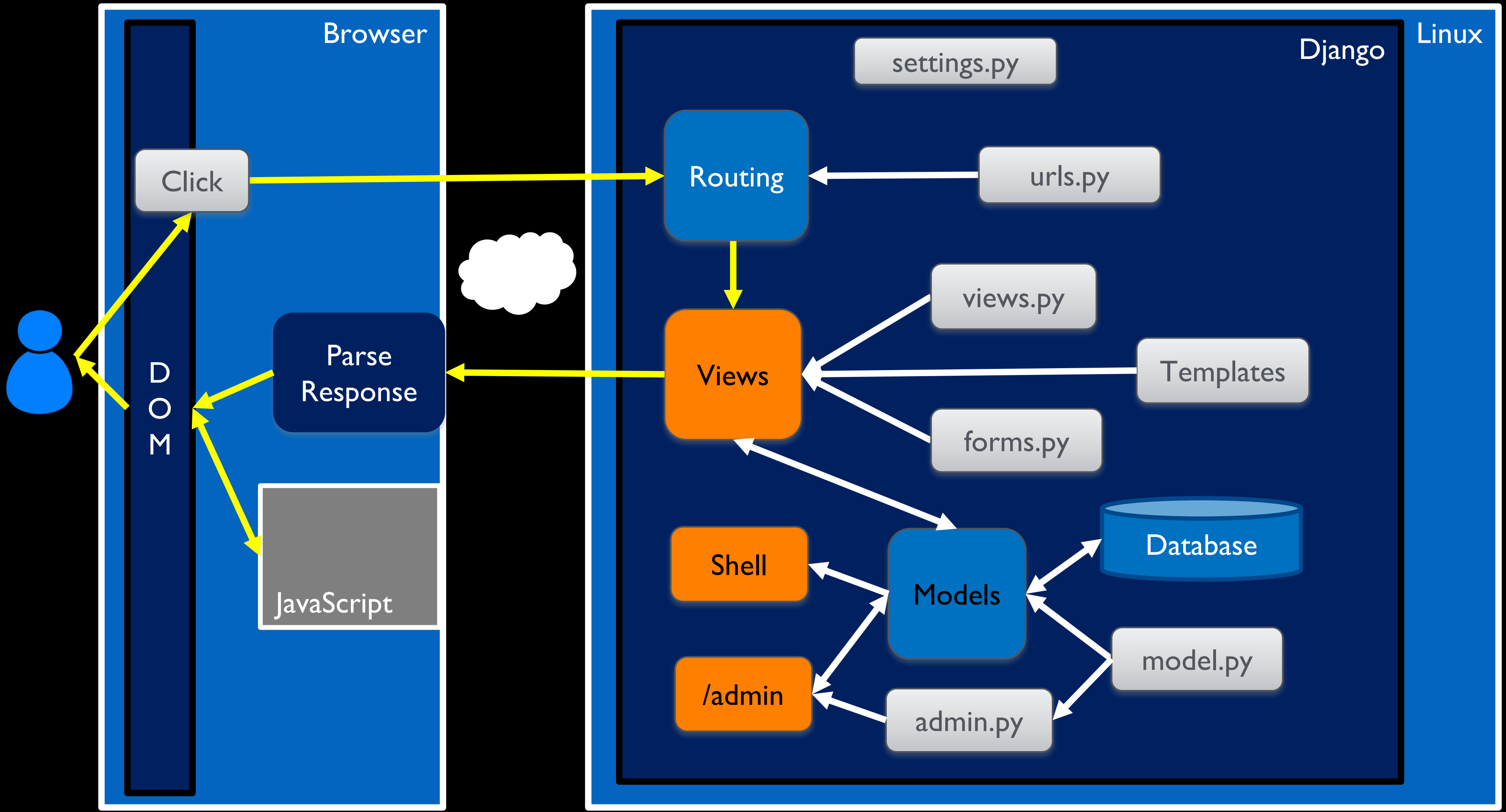
Table of Contents

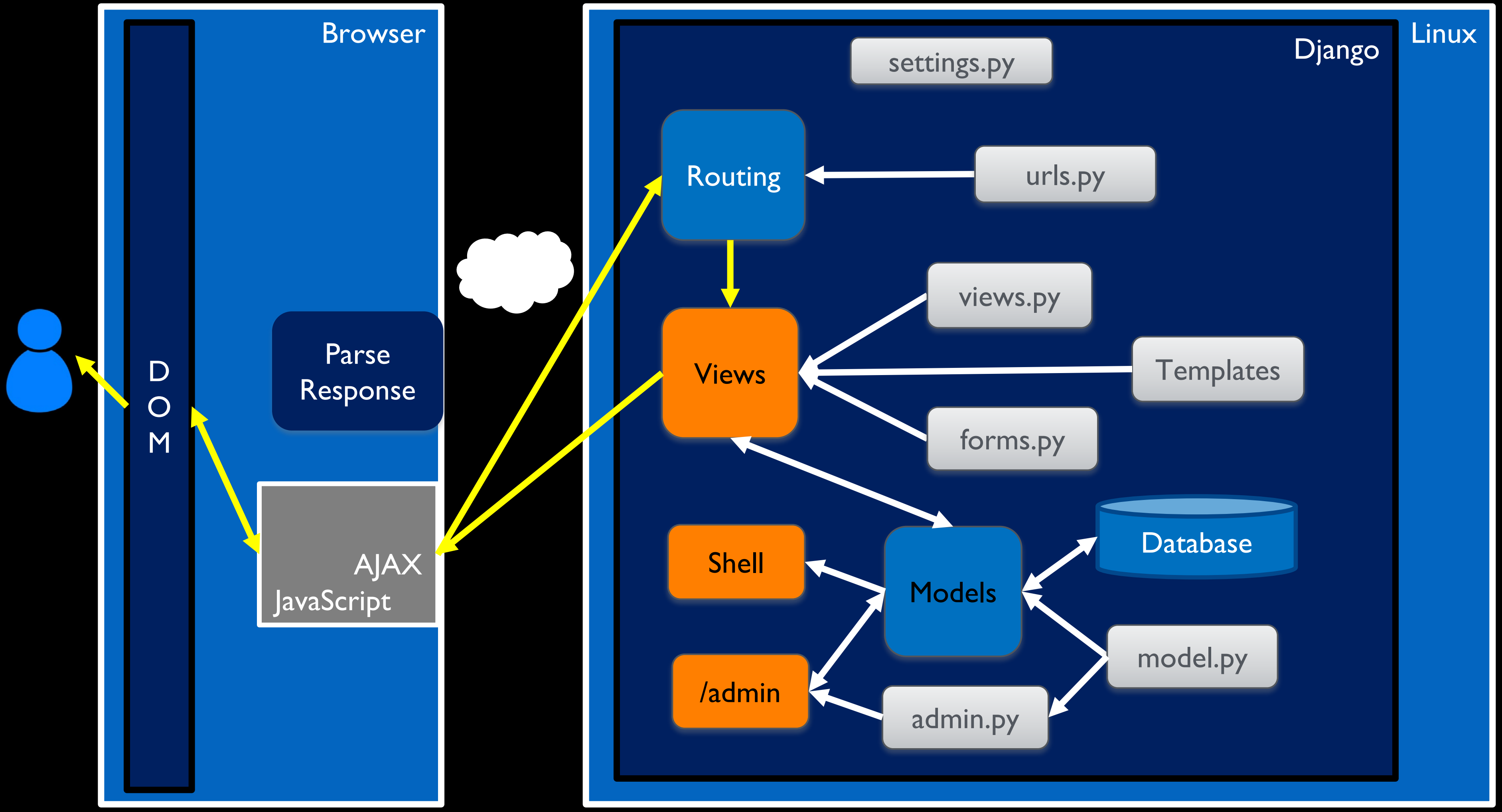
This slide deck consists of slides used in 2 lecture videos in Week 5. Below is a list of shortcut hyperlinks for you to jump into specific sections.

Using AJAX / JSON

Dr. Charles Severance
www.dj4e.com

https://samples.dj4e.com/chat
https://github.com/csev/dj4e-samples/tree/master/chat


Browser
Click
D O M
Parse Response
JavaScript
Linux
Django
settings.py
Routing
urls.py
Views
views.py
Templates
forms.py
Shell
Models
Database
/admin
model.py
admin.py

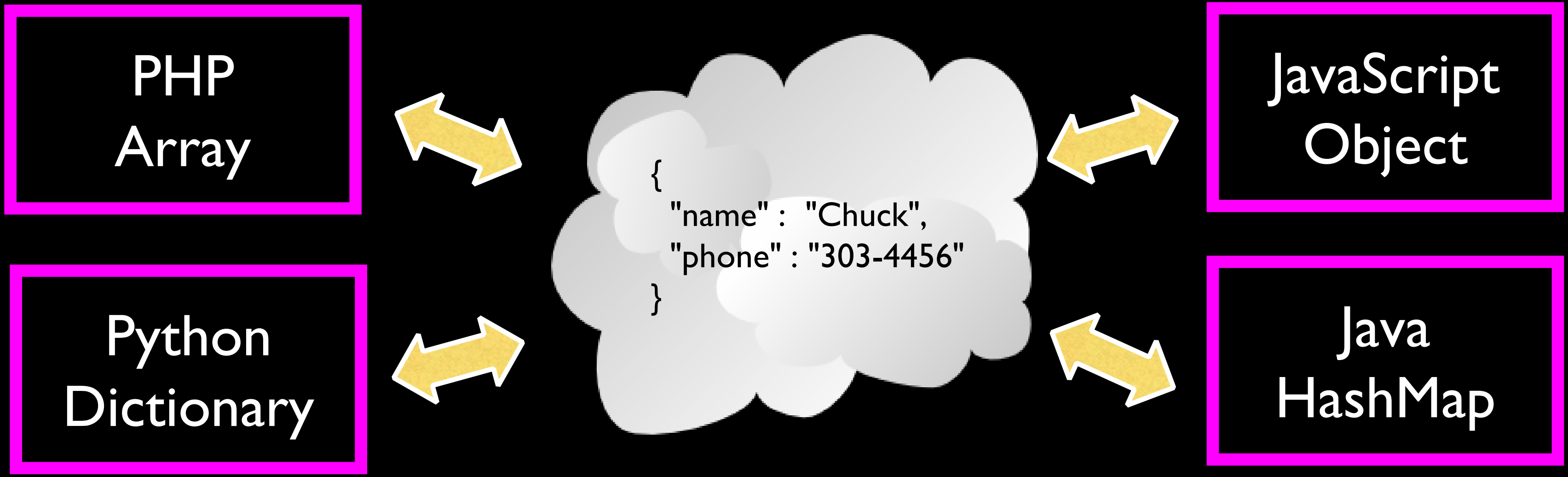

Browser
D O M
Parse Response
AJAX JavaScript
Linux
Django
settings.py
Routing
urls.py
Views
views.py
Templates
forms.py
Database
Shell
Models
model.py
/admin
admin.py

Data on the Web (2003)

- With the HTTP Request/Response well understood and well supported, there was a natural move toward exchanging data between programs using these protocols.

- We needed to come up with an agreed way to represent data going between applications and across networks.

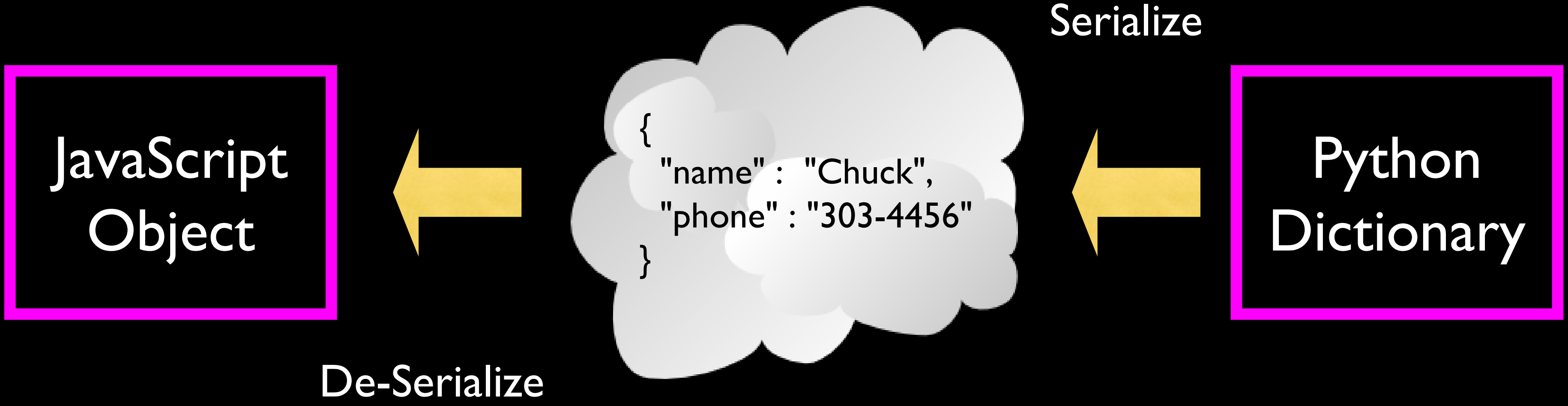
Agreeing on a “Wire Format”



```
{
  "name" :  "Chuck",
  "phone" : "303-4456"
}
```

a.k.a. “Wire Protocol” - What we send on the “wire”

JSON is a “Wire Format”

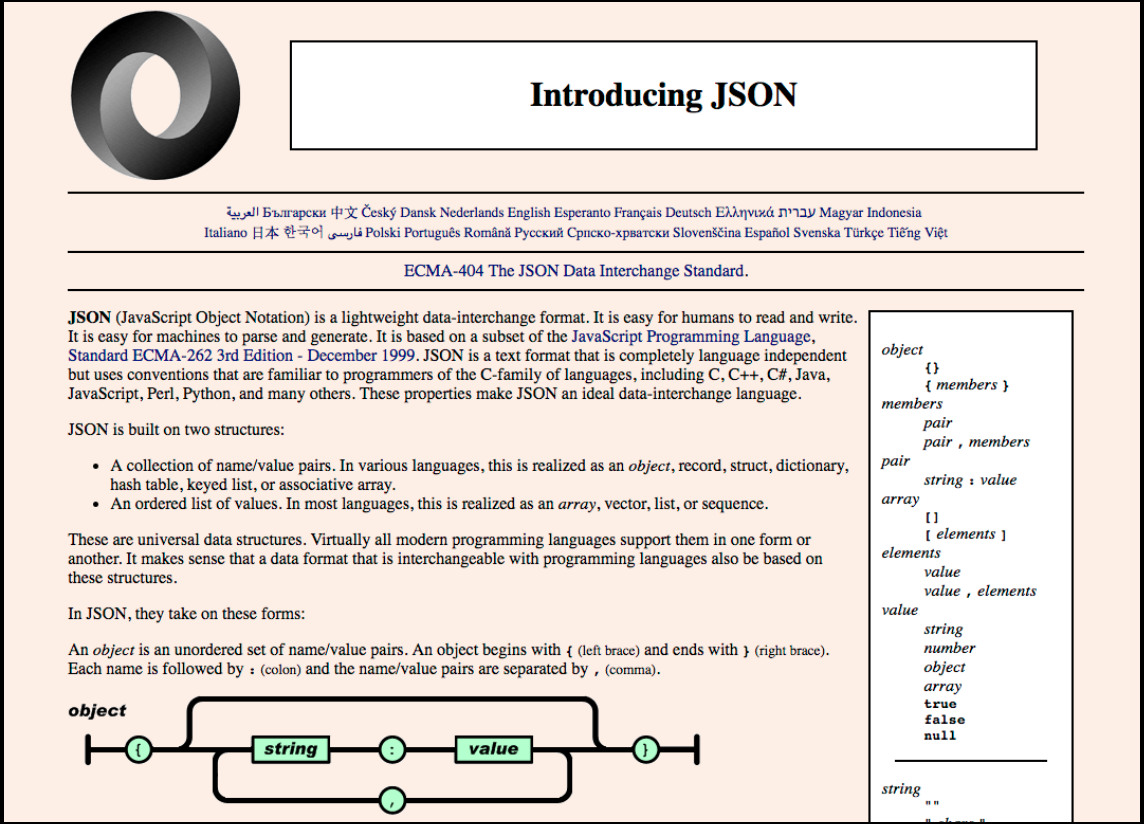


```
{
  "name" :  "Chuck",
  "phone" : "303-4456"
}
```

JavaScript Object Notation

- Douglas Crockford – "Discovered" JSON
- Object literal notation in JavaScript

https://www.youtube.com/watch?v=kc8BAR7SHJI



Introducing JSON

العربية Български 中文 Český Dansk Nederlands English Esperanto Français Deutsch Ελληνικά עברית Magyar Indonesia
Italiano 日本 한국어 فارسی Polski Português Română Русский Српско-хрватски Slovenščina Español Svenska Türkçe Tiếng Việt

ECMA-404 The JSON Data Interchange Standard.

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence.

These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures.

In JSON, they take on these forms:

An *object* is an unordered set of name/value pairs. An object begins with `{` (left brace) and ends with `}` (right brace). Each name is followed by `:` (colon) and the name/value pairs are separated by `,` (comma).

```
object
        {}
        { members }
members
        pair
        pair , members
pair
        string : value
array
        []
        [ elements ]
elements
        value
        value , elements
value
        string
        number
        object
        array
        true
        false
        null

string
        ""
```

www.json.org

Derived from the JavaScript "constant" syntax

Similar to Python Dictionary syntax

```
who = {
    "name": "Chuck",

    "age": 29,

    "college" : true,

    "offices" : [ "3350DMC", "3437NQ" ],

    "skills" : {
       "fortran": 10,
       "C++" : 5,
       "C": 10,
       "python" : '7'
    }
};
```

String

Integer

Boolean

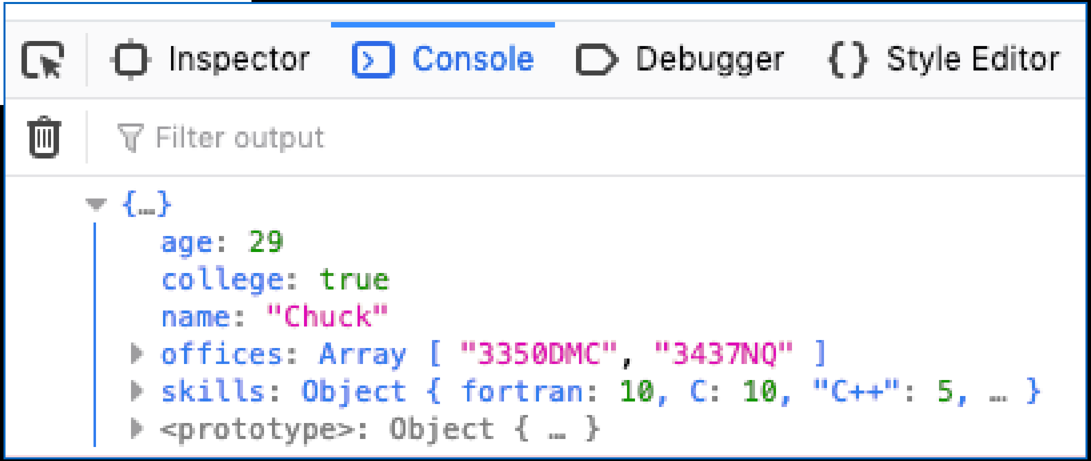
List/Array

Object

JSON Syntax

https://samples.dj4e.com/chat/syntax

```
<script type="text/javascript">
who = {
    "name": "Chuck",
    "age": 29,
    "college": true,
    "offices" : [ "3350DMC", "3437NQ" ],
    "skills" : { "fortran": 10, "C": 10,
        "C++": 5, "python" : 7 }
};
console.log(who);
</script>
```



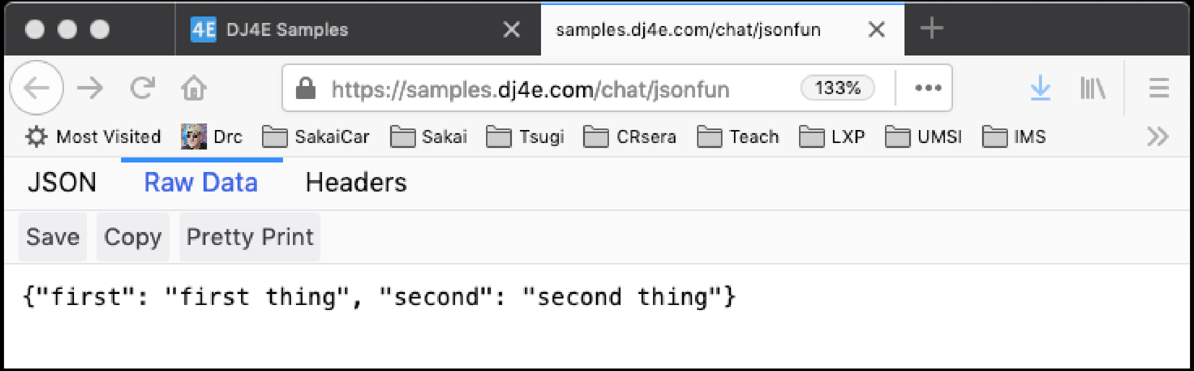
```
▾ {…}
    age: 29
    college: true
    name: "Chuck"
  ▸ offices: Array [ "3350DMC", "3437NQ" ]
  ▸ skills: Object { fortran: 10, C: 10, "C++": 5, … }
  ▸ <prototype>: Object { … }
```

https://samples.dj4e.com/chat/jsonfun

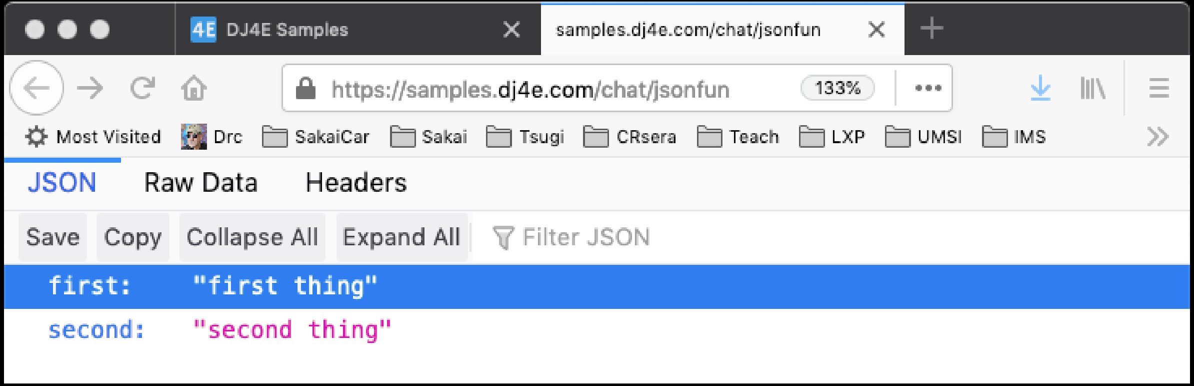
```
urls.py:

 path('jsonfun', views.jsonfun, name='jsonfun'),
```

```
views.py:

import time
from django.http import JsonResponse

def jsonfun(request):
    time.sleep(2)
    stuff = {
        'first': 'first thing',
        'second': 'second thing'
    }
    return JsonResponse(stuff)
```

https://samples.dj4e.com/chat/jsonfun

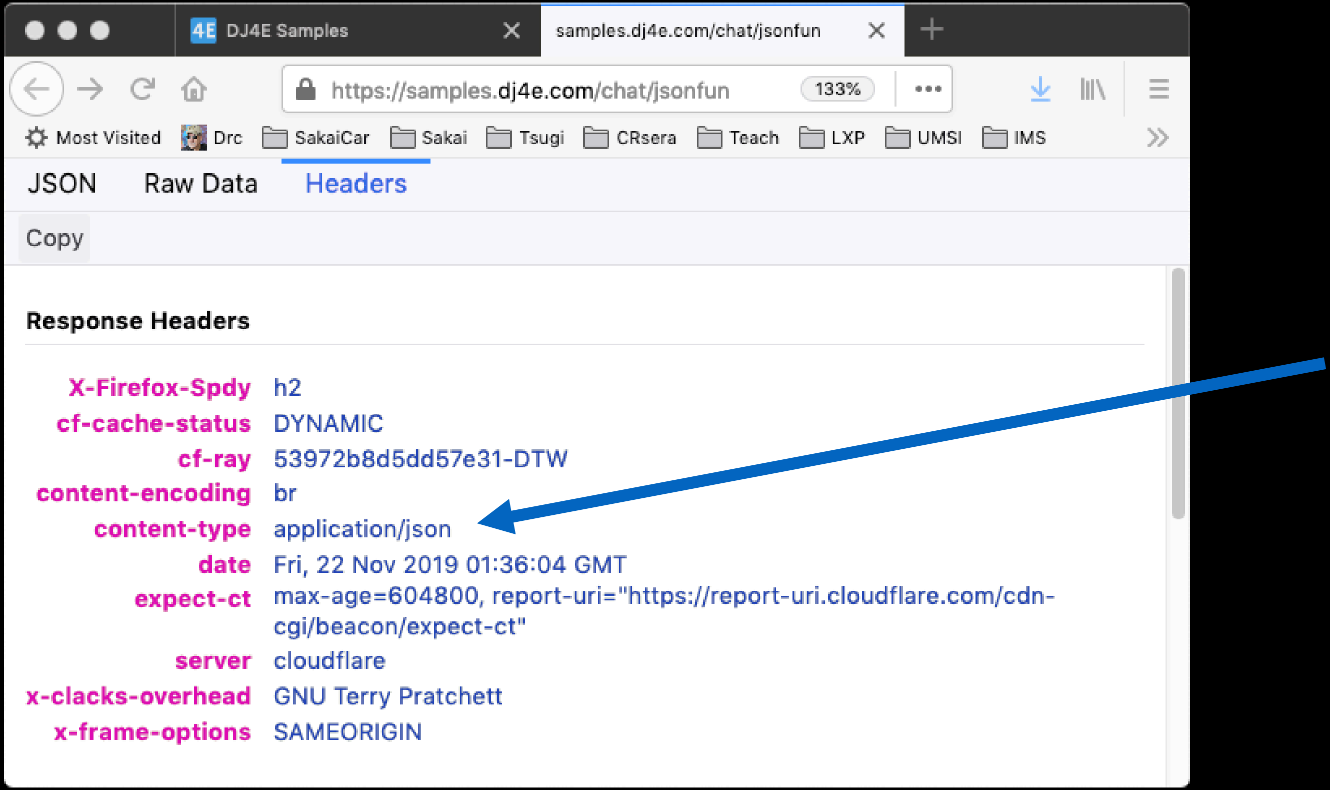


```
{"first": "first thing", "second": "second thing"}
```



```
first:    "first thing"
second:   "second thing"
```

https://samples.dj4e.com/chat/jsonfun

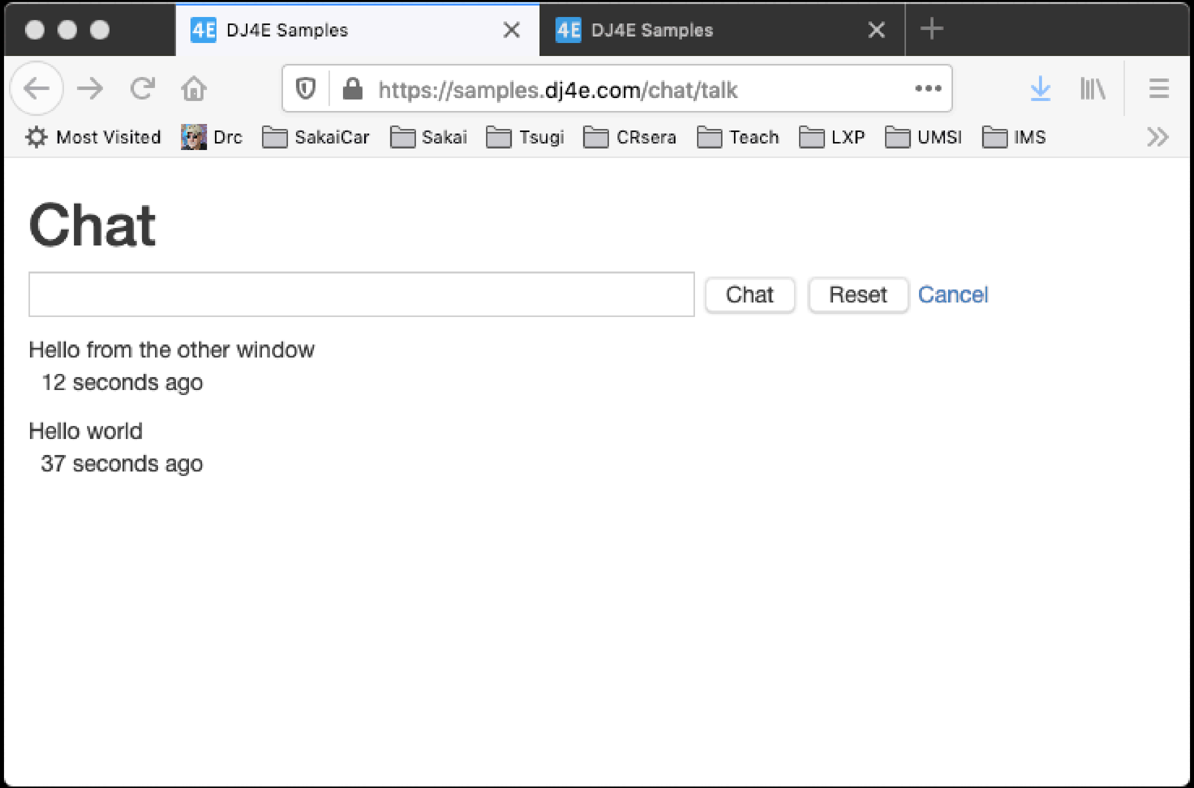


Response Headers

X-Firefox-Spdy	h2
cf-cache-status	DYNAMIC
cf-ray	53972b8d5dd57e31-DTW
content-encoding	br
content-type	application/json
date	Fri, 22 Nov 2019 01:36:04 GMT
expect-ct	max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
server	cloudflare
x-clacks-overhead	GNU Terry Pratchett
x-frame-options	SAMEORIGIN

A Chat App Using JSON

https://samples.dj4e.com/chat/talk

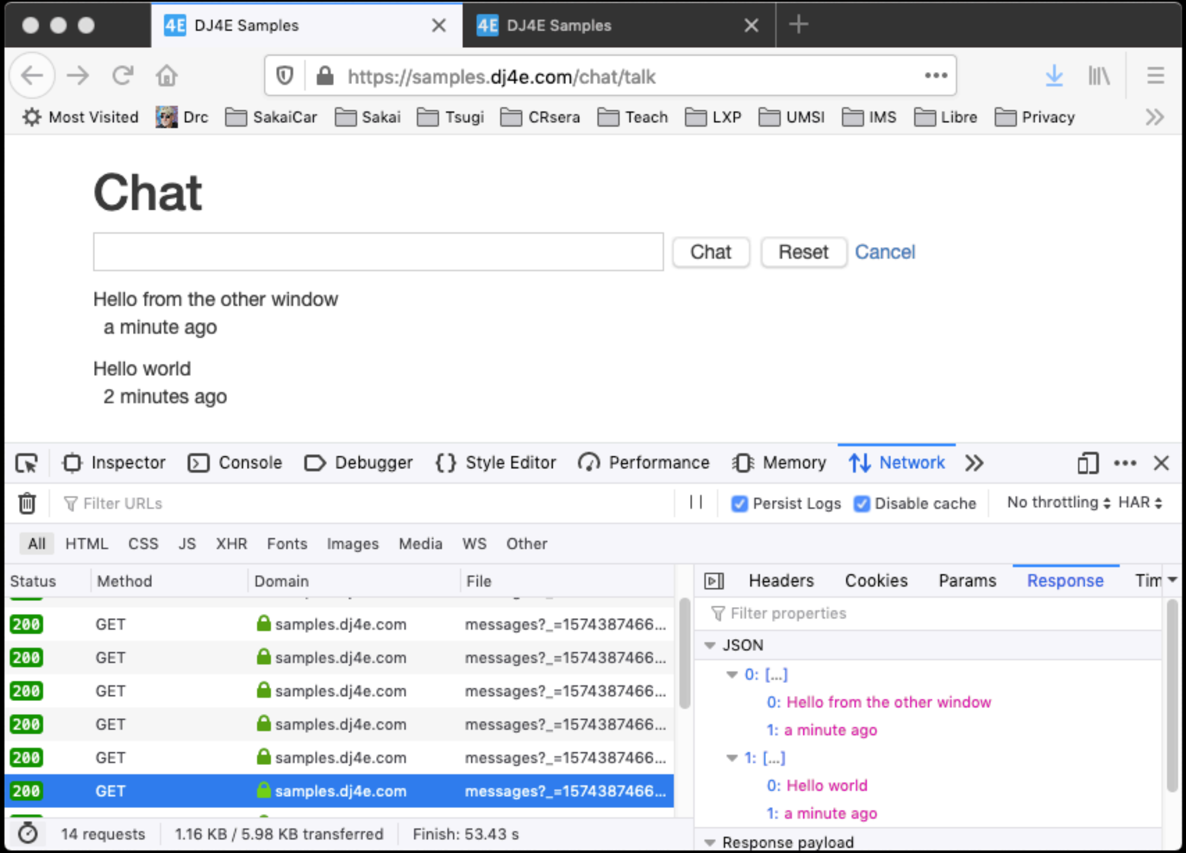


Chat

Hello from the other window
12 seconds ago

Hello world
37 seconds ago

https://samples.dj4e.com/chat/talk



Chat

Chat Reset Cancel

Hello from the other window
a minute ago

Hello world
2 minutes ago

Status	Method	Domain	File
200	GET	samples.dj4e.com	messages?_=1574387466...
200	GET	samples.dj4e.com	messages?_=1574387466...
200	GET	samples.dj4e.com	messages?_=1574387466...
200	GET	samples.dj4e.com	messages?_=1574387466...
200	GET	samples.dj4e.com	messages?_=1574387466...
200	GET	samples.dj4e.com	messages?_=1574387466...

14 requests 1.16 KB / 5.98 KB transferred Finish: 53.43 s

```
JSON
  0: [...]
    0: Hello from the other window
    1: a minute ago
  1: [...]
    0: Hello world
    1: a minute ago
Response payload
```

```
urls.py:

    path('talk', views.TalkMain.as_view(), name='talk'),
    path('messages', views.TalkMessages.as_view(), name='messages'),

   url(r'^static/(?P<path>.*)$', serve,
        {'document_root': os.path.join(BASE_DIR, 'static'), 'show_indexes': True},
        name='static'
  )
```

```
models.py:

class Message(models.Model) :
    text = models.TextField();
    owner = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)

    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

https://samples.dj4e.com/chat/talk

```
views.py:

class TalkMain(LoginRequiredMixin, View) :
    def get(self, request):
        return render(request, 'chat/talk.html')

    def post(self, request) :
        message = Message(text=request.POST['message'], owner=request.user)
        message.save()
        return redirect(reverse('chat:talk'))
```

https://samples.dj4e.com/chat/talk

templates/chat/talk.html (1 of 3):

```
{% extends 'base_bootstrap.html' %}
{% block content %}
<h1>Chat</h1>
<form method="post">
{% csrf_token %}
<input type="text" name="message" size="60"/>
<input type="submit" value="Chat"/>
<input type="submit" name="reset" value="Reset"/>
<a href="{% url 'chat:main' %}" target="_blank">Cancel</a>
</p>
</form>

<div id="chatcontent">
<img src="{% url 'chat:static' 'spinner.gif' %}" alt="Loading..."/>
</div>
```

https://samples.dj4e.com/chat/talk

templates/chat/talk.html (2 of 3):

```
<script type="text/javascript">
function updateMsg() {
    console.log('Requesting JSON');
    $.getJSON('{% url 'chat:messages' %}', function(rowz){
      console.log('JSON', rowz);
      $('#chatcontent').empty();
      for (var i = 0; i < rowz.length; i++) {
        arow = rowz[i];
        $('#chatcontent').append('<p>'+arow[0] +
            '<br/>&nbsp;&nbsp;'+arow[1]+"</p>\n");
      }
      setTimeout('updateMsg()', 4000);
  });
}
```

```
[
 ["Hello from the other window", "13 minutes ago"],
 ["Hello world", "14 minutes ago"]
]
```

https://samples.dj4e.com/chat/talk

templates/chat/talk.html (3 of 3):

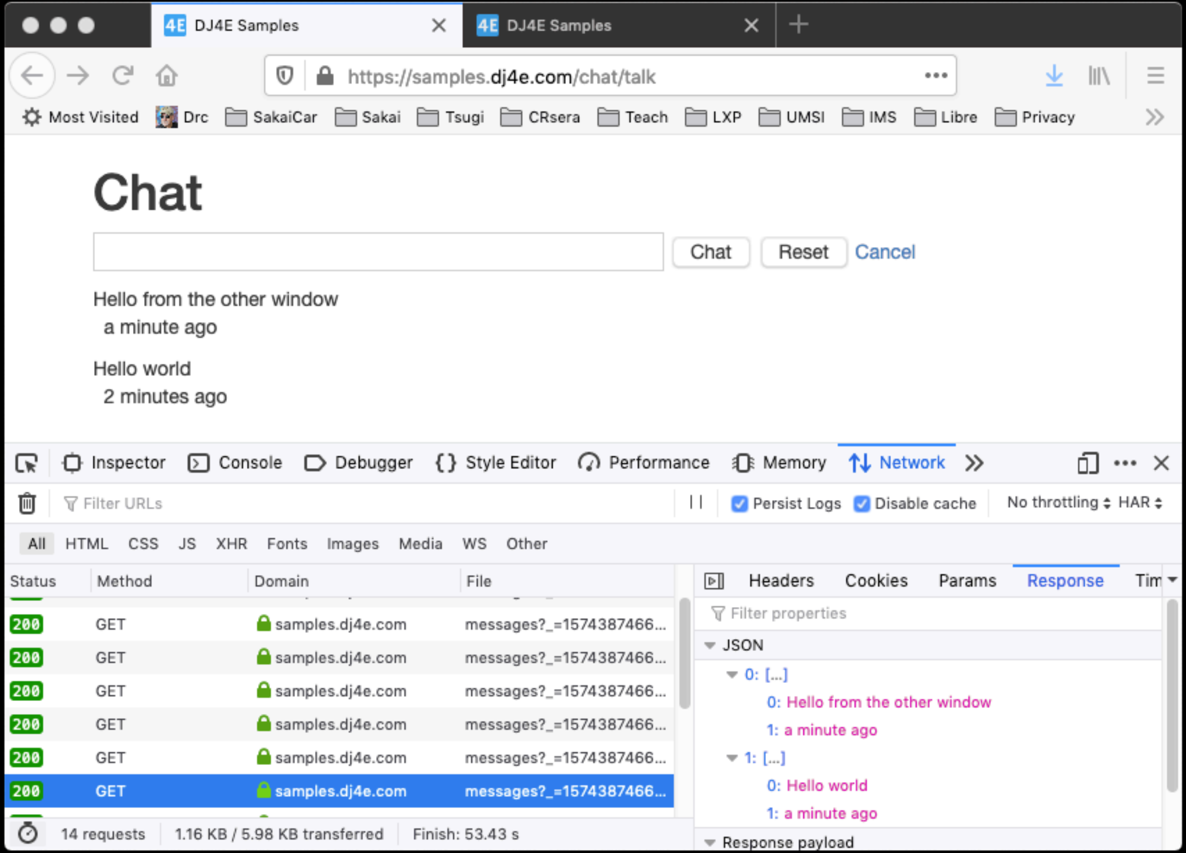
```
// Make sure JSON requests are not cached
$(document).ready(function() {
  $.ajaxSetup({ cache: false });
  updateMsg();
});
</script>
{% endblock %}
```

https://samples.dj4e.com/chat/messages

```
views.py:

class TalkMessages(LoginRequiredMixin, View) :
    def get(self, request):
        messages = Message.objects.all().order_by('-created_at')[:10]
        results = []
        for message in messages:
            result = [message.text, naturaltime(message.created_at)]
            results.append(result)
        return JsonResponse(results, safe=False)
```

```
[
 ["Hello from the other window", "13 minutes ago"],
 ["Hello world", "14 minutes ago"]
]
```

https://samples.dj4e.com/chat/talk



Chat

Hello from the other window
a minute ago

Hello world
2 minutes ago

Status	Method	Domain	File
200	GET	samples.dj4e.com	messages?_=1574387466...
200	GET	samples.dj4e.com	messages?_=1574387466...
200	GET	samples.dj4e.com	messages?_=1574387466...
200	GET	samples.dj4e.com	messages?_=1574387466...
200	GET	samples.dj4e.com	messages?_=1574387466...
200	GET	samples.dj4e.com	messages?_=1574387466...

14 requests 1.16 KB / 5.98 KB transferred Finish: 53.43 s

```
JSON
  0: [...]
    0: Hello from the other window
    1: a minute ago
  1: [...]
    0: Hello world
    1: a minute ago
Response payload
```

Summary

- JSON is very simple and powerful.
- It is well supported and performance in many languages.
- JavaScript / jQuery and Python/Django have excellent support.

Acknowledgements / Contributions

Initial Development: Charles Severance, University of Michigan School of Information

Insert new Contributors and Translators here including names and dates

Continue new Contributors and Translators here

Additional Source Information

- Image of Douglas Crockford is Copyright Charles R. Severance and licensed as CC-BY