

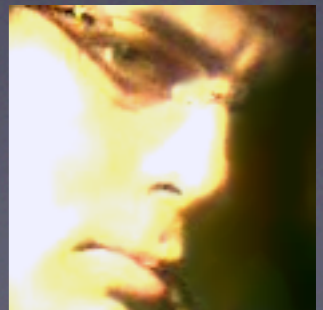
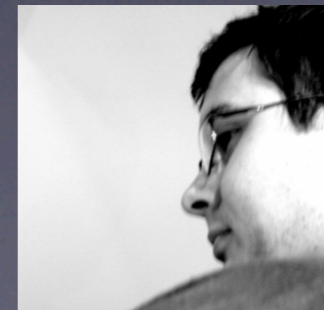
The Distributed Social Network

How we free ourselves, right now.

Hello...

Ben Ward

- ben@ben-ward.co.uk
- Web Developer at Yahoo! Europe
- Admin at microformats.org

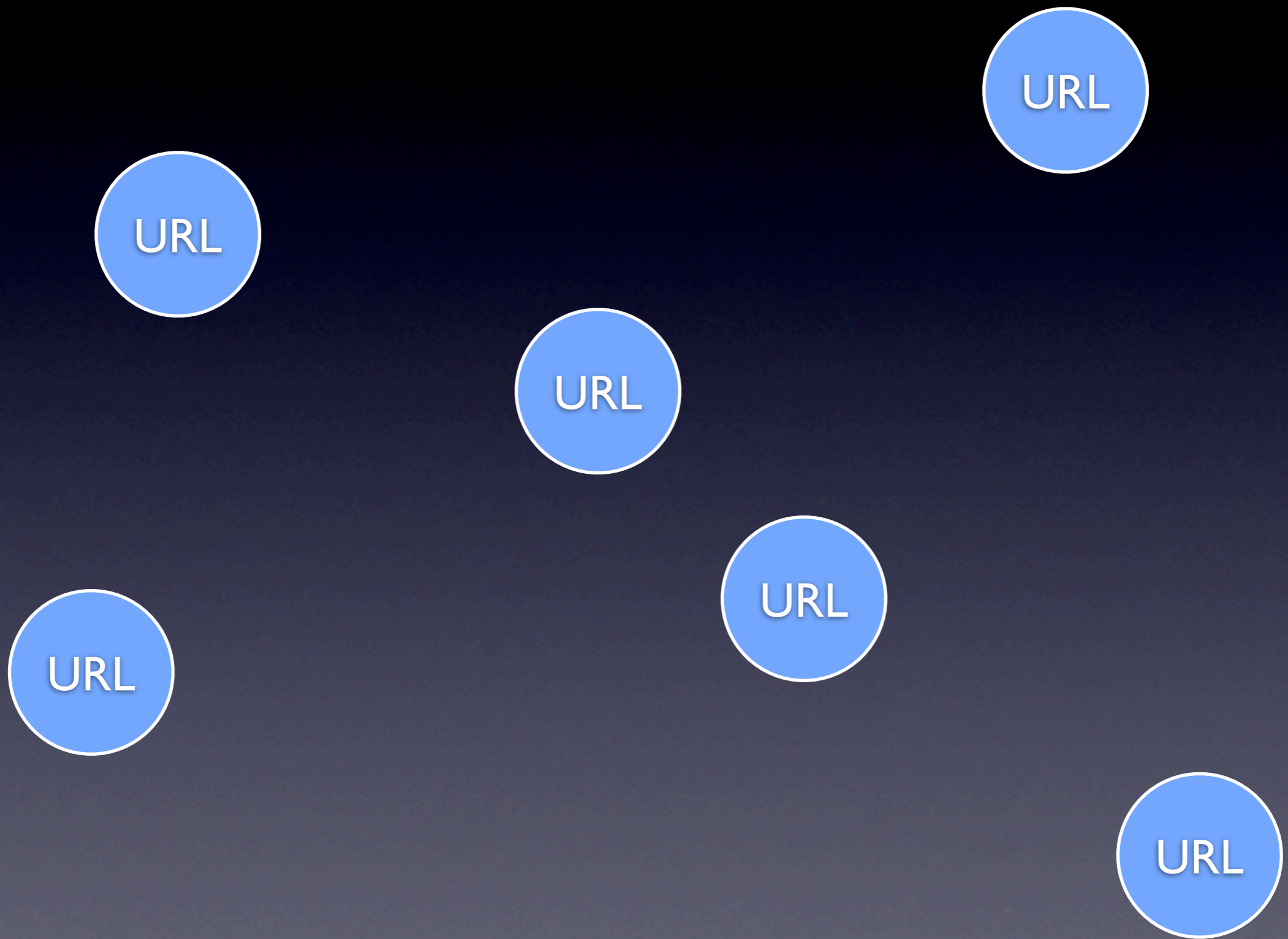


Introduce me.

Y!:

µf: Admin since last year, contributor to work on Listings format, recipe format (nom, etc.)

In the beginning...



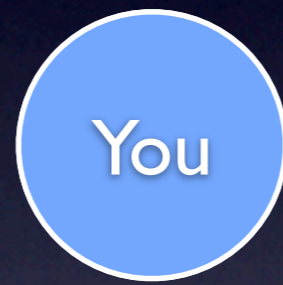
In the beginning... was the URL

... this was generally regarded as a good idea, and so **in short order** there were many URLs.

[Click]

- Beautifully
- URLs underpin everything we do on the internet.
- Technically, RESOURCES
- On the net, identity is a resource too

Identity



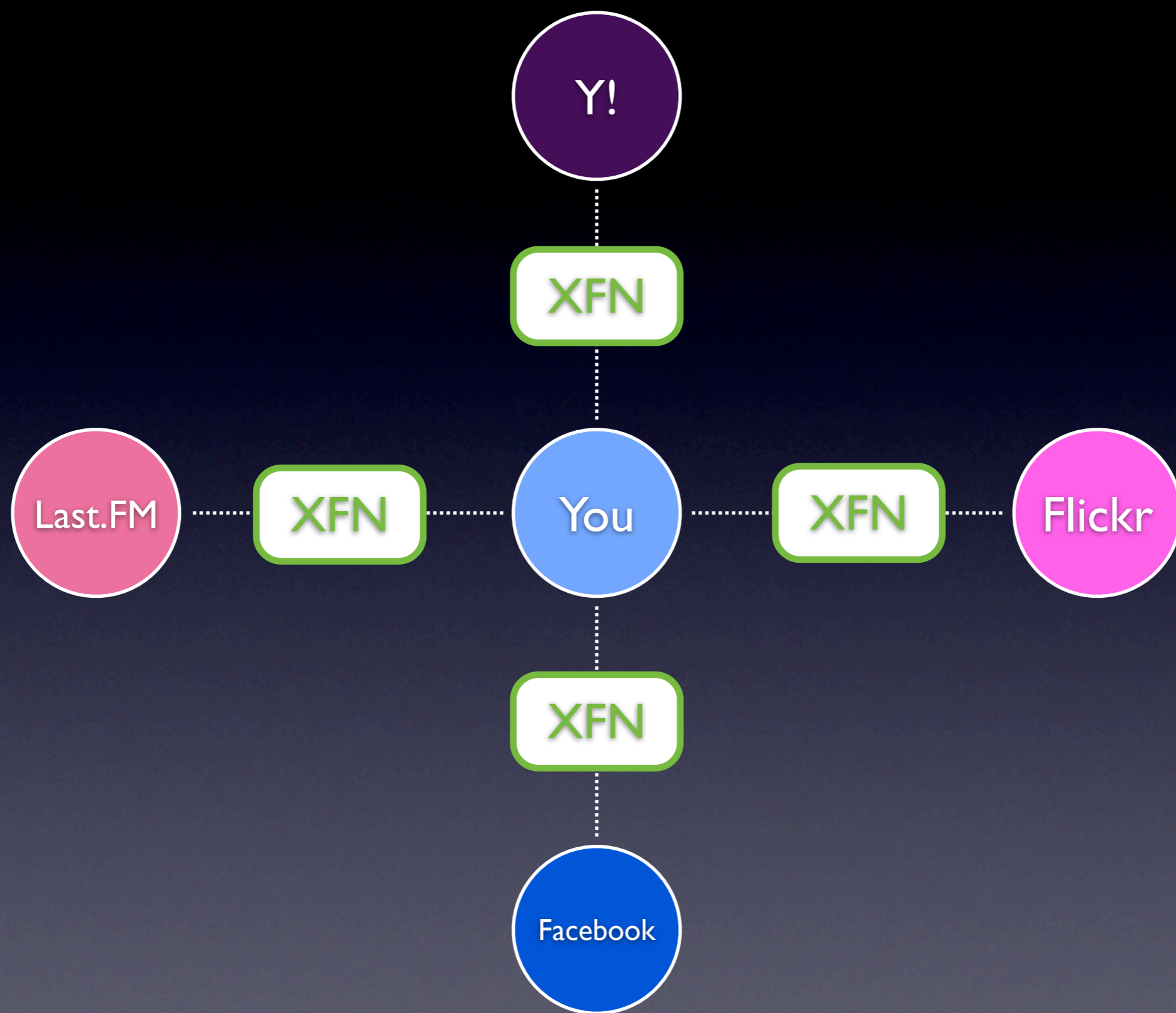
Even in the web before OpenID — which makes the whole thing more technical — single URLs have mapped to identity.

People have used URLs to informally represent themselves:

- Homepages
- Blogs
- Profiles

Therefore, some URLs are representing people; one of them represents you.

Disparate Identity



5

Except, the URL to Identity relationship is one-to-many; not 1:1.

[Click]

We all have one identity — well, most of us — but we have more than one identifying URL. We use hosted networks and services on other domains all the time. That's the whole point of decentralisation!

[Click]

The first step of building the decentralised social network is to link together these disparate parts. Inform the network of who you are, even when your identity is split across infinite boundaries.

[Click]

We do this with a technology called XFN: The XHTML Friends network. XFN is a microformat to describe social relationships on regular hyperlinks. It lets you link to other people, but it also lets you link to yourself.

Making Claims



```
<a href="http://flickr.com/photos/benward" rel="me">Photos</a>
```

```
<a href="http://ben-ward.co.uk" rel="me">Homepage</a>
```

Identity consolidation is built on a single property of XFN. `rel="me"`

[Click]

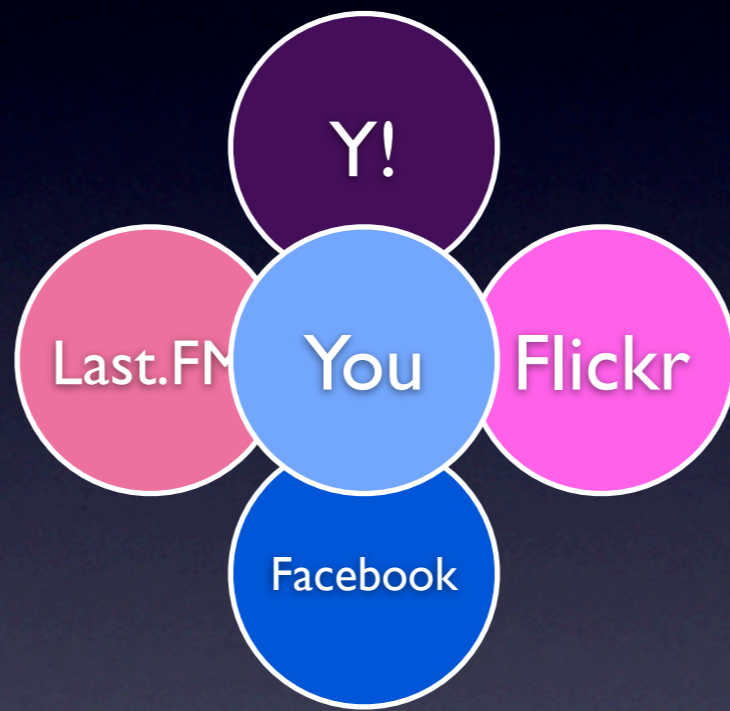
Where one site links to another with `rel=me`, that is a one-way relationship. It is a 'claim' that the source site represents the same identity as the target.

[Click]

Consolidation of identity requires that the second site links back to the source, also with `rel=me`. Thus both sites claim each other, and so the claim is verified.

Note that it doesn't communicate which node is the 'one true identifier'. That's a different matter, not actually a problem. Perhaps aesthetic.

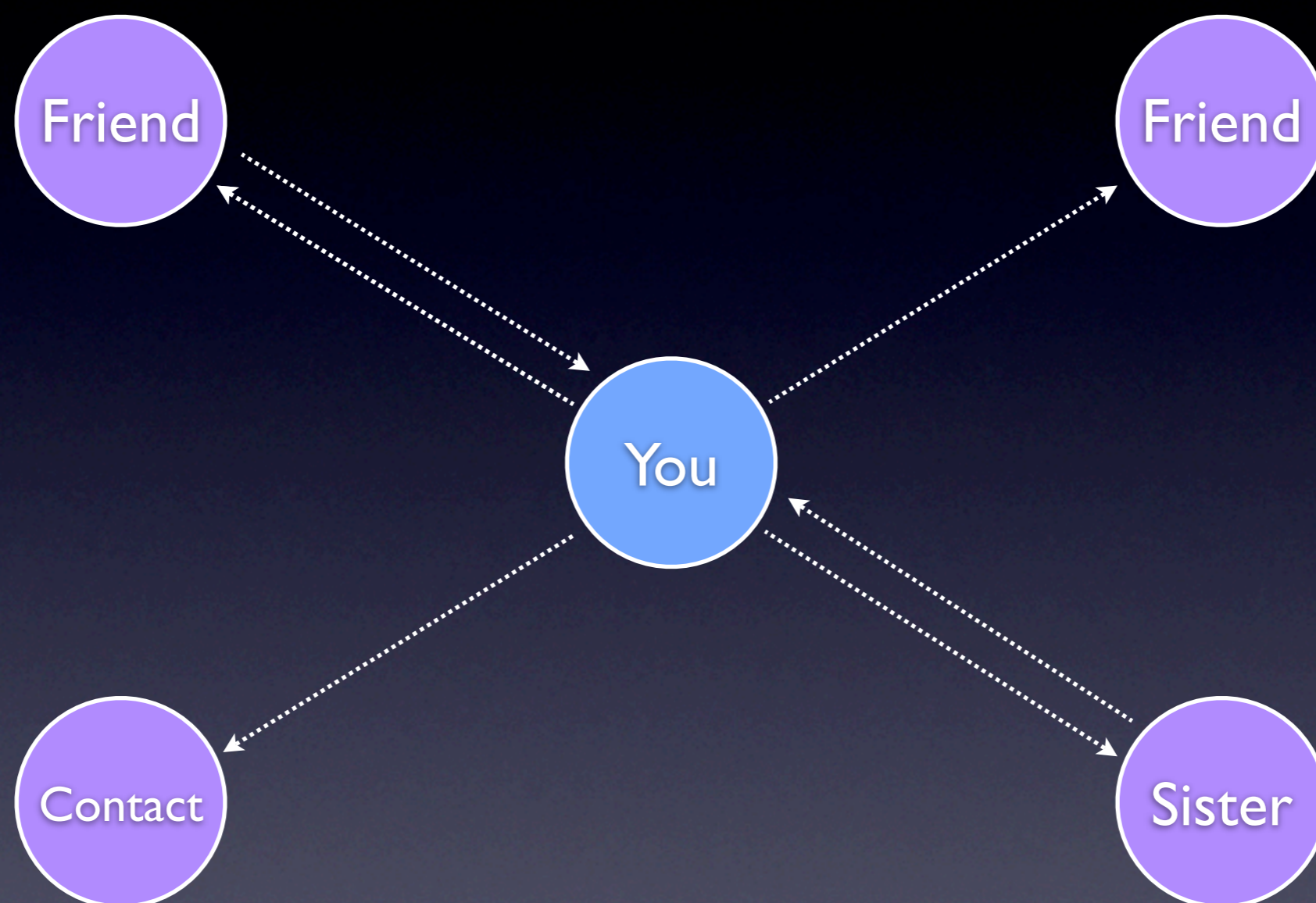
Consolidated Identity



So, with the relationships between each identity claimed, they can be consolidated.

More power to **you**

The Graph



XFN is used to consolidate disparate identity with `rel=me`, but it also describes relationships between different identities.

`rel=friend, acquaintance, contact`

The same claim system is used, enhancing hyperlinks between you and your friends. That could be on a blogroll, a passing mention in a blog post, or the address-book-esque display of a social network 'contacts' page.

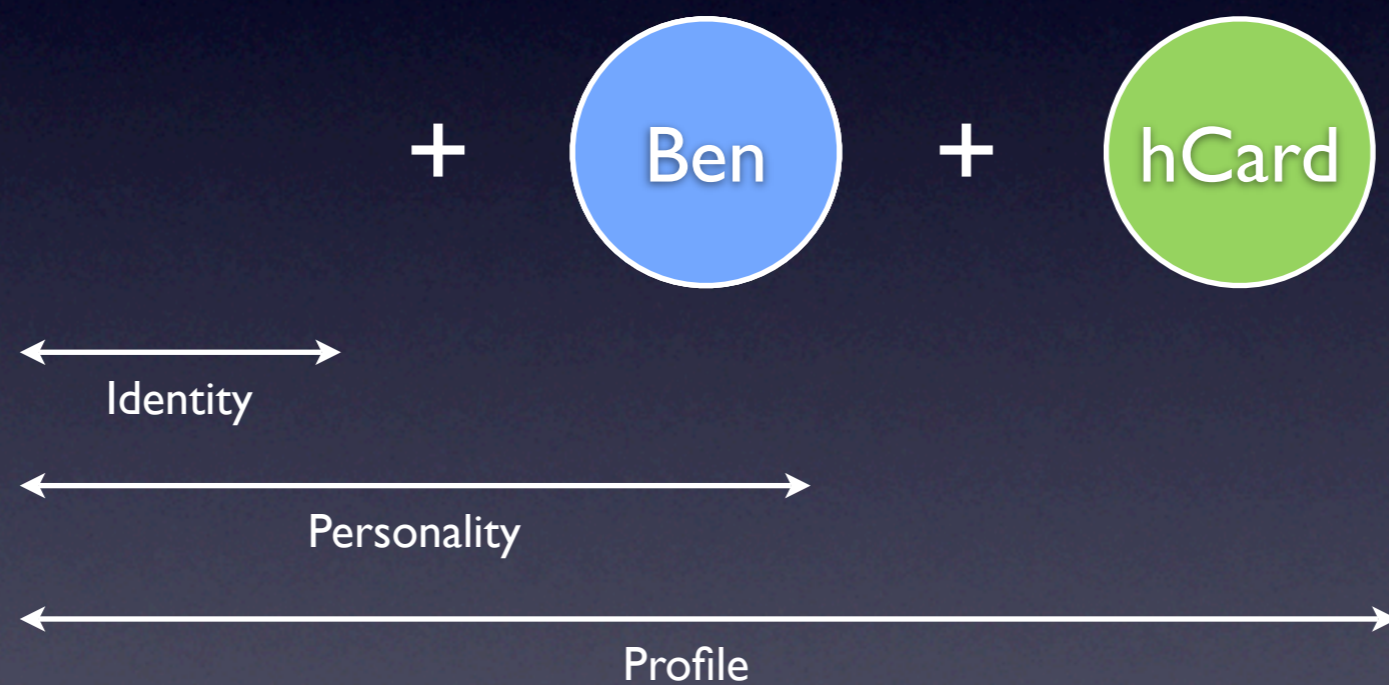
— that social network would be part of your consolidated identity, of course.

XFN Relationships

- Friends, acquaintances
- Contacts ('following')
- Family (kin, siblings, parents, spouse)

XFN is in fact more granular than current social networks require.
Three tier 'friendship' -> 'Contact'|'Acquaintance' is perhaps most accurate on most sites
Many-tier family -> Flickr just says 'family'

The Information



10

So we have **identity**, and we have **relationships**.

At this point --> **just related URLs**.

That's a useful service --> it's not enough.

At each URL --> **content**.

[Click]

Identity URL --> content is contextual to the identity.

That's **HTML**.

Normally, content **is the resource**, with identity is also **describes the resource**.

Turns '**Identity**' --> '**Personality**'

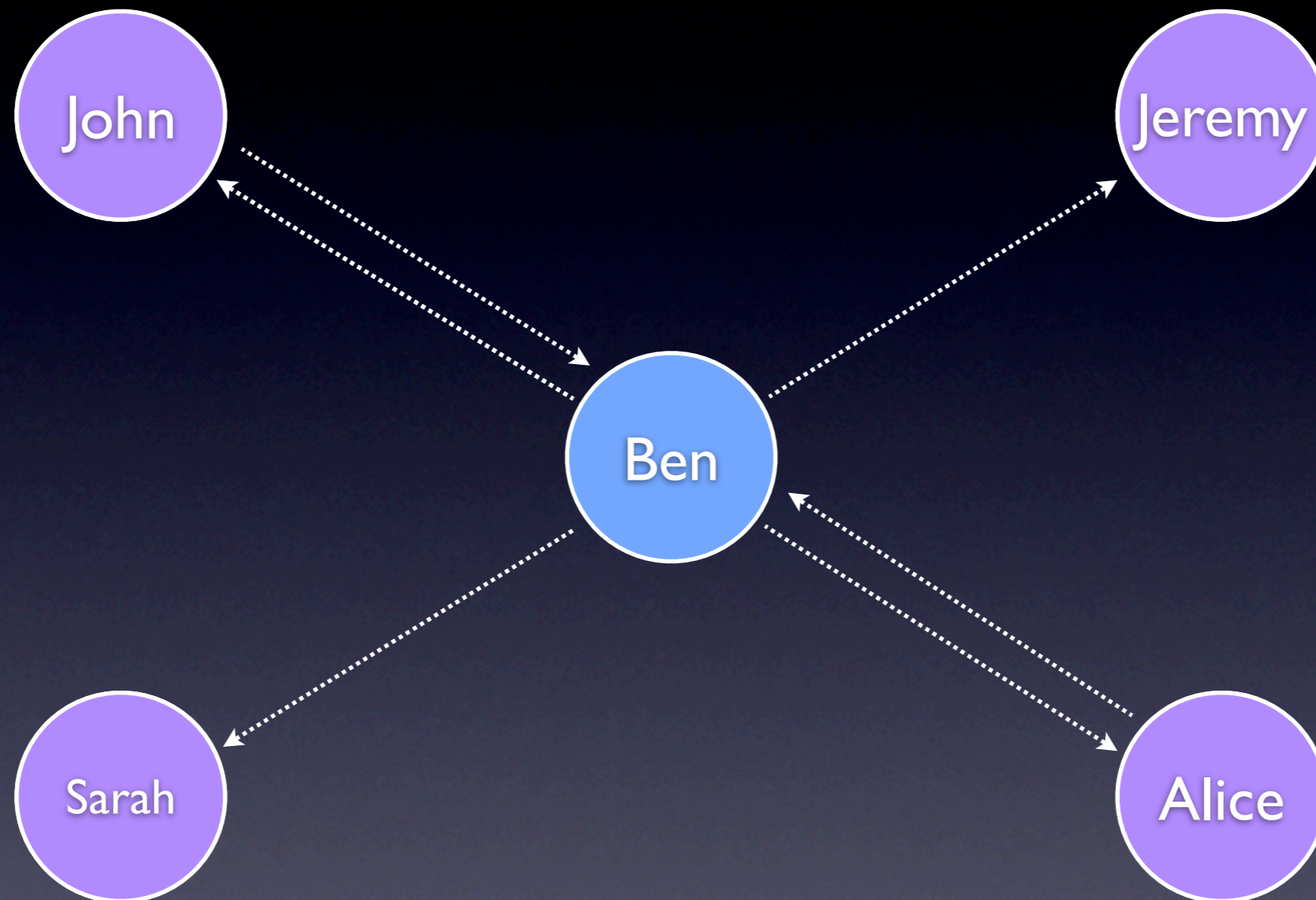
[Click]

Then **add hCard** (or FOAF). Structured profile information.

So we take identifiers and content, structure them, and take:

Identity -> Personality -> Profile -> a Person

The Network



So the social network is now enhanced:

- Relationships happen at the URL level
- Profiles are expressed at the HTML level

We stop working with **resources**
Start working with **people**

Consumption

- Use simple, powerful semantics
- Crawlable data
- Indexable data

So that's the information. How do we parse it?

2 real options:

- Crawling/Spidering – On demand, up-to-date, high cost query, SLOW
- Indexing (Goog, Y!... etc.) – Precached, low-cost query, fast, CACHE ROT

Google Social Graph API

<http://code.google.com/apis/socialgraph>

People who link to you as a contact

to  ben-ward.co.uk/

adactio.com/journal.php/ colleague friend met
adactio.com/journal/ acquaintance colleague friend met
cazmockett.blogspot.com/ acquaintance colleague met
dconstruct06.madgex.com/feeds/default.aspx acquaintance colleague met
dconstruct06.madgex.com/feeds/default.aspx colleague friend met
dconstruct06.madgex.com/feeds/default.aspx acquaintance colleague met
dconstruct06.madgex.com/feeds/post.aspx acquaintance colleague met
dconstruct06.madgex.com/feeds/post.aspx colleague friend met
fberriman.com/ co-worker colleague friend met
klauskomenda.com/about/ co-worker contact met
klauskomenda.com/code/geolinkr/ co-worker contact met
kurafire.net/log/ colleague friend met
kurafire.net/more/destinations colleague friend met
lab.backnetwork.com/examples/1/page1.htm contact met
↔ nascentguruism.com/ co-worker colleague friend met
sq.wordpress.com/ co-worker colleague friend met
thedredge.org/ colleague friend met
tomlovesyou.com/ colleague friend met
updowninbetween.wordpress.com/ acquaintance colleague met
v3.mysticg.com/ friend
veeliam.wordpress.com/ co-worker colleague friend met
alphanor.org/ friend
klauskomenda.com/ co-worker contact met
simonjobling.com/ colleague friend met
zaxbypass.com/ co-worker colleague friend met

The Google Social Graph is the first useful implementation of an index (Technorati did a search tool in their 'kitchen').

Already explained by others, but from a consumption POV, this is a snippet of the output you can get.

So, you want to build a social application?

- When I sign up, ***ask for my URL***
- ***Generate my profile*** from my hCard
- ***Find my contacts*** from Google's Social Graph API
- ***Build my buddy list*** from those contacts already in your database
- Save my time, and ***make me happy.***

- Ideally, the URL is going to be an OpenID.

Next steps

- Indexed profiles
- That 'Privacy' Thing