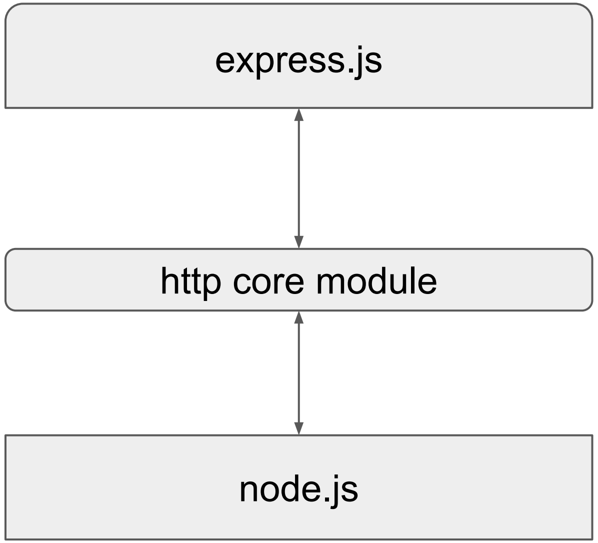
Tutorial @2

Node.js & Express web framework & Pug template engine

>>> Pug@1 :: recall express framework

Express is a tool for building web applications that run on node.js.

It runs on node.js and uses http core modules functionality behind the scene.



>>> Pug@2 :: express framework example

1.Create a project folder

2.Inside this folder create

package.json file with npm init

3.Download express using

npm install express --save

3.Create your main file. i.e. app.js

4.In your main file write the code that is shown in the box

5.Save and run main file with node <filename>

6. Go to your browser and check whether your express server is working

go to this url: 127.0.0.1:3000

app.js

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


app.get('/', function (req,res) {
      res.send('<h1>Hello World!</h1>')
})


app.listen(3000, function () {
      console.log('Server is running on port 3000')
})
```

terminal

```
> node app.js
> Server is running on port 3000
```

>>> Pug@3 :: briefly discuss the previous example

In this example we are using request handler function and sending response to the client that requested the data from root url of the website.
We are sending only data enclosed in <h1> tag which is not enough for well designed websites.
We usually use separate .html files for defining structure of the website, but these html files are static and they cannot incorporate additional data dynamically.
So we need tool which can grab data from the application layer, generate static .html file with the very data and send it to the client.
In this tutorial we will use tool called Pug for this purpose.

```
var express = require(‘express’)
var app = express()


app.get(‘/’, function (req,res) {
      res.send(‘<h1>Hello World!</h1>’)
})


app.listen(3000)
```

>>> Pug@4 :: Pug introduction

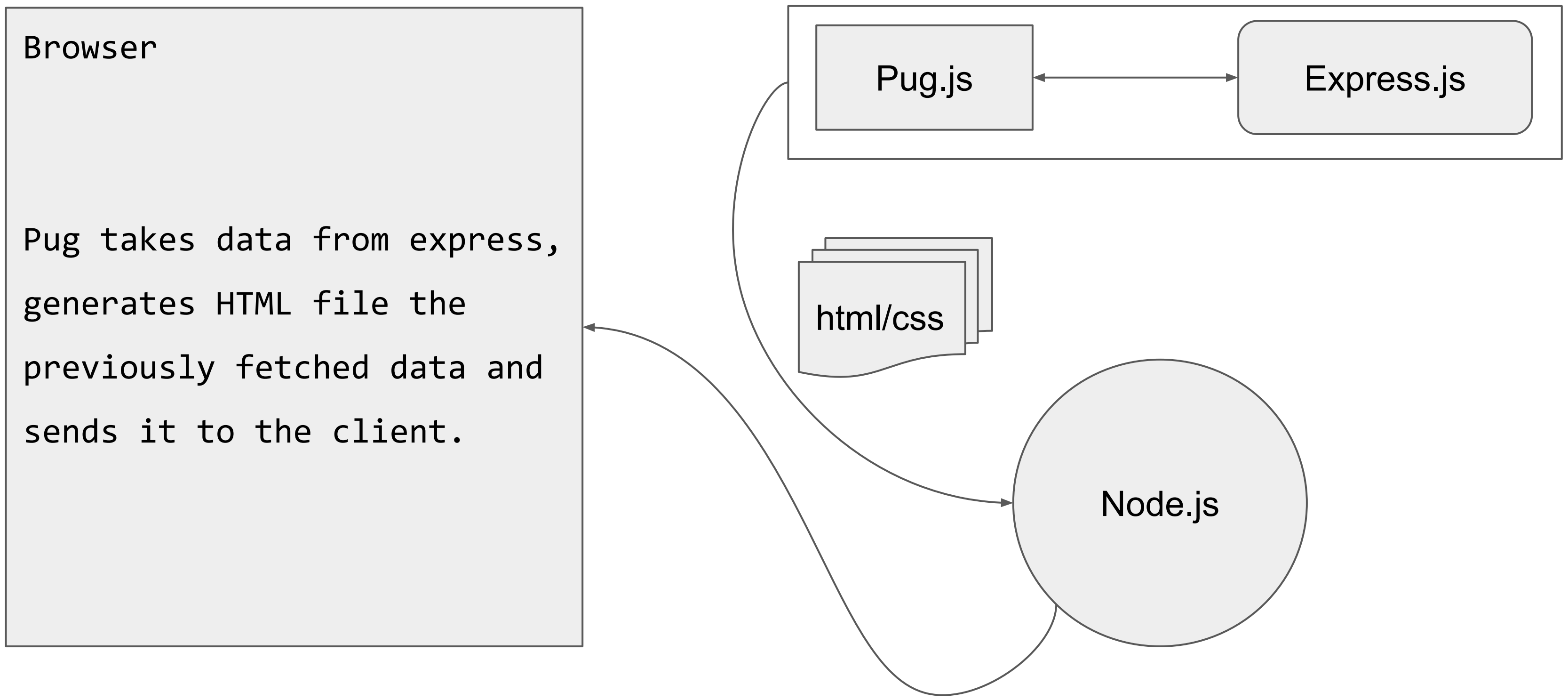
To understand Pug, you need to remember that the browser reads HTML and CSS and then displays formatted images and text to the client based on what that HTML and CSS tells it to do.

Pug is the middleman. **Pug** is a *template engine* for **Node.js**.

A *template engine* allows us to inject data and then produce HTML.

In short: **At run time, Pug (and other template engines) replace variables in our file with actual values, and then send the resulting HTML string to the client.**

>>> Pug@5 :: pug introduction



Pug takes data from express, generates HTML file the previously fetched data and sends it to the client.

>>> Pug@6 :: pug and html > same code, different languages

Pug code is written in the .pug file.

```
html
  head
     title My website
  body
    main.container
       h1 Homepage
```

and then converted to html code.

```
<html>
     <head>
          <title>My website</title>
     </head>

     <body>
          <main class=”container”>
               <h1>Homepage</h1>
          </main>
     </body>
</html>
```

>>> Pug@7 :: pug is good and bad

Good

When you write with Pug, you write code that looks like paragraphs. This greatly improves code-readability and streamlines projects with multiple developers.

There are no closing tags with Pug. Pug makes use of indentation to determine the nesting of tags. There are even shorthands for classes (.) and IDs (#)

Most importantly, we can write JavaScript that actually (almost/kind of) looks like JavaScript within our pug files.

Bad

With Pug, white-space matters. And it matters big time. The slightest mistake in your formatting/indenting/spacing means big problems for your code.

With Pug, you can’t copy HTML from anywhere, you have to convert everything to Pug before you can use it.

>>> Pug@8 :: how to install pug and use it in express

1. Use npm install pug --save command to install package to the project
2. Inside main file indicate that you want to use <pug> as the template engine
3. Create in the root folder special folder views to keep your .pug templates there. It is required.
4. Render .pug files using render function of express. Check next slide for code example.

project structure

```
root/
    app.js
    package.json
    package-lock.json
    views/
        index.pug
    node_modules
```

>>> Pug@9 :: how to install pug and use it in express

app.js

```
var express = require(‘express’)
var app = express()

app.set(‘view engine’, ‘pug’)

app.get(‘/’, function (req, res){
     res.render(‘index’, {name: ‘John Doe’})
})

app.listen(3000) <- can be any port number
```

index.pug

```
html
  head
    title My website
  body
    h1= name
```

>>> Pug@10 :: how to install pug and use it in express

```
app.set(‘view engine’, ‘pug’)

app.get(‘/’, function (req, res){
      res.render(‘index’, {name: ‘John Doe’})
})
```

This function is sets the pug as a main template engine for express. It is built-in function and you just have to follow the requirements.

This function sends the response to the requested url to the client. The (1st) argument of this function is the .pug file name that must be rendered and the (2nd) argument is the data we want to dynamically pass to the pug engine so it will generate .html file with the very data. The data is old plain javascript object. It is also called context

>>> Pug@11 :: how to install pug and use it in express

```
html
  head
    title My website
  body
    h1= name
```

This is simple .pug file with just several tags. It looks like .html but without opening and closing tags. See previous slides for more info.

In this part of .pug file we are 'saying' that h1 tags must have the value coming from the express.

So, if we want to assign a value to some tag we use the shown example.

If we want hardcode the data, we don't need equal sign. We just use space instead.

>>> Pug@12 :: Practice

1. Create project folder
2. Inside initiate a project/create package.json file <npm init>
3. Create main file <app.js>
4. Install express and pug packages <npm install express pug --save>
5. Create folder <views> inside project folder
6. Create <index.pug> file inside <views> folder
7. Define simple web page structure inside <index.pug>
8. Serve this file using <app.render()> function