In [1]:

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
import numpy as np

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.utils import to_categorical
```

In [2]:

```
(X_train, y_train), (X_test,y_test) = datasets.cifar10.load_data()
```

In [3]:

```
X_train.shape
```

Out[3]:

```
(50000, 32, 32, 3)
```

In [4]:

```
X_test.shape
```

Out[4]:

```
(10000, 32, 32, 3)
```

In [5]:

```
np.unique(X_train)
```

Out[5]:

```
array([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,
        26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,
        39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,
        52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,
        65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,
        78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
        91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103,
       104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
       117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129,
       130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
       143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155,
       156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168,
       169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181,
       182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194,
       195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,
       208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220,
       221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233,
       234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246,
       247, 248, 249, 250, 251, 252, 253, 254, 255], dtype=uint8)
```

In [6]:

```
y_train = y_train.reshape(-1,)
```

In [7]:

```
y_test = y_test.reshape(-1,)
```

In [8]:

```
# reshape
X_train = X_train.reshape((50000, 32, 32, 3))
X_test = X_test.reshape((10000, 32, 32, 3))
```

In [9]:

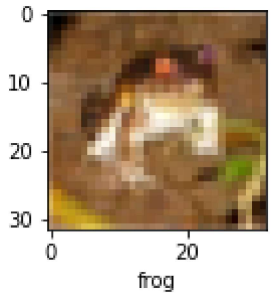
```
classes = ["airplane","automobile","bird","cat","deer","dog","frog","horse","ship","truck"]
```

In [10]:

```
def plot_sample(X, y, index):
    plt.figure(figsize = (15,2))
    plt.imshow(X[index])
    plt.xlabel(classes[y[index]])
```

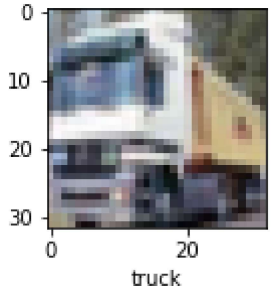
In [11]:

```
plot_sample(X_train, y_train, 0)
```



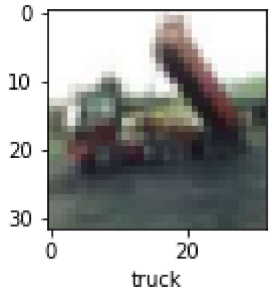
In [12]:

```
plot_sample(X_train, y_train, 1)
```



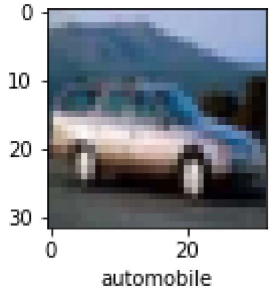
In [13]:

```
plot_sample(X_train, y_train, 2)
```



In [14]:

```
plot_sample(X_train, y_train, 4)
```



In [15]:

```
# convolution base
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 4, 4, 64)          36928
=================================================================
Total params: 56,320
Trainable params: 56,320
Non-trainable params: 0
_________________________________________________________________
```

In [16]:

```
model.add(Flatten())
model.add(Dense(10, activation='softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 4, 4, 64)          36928
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 10)                10250
=================================================================
Total params: 66,570
Trainable params: 66,570
Non-trainable params: 0
_________________________________________________________________
```

In [17]:

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'
model.fit(X_train, y_train, epochs=5)
```

```
Epoch 1/5
1563/1563 [==============================] - 34s 21ms/step - loss: 2.6786 -
accuracy: 0.3011
Epoch 2/5
1563/1563 [==============================] - 33s 21ms/step - loss: 1.3519 -
accuracy: 0.5155
Epoch 3/5
1563/1563 [==============================] - 32s 21ms/step - loss: 1.2093 -
accuracy: 0.5778
Epoch 4/5
1563/1563 [==============================] - 32s 21ms/step - loss: 1.1215 -
accuracy: 0.6080
Epoch 5/5
1563/1563 [==============================] - 32s 20ms/step - loss: 1.0528 -
accuracy: 0.6325
```

Out[17]:

```
<tensorflow.python.keras.callbacks.History at 0x1f6c659aa60>
```

In [18]:

```
test_loss, test_accuracy = model.evaluate(X_test,y_test)
print("test loss -",test_loss)
print("test accuracy -",test_accuracy)
```

```
313/313 [==============================] - 1s 4ms/step - loss: 1.1454 - accu
racy: 0.6112
test loss - 1.145400047302246
test accuracy - 0.6111999750137329
```

In [ ]:

```

```