

# Tutorial 6

Requests, Frontend/Backend

# Exercise 4

- also think about resetting boxes, text, ...

```
// Task 2.4

function palindrome(word) {
  const palindromeContainer = document.querySelector("#palindromeContainer");
  // remove old boxes from previous analysis
  palindromeContainer.replaceChildren();

  for (let i = 0; i < wordLength(word); i++) {
    const char = word[i];
    const opposingChar = word[wordLength(word) - i - 1];
    const isPalindrome = char.toLowerCase() === opposingChar.toLowerCase();

    const palindromeBox = document.createElement("span");
    palindromeBox.classList.add("palindromeBox");

    if (isPalindrome) {
      palindromeBox.classList.add("green");
    } else {
      palindromeBox.classList.add("red");
    }
    palindromeContainer.appendChild(palindromeBox);
  }
}
```

This line removes any existing content within the `palindromeContainer` element. This is done to clear any previous analysis results before displaying the current one.

*Enter a noun below and start analyzing!*

Analyze!

**Word entered:** abc

The language of your word is: The language of the word cannot be determined.

Your word has the length: 3

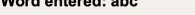
The first letter is capitalized: false

Number of distinct letters: 3

Ratio of consonants to vowels in the word: 2.00

Inverted the case of the word: ABC

Worth of the word: 6

Palindromeness:  


# Exercise 4

- check what is inside the element if you reset it → maybe you also remove a title (if title element is inside the element for the palindrome boxes for example)
- and check where you put your elements in the html file → inside div .analysis if you want it to be inside the lighter square else it will be outside the field

outside

Enter a noun below and start analyzing!

abc

**Word entered: abc**

The language of your word is: The language of the word cannot be determined.

Your word has the length: 3


The first letter is capitalized: No

Number of distinct letters: 3

Ratio of consonants to vowels in the word: 2

Inverted the case of the word: ABC

Worth of the word: 6



inside

Enter a noun below and start analyzing!

abc

**Word entered: abc**

The language of your word is: The language of the word cannot be determined.

Your word has the length: 3


The first letter is capitalized: No

Number of distinct letters: 3

Ratio of consonants to vowels in the word: 2

Inverted the case of the word: ABC

Worth of the word: 6



# HTTP Requests

- Interaction from client (browser) and server
- Client sends a request containing:
  - The path to the requested resource
  - The header containing user authentication, cookies etc.
  - A request body, for sending data to the server
- GET:
  - Retrieve data from the server
  - No request body!
- POST:
  - Put some data into the server's database -> Data is sent in the request body
  - Information about what type of data sent is in the header-> e.g. "application/json"

# HTTP Responses

- Some data: html, json, xml
- A status code. Most common:
  - 200 OK
  - 302 found (redirect)
  - 403 forbidden (client has no access rights)
  - 404 not found
  - 500 server error
- Full list: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>

# AJAX

- **Asynchronous JavaScript and Xml**
- A way to make the website update incrementally, without reloading or stopping other processes.
- Different ways to implement, we use `fetch()`
- `fetch()` returns a promise that resolves to the response or an error.

# Promises

For more info see:

<https://javascript.info/promise-basics>

or <https://javascript.info/fetch>

- An object in JS that has three states:
  - pending
  - fulfilled -> returns a value
  - rejected -> returns an error
- It takes as an argument the function to be executed when fulfilled and when rejected
- The `.then()` method executes one of the functions when the promise is resolved
- The `.catch()` method is executed when the promise is rejected or an error occurred

# Example

Situation:

You want to bake a Raspberry Heart Cheescake with your roommate. Since you will be at home all day solving exercises, you ask your roommate to buy the ingredients on their way home.



# Example

```
const getChocolate =  
  new Request("Zurich/Hofwiesenstr350/Migros/buy=whiteChocolate",  
    {method: "GET"})  
  
fetch(getChocolate).then(function(groceries){  
  
  bakeCake(groceries.whiteChocolate, *otherIngredients);  
  
}).catch(error){  
  respondTo(error);  
  watchMovie();  
}
```

# JS Syntax Variants

The **await** keyword has similar functionality as the **.then()** method.

The **await** keyword can only be employed with functions declared as asynchronous with the **async** keyword.

**await** keyword is used in conjunction with the **async** function to make asynchronous programming more readable and manageable. It's primarily used with promises to pause the execution of a function until the promise is resolved or rejected

```
async function myFunc(){
  let response = await fetch(request);

  if (response.ok){//HTTP-status is 200-299
    let json = await response.json();
    console.log(json)
  }else{
    alert("HTTP-Error: "+response.status);
  }
}
```

# Requests

```
const element = document.querySelector("html-element");  
element.addEventListener("type-of-event", (event) => {  
  const request = new Request("API endpoint", {method: "GET"});  
  fetch(request)  
    .then((response) => response.json())  
    .then((data) => {  
      // Manipulation of retrieved data comes here  
    })  
    .catch((error) => {  
      // Error handling comes here  
    });  
});
```

This line selects an HTML element with the tag name "html-element" and stores it in the variable

This line adds an event listener to the `element` you selected. It listens for the "type-of-event" (which should be replaced with an actual event type, like "click" or "submit"). When this event occurs on the selected HTML element, the callback function provided in the second argument is executed.

Inside the event listener's callback function:

- a. It creates a new `Request` object targeting an "API endpoint" and using the HTTP method "GET."
- b. It makes a fetch request using the `fetch` function to retrieve data from the specified API endpoint.
- c. It handles the response from the API by parsing it as JSON using `.then((response) => response.json())`.
- d. After parsing the response, it processes the data inside the next `.then()` block.
- e. It provides an error handling mechanism using `.catch()` to deal with errors that may occur during the fetch request.

# Requests

```
const element = document.querySelector("html-element");
element.addEventListener("type-of-event", async (event) => {
  const request = new Request("API endpoint", {method: "GET"});
  try {
    const response = await fetch(request);
    const data = await response.json();
    // Manipulation of retrieved data comes here
  } catch (error) {
    // Error handling comes here
  }
});
```

This line adds an event listener to the `element` you selected.

When this event occurs on the selected HTML element, the callback function provided in the second argument is executed. The callback function is marked as `async`, which means it can use `await` for asynchronous operations.

Inside the event listener's callback function:

- It creates a new `Request` object
- It makes a fetch request using the `await` keyword, which allows for asynchronous waiting, making the code more concise and easier to read. **It waits for the fetch request to complete before continuing.**
- It then processes the response by parsing it as JSON using `await response.json()`.
- After parsing the response, it processes the data inside the same `try` block.

# JavaScript and Databases

What would we need to implement this page, using Flask?

What do we see and Ideas how to implement this?

(spoiler warning for the message displayed in the screenshot...)

## Message Board

2023-10-26 18:44:59.539812

 [not implemented]



2023-10-26 18:36:44.883011

 [not implemented]

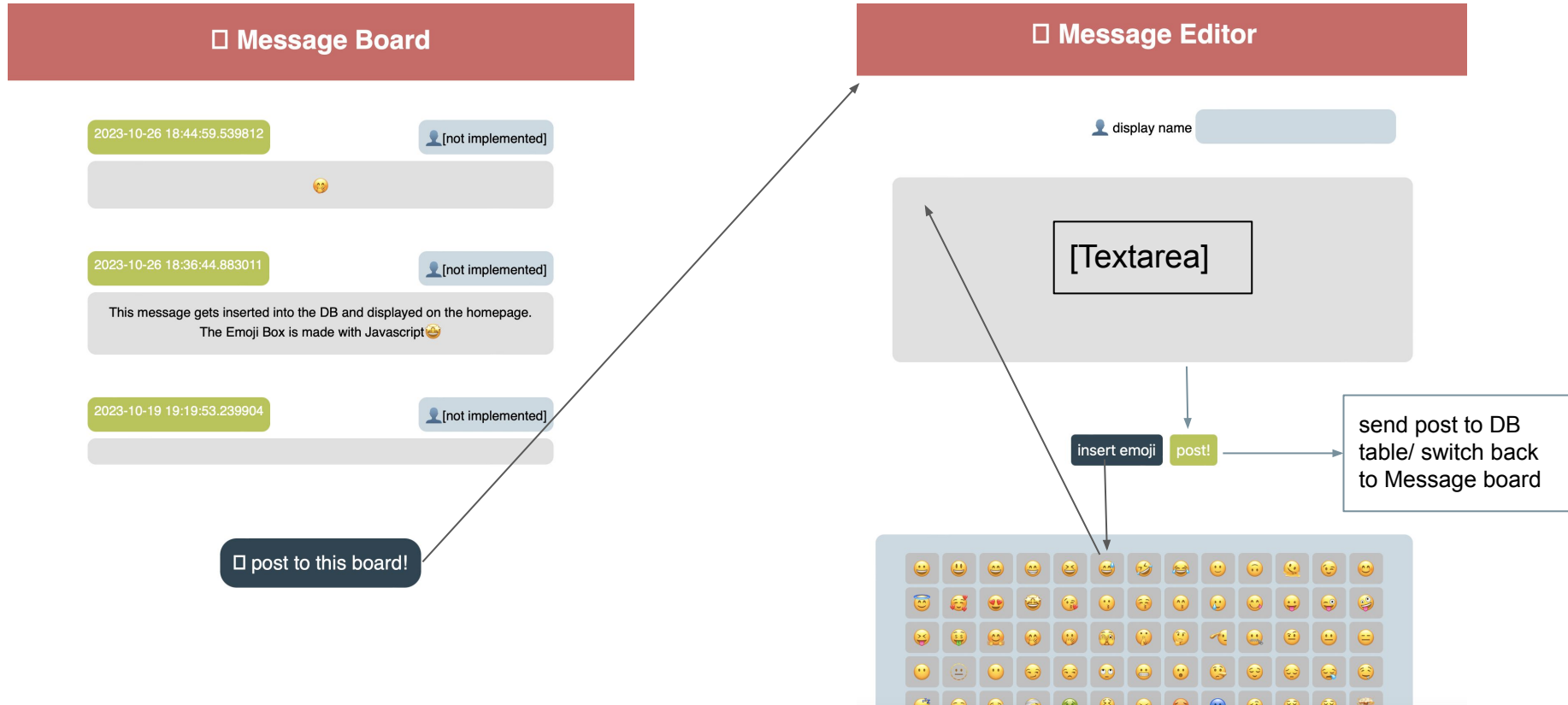
This message gets inserted into the DB and displayed on the homepage.  
The Emoji Box is made with Javascript 🙌

2023-10-19 19:19:53.239904

 [not implemented]

post to this board!

# Switch between pages for Viewing and Posting



# Connection of Frontend and Backend

Combine Javascript with Python code in Flask:

- Create an emoji container with Javascript
- Create a Form in the HTML:
  - define method (post)
  - define action (link to a python function)
  - Assign elements you want to access with CSS or JavaScript later on classes and/or ids
- Write JS functions to interact with the html after certain buttons are clicked

```
@app.post('/post-message')
def post_message():
    if request.form["submit"]:
        #displayname = request.form["displayname"]
        message = request.form["message"]
        db = get_db()
        cur = db.cursor()
        try:
            cur.execute("INSERT INTO messages (message, created_on) VALUES (%s, NOW())",
                        (message,))
            db.commit()
            cur.close()
            return app.redirect("/")
        except Exception as e:
            print(e)
            cur.close()
            return "Error while querying the database"
```

```
<!-- Form for posting a message. -->
<form method="post" action="/post-message">
    <div class="row" id="displayname-container">
        <!-- Input field for the user's display name. -->
        <label for="displayname">👤 display name</label>
        <input
            type="text"
            for="displayname"
            name="displayname"
            class="text-input"
            id="displayname"
        />
    </div>
    <div class="row">
        <!-- Textarea for entering the message. -->
        <textarea name="message" class="text-input" id="message"></textarea>
    </div>
    <div class="row">
        <!-- Button to insert emojis and submit the form. -->
        <button class="btn blue-button" id="show-emojis">insert emoji</button>
        <input
            type="submit"
            class="btn green-button"
            id="send-message"
            name="submit"
            value="post!"
        />
    </div>
</form>
<!-- Container for displaying emojis. -->
<div class="row emoji-container"></div>
```

# Create a JSON output to fetch with JS

What do we want?

- Click button -> get random message from DB “messages” table
- Click “X” removes the displayed message.
- Doesn't affect the rest of the HTML

## □ Message Board

Get Random Message X

["lets see if this new message will show up in the random message display 🤔"]

```
new *
@app.route('/get_data', methods=['GET'])
def get_data():
    db = get_db()
    cursor = db.cursor()
    cursor.execute('SELECT message FROM messages')
    data = cursor.fetchall()
    cursor.close()
    return jsonify(data)

fetch(input: '/get_data') Promise<Response>
.then(response => {
    if (!response.ok) {
        throw new Error('Network response was not ok');
    }
    return response.json();
}) Promise<any>
.then(data => {
    fetchedData = data;
}) Promise<any>
.catch(error => {
    // Handle errors
    console.error('Error:', error);
});
```



# Demo: Message Board

- Interact with DB
- Switch between pages, redirect
- Interact through HTML with JS and Python
- “Post” and “Get” methods Examples
- (Free to experiment with the provided code)



**HTML**



**HTML + CSS**



**HTML + CSS + JAVA SCRIPT**

# Exercise 5

- JS and API
- Part 1:
  - GET-Request to get a random cat image and display it on simple webpage
- Part 2:
  - GET-Request to dictionary and extract needed information

## Cat Picture Generator

Give me a Cat picture!



Word Analytics

house

☐ Phonetics

**Word entered: house**

Definitions:

1. noun: A structure built or serving as an abode of human beings.

Word Analytics

house

☒ Phonetics

**Word entered: house (/haʊs/)**

Definitions:

1. noun: A structure built or serving as an abode of human beings.

# Some more tutorials and examples

- <https://javascript.info/>
- <https://htmldog.com/>