



НИЕ ВЯРВАМЕ ВЪВ ВАШЕТО БЪДЕЩЕ

Преговор

- Какви са основните характеристики на `viewport` таг-а?
 - `width=device-width`
 - `initial-scale=1`
- Как можем да зададем стилове за екрани с ширина между 600 и 800px?
 - `@media (min-width: 600px) and (max-width: 800px) { ... }`
+ meta viewport tag
 - `<link`
 `rel="stylesheet"`
 `media="(min-width: 600px) and (max-width: 800px)"`
 `href="...">`
+ meta viewport tag

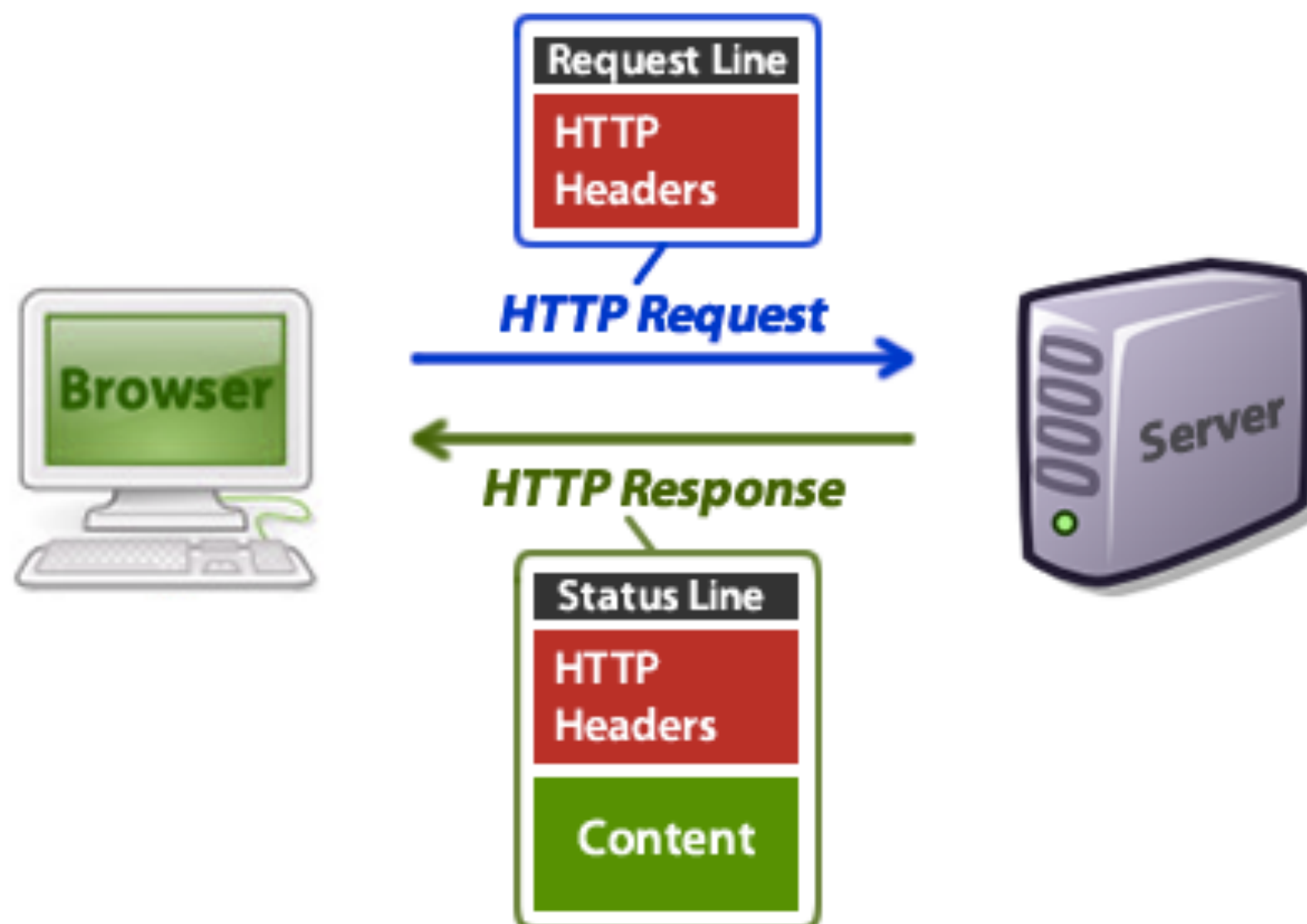
HTTP

requests & responses



HTTP

- Hyper Text Transfer Protocol
- Това е протокол (установени правила) за това как клиента (браузъра) и сървъра (web сървър) обменят информация
- Комуникацията между браузъра и сървъра се осъществява чрез HTTP съобщения, като тези, които са изпратени от браузъра се наричат заявка (*request*), а изпратените от сървъра - отговор (*response*)
- HTTP (и в частност HTTPS) се грижи за целия web трафик в интернет



http request

- Всеки път когато искаме да отворим дадена страница, браузъра (клиента) прави HTTP заявка, за да получи `index.html`
- Браузъра прави *request* към сървъра, като му изпраща съобщение (*message*), което съдържа следните реквизити:
 - request line
 - http headers
 - body parameters (само при POST заявки)
- След като получи `index.html`, прави отделен HTTP request за всеки един свързан файл от страницата

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

method	path	protocol
GET	/tutorials/other/top-20-mysql-best-practices/	HTTP/1.1

```
Host: net.tutsplus.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: PHPSESSID=r2t5uvjq435r4q7ib3vtdjq120
Pragma: no-cache
Cache-Control: no-cache
```

HTTP headers as Name: Value

Вижте заявките и съответните отговори в таб-а Network на Developer Tools

response

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

protocol **status code**

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
```

HTTP headers as Name: Value

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 ... -->
```

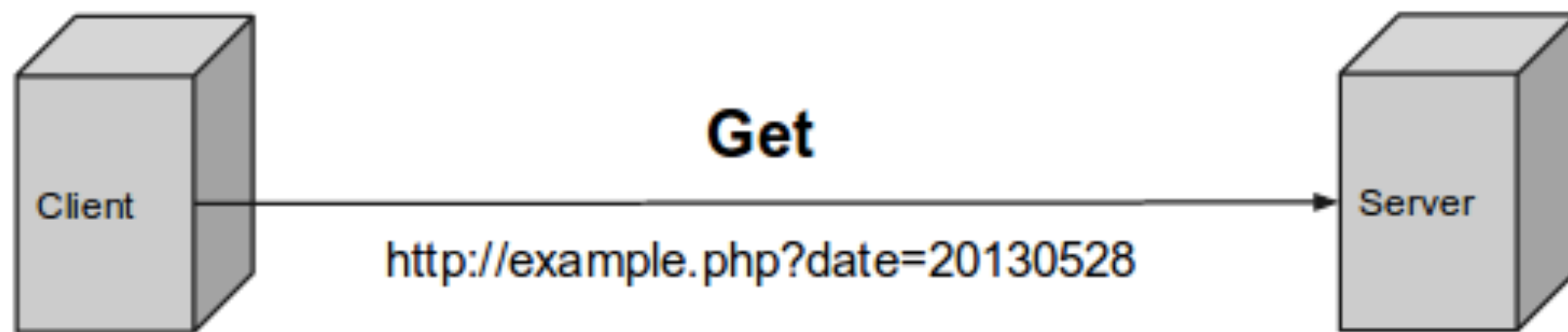

status code

- 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>

Други протоколи в интернет

- Не всичкият трафик в интернет е http/https (т.е. не всичкият трафик е *web трафик*)
- в интернет има също така и ftp трафик, където FTP е протокол за пренос на файлове (**F**ile **T**ransfer **P**rotocol)
- въпреки, че през HTTP също могат да се пренасят файлове, FTP е различен от HTTP
- основната разлика е, че HTTP трафика се случва между уеб клиент (браузъра) и уеб сървър, а FTP трафика е между FTP сървър и FTP клиент (например FileZilla)
- други протоколи са SMTP, IMAP, UDP и т.н.

POST vs GET



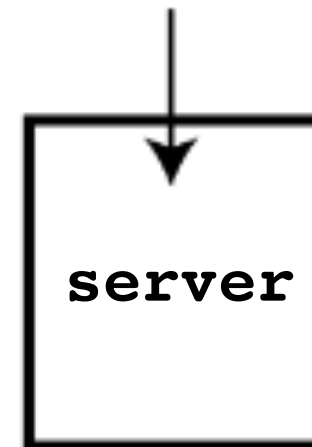
Form Data, JSON Strings, Query Parameters, View States, etc

POST vs GET

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

Using GET

`http://www.somedomain.com/register.asp?name=jobe&email=jobe@electrotank.com`



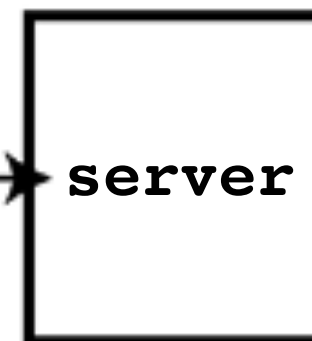
Using POST

`http://www.somedomain.com/register.asp`



HTTP Request

`name=jobe&
email=jobe@
electrotank.com`





WEB FORMS

<form>

- 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*:

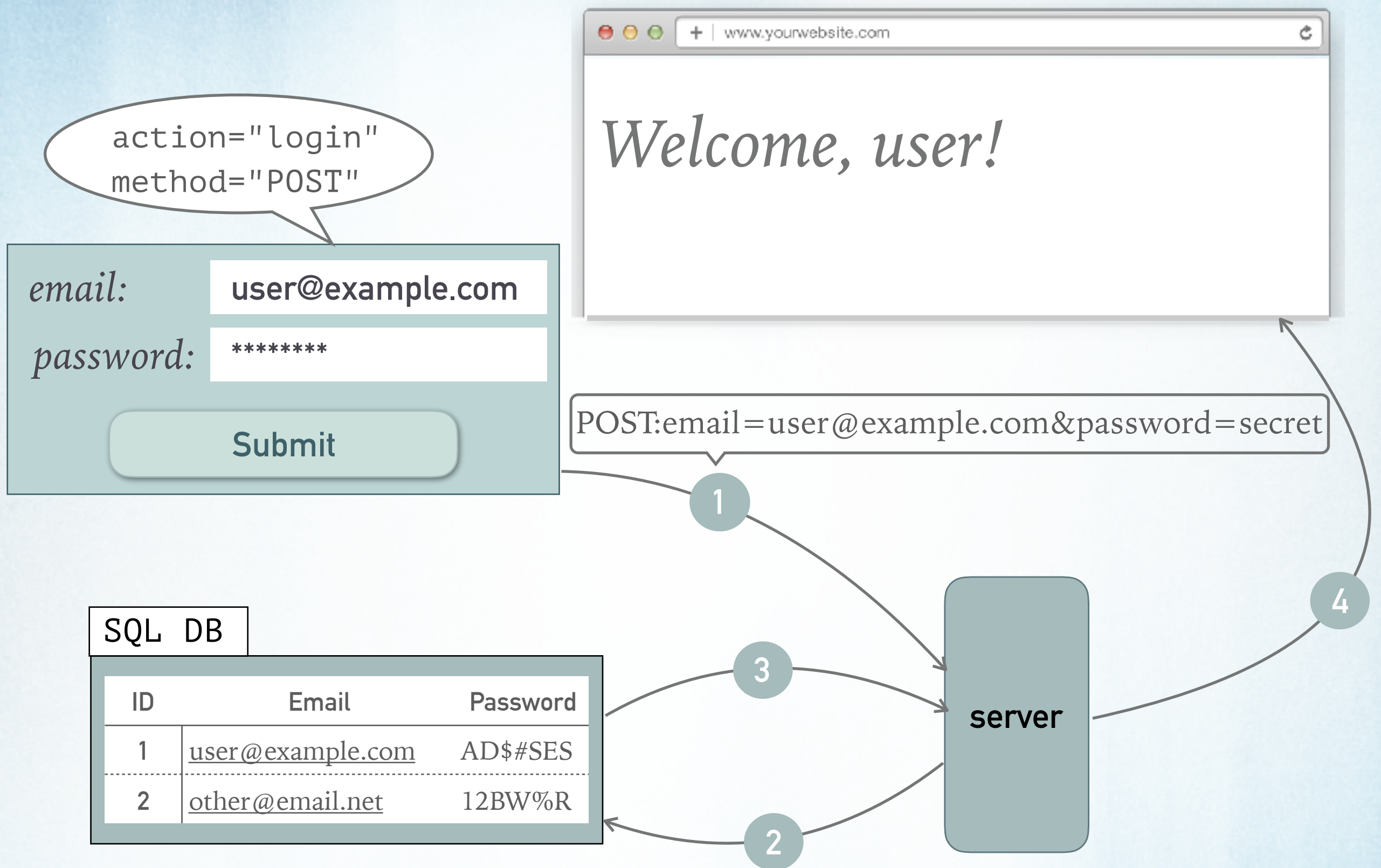
<form method="post"></form>

- form е HTML елемент, който се използва единствено за групиране на няколко други HTML елементи, наречени *полета*
- Елементите на формуляра, могат да бъдат:
текстови полета, падащи менюта, радио-бутони, checkboxes, бутони и скрити полета

- *form* елементът има два специфични атрибута, които определят къде ще отиде информацията, която се въвежда от потребителя и как
- **method**
указва типа на заявката, която изпращаме (POST или GET)
- **action**
указва физически адрес на backend приложението

<form method="post" action="api/send-feedback.php">

- action и method заедно определят уникален адрес т.нар. endpoint, където бекенд приложението приема заявки. Всички endpoints на дадено бекенд приложение формират т.нар. API (access program interface) . Често казваме API на самото приложение.



Полета на уеб формурярите

- Полетата за въвеждане на данни от потребителя ще наричаме накратко "инпут полета"
- Синтаксис:
<input type="..." name="..." value="...">
- Имат три важни атрибута:
 - **name**: задава име на параметъра, който ще се изпрати към сървъра
 - **type**: указва на браузъра какъв тип е полето: text, password, radio, date, submit, ...
 - **value**: не е задължителен (освен за полетата с избор на опции), но когато се използва, задава предварителна стойност на полето
- input елемента няма затварящ таг!

Поле за въвеждане на текст

- `<input type="text">` - за имена, телефон, имейл и т.н..
- `<input type="password">`
не показва изписаният от потребителя текст
- `<input type="email">`
същото като text, но се прилага валидация за email
- `<textarea cols="20" rows="5">A pre-filled text here</textarea>`
това е поле за писане на дълъг текст (използвайте CSS width и height вместо cols и rows)
- HTML5 типове на полета: url, date, color, number, range и др.

Текстови полета

Get in touch

text input → Your name:

text input → email:

Your message:

text area →

Бутони, които изпращат

- Бутони за изпращане (submit)
 - `<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 изпращат

- 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:

Spoken languages:

checkboxes

→ ☐ English

☐ Bulgarian

☐ French

Do you like my page?

radio

→ ☐ Yes ☐ No

Checkbox

- Списък от опции без ограничение на избора

```
<label for="accept-box">Приемам условията за ползване</label>  
<input type="checkbox" name="accept" id="accept-box">
```

- визуализира 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*

Radio бутони

- Списък от опции с възможност за максимум един избор
- `<input type="radio" name="choise" value="1">`
- визуализира се като кръгче и има смисъл от него само ако има поне още един такъв, който да е в същата група 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)

```
<select>
```

```
  <option value="...">Option 1</option>
```

```
  <option value="...">Option 2</option>
```

```
  ...
```

```
</select>
```

<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></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 искаме 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*
- обикновено се използват от *javascript* за динамично генерирани стойности (*value*)
- Например за изпращане на геолокацията на потребителя, за изпращане на потребителски ID и други

Други полета

- Файл ъплоад:

`<input type="file">`

- HTML5 нови типове полета:

Имейл	-	<code><input type="email"></code>
URL	-	<code><input type="url"></code>
Телефон	-	<code><input type="tel"></code>
Избор на цвят	-	<code><input type="color"></code>
Дата	-	<code><input type="date"></code>
Обхват	-	<code><input type="range"></code>
Число	-	<code><input type="number"></code>

Други специални input 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 валидацията
- `selected`
за предварително избрана опция (*option*) на *select box*
- `checked` - за чекбокс и радио

Задача за упражнение

- Направете контактна форма, която да има следните полета:
 - Имена
 - email
 - език (Английски, Немски, Български)
 - телефон
 - съдържание
 - бутон за изпращане
 - reset бутон
- Свърете с картинката от следващия слайд

contact.us

Свържете се с нас!

Вашите имена

Име Фамилия

Вашият имейл

email@example.com

Избор на език

Български ▾

Телефон

+359xxxxxxxxx

Съобщение

Опишете Вашето запитване тук

Изтрий

Изпрати

Задача

Създайте страница, която да изглежда по следните 4 начина в зависимост от размера на екрана:



<http://swift-academy.zenlabs.pro/misc/tasks/lesson6/circles-task.zip>

Задача

- Дефинирайте 4 различни състояния за следните размери на екрана:
 - < 459 (small)
 - 460 - 679 (medium)
 - 680 - 999 (large)
 - > 1000 (extra-large)
- Начина, по който страницата трябва да изглежда във всяко едно от тях, можете да видите в съответния скрийншот в папката samples.
- Използвайте [Just A Color Picker](#) за да вземете правилните цветове от примера

Въпроси?

Полезни връзки

- Валидация на форми:

<https://www.sitepoint.com/html5-form-validation/>

https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Form_validation

<http://www.the-art-of-web.com/html/html5-form-validation/>

- Fieldsets

<https://developer.mozilla.org/en/docs/Web/HTML/Element/fieldset>

Примери

<http://swift-academy.zenlabs.pro/lessons/lesson6/examples>

Домашно

<http://swift-academy.zenlabs.pro/lessons/lesson6/homework>