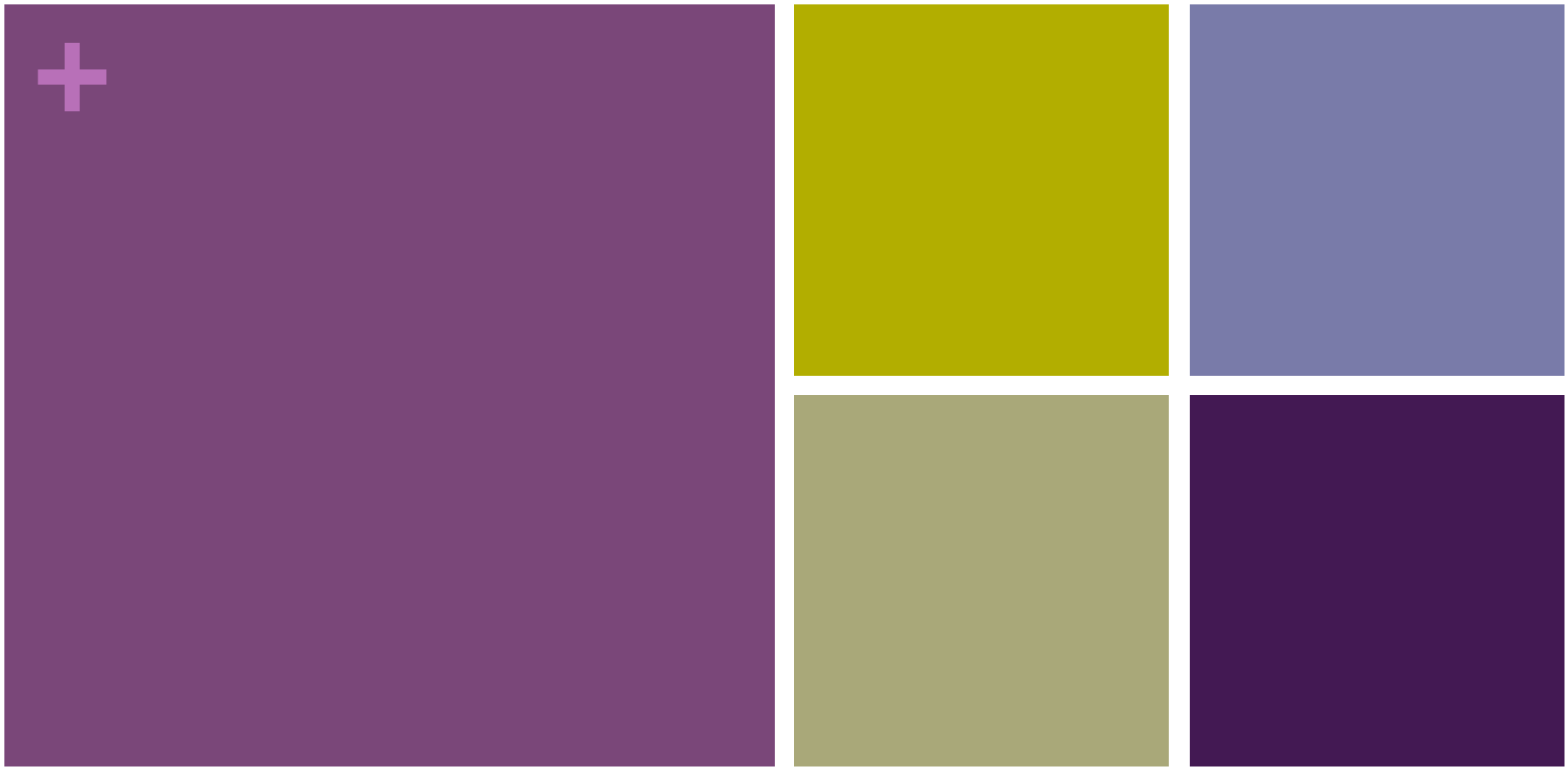
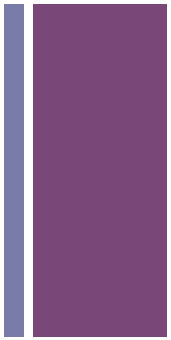


Javascript

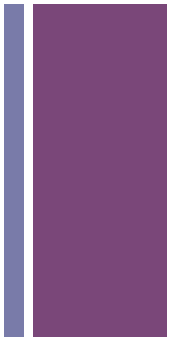
Badrinarayan Parthasarathy

Javascript - Introduction



- Also known as ECMAscript.
- Developed by Netscape
- JavaScript - entirely different from Java.
- Prototype based object-oriented scripting language.
- Dynamic, weakly typed and has first class functions.
- Used in:
  - Web Pages: Used to write functions that are embedded in or included from HTML pages that interact with the Document Object Model of that page.
  - Javascript interpreters, embedded in apps. Used to provide object model and access to the host.

Javascript – Structure



<HTML>

<HEAD>

<TITLE> My JavaScript Page</TITLE>

<SCRIPT>

-Javascript code here

</SCRIPT>

</HEAD>

What a Javascript can do….

- Gives HTML designers a programming tool.
- Used to insert text in a html page.
- React to events. Eg: User click on HTML element.
- Read and write HTML elements.
- Validate data, before submission to server
- Used to store retrieve cookies from the host.

Elements of a JavaScript

- **Variables:**

  var x; var carname="volvo";

- **Arrays:**

  myCar[0]="a"; myCar[1]="b"; myCar[2]="c";

- **Functions**

  function functionname(var1, var2..)

  { function body; (return val;) }

Statements and Comments

- Statements are case sensitive
- Statements – command to a browser

  document.write("Hello Dolly"); , writes 'Hello Dolly' to the webpage.
- Semicolon at the end, not a compusion. Useful for writing multiple statements on a single line.
- Statements – grouped together in blocks. Starts with '{' and ends with '}'.
- Blocks are used for defining functions.
- Single line comments : //single line comments here
- Multi line comments : /*multiline comments here*/

Operators

- Arithmetic : +, -, *, /, %(modulus),

  ++(increment), --(decrement).

- Assignment : = , +=, -=, *=, /=, %=.

- + operator can be used to concatenate strings.

  txt1 = “nice”; txt2 = “dog”; txt=txt1+” “+txt2;

  output: nice dog

- Number + String results in a string.

JavaScript Operators

- Comparison: ==, !=, >=, >, <, <=, ===(is exactly equal to value and type)

  Eg: if (number < x) { …… }

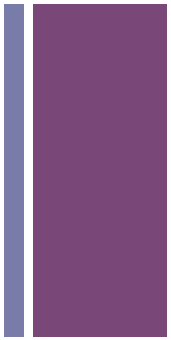
- Logical : &&, | |, !

  Eg if (x==5 || y>5) { …… }

- Conditional: '?'

  variable = (condition)? Val1:Val2 ;

Conditional Statements - I



- if
- if.. else,
- if.. else if.. else
- Eg: if (condition) {}

  else if (condition) {}

  else (condition) {}

Conditional Statements - II

- **Switch.. Case** : Selects one of many blocks of code.
- Structure:

```
Switch (n){
  case 1: {block 1;} break;
  case 2: {block 2;} break;
  default: break;}
```

Pop-up Boxes

- ALERT BOXES: Used to make sure that the information comes through to the user.
  - The user needs to click 'OK' to proceed.
- CONFIRM BOX: Used to verify or accept something.
  - Box returns true on 'OK' and false on 'CANCEL'.
- PROMPT BOX: If we want the user to input a value before entering a page.
  - Box returns input value on 'OK' , 'CANCEL' returns null.

Looping Constructs

- **'for'** loop:

  for (var=start_val; var<=end_val; var = var+inc_val) {…..}

- **'while'** loop:

  while(condition) {…..}

- **'do… while'** loop:

  do {…..} while (condition)

- **Break:** exit the loop, execute code after the loop.

- **Continue:** iterates to the next value in the loop.

JavaScript Events

- Events, used to trigger any JavaScript
- Events include:

  click of mouse, loading of an image/web page, mouse over hot-spot, selecting input field in HTML form, submitting an HTML form, keystroke
- OnLoad, OnUnload
- onFocus, onBlur, onChange
- onSubmit, onMouseOver
- [Illustrate 'onMouseOver' example provided in the PDF]

Try… Catch blocks

- Try Block : used to test the code within it for errors.
- Catch : Used for error handling.

  try { //Run some code here }

  catch(err) { //Handle errors here }

- Throw : Creates an exception. Controls program flow and helps give valid error messages.

Try… Catch blocks

- Example:

```
try { if(x>10) { throw "Err1"; } }
catch(er) { if(er=="Err1")
{ alert("Error! The value is too high"); }}
```

Special Characters

Code	Output
\’	Single quote
\”	Double quote
\\	Backslash
\n	newline
\r	Carriage return
\f	Form feed
\t	Tab
\b	backspace

Objects in JS

- JavaScript-an OOP language
- Properties: Values associated with the object
- Methods: Actions performed on Objects
- <script type="text/javascript">

  var str="Hello world!"; // property of JS in-built object.

  document.write(str.toUpperCase()); //mthd applied on 'str'.

  </script>

Objects in JS

- Inbuilt objects in JavaScript
- **String:** Used to manipulate text.

  Some of the methods include concat(), replace(), slice() etc.
- **Date:** Used to work with date and times.
- Instantiating Date:

  new Date() // current date and time

  new Date(milliseconds) //milliseconds since 1970/01/01

  new Date(dateString)

  new Date(year, month, day, hours, minutes, seconds, milliseconds)

Objects in JS

- Array: Can hold more than one value at a time.

  var myCar = new Array();

  myCar[0]="A";

  myCar[1]="B";

  myCar[2]="C";

- Methods: push(), pop(), reverse() etc. on an Array.

Objects in JS

- **Boolean**: "true" or "false".

  var myBoolean=new Boolean();

  Methods: toString(), valueOf().

- **Math**: Allows us to perform mathematical tasks.

  Math Constants: Math.E, Math.PI, Math.SQRT2, Math.SQRT1_2, Math.LN2, Math.LN10, Math.LOG2E, Math.LOG10E

  Math Methods: document.write(Math.round(4.7));, document.write(Math.random());

Objects in JS

- **RegExp:** Describes a pattern of characters.
- Used for searching pattern in a text.

  var patt = /pattern/modifiers

  modifiers – specifies if a search should be global *(g)*, case-insensitive *(i)* ..
- var str="Is this all there is?"; var patt1=/is/gi;

  //o/p = Is this all there is?

Objects in JS

- **RegExp** methods…
- test(): used to check if a pattern is present in an expression.

```
var patt1=new RegExp("e");

document.write(patt1.test("The best things in life are free"));

// o/p: true
```

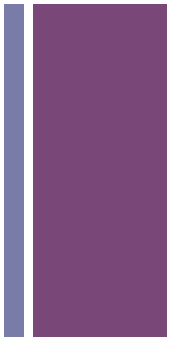
- exec(): searches a string for a specified value, and returns the text of the found value. Else returns NULL.

```
var patt1=new RegExp("e");

document.write(patt1.exec("The best things in life are free"));

//o/p = e
```

Advanced JS - Browser



- Useful to detect the visitors' browser details.
- 'navigator' object contains the details.
- Accessing the details:

  navigator.appCodeName, navigator.appName, navigator.appVersion, navigator.cookieEnabled, navigator.platform, navigator.userAgent

Advanced JS - Cookies

- **Cookie:** **variable stored in the visitors computer**. Cookie value retrieved every time a page is requested by user.
- JavaScript allows both setting and retrieving cookie values.
- Eg: name cookie, password cookie, date cookie.
- Eg: getCookie(), setCookie(), checkCookie() etc.

Advanced JS - Validation

- Validate data in HTML forms before submission to a server.
- Validate e-mail address, required fields, date etc.
- Example:

```
function validateForm()
{var x=document.forms["myForm"]["fname"].value
if (x==null || x=="")
{  alert("First name must be filled out");  return false;  }}
```

Advanced JS – Timing Events

- Possible to execute a code after a specified timing interval.
- Methods:
  - var t = setTimeout("JS statement", milliseconds);
  - clearTimeout(timeout_variable);
- To get an infinite loop, we make use of recursive functions.

Advanced JS – User Defined Objects

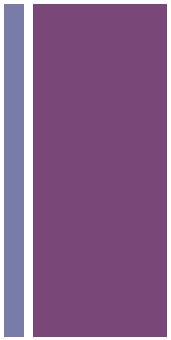
- Collection of properties and methods, in a single entity.
- Accessing object property : **objName.propName**
- Accessing object method: **objName.method()**
- Illustrated example in the 'Example' PDF:

  creating an instance of an object(), constructors.

Advanced JS – User Defined Objects

- Creation of JavaScript object Requires two steps:
  - Step 1 : Declaring an Object using a function.
  - Step 2 : Instantiating the newly created object by using 'new' keyword.
- Example: function newObject(parameter) {}
- Var myObject = new newObject("Badri");

Questions??



Thank You!