Data visualization cheat sheet

for numpy, pandas, pytorch, tensor

kai

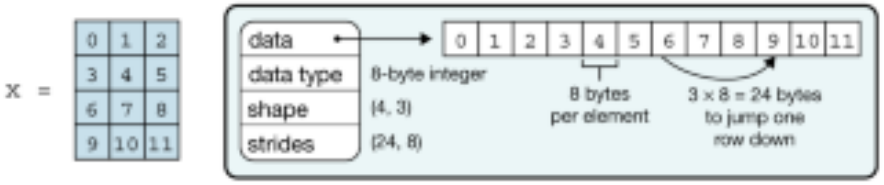
Cheat Sheet

numpy

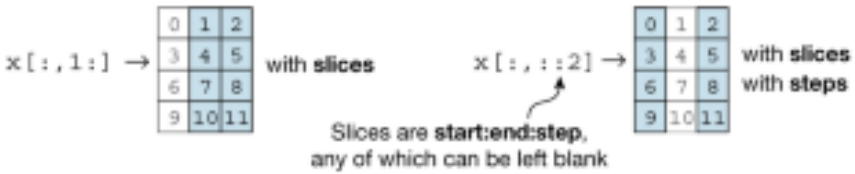
NumPy

https://numpy.org/

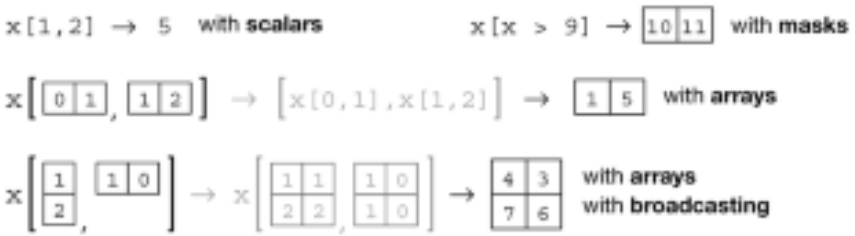
a Data structure



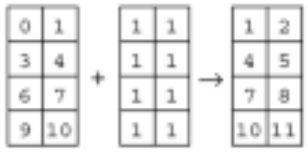
b Indexing (view)



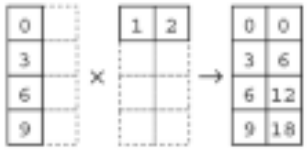
c Indexing (copy)



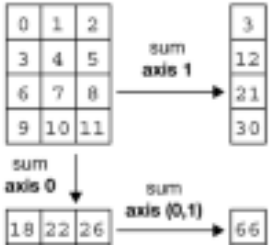
d Vectorization



e Broadcasting



f Reduction

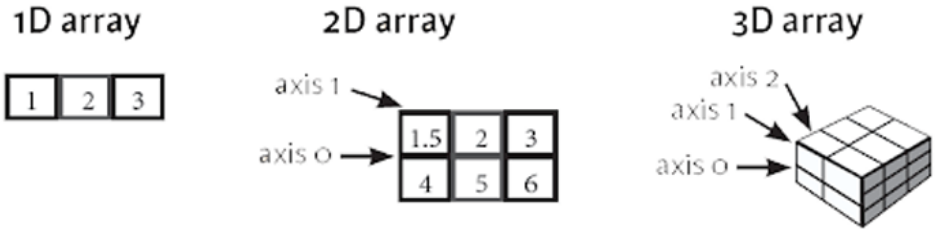


g Example

```
In [1]: import numpy as np

In [2]: x = np.arange(12)

In [3]: x = x.reshape(4, 3)

In [4]: x
Out[4]:
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])

In [5]: np.mean(x, axis=0)
Out[5]: array([4.5, 5.5, 6.5])

In [6]: x = x - np.mean(x, axis=0)

In [7]: x
Out[7]:
array([[-4.5, -4.5, -4.5],
       [-1.5, -1.5, -1.5],
       [ 1.5,  1.5,  1.5],
       [ 4.5,  4.5,  4.5]])
```

numpy

NumPy



> Subsetting, Slicing, Indexing

Subsetting

```
>>> a[2] #Select the element at the 2nd index
 3
>>> b[1,2] #Select the element at row 1 column 2 (equivalent to b[1][2])
 6.0
```

Slicing

```
>>> a[0:2] #Select items at index 0 and 1
 array([1, 2])
>>> b[0:2,1] #Select items at rows 0 and 1 in column 1
 array([ 2., 5.])
>>> b[:1] #Select all items at row 0 (equivalent to b[0:1, :])
 array([[1.5, 2., 3.]])
>>> c[1,...] #Same as [1,:,:]
 array([[[ 3., 2., 1.],
         [ 4., 5., 6.]]])
>>> a[ : :-1] #Reversed array a array([3, 2, 1])
```

Boolean Indexing

```
>>> a[a<2] #Select elements from a less than 2
 array([1])
```

Fancy Indexing

```
>>> b[[1, 0, 1, 0],[0, 1, 2, 0]] #Select elements (1,0),(0,1),(1,2) and (0,0)
 array([ 4. , 2. , 6. , 1.5])
>>> b[[1, 0, 1, 0]][:,[0,1,2,0]] #Select a subset of the matrix's rows and columns
 array([[ 4. ,5. , 6. , 4. ],
        [ 1.5, 2. , 3. , 1.5],
        [ 4. , 5. , 6. , 4. ],
        [ 1.5, 2. , 3. , 1.5]])
```

Transposing Array

```
>>> i = np.transpose(b) #Permute array dimensions
>>> i.T #Permute array dimensions
```

Changing Array Shape

```
>>> b.ravel() #Flatten the array
>>> g.reshape(3,-2) #Reshape, but don't change data
```

Adding/Removing Elements

```
>>> h.resize((2,6)) #Return a new array with shape (2,6)
>>> np.append(h,g) #Append items to an array
>>> np.insert(a, 1, 5) #Insert items in an array
>>> np.delete(a,[1]) #Delete items from an array
```

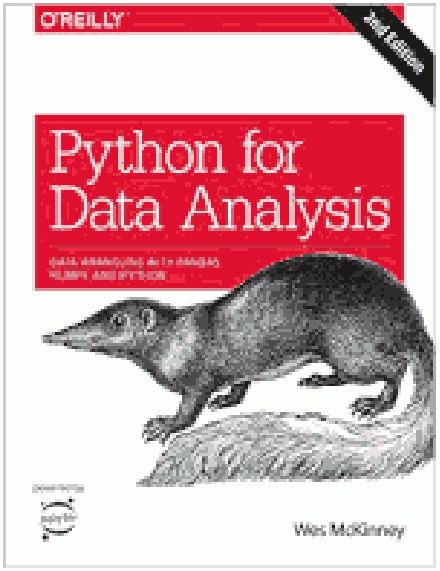
Combining Arrays

```
>>> np.concatenate((a,d),axis=0) #Concatenate arrays
 array([ 1, 2, 3, 10, 15, 20])
>>> np.vstack((a,b)) #Stack arrays vertically (row-wise)
 array([[ 1. , 2. , 3. ],
        [ 1.5, 2. , 3. ],
        [ 4. , 5. , 6. ]])
>>> np.r_[e,f] #Stack arrays vertically (row-wise)
>>> np.hstack((e,f)) #Stack arrays horizontally (column-wise)
 array([[ 7., 7., 1., 0.],
        [ 7., 7., 0., 1.]])
>>> np.column_stack((a,d)) #Create stacked column-wise arrays
 array([[ 1, 10],
        [ 2, 15],
        [ 3, 20]])
>>> np.c_[a,d] #Create stacked column-wise arrays
```

Link - https://media.datacamp.com/legacy/image/upload/v1676302459/Marketing/Blog/Numpy_Cheat_Sheet.pdf

PANDAS

https://pandas.pydata.org/



NUMFOCUS
OPEN CODE = BETTER SCIENCE

TWO SIGMA

d-fine

VOLTRON DATA

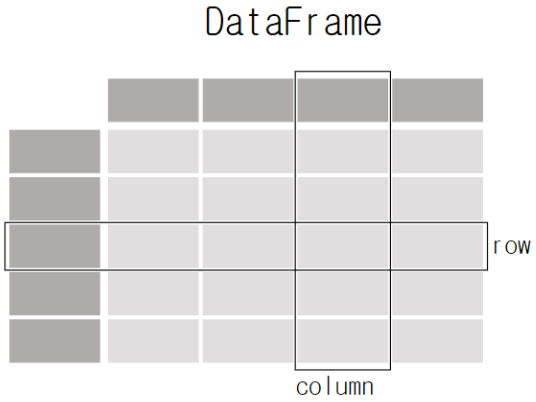
Quansight Labs

NVIDIA

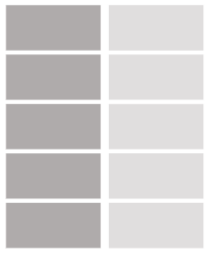
TIDELIFT

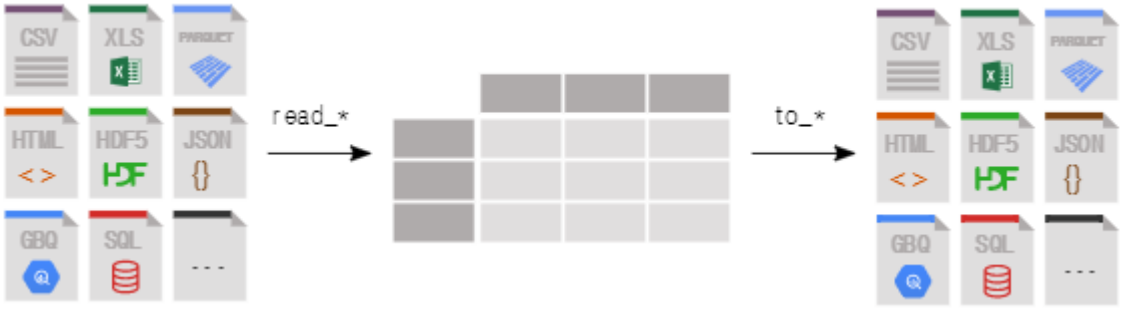
Chan Zuckerberg Initiative

bodo.ai



Each column in a `DataFrame` is a `Series`





Creating DataFrames

https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
df = pd.DataFrame(
        {"a" : [4, 5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
       index = [1, 2, 3])
```

Specify values for each column.

```
df = pd.DataFrame(
    [[4, 7, 10],
     [5, 8, 11],
     [6, 9, 12]],
    index=[1, 2, 3],
    columns=['a', 'b', 'c'])
```

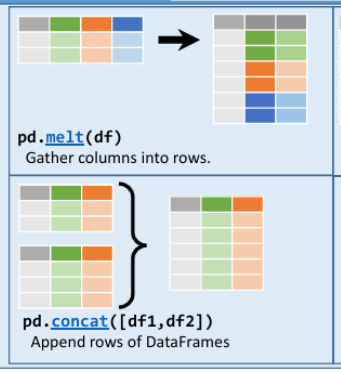
Specify values for each row.

		a	b	c
N	v			
D	1	4	7	10
	2	5	8	11
e	2	6	9	12

```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
index = pd.MultiIndex.from_tuples(
        [('d', 1), ('d', 2),
         ('e', 2)], names=['n', 'v']))
```

Create DataFrame with a MultiIndex

Reshaping Data – Change layout, sorting, reindexing, renaming



`pd.melt(df)`
Gather columns into rows.

`df.pivot(columns='var', values='val')`
Spread rows into columns.

`pd.concat([df1,df2])`
Append rows of DataFrames

`pd.concat([df1,df2], axis=1)`
Append columns of DataFrames

`df.sort_values('mpg')`
Order rows by values of a column (low to high).

`df.sort_values('mpg', ascending=False)`
Order rows by values of a column (high to low).

`df.rename(columns = {'y':'year'})`
Rename the columns of a DataFrame

`df.sort_index()`
Sort the index of a DataFrame

`df.reset_index()`
Reset index of DataFrame to row numbers, moving index to columns.

`df.drop(columns=['Length', 'Height'])`
Drop columns from DataFrame

Subset Variables - columns

`df[['width', 'length', 'species']]`
Select multiple columns with specific names.

`df['width']` *or* `df.width`
Select single column with specific name.

`df.filter(regex='regex')`
Select columns whose name matches regular expression *regex*.

Using query

query() allows Boolean expressions for filtering rows.

```
df.query('Length > 7')
df.query('Length > 7 and Width < 8')
df.query('Name.str.startswith("abc")',
         engine="python")
```

Subsets - rows and columns

Use `df.loc[]` and `df.iloc[]` to select only rows, only columns or both.
Use `df.at[]` and `df.iat[]` to access a single value by row and column.
First index selects rows, second index columns.

`df.iloc[10:20]`
Select rows 10-20.

`df.iloc[:, [1, 2, 5]]`
Select columns in positions 1, 2 and 5 (first column is 0).

`df.loc[:, 'x2':'x4']`
Select all columns between x2 and x4 (inclusive).

`df.loc[df['a'] > 10, ['a', 'c']]`
Select rows meeting logical condition, and only the specific columns .

`df.iat[1, 2]` Access single value by index

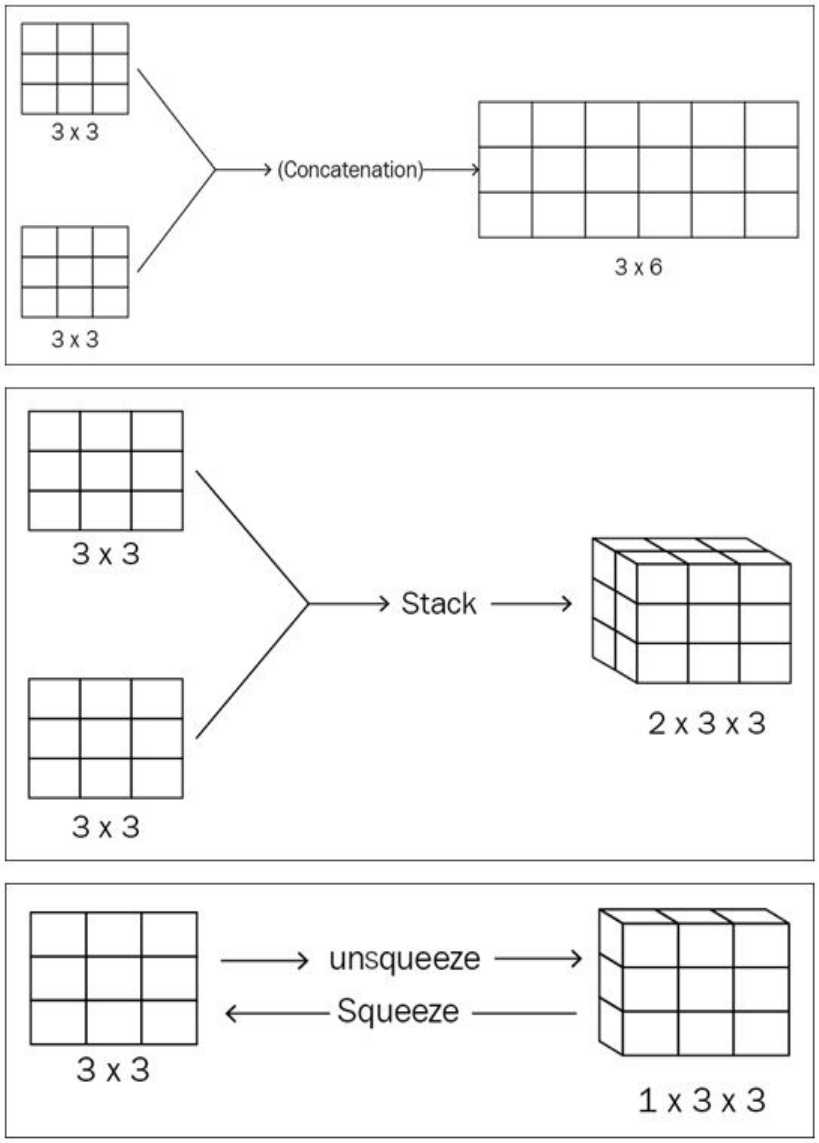
`df.at[4, 'A']` Access single value by label

Pytorch matrix and tensor

Type	Scalar	Vector	Matrix	Tensor
Definition	a single number	an array of numbers	2-D array of numbers	k-D array of numbers
Notation	x	$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$	$\boldsymbol{X} = \begin{bmatrix} X_{1,1} & X_{1,2} & \cdots & X_{1,n} \\ X_{2,1} & X_{2,2} & \cdots & X_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ X_{m,1} & X_{m,2} & \cdots & X_{m,n} \end{bmatrix}$	$\mathbf{X}$ $X_{i,j,k}$
Example	1.333	$\boldsymbol{x} = \begin{bmatrix} 1 \\ 2 \\ \vdots \\ 9 \end{bmatrix}$	$\boldsymbol{X} = \begin{bmatrix} 1 & 2 & \cdots & 4 \\ 5 & 6 & \cdots & 8 \\ \vdots & \vdots & \vdots & \vdots \\ 13 & 14 & \cdots & 16 \end{bmatrix}$	$\mathbf{x} = \begin{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} & \begin{bmatrix} 10 & 20 & 30 \\ 50 & 60 \\ 80 & 90 \end{bmatrix} & \begin{bmatrix} 100 & 200 & 300 \\ 00 & 600 \\ 00 & 900 \end{bmatrix} \end{bmatrix}$
Python code example	x = np.array(1.333)	x = np.array([1,2,3, 4,5,6, 7,8,9])	x = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12], [13,14,15,16]])	x = np.array([[[1, 2, 3], [4, 5, 6], [7, 8, 9]], [[10, 20, 30], [40, 50, 60], [70, 80, 90]], [[100, 200, 300], [400, 500, 600], [700, 800, 900]]])
Visualization				 3-D Tensor

https://www.kaggle.com/code/jarfo1/a-world-of-tensors-and-differentiable-computing

Pytorch matrix and tensor



https://www.codementor.io/@packt/how-to-perform-basic-operations-in-pytorch-code-10al39a4c4

pytorch

1 Load data 2 Define model 3 Train model 4 Evaluate model

General

PyTorch is a open source machine learning framework. It uses **torch.Tensor** – multi-dimensional matrices – to process. A core feature of neural networks in PyTorch is the autograd package, which provides automatic derivative calculations for all operations on tensors.

import torch	Root package	**torch.randn(*size)**	Create random tensor
import torch.nn as nn	Neural networks	**torch.Tensor(L)**	Create tensor from list
from torchvision import datasets, models, transforms	Popular image datasets, architectures & transforms	**tnsr.view(a,b, ...)**	Reshape tensor to size (a, b, ...)
import torch.nn.functional as F	Collection of layers, activations & more	**requires_grad=True**	tracks computation history for derivative calculations

Layers

nn.Linear(m, n): Fully Connected layer (or dense layer) from m to n neurons

nn.ConvXd(m, n, s): X-dimensional convolutional layer from m to n channels with kernel size s; X ∈ {1, 2, 3}

nn.Flatten(): Flattens a contiguous range of dimensions into a tensor

nn.MaxPoolXd(s): X-dimensional pooling layer with kernel size s; X ∈ {1, 2, 3}

nn.Dropout(p=0.5): Randomly sets input elements to zero during training to prevent overfitting

nn.BatchNormXd(n): Normalizes a X-dimensional input batch with n features; X ∈ {1, 2, 3}

nn.Embedding(m, n): Lookup table to map dictionary of size m to embedding vector of size n

nn.RNN/LSTM/GRU: Recurrent networks connect neurons of one layer with neurons of the same or a previous layer

torch.nn offers a bunch of other building blocks.
A list of state-of-the-art architectures can be found at https://paperswithcode.com/sota.

PyTorch Tutorial for Reshape, Squeeze, Unsqueeze, Flatten and View - MLK - Machine Learning Knowledge

Link - https://www.stefanseegerer.de/media/pytorch-cheatsheet-EN.pdf

Define model

There are several ways to define a neural network in PyTorch, e.g. with **nn.Sequential** (a), as a class (b) or using a combination of both.

```
model = nn.Sequential(
    nn.Conv2D(■,■,■)
    nn.ReLU()
    nn.MaxPool2D(■)
    nn.Flatten()
    nn.Linear(■,■)
)
```
a

```
class Net(nn.Module):
  def __init__():
    super(Net, self).__init__()

      self.conv
        = nn.Conv2D(■,■,■)

      self.pool
        = nn.MaxPool2D(■)

      self.fc = nn.Linear(■,■)

  def forward(self, x):
      x = self.pool(
            F.relu(self.conv(x))
          )

      x = x.view(-1,■)

      x = self.fc(x)

    return x
model = Net()
```
b

Train model

LOSS FUNCTIONS

PyTorch already offers a bunch of different loss fuctions, e.g.:

nn.L1Loss	Mean absolute error
nn.MSELoss	Mean squared error (L2Loss)
nn.CrossEntropyLoss	Cross entropy, e.g. for single-label classification or unbalanced training set
nn.BCELoss	Binary cross entropy, e.g. for multi-label classification or autoencoders

OPTIMIZATION (torch.optim)

Optimization algorithms are used to update weights and dynamically adapt the learning rate with gradient descent, e.g.:

optim.SGD	Stochastic gradient descent
optim.Adam	Adaptive moment estimation
optim.Adagrad	Adaptive gradient
optim.RMSProp	Root mean square prop

GPU Training

device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')

If a GPU with CUDA support is available, computations are sent to the GPU with ID 0 using **model.to(device)** or **inputs, labels = data[0].to(device), data[1].to(device).**

```
import torch.optim as optim

# Define loss function
loss_fn = nn.CrossEntropyLoss()

# Choose optimization method
optimizer = optim.SGD(model.parameters(),
                      lr=0.001, momentum=0.9)

# Loop over dataset multiple times (epochs)
for epoch in range(2):
    model.train() # activate training mode
    for i, data in enumerate(train_loader, 0):
        # data is a batch of [inputs, labels]
        inputs, labels = data

        # zero gradients
        optimizer.zero_grad()

        # calculate outputs
        outputs = model(inputs)
        # calculate loss & backpropagate error
        loss = loss_fn(outputs, labels)
        loss.backward()
        # update weights & learning rate
        optimizer.step()
```

Evaluate model

The evaluation examines whether the model provides satisfactory results on previously withheld data. Depending on the objective, different metrics are used, such as acurracy, precision, recall, F1, or BLEU.

model.eval()	Activates evaluation mode, some layers behave differently
torch.no_grad()	Prevents tracking history, reduces memory usage, speeds up calculations

Link - https://www.stefanseegerer.de/media/pytorch-cheatsheet-EN.pdf

matplotlib

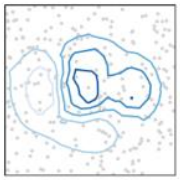
matplotlib

Matplotlib: Visualization with Python

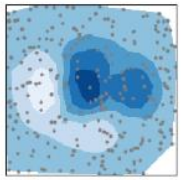
Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many file formats.
- Embed in JupyterLab and Graphical User Interfaces.
- Use a rich array of third-party packages built on Matplotlib.

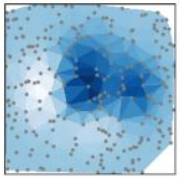
Try Matplotlib (on Binder) →



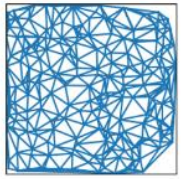
tricontour(x, y, z)



tricontourf(x, y, z)



tripcolor(x, y, z)



triplot(x, y)

https://matplotlib.org/stable/plot_types/index.html

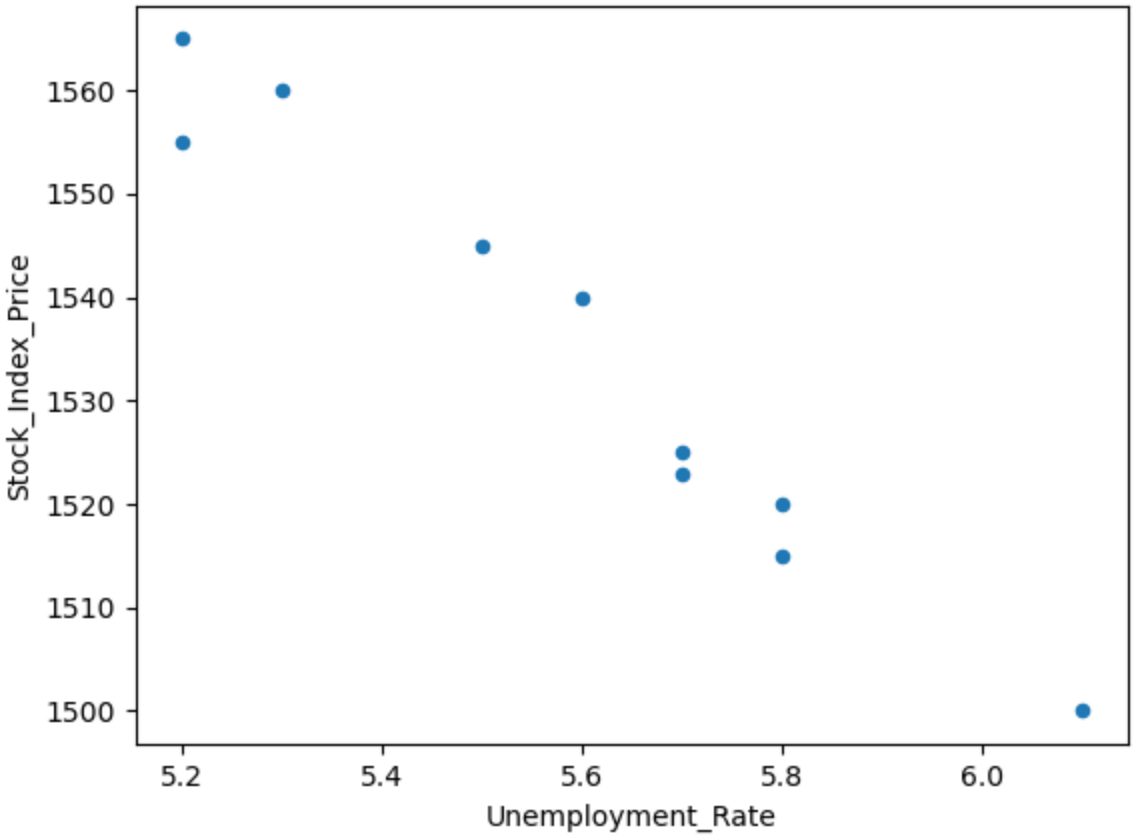
Example code

Data Analysis

```
import pandas as pd
import matplotlib.pyplot as plt

data = {'Unemployment_Rate': [6.1,5.8,5.7,5.7,5.8,5.6,5.5,5.3,5.2,5.2],
        'Stock_Index_Price': [1500,1520,1525,1523,1515,1540,1545,1560,1555,1565]
       }

df = pd.DataFrame(data,columns=['Unemployment_Rate','Stock_Index_Price'])
df.plot(x ='Unemployment_Rate', y='Stock_Index_Price', kind = 'scatter')
plt.show()
```

https://matplotlib.org/stable/api/index.html

Data Analysis



Data Analysis

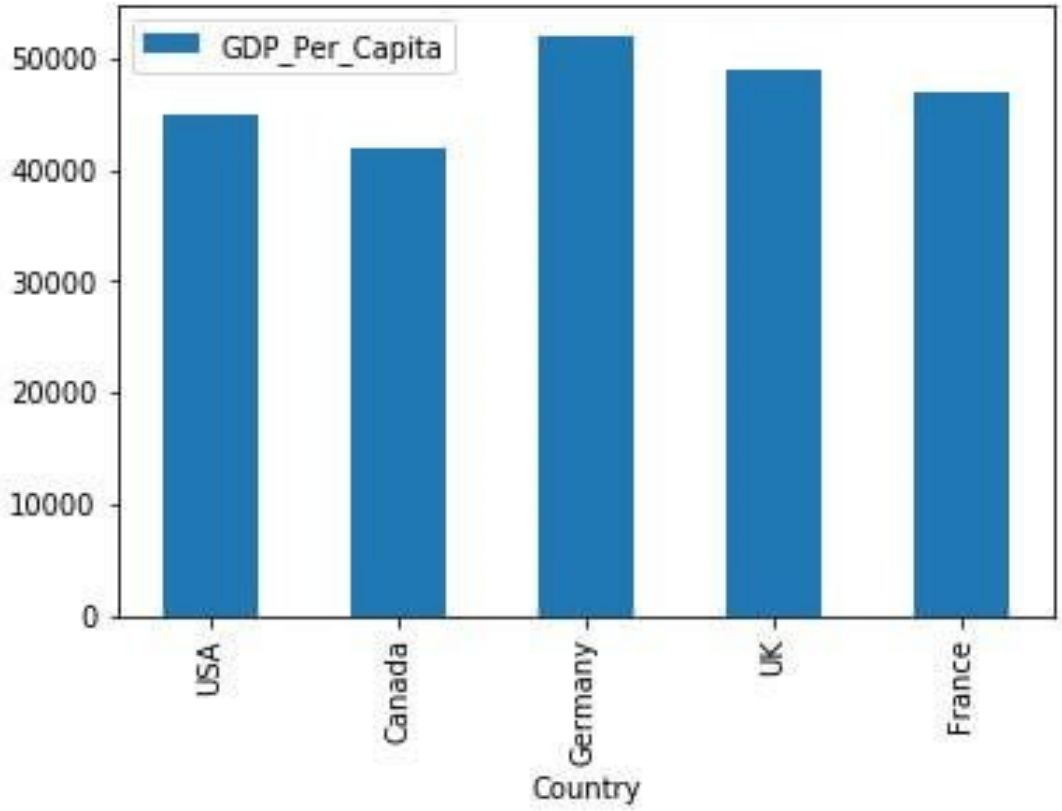
```
import pandas as pd
import matplotlib.pyplot as plt

data = {'Year': [1920,1930,1940,1950,1960,1970,1980,1990,2000,2010],
        'Unemployment_Rate': [9.8,12,8,7.2,6.9,7,6.5,6.2,5.5,6.3]
       }

df = pd.DataFrame(data,columns=['Year','Unemployment_Rate'])
df.plot(x ='Year', y='Unemployment_Rate', kind = 'line')
plt.show()
```

Data Analysis

Data Analysis

```
import pandas as pd
import matplotlib.pyplot as plt

data = {'Country': ['USA','Canada','Germany','UK','France'],
        'GDP_Per_Capita': [45000,42000,52000,49000,47000]
       }

df = pd.DataFrame(data,columns=['Country','GDP_Per_Capita'])
df.plot(x ='Country', y='GDP_Per_Capita', kind = 'bar')
plt.show()
```

Data Analysis

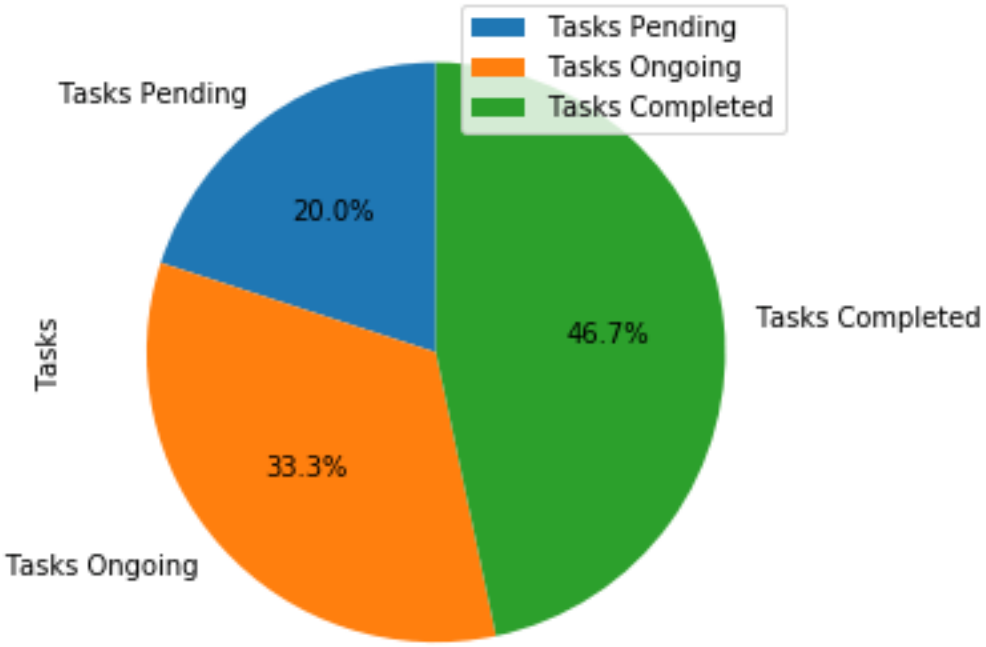


Data Analysis

```
import pandas as pd
import matplotlib.pyplot as plt

data = {'Tasks': [300,500,700]}
df = pd.DataFrame(data,columns=['Tasks'],index = ['Tasks Pending','Tasks Ongoing','Tasks Completed'])

df.plot.pie(y='Tasks',figsize=(5, 5),autopct='%1.1f%%', startangle=90)
plt.show()
```

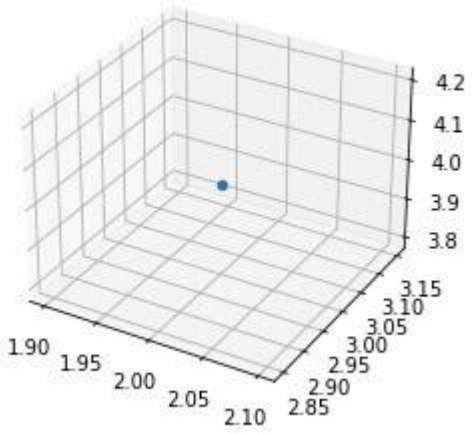
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.html
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.pie.html

Data Analysis



Data Analysis

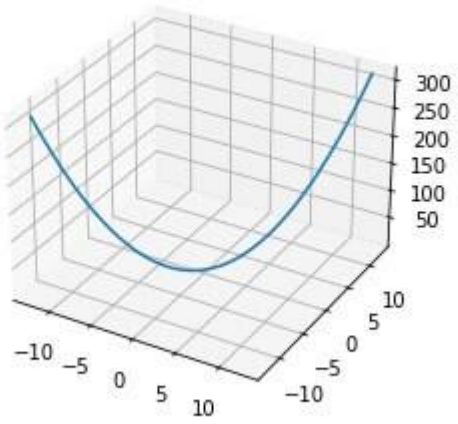
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(4,4))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(2,3,4) # plot the point (2,3,4) on the figure
plt.show()
```



https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.figure.html
https://www.statology.org/fig-add-subplot/

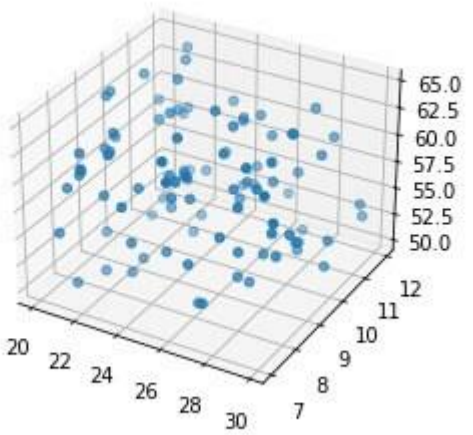
Data Analysis

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

x = np.linspace(-4 * np.pi, 4 * np.pi, 50)
y = np.linspace(-4 * np.pi, 4 * np.pi, 50)
z = x**2 + y**2
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot(x,y,z)

plt.show()
```



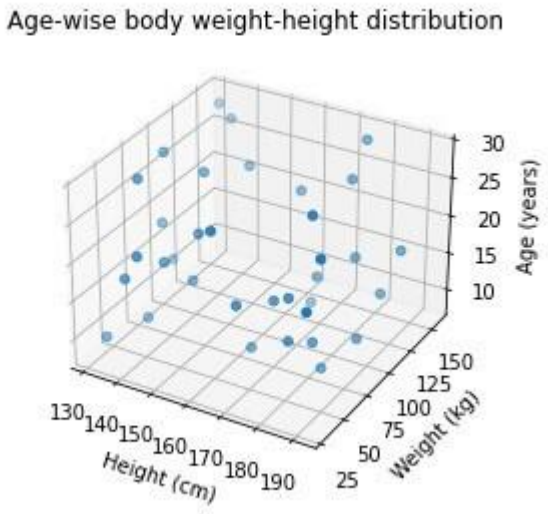
Data Analysis

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

np.random.seed(42)
xs = np.random.random(100)*10 + 20
ys = np.random.random(100)*5 + 7
zs = np.random.random(100)*15 + 50

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(xs,ys,zs)

plt.show()
```



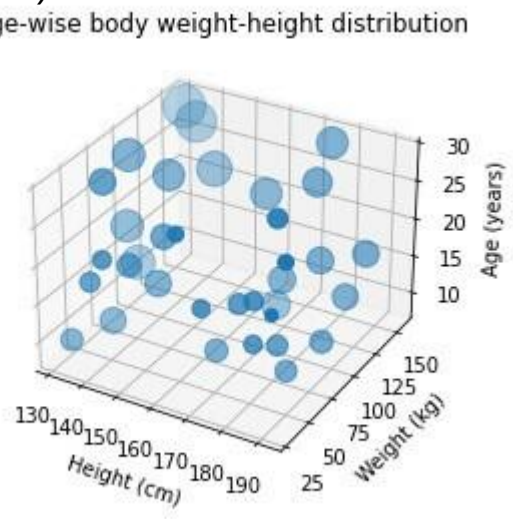
Data Analysis

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

np.random.seed(42)
ages = np.random.randint(low = 8, high = 30, size=35)
heights = np.random.randint(130, 195, 35)
weights = np.random.randint(30, 160, 35)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(xs = heights, ys = weights, zs = ages)
ax.set_title("Age-wise body weight-height distribution")
ax.set_xlabel("Height (cm)")
ax.set_ylabel("Weight (kg)")
ax.set_zlabel("Age (years)")

plt.show()
```



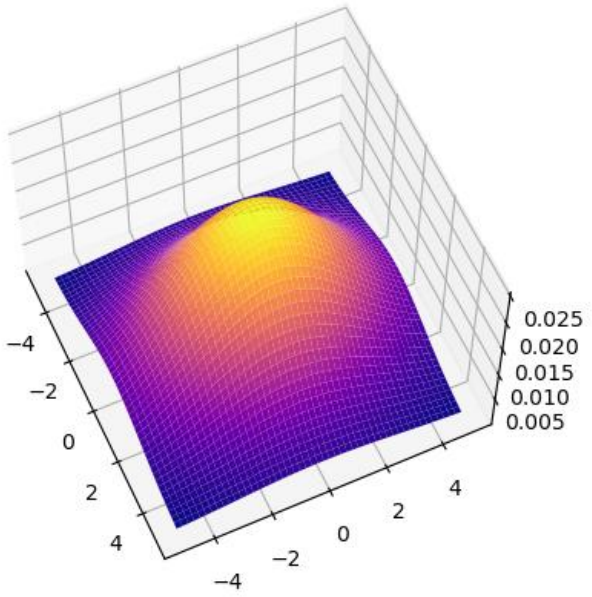
Data Analysis

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

np.random.seed(42)
ages = np.random.randint(low = 8, high = 30, size=35)
heights = np.random.randint(130, 195, 35)
weights = np.random.randint(30, 160, 35)
bmi = weights / ((heights * 0.01)**2)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(xs = heights, ys = weights, zs = ages, s=bmi*5 )
ax.set_title("Age-wise body weight-height distribution")
ax.set_xlabel("Height (cm)")
ax.set_ylabel("Weight (kg)")
ax.set_zlabel("Age (years)")
plt.show()
```



Data Analysis

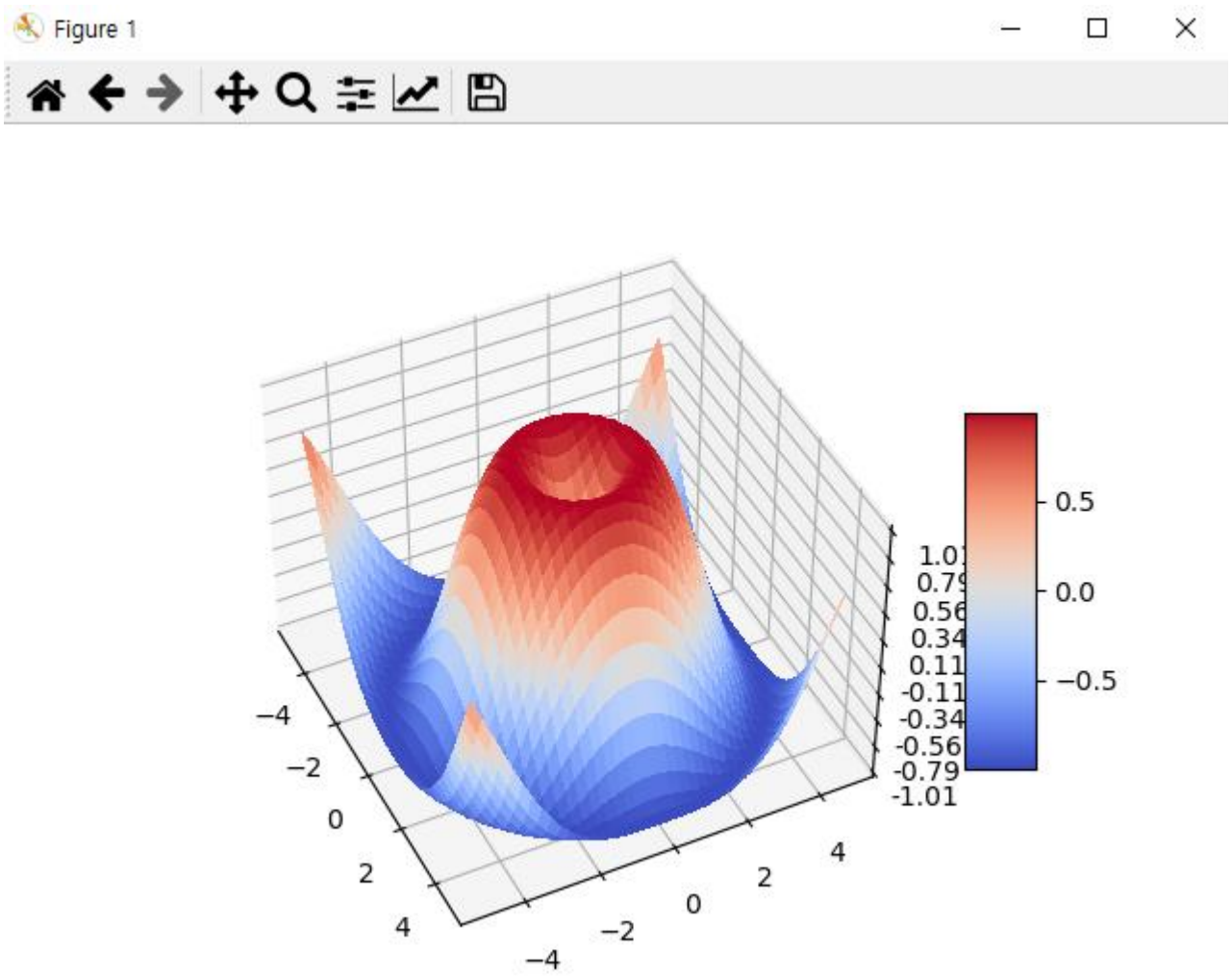
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
from scipy.stats import multivariate_normal

X = np.linspace(-5,5,50)
Y = np.linspace(-5,5,50)
X, Y = np.meshgrid(X,Y)
X_mean = 0; Y_mean = 0
X_var = 5; Y_var = 8

pos = np.empty(X.shape+(2,))
pos[:,:,0]=X
pos[:,:,1]=Y

rv = multivariate_normal([X_mean, Y_mean],[[X_var, 0], [0, Y_var]])

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, rv.pdf(pos), cmap="plasma")
plt.show()
```



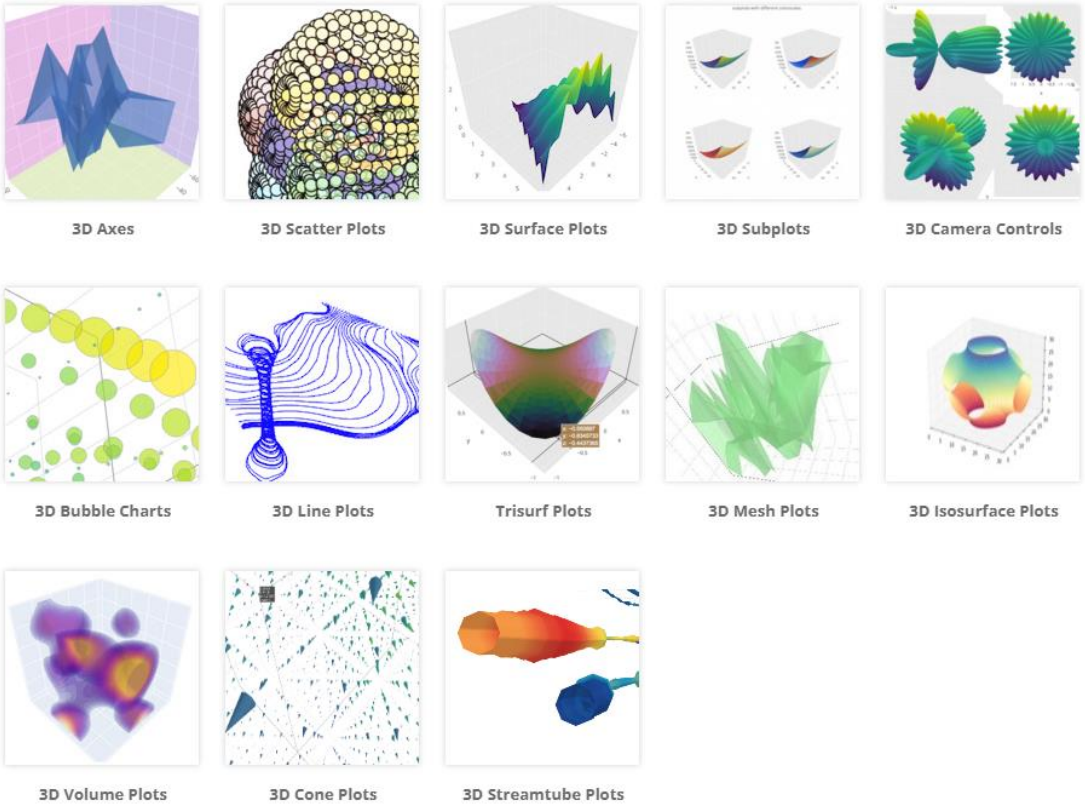
Data Analysis

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
import numpy as np


fig = plt.figure()
ax = fig.gca(projection='3d')

# Make data.
X = np.arange(-5, 5, 0.25)
Y = np.arange(-5, 5, 0.25)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)

# Plot the surface.
surf = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm,
                       linewidth=0, antialiased=False)

# Customize the z axis.
ax.set_zlim(-1.01, 1.01)
ax.zaxis.set_major_locator(LinearLocator(10))
ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))

# Add a color bar which maps values to colors.
fig.colorbar(surf, shrink=0.5, aspect=5)

plt.show()
```

Data Analysis



Data Analysis



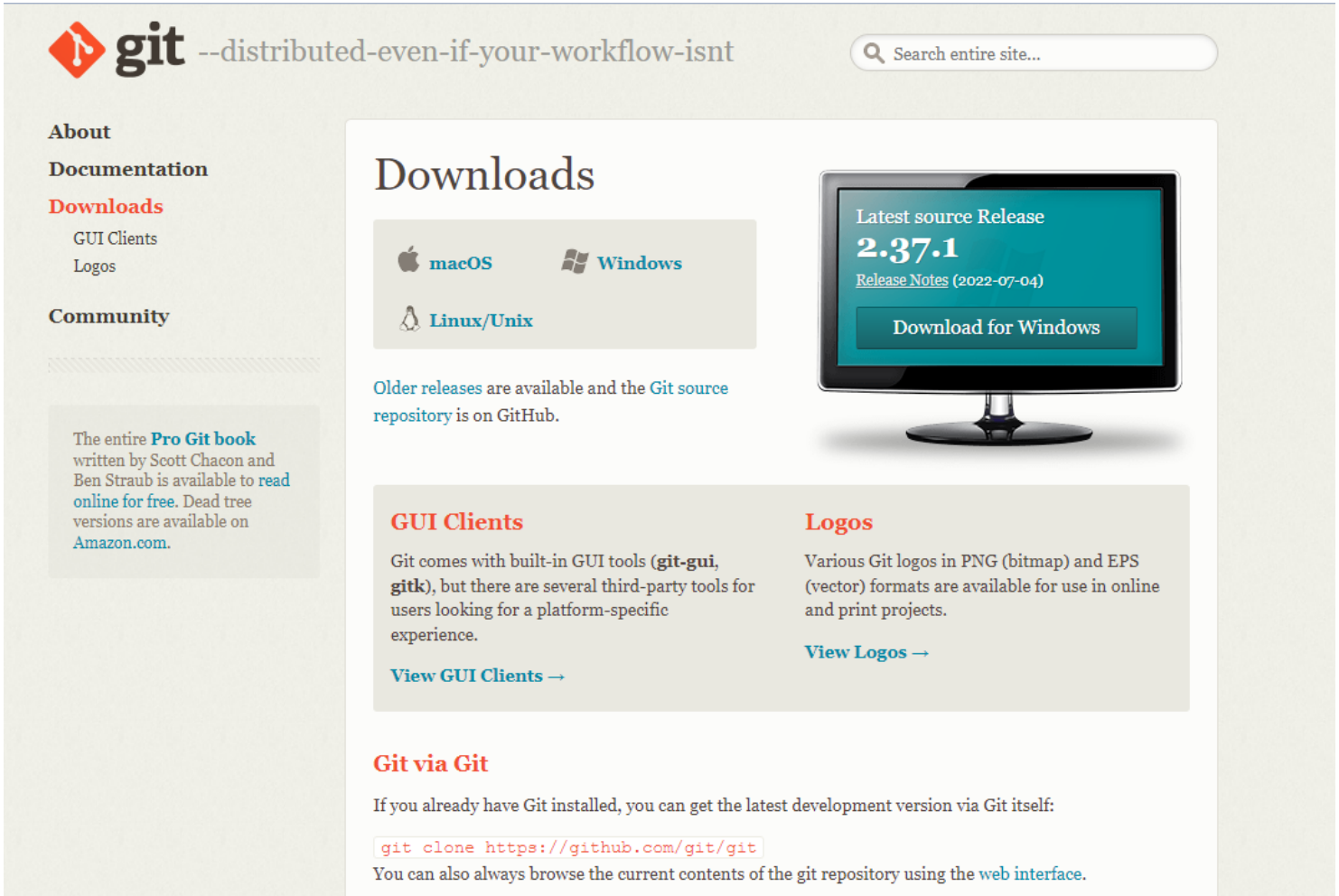
https://plotly.com/python/3d-charts/

Game

pip install pygame

Game

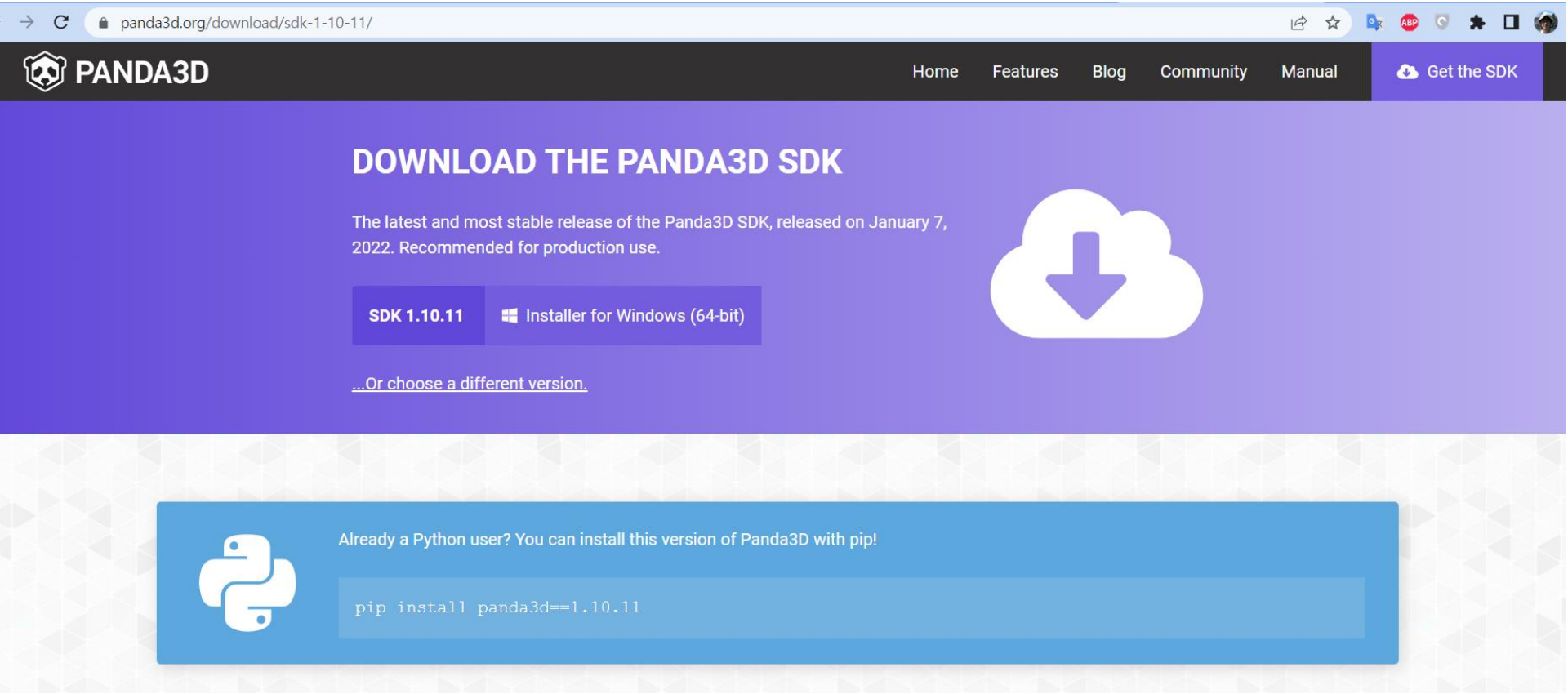
https://git-scm.com/downloads



https://github.com/Hakkush-07/pygame-3D

Game

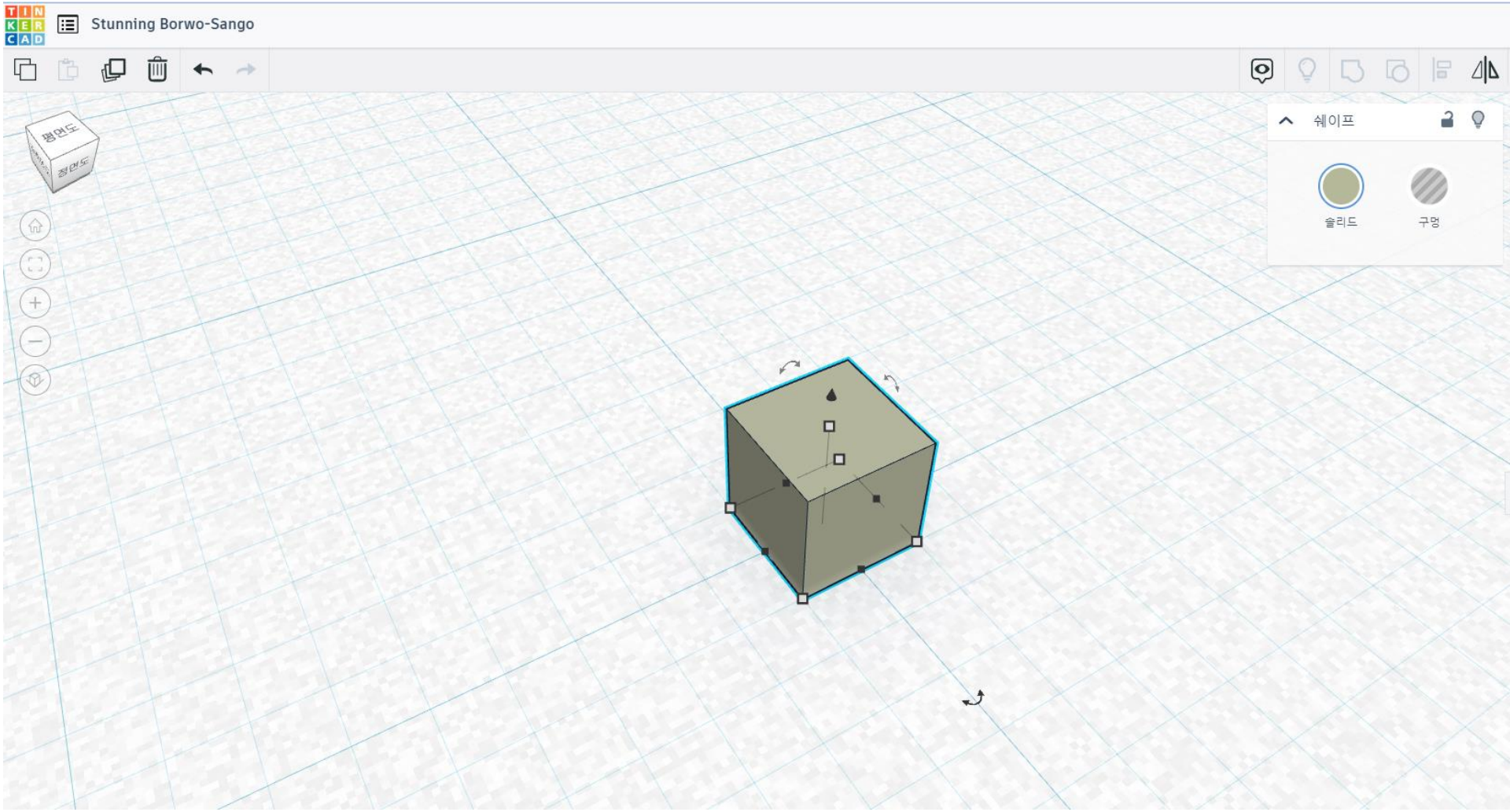
https://www.panda3d.org/download/sdk-1-10-11/



pip install panda3d

Modeling

https://www.tinkercad.com/things/fcTDgpaAdB3-stunning-borwo-sango/edit



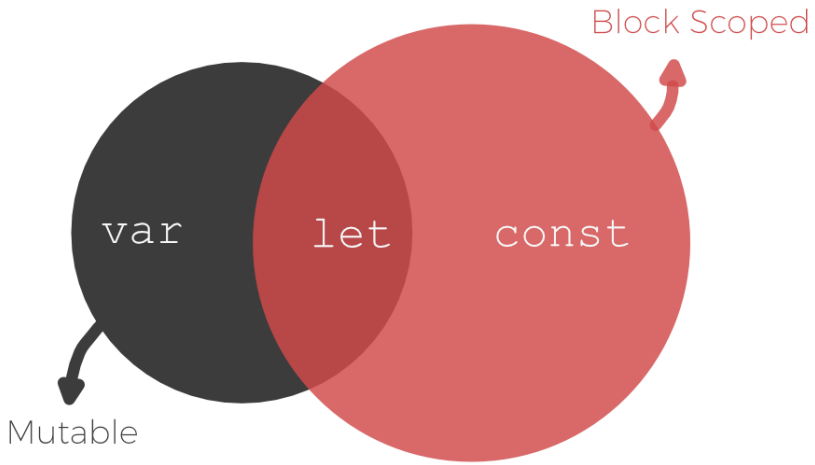
Modeling

```
import numpy as np
from stl import mesh

# Define the 8 vertices of the cube
vertices = np.array([₩
    [-1, -1, -1],
    [+1, -1, -1],
    [+1, +1, -1],
    [-1, +1, -1],
    [-1, -1, +1],
    [+1, -1, +1],
    [+1, +1, +1],
    [-1, +1, +1]])
# Define the 12 triangles composing the cube
faces = np.array([₩
    [0,3,1],
    [1,3,2],
    [0,4,7],
    [0,7,3],
    [4,5,6],
    [4,6,7],
    [5,1,2],
    [5,2,6],
    [2,3,6],
    [3,7,6],
    [0,1,5],
    [0,5,4]])

# Create the mesh
cube = mesh.Mesh(np.zeros(faces.shape[0], dtype=mesh.Mesh.dtype))
for i, f in enumerate(faces):
    for j in range(3):
        print(vertices[f[j],:])
        cube.vectors[i][j] = vertices[f[j]]

# Write the mesh to file "cube.stl"
cube.save('cube.stl')
```

javascript

javascript

```
let input;
if (input === undefined) {
  doThis();
} else {
  doThat();
}
```




```
const n = null;
console.log(n * 32); // Will log 0 to the
console
```

```
foo(); // "bar"

/* Function declaration */
function foo() {
  console.log('bar');
}
```

javascript

```
class Car {
  constructor(name, year) {
    this.name = name;
    this.year = year;
  }
  age() {
    let date = new Date();
    return date.getFullYear() - this.year;
  }
}

let myCar = new Car("Ford", 2014);
document.getElementById("demo").innerHTML =
"My car is " + myCar.age() + " years old.";
```

node.js

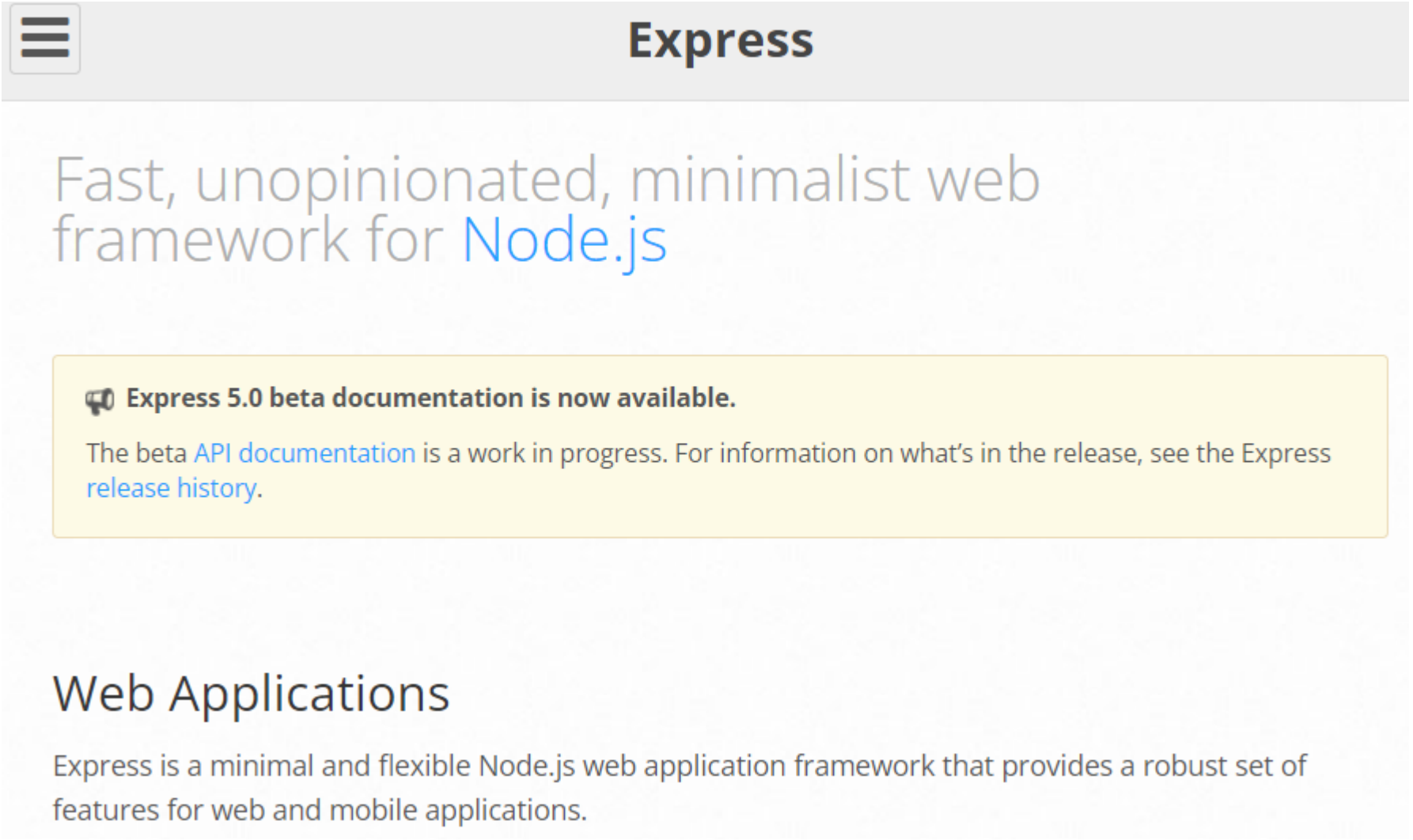
- About
- Governance
- Community
- Working Groups
- Releases
- Resources
- Trademark
- Privacy Policy

About Node.js®

As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications. In the following "hello world" example, many connections can be handled concurrently. Upon each connection, the callback is fired, but if there is no work to be done, Node.js will sleep.

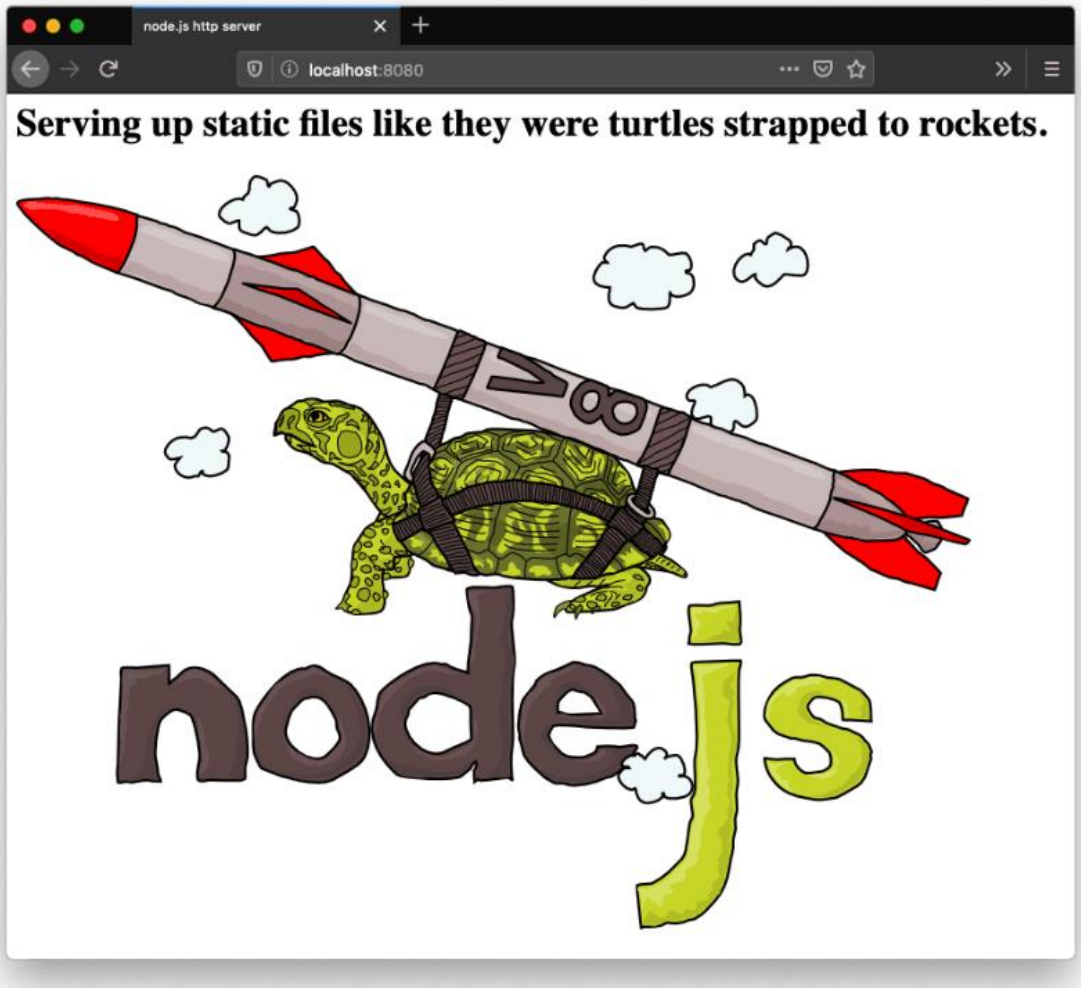
```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}
});
```

node.js

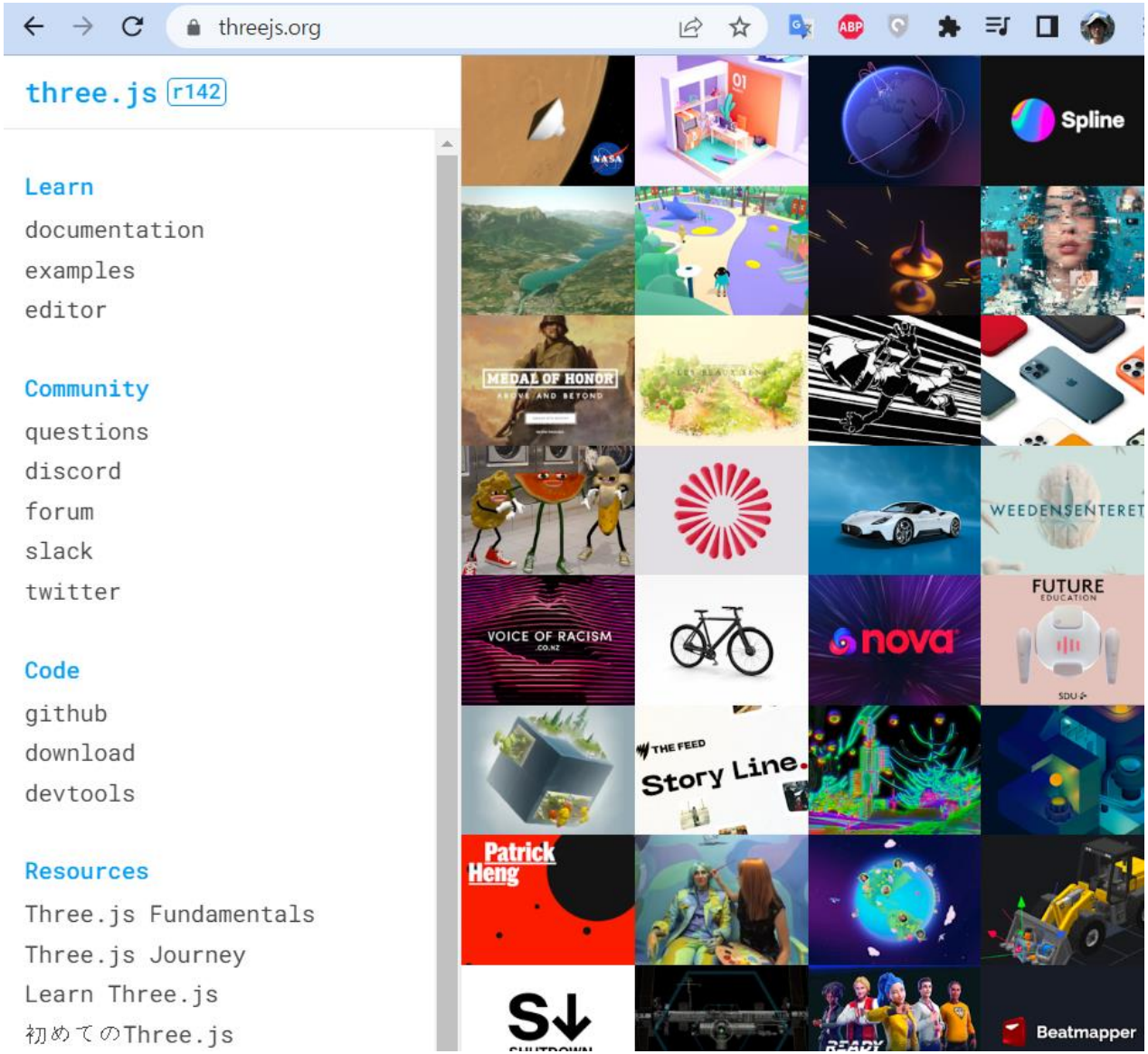

Express
Fast, unopinionated, minimalist web framework for Node.js
Express 5.0 beta documentation is now available.
The beta API documentation is a work in progress. For information on what's in the release, see the Express release history.
Web Applications
Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

node.js

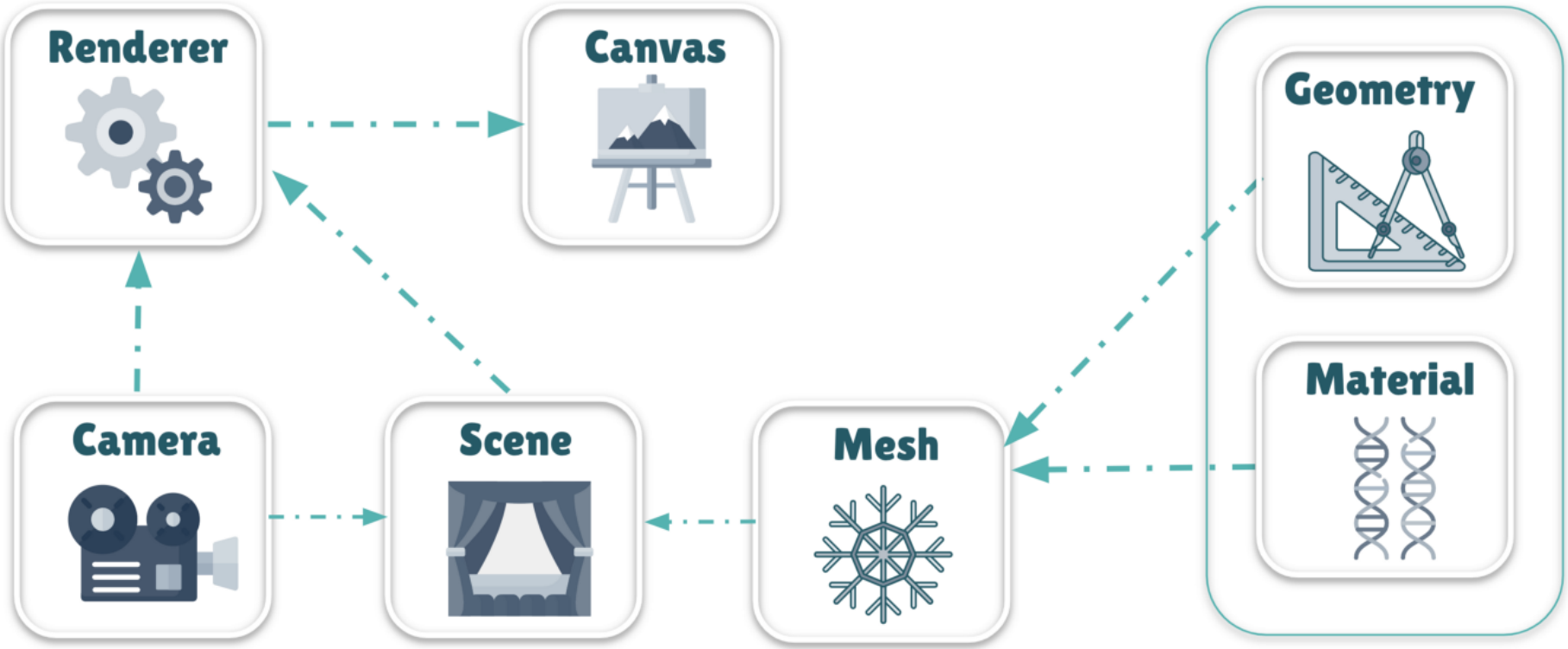
`http-server` is a simple, zero-configuration command-line static HTTP server. It is powerful enough for production usage, but it's simple and hackable enough to be used for testing, local development and learning.



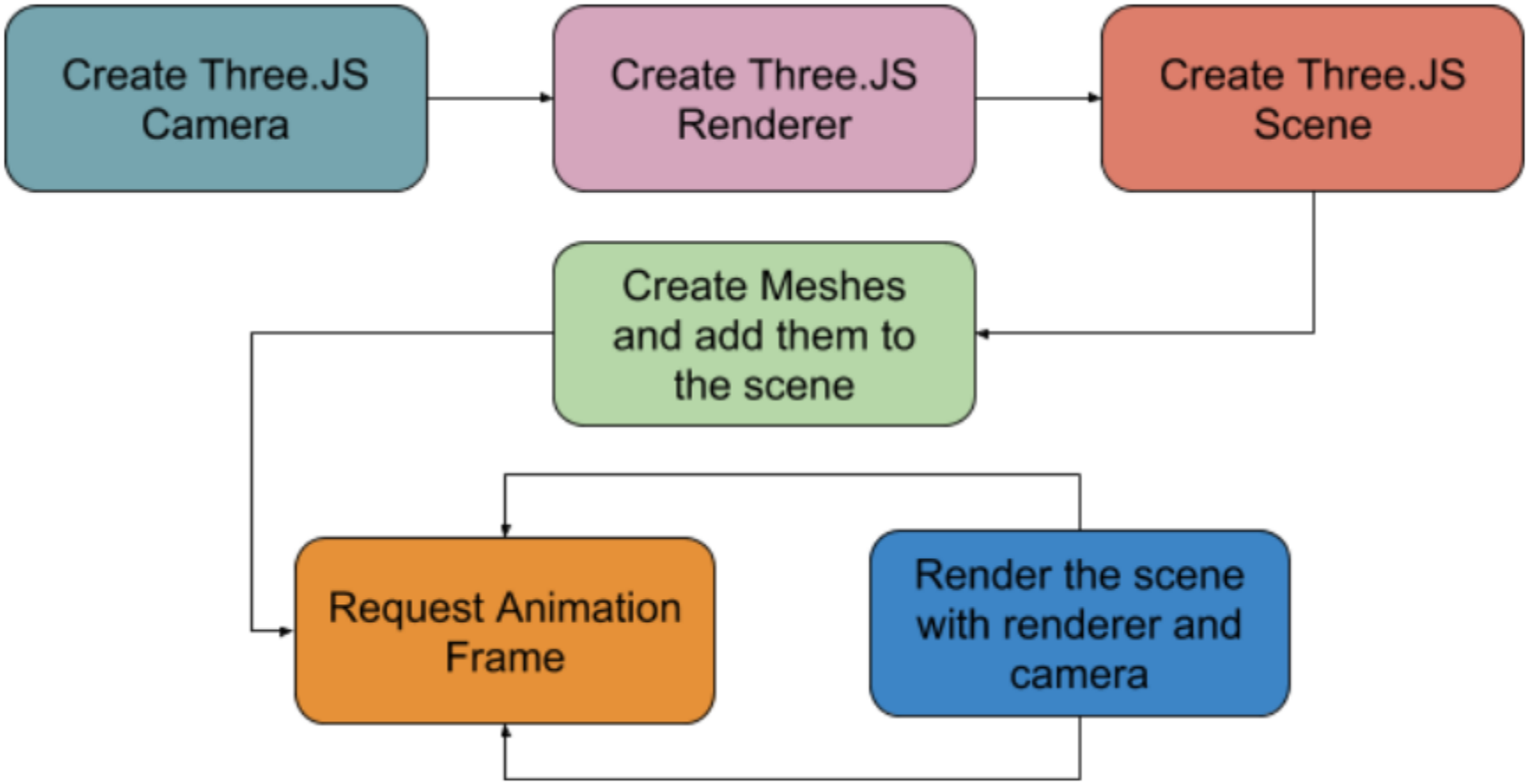
node.js


threejs.org
three.js r142
Learn
documentation
examples
editor
Community
questions
discord
forum
slack
twitter
Code
github
download
devtools
Resources
Three.js Fundamentals
Three.js Journey
Learn Three.js
初めてのThree.js
Spline
MEDAL OF HONOR
VOICE OF RACISM
nova
FUTURE
EDUCATION
THE FEED
Story Line.
Patrick
Heng
Beatmapper

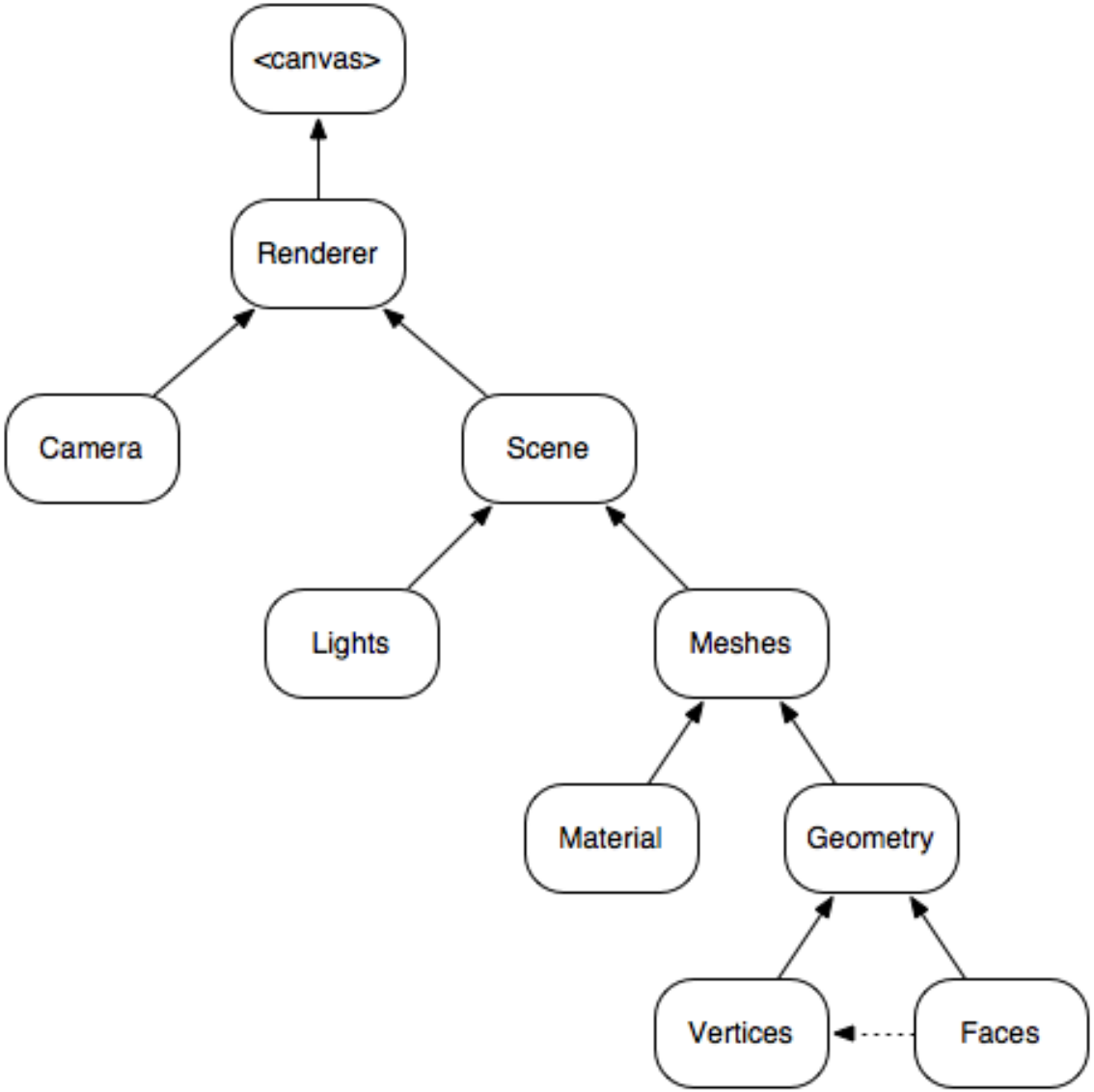
node.js



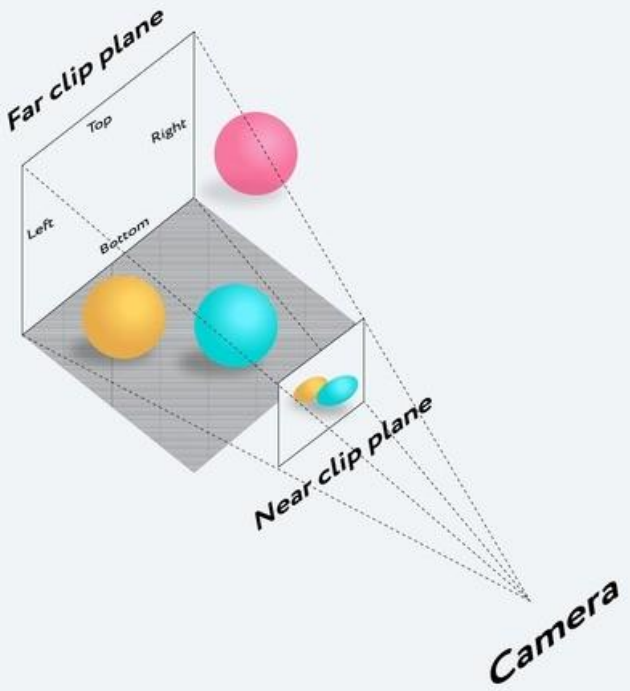
node.js


Create Three.JS Camera
Create Three.JS Renderer
Create Three.JS Scene
Create Meshes and add them to the scene
Request Animation Frame
Render the scene with renderer and camera

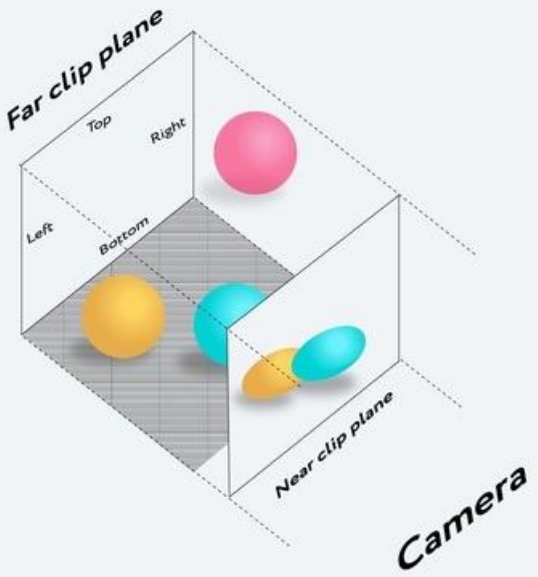
node.js


<canvas>
Renderer
Camera
Scene
Lights
Meshes
Material
Geometry
Vertices
Faces

node.js



Perspective



Orthographic

node.js

npm install express
npm install three
npm install http-server

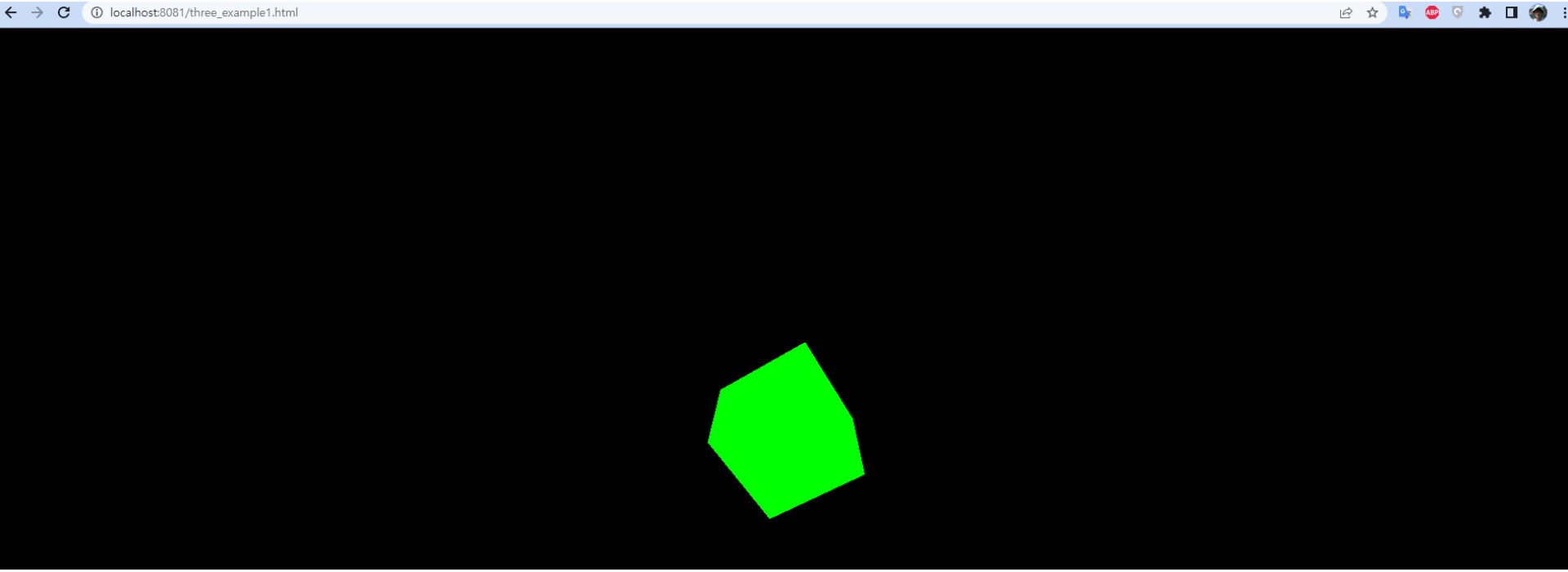
node.js

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My first three.js app</title>
    <style>
      body { margin: 0; }
    </style>
  </head>
  <body>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/three.js/r83/three.js"></script>
    <script>
      const scene = new THREE.Scene();
      const camera = new THREE.PerspectiveCamera( 75, window.innerWidth / window.innerHeight, 0.1,
1000 );

      const renderer = new THREE.WebGLRenderer();
      renderer.setSize( window.innerWidth, window.innerHeight );
      document.body.appendChild( renderer.domElement );
```

node.js

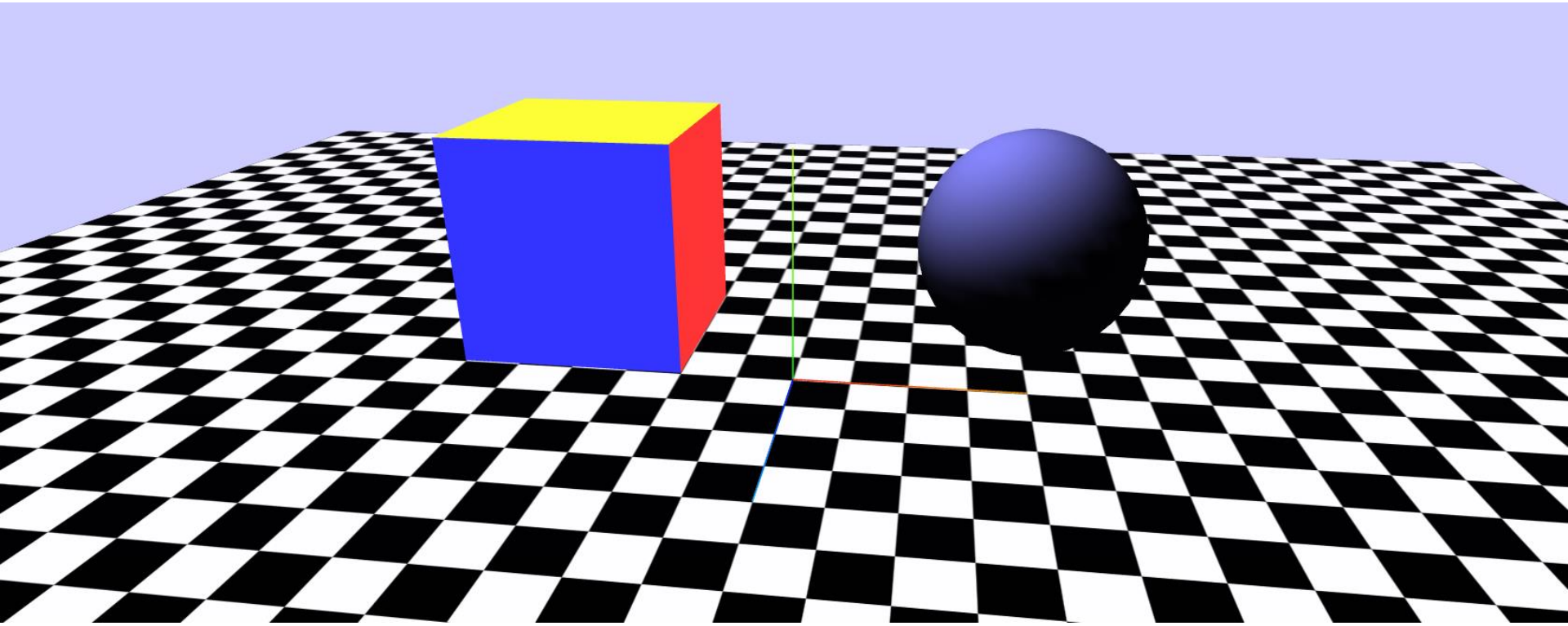
```
        const geometry = new THREE.BoxGeometry( 1, 1, 1 );
        const material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );
        const cube = new THREE.Mesh( geometry, material );
        scene.add( cube );

        camera.position.z = 5;

        function animate() {
            requestAnimationFrame( animate );

            cube.rotation.x += 0.01;
            cube.rotation.y += 0.01;

            renderer.render( scene, camera );
        };

        animate();
    </script>
  </body>
</html>
```

node.js



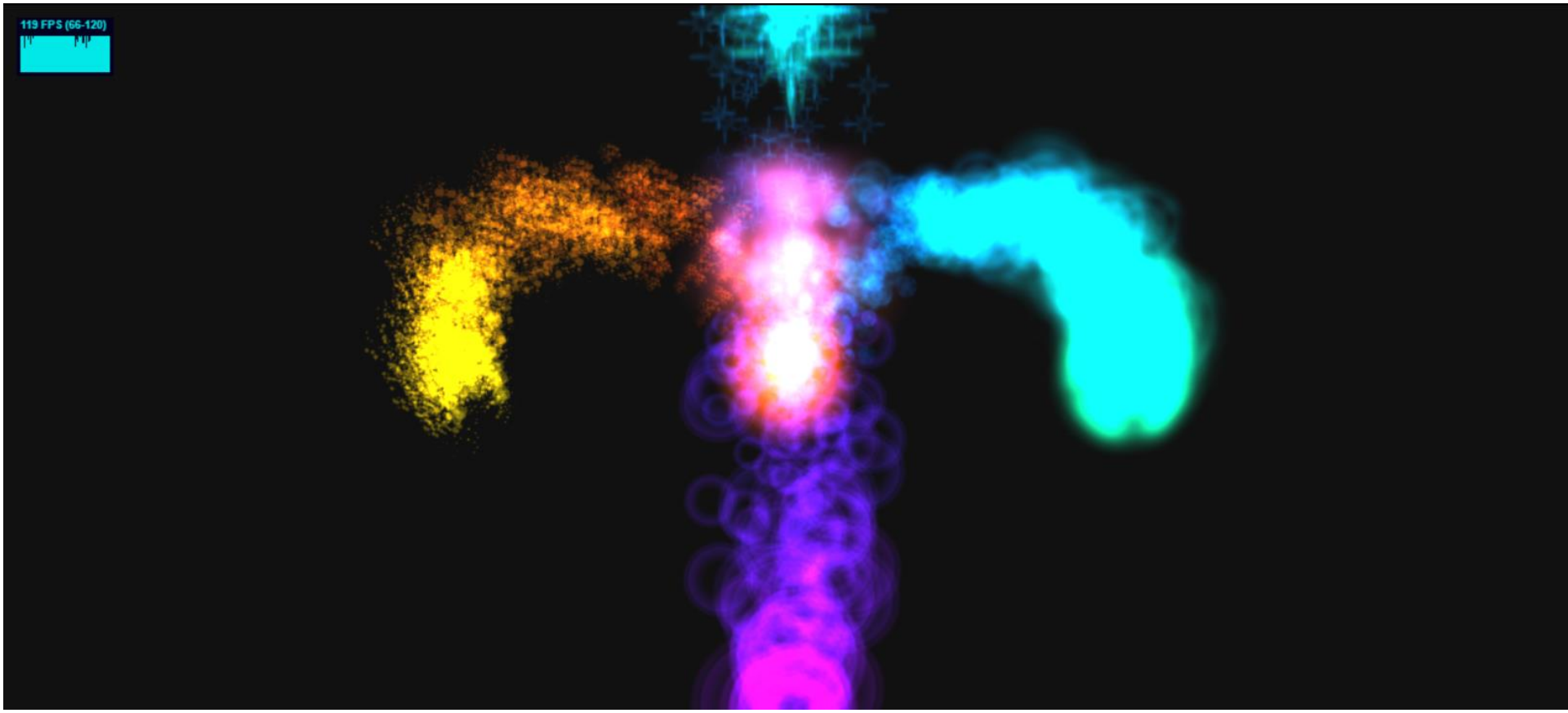
node.js

https://github.com/stemkoski/stemkoski.github.com



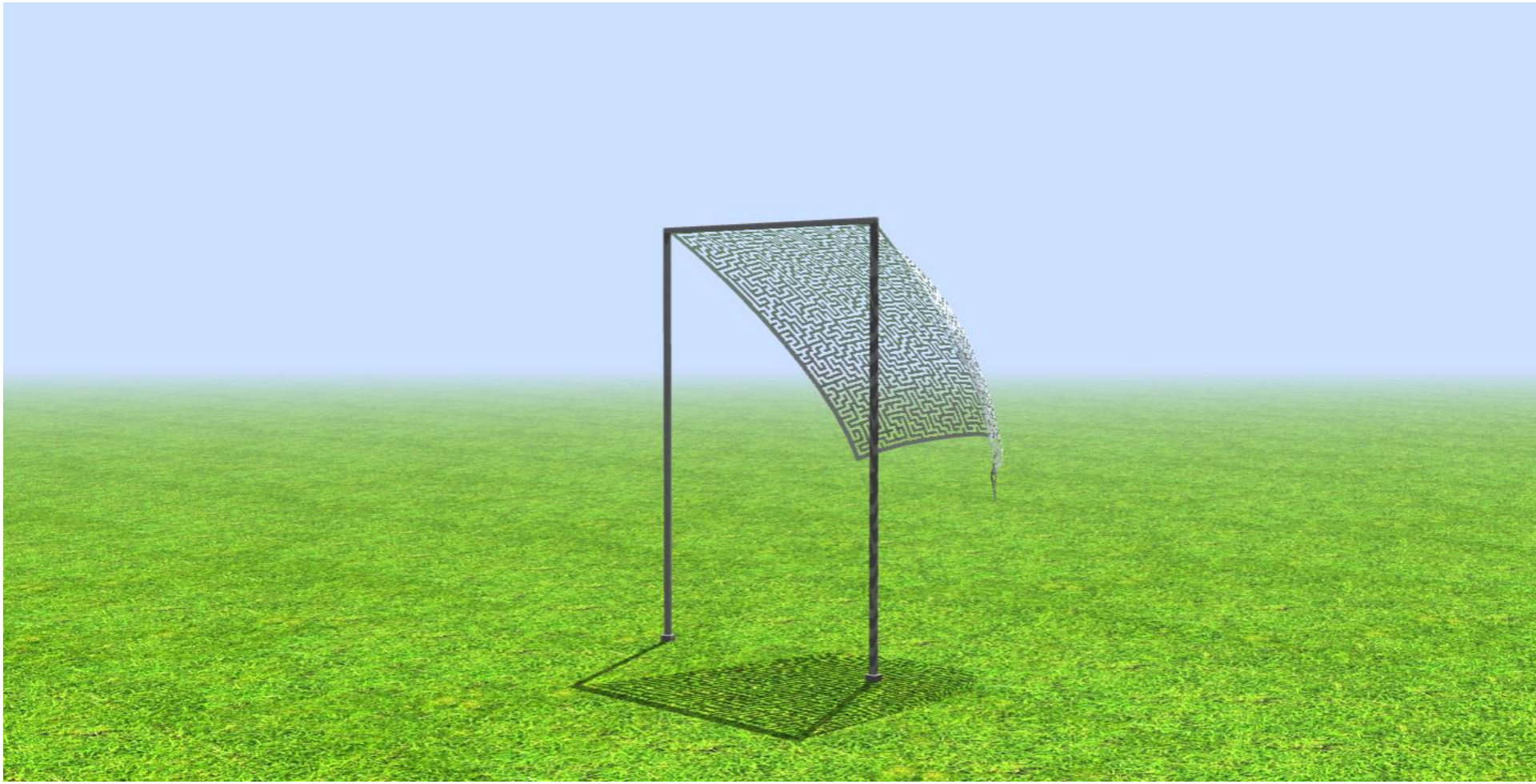
node.js

https://three-nebula.org/examples/gpu-renderer



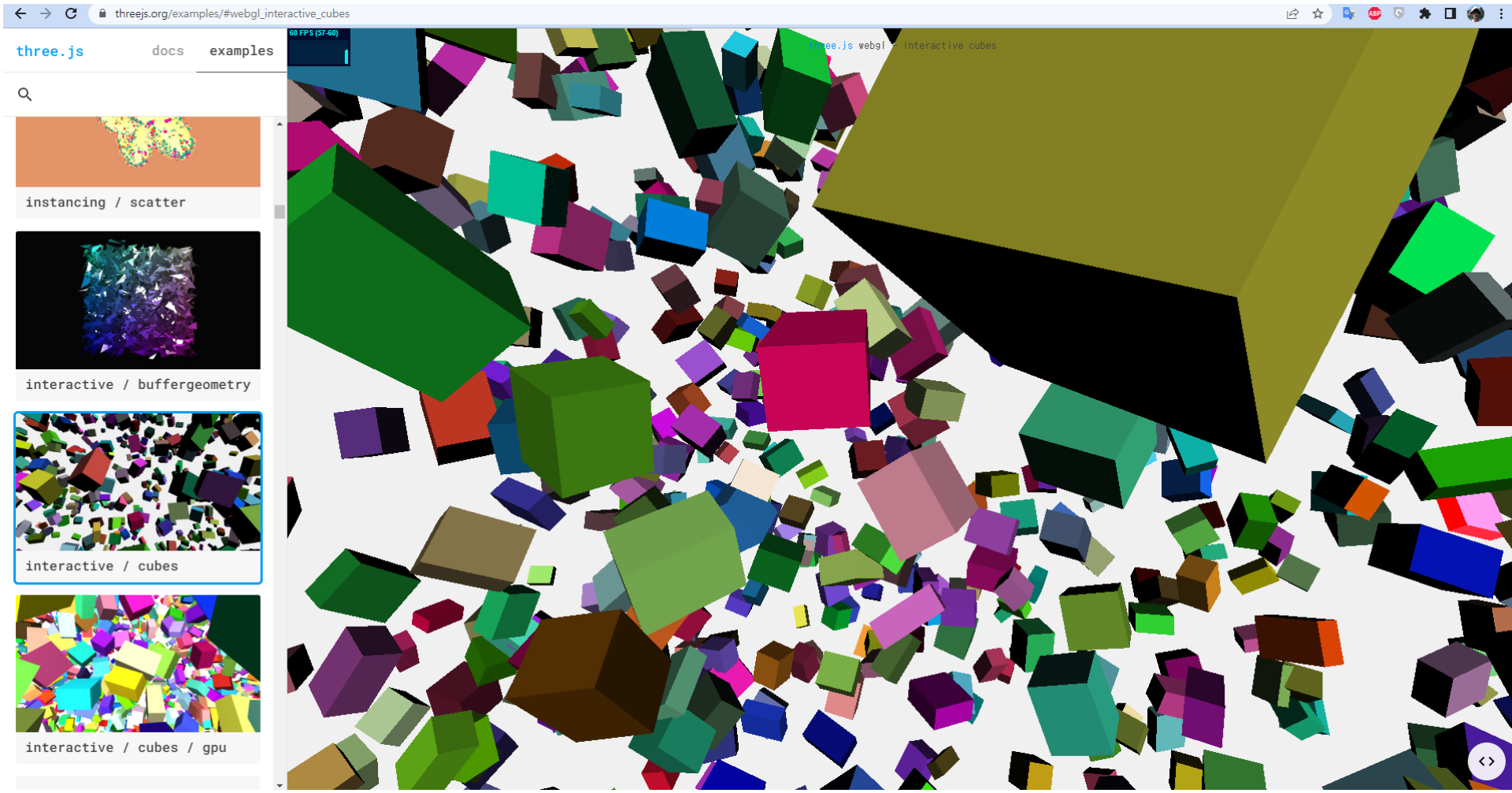
node.js

https://cs.wellesley.edu/~cs307/threejs/r67/examples/#webgl_animation_cloth



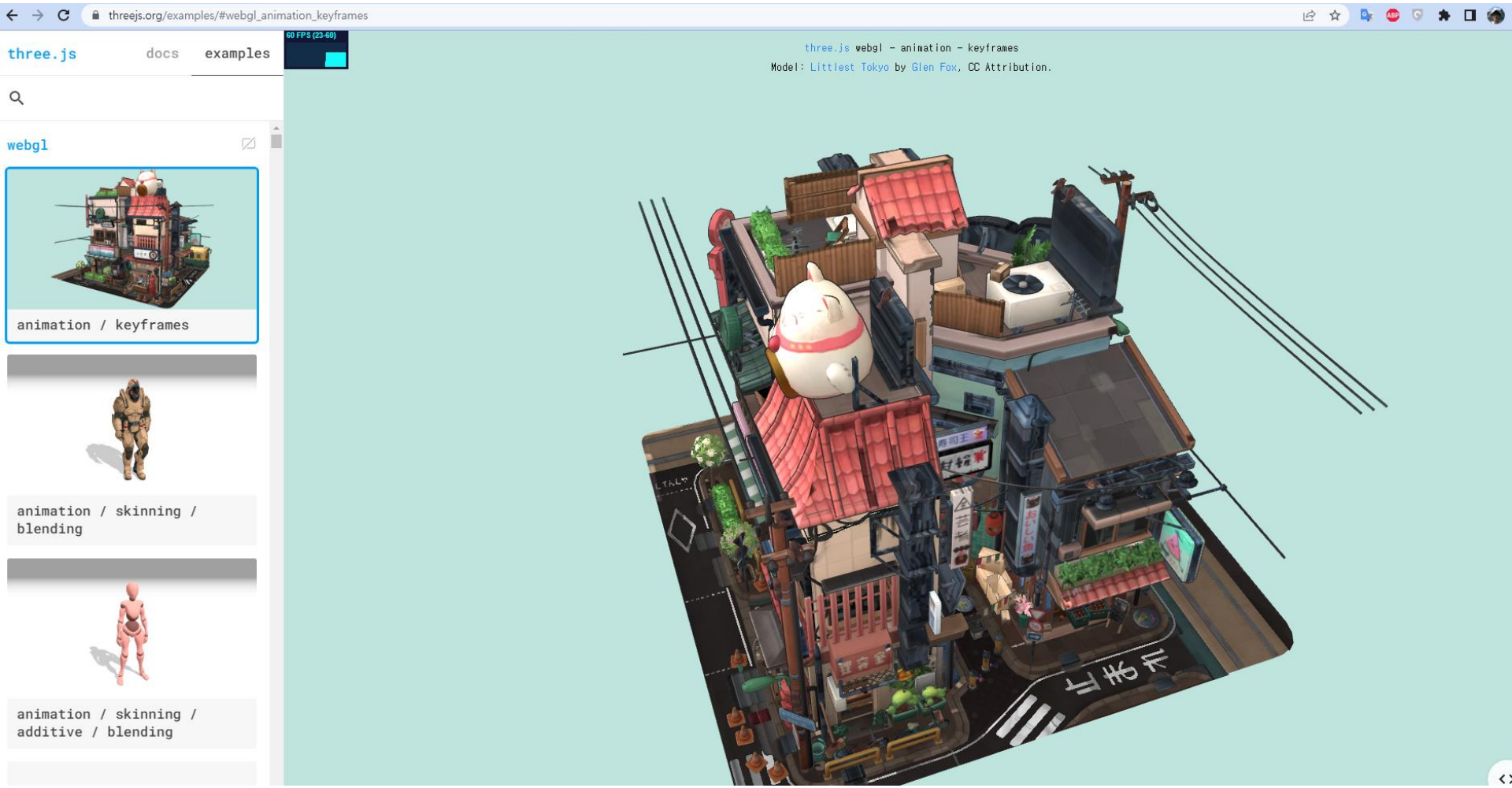
node.js

https://threejs.org/examples/#webgl_interactive_cubes



node.js

https://threejs.org/examples/#webgl_animation_keyframes



node.js

https://cesium.com/use-cases/digital-twins/

DX


HOW IT FEELS TO MASTER DIGITAL TRANSFORMATION
makeameme.org