Department of
COMPUTER SCIENCE
HONG KONG BAPTIST UNIVERSITY

COMP7980 – Dynamic Web and Mobile Programming
COMP7270 – Web and Mobile Programming

Chapter 6 AJAX & Restful API

Course Instructors: Dr. Ma Shichao, Dr. Zhang Ce, and Mr. Jiang Jintian

Announcement

- Project Demonstration
  - April 9 and April 16, 2025
  - Should submit the project report and source code before April 9
  - Tentative report outlines:
    - Introduction
    - Database design
    - UI design
    - Extra features
    - Teammates' responsibilities
    - …

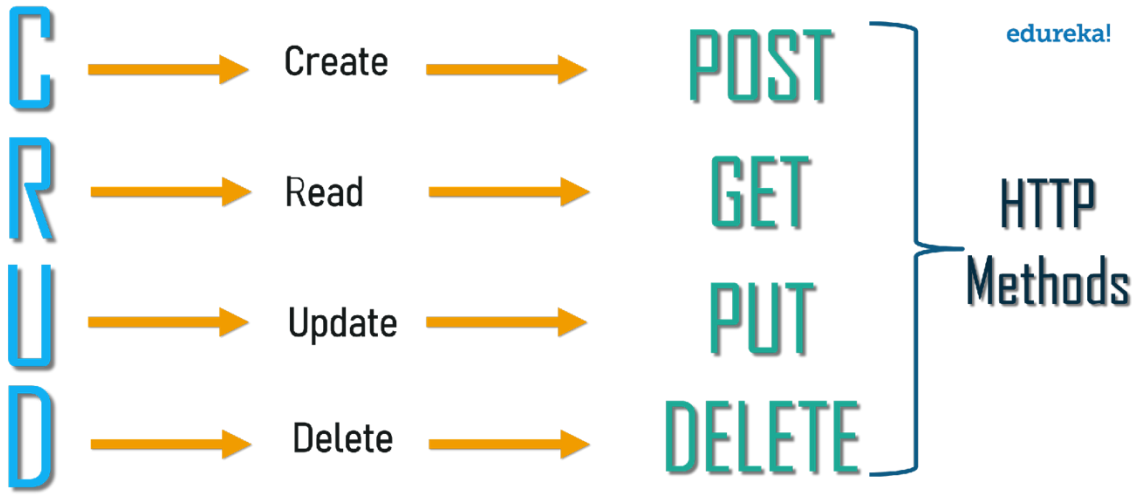
Preparation for Lab

- Download lab06-materials from Moodle
- Extract it to a folder
  - Rename the folder name to ‘lab06’
- Get into the folder and install the libraries
  - `cd lab06`
  - `npm install`
- Setup the MongoDB database
  - In the file `utils/db.js`, please assign the variable `process.env.MONGODB_URI` as your own connection string
- Test the code
  - Run `npm start` and access `localhost:3000`

Restful API

- A **RESTful API** (Representational State Transfer API) is an architectural style and approach for **designing web services** that enable communication between systems over the internet.
- It follows a set of principles and constraints, which allow for the **creation, retrieval, update, and deletion (CRUD) operations** on **resources**.

Restful API



- Uniform Interface:
  - RESTful APIs use a uniform set of **well-defined methods** and standard HTTP verbs such as GET, POST, PUT, PATCH, and DELETE to perform **CRUD operations** on resources. These methods provide a consistent and predictable way to interact with the API.

Restful API

- Resource-Oriented: **Resources** are the **key entities exposed** by a RESTful API. Each resource is identified by a unique URI (Uniform Resource Identifier) and can be represented in different formats, such as JSON or XML. **Clients** interact with these resources through the **API's endpoints**.
- **Representation**: Resources are represented in a format that can be easily understood by clients, such as **JSON (JavaScript Object Notation)** or XML (eXtensible Markup Language). The representation includes the data and any associated metadata.

RESTful API Endpoints

Bookings is the resource.
Representation is JSON

Non-RESTful (current implementation)	RESTful
POST /booking	**POST** /bookings
GET /booking/**read**/:id	**GET** /bookings/:id
POST /booking/**update**/:id	**PUT** /bookings/:id
POST /booking/**delete**/:id	**DELETE** /bookings/:id

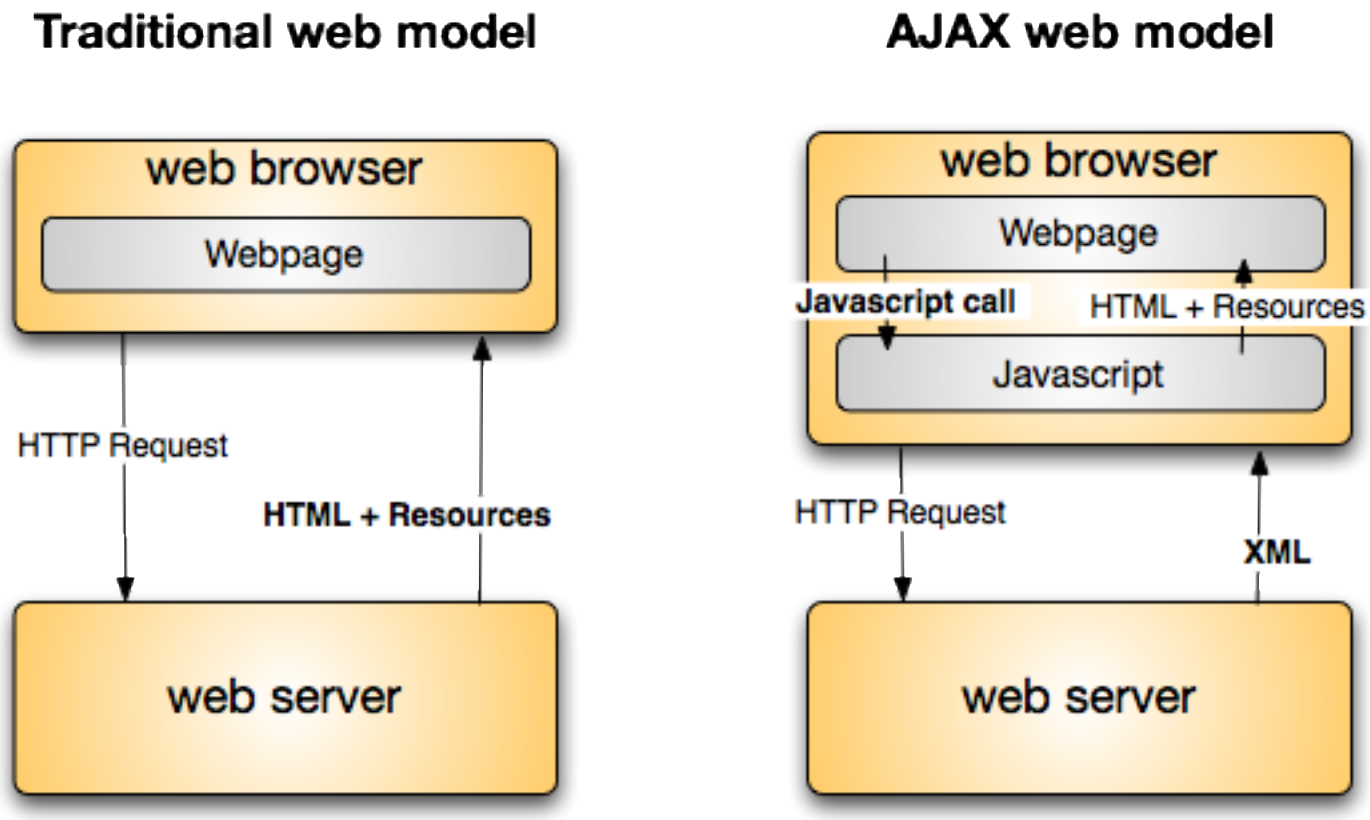
HTTP Methods and HTML Form

- Please note that **HTML forms only understand HTTP methods** GET and POST.
- Using a DELETE **form method** will be **treated as a** POST.
- Thus, we should use **Ajax** request for non GET or POST requests.

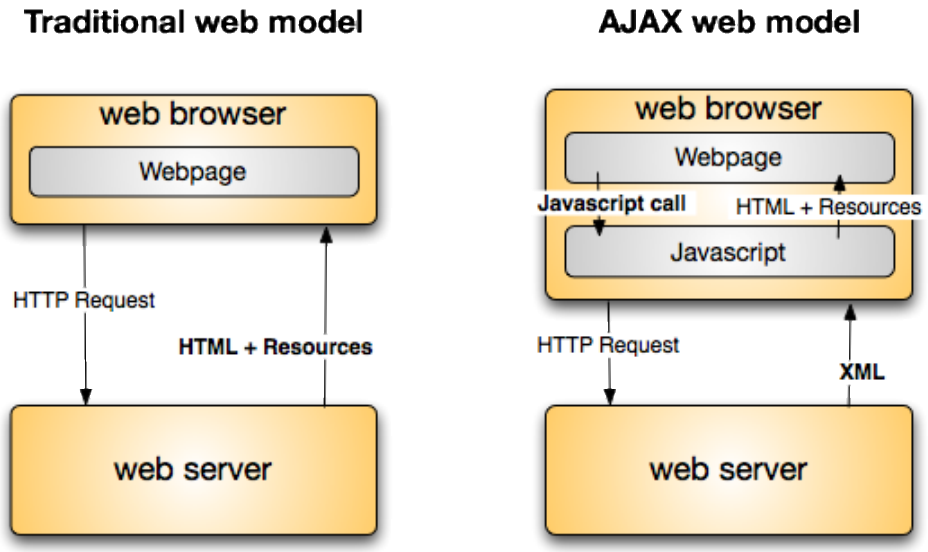
AJAX

AJAX

- Asynchronous JavaScript and XML
- The HTTP **request** is now initiated by client-side **JavaScript**.
- Server **response** only contains data (like **XML**), but not the entire page.



AJAX



- **XML** was once the go-to choice of data exchange format.
  - Now, **JSON** is more often used as the response data format.
- The client-side JavaScript will receive this data response and **update the current web page**.
  - As such, we are able to update a web page **without reloading it**.

Benefits of AJAX

- Ajax can **reduce the traffic travels** between the client and the server.
  - Usually, only data (in JSON or XML formats) will be sent, **HTML and CSS codes are not transmitted**.
- The **server response time is faster** so increases performance and speed.
  - **Data aren't processed in the server side**, like being included in html.

Making AJAX API Calls

- Here are some possible ways to make an Ajax call:

XMLHTTPRequest	Built-in browser object, **old standard**, support old browsers.
Fetch	**Built-in object for modern browsers**, more straight-forward interface.
Axios	Open-source library. **Better error handling** Works both in browsers and **Node.js**.…
JQuery	Open-source library. Doesn’t have much to offer as compared to fetch.

Fetch API

- The Fetch API provides a global fetch method
- Making an API call with fetch()

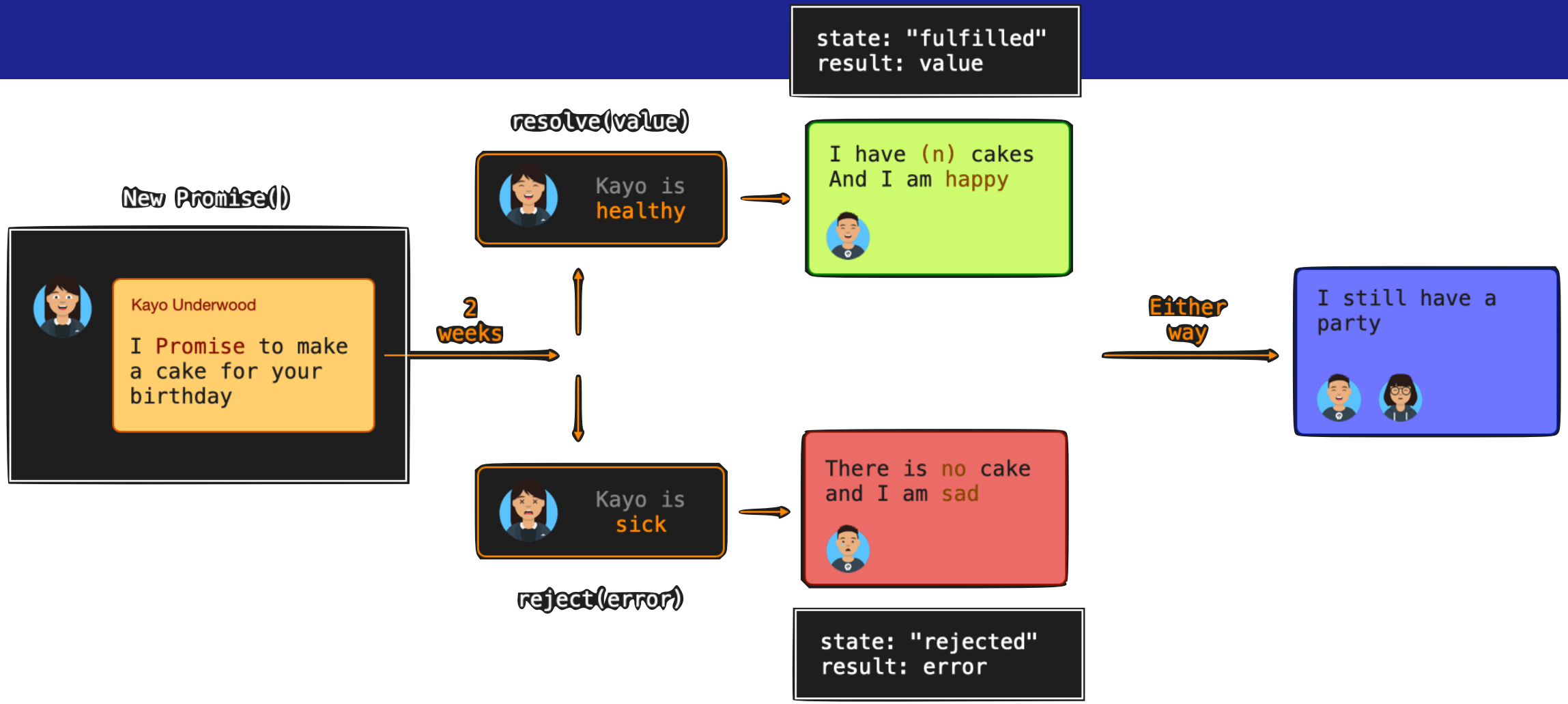
```
const response = await fetch(`/api/bookings?page=${page}&perPage=${perPage}`);
```

- fetch doesn’t perform automatic transformations, so we have to convert JSON data with response.json():

```
// convert the response to json
const json = await response.json();
```

Fetch API

- The fetch function returns a **Promise** that **resolves** to a Response object representing the server's response to the request.
- The response.json() method is called on the Response object to extract the JSON data from the response. **This method also returns a Promise** that resolves to the parsed JSON data.
- **The function awaits the resolution** of the response.json() Promise and assigns the parsed JSON data to the json variable.



Synchronous = happens at the same time. **Asynchronous** = doesn't happen at the time

Based on real-life scenario

Made by Thu Nghiem • Founder at DevChallenges.io

URLSearchParams

- On client side JavaScript, the **query string** part of a URL is available via window.location.search. For example,

?perPage=2&page=2

- To retrieve the **search/query parameters**, we can use URLSearchParams, and its utility methods

```
const urlParams = new URLSearchParams(window.location.search);
renderPage(1, urlParams.get("perPage"))
```

OnSubmit

```
<form onsubmit="handleSubmit(event)">
```

- The onsubmit attribute allows us to specify **a function that will be executed when the form is being submitted**.
- The original form submission **will also be executed** in general.
  - event.preventDefault() will cancel the **default submission**.

```
async function handleSubmit(event) {

    event.preventDefault();
```

FormData

- The FormData interface provides a way to easily **construct a set of key/value pairs representing form data (fields and their values)**
- event.target refers to the **form element** that **triggered the submission** event.

```
async function handleSubmit(event) {
    // prevent the default behaviour
    event.preventDefault();
    // get the id from the url
    const urlParams = new URLSearchParams(window.location.search);
    let id = urlParams.get("id")
    // get the form data
    const formData = new FormData(event.target);
```

URL-encoded form data

```
// A function to update a booking with www-form-urlencoded data
async function updateBooking(id, booking) {

    const response = await fetch(`/api/bookings/${id}`, {
        method: 'PUT',
        headers: {
            'Content-Type': 'application/x-www-form-urlencoded'
        },
        body: new URLSearchParams(booking)
    });
    // convert the response to json
    const json = await response.json();
    // return the json
    return json;
}
```

The form data

application/x-www-form-urlencoded

- A MIME type (Media Type) that specifies the format of data being sent in an HTTP request or response when using the **URL-encoded form data format**.
- When a form is submitted using the **application/x-www-form-urlencoded** format, the form data is encoded in a **key-value pair format**, where each field name and its corresponding value are joined by an equal sign (=), and multiple pairs are separated by an ampersand (&). For example, a key-value pair name=John would be URL-encoded as **name=John**.
- This format is commonly used in HTML forms and is the **default format when submitting HTML forms** without specifying an explicit enctype attribute.

Client-side redirect

- The window.location.href can redirect the user to a new URL.

```
/ A function to display a confirm box for delete, display the response
and redirect to the bookings page
async function handleDelete() {
    // get the id from the url
    const urlParams = new URLSearchParams(window.location.search);
    let id = urlParams.get("id")
    // display a confirm box
    if (confirm(`Are you sure you want to delete booking ${id}?`)) {
        // delete the booking
        const deletedBooking = await deleteBooking(id);
        // display the response
        alert(JSON.stringify(deletedBooking));
        // redirect to the bookings page
        window.location.href = "/bookings.html";
    }
}
```