|  |  |
| --- | --- |
| **Topics** | **Solutions** |
| Datetime picker | $(function () {  $('#TransactionDate').datepicker({ dateFormat: 'dd-MM-yy', type: Text }).click(function () { $(this).focus(); });  }); // javascript code  ----------------------------  <td>  <input type="text" id="TransactionDate" name="TransactionDate" class="form-control" value="@DateTime.Now.ToString("dd-MMMM-yyyy")" readonly="readonly" /> <span class="error">Transaction Date is required.</span></td>  //C# code |
| Autocomplete | $("#AccountNumber").autocomplete({  source: function(request, response) {  $.ajax({  url: "/Transaction/Transaction/AutoCompleteAccountNumber",  type: "GET",  dataType: "json",  data: { term: request.term },  success: function(data) {  response($.map(data, function(item) {  return { label: item, value: item };  }));  }  });  },  messages: {  noResults: "",  results: ""  }  });  // Action method🡪  [HttpGet]  public JsonResult AutoCompleteAccountNumber(string term)  {  var result = objTransactionBll.GetAllAccountNumberForAutocomplete(term);  return Json(result, JsonRequestBehavior.AllowGet);  } |
| Change Event | $("#AccountNumber").change(function() {  var id = $("#AccountNumber").val();  var json = { id: id };  $.ajax({  type: "POST",  url: "/Transaction/Transaction/GetCustomerInfoByAccountNumber",  contentType: "application/json",  data: JSON.stringify(json),  success: function(result) {  // console.log(result[0]);  $('#CustomerName').val(result[0].CustomerName);  $("#CustomerName").attr("readonly", "readonly");  $('#AccountTypeName').val(result[0].AccountTypeName);  $("#AccountTypeName").attr("readonly", "readonly");  $('#Balance').val(result[0].Balance);  $("#Balance").attr("readonly", "readonly");  $('#AccountSetupId').val(result[0].AccountSetupId);  $('#InterestRate').val(result[0].InterestRate);  $('#CustomerImage').attr('src', result[0].CustomerImage);  },  });  }); |
| Allow only positive numbers | <script type="text/javascript">  function IsDouble(evt) {  evt = (evt) ? evt : window.event;  var charCode = (evt.which) ? evt.which : evt.keyCode;  if (charCode > 31 && (charCode < 48 || charCode > 57) && charCode != 46) {  return false;  }  else if (charCode == 13) {  return false;  }  status = "";  return true;  }  //🡪 in text box use: onkeypress="return IsDouble(event);" |
| After Submit button click  Ajax jeson data post | $('#submit').click(function() {    var isAllValid = true;    if ($('#AccountNumber').val().trim() == '') {  $('#AccountNumber').siblings('span.error').css('visibility', 'visible');  isAllValid = false;  } else {  $('#AccountNumber').siblings('span.error').css('visibility', 'hidden');  }  if ($('#TransactionType').val().trim() == 0) {  $('#TransactionType').siblings('span.error').css('visibility', 'visible');  isAllValid = false;  } else {  $('#TransactionType').siblings('span.error').css('visibility', 'hidden');  }  if ($('#Amount').val().trim() == '') {  $('#Amount').siblings('span.error').css('visibility', 'visible');  isAllValid = false;  } else {  $('#Amount').siblings('span.error').css('visibility', 'hidden');  }  var transactionType = $("#TransactionType option:selected").text();  var amount = parseFloat($('#Amount').val());  var currentBalance = parseFloat($('#Balance').val());  if (transactionType == 'Withdraw' && (currentBalance - 100) < amount)  {  isAllValid = false;    }      if ($('#TransactionDate').val().trim() == '') {  $('#TransactionDate').siblings('span.error').css('visibility', 'visible');  isAllValid = false;  } else {  $('#TransactionDate').siblings('span.error').css('visibility', 'hidden');  }    if (isAllValid) {    var data = {  AccountNumber: $('#AccountNumber').val(),  TransactionType: $('#TransactionType option:selected').val(),  TransactionDate: $('#TransactionDate').val(),  InterestRate: $('#InterestRate').val(),  Amount: $('#Amount').val(),  AccountSetupId: $('#AccountSetupId').val()  }  console.log(data);  $.ajax({  url: '@Url.Action("Index", "Transaction")',  type: "POST",  data: JSON.stringify(data),  dataType: "JSON",  contentType: "application/json",  success: function (d) {  if (d.status === true) {  alert('Successfully done.');    $("#AccountNumber").val('');  $("#TransactionType").val('');  $("#TransactionDate").val('');    }  else {  alert('Failed');  }  $('#submit').val('Save');  window.location.reload();    },  error: function () {  alert('Error. Please try again.');  $('#submit').val('Save');  }  });  }  });    }); |
| Check stock or amount with current value  With proper message | //---checking current balance and withdraw amount  $("#Amount").keyup(function () {  var currentBalance = parseFloat($('#Balance').val());    var transactionType = $("#TransactionType option:selected").text();    var status = $("#MessageForExistChek"); //DIV object to display the status message  var ttDropdownValue = $.trim(transactionType);  if (ttDropdownValue == 'Withdraw') {  var amount = parseFloat($('#Amount').val());    if ((currentBalance-100) >= amount) {    status.html("<font color=green>'<b>" + amount + "</b>' TK. is Available!</font>");  }  else {  status.html("<font color=red>'<b>" + amount + "</b>'TK. your balance is not sufficient.! and you have to keep minimum 100TK in your account.</font>");  //$("#submit").attr("disabled", "disabled");    }    }    });  // specific text box  <input type="text" id="Amount" name="Amount" class="form-control" onkeypress="return IsDouble(event);" />  @Html.Label(" ", new { @class = "col-md-10", id = "MessageForNumberToWordConvert" }) @Html.Label(" ", new { @class = " col-md-offset-4", id = "MessageForExistChek" }) |
| Number to Letter conversion  Exm. 100 it will convert in One hundred. | $("#Amount").keyup(function () {  var amount = parseFloat($('#Amount').val());  var status = $("#MessageForNumberToWordConvert"); //DIV object to display the status message  var user = $.trim(amount);  if (user.length >= 1) {  status.html("Checking....") //While our Thread works, we will show some message to indicate the progress  //jQuery AJAX Post request  $.post("/Transaction/Transaction/GetConvertNumberToWord", { number: amount },  function (data) {  status.html("<font color=green>'<b>" + data + "</b>' TK Only</font>");  });  } else {  status.html("Need more characters...");  }  });  // Action mehod  public JsonResult GetConvertNumberToWord(int number)  {  var exsits = objTransactionBll.GetConvertNumberToWord(number);  return new JsonResult { Data = exsits };  }  // BLL mehod  public string GetConvertNumberToWord(int number)  {  if (number == 0)  return "zero";  if (number < 0)  return "minus " + GetConvertNumberToWord(Math.Abs(number));  string words = "";  if ((number / 1000000000) > 0)  {  words += GetConvertNumberToWord(number / 1000000000) + " Billion ";  number %= 1000000000;  }  if ((number / 10000000) > 0)  {  words += GetConvertNumberToWord(number / 10000000) + " Crore ";  number %= 10000000;  }  if ((number / 1000000) > 0)  {  words += GetConvertNumberToWord(number / 1000000) + " Million ";  number %= 1000000;  }  if ((number / 100000) > 0)  {  words += GetConvertNumberToWord(number / 100000) + " Lakh ";  number %= 100000;  }  if ((number / 1000) > 0)  {  words += GetConvertNumberToWord(number / 1000) + " Thousand ";  number %= 1000;  }  if ((number / 100) > 0)  {  words += GetConvertNumberToWord(number / 100) + " Hundred ";  number %= 100;  }  if (number > 0)  {  if (words != "")  words += "and ";  var unitsMap = new[] { "zero", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine", "Ten", "Eleven", "Twelve", "Thirteen", "Fourteen", "Fifteen", "Sixteen", "Seventeen", "Eighteen", "Nineteen" };  var tensMap = new[] { "zero", "Ten", "Twenty", "Thirty", "Forty", "Fifty", "Sixty", "Seventy", "Eighty", "Ninety" };  if (number < 20)  words += unitsMap[number];  else  {  words += tensMap[number / 10];  if ((number % 10) > 0)  words += "-" + unitsMap[number % 10];  }  }  return words ;  } |
| Append table data after change event | ///this is table  <table class="table" id="hideTableAll" style="width: 90%">  <tr class="borderSet">  <th>Transaction Date</th>  <th>Installment No</th>  <th>Installment Month</th>  <th>Installment Amount</th>  </tr>  <tbody id="allDataShowForCssDeposit"></tbody>  </table>  ///jQuery method (after change text box value)  $("#AccountNumber").change(function () {  var AccountNumber = $("#AccountNumber").val();  var json = {  AccountNumber: AccountNumber  };  $.ajax({  type: "POST",  url: '@Url.Action("GetAllAccountInfoByAccountNo", "CSS")',  contentType: "application/json; charset=utf-8",  data: JSON.stringify(json),  success: function (data) {  $.each(data, function (key, value) {  $("#hideTableAll").show();  $("#allDataShowForCssDeposit").append('<tr class="borderSet"> <td class="borderSet">' + formateDate(value.TransactionDate) + '</td><td class="borderSet">' + value.InstallmentNumber + '</td><td class="borderSet">' + monthNumToName(value.InstallmentMonth) + ' - ' + value.InstallmentYear + '</td><td class="borderSet">' + value.Amount + '</td></tr> ');  });  }  });  }); |
| Populate dropdown in ajax | $(function () {  $("#AccountType").empty();  $.ajax({  type: "POST",  url: '@Url.Action("GetAccontTypeByAccountTitle", "Account")',  contentType: "application/json; charset=utf-8",  success: function (data) {  $("#AccountType").append('<option value="0">--Select Type--</option>');  $.each(data, function (key, value) {  $("#AccountType").append('<option value=' + value.AccountSetupId + '>' + value.AccountTypeName + '</option>');  });  }  });  }); |
| Populate dropdown after textbox change event | $("#AccountTitle").change(function () {  var AccountTitle = $("#AccountTitle").val();  $("#AccountType").empty();  var json = {  AccountTitle: AccountTitle  };  $.ajax({  type: "POST",  url: '@Url.Action("GetAccontTypeByAccountTitle", "Account")',  contentType: "application/json; charset=utf-8",  data: JSON.stringify(json),  success: function (data) {  $("#AccountType").append('<option value="0">--Select Type--</option>');  $.each(data, function (key, value) {  $("#AccountType").append('<option value=' + value.AccountSetupId + '>' + value.AccountTypeName + '</option>');  });  }  });  }); |
| Month number to name And  Name to number convert | function monthNumToName(monthnum) {  return months[monthnum - 1] || '';  }  function monthNameToNum(monthname) {  var month = months.indexOf(monthname);  return month ? month + 1 : 0;  }  // function call  monthNumToName(month) |
| Check box use in table row and get that checked value | $('#chkAll').click(function (e) {  var table = $(e.target).closest('table');  $('td input:checkbox', table).prop('checked', this.checked);  });  ///get checked value  function getCheckedItems(totalItems) {  var items = [];  $('#SaleData').find('input[type="checkbox"]:checked').each(function () {  var id = this.id;  totalItems.forEach(function (item) {  if (id == item.InstallmentNumber) {  items.push(item);  }  })  });  return items;  } |
| Dynamic sql query | ALTER PROCEDURE [dbo].[uspGetDailyLocalSaleReport]    @ConsignmentNumber nvarchar(100)=null,  @FromDate datetime=null,  @ToDate datetime=null,  @ProductId int=null,  @TruckDetailId int=null,  @CustomerId smallint=null  AS  BEGIN  Declare @S\_SQL as VARCHAR(2500)  SET @S\_SQL=' SELECT DISTINCT s.ConsignmentNumber,cu.CustomerId,cu.CustomerAddress,cu.CustomerName,s.SellingDate,sd.TruckNumber,p.ProductId,p.ProductName,sd.SaleUnitBag,  sd.SaleUnitKG,sd.SalePrice,(sd.SaleUnitKG\*sd.SalePrice) AS TotalAmount  FROM Sale s  LEFT JOIN SaleDetail sd ON s.SaleId=sd.SaleId  LEFT JOIN CustomerDetail cu ON s.CustomerId=cu.CustomerId  LEFT JOIN Product p ON sd.ProductId=p.ProductId  WHERE '  if @FromDate <> '' AND @ToDate <> ''  SET @S\_SQL = @S\_SQL + ' s.SellingDate between ''' + Convert(nvarchar(100),@FromDate) + ''' AND ''' + Convert(nvarchar(100),@ToDate) + ''''  if @ConsignmentNumber <> ''  BEGIN  SET @S\_SQL = @S\_SQL + ' AND '  SET @S\_SQL = @S\_SQL + ' s.ConsignmentNumber = '''+ @ConsignmentNumber +''''  END  if @ProductId <> 0  BEGIN  SET @S\_SQL = @S\_SQL + ' AND '  SET @S\_SQL = @S\_SQL + ' p.ProductId = ' + CONVERT(varchar(200), CASE WHEN IsNumeric(CONVERT(VARCHAR(12), @ProductId)) = 1 then CONVERT(VARCHAR(12), @ProductId) else 0 End)  END  if @CustomerId <> 0  BEGIN  SET @S\_SQL = @S\_SQL + ' AND '  SET @S\_SQL = @S\_SQL + ' cu.CustomerId = ' + CONVERT(varchar(200), CASE WHEN IsNumeric(CONVERT(VARCHAR(12), @CustomerId )) = 1 then CONVERT(VARCHAR(12), @CustomerId ) else 0 End)  END  PRINT @S\_SQL  EXEC (@S\_SQL)  END |
| Union All SQL | ALTER PROCEDURE [dbo].[uspSalesLedgerAccountReport]  @FromDate varchar(150),  @ToDate varchar(150)  AS  BEGIN  SELECT \* FROM (  SELECT  @FromDate as SellingDate,  'Opening Balance' AS Header,  '-' as Type,  '-' as ChequeNumber,  '-' AS OrderNo,  0 as Debit,  ISNULL((sale\_amount - purchase\_amount - expense\_amount),0) AS Credit  FROM (  SELECT  'SALES' AS sales\_heder,  SUM(A.sale\_amount) AS sale\_amount,  'PURCHASE' AS purchase\_heder,  SUM(A.purchase\_amount) AS purchase\_amount,  'DAILY EXPENSE' AS expense\_heder,  SUM(A.expense\_amount) AS expense\_amount  FROM (  SELECT  'SALES' AS sales\_heder,  SUM(ISNULL(TotalAmount, 0)) AS sale\_amount,  NULL AS purchase\_header,  NULL AS purchase\_amount,  NULL AS expense\_header,  NULL AS expense\_amount  FROM Sale  WHERE SellingDate < @FromDate  UNION ALL  SELECT  NULL AS sales\_heder,  NULL AS sale\_amount,  'PURCHASE' AS purchase\_heder,  SUM(ISNULL(TotalAmount, 0)) AS purchase\_amount,  NULL AS expense\_header,  NULL AS expense\_amount  FROM purchase  WHERE PurchaseDate < @FromDate  UNION ALL  SELECT  NULL AS sales\_heder,  NULL AS sale\_amount,  NULL AS purchase\_header,  NULL AS purchase\_amount,  'DAILY EXPENSE' AS expense\_heder,  SUM(ISNULL(TotalAmount, 0)) AS expense\_amount  FROM DailyExpense  WHERE Date < @FromDate  ) AS A) AS B  UNION ALL  SELECT  s.SellingDate,  c.CustomerName,  CASE  WHEN s.PaymentMethod = 1 THEN 'Cash'  ELSE 'Cheque'  END AS Type,  ISNULL(ChequeNumber,'-') as ChequeNumber,  ISNULL(s.OrderNumber,'-'),  0 AS Debit,  ISNULL(s.PaidAmount,0) AS Credit  FROM Sale AS s  LEFT JOIN CustomerDetail AS c  ON s.CustomerId = c.CustomerId  WHERE s.SellingDate BETWEEN @FromDate AND @ToDate  ) AS D  END |
| Number(07) Month to Letter(July) month Convert SQL | SELECT DATENAME(month, DATEADD(month, @mydate-1, CAST('2008-01-01' AS datetime))) |
| Get month , year from datetime | declare @month int  declare @year int  set @month= DATEPART(mm,@ToDate)  set @year= DATEPART(YEAR,@ToDate) |
| Html view | @model TonyMultipurpose.Areas.TermLoan.Models.Guarantor  @{  Layout = "~/Views/Shared/\_Layout.cshtml";  }  @{  var prod = ViewData["MyProduct"];  }  <!DOCTYPE html>  <html>  <head>  </head>  <body>  @using (Html.BeginForm("Save", "TermLoan", FormMethod.Post, new { @class = "form-horizontal" }))  {  @Html.AntiForgeryToken()  if (Model != null && Model.GuarantorId > 0)  {  @Html.HiddenFor(d => d.GuarantorId)  }  <div>  <div class="col-md-12">  <div class="col-sm-3 col-md-6 col-lg-5">  <div class="col-sm-7 col-md-4">  <label class="control-label">Generate User Id</label>  </div>  <div class="col-sm-11 col-md-7 col-lg-8 ">  <input type="text" readonly="readonly" name="UserUniqId" class="form-control" value="@ViewBag.showData" />  </div>  </div>  <div class="col-sm-3 col-md-6 col-lg-5">  <div class="col-sm-7 col-md-4">  </div>  <div class="col-sm-11 col-md-7 col-lg-8 ">  </div>  </div>  </div>  <div class="col-sm-3 col-md-10 col-lg-12" id="divRefress">  <table class="table-responsive table" id="hideTableAll">  <tr class="borderSet">  <th>Name</th>  <th>NID</th>  <th>Phone No</th>  <th>Edit</th>  </tr>  <tbody id="allDataShowForCssDeposit"></tbody>  </table>  </div>  </div>  <div class="modal-footer">  <p></p>  </div>  </div>  </div>  } |
|  | public class CommonGetway  {  string Connection = WebConfigurationManager.ConnectionStrings["TonyMultipurposeDB"].ConnectionString.ToString();  public SqlConnection objConnection { get; set; }  public SqlCommand objCommand { get; set; }  public SqlDataReader objDataReader { get; set; }  public SqlDataAdapter objDataAdapter { get; set; }  public DataTable dt { get; set; }  public CommonGetway()  {  objConnection = new SqlConnection(Connection);  }  } |
|  | public string ExistingCheckUniqueNumber(string existingCheck)  {  string lastUniqueNumber = string.Empty;  objCommand = new SqlCommand("TestspGetExistUniqNumber", objConnection);  objCommand.CommandType = CommandType.StoredProcedure;  objCommand.Parameters.Add(new SqlParameter("ExistingCheck", existingCheck));  objConnection.Open();  objDataReader = objCommand.ExecuteReader();  if (objDataReader.HasRows)  {  while (objDataReader.Read())  {  lastUniqueNumber = objDataReader["LastUniqueNumber"].ToString();  }  }  else  {  string b = "Faield";  }  objConnection.Close();  return lastUniqueNumber;  }  public int UniqueNumberSave(string UserUniqId, string UserTempId)  {  objCommand = new SqlCommand("TestspUniqueNumberSave", objConnection);  objCommand.CommandType = CommandType.StoredProcedure;  objCommand.Parameters.Add(new SqlParameter("UserUniqId", UserUniqId));  objCommand.Parameters.Add(new SqlParameter("UserTempId", UserTempId));  objConnection.Open();  int RowAffacted = objCommand.ExecuteNonQuery();  if (RowAffacted > 0)  {  string a = "Success";  }  else  {  string b = "Faield";  }  objConnection.Close();  return RowAffacted;  } |
|  |  |