SWIFTACADEMY

НИЕ ВЯРВАМЕ ВЪВ ВАШЕТО БЪДЕЩЕ

Събития (Events) - Част 1

SWIFTACADEMY

НИЕ ВЯРВАМЕ ВЪВ ВАШЕТО БЪДЕЩЕ

Front-End Development

Събития

- Представете си, че искаме да следим кога в любимият ни магазин правят разпродажба

- (да, това ще бъде "събитие")

- но не можем просто да стоим пред вратата му и да го гледаме в очакване (нито пък постоянно да наблюдаваме уебсайта му)

- Едно от решенията би било да накараме някой да го наблюдава вместо нас

- това ще бъде т. нар. **EventListener**

Capturing & Handling

SWIFTACADEMY

НИЕ ВЯРВАМЕ ВЪВ ВАШЕТО БЪДЕШЕ

Представете си, че не само искаме да разберем кога има разпродажба, но и да предприемем някакви конкретни действия, всеки път, когато това стане.

Например:

- да проверим кой са намалените артикули
- да разберем дали можем да комбинираме отстъпки и дали разпродажбата важи при онлайн покупка
- или може би просто искаме да си купим нещо конкретно

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 магазина, ще трябва да кажем на нашият *EventListener*, че когато види че има разпродажба (**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 пресметне сумите на отстъпките и ще направи поръчка)

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 нашето приложение/уебсайт

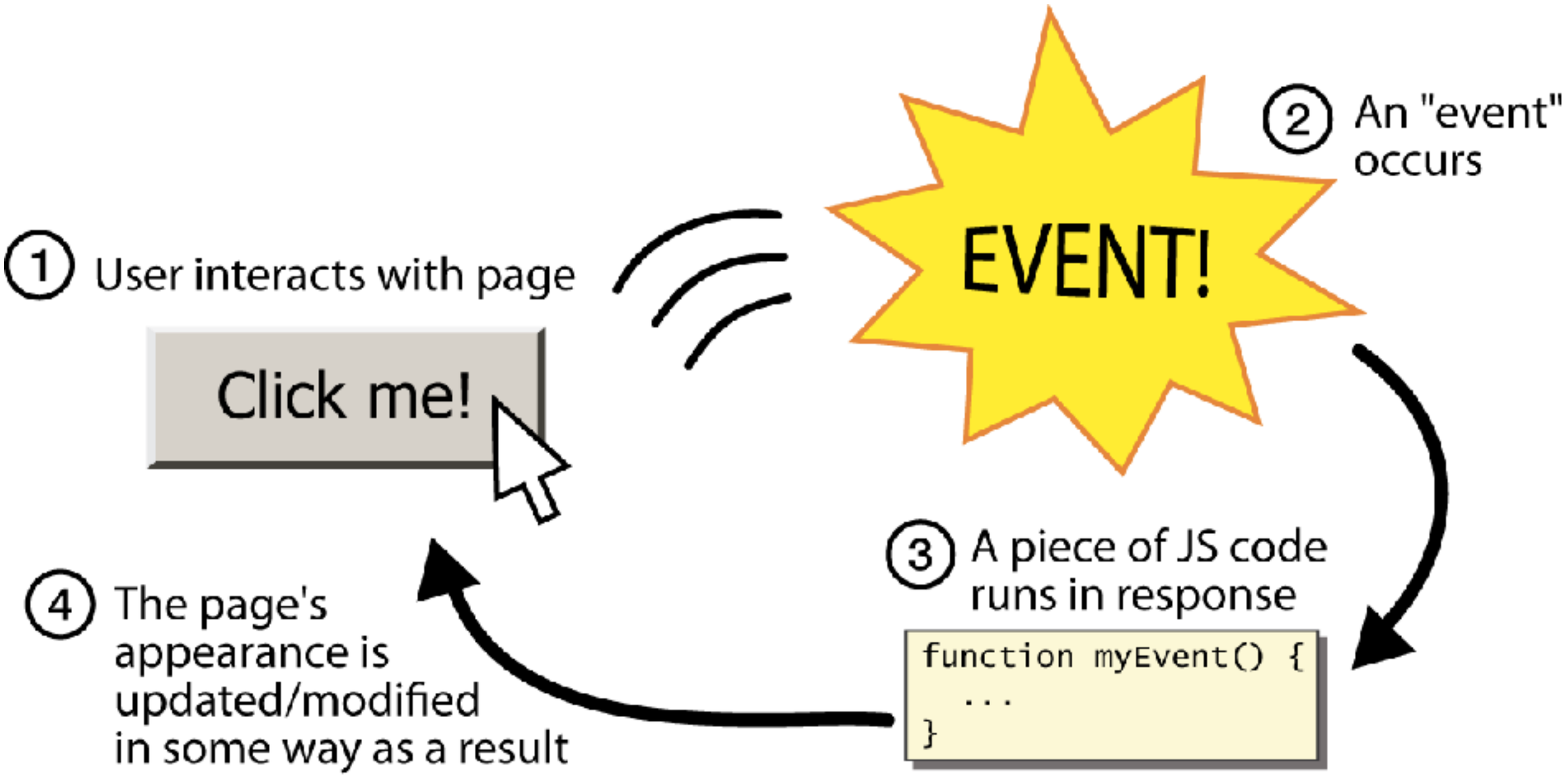
Събития - Част 2

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**

Кое от следните е събитие?

- натискане на бутон **—> click / onclick**
- минаване на мишката над елемент **—> mouseover / hover**
- промяна на размера на прозореца на браузъра **—> resize**
- скролване с мишката **—> scroll**
- изпращане на форма (submit) **—> submit**
- писане в инпут поле **—> keypress / keyup / keydown**


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/dWWwrW

Помните ли какво беше DOM?

"представянето на HTML документа като javascript обект"

DOMContentLoaded

- Това е първото събитие, което настъпва след отварянето на дадена страница
- 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!');
});
```

или с jQuery:

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

callback

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)

Синтаксис

Native JS:

```
element.addEventListener(eventName, function() {
  console.log('Event registered');
}, useCapture);
```

jQuery:

```
$(domSelector).eventName(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

Задача

- Напишете скрипт, който да извежда следните 2 заглавия в една страница:
    - "PAGE DOM LOADED" - при настъпване на DOMContentLoaded събитието
    - "PAGE BOM LOADED" - при wibdow.onload
- Направете обработка на събитието клик (където и да е върху страницата), като изписвате текста "click captured" и сменяте цвета на фона на страницата (можете да редувате 2 цвята с classList.toggle функцията)

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 изполва 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!*

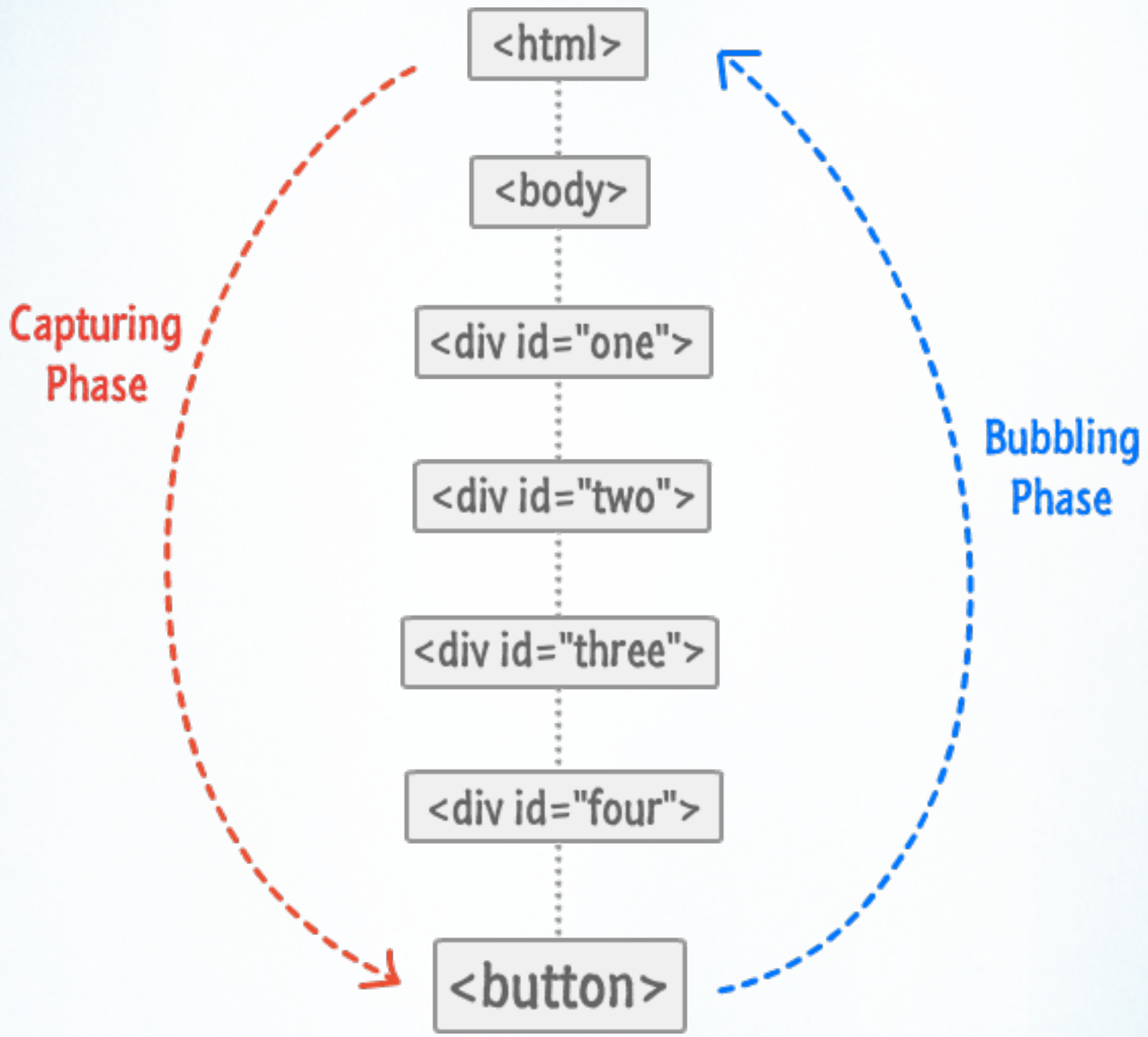
Въпроси?

Capturing & bubbling

- Тези 2 понятия се отнасят за 2те фази при регистрирането и обработката на събитието:

  - capturing (улавяне) на събитието е фазата, в която се изпълняват всички "обработки" на събитието, регистрирани в capture фаза.
    Изпълнението става в посока от най-външния елемент, обхващащ target-а на събитието (event target-а), към най-вътешния (самият event target)

  - bubbling (изплуване) на събитието е фазата, в която се изпълняват всички "обработки" на събитието, регистрирани в bubble фаза.
    Тази фаза се изпълнява след capture фазата и реда на изпълнение е в обратна посока. Т.е. от най-вътрешния, към най-външния елемент (от там идва "изплуване")


<html>
<body>

<button>
Capturing Phase
Bubbling Phase

Capturing & bubbling

- capture фазата е преди bubble фазата
- по подразбиране всички събития се регистрират в bubble фазата
- за да регистрираме обработка на събитие в capture фазата, използваме опционалния 3-ти параметър на addEventListener със стойност true:

```
element.addEventListener(eventName, callback, true);
```

- https://www.w3.org/TR/DOM-Level-3-Events/#event-flow
- https://www.quirksmode.org/js/events_order.html#link4
- https://www.youtube.com/watch?v=sfKDOOJgbSI

removeEventListener

- Изпълнението на обработката на едно събитие се нарича dispatching
- Ако искаме да премахнем listener-а (и така да отменим изпълняването на обраротващата функция) след първата обработка, можем да използваме 2 подхода:
  - по-лесен:

```
element.addEventListener(eventName, callback, { once: true });
```

  - по-труден:

```
element.removeEventListener(eventName, callback, useCapture);
```

където сойностите на eventName, callback и useCapture, трябва да съвпадат със стойностите зададени на addEventListener при създаването му

Примери

addEvent!.. No ..remove! No ... AAAAAAGH!

Примери

- http://codepen.io/jenie/pen/vKBxBz
- http://codepen.io/jenie/pen/qNWojq?editors=1011
- http://codepen.io/jenie/pen/oLvZLy?editors=1010

Събития

Focus Events

Event Name	Fired When
`focus`	An element has received focus (does not bubble).
`blur`	An element has lost focus (does not bubble).

Form Events

Event Name	Fired When
`reset`	The reset button is pressed
`submit`	The submit button is pressed

https://developer.mozilla.org/en-US/docs/Web/Events

Събития

Clipboard Events

Event Name	Fired When
`cut`	The selection has been cut and copied to the clipboard
`copy`	The selection has been copied to the clipboard
`paste`	The item from the clipboard has been pasted

Keyboard Events

Event Name	Fired When
`keydown`	ANY key is pressed
`keypress`	ANY key except Shift, Fn, CapsLock is in pressed position. (Fired continously.)
`keyup`	ANY key is released

https://developer.mozilla.org/en-US/docs/Web/Events

Събития

Drag & Drop Events

Event Name	Fired When
dragstart	The user starts dragging an element or text selection.
drag	An element or text selection is being dragged (Fired continuously every 350ms).
dragend	A drag operation is being ended (by releasing a mouse button or hitting the escape key).
dragenter	A dragged element or text selection enters a valid drop target.
dragover	An element or text selection is being dragged over a valid drop target. (Fired continuously every 350ms.)
dragleave	A dragged element or text selection leaves a valid drop target.
drop	An element is dropped on a valid drop target.

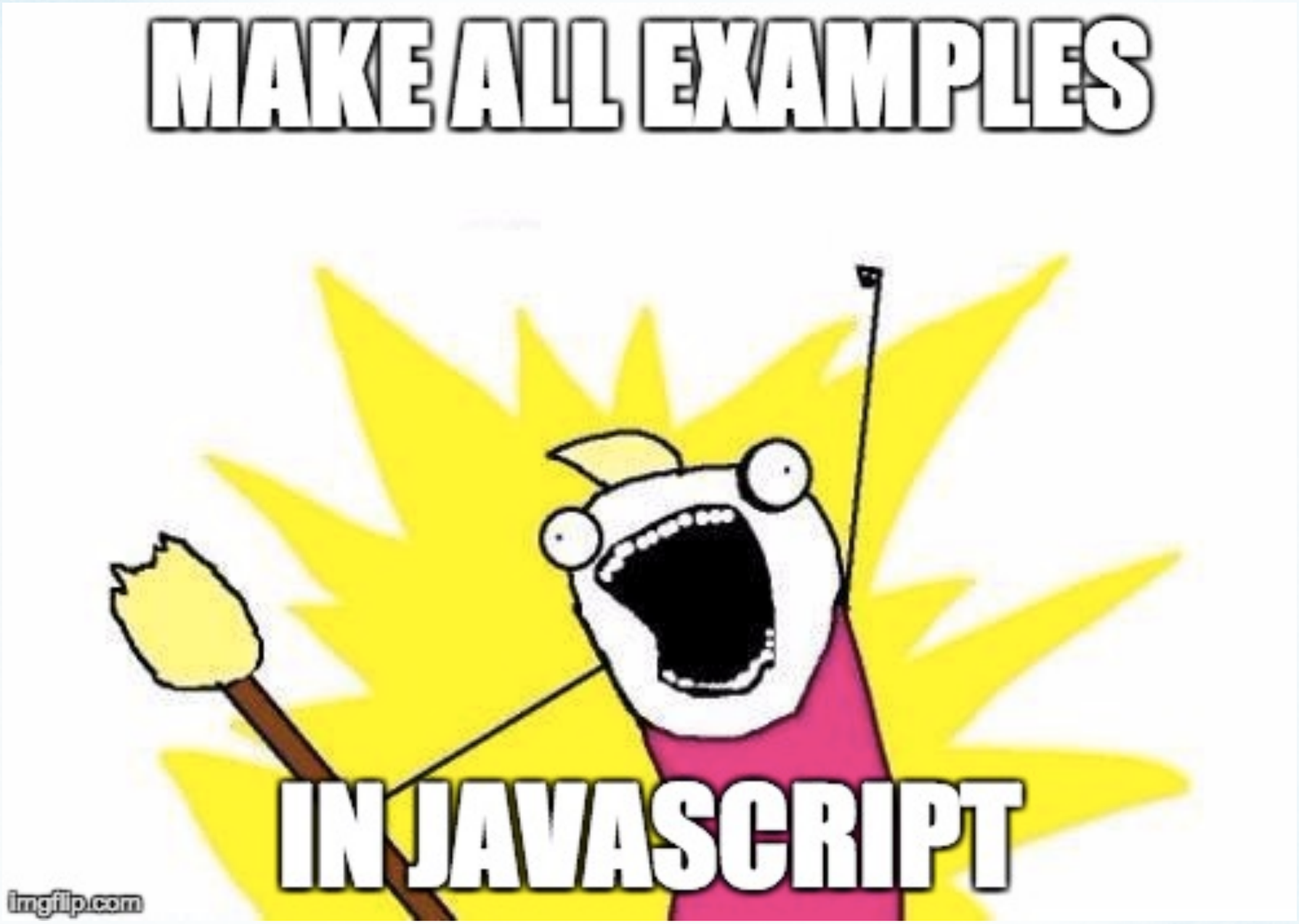
https://developer.mozilla.org/en-US/docs/Web/Events

Mouse Events

Event Name	Fired When
mouseenter	A pointing device is moved onto the element that has the listener attached.
mouseover	A pointing device is moved onto the element that has the listener attached or onto one of its children.
mousemove	A pointing device is moved over an element. (Fired continously as the mouse moves.)
mousedown	A pointing device button is pressed on an element.
mouseup	A pointing device button is released over an element.
click	A pointing device button (ANY button; soon to be primary button only) has been pressed and released on an element.
dblclick	A pointing device button is clicked twice on an element.
contextmenu	The right button of the mouse is clicked (before the context menu is displayed).
mouseleave	A pointing device is moved off the element that has the listener attached.
mouseout	A pointing device is moved off the element that has the listener attached or off one of its children.
select	Some text is being selected.

jQuery events

- click()
- focus()
- keyup(), keydown(), keypress()
- mouseOver(), mouseEnter(), 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
  - https://developer.mozilla.org/en-US/docs/Web/Events
- Capture & bubble: https://www.youtube.com/watch?v=sfKDOOJgbSI

KEEP CALM AND LEARN JAVASCRIPT

Примери

http://swift-academy.zenlabs.pro/lessons/lesson19/examples/download.zip

Домашно

http://swift-academy.zenlabs.pro/lessons/lesson19/homework