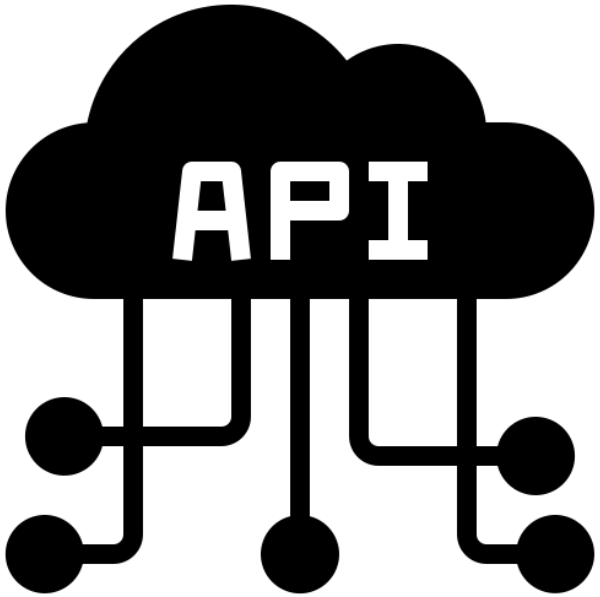


Creating an API

SENG 4640
Software Engineering for Web Apps
Winter 2023

Sina Keshvadi
Thompson Rivers University

Here we are in the last part of the programming section of the course.

Let’s wrap things up by an example that combines a lot of the different things we've seen so far.

In this example, we're going to have some client side JavaScript, that is JavaScript running in the browser using a library like jQuery.

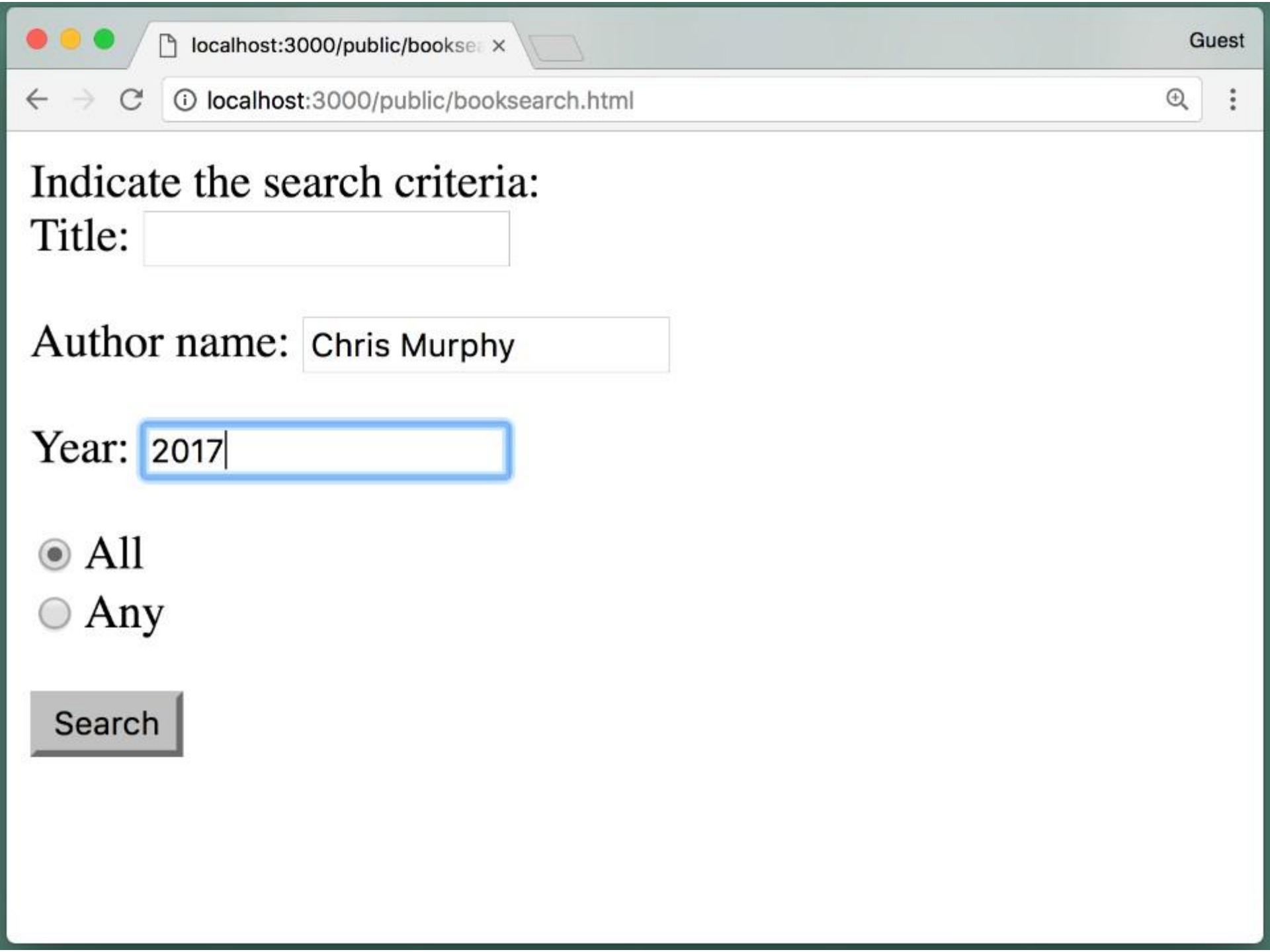
And then we'll have some server side JavaScript using Node and Express.

And we'll see how they communicate with each other to create some interactive web app.

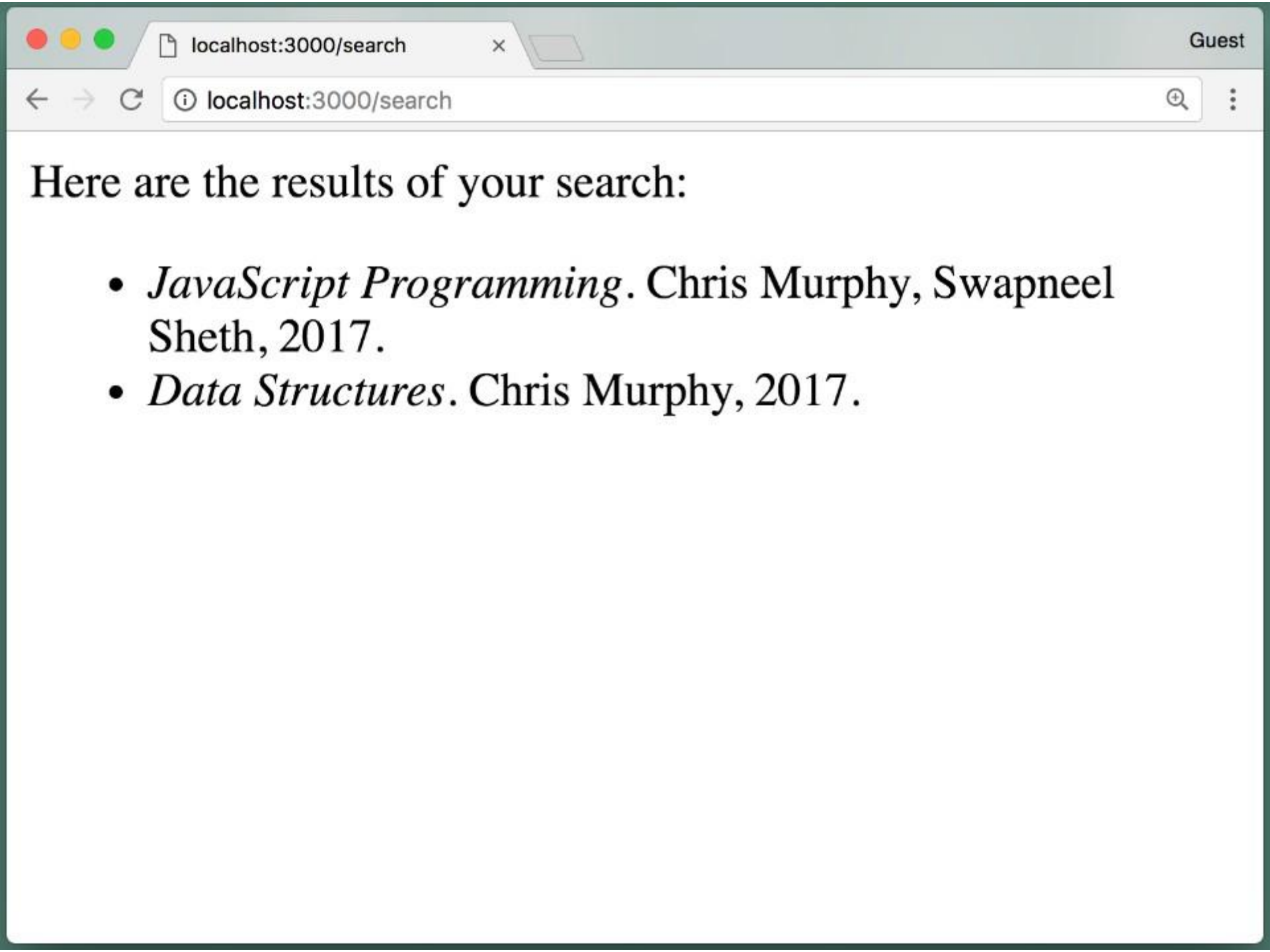
Review

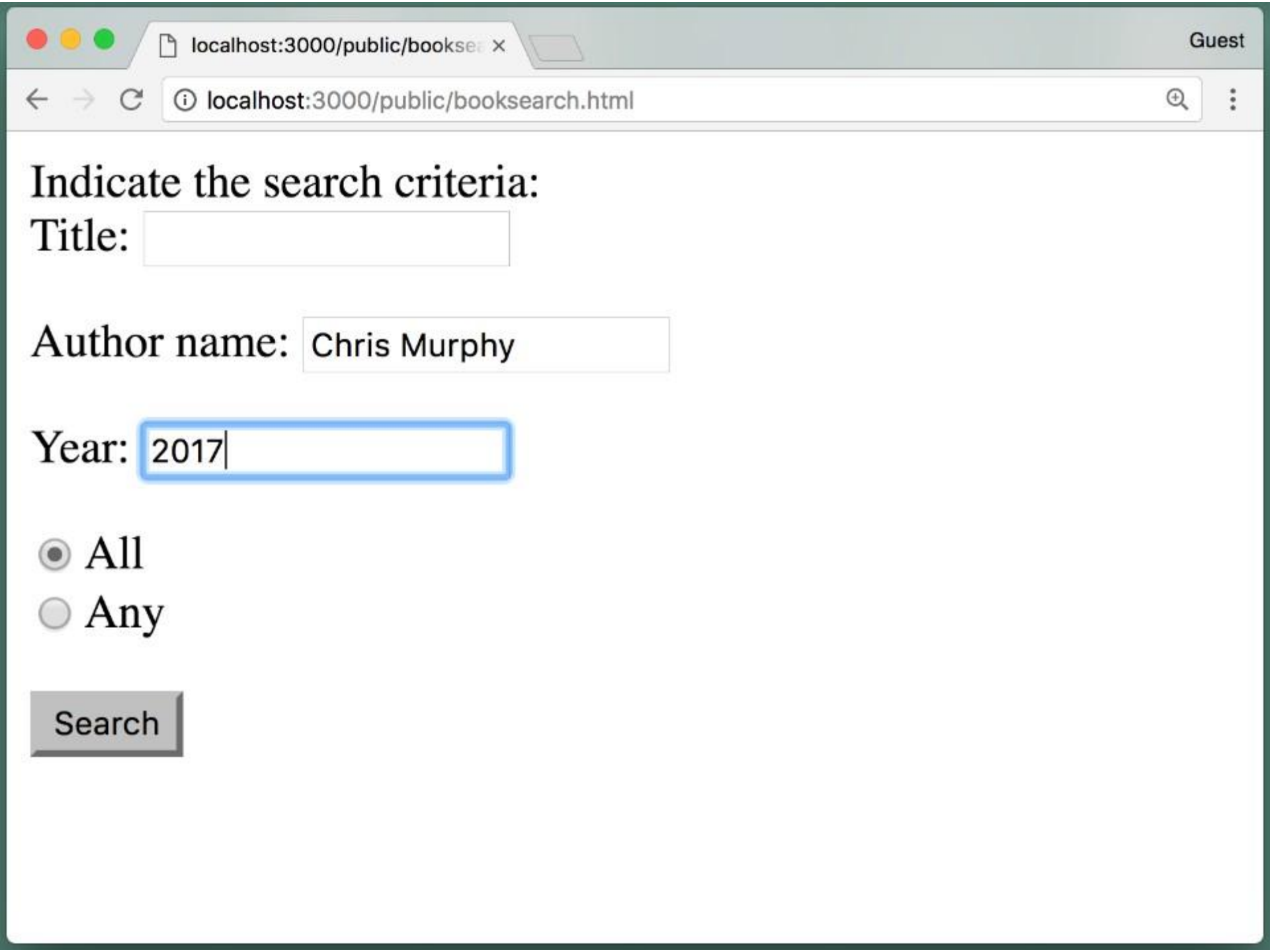
- Node.js and Express allow us to build server-side web apps in JavaScript

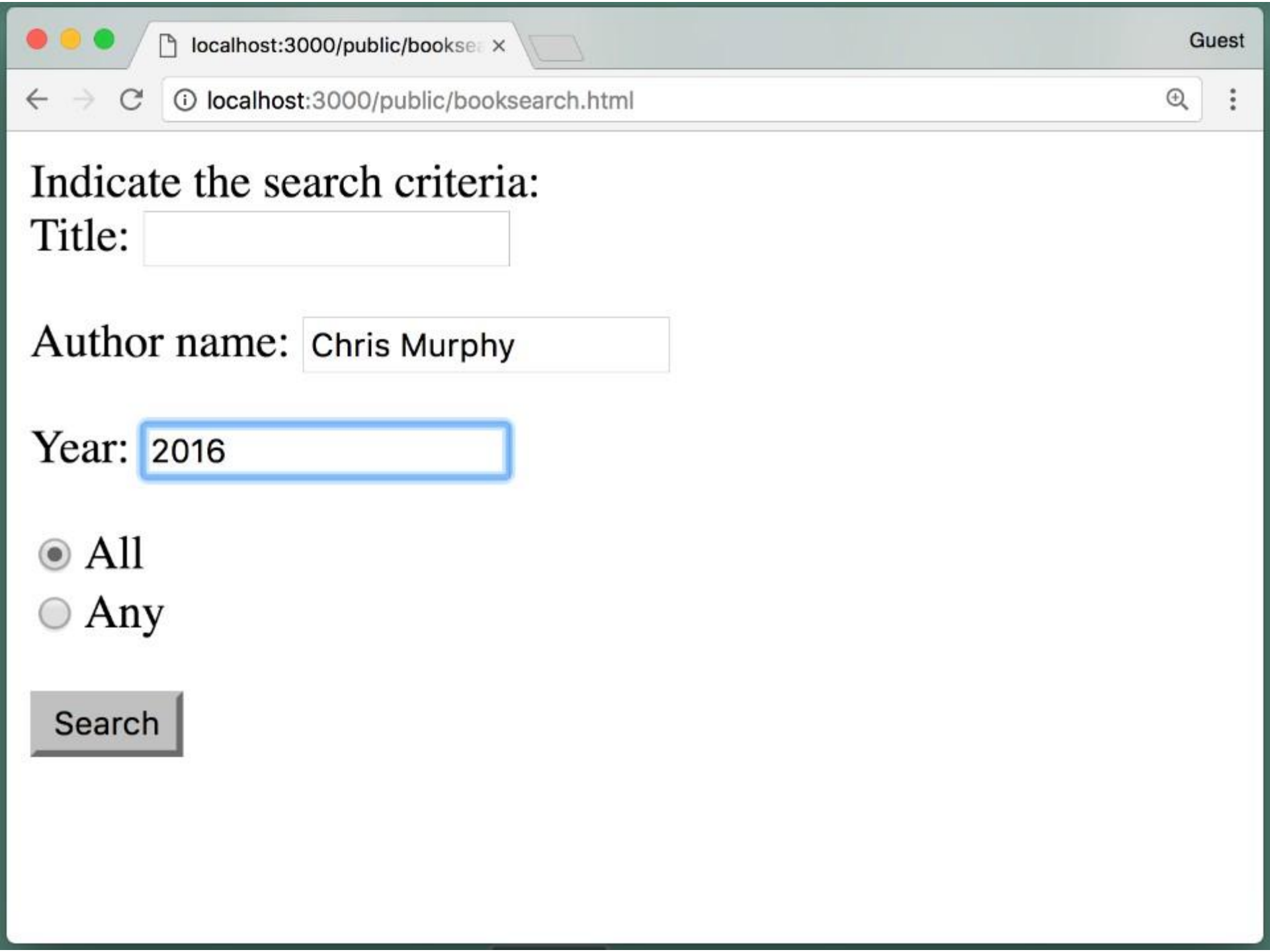
- MongoDB allows us to store data as documents and query them via JavaScript

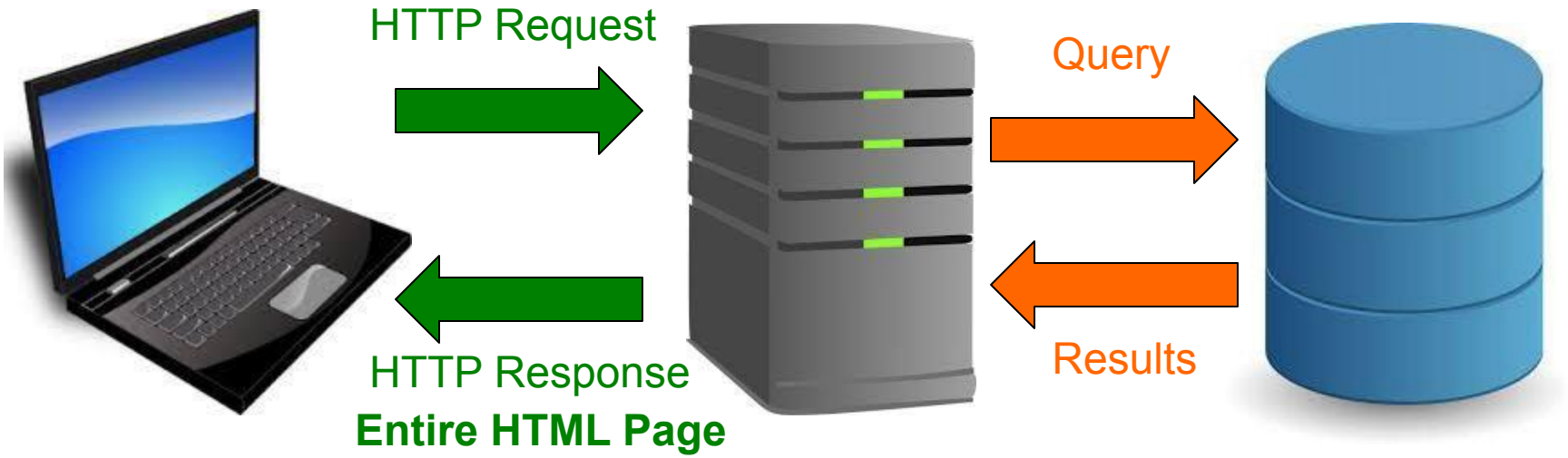

localhost:3000/public/booksea
Guest
localhost:3000/public/booksearch.html
Indicate the search criteria:
Title:
Author name:
Chris Murphy
Year:
2017
All
Any
Search

```
// index.js
async function searchAll(req, res) {
 const query = {};
 if (req.body.title) {
   query.title = req.body.title;
 }
 if (req.body.name) {
   query["authors.name"] = req.body.name;
 }
 if (req.body.year) {
   query.year = req.body.year;
 }

 try {
   const books = await Book.find(query);
   res.render("books", { books });
 } catch (err) {
   res.type("html").status(500);
   res.send("Error: " + err);
 }
}
```

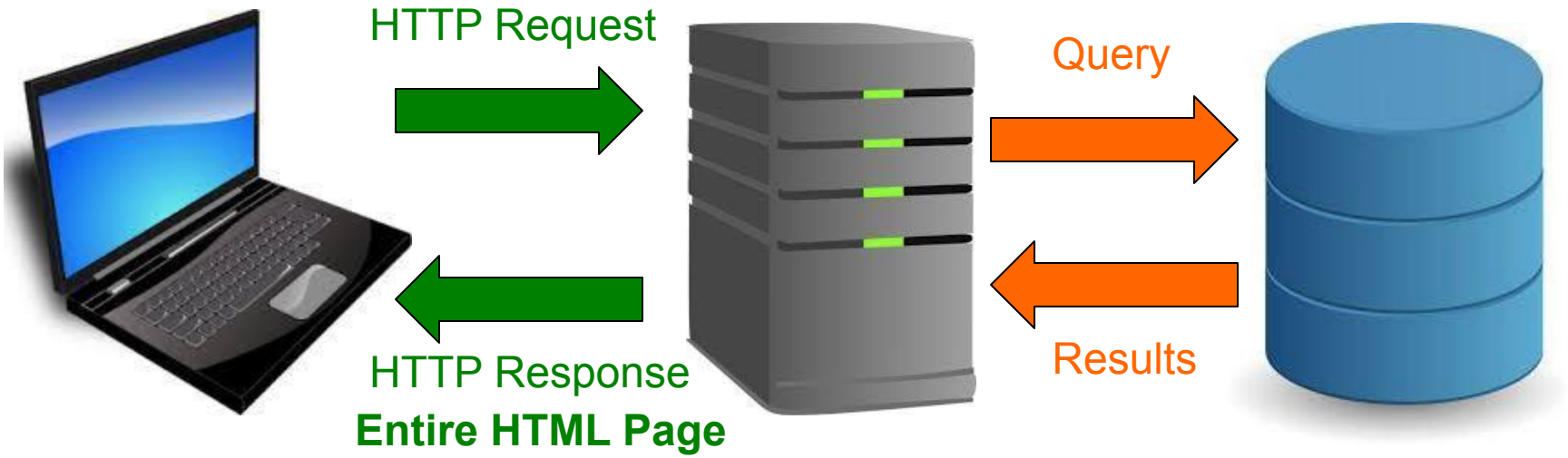

localhost:3000/search
Guest
localhost:3000/search
Here are the results of your search:
• *JavaScript Programming*. Chris Murphy, Swapneel Sheth, 2017.
• *Data Structures*. Chris Murphy, 2017.


localhost:3000/public/booksea
Guest
localhost:3000/public/booksearch.html
Indicate the search criteria:
Title:
Author name:
Chris Murphy
Year:
2017
All
Any
Search


localhost:3000/public/booksea
Guest
localhost:3000/public/booksearch.html
Indicate the search criteria:
Title:
Author name:
Chris Murphy
Year:
2016
All
Any
Search



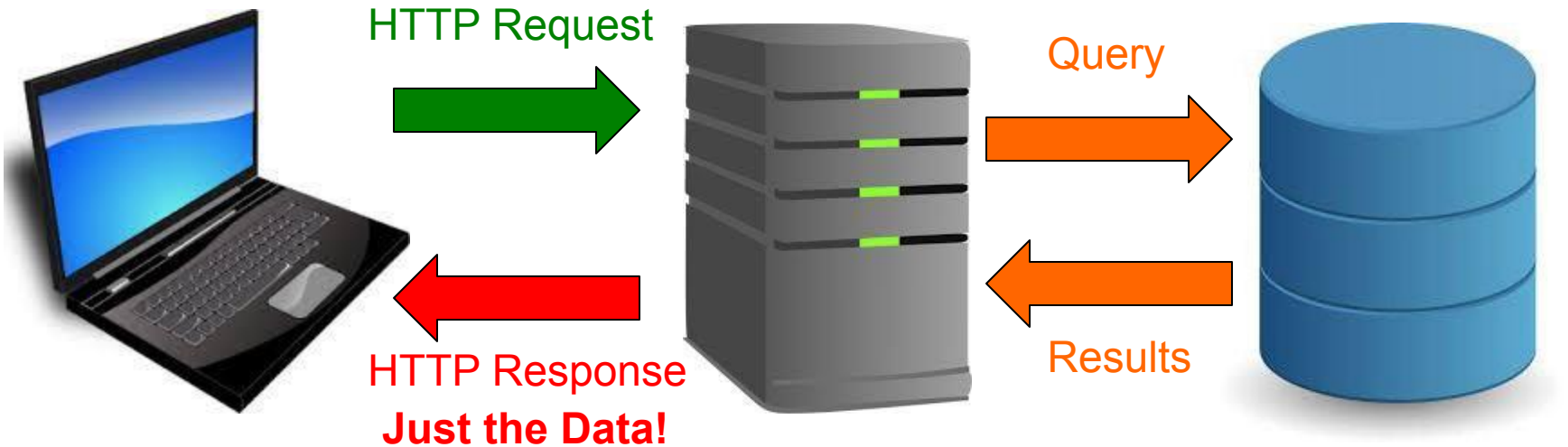
- The client first sends the HTTP Request to the Server as a result of submitting the form.
- The server is running our Node Express JavaScript code. It sends the query to MongoDB.
- MongoDB sends back the results.
- The server then takes those results, renders the HTML and sends it back to the client as part of the HTTP response, that is, it's sending back the entire HTML page.



- Now the user does another search, it sends another HTTP request, that causes another query to be sent to Mongo, Mongo gives back the results, server constructs the HTML again. And sends back the HTTP response, including the entire HTML page.

In this particular example, it's probably not a big deal that it's re-rendering the entire HTML page.

After all, that HTML page was pretty small and pretty simple.


HTTP Request
Query
HTTP Response
Just the Data!
Results
Client
Server
Database

What is an API?

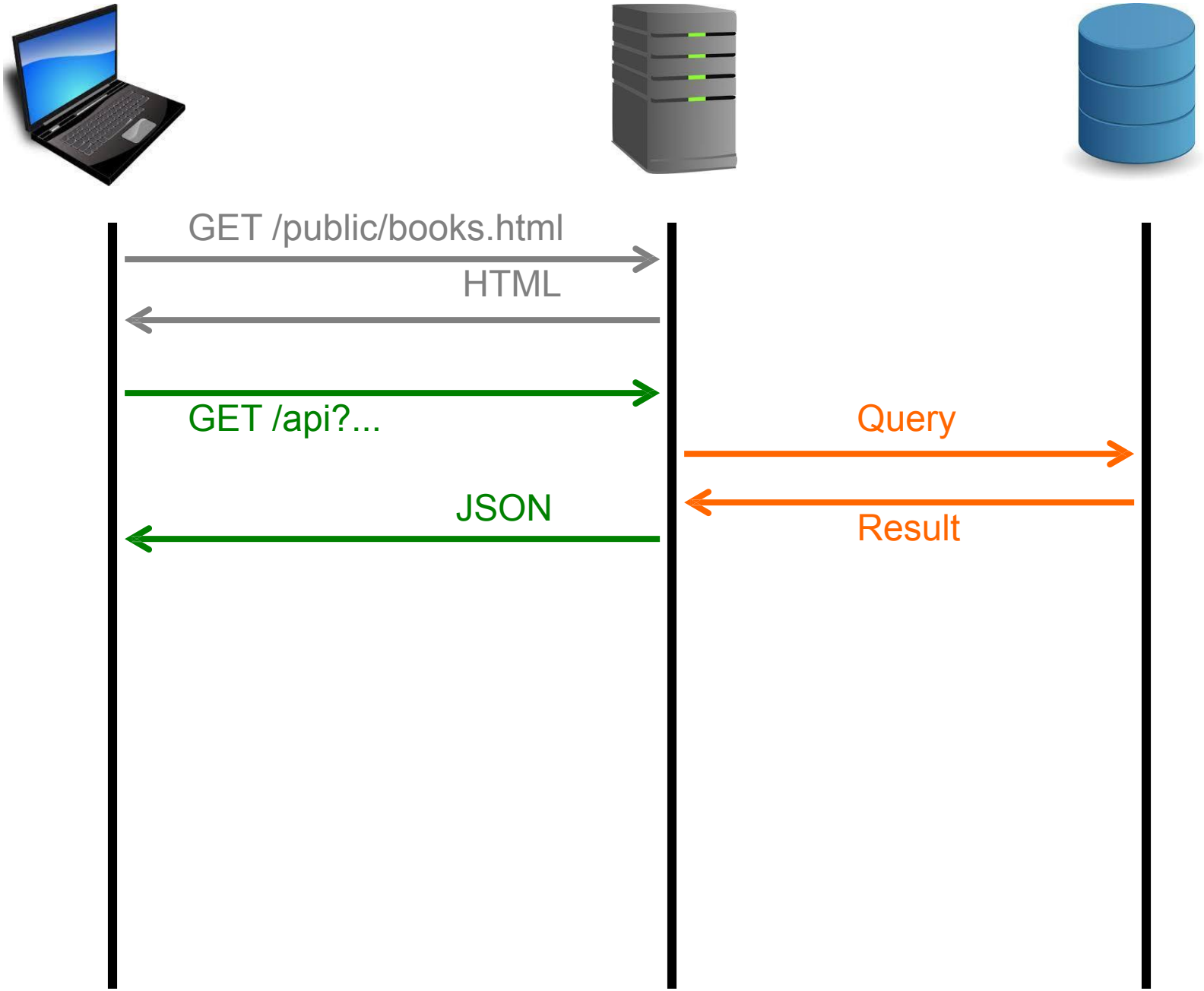
- An API is an **Application Programming Interface**
- It is a URL or a set of URLs that returns pure data to requests
- APIs can be used to incorporate data and functionality from other sources in your webapp

What is an API?

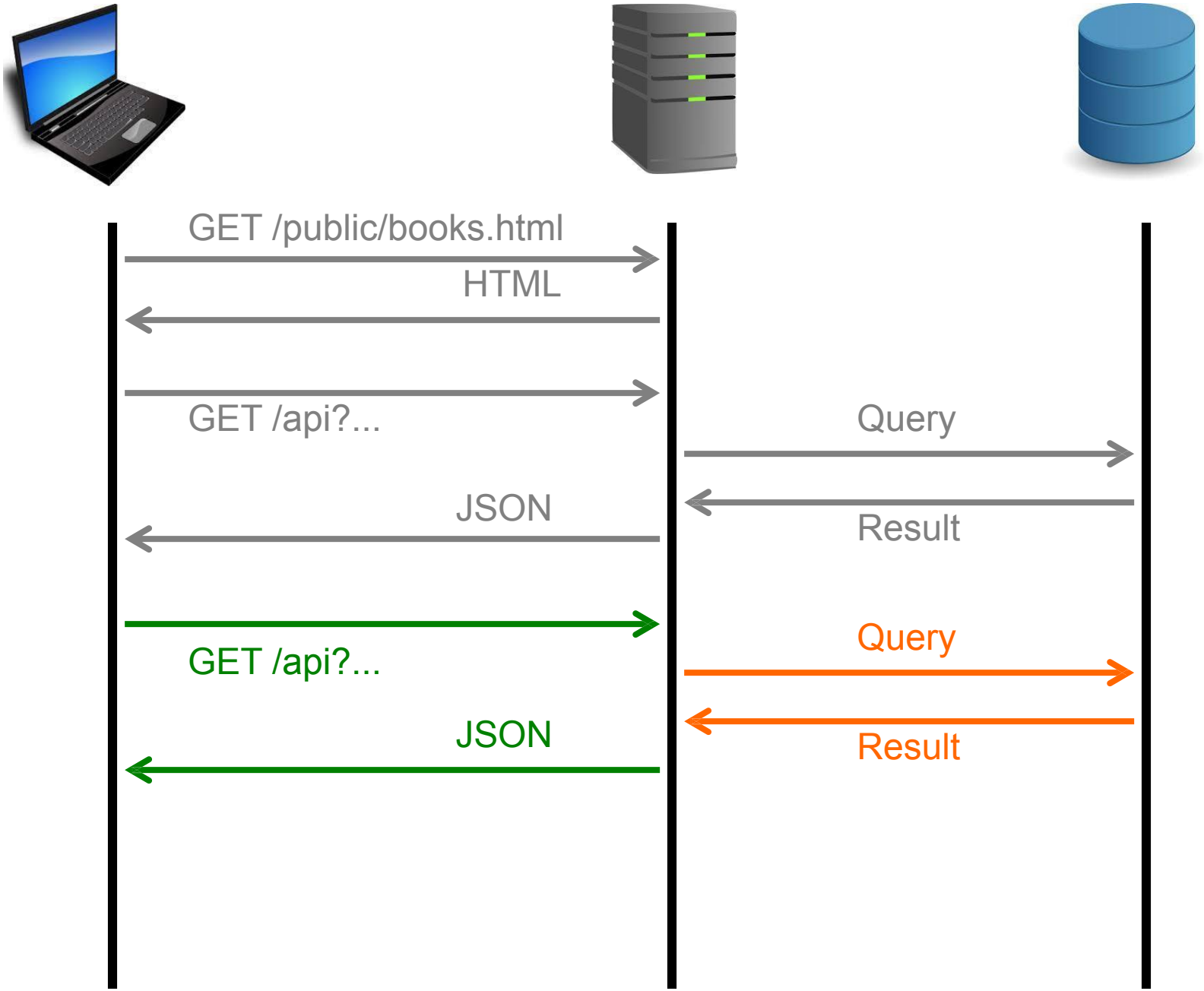
- An API is an **Application Programming Interface**
- It is a URL or a set of URLs that returns pure data to requests
- APIs can be used to incorporate data and functionality from other sources in your webapp
- **You can also create your own API using Node.js to return JSON data to HTTP requests**

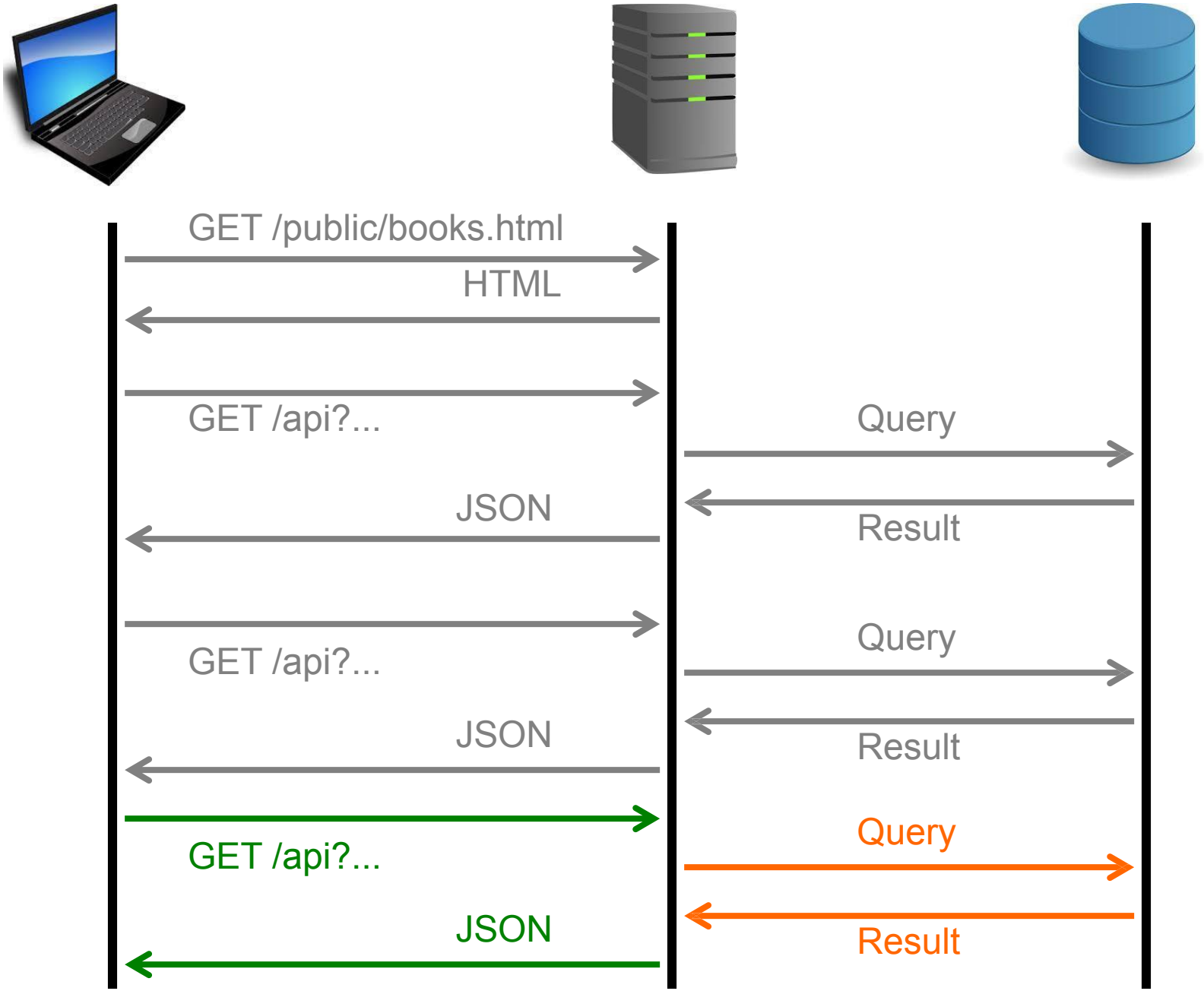
GET /public/books.html

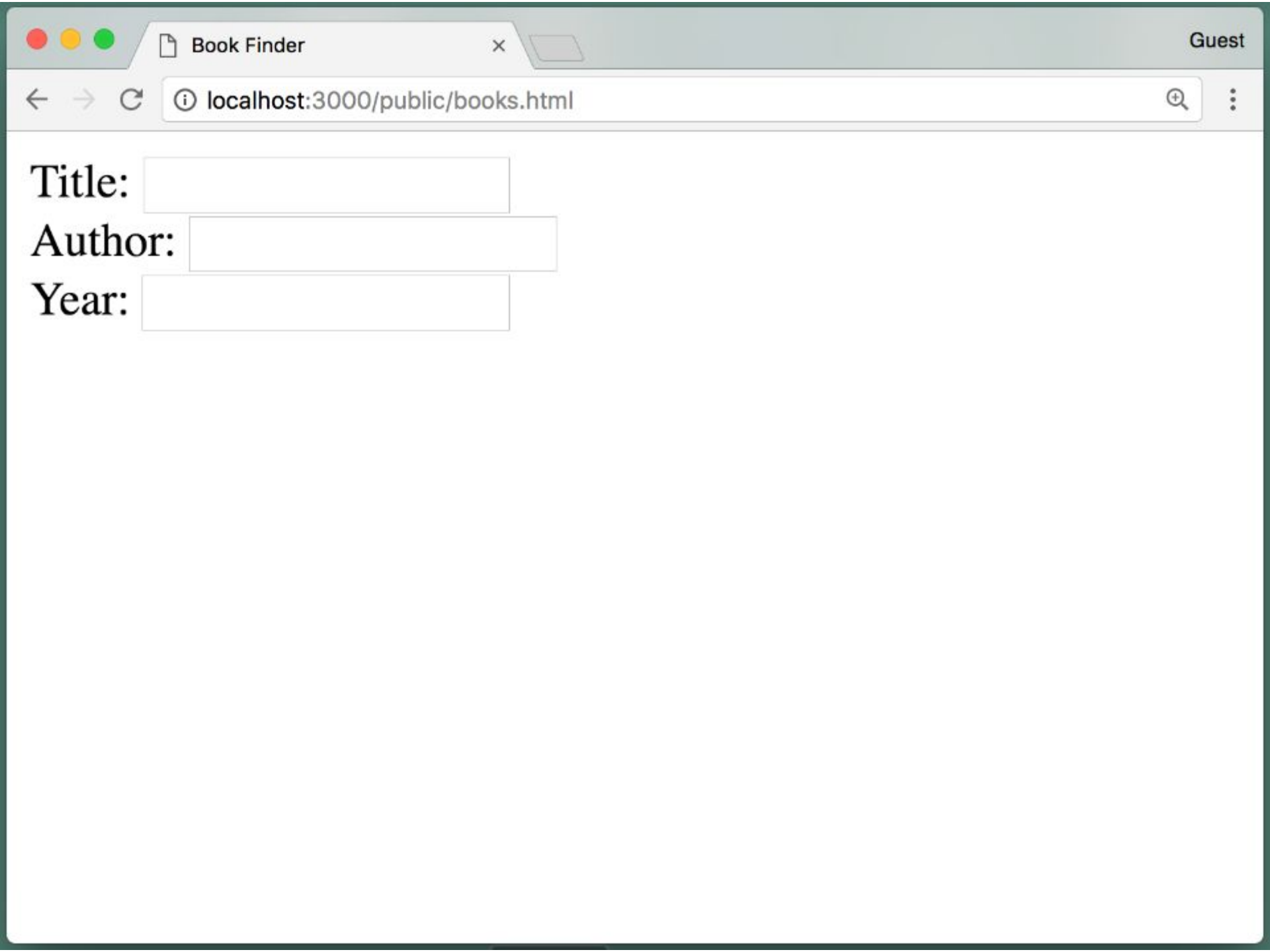
HTML


GET /public/books.html
HTML
GET /api?...
Query
JSON
Result

Now our client would have to have some sort of code running in the browser that will take that JSON data and render the HTML accordingly.

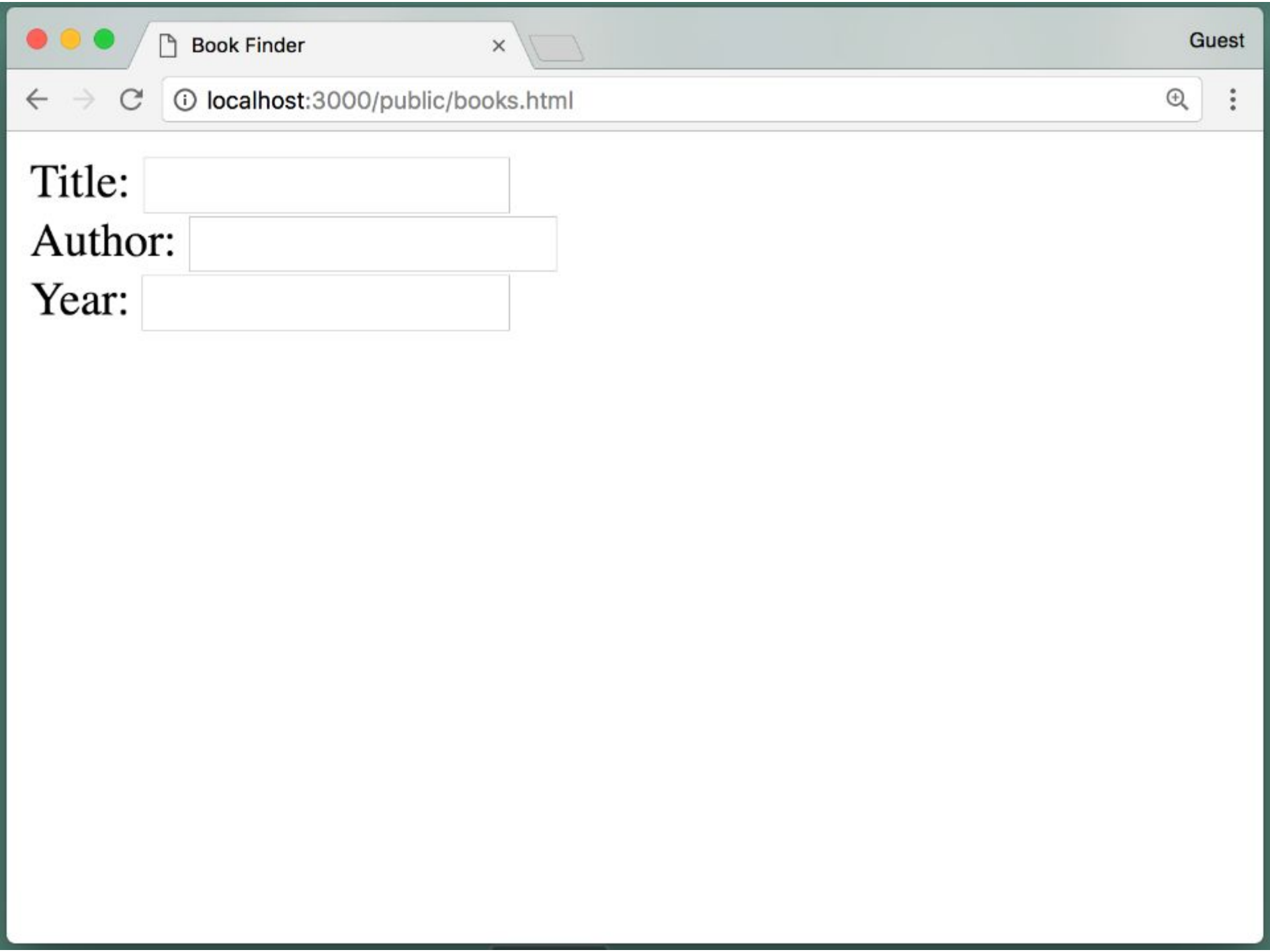

GET /public/books.html
HTML
GET /api?...
Query
JSON
Result
GET /api?...
Query
JSON
Result

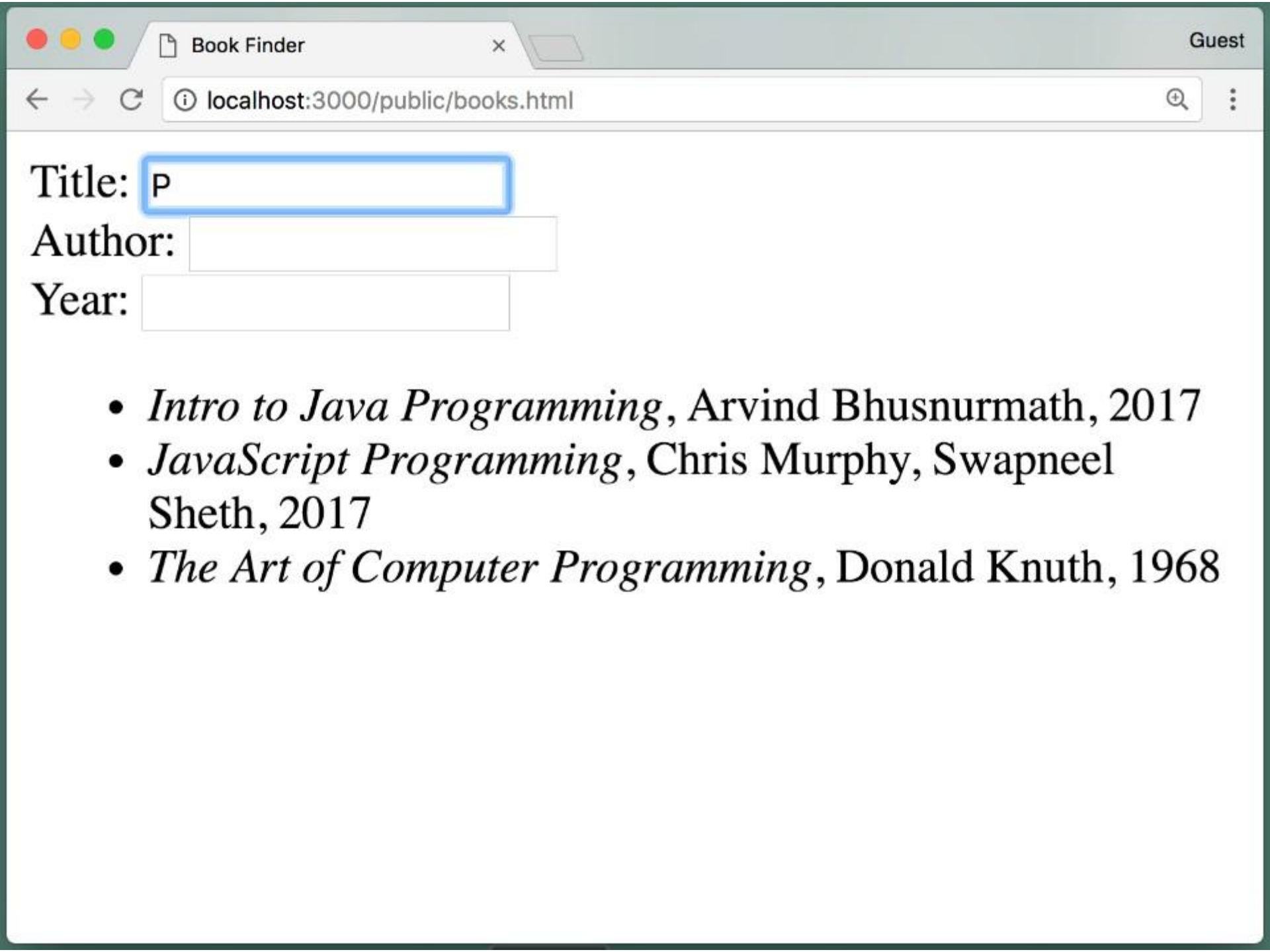

GET /public/books.html
HTML
GET /api?...
Query
JSON
Result
GET /api?...
Query
JSON
Result
GET /api?...
Query
JSON
Result

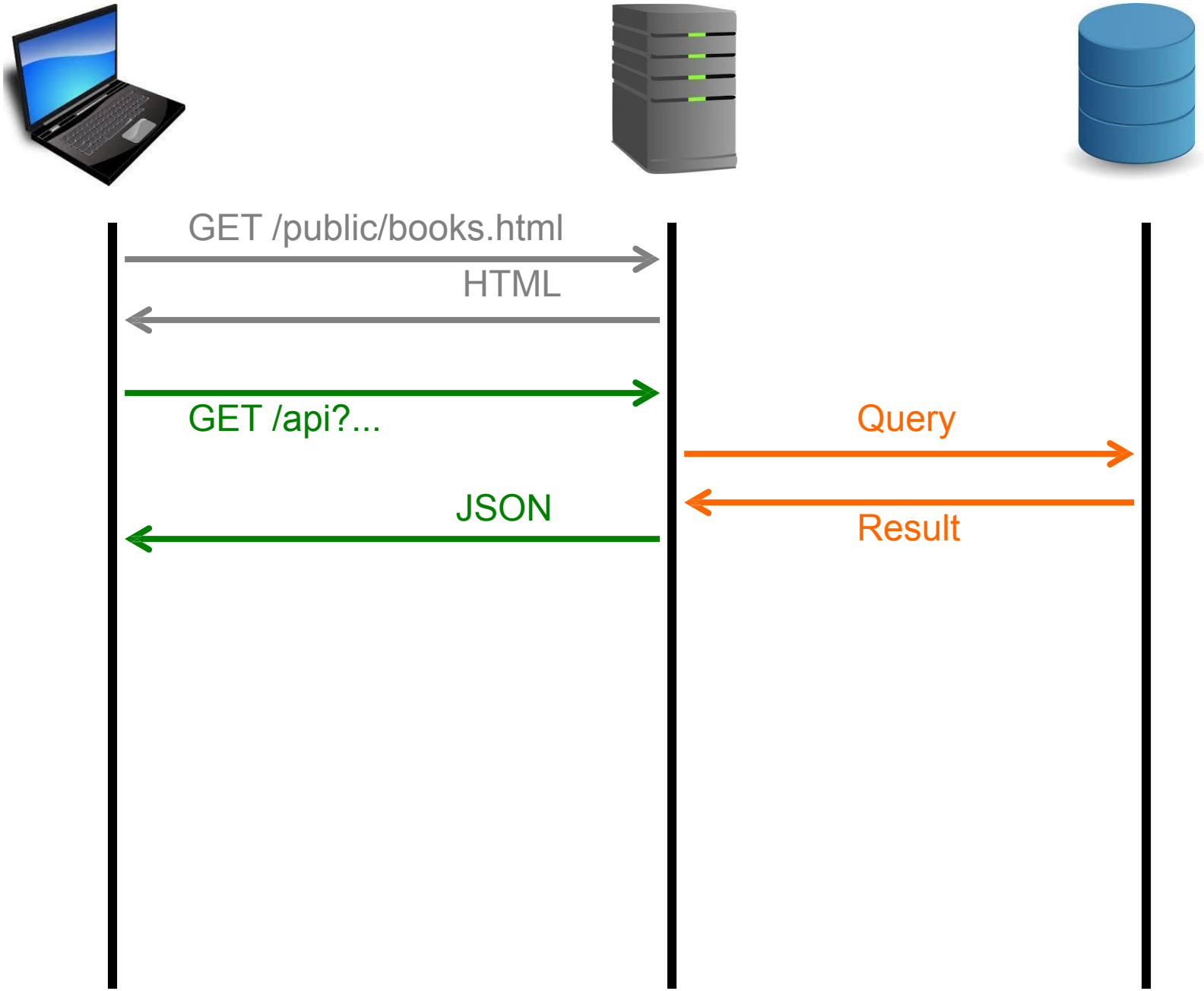

Book Finder
Guest
localhost:3000/public/books.html
Title:
Author:
Year:

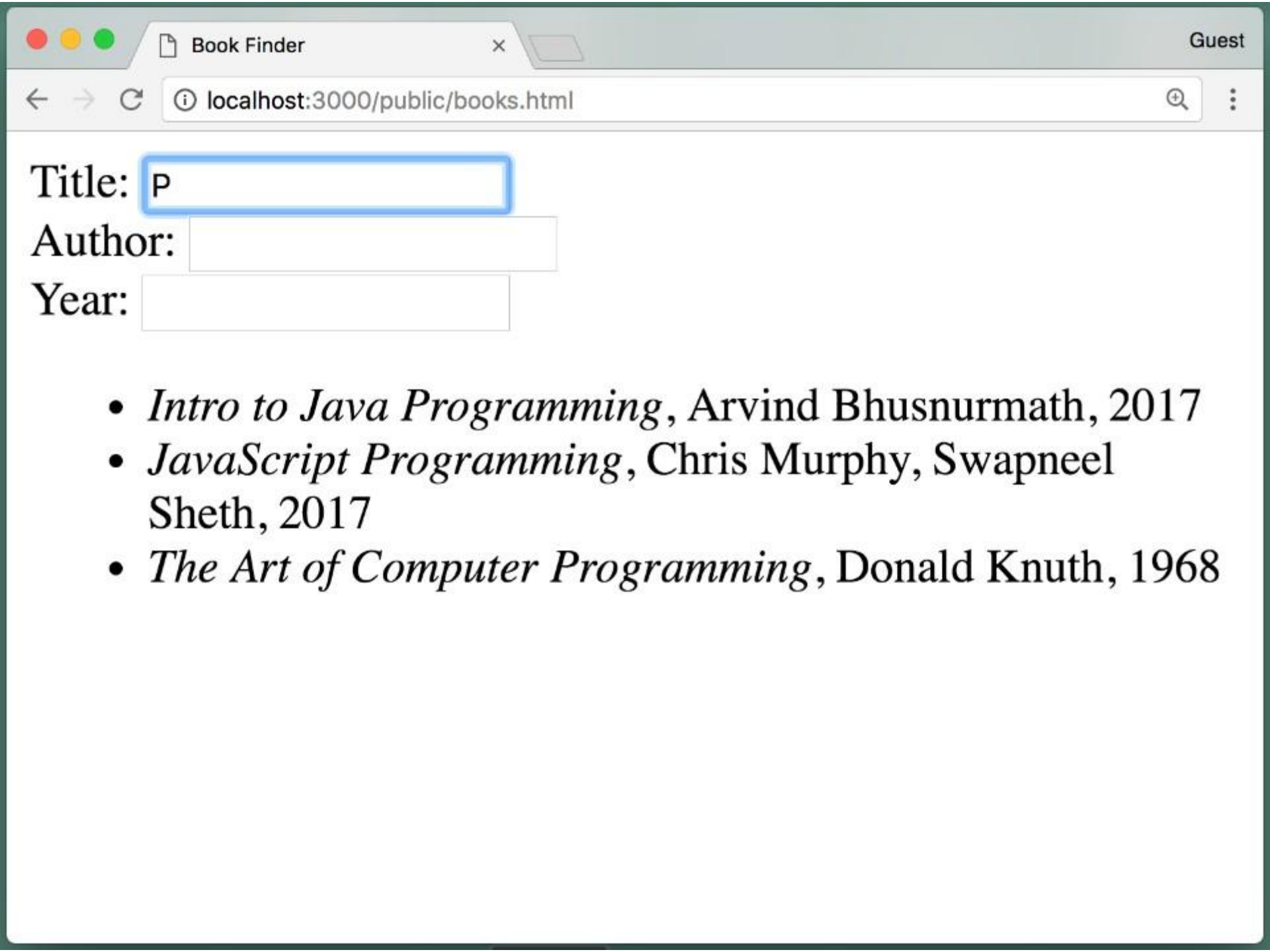
GET /public/books.html

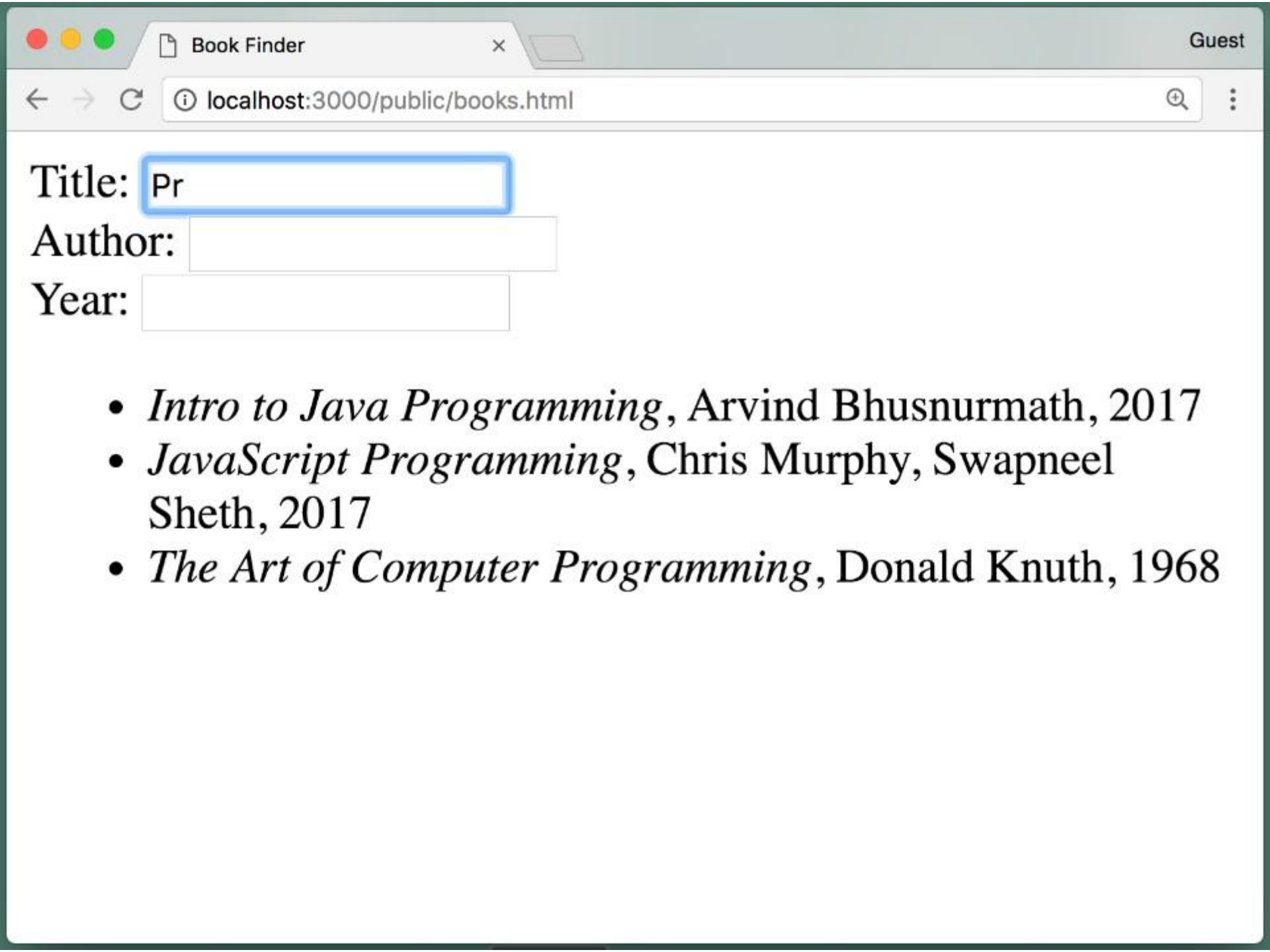
HTML

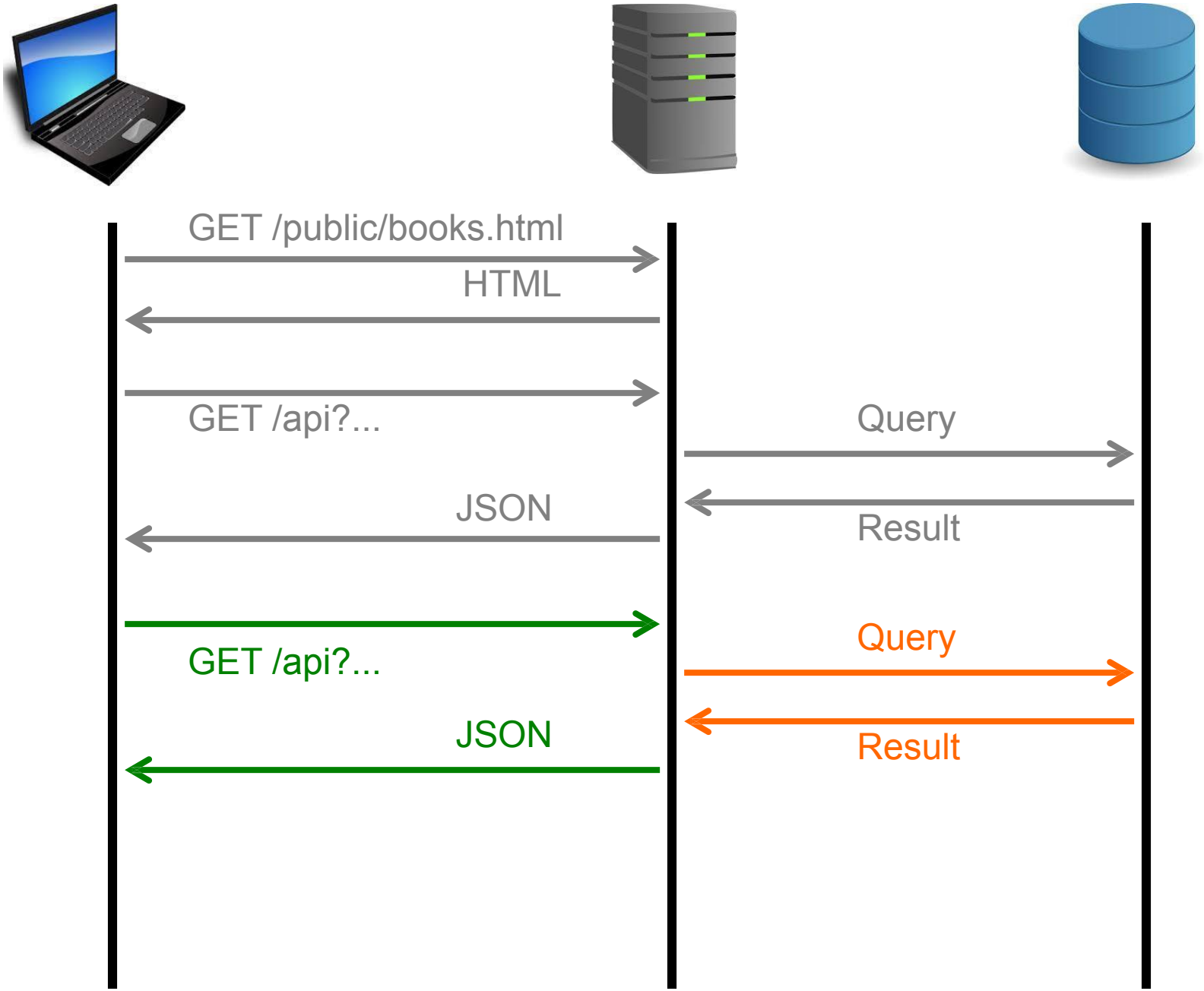

Book Finder
Guest
localhost:3000/public/books.html
Title:
Author:
Year:

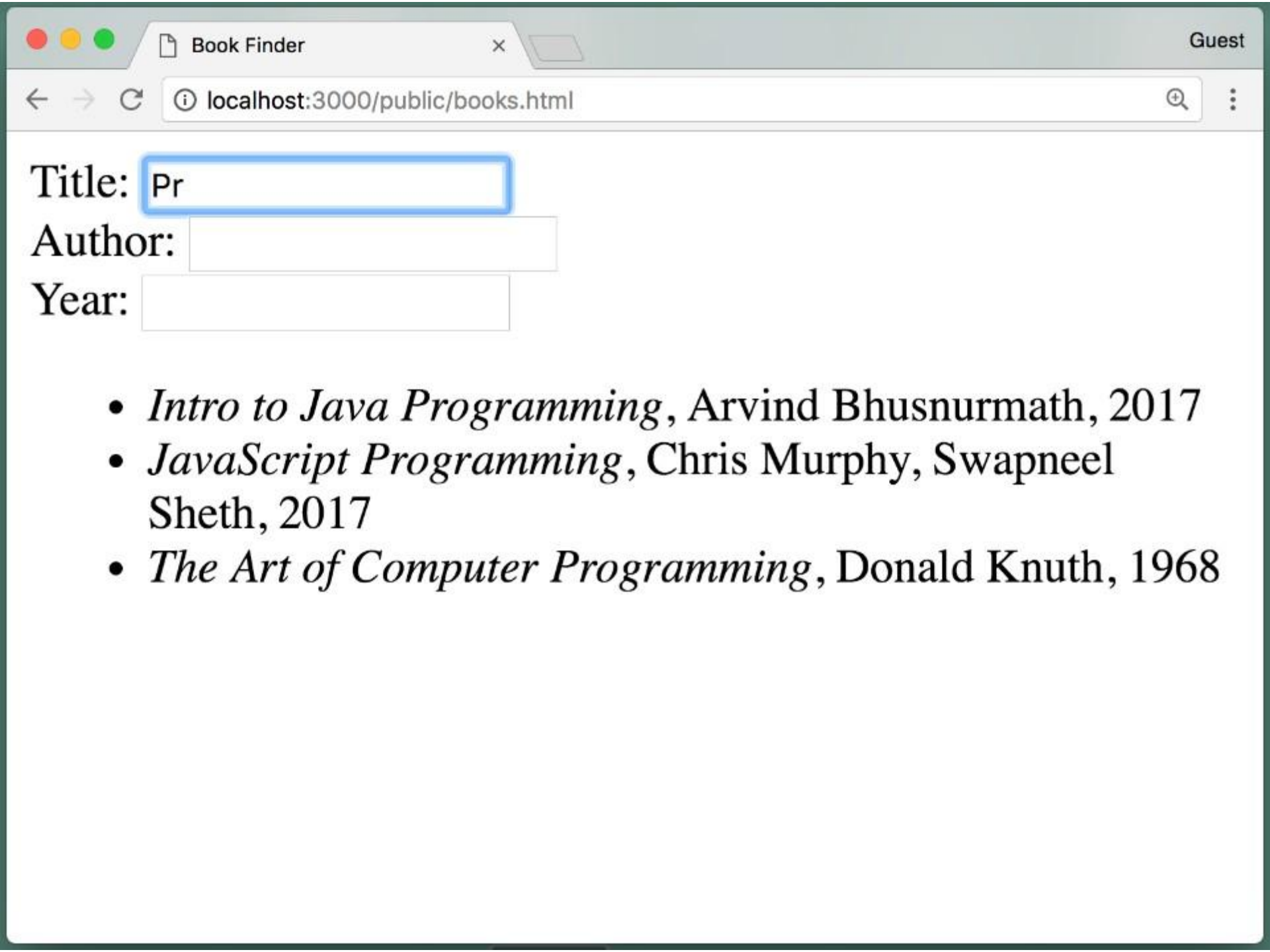

Book Finder
Guest
localhost:3000/public/books.html
Title: P
Author:
Year:
Intro to Java Programming, Arvind Bhusnurmath, 2017
JavaScript Programming, Chris Murphy, Swapneel Sheth, 2017
The Art of Computer Programming, Donald Knuth, 1968

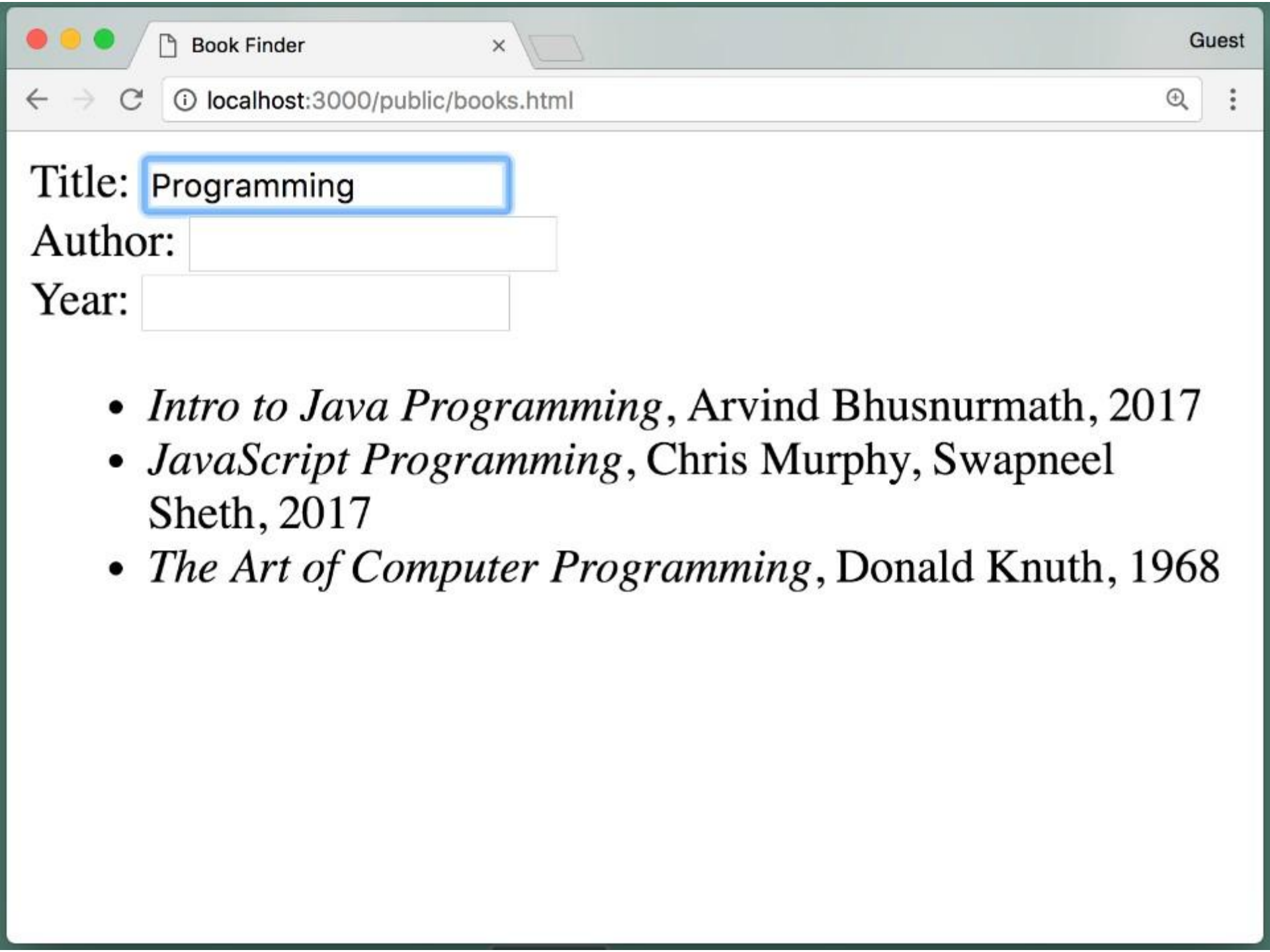

GET /public/books.html
HTML
GET /api?...
Query
JSON
Result

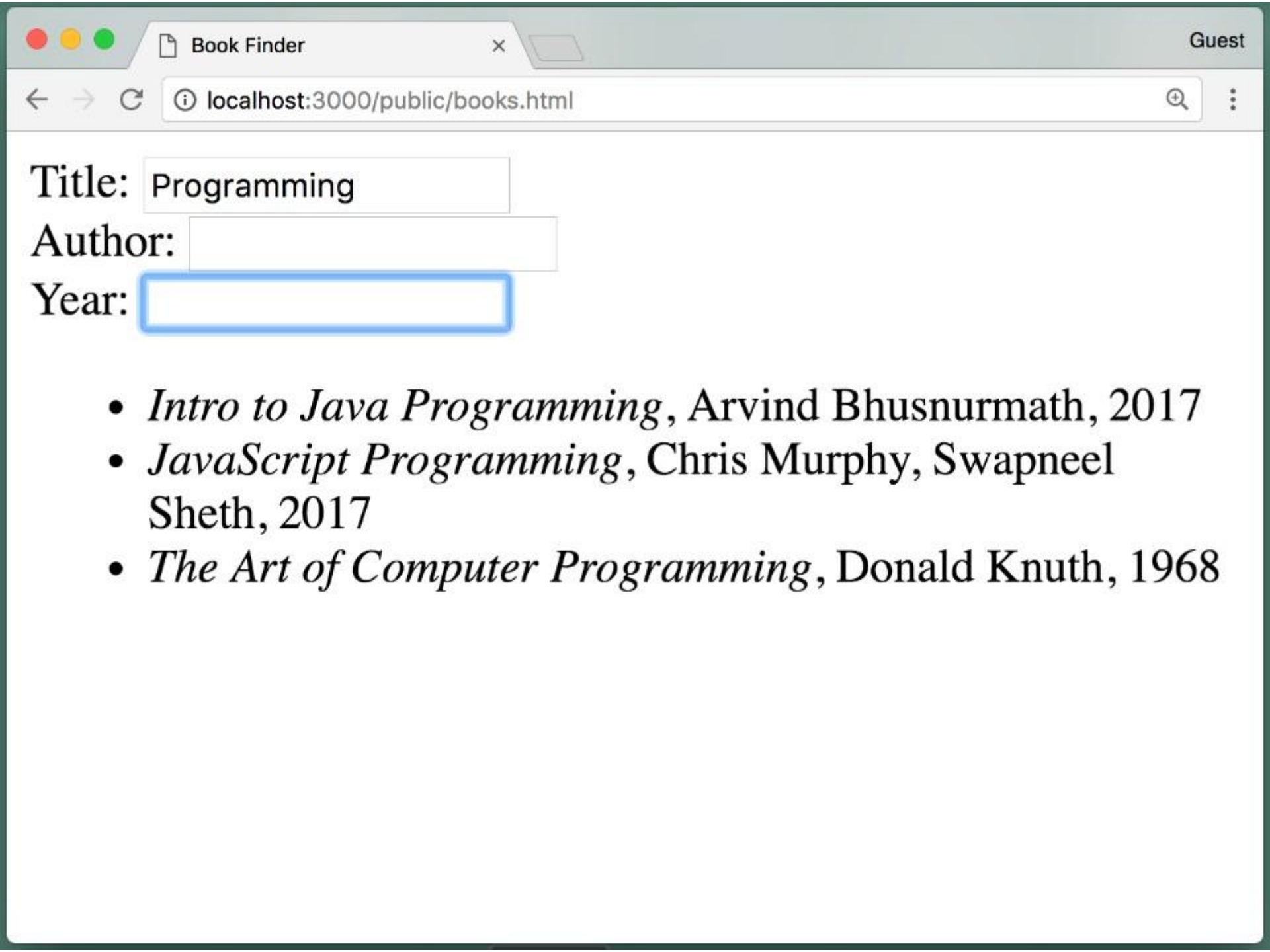

Book Finder
Guest
localhost:3000/public/books.html
Title: P
Author:
Year:
Intro to Java Programming, Arvind Bhusnurmath, 2017
JavaScript Programming, Chris Murphy, Swapneel Sheth, 2017
The Art of Computer Programming, Donald Knuth, 1968

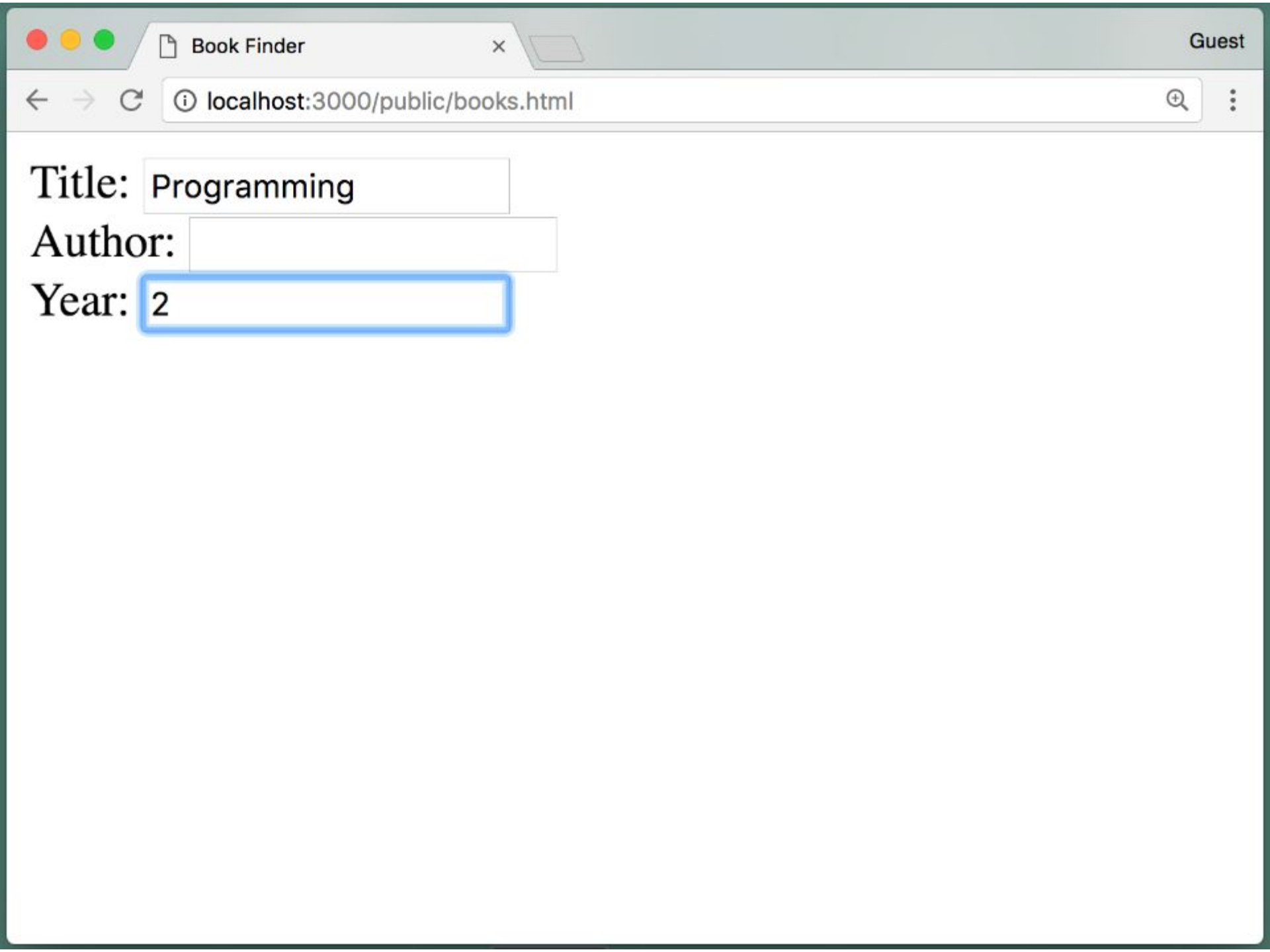

Book Finder
Guest
localhost:3000/public/books.html
Title: Pr
Author:
Year:
Intro to Java Programming, Arvind Bhusnurmath, 2017
JavaScript Programming, Chris Murphy, Swapneel Sheth, 2017
The Art of Computer Programming, Donald Knuth, 1968

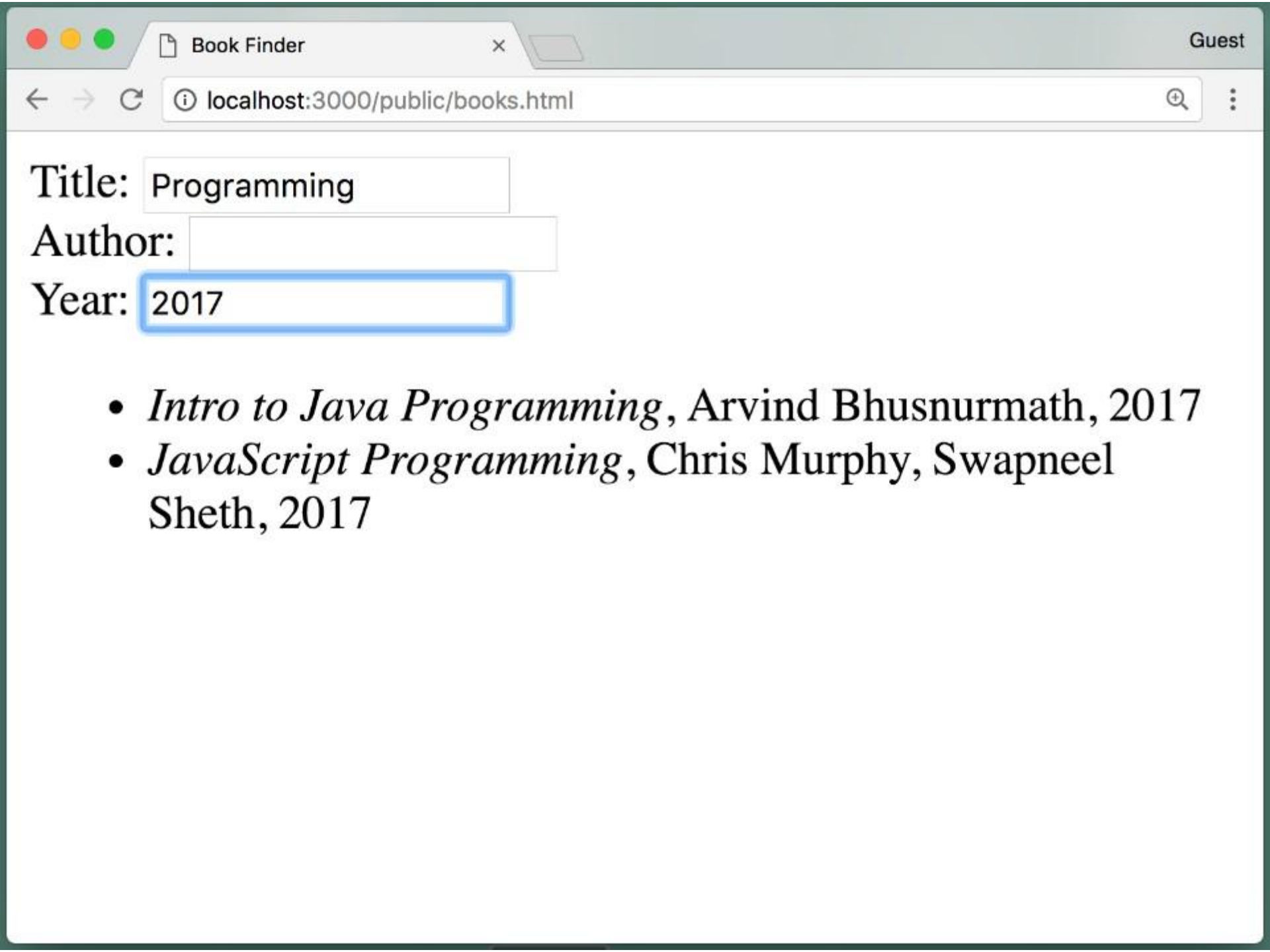

GET /public/books.html
HTML
GET /api?...
Query
JSON
Result
GET /api?...
Query
JSON
Result


Book Finder
Guest
localhost:3000/public/books.html
Title: Pr
Author:
Year:
Intro to Java Programming, Arvind Bhusnurmath, 2017
JavaScript Programming, Chris Murphy, Swapneel Sheth, 2017
The Art of Computer Programming, Donald Knuth, 1968


Book Finder
Guest
localhost:3000/public/books.html
Title: Programming
Author:
Year:
Intro to Java Programming, Arvind Bhusnurmath, 2017
JavaScript Programming, Chris Murphy, Swapneel Sheth, 2017
The Art of Computer Programming, Donald Knuth, 1968


Book Finder
Guest
localhost:3000/public/books.html
Title: Programming
Author:
Year:
Intro to Java Programming, Arvind Bhusnurmath, 2017
JavaScript Programming, Chris Murphy, Swapneel Sheth, 2017
The Art of Computer Programming, Donald Knuth, 1968


Book Finder
Guest
localhost:3000/public/books.html
Title: Programming
Author:
Year: 2


Book Finder
Guest
localhost:3000/public/books.html
Title: Programming
Author:
Year: 2017
Intro to Java Programming, Arvind Bhusnurmath, 2017
JavaScript Programming, Chris Murphy, Swapneel Sheth, 2017

Obviously in this example there's going to be some code running on the server.

Obviously in this example there's going to be some code running on the server.

It's going to accept all of the HTTP request, send the query to Mongo, gets the results back, construct an HTTP response, which might be JSON and then send it back to the browser.

Obviously in this example there's going to be some code running on the server.

It's going to accept all of the HTTP request, send the query to Mongo, gets the results back, construct an HTTP response, which might be JSON and then send it back to the browser.

There is also code running in the browser because there has to be some JavaScripts code there that's getting that JSON back from the server, and then it's updating the HTML accordingly.

Obviously in this example there's going to be some code running on the server.

It's going to accept all of the HTTP request, send the query to Mongo, gets the results back, construct an HTTP response, which might be JSON and then send it back to the browser.

There is also code running in the browser because there has to be some JavaScripts code there that's getting that JSON back from the server, and then it's updating the HTML accordingly.

Let’s look at that JavaScript code, starting with what's on the server in our Node express app.

```
// index.js
var express = require("express");
var app = express();

app.set("view engine", "ejs");

var bodyParser = require("body-parser");
app.use(bodyParser.urlencoded({ extended: true }));

var Book = require("./Book.js");

app.use("/public", express.static("public"));
```

... (next slides)

```
// index.js
var express = require("express");
var app = express();

app.set("view engine", "ejs");

var bodyParser = require("body-parser");
app.use(bodyParser.urlencoded({ extended: true }));

var Book = require("./Book.js");

app.use("/public", express.static("public"));
```

... (next slides)

We saw in the previous lesson how we could create the book schema in ebook.js.

```
// index.js
... (Previous slide)
app.use("/api", async (req, res) => {
 try {
   const query = {};
   if (req.query.title) {
     query.title = { $regex: req.query.title };
   }
   if (req.query.name) {
     query["authors.name"] = { $regex: req.query.name };
   }
   if (req.query.year) {
     query.year = req.query.year;
   }
   if (Object.keys(query).length ===0) {
     res.json({});
   } else {
     const books = await Book.find(query);
     res.json(books);
   }
 } catch (err) {
   res.type("html").status(500);
   res.send("Error: " + err);
 }
});
```

```
// index.js
... (Previous slide)
app.use("/api", async (req, res) => {
 try {
   const query = {};
   if (req.query.title) {
     query.title = { $regex: req.query.title };
   }
   if (req.query.name) {
     query["authors.name"] = { $regex: req.query.name };
   }
   if (req.query.year) {
     query.year = req.query.year;
   }
   if (Object.keys(query).length === 0) {
     res.json({});
   } else {
     const books = await Book.find(query);
     res.json(books);
   }
 } catch (err) {
   res.type("html").status(500);
   res.send("Error: " + err);
 }});
```

we'll specify /api as the URI that we're looking for.
And then we have our callback function.

```
// index.js
... (Previous slide)
app.use("/api", async (req, res) => {
 try {
   const query = {};
   if (req.query.title) {
     query.title = { $regex: req.query.title };
   }
   if (req.query.name) {
     query["authors.name"] = { $regex: req.query.name };
   }
   if (req.query.year) {
     query.year = req.query.year;
   }
   if (Object.keys(query).length === 0) {
     res.json({});
   } else {
     const books = await Book.find(query);
     res.json(books);
   }
 } catch (err) {
   res.type("html").status(500);
   res.send("Error: " + err);
 }});
```

we're going to construct our query object. This is going to be an all query, or an and query. So we just have one object that has all of the properties that we want to search for.

```
// index.js
... (Previous slide)
app.use("/api", async (req, res) => {
 try {
   const query = {};
   if (req.query.title) {
     query.title = { $regex: req.query.title };
   }
   if (req.query.name) {
     query["authors.name"] = { $regex: req.query.name };
   }
   if (req.query.year) {
     query.year = req.query.year;
   }
   if (Object.keys(query).length === 0) {
     res.json({});
   } else {
     const books = await Book.find(query);
     res.json(books);
   }
 } catch (err) {
   res.type("html").status(500);
   res.send("Error: " + err);
 }});
```

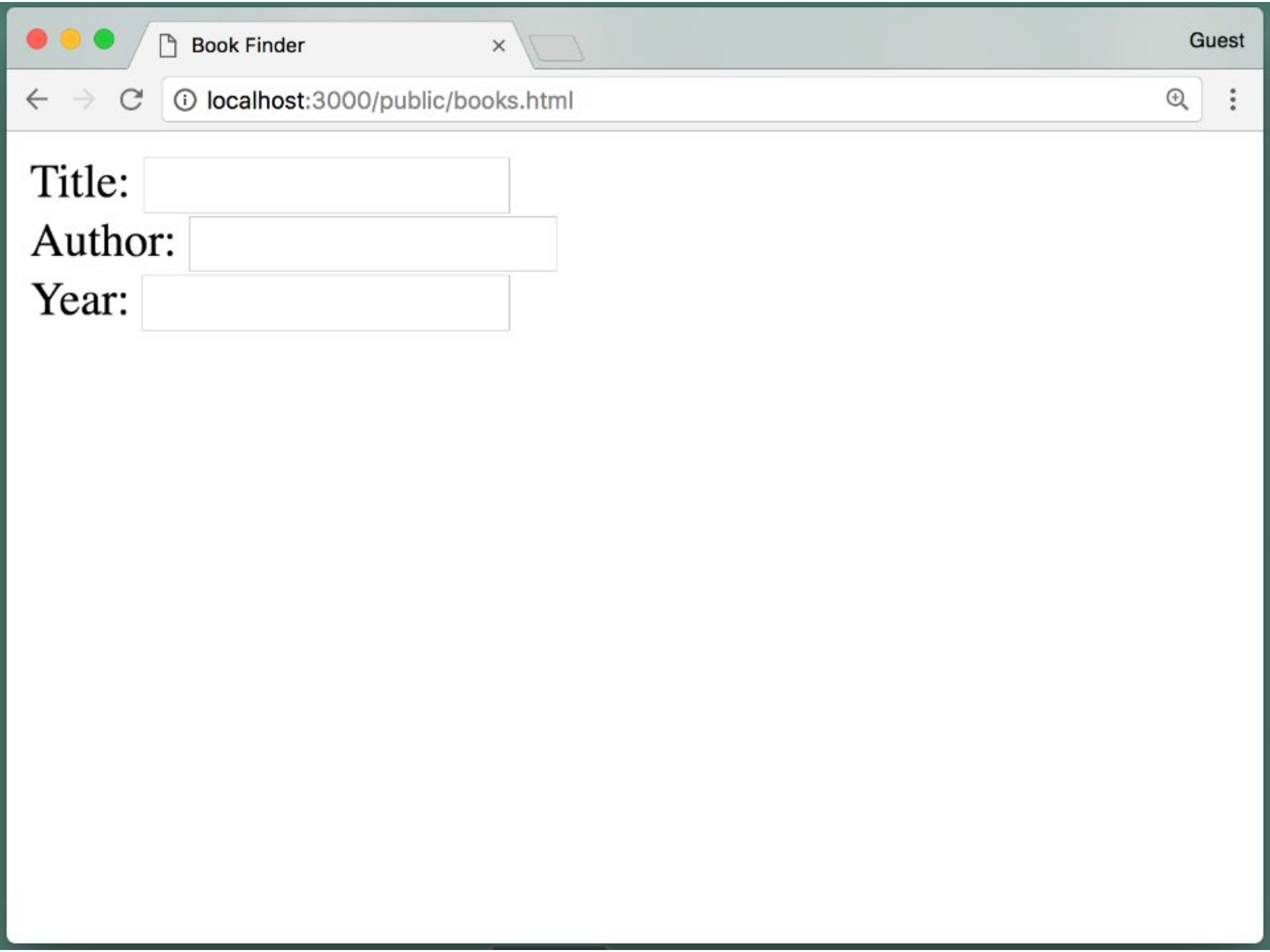
In the previous lesson, we got the values out of the body of the request because they were being submitted as part of a form. But we're not using form submission here, rather, the title, in this case, would come right out of the query.
Which is the part of the URL that follows the question mark.

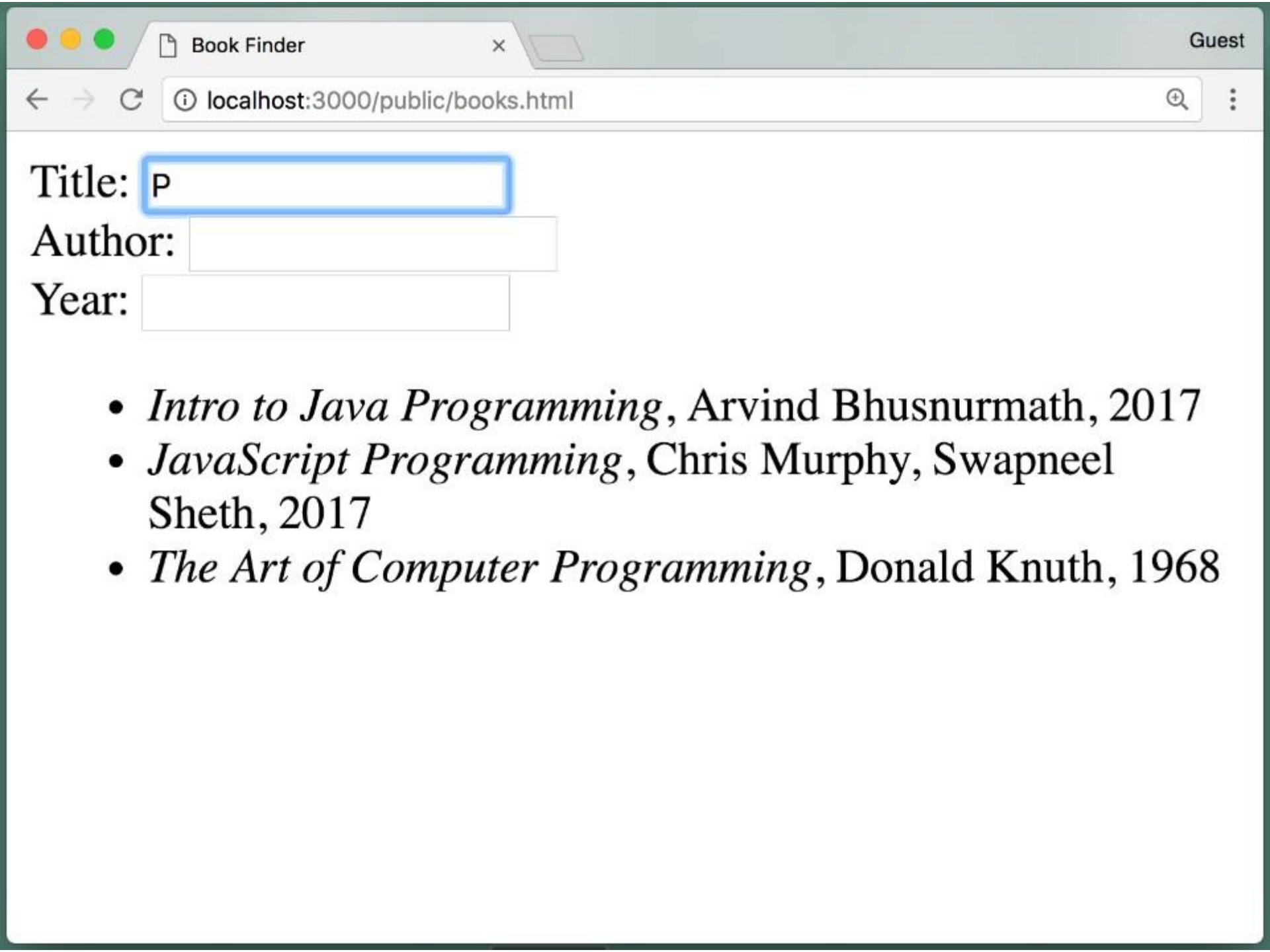
```
// index.js
... (Previous slide)
app.use("/api", async (req, res) => {
 try {
   const query = {};
   if (req.query.title) {
     query.title = { $regex: req.query.title };
   }
   if (req.query.name) {
     query["authors.name"] = { $regex: req.query.name };
   }
   if (req.query.year) {
     query.year = req.query.year;
   }
   if (Object.keys(query).length === 0) {
     res.json({});
   } else {
     const books = await Book.find(query);
     res.json(books);
   }
 } catch (err) {
   res.type("html").status(500);
   res.send("Error: " + err);
 }});
```

So we've constructed our query that we're going to pass to Mongo in order to find all of the books.

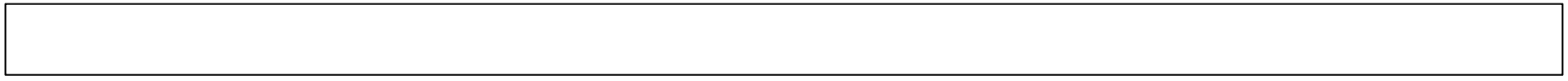
```
// index.js
... (Previous slide)
app.use("/api", async (req, res) => {
 try {
   const query = {};
   if (req.query.title) {
     query.title = { $regex: req.query.title };
   }
   if (req.query.name) {
     query["authors.name"] = { $regex: req.query.name };
   }
   if (req.query.year) {
     query.year = req.query.year;
   }
   if (Object.keys(query).length === 0) {
     res.json({});
   } else {
     const books = await Book.find(query);
     res.json(books);
   }
 } catch (err) {
   res.type("html").status(500);
   res.send("Error: " + err);
 }});
```

```
// index.js
... (Previous slide)
app.use("/api", async (req, res) => {
 try {
   const query = {};
   if (req.query.title) {
     query.title = { $regex: req.query.title };
   }
   if (req.query.name) {
     query["authors.name"] = { $regex: req.query.name };
   }
   if (req.query.year) {
     query.year = req.query.year;
   }
   if (Object.keys(query).length === 0) {
     res.json({});
   } else {
     const books = await Book.find(query);
     res.json(books);
   }
 } catch (err) {
   res.type("html").status(500);
   res.send("Error: " + err);
 }});
```

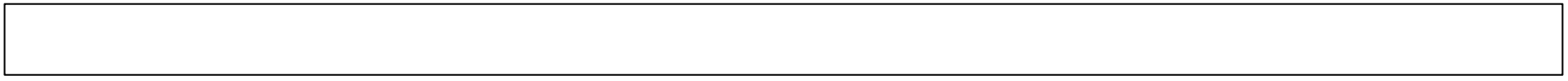

Book Finder
Guest
localhost:3000/public/books.html
Title:
Author:
Year:


Book Finder
Guest
localhost:3000/public/books.html
Title: P
Author:
Year:
Intro to Java Programming, Arvind Bhusnurmath, 2017
JavaScript Programming, Chris Murphy, Swapneel Sheth, 2017
The Art of Computer Programming, Donald Knuth, 1968

```
<!DOCTYPE html>
<html>

<head>
 <title>Book Finder</title>
 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
</head>

<body>
 <form>
   Title: <input name='title'><br>
   Author: <input name='author'><br>
   Year: <input name='year'><br>
 </form>
 <ul><span id='results'></span></ul>
 <script>
//  Next Slide
 </script>

</body>

</html>
```



```
$("input").on("change input textInput", () => {
   var title = $("input[name='title']").val();
   var author = $("input[name='author']").val();
   var year = $("input[name='year']").val();

   var query = '?';
   if (title) query += 'title=' + title + '&';
   if (author) query += 'name=' + author + '&';
   if (year) query += 'year=' + year;

   //$("#results").html(query);

   var url = 'http://localhost:3000/api/' + query;

   $.getJSON(url, (books, status)=> {

     var results = $("#results");
     results.html('');

     var output = '';
     books.forEach((book)=> {
       results.append('<li><i>' + book.title + '</i>, ');
       book.authors.forEach((author)=> {
         if (author.name) results.append(author.name +', ');
       });
       results.append(book.year +'</li>');
     }); }); });
```



```
$("input").on("change input textInput", () => {
   var title = $("input[name='title']").val();
   var author = $("input[name='author']").val();
   var year = $("input[name='year']").val();

   var query = '?';
   if (title) query += 'title=' + title + '&';
   if (author) query += 'name=' + author + '&';
   if (year) query += 'year=' + year;

   //$("#results").html(query);

   var url = 'http://localhost:3000/api/' + query;

   $.getJSON(url, (books, status) => {

     var results = $("#results");
     results.html('');

     var output = '';
     books.forEach((book) => {
       results.append('<li><i>' + book.title + '</i>, ');
       book.authors.forEach((author) => {
         if (author.name) results.append(author.name + ', ');
       });
       results.append(book.year + '</li>');
     }); }); });
```

We start out by using a jQuery selector to select all of the input elements in the page. We want this code to run for an action on any of those three inputs.

```
$("input").on("change input textInput", () => {
   var title = $("input[name='title']").val();
   var author = $("input[name='author']").val();
   var year = $("input[name='year']").val();

   var query = '?';
   if (title) query += 'title=' + title + '&';
   if (author) query += 'name=' + author + '&';
   if (year) query += 'year=' + year;

   //$("#results").html(query);

   var url = 'http://localhost:3000/api/' + query;

   $.getJSON(url, (books, status) => {

     var results = $("#results");
     results.html('');

     var output = '';
     books.forEach((book) => {
       results.append('<li><i>' + book.title + '</i>, ');
       book.authors.forEach((author) => {
         if (author.name) results.append(author.name + ', ');
       });
       results.append(book.year + '</li>');
     }); }); });
```

The event for which we're defining the call back function is the **on** event.
on is just the general way of saying, look for these specific events, and then call this call back function.

```
$("input").on("change input textInput", () => {
   var title = $("input[name='title']").val();
   var author = $("input[name='author']").val();
   var year = $("input[name='year']").val();

   var query = '?';
   if (title) query += 'title=' + title + '&';
   if (author) query += 'name=' + author + '&';
   if (year) query += 'year=' + year;

   //$("#results").html(query);

   var url = 'http://localhost:3000/api/' + query;

   $.getJSON(url, (books, status) => {

     var results = $("#results");
     results.html('');

     var output = '';
     books.forEach((book) => {
       results.append('<li><i>' + book.title + '</i>, ');
       book.authors.forEach((author) => {
         if (author.name) results.append(author.name + ', ');
       });
       results.append(book.year + '</li>');
     }); }); });
```

In jQuery lecture, we saw that we could pass an object to the **on** function, where the properties were the names of the events, and their values were the call back functions. Another way of using the on function is to pass a string that lists the name of the events for which were listening.

```
$("input").on("change input textInput", () => {
   var title = $("input[name='title']").val();
   var author = $("input[name='author']").val();
   var year = $("input[name='year']").val();

   var query = '?';
   if (title) query += 'title=' + title + '&';
   if (author) query += 'name=' + author + '&';
   if (year) query += 'year=' + year;

   //$("#results").html(query);

   var url = 'http://localhost:3000/api/' + query;

   $.getJSON(url, (books, status) => {

     var results = $("#results");
     results.html('');

     var output = '';
     books.forEach((book) => {
       results.append('<li><i>' + book.title + '</i>, ');
       book.authors.forEach((author) => {
         if (author.name) results.append(author.name + ', ');
       });
       results.append(book.year + '</li>');
     }); }); });
```

different browsers consider these events to have different names. But we want this code to run whenever the user types anything into any of those fields.

```
$("input").on("change input textInput", () => {
   var title = $("input[name='title']").val();
   var author = $("input[name='author']").val();
   var year = $("input[name='year']").val();

   var query = '?';
   if (title) query += 'title=' + title + '&';
   if (author) query += 'name=' + author + '&';
   if (year) query += 'year=' + year;

   //$("#results").html(query);

   var url = 'http://localhost:3000/api/' + query;

   $.getJSON(url, (books, status) => {

     var results = $("#results");
     results.html('');

     var output = '';
     books.forEach((book) => {
       results.append('<li><i>' + book.title + '</i>, ');
       book.authors.forEach((author) => {
         if (author.name) results.append(author.name + ', ');
       });
       results.append(book.year + '</li>');
     }); }); });
```

```
$("input").on("change input textInput", () => {
   var title = $("input[name='title']").val();
   var author = $("input[name='author']").val();
   var year = $("input[name='year']").val();

   var query = '?';
   if (title) query += 'title=' + title + '&';
   if (author) query += 'name=' + author + '&';
   if (year) query += 'year=' + year;

   //$("#results").html(query);

   var url = 'http://localhost:3000/api/' + query;

   $.getJSON(url, (books, status) => {

     var results = $("#results");
     results.html('');

     var output = '';
     books.forEach((book) => {
       results.append('<li><i>' + book.title + '</i>, ');
       book.authors.forEach((author) => {
         if (author.name) results.append(author.name + ', ');
       });
       results.append(book.year + '</li>');
     }); }); });
```

start with a query string that begins with a question mark.

```
$("input").on("change input textInput", () => {
   var title = $("input[name='title']").val();
   var author = $("input[name='author']").val();
   var year = $("input[name='year']").val();

   var query = '?';
   if (title) query += 'title=' + title + '&';
   if (author) query += 'name=' + author + '&';
   if (year) query += 'year=' + year;

   //$("#results").html(query);

   var url = 'http://localhost:3000/api/' + query;

   $.getJSON(url, (books, status) => {

     var results = $("#results");
     results.html('');

     var output = '';
     books.forEach((book) => {
       results.append('<li><i>' + book.title + '</i>, ');
       book.authors.forEach((author) => {
         if (author.name) results.append(author.name + ', ');
       });
       results.append(book.year + '</li>');
     }); }); });
```

the entire URL to request the data from the node express server using the API that we wrote.

```
$("input").on("change input textInput", () => {
   var title = $("input[name='title']").val();
   var author = $("input[name='author']").val();
   var year = $("input[name='year']").val();

   var query = '?';
   if (title) query += 'title=' + title + '&';
   if (author) query += 'name=' + author + '&';
   if (year) query += 'year=' + year;

   //$("#results").html(query);

   var url = 'http://localhost:3000/api/' + query;

   $.getJSON(url, (books, status) => {

     var results = $("#results");
     results.html('');

     var output = '';
     books.forEach((book) => {
       results.append('<li><i>' + book.title + '</i>, ');
       book.authors.forEach((author) => {
         if (author.name) results.append(author.name + ', ');
       });
       results.append(book.year + '</li>');
     }); }); });
```

We've seen this before, when we did the New York Times API example, with React. But we're going to use a library, from jQuery, called Ajax, which defines a function in the jQuery library, called getJSON.

```
$("input").on("change input textInput", () => {
   var title = $("input[name='title']").val();
   var author = $("input[name='author']").val();
   var year = $("input[name='year']").val();

   var query = '?';
   if (title) query += 'title=' + title + '&';
   if (author) query += 'name=' + author + '&';
   if (year) query += 'year=' + year;

   //$("#results").html(query);

   var url = 'http://localhost:3000/api/' + query;

   $.getJSON(url, (books, status) => {

     var results = $("#results");
     results.html('');

     var output = '';
     books.forEach((book) => {
       results.append('<li><i>' + book.title + '</i>, ');
       book.authors.forEach((author) => {
         if (author.name) results.append(author.name + ', ');
       });
       results.append(book.year + '</li>');
     }); }); });
```

When the data comes back from the server as JSON we're going to invoke this callback function where the data that came back will be this parameter, books.

```
$("input").on("change input textInput", () => {
   var title = $("input[name='title']").val();
   var author = $("input[name='author']").val();
   var year = $("input[name='year']").val();

   var query = '?';
   if (title) query += 'title=' + title + '&';
   if (author) query += 'name=' + author + '&';
   if (year) query += 'year=' + year;

   //$("#results").html(query);

   var url = 'http://localhost:3000/api/' + query;

   $.getJSON(url, (books,status) => {

     var results = $("#results");
     results.html('');

     var output = '';
     books.forEach((book) => {
       results.append('<li><i>' + book.title + '</i>, ');
       book.authors.forEach((author) => {
         if (author.name) results.append(author.name + ', ');
       });
       results.append(book.year + '</li>');
     }); }); });
```

In this example, we're ignoring the status. Perhaps it would be better to see the status of the HTTP response so we could handle a error but assuming that everything is okay.

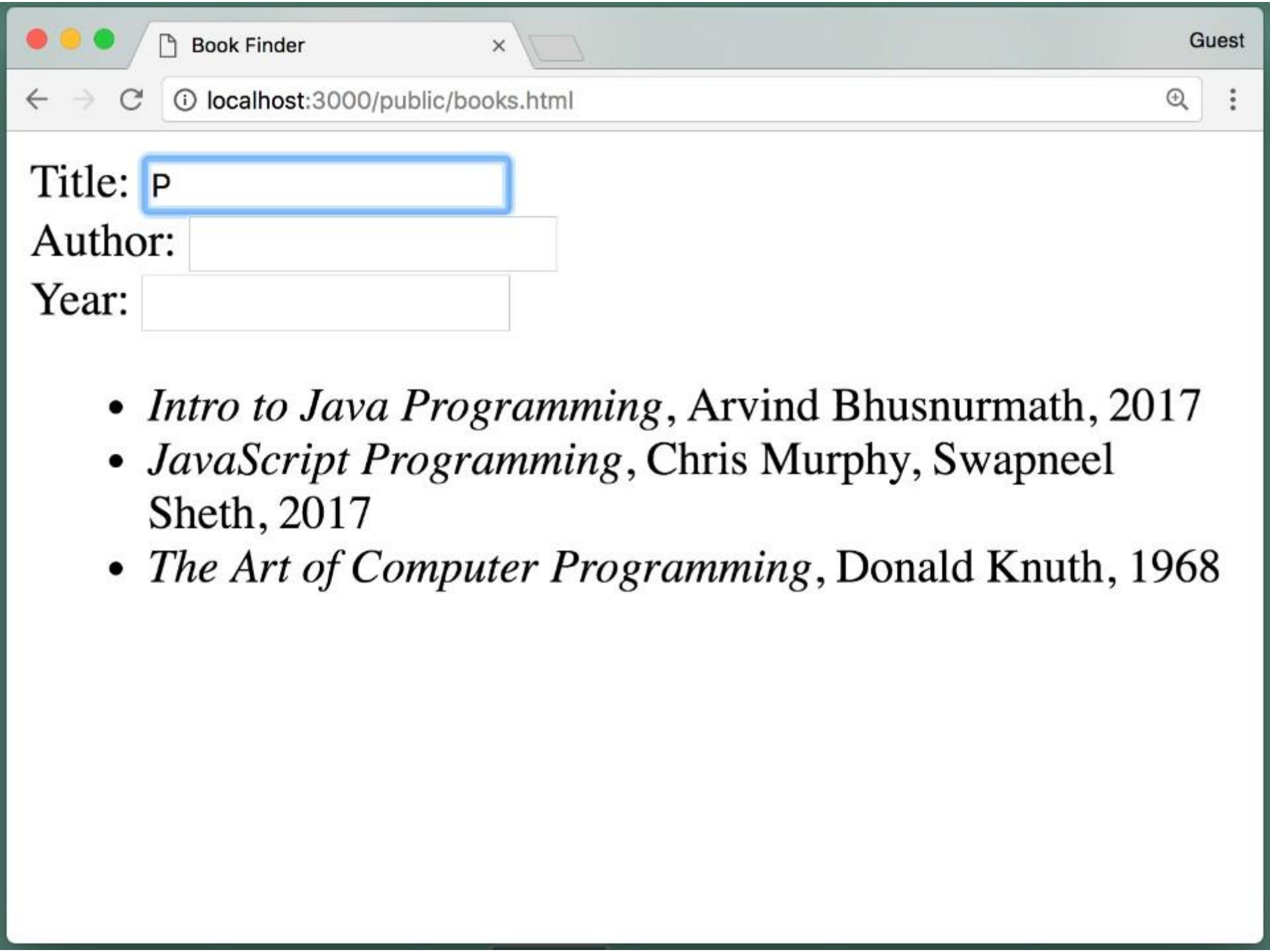
```
$("input").on("change input textInput", () => {
   var title = $("input[name='title']").val();
   var author = $("input[name='author']").val();
   var year = $("input[name='year']").val();

   var query = '?';
   if (title) query += 'title=' + title + '&';
   if (author) query += 'name=' + author + '&';
   if (year) query += 'year=' + year;

   //$("#results").html(query);

   var url = 'http://localhost:3000/api/' + query;

   $.getJSON(url, (books, status) => {

     var results = $("#results");
     results.html('');

     var output = '';
     books.forEach((book) => {
       results.append('<li><i>' + book.title + '</i>, ');
       book.authors.forEach((author) => {
         if (author.name) results.append(author.name + ', ');
       });
       results.append(book.year + '</li>');
     }); }); });
```

Let's update the HTML content using jQuery.
Clear the HTML content by using the HTML function with an empty string as the argument.

```
$("input").on("change input textInput", () => {
   var title = $("input[name='title']").val();
   var author = $("input[name='author']").val();
   var year = $("input[name='year']").val();

   var query = '?';
   if (title) query += 'title=' + title + '&';
   if (author) query += 'name=' + author + '&';
   if (year) query += 'year=' + year;

   //$("#results").html(query);

   var url = 'http://localhost:3000/api/' + query;

   $.getJSON(url, (books, status) => {

     var results = $("#results");
     results.html('');

     var output = '';
     books.forEach((book) => {
       results.append('<li><i>' + book.title + '</i>, ');
       book.authors.forEach((author) => {
         if (author.name) results.append(author.name + ', ');
       });
       results.append(book.year + '</li>');
     }); }); });
```

books is an array of objects, and each object has the title and year properties, and an authors property which is an array of objects.

```
$("input").on("change input textInput", () => {
   var title = $("input[name='title']").val();
   var author = $("input[name='author']").val();
   var year = $("input[name='year']").val();

   var query = '?';
   if (title) query += 'title=' + title + '&';
   if (author) query += 'name=' + author + '&';
   if (year) query += 'year=' + year;

   //$("#results").html(query);

   var url = 'http://localhost:3000/api/' + query;

   $.getJSON(url, (books, status) => {

     var results = $("#results");
     results.html('');

     var output = '';
     books.forEach((book) => {
       results.append('<li><i>' + book.title + '</i>, ');
       book.authors.forEach((author) => {
         if (author.name) results.append(author.name + ', ');
       });
       results.append(book.year + '</li>');
     }); }); });
```


Book Finder
Guest
localhost:3000/public/books.html
Title: P
Author:
Year:
Intro to Java Programming, Arvind Bhusnurmath, 2017
JavaScript Programming, Chris Murphy, Swapneel Sheth, 2017
The Art of Computer Programming, Donald Knuth, 1968

Summary

- We can use **server-side** tools such as Node.js, Express, and MongoDB to develop APIs to make data available on the Web

- We can use **client-side** tools such as jQuery, React.js, and D3.js to access that data and display it in a Web page