Creating An Expression

12 min

POST is the HTTP method verb used for creating new resources. Because POST routes create new data, their paths do not end with a route parameter, but instead end with the type of resource to be created.

For example, to create a new monster, a client would make a POST request to /monsters. The client does not know the id of the monster until it is created and sent back by the server, therefore POST /monsters/:id doesn't make sense because a client couldn't know the unique id of a monster before it exists.

First, we create a new element using createElement() from **utils.js**:

```
app.post('/monsters', (req, res, next) => {
  const receivedExpression = createElement('monsters', req.query);
```

If our new element is valid, we push() it to our array of elements:

```
monsters.push(receivedExpression);
```

Express uses .post() as its method for POST

Preview: Docs XMLHttpRequest is a built-in browser object that allows HTTP requests to be made in JavaScript. It is basically used to fetch data from APIs.

requests

. POST requests can use many ways of sending data to create new resources, including query

Preview: Docs Strings are any grouping of characters (letters, spaces, numbers, or symbols) surrounded by quotes or backticks.

strings

.

The HTTP status code for a newly-created resource is 201 Created.

Instructions

1. Checkpoint 1 Passed

1.

Create a POST /expressions route. It should send, create, and add a new expression to the expressions array if it is a valid new expression (meaning it has an emoji and name key). It should send back the new element with a 201 status code if it is valid, and it should send a 400 status code if the object is not valid.

You can use the createElement(elementType, objectToCreate) helper function to create a valid expression. The first argument is the type of element, so it should be 'expressions' in this case. The second argument should be the query object with an emoji and a name property. This function will return false if the objectToCreate does not contain all necessary key-value pairs, and it will return the newly-created element if object to create is valid. It does not add the created element to any arrays, you will need to do so yourself.

Don’t forget to restart your server and test as you implement the functionality. To test your route, use the POST tab in the upper left corner. Select a name and emoji and send the request to see if your route works as intended.

Hint

You can use the .push() method of the expressions array to add a new element after it has been created, for example:

```
const newElement = createElement('emoji', {name: 'example', emoji: ':)'});
if (newElement) {
  elements.push(newElement);
}
```

app.js

```
const express = require('express');
const app = express();

// Serves Express Yourself website
app.use(express.static('public'));

const { getElementById, getIndexById, updateElement,
  seedElements, createElement } = require('./utils');

const expressions = [];
seedElements(expressions, 'expressions');

const PORT = process.env.PORT || 4001;

app.get('/expressions', (req, res, next) => {
```

```
  res.send(expressions);
});

app.get('/expressions/:id', (req, res, next) => {
  const foundExpression = getElementById(req.params.id, expressions);
  if (foundExpression) {
    res.send(foundExpression);
  } else {
    res.status(404).send();
  }
});

app.put('/expressions/:id', (req, res, next) => {
  const expressionIndex = getIndexById(req.params.id, expressions);
  if (expressionIndex !== -1) {
    updateElement(req.params.id, req.query, expressions);
    res.send(expressions[expressionIndex]);
  } else {
    res.status(404).send();
  }
});

// Add your POST handler below:
app.post('/expressions', (req, res, next) => {
  const newExpression = createElement('expressions', req.query);
  if (newExpression !== false) {
    expressions.push(newExpression);
    res.status(201).send(newExpression);
  } else {
    res.status(400).send();
```

```
  }
})

app.listen(PORT, () => {
  console.log(`Listening on port ${PORT}`);
});
```

utils.js

```
let expressionIdCounter = 0;
let animalIdCounter = 0;

const getElementById = (id, elementList) => {
  return elementList.find((element) => {
    return element.id === Number(id);
  });
};

const getIndexById = (id, elementList) => {
  return elementList.findIndex((element) => {
    return element.id === Number(id);
  });
};

const createElement = (elementType, queryArguments) => {
  if (queryArguments.hasOwnProperty('emoji') &&
      queryArguments.hasOwnProperty('name')) {
    let currentId;
    if (elementType === 'expressions') {
      expressionIdCounter += 1;
      currentId = expressionIdCounter;
```

```
    } else {
      animalIdCounter += 1;
      currentId = animalIdCounter;
    }
    return {
      'id':    currentId,
      'emoji': queryArguments.emoji,
      'name':  queryArguments.name,
    };
  } else {
    return false;
  }
};

const updateElement = (id, queryArguments, elementList) => {
  const elementIndex = getIndexById(id, elementList);
  if (elementIndex === -1) {
    throw new Error('updateElement must be called with a valid id
parameter');
  }
  if (queryArguments.id) {
    queryArguments.id = Number(queryArguments.id);
  }
  Object.assign(elementList[elementIndex], queryArguments);
  return elementList[elementIndex];
};

const seedElements = (arr, type) => {
  if (type === 'expressions') {
```

```
    arr.push(createElement('expressions', {'emoji': '😀', 'name':
'happy'}));

    arr.push(createElement('expressions', {'emoji': '😎', 'name':
'shades'}));

    arr.push(createElement('expressions', {'emoji': '😴', 'name':
'sleepy'}));

  } else if (type === 'animals') {

    arr.push(createElement('animals', {'emoji': '🐶', 'name':
'Pupper'}));

    arr.push(createElement('animals', {'emoji': '🐍', 'name':
'Snek'}));

    arr.push(createElement('animals', {'emoji': '😺', 'name':
'Maru'}));

  } else {

    throw new Error(`seed type must be either 'expression' or
'animal'`);

  }

};


module.exports = {

  createElement: createElement,

  getIndexById: getIndexById,

  getElementById: getElementById,

  updateElement: updateElement,

  seedElements: seedElements,

};
```