

# jQuery - AJAX

---

FRONT-END DEVELOPMENT KURS

# AJAX

---

- Asynchronous JavaScript and XML
- Tehnika za kreiranje brzih dinamičkih web stranica
- Kod klasičnih web stranica (bez AJAXa) potrebno je osvežiti čitavu web stranicu kako bi se promenio sadržaj
- AJAX dozvoljava da se web stranica osvežava asinhrono, razmenom male količine podataka sa serverom. Ovo omogućava da se osvežavaju delovi stranice

# AJAX

---

- Obrada zahteva na serveru može biti veoma vremenski zahtevna. Ovo može prouzrokovati da web aplikacija zakoči ili stane
- Sa AJAXom, JavaScript ne mora da čeka na odgovor od servera već može da
  - Izvršava druge skriptove dok čeka odgovor
  - Obradi odgovor tek onda kada je on spreman

# AJAX

---

- Nije nova tehnologija već je kombinacija postojećih tehnologija:
  - **XMLHttpRequest object** – asinhrona razmena podataka sa serverom
  - **JavaScript/DOM** – izmena strukture i sadržaja bez ponovnog učitavanja stranice
  - **CSS** – uređivanje izgleda stranice
  - **XML (danas JSON)** – format podataka koji se razmenjuju

# AJAX

## Browser

Registrovan je događaj:

- Kreiraj **XMLHttpRequest** objekat
- Pošalji HTTP zahtev

HTTP/HTTPS  
zahtev

## Server

- Procesiranje HTTP zahteva
- Kreiranje odgovora i slanje podataka nazad u browser

## Browser

**JavaScript/DOM:**

- Procesiranje dobijenih podataka
- Osvežavanje sadržaja stranice

HTTP/HTTPS odgovor  
XML/JSON/HTML/tekst

# AJAX Prednosti

---

- Podrжан od strane svih modernih browsera
- Web stranica je prijatnija za korišćenje
  - Ne gubi se sav operacioni sadržaj dok se stranica učitava
  - Instant odgovor – korisnik ne mora da čeka sledeću stranicu
  - Ne gubi se pozicija na koju je korisnik skrolovao
  - Može se definisati akcija i na mouse move i slične događaje
  - Navigacija na stranici je lepša u odnosu na standardno back/forward na browseru
  - Osećaj desktop aplikacije

# AJAX Prednosti

---

- Uvećane performanse i brzina odgovora
- Učitavaju se samo podaci koji su neophodni za osvežavanje strane tako da je saobraćaj između klijenta i servera manji
- Odvajanje prezentacije podataka od dobavljanja podataka
- Data driven (nasuprot klasičnim web aplikacijama koje su page-driven)

# AJAX Mane

---

- Usled dinamičkog učitavanja sadržaja otežano je:
  - registrovanje stanja u istoriji browsera (problem back/forward navigacije)
  - bookmarkovanje nekog konkretnog stanja
  - indeksiranje od strane pretraživača (dinamički proizveden sadržaj u opštem slučaju nije vidljiv crawlerima)
- Korisnici koji koriste browsere koji nemaju podršku ili imaju onesposobljen JavaScript neće biti u stanju da koriste funkcionalnost obezbeđenu AJAXom



# jQuery – Same-origin policy

---

- Same-origin policy - bitan koncept u sigurnosnom modelu web aplikacije
- On dozvoljava skriptama koje rade na stranicama poreklom sa istog sajta da pristupe međusobnom objektnom modelu dokumenata (DOM) bez posebnih ograničenja, ali sprečava pristup DOMu drugih sajtova.
- Same-origin policy se takodje primenjuje na XMLHttpRequest-ove osim u slučaju da server pruža Access-Control-Allow-Origin (CORS) zaglavlje.

# jQuery - AJAX

---

- jQuery pojednostavljuje AJAX funkcionalnosti
- Različiti brauzeri podržavaju različitu sintaksu AJAX poziva
- jQuery rešava ovaj problem, pa se AJAX poziv svodi na jednu liniju koda

# jQuery – load()

---

- Asinhrono učitavanje sadržaja sa servera i smeštanje tog sadržaja u selektovani element

```
$(selector).load(URL,data,callback);
```

- Opšti oblik:

```
$("#server_message").load("http://localhost:3000/api/some_url");
```

# jQuery – get()

---

- Asinhrono zahteva resurs sa servera

```
$.get(URL, data, successCallback, dataType);
```

- Opšti oblik:

```
$.get("http://localhost:3000/api/articles", function(data,status){  
    // manipulacija DOM stablom na  
    // osnovu pristiglih podataka (data)  
});
```

# jQuery – post()

---

- Asinhrono zahteva resurs sa servera

```
$.post(URL, data, successCallback, dataType);
```

- Opšti oblik:

```
$.post("demo_ajax_gethint.asp", {suggest: txt}, function(result){  
    $("span").html(result);  
});
```

# jQuery – Obrada događaja

---

- `beforeSend()`
  - funkcija koja će se pozvati pre slanja zahteva, npr. postavka custom header-a u zahtevu
  - metoda može da vrati `false` čime će se zahtev ukinuti
- `success()` ili `done()`
  - funkcija koja se poziva u slučaju da je zahtev serveru uspešno razrešen
- `error()` ili `fail()`
  - funkcija koja se poziva u slučaju da je zahtev serveru razrešen neuspehom
- `complete()`
- funkcija koja se poziva kada se zahtev izvrši (bez obzira na ishod), nakon izvršavanja `success()` i `error()`

# jQuery – Obrada događaja

---

```
$.post("http://localhost:8080/Primeri_AJAX/PostArticle",  
    ...  
)  
.error(function(data,status){ alert("Request failed!");})  
.success(function(data,status){alert("Request successful");});
```

- Više na: <http://api.jquery.com/category/ajax/>

# jQuery – AJAX Pozivi

---

- `load(URL, data, callback)`
  - učitavanje URLa u DOM kontejner, GET ili POST
- `getJSON(URL, data, successCallback)`
  - učitavanje JSON objekta pomoću GET metoda
- `getScript(URL, successCallback)`
  - učitavanje JavaScripta sa servera, skripta se automatski izvršava



# jQuery – AJAX Pozivi

---

- `get(URL, data, successCallback, dataType)`
  - Generički AJAX GET zahtev
- `post(URL, data, successCallback, dataType)`
  - Generički AJAX POST zahtev
- `ajax(options)`
  - AJAX zahtevi opšte svrhe (viši nivoi apstrakcije), GET ili POST

Podsetnik: Modul 1 - HTTP.pdf

Primer: 1. Pozivi

API u primerima: <https://httpbin.org/>

# URL - Podsetnik

---

http://www.domain.com:1234/path/to/resource?a=b&x=y

The diagram illustrates the components of the URL `http://www.domain.com:1234/path/to/resource?a=b&x=y`. Red horizontal brackets are placed under each component, with red vertical lines connecting them to labels below or above the URL. The labels are: **protocol** (under `http`), **host** (under `www.domain.com`), **port** (above `:1234`), **resource path** (under `/path/to/resource`), and **query** (above `?a=b&x=y`).

protocol

host

port

resource path

query

# URL - Podsetnik

---

- Prenos parametara se može vršiti na dva načina:
- Query parametri:
  - `http://www.domain.com/path/to/resource?p1=v1&p2=v2&p3=v3`
- URL parametri:
  - Šablon:
  - `http://www.domain.com/path/to/resource/{p1}/....`
  - Izgled:
  - `http://www.domain.com/path/to/resource/v1/....`
- Moguće je kombinovati ova dva pristupa

# APIs

---

- Korisnički interfejsi (User interfaces) omogućavaju ljudima da interaguju sa programima.
- APIs (Application Programming Interfaces) omogućavaju interakciju između programa (i skripta).
- API je skup protokola i rutina koje definiše načine na koje aplikacije mogu da zahtevaju usluge (servise) od drugih aplikacija.

# APIs

---

- NPR.:
  - DOM je API koji omogućava skriptama da pristupe i menjaju sadržaj web stranica dok su učitane u browseru.
  - jQuery je JavaScript fajl sa API-jem. Omogućava selektovanje elemenata, i upotrebu metoda za obradu tih elemenata.
  - Web sajtovi kao što su Facebook, Google, Twitter omogućavaju pristup svojim platformama, putem APIa, kako bi pristupili određenim funkcionalnostima koje poseduju ili podacima (npr. Like dugme, ugrađivanje Google mapa za prikaz mesta prodavnice).

# API Key

---

- Većina web APIa zahteva korišćenje API ključa (API Key)
- API Key je kod koji se šalje za API zahtevima
- Služi za identifikaciju:
  - Aplikacije koji vrši API pozive
  - Njenog developera ili korisnika
- Koristi se za praćenje i kontrolu korišćenja APIa

# FLICKER API

---

- Dokumentacija: <https://www.flickr.com/services/api/>

# FLICKR API

---

- Da bi koristili flickr api potrebno je napraviti API Key
- <https://www.flickr.com/services/api/keys/>
- API Key za primer:  
325b6733bb2c01b6ee5a9d73e9eb8cf9



# FLICKR API – REST Request format

---

- Za preuzimanje podatak putem APIa koristićemo REST, odnosno HTTP GET i POST akcije
- FLICKR REST endpoint URL:

<https://api.flickr.com/services/rest/>

- Primer korišćenja:

`https://api.flickr.com/services/rest/?method=flickr.test.echo  
&name=value`

# FLICKR API – Metode u primeru

---

- flickr.people.findByEmail
- flickr.people.findByUsername
- flickr.urls.lookupUser
- flickr.people.getInfo
- flickr.people.getPublicPhotos
- flickr.galleries.getList
- flickr.photos.comments.getList

# FLICKR API – Metode u primeru

---

- Primer poziva:

```
https://api.flickr.com/services/rest/?method=flickr.people.findByUsername&api_key=325b6733bb2c01b6ee5a9d73e9eb8cf9&username=george&format=json&jsoncallback=
```

- Primer odgovora:

```
jsonFlickrApi({"user":{"id":"34427469121@N01","nsid":"34427469121@N01","username":{"_content":"George"}},"stat":"ok"})
```

# URL – Podsetnik - FLICKR

---

`https://api.flickr.com/services/rest/?method=flickr.people.findByName&api_key=...&username=george&format=json&jsoncallback=`

- protocol
- host
- resource path
- query

# URL - Podsetnik

---

- Ono što se prenosi u query(upit) sekciji se nazivaju parametri(argumenti)
- Dati su u formatu ime=vrednost:
  - method=flickr.people.findByUsername
  - format=json
  - extras=url\_s,url\_o
  - api\_key=...
- Spajaju se pomocu & simbola: ime1=vrednost1&ime2=vrednost2&...

# URL - Podsetnik

---

`http://api.openweathermap.org/data/2.5/weather?q=London,uk&appid=b1b15e88fa797225412429c1c50c122a`

- Argumenti:
  - q=London,uk
  - appid=...

# FLICKR API - Endpoint

---

<https://api.flickr.com/services/rest/?method=flickr.test.echo&name=value>

Iz navedenog vidimo da url ka flickr api-ju:

- <https://api.flickr.com/services/rest/>
- A za poziv odredjene metode koristimo method argument, i ostale koji su potrebni navedenoj metodi.
- Pošto nam je potreban odgovor u JSON formatu, i pošto flickr vraća podatke u JSONP formatu, za potrebe jQuery-ja, dodajemo dodatne argumente:
  - format=json
  - jsoncallback=?

# FLICKR API – findByUsername

**flickr.people.findByUsername**

Naziv metode

Return a user's NSID, given their username.

## Authentication

This method does not require authentication.

## Arguments

### Argumenti

**api\_key** (Required)

Your API application key. [See here](#) for more details.

**username** (Required)

The username of the user to lookup.

Argumenti koje je obavezno proslediti pri pozivu metode su obeleženi sa (Required).

(Optional) - ne moramo proslediti argument



# FLICKR API - findByUsername

---

Na url:

- <https://api.flickr.com/services/rest/>?

- Dodajemo parametre:

- method=flickr.people.findByUsername
- api\_key=325b6733bb2c01b6ee5a9d73e9eb8cf9
- username=george
- format=json
- jsoncallback=?

`https://api.flickr.com/services/rest/?method=flickr.people.findByUsername&api_key=325b6733bb2c01b6ee5a9d73e9eb8cf9&username=george&format=json&jsoncallback=?`

# FLICKR API - findByUsername

---

- jQuery AJAX, primer poziva:
  - `$.get(URL, callback, "json");`
  - `$.getJSON(URL, callback);`
- URL predstavlja URL sa prethodnog slajda.

# FLICKR API - findByUsername

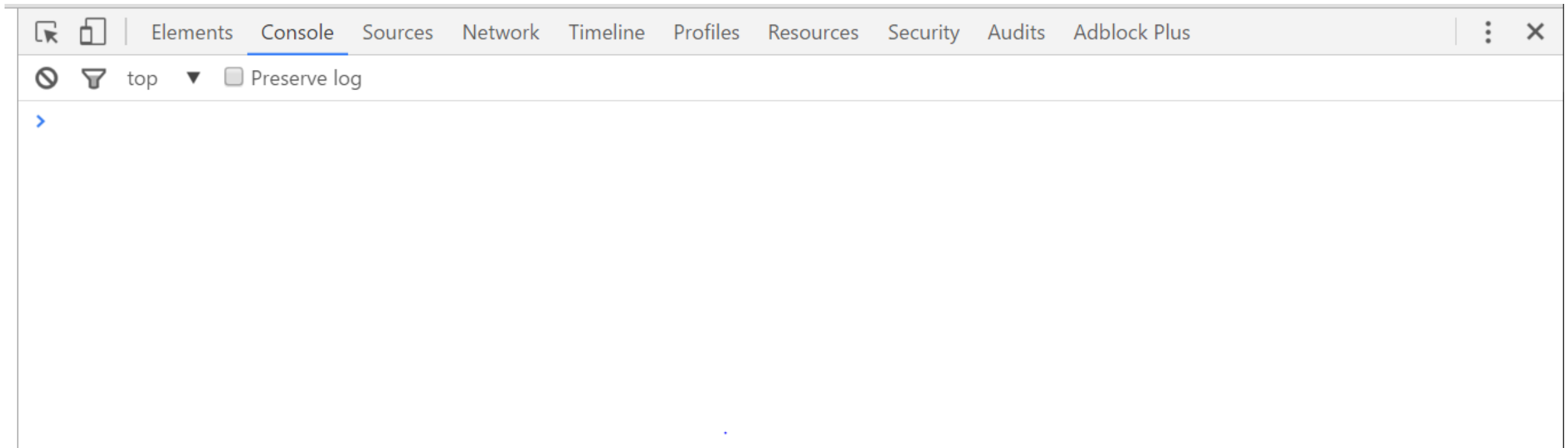
---

- Odgovor:
  - `jsonFlickrApi({"user":{"id":"34427469121@N01","nsid":"34427469121@N01","username":{"_content":"George"}},"stat":"ok"})`
- Za proveru sadržaja odgovor u obliku JavaScript objekta moguće je iskoristi developer tools browsera.
- Više na linku:
- <https://developer.chrome.com/devtools>

# FLICKR API – findByUsername - Odgovor

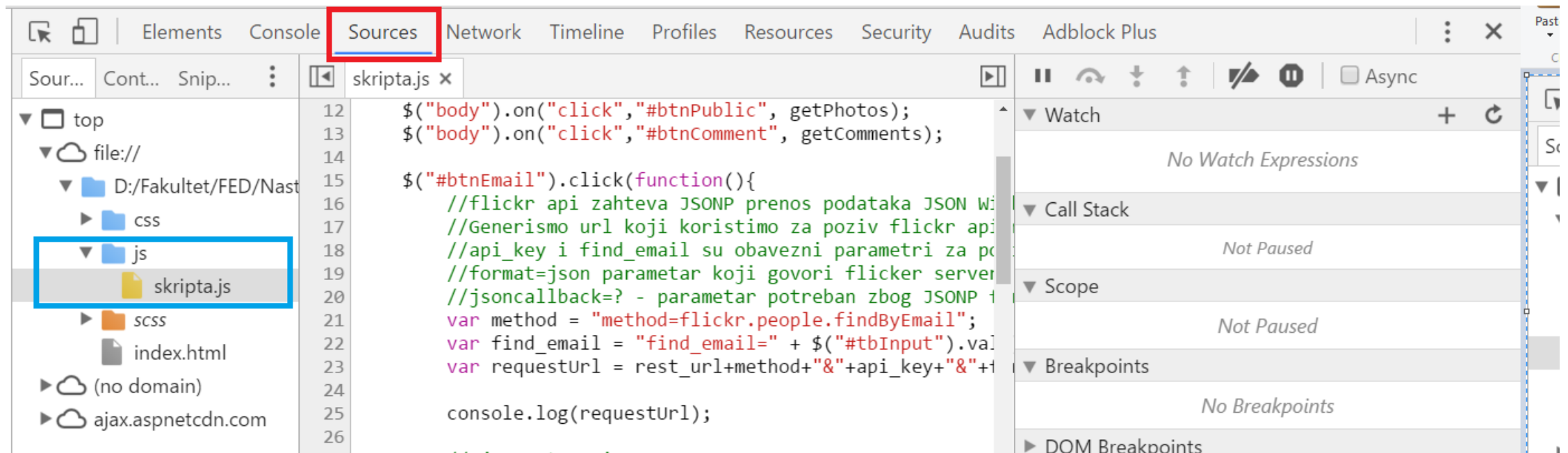
---

- Sa F12 otvaramo developer tools u Chromu:



# FLICKR API – findByUsername - Odgovor

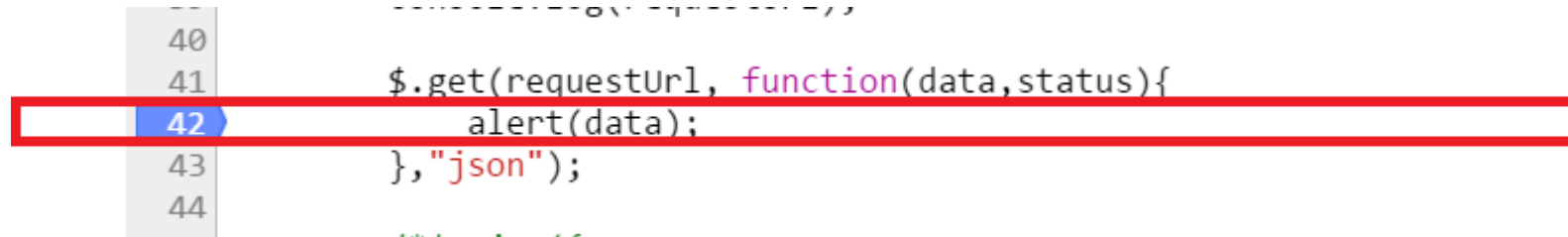
- Potrebno je da kliknemo na tab **Sources**, i da **pronadjemo nas .js fajl**.



# FLICKR API – findByUsername - Odgovor

---

- Sad možemo da pronađemo mesto naše callback funkcije, i da postavimo **breakpoint**.
- **Breakpoint** postavljamo tako što kliknemo na broj linije u kome se naš kod nalazi.



A screenshot of a code editor with a light gray background. On the left, line numbers 40, 41, 42, 43, and 44 are listed. Line 42 is highlighted with a blue background and a white arrow pointing to the right. A red horizontal bar is drawn across the code on line 42, indicating a breakpoint. The code on line 42 is `alert(data);`. The code on line 41 is `$.get(requestUrl, function(data,status){` and on line 43 is `}, "json");`.

```
40  
41 $.get(requestUrl, function(data,status){  
42     alert(data);  
43 }, "json");  
44
```

- Time postićemo da naš kod stane sa izvršavanjem kad dođe do linije u kojoj se breakpoint nalazi.

# FLICKR API – findByUsername - Odgovor

---

- Da bi naš kod došao do linije u kojoj se breakpoint nalazi, moramo pozvati izvršavanje odgovarajuće funkcije.
- Npr. Breakpoint se nalazi u callback funkciji \$.get zahteva za pronalaženje korisnika po njegovom korisničkom imenu.
- Navedeni \$.get zahtev se pokreće klikom na dugme: Pronadji UserName.
- Što znači da treba da kliknemo na dugme Pronadji UserName.

# FLICKR API – findByUsername - Odgovor

- Breakpoint:

The screenshot displays a web browser's developer tools interface. The left pane shows the 'Paused in debugger' state with a call stack. The middle pane shows the source code of 'skripta.js' with a breakpoint set at line 42. The right pane shows the 'Watch' and 'Scope' sections.

**Call Stack:**

- (anonymous function) skripta.js:42
- fire jquery-1.9.0.js:1017
- self.fireWith jquery-1.9.0.js:1127
- done jquery-1.9.0.js:8021
- script.onload.script.onreadystatechange jquery-1.9.0.js:8278

**Scope:**

- Local
  - data: Object
  - status: "success"
  - this: Object
- Closure (undefined)
- Global window

**Breakpoints:**




- skripta.js:42 alert(data);

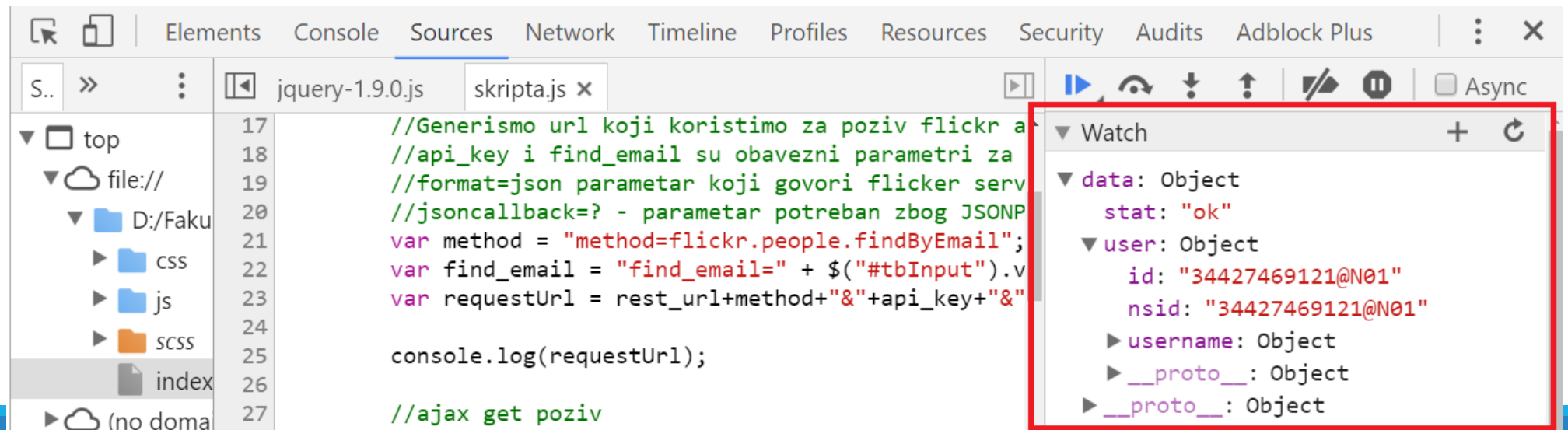
**Console:**

```
console.log(requestUrl);
//ajax get poziv
//requestUrl - url na koji saljemo zahtev https://api.flickr.
//postaviKorisnika - callback funkcija koja vrsi obradu odgov
//dataType - "json" parametar koji govori o formatu odgovora,
$.get(requestUrl, postaviKorisnika, "json");
});
$("#btnUsername").click(function(){
  var method = "method=flickr.people.findByUsername";
  var username = "username=" + $("#tbInput").val();
  var requestUrl = rest_url+method+"&"+api_key+"&"+username+"&";
  console.log(requestUrl);
  $.get(requestUrl, function(data,status){
    alert(data);
  }, "json");
});
/*$.ajax({
  method: "get",
  url: requestUrl,
  dataType: "jsonp",
  success: postaviKorisnika
});*/
});
```



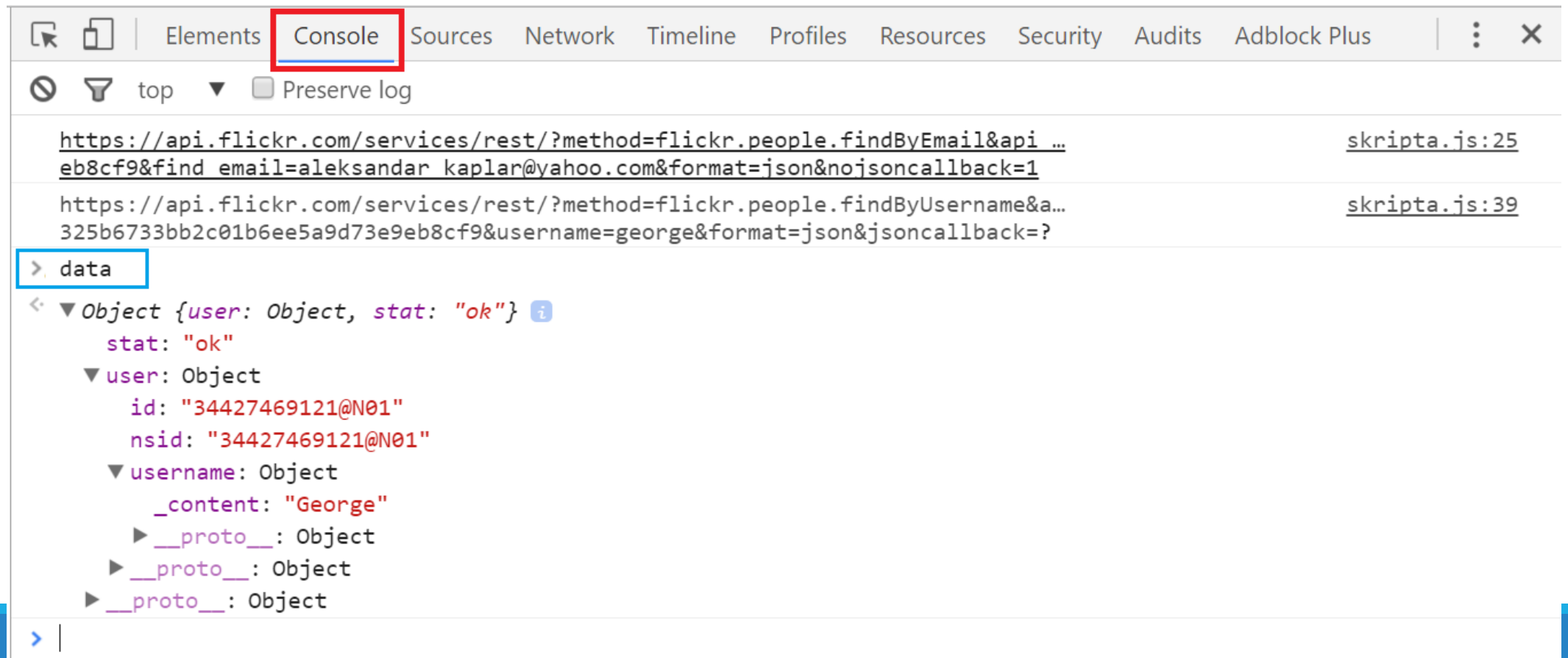
# FLICKR API – findByUsername - Odgovor

- Da bi nastavili izbršavanje koda potrebno je kliknuti na dugme: , a da bi prešli na izvršavanje sledeće linije koda: 
- Da bi pratili promenu naše promenljive možemo je dodati u Watch, klikom na dugme: 



# FLICKR API – findByUsername - Odgovor

- Ili možemo preći na tab **Console**, i ukucati **ime promenljive**:



```
https://api.flickr.com/services/rest/?method=flickr.people.findByEmail&api ... skripta.js:25
eb8cf9&find_email=aleksandar_kaplar@yahoo.com&format=json&nojsoncallback=1

https://api.flickr.com/services/rest/?method=flickr.people.findByUsername&a... skripta.js:39
325b6733bb2c01b6ee5a9d73e9eb8cf9&username=george&format=json&jsoncallback=?

> data
< ▼ Object {user: Object, stat: "ok"} ⓘ
  stat: "ok"
  user: Object
    id: "34427469121@N01"
    nsid: "34427469121@N01"
    username: Object
      _content: "George"
      __proto__: Object
    __proto__: Object
  __proto__: Object
```

# OPENWEATHER API

---

- <http://openweathermap.org/api>
- OpenWeather API kao podrazumevan format odgovora koristi JSON format.
- Što znači da nisu potrebni atributi format=json i jsoncallback=?, kao za flicker api.
- Pogledati rešenja zadatka, T8.