Topic: Date methods , callback, for each and map

Date Methods

Creating Dates

Var now=new Date();

Specific Date and Time

let specificDate = new Date('2024-06-12T10:20:30Z');

Getting Date Components

let year = now.getFullYear();// year

let month = now.getMonth(); //0 -11

let day = now.getDate(); // 1-31

Day of the Week

let dayOfWeek = now.getDay(); // 0-6 (0 = Sunday, 6 = Saturday)

Hours, Minutes, Seconds, Milliseconds

let hours = now.getHours(); // 0-23

let minutes = now.getMinutes(); // 0-59

let seconds = now.getSeconds(); // 0-59

let milliseconds = now.getMilliseconds(); // 0-999

Setting Date Components

now.setFullYear(2025);

now.setMonth(6); // July

now.setDate(15);

Set Hours, Minutes, Seconds, Milliseconds

now.setHours(15);

now.setMinutes(30);

now.setSeconds(45);

now.setMilliseconds(500);

Formatting Date and Time

JavaScript provides methods to format dates as strings in different formats:

toDateString()

let dateStr = now.toDateString(); // e.g., "Wed Sep 22 2024"

toTimeString()

let timeStr = now.toTimeString(); // e.g., "15:30:45 GMT+0530 (India Standard Time)"

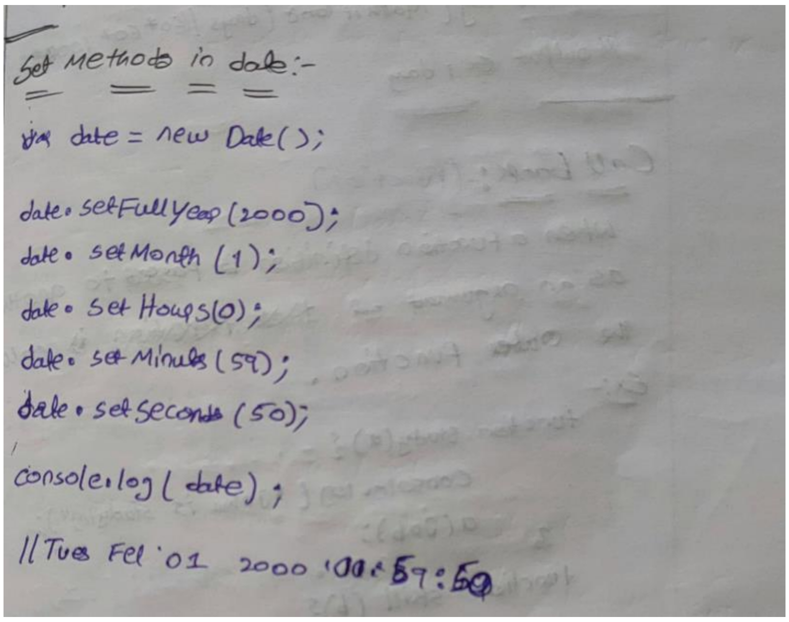
toLocaleDateString()

let localDateStr = now.toLocaleDateString(); // e.g., "9/22/2024" in US format

toLocaleTimeString()

let localTimeStr = now.toLocaleTimeString(); // e.g., "3:30:45 PM" in US format

These methods allow you to present dates in a human-readable or standardized format.



Set methods in date:-

```
var date = new Date();

date.setFullYear(2000);
date.setMonth(1);
date.setHours(0);
date.setMinutes(59);
date.setSeconds(50);

console.log(date);

//Tues Feb 01 2000 00:59:50
```

Age calculator

```
 <input type="date" id="el1" />
<button onclick="fun()">clicke me</button>

<script>
  function fun() {
    var d = document.getElementById("el1").value;

    var olddate = new Date(d);
    var newdate = new Date();

    var year = newdate.getFullYear() - olddate.getFullYear();

    var milliseconds = newdate - olddate;
    var days = Math.floor(milliseconds / (60 * 60 * 24 * 1000));
    console.log(days);
    console.log(year);
  }
</script>
```

Callbacks

callback is a function defintion is passed as an argument to another function and is executed after some operation has been completed. This is a powerful feature that allows for asynchronous programming, enabling tasks to run concurrently without blocking the main execution thread.

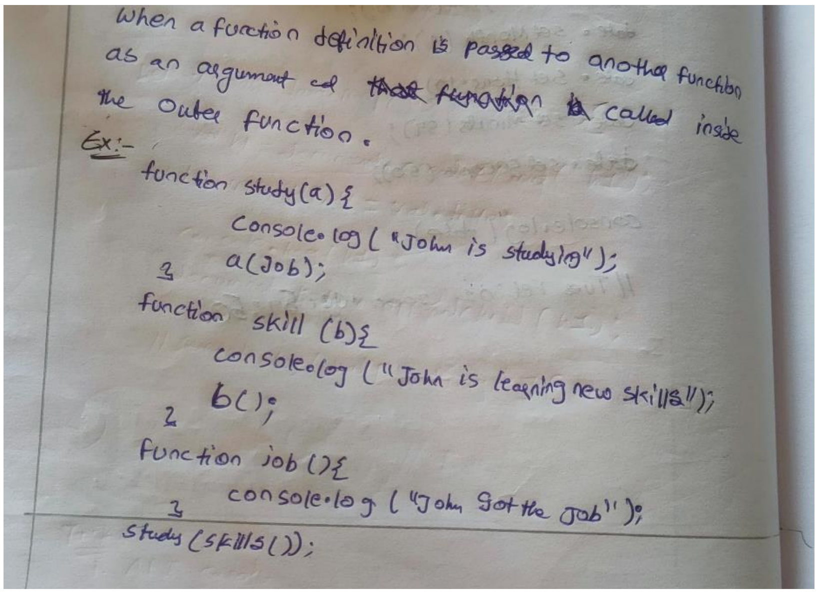
Defining the Callback:

```
// Step 1: Define the callback function
function myCallback() {
  console.log("Callback function executed!");
}
```

The callback function can be defined separately or inline.

```
// Defining separately
function myCallback() {
  console.log("Callback executed!");
}
doSomething(myCallback);

// Defining inline
doSomething(function () {
  console.log("Callback executed!");
});
```

Step 1: Define a Function that Takes Another Function as a Parameter(HOF)
Step 2: Define a function that takes another function as a parameter
Step 3: Execute the callback function
Step 4: Pass the callback function as an argument



When a function definition is passed to another function as an argument ad ~~that function is~~ called inside the Outer function.

Ex:-

```
function study(a){
        Console.log("John is studying");
        a(Job);
    }
function skill (b){
        console.log("John is learning new skills");
        b();
    }
Function job (){
        console.log("John Got the Job");
    }
study(skills());
```

Ex-1

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

function mySecond(a) {
  console.log("Goodbye");
}

var a=myFirst();
mySecond(a);
//Hello
//Goodbye
```

Ex-2

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

function mySecond() {
    myFirst()
  console.log("Goodbye");
}

mySecond();
```

The problem with the first example above, is that you have to call two functions to display the result. The problem with the second example, is that you cannot prevent the function from displaying the result when it invokes every time.

```
function myFirst() {
  console.log("Hello");
}
function mySecond(a) {
  console.log("Goodbye");
}

mySecond(myFirst());//can call two functions at a time
mySecond()//one function
```

Usage of Callbacks

Callbacks are commonly used in situations where you want to perform tasks asynchronously,
such as:
Event Handling
Array Methods
Higher-Order Functions
Asynchronous Operations

Array Iteration and Transformation Methods:

Array for each

Array.forEach() is a method in JavaScript used to iterate over elements in an array. It executes a provided function once for each array element

Syntax:

```
array.forEach(function(value, index, array) {
  // Your code here
});
```

value: The current item in a array.
index (optional): The index of the item in the array.
array (optional): The array that forEach()

```
const numbers = [1, 2, 3, 4, 5];

numbers.forEach(function(val, index) {
  console.log(`Element at index ${index} is ${val }`);
});
//output
// Element at index 0 is 1
// Element at index 1 is 2
// Element at index 2 is 3
// Element at index 3 is 4
// Element at index 4 is 5
```

You can also use arrow functions for a more concise syntax:

```
numbers.forEach((number, index) => {
  console.log(`Element at index ${index} is ${number}`);
});
```

One important thing to note about forEach() is that it doesn't return anything. It simply iterates over the array. If you need to transform the elements of the array and create a new array based on those transformations, you might want to use methods like map() instead.

```
array.forEach(function(element) {
  return element * 2; // This return statement has no effect
});
```

Map method:

The map() method in JavaScript is used to create a new array by calling a provided function on every element in the calling array. It doesn't change the original array; instead, it returns a new array with the results of applying the provided function to each element.

Syntax:

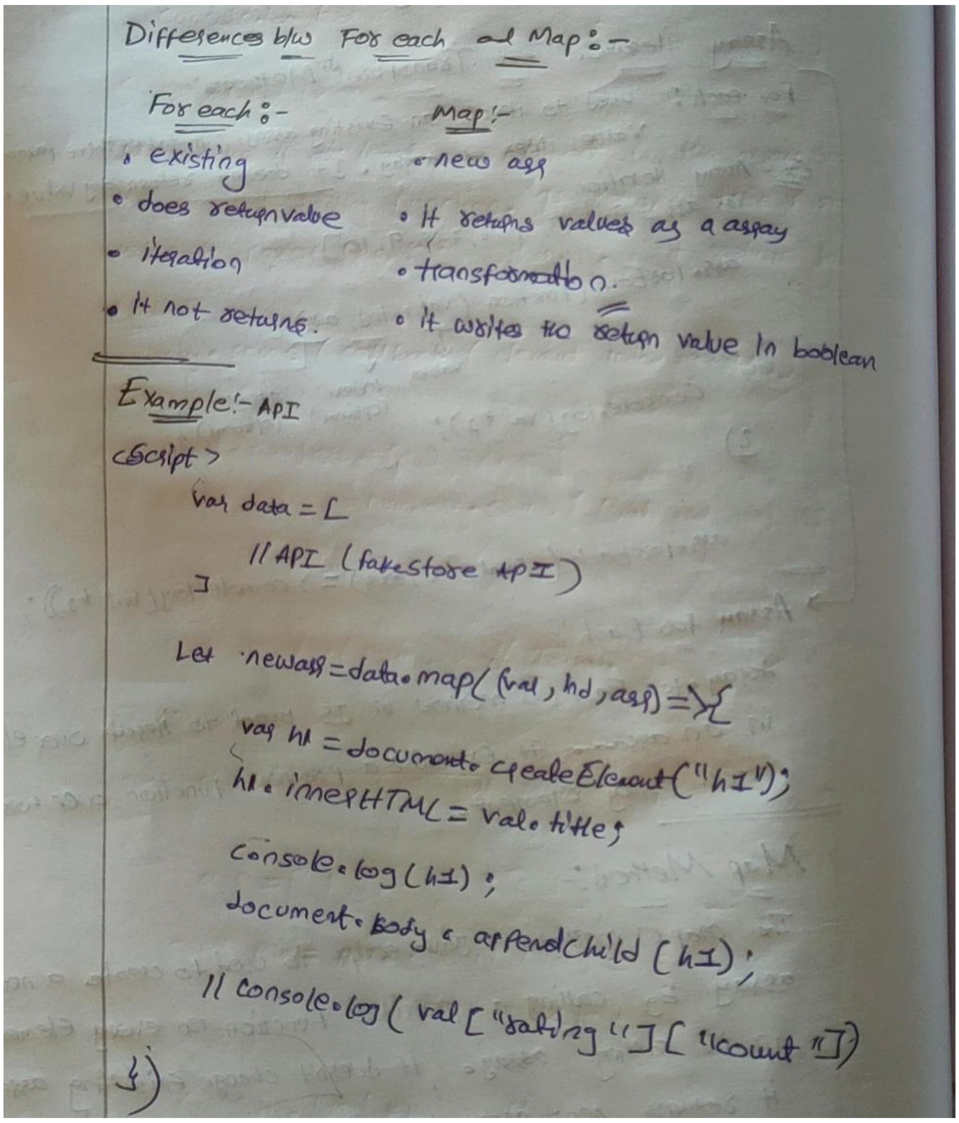
```
const newArray = array.map(function callback(currentValue, index, array) {
 // Return element for newArray
});
```

value: The current item in a array.
index (optional): The index of the item in the array.
array (optional): The array that forEach()

```
const numbers = [1, 2, 3, 4, 5];

const doubledNumbers = numbers.map(function(number) {
  return number * 2; // Return value determines the value
});
console.log(doubledNumbers); // Output: [2, 4, 6, 8, 10]
```

You can also use arrow functions for more concise syntax:

```
const numbers = [1, 2, 3, 4, 5];
const doubledNumbers = numbers.map(number => number * 2);
console.log(doubledNumbers); // Output: [2, 4, 6, 8, 10]
```



Differences b/w For each and Map:-

For each:-	Map:-
• existing	• new arr
• does return value	• It returns values as a array
• iteration	• transformation.
• It not returns.	• It writes to return value in boolean

Example:- API

```
<Script>
    var data = [
        // API (fakestore API)
    ]

    Let newarr = data.map((val, ind, arr) => {
        var h1 = document.createElement("h1");
        h1.innerHTML = val.title;
        console.log(h1);
        document.body.appendChild(h1);
      // console.log(val["rating"]["count"])
})
```

Filter:-

- iteration & transformation
- it returns values
- it returns values which are satisfying the condition.

Ex: arr = [1,2,3,4,5,6,7];

```
new_arr = arr.filter( (val, ind, arr) => {
        return val%2 ==0
})
(or)
new_arr = arr.filter(val => val%2 ==0)

console.log (new_arr);
```

Reduce(): used ~~for~~ to ~~various~~ reduce the elements of an array to single value.

Accumulator:- The accumulated value computed from previous iteration.

Ex:- sum of an Array.

```
arr = [1,2,3,4,5]

new_arr= arr.reduce ( (count, val, ind, arr) => {       → accumlator
   return val+count;                                    → accumlator
}, 0)                                                   → accumlator value
console.log (new_arr); // 15
```

Reduce Right Method:-

used to reduce the elements of an array to a single value but it process the array from right to left.

Sort method

- T~~o~~ remove lexographical

```
arr = [12, 1, 12, 10021, 63, 3456, 322];
arr.sort(a,b) =>{
        return a-b;
})
console.log (arr);
```

Some Method :-

It checks if at least one element in the array satisfies the provided testing function. It returns true if any element passes the test, Otherwise, it returns false.

Syntax:-

```
array.some ( callback ( element , index, array)).
```

Ex:-

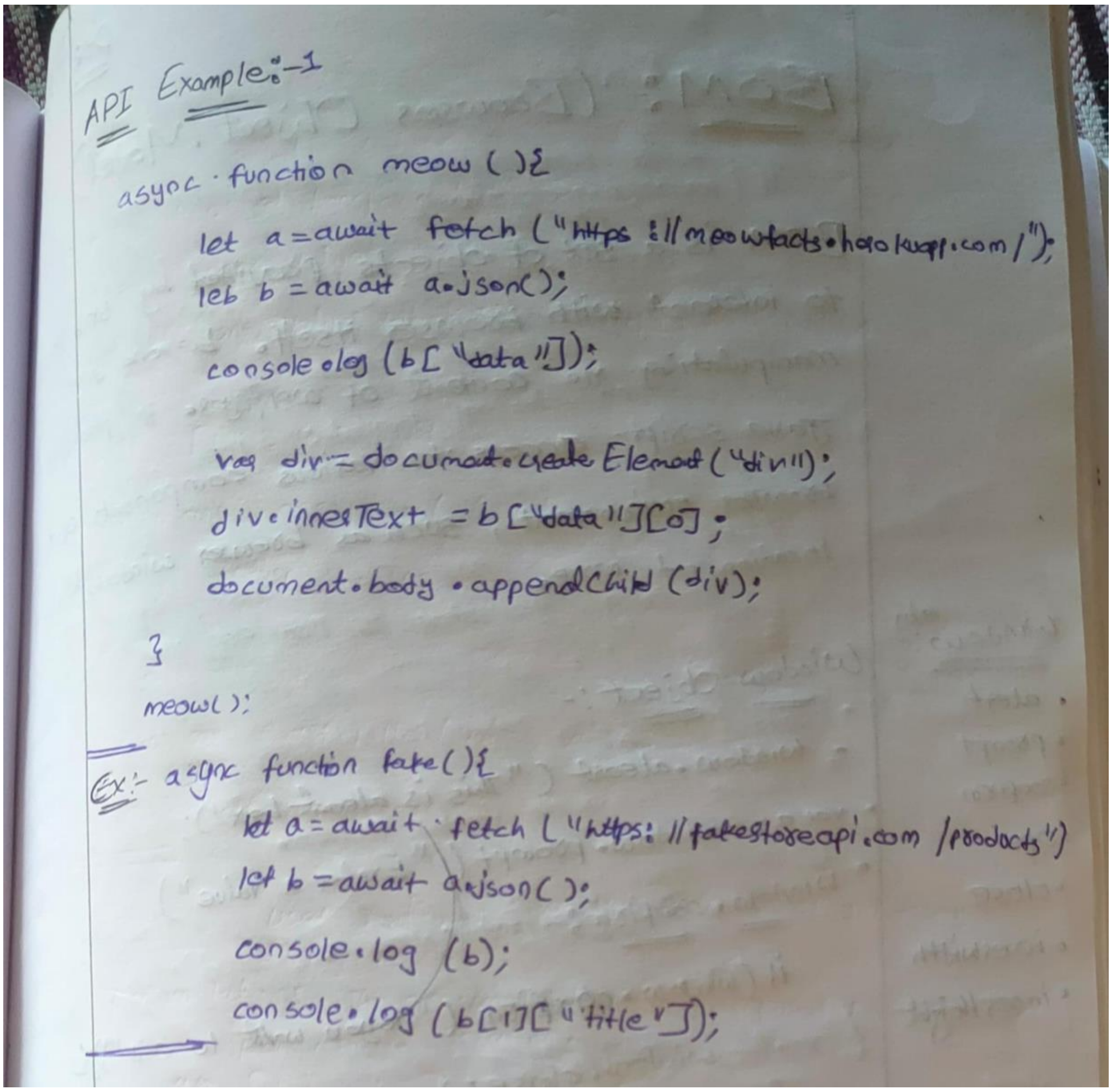
```
const numbers = [1,2,3,4,5]
const num2 = numbers.some ( (val ind arr) =>{
        return val =3;
})
console.log(num2) // True.
```

every Method :-

- it should statisfies all elements present in array.

Ex:-

```
arr = [1,2,3,4,5,6,7]
const num = arr.every( (val) => {
                    return arr<10;
})
console.log (num); // True
```



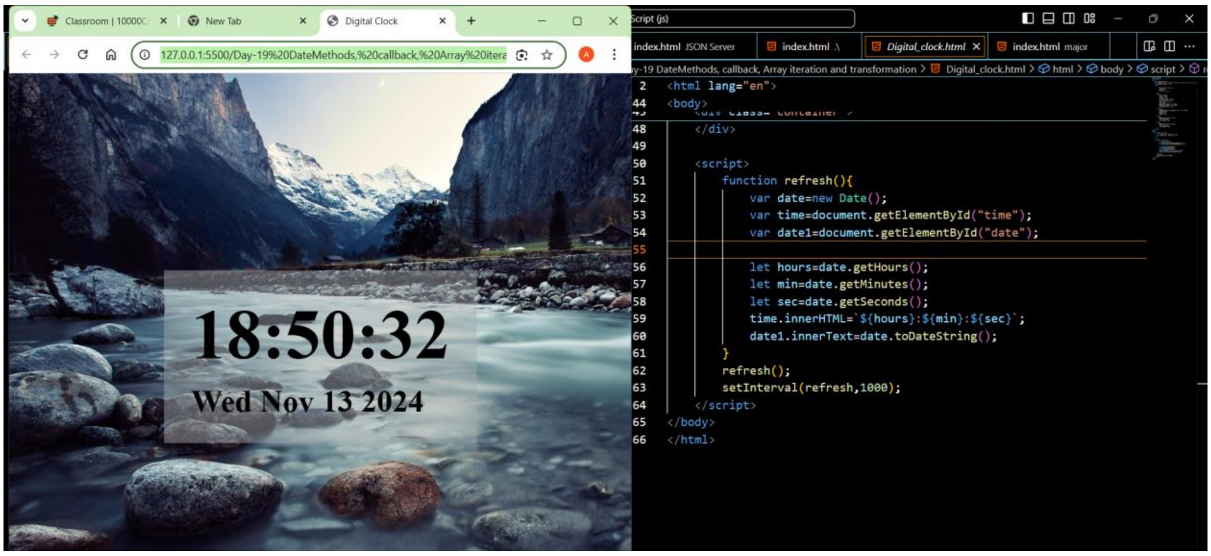
API Example:-1

```
async function meow (){
    let a=await fetch("https://meowfacts.herokuapp.com/");
    leb b =await a.json();
    console.log (b["data"]);

    var div=document.createElement("div");
    div.innerText =b["data"][0];
    document.body.appendChild (div);
}
meow();
```

Ex:-
```
async function fake(){
    let a=await fetch ("https://fakestoreapi.com/products")
    let b=await a.json();
    console.log (b);
    console.log (b[1]["title"]);
```

Task:

1. **Digital Clock**



18:50:32

Wed Nov 13 2024

```
<html lang="en">
<body>
    </div>

    <script>
        function refresh(){
            var date=new Date();
            var time=document.getElementById("time");
            var date1=document.getElementById("date");

            let hours=date.getHours();
            let min=date.getMinutes();
            let sec=date.getSeconds();
            time.innerHTML=`${hours}:${min}:${sec}`;
            date1.innerText=date.toDateString();
        }
        refresh();
        setInterval(refresh,1000);
    </script>
</body>
</html>
```

2. API Cards

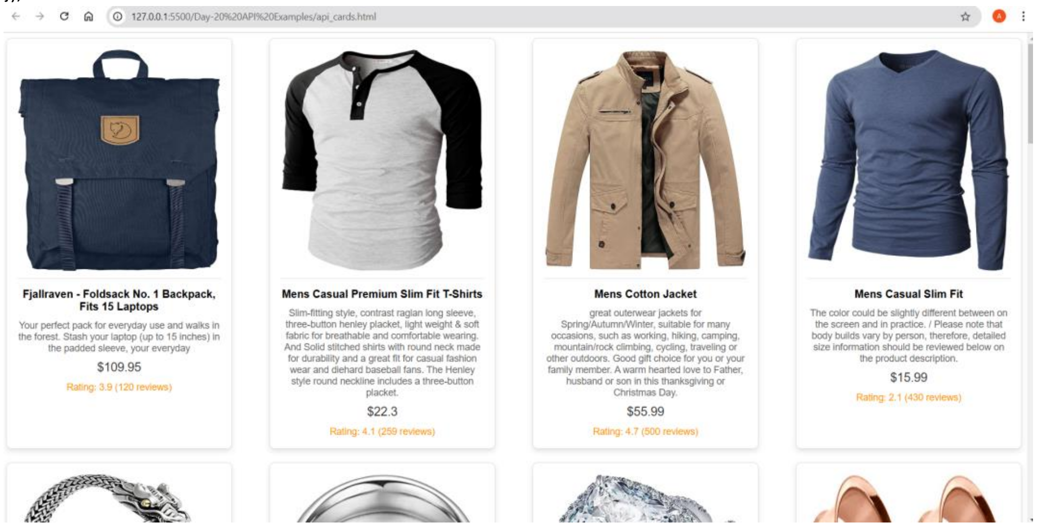
```
const products = [
    //Fake Store API Data
  ]
const container = document.getElementById('card-container');
```

// Function to create a card for each product

```
function createProductCard(product) {
   const card = document.createElement('div');
   card.className = 'card';

   const image = document.createElement('img');
   image.src = product.image;

   const title = document.createElement('div');
   title.className = 'title';
   title.textContent = product.title;

   const description = document.createElement('div');
   description.className = 'description';
   description.textContent = product.description;

   const price = document.createElement('div');
   price.className = 'price';
   price.textContent = `$${product.price}`;

   const rating = document.createElement('div');
   rating.className = 'rating';
   rating.textContent = `Rating: ${product.rating.rate} (${product.rating.count} reviews)`;

   card.appendChild(image);
   card.appendChild(title);
   card.appendChild(description);
   card.appendChild(price);
   card.appendChild(rating);

   return card;
}
```

// Loop through the products and create cards

```
products.forEach(product => {
   const productCard = createProductCard(product);
   container.appendChild(productCard);
});
```



Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops

Your perfect pack for everyday use and walks in the forest. Stash your laptop (up to 15 inches) in the padded sleeve, your everyday

$109.95

Rating: 3.9 (120 reviews)

Mens Casual Premium Slim Fit T-Shirts

Slim-fitting style, contrast raglan long sleeve, three-button henley placket, light weight & soft fabric for breathable and comfortable wearing. And Solid stitched shirts with round neck made for durability and a great fit for casual fashion wear and diehard baseball fans. The Henley style round neckline includes a three-button placket.

$22.3

Rating: 4.1 (259 reviews)

Mens Cotton Jacket

great outerwear jackets for Spring/Autumn/Winter, suitable for many occasions, such as working, hiking, camping, mountain/rock climbing, cycling, traveling or other outdoors. Good gift choice for you or your family member. A warm hearted love to Father, husband or son in this thanksgiving or Christmas Day.

$55.99

Rating: 4.7 (500 reviews)

Mens Casual Slim Fit

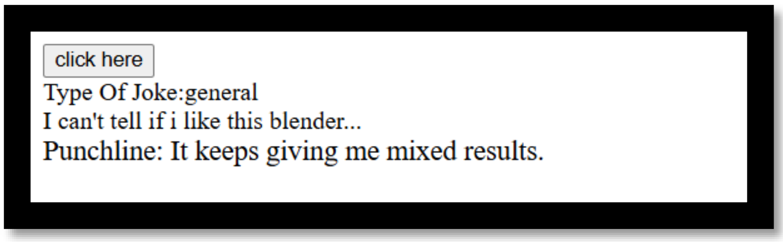
The color could be slightly different between on the screen and in practice. / Please note that body builds vary by person, therefore, detailed size information should be reviewed below on the product description.

$15.99

Rating: 2.1 (430 reviews)

3. Genrate Jokes Using API:

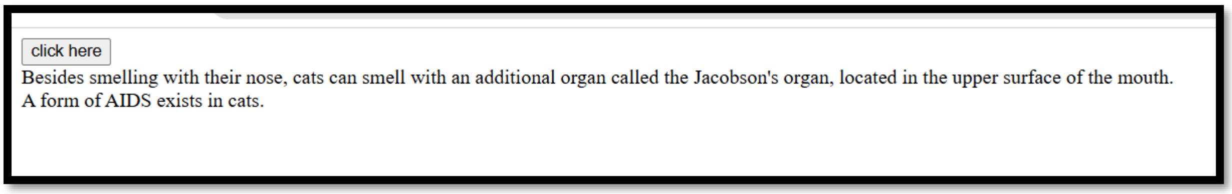
```
<button onclick="jokes()">click here</button>
  <div class="container"></div>

  <script>
    async function jokes(){
      let a=await fetch("https://official-joke-api.appspot.com/random_joke");
      let b=await a.json();

      // console.log(b);
      // console.log(b["setup"]);

      var span=document.createElement("span");
      span.innerHTML="Type Of Joke:";
      var span1=document.createElement("span");
      span1.innerHTML=b["type"];
      span1.style.paddingBottom="15px";
      var div1=document.createElement("div");
      div1.innerHTML=b["setup"];

      var div2=document.createElement("div");
      div2.innerHTML="Punchline: "+b["punchline"];
      div2.style.fontSize="18px"

      var div=document.getElementsByClassName("container")[0];
      span.appendChild(span1);
      div.appendChild(span);
      div.appendChild(div1);
      div.appendChild(div2);

      // document.body.appendChild(div);
    }
    // jokes();
  </script>
```



```
Type Of Joke:general
I can't tell if i like this blender...
Punchline: It keeps giving me mixed results.
```

4. Genrate Meow facts using API

```
  <button onclick="me()">click here</button>
  <script>
    async function me(){
      let a=await fetch("https://meowfacts.herokuapp.com/");
      let b=await a.json();

      console.log(b["data"][0]);

      var div=document.createElement("div");
      div.innerHTML=b["data"][0];
      document.body.appendChild(div);
    }
    // me();
  </script>
```



Besides smelling with their nose, cats can smell with an additional organ called the Jacobson's organ, located in the upper surface of the mouth.
A form of AIDS exists in cats.