библиотека jquery
быстрый взгляд

CDN -- Content Delivery Network

нам нужно для сайта чтобы у браузера юзера была какая то библиотека (например jquery)
мы можем

1. просто эту библиотеку скачать на сервер и присылать на запросы
2. брать ее с CDN
   CDN это сервера по всему миру на которых лежат всякие популярные библиотеки
   их предоставляют всякие компании
   из плюсов

- то что все может работать быстрее (если пользователь далеко от нас но не от cdn)
- когда юзер впервые зайдет на сайт вполне возможно что с cdn-а уже будет скачан jquery (т к он популярный), и значит браузре не будет его снова скачивать и наш сайт отобразится быстрее
  минусы
- сайт не работает без интернета
- проблемы с cdn (их дудосят, инженеры проебались и т п)

подключаем так:

```
<script src="/js/jquery-3.6.1.min.js"></script>
```

#todo добавить ссылку сюда в лекцию css

так размещается на страничке js код

```
<#import "commons.ftlh" as c>

<@c.page>
    Welcome on Codeforces!
    <script>
        alert(1);
    </script>
</@c.page>
```

функция $(); это магическая jquery-функция, которая вызовет наш код примерно тогда когда мы хотим, когда дерево DOM уже будет построено

если в $() передается строка, которая может быть распаршена как css селектор, осущетсвляется запрос чтобы найти все соответсвующие элементы

прикол:
если у ноды a был сын img а потом мы сделали ("a").text("Hello") , сын убьется и вместо него подвесится нода текста

```
.click(fun)
вызовет функцию при клике на элемент
до того как напимер перейти по ссылке
```

```
если fun вернет false то исполнение события прервется (мы не перейдем по ссылке)

это работает так: все обработчики какого то события лежат в стеке, fun на вершине стека,
и если он возвращает false программа выходит из стека
"не передаем дальше в цепочку обработчиков действие клик"
```

внутри fun:

```
$(this).text("Clicked!");
```

у js функций есть контекст вызова, мы обращаемся к нему
this -- типа DOM элемент, а $(this) -- он но превращенный в jquery (теперь у него есть .css, .text и т п)

мы ловим формы на submit а не клик потому что понятно почему это правильнее
соглашение: в названии переменной использовать $ (в начале) если в ней хранится jquery

асинхронный неблокирующий запрос на сервер

-- короче интуитивно

```
$.ajax({
    method: "POST",
    data: { action: "register", login, password },
    dataType: "json",
    success: function (response) {
        return false;
    }
});
```

success -- если (когда) сервер вернет нам ответ

прицип подход AJAX

асинхронный javascript and xml
ну по факту у нас это будет javascript and json

аннотация

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
public @interface Json {
}
```

аннотация retention runtime говорит что данная (json здесь) аннотация будет нужна в рантайме поэтому не вырезай ее на этапе компиляции
target -- к чему применят аннотацию

```
boolean json = method.getAnnotation(Json.class) != null;
```

```
@Json
private void register(HttpServlet
    User user = new User();
```

"Шаблонизация на стороне браузера":

```
<template>
    <tr>
        <td class="user-id"></td>
        <td class="user-login"></td>
        <td class="user-creationTime"></td>
    </tr>
</template>
```

```
response["users"].forEach(function (user) {
    const $tpl = $($users.find("template").prop("content")).clone();
});
```

```
$users.find("tbody").append($tpl);
```

```
$tpl.find("tr").attr("data-userId", user.id);
```

```
$.notify("Creatic
("${message?js_string}"
```

```
window.notify = function (messa
```

window это типа супер глобальнгая переменная и мы определили у нее метод notify и теперь сможем его вызывать

```
notify("${m
```

упражнение

так а зачем нужен ajax?

почему

```
.prop("content")
```