DORS/CLUC 2022

Design of the Node.js module according to Unix principles – *tinyHTMLizer*

Kruno Peter

Research and Teaching Department

Andrija Stampar Teaching Institute of Public Health

individual developer

resource scarcity: time, knowledge, experience

approach to complexity of software development

*divide et impera** – the principle of decomposition

** The technique of mastering complexity has been known since ancient times: 'divide et impera' (divide and rule).* (Dijkstra 1965)

the Unix philosophy* (1969.)

- *Make it simple.*
- *Make each program do one thing well.*

* Cooke, D, Urban, J & Hamilton, S 1999, 'Unix and Beyond: An Interview with Ken Thompson', Computer, May 1999, pp. 58-64

Campbell-Kelly, M, Aspray, W 1996, 'Computer: A History of the Information Machine', New York: BasicBooks.

Wikipedia 2022, 'Unix philosophy', viewed 14 April 2022, <https://en.wikipedia.org/wiki/Unix_philosophy>

the Node.js philosophy* (2009.)

- a small core
- an ecosystem of small modules

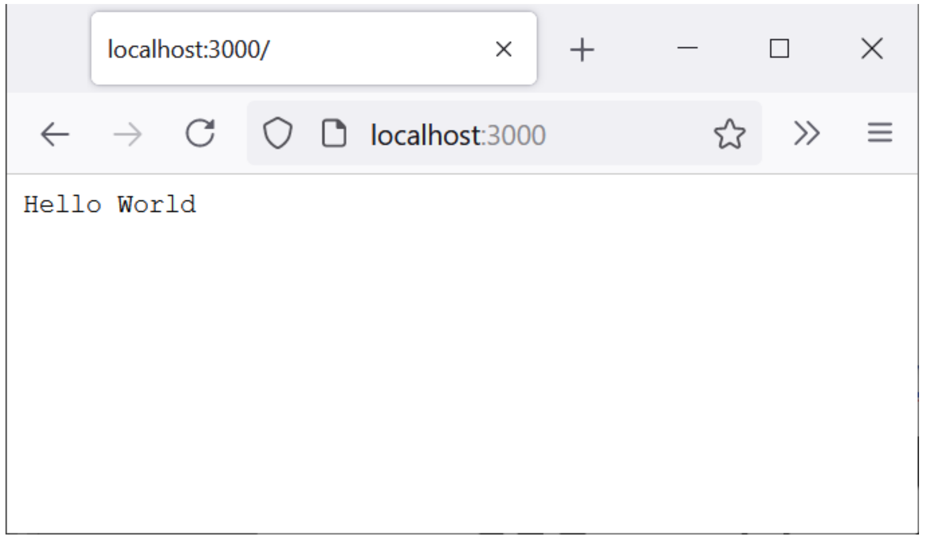
* Evrone, 'Interview with Ryan Dahl, Creator of Node.js', viewed 18 May 2022, <https://evrone.com/ryan-dahl-interview>

Casciaro, M, Mammino, L 2016, 'Node.js Design Patterns, Second Edition', Birmingham: Packt Publishing.

Hughes-Croucher, T, Wilson, M 2012, 'Node: Up and Running: Scalable Server-Side Code with JavaScript', Sebastopol: O'Reilly Media.

```
// a simple Web server - from https://nodejs.org/en/about/
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

[ ]



```
Hello World
```

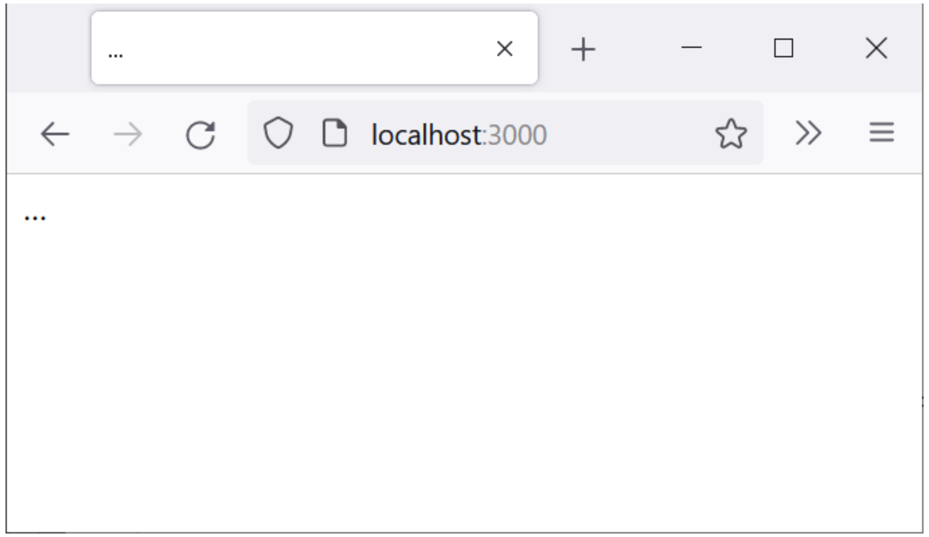
[an upgrade: generating HTML code (but not only <p>*Hello!*</p>) –

a component: a simple interface and structure, and no dependencies]

a simple HTML code generator

- a clean function packed in a module
- all its arguments are optional
- the function htmlize(content, title, style, script) – content and title are '...' by default
- a module without dependencies – tinyHTMLizer
- GitHub: https://github.com/kruno-peter/tinyHTMLizer

```
// a simple Web server - returns an (almost) empty Web page
const http = require('http');
const tinyHTMLizer = require('./tinyHTMLizer');    // shorter: th

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/html');    // instead of text/plain
  res.end(tinyHTMLizer.htmlize());    // htmlize! (without any args)
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```



Grammar

```
<webPage> ::= <docType> <html1> <head> <body> <html2>
<head> ::= <head1> <charset> <titled> [<styled>] [<scripted>] <head2>
<titled> ::= <title1> <title> <title2>
<styled> ::= <style1> <style> <style2>
<scripted> ::= <script1> <script> <script2>
<body> ::= <body1>  <content> <body2>

<docType> ::= '<!doctype html>'
<html1> ::= '<html lang="en">'
<html2> ::= '</html>'
<head1> ::= '<head>'
<head2> ::= '</head>'
<charset> ::= '<meta charset="utf-8">'
<title1> ::= '<title>'
<title2> ::= '</title>'
<style1> ::= '<style>'
<style2> ::= '</style>'
<script1> ::= '<script>'
<script2> ::= '</script>'
<body1> ::= '<body>'
<body2> ::= '</body>'

<title> ::= '...'
<content> ::= '...'
```

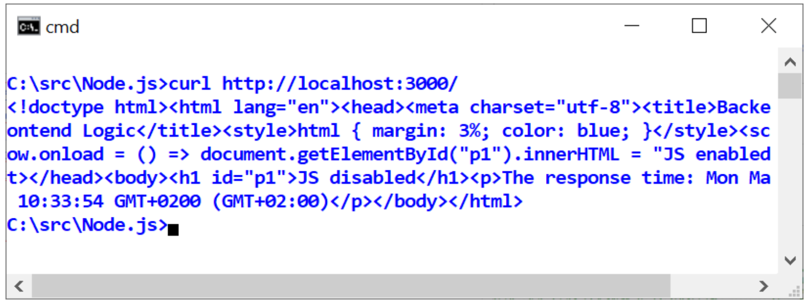
Web page code*

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>...</title>
</head>
<body>
...
</body>
</html>
```

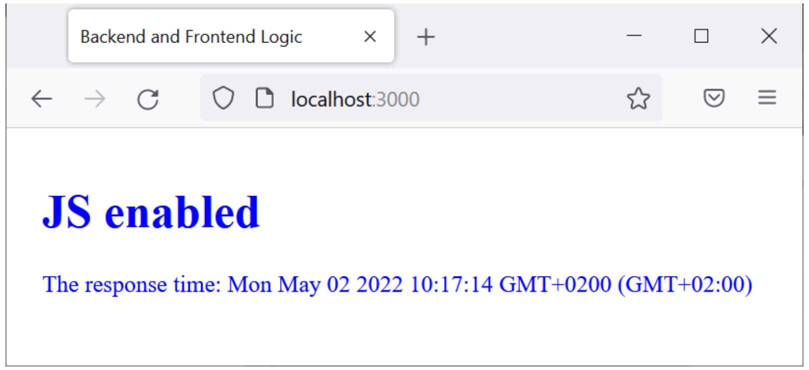
unit testing: <title>, <content> => <webPage>

* HTML code validated by using W3C Markup Validator on 15 April 2022

```
// a simple HTMLized Web app:
// it shows the server response time and tests whether JavaScript is enabled
const http = require('http');
const tinyHTMLizer = require('./tinyHTMLizer');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  const content = '<h1 id="p1">JS disabled</h1><p>The response time: ' + new Date() + '</p>';
  const title = 'Backend and Frontend Logic';
  const style = 'html { margin: 3\%; color: blue; }';   // watch out! backslash%
  const script = 'window.onload = () => document.getElementById("p1").innerHTML = "JS enabled";';
  const webPage = tinyHTMLizer.htmlize(content, title, style, script);

  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/html');
  res.end(webPage);
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```



```
C:\src\Node.js>curl http://localhost:3000/
<!doctype html><html lang="en"><head><meta charset="utf-8"><title>Backe
ontend Logic</title><style>html { margin: 3%; color: blue; }</style><sc
ow.onload = () => document.getElementById("p1").innerHTML = "JS enabled
t></head><body><h1 id="p1">JS disabled</h1><p>The response time: Mon Ma
 10:33:54 GMT+0200 (GMT+02:00)</p></body></html>
C:\src\Node.js>
```



JS enabled

The response time: Mon May 02 2022 10:17:14 GMT+0200 (GMT+02:00)

Grammar

```
<webPage> ::= <docType> <html1> <head> <body> <html2>
<head> ::= <head1> <charset> <titled> [<styled>] [<scripted>] <head2>
<titled> ::= <title1> <title> <title2>
<styled> ::= <style1> <style> <style2>
<scripted> ::= <script1> <script> <script2>
<body> ::= <body1>  <content> <body2>

<docType> ::= '<!doctype html>'
<html1> ::= '<html lang="en">'
<html2> ::= '</html>'
<head1> ::= '<head>'
<head2> ::= '</head>'
<charset> ::= '<meta charset="utf-8">'
<title1> ::= '<title>'
<title2> ::= '</title>'
<style1> ::= '<style>'
<style2> ::= '</style>'
<script1> ::= '<script>'
<script2> ::= '</script>'
<body1> ::= '<body>'
<body2> ::= '</body>'

<title> ::= 'Backend and Frontend Logic'
<style> ::= 'html { margin: 3%; color: blue; }'
<script> ::= 'window.onload = () => document.getElementById("p1").innerHTML = "JS enabled";'
<content> ::= '<h1 id="p1">JS disabled</h1><p>The response time: Mon May 02 2022 10:17:14 GMT+0200 (GMT+02:00)</p>'
```

unit testing: <title>, <style>, <script>, <content> => <webPage>

Web page code*

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Backend and Frontend Logic</title>
  <style>
    html { margin: 3%; color: blue; }
  </style>
  <script>
    window.onload = () => document.getElementById("p1").innerHTML = "JS enabled";
  </script>
</head>
<body>
  <h1 id="p1">JS disabled</h1>
  <p>The response time: Mon May 02 2022 10:17:14 GMT+0200 (GMT+02:00)</p>
</body>
</html>
```

* HTML code validated by using W3C Markup Validator on 2 May 2022

```
// tinyHTMLizer.js - a Node.js module - a simple HTML generator
// htmlize(content, title, style, script) - returns a simple Web page (en)
// all arguments are optional ('content' and 'title' are dots by default)
// tip: incrementally add arguments and test the application

exports.htmlize = (content = '...', title = '...', style, script) => {

  // HTML tags
  const docType = '<!doctype html>';
  const html1 = '<html lang="en">';
  const html2 = '</html>';
  const head1 = '<head>';
  const head2 = '</head>';
  const charset = '<meta charset="utf-8">';
  const title1 = '<title>';
  const title2 = '</title>';
  const style1 = '<style>';
  const style2 = '</style>';
  const script1 = '<script>';
  const script2 = '</script>';
  const body1 = '<body>';
  const body2 = '</body>';

  // processing arguments - omitting unnecessary
  let styled = style1 + style + style2;
  if (style == undefined) {          // == works fine
    styled = '';
  }

  let scripted = script1 + script + script2;
  if (script == undefined) {
    scripted = '';
  }

  // assembling the webpage - concatenation
  let head = head1 + charset + title1 + title + title2 + styled + scripted + head2;
  let body = body1 + content + body2;
  let webPage = docType + html1 + head + body + html2;

  return webPage;

};
```

discussion

- there is not a universal approach to software development* – other problems and possibilities arise from the solution
- a minimalistic htmlize() with two arguments – content and title
- move htmlize() from the module to the app
- htmlize() suitable for simple apps (prototyping, education)
- using a template to avoid mixing HTML, CSS and JavaScript code

* Brooks, F 1995 (1986), 'No Silver Bullet – Essence and Accident in Software Engineering', The Mythical Man-Month, Addison-Wesley

conclusion ✓

- designing simple modules to avoid complexity
- process: iterating and incrementing in short cycles