

Materiali mesimor per javen 3 - 7 mars 2025 Scantech

E hane - 3 mars 2025:

Sesioni i pare: Perseritje me funksione (definimi dhe thirrja e funksionit, arrow functions)

Sesioni i dyte: Implement a function that filters even numbers from an array.

```
//funksioni per ti gjetur numrat cift tek array: myNumbers
function filterEvenNumbers(numbers) {
    return numbers.filter(num => num % 2 === 0);
}

// Example usage
const myNumbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
const evenNumbers = filterEvenNumbers(myNumbers);

console.log(evenNumbers); // Output: [2, 4, 6, 8, 10]
```

E merkure - 5 mars 2025:

Sesioni i pare: Callbacks and problem “callback hell” and solution with Promises
(and a small introduction to async/await)

Shembulli i nje asynchronous callback te thjeshte:



```
//Metoda "Callback"
function fetchDataCallback(callback) {
  setTimeout(() => {
    callback("Data fetched using callback!");
  }, 1000);
}

// Shembulli i perdorimit te funksionit me Callback
fetchDataCallback((result) => {
  console.log(result); // Output after 1 second: Data fetched using callback!
});
```

Shembulli i njejte por me Promise:

```
// Metoda Promise
function fetchDataPromise() {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve("Data fetched using Promise!");
    }, 1000);
  });
}

// Shembulli i perdorimit te funksionit me Promise
fetchDataPromise().then((result) => {
  console.log(result); // Output after 1 second: Data fetched using Promise!
});
```

Shembull i “callback hell”:

```
function step1(callback) {
  setTimeout(() => {
    console.log("Step 1 completed");
    callback();
  }, 1000);
}

function step2(callback) {
  setTimeout(() => {
    console.log("Step 2 completed");
    callback();
  }, 1000);
}

function step3(callback) {
  setTimeout(() => {
    console.log("Step 3 completed");
    callback();
  }, 1000);
}

// Nested callbacks
step1(() => {
  step2(() => {
    step3(() => {
      console.log("All steps completed!");
    });
  });
});
```

Zgjidhje Promises:



```
function step1() {
  return new Promise(resolve => {
    setTimeout(() => {
      console.log("Step 1 completed");
      resolve();
    }, 1000);
  });
}

function step2() {
  return new Promise(resolve => {
    setTimeout(() => {
      console.log("Step 2 completed");
      resolve();
    }, 1000);
  });
}

function step3() {
  return new Promise(resolve => {
    setTimeout(() => {
      console.log("Step 3 completed");
      resolve();
    }, 1000);
  });
}

// Chaining Promises
step1()
  .then(() => step2())
  .then(() => step3())
  .then(() => console.log("All steps completed ↓"));
```

Permbledhje:

|  Summary | | |
|--|---|--|
| Concept | What It Does | Example |
| Callback | A function passed into another function to execute later. | <code>fetchData(callback)</code> |
| Promise | Represents an async task that may succeed or fail. | <code>new Promise((resolve, reject) => {...})</code> |
| <code>.then()</code> | Runs when a promise resolves (success). | <code>.then(response => console.log(response))</code> |
| <code>.catch()</code> | Runs when a promise rejects (failure). | <code>.catch(error => console.log(error))</code> |
| Next Step: Use <code>async/await</code> for even better readability!  | | |

Sesioni i dyte: Diskutime mbi shembuj ne boten reale per *Database calls* edhe *API Requests*

E premte - 7 mars 2025:

Sesioni i pare: Konvertimi i Promises ne Asynchronous Functions (komandat *async/await*) dhe error handling me metoden *try {} catch {}*

Sesioni i dyte: detyre per: Konsumimi i te dheneve me metoden *fetch()* duke perdorur error handling me metoden *try {} catch {}*.

Linku i Fake API: <https://jsonplaceholder.typicode.com/>

Linku: <https://cocktail-recipes.surge.sh/>

Shembulli:

```
async function handleSearch(event) {
  if (event) event.preventDefault();

  let searchValue = searchInput.value.trim();
  if (!searchValue) return;

  try {
    let response = await fetch(`https://www.thecocktaildb.com/api/json/v1/1/search.php?s=${searchValue}`);
    if (!response.ok) throw new Error("Failed to fetch data");

    let data = await response.json();
    console.log("Fetched Data:", data);

    if (data.drinks) {
      console.log("Cocktails Found:", data.drinks.map(drink => drink.strDrink));
    } else {
      console.log("No cocktails found.");
    }
  } catch (error) {
    console.error("Error fetching data:", error);
  }
}
```