SWIFTACADEMY

НИЕ ВЯРВАМЕ ВЪВ ВАШЕТО БЪДЕЩЕ

“

Because JavaScript can be used without understanding, the understanding of the language is often never attained.

”

– You Don't Know JS

Document Object Model

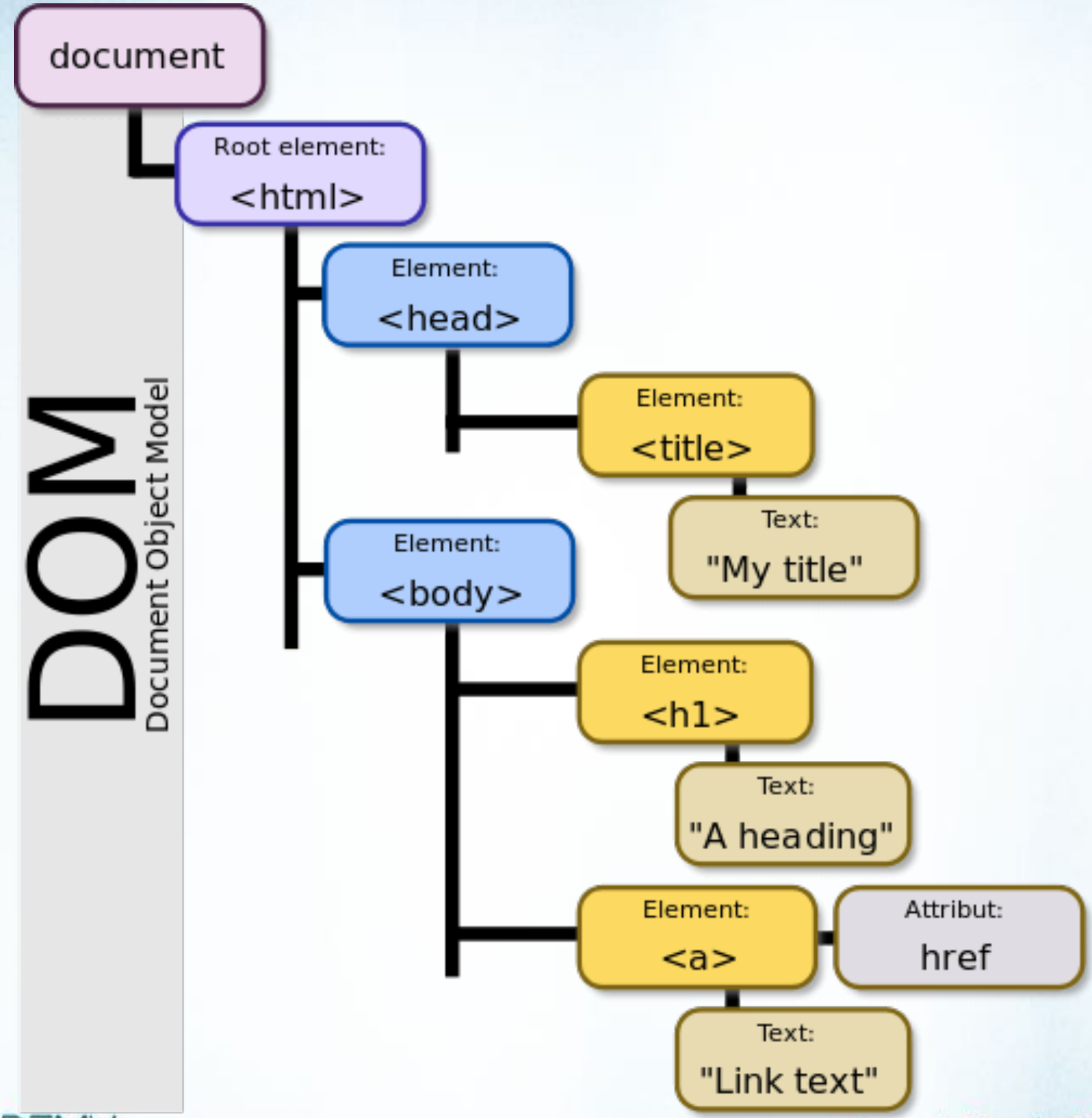
SWIFTACADEMY

НИЕ ВЯРВАМЕ ВЪВ ВАШЕТО БЪДЕШЕ

Front-End Development

Обектен модел на документа

- Ако си спомняте html страниците се наричаха документи
- Също така, html кода беше дървовидна структура с root елемент (<html>)
- А как ви се струва, да имахме начин да достъпваме и променяме отделни елементи от html кода с JavaScript?
- Всъщност можем -> благодарение на глобалният обект *document*
- Начинът, по който този обект е изграден, се нарича DOM


DOM
Document Object Model
document
Root element:
<html>
Element:
<head>
Element:
<title>
Text:
"My title"
Element:
<body>
Element:
<h1>
Text:
"A heading"
Element:
<a>
Attribut:
href
Text:
"Link text"

DOM selectors

- document.body
- document.getElementById("myLittlePony")
- document.getElementsByClassName("awesome")
- document.getElementsByTagName("div")
- document.querySelector("a.top-link")
- document.querySelectorAll("a:visited")

!!!

getElementById & getElementsByClassName

- Тези два DOM селектора, ни позволяват съответно:
  - да получим елемента, имащ конкретно ID
  - да получим списък с елементите, имащи конкретен клас
- Резултата от getElementById е обект от тип Element
- Резултата от getElementsByClassName е обект с тип HTMLCollection
- Т.е. получаваме или DOM елемент, или колекция (списък) от DOM елементи.
- DOM елементите ни позволяват да управляваме съдържанието на уеб страницата през JavaScript-а

querySelector
& querySelectorAll

- Тези два DOM селектора, получават като аргумент класически CSS селектор (това, което използваме в CSS за да задаваме стилове на елементите) и ни позволяват съответно:
  - да получим първият елемент, отговарящ на дадения селектор
  - да получим списък с елементите, отговарящи на дадения селектор
- Резултата от querySelector е обект от тип Element
- Резултата от querySelectorAll е обект с тип NodeList
- Пример:

```
var nav = document.querySelector("nav");
var listItems = document.querySelectorAll(".list-item");
```

Пример:

HTML

```
<div class="square"></div>
```

CSS

```
.square {
  width: 100px;
  height: 100px;
  background-color: pink;
}

.circle {
  border-radius: 100%;
}
```

JS

```
document.querySelector(".square").style.backgroundColor = "lightgreen";
document.querySelector(".square").classList.add("circle")
document.querySelector(".circle").style.border = "2px solid blue";
```

https://codepen.io/jenie/pen/mmRLyq

Element - полета

- classList - дава достъп до класовете на елемента (могат да се променят с методите add, remove и toggle).

  Пример:

  ```
  document.querySelector("image").classList.toggle("visible")
  ;
  ```

- **innerHTML** - прочита/задава HTML съдържание на елемента

  Пример:

  ```
  document.querySelector("h1").innerHTML = "<em>Hello!</em>";
  ```

- **parentElement** - връща parent елемента (наследен от Node интерфейса)
- **textContent** - връща или задава текстово съдържание на текущия елемент (наследен от Node интерфейса)

Element - полета

- Специфични полета:
  - **value** - ако е инпут поле
  - **src** - ако е картинка
  - други (по-рядко се ползват): **style**, **id**, **tagName**, **className**, **attributes**
- Пример:

```
function setBorder(element) {
  element.style.border = "1px solid black";
}
document.querySelectorAll("div").forEach(setBorder);
```

Element - методи

- **addEventListener()** - закача функция, към определено събитие, което може да се случи с елемента (например click или mouseover)
- **getAttribute(attrName)** - връща стойността на даден атрибут
- **setAttribute(attrName, attrValue)** - променя стойността на атрибут. Пример:

```
document.querySelector("#submit").setAttribute("disabled", true);
```

- **removeAttribute(attrName)** - премахва атрибут
- **hasAttribute(attrName)** - връща true или false

Упражнение

- Като използвате следния codepen: https://codepen.io/jenie/pen/pPRQMY
- Направете следното само чрез JavaScript:

1. Задайте различни цветове за background на всеки един от трите div-а (например: lightgreen, lightyellow и coral)
2. Поставете форматиран текст във втория div (такъв, който съдържа html форматиране)
3. Разменете текстовете на първия и последния div

Още методи на елемента

- Всички селектори, които изпълняваме на document обекта, могат да се изпълнят и на Element обектите.
- Разликата е, че когато ги изпълняваме на елемента, търсенето става само в неговите наследници (елементите, които се съдържат в него)
- Пример:

```
var tabs = document.querySelectorAll(".tab");
tabs.forEach(function(tab) {
  var tabImg = tab.querySelector("img");
  tabImg.style.maxWidth = "100%";
});
```

Методи на елемента, наследени от Node

- appendChild() - закача нов HTML елемент (създаден чрез document.createElement() метода) след последния child елемент
- removeChild(childElement) - изтрива дадения childElement от децата на текущия елемент
- cloneNode() - връща копие на текущия елемент
- Пример:

```
var list = document.querySelector("ul");
var lastLi = list.querySelector("li:last-child");
var newLi = lastLi.cloneNode(true);
list.appendChild(newLi);
```

Form селектори

- Когато имаме именуван уеб формуляр, можем да го селектираме в JavaScript директно през името му
- html:
  ```
  <form name="myForm">
    ...
  </form>
  ```
- javascript:
  `document.myForm`
- Същото важи и за полетата на формуляра, които се селектват по име на формуляр, последван от име на поле:
  `document.myForm.email`

Въпроси?

Browser Object Model

window

- При зареждането на страницата, браузъра създава *най-глобалният* обект: **window**
- **window** представлява глобалния контекст на приложението и съдържа в себе си всички останали глобални функции и променливи (т.е. всеки път, когато създадем глобална променлива или функция, тя се закача за window)
  - Там например е глобалният обект **document**: window.document
  - Или дори конзолата: window.console
- Начина, по който **window** обекта е изграден, се нарича **BOM**

window.location

- Съдържа информация за URL-а на документа
- може да refresh-ва страницата:

```
window.location.reload();
```

- Може да зарежда друг сайт в същия прозорец:

```
window.location.replace("https://
developer.mozilla.org/en-US/docs/Web/API/
Window/location")
```

- може да чете GET параметрите в url-а

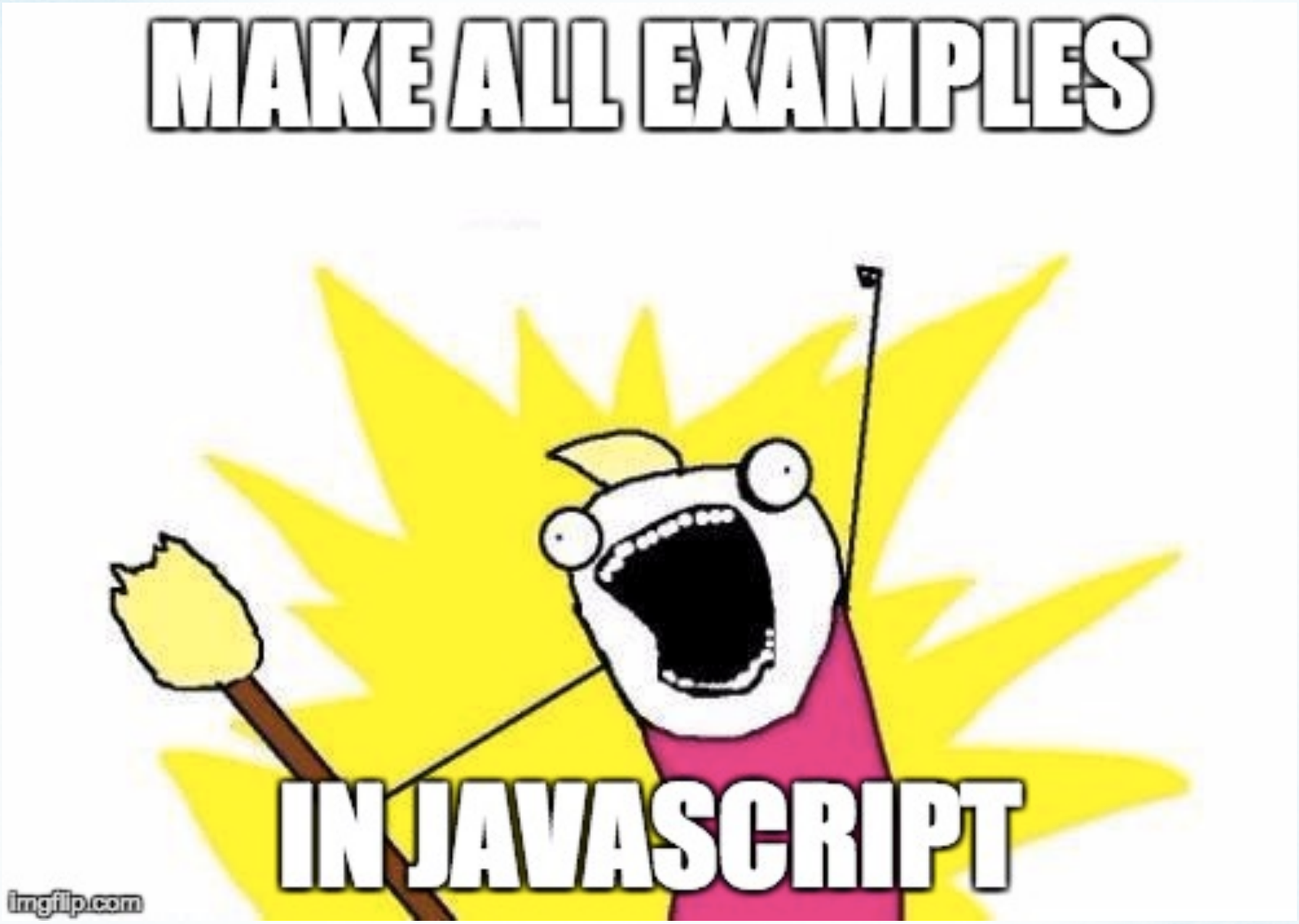
setTimeout() и setInterval()

- setTimeout(callback, timeout); // използваме го, за да отложим изпълнението на callback функцията
- setInterval(callback, timeout); // използваме го, за да накараме callback функцията да се изпълнява през определен интервал от време
- Example:

```
var intervalID = setInterval(alert, 1000); // Will alert every second.

clearInterval(intervalID); // Will clear the timer.

setTimeout(alert, 1000); // Will alert once, after a second.
```


MAKE ALL EXAMPLES
IN JAVASCRIPT
imgflip.com

Упражнение 2

- Като използвате следния codepen: https://codepen.io/jenie/pen/ZKLVBE
- Направете така, че:

1. 30 секунди след зареждането на страницата, да се покаже контейнера с ID showMe
2. След като се покаже първата картинка, на всеки 5 секунди да се показва и скрива втората картинка
3. В последния контейнер да се покаже домейна (host) на страницата

Примери в codepen

- https://codepen.io/jenie/pen/mmRLyq?editors=1010

- https://codepen.io/jenie/pen/VbPNzM?editors=0011

- https://codepen.io/jenie/pen/eWgwzL

KEEP CALM AND LEARN JAVASCRIPT

Примери

http://swift-academy.zenlabs.pro/lessons/lesson18/examples/download.zip

Домашно

http://swift-academy.zenlabs.pro/lessons/lesson18/homework