User Params

Reading from the URL

Level 3 - Part I

Always returning all the Blocks

To improve efficiency, we want to be able to **limit** the number of results returned

app.js

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

app.get('/blocks', function(request, response) {
  var blocks = ['Fixed', 'Movable', 'Rotating'];
  response.json(blocks);
});

app.listen(3000);
```

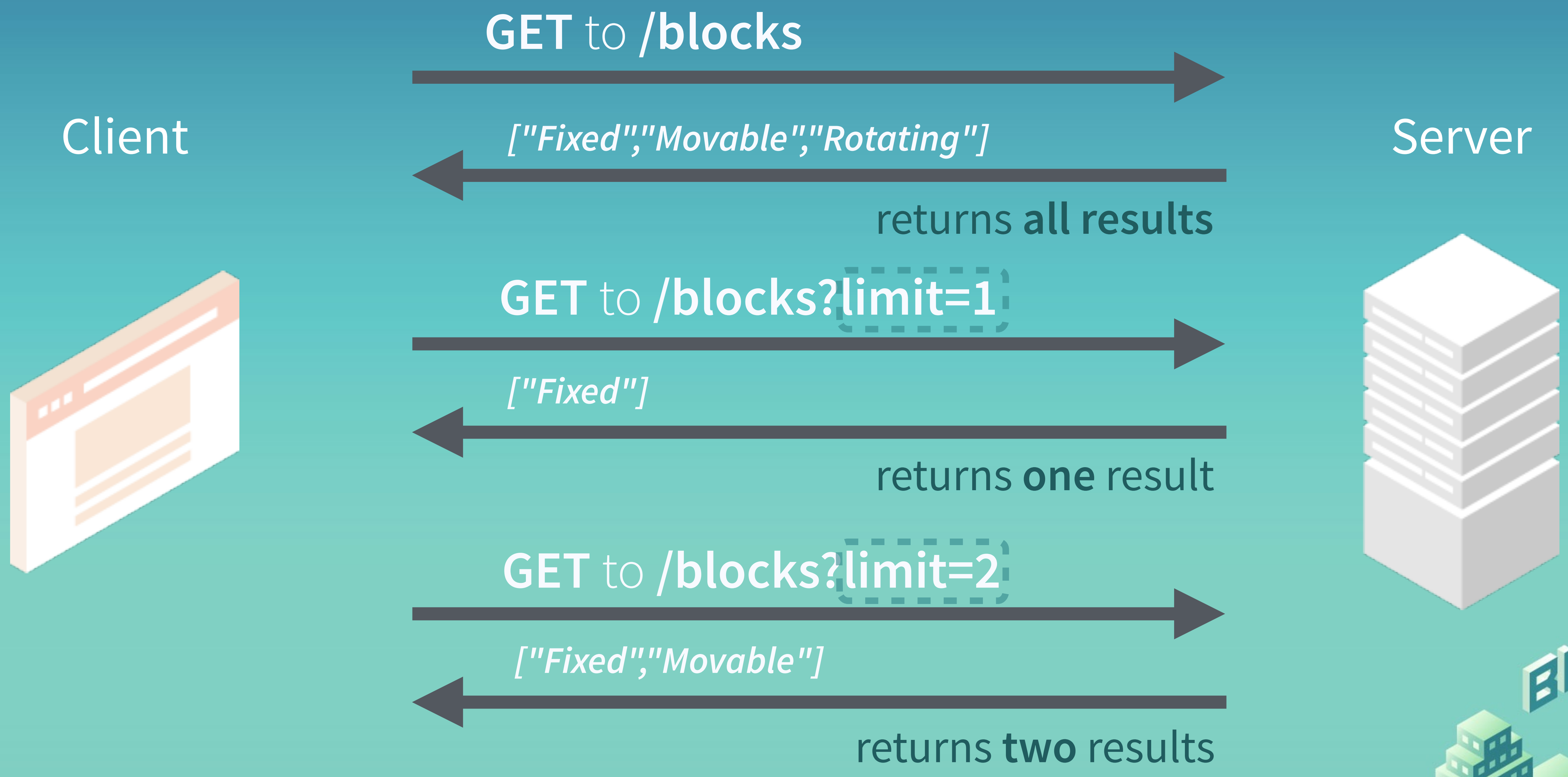
```
$ curl http://localhost:3000/blocks

["Fixed","Movable","Rotating"]
```

always returns **all the Blocks**

Limiting the number of Blocks returned

Query strings are a great way to **limit** the number of results returned from an endpoint



Reading query string parameters

Use **request.query** to access query strings

app.js

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

app.get('/blocks', function(request, response) {
  var blocks = ['Fixed', 'Movable', 'Rotating'];
  if (request.query.limit >= 0) {

  } else {
    response.json(blocks);
  }
});

app.listen(3000);
```

true when a numeric value for **limit** is part of the URL

returns **all results**

Reading query string parameters

The **slice** function returns a portion of an Array

app.js

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

app.get('/blocks', function(request, response) {
  var blocks = ['Fixed', 'Movable', 'Rotating'];
  if (request.query.limit >= 0) {
    response.json(blocks.slice(0, request.query.limit));
  } else {
    response.json(blocks);
  }
});

app.listen(3000);
```

returns **limited** results

Limiting results using curl

URLs with query strings can be used with **curl**

```
$ curl http://localhost:3000/blocks?limit=1

["Fixed"]
```

```
$ curl http://localhost:3000/blocks?limit=2

["Fixed","Movable"]
```

limiting results

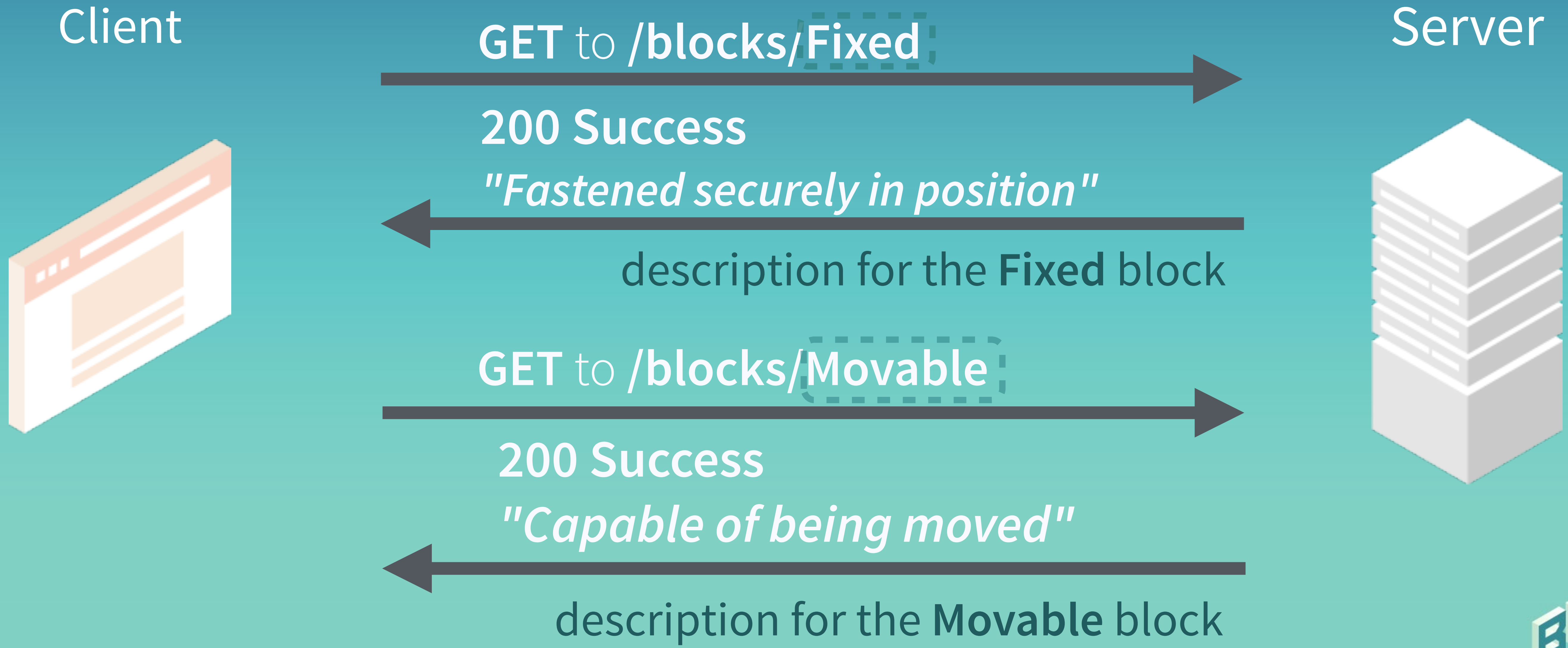
```
$ curl http://localhost:3000/blocks

["Fixed","Movable","Rotating"]
```

all results when no limit is used

Returning description for a specific Block

We can use **meaningful URLs** to return the description for specific types of Blocks



GET to **/blocks/Fixed**

200 Success

"Fastened securely in position"

description for the **Fixed** block

GET to **/blocks/Movable**

200 Success

"Capable of being moved"

description for the **Movable** block

Creating Dynamic Routes

Placeholders can be used to name arguments part of the URL path

app.js

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

app.get('/blocks/:name', function(request, response) {
  ...
});

app.listen(3000);
```

creates **name** property on the **request.params** object

```
request.params.name
```

Expanding our blocks list

In order to store additional information on blocks, we'll move from an Array to a JavaScript **object**

app.js

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

var blocks = {
  'Fixed': 'Fastened securely in position',
  'Movable': 'Capable of being moved',
  'Rotating': 'Moving in a circle around its center'
};
app.get('/blocks/:name', function(request, response) {
  ...
});

app.listen(3000);
```

can now be accessed from other routes in the file

Reading route parameters

We use **request.params.name** to look up the Block’s **description**

app.js

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

var blocks = {
  'Fixed': 'Fastened securely in position',
  'Movable': 'Capable of being moved',
  'Rotating': 'Moving in a circle around its center'
};
app.get('/blocks/:name', function(request, response) {
  var description = blocks[request.params.name];
  ...
});

app.listen(3000);
```

Returning block description

Responding with description and proper status code

app.js

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

var blocks = {
  'Fixed': 'Fastened securely in position',
  'Movable': 'Capable of being moved',
  'Rotating': 'Moving in a circle around its center'
};
app.get('/blocks/:name', function(request, response) {
  var description = blocks[request.params.name];
  response.json(description);
});

app.listen(3000);
```

defaults to **200 Success** status code

Testing dynamic routes with curl

Returns proper **status codes** and response bodies

```
$ curl -i http://localhost:3000/blocks/Fixed

HTTP/1.1 200 OK
"Fastened securely in position"
```

```
$ curl -i http://localhost:3000/blocks/Movable

HTTP/1.1 200 OK
"Capable of being moved"
```

The **-i** option tells curl to **include** response headers in the output

Fixing the response for URLs not found

The status code does not indicate an **invalid** URL

```
$ curl -i http://localhost:3000/blocks/Banana

HTTP/1.1 200 OK

```

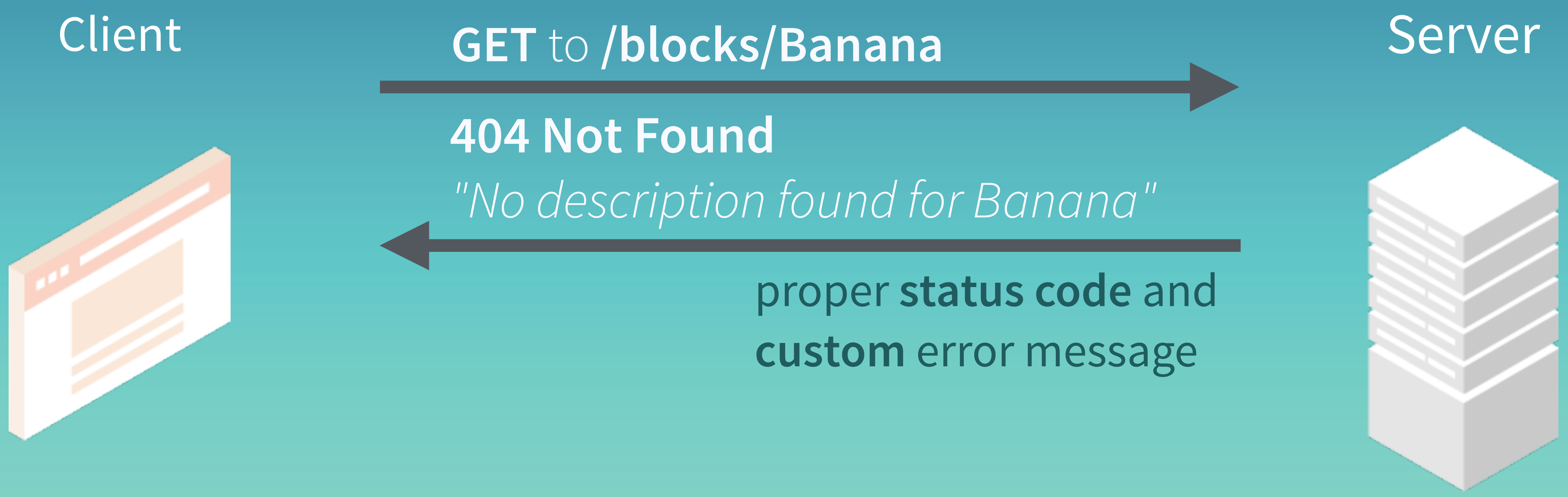
this Block does **not** exist

successful status code
for **unsuccessful** request

blank response

Handling Blocks not found

We must return a **404 Not Found** status code and an informative error message when a Block is not found


Client
GET to /blocks/Banana
Server
404 Not Found
"No description found for Banana"
proper status code and custom error message

Responding from not found URLs

Trying to access non-existing properties in JavaScript objects returns **undefined**

app.js

```
...
var blocks = {
  'Fixed': 'Fastened securely in position',
  'Movable': 'Capable of being moved',
  'Rotating': 'Moving in a circle around its center'
};
app.get('/blocks/:name', function(request, response) {
  var description = blocks[request.params.name];
  if (!description) {
    ...
  } else {
    response.json(description);
  }
});
...
```

returns **undefined** when no property is found for a given Block name

checks for the presence of a description to determine the response

Responding with Not Found

Use the **status** function to set a custom HTTP status code

app.js

```
...
var blocks = {
  'Fixed': 'Fastened securely in position',
  'Movable': 'Capable of being moved',
  'Rotating': 'Moving in a circle around its center'
};
app.get('/blocks/:name', function(request, response) {
  var description = blocks[request.params.name];
  if (!description) {
    response.status(404)
  } else {
    response.json(description);
  }
});
...
```

sets the **404 Not Found** status code

Responding with Not Found

Respond with a custom JSON error message

app.js

```
...
var blocks = {
  'Fixed': 'Fastened securely in position',
  'Movable': 'Capable of being moved',
  'Rotating': 'Moving in a circle around its center'
};
app.get('/blocks/:name', function(request, response) {
  var description = blocks[request.params.name];
  if (!description) {
    response.status(404).json('No description found for ' + request.params.name);
  } else {
    response.json(description);
  }
});
...
```

informative error message

Testing invalid routes with curl

Returns proper **status code** and informative error message

```
$ curl -i http://localhost:3000/blocks/Banana

HTTP/1.1 404 Not Found
"No description found for Banana"
```