

# TECHSLIDES

Data, Maps, Usability, and Performance



## Client-Side JavaScript to Node.js

Last updated on **October 16, 2014** in [Development](#)



JavaScript is mostly used on the client-side and in the browser, enabling all kinds of dynamic interactions on the page. When Brendan Eich created the language for Netscape in 1995, there were already server side implementations but [node.js](#) made it really popular. Node is a cross-platform runtime environment that is event driven and non-blocking and uses the [Google V8 JavaScript engine](#) to execute code. So, if you already know JS, you can now code on the server. There are a ton of use cases that this enables but today I want to focus on a simple concept of moving your client-side JavaScript code to Node.js so that you can hide or protect your JS functions from being visible to end users.

Let's say you have a simple HTML page where users submit a form that is processed by client-side JavaScript via the form onsubmit event. For example, a simple addition calculator where the end user provides 2 numbers and JavaScript adds them up and returns and prints the result on the screen. Here is an example:

```
1  <!doctype html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Form Calculator Add Example</title>
6  </head>
7  <body>
8      <form name="myForm" action="" onsubmit="return calc(this['A'].value,this['B'].value);" method="post">
9          <input type="text" name="A"> +
10         <input type="text" name="B"> =
11         <span id="result"></span>
12         <br>
13         <input type="submit" value="Submit">
14     </form>
15     <script>
16         function calc(a,b){
17             //var a = document.forms["myForm"]["A"].value;
18             //var b = document.forms["myForm"]["B"].value;
19             document.querySelector("#result").textContent = Number(a)+Number(b);
20             return false;
21         }
22     </script>
```

```
23 </body>
24 </html>
```

form.html hosted with ❤ by GitHub

[view raw](#)

How can you transform that to Node.js so that you can hide the JavaScript function to end users? In our example, the onsubmit event handler passes form data into a client-side JS function but a form traditionally (without onsubmit) will post the data to the server (makes a HTTP request to a path specified in the form action parameter). Right now, we have no server setup to do anything with that form data so removing the onsubmit event will result in a page refresh.

Node.js will be our server and it will grab the data and pass it to the same JavaScript functions you had on the front-end but now everything lives on the server-side and end users cannot see those functions. Node should be [easy to install](#) and the first thing you want your script to do is to load the built-in http module to create an HTTP server that will listen for web requests and respond. If you save the file below as **form1.js** and run **node form1.js** in your command-line interface you should be able to hit **http://localhost:8000/** in your browser and see the text “hello”:

```
1 // Load the http module to create an http server.
```

```
2  var http = require('http');
3
4  // Create a function to handle every HTTP request
5  function handler(req, res){
6      res.setHeader('Content-Type', 'text/html');
7      res.writeHead(200);
8      res.end("<html><body><h1>Hello</h1></body></html>");
9  };
10
11 // Create a server that invokes the `handler` function upon receiving a request
12 http.createServer(handler).listen(8000, function(err){
13     if(err){
14         console.log('Error starting http server');
15     } else {
16         console.log("Server running at http://127.0.0.1:8000/ or http://localhost:8000/");
17     };
18 });
```

form1.js hosted with ❤ by GitHub

[view raw](#)

Now, the browser, by default makes HTTP GET requests to a server because it reads web pages. Your form is trying to submit something so it is making an HTTP POST request. So, we will write a conditional in our node.js script and respond with the calculator form on a GET request and for the POST, we will capture the form data, process it, and respond with a form and the result of our calculation. Here is how that looks:

```
1 // Load the http module to create an http server.
2 var http = require('http');
3
4 // Create a function to handle every HTTP request
5 function handler(req, res){
6
7     var form = '';
8
9     if(req.method == "GET"){
10
11         form = '<!doctype html> \
12 <html lang="en"> \
13 <head> \
14     <meta charset="UTF-8"> \
15     <title>Form Calculator Add Example</title> \
16 </head> \
17 <body> \
18     <form name="myForm" action="/" method="post">\
19         <input type="text" name="A"> + \
20         <input type="text" name="B"> = \
21         <span id="result"></span> \
22         <br> \
23         <input type="submit" value="Submit"> \
24     </form> \
25 </body> \
26 </html>';
27
28 //respond
29 res.setHeader('Content-Type', 'text/html');
30 res.writeHead(200);
31 res.end(form);
32
```

```
33 } else if(req.method == 'POST'){
34
35     //read form data
36     req.on('data', function(chunk) {
37
38         //grab form data as string
39         var formdata = chunk.toString();
40
41         //grab A and B values
42         var a = eval(formdata.split("&")[0]);
43         var b = eval(formdata.split("&")[1])
44
45         var result = calc(a,b);
46
47         //console.log(chunk.toString());
48         //console.log(a);
49         //console.log(b);
50         //console.log(result);
51
52         //fill in the result and form values
53         form = '<!doctype html> \
54 <html lang="en"> \
55 <head> \
56     <meta charset="UTF-8"> \
57     <title>Form Calculator Add Example</title> \
58 </head> \
59 <body> \
60     <form name="myForm" action="/" method="post">\
61         <input type="text" name="A" value="'+a+'"> + \
62         <input type="text" name="B" value="'+b+'"> = \
63         <span id="result">'+result+'</span> \
64         <br> \
```

```
65     <input type="submit" value="Submit"> \
66   </form> \
67 </body> \
68 </html>';
69
70   //respond
71   res.setHeader('Content-Type', 'text/html');
72   res.writeHead(200);
73   res.end(form);
74
75   });
76
77   } else {
78     res.writeHead(200);
79     res.end();
80   };
81
82 };
83
84 //js functions running only in Node.JS
85 function calc(a,b){
86   return Number(a)+Number(b);;
87 }
88
89 // Create a server that invokes the `handler` function upon receiving a request
90 http.createServer(handler).listen(8000, function(err){
91   if(err){
92     console.log('Error starting http server');
93   } else {
94     console.log("Server running at http://127.0.0.1:8000/ or http://localhost:8000/");
95   };
96 });
```



The code could be refactored in many ways but to show everything in a single file, I think this is a good example of transforming a form with client-side JavaScript to a server-side Node.js version. It's a bit of pain to grab Form Data so if you are doing this for a serious project, I would recommend a framework like [Express](#) which makes writing web apps much easier.

This form example makes a page request to the server but you could make it feel like the original example by using AJAX (XMLHttpRequest) to pass and retrieve data from the server without a full page refresh. All you need is a few lines of additional client-side JS code to make the Ajax requests via the onsubmit form event. I also transformed the Node response function to only respond with the result instead of the full page. Here is how that would look like with Ajax and Node.js in a single file:

```
1 // Load the http module to create an http server.
2 var http = require('http');
3
4 // Create a function to handle every HTTP request
5 function handler(req, res){
6
7     var form = '';
```

```
8
9  if(req.method == "GET"){
10
11      form = '<!doctype html> \
12 <html lang="en"> \
13 <head> \
14     <meta charset="UTF-8"> \
15     <title>Form Calculator Add Example</title> \
16 </head> \
17 <body> \
18     <form name="myForm" action="" onsubmit="return ajax();"method="post">\
19         <input type="text" name="A"> + \
20         <input type="text" name="B"> = \
21         <span id="result"></span> \
22         <br> \
23         <input type="submit" value="Submit"> \
24     </form> \
25     <script> \
26         function ajax(){ \
27             var a = document.forms["myForm"]["A"].value; \
28             var b = document.forms["myForm"]["B"].value; \
29             var formdata = "A="+a+"&B="+b; \
30             \
31             xmlhttp = new XMLHttpRequest(); \
32             xmlhttp.onreadystatechange=function(){ \
33                 if(xmlhttp.readyState==4 && xmlhttp.status==200){ \
34                     document.getElementById("result").innerHTML=xmlhttp.responseText; \
35                 }; \
36             }; \
37             xmlhttp.open("POST","",true); \
38             xmlhttp.send(formdata); \
39             return false; \
```

```
40     } \
41     </script> \
42 </body> \
43 </html>';
44
45     //respond
46     res.setHeader('Content-Type', 'text/html');
47     res.writeHead(200);
48     res.end(form);
49
50     } else if(req.method == 'POST'){
51
52         //read form data
53         req.on('data', function(chunk) {
54
55             //grab form data as string
56             var formdata = chunk.toString();
57
58             //grab A and B values
59             var a = eval(formdata.split("&")[0]);
60             var b = eval(formdata.split("&")[1])
61
62             var result = calc(a,b);
63
64             //fill in the result and form values
65             form = result.toString();
66
67             //respond
68             res.setHeader('Content-Type', 'text/html');
69             res.writeHead(200);
70             res.end(form);
71
```

```
72     });
73
74     } else {
75         res.writeHead(200);
76         res.end();
77     };
78
79 };
80
81 //js functions running only in Node.JS
82 function calc(a,b){
83     return Number(a)+Number(b);;
84 }
85
86 // Create a server that invokes the `handler` function upon receiving a request
87 http.createServer(handler).listen(8000, function(err){
88     if(err){
89         console.log('Error starting http server');
90     } else {
91         console.log("Server running at http://127.0.0.1:8000/ or http://localhost:8000/");
92     };
93 });
```

Tags: [ajax](#), [JavaScript](#), [nodejs](#)




0 Comments

TechSlides

 Login ▾

 Recommend 6

 Share

Sort by Best ▾






Start the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

Name

Be the first to comment.

 Subscribe  Add Disqus to your siteAdd DisqusAdd  Disqus' Privacy PolicyPrivacy PolicyPrivacy PolicyPrivacy

## Archives

[August 2016](#)  
[July 2016](#)

## Categories

[Data Visualization](#)  
[Design](#)

June 2016  
July 2015  
June 2015  
May 2015  
April 2015  
March 2015  
February 2015  
January 2015  
December 2014  
November 2014  
October 2014  
September 2014  
August 2014  
April 2014  
March 2014  
February 2014  
January 2014  
December 2013  
November 2013  
October 2013  
September 2013

Development  
Domains  
SEO  
Social Media  
Uncategorized

August 2013  
July 2013  
June 2013  
May 2013  
April 2013  
March 2013  
February 2013  
January 2013  
December 2012  
November 2012  
October 2012  
September 2012  
August 2012  
July 2012  
June 2012  
May 2012

Copyright ©2016 TechSlides, All Rights Reserved