

# JSON CheatSheet

(in cybersecurity terms)

{JSON}

JavaScript Object Notation





## JSON cheat sheet (in cybersecurity terms)



تهیه شده توسط تیم TryHackBox

The Chaos

لينك زير جهت حمايت از جامعه آموزش رايگان :

<https://daramet.com/TryHackBox>



ما را به دوستانتان معرفی کنید.

دیگر کانال ها و شبکه های اجتماعی ما را دنبال کنید:

کانال های تلگرام ما

آموزش تست نفوذ و Red Team

[@TryHackBox](#)

رودمپ های مختلف:

[@TryHackBoxOfficial](#)

داستان های هک:

[@TryHackBoxStory](#)

آموزش برنامه نویسی:

[@TryCodeBox](#)

رادیو زیروپاد ( پادکست ها ) :

[@RadioZeroPod](#)

ایнстاستاگرام :

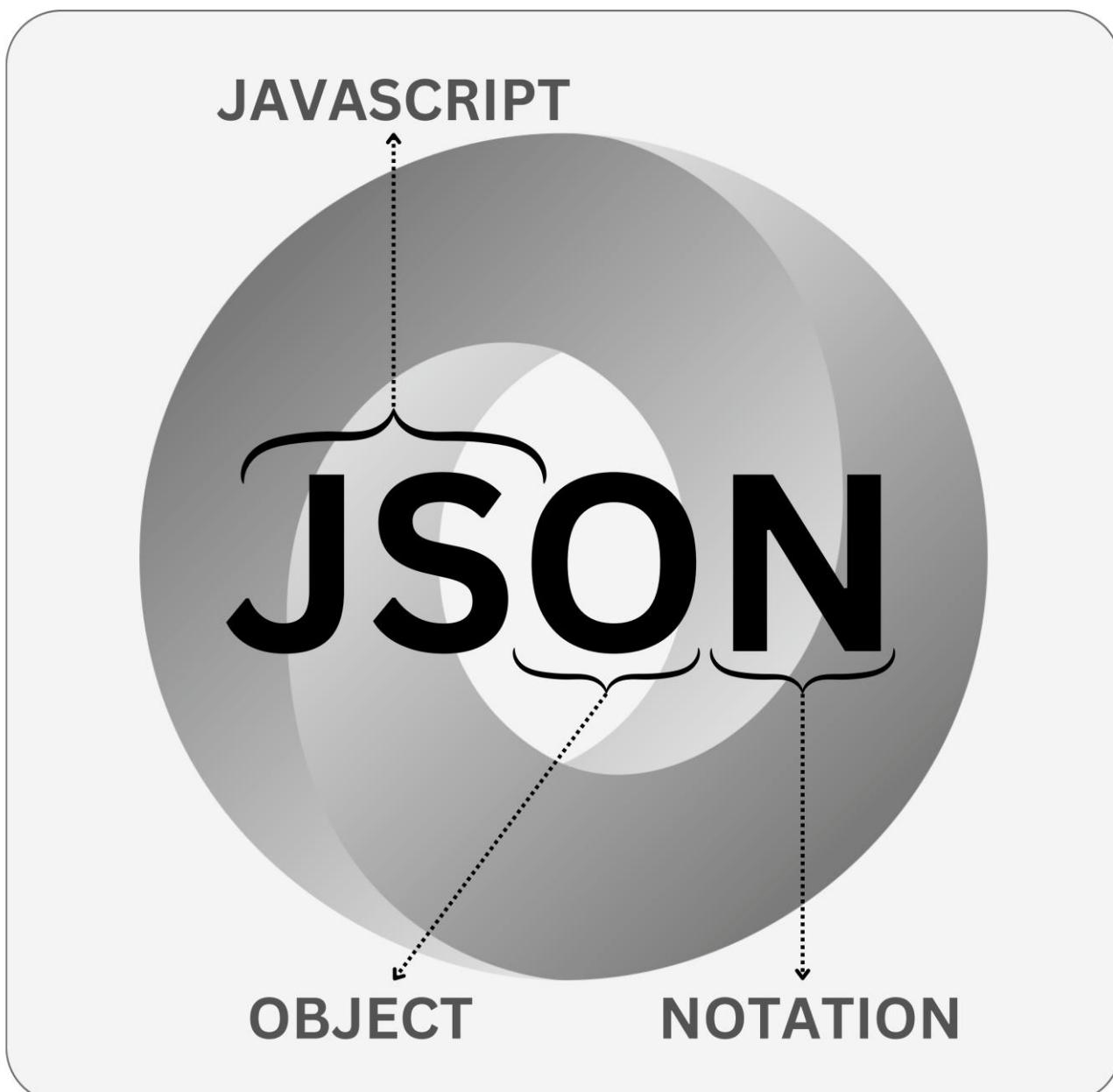
<http://www.instagram.com/TryHackBox>

یوتیوب:

<https://youtube.com/@tryhackbox>



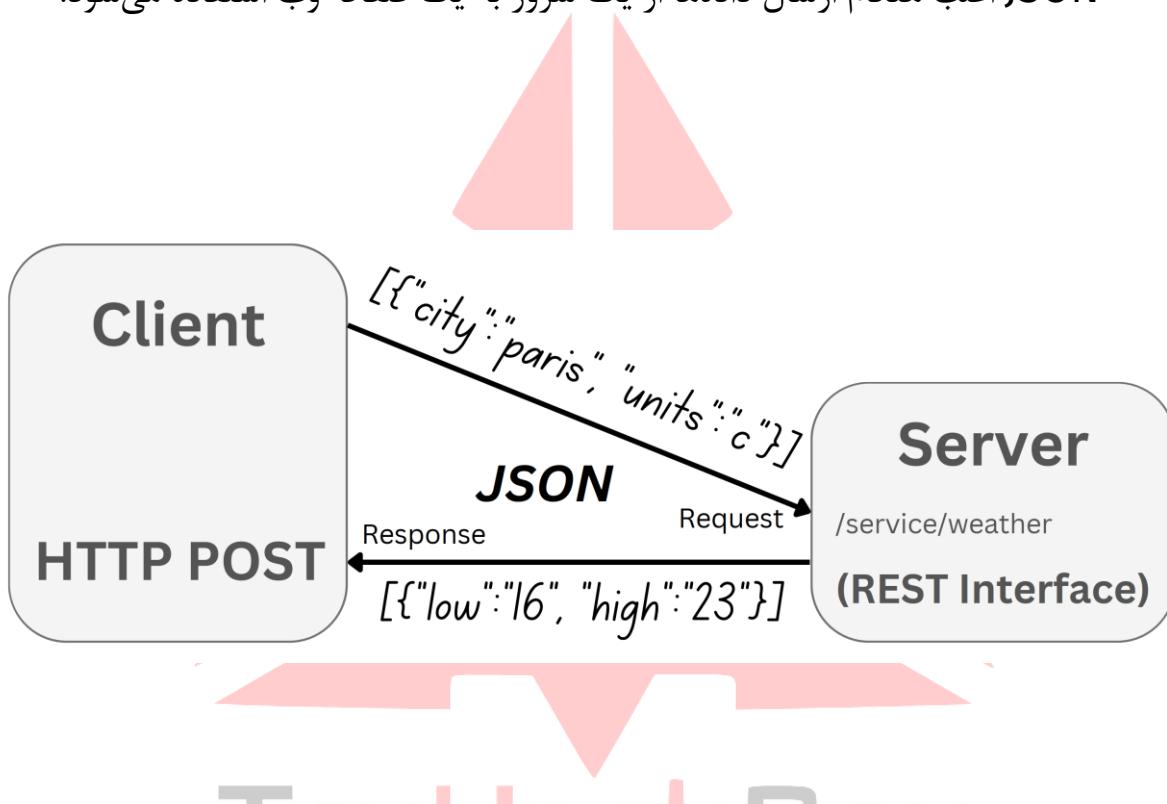
## I. مقدمه مفهومی گرافیکی بر JSON





## خلاصه کلامی JSON

- JSON مخفف عبارت JavaScript Object Notation است.
- JSON یک فرمت سبک برای ذخیره و انتقال داده‌ها است.
- JSON "خدودوصیف‌گر" و قابل فهم است.
- JSON اغلب هنگام ارسال داده‌ها از یک سرور به یک صفحه وب استفاده می‌شود.



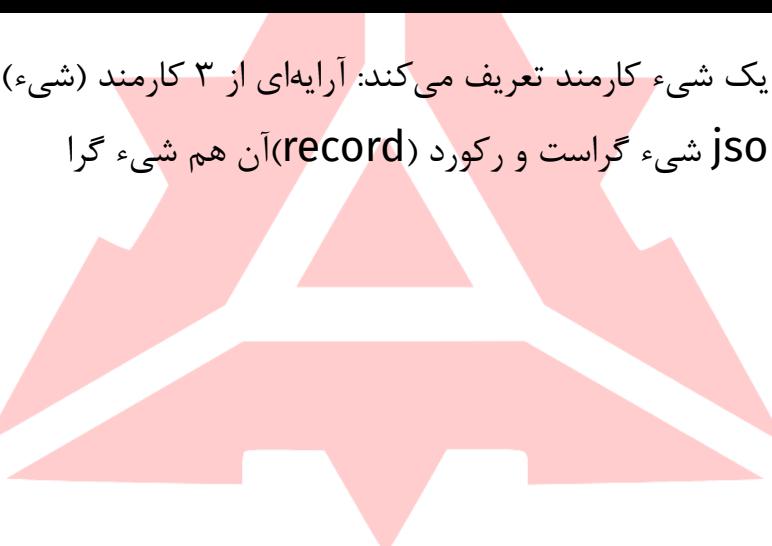
TryHackBox



## خلاصه مثالی از JSON

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

- این مثال یک شیء کارمند تعریف می‌کند: آرایه‌ای از ۳ کارمند (شیء)
- چرا که json شیء گر است و رکورد (record) آن هم شیء گرا



TryHackBox



## خلاصه JSON در KEY AND VALUES

دو بخش اصلی که JSON را تشکیل می‌دهند، کلیدها و مقادیر هستند. این دو با هم یک جفت کلید/مقدار را تشکیل می‌دهند:

- کلید: یک کلید همیشه یک رشته است که در علامت نقل قول قرار می‌گیرد.
- مقدار: یک مقدار می‌تواند یک رشته، عدد، عبارت بولی، آرایه یا شیء باشد.



کلید «name» و مقدار «aditya» است.

TryHackBox



## خلاصه JSON.parse()

هنگام دریافت داده از یک وب سرور، داده همیشه یک رشته است.

تصور کنید که ما این متن را از یک وب سرور دریافت کردیم :

```
'{ "name": "Aditya", "age": 19, "country": "India"}'
```

از تابع جاوا اسکریپت JSON.parse() برای تبدیل متن به یک شیء جاوا اسکریپت استفاده کنید :

```
var obj = JSON.parse('{"name": "Aditya", "age": 19, "country": "India"}');
```

TryHackBox

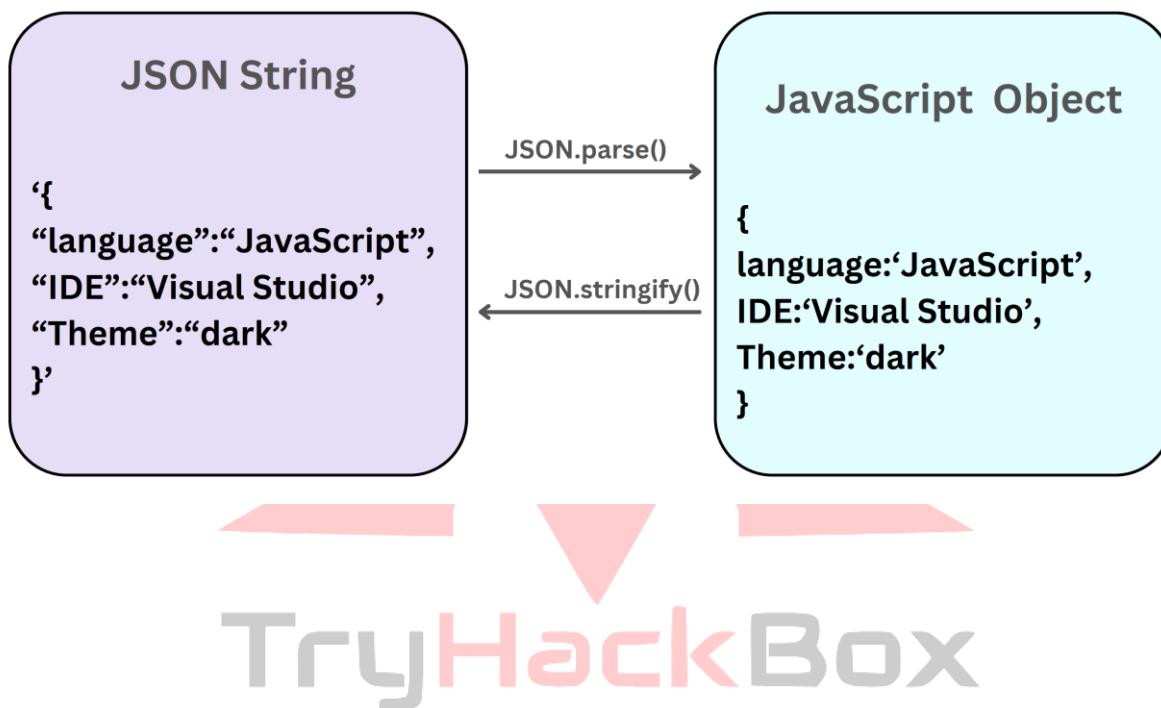


## خلاصه JSON. stringify()

متده (`JSON.stringify()`) یک شیء یا مقدار جاوا اسکریپت را به یک رشته JSON تبدیل می‌کند، و در صورت مشخص شدن یک تابع جایگزین، مقادیر را به صورت اختیاری جایگزین می‌کند یا در صورت مشخص شدن یک آرایه جایگزین، فقط ویژگی‌های مشخص شده را در بر می‌گیرد.

### SYNTAX

`JSON.stringify(value, replacer, space)`





## خلاصه JSON VALUES

- string
- number
- object (JSON object)
- array
- boolean
- null

یک فرمت داده مینیمال است. تنها با یادگیری چند اصل کلیدی می‌توانید کل یک سایت را رمزگشایی کنید.



TryHackBox



برگه‌ی تقلب

JSON



## II. برگه تقلب JSON

- JavaScript Object Notation Json
- Json خواندن و نوشتنش راحت است.
- Json یک قالب مستقل از زبان برای تبادل داده است.
- Json پسوند فایل‌های "Json" است.
- Json نوعی "application/json" می‌باشد.

### ساختار JSON

- داده‌ها به صورت جفت کلید/متغیر (key/value) ارائه می‌شوند.
- هر جفت را کاما از هم جدا می‌شوند.
- آکولادها {} اجزاء را تعیین می‌کنند.
- براکت‌ها [] آرایه‌ها را مشخص می‌کنند.

```
{  
    "key1": "value1",  
    "key2": "value2",  
    "key3": "value3"  
}
```



```
● ● ●  
{  
    "name": "JSON",  
    "age": 23,  
    "gender": "M",  
    "married": true,  
    "children": [  
        {"name": "foo", "age": 9, "gender": "M"},  
        {"name": "Bar", "age": 7, "gender": "F"}  
    ]  
}  
"key3": "value3"  
}
```

مثال از ساده‌ترین ساختار JSON

## انواع داده در JSON

### Number

یک عدد صحیح در JSON .

مثال : {"age": 30} .

یک عدد اعشاری در JSON .

### String

مقادیر رشته‌ای مجموعه‌ای از کاراکترهای یونیکد به طول صفر یا بیشتر هستند که با دو علامت کوتیشن ("") احاطه می‌شوند.

مثال : {"name": "TryHackBox"}

### Boolean

مقادیر false یا true که برای نمایش درستی یا نادرستی یک مقدار استفاده می‌شوند.

مثال : {"married": false}



## Array

آرایه مجموعه‌ای مرتب از مقادیر است که بین برآکت‌ها [ ] قرار می‌گیرد. مقادیر آرایه می‌توانند شامل اشیاء JSON باشند با ساختار جفت کلید/ مقدار مثال:

```
{  
  "students": [  
    {"firstName": "Fina", "lastName": "Bar"},  
    {"firstName": "Baz", "lastName": "Qux"}  
  ]  
}
```

## Object

مجموعه‌ای از جفت‌های کلید/ مقدار (key/value) که بین آکولادها { } قرار می‌گیرند و با کاما از هم جدا می‌شوند. مثال:

```
{  
  "name": "TryHackBox",  
  "url": "https:// TryHackBox.com"  
}
```

## Null

نشان‌دهنده مقدار خالی یا عدم وجود اطلاعات.  
مثال : {"bloodType": null}



## کاراکترهای خاص در JSON

در JSON ، برخی کاراکترها نیاز به Escape (فار) دارند و با استفاده از بکاصلش (\) نمایش داده می شوند :

### Double Quote (" ) .1

\": Escape Sequence 。

"This is a \"quoted\" string" مثال : 。

### Backslash (\) .2

\\: Escape Sequence 。

"This is a backslash:\\" مثال : 。

### Forward Slash (/) .3

\/: اختياری است، اما توصیه می شود : Escape Sequence 。

"http://\\example.com" مثال : 。

### Backspace (\b) .4

Escape Sequence: \b 。

"Before\bAfter" مثال : 。

### Form Feed (\f) .5

/f: Escape Sequence 。

"Line1\fline2" مثال : 。



## Unicode Character (\u) .6

\uxxxx : Escape Sequence 。

"This is a Unicode character: \u00E9" ۔ مثال :

توضیح: این کاراکتر به همراه رقم هگزادسیمال، یک کاراکتر یونیکد را نمایش می‌دهد.

## Newline (\n) .7

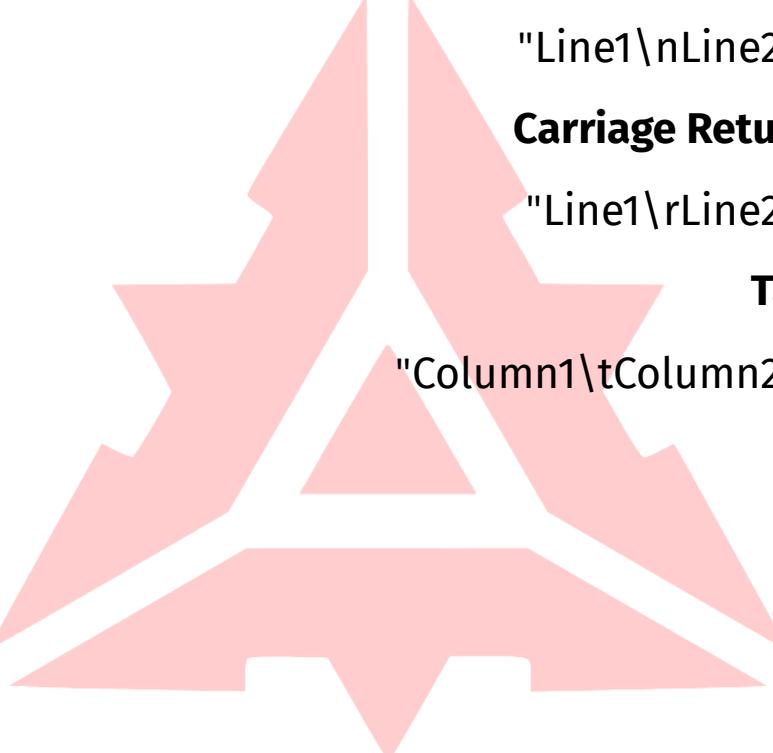
"Line1\nLine2" ۔

## Carriage Return (\r) .8

"Line1\rLine2" ۔

## Tab (\t) .9

"Column1\tColumn2" ۔



TryHackBox

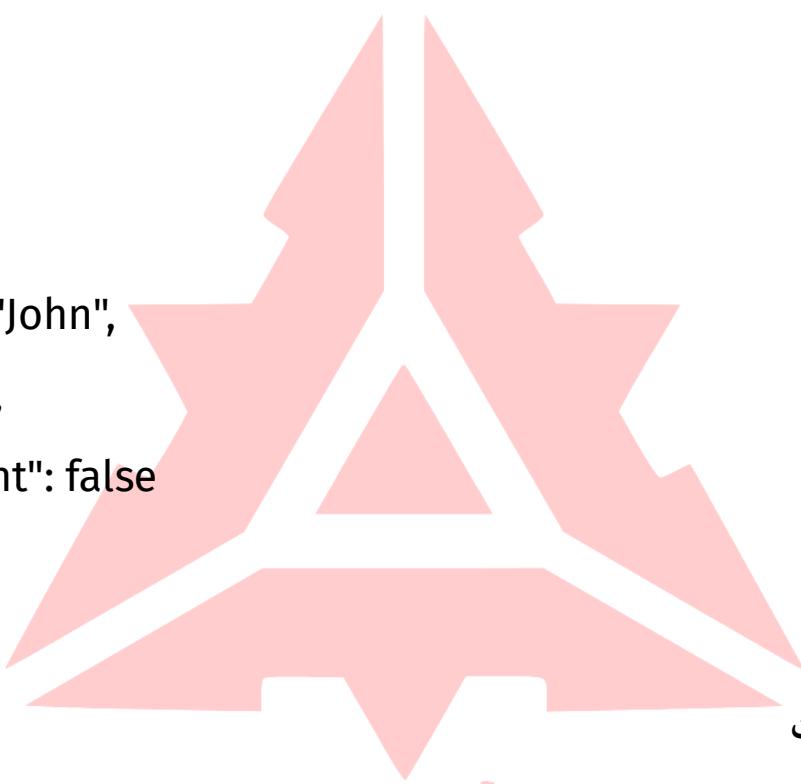


## آبجکت در JSON

- مجموعه‌ای از جفت‌های کلید-مقدار (Key-Value) است.
- بین آکولادهای {} قرار می‌گیرد.
- کلیدها همیشه رشته (String) و در دبل کوپیشن "" هستند.
- مقادیر می‌توانند از هر نوع داده‌ای (عدد، رشته، بولین، آرایه، آبجکت دیگر، یا null) باشند.

مثال ساده:

```
{  
  "name": "John",  
  "age": 30,  
  "isStudent": false  
}
```



ساختار آبجکت

**TryHackBox**

- کلیدها باید منحصر به فرد باشند.
- مقادیر با کاما از هم جدا می‌شوند.
- تورفت (Indent) اختیاری است (برای خوانایی بهتر).



## مثال با انواع داده‌ها:

```
{  
  "name": "Alice",           // String  
  "age": 25,                 // Number  
  "isActive": true,          // Boolean  
  "courses": ["Math", "Science"], // Array  
  "address": {                // Nested Object  
    "city": "Tehran",  
    "postalCode": "12345"  
  },  
  "notes": null               // Null  
}
```

## (Nested Objects) آبجکت‌های تو در تو

- یک آبجکت می‌تواند داخل آبجکت دیگر باشد.

مثال:

```
json  
{  
  "employee": {  
    "name": "Ali",  
    "department": {  
      "name": "IT",  
      "floor": 3  
    }  
  }  
}
```

## دسترسی به مقادیر آبجکت

- در زبان‌های برنامه‌نویسی با استفاده از:

object.key (.) مانند نقطه .

براکت و نام کلید مانند [ ] object["key"] مانند.



## مثال در JavaScript

```
● ● ●  
  
const person = {  
    "name": "John",  
    "age": 30  
};  
console.log(person.name); // خروجی: John  
console.log(person["age"]); // خروجی: 30
```

## کاربردهای رایج

- ذخیره‌سازی داده‌های ساختار یافته مثل پروفایل کاربران.
- تبادل داده بین سرور و کلاینت (API Responses)
- پیکربندی تنظیمات نرم افزارها.

## مثال API Response

```
● ● ●  
  
{  
    "status": "success",  
    "data": {  
        "id": 1,  
        "title": "JSON Tutorial"  
    }  
}
```



## خطاهای رایج

- فراموش کردن دبل کوئیشن برای کلیدها

✗ { name: "John" }  
✓ { "name": "John" }

- استفاده از کامای اضافی در انتهای

✗ { "a": 1, "b": 2, }  
✓ { "a": 1, "b": 2 }

ابزارهای مفید

- اعتبارسنجی JSON: [jsonlint.com](http://jsonlint.com)

- فرماتدهنده (Formatter): [jsonformatter.org](http://jsonformatter.org)

TryHackBox



## آرایه در JSON

- مجموعه‌ای مرتب از مقادیر (**Values**) است.

- بین براکت‌های [] قرار می‌گیرد.

- مقادیر می‌توانند از هر نوع داده‌ای عدد، رشته، بولین، آبجکت، آرایه دیگر، یا null باشند.

- مقادیر با کاما ، از هم جدا می‌شوند.

مثال ساده:

```
["apple", "banana", "cherry"]
```

ساختار آرایه

- ترتیب عناصر مهم است برخلاف آبجکت‌ها که کلیدها **unordered** هستند

- ایندیکس (**Index**) از صفر شروع می‌شود.

مثال با انواع داده‌ها:

```
● ○ ●
[
  42,           // Number
  "Hello",      // String
  true,          // Boolean
  null,          // Null
  { "name": "John" }, // Object
  [1, 2, 3]       // Nested Array
]
```



## آرایه‌های تو در تو (Nested Arrays)

- یک آرایه می‌تواند شامل آرایه‌های دیگر باشد.

مثال:

```
[  
  [1, 2],  
  ["a", "b"],  
  [true, false]  
]
```

آرایه از آبجکت‌ها

- پر کاربرد در API‌ها و ذخیره‌سازی داده‌های ساختاری‌افته.

مثال:

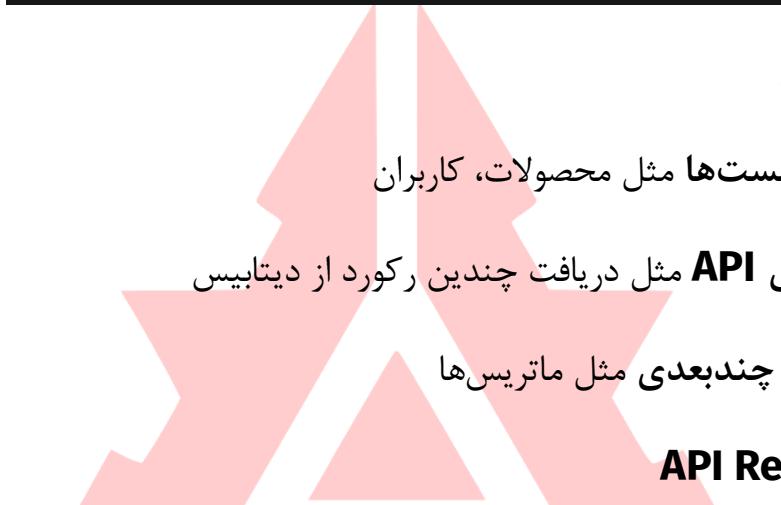
```
[  
  { "id": 1, "name": "Alice" },  
  { "id": 2, "name": "Bob" }  
]
```



## دسترسی به مقادیر آرایه

- در زبان‌های برنامه‌نویسی با اندیکس قابل دسترسی است.

### مثال در JavaScript



```
const fruits = ["apple", "banana", "cherry"];
console.log(fruits[0]); // خروجی: apple
```

### کاربردهای رایج

- ذخیره لیست‌ها مثل محصولات، کاربران
- پاسخ‌های API مثل دریافت چندین رکورد از دیتابیس
- داده‌های چندبعدی مثل ماتریس‌ها

### مثال API Response



```
{
  "status": "success",
  "data": [
    { "id": 1, "title": "Book A" },
    { "id": 2, "title": "Book B" }
  ]
}
```



## خطاهای رایج

- کامای اضافی در انتهای

✗ [1, 2, 3,]

✓ [1, 2, 3]

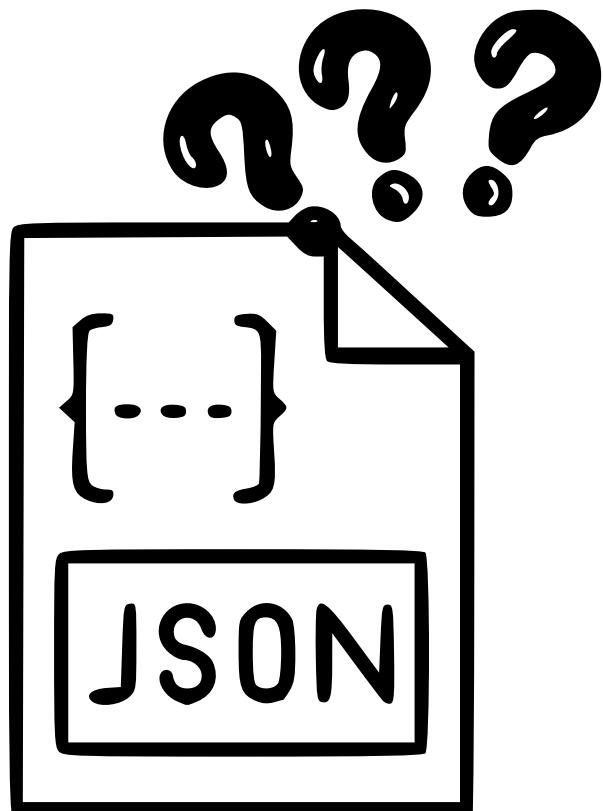
- فراموش کردن براکت‌ها

✗ "fruits": "apple", "banana"  
✓ "fruits": ["apple", "banana"]

ابزارهای مفید

- اعتبارسنجی JSON: [jsonlint.com](http://jsonlint.com)
- به آبجکت JSON در JavaScript تبدیل JSON.parse()

TryHackBox



**EXPLOIT FROM JSON**



### III. اهمیت JSON در هک، امنیت و مدیریت داده‌ها

فرمت اصلی ارتباط داده بین کلاینت و سرور JSON (JavaScript Object Notation) در بسیاری از API‌ها، اپلیکیشن‌های وب، موبایل و حتی سیستم‌های IoT هست. برای هکرهای اخلاقی، تحلیل‌گران امنیتی و مهندسان معکوس، توانایی درک و دستکاری JSON بسیار حیاتی است.

چرا JSON برای هکرهای مهم است؟

**1. بررسی پاسخ‌های API** : بسیاری از API‌ها، مخصوصاً RESTful API‌ها، از JSON برای بازگشت داده استفاده می‌کنند. بررسی این پاسخ‌ها اطلاعات حساسی را افشا می‌کند، مانند:

\* توکن‌ها

\* شناسه‌های کاربران

\* مسیرهای مخفی

\* داده‌های کانفیگ

**2. دستکاری داده‌ها برای تست امنیت**: در تست نفوذ، JSON معمولاً به عنوان ورودی سمت کلاینت ارسال می‌شود. تست امنیت می‌تواند:

\* داده‌ها را تزریق کند (SQLi, XSS, etc)

\* ساختار را دستکاری کند

\* مقادیر را تغییر دهد تا رفتار سیستم بررسی شود

**3. استفاده در Json** : JWT (JSON Web Token) است که در احراز هویت نقش حیاتی دارد. درک فرمت JSON کمک می‌کند تا آسیب‌پذیری‌هایی مانند Token Forgery یا None Algorithm Bypass شوند.



## انواع حملات با JSON

### JSON Injection

- تزریق مقادیر در فیلدهای JSON برای بررسی نحوه اعتبارسنجی و فیلتر دادهها.
- ابزار مفید : Burp Suite + JSON Beautifier + Intruder
- مثال : `{"user":"admin","pass":"1234'||'1=='1"}`

### Testing JSON-based API Endpoints

- بررسی POST, PUT, PATCH endpoint های API با بارهای مخرب برای تست فیلترینگ سرور.
- ابزار مفید : Postman, Fiddler, curl
- بررسی فیلدهای پنهان یا مجاز از طریق:
  - اضافه کردن فیلد جدید
  - حذف فیلدهای ضروری
  - تغییر نوع داده (type confusion)

### JWT Analysis & Manipulation

- بررسی ساختار token ها (header.payload.signature)
- تست موارد زیر:
  - تغییر الگوریتم به none
  - جعل کلید برای امضای مجدد
  - بررسی افشای کلید در پاسخها

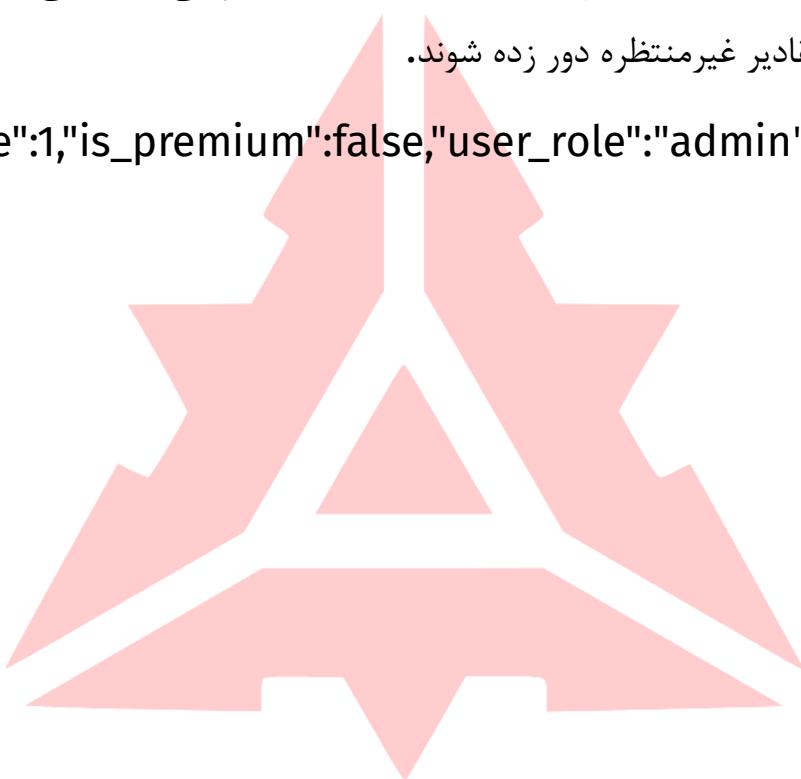


## JSON Parser Attacks

- برخی زبان‌ها و پارسراها در مقابل JSON‌های نادرست یا crafted شده آسیب‌پذیر هستند مثلاً Prototype Pollution
- مثال: `{"proto__":{"admin":true}__"}`

## Logic Abuse via JSON

- سناریوهای تجاری که بر پایه داده‌های JSON تصمیم می‌گیرند، می‌توانند از طریق ارسال مقادیر غیرمنتظره دور زده شوند.
- مثال: `{"price":1,"is_premium":false,"user_role":"admin"}`



TryHackBox



## ابزار های نفوذ با JSON

### جدول ابزارها

| ابزار             | نوع حمله        | مثال کاربرد                               |
|-------------------|-----------------|---|
| <b>Burp Suite</b> | JSON Injection  | تغییر ساختار JSON برای دور زدن احراز هویت |
| <b>Postman</b>    | API Testing     | اتوماسیون حملات Brute Force               |
| <b>jq</b>         | Data Extraction | استخراج ایمیل‌ها و پسورد‌ها از پاسخ‌ها    |
| <b>SQLMap</b>     | SQL Injection   | تزریق SQL از طریق پارامترهای JSON         |
| <b>Node.js</b>    | Deserialization | ساخت Prototype های payload Pollution      |
| <b>Python</b>     | Automation      | ارسال خودکار درخواست‌های مخرب             |
| <b>Wfuzz</b>      | Fuzzing         | پیدا کردن پارامترهای آسیب‌پذیر            |

TryHackBox