webdev

Billy Janitsch and Ben Kuhn

Us

Ben makes things work (back-end)

Billy makes them pretty (front-end)

Gimblium: online game dev studio

Harvard Class: course shopping madness

Idea

"Guys, I have a great idea for a website. It's like Facebook, but for cats."

break it down

idea - - - - - > implementation

Breakdown

Design: ask a series of hows and whys

Function: break down into components (profiles, posts, comments)

Make a priority list

Have specifics in mind, but leave room for change and exploration

structure

idea - - - - - > implementation

Basics

Client - the browser and the stuff it displays/runs (HTML, CSS, Javascript); do most work here

Server - sends data to the client; ideally just permissions checking + database queries

Components of a web app

Complexity is your biggest enemy, so try to keep your components separate

Some trendy but useful buzzwords

- "Models" - the data or information your app deals with
- "Views" - how you present that data to users
- "Controllers" - the logic; how you change your model in response to user input

Models

Important thing: live on both client and server

When designing: what kind of data? what kind of queries?

E.g., cat FB post: author, text, recipient

Queries: by author, by recipient, by date

Oops, have to add another field

Models on the server

Figure out which queries absolutely have to be server-side (for data reasons)

Client-side searching/filtering is faster

Keep it simple, but don't send across too much

Hard to change DB, so consider a JSON blob

Client-server communication

Keep requests simple: get/create/update

```
GET http://fb.cat/post.php?id=1
POST /wallpost.php?user=2 { "text":"meow" }
POST /propic.php?user=1 <image data>
```

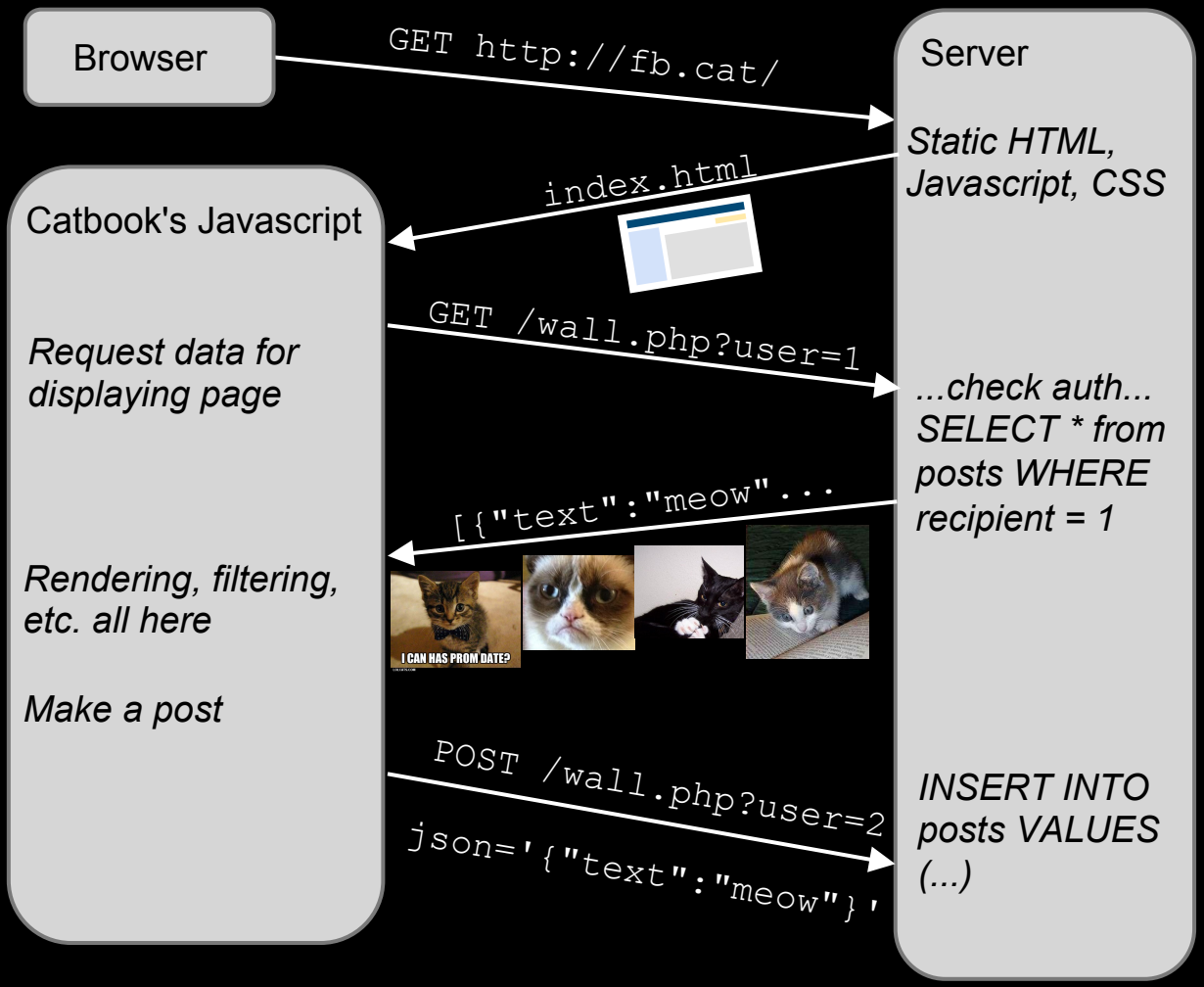
Best to pre-fetch the largest reasonable amount of data

```
GET /wall.php?user=1
GET /wall.php?user=1&since=10-31-2012
```

Send data in JSON

```
$.getJSON('/wall.php?user=1', ...);
```

Example


Browser
GET http://fb.cat/
Server
Static HTML, Javascript, CSS
index.html
Catbook's Javascript
GET /wall.php?user=1
Request data for displaying page
...check auth... SELECT * from posts WHERE recipient = 1
[{"text":"meow"...
Rendering, filtering, etc. all here
I CAN HAS PROM DATE?
Make a post
POST /wall.php?user=2
json='{"text":"meow"}'
INSERT INTO posts VALUES (...)

Fancy client-side stuff

Vanilla Javascript painful, but libraries/tools help a lot

jQuery for manipulating HTML easily

```
$("#fader").click(function () { $
  (this).delay(500).fadeOut() });
```

Underscore.js for utility functions

```
_.shuffle(_.uniq(_.flatten(lsts)));
```

Backbone.js for better architecture

Backbone.js

Javascript "models" and "collections": objects and lists that can trigger events when they change

```
posts.on("add", renderWall);
post.on("change:likes", function () {
    makeNotification("Someone liked your post");
});
post.on("change", function() {
    this.view.render();
});
```

Saves tons of complexity

Views

Technically pretty straightforward

jQuery jQuery jQuery

For complicated UI, look for libraries

Billy will explain more about hard parts

More advanced techniques

Fancy HTML5 stuff: local storage, websockets, single-page apps

CoffeeScript: compiles down to Javascript

```
squares = (x*x for x in [10..1]);
```

Other server languages - Ruby, Python, node.js

Search around if you're stuck on something

General points

Complexity is the devil - do the stupid easy way

Write first, clean up/throw away later

Hard work on client, data and auth on server

Libraries make your life better

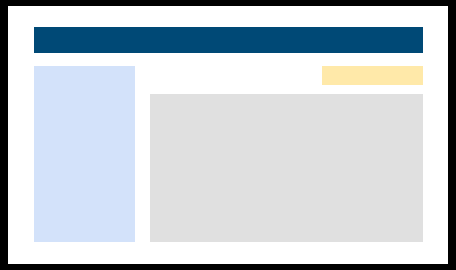
Search Web for more resources

design

idea - - - - - > implementation

UI Design

UI Design



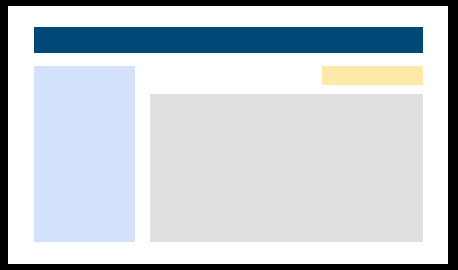
Work with a color scheme and typeface

Use bold colors sparingly

Be minimal

Mock up multiple designs (paper or PS)

UI Implementation



HTML/PHP: content, division thereof
See CS50

CSS: color, type, positioning, decorations
See Ben Shryock's seminar

js/jQuery: animations, dynamic data
See Vipul Shkhawat's seminar

UI vs. UX

Experience is more than just interface

A sexy design is necessary but not sufficient

UX Design

It doesn't matter if a user can do X, it matters if a user can easily figure out how to do X

Optimize for common use cases

Show, don't explain

Test both functionality and usability

project management

idea - - - - - > implementation

Team

Size affects performance: more people allow more work but require more communication

Balance of skills (back-end, front-end)

Fun + motivation is key

Iteration

Work in functional spurts

After each, pause and ask questions

Abandon ideas that aren't working, embrace new ones that might

Testers are useful - especially new ones

Good Practices

// Comment /* seriously though don't leave the // in the final slideshow it's kind of cheesy */

Clean up regularly

Version control: git
See Tommy MacWilliam's seminar

k thx bai

idea - - - - - > implementation

questions?