Module 6

Lecture 4

Alternative Paradigms: Scripting Languages

Contents

Common characteristics,

- Different Problem domains for using scripting,
- Use of scripting in Web development server and clients side scripting,

- Innovative features of scripting languages –
  - Names and Scopes,
  - string and pattern manipulation,
  - data types,
  - object orientation.

Innovative feature: Pattern Matching

- Regular expressions (REs) are present in many scripting languages and related tools employ extended versions of the notation
- Regular Expression provides an ability to match a “string of text” in a very flexible and concise manner.

- A “string of text” can be further defined as a single character, word, sentence or particular pattern of characters.

[ ]: Matches any one of a set characters
[ ] with hyphen: Matches any one of a range characters
^: The pattern following it must occur at the beginning of each line ^ with
[ ] : The pattern must not contain any character in the set specified $:
The pattern preceding it must occur at the end of each line . (dot):
Matches any one character
\ (backslash): Ignores the special meaning of the character following
it *: zero or more occurrences of the previous character
(dot).*: Nothing or any numbers of characters.

Innovative feature: Pattern Matching with RE

- [ ] : Matches any one of a set characters

Ex1: $grep “New[abc]” filename

It specifies the search pattern as : Newa , Newb or Newc

• Use [ ] with hyphen: Matches any one of a range characters

Ex1: $grep "New[a-e]" filename

It specifies the search pattern as: Newa , Newb or Newc , Newd, Newe

• Use ^: The pattern following it must occur at the beginning of each line

Ex1: $grep "^san" filename

Search lines beginning with san. It specifies the search pattern as: sanjeev ,sanjay, sanrit , sanchit , sandeep etc.

• Use ^ with [ ]: The pattern must not contain any character in the set specified

Ex1: $grep "New[^a-c]" filename

It specifies the pattern containing the word "New" followed by any character other than an 'a','b', or 'c'

Pattern Matching in JavaScript

• Commonly used JavaScript's built-in methods for performing pattern-matching

Function	What it Does
exec()	Search for a match in a string. It returns an array of information or null on mismatch.
test()	Test whether a string matches a pattern. It returns true or false.
search()	Search for a match within a string. It returns the index of the first match, or -1 if not found.
replace()	Search for a match in a string, and replaces the matched substring with a replacement string.
match()	Search for a match in a string. It returns an array of information or null on mismatch.
split()	Splits up a string into an array of substrings using a regular expression.

Pattern Matching in JavaScript

- In JavaScript, regular expressions are represented by **RegExp** object •

var regex = /^Mr\./; // Literal syntax • **Example1:**

```
<script>
var regex = /ca[kf]e/;
var str = "He was eating cake in the cafe.";
if(regex.test(str)) { // Test the string against the regular expression alert("Match
    found!");
} else {
    alert("Match not found.");
}
</script>
```

• **Example2:**

```
var regex = /ca[kf]e/;
var matches = str.match(regex);

alert(matches.length); // Outputs: 2
```

Pattern Matching in JavaScript

let text = "Visit SFIT!"; let n =

• **Example3:**

text.search(" SFIT ");
document.write(n);

- **<u>Example4:</u>**

let result = text.replace("Microsoft", " SFIT
"); document.write(result);

- **<u>Example5:</u>**

let text = "The best things in life are
free"; let result = /life/.exec(text);
document.write(result);

let text = "Visit Microsoft!";

//returns the position of the match as 6:

Pattern Matching in JavaScript

- **Example6:**

```
<script>
let text = "The best things in life are free";
const pattern = /[aeiou]/g; //The "g" modifier specifies a global match.  let output =
"";
while((result = pattern.exec(text)) !== null) {
 output += result[0] + " " + pattern.lastIndex + "\n";
}
document.write(output);
</script>
```

Innovative feature: Data Types

- As we have seen, scripting languages don’t generally require the declaration of types for variables
- Most perform extensive run-time checks to make sure that values are never used in inappropriate ways
- Some languages (e.g., Scheme, Python, and Ruby) are relatively strict about this checking
- Perl and Tcl takes the position that programmers should check for the errors they care about.

- When the programmer wants to convert from one type to another, it must say so explicitly

Innovative feature: Data Types

- Numeric types have a bit more variation across languages, but emphasis is universally that the programmer shouldn't worry about the issue unless necessary.
- Some of these even store numbers as strings, so calculations may not always be what you expect, although most do a good job of auto-converting if needed.
- For composite types, a heavy emphasis is on mappings (also called dictionaries, hashes, or associated arrays).

- Generally, these are similar to arrays, but access time depends upon a hash funtion.
- Example of dictionary:

director = {}

director['Star Wars'] = 'George Lucas'

director['The Princess Bride'] = 'Rob Reiner'

print director['Star Wars']

JavaScript Data Types

- There are eight basic data types in JavaScript.

1. Number
   - let n = 123; n = 12.345;
2. BigInt: larger numbers, larger than ±(2^{53}-1)
   - const bigInt = 1234567890123456789012345678901234567890n;

3. String

– let str = "Hello"; let str2 = ‘Single quotes’

4. Boolean

– let x = true; let y = false;

5. The “null” value

– let name = null;

6. The “undefined” value

– let age; alert(age); // shows "undefined"

JavaScript Data Types

7. The typeof operator

• returns the type of the // "number" // "bigint“
argument typeof 0;

typeof 10n;
typeof Symbol("id");

typeof Math; // "symbol“ // "object"

8. Objects and Symbols

- Objects are used to store keyed collections of various data and more complex entities. E.g.

```
const person = {firstName:"John", age:50, eyeColor:"blue"};
const person = {
  firstName: "John",
  age: 50,
  eyeColor: "blue“ };
```

JavaScript Data Types

- Symbols are immutable (cannot be changed) and are unique. – const value1 = Symbol('hello’);
  – const value2 = Symbol('hello’);
  – console.log(value1 === value2); // false
  – Though value1 and value2 both contain

the same description, different.

– You can add symbols as a key in an object using square brackets []. let id = Symbol("id");

let person = {

they are

name: "Jack",

123 };

// adding symbol as a key

[id]: 123 // not "id":

console.log(person); // {name: "Jack", Symbol(id): 123}

Innovative feature: Object Orientation

- Perl-5 has features that allow one to program in an object-oriented style. It uses a value model for variables; objects are always accessed via pointers.
- PHP and JavaScript have cleaner, more conventional-looking object oriented features. In PHP and JavaScript, a variable can hold either a value of a primitive type or a reference to an object of composite type.
- Both allow the programmer to use a more traditional imperative style

Innovative feature: Object Orientation

- Python and Ruby are explicitly and uniformly object-oriented and use a uniform reference model
- Classes are themselves objects in Python and Ruby, much as they are in Smalltalk
- Classes are types in PHP, much as they are in C++, Java, or C# • Classes in Perl are simply an alternative way of looking at packages (namespaces)
- JavaScript, remarkably, has objects but no classes
- Both PHP and JavaScript are more explicitly object oriented

Objects in JavaScript

- JavaScript is an object-based language. Everything is an object in JavaScript.
- JavaScript is template based not class based.
- There are 3 ways to create objects.
  - By object literal
  - By creating instance of Object directly (using new keyword)
  - By using an object constructor (using new keyword)

•JavaScript Object by object literal

object={property1:value1,property2:value2.....propertyN:valueN

} • **Example1:**

```
<script>
emp={id:102,name:"Shyam Kumar",salary:40000}
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
```

Objects in JavaScript

•JavaScript Object by creating instance of an object

```
var objectname=new Object();
```

• **Example2:**

```
<script>
var emp=new Object();
emp.id=101;
emp.name="Ravi Malik";
emp.salary=50000;
```

//**new** keyword is used to create object.

```
document.write(emp.id+" "+emp.name+"
"+emp.salary); </script>
```

Objects in JavaScript

•JavaScript Object by using an object constructor

```
var objectname=new Object();
```

• **Example3:**

```
<script>
```

```
function emp(id,name,salary){
   this.id=id;
   this.name=name;
   this.salary=salary;
 }
// constructor
// this keyword refers to current
object

e=new emp(103,"Vimal Jaiswal",30000);
document.write(e.id+" "+e.name+"
"+e.salary); </script>
```

Thank You