



НИ Е ВЯРВАМЕ ВЪВ ВАШЕТО БЪДЕЩЕ

Събития



И така .. бърз преговор :)

- Кое от следните е събитие?
 - звъни ми телефона —> **Да**
 - страницата зарежда бавно —> **не е събитие, а състояние**
 - купувам си кафе —> **Да**
 - няма кафе (даден артикул е изчерпан)
\\ —> **това е state**

Събития

- Събитията ни показват, че нещо току-що се е случило
- Събитията възникват, а `eventListener`-ите ги регистрират
- Използваме ги, за да следим какво се случва и да зададем съответна реакция
- Реакцията ни се нарича обработка на събитието
- По подробно - в лекция 11:
<http://zenlabs.pro/courses/lessons/lesson11/lesson.pdf>

Ок, май стана ясно вече ..



DOMContentLoaded

- Това е първото събитие, което се случва при зареждането на страницата
- Възниква в момента, в който DOM-а е зареден в `document` обекта и браузъра визуализира страницата
- Най често го ползваме за скриптове, които ще генерират някакво съдържание или ще покажат някой досаден изкачащ прозорец, съобщаващ, че днес има промоция на нещо
- Идеята е, че всички скриптове, които генерират съдържание, ползват `appendChild` метода, за да закачат съдържанието към елемент от DOM-а
- Преди DOM-а да е готов, обаче, това няма как да стане

Как се ползва:

```
document.addEventListener('DOMContentLoaded', function() {  
    console.log('the DOM is ready!');  
});
```

или с jQuery:

```
$(document).ready(function() {  
    console.log('the DOM is ready!');  
});
```

window onload

- Подобно на DOMContentLoaded, но с разликата, че това събитие възниква чак след като абсолютно всичко е заредено (много неща продължават да се даунлоад-ват и да се парсват дори и след зареждането на DOM-а, като например скриптовете в края на html-а)
- Т.е. window onload се случва след DOMContentLoaded
- Освен това, когато се случи това събитие, то изпълнява функцията window.onload:

```
window.onload = function() {  
    console.log("The BOM is ready");  
};
```

- <https://developer.mozilla.org/en-US/docs/Web/API/GlobalEventHandlers/onload>

addEventListener()

- За да регистрираме и обработим събитие, използваме `addEventListener()` метода на DOM обектите
- синтаксис:
`document.addEventListener('DOMContentLoaded', callback);`
- където `callback` референция към функция или анонимна функция
- тази функция се нарича обработваща (или `handler`) и приема като първи аргумент `event` обекта, който носи информация за текущия `event` (текущото събитие)

Event обекта

- При възникване на събитие, то се присвоява на обект от класа Event
- не ни трябва да знаем всичко за този клас. Това, което трябва да знаем за ивента са точно 2 неща:
 - винаги се подава като параметър на обработващата функция (handler-a)
 - има си дефолтен handler (функция за обработка по подразбиране), който винаги се изпълнява при възникването на ивента, независимо дали ние сме го хендълнали или не
- За да откажем изпълнението на дефолтния хендлър използваме метода `preventDefault()`:

`e.preventDefault()`

Примери



**addEventListener... No ...prevent! No ...
AAAAAAGH!**

Въпроси?

JQuery events

- Всичко е на едно място и става по един начин
- cross-device & cross-browser (почти)
- бави зареждането
- но пък ускорява разработката и предпазва от cross-browser проблеми
- <https://api.jquery.com/category/events/>

JQuery events

- `click()`
- `focus()`
- `keyup()`, `keydown()`, `keypress()`
- `mouseover()`, `mouseout()`
- `blur()`
- Drag & Drop example

MAKE ALL EXAMPLES



Въпроси?

Полезни връзки



**KEEP
CALM
AND
LEARN
JAVASCRIPT**

Примери

<http://zenlabs.pro/courses/lessons/lesson17/examples.zip>

Домашно

<https://github.com/zzeni/swift-academy-homeworks/blob/fe-03/tasks/L17>