EXPRESS ROUTES CODE CHALLENGES

Code Challenge

Code Challenge

Instructions

1.

Require the `'express'` package and save it to a variable named `express`. Then, create an Express instance and save it to a variable called `app`.

app.js

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

Code Challenge

Code Challenge

Instructions

1.

Start your server listening on the port defined by the `PORT` variable.

app.js

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

const app = express();

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

// Add your code below:
app.listen(PORT);
```

Code Challenge

Code Challenge

Instructions

1.

Write a GET /sausages route that sends back the whole `sausageTypes` array.

app.js

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

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

const sausageTypes = ['bratwurst', 'andouille', 'chorizo', 'boudin', 'kielbasa'];

app.get('/sausages', (req, res, next) => {
  res.send(sausageTypes);
})

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

Code Challenge

Code Challenge

Instructions

1.

Fix the route so that it sends back the array of metal building materials.

app.js

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

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

const buildingMaterials = {
  wood: ['plywood', '2x4s', 'cedar shingles'],
  metal: ['steel girders', 'wall studs', 'rebar'],
};

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

```
app.get('/metals', (req, res, next) => {
  const arrayToSend = buildingMaterials.metal;
  res.send(arrayToSend);
});
```

Code Challenge

Code Challenge

Instructions

1.

Complete the GET /battlefields/:name route. Send back the battlefield object if a battlefield exists, and set the status to 404 and send an empty response if it does not.

app.js

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

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

const battlefields = {
  fortSumter: {
    state: 'SC',
  },
  manassas: {
    state: 'VA',
  },
  gettysburg: {
    state: 'PA',
  },
  antietam: {
    state: 'MD',
  }
}

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

```
app.get('/battlefields/:name', (req, res, next) => {
  const battlefieldName = req.params.name;
  const battlefield = battlefields[battlefieldName];
  if (battlefield) {
    res.send(battlefield);
  } else {
    res.status(404).send();
  }
});
```

Code Challenge

Code Challenge

Instructions

1.

Write a route to handle PUT requests to `/currencies/:name/countries`.

The `:name` param represents the currency name in the `currencies` object. The updated array of countries that use this currency will be sent in a query.

This route handler should replace the `countries` array of the correct single-currency object with an updated array from the query. It should send the updated array as a response.

Hint

Checkout the Express docs for help on using req.params and req.query!

app.js

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

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

const currencies = {
  diram: {
    countries: ['UAE', 'Morocco'],
  },
  real: {
    countries: ['Brazil'],
  },
```

```
  dinar: {
    countries: ['Algeria', 'Bahrain', 'Jordan', 'Kuwait'],
  },
  vatu: {
    countries: ['Vanuatu'],
  },
  shilling: {
    countries: ['Tanzania', 'Uganda', 'Somalia', 'Kenya'],
  },
};

app.put('/currencies/:name/countries', (req, res, next) => {
  const countries = req.query;
  currencies[req.params.name] = countries;
  res.send(currencies[req.params.name]);
});

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

Code Challenge

Code Challenge

Instructions

1.

Create a POST /soups route. It should add a new soup name to the `soups` array from the `name` property of the `req.query` object. It should also set a status code for 'Created'

app.js

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

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

const soups = ['gazpacho', 'borscht', 'primordial', 'avgolemono', 'laksa'];

app.post('/soups', (req, res, next) => {
```

```
  const newSoup = req.query.name;
  soups.push(newSoup);
  res.status(201).send(newSoup);
});

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

Code Challenge

Code Challenge

Instructions

1.

Write a route to handle DELETE requests to `/puddings/:flavor`. It should delete the correct pudding and send a 204 response if the pudding name exists and send a 404 response if it does not.

You can use the helper functions `findPuddingIndex` to find the index of the pudding by flavor and `deletePuddingAtIndex` to delete a pudding from the `puddingFlavors` array by index.

Hint

`findPuddingIndex` returns -1 if a pudding name is not included in `puddingFlavors`.

app.js

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

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

const puddingFlavors = ['chocolate', 'banana', 'butterscotch', 'pistachio'];

const findPuddingIndex = (name) => {
  return puddingFlavors.indexOf(name);
}

const deletePuddingAtIndex = (index) => {
```

```
  puddingFlavors.splice(index, 1);
}

// Your code here!
app.delete('/puddings/:flavor', (req, res, next) => {
  const puddingIndex = findPuddingIndex(req.params.flavor);
  if (puddingIndex !== -1) {
    deletePuddingAtIndex(puddingIndex);
    res.status(204).send();
    } else {
        res.status(404).send();
    }
});


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

Code Challenge

Code Challenge

Instructions

1.

Mount the `sauceRouter` with `app.use` so that a GET /sauces request sends back the `sauces` array.

Hint

To use a router with `app.use()`, the syntax is `app.use(path, routerInstance);`

app.js

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

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

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

const pastas = ['agnolotti', 'cavatelli', 'gemelli', 'tortellini'];

app.get('/pastas', (req, res, next) => {
  res.send(pastas);
});

const sauceRouter = express.Router();
// Add your code here:
app.use('/sauces', sauceRouter);

const sauces = ['carbonara', 'primavera', 'bolognese', 'puttanesca', 'fra diavolo
']

sauceRouter.get('/', (req, res, next) => {
  res.send(sauces);
});
```

Code Challenge
Code Challenge

Instructions

1.
Create two routers - `mountainsRouter` and `mountainRangesRouter`. Mount them at `/mountains` and `/mountain-ranges`, respectively.

Create a route handler with `mountainsRouter` to send back the `mountains` array in response to a GET /mountains request.

Create a route handler with mountainRangesRouter to send back the `mountainRanges` array in response to a GET /mountain-ranges request.

Hint
Use `app.use()` to mount the routers at their specific base paths, and then set up `.get()` handlers for each router at their base path, `'/'`.

app.js

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

const mountainsRouter = express.Router();
const mountainRangesRouter = express.Router();

app.use('/mountains', mountainsRouter);
app.use('/mountain-ranges', mountainRangesRouter);

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

mountainsRouter.get('/', (req, res, next) => {
  res.send(mountains);
});

mountainRangesRouter.get('/', (req, res, next) => {
  res.send(mountainRanges);
});

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

const mountains = ['denali', 'olympus', 'kilimanjaro', 'matterhorn'];
const mountainRanges = ['alps', 'andes', 'himalayas', 'rockies'];
```