FE101

Building the Front End for an Existing API

What is an API?

API stands for “Application Programming Interface”

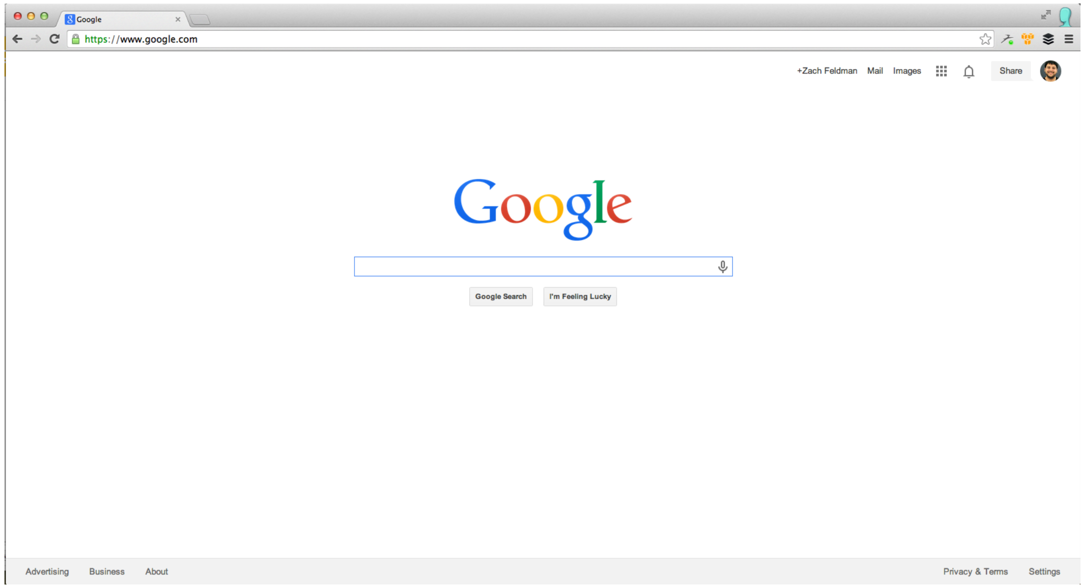
It is simple a publicly accessible way to program any black-box application, usually through a well-defined syntax

Web APIs using different *HTTP verbs* to access different functionality at different URLs

HTTP Request Types

GET

A GET request asks for information on a certain resource. It typically returns back an HTML page, or in Rails terms, a view

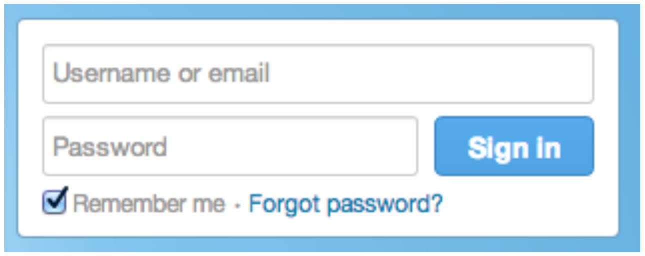


Issuing a GET request to Google.com returns an HTML file, or view, for their main home page.

HTTP Request Types

POST

A POST request contains data that the server is meant to process, typically to create a new version of the entity. POST requests typically do not serve back a view, but instead redirect to one



When I submit the log in form on the Twitter.com homepage, a POST request containing my Username or email and password is sent to the server for processing. If my information is correct, it redirects me to my profile page view.

HTTP Requests

PUT

A PUT request contains data that the server is meant to process, typically to update a set of existing data. PUT requests redirect to a view.

DELETE

A DELETE request is meant to delete the specified resource

RESTful Routing

REST stands for: RepresEntational State Transfer

Implementing a RESTful API allows your website to be more predictable and easy to use, as well as organized

Different HTTP verbs are used on the same URL (resource) by the user to represent the requested state of that resource

RESTful Routing

For example, a basic RESTful routing scheme for a **users** resource might look like this:

HTTP Verb	URL	Action
GET	/users	list an index of all users
POST	/users	create a new user
GET	/users/new	display a form to create a new user (signup form)
GET	/users/:id/edit	display a form to edit an existing user with the ID :id
GET	/users/:id	show the user with the id :id
PUT	/users/:id	update the user with id :id
DELETE	/users/:id	destroy the user with id :id

JSON - JavaScript Object Notation

“a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate.”
-json.org

```
{
  “animals”: [
    “zebra”,
    “ostrich”,
    “baboon”
  ]
}
```

JSON - JavaScript Object Notation

JSON is parsed into a JavaScript object by using the JSON.parse method and passing it the JSON as a string.

```
var parsed = JSON.parse('{"jelly":
"fish"}')
parsed.jelly
>>"fish"
```

Most JSON is returned as the result of an **AJAX call.**

AJAX Requests

AJAX stands for
Asynchronous **J**avaScript **A**nd **X**ML

The meaning: web applications can send or receive data from the server without requesting an entire page - instead, they can just request or send an arbitrary amount of data

Making AJAX Requests with jQuery

The easiest and most cross-browser compatible way to make an AJAX request is with jQuery.

```
$.ajax({
  type: “POST”,
  url: “/request”,
  data: {q: “131 Humboldt St”}
})
```

Making AJAX Requests with jQuery

In order to parse the data returned when the request is complete, use the .done() method and give it an anonymous function as an argument:

```
$.ajax({
    type: “POST”,
    url: “/request”,
    data: {q: “131 Humboldt St”}
}).done(
   function(d){
     alert(‘data returned is: ’ + d);
   }
)
```

Making AJAX Requests with jQuery

Try making this request from the console on the NYCDA home page.

```
$.ajax({
   type:"POST",
   url:"http://nycda.com"
}).done(
  function(d){
    console.log(d)
  }
)
```

Parsing Returned AJAX Data

- When you make an AJAX request, AJAX data may be returned as an object within a string

- To convert this to a valid JavaScript object, use the JSON object’s parse method:

```
a = JSON.parse(‘"ip":"192.76.2.114","country_code":"US"}’)
a.ip
>>”192.76.2.114”
```

Cross-domain considerations

- You cannot normally make an AJAX request across different domains

- To get around this, you can make a request to a URL local to your domain that makes a cross-domain request for you. For instance, a request to a Sinatra route which itself makes a request and returns the data.

JSONP

- JSONP or "JSON with padding" is a communication technique used in JavaScript programs running in web browsers to request data from a server in a different domain, something prohibited by typical web browsers because of the same-origin policy.

```
$.ajax({dataType: “jsonp”});
```

The Dribbble API

- Dribbble is a website where designers can showcase their work

- They also have a really nicely documented API

Let’s call their API!

```
$.ajax({
        url: "http://api.dribbble.com/shots/21603",
        type: "GET",
        success: function(data) {
            console.log(data);
        },
        error: function() {
            console.log("Sorry. Ajax request failed. :-( ");
        },
        dataType: 'jsonp'
    });
```

Exercise: Show off a shot

Call the Dribbble API for a single shot and display the following data:

- the image itself
- the shot title
- the player
- the number of likes
- the number of comments
- ...and anything else you think would look nice!