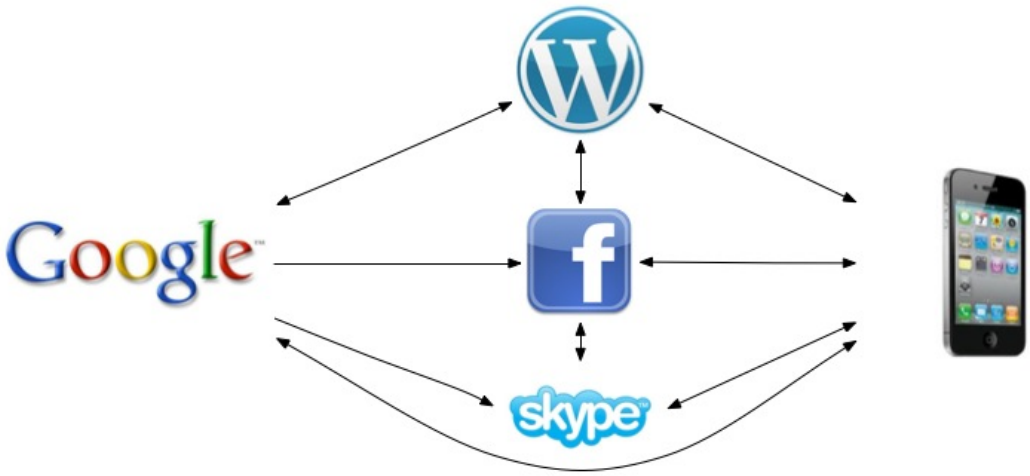

Tyler Sparks

Protocol Dispute:

Choosing the right tools for an API

-Background Info-

Web applications need the ability to connect to other data sources

Web applications may be written in a diverse set of languages (Ruby, PHP, ASP.NET, Python, Perl, etc)

2 types of base level request format: XML and JSON

These define the structure of information transmitted, not the rules of the request architecture

Examples of request architecture are RESTful, SOAP, CORBA, and JSON-RPC

-Key Ideas-

- API's facilitate the flow of information from one discrete application or location to another (ocassionally entire systems are built using an OO Private API model, but this is more of an exception)
- The choice of data fomat and request structure affects the applications that developers choose to connect to
- The web is all about organization and access of information, typically applications which support this paradigm are more succesful than those which do not

-Data Comparisons-

	JSON	XML
Performance	++	-
Human Readability	-	+
Reserve Words	--	++
Extensibility	-	++

Exchange Format	Data	Document
Support	Language Dependant	Language Dependant
Simplicity	++	+
Data Types	--	++
Security	--	++

JSON Format

```
[ {
"Record": "Contact",
"data": [
{
"value": "First",
"content":"James"
},
{
"value": "Last",
"content": "Smith"
},
{
"value": "Email",
"content": "me@ymail.com"
} ]
} ]
```

XML Format

```
<?xml version="1.0" ?>

    <record>

        <contact id="0540587">

            <first>James</first>
            <last>Smith</last>
            <email>me@ymail.com</email>

        </contact>

    </record>
```

-Request Comparisons-

	RESTful	SOAP
Supported Data Types	Any	XML
Return Type	Any + HTTP Response	XML
Protocol	HTTP(s)	*All
Information Retrieval	GET	Custom Request, Usually an RPC with POST
Information Submission	*PUT, POST	
Information Deletion	*DELETE	

Operation State	*Too simple to maintain state	Maintains state as part of of architecture
Asynchronous Calls	Can be done through additional tools (AJAX)	Easy to dictate as a part of the request
Caching	Yes	No

-Example Factors in Decision Process-

What platform will I be runnning on?

What language is the application built in?

How is the data best represented?

What integration partners am I interested in?

What are the device limitations?

What are the developers familiar with?

Do I need to scale?

-Conclusion-

REST is most conducive to the original design of the web, it uses the base components of the HTTP specification. If in doubt, use REST!

XML has the advantage of being human readable and works very nicely in the browser and with well formed REST implementations

Certain machines and situations may impose limitations best circumvented by changing the data type or request architecture

Small projects may not need to use a full request implementation (although they could arguably benefit from doing so)