**Vanilla Frameworks (Vanilla JS)**

ДОКУМЕНТАЦИЯ

Изготвили: Симеон Мечков ФН 1601681071

Красимира Пискачева ФН 1601681051

Съдържание:

1. Кратко описание
2. Как се вгражда?
3. Технологии и принципи
4. Компоненти
5. Примери

**1. Кратко описание**

JavaScript (Vanilla.JS) е олекотен, интерпретиран или своевременно компилиран език за програмиране с първокласни функции. Въпреки че е най-известен като скриптов език за уеб страници, много среди, които не са в браузър, също го използват, като Node.js, Apache CouchDB и Adobe Acrobat. JavaScript е базиран на прототип, мулти-парадигма, еднопоточен, динамичен език, поддържащ обектно-ориентирани, императивни и декларативни (напр. Функционално програмиране) стилове. Понятието *VANILLA* се използва често като описание на нещо опростено, обикновено, изчистено, стандартно.

Стандартът за JavaScript е ECMAScript. От 2012 г. всички съвременни браузъри напълно поддържат ECMAScript 5.1. По-старите браузъри поддържат най-малко ECMAScript 3. На 17 юни 2015 г. ECMA International публикува шестата основна версия на ECMAScript, която официално се нарича ECMAScript 2015 и първоначално се нарича ECMAScript 6 или ES6. Оттогава стандартите на ECMAScript са на цикли на годишно пускане.

**2.Как се вгражда / инсталира?**

Vanilla JS е изключително лесен и удобен за използване тъй като е толкова известен и разпространен, че се поддържа от повечето стандартни браузъри. Поради тази причина няма нужда да се инсталира. Той се интерпретира от браузърите по подразбиране.

За да вградим Vanilla JS в нашия проект трябва да добавим следния код в html частта.

<script src="path/to/vanilla.js"></script>

**3.Технологии и принципи**

В JavaScript могат да се приложат S.O.L.I.D принципите:

* ***Single Responsibility Principle****:* Всичко трябва да има само една причина да се промени. Това ще помогне на разработчиците да разберат контекста и отговорността на това, което изграждат и когато има нужда от промяна.
* ***Open-Closed Principle***: Промяна в поведението трябва да е възможна без промяна на съществуващ код, напр. чрез използване на разширени точки и създаване на код, който може да бъде включен.
* **Liskov substitution principle**: Принципът определя, че обектите от свръхклас трябва да бъдат заменими с обекти от неговите подкласове, без да се нарушава приложението. Това изисква обектите от подкласовете ви да се държат по същия начин като обектите от вашия суперклас.
* ***Interface Segregation Principle:*** Клиентът не трябва да бъде принуждаван да зависи от интерфейсите, които не използва. Проблем е, че в JS няма явни интерфейси, но има начини за това.
* ***Dependency Inversion Principle:*** Състои се от две концепции, абстракция, която гласи, че трябва да зависим от абстракциите, а не от конкретните реализации и собствеността, която гласи, че прилагането на ниско ниво трябва да зависи от концепциите на високо ниво.

4.Компоненти

1. Стандартни обекти:

* Масиви (Array)
  + Масивите са обекти, подобни на списък, чийто прототип има методи за извършване на операции на преминаване и мутация.
  + Масивите не могат да използват низове като индекси на елементи (като в асоциативен масив), но трябва да използват цели числа
* Boolean
  + Boolean приема стойности true и false.
* Date
  + Обектите за дата на JavaScript представляват един момент във формат, независим от платформата.
  + Има 4 начина за създаване на дата чрез JavaScript:

## New Date() - създава дата обект с текущи дата и време.

* + new Date(year, month, day, hours, minutes, seconds, milliseconds) - създава обект с уточнени дата и време.

## new Date(*dateString*) - позволява да добавим string елемент. Пример: var d = new Date("February 25, 2020 10:20:00");

## new Date(*milliseconds*)

* Error
  + Обектите за грешка се показват, когато възникнат грешки по време на изпълнение. Обектът Грешка може да се използва и като основен обект за дефинирани от потребителя изключения.
* Function
  + Функцията е процедура на JavaScript - набор от оператори, който изпълнява задача или изчислява стойност. За да използвате функция, трябва да я дефинирате някъде в обхвата, от който искате да я извикате.
* JSON = JavaScript Object Notation
  + JSON е формат за съхранение и транспортиране на данни.
  + често се използва, когато данните се изпращат от сървър до уеб страница.
* Math
  + Math е вграден обект, който има свойства и методи за математически константи и функции. Не е функционален обект.
* Number
  + Това е обвиващ обект, който позволява работата с числови стойности. Обект номер се създава с помощта на конструктора Number ().
* Object
  + Обект, е референтен тип данни. Променливите, на които е присвоена референтна стойност, получават референция или указател към тази стойност.
  + Тази справка или указател сочи местоположението в паметта, където се съхранява обекта.
  + Пример за създаване на обект: var person = {firstName:"Jane", lastName:"Doe", age:23, eyeColor:"green", hairColor:”blonde”};
* RegExp
  + Използва се за свързване на текст с шаблон
* String
  + Глобалният обект String е конструктор за низове или последователност от знаци.
  + Задаваме string параметри като ги поставим в “ ”.
* Map
  + Обектът Map съдържа двойки ключ-стойност и запомня оригиналния ред на вмъкване на ключовете. Всяка стойност (както обекти, така и примитивни стойности) може да се използва като ключ или като стойност.
* Set
  + Този обект позволява да се съхраняват уникални стойности от всякакъв тип. Не е от значение дали тези стойности са примитивни или са препратки към обекти.
  + Set обектите са колекции от стойности.
* WeakMap
  + Този обект представлява съвкупност от двойки ключ / стойност, в които ключовете са слабо препратени. Ключовете трябва да са обекти и стойностите могат да бъдат произволни стойности.
* WeakSet
  + WeakSet обектът представлява колекция от обекти. В WeakSet обектите могат да се появяват само един път. Те са уникални.

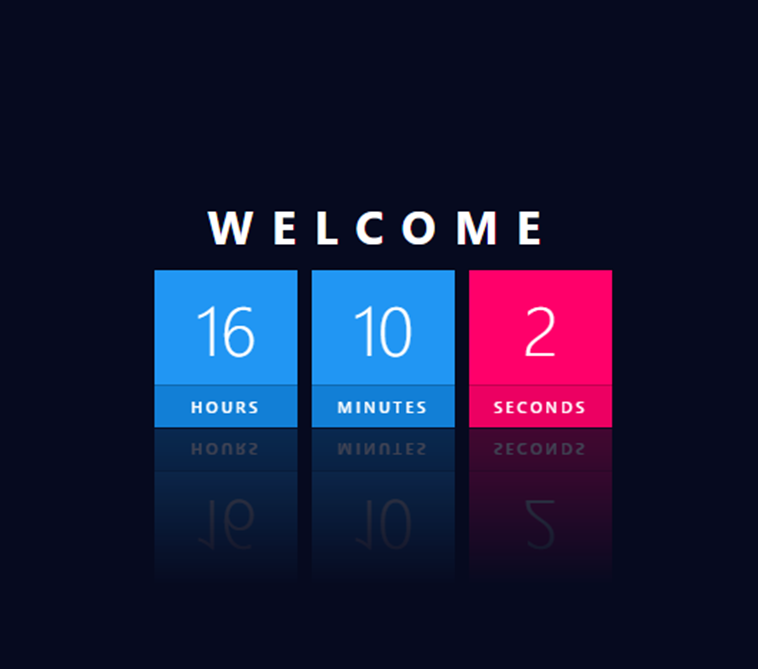
2.Оператори:

* instanceof
  + Този оператор проверява дали свойството прототип на конструктор се появява навсякъде във веригата на прототипа на обект.
* Typeof
  + Операторът typeof връща низ, указващ типа на неоценения операнд.
  + използва се за получаване на типа данни на своя операнд. Операндът може да бъде или буквален, или структура от данни, като променлива, функция или обект.
* New
  + позволява на разработчиците да създадат екземпляр от дефиниран от потребителя тип обект или от един от вградените типове обекти, който има функция конструктор.
* this

Условни оператори и декларации

* do-while, for-in, for-of, try-catch, let, var, const, if-else, switch

# 5. Примери

****

function clock() {

var hours = document.getElementById('hours');

var minutes = document.getElementById('minutes');

var seconds = document.getElementById('seconds');

var h = new Date().getHours();

var m = new Date().getMinutes();

var s = new Date().getSeconds();

hours.innerHTML = h;

minutes.innerHTML = m;

seconds.innerHTML = s;

}

var clockTickingInterval = setInterval(clock, 1000);



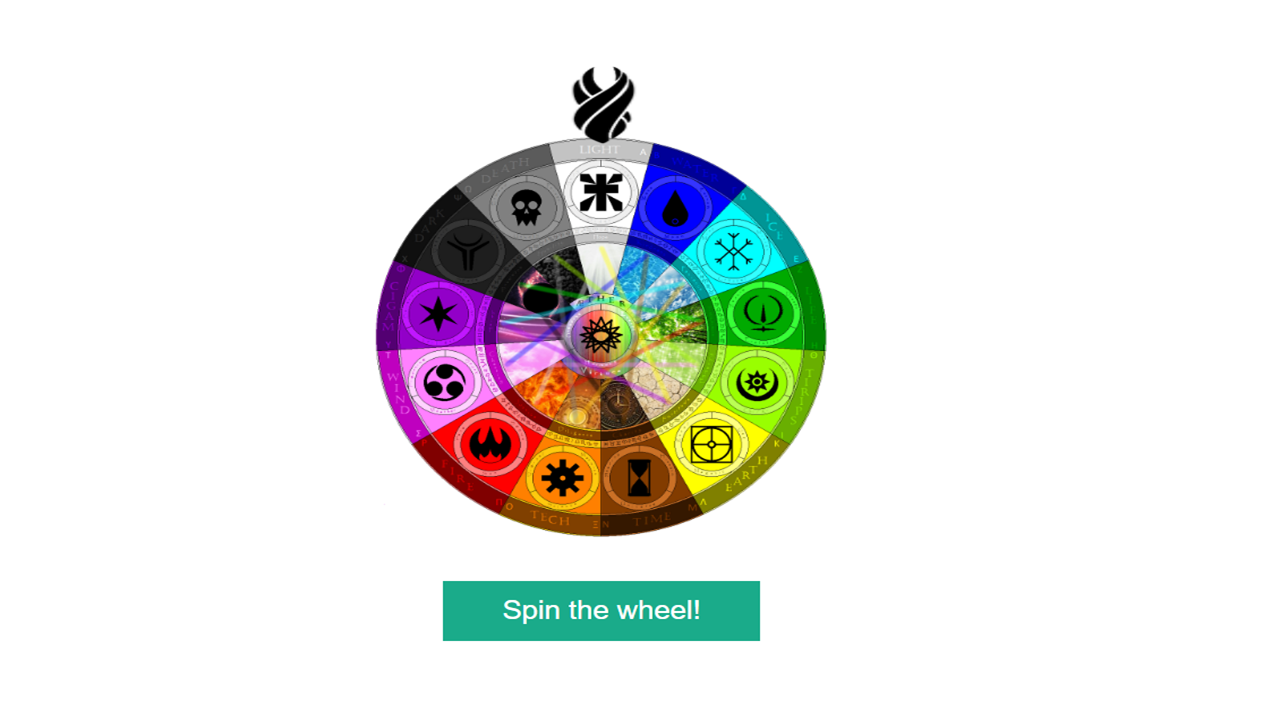
var slide = document.getElementById('slide2');

window.onmousemove = function (e) {

var x = e.clientX;

slide.style.left = x + 'px';

}



var wheel = document.getElementById('wheel');

var spinButton = document.getElementById('spin-wheel-button');

var deg = 0;

spinButton.addEventListener('click', function () {

spinButton.style.pointerEvents = 'none';

deg = Math.floor(5000 + Math.random() \* 5000);

wheel.style.transition = 'all 10s ease-out';

wheel.style.transform = `rotate(${deg}deg)`;

wheel.classList.add('blur');

});

wheel.addEventListener('transitionend', function () {

wheel.classList.remove('blur');

spinButton.style.pointerEvents = 'auto';

wheel.style.transition = 'none';

var actualDeg = deg % 360;

wheel.style.transform = `rotate(${actualDeg}deg)`;

});



var paralaxContainer = document.getElementById('layered-container');

function parallax(e) {

paralaxContainer.querySelectorAll('.layer').forEach(layer => {

var speed = layer.getAttribute('data-speed');

var x = (window.innerWidth - e.pageX \* speed) / 250;

var y = (window.innerWidth - e.pageX \* speed) / 250;

layer.style.transform = `translate(${-x}px) translateY(${y}px)`;

})

}

paralaxContainer.addEventListener('mousemove', parallax);

# Използвана литература

1. <https://www.infoq.com/news/2014/01/solid-principles-javascript/>
2. <http://vanilla-js.com/>
3. <https://developer.mozilla.org/en-US/docs/Web/JavaScript>