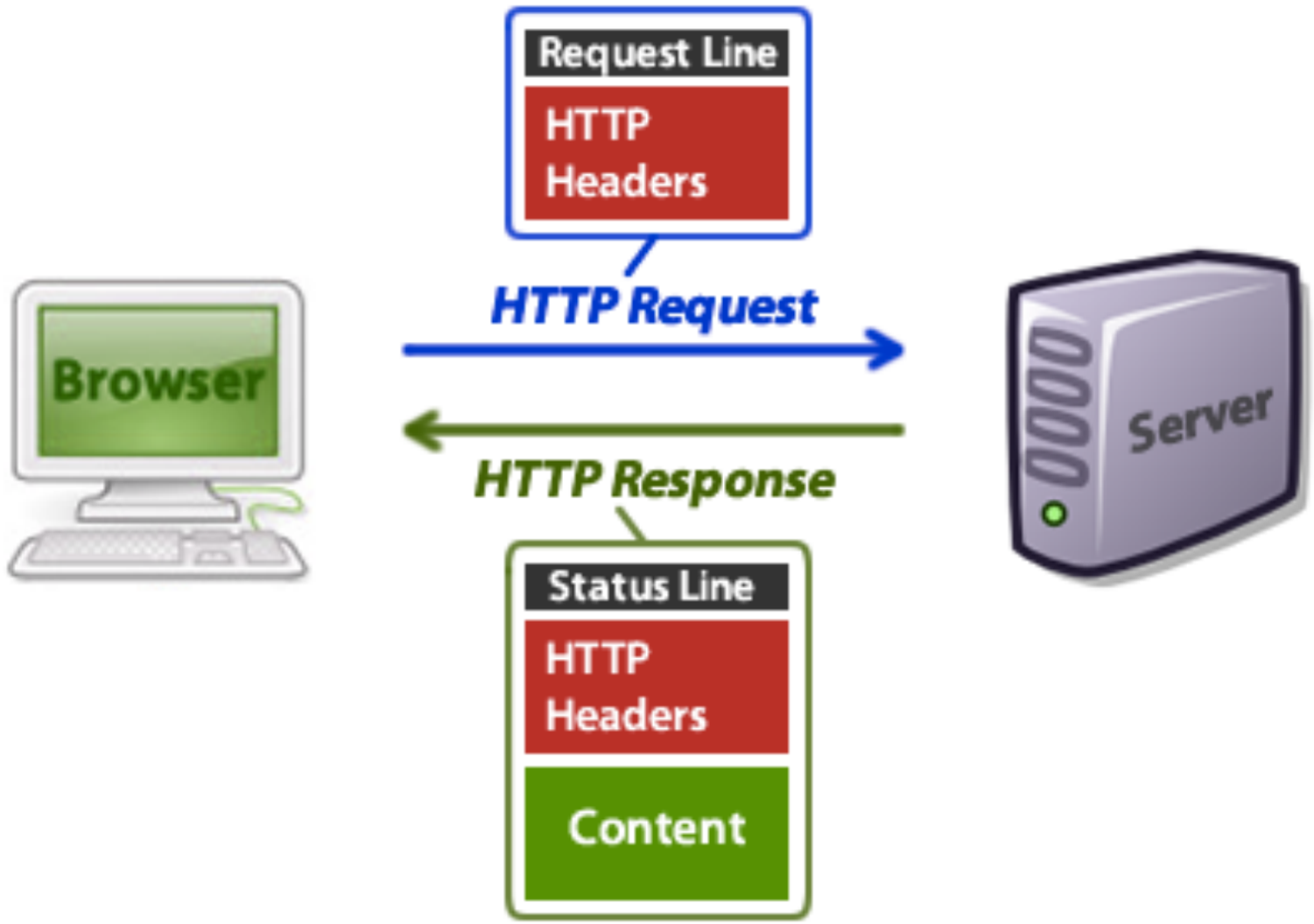
SWIFTACADEMY

НИЕ ВЯРВАМЕ ВЪВ ВАШЕТО БЪДЕЩЕ

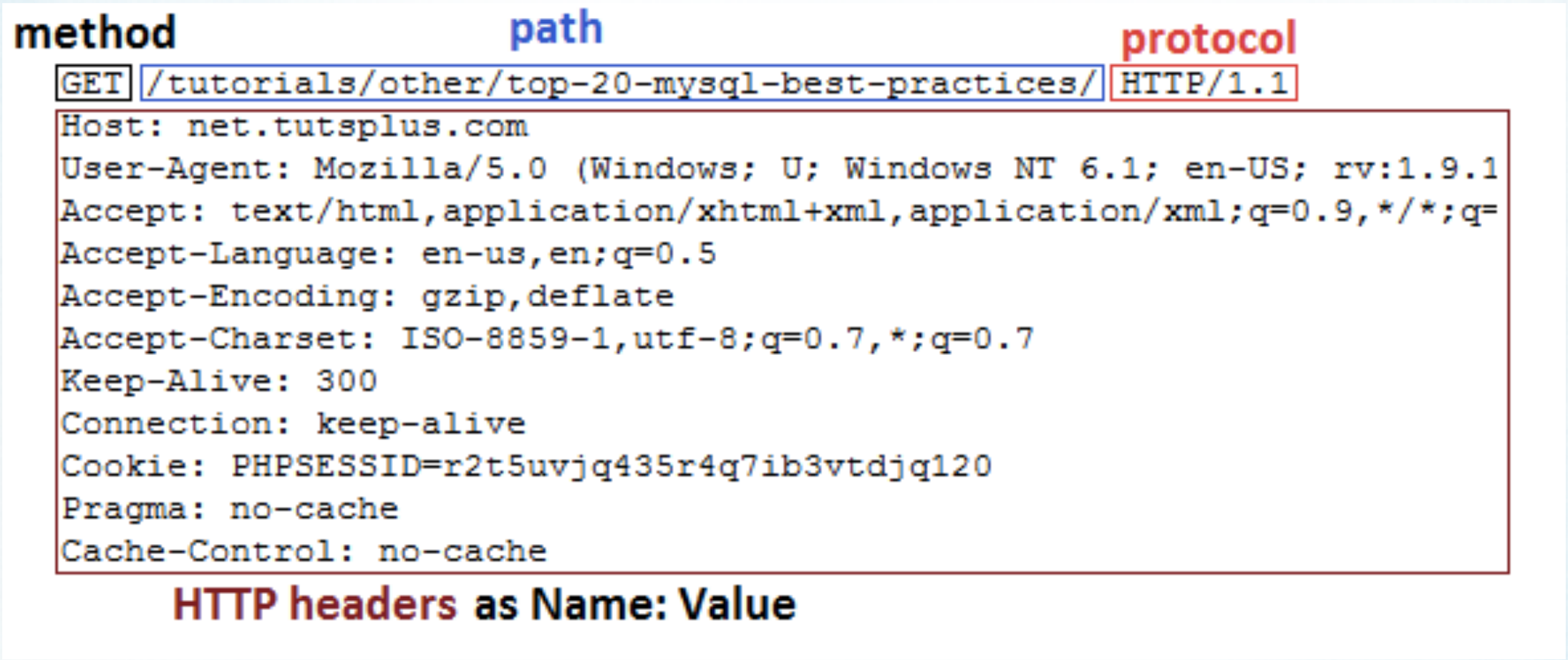
HTTP

requests & responses


Request Line
HTTP Headers
HTTP Request
Browser
Server
HTTP Response
Status Line
HTTP Headers
Content

- Hyper Text Transfer Protocol
- Това е протокол (набор от правила) за това как клиента и сървъра обменят информация
- Клиента (браузъра) прави *request* към сървъра, като му изпраща съобщение (*message*), което съдържа:
  - request line
  - http headers
  - body (само при POST заявки)

Request line-а се състои от **име на метод**, **път** и **протокол** (име и версия):



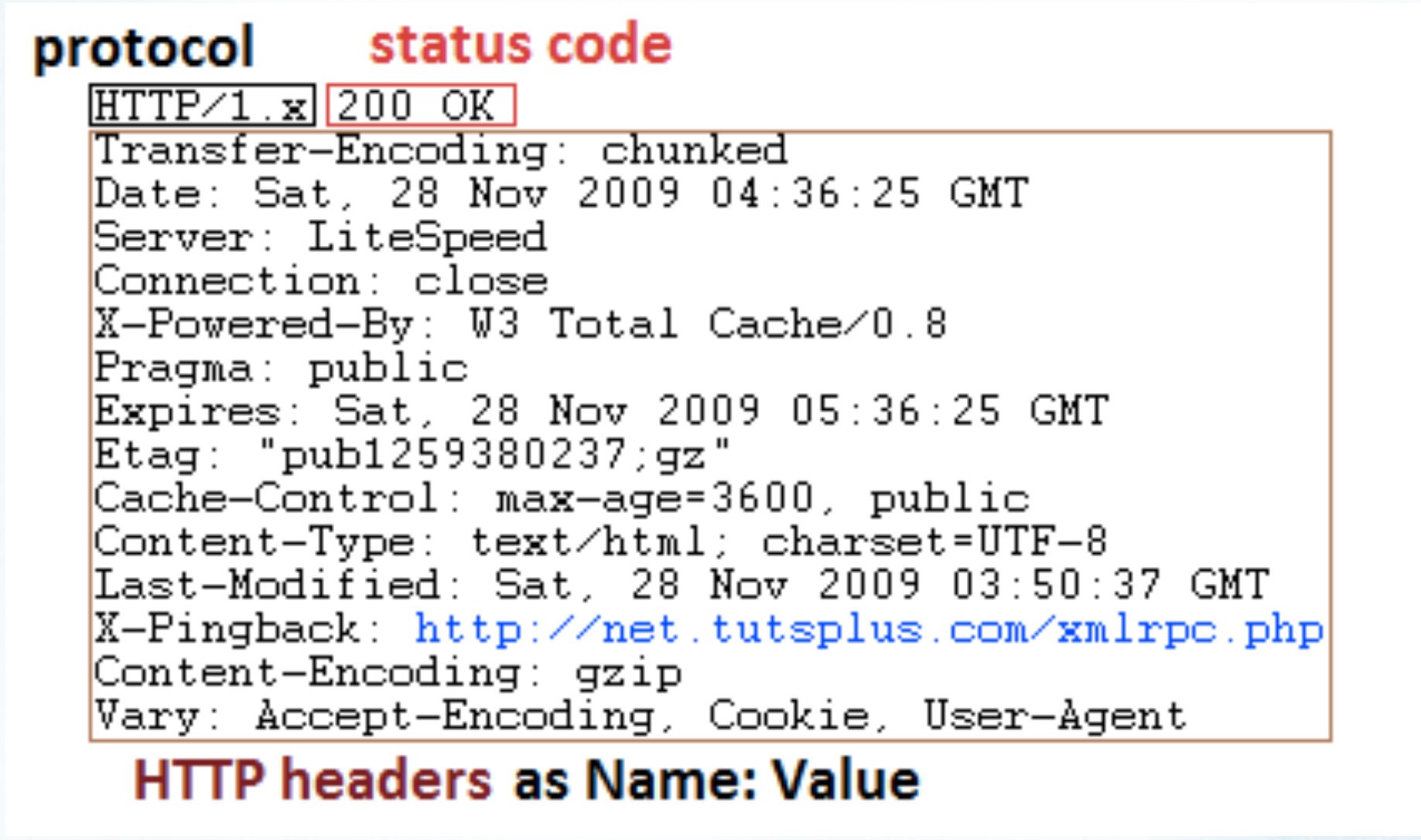
- Когато *request*-а стигне до сървъра, сървъра го “прочита” и намира документа, който е поискан в *request*-а
- За да изпрати документа към килента, сървъра съставя т.нар *response* message (отговор), който се състои от:
  - status line
  - http headers
  - content (самия документ, например например съдържанието на index.html)

```
HTTP/1.x 200 OK
Transfer-Encoding: chunked
Date: Sat, 28 Nov 2009 04:36:25 GMT
Server: LiteSpeed
Connection: close
X-Powered-By: W3 Total Cache/0.8
Pragma: public
Expires: Sat, 28 Nov 2009 05:36:25 GMT
Etag: "pub1259380237;gz"
Cache-Control: max-age=3600, public
Content-Type: text/html; charset=UTF-8
Last-Modified: Sat, 28 Nov 2009 03:50:37 GMT
X-Pingback: http://net.tutsplus.com/xmlrpc.php
Content-Encoding: gzip
Vary: Accept-Encoding, Cookie, User-Agent

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8
<title>Top 20+ MySQL Best Practices - Nettuts+</title>
<!-- ... rest of the html ... -->
```

Винаги първият ред от отговор-а съдържа т.нар. *status line*, който от своя страна се състои от:

- протокол (име и версия)
- status код



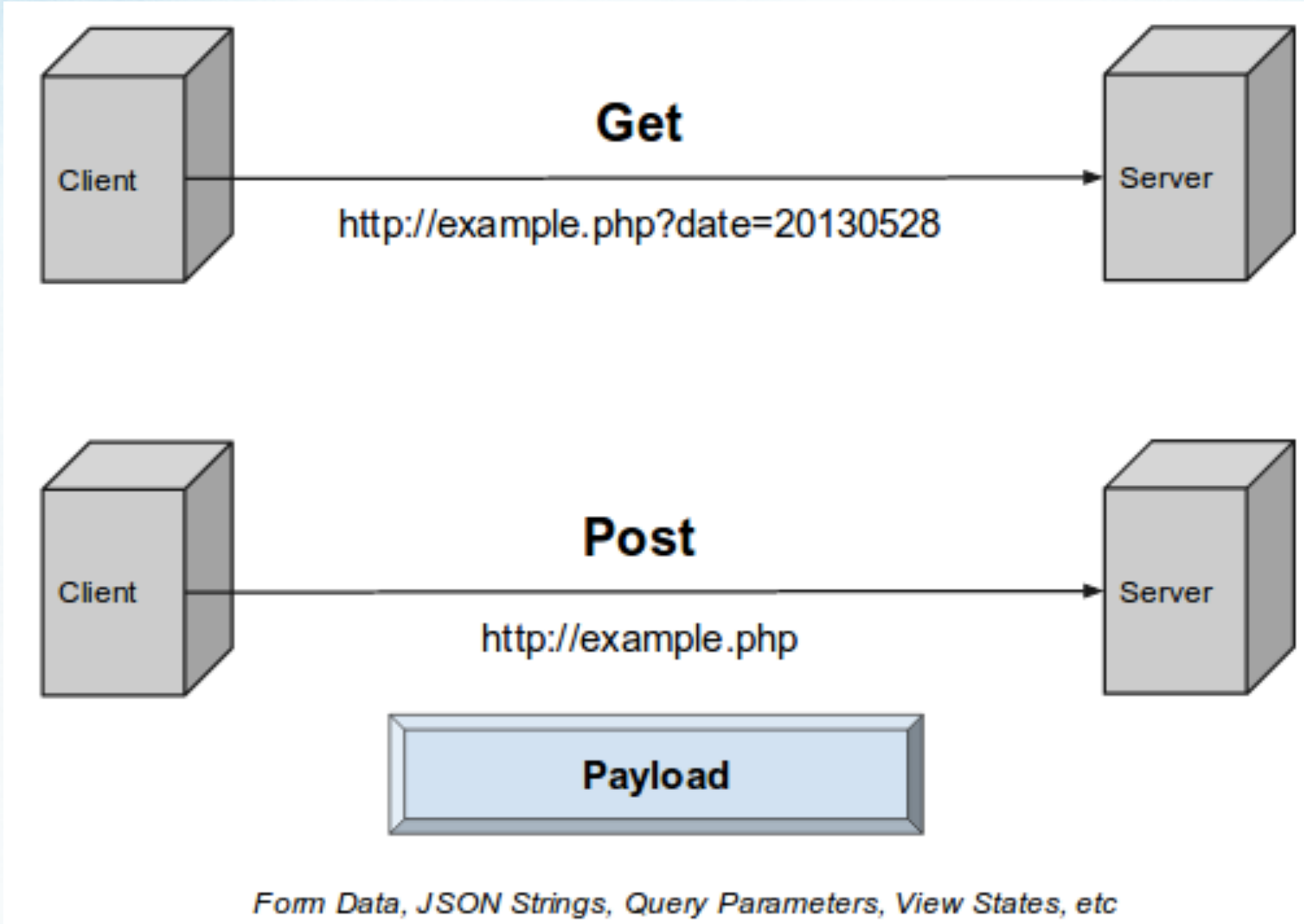
- От върнатият *status code*, клиента (браузърът) може да разбере дали *request*-а (заявката, която е направил) е бил успешен или не
  - 200 - OK
  - 404 - Not found
  - 403 - Forbidden
  - http://code.tutsplus.com/tutorials/http-headers-for-dummies--net-8039

POST vs GET

SWIFTACADEMY

НИЕ ВЯРВАМЕ ВЪВ ВАШЕТО БЪДЕЩЕ

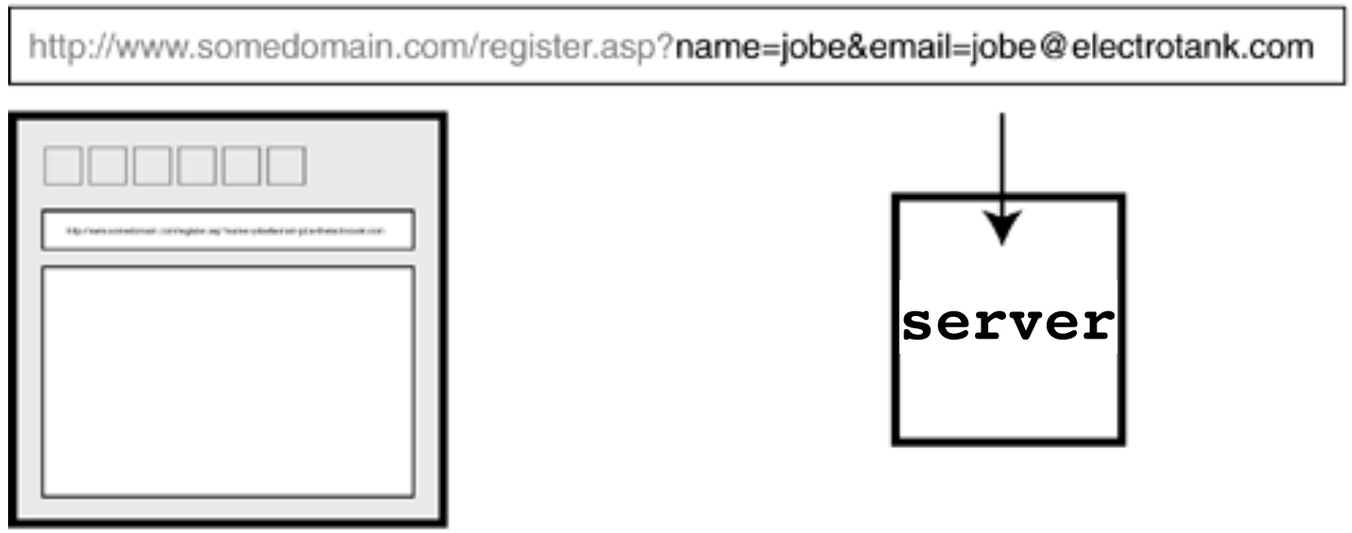
Front-End Development



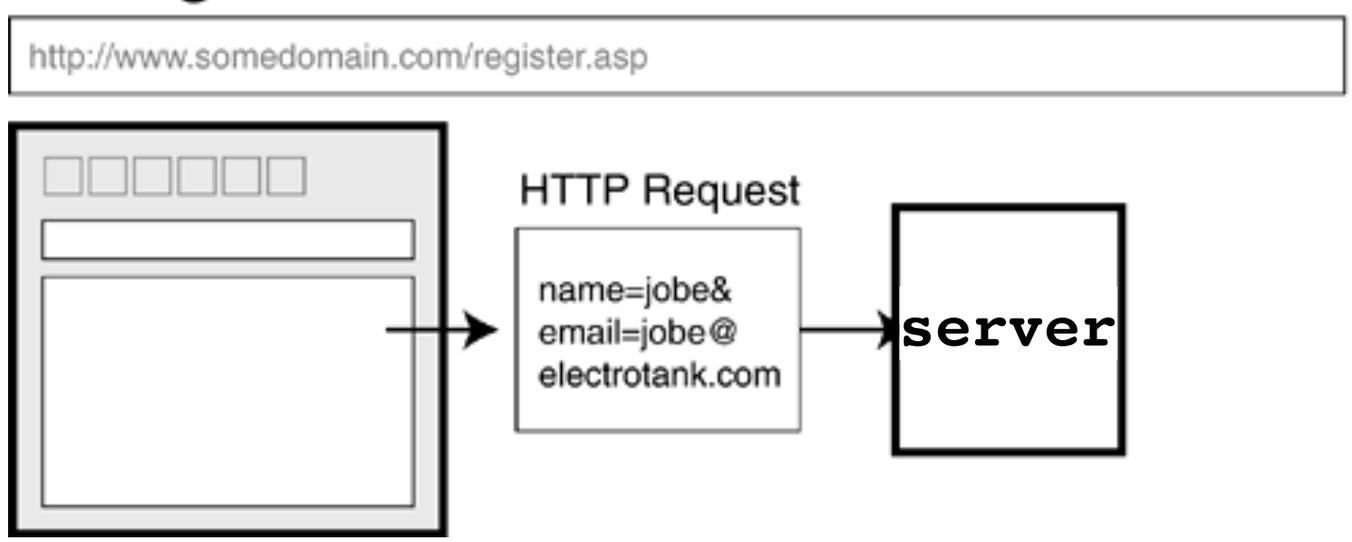
Form Data, JSON Strings, Query Parameters, View States, etc

- GET: заявка от клиента към сървъра за получаване на даден ресурс
  - request parameters in the request line
  - body: no
  - secure: NO
- POST: подаване на набор от параметри от клиента към сървъра с цел модифициране на ресурс
  - request parameters: in the body of the request
  - secure: YES
- http://www.w3schools.com/tags/ref_httpmethods.asp

Using GET



Using POST



WEB FORMS

SWIFTACADEMY

НИЕ ВЯРВАМЕ ВЪВ ВАШЕТО БЪДЕЩЕ

Front-End Development

- Използваме web форми (формуляри), когато е необходимо потребителят да въведе/предостави определена информация в нашият уебсайт

- За да добавим web-форма към нашият html, използваме таг-а *form*:

<form method="post"></form>

- form е отделен композитен елемент, който съдържа набор от елементи, наречени *полета*

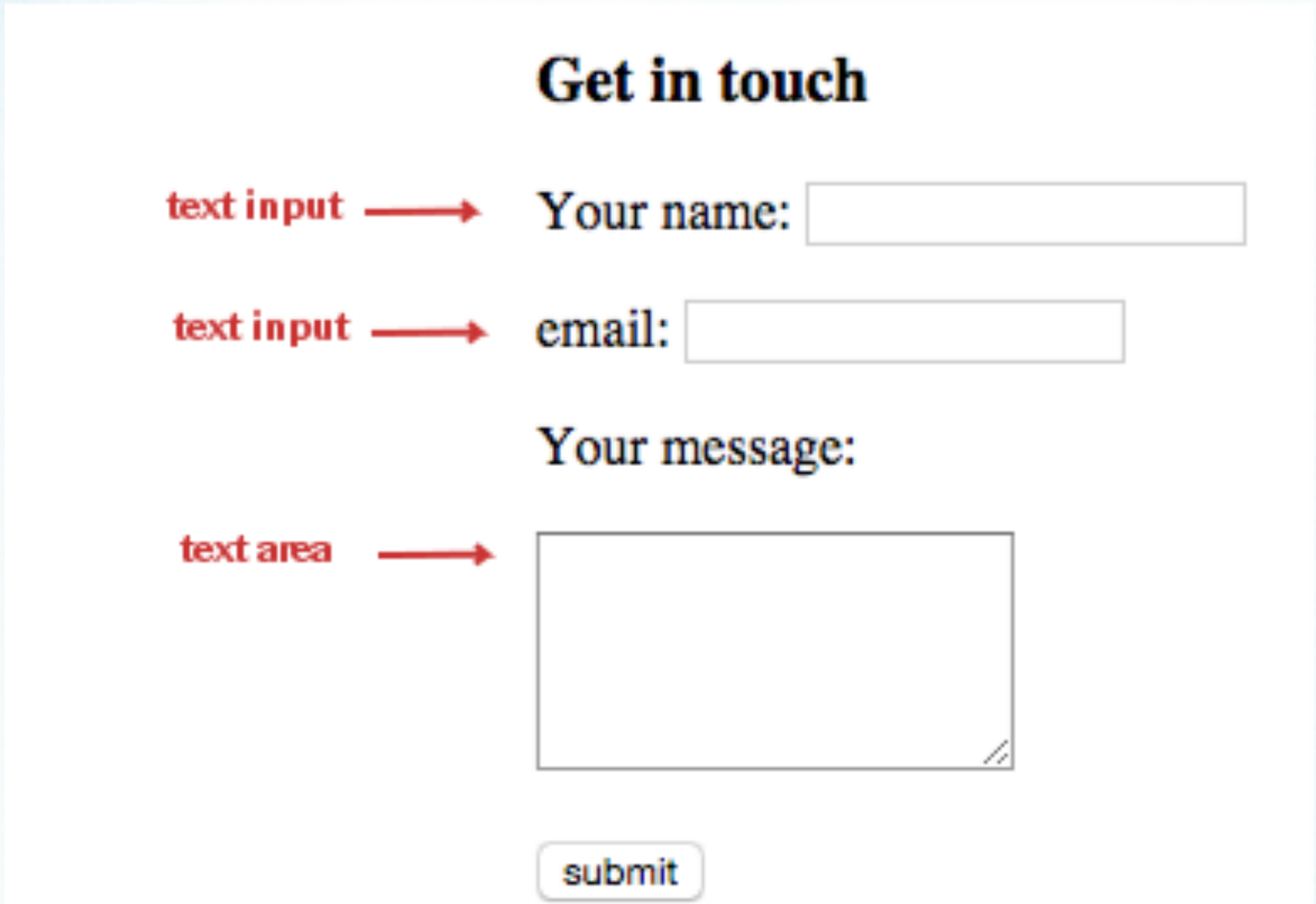
- Елементите на една web форма, могат да бъдат текстови полета (такива в които се въвежда текст), падащи менюта, радио-бутони (тези кръгличките), checkboxes, бутони и скрити полета

- *form* елементът има няколко специфични атрибута, които определят къде ще отиде информацията, която се въвежда от потребителя и как:
  - action=“api/send-feedback.php”
  - method=“post”
- Когато някой *събмитне* (изпрати) формата, се създава POST или GET заявка в зависимост от избрания *метод*. След това тази заявка се изпраща до указаният адрес в *action* атрибута
- Съответният уеб сървър трябва да знае как да “обработи” заявката и да върне съответното съдържание, отговарящо на резултата от обработката

Полета за въвеждане на текст

- <input type="text">
- ползваме ги, когато искаме юзерът да въведе имена, телефон, имейл и т.н. (в HTML 5 има отделни типове *input* за email, tel, url и др.)
- HTML5: <input type="email">
- input елемент-а няма затварящ таг!

- Полета за пароли
  - <input type="password">
  - съдържанието на това поле не се визуализира (вместо това излизат точки или звездички)
- Textarea
  - Използваме това поле за въвеждане на по-голям обем текст. Например при съобщение във форма за контакт.
  - <textarea>A pre-filled text here</textarea>
  - има атрибути *cols* и *rows*
  - има затварящ таг!


Get in touch
text input
Your name:
text input
email:
Your message:
text area
submit

Бутони

- Бутони за събмитване
  - <button type="submit"></button>
  - <input type="submit">
    - button има затварящ таг, докато input няма!
- За ресет-ване на форма
  - <input type="reset">
  - изтрива всичко попълнено във формата

- Бутони, които не събмитват уеб формата
  - <input type="button">
  - <button></button>
- Две основни разлики между *input* и *button*:
  - *input* може да се ползва само в контекста на *form*, а *button* - навсякъде
  - *input* не може да има "богато" съдържание, защото няма затварящ таг, докато *button* може

Списъци


drop down
(select box)
Select a country:
Bulgaria
Spoken languages:
checkboxes
English
Bulgarian
French
Do you like my page?
radio
Yes
No

- Списък от опции без ограничение на избора
  - <input type="checkbox">
  - виуализира се като квадратче и служи за отбелязване на избор от рода на:

```
[x] Приемам условията за ползване
```

- ако искаме да предоставим избор от повече опции, можем да групираме няколко бутона, чрез атрибута *name*

- Списък от опции с възможност за за максимум един избор
  - <input type="radio">
  - виуализира се като кръгче и има смисъл от него само ако има поне още един такъв, който да е в същата гупа от радио бутони
  - за да групираме няколко радио бутона, трябва да използваме атрибута *name*
  - Ако искаме да използваме само 1 радио бутон, значи ни трябва *checkbox*, а не *radio*

Падащи менюта

```
<select>
 <option>Option 1</option>
 <option>Option 2</option>
 ...
</select>
```

- представлява композиция от един външен **<select>** елемент и неограничен брой вътрешни **<option>** елементи
- нарича се още: *select box* или *dropdown menu*
- може да бъде със *single* или *multiple* choice
- ползва се ако имаме избор от прекалено много опции (повече от 5)
- имаме възможност за групиране на опциите с елемента **<optgroup>**
- http://www.w3schools.com/tags/tag_optgroup.asp

name и value атрибутите

- всяка web форма служи за изготвянето на списък от параметри, които да се пратят с POST или GET заявката към сървъра

- тези параметри се подават във формат **ключ=стойност** (виж слайдовете за *HTTP requests*)

- всеки един от тези параметри отговаря на дадено инпут поле от уеб формата, която е изпратила заявката

- именно по тази причина, инпут полетата имат *name* и *value* атрибути

- name атрибутът
  - Чрез този атрибут задаваме името на параметъра, който ще се изпрати при събмит на форма-та
  - Той трябва да е уникален, освен ако не се използва за група от еднотипни полета като *checkboxes* или *radio buttons*
- value атрибутът
  - Чрез този атрибут задаваме стойността на параметъра, който ще се изпрати при събмит на форма-та
  - Използва се при *hidden* полетата или когато изкаме да зададем предварително попълнена стойност на някое поле (Например: за дати - днешна дата, при редакция на вече попълнени данни и т.н.)

hidden полета

- Използват се за изпращане на данни, които не се въвеждат директно от потребителя
- <input type="hidden" name="key" value="123426536">
- имат смисъл, само ако са им зададени *name* и *value*
- обикновенно се използват от *javascript* за динамично генерирани стойности (*value*)

Други полета

- Файл ъплоад:

```
<input type="file">
```

- HTML5 нови типове полета:
  - <input type="email">
  - <input type="url">
  - <input type="tel">
  - <input type="color">
  - <input type="date">
  - <input type="range">

Други специални инпут атрибути

- **placeholder**="Enter some text here"
  задава упътващ текст на полето
- **readonly**
  не разрешава промяна на стойността на полето
- **disabled**
  disabled полетата не се изпращат като параметри
- **novalidate**
  отказва HTML5 валидацята
- **selected**
  за предварително избрана опция (*option*) на *select box*
- **checked** - за чекбокс и радио

Въпроси?

Примери

http://zenlabs.pro/courses/lessons/lesson8/examples.zip

Домашно

https://github.com/zzeni/swift-academy-homeworks/blob/fe-03/tasks/L08