SWIFTACADEMY

НИЕ ВЯРВАМЕ ВЪВ ВАШЕТО БЪДЕЩЕ

Събития (Events) - Част 1

Събития

- Представете си, че искаме да следим кога входната врата се отваря
- (да, това ще бъде "събитие")
- но не искаме да стоим и да я гледаме постоянно в очакване
- Едно от решенията би било да накараме някой да я наблюдава вместо нас
- това ще бъде т. нар. **EventListener**

Capturing & Handling

SWIFTACADEMY

НИЕ ВЯРВАМЕ ВЪВ ВАШЕТО БЪДЕЩЕ

Представете си, че не само искаме да разберем дали вратата се отваря, но и да предприемем някакви конкретни действия, всеки път, когато това стане.

Например:

- бихме могли да предположим, че някой влиза и да кажем "Добър ден"

- при допълнителен оглед е възможно да установим, че става течение и да затворим прозореца

- или може би просто искаме да преброим колко пъти ще се отвори за деня

Обработка

- Действията, които предприемаме при възникването на събитие, се наричат обработка на събитието (**event handling**)

- Това как ще обработим дадено събитие, зависи изцяло от нас

- В случая с вратата, ще трябва да кажем на нашият *EventListener*, че когато види вратата да се отваря (**event capturing**), трябва да подаде сигнал към *EventHandler*-а

- *EventHandler*-а от своя страна ще извърши обработката на събитието, като следва нашите инструкции (например ще увеличи брояча с +1)

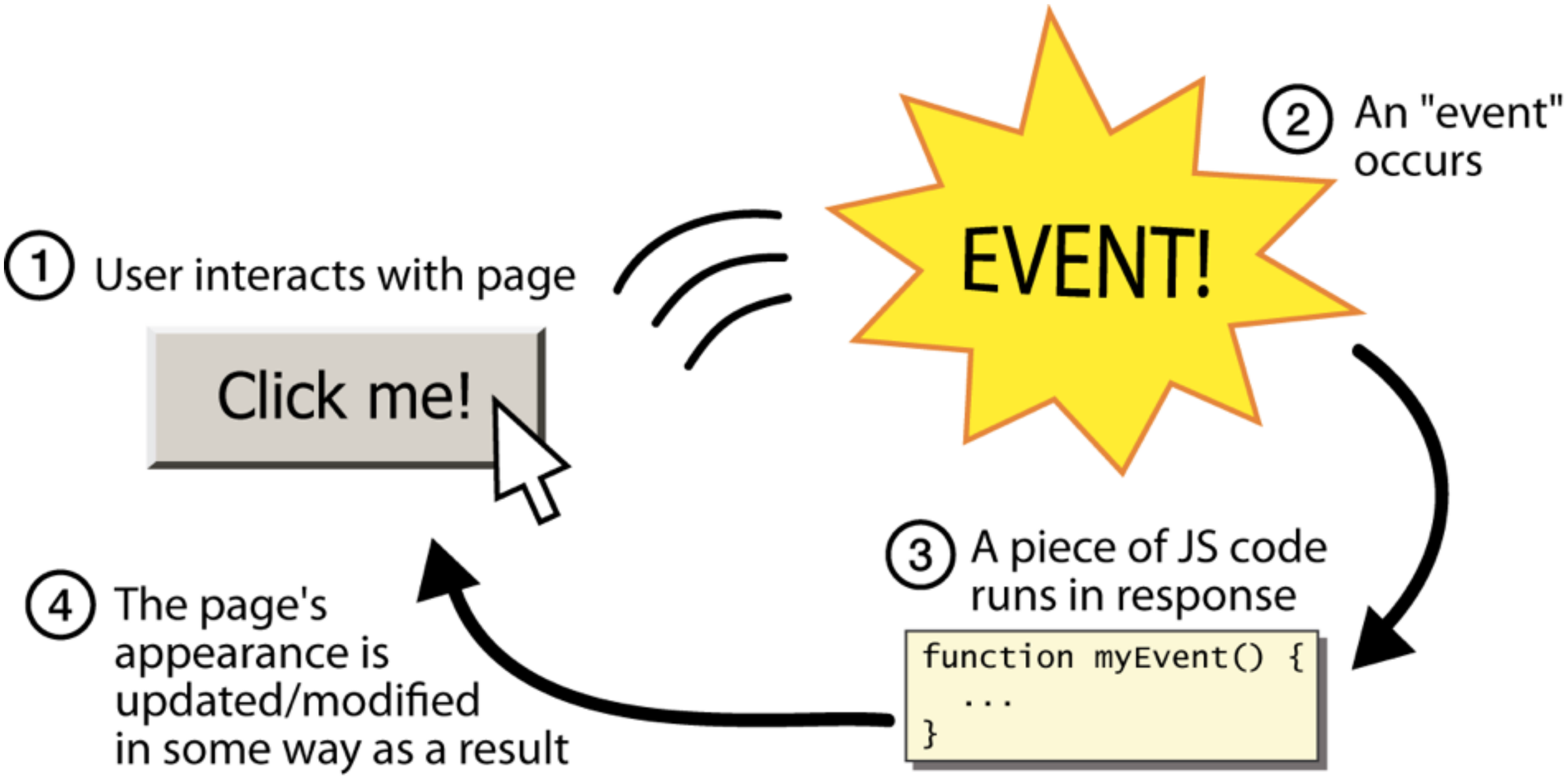
Заключение

- Събитията настъпват в момента, в който нещо премине от едно състояние в друго
- 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
- Избраната от нас обработка на различните събития определя поведението (**behaviour**-а) на нашата програма/приложение

Събития - Част 2

Кое от следните е събитие?

- Телефона звъни **—> Да**
- страницата зарежда бавно **—> не е събитие, а състояние**
- Някой ме поздравява **—> Да**
- Кафе-машината няма чашки **\ —> това е state**
- Кафе-машината дава сигнал, че кафето е готово **\ —> това е събитие**


1 User interacts with page
Click me!
EVENT!
2 An "event" occurs
3 A piece of JS code runs in response
function myEvent() {
...
}
4 The page's appearance is updated/modified in some way as a result

Пример:

http://codepen.io/jenie/pen/bBowNd?editors=0010

Помните ли какво беше DOM?

"представянето на HTML документа като javascript обект"

DOMContentLoaded

- Това е първото събитие, което настъпва изобщо
- Възниква в момента, в който html-а е зареден в **document** обекта и в който браузъра започва да визуализира страницата
- Нашият Javascrpt код винаги се изпълнява веднага, след като браузърът го даунлоадне. Това може да е проблем ако скриптът ни се опита да ползва елементи от DOM дървото, а браузъра все още не е даунлоаднал целия HTML на страницата
- За да решим този проблем или трябва да сложим целият JS код да се изпълнява след настъпването на DOMContentLoaded или да включваме js файловете най-отдолу в HTML-а (точно преди да затворим <body> тагът)

Как се ползва

Native JS:

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

callback

или с jQuery:

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

Синтаксис

Native JS:

```
DOM-ELEMENT.addEventListener('EVENT-NAME', function() {
  console.log('Event registered');
});
```

jQuery:

```
$('DOM_SELECTOR').EVENT-NAME(function() {
  console.log('Event registered');
});
```

Задача

- Направете страница с 2 бутона
- При клик на бутон 1 фонът на страницата става син
- При клик на бутон 2 - фонът на страницата става оранжев, ако е бил бял и червен ако е бил син
- При дабълклик на бутон 1 - фонът на страницата става бял
- `document.body.style.backgroundColor = "blue"`

Решения на задачата:

- Native JS:
  - http://codepen.io/jenie/pen/BQwRJm?editors=1010
- jQuery:
  - http://codepen.io/jenie/pen/wordpp?editors=0011

window onload

- 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
- window onload е еквивалента на DOMContentLoaded събитието, но настъпва чак след като се зареди абсолютно всичко от страницата
- 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 е 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, се създава нов обект от класа Event, който съдържа в себе си информацията за събитието
- Важното, което трябва да знаем за event обекта е:
  - винаги се получава като първи аргумент в обработващата функция (handler-a)
  - носи информация за текущото събитие
  - може да се използва за отказване на страндартната обработка (тази по подразбиране) на събитието, чрез метода **preventDefault()**
- http://www.w3schools.com/jsref/dom_obj_event.asp

target на събитието

- Елементът, върху който се изпълнява събитието (този, чието състояние се променя в резултат на настъпилото събитие), се нарича **target** на събитието
- Този target може да се получи от event аргумента на handler функцията по следния начин: **event.target**
- Пример:

```
$("button").click(function(event) {
  console.log('here I am:', event.target);
});
```

- https://developer.mozilla.org/en-US/docs/Web/API/Event/target

this на събитието

- Това е елементът, чрез който е прихванато събитието (този на който е закачен **eventListener** за съответното събитие)
- Можем да го използваме само в **callback** функцията
- Пример:

```
$("document").click(function(event) {
  console.log(event.target); // => "<button>click me!</
button>"
  console.log(this); // => document
});
```

- *Важно: често **this** и **target** на event-а са едно и също, но не винаги!*

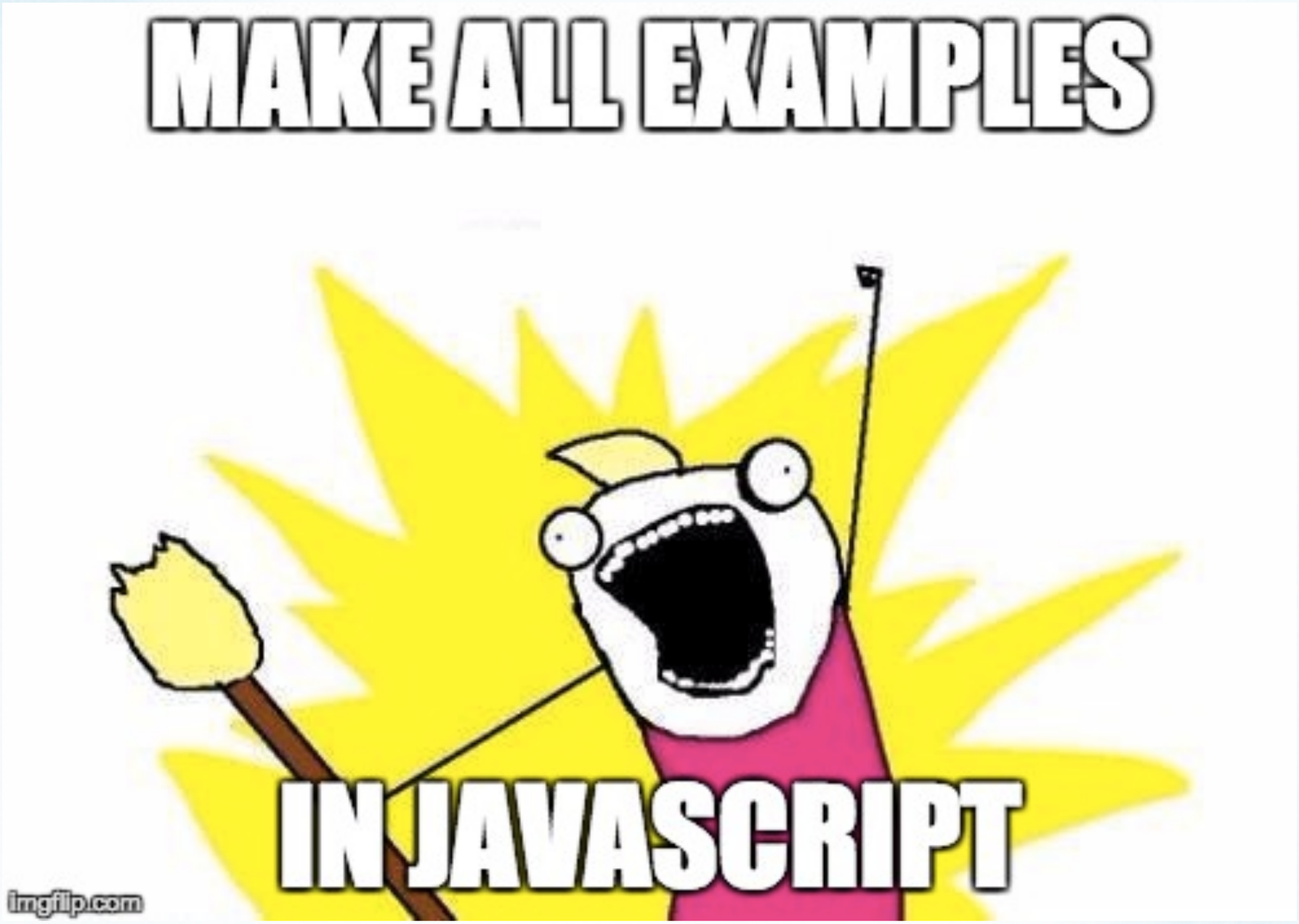
Въпроси?

Примери

addEvent!.. No ..prevent! No ... AAAAAAGH!

jQuery events

- click()
- focus()
- keyup(), keydown(), keypress()
- mouseOver(), mouseOut()
- blur()
- https://api.jquery.com/category/events/


MAKE ALL EXAMPLES
IN JAVASCRIPT
imgflip.com

Въпроси?

Полезни връзки

- MDN event docs:
  - https://developer.mozilla.org/en-US/docs/Web/Events
  - https://developer.mozilla.org/en-US/docs/Web/API/Event
  - https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener
- Code pens:
  - http://codepen.io/jenie/pen/vKBxBz
  - http://codepen.io/jenie/pen/qNWojq?editors=1011
  - http://codepen.io/jenie/pen/oLvZLy?editors=1010


KEEP
CALM
AND
LEARN
JAVASCRIPT

Примери

http://swift-academy.zenlabs.pro/lessons/lesson17/examples/download.zip

Домашно

http://swift-academy.zenlabs.pro/lessons/lesson17/homework