

Module 10: Microservices Application

Demo Document 2

edureka!

edureka!

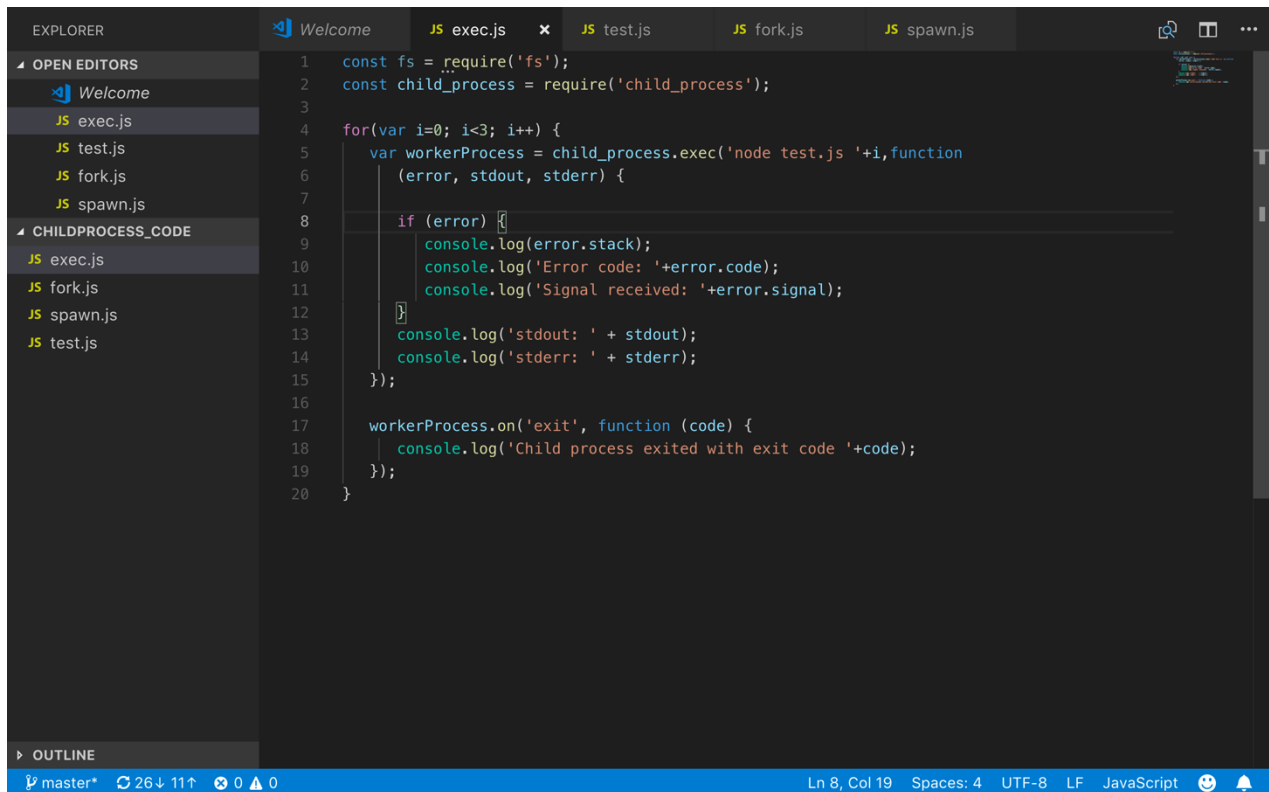
© Brain4ce Education Solutions Pvt. Ltd.

Working Of Child Process

DEMO Steps:

Step 1: Let us create two .js files named test.js and exec.js.

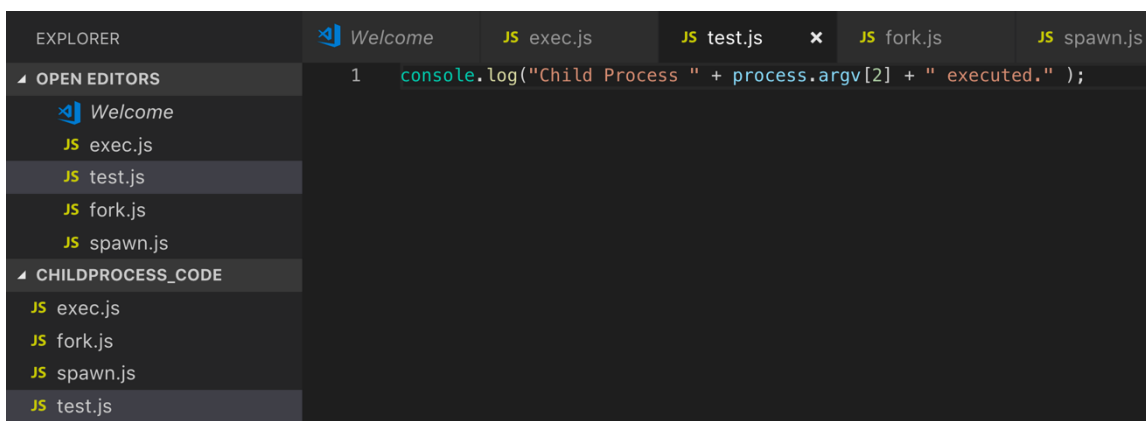
child_process.exec method runs a command in a shell and buffers the output

A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a file tree with 'OPEN EDITORS' and 'CHILDPROCESS_CODE' sections. The 'OPEN EDITORS' section lists 'Welcome', 'exec.js', 'test.js', 'fork.js', and 'spawn.js'. The 'CHILDPROCESS_CODE' section lists 'exec.js', 'fork.js', 'spawn.js', and 'test.js'. The main editor area has tabs for 'Welcome', 'exec.js', 'test.js', 'fork.js', and 'spawn.js'. The 'exec.js' tab is active, displaying the following JavaScript code:

```
1  const fs = require('fs');
2  const child_process = require('child_process');
3
4  for(var i=0; i<3; i++) {
5      var workerProcess = child_process.exec('node test.js '+i,function
6          (error, stdout, stderr) {
7
8          if (error) {
9              console.log(error.stack);
10             console.log('Error code: '+error.code);
11             console.log('Signal received: '+error.signal);
12         }
13         console.log('stdout: ' + stdout);
14         console.log('stderr: ' + stderr);
15     });
16
17     workerProcess.on('exit', function (code) {
18         console.log('Child process exited with exit code '+code);
19     });
20 }
```

The status bar at the bottom indicates 'master', '26 ↓ 11 ↑', '0 ▲ 0', 'Ln 8, Col 19', 'Spaces: 4', 'UTF-8', 'LF', 'JavaScript', and a notification bell.

Step 2: In test.js write console.log and we will be executing this file with Exec.js

A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a file tree with 'OPEN EDITORS' and 'CHILDPROCESS_CODE' sections. The 'OPEN EDITORS' section lists 'Welcome', 'exec.js', 'test.js', 'fork.js', and 'spawn.js'. The 'CHILDPROCESS_CODE' section lists 'exec.js', 'fork.js', 'spawn.js', and 'test.js'. The main editor area has tabs for 'Welcome', 'exec.js', 'test.js', 'fork.js', and 'spawn.js'. The 'test.js' tab is active, displaying the following JavaScript code:

```
1  console.log("Child Process " + process.argv[2] + " executed." );
```

The status bar at the bottom indicates 'master', '26 ↓ 11 ↑', '0 ▲ 0', 'Ln 1, Col 1', 'Spaces: 4', 'UTF-8', 'LF', 'JavaScript', and a notification bell.

Step 3: Verify the Output, once the server has started

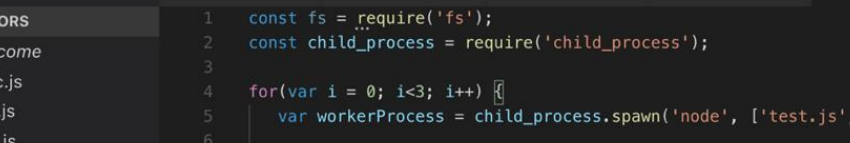
```
childprocess_code — -bash — 80x24
Avyaans-MacBook-Pro:childprocess_code avi$ node exec.js
Child process exited with exit code 0
stdout: Child Process 0 executed.

stderr:
Child process exited with exit code 0
stdout: Child Process 2 executed.

stderr:
Child process exited with exit code 0
stdout: Child Process 1 executed.

stderr:
Avyaans-MacBook-Pro:childprocess_code avi$
```

Step 4: Instead of using `exec` now try code using `child_process.spawn()`. Now create one new `spawn.js` file and execute the same test file with `spawn`



```
1  const fs = require('fs');
2  const child_process = require('child_process');
3
4  for(var i = 0; i<3; i++) {
5      var workerProcess = child_process.spawn('node', ['test.js', i]);
6
7      workerProcess.stdout.on('data', function (data) {
8          console.log('stdout: ' + data);
9      });
10
11     workerProcess.stderr.on('data', function (data) {
12         console.log('stderr: ' + data);
13     });
14
15     workerProcess.on('close', function (code) {
16         console.log('child process exited with code ' + code);
17     });
18 }
```

Step 5: Now run node spawn.js in command prompt and output of spawn with child process

```
Avyaans-MacBook-Pro:childprocess_code avi$ node spawn.js
stdout: Child Process 0 executed.

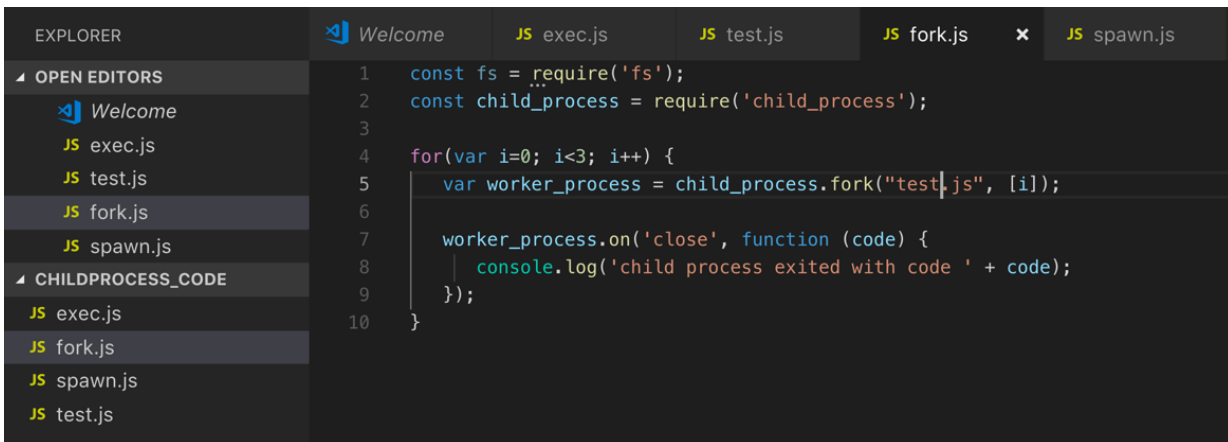
stdout: Child Process 1 executed.

child process exited with code 0
stdout: Child Process 2 executed.

child process exited with code 0
child process exited with code 0
Avyaans-MacBook-Pro:childprocess_code avi$
```

Step 6: FORK() method `child_process.fork` method is a special case of `spawn()` to create Node processes.

Create two js files named `fork.js` and `test.js`



The screenshot shows the VS Code editor interface. The Explorer sidebar on the left lists the files: `Welcome`, `exec.js`, `test.js`, `fork.js`, `spawn.js`, and a folder named `CHILDPROCESS_CODE` containing `exec.js`, `fork.js`, `spawn.js`, and `test.js`. The main editor area shows the code for `fork.js`:

```
1  const fs = require('fs');
2  const child_process = require('child_process');
3
4  for(var i=0; i<3; i++) {
5      var worker_process = child_process.fork("test.js", [i]);
6
7      worker_process.on('close', function (code) {
8          console.log('child process exited with code ' + code);
9      });
10 }
```

Step 7: Verify the output, once the server has started.



The screenshot shows a terminal window titled `childprocess_code — -bash — 80x24`. The prompt is `Avyaans-MacBook-Pro:childprocess_code avi$`. The command `node fork.js` has been executed, resulting in the following output:

```
Child Process 0 executed.
Child Process 2 executed.
child process exited with code 0
child process exited with code 0
Child Process 1 executed.
child process exited with code 0
Avyaans-MacBook-Pro:childprocess_code avi$
```