

# SER 321 B Session

**SI Session**

**Thursday, March 27th 2025**

*7:00 pm - 8:00 pm MST*

# Agenda



Review HTTP

JSON

Syntax

Structure Overview

Hands-On Practice

# SI Session Expectations

Thanks for coming to the **SER 321** SI session. We have a packed agenda and we are going to try to get through as many of our planned example problems as possible. This session will be recorded and shared with others.

- If after this you want to see additional examples, please visit the drop-in tutoring center.
- We will post the link in the chat now and at the end of the session.
  - [tutoring.asu.edu](https://tutoring.asu.edu)
- Please keep in mind we are recording this session and it will be made available for you to review 24-48 hours after this session concludes.
- Finally, please be respectful to each other during the session.

# Interact with us:

## Zoom Features



### Zoom Chat

- Use the chat feature to interact with the presenter and respond to presenter's questions.
- Annotations are encouraged

We will focus on ***FOUR*** request types

1.

2.

3.

4.

## SER 321

### HTTP Requests

What's the difference?

1. GET

2. POST

3. PUT

4. DELETE

Idempotent

# SER 321

## HTTP Matching

Match the HTTP response code with its meaning:

Code:

1XX

2XX

3XX

4XX

5XX

Meaning:

User Error

Server Error

Information

Redirect

Success

# SER 321

## HTTP Matching

Match the HTTP response code with its meaning:

Code:

1XX

2XX

3XX

4XX

5XX

Meaning:

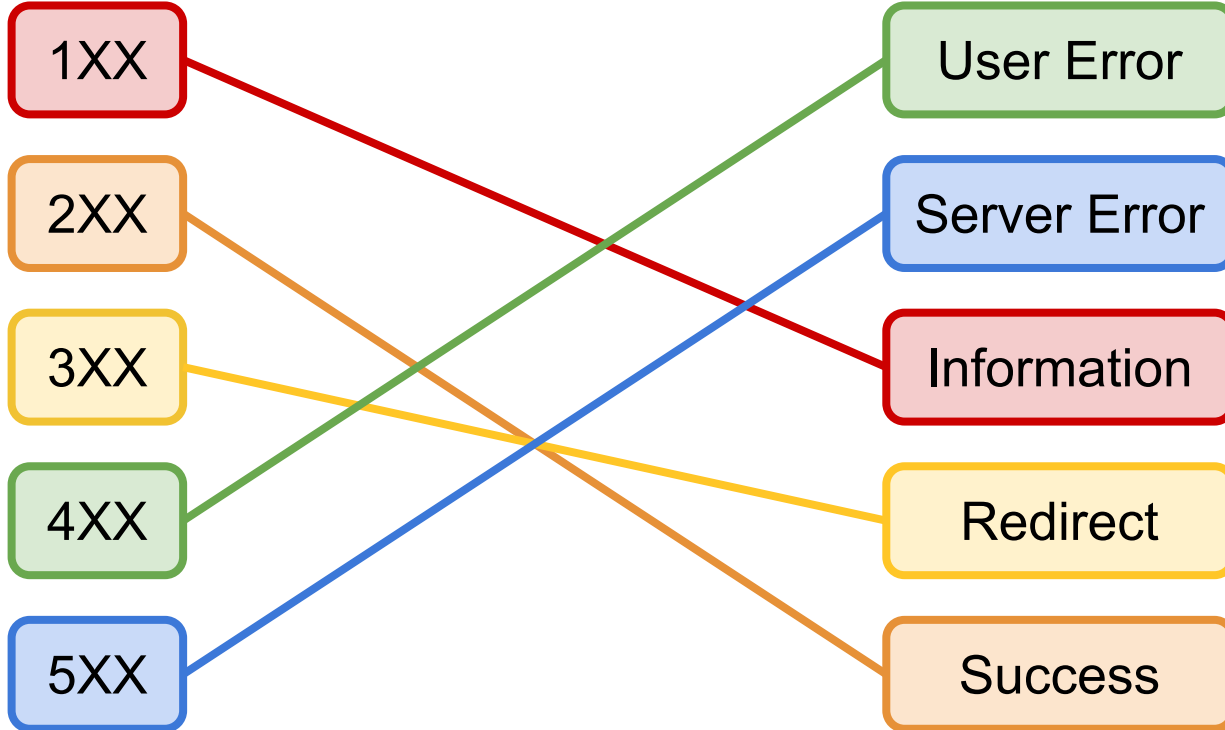
User Error

Server Error

Information

Redirect

Success





# SER 321

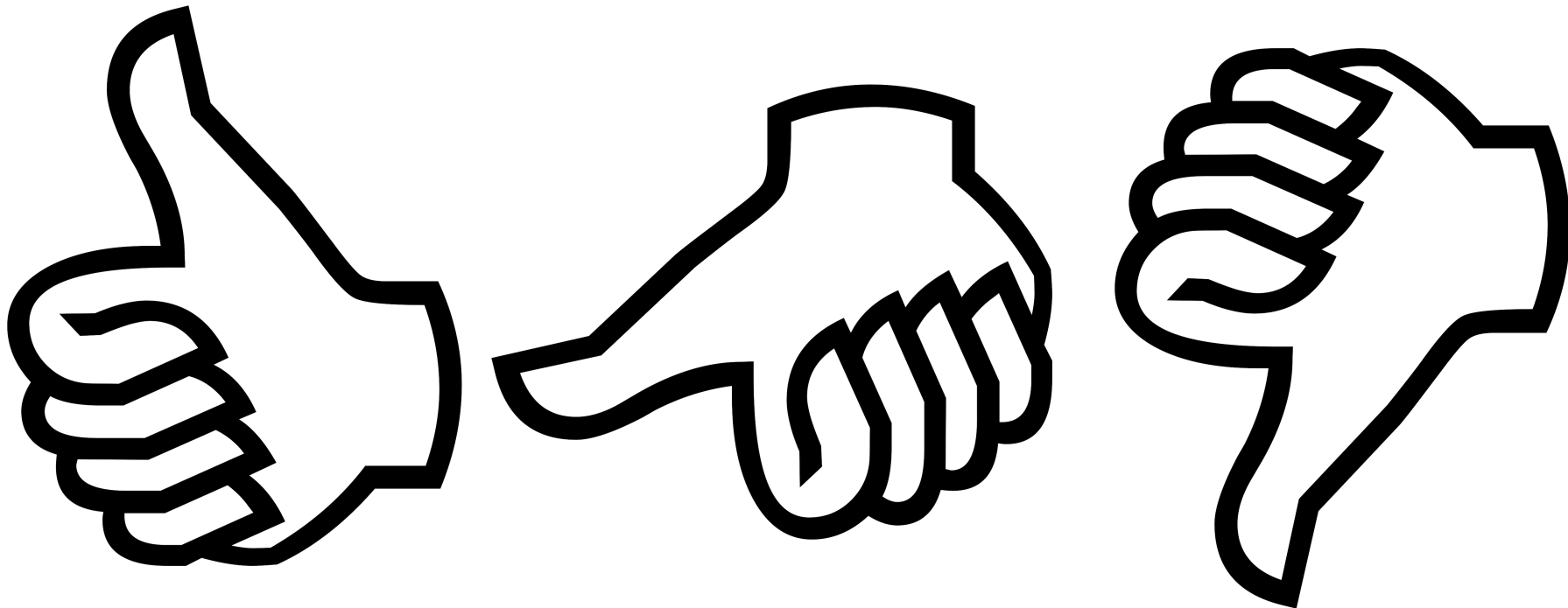
## HTTP Responses

1XX	Information	Rare			
2XX	Success	200	201	204	
3XX	Redirect	301			
4XX	User Error	400	403	404	405
5XX	Server Error	501	503		

**SER 321**

**JSON**

How comfortable are we with JSON?



**SER 321**

**JSON**

**Object**

**Array**

**Member**

{ }

[ ]

“name” : value

**SER 321**

**JSON**

How many Objects?

How many Arrays?

How many Members?

```
{  
  "name": "lab3vue_act3_kgrinne3",  
  "version": "0.0.0",  
  "private": true,  
  "scripts": {  
    "dev": "vite",  
    "build": "vite build",  
    "preview": "vite preview"  
  },  
  "dependencies": {  
    "vue": "^3.3.4"  
  },  
  "devDependencies": {  
    "@vitejs/plugin-vue": "^4.3.1",  
    "vite": "^4.4.9"  
  }  
}
```

# SER 321

## JSON

### What do we see?

```
1. {  
    "name" : "katie",  
    "role" : "student",  
    "course" : "ser321"  
}
```

```
2. [  
    {"name" : "katie"}  
]
```

```
3. {  
    "name" : "katie",  
    "pets" : {  
        "dog" : "smokey"  
    }  
}
```

```
4. {  
    "submissions" : [  
        {"name" : "katie"},  
        {"name" : "zac"}  
    ]  
}
```

# SER 321

## JSON

Do you see any errors?

```
1. {  
    "name" : "katie",  
    "role" : "student",  
    "course" : "ser321"  
}
```

```
2. [{  
    "name" : "katie",  
    "name" : "zac"  
}]
```

```
3. {  
    "name" : "katie",  
    "pets" : {  
        "dog" : "smokey",  
        "dog" : "samie"  
    }  
}
```

```
4. {  
    "submissions" : [  
        {"name" : "katie"},  
        {"name" : "zac"}  
    ]  
}
```

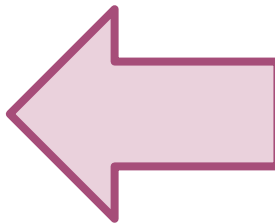
**SER 321**

**JSON**

Can we fix these?

```
[  
  {"name": "katie"},  
  {"name": "zac"}  
]
```

**Correct!**



```
[{  
  "name": "katie",  
  "name": "zac"  
}]
```

**WRONG**

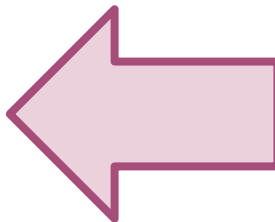
**SER 321**

**JSON**

Can we fix these?

```
[  
  {"name": "katie"},  
  {"name": "zac"}  
]
```

**Correct!**

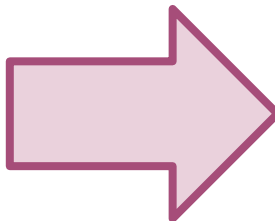


```
[{  
  "name": "katie",  
  "name": "zac"  
}]
```

**WRONG**

```
{  
  "name": "katie",  
  "pets": {  
    "dog": "smokey",  
    "dog": "samie"  
  }  
}
```

**WRONG**



```
{  
  "name": "katie",  
  "pets": {  
    {"dog": "smokey"},  
    {"dog": "samie"}  
  }  
}
```

**Correct!**



# SER 321

## JSON in Assignments

Starter Code files use **org.json**

If you have a different JSON library you would prefer to use, you *can* use that

This library is **very** intuitive, so I recommend using it!

### JSONObject

A JSONObject is an unordered collection of name/value pairs.

### JSONArray

A JSONArray is an ordered sequence of values.

# SER 321

## JSON in Assignments

Very Similar to a **MAP** object

### JSONObject

A JSONObject is an unordered collection of name/value pairs.

boolean

`has(String key)`

Determine if the JSONObject contains a specific key.

```
boolean hasStu = json.has("student");
```

## SER 321

JSONObject	<code>put(String key, boolean value)</code> Put a key/boolean pair in the JSONObject.
JSONObject	<code>put(String key, Collection&lt;?&gt; value)</code> Put a key/value pair in the JSONObject, where the value will be a JSONArray which is produced from a Collection.
JSONObject	<code>put(String key, double value)</code> Put a key/double pair in the JSONObject.
JSONObject	<code>put(String key, float value)</code> Put a key/float pair in the JSONObject.
JSONObject	<code>put(String key, int value)</code> Put a key/int pair in the JSONObject.
JSONObject	<code>put(String key, long value)</code> Put a key/long pair in the JSONObject.
JSONObject	<code>put(String key, Map&lt;?,?&gt; value)</code> Put a key/value pair in the JSONObject, where the value will be a JSONObject which is produced from a Map.
JSONObject	<code>put(String key, Object value)</code> Put a key/value pair in the JSONObject.

Put will add to the Object

```
json.put("term", "spring");
```

# SER 321

## JSON in Assignments

**Need to know the data type**  
when pulling from the  
JSONObject!

```
JSONObject json = {  
    "Student" : "katie",  
    "term" : "fall"  
}
```

`String term = json.get("term");`



`String term = json.getString("term");`



Object	<code>get(String key)</code> Get the value object associated with a key.
BigDecimal	<code>getBigDecimal(String key)</code> Get the BigDecimal value associated with a key.
BigInteger	<code>getBigInteger(String key)</code> Get the BigInteger value associated with a key.
boolean	<code>getBoolean(String key)</code> Get the boolean value associated with a key.
double	<code>getDouble(String key)</code> Get the double value associated with a key.
<code>&lt;E extends Enum&lt;E&gt;&gt;</code> E	<code>getEnum(Class&lt;E&gt; clazz, String key)</code> Get the enum value associated with a key.
float	<code>getFloat(String key)</code> Get the float value associated with a key.
int	<code>getInt(String key)</code> Get the int value associated with a key.
JSONArray	<code>getJSONArray(String key)</code> Get the JSONArray value associated with a key.
JSONObject	<code>getJSONObject(String key)</code> Get the JSONObject value associated with a key.
long	<code>getLong(String key)</code> Get the long value associated with a key.
static String[]	<code>getNames(JSONObject jo)</code> Get an array of field names from a JSONObject.
static String[]	<code>getNames(Object object)</code> Get an array of public field names from an Object.
	<code>getNumber(String key)</code> Get the Number value associated with a key.
	<code>getString(String key)</code> Get the string associated with a key.

SER 321

JSON in Assignments

JSONArray

No HAS

Can target indices

Object	<code>get(int index)</code> Get the object value associated with an index.
BigDecimal	<code>getBigDecimal(int index)</code> Get the BigDecimal value associated with an index.
BigInteger	<code>getBigInteger(int index)</code> Get the BigInteger value associated with an index.
boolean	<code>getBoolean(int index)</code> Get the boolean value associated with an index.
double	<code>getDouble(int index)</code> Get the double value associated with an index.
<code>&lt;E extends Enum&lt;E&gt;&gt; E</code>	<code>getEnum(Class&lt;E&gt; clazz, int index)</code> Get the enum value associated with an index.
float	<code>getFloat(int index)</code> Get the float value associated with a key.
int	<code>getInt(int index)</code> Get the int value associated with an index.
JSONArray	<code>getJSONArray(int index)</code> Get the JSONArray associated with an index.
JSONObject	<code>getJSONObject(int index)</code> Get the JSONObject associated with an index.
long	<code>getLong(int index)</code> Get the long value associated with an index.
Number	<code>getNumber(int index)</code> Get the Number value associated with a key.
String	<code>getString(int index)</code> Get the string associated with an index.

# SER 321

## JSON Practice

JSONObject json =

How would we...

Check for the name member?

boolean hasName =

Get the project name?

String name =

```
{
  "name": "lab3vue_act3_kgrinne3",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "vue": "^3.3.4"
  },
  "devDependencies": {
    "@vitejs/plugin-vue": "^4.3.1",
    "vite": "^4.4.9"
  }
}
```

**SER 321**

**JSON Practice**

[org.json Docs](#)

[JSON Guide](#)

JSONObject json =

How would we...

Get the dev value?

JSONObject scripts =

String dev=

```
{
  "name": "lab3vue_act3_kgrinne3",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "vue": "^3.3.4"
  },
  "devDependencies": {
    "@vitejs/plugin-vue": "^4.3.1",
    "vite": "^4.4.9"
  }
}
```

Step 1:  
Get the scripts  
Object

Step 2:  
Get the dev  
value

# SER 321

## JSON Practice

JSONObject json =

How would we...

Build the json object?

You *could* do it all inline,  
but let's go one object at  
a time for clarity

```
{  
  "name": "lab3vue_act3_kgrinne3",  
  "version": "0.0.0",  
  "private": true,  
  "scripts": {  
    "dev": "vite",  
    "build": "vite build",  
    "preview": "vite preview"  
  },  
  "dependencies": {  
    "vue": "^3.3.4"  
  },  
  "devDependencies": {  
    "@vitejs/plugin-vue": "^4.3.1",  
    "vite": "^4.4.9"  
  }  
}
```



# SER 321

## JSON Practice

JSONObject json =

Answer:

```
{
  "name": "lab3vue_act3_kgrinne3",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "vue": "^3.3.4"
  },
  "devDependencies": {
    "@vitejs/plugin-vue": "^4.3.1",
    "vite": "^4.4.9"
  }
}
```

## SER 321

### JSON Practice

JSONObject json =

Answer:

```
JSONObject json = new JSONObject();  
json.put("name", "lab3vue_act3_kgrinne3");  
json.put("version", "0.0.0");  
json.put("private", "true");
```

```
{  
  "name": "lab3vue_act3_kgrinne3",  
  "version": "0.0.0",  
  "private": true,  
  "scripts": {  
    "dev": "vite",  
    "build": "vite build",  
    "preview": "vite preview"  
  },  
  "dependencies": {  
    "vue": "^3.3.4"  
  },  
  "devDependencies": {  
    "@vitejs/plugin-vue": "^4.3.1",  
    "vite": "^4.4.9"  
  }  
}
```

# SER 321

## JSON Practice

JSONObject scripts =

Answer:

```
{
  "name": "lab3vue_act3_kgrinne3",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "vue": "^3.3.4"
  },
  "devDependencies": {
    "@vitejs/plugin-vue": "^4.3.1",
    "vite": "^4.4.9"
  }
}
```

## SER 321

### JSON Practice

JSONObject scripts =

Answer:

```
JSONObject scripts = new JSONObject();
scripts.put("dev", "vite");
scripts.put("build", "vite build");
scripts.put("preview", "vite preview");
json.put("scripts", scripts.toMap());
```

```
{
  "name": "lab3vue_act3_kgrinne3",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "vue": "^3.3.4"
  },
  "devDependencies": {
    "@vitejs/plugin-vue": "^4.3.1",
    "vite": "^4.4.9"
  }
}
```

# SER 321

## JSON Practice

JSONObject depend =

Answer:

```
{
  "name": "lab3vue_act3_kgrinne3",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "vue": "^3.3.4"
  },
  "devDependencies": {
    "@vitejs/plugin-vue": "^4.3.1",
    "vite": "^4.4.9"
  }
}
```

## SER 321

### JSON Practice

JSONObject depend =

Answer:

```
JSONObject depend = new JSONObject();
depend.put("vue", "^3.3.4");
json.put("dependencies", depend.toMap());
```

```
{
  "name": "lab3vue_act3_kgrinne3",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "vue": "^3.3.4"
  },
  "devDependencies": {
    "@vitejs/plugin-vue": "^4.3.1",
    "vite": "^4.4.9"
  }
}
```

# SER 321

## JSON Practice

JSONObject devDep =

Answer:

```
{
  "name": "lab3vue_act3_kgrinne3",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "vue": "^3.3.4"
  },
  "devDependencies": {
    "@vitejs/plugin-vue": "^4.3.1",
    "vite": "^4.4.9"
  }
}
```

# SER 321

## JSON Practice

JSONObject devDep =

Answer:

```
JSONObject devDep = new JSONObject();
devDep.put("@vitejs/plugin-vue", "^4.3.1");
devDep.put("vite", "^4.4.9");
json.put("devDependencies", devDep.toMap());
```

```
{
  "name": "lab3vue_act3_kgrinne3",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "vue": "^3.3.4"
  },
  "devDependencies": {
    "@vitejs/plugin-vue": "^4.3.1",
    "vite": "^4.4.9"
  }
}
```



**SER 321**

**JSON Practice**

JSONObject json =

Answer:

```
{
  "lat": 42.3434,
  "lon": -88.0412,
  "timezone": "America/Chicago",
  "timezone_offset": -18000,
  "current": {
    "dt": 1743090240,
    "sunrise": 1743075764,
    "sunset": 1743120715,
    "temp": 49.12,
    "feels_like": 46.17,
    "pressure": 1023,
    "humidity": 60,
    "dew_point": 35.85,
    "uvi": 1.36,
    "clouds": 100,
    "visibility": 10000,
    "wind_speed": 6.91,
    "wind_deg": 240,
    "weather": [
      {
        "id": 804,
        "main": "Clouds",
        "description": "overcast clouds",
        "icon": "04d"
      }
    ]
  }
}
```

# SER 321

## JSON Practice

JSONObject json =

Answer:

```
JSONObject current = json.getJSONObject("current");  
int clouds = current.getInt("clouds");
```

```
{  
  "lat": 42.3434,  
  "lon": -88.0412,  
  "timezone": "America/Chicago",  
  "timezone_offset": -18000,  
  "current": {  
    "dt": 1743090240,  
    "sunrise": 1743075764,  
    "sunset": 1743120715,  
    "temp": 49.12,  
    "feels_like": 46.17,  
    "pressure": 1023,  
    "humidity": 60,  
    "dew_point": 35.85,  
    "uvi": 1.36,  
    "clouds": 100,  
    "visibility": 10000,  
    "wind_speed": 6.91,  
    "wind_deg": 240,  
    "weather": [  
      {  
        "id": 804,  
        "main": "Clouds",  
        "description": "overcast clouds",  
        "icon": "04d"  
      }  
    ]  
  }  
}
```

**SER 321**

**JSON Practice**

JSONObject json =

Answer:

```
{
  "lat": 42.3434,
  "lon": -88.0412,
  "timezone": "America/Chicago",
  "timezone_offset": -18000,
  "current": {
    "dt": 1743090240,
    "sunrise": 1743075764,
    "sunset": 1743120715,
    "temp": 49.12,
    "feels_like": 46.17,
    "pressure": 1023,
    "humidity": 60,
    "dew_point": 35.85,
    "uvi": 1.36,
    "clouds": 100,
    "visibility": 10000,
    "wind_speed": 6.91,
    "wind_deg": 240,
    "weather": [
      {
        "id": 804,
        "main": "Clouds",
        "description": "overcast clouds",
        "icon": "04d"
      }
    ]
  }
}
```

# SER 321

## JSON Practice

JSONObject json =

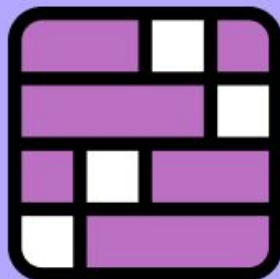
Answer:

```
JSONObject current = json.getJSONObject("current");  
  
JSONObject weather =  
    current.getJSONObject("weather");  
  
String description =  
    weather.getString("description");
```

```
{  
  "lat": 42.3434,  
  "lon": -88.0412,  
  "timezone": "America/Chicago",  
  "timezone_offset": -18000,  
  "current": {  
    "dt": 1743090240,  
    "sunrise": 1743075764,  
    "sunset": 1743120715,  
    "temp": 49.12,  
    "feels_like": 46.17,  
    "pressure": 1023,  
    "humidity": 60,  
    "dew_point": 35.85,  
    "uvi": 1.36,  
    "clouds": 100,  
    "visibility": 10000,  
    "wind_speed": 6.91,  
    "wind_deg": 240,  
    "weather": [  
      {  
        "id": 804,  
        "main": "Clouds",  
        "description": "overcast clouds",  
        "icon": "04d"  
      }  
    ]  
  }  
}
```

Connections!

The New York Times **Games**



**Connections**

# SER 321

Scratch Space

## Upcoming Events

### SI Sessions:

- Sunday, March 30th at 7:00 pm MST
- Tuesday, April 1st at 10:00 am MST
- Thursday, April 3rd at 7:00 pm MST

### Review Sessions:

- TBD

# Questions?

## Survey:

<https://asuasn.info/ASNSurvey>





# More Questions?

Check out our other resources!

tutoring.asu.edu



## Academic Support

Academic Support Network (ASN) provides a variety of free services in-person and online to help currently enrolled ASU students succeed academically.

### Services



#### Subject Area Tutoring

Need in-person or online help with math, science, business, or engineering courses? Just hop into our Zoom room or drop into a center for small group tutoring. We'll take it from there.

[Need help using Zoom?](#)

[View the tutoring schedule](#)

[View digital resources](#)

Go to Zoom



#### Writing Tutoring

Need help with undergraduate or graduate writing assignments? Schedule an in-person or online appointment, access your appointment link, or wait in our drop-in queue.

[Access your appointment link](#)

[Access the drop-in queue](#)

Schedule Appointment



#### Online Study Hub

Join our online peer communities to connect with your fellow Sun Devils. Engage with our tools to search our bank of resources, videos, and previously asked questions. Or, ask our Tutorbot questions.

Now supporting courses in Math, Science, Business, Engineering, and Writing.

Online Study Hub

1-

Go to Zoom

2-

[Need help using Zoom?](#)

[View the tutoring schedule](#)

[View digital resources](#)






1. Click on 'Go to Zoom' to log onto our Online Tutoring Center.
2. Click on 'View the tutoring schedule' to see when tutors are available for specific courses.

# More Questions?

## Check out our other resources!

[tutoring.asu.edu/online-study-hub](https://tutoring.asu.edu/online-study-hub)

 **Academic Support Network**

 [Services](#)  [Faculty and Staff Resources](#) [About Us](#) 

[University College](#)

## Online Study Hub

Online peer communities for students and tutors, YouTube channels, and Tutorbots.



### What are online peer communities?

Individual courses have an online peer community that allows you to connect with your peers to post and answer questions and to develop study groups.



### How can tutoring center videos help?

Videos can help supplement the learning you're doing in and outside of class and include step-by-step methods for how to understand concepts.



### How does the Tutorbot work?

You can ask the Tutorbot questions about course concepts and the Tutorbot will recommend additional resources and examples to help address your questions.

Select a subject

- Any -

[Apply](#)



**Academic Support Network**



[Services](#) 

[Faculty and Staff Resources](#)

[About Us](#) 

[University College](#)

Select a subject

- Any -

[Apply](#)

**Business**


### ACC 231

Uses of Accounting Info I

 [Peer Community](#)

### ACC 241

Uses of Accounting Info II

 [Peer Community](#)

### CIS 105

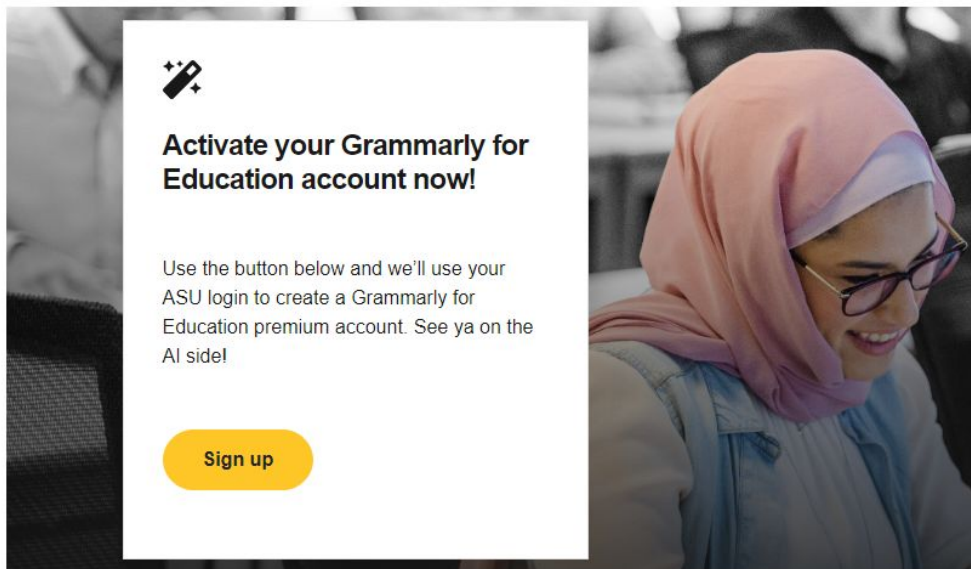
Computer Applications and Information Technology

 [Peer Community](#)

Don't forget to check out the Online Study Hub for additional resources!

# Expanded Writing Support Available

Including Grammarly for Education, at no cost!



[tutoring.asu.edu/expanded-writing-support](https://tutoring.asu.edu/expanded-writing-support)

\*Available slots for this pilot are limited

## Additional Resources

- [Course Repo](#)
- [Gradle Documentation](#)
- [GitHub SSH Help](#)
- [Linux Man Pages](#)
- [OSI Interactive](#)
- [MDN HTTP Docs](#)
  - [Requests](#)
  - [Responses](#)
- [JSON Guide](#)
- [org.json Docs](#)