Προγραμματισμός και Συστήματα στον Παγκόσμιο Ιστό

NodeJS
Express

Δρ. Δημήτριος Κουτσομητρόπουλος

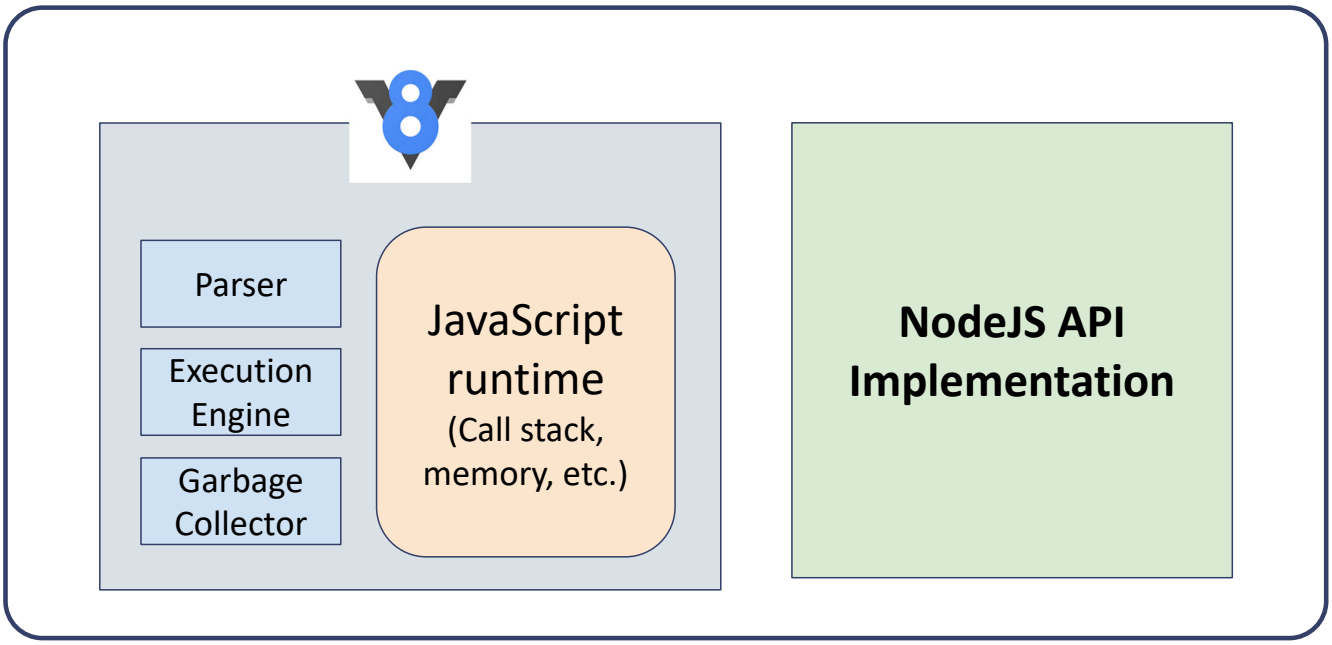
NodeJS

NodeJS:
- Περιβάλλον εκτέλεσης JavaScript (runtime) γραμμένο σε C++.
- Μπορεί να ερμηνεύσει και να εκτελέσει JavaScript.
- Περιλαμβάνει το NodeJS API.

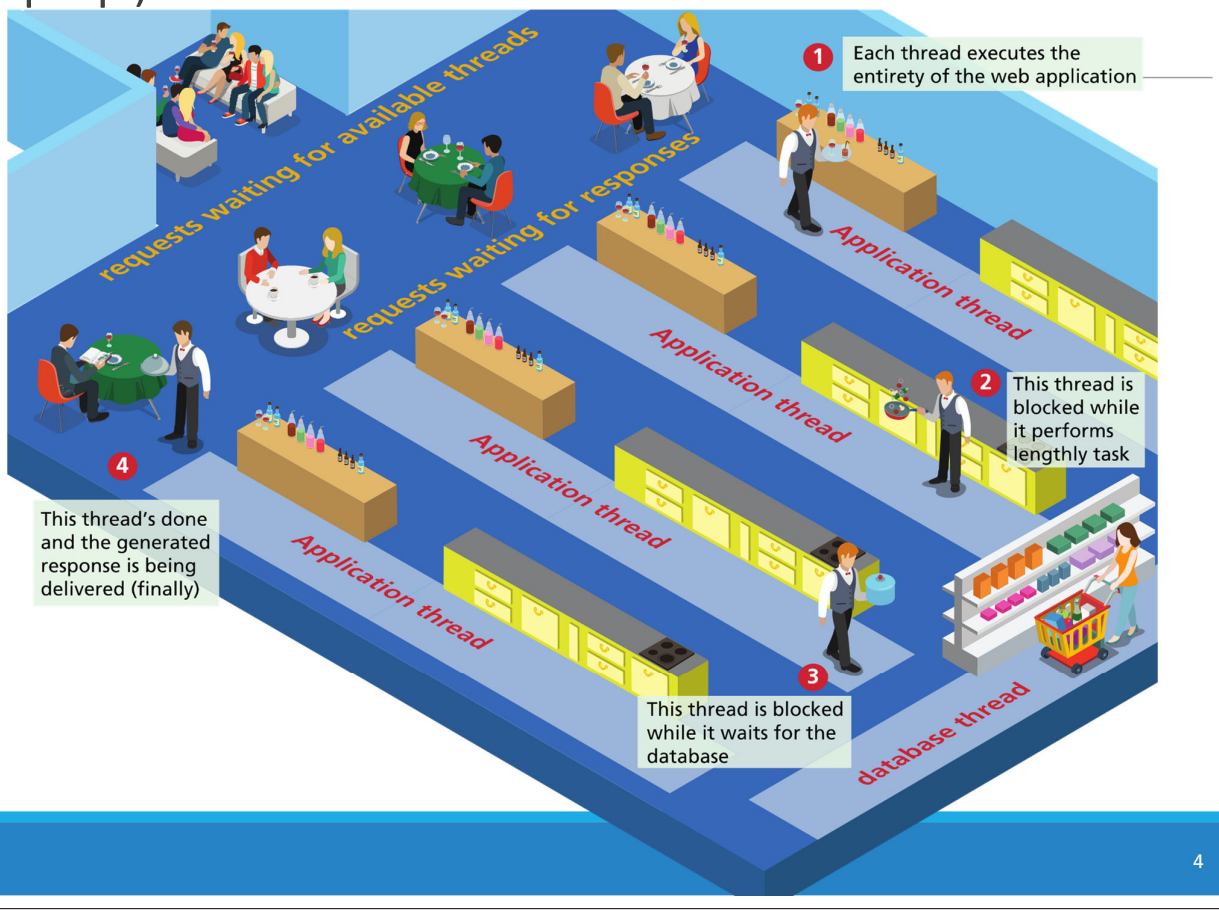
NodeJS API:
- Σύνολο από βιβλιοθήκες JavaScript χρήσιμων για τη δημιουργία προγραμμάτων.

V8 (από τον Chrome):
- Ο διερμηνευτής JavaScript ("engine") που χρησιμοποιεί το NodeJS για να διερμηνεύσει, να μεταγλωττίσει και να εκτελέσει JavaScript κώδικα.

NodeJS, V8, NodeJS APIs



Multi-threaded αρχιτεκτονική (apache, php)



Single-threaded nodeJS

εντολή node

Εκτελώντας node χωρίς όνομα αρχείου τρέχει ένα REPL loop

- Παρόμοια με την κονσόλα:

```
$ node
> let x = 5;
undefined
> x++
5
> x
6
```

- Εκτέλεση αρχείου JS:

```
$ node simple-script.js
```

Node για servers

server.js:

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

const server = http.createServer();

server.on('request', function(req, res) {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.on('listening', function() {
  console.log('Server running!');
});

server.listen(3000);
```


Node για servers

Include the HTTP NodeJS library

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

const server = http.createServer();
```

When the server gets a request, send back "Hello World" in plain text

```
server.on('request', function(req, res) {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});
```

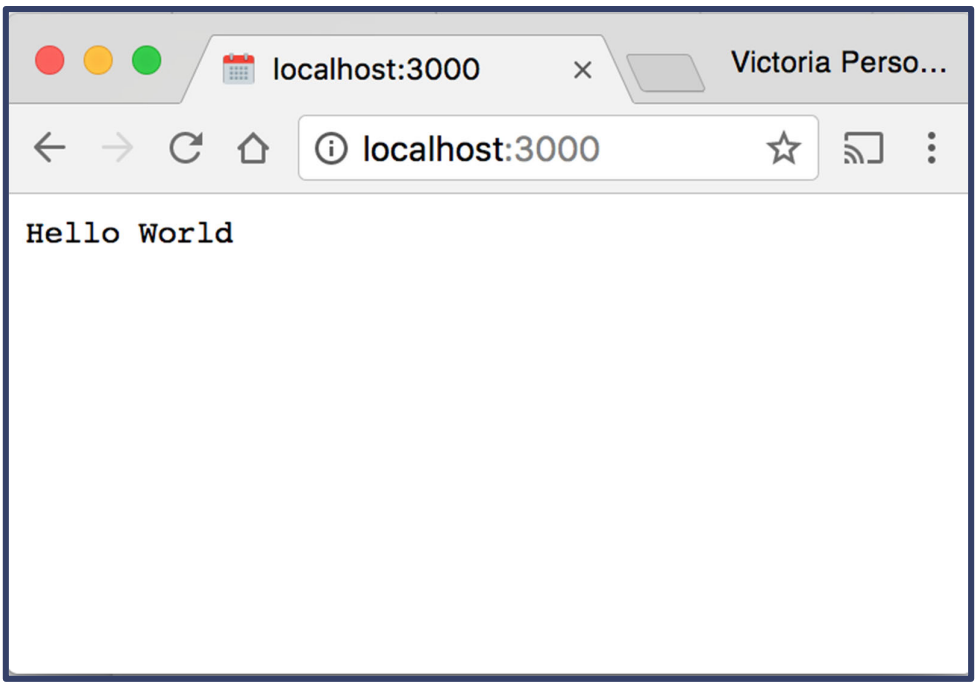
When the server is started, print a log message

```
server.on('listening', function() {
  console.log('Server running!');
});
```

Start listening for messages!

```
server.listen(3000);
```

Server response



Node για servers

Τα NodeJS server APIs είναι χαμηλού επιπέδου:

- Το request φτιάχνεται χειρωνακτικά
- Το response φτιάχνεται χειρωνακτικά
- Χρειάζεται πολύς κώδικας για επεξεργασία

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

http.createServer(function(request, response) {
  var headers = request.headers;
  var method = request.method;
  var url = request.url;
  var body = [];
  request.on('error', function(err) {
    console.error(err);
  }).on('data', function(chunk) {
    body.push(chunk);
  }).on('end', function() {
    body = Buffer.concat(body).toString();
    // BEGINNING OF NEW STUFF

    response.on('error', function(err) {
      console.error(err);
    });

    response.statusCode = 200;
    response.setHeader('Content-Type', 'application/json');
    // Note: the 2 lines above could be replaced with this next one:
    // response.writeHead(200, {'Content-Type': 'application/json'})

    var responseBody = {
      headers: headers,
      method: method,
      url: url,
      body: body
    };

    response.write(JSON.stringify(responseBody));
    response.end();
    // Note: the 2 lines above could be replaced with this next one:
    // response.end(JSON.stringify(responseBody))

    // END OF NEW STUFF
  });
}).listen(8080);
```

ExpressJS

ExpressJS

Χρήση της βιβλιοθήκης ExpressJS για server-side λειτουργίες στο Node:

Χωρίς ExpressJS:

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

const server = http.createServer();

server.on('request', function(req, res) {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.on('listening', function() {
  console.log('Server running!');
});

server.listen(3000);
```

Με ExpressJS:

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

app.get('/', function (req, res) {
  res.send('Hello World!');
})

app.listen(3000, function () {
  console.log('Example app listening on port 3000!');
})
```

ExpressJS

Το Express δεν περιλαμβάνεται στο NodeJS APIs.

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

```
module.js:327
    throw err;
    ^

Error: Cannot find module 'express'
    at Function.Module._resolveFilename
```

Χρειάζεται εγκατάσταση μέσω npm.

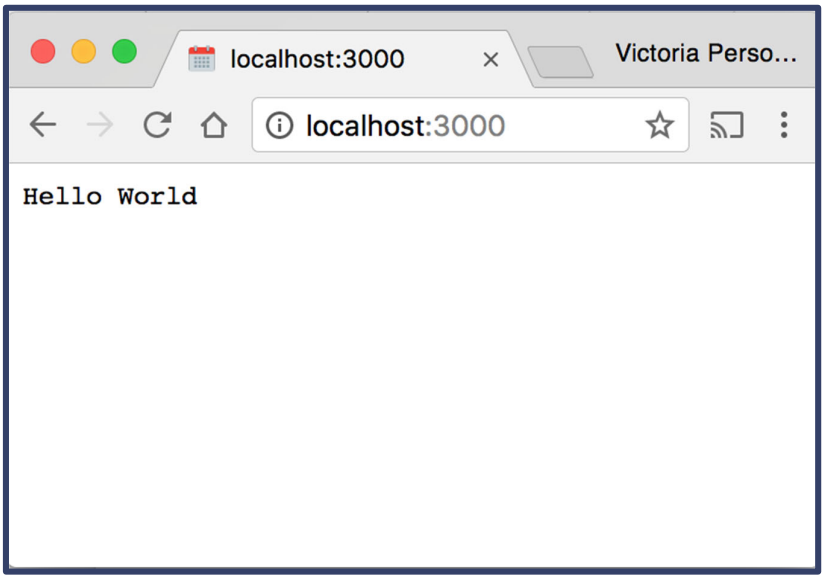
npm

Το npm εγκαθίσταται μαζί με το node:

- **npm**: Node Package Manager*:
  εργαλείο που εγκαθιστά **πακέτα** (βιβλιοθήκες και εργαλεία)
  γραμμένα σε JavaScript και συμβατά με το NodeJS
- Πακέτα μπορούν να βρεθούν στο online repository:
  https://www.npmjs.com/

Παράδειγμα Express

```
$ npm install express
$ node server.js
Example app listening on port 3000!
```



ExpressJS

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

Η `require()` φορτώνει το ExpressJS module.

Το module αυτό περιέχει μια συνάρτηση που δημιουργεί ένα νέο Express Application object.

ExpressJS

```
app.listen(3000, function () {
  console.log('Example app listening on port 3000!');
})
```

Η ExpressJS `listen()` είναι ταυτόσημη με την NodeJS `listen()` συνάρτηση:

- Προσαρτά τη διαδικασία του server στο συγκεκριμένο **αριθμό port**.
- Έτσι μηνύματα που στέλνονται στο port3000 του ΛΣ θα δρομολογούνται σε αυτή τη διαδικασία server.
- Η παράμετρος-συνάρτηση είναι ένα callback που θα εκτελεστεί όταν ο server ξεκινήσει να ακούει για HTTP μηνύματα (όταν προσαρτηθεί στο port 3000)

ExpressJS Routes

```
app.get('/', function (req, res) {
  res.send('Hello World!');
})
```

app.***method***(***path***, ***handler***)

- Καθορίζει τον χειρισμό από τον server αιτημάτων HTTP ***method*** (`get` | `post` | ...) που γίνονται στο URL/***path***
- Η callback συνάρτηση θα εκτελεστεί κάθε φορά που υπάρχει ένα νέο αίτημα.
- Παράδειγμα: Όταν υπάρξει GET request στο http://localhost:3000/, απάντησε με το κείμενο "Hello World!"

ExpressJS Routes

Μπορούμε να έχουμε και άλλα routes στο Express:

```
app.get('/', function (req, res) {
  res.send('Main page!');
});

app.get('/hello', function (req, res) {
  res.send('GET hello!');
});

app.post('/hello', function (req, res) {
  res.send('POST hello!');
});
```


Επικοινωνία με τον server

Αποστολή HTTP requests

Πώς μπορούμε να στείλουμε HTTP requests στον server;

1. Πλοήγηση στο http://localhost:3000/<path> από τον browser
   - Όμως, μόνο GET requests

2. Χρήση fetch()

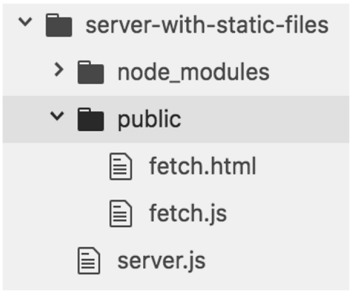
- Μπορούμε να στείλουμε οποιοδήποτε τύπο HTTP request
- Όμως: πρόβλημα λόγω CORS (http://localhost:3000 vs. file:///)

3. `curl` command-line tool
   - $ `curl` --request POST http://localhost:3000/hello

Cross-origin λύσεις

Λύση 1: Θέτουμε κεφαλίδα Access-Control-Allow-Origin πριν την αποστολή του response:

```
app.get('/', function (req, res) {
  res.header("Access-Control-Allow-Origin", "*");
  res.send('Main page!');
});
```



Λύση 2: Φόρτωση του κώδικα fetch *στατικά* **από τον ίδιο server:**

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

app.use(express.static('public'));

app.get('/', function (req, res) {
  res.send('Main page!');
```

Λέει στον server να εξυπηρετεί απευθείας τα αρχεία στο directory public

Οπότε ο server εξυπηρετεί:
http://localhost:3000/fetch.html
http://localhost:3000/fetch.js
relative to the static directory

Αλλαγή μεθόδου στη `fetch()`

```
app.post('/hello', function (req, res) {
  res.send('POST hello!');
});
```

server

```
fetch('/hello', { method: 'POST' })
    .then(onResponse)
    .then(onTextReady);
```

client

Μπορούμε να αλλάξουμε την μέθοδο HTTP μέσω παραμέτρου στη fetch (), που καθορίζει ένα options object:

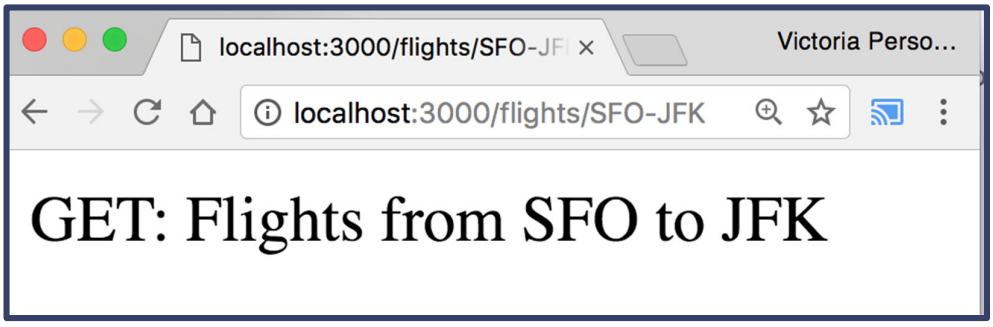
- `method:` καθορίζει την HTTP μέθοδο του αιτήματος, π.χ. POST, PUT, PATCH, DELETE κλπ.
  - GET είναι η προκαθορισμένη τιμή (default).

Route parameters

Μπορούμε να ορίσουμε μία ή περισσότερες *παραμέτρους δρομολόγησης* μέσα στο URL και να τις διαβάσουμε από το **`req.params`** (docs):

- Ξεκινούν με :
- Τα . και – ερμηνεύονται ως έχουν

```
app.get('/flights/:from-:to', function (req, res) {
  const routeParams = req.params;
  const from = routeParams.from;
  const to = routeParams.to;
  res.send('GET: Flights from ' + from + ' to ' + to);
});
```



Query parameters

Μπορούμε να διαβάσουμε τις *παραμέτρους ερωτήματος* (?) από το **`req.query`**:

```
app.get('/hello', function (req, res) {
  const queryParams = req.query;
  const name = queryParams.name;
  res.send('GET: Hello, ' + name);
});
```

Request: `http://localhost:3000/hello?name=Dimitris`
Response: `GET: Hello, Dimitris`

POST message body

Μπορούμε να στείλουμε query parameters και μέσω POST
Είναι όμως κακή πρακτική. Συνήθως στέλνουμε τα δεδομένα στο message body

- για αυτό χρησιμοποιούμε POST!

```
const message = {
    name: 'Dimitris',
    email: 'koutsomi@ceid'};
const serializedMessage = JSON.stringify(message);
fetch('/helloemail', { method: 'POST', body: serializedMessage })
    .then(onResponse)
    .then(onTextReady);
```

Επεξεργασία POST μηνύματος με Express

```
app.post('/helloemail', function (req, res) {
  let data = '';
  req.setEncoding('utf8');
  req.on('data', function(chunk) {
     data += chunk;
  });

  req.on('end', function() {
    const body = JSON.parse(data);
    const name = body.name;
    const email = body.email;
    res.send('POST: Name: ' + name + ', email: ' + email);
  });
});
```

Χρειάζεται χειρισμός των events του **request object**

- `data, end`
- Κληρονομούνται από το nodeJS

body-parser

Μπορούμε να χρησιμοποιήσουμε τη body-parser βιβλιοθήκη:

```
const bodyParser = require('body-parser');
const jsonParser = bodyParser.json();
```

Δεν ανήκει στο nodeJS API:

- `$ npm install body-parser`

Δημιουργεί έναν JSON parser

- Περνιέται ως παράμετρος στα routes που θα δεχτούν message body

Προσπέλαση απευθείας με το `req.body`:

```
app.post('/helloparsed', jsonParser, function (req, res) {
  const body = req.body;
  const name = body.name;
  const email = body.email;
  res.send('POST: Name: ' + name + ', email: ' + email);
});
```

Σώμα μηνύματος POST

Τέλος χρειάζεται να προσθέσουμε JSON content-type κεφαλίδες στο μήνυμα που θα γίνει POST από την πλευρά της fetch():

```
const message = {
    name: 'Dimitris',
    email: 'koutsomi@ceid'};
const fetchOptions = {
  method: 'POST',
  headers: {
    'Accept': 'application/json',
    'Content-Type': 'application/json'
  },
  body: JSON.stringify(message)
};
fetch('/helloparsed', fetchOptions)
  .then(onResponse)
  .then(onTextReady);
```

Response:

```
POST: Name: Dimitris, email: koutsomi@ceid
```


Σχεδιαστικές συμβάσεις

GET vs POST

- Χρήση GET για requests ανάκτησης δεδομένων, όχι εγγραφής
- Χρήση POST για requests εγγραφής δεδομένων, όχι ανάκτησης

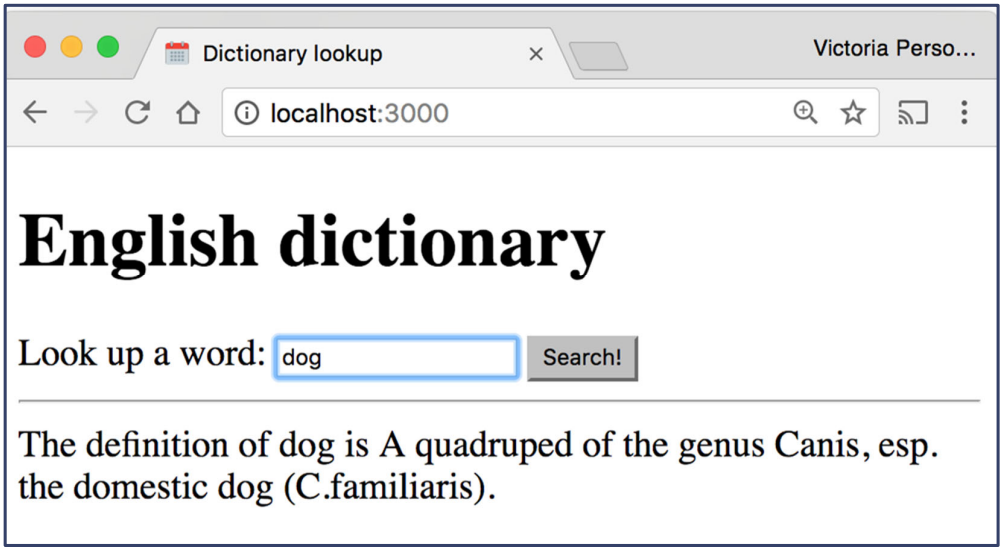
Route vs Query params

- Χρήση route παραμέτρων για τις παραμέτρους που είναι απαραίτητες για το request
- Χρήση query παραμέτρων για:
  - Όσες είναι προαιρετικές
  - Οι τιμές τους μπορεί να έχουν κενά

Παράδειγμα: Λεξικό

Παράδειγμα

Έστω αρχείο `dictionary.json` που περιέχει ζευγάρια λέξεων/τιμών.
Η εφαρμογή λεξικού επιτρέπει την αναζήτηση μιας λέξης και εμφάνιση του ορισμού της.



Αναζήτηση στο λεξικό

```
// Load a JSON file containing english words.
const englishDictionary = require('./dictionary.json');

app.use(express.static('public'));

function onPrintWord(req, res) {
  const routeParams = req.params;
  const word = routeParams.word;

  const key = word.toLowerCase();
  const definition = englishDictionary[key];

  res.send(`The definition of ${word} is ${definition}`);
}
app.get('/print/:word', onPrintWord);
```


Fetch από το λεξικό

```
async function onSearch(event) {
  event.preventDefault();
  const input = document.querySelector('#word-input');
  const word = input.value.trim();
  const result = await fetch('/print/' + word);
  const text = await result.text();
  const re
  results.
}

const form = document.querySelector('#search');
form.addEventListener('submit', onSearch);
```

```
<form id="search">
  Look up a word: <input type="text" id="word-input"/>
  <input type="submit" value="Search!">
</form>
```

Απόκριση JSON

Αν θέλουμε να επιστρέψουμε JSON, μπορούμε να χρησιμοποιήσουμε τη μέθοδο `res.json(`***object***`)` αντί για `res.send(`*string*`)`:

```
app.get('/', function (req, res) {
  const response = {
    greeting: 'Hello World!',
    awesome: true
  }
  res.json(response);
});
```

Η παράμετρος που περνιέται στη `res.json()` θα πρέπει να είναι ένα αντικείμενο JavaScript.

Απόκριση JSON από το λεξικό

```
function onLookupWord(req, res) {
  const routeParams = req.params;
  const word = routeParams.word;

  const key = word.toLowerCase();
  const definition = englishDictionary[key];

  res.json({
    word: word,
    definition: definition
  });
}
app.get('/lookup/:word', onLookupWord);
```

Fetch από το λεξικό (JSON)

```
async function onSearch(event) {
  event.preventDefault();
  const input = document.querySelector('#word-input');
  const word = input.value.trim();

  const results = document.querySelector('#results');
  results.cla
  const resul
  const json =
  results.classList.remove('hidden');
  const wordDisplay = results.querySelector('#word');
  const defDisplay = results.querySelector('#definition');
  wordDisplay.textContent = json.word;
  defDisplay.textContent = json.definition;
}
```

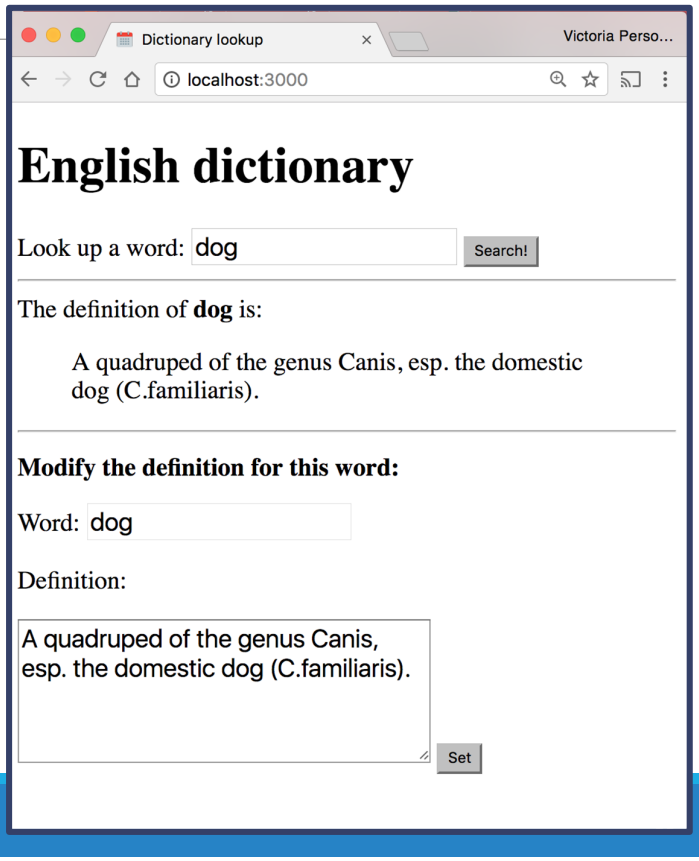
```
<div id="results" class="hidden">
  The definition of <strong id="word"></strong> is:
  <blockquote id="definition"></blockquote>
</div>
```


Αποτέλεσμα

Αποθήκευση δεδομένων

Πώς μπορούμε να στείλουμε δεδομένα πίσω στο λεξικό;

- Π.χ. να τροποποιήσουμε ή να προσθέσουμε τον ορισμό μιας λέξης
- Θα στείλουμε τα περιεχόμενα της φόρμας με POST



fs-extra

Χρήση της βιβλιοθήκης `fs-extra` για να γράψουμε στο αρχείο dictionary.json.

- `fs`: NodeJS API βιβλιοθήκη
  - Χρησιμοποιεί callbacks
- `fs-extra`: npm βιβλιοθήκη
  - Χρησιμοποιεί callbacks Ή promises
  - `fs.writeJson(`***`fileName, object`***`)`

Server: εγγραφή δεδομένων

```
async function onSetWord(req, res) {
  const routeParams = req.params;
  const word = routeParams.word;
  const definition = req.body.definition;
  const key = word.toLowerCase();
  englishDictionary[key] = definition;
  await fse.writeJson('./dictionary.json', englishDictionary);
  res.json({ success: true});
}
app.post('/set/:word', jsonParser, onSetWord);
```


Client: `fetch()`

```
async function onSet(event) {
  event.preventDefault();
  const setWordInput = results.querySelector('#set-word-input');
  const setDefInput = results.querySelector('#set-def-input');
  const word = setWordInput.value;
  const def = setDefInput.value;

  const message = {
    definition: def
  };
  const fetchOptions = {
    method: 'POST',
    headers: {
      'Accept': 'application/json',
      'Content-Type': 'application/json'
    },
    body: JSON.stringify(message)
  };
  await fetch('/set/' + word, fetchOptions);
```

```
<form id="set">
  <h2>
    Modify the definition for this word:
  </h2>
  Word: <input id="set-word-input" type="text" readonly/>
  <p>
    Definition:
  </p>
  <textarea id="set-def-input"></textarea>
  <input type="submit" value="Set">
  <p id="status"></p>
</form>
```

Πηγές:V. Kirst, Stanford CS-193X course; Mozilla Development Network, https://developer.mozilla.org; Connolly, Hoar, "Fundamentals of Web Development, 2nd ed