Front End React Development
sesi 8

Document Object Model
- DOM -

Document Object Model - DOM

Dalam mengembangkan web, kita harus menyadari bahwa kita sekaligus membuat Document Object Model (DOM) yang tersusun dalam dokumen HTML. Dengan pengetahuan DOM, kita bisa secara lebih lengkap mengetahui dan mampu untuk membuat interaksi pada halaman web menggunakan JavaScript.

DOM API di HTML umumnya adalah untuk node ataupun objek element, document, dan window. Setiap hal tersebut memiliki berbagai property (nilai) dan method (aksi), bahkan bisa juga dipasang sebuah penangan kejadian (event handler) sehingga jika ada kejadian tertentu dilakukan suatu statement akan dijalankan.

- window: frame, parent, self, top
- history
- location
- document
- element: body, h1, p, button, dll

Document Object Model - DOM

Mulai dari bagian ini, snippet atau potongan kode ini akan berlanjut/bersambung. Sebelum mencoba melakukan seleksi dan manipulasi, kita coba asumsikan penggunaan Kita bisa melakukan seleksi terhadap DOM dengan mengunakan beberapa sintaks berikut:

```
1  <html>
2    <head>
3      <title>Contoh Webpage Standard</title>
4    </head>
5    <body>
6      <div id="page-title">Sample Page Title</div>
7      <h1>Test Sample Heading</h1>
8      <div class="page-box">Page Box 1</div>
9      <div class="page-box">Page Box 2</div>
10     <div class="page-box">Page Box 3</div>
11     <script src="js-simple-dom-script.js"></script>
12   </body>
13 </html>
```

HACKTIV8

Document Object Model - DOM

```
1 var pageTitleElement =
document.getElementById("page-title");

2 // Menyeleksi DOM berdasarkan Id element dan
menampungnya ke dalam variabel. Isinya merupakan object
HTML element

3

4 var pageBoxElements =
document.getElementsByClassName("page-box");

5 // Menyeleksi DOM berdasarkan nama class element dan
menampungnya ke dalam variabel. Isinya merupakan array
dari object HTML element, walau <h1> hanya ada 1.

6

7 var pageHeadings =
document.getElementsByTagName("h1");

8 // Menyeleksi DOM berdasarkan tag <h1> dan
menampungnya ke dalam variabel. Isinya merupakan array
dari object HTML element
```

Mengakses isi HTML dari DOM

```
1 var pageTitleElement =
document.getElementById("page-title");

2 // Menyeleksi DOM berdasarkan Id element dan
menampungnya ke dalam variabel. Isinya merupakan object
HTML element

3

4 var pageBoxElements =
document.getElementsByClassName("page-box");

5 // Menyeleksi DOM berdasarkan nama class element dan
menampungnya ke dalam variabel. Isinya merupakan array
dari object HTML element, walau <h1> hanya ada 1.

6

7 var pageHeadings =
document.getElementsByTagName("h1");

8 // Menyeleksi DOM berdasarkan tag <h1> dan
menampungnya ke dalam variabel. Isinya merupakan array
dari object HTML element
```

```
9

10 var pageTitleElementsContent =
pageTitleElement.innerHTML;

11 console.log('isi <div id="page-title"> :' +
pageTitleElementsContent);

12 // isi <div id="page-title"> adalah Sample Page Title

13

14 var pageBoxElementsContent =
pageBoxElements.innerHTML;

15 console.log('isi <div class="page-box"> :' +
pageBoxElementsContent);

16 // isi <div class="page-box"> adalah undefined!
```

Mengakses isi HTML dari DOM

```
1 var pageTitleElement =
document.getElementById("page-title");

2 // Menyeleksi DOM berdasarkan Id element dan
menampungnya ke dalam variabel. Isinya merupakan object
HTML element

3

4 var pageBoxElements =
document.getElementsByClassName("page-box");

5 // Menyeleksi DOM berdasarkan nama class element dan
menampungnya ke dalam variabel. Isinya merupakan array
dari object HTML element, walau <h1> hanya ada 1.

6

7 var pageHeadings =
document.getElementsByTagName("h1");

8 // Menyeleksi DOM berdasarkan tag <h1> dan
menampungnya ke dalam variabel. Isinya merupakan array
dari object HTML element
```

```
9

10 var pageTitleElementContent =
pageTitleElement.innerHTML;

11 console.log('isi <div id="page-title"> :' +
pageTitleElementContent);

12 // isi <div id="page-title"> adalah Sample Page Title

// Meloop array pageBoxElements

13

14 for(var i = 0; i < pageBoxElements.length; i++) {

15 var currentPageBoxElement          =
pageBoxElements[i];

16 var currentPageBoxElementContent  =
currentPageBoxElement.innerHTML;

17
```

HACKTIV8

Mengakses isi HTML dari DOM

```
18 // Mengambil isi elemen pageBoxElements yang kedua,
yaitu index ke 1

19 var secondPageBoxElement          =
pageBoxElements[1];

20 var secondpageBoxElementContent  =
secondPageBoxElement.innerHTML;

21

22 // Mengambil isi elemen pageBoxElements yang ketiga,
yaitu index ke 2

23 var thirdPageBoxElement          = pageBoxElements[2];

24 var thirdpageBoxElementContent  =
thirdPageBoxElement.innerHTML;
```

```
25

26 // Menampilkan isi elemen dengan console.log

27 console.log('isi <div class="page-box"> yang
pertama:' + firstpageBoxElementContent);

28 console.log('isi <div class="page-box"> yang kedua:'
+ secondpageBoxElementContent);

29 console.log('isi <div class="page-box"> yang ketiga:'
+ thirdpageBoxElementContent);
```

DOM Transversing

Apa itu DOM Transversing?

Di layout HTML yang cukup kompleks, kita akan bertemu dengan banyak element HTML yang memiliki hubungan parent-child yang dalam, dan pada saat kita menggunakan JavaScript untuk menseleksi atau memanipulasinya, tidak mungkin kita harus memberikan id atau class ke semua element-nya. Kita bisa menseleksi element HTML dari parent atau dari childnya. Untuk lebih dalam memahami hal ini, kamu harus telah mengerti hierarki parent-child yang terjadi di susunan HTML. Tapi tenang saja, kita akan mengulas ulang hal tersebut.

DOM Transversing

Apa itu DOM Transversing?

Di layout HTML yang cukup kompleks, kita akan bertemu dengan banyak element HTML yang memiliki hubungan parent-child yang dalam, dan pada saat kita menggunakan JavaScript untuk menseleksi atau memanipulasinya, tidak mungkin kita harus memberikan id atau class ke semua element-nya. Kita bisa menseleksi element HTML dari parent atau dari childnya. Untuk lebih dalam memahami hal ini, kamu harus telah mengerti hierarki parent-child yang terjadi di susunan HTML. Tapi tenang saja, kita akan mengulas ulang hal tersebut.

```html
<html>
  <head></head>
  <body>
    <h1></h1>
    <div id="contoh-div-1">
      <p id="contoh-p-1">
        <strong></strong>
        <em></em>
      </p>
    </div>
    <div id="contoh-div-2">
      <h2></h2>
      <p id="contoh-p-2"></p>
      <ul>
```

```
      <li></li>
      <li></li>
    </ul>
  </div>
  <script src="dom-transverse-1-intro.js"></script>
  <script
src="dom-transverse-2-siblings.js"></script>
  <script
src="dom-transverse-3-chaining-selectors.js"></script>
</body>
</html>
```

HACKTIV8

DOM Transversing

html : merupakan parent paling atas

head : merupakan child dari html body : merupakan child dari html, sibling dari head h1 : merupakan child dari body

div id="contoh-div-1" : merupakan child dari body, sibling dari <h1>

p id="contoh-p-1" : merupakan child dari div id="contoh-div-1"

strong : merupakan child dari p id="contoh-p-1"

em : merupakan child dari p id="contoh-p-1", sibling dari strong

div id="contoh-div-2" : merupakan child dari body, sibling dari h1 dan div id="contoh-div-1"

h2 : merupakan child dari div id="contoh-div-2" p id="contoh-p-2" : merupakan child dari div id="contoh-div-2", sibling dari h2

ul : merupakan child dari div id="contoh-div-2", sibling dari h2 dan p id="contoh-p-2"

li : merupakan child dari ul

Parent - Child

Untuk mulai mengenai transerving atau penjelajahan di dalam DOM, kita coba mulai dengan menjelajahi hubungan Parent - Child. Contoh pertama kita mulai dengan menseleksi <body> dan mendapatkan element HTML apa saja yang menjadi children dari <body>.

```
/*
=====================================================
Menseleksi DOM berdasarkan hubungan Parent - Child
=====================================================
*/

```

```
// Menseleksi element <body>
var body = document.body;

// Mendapatkan element children dari <body>
var bodyChilds = body.children;

// Menampilkan DOM yang menjadi child dari <body>
dalam bentuk array
console.log(bodyChilds); // h1, div
id="contoh-div-1", div id="contoh-div-2", scripts js
```

```
/*
=====================================================
Menseleksi DOM berdasarkan hubungan Parent - Child
=====================================================
*/

// Menseleksi element <body>
var body = document.body;

// Mendapatkan element children dari <body>
var bodyChilds = body.children;

// Menampilkan DOM yang menjadi child dari <body>
```

```
console.log(bodyChilds);

// Menseleksi element <div id="contoh-div-1">
var contohDiv1 =
document.getElementById('contoh-div-1');

// Mendapatkan element children dari <div
id="contoh-div-1"> dalam bentuk array
var contohDiv1Childs = contohDiv1.children;
```

```

// Mendapatkan element children dari <div
id="contoh-div-1"> dalam bentuk array
var contohDiv1Childs = contohDiv1.children;
console.log(contohDiv1FirstChild); // <p
id="contoh-p-1">...</p>

// Note: Walaupun children mungkin hanya 1 element,
tetap tertampung dalam array!
```

HACKTIV8

Siblings

Apabila sebelumnya kita mempelajari hubungan DOM sebagai parent dan child, sekarang kita akan membahas tentang hubungan antar sibling. Sibling, layaknya saudara kandung dalam analogi keluarga, merupakan DOM yang merupakan child dari parent yang sama.

```
/*
==========================================
Menseleksi DOM berdasarkan hubungan Sibling
==========================================
*/

// Menseleksi element <div id="contoh-div-1">
var contohDiv1 =
document.getElementById('contoh-div-1');

// Mendapatkan sibling setelah <div
id="contoh-div-1">
var contohDiv1NextSibling =
contohDiv1.nextElementSibling;

console.log(contohDiv1NextSibling); // <div
id="contoh-div-2">...</div>
```

```
/*
==========================================
Menseleksi DOM berdasarkan hubungan Sibling
==========================================
*/

// Menseleksi element <div id="contoh-div-1">
var contohDiv1 =
document.getElementById('contoh-div-1');

// Mendapatkan sibling setelah <div
id="contoh-div-1">
var contohDiv1NextSibling =
contohDiv1.nextElementSibling;

console.log(contohDiv1NextSibling); // <div
id="contoh-div-2">...</div>

// Mendapatkan sibling sebelum <div
id="contoh-div-1">
var contohDiv1PrevSibling =
contohDiv1.previousElementSibling;

console.log(contohDiv1PrevSibling); // <h1></h1>
```

HACKTIV8

Siblings

Apabila sebelumnya kita mempelajari hubungan DOM sebagai parent dan child, sekarang kita akan membahas tentang hubungan antar sibling. Sibling, layaknya saudara kandung dalam analogi keluarga, merupakan DOM yang merupakan child dari parent yang sama.

```
/*
==========================================
Menseleksi DOM berdasarkan hubungan Sibling
==========================================
*/

// Menseleksi element <div id="contoh-div-1">
var contohDiv1 =
document.getElementById('contoh-div-1');

// Mendapatkan sibling setelah <div
id="contoh-div-1">
var contohDiv1NextSibling =
contohDiv1.nextElementSibling;

console.log(contohDiv1NextSibling); // <div
id="contoh-div-2">...</div>
```

```
/*
==========================================
Menseleksi DOM berdasarkan hubungan Sibling
==========================================
*/

// Menseleksi element <div id="contoh-div-1">
var contohDiv1 =
document.getElementById('contoh-div-1');

// Mendapatkan sibling setelah <div
id="contoh-div-1">
var contohDiv1NextSibling =
contohDiv1.nextElementSibling;

console.log(contohDiv1NextSibling); // <div
id="contoh-div-2">...</div>

// Mendapatkan sibling sebelum <div
id="contoh-div-1">
var contohDiv1PrevSibling =
contohDiv1.previousElementSibling;

console.log(contohDiv1PrevSibling); // <h1></h1>
```

HACKTIV8

DOM Creation
Apa itu DOM Creation?

Dalam aplikasi web, kita tidak terbatas memanfaatkan DOM untuk dimanipulasi atau ditambahkan event-event tertentu. Kita juga dapat membuat DOM baru menggunakan JavaScript. Bahkan, kita dapat membuat sebuah halaman web yang full dengan element HTML namun script HTML kita sangt sedikit, dan hampir seluruhnya di render menggunakan JavaScript. Prinsip inilah yang dilakukan oleh Angular, React, dan teknologi front-end lainnya untuk merender DOM.

```html
<html>
  <head>
    <title>Super Simple Web Page</title>
  </head>
  <body>
    <div id="main">
      <div id="inside-main">
        <h1>Heading Sample 1</h1>
        <button>Click Me!</button>
      </div>
    </div>
  </body>
</html>
```

```html
<html>
<head>
  <title>Super Simple Web Page - Fresh From the JS</title>
</head>
<body>
  <!-- it's empty, just contain a js file -->
  <script src="js-dom-creation-script.js"></script>
</body>
</html>
```

```js
// Pertama, kita seleksi terlebih dahulu <body>
var body = document.body;

// Kemudian, kita buat sebuah element HTML <div> menggunakan createElement
var mainDiv = document.createElement('div');

// Untuk membuat <div id="main">, maka kita harus membuat HTML attribute id
var mainDivAttrId = document.createAttribute('id');

// Untuk memberikan nilai kepada id, maka kita gunakan .value
mainDivAttrId.value = "main";

// id="main" kita sudah siap. Sekarang kita harus menambahkan attribute tersebut ke mainDiv
mainDiv.setAttributeNode(mainDivAttrId);

// mainDiv kita sudah menjadi <div id="main">. Saatnya kita tambahkan ke dalam <body>
// Karena Kita akan meletakkan <div id="main"> di dalam <body>, maka kita gunakan appendChild
body.appendChild(mainDiv);
```

HACKTIV8

REGEX

REGEX

Regular Expression atau sering disingkat RegExp atau RE adalah suatu mekanisme pencocokan pola (pattern matching) yang dibuat dengan menggunakan karakter-karakter khusus. Fungsinya sangat beragam, mulai dari memeriksa apakah sebuah inputan sudah sesuai atau belum (test), untuk membuat fitur pencarian (search) atau penggantian string (replace).

Contoh method yang menggunakan RegExp sudah kita lihat pada pembahasan tentang string object. Diantaranya method search(), match() dan replace(). Selain itu juga terdapat beberapa method yang khusus "melekat" ke RegExp Object javascript.

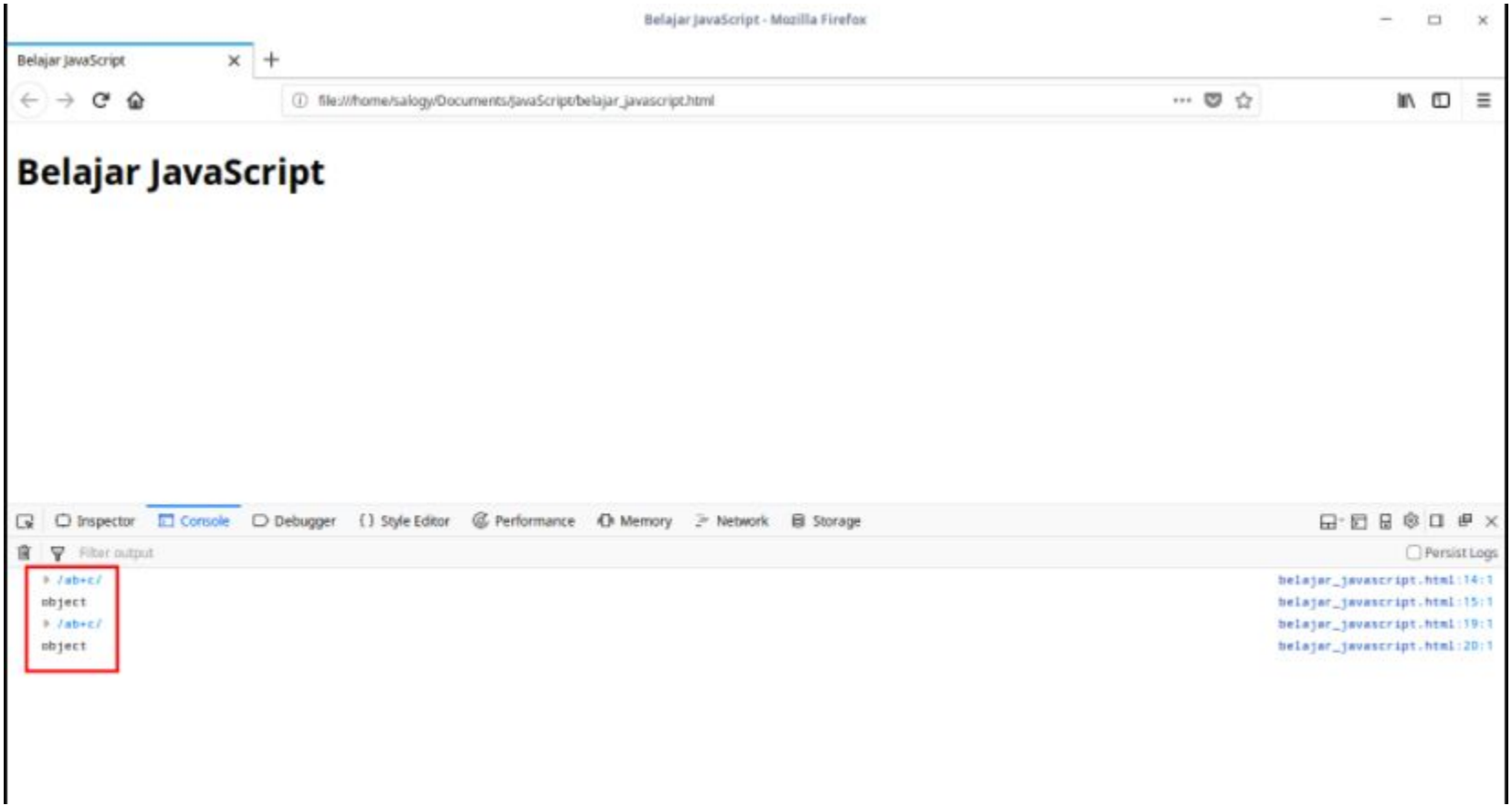
Penggunaan paling banyak dari RegExp adalah untuk proses validasi form. Sebagai contoh, bagaimana caranya kita memastikan seseorang sudah menginput alamat email dengan benar? Apakah yang diinput di kolom total pemesanan sudah berupa angka? atau malah huruf? Bagaimana cara memastikan inputan password yang syaratnya harus dibuat dari 6 karakter dan mengandung minimal 1 angka dan 1 huruf besar? Ini semua ditangani oleh RegExp.

Cara Membuat RegExp Object

Di dalam javscript, regular expression ditempatkan ke dalam object sendiri, yakni RegExp object. Sama seperti mayoritas object bawaan javascript lain, kita memiliki 2 cara penulisan menggunakan object constructor atau cara langsung (literal).

Berikut contoh pembuatan RegExp di dalam javascript:

```
1  var foo = /ab+c/;
2
3  console.log(foo);  // /ab+c/
4  console.log(typeof foo);  // object
5
6  var bar = new RegExp("ab+c");
7
8  console.log(bar);  ///ab+c/
9  console.log(typeof bar);  // object
```


Belajar JavaScript - Mozilla Firefox
Belajar JavaScript
file:///home/salogy/Documents/javaScript/belajar_javascript.html
Belajar JavaScript
Inspector
Console
Debugger
Style Editor
Performance
Memory
Network
Storage
Filter output
Persist Logs
/ab+c/
object
/ab+c/
object
belajar_javascript.html:14:1
belajar_javascript.html:15:1
belajar_javascript.html:19:1
belajar_javascript.html:20:1

HACKTIV8

Reg Exp Object Method

RegExp object memiliki beberapa method dan property. Sebagian besar dari method ini merupakan fitur lanjutan yang jarang dipakai. Kita hanya akan membahas 2 diantaranya: method test() dan exec().

Method RegExp.prototype.test()

Method test() digunakan untuk memeriksa apakah sebuah string lolos dari pola regular expression yang diinput. Jika lolos, hasilnya true. Jika tidak, hasilnya false.

Method ini mirip seperti include() dari String Object. Bedanya pada method test() pengecekan string menggunakan pola regular expression. Berikut contoh penggunaannya:

```
1  var foo = "Belajar JavaScript";
2  var pola = /JavaScript/;
3
4  console.log(pola.test(foo));  // true
5  console.log(/Belajar/.test(foo));  // true
6  console.log(/belajar/.test(foo));  // false
```

Method RegExp.prototype.exec()

Method exec() berfungsi untuk mencari karakter atau kata yang cocok dengan pola regular expression, kemudian menyimpan hasilnya ke dalam array.

Berikut contoh penggunaannya:

```
1 var foo = "1 jam sama dengan 60 menit, juga sama dengan 3600 detik";
2 var pola = /\d+/;
3
4 console.log(pola.exec(foo));  // Array ["1"]
```

Bagaimana cara mencari "semua" digit? Kita bisa menambahkan sebuah flag g atau penanda di dalam pola regular expression. Pola tersebut menjadi seperti ini:

```
var foo = "1 jam sama dengan 60 menit, juga sama dengan 3600 detik";
var pola = /\d+/g;

console.log(pola.exec(foo));  // Array ["1"]
console.log(pola.exec(foo));  // Array ["60"]
console.log(pola.exec(foo));  // Array ["3600"]
console.log(pola.exec(foo));  // null
```

Pola Reguler Expression

Mempelajari pola reguler expression bisa dibilang "gampang-gampang susah". Karakter penyusun pola regexp tidak begitu banyak, tapi hasil dari pola yang ditulis cukup susah untuk dibaca. Agar mudah dipahami, kita akan banyak menggunakan contoh kode program.

Pola RegExp Sebagai String

Mari kita mulai dari pola yang paling sederhana yakni jika isi regular expression berbentuk string biasa yang terdiri dari sebuah kata atau berberapa karakter. Jika ditulis seperti ini, pola tersebut akan cocok selama di dalam string terdapat kata tersebut (di posisi mana saja). Berikut contohnya:

```
1  var foo = "Belajar JavaScript";
2
3  console.log(/JavaScript/.test(foo));  //true
4  console.log(/JavaScript/.test(foo));  //false
5  console.log(/Belajar/.test(foo));  // true
6  console.log(/belajar/.test(foo));  // false
7  console.log(/Java/.test(foo));  // true
8  console.log(/ajar/.test(foo));  // true
```

Pola Character Set

Pola selanjutnya adalah character set, dimana kitaa bisa membuat syarat bahwa hanya karakter tertentu saja yang boleh ditulis. Ini dibuat menggunakan tanda kurung siku: [dan]. Hanya karakter yang ada di dalam tanda kurung ini saja yang akan memenuhi syarat.

```
var pola = /[abcde]../;

console.log(pola.test("abaa"));  // true
console.log(pola.test("fba"));  // false
console.log(pola.test("1dd"));  // false
console.log(pola.test("add"));  // true
console.log(pola.test(" dd"));  // false
console.log(pola.test("belajar"));  // true
```

Pola /[abcde]../ artinya, digit pertama hanya bisa diisi oleh salah satu dari huruf a, b, c, d atau e. Kemudian salah satu huruf tersebut diikuti setidaknya oleh 2 karakter lain (bebas mau karakter apa saja).

Dengan demikian, string yang memenuhi syarat ini adalah: "abaa", "add" dan "belajar". String "fba" menjadi false karena digit pertama diawali huruf f. String "1dd" juga false karena karakter pertama berupa angka, sedangkan string " dd" hasilnya false karena karakter pertama berisi spasi. Pola /[abcde]../ bisa juga ditulis menjadi: /[a-e]../.