



# Pemrograman Berbasis Web

Pertemuan 11

**Nori Wilantika**  
**Lya Hulliyyatus Suadaa**  
**Yeni Rimawati**

Politeknik Statistika STIS  
Prodi DIV Komputasi Statistik

# 11

AJAX, JSON, dan API

The background is a light cream color, decorated with several abstract elements. In the top left, there is a horizontal red brushstroke and a cluster of small black dots. On the left side, there is a large, textured grey circle with a black starburst pattern above it. In the bottom left, there is a pink brushstroke and a cluster of small black dots. On the right side, there is a large, textured yellow circle, a thin black wavy line with two small yellow circles attached to it, and a grey brushstroke at the bottom right.

AJAX



Pilih Jurusan :

No	NIK	Nama	Jurusan
1	147343998	Chika Lestari	SI
2	136349343	Kory Ubi	SI
3	135610157	Dahlan Iskan	SI
4	135310156	Safitri Ayu	SI
5	1253434	Setiawan	SI

- Pemrosesan request dari client menggunakan server-side programming konvensional:
  - User memilih jurusan melalui drop-down list
  - Client (browser) mengirimkan formulir ke sebuah server-side script
  - Server melakukan queries data mahasiswa untuk jurusan yang dipilih user dan generate HTML yang berisi tabel yang memuat data hasil query
- Keseluruhan halaman harus direload ulang





localhost/belajar\_ajax/

Pilih Jurusan :  ▼

No	NIK	Nama	Jurusan
----	-----	------	---------



# APA ITU AJAX?

- AJAX = Asynchronous JavaScript And **XML**, adalah sebuah teknik untuk membuat halaman web yang cepat dan dinamis.
- AJAX memungkinkan halaman web diperbarui secara asinkron dengan mengirim dan mengambil data dari server di belakang layar (tanpa memuat ulang halaman).
- AJAX is not a programming language.
- AJAX is a misleading name. Historically, data was transferred in **XML** format. Nowadays it is equally common to transport data as plain text or **JSON** text.

Provinsi

Jawa Tengah

Kabupaten

Pilih Kabupaten

KAB. BREBES

KAB. CILACAP

KAB. DEMAK

KAB. GROBOGAN

KAB. JEPARA

KAB. KARANGANYAR

KAB. KEBUMEN

KAB. KENDAL

KAB. KLATEN

KAB. KUDUS

KAB. MAGELANG



belajar php|

belajar php

belajar php **dasar**

belajar php **untuk pemula**

belajar php **mysql**

belajar php **dari nol**

belajar php **codeigniter**

belajar php **dasar pdf**

belajar php**maker**

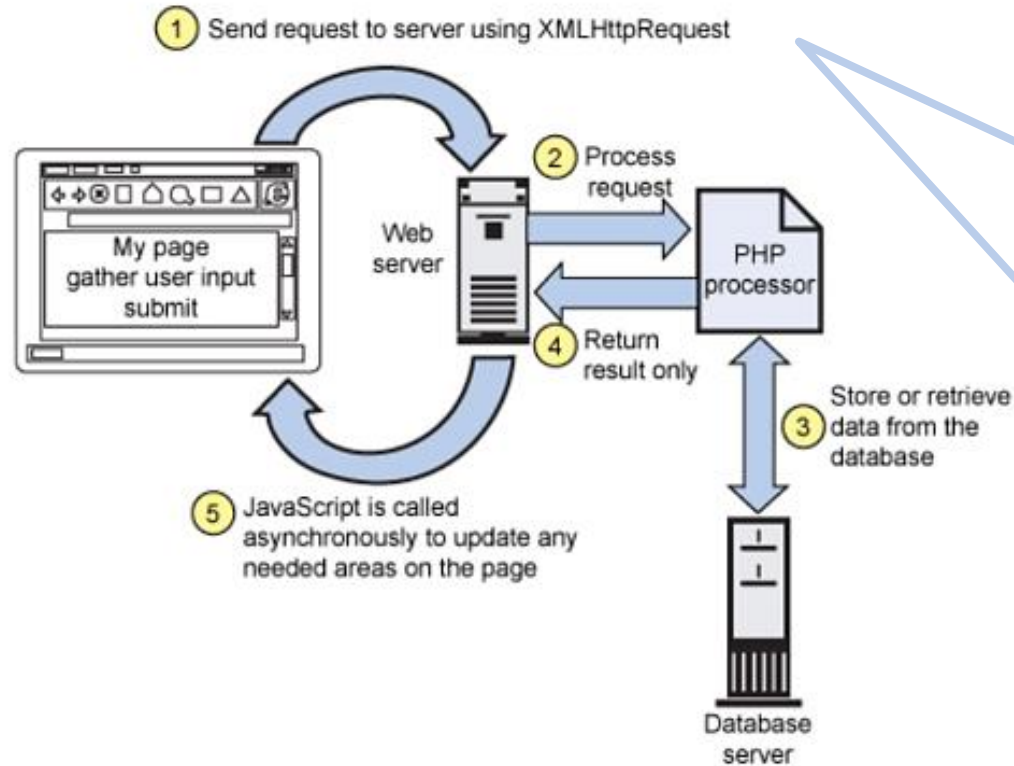
belajar php **dari dasar samai mahir**



# CARA KERJA AJAX

- AJAX hanya menggunakan kombinasi dari:
  - Browser built-in XMLHttpRequest object (untuk request data dari web server)
  - JavaScript dan HTML DOM (untuk menampilkan atau menggunakan data)
- Dengan kata lain, AJAX adalah sekumpulan metode JavaScript yang berkaitan dengan objek XMLHttpRequest.
- Pada sisi server, PHP scripts biasa digunakan untuk menerima/mengirim data dan menangani transaksi database.
- Versi lama dari Internet Explorer menggunakan ActiveXObject. Modern Browsers dapat menggunakan Fetch API daripada objek XMLHttpRequest.





The keystone of AJAX is the XMLHttpRequest object.

Tahapan pembuatan AJAX:

1. Membuat objek XMLHttpRequest object
2. Mendefinisikan callback function
3. Membuka objek XMLHttpRequest dan mengirim request ke server

K. Ramirez: Build Ajax-based Web sites with PHP. IBM, 2 Sep 2008.  
<https://www.ibm.com/developerworks/library/wa-aj-php/>  
[accessed 03 Dec 2020]



# XMLHttpRequest Object

1. Membuat objek XMLHttpRequest object

```
variable = new XMLHttpRequest();
```

2. Mendefinisikan fungsi callback -> kode yang dieksekusi ketika respon dari server sudah ready.

```
xhttp.onload = function() {  
    // What to do when the response is ready  
}
```

3. Membuka objek XMLHttpRequest dan mengirim request ke server, menggunakan metode open() dan send() pada objek XMLHttpRequest.

```
xhttp.open("GET", "ajax_info.txt");  
xhttp.send();
```



# SEND A REQUEST TO A SERVER

```
xhttp.open ("GET", "demo_get2.php?fname=Henry&lname=Ford", true);
```


Tipe dari request:  
GET or POST

File yg melakukan aksi pada server.  
Dapat berupa file apa saja, seperti  
.txt dan .xml, atau server scripting  
files seperti .asp dan .php

true (asynchronous) atau  
false (synchronous)

POST request:

```
xhttp.open("POST", "demo_post2.asp");  
xhttp.send("fname=Henry&lname=Ford");
```





# SERVER RESPONSE

- Server Response Properties

Property	Description
responseText	get the response data as a string
responseXML	get the response data as XML data

- Penggunaan ResponseText:

```
document.getElementById("demo").innerHTML = xhttp.responseText;
```

- Server Response Methods

Method	Description
getResponseHeader()	Returns specific header information from the server resource
getAllResponseHeaders()	Returns all the header information from the server resource

# XMLHttpRequest Object Method

Method	Description
<code>new XMLHttpRequest()</code>	Creates a new XMLHttpRequest object
<code>abort()</code>	Cancels the current request
<code>getAllResponseHeaders()</code>	Returns header information
<code>getResponseHeader()</code>	Returns specific header information
<code>open(<i>method,url,async,user,psw</i>)</code>	<p>Specifies the request</p> <p><i>method</i>: the request type GET or POST <i>url</i>: the file location <i>async</i>: true (asynchronous) or false (synchronous) <i>user</i>: optional user name <i>psw</i>: optional password</p>
<code>send()</code>	<p>Sends the request to the server</p> <p>Used for GET requests</p>
<code>send(<i>string</i>)</code>	<p>Sends the request to the server.</p> <p>Used for POST requests</p>
<code>setRequestHeader()</code>	Adds a label/value pair to the header to be sent

# XMLHttpRequest Object Properties

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the <a href="#">Http Messages Reference</a>
statusText	Returns the status-text (e.g. "OK" or "Not Found")



```
function showResult(str) {  
    var xmlhttp=new XMLHttpRequest();  
  
    xmlhttp.onreadystatechange=function() {  
        if (this.readyState==4 && this.status==200) {  
            document.getElementById("txtHint").innerHTML = this.responseText;  
        }  
    }  
    xmlhttp.open("GET","livesearch.php?q="+str,true);  
    xmlhttp.send();  
}
```

The background is a light cream color, decorated with several abstract elements. In the top left, there is a horizontal red brushstroke and a cluster of small black dots. On the left side, there is a large, textured grey circle with a black starburst pattern above it. In the bottom left, there is a pink brushstroke and a cluster of small black dots. On the right side, there is a large, textured yellow circle, a thin black wavy line with two small yellow circles attached to it, and a grey brushstroke at the bottom right.

# JSON





# JSON

- AJAX dapat digunakan untuk komunikasi interaktif dengan file XML atau file JSON. Namun saat ini lebih sering menggunakan JSON.
- JSON adalah singkatan dari JavaScript Object Notation, dan merupakan sintaks untuk menyimpan dan bertukar data.
- JSON tidak bergantung pada bahasa pemrograman yang digunakan (language independent). Sintaks JSON diturunkan dari notasi objek JavaScript, tetapi format JSON hanya berupa teks.

## JSON

```
{ "name": "John" }
```

## JavaScript

```
{ name: "John" }
```

- Karena format JSON adalah format berbasis teks, JSON dapat dengan mudah dikirim ke dan dari server, dan digunakan sebagai format data oleh bahasa pemrograman apa pun.



# JSON VS XML

Baik JSON dan XML dapat digunakan untuk menerima data dari web server.

## PERSAMAAN

- Baik JSON dan XML "self describing" (dapat dibaca manusia)
- Baik JSON dan XML bersifat hierarkis (nilai di dalam nilai)
- Baik JSON dan XML dapat diuraikan dan digunakan oleh banyak bahasa pemrograman
- Baik JSON dan XML dapat diambil (fetched) dengan XMLHttpRequest

## PERBEDAAN

- JSON tidak menggunakan tag akhir
- Penulisan JSON lebih pendek
- JSON lebih cepat untuk dibaca dan ditulis
- JSON dapat menggunakan array
- XML harus diurai (parsed) dengan parser XML. JSON dapat diuraikan (parsed) oleh fungsi JavaScript standar.

## Contoh JSON

```
{ "employees": [  
  { "firstName": "John", "lastName": "Doe" },  
  { "firstName": "Anna", "lastName": "Smith" },  
  { "firstName": "Peter", "lastName": "Jones" }  
]}
```

## Contoh XML

```
<employees>  
  <employee>  
    <firstName>John</firstName> <lastName>Doe</lastName>  
  </employee>  
  <employee>  
    <firstName>Anna</firstName> <lastName>Smith</lastName>  
  </employee>  
  <employee>  
    <firstName>Peter</firstName> <lastName>Jones</lastName>  
  </employee>  
</employees>
```

```
{
  "data": [
    {
      "id": 1,
      "kode_awal": "08140",
      "reg_area": "Jakarta ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 2,
      "kode_awal": "081410",
      "reg_area": "Jakarta ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 3,
      "kode_awal": "081411",
      "reg_area": "Jakarta ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 4,
      "kode_awal": "081420",
      "reg_area": "Balikpapan ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 5,
      "kode_awal": "0814205",
      "reg_area": "Samarinda ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 6,
      "kode_awal": "081421",
      "reg_area": "Banjarmasin ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 7,
      "kode_awal": "081422",
      "reg_area": "Pontianak ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 8,
      "kode_awal": "081423",
      "reg_area": "Manado ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 9,
      "kode_awal": "081424",
      "reg_area": "Palu ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 10,
      "kode_awal": "0814245",
      "reg_area": "Kendari ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 11,
      "kode_awal": "081425",
      "reg_area": "Makassar ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 12,
      "kode_awal": "081426",
      "reg_area": "Ambon ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 13,
      "kode_awal": "081427",
      "reg_area": "Jayapura ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 14,
      "kode_awal": "081428",
      "reg_area": "Palangkaraya ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 15,
      "kode_awal": "081430",
      "reg_area": "Medan ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 16,
      "kode_awal": "081434",
      "reg_area": "Banda Aceh ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 17,
      "kode_awal": "081435",
      "reg_area": "Padang ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 18,
      "kode_awal": "081436",
      "reg_area": "Batam ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 19,
      "kode_awal": "081437",
      "reg_area": "Pekanbaru ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 20,
      "kode_awal": "081438",
      "reg_area": "Palembang ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 21,
      "kode_awal": "081439",
      "reg_area": "Bengkulu ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 22,
      "kode_awal": "0814395",
      "reg_area": "Jambi ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 23,
      "kode_awal": "081440",
      "reg_area": "Bandarlampung ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 24,
      "kode_awal": "081450",
      "reg_area": "Surabaya ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 25,
      "kode_awal": "081455",
      "reg_area": "Malang ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 26,
      "kode_awal": "081456",
      "reg_area": "Madiun ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 27,
      "kode_awal": "081457",
      "reg_area": "Denpasar ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 28,
      "kode_awal": "081458",
      "reg_area": "Negeri ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 29,
      "kode_awal": "081459",
      "reg_area": "Jember ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 30,
      "kode_awal": "081460",
      "reg_area": "Barong ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 31,
      "kode_awal": "081463",
      "reg_area": "Sukabumi ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 32,
      "kode_awal": "081464",
      "reg_area": "Sukoharjo ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 33,
      "kode_awal": "081465",
      "reg_area": "Semarang ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 34,
      "kode_awal": "081467",
      "reg_area": "Yogyakarta ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    },
    {
      "id": 35,
      "kode_awal": "081468",
      "reg_area": "Yogyakarta ",
      "kartu": "Indosat M2 ",
      "provider": "INDOSAT"
    }
  ]
}
```



# JSON PARSE (JAVASCRIPT)

- Penggunaan umum JSON: pertukaran data ke/dari web server.
- Ketika client menerima JSON data dari web server, data selalu berbentuk string.
- Parse data menjadi objek JavaScript dengan **JSON.parse**
- Misalkan kita menerima data JSON berikut dari server:

```
'{"name":"John", "age":30, "city":"New York"}'
```

- Terlebih dahulu konversikan menjadi objek JavaScript agar dapat digunakan di client

```
<script>  
const myObj = JSON.parse(myJSON);  
document.getElementById("demo").innerHTML = myObj.name;  
document.getElementById("demo2").innerHTML = myObj["age"];  
</script>
```

- Jika data JSON yang diterima berisi JSON Array, maka parse data menggunakan JSON.parse() akan menghasilkan Array Javascript.

```
'["Ford", "BMW", "Fiat"]'
```



# JSON STRINGIFY (JAVASCRIPT)

- Ketika client mengirim JSON data ke web server, data harus berbentuk string.
- Gunakan fungsi **JSON.stringify()** untuk konversi data ke string dan siap dikirim.
- Misalkan kita memiliki object JavaScript berikut:

```
{name: "John", age: 30, city: "New York"};
```

- Terlebih dahulu kita konversikan menjadi string agar dapat dikirimkan ke server:

```
<script>  
const obj = {name: "John", age: 30, city: "New York"};  
const myJSON = JSON.stringify(obj);  
</script>
```

- JSON.stringify() juga dapat digunakan pada Array Javascript.



# JSON OBJECT LITERALS

- This is a JSON string:

```
'{"name":"John", "age":30, "car":null}'
```

- Inside the JSON string there is a JSON object literal. JSON object literals are surrounded by curly braces {}.

```
{"name":"John", "age":30, "car":null}
```

Key.  
Harus String

Value.

Tipe data value:

- string
- number
- object
- array
- boolean
- null

- Keys dan values dipisahkan dengan titik dua.
- Antar pasangan key-values dipisahkan dengan koma
- Dari JSON object literal dapat dibuat Javascript object menggunakan fungsi `JSON.parse()`
- JSON values cannot be one of the following data types: function, date, undefined

# Looping an Object: **for-in** Loop

```
'{"name":"John", "age":30, "car":null}'
```

```
<!DOCTYPE html>
<html>
<body>

<h2>Looping Object Properties</h2>
<p id="demo"></p>

<script>
const myObj = JSON.parse(myJSON);
let text = "";
for (const x in myObj) {
  text += x + ", ";
}
document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Output: name, age, car,

```
<!DOCTYPE html>
<html>
<body>

<h2>Looping JavaScript Object Values</h2>
<p id="demo"></p>

<script>
const myObj = JSON.parse(myJSON);
let text = "";
for (const x in myObj) {
  text += myObj[x] + ", ";
}
document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Output: John, 30, null,





# JSON ARRAY LITERALS

- This is a JSON string:

```
'["Ford", "BMW", "Fiat"]'
```

- Inside the JSON string there is a JSON array literal:

```
["Ford", "BMW", "Fiat"]
```

- Sama seperti JSON Object literals, dari JSON array literals dapat dibuat Javascript array menggunakan fungsi JSON.parse()
- Arrays in JSON are almost the same as arrays in JavaScript.
- In JSON, array values must be of type string, number, object, array, boolean or null.
- In JavaScript, array values can be all of the above, plus any other valid JavaScript expression, including functions, dates, and undefined.

# Mengakses Array Values

```
'["Ford", "BMW", "Fiat"]';
```

```
const myArray = JSON.parse(myJSON);  
document.getElementById("demo").innerHTML = myArray[0];
```

Output: Ford

```
'{"name":"John", "age":30, "cars":["Ford", "BMW", "Fiat"]}';
```

```
const myObj = JSON.parse(myJSON);  
document.getElementById("demo").innerHTML = myObj.cars[0];
```

Output: Ford

```
const myObj = JSON.parse(myJSON)  
let text = "";  
for (let i in myObj.cars) {  
  text += myObj.cars[i] + ", ";  
}  
document.getElementById("demo").innerHTML = text;
```

Output: Ford, BMW, Fiat,

```
const myObj = JSON.parse(myJSON);  
let text = "";  
for (let i = 0; i < myObj.cars.length; i++) {  
  text += myObj.cars[i] + ", ";  
}  
document.getElementById("demo").innerHTML = text;
```

Output: Ford, BMW, Fiat,



# SUMMARY

- Jika terdapat data dalam bentuk JavaScript object atau array pada client, data tersebut bisa dikonversi menjadi object literal atau array literal dalam JSON, menggunakan JSON Stringify, lalu dikirimkan ke server.
- Jika client menerima data dalam format JSON format dari server, data tsb juga dapat dikonversi dengan mudah menjadi JavaScript object atau array menggunakan JSON Parse, lalu diproses seperti JavaScript objek atau array biasa.
- Client dapat merequest JSON dari server dengan menggunakan **AJAX Request**.
- PHP pun memiliki built-in functions untuk handle JSON:
  - **json\_encode()** digunakan untuk mengencode nilai ke format JSON.
  - **json\_decode()** digunakan untuk mengkonversi atau mengubah objek/array dalam JSON menjadi PHP objek atau PHP array asosiatif.



# Request JSON from Server Using AJAX

```
keyword.addEventListener('keyup', function() {
    var xmlhttp = new XMLHttpRequest();

    xmlhttp.onreadystatechange = function() {
        if(xmlhttp.readyState== 4 && xmlhttp.status==200) {
            myObj = JSON.parse(this.responseText);
            let text = "";
            for (let x in myObj) {
                text += myObj[x].name + "<br>";
            }
            document.getElementById("demo").innerHTML = text;
        }
    }

    xmlhttp.open('GET', 'dbconn_customers_byname.php?keyword='+keyword.value, true);
    xmlhttp.send();
})
```

# PHP Functions to Handle JSON

```
$conn = new PDO("mysql:host=$servername;dbname=$db_name",$username,$password);
$conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION,
| | | | | PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC);

$name=$_GET["keyword"];
$sql="SELECT * FROM customers WHERE name LIKE '%$name%' ";
$result = $conn->query($sql);
$rows = $result->fetchAll();

echo json_encode($rows);
```

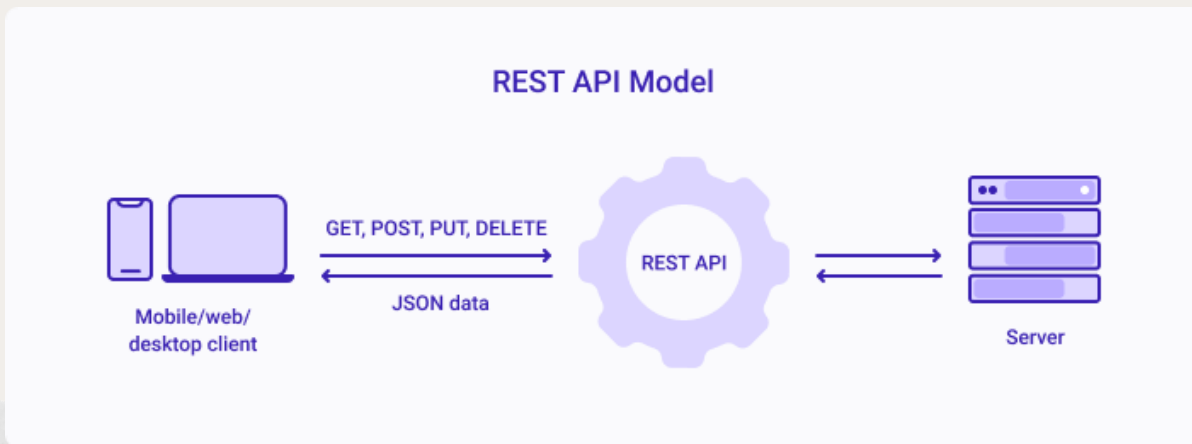
The background is a light cream color, decorated with several abstract elements. In the top left, there is a horizontal red brushstroke and a cluster of small black dots. On the left side, there is a large, textured grey circle with a black starburst pattern above it. In the bottom left, there is a pink brushstroke and a cluster of small black dots. On the right side, there is a large, textured yellow circle, a thin black wavy line, and a small yellow circle. In the bottom right, there is a grey brushstroke and a small yellow circle.

API



# What is API?

- Application Programming Interface (API): a contract (or specification) which allows two or more software applications to talk to each other through a well-defined computing interface.
- We are not required to know how an API works internally, but all we will need to know is how to interact with the API





# API Contract

- An API contract is the documentation that describes how the API works and how it should be used.
- API contract includes, at least:
  - Endpoints urls,
  - Actions of each endpoint,
  - Arguments,
  - Examples of the responses

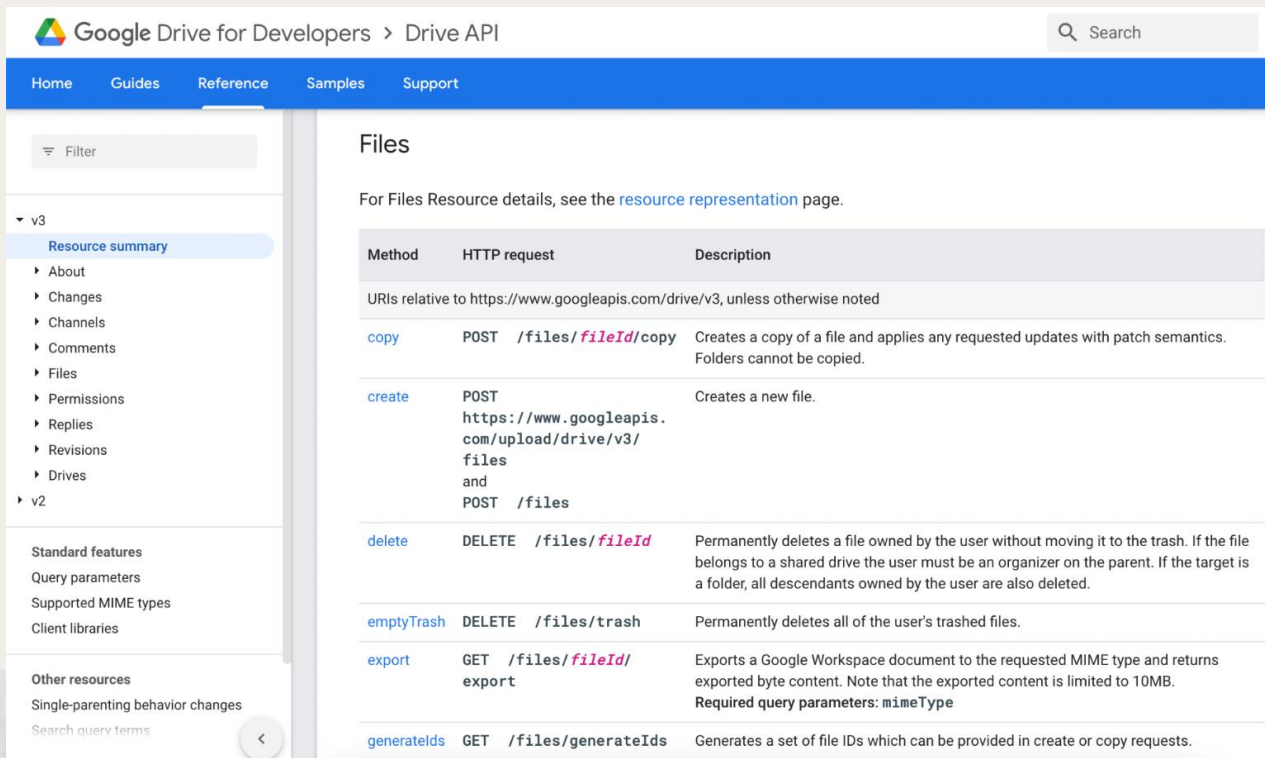
Imagine we want to order goods on the marketplace:

- ▷ What endpoint should be accessed? (tokopedia, Bukalapak, shopee)
- ▷ What arguments should be included? (item name, item quantity, seller name, shipping options, payment method)
- ▷ What is the expected response? (OK, out of stock, item not available, seller is on holiday, etc.)



# Real Life Example: Google APIs

- Google Drive's API (<https://developers.google.com/drive/api/v3/reference> )



The screenshot shows the Google Drive for Developers API reference page for the v3 Files resource. The page has a blue header with navigation links: Home, Guides, Reference (selected), Samples, and Support. A search bar is in the top right. On the left, a sidebar shows the v3 API structure with 'Resource summary' selected, listing various resources like About, Changes, Channels, Comments, Files, Permissions, Replies, Revisions, and Drives. Below this are sections for 'Standard features' (Query parameters, Supported MIME types, Client libraries) and 'Other resources' (Single-parenting behavior changes, Search query terms). The main content area is titled 'Files' and includes a note: 'For Files Resource details, see the [resource representation](#) page.' Below this is a table listing API methods.

Method	HTTP request	Description
URIs relative to <a href="https://www.googleapis.com/drive/v3">https://www.googleapis.com/drive/v3</a> , unless otherwise noted		
<a href="#">copy</a>	POST <code>/files/<i>fileId</i>/copy</code>	Creates a copy of a file and applies any requested updates with patch semantics. Folders cannot be copied.
<a href="#">create</a>	POST <code>https://www.googleapis.com/upload/drive/v3/files</code> and POST <code>/files</code>	Creates a new file.
<a href="#">delete</a>	DELETE <code>/files/<i>fileId</i></code>	Permanently deletes a file owned by the user without moving it to the trash. If the file belongs to a shared drive the user must be an organizer on the parent. If the target is a folder, all descendants owned by the user are also deleted.
<a href="#">emptyTrash</a>	DELETE <code>/files/trash</code>	Permanently deletes all of the user's trashed files.
<a href="#">export</a>	GET <code>/files/<i>fileId</i>/export</code>	Exports a Google Workspace document to the requested MIME type and returns exported byte content. Note that the exported content is limited to 10MB. <b>Required query parameters:</b> <code>mimeType</code>
<a href="#">generateIds</a>	GET <code>/files/generateIds</code>	Generates a set of file IDs which can be provided in create or copy requests.



# Real Life Example: Public API in Indonesia

- <https://github.com/farizdotid/DAFTAR-API-LOKAL-INDONESIA>



# Accessing API

- Using a specific HTTP method on a particular type of call
- Equivalent to CRUD (create, read, update, delete) operations
  - POST method → create operation
  - GET method → read operation
  - PUT or PATCH method → update operation
    - PUT → update entirely
    - PATCH → update partially
  - DELETE method → delete operation



# Example

Access Public API from: <https://reqres.in/>

<https://reqres.in/api/users?page=1>

```
{
  "page": 1,
  "per_page": 6,
  "total": 12,
  "total_pages": 2,
  "data": [
    {
      "id": 1,
      "email": "george.bluth@reqres.in",
      "first_name": "George",
      "last_name": "Bluth",
      "avatar": "https://reqres.in/img/faces/1-image.jpg"
    },
  ],
}
```

```
{
  "id": 2,
  "email": "janet.weaver@reqres.in",
  "first_name": "Janet",
  "last_name": "Weaver",
  "avatar": "https://reqres.in/img/faces/2-image.jpg"
},
{
  "id": 3,
  "email": "emma.wong@reqres.in",
  "first_name": "Emma",
  "last_name": "Wong",
  "avatar": "https://reqres.in/img/faces/3-image.jpg"
},
}
```



# GET

GET



<https://reqres.in/api/users/2>

Send



```
{
  "data": {
    "id": 2,
    "email": "janet.weaver@reqres.in",
    "first_name": "Janet",
    "last_name": "Weaver",
    "avatar": "https://reqres.in/img/faces/2-image.jpg"
  },
  "support": {
    "url": "https://reqres.in/#support-heading",
    "text": "To keep ReqRes free, contributions towards server costs are appreciated!"
  }
}
```



# POST

POST

https://reqres.in/api/users

Send

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettingsCookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	name	Iya			
<input checked="" type="checkbox"/>	job	lecturer			

```
{
  "id": "165",
  "createdAt": "2023-05-29T02:03:55.854Z"
}
```



# POST

POST

▼

https://reqres.in/api/users

Send

▼

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

☐ none

☒ form-data

☐ x-www-form-urlencoded

☐ raw

☐ binary

☐ GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	name	Iya			
<input checked="" type="checkbox"/>	job	lecturer			
<input checked="" type="checkbox"/>	photo	File ▼ Select Files			
	Key	Text Value	Description		
		File			



# POST

POST

https://reqres.in/api/users

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON

Beautify

1

{

2

"name": "lya",

3

"job": "lecturer"

4

}

```
{
  "name": "lya",
  "job": "lecturer",
  "id": "803",
  "createdAt": "2023-05-29T02:25:23.839Z"
}
```



# PUT

PUT ▼ https://reqres.in/api/users/2 Send ▼

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼ Beautify

```
1 {  
2   ... "name": "lya",  
3   ... "job": "lecturer"  
4 }
```

```
{  
  "name": "lya",  
  "job": "lecturer",  
  "updatedAt": "2023-05-29T02:24:26.310Z"  
}
```



# PATCH

PATCH

https://reqres.in/api/users/2

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON

1

{

2

"name": "lya"

3

}

Beautify

```
{  
  "name": "lya",  
  "updatedAt": "2023-05-29T02:39:07.716Z"  
}
```



# DELETE

DELETE



https://reqres.in/api/users/2

Send



204 No Content



# REFERENSI

Materi disadur dari:

1. COMP519 Web Programming (2020-21). Diakses pada Januari 2021, dari <https://cgi.csc.liv.ac.uk/~ullrich/COMP519/>
2. W3Schools Online Web Tutorials. Diakses pada Januari 2021, dari <https://www.w3schools.com/>
3. <https://restfulapi.net/>
4. Web programming UNPAS youtube channel:  
<https://youtube.com/playlist?list=PLFIM0718LjIW7AsIbnhFg15t9yx4H-sQ0>

# Thanks!

Do you have any questions?



wilantika@stis.ac.id

lya@stis.ac.id

yeni.rima@stis.ac.id

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik