JAVASCRIPT

TIMEOUTS. DOM. EVENT LOOP. XHR.

СЪДЪРЖАНИЕ

TIMEOUTS

SETTIMEOUT И CLEARTIMEOUT

- Изпълнява дадена функция след определено време
- Времето за изпълнение на функцията не е гарантирано, а минимално

SETTIMEOUT

```
setTimeout(function() {
  // do something
}, 1000);
```

ПРИМЕРИ

```
function test() {
        console.log("Hello");
}
setTimeout(test, 1000);

console.log("World");
```

```
function test() {
        console.log("Hello");

        setTimeout(test, 1000);
}

setTimeout(test, 1000);
```

CLEARTIMEOUT

```
function test() {
        console.log("Hello");
}
let timerId = setTimeout(test, 1000);
clearTimeout(timerId);

console.log("World");
```

SETINTERVAL И CLEARINTERVAL

- Повтаря изпълнението на дадена функция на определен интервал от време

SETINTERVAL

```
function test() {
        console.log("Hello");
}

setInterval(test, 1000);
```

CLEARINTERVAL

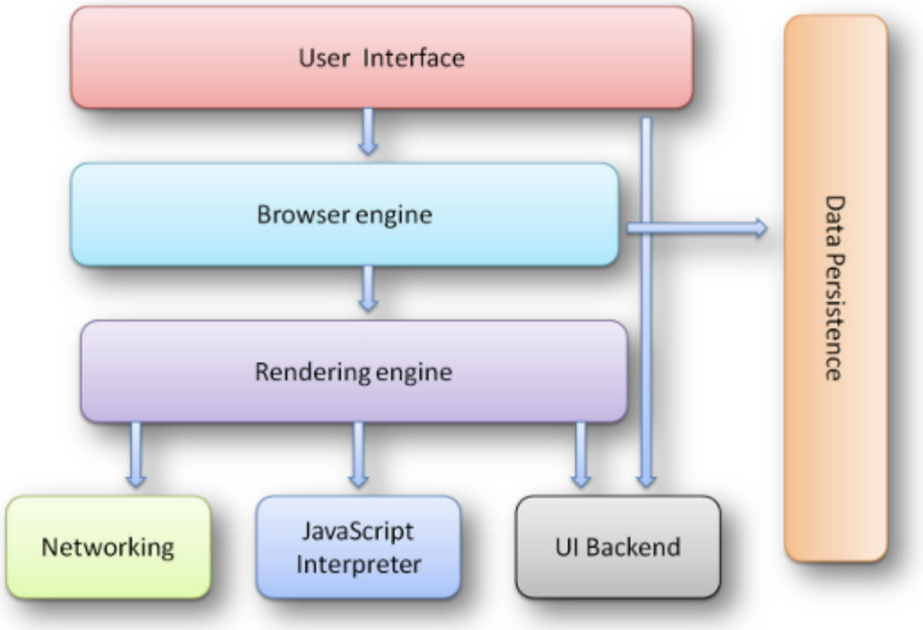
```
function test() {
        console.log("Hello");
}

let timerId = setInterval(test, 1000);
clearInterval(timerId);
```

SETINTERVAL VS SETTIMEOUT

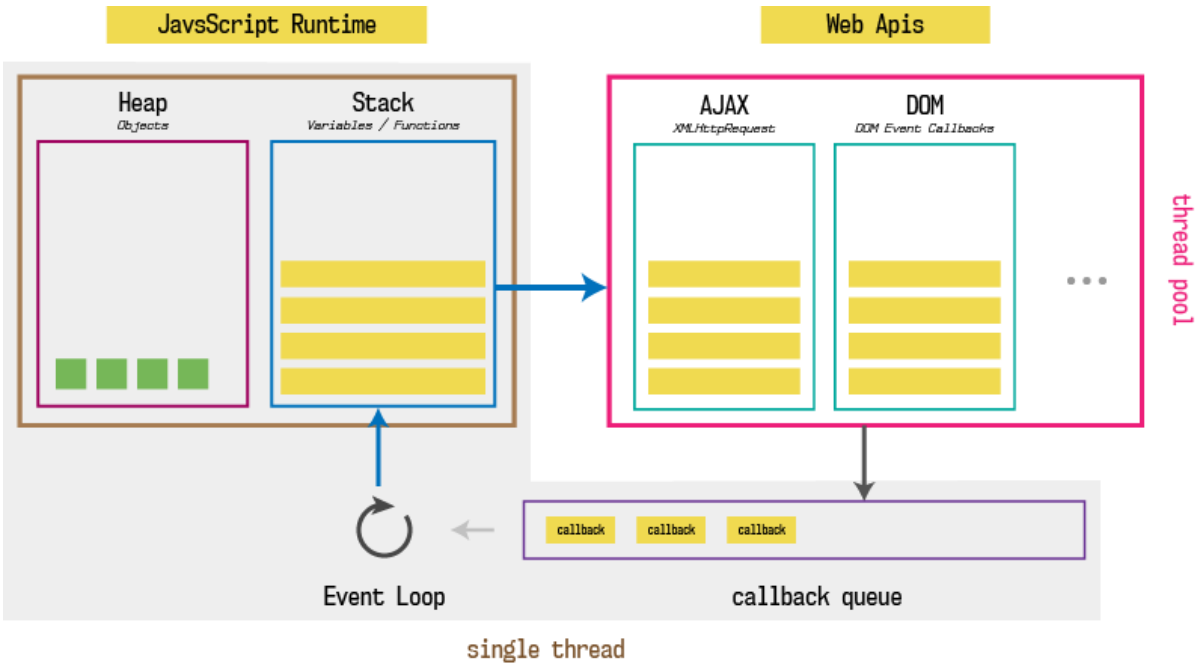
- **Understanding timers: setTimeout and setInterval**
- **Пример за setTimeout**
- **Пример за setInterval**

EVENT LOOP

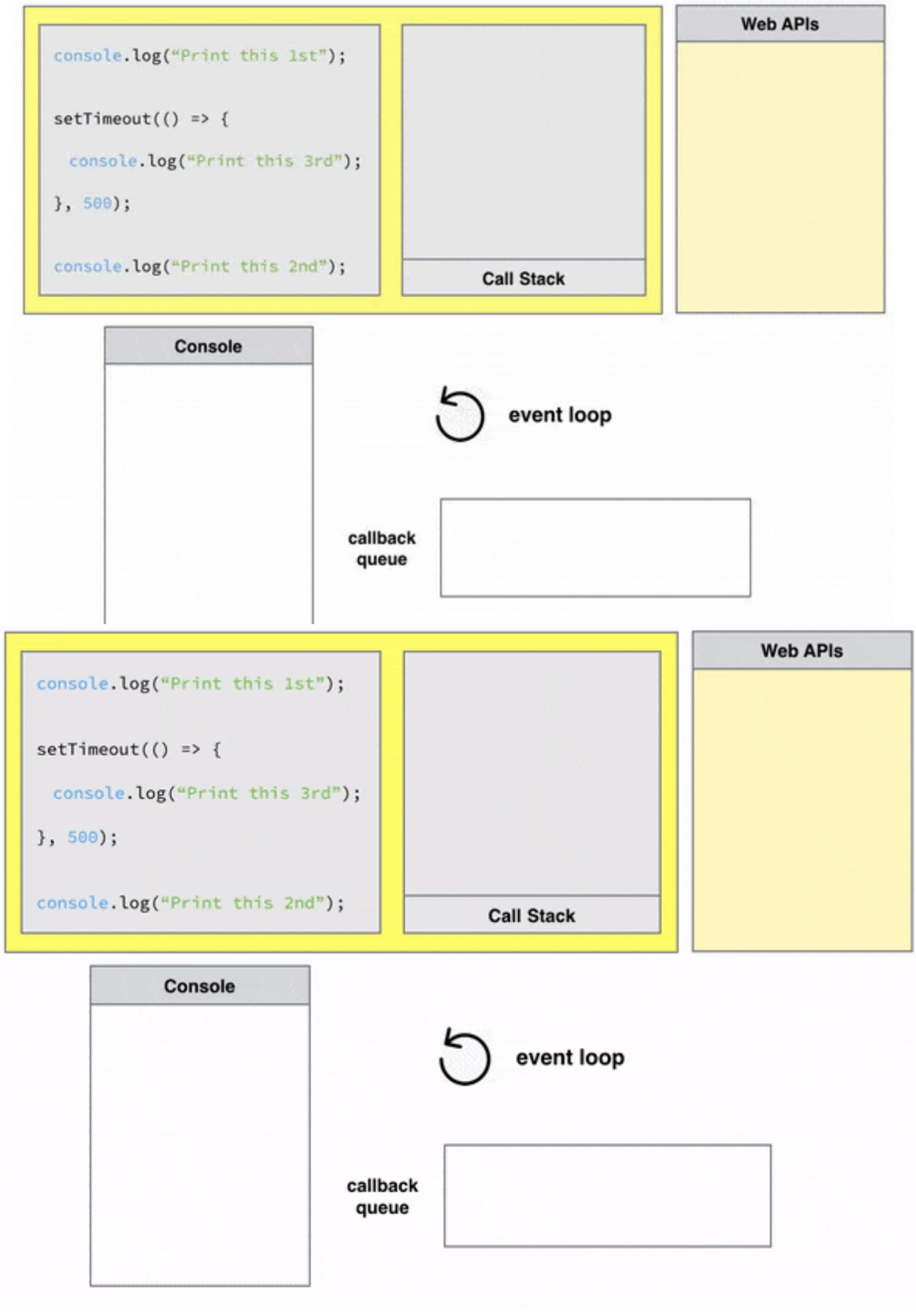
Архитектура на браузъра:



EVENT LOOP


JavsScript Runtime
Web Apis
Heap
Objects
Stack
Variables / Functions
AJAX
XMLHttpRequest
DOM
DOM Event Callbacks
thread pool
callback
callback
callback
Event Loop
callback queue
single thread

EVENT LOOP


console.log("Print this 1st");
setTimeout(() => {
console.log("Print this 3rd");
}, 500);
console.log("Print this 2nd");
Call Stack
Web APIs
Console
event loop
callback queue
console.log("Print this 1st");
setTimeout(() => {
console.log("Print this 3rd");
}, 500);
console.log("Print this 2nd");
Call Stack
Web APIs
Console
event loop
callback queue

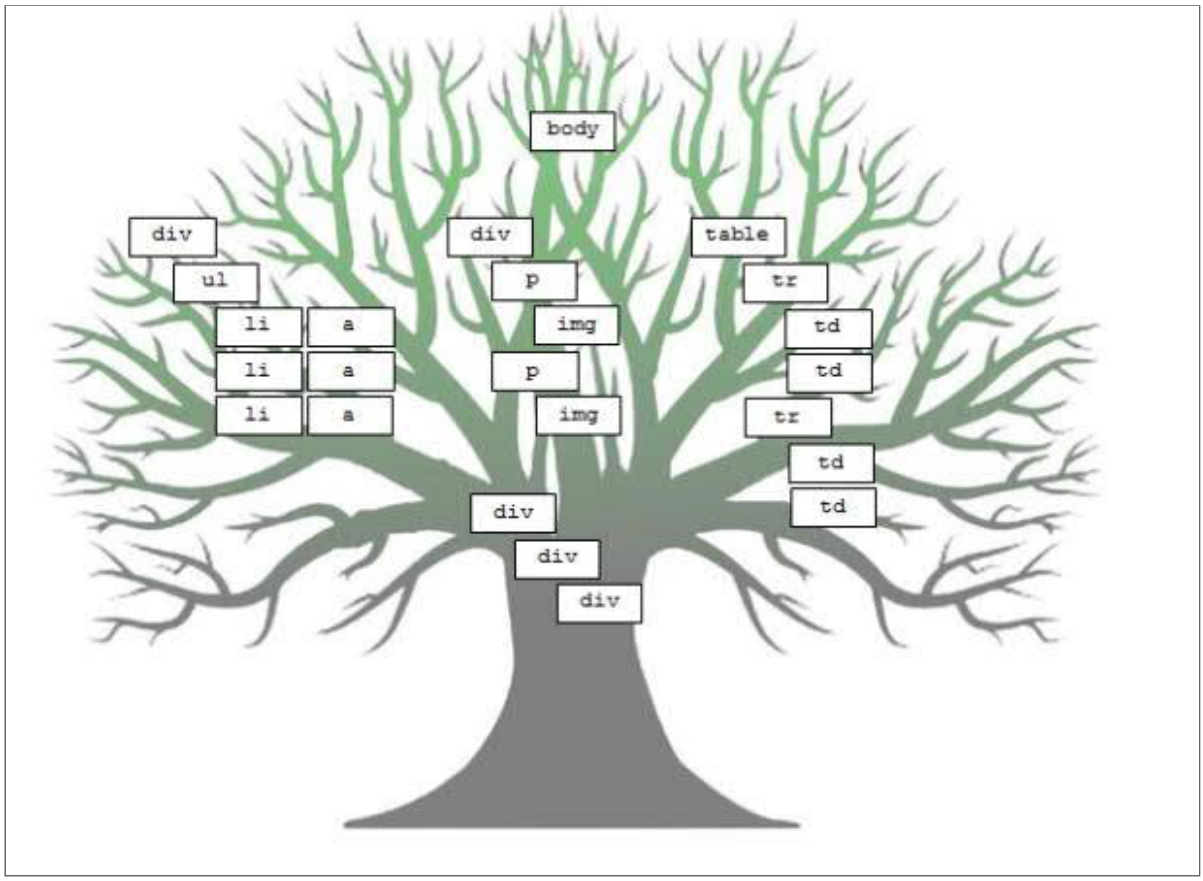
EVENT LOOP ПРИМЕРИ

- **setTimeout 1**
- **setTimeout 2**

ПОЛЕЗНИ ВРЪЗКИ

- **What the heck is the event loop anyway?**

DOM МАНИПУЛАЦИИ



КАКВО Е DOM?

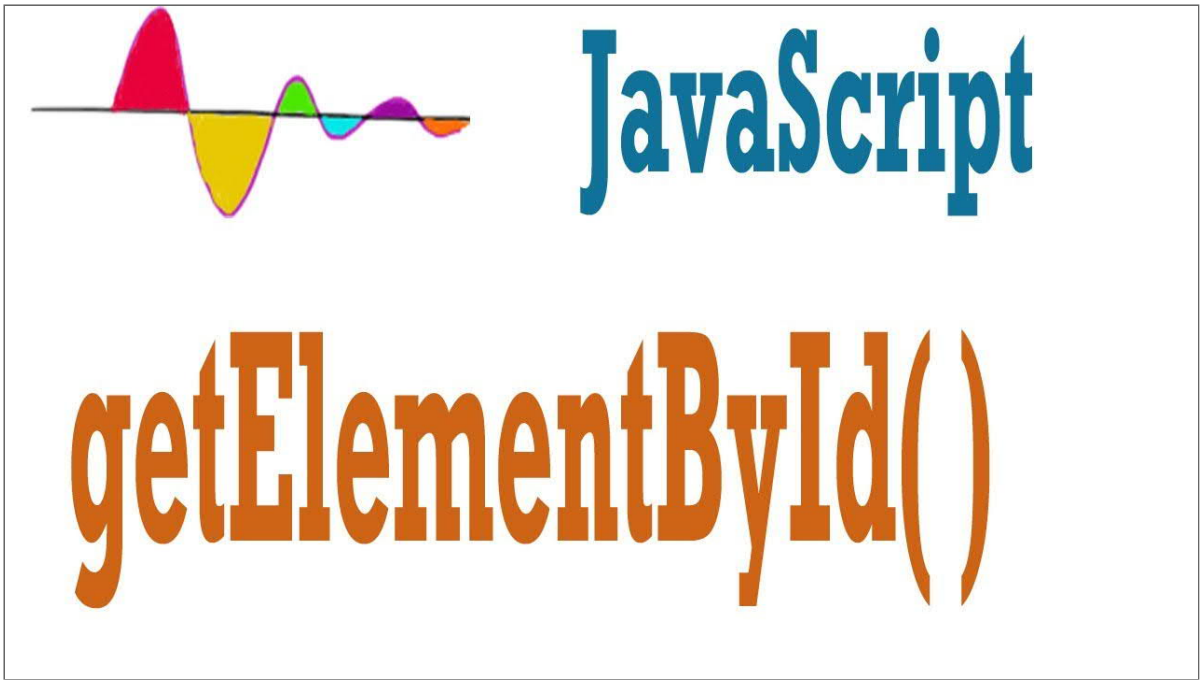
Document Object Model

Структурира се като дърво, което представя какво има на HTML/XML/SVG документ

HTML ЕЛЕМЕНТ

- div, p, a, head, button, други?
- Тяхното поведение се дефинира чрез JavaScript

ДОСТЪПВАНЕ НА DOM API ПРЕЗ JAVASCRIPT



DOM СЕЛЕКТОРИ

- document.getElementById()
- document.getElementsByTagName()
- document.getElementsByClassName()
- document.querySelector()
- document.querySelectorAll()

ПРИМЕР

```
<div id="my-div">
        <div id="inner-div">
                <ul>
                        <li class="first-li">1.1</li>
                        <li class="first-li">1.2</li>
                        <li class="second-li">2.1</li>
                        <li class="second-li">2.2</li>
                </ul>
        </div>
</div>
```

```
document.getElementById("my-div"); // връща първия див
document.getElementsByClassName("second-li"); // предположете? :)"
document.getElementsByTagName("li") // всички li
document.querySelector("li") // първото li в DOM
```

СЪЗДАВАНЕ И ДОБАВЯНЕ НА DOM ЕЛЕМЕНТИ В СТРАНИЦА

СЪЗДАВАНЕ НА DOM ЕЛЕМЕНТИ

- document.createElement(tagName)
-
```
var myDiv = document.createElement("div");
console.log(myDiv);
```

ДОБАВЯНЕ НА DOM В СТРАНИЦА

- element.appendChild()
- element.removeChild()

```
var myDiv = document.createElement("div");
document.body.appendChild(myDiv);
```

МАНИПУЛИРАНЕ НА HTML ЕЛЕМЕНТИ

- element.style.[___propety_name___]
- element.hasAttribute()
- element.toString()
- element.innerHTML
- element.textContent
- element.children
- И още много други..

ДОБАВЯНЕ НА СЪБИТИЯ

- click
- mousedown
- mouseup
- addEventListener
- и други

ПРИМЕР

```
var myButton = document.createElement("Button");
myButton.addEventListener("click", function() {
        alert("Hi there!");
});
```

AJAX & XHR

КАКВО Е AJAX

- AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.
- AJAX is not a programming language.
- AJAX just uses a combination of:
    - A browser built-in XMLHttpRequest object (to request data from a web server)
    - JavaScript and HTML DOM (to display or use the data)

AJAX@W3CSchool

КАКВО Е XHR (XMLHTTPREQUEST)

- Обект за комуникация със сървър
- Изпълнява заявка до сървъра, без пълно презареждане на страницата
- Работи и с други формати освен с XML, например JSON
- Използва се при AJAX комуникация
- Операциите с него са (по подразбиране) асинхронни

XHR СИНТАКСИС

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
                // do whatever you need with the response
        }
};
xhttp.open("GET", "server-url", true);
xhttp.send();
```

ОСНОВНИ МЕТОДИ И СВОЙСТВА НА XHR (1/2)

- **open** - инициализира заявката
- **send** - изпраща заявката

ОСНОВНИ МЕТОДИ И СВОЙСТВА НА XHR (2/2)

- **addEventListener("load", callbackFunction)** - изпълнява функция когато отговорът от сървъра е готов
- **addEventListener("error", callbackFunction)** - изпълнява функция когато има грешка в отговора от сървъра или изобщо няма отговор

```
var xhttp = new XMLHttpRequest();
xhttp.addEventListener("success", function() {
      // do whatever you need with the response
});

xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

- Всички методи и свойства на обекта може да прочетете **тук**

CORS

- Cross-Origin Resource Sharing
- Кои ресурси са разрешени и кои не (по подразбиране)?
- От какво ни предпазва?

FETCH API

- Подобно на XHR, но много по-лесно за използване и с повече удобни опции
- Не се поддържа в IE11, но има polyfill

СИНТАКСИС

```
fetch("__url_to_fetch__").then(function(response) {
        console.log(response);
});

fetch("__url_to_fetch__", options);
```

ПРИМЕР

```
let url = "https://maps.googleapis.com/maps/api/geocode/json?addres

fetch(url).then(function (response) {
        if (response.ok) {
                response.json().then(function(jsonData) {
                        console.log(jsonData);
                });
        }
});
```

OPTIONS ОБЕКТ

```
let myInit = { method: 'GET',
               mode: 'cors',
               cache: 'default' };

fetch('/flowers.jpg', myInit).then(function(response) {
        return response.blob();
}).then(function(myBlob) {
        let objectURL = URL.createObjectURL(myBlob);
        myImage.src = objectURL;
});
```

- method: GET, POST, PUT, DELETE, etc.
- mode: no-cors, cors, same-origin
- headers: обект съдържащ HTTP headers
- body: тялото на POST request
- и други...

ANY QUESTIONS?

name: "Vladislav Iliev"
email: "**vladislav.iliev@sap.com**

name: "Hristo Petrov"
email: "**h.petrov@sap.com**