WRITING PDF – continuation

FirstWritingPDF.js U | India.pdf (read-only) U

Javascript > Lecture 30 > FirstWritingPDF.js > createScoreCard

```
// npm install pdf-lib
// node FirstWritingPDF.js --source=teams.json --dest=worldCup

let minimist = require("minimist");
let fs = require("fs");
let path = require("path");
let pdf = require("pdf-lib");

let args = minimist(process.argv);

let teamsJSON = fs.readFileSync(args.source, "utf-8");
let teams = JSON.parse(teamsJSON); //conversion to JSON to JSO -> array of teams object

fs.mkdirSync(args.dest); // creation of worldCup folder

for(let i = 0; i < teams.length; i++){
    let teamFolder = path.join(args.dest, teams[i].name);
    fs.mkdirSync(teamFolder); // team name folder creation of each team (India, Australia, England)
                              //in worldCup folder -> eg worldCup\India
    for(let j = 0; j < teams[i].matches.length; j++){
        let matchFileName = path.join(teamFolder,teams[i].matches[j].vs + ".pdf");
        createScoreCard(teams[i].name, teams[i].matches[j], matchFileName);
    }
}

function createScoreCard(teamName, match, matchFileName){
  // this function creates pdf for match in appropriate folder with correct details
  //here we will use pdf-lib to create the pdf

  let t1 = teamName;
  let t2 = match.vs;
  let result = t1 + " " + match.result;

  let pdfDocument = pdf.PDFDocument; // taken out the pdfDocument from pdf-lib -> pdf-lib has a property called as PDFDocument
  let originalTemplateBytes = fs.readFileSync("Template.pdf"); // getting the bytes from template
  let prmToLoadBytes = pdfDocument.load(originalTemplateBytes); // promise that I will load the Bytes
  prmToLoadBytes.then(function(pdfdoc){ // if I load the bytes and fulfill the promise, then I will call function(pdfdoc)
    let page = pdfdoc.getPage(0);
    page.drawText("Hello World");
    let promiseToSave = pdfdoc.save();
    promiseToSave.then(function(changedBytes){
      fs.writeFileSync(matchFileName, changedBytes);
    })
  })
}
```

Worldcup 2019

Team 1	
Team 2	
Result	

Hello World

FirstWritingPDF.js U

Javascript > Lecture 30 > FirstWritingPDF.js > createScoreCard > prmToLoadBytes.then() callback > promiseToSave.then() callback

```
// npm install pdf-lib
// node FirstWritingPDF.js --source=teams.json --dest=worldCup

let minimist = require("minimist");
let fs = require("fs");
let path = require("path");
let pdf = require("pdf-lib");

let args = minimist(process.argv);

let teamsJSON = fs.readFileSync(args.source, "utf-8");
let teams = JSON.parse(teamsJSON); //conversion to JSON to JSO -> array of teams object

fs.mkdirSync(args.dest); // creation of worldCup folder

for(let i = 0; i < teams.length; i++){
    let teamFolder = path.join(args.dest, teams[i].name);
    fs.mkdirSync(teamFolder); // team name folder creation of each team (India, Australia, England)
                              //in worldCup folder -> eg worldCup\India
    for(let j = 0; j < teams[i].matches.length; j++){
        let matchFileName = path.join(teamFolder,teams[i].matches[j].vs + ".pdf");
        createScoreCard(teams[i].name, teams[i].matches[j], matchFileName);
    }
}

function createScoreCard(teamName, match, matchFileName){
  // this function creates pdf for match in appropriate folder with correct details
  //here we will use pdf-lib to create the pdf

  let t1 = teamName;
  let t2 = match.vs;
  let result = t1 + " " + match.result;

  let pdfDocument = pdf.PDFDocument; // taken out the pdfDocument from pdf-lib -> pdf-lib has a property called as PDFDocument
  let originalTemplateBytes = fs.readFileSync("Template.pdf"); // getting the bytes from template
  let prmToLoadBytes = pdfDocument.load(originalTemplateBytes); // promise that I will load the Bytes
  prmToLoadBytes.then(function(pdfdoc){ // if I load the bytes and fulfill the promise, then I will call function(pdfdoc)
      let page = pdfdoc.getPage(0);
      page.drawText(result);
      let promiseToSave = pdfdoc.save();
      promiseToSave.then(function(changedBytes){  // promise that I will save the file and if so then I will give you the Bytes
          fs.writeFileSync(matchFileName, changedBytes);
      })
  })
}
```

Template

Worldcup 2019

Team 1	
Team 2	
Result	

Australia Win

```
function createScoreCard(teamName, match, matchFileName){
  // this function creates pdf for match in appropriate folder with correct details
  //here we will use pdf-lib to create the pdf

  let t1 = teamName;
  let t2 = match.vs;
  let result = t1 + " " + match.result;

  let pdfDocument = pdf.PDFDocument; // taken out the pdfDocument from pdf-lib -> pdf-lib has a property called as PDFDocument
  let originalTemplateBytes = fs.readFileSync("Template.pdf"); // getting the bytes from template
  let prmToLoadBytes = pdfDocument.load(originalTemplateBytes); // promise that I will load the Bytes
  prmToLoadBytes.then(function(pdfdoc){ // if I load the bytes and fulfill the promise, then I will call function(pdfdoc)
    let page = pdfdoc.getPage(0);
    page.drawText(t1, {
     x: 320,
     y: 710,
     size: 10
    });
    page.drawText(t2, {
     x: 320,
     y: 696,
     size: 10
    });
    page.drawText(result, {
      x: 320,
      y: 681,
       size: 10
    });
    let promiseToSave = pdfdoc.save();
    promiseToSave.then(function(changedBytes){  // promise that I will save the file and if so then I will give you the Bytes
      fs.writeFileSync(matchFileName, changedBytes);
    })
  })
}
```

the coordinates are specified to position the text at appropriate place

Worldcup 2019

Team 1	England
Team 2	Australia
Result	England Loss

TIMER

- The setInterval() method calls a function or evaluates an expression at specified intervals (in milliseconds).
- The setInterval method will continue calling the function until clearInterval() is called or window is closed.
- The ID value returned by setInterval() is used as the parameter for the clearInterval() method.

JS FirstTimer.js U

Javascript > Lecture 30 > JS FirstTimer.js > ...

```
1   // node FirstTimer.js --n=10 --d=500
2
3   let minimist = require("minimist");
4   let args = minimist(process.argv);
5
6   let count = args.n;
7   let id = setInterval(function(){
8       console.log(count + " time-units to go.")
9       count--;
10
11      if(count == 0){
12          clearInterval(id);
13          console.log("Timeout.")
14      }
15  },args.d)
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
→  Lecture 30 git:(main) ✗ node FirstTimer --n=10 --d=500
10 time-units to go.
9 time-units to go.
8 time-units to go.
7 time-units to go.
6 time-units to go.
5 time-units to go.
4 time-units to go.
3 time-units to go.
2 time-units to go.
1 time-units to go.
Timeout.
→  Lecture 30 git:(main) ✗
```

JS FirstTimer.js U

Javascript > Lecture 30 > JS FirstTimer.js > ...

```
1   // node FirstTimer.js --n=10 --d=500
2
3   let minimist = require("minimist");
4   let args = minimist(process.argv);
5
6   let count = args.n;
7   let time = args.d;
8   let id = setInterval(function(){
9       console.log(count + " time-units to go.")
10      count--;
11
12      if(count == 0){
13
14          console.log("Timeout.");
15          clearInterval(id);
16      }
17  },time);
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
→  Lecture 30 git:(main) ✗ node FirstTimer --n=10 --d=500
10 time-units to go.
9 time-units to go.
8 time-units to go.
7 time-units to go.
6 time-units to go.
5 time-units to go.
4 time-units to go.
3 time-units to go.
2 time-units to go.
1 time-units to go.
Timeout.
→  Lecture 30 git:(main) ✗
```