

More topics in JavaScript

Closure
Callback function
Promise
Fetch

Closure

Global Variables

- לפונקציה יש גישה לכל המשתנים שלה
- כמו בדוגמא הבאה : המשתנה a

```
<body>
  <p id="demo"></p>

  <script>
    myFunction();

    function myFunction() {
      let a = 4;
      document.getElementById("demo").innerHTML = a * a;
    }
  </script>
</body>
```

Global Variables

- לפונקציה יש גישה לכל המשתנים המוגדרים מחוץ לפונקציה
- כמו בדוגמא הבאה : המשתנה a

```
<body>
  <p id="demo"></p>
  <script>
    let a = 4;
    myFunction();

    function myFunction() {
      document.getElementById("demo").innerHTML = a * a;
    }
  </script>
</body>
```

Global Variables

- משתנים המוגדרים ללא var, let, const יהיו גלובליים

```
<body>
  <p id="demo"></p>

  <script>
    myFunction();
    document.getElementById("demo").innerHTML = a * a;

    function myFunction() {
      a = 4;
    }
  </script>
</body>
```

Closure

- Closure היא פונקציה פנימית (inner function) אשר יש לה מצביע למצב שעוטף אותה (outer function)

- במילים אחרות clouser נותן גישה ל:

- ✓ למשתנים של הפונקציה עצמה

- ✓ למשתנים של הפונקציה שעוטפת אותה

- ✓ למשתנים הגלובליים

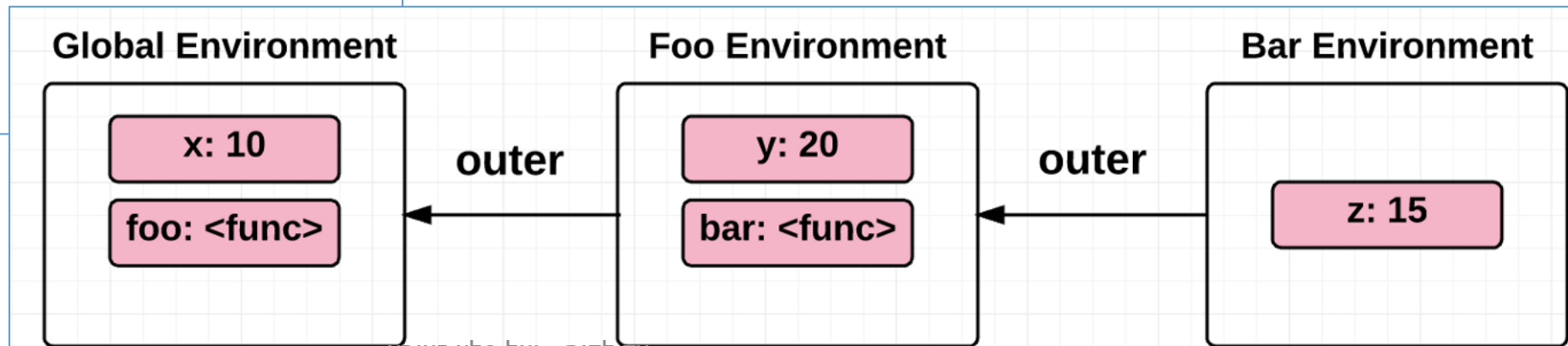
- פונקציה כזו היא פונקציה ש"זוכרת" את הסביבה שיצרה אותה

Scope

מה יודפס?

```
<script>
var x = 10;

function foo() {
  let y = 20;
  function bar() {
    let z = 15;
    return x + y + z;
  }
  return bar;
}
let a = foo();
console.log(a())
</script>
```



מה יודפס?
איזה x ילקח במקרה זה ?

```
var x = 10;

function foo() {
    var y = x + 5;
    return y;
}

function bar() {
    var x = 2;
    return foo();
}

console.log(foo());
console.log(bar());
```



```
<script>
  function sayHello() {
    let say = function () { console.log(hello); }
    // Local variable that ends up within the closure
    let hello = 'Hello, world!';
    return say;
  }
  let sayHelloClosure = sayHello();
  sayHelloClosure(); // 'Hello, world!'
</script>
```

משתנה hello מוגדר אחרי פונקציה אנונימית אך לפונקציה עדין יש גישה אליו, זאת מכיוון שהמשתנה כבר הוגדר כמשתנה הנמצא ב scope של הפונקציה בזמן שהפונקציה נוצרה ולכן כבר זמין בזמן הריצה של sayHello.

אנו רואים שיש לנו גישה למשתנים של הפונקציה העוטפת גם לאחר שהפונקציה חוזרת.

```
var x = 10;
function foo(a) {
var b = 20;

function bar(c) {
var d = 30;
return boop(x + a + b + c + d);
}

function boop(e) {
return e * -1;
}

return bar;
}

var moar = foo(5); // Closure
moar(15);
```

```
var x = 10;
function foo(a) {
var b = 20;

function bar(c) {
var d = 30;
return boop(x + a + b + c + d);
}

function boop(e) {
return e * -1;
}

return bar;
}

var moar = foo(5); // Closure
moar(15);
```

הפונקציה `foo(5)` מחזירה את `bar` לתוך המשנה
`moar` קריאה ל `moar(15)` מפעילה את `bar` עם
הפרמטר 15, בתוך `a` קיים כבר הערך 5 מהקריאה ל
`.foo(5)`
`bar` מפעילה את `boop` והערך המוחזר הוא:
 $(10+5+20+15+30)*-1$

Use Cases of Closures

Data Encapsulation

- Closure מאפשר ניהול המידע – על ידי הסתרה של משתנים מסויימים, אפשרות שינוי משתנים רק ממקום מסויים
- דוגמא:

```
function createCounter() {  
  let count = 0;  
  
  function increment() {  
    count++;  
    return count;  
  }  
  
  return increment;  
}
```

```
const counter1 = createCounter();  
const counter2 = createCounter();
```

```
console.log(counter1()); // Output: 1  
console.log(counter2()); // Output: 1
```

```
console.log(counter1()); // Output: 2  
console.log(counter2()); // Output: 2
```

מתבצעת אנקפסולציה למשתנה ה `count`
הגישה היחידה למשתנה היא על ידי
`increment` שיבצע הגדולה ויחזיר את הערך.

קריאה ל `createCounter` מחזירה את
`increment`.

מספר קריאות ל `createCounter` יחזירו
closures שונים לכל אחד state משלו.

Use Cases of Closures

Event handlers with preserved state

```
<button class="counter">A</button>  
<button class="counter">B</button>
```

```
function createCounter(name) {  
  let count = 0;  
  
  return function () {  
    count++;  
    console.log(` ${name}: ${count}`);  
  };  
}
```

```
document.querySelectorAll(".counter").forEach(btn => {  
  btn.addEventListener("click", createCounter(btn.innerText));  
});
```

Without closure:

```
let count = 0;  
button.addEventListener("click", () => {  
  count++;  
});
```

Button A remembers its own count

Button B remembers its own count

Each handler has its own closure

Use Cases of Closures

Functional Programming

Closure מאפשר יצירת פונקציות שמקבלות פונקציות כפרמטרים ומחזירות פונקציות

```
function createMultiplier(factor) {  
    return function (x) {  
        return x * factor;  
    };  
}
```

```
const double = createMultiplier(2);  
const triple = createMultiplier(3);
```

```
console.log(double(5)); // Output: 10  
console.log(triple(5)); // Output: 15
```

קוד צפוי ויציב יותר

פחות באגים

בדיקות פשוטות

תחזוקה קלה

Callback function

Callback function

פונקציה אשר נשלחת כפרמטר לפונקציה אחרת, ואז נקראת בתוך הפונקציה אליה הגיעה כפרמטר.

```
function operation(val1, val2, callback) {  
    callback(val1, val2)  
}  
  
function sum(a, b) {  
    console.log(a + b)  
}  
  
function divide(a, b) {  
    console.log(a / b)  
}  
  
operation(6, 5, sum)  
operation(9, 3, divide)
```

CallBack Function Use cases- Asynchronous Operations

יצירת אירוע בצורה אסינכרונית

```
<script>
setTimeout(sayHi, 3000);

function sayHi() {
    console.log("Hi")
}
</script>
```

```
<script>
setTimeout(function () { myFunction("I love You !!!"); }, 3000);

function myFunction(value) {
    document.getElementById("demo").innerHTML = value;
}
</script>
```

CallBack Function Use cases

קריאה לפונקציה כתגובה לאירוע

```
document.getElementById("myButton").addEventListener("click", function ()  
{  
    console.log("Button clicked.");  
});
```

• נראה מיד שימוש ב callback עבור תגובה לקריאת http

Fetch API (Promise)

Fetch API

פונקציה המאפשרת לבצע קריאות HTTP ולקבל תגובה

API

Application Programming Interface

סט של כללים וכלים המאפשרים לתוכנות שונות לדבר זו עם זו ולחלוק מידע
גם אם הן מפותחות על ידי ארגונים שונים וגם אם הן רצות על פלטפורמות
שונות.

Web API Basics

- **Endpoints** – URL שה API חושף, כל endpoint מיצג פונקציה שה API יבצע
- **Request and Response Format** – XML , JSON
- **Authentication and Authorization** – הרבה פעמים יש צורך שרק משתמשים מורשים יוכלו לגשת ל API. לצורך כך משתמשים ב API Key או Token

Fetch API

- Fetch api מאפשר לנו לבצע קריאות HTTP ב javascript
- כל הבראוזרים הפופולרים כיום תומכים ב fetch api
- Fetch api מספק פונקציות javascript גלובליות שניתן להשתמש בה הנקראות fetch, והיא דואגת להחזיר resources מה server

Syntax

```
fetch(URL [, options])
```

URL : a URL object that represents the path of the resource to be fetched

Options(optional): Any further options such as:

- Method: The request method is either GET POST PUT DELETE.
- Headers : for example 'Content-Type': 'application/json'
- Body: json string
- Mode
- Credentials
- Cache

How fetch() works

```
<script>
fetch('url')
  .then(response => {

    //handle response
    return response.json()
  })
  .then(data => {

    //handle data

  })
  .catch(error => {

    //handle error

  });
</script>
```

Calling an External API Using Fetch

- נשתמש ב API חינמי הנקרא JSONPlaceholder API
- אינו דורש אימות משתמש או API key
- <https://jsonplaceholder.typicode.com/guide/>

Fetch

- נשתמש בפונקצית fetch של javascript על מנת לפנות ל api חיצוני שיחזיר נתונים,
- את הנתונים נדפיס ללוג

```
fetch('https://jsonplaceholder.typicode.com/posts')
  .then(response => {
    // Check if the response is successful
    return response.json(); // Parse the JSON from the response
  })
  .then(data => {
    // Use the data from the response
    console.log(data);
  })
  .catch(error => {
    // Handle any errors
    console.error('Error:', error);
  });
```

Fetch with error handling

```
// Fetch data from the JSONPlaceholder API
fetch('https://jsonplaceholder.typicode.com/posts')
.then(response => {
    if (!response.ok) {

        throw new Error(`HTTP error! Status: ${response.status}`);
    }
    // Return the response data as JSON
    return response.json();
})
.then(data => {
    // Handle the JSON data
    console.log('Fetched Data:', data);
    // Example: Display the titles of the posts
    data.forEach(post => {
        console.log(`Post ${post.id}: ${post.title}`);
    });
})
.catch(error => {
    console.error('There was a problem with the fetch operation:', error);
});
```

Getting a resource

```
fetch('https://jsonplaceholder.typicode.com/posts/1')  
  .then((response) => response.json())  
  .then((json) => console.log(json))  
  .catch(error => {  
    console.error('There was a problem with the fetch operation:', error);  
  });
```


Getting resources

```
fetch('https://jsonplaceholder.typicode.com/posts')  
  .then((response) => response.json())  
  .then((json) => console.log(json))  
  .catch(error => {  
    console.error('There was a problem with the fetch operation:', error);  
  });
```

```
<body>
<h1>Post Titles</h1>
<ul id="post-list"></ul>
<script>
// JavaScript code goes here
fetch('https://jsonplaceholder.typicode.com/posts')
  .then(response => {
    if (!response.ok) {
      throw new Error(`HTTP error! Status: ${response.status}`);
    }
    return response.json();
  })
  .then(data => {
    console.log(data);
    const postList = document.getElementById('post-list');
    data.forEach(post => {
      const listItem = document.createElement('li');
      listItem.innerText = `Post ${post.id}: ${post.title}`;
      postList.appendChild(listItem);
    });
  })
  .catch(error => {
    console.error('Fetch error:', error);
  });
</script>
</body>
```

תרגיל כיתה

ה api אליו תפנו יחזיר לכם את רשימת החגים בארה"ב

אתם מעוניינים אך ורק בשדות של date ו name

החזירו את המידע ל client והציגו את המידע בדף html

להלן ה api endpoint

<https://date.nager.at/api/v3/publicholidays/2026/US>

שאלות?

PROMISE

```
fetch('https://jsonplaceholder.typicode.com/todos/1')  
  .then(response => response.json())  
  .then(json => console.log(json))  
  .catch(error => console.log(error));
```

ASYNC/AWAIT

הצורה המועדפת על
המתכנתים היא יותר זורמת
עם הקוד

```
async function runProcess() {  
  try {  
    const response = await fetch('https://jsonplaceholder.typicode.com/todos/1');  
    const json = await response.json();  
    console.log(json);  
  } catch (error) {  
    console.log(error);  
  }  
}
```

צד לקוח - יעל סלע זעירא
runProcess();