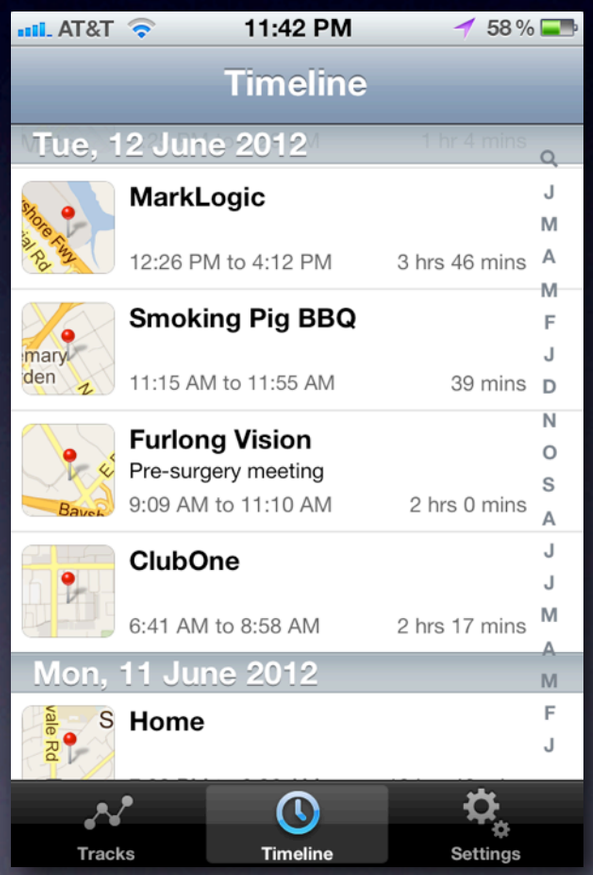
Using Parse.com as Your Server-Side

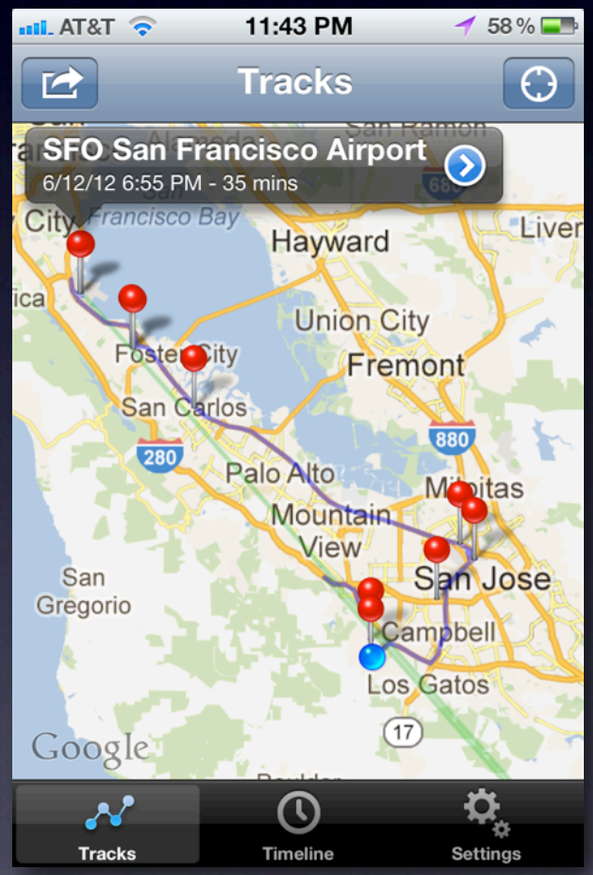
Jason Hunter

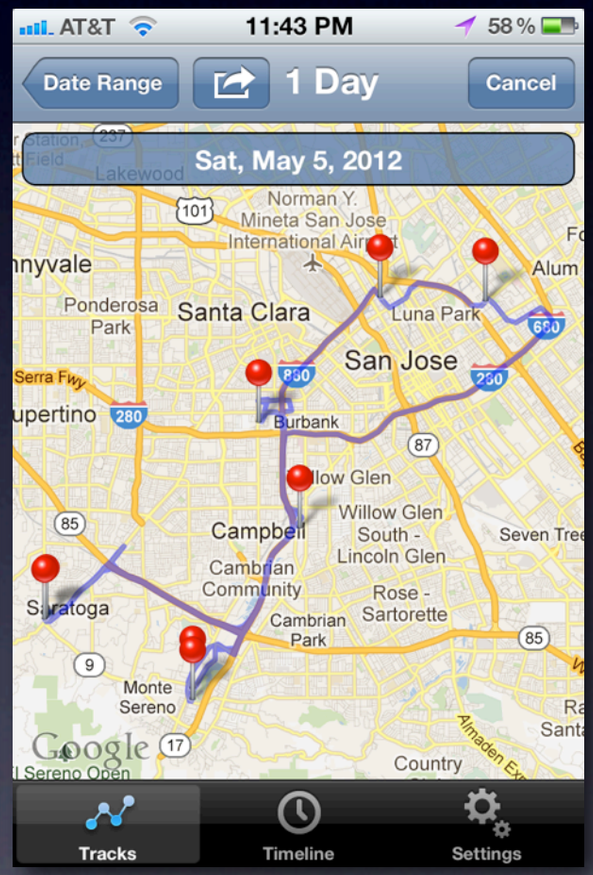
About Me

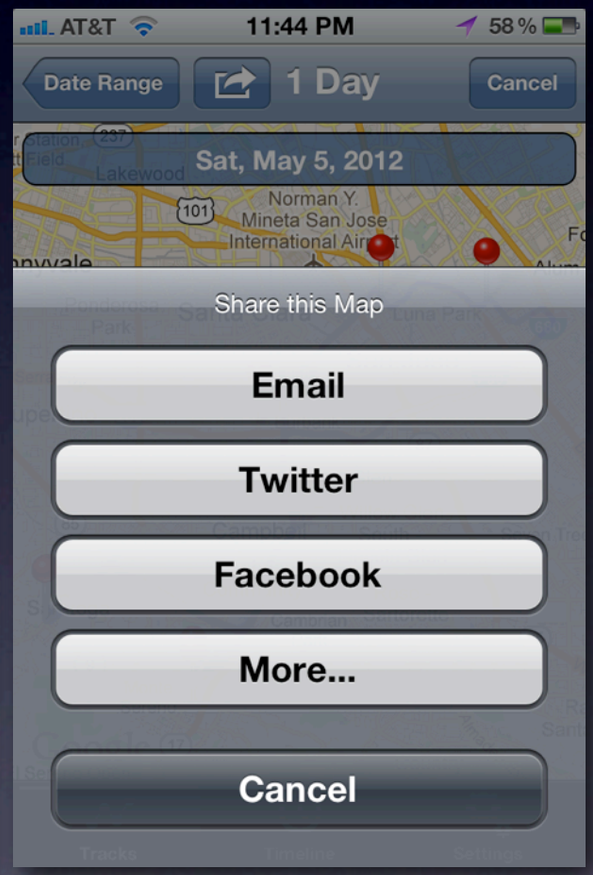
- Chief Architect at MarkLogic
- Weekend iOS hacker
- No affiliation with Parse.com
- @hunterhacker

I Need a Cloud









Parse.com

- Platform as a Service (PaaS)
- "The mobile app platform for developers"
- Parse handles the server side; lets you focus on your app
- Includes SDKs for iOS, Android, REST, & JavaScript

Parse Features

- Store and query (schemaless) objects
- Store binaries
- Cross-platform push notifications
- Complete user management; Twitter & FB
- Background communication

Parse

Home | Testimonials | Pricing | Docs | About | Blog

Sign Up | Log In

The mobile app platform for developers

Add a powerful and scalable backend in minutes for your Social Networking App

Try it for free

For iOS, Android & JavaScript

I couldn't imagine being able to build the Band of the Day backend as quickly and reliably without Parse.

Chris Stevenson - CTO, 955 Dreams
App of the Year 2011 Runner Up

Data | Push | Users | Social | Location | Files

Parse Tech Stack

"You should be ready to handle Ruby, Amazon Web Services, MongoDB, MySQL, Rails, Unicorn"

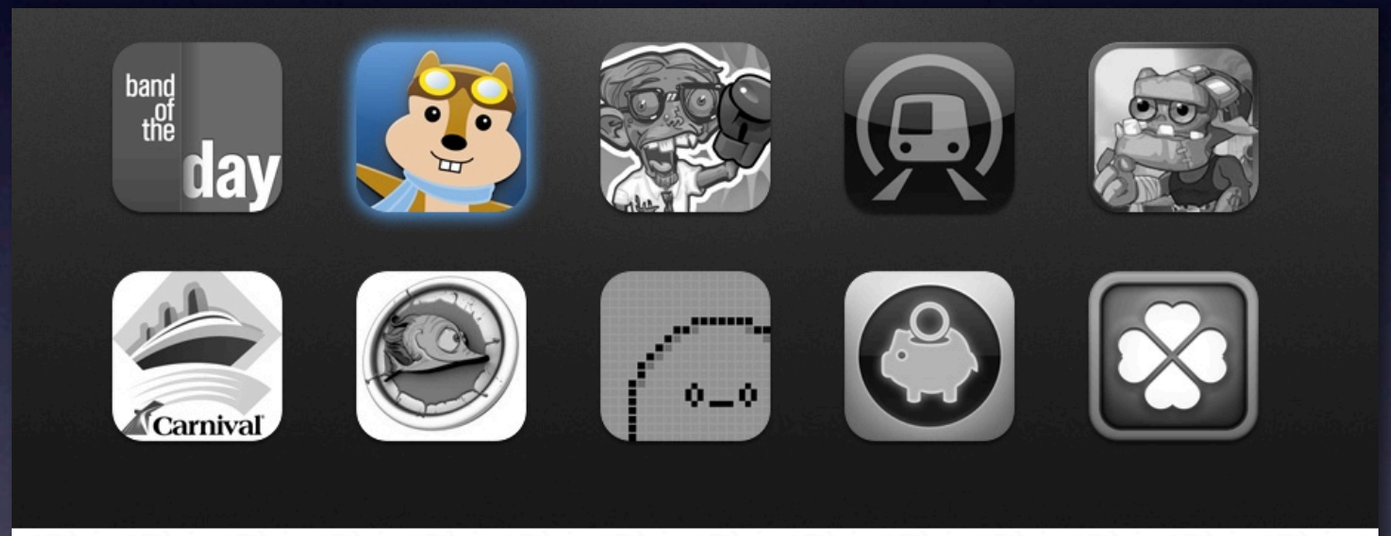
-- http://parse.com/jobs

Pricing

	Basic	Pro	Enterprise
	$0 per month	$199 per month	30 days free trial
API Requests per month	1,000,000 7¢ per 1,000 over	15,000,000 5¢ per 1,000 over	Contact us for a plan that fits your needs enterprise@parse.com
Pushes per month	1,000,000 7¢ per 1,000 over	5,000,000 5¢ per 1,000 over	
File Storage	1GB 20¢ per 1GB over	10GB 15¢ per 1GB over	
Roles for Access Control	1	5 $15/extra	
End-User Branding		✓	✓
Multiple Push Certificates		✓	✓
Auto-scaling Platform	✓	✓	✓

Who's Using It?

- Hipmunk is probably the most famous app



Hipmunk

by Hipmunk

Hipmunk takes the agony out of flight and hotel search. With Hipmunk you'll see all the relevant flights on a timeline, helping you visualize the right choice.

View in iTunes

Pros and Cons

- Handles so much of what you need to do
- Easy automatic scaling
- Shockingly frequent feature updates
- The tech stack is out of your control
- You're dependent 24x7 on a small startup

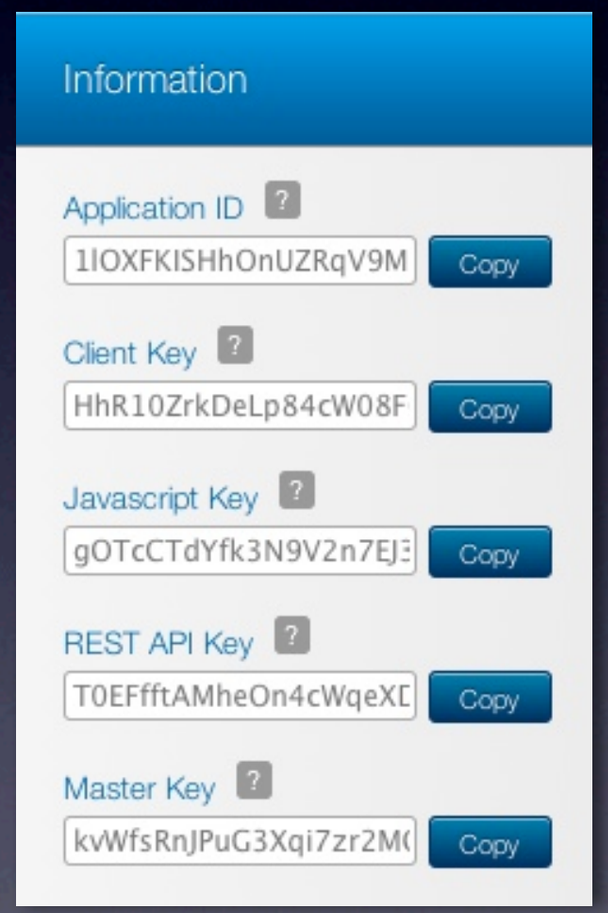
Competitors

- Appcelerator (nee Cocoafish)
- Kinvey
- StackMob
- Raw Amazon Web Services

Storing Objects

Registration

- You register with Parse to get a client key and an application id (i.e. one for test, one for production)
- In your AppDelegate you put this code:




```
#import "Parse/Parse.h"

[Parse setApplicationId:@"Your Application Id"
              clientKey:@"Your Client Key"];
```

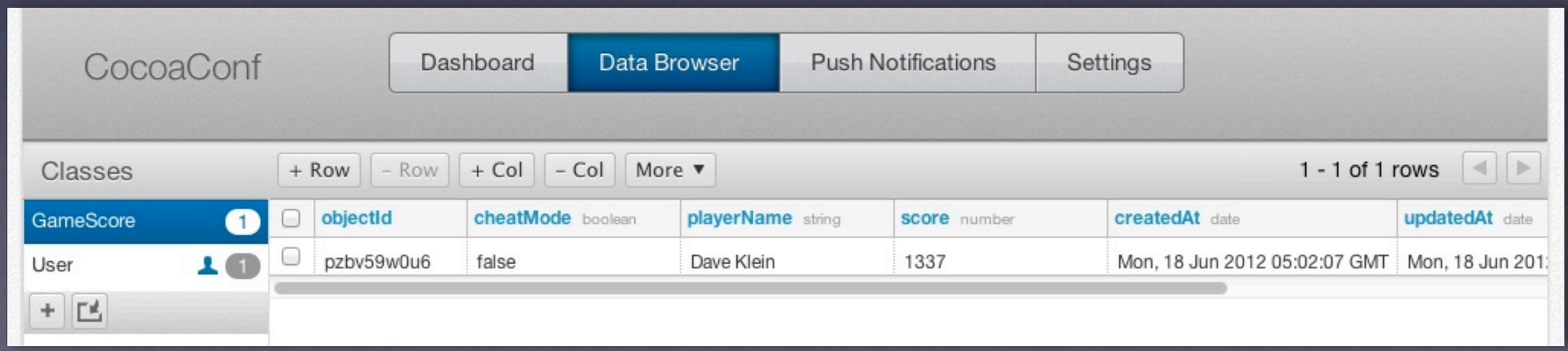
PFObject

- A PFObject contains key-value pairs of JSON-compatible data
- Strings, numbers, booleans, arrays, and dictionaries; plus date, data, and null
- No schema required
- You do provide a "class name" for grouping

Saving an Object

- Construct a PFObject, give a class name, assign values, save.

```
PFObject *gameScore = [PFObject objectWithClassName:@"GameScore"];
[gameScore setObject:@"Dave Klein" forKey:@"playerName"];
[gameScore setObject:[NSNumber numberWithInt:1337] forKey:@"score"];
[gameScore setObject:[NSNumber numberWithBool:NO] forKey:@"cheatMode"];
[gameScore save];
```



Retrieving an Object

- Construct a PFQuery, fetch by constraint

```
PFQuery *query = [PFQuery queryWithClassName:@"GameScore"];

// To fetch by objectId
PFObject *gameScore = [query getObjectWithId:@"pzbv59w0u6"];

// To fetch by properties
[query whereKey:@"playerName" containsString:@"Klein"];
[query whereKey:@"score" greaterThan:[NSNumber numberWithInt:1000]];
PFObject *gameScore = [query getFirstObject];
```

Viewing an Object

- Treat it like a dictionary

```
int score = [[gameScore objectForKey:@"score"] intValue];
NSString *playerName = [gameScore objectForKey:@"playerName"];
BOOL cheatMode = [[gameScore objectForKey:@"cheatMode"] boolValue];

NSString *objectId = gameScore.objectId;
NSDate *updatedAt = gameScore.updatedAt;
NSDate *createdAt = gameScore.createdAt;

// Later, to update with more recent data from the server
[myObject refresh];
```

Access via REST

- Use REST to interact with the data from any server, such as to drive a support site

```
curl -X GET \
  -H "X-Parse-Application-Id: 1lOXFKISHhOnUZ..." \
  -H "X-Parse-REST-API-Key: T0EFfftAMheOn4cW..." \
  https://api.parse.com/1/classes/GameScore/pzbv59w0u6

{
  "playerName": "Dave Klein",
  "cheatMode": false,
  "score": 1337,
  "createdAt": "2012-06-18T05:02:07.619Z",
  "updatedAt": "2012-06-18T05:02:07.619Z",
  "objectId": "pzbv59w0u6",
  "ACL": {"ro4mVykHOd":{"write":true,"read":true}}
}
```

Many Data Types

```
NSNumber *number = [NSNumber numberWithInt:42];
NSString *string = [NSString stringWithFormat:@"the number is %i", number];
NSDate *date = [NSDate date];
NSData *data = [@"foo" dataUsingEncoding:NSUTF8StringEncoding];
NSArray *array = [NSArray arrayWithObjects:string, number, nil];
NSDictionary *dictionary =
 [NSDictionary dictionaryWithObjectsAndKeys:number, @"number",
  string, @"string", nil];
NSNull *null = [NSNull null];

PFObject *bigObject = [PFObject objectWithClassName:@"BigObject"];
[bigObject setObject:number     forKey:@"myNumber"];
[bigObject setObject:string     forKey:@"myString"];
[bigObject setObject:date       forKey:@"myDate"];
[bigObject setObject:data       forKey:@"myData"];
[bigObject setObject:array      forKey:@"myArray"];
[bigObject setObject:dictionary forKey:@"myDictionary"];
[bigObject setObject:null       forKey:@"myNull"];
[bigObject save];
```

The JSON Data Model


```
{
   "myNull":null,
   "myData":{
      "__type":"Bytes",
      "base64":"Zm9v"
   },
   "myDate":{
      "__type":"Date",
      "iso":"2012-06-24T22:47:34.791Z"
   },
   "myNumber":42,
   "myDictionary":{
      "number":42,
      "string":"the number is 110430992"
   },
   "myArray":[
      "the number is 110430992",
      42
   ],
   "myString":"the number is 110430992",
   "createdAt":"2012-06-24T22:47:31.925Z",
   "updatedAt":"2012-06-24T22:47:31.925Z",
   "objectId":"559eXflpSv",
   "ACL":{"ro4mVykHOd":{"write":true,"read":true}}
}
```

Background Save

- Network access shouldn't go on the main thread

```
[gameScore saveInBackground];  // no callback

[gameScore saveInBackgroundWithBlock:^(BOOL succeeded, NSError *error) {
    if (!error) {
        // The gameScore saved successfully.
    } else {
        // There was an error saving the gameScore.
    }
}];

[gameScore saveEventually];  // works even if presently offline!
```

Background Query

```
PFQuery *query = [PFQuery queryWithClassName:@"GameScore"];
[query getObjectInBackgroundWithId:@"pzbv59w0u6"
                             block:^(PFObject *gameScore, NSError *error) {
        if (!error) {
            // Request succeeded, log the score
            NSLog(@"Score was: %d", [[gameScore objectForKey:@"score"] intValue]);
        } else {
            // Log the failure
            NSLog(@"Error: %@ %@", error, [error userInfo]);
        }
    }];
```

Relational Data

Modeling Relationships

- Just store a PFObject as a value inside another PFObject

```
PFObject *myPost = [PFObject objectWithClassName:@"Post"];
[myPost setObject:@"CocoaConf" forKey:@"title"];
[myPost setObject:@"The next CocoaConf will be in..." forKey:@"content"];

PFObject *myComment = [PFObject objectWithClassName:@"Comment"];
[myComment setObject:@"Great, right next to me!" forKey:@"content"];

// Add a relation between the Post and Comment
[myComment setObject:myPost forKey:@"parent"];

// This will save both myPost and myComment
[myComment saveInBackground];
```

Modeling Relationships

- Or refer by objectId

```
[myComment setObject:
    [PFObject objectWithoutDataWithClassName:@"Post" objectId:@"1zEcyElZ80"]
    forKey:@"parent"];
```

Faulting

- By default, related PFObjects must be explicitly fetched

```
PFObject *post = [fetchedComment objectForKey:@"parent"];
[post fetchIfNeededInBackgroundWithBlock:^(PFObject *object, NSError *error) {
  NSString *title = [post objectForKey:@"title"];
}];
```

Sets of Objects

- To model many-to-many use a PFRelation
  - Like an array, but with more control

- A user can "like" many blog posts

```
PFUser *user = [PFUser currentUser];
PFRelation *relation = [user relationforKey:@"likes"];
[relation addObject:post];
[user saveInBackground];
```

Sets of Objects

- Retrieve all stored likes about "Cocoa"

```
PFUser *user = [PFUser currentUser];
PFRelation *relation = [user relationforKey:@"likes"];
PFQuery *query = [relation query];
[query whereKey:@"title" containsString:@"Cocoa"];
[query findInBackgroundWithBlock:^(NSArray *result, NSError *error) {
   if (error) {
      // There was an error
   } else {
      // result has all the Cocoa posts the current user liked.
   }
}];
```

More Advanced Query

- Find comments for posts with images

```
PFQuery *innerQuery = [PFQuery queryWithClassName:@"Post"];
[innerQuery whereKeyExists:@"image"];
PFQuery *query = [PFQuery queryWithClassName:@"Comment"];
[query whereKey:@"post" matchesQuery:innerQuery];

[query findObjectsInBackgroundWithBlock:^(NSArray *comments, NSError *error) {
   // comments holds the comments for posts with images
}];
```

Caching

- Parse maintains a client-side cache for disconnected access
- On a query you specify the cache policy:
  - Don't use a cache
  - Only use a cache
  - Pull from net, but save results to cache
  - Try the cache, fallback to network
  - Try the network, fallback to cache
  - Try the cache first, then use network to verify

Large Files

Saving Opaque Files

- A PFFile can store up to 10 Megs of unindexed data (vs ~128 KB in a PFObject)
  - Constructed as NSData
  - Give a filename (doesn't have to be unique)
  - The extension matters; it controls the Content-Type used for direct downloads

Saving an Image

- Image -> NSData -> PFFile -> PFObject

```
NSData *imageData = UIImagePNGRepresentation(image);
PFFile *imageFile = [PFFile fileWithName:@"image.png" data:imageData];
[imageFile save];

PFObject *userPhoto = [PFObject objectWithClassName:@"UserPhoto"];
[userPhoto setObject:@"My trip to Hawaii!" forKey:@"imageName"];
[userPhoto setObject:imageFile             forKey:@"imageFile"];
[userPhoto save];
```

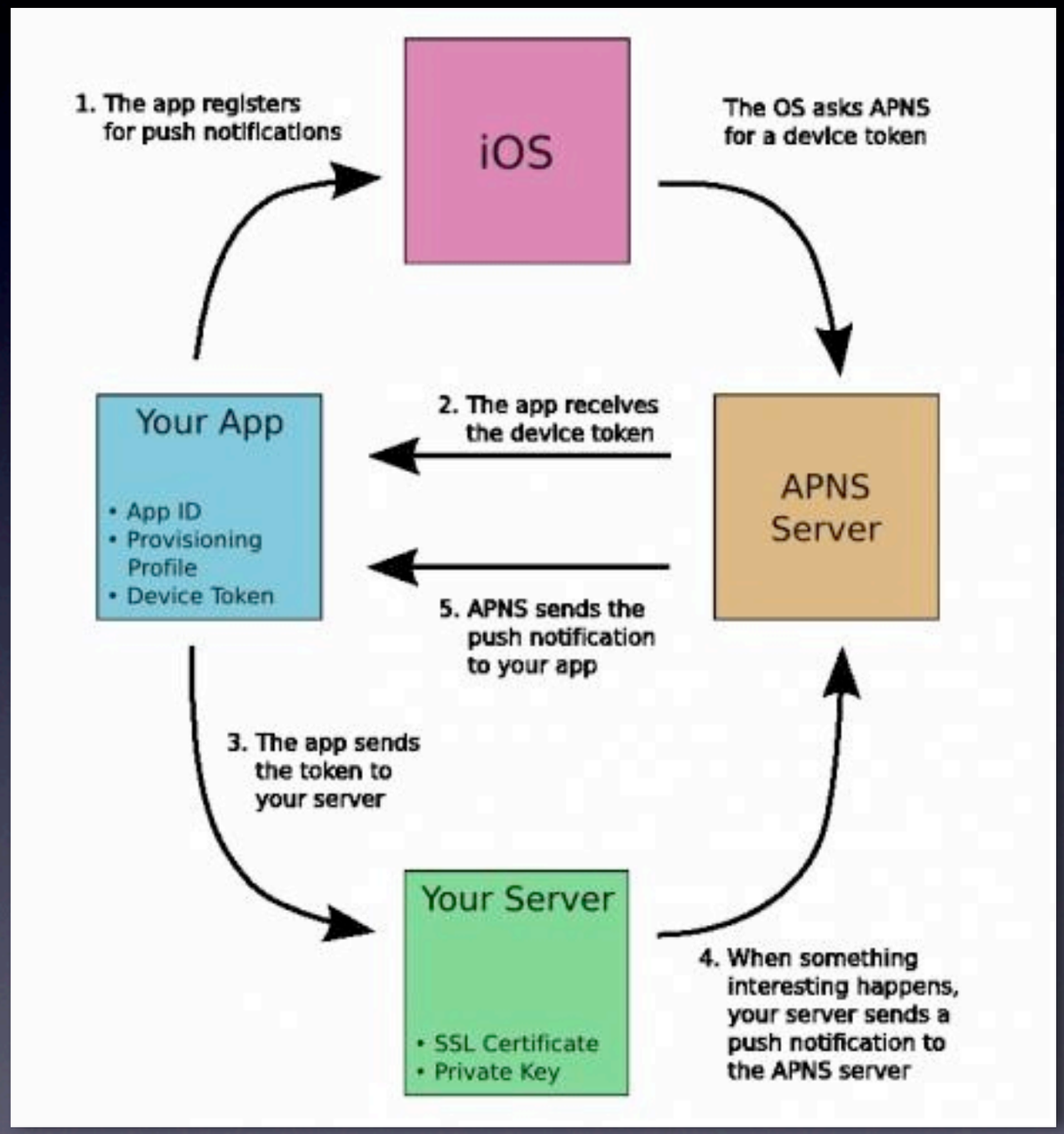
Watching Uploads

- Large uploads should be done in the background, can have progress tracking

```
NSData *data = ...;
PFFile *file = [PFFile fileWithName:@"big.mov" data:data];
[file saveInBackgroundWithBlock:^(BOOL succeeded, NSError *error) {
   // Handle success or failure here ...
} progressBlock:^(int percentDone) {
   // Update your progress here. percentDone will be between 0 and 100.
}];
```

Push Notifications

"Your Server"?



http://www.raywenderlich.com/3443/apple-push-notification-services-tutorial-part-12

Registration

- First tell the server about yourself

```
- (void)application:(UIApplication *)application
 didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)newDeviceToken
   {
   [PFPush storeDeviceToken:newDeviceToken];
   // Subscribe this user to the broadcast channel, ""
   [PFPush subscribeToChannelInBackground:@"" block:^(BOOL s, NSError *error) {
       if (s) {
          NSLog(@"Successfully subscribed to the broadcast channel.");
       } else {
          NSLog(@"Failed to subscribe to the broadcast channel.");
       }
   }];
}
```

Sending a Message

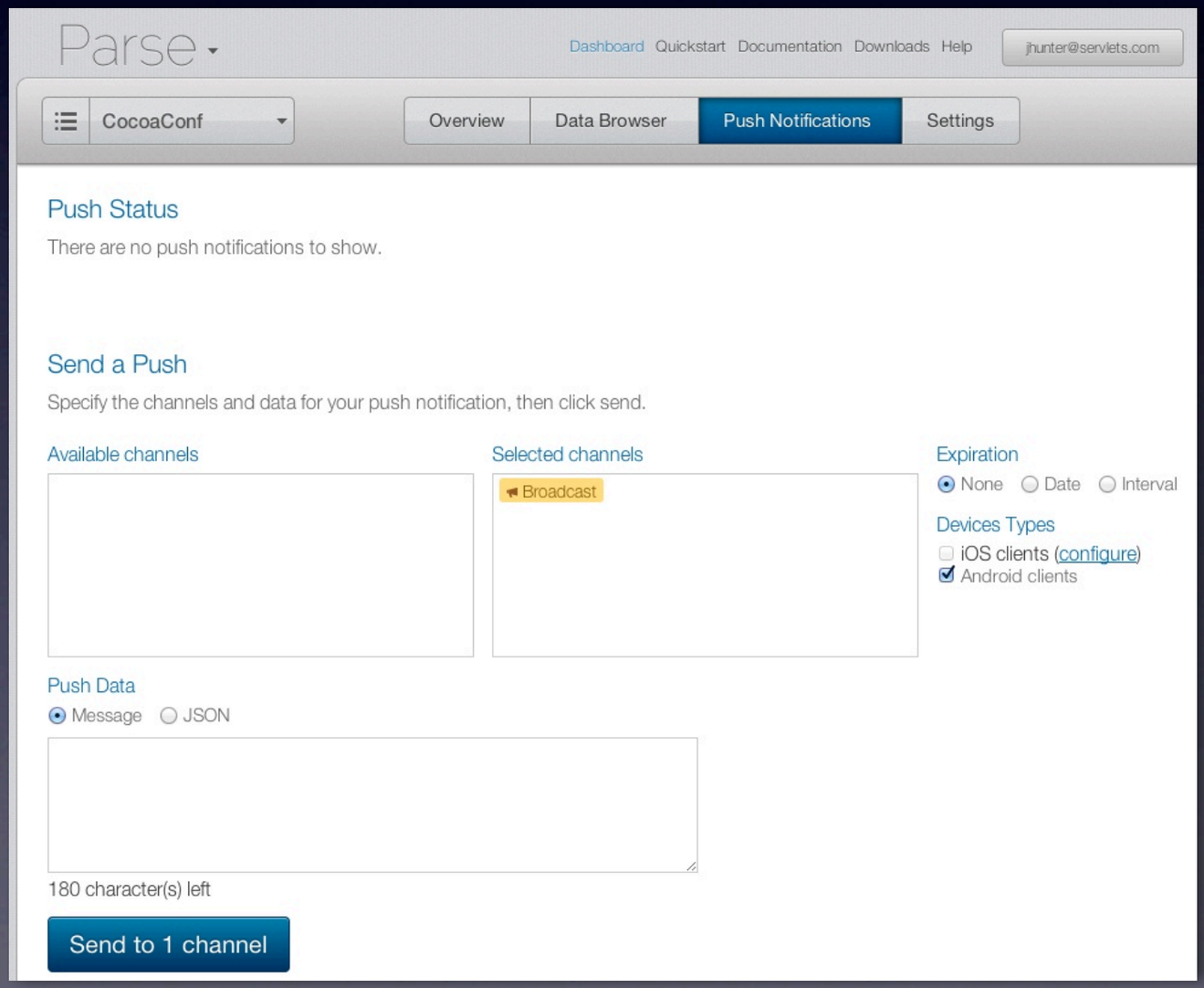
- Send via the SDK or REST

```
[PFPush sendPushMessageToChannelInBackground:@""
                                 withMessage:@"Hello World!"];
```

```
curl -X POST \
  -H "X-Parse-Application-Id: ${APPLICATION_ID}" \
  -H "X-Parse-REST-API-Key: ${REST_API_KEY}" \
  -H "Content-Type: application/json" \
  -d '{ "channel": "",
        "type": "ios"
        "data": { "alert":"Hello World!" } }' \
  https://api.parse.com/1/push
```

Sending a Message

- Or send from the web interface!



Receiving a Message

- The notification comes in the normal way
  - Parse provides a modal alert UI

```
- (void)application:(UIApplication *)application
 didReceiveRemoteNotification:(NSDictionary *)userInfo {
    [PFPush handlePush:userInfo];
}
```

Users

User Management

- Parse handles it all
  - Signing up, logging in, logging out, anonymous users, email verification, forgotten passwords, role-based security
- Login with Facebook or Twitter
- It even provides UI screens!

User Sign-Up

```
- (void)myMethod {
    PFUser *user = [PFUser user];
    user.username = @"my name";
    user.password = @"my pass";
    user.email = @"email@example.com";

    // other fields can be set just like with PFObject
    [user setObject:@"415-392-0202" forKey:@"phone"];

    [user signUpInBackgroundWithBlock:^(BOOL succeeded, NSError *error) {
      if (!error) {
         // Hooray! Let them use the app now.
      } else {
         NSString *errorString = [[error userInfo] objectForKey:@"error"];
         // Probably the username is taken
      }
    }];
}
```

User Log-In

```
[PFUser logInWithUsernameInBackground:@"myname" password:@"mypass"
                block:^(PFUser *user, NSError *error) {
                    if (user) {
                        // Do stuff after successful login.
                    } else {
                        // The login failed. Check error to see why.
                    }
                }];
```

Email Verification

- Parse can verify emails for you (optional)
- Fires an email with a clickable link
- Results held in the emailVerified property

Accessing the User

```
PFUser *currentUser = [PFUser currentUser];
if (currentUser) {
    // Do stuff with the user
    NSString *phone = [user objectForKey:@"phone"];
    [currentUser incrementKey:@"runCount"];
    [currentUser saveInBackground];
} else {
    // Show the signup or login screen
}
```

Anonymous Users

- You can start a user as anonymous
  - Holds data like any other user
  - Can "upgrade" by signing up, or linking with Twitter or Facebook
  - On logout (or login) their data will be tossed
  - Can be created w/o network access!

Automated Anonymity

```
[PFUser enableAutomaticUser];
[[PFUser currentUser] incrementKey:@"runCount"];
[[PFUser currentUser] saveInBackground];

if ([PFAnonymousUtils isLinkedWithUser:[PFUser currentUser]]) {
   [self enableSignUpButton];
} else {
   [self enableLogOutButton];
}
```

Security

- Parse can restrict data access by user
  - List which users can read and write, or make an object publicly visible
  - Create ACLs for more advanced uses
  - Assign a default ACL for new data

Controlling Visibility

- The default is public visiblity (to anyone with your application or master key)

- You can override this on an object:

```
PFACL *readOnlyACL = [PFACL ACL];
[readOnlyACL setPublicReadAccess:YES];
[readOnlyACL setPublicWriteAccess:NO];
[object setACL:readOnlyACL];
```

Default ACLs

- You can re-assign the defaults:

```
PFACL *defaultACL = [PFACL ACL];

if (wantPublicReadAccess) {
   [defaultACL setPublicReadAccess:YES];
}
if (wantPublicWriteAccess) {
   [defaultACL setPublicWriteAccess:YES];
}

// Current user perms are (oddly) provided as the second arg
[PFACL setDefaultACL:defaultACL withAccessForCurrentUser:YES];
```

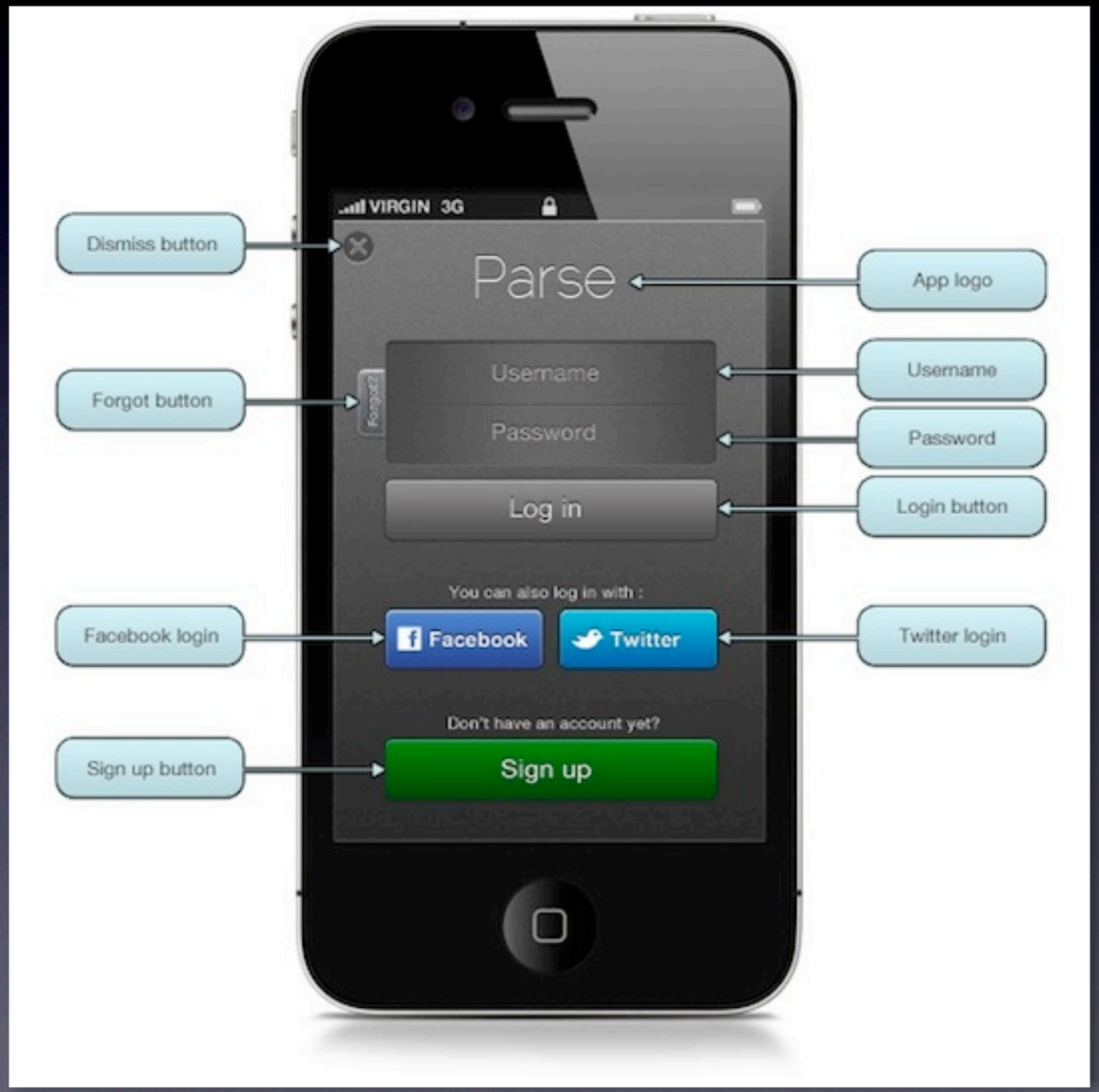
Graphical Interface

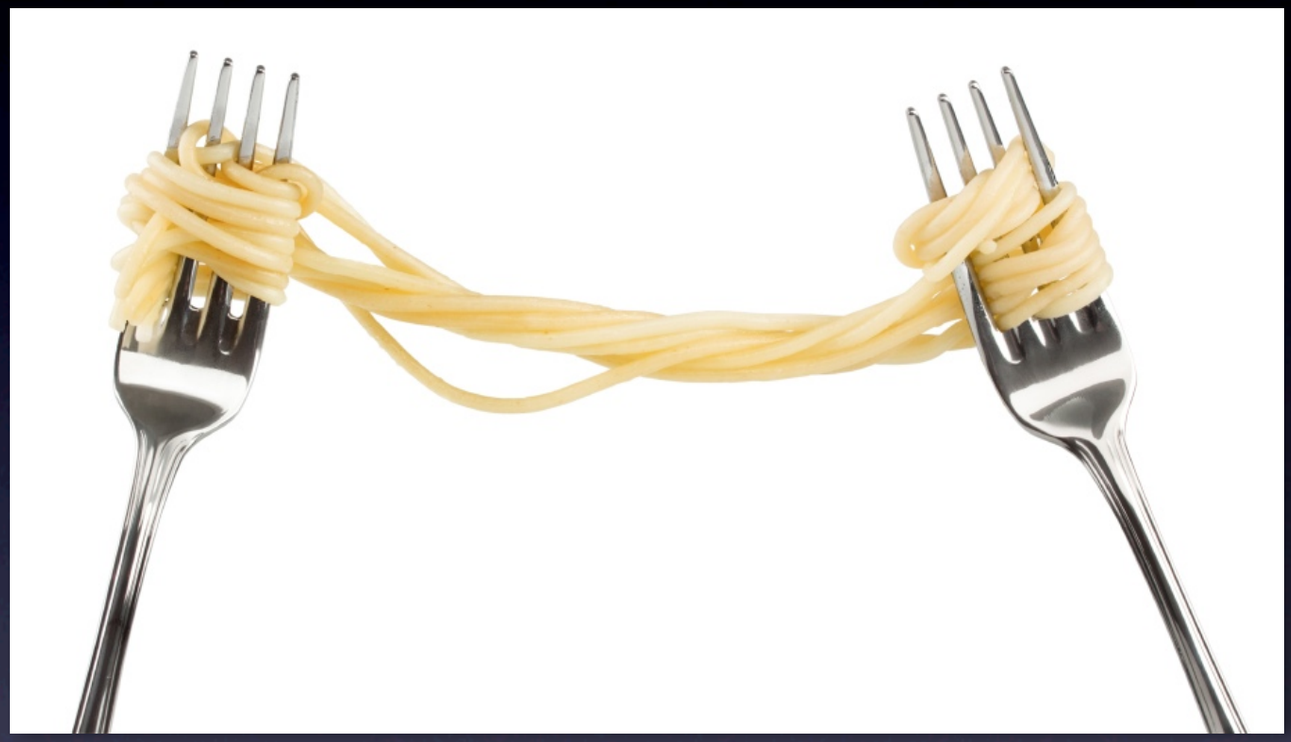
Parse Views

- Parse provides Views and Controllers to help with common tasks
  - Sign up, log in, results table, image view
  - Heavily customizable

PFSignUpViewController

PFLoginViewController





Syncing to CoreData

FTASync

- https://github.com/itsniper/FTASync
- Syncs CoreData to a Parse back-end
- Open source (MIT), new, experimental

FTASyncParent Entity

- Your entities extend FTASyncParent
  - String objectId
  - BOOL createdHere (Default: YES)
  - Int16 syncStatus (Default: 2)
  - Date updatedAt

Run the Sync

- Presently just global sync

```
[[FTASyncHandler sharedInstance] syncWithCompletionBlock:^{
      NSLog(@"Completion Block Called");
  } progressBlock:^(float progress, NSString *message) {
      NSLog(@"PROGRESS UPDATE: %f - %@", progress, message);
  }];
```

So Get Cooking!

blog.parse.com

Archives

22 Jun 2012	New Atomic Operations for Arrays
21 Jun 2012	AWS Resiliency Improvements
18 Jun 2012	The New App Dashboard
15 Jun 2012	Even More Ways to Protect Your Data
13 Jun 2012	Parse and Box Partner to Innovate in the Enterprise
13 Jun 2012	Recent Query Improvements
12 Jun 2012	Behind the Moustache
11 Jun 2012	More Ways to Lock Down Your Data
30 May 2012	Parse launches JavaScript SDK: Parse for Websites
25 May 2012	Let's Role!
22 May 2012	Import your CSV data to Parse
21 May 2012	Pushing to multiple iOS apps simultaneously
17 May 2012	PFFile: Because the Internet is not a Truck
17 May 2012	A More Scalable Many-to-Many Approach
16 May 2012	ACLs in the Data Browser
10 May 2012	A Redesigned App Management Console
07 May 2012	Express Your Brand With Parse
04 May 2012	Changelogs
30 Apr 2012	Managing Push Subscriptions with Installation Objects
26 Apr 2012	AnyWall: An Example Geolocation App with Tutorial
24 Apr 2012	Protect Your App By Limiting User Authentication Methods

Thanks!

- @hunterhacker
- jhunter@servlets.com