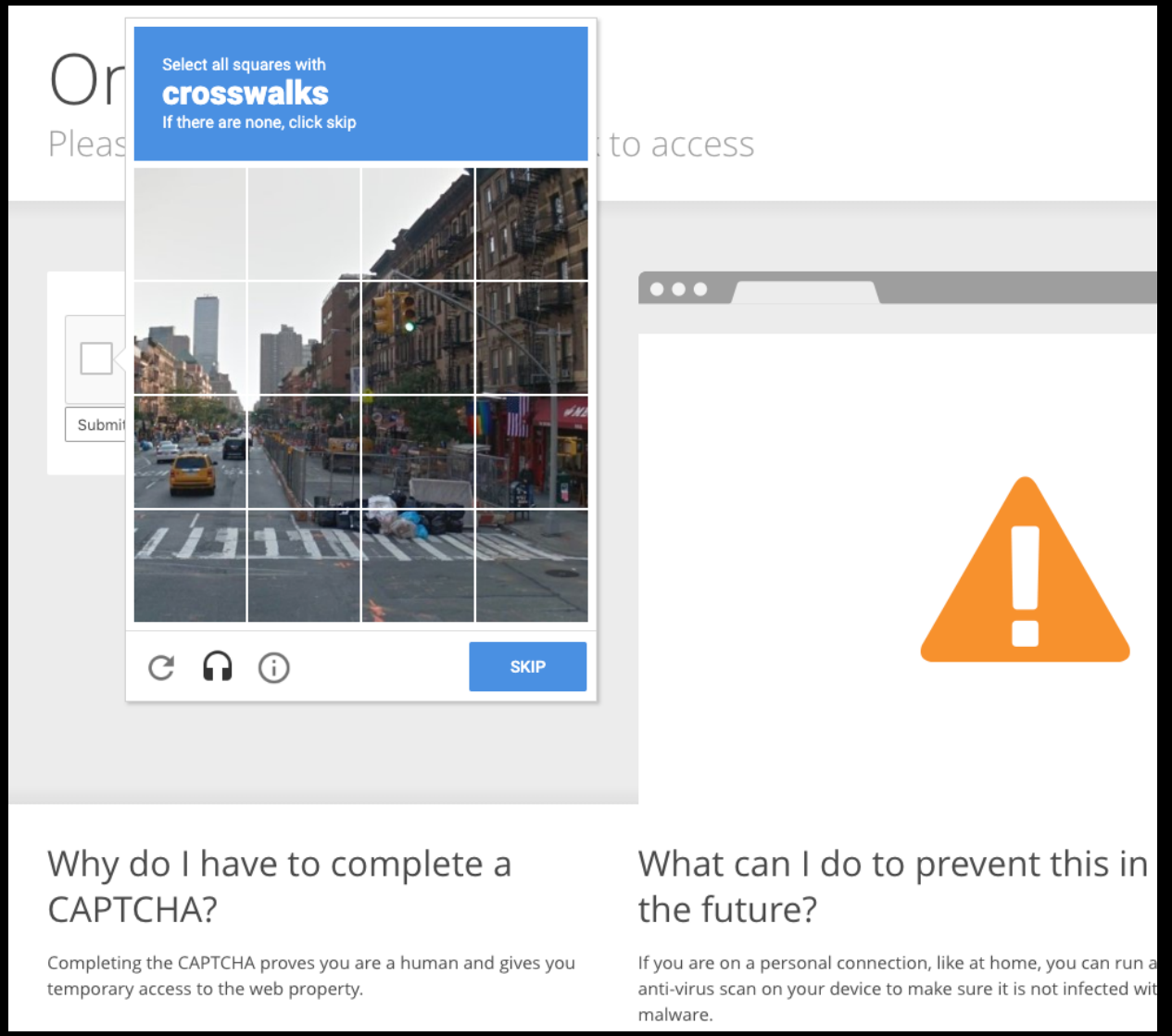

JSON
JavaScript Object Notation
DAVID BOMBAL
EDUCATE.MOTIVATE

Human?

- Are you a human or a robot?
- Machines struggle with unformatted data



DAVID BOMBAL
EDUCATE.MOTIVATE


DAVID BOMBAL
EDUCATE.MOTIVATE

- Manufacturer: Telsa
- Model: X
- Color: White
- Fuel: Electric
- MPG city: 99
- MPG highway: 93

DAVID BOMBAL
EDUCATE.MOTIVATE

JSON Data

- JSON data is written as name/value pairs:
  - A name / value pair consists of a field name (in double quotes), followed by a colon, followed by a value:
  - "firstname" : "David"
  - Or, "lastname" : "Bombal"

JSON Data types: Object

- Object
  - Unordered collection of key:value pairs
  - Data surrounded by curly braces { }
  - Key and value are separated by a colon
  - Each key/value pair is separated by a comma (not the last!). Trailing commas must not be used.
  - Use double quotes, not single quotes.
  - Boolean values must be lowercase (different to Python)
  - Example:
    - `{"firstName": "David", "lastName": "Bombal"}`

JSON Data types: Object

- Object
  - Another example:

```
{
        "firstName" : "David",
        "lastName" : "Bombal",
        "website" : "davidbombal.com"
}
```

JSON Data types

- Array
  - Ordered list of values
  - Uses square brackets [ ]
  - Can store multiple types:
    - JSON array can store valid JSON data types such as string, number, Boolean, object, null or another array
  - Values must be separated by comma.

Practical

- Use the DevNet Nexus always on sandbox
  - DNS name: sbx-nxos-mgmt.cisco.com
  - Protocol: SSH
  - Port: 8181
  - Username: admin
  - Password: Admin_1234!
- Example:
  - Mac: ssh -p 8181 admin@sbx-nxos-mgmt.cisco.com
  - Windows: Use Putty to connect

JSON Data types

- Object
  - Unordered collection of key:value pairs
  - Data surrounded by curly braces { }
- show version | json-pretty

```
{
    "chassis_id": "Nexus9000 9000v Chassis",
    "cpu_name": "Intel(R) Xeon(R) CPU E5-4669 v4 @ 2.20GHz",
   "manufacturer": "Cisco Systems, Inc."
}
```

JSON Data types

- Object
  - Unordered collection of key:value pairs
  - Data surrounded by curly braces { }

```
sbx-n9kv-ao# show int vlan 100 | json-pretty
{
    "TABLE_interface": {
        "ROW_interface": {
            "interface": "Vlan100",
            "svi_admin_state": "up",
            "svi_line_proto": "up",
            "svi_mac": "0050.56bb.ab06",
            "svi_ip_addr": "172.16.100.1",
            "svi_ip_mask": "24",
            "svi_mtu": "1500",
            "svi_bw": "1000000",
            "svi_delay": "10",
            "svi_tx_load": "1",
            "svi_rx_load": "1",
            "svi_arp_type": "ARPA",
            "svi_time_last_cleared": "never",
            "svi_ucast_pkts_in": "0",
            "svi_ucast_bytes_in": "0"
        }
    }
}
sbx-n9kv-ao#
```

```
sbx-n9kv-ao# show int eth1/5 | json-pretty
{
    "TABLE_interface": {
        "ROW_interface": {
            "interface": "Ethernet1/5",
            "state": "up",
            "admin_state": "up",
            "share_state": "Dedicated",
            "eth_hw_desc": "100/1000/10000 Ethernet",
            "eth_hw_addr": "0050.56bb.ab06",
            "eth_bia_addr": "0050.56bb.ab0b",
            "desc": "L3 Link",
            "eth_ip_addr": "172.16.1.1",
            "eth_ip_mask": "30",
            "eth_ip_prefix": "172.16.1.0",
            "eth_mtu": "1500",
            "eth_bw": "1000000",
            "eth_dly": "10",
            "eth_reliability": "255",
            "eth_txload": "1",
            "eth_rxload": "1",
            "encapsulation": "ARPA",
            "medium": "broadcast",
            "eth_duplex": "full",
            "eth_speed": "1000 Mb/s",
            "eth_beacon": "off",
            "eth_autoneg": "on",
            "eth_in_flowctrl": "off",
            "eth_out_flowctrl": "off",
            "eth_mdix": "off",
            "eth_swt_monitor": "off",
            "eth_ethertype": "0x8100",
            "eth_storm_supp": "0",
            "eth_runts": "0",
            "eth_giants": "0",
            "eth_crc": "0",
            "eth_nobuf": "0",
            "eth_inerr": "0",
            "eth_frame": "0",
            "eth_overrun": "0",
            "eth_underrun": "0",
            "eth_ignored": "0",
            "eth_watchdog": "0",
            "eth_bad_eth": "0",
            "eth_bad_proto": "0",
            "eth_in_ifdown_drops": "0",
            "eth_dribble": "0",
            "eth_indiscard": "0",
            "eth_inpause": "0",
            "eth_outucast": "0",
            "eth_outmcast": "0",
            "eth_outbcast": "0",
            "eth_outpkts": "0",
            "eth_outbytes": "0",
            "eth_jumbo_outpkts": "0",
            "eth_outerr": "0",
            "eth_coll": "0",
            "eth_deferred": "0",
            "eth_latecoll": "0",
            "eth_lostcarrier": "0",
            "eth_nocarrier": "0",
            "eth_babbles": "0",
            "eth_outdiscard": "0",
            "eth_outpause": "0"
        }
    }
}
sbx-n9kv-ao#
```

JSON Data types

- Array
  - Ordered list of values
  - Uses square brackets [ ]

```
{
    "Interfaces": [
            {
              "intf-name": "Vlan100",
              "admin-state": "up",
              "prefix": "10.1.100.1",
            },

            {
              "intf-name": "Vlan200",
              "admin-state": "up",
              "prefix": "10.1.200.1",
            }
    ]
}
```

Python Script 1

- Run Python On-Box :

```
from cli import *
cmd1 = 'show ip int brief | json-pretty'
output1 = cli(cmd1)
print(output1)
```

Python Script 2

- Run Python On-Box :
  - Extract
    - TABLE_intf dictionary
    - ROW_intf dictionary
  - Select element in the index
  - Select the key

Python Script 2

- Run Python On-Box :

```
from cli import *
import json
cmd1 = 'show ip int brief | json-pretty'
json_data = cli(cmd1)
json_final = json.loads(json_data)
print(json_final['TABLE_intf']['ROW_intf'][0]['prefix'])
```

Python Script 3 Step 1

- Run Python remotely using Ubuntu:
  - Step 1: Install Python and Netmiko

```
sudo apt install python3-pip
sudo pip3 install -U netmiko
```

Python Script 3 Step 2

- Run Python remotely using Ubuntu VM: Print out interfaces to test our script:

```
#!/usr/bin/env python
from netmiko import ConnectHandler

nx_os = {
    'device_type': 'cisco_ios',
    'ip': 'sbx-nxos-mgmt.cisco.com',
    'username': 'admin',
    'password': 'Admin_1234!',
    'port': 8181
}

net_connect = ConnectHandler(**nx_os)
output = net_connect.send_command('show ip int brief')
print(output)
```

Python Script 4

- Print JSON output:

```
#!/usr/bin/env python

from netmiko import ConnectHandler

nx_os = {
    'device_type': 'cisco_ios',
    'ip': 'sbx-nxos-mgmt.cisco.com',
    'username': 'admin',
    'password': 'Admin_1234!',
    'port': 8181
}

net_connect = ConnectHandler(**nx_os)
output = net_connect.send_command('show ip int brief | json-pretty')
print(output)
```

Python Script 5

- Find some data:

```
#!/usr/bin/env python
from netmiko import ConnectHandler
import json
nx_os = {
    'device_type': 'cisco_ios',
    'ip': 'sbx-nxos-mgmt.cisco.com',
    'username': 'admin',
    'password': 'Admin_1234!',
    'port': 8181
}
net_connect = ConnectHandler(**nx_os)
output = net_connect.send_command('show ip int brief | json-pretty')
json_data = json.loads(output)
print(json_data['TABLE_intf']['ROW_intf'][0]['intf-name'])
print(json_data['TABLE_intf']['ROW_intf'][0]['prefix'])
```

Python Script 6 Part 1

- Use a loop to find IP addresses :

```
#!/usr/bin/env python
from netmiko import ConnectHandler
import json
nx_os = {
    'device_type': 'cisco_ios',
    'ip': 'sbx-nxos-mgmt.cisco.com',
    'username': 'admin',
    'password': 'Admin_1234!',
    'port': 8181
}
```

Python Script 6 Part 2

- Use a loop to find IP addresses :

```
net_connect = ConnectHandler(**nx_os)
output = net_connect.send_command('show ip int brief | json-pretty')
json_data = json.loads(output)
int_number = len(json_data['TABLE_intf']['ROW_intf'])

for x in range(int_number):
    print(json_data['TABLE_intf']['ROW_intf'][x]['intf-name'])
    print(json_data['TABLE_intf']['ROW_intf'][x]['prefix'])
```


JSON
JavaScript Object Notation
DAVID BOMBAL
EDUCATE.MOTIVATE