend %default-type-size;
}
footer {
  @extend %default-type-size;
  color: #eeeeee;
}
```

```
p, footer {
  font-size: 16px;
  line-height: 1.5em;
}
footer {
  color: #eeeeee;
}
```

SASS *contd*

Nesting

```scss
nav {

  ul {
    margin: 0;
    padding: 0;
    list-style: none;

    li {
      display: inline-block;
    }
  }
}
```

```css
nav ul {
  margin: 0;
  padding: 0;
  list-style: none;
}

nav ul li {
  display: inline-block;
}
```

SASS *contd*

@import & Partials

```scss
// _reset.scss
html,
body,
ul,
ol {
  margin: 0;
  padding: 0;
}
```

```scss
// base.scss
@import 'reset';

body {
  font-size: 100% Helvetica, sans-serif;
  background-color: #efefef;
}
```

```css
/* base.css */
html, body, ul, ol {
  margin: 0;
  padding: 0;
}
body {
  font-size: 100% Helvetica, sans-serif;
  background-color: #efefef;
}
```

SASS *contd*

Sass while powerful can be misused. You can create brittle, inflexible, and unmaintainable CSS with Sass as with vanilla CSS.

What you need is an architecture that is modular and scalable.

SMACSS

Scalable and Modular Architecture for CSS

"SMACSS is becoming one of the most useful contributions to front-end discussion in years"
Scalable and Modular Architecture for CSS
A flexible guide to developing sites small and large.
by Jonathan Snook

By Jonathan Snook a web developer and designer, formerly at Yahoo!, He worked on the redesign of Yahoo! mail

learn more at smacss.com

SMACSS

At the very core of SMACSS is categorization

There are five types of categories:

1. Base
2. Layout
3. Module
4. State
5. Theme

SMACSS

Base

Base rules are the defaults.

```css
html, body, form {
  margin: 0;
  padding: 0;
}

input[type="text"] {
  border: 1px solid #999;
}

a {
  color: #039;
}

a:hover {
  color: #03C;
}
```

SMACSS

Layout

Layout rules divide the page into sections. Layouts hold one or more modules together.

```css
#article {
  width: 80%;
  float: left;
}

#sidebar {
  width: 20%;
  float: right;
}

.l-fixed #article {
  width: 600px;
}

.l-fixed #sidebar {
  width: 200px;
}
```

SMACSS

Module

Modules are the reusable, modular parts of our design. They are the callouts, the sidebar sections, the product lists and so on.

```
.pod {
  width: 100%;
  background: #ddd;
}

.pod input[type=text] {
  width: 50%;
  border: 1px solid #666;
}
```

SMACSS

Module *contd*

```
.pod {
  width: 100%;
  background: #ddd;
}

.pod input[type=text] {
  width: 50%;
  border: 1px solid #666;
}

.pod-callout {
  width: 200px;
}

.pod-callout input[type=text] {
  width: 180px;
}
```

```
<div class="pod pod-callout"> ... </div>
```

SMACSS

State

State rules are ways to describe how our modules or layouts will look when in a particular state. Is it hidden or expanded? Is it active or inactive?

```
.tab {
  background-color: purple;
  color: white;
}

.is-tab-active {
  background-color: white;
  color: black;
}
```

SMACSS

Theme

Theme rules are similar to state rules in that they describe how modules or layouts might look.

```css
/* in module-name.css */
.mod {
  border: 1px solid;
}

/* in theme.css */
.mod {
  border-color: blue;
}
```

Designing Web Systems

The goal is to empower designers and developers so they aren’t forced to add counter styles to the bottom of the stylesheet.

```
.maintext {
  color: red !important;
}
```

Contributing to an Open Source Community

The goal is to empower designers and developers so they aren’t forced to add counter styles to the bottom of the stylesheet. To enable them to easily change colors and branding. To enable them to dig in deeper and make substanial changes.

To avoid this.

```css
.maintext {
  color: red !important;
}
```

Contributing to an Open Source Community

People will generally be supportive of your efforts.

- Get on the mailing list: get a sense of the community
- Read the FAQ and look through their documentation
- See if there’s a post on their bug list or issue queue
- If you need help; check to see if your question has been asked and answered before

Contributing to an Open Source Community

Not a developer? You can contribute in other ways.

- Usability testing
- QA testing
- Documentation and writing
- Community organization

Contributing to an Open Source Community

Four Levels of Community Engagement

- Implementation
- Evangelism
- Documentation
- Support

Bringing Innovation to a Risk Averse Culture

People demand innovation but without change.

It's a long process

Getting the new UI into Production

Or the 12 tasks of Hercules!

It's was a long process

- Design review with the group responsible for the University’s Branding and Design
- Two technical reviews:
  - with the internal front-end developers
  - with the community — Gonzalo Silverio the original Sakai portal developer — he asked for the work to be committed back to Sakai.

Getting the new UI into Production

Or the 12 tasks of Hercules!

- Two rounds of informal usability testing for fine-tuning
- A formal A/B usability test
- Two reviews with the faculty's User Advisory Group. – They contributed a number of valuable suggestions that were implemented.
- Two rounds of Quality Assurance testing by the NYU Classes service team
- Two rounds of CSS bug fixing.

In Conclusion

Why you need to do a Usability Test!

- Insight Into Understanding the Problem
- Focus on the Users Problems and not the Technical or Aesthetic Issues
- Observed User Behavior Rather Than Opinion
- Creates Awareness of Issues to Managers, and Developers
- Builds a Consensus to Solving the Issues

In Conclusion

Why you need to do a Usability Test!

- Communicated the results: Problems and Solutions
- Quick wins are best
- Focus on the most serious problems

In Conclusion

- When possible do more
- Embrace open source
- Contribute back to the community in your own way

In Conclusion

- Change is hard and it takes a long time
- You need people on your side
- You should be able to prove your changes work
- Communication is key