Internet Programming

Form Processing

Dan Brandon, Ph.D., PMP

Form Submit & Reset Buttons

- There are two special buttons that you normally put on each form: the submit and reset buttons
- The submit button, by default, will send the form data (after it's data has been 'URLencoded') to the designated server program/process (action=…)
- The reset button by default will clear the form
- You can override the behavior of either of these buttons

Overriding Submit/Reset Button Behavior

[can also trigger from ‘onClick of buttons]

```
<HTML><HEAD><TITLE>Forms Processing</TITLE></HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
      function verify(i) {
               var s = "submit";
               if (i == 1) s = "reset";
               window.event.returnValue = false;
               if (confirm ("Are you sure you want to " + s + " ?"))
                                 window.event.returnValue = true;
      }
// --></SCRIPT>
<BODY ONLOAD="document.forms[0].elements[0].focus();">
<H1>Forms Processing Example</H1>
<FORM ONSUBMIT="verify(0)" ONRESET="verify(1)">
Please enter the folowing information, then hit "submit":<P>
<H3>Field0:<INPUT class="black" TYPE="TEXT" Name="F0"></H3><BR>
<H3>Field1:<INPUT class="black" TYPE="TEXT" Name="F1"></H3><BR>
<H3>Field2:<INPUT class="black" TYPE="TEXT" Name="F2"></H3><BR>
<INPUT class="black" TYPE="SUBMIT" VALUE="Submit" Name="Submit">
<INPUT class="black" TYPE="RESET" VALUE="Reset" Name="Reset">
</FORM></BODY></HTML>
```

Form Validation (onBlur & onSubmit)

- Form processing involves the input, validation, and transmittal of data from the browser to the information server (action=)
- Data may be validated when the user tabs or clicks off of a field (onBlur), when the form submit button is hit, or when the data gets to the server
- The most efficient and most responsive way to do validation (from HTML forms) is on the browser via JS
- Many current web applications do not handle this area well (error messages, focus, colors, etc.)
- Project 2 deliverable !!!

Validation (con't)

- The validation of input submitted on a form often takes the bulk of the time in building a web form
- To avoid bad user input, one needs to validate the data before processing
- Validation is best done at the time of accepting the input; the earlier you can prevent bad data, the better
- Validation was historically completed by server-side processing, which was executed once the user submitted the form
  - In this type of validation, the user provided inputs were submitted to the server, which then processed the input and determined if the inputs were incorrect or incomplete and sent back an appropriate error message
- With the rise of JavaScript, browsers began support for client-side validation through specialized validation functions and libraries

Field Level Validation
(onBlur or onChange of the field)

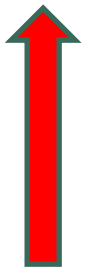
- <html><head>
- <title>My Document</title>
- <script language="JavaScript">
- <!-- start of script --
- function ValidateAge(formObject) {
  - if (formObject.value <=18) {
    - alert("Age must be greater than 18 !");
    - formObject.value = 0;
  - }
- }
- <!-- end of script --></script></head>

Instead of using "formObject" we could write document.myForm.age

Look at next slide first !

Field Level Validation (con't)

- <body>
- <form name = "myForm" method="post">
- Enter Data:<p>
- Name: <input type="text" name="name" size = 30>
- Age: <input type="number" name="age" size =5 onBlur="ValidateAge(this);">
- </form>
- </body>
- </html>



Referring to the value entered:

- The value that is entered in a form field needs to be available to the function doing the validation
- There are several ways to do this, not all are compatible across browsers:
  - Hardcode object names
    - function checkMyField() {
      - document.myForm.myField.value
  - Passing the value to the function as an argument (“item”);
    - function qty_check(item, … // use ‘this’ on call to function
      - item.value
  - Using the form object and position on form:
    - function check(i, …
      - document.forms[0].elements[i].value
  - Using the event object:
    - function check (…
      - event.srcElement.value

Class Exercise

- Rewrite the previous validation example to use:
  - document.form-name.field-name.vaue
- Instead of the “this” object

Don’t look ahead …

onBlur2.html

```
<html>
<head>
        <title>My Document</title>
        <script language="JavaScript">
        function ValidateAge() {
                if (document.myForm.age.value <=18) {
                        alert("Age must be greater than 18 !");
                        document.myForm.age.value = 0;
                }
        }
        </script>
</head>
<body>
        <form name = "myForm" method="post">
                Enter Data:<p>
                Name: <input type="text" name="name" size = 30>
                Age: <input type="number" name="age" size =5 onBlur="ValidateAge()">
        </form>
</body>
</html>
```

Submit Level Validation

(onClick of submit button, or onSubmit of form)

- <html><head><title>My Document</title>
- <script language="JavaScript">
- <!-- start of script --
- function validateForm(formObject) {
  - if (formObject.value =="") {
    - alert("must enter some comments");
    - return false; // form will not be submitted
  - }
  - else
    - return true;
- }
- <!-- end of script --></head>

See next slide first !

Submit Level Validation (con't)

- <body>
- <form name = "evalForm" method="post" action="http://www.cbu.edu/cgi-bin/myProg.cgi" onSubmit="validateForm(evalForm.eval);">
- Enter Data:<p>
- Name: <input type="text" name="name" size = 30>
- Course Evaluation:
- <textarea name="eval" rows=5 cols=40 </textarea><p>
- <input type="submit" value=Submit">
- </form></body></html>

Setting Focus

- It is desirable to set focus in a form (or elsewhere) for your user
- This code sets the focus to the first field of the first form on a page:

```
document.forms[0].elements[0].focus();
```

Validation Example at both the Field and Form Level

```
<BODY>
<FORM NAME="widget_order" ACTION="lwapp.html“
        METHOD="post">
        How many widgets today?
        <INPUT TYPE="text" NAME="quantity“
                onChange="qty_check(this, 0, 999)">
        <BR>
        <INPUT TYPE="button" VALUE="Enter Order“
                onClick="validateAndSubmit(this.form)">
</FORM>
</BODY>
```

Or using a “submit” button
[can also use onClick for submit button]

- <FORM NAME="widget_order" ACTION="lwapp.html“
- METHOD="post“
- onSubmit="return qty_check(theform.quantity, 0, 999)">

...

- <INPUT TYPE="submit">
- ...
-

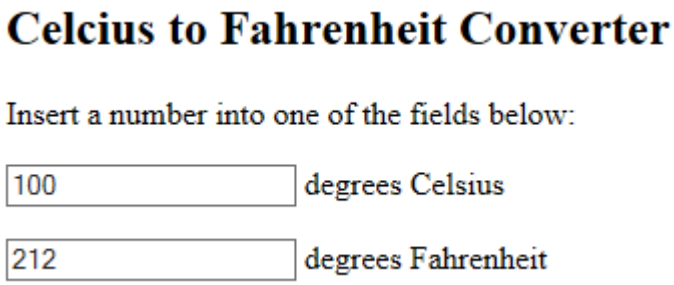
</FORM>

Validation Example at both the Field and Form Level (con't)

```
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
function isaPosNum(s) {
   return (parseInt(s) > 0)}
function qty_check(item, min, max) {
   var returnVal = false
   if (!isaPosNum(item.value))
      alert("Please enter a positive number")
   else if (parseInt(item.value) < min)
      alert("Please enter a " + item.name + " greater than " + min)
   else if (parseInt(item.value) > max)
      alert("Please enter a " + item.name + " less than " + max)
   else
      returnVal = true
   return returnVal}
function validateAndSubmit(theform) {
   if (qty_check(theform.quantity, 0, 999)) {
      alert("Order has been Submitted")
      return true
   }
   else {
      alert("Sorry, Order Cannot Be Submitted!")
      return false
   }
}
</SCRIPT>
</HEAD>
```

Keypress Events

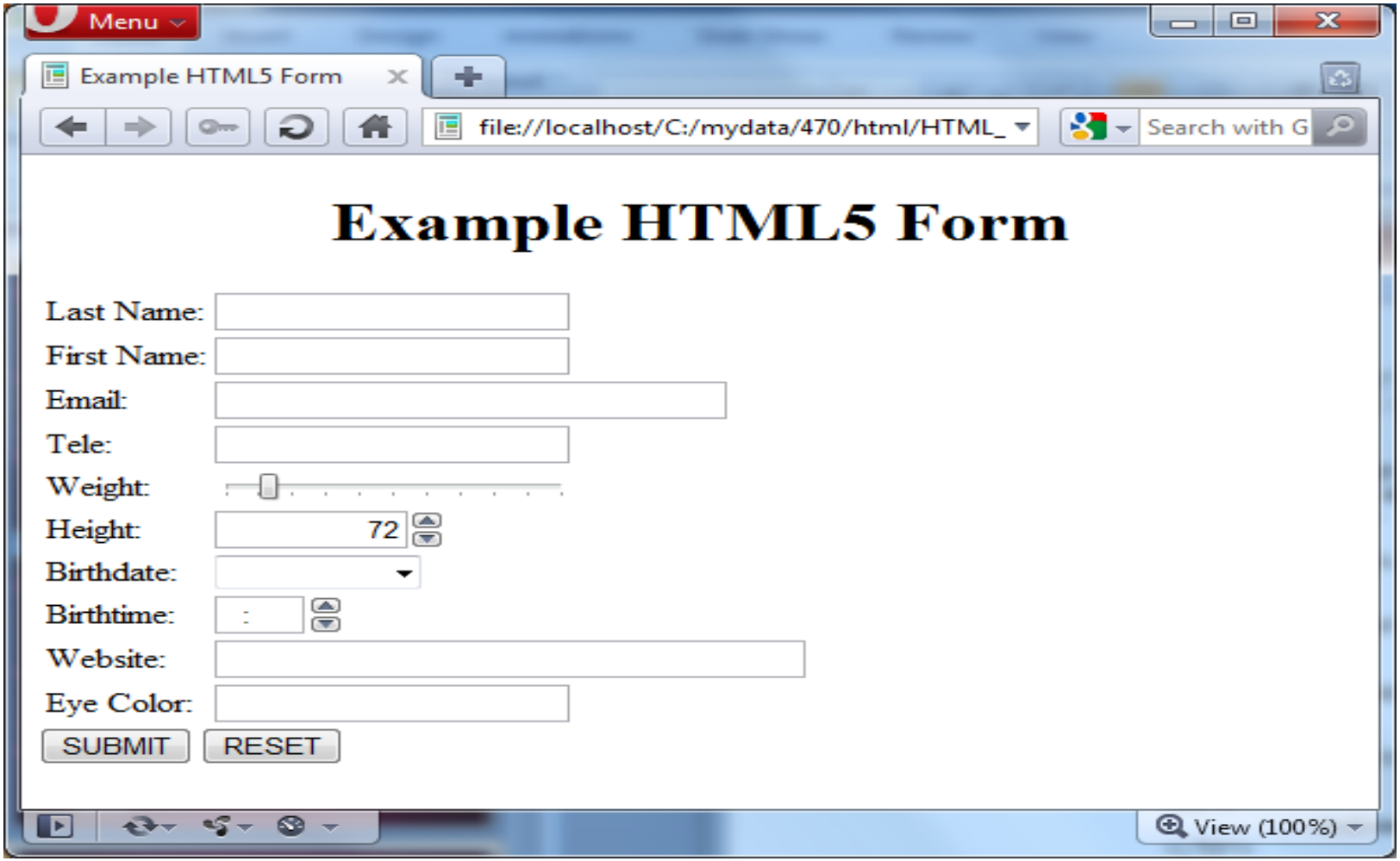
```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<script>
function convert(degree) {
    var x;
    if (degree == "C") {
        x = document.getElementById("celcius").value * 9 / 5 + 32;
        document.getElementById("fahrenheit").value = Math.round(x);
    } else {
        x = (document.getElementById("fahrenheit").value -32) * 5 / 9;
        document.getElementById("celcius").value = Math.round(x);
    }
}
</script>
</HEAD>
<BODY>
<h2>Celcius to Fahrenheit Converter </h2>
<p>Insert a number into one of the fields below:</p>
<p><input id="celcius" onkeyup="convert('C')"> degrees Celsius</p>
<p><input id="fahrenheit" onkeyup="convert('F')"> degrees Fahrenheit</p>
</BODY>
</HTML>
```



HTML 5 Extensions for Forms

- HTML 5 provides new type of form controls which provides some error checking and entry aids such as “pickers”:
  - range (exact number not important)
  - number
  - telephone number
  - color
  - date, time, datetime (UTC), datetime-local, month, week
  - email
  - image
  - search
  - url
  - Password
- Required, numeric checking and limit checking included

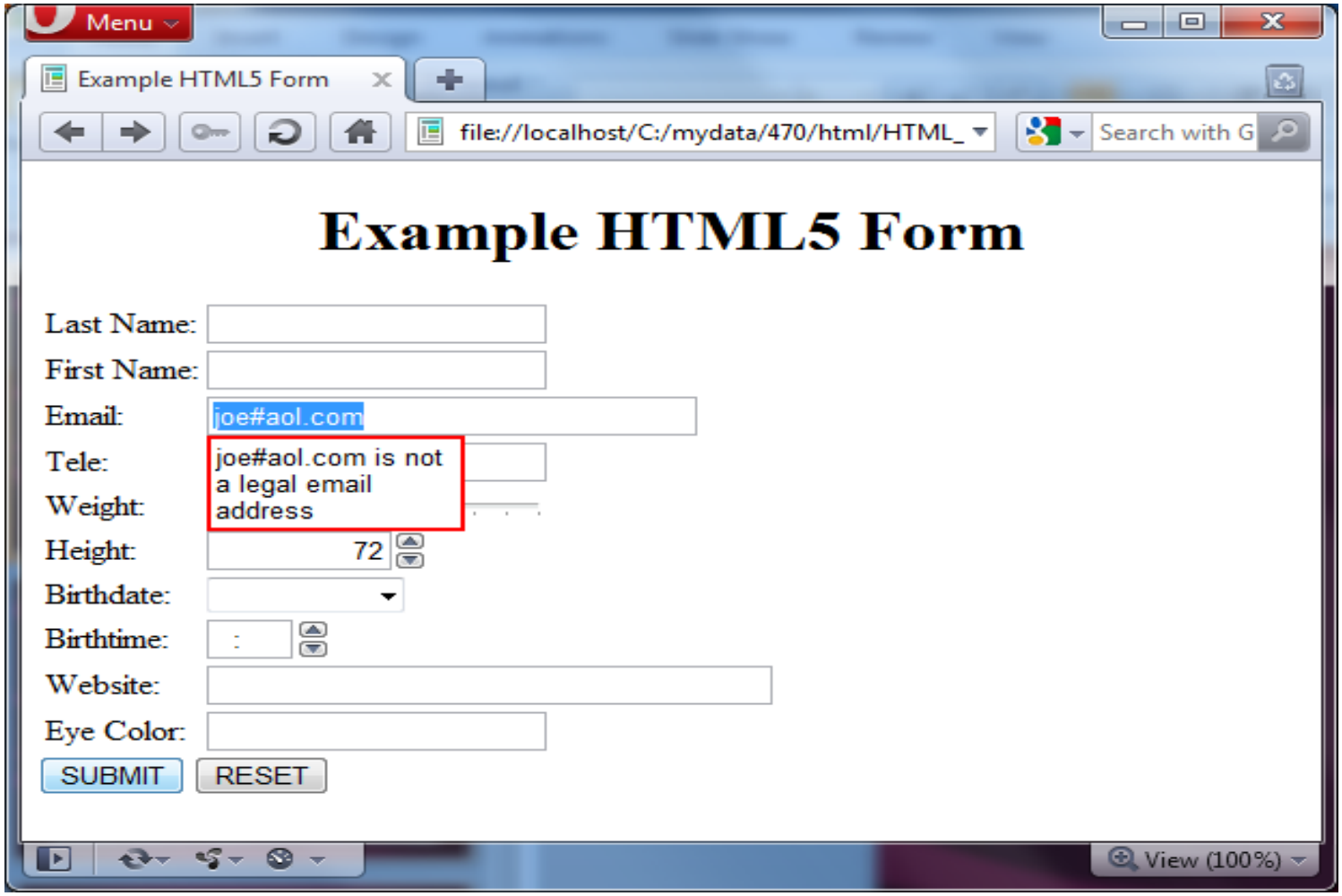
Example Form



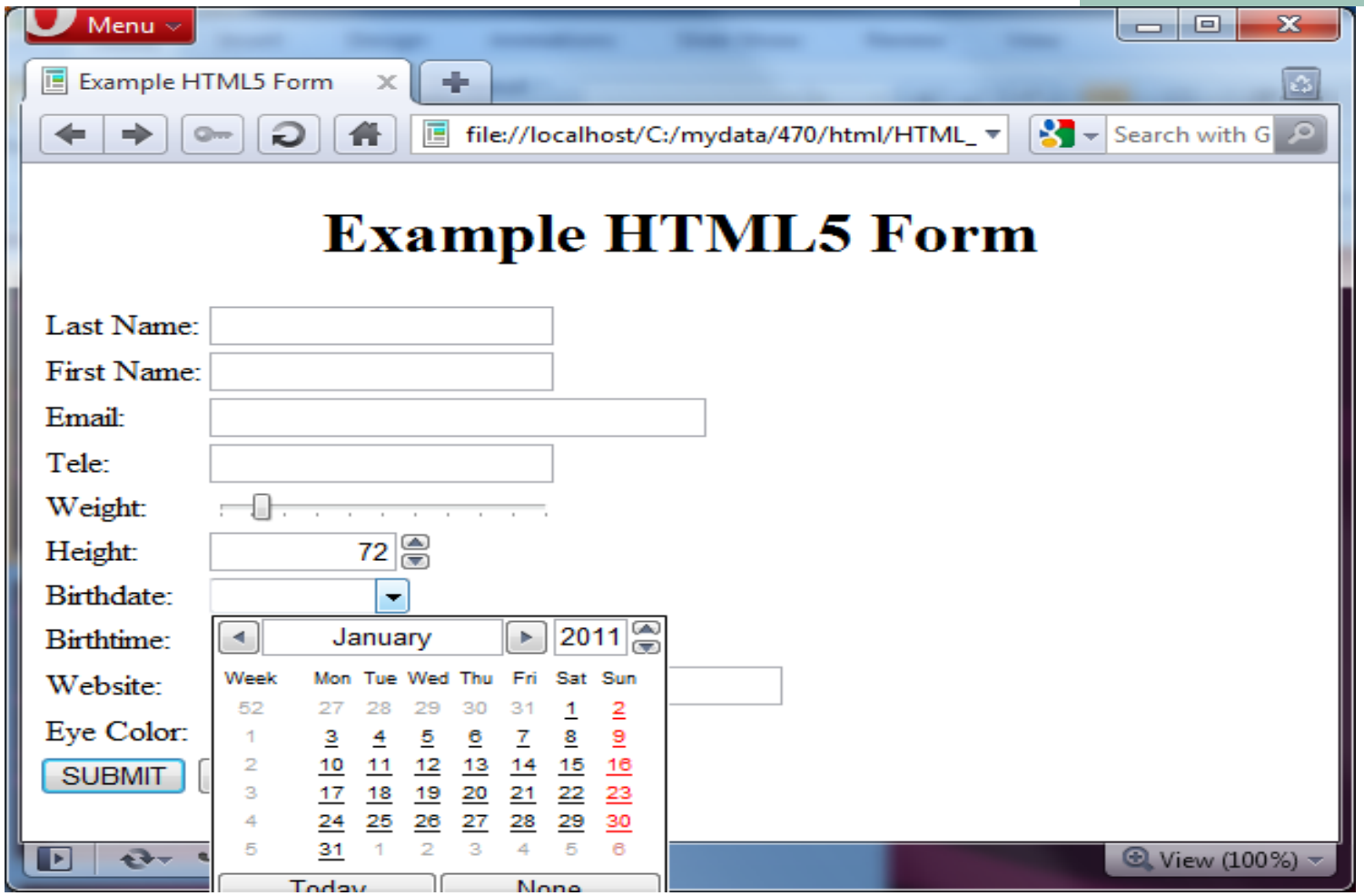
Example HTML 5 Form Code

```
<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8">
		<title>Example HTML5 Form</title>
	</head>
	<body>
		<h1 style="text-align:center">Example HTML5 Form</h1>
		<form method="post" action="echo.php">
		<table align="left">
			<tr>
				<td>Last Name:</td><td><input name="lname' placeholder="Enter your last name" autofocus></td>
			</tr>
			<tr>
				<td>First Name:</td><td><input name="fname' placeholder="Enter your first name"></td>
			</tr>
			<tr>
				<td>Email:</td><td><input type="email" name="email' placeholder="Enter your email address"></td>
			</tr>
			<tr>
				<td>Tele:</td><td><input type="tel" name="tele' placeholder="Enter your telephone number"></td>
			</tr>
			<tr>
				<td>Weight:</td><td><input type="range" min="50" max="1000" step="10" value="170" name="weight" placeholder="Enter your weight in
pounds"></td>
			</tr>
			<tr>
				<td>Height:</td><td><input type="number" min="36" max="96" step="2" value="72" name="height" placeholder="Enter your weight in
pounds"></td>
			</tr>
			<tr>
				<td>Birthdate:</td><td><input type="date" name="bdate' placeholder="Enter your birthdate"></td>
			</tr>
			<tr>
				<td>Birthtime:</td><td><input type="time" name="btime' placeholder="Enter your birth time (24 hour)"></td>
			</tr>
			<tr>
				<td>Website:</td><td><input type="url" name="web' placeholder="Enter your web site"></td>
			</tr>
			<tr>
				<td>Eye Color:</td><td><input type="color" name="ecolor' placeholder="Enter your eye color"></td>
			</tr>
		</table>
		<br clear="all">
		<input type="submit" value="SUBMIT">
		<input type="reset" value ="RESET">
		</form>
	</body>
</html>
```

Example Form (con't)



Example Form (con't)



Styling and Error Checking

- <!DOCTYPE html>
- <html>
- <head>
- <style>
  - input:required:invalid {background-color: lightyellow;}
- </style>
- </head>
- <body>
- <header><h1>Validation demo</h1> Demo showing HTML5 validation </header>
- <footer><h1></h1> End of Demo </footer>
- <form id="myform" action="#">
  - <input id="name" required>
  - <input id="phone">
  - <input id="zip" required>
  - <input type="submit">
- </form>
- </body>
- </html>

External JavaScript Files

- You can keep your JS separate form your HTML file
- For example, you can maintain your JS form validation code in one file that is used with all your HTML files:
  - <SCRIPT LANGUAGE="JavaScript" src="myForms.js" ></SCRIPT>

Example of External JS Files

```
<HTML>
<HEAD><TITLE>Forms Processingr</TITLE>
<SCRIPT LANGUAGE="JavaScript">
      var helpText = ["F0 Help Text", "F1 Help Text", "F2 Help Text"];
      var required = [1, 1, 0]; //List of required fields
</SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="forms.js"></SCRIPT>
<LINK REL="stylesheet" TYPE="text/css" HREF="forms.css">
</HEAD>
<BODY ONLOAD="document.forms[0].elements[0].focus();">
<H1>Forms Processing Example</H1>
<FORM Method="POST" ACTION="myCGI">
Please enter the folowing information, then hit "submit":
<P>
<H3 class="black" ID="F0L">Field0:<INPUT class="black" TYPE="TEXT"
          Name="F0" ONFOCUS="over(0)" ONBLUR="out(0)"></H3><BR>
<H3 class="black" ID="F1L">Field1:<INPUT class="black" TYPE="TEXT"
          Name="F1" ONFOCUS="over(1)" ONBLUR="out(1)"></H3><BR>
<H3 class="black" ID="F2L">Field2:<INPUT class="black" TYPE="TEXT"
          Name="F2" ONFOCUS="over(2)" ONBLUR="out(2)"></H3><BR>
<INPUT class="black" TYPE="SUBMIT" VALUE="Submit" Name="Submit">
<INPUT class="black" TYPE="RESET" VALUE="Reset" Name="Reset">
</FORM>
</BODY></HTML>
```

.js & .css External Files

forms.js

```
var req = 1;
function over(i) {
      window.status = helpText[i];
      event.srcElement.className = "blue";
}
function out(i){
      window.status = "";
      if (req == 0) {
                req = 1;
                return;
      }
      if (required[i] == 1 &&
event.srcElement.value == "") {
                alert ("This is a required
field");
                req = 0; // turn off req
check on re-focus

      document.forms[0].elements[i].focus();
                return;
      }
      event.srcElement.className = "black";
      req = 1;
}
```

- forms.css
  - .blue {color: blue}
    .red {color:red}
  - .black color:black}

Advantages of building and maintaining your own library of .js and .css files ! Manually do HTML or use “two-way” wysiwyg editiors

Validation of Individual Form Fields

- Each field on your form should be fully checked before the data of the form is used (ie to update a database)
- You can write (or find, or buy) code to validate each type of common input such as names, numbers, currency amounts, dates, times, ss numbers, credit card numbers, states, zip codes, etc.

Server Validation

- Not all validation may be possible on the browser since data may need to be validated against dynamic or database values that reside on the information server or database server (i.e. foreign keys)
- In that case the data can either be validated by the server process (action=), or by new AJAX capabilities within JavaScript

Server Generated HTML/JS

- To include dynamic validation information and/or dynamic content, either the HTML and/or associated JavaScript can be generated (written by) a program running on the server; the browser page would simply have a link to the program which generates the HTML/JS code
- This is available to a limited extend under CGI or Microsoft's "Active Server Pages" and to a greater extent under Cold Fusion, PHP, Python, Java "Servlets" or "Java Server Pages"
- The most effective and efficient corporate e-commerce web sites use dynamically generated HTML/JS

Timeouts

- <html><head><title>My Document</title>
- <script language="JavaScript">
- <!-- start of script --
- function timer(seconds) {
  - // arguments to setTimer are function name & miliseconds
  - setTimer('document.back()', seconds*1000);
- }
- <!-- end of script -->
- </head>

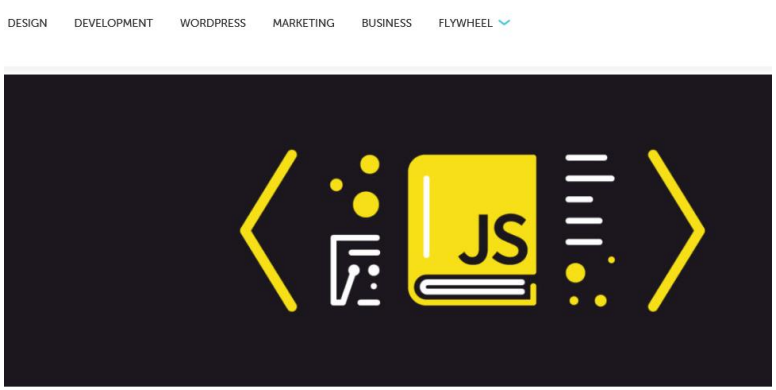
- <body onLoad="timer(30);">
- Password Screen...<hr>
- <form name = "mylForm" method="post" action="http://www.cbu.edu/cgi-bin/myProg.cgi">
- Enter User Name and password:<p>
- Name: <input type="text" name="name" size = 10>
- Password: <input type="text" name="pw" size = 10>
- You will be returned to previous page in 30 seconds.<p>
- <input type="submit" value=Submit">
- </form>
- </body>
- </html>

Executing JS Code From Links

- Click Here to …
- Or a function call:
- <Click Here to …
- Or:
- …

JS Libraries and Sources

- There are many free and commercial JS libraries available
- These include libraries for form validation as well as other common JS functions

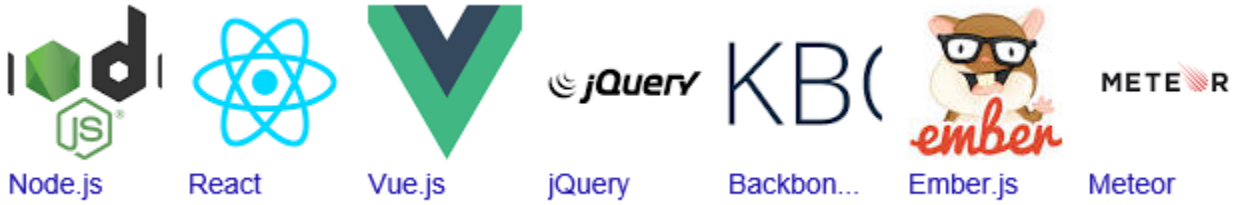


Other DOM Events

- There are many other types of events including:
  - Clipboard events (cut, paste, …)
  - Data binding events
  - Keyboard events
  - Mouse moving and dragging
  - Printing
  - Other …
- Many of these are now supported in HTML5 – covered later in this course

Advanced JavaScript

- JS has many more features and capabilities beyond those covered in this course:
  - Unicode support
  - Bitwise operations
  - Regular Expressions
  - Exception Handling (try, catch, throw)
  - Classes and Inheritance
  - Server side operation



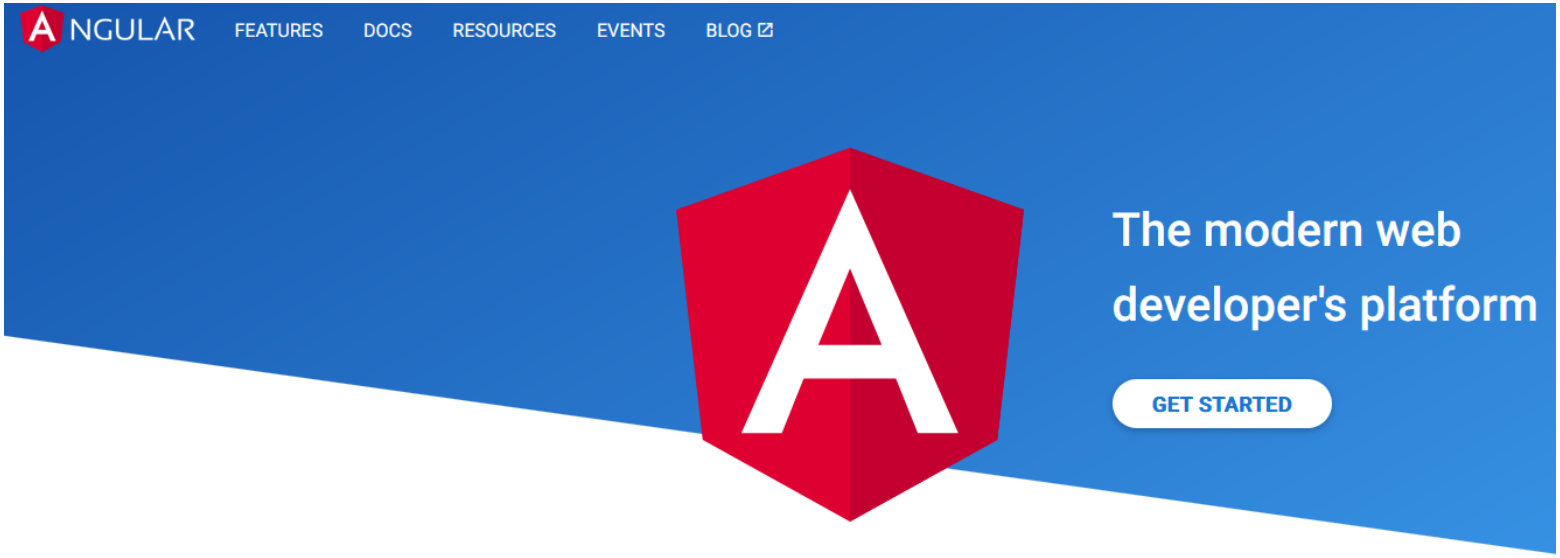
jQuery

- It is difficult to write and maintain JavaScript to support all browsers, and changes to browsers in time
- jQuery is a JavaScript library that permits you to reference DOM objects in a more direct manner
- JQuery also has operators and functions that make some things in JS easier
- One needs to download that code from http://jquery.com, and then refer to it as an external Javascript library
- For example:
  - This: document.getElementById (‘xxx’).innerHTML = “yyy”;
  - Is replaced by: $(‘xxx’).text(‘yyy’);

Angular

- Angular is a development platform form Google, built on TypeScript (a strong typing version of JavaScript)
- As a platform, Angular includes:
  - A component-based framework for building scalable web applications
  - A collection of well-integrated libraries that cover a wide variety of features, including routing, forms management, client-server communication, and more
  - A suite of developer tools to help you develop, build, test, and update your code
- Angular can scale from single-developer projects to enterprise-level applications
- The Angular ecosystem consists of a diverse group of millions developers, library authors, and content creators

Angular (con’t)



DEVELOP ACROSS ALL PLATFORMS

Learn one way to build applications with Angular and reuse your code and abilities to build apps for any deployment target. For web, mobile web, native mobile and native desktop.

Angular (con't)

- Angular is based upon "components"; the following is a minimal Angular component.

```
import { Component } from '@angular/core';
@Component({
  selector: 'hello-world',
  template: `
    <h2>Hello World</h2>
    <p>This is my first component!</p> ‘
 })
export class HelloWorldComponent {
  // The code in this class drives the component's behavior.
}
```

- To use this component, you write the following in a template:

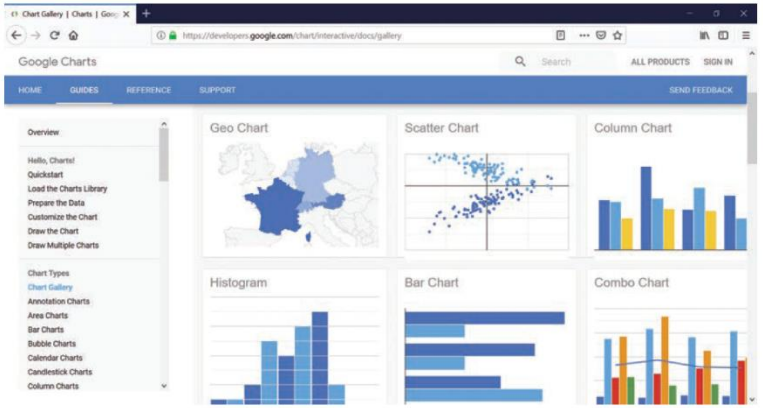
```
<hello-world></hello-world>
```

- When Angular renders this component, the resulting DOM looks like this:

```
<hello-world>
    <h2>Hello World</h2>
    <p>This is my first component!</p>
</hello-world>
```

Google Charts

- Advantages of using Google Charts
  - It is free to use
  - It supports web and mobile solutions
  - It is easy to integrate



Used with permission of Google

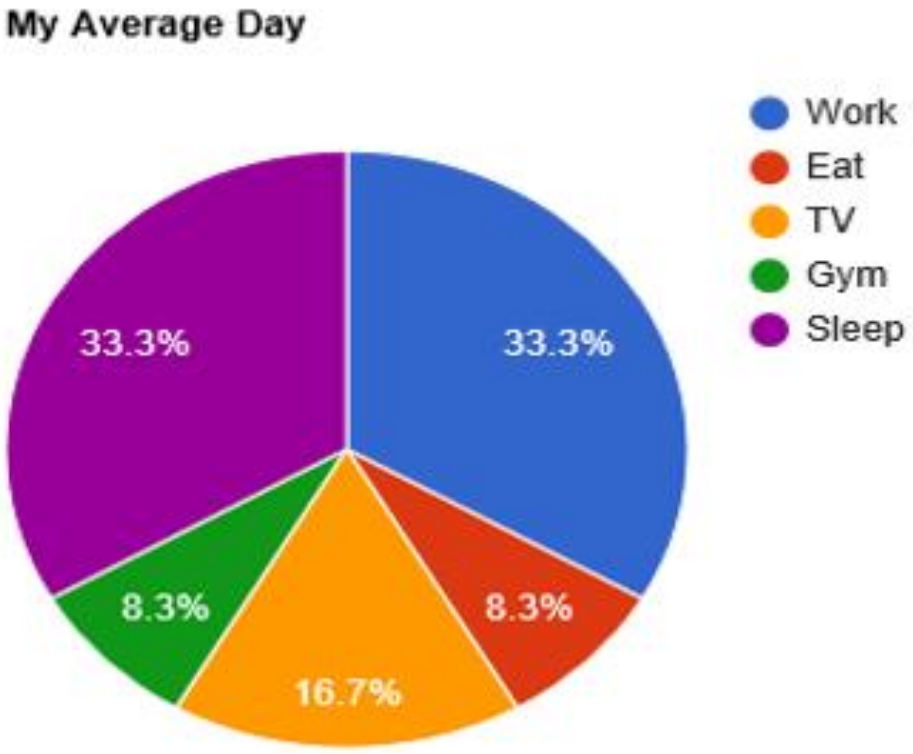
Google Charts (con't)

- **Google Charts** is a pure JavaScript based charting library
- Google Charts provides wide variety of charts; for example, line charts, spline charts, area charts, bar charts, pie charts and so on
- The most common way to use **Google Charts** is with JavaScript that you embed in your web page
  - You load some **Google Chart** libraries
  - List the data to be charted
  - Select options to customize your **chart**
  - **C**reate a **chart** object with an id that you choose

Google Charts (con't)

```
<script type="text/javascript">
// Load google charts
google.charts.load('current', {'packages':['corechart']});
google.charts.setOnLoadCallback(drawChart);

// Draw the chart and set the chart values
function drawChart() {
  var data = google.visualization.arrayToDataTable([
  ['Task', 'Hours per Day'],
  ['Work', 8],
  ['Friends', 2],
  ['Eat', 2],
  ['TV', 2],
  ['Gym', 2],
  ['Sleep', 8]
]);

  // Optional; add a title and set the width and height of the chart
  var options = {'title':'My Average Day', 'width':550, 'height':400};

  // Display the chart inside the <div> element with id="piechart"
  var chart = new google.visualization.PieChart(document.getElementById('piechart'));
  chart.draw(data, options);
}
</script>
```

Google Charts (con't)



Dual y-Axis Chart (Google Charts)

DualYChart - Notepad

File Edit Format View Help

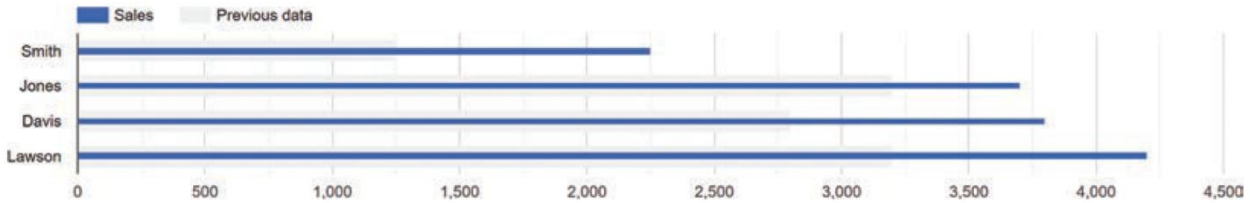
```
<html>
<head>
  <script type="text/javascript"
src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript">
      google.charts.load('current', {'packages':['line']});
      google.charts.setOnLoadCallback(drawChart);
      google.charts.load('current', {'packages':['line', 'corechart']});
      google.charts.setOnLoadCallback(drawChart);

    function drawChart() {
      var chartDiv = document.getElementById('chartDiv');
      var data = new google.visualization.DataTable();
      data.addColumn('date', 'Month');
      data.addColumn('number', "Revenues");
      data.addColumn('number', "Expenses");

      data.addRows([
        [new Date(2019, 0),  5000, 2000],
        [new Date(2019, 1),  6500, 2200],
        [new Date(2019, 2),  7500, 2300],
        [new Date(2019, 3),  5900, 2800],
        [new Date(2019, 4),  8000, 3200],
        [new Date(2019, 5),  8500, 3500],
        [new Date(2019, 6),  9000, 4000],
        [new Date(2019, 7),  7500, 3800],
        [new Date(2019, 8),  7000, 3500],
        [new Date(2019, 9),  6500, 3250],
        [new Date(2019, 10), 6000, 2800],
        [new Date(2019, 11), 5500, 2500]
      ]);
```

DualYChart - Notepad

File Edit Format View Help

```
      var materialOptions = {
        chart: {
          title: 'Revenues versus Expenses'
        },
        width: 900,
        height: 500,
        series: {
          // Gives each series an axis name that matches the Y-axis below.
          0: {axis: 'Revenues'},
          1: {axis: 'Expenses'}
        },
        axes: {
          // Adds labels to each axis; they don't have to match the axis names.
          y: {
            Temps: {label: 'Revenues'},
            Daylight: {label: 'Expenses'}
          }
        }
      };

      function drawMaterialChart() {
        var materialChart = new google.charts.Line(chartDiv);
        materialChart.draw(data, materialOptions);
      }

      drawMaterialChart();
    }
  </script>
</head>
<body>
  <div id="chartDiv"></div>
</body>
</html>
```

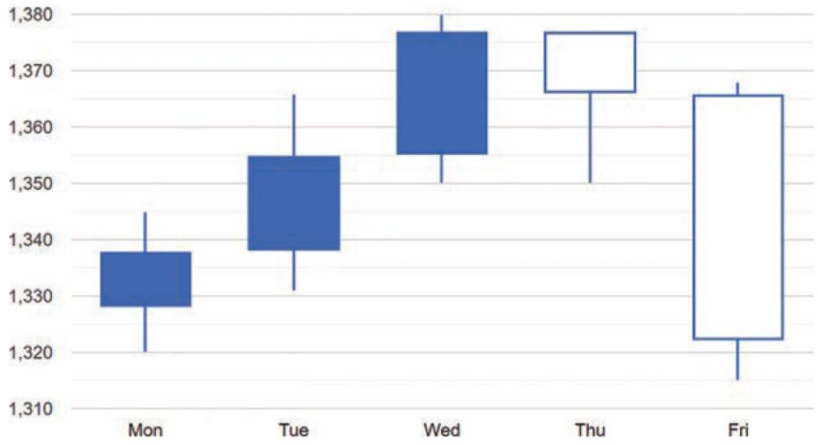
Diff Chart (Google Charts)



Used with permission of Google

```
<html>
<head>
  <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
  <script type="text/javascript">
    google.charts.load('current', {packages:['corechart']});
    google.charts.setOnLoadCallback(drawChart);

  function drawChart() {
    var Sales2018Data = google.visualization.arrayToDataTable([
      ['Name', 'Sales'],
      ['Smith', 1250],
      ['Jones', 3200],
      ['Davis', 2800],
      ['Lawson', 3200]
    ]);

    var Sales2019Data = google.visualization.arrayToDataTable([
      ['Name', 'Sales'],
      ['Smith', 2250],
      ['Jones', 3700],
      ['Davis', 3800],
      ['Lawson', 4200]
    ]);

    var barChartDiff = new google.visualization.BarChart(document.getElementById('chartDiv'));
    var diffData = barChartDiff.computeDiff(Sales2018Data, Sales2019Data);
    var options = { legend: { position: 'top' } };
    barChartDiff.draw(diffData, options);
  }
</script></head>
<body>
  <div id="chartDiv"></div>
</body>
</html>
```

Candlestick Chart (Google Charts)



StockPrices - Notepad

File Edit Format View Help

```
<html>
  <head>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript">
      google.charts.load('current', {'packages':['corechart']});
      google.charts.setOnLoadCallback(drawChart);

  function drawChart() {
    var data = google.visualization.arrayToDataTable([
      ['Mon', 1320, 1328, 1338, 1345],
      ['Tue', 1331, 1338, 1355, 1366],
      ['Wed', 1350, 1355, 1377, 1380],
      ['Thu', 1377, 1377, 1366, 1350],
      ['Fri', 1368, 1366, 1322, 1315]
    ], true);

    var options = {
      legend:'none'
    };

    var chart = new google.visualization.CandlestickChart(document.getElementById('chartDiv'));

    chart.draw(data, options);
  }
    </script>
  </head>
  <body>
    <div id="chartDiv" style="width: 900px; height: 500px;"></div>
  </body>
</html>
```

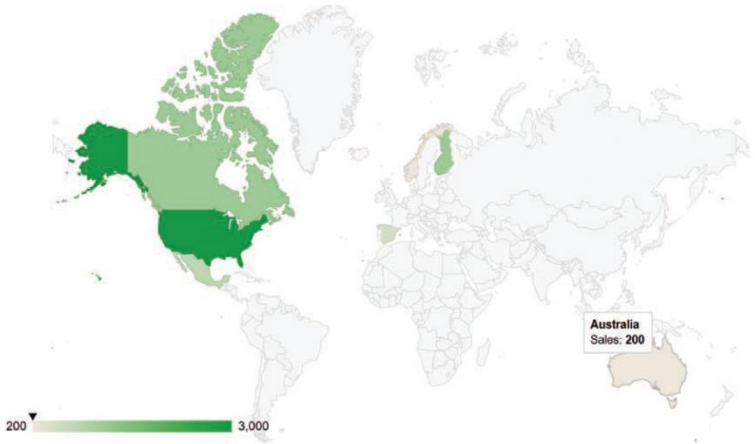
Gauge Chart (Google Charts)

ShowGauge - Notepad

File Edit Format View Help

```
<html>
  <head>
   <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
   <script type="text/javascript">
      google.charts.load('current', {'packages':['gauge']});
      google.charts.setOnLoadCallback(drawChart);

      function drawChart() {

        var data = google.visualization.arrayToDataTable([
          ['Label', 'Value'],
          ['Revenues', 80],
          ['Expenses', 55]
        ]);

        var options = {
          width: 400, height: 120,
          redFrom: 90, redTo: 100,
          yellowFrom:75, yellowTo: 90,
          minorTicks: 5
        };

        var chart = new google.visualization.Gauge(document.getElementById('chartDiv'));
        data.setValue(0, 1, 85);
        data.setValue(1, 1, 70);
        chart.draw(data, options);
      }
    </script>
  </head>
  <body>
    <div id="chartDiv" style="width: 400px; height: 120px;"></div>
  </body>
</html>
```

Geochart (Google Charts)

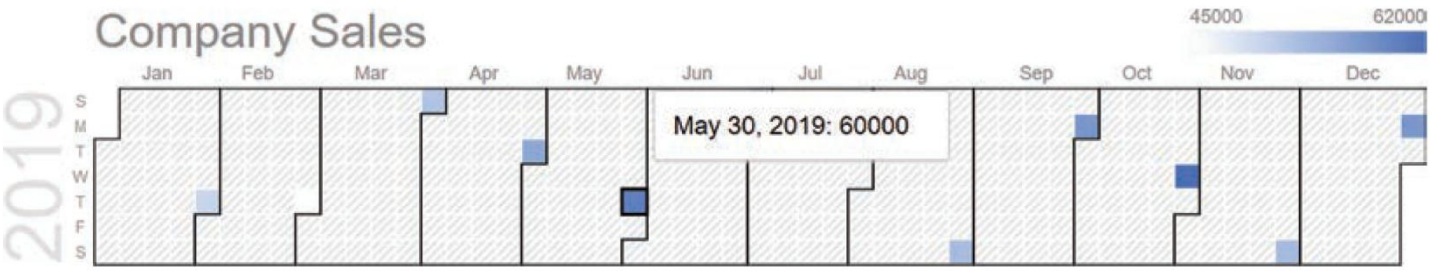


*ShowWorld - Notepad

File Edit Format View Help

```
<html>
  <head>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript">
      google.charts.load('current', {
        'packages':['geochart'],
        // Note: you will need to get a mapsApiKey for your project.
        // See: https://developers.google.com/chart/interactive/docs/basic_load_libs#load-settings
        'mapsApiKey': 'AIzaSyD-9tSrke72PouQMnMX-a7eZSW0jkFMBWY'
      });
      google.charts.setOnLoadCallback(drawRegionsMap);

      function drawRegionsMap() {
        var data = google.visualization.arrayToDataTable([
          ['Country', 'Sales'],
          ['Finland', 1200],
          ['United States', 3000],
          ['Mexico', 800],
          ['Canada', 1200],
          ['Spain', 500],
          ['Norway', 200],
          ['Australia', 200]
        ]);
        var options = {};
        var chart = new google.visualization.GeoChart(document.getElementById('chartDiv'));
        chart.draw(data, options);
      }
    </script>
  </head>
  <body>
    <div id="chartDiv" style="width: 900px; height: 500px;"></div>
  </body>
</html>
```

Calendar Chart (Google Charts)



Used with permission of Google

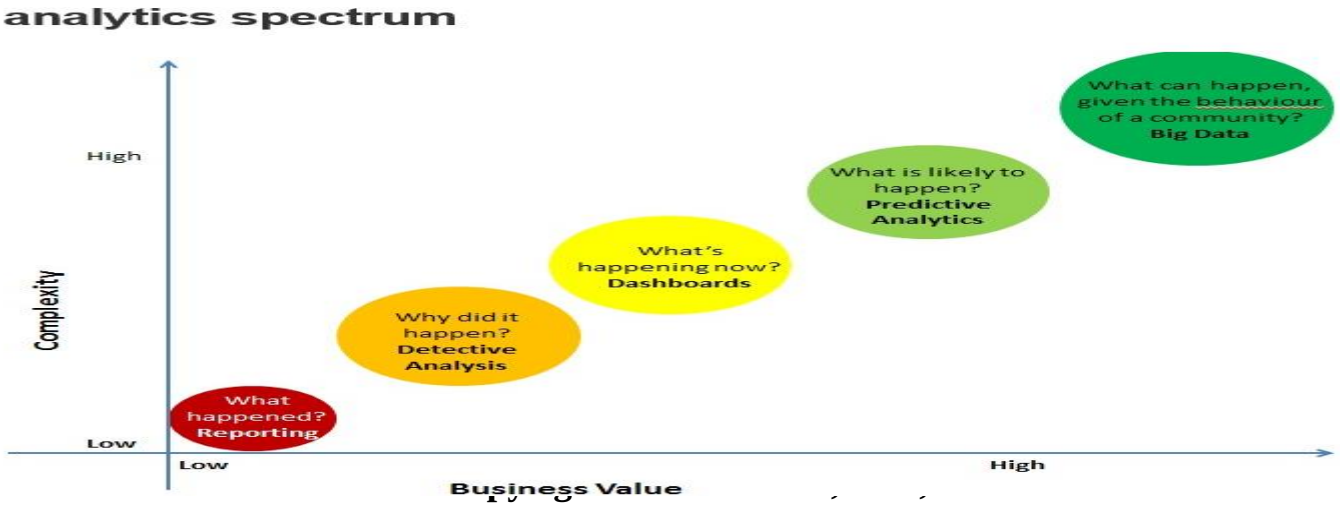
*ShowCalendar - Notepad

File Edit Format View Help

```
<html>
  <head>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript">
      google.charts.load("current", {packages:["calendar"]});
      google.charts.setOnLoadCallback(drawChart);

    function drawChart() {
        var dataTable = new google.visualization.DataTable();
        dataTable.addColumn({ type: 'date', id: 'Date' });
        dataTable.addColumn({ type: 'number', id: 'Sales' });
        dataTable.addRows([
           [ new Date(2019, 0, 31), 50000 ],
           [ new Date(2019, 1, 28), 45000 ],
           [ new Date(2019, 2, 31), 52000 ],
           [ new Date(2019, 3, 30), 55500 ],
           [ new Date(2019, 4, 30), 60000 ],
           [ new Date(2019, 5, 30), 55000 ],
           [ new Date(2019, 6, 30), 47000 ],
           [ new Date(2019, 7, 31), 53000 ],
           [ new Date(2019, 8, 30), 57500 ],
           [ new Date(2019, 9, 30), 62000 ],
           [ new Date(2019, 10, 30), 53000 ],
           [ new Date(2019, 11, 30), 57500 ],
        ]);
        var chart = new google.visualization.Calendar(document.getElementById('chartDiv'));
        var options = { title: "Company Sales", height: 350, };
        chart.draw(dataTable, options);
    }
    </script>
  </head>
  <body>
    <div id="chartDiv" style="width: 1000px; height: 350px;"></div>
  </body>
```

Business and Analytics

- Business now use information to sell more stuff (identify customers, segment customers, up selling, cross selling, etc.)
- Get our message to more customers in a more cost effective manner (segmentation & micro-segmentation)

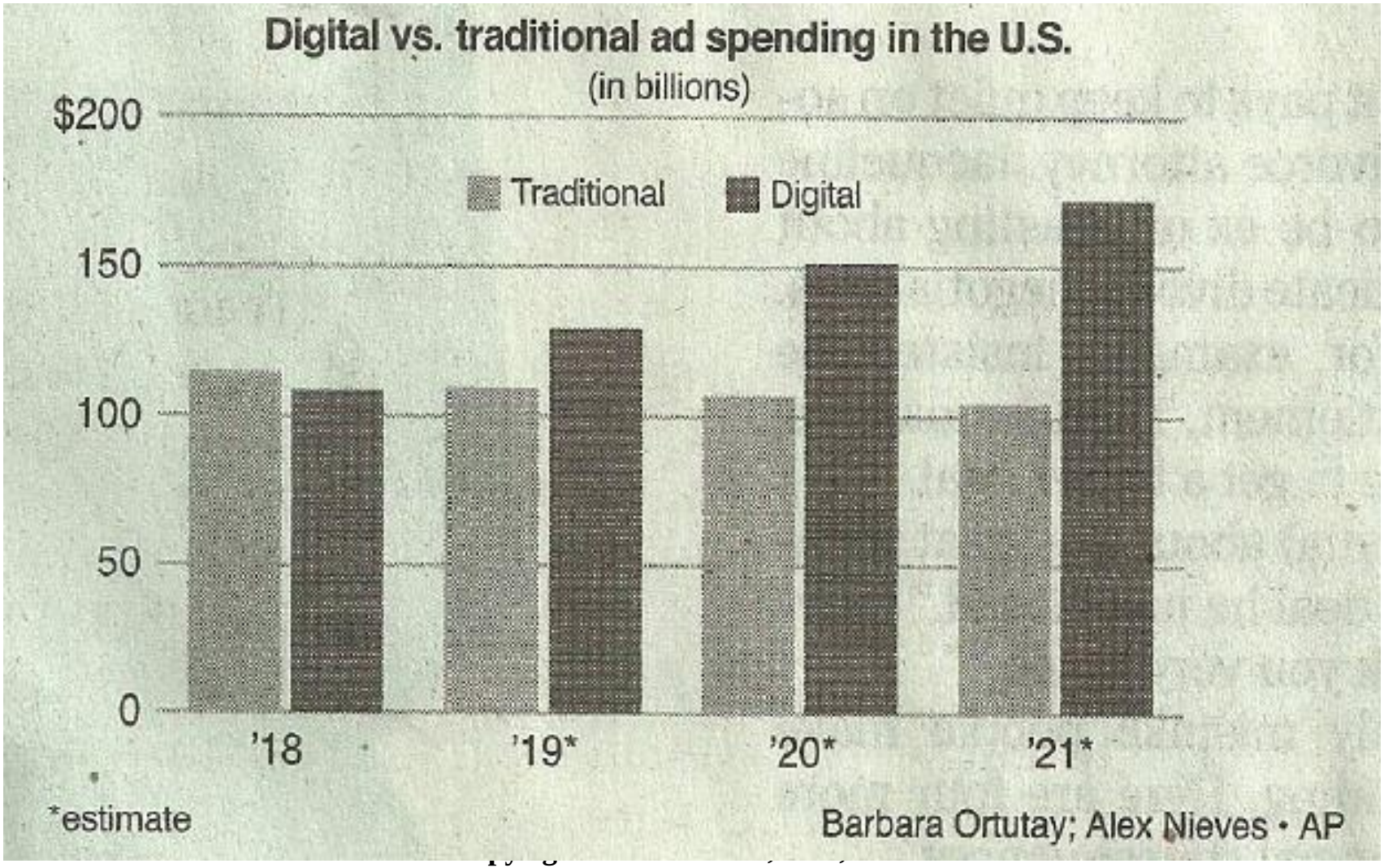


Traditional vs Digital Marketing

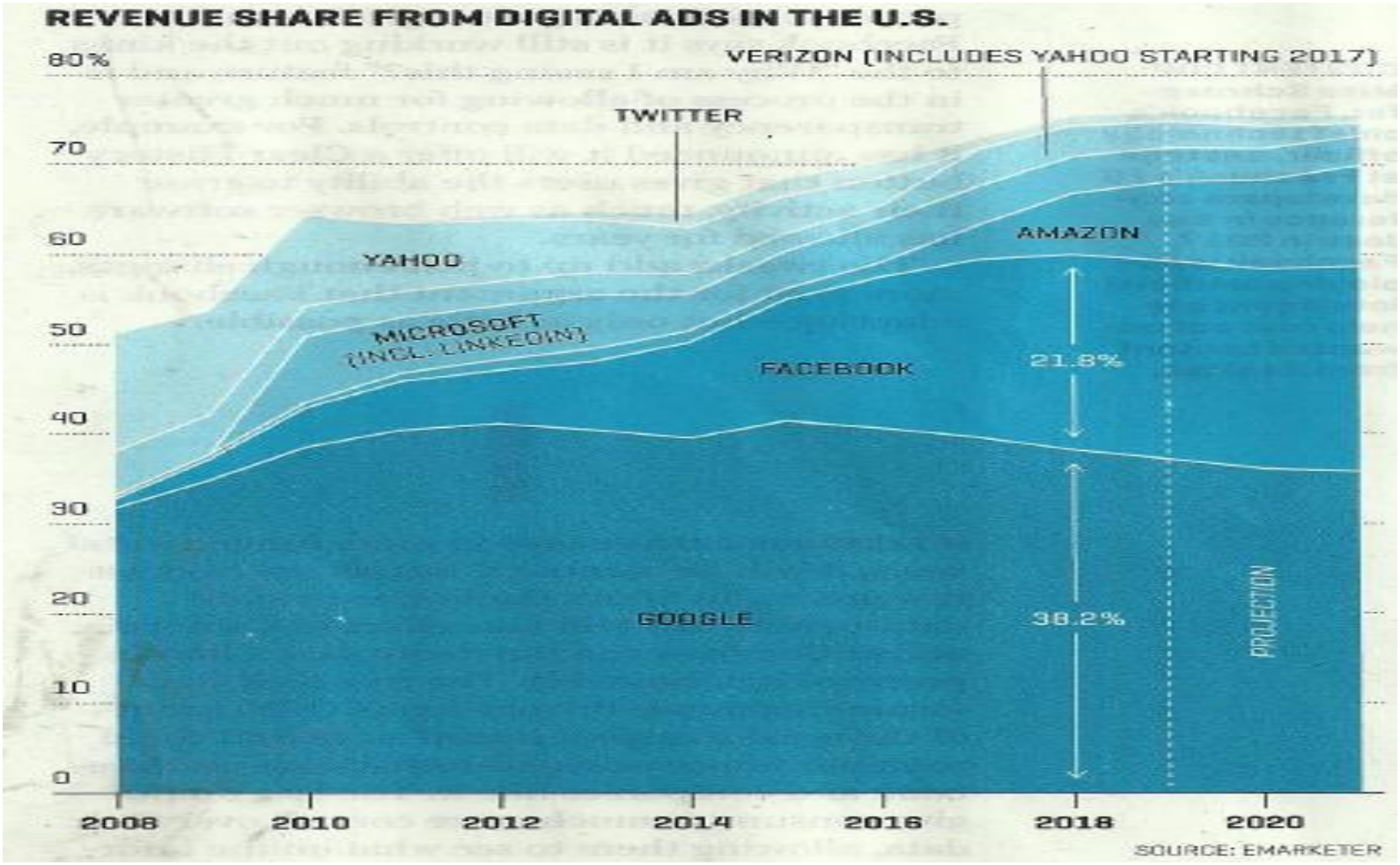
Digital Marketing

- US advertisers now spend more for digital adds than all other media (TV, radio, print, etc.) about $100 billion- growing 19% annually
- Internet advertising is growing at 35% per year versus 5% for TV
- 37% of digital ads go to Google and 21% to Facebook, with Amazon and Twitter getting most of the rest
- Mobile phone and video ads are growing the fastest
- Soon there will be live chats in which robot salespersons swap instant messages with web/smartphone users about products/services

US: Traditional vs Digital Marketing



Digital Marketing Major Players



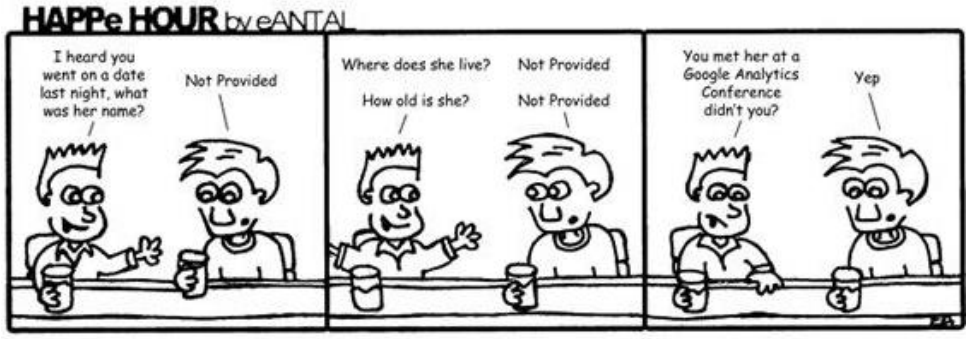
Digital Marketing Environment

Google Analytics

- Google Analytics (GA) allows organizations to generate, analyze, and visualize information about their website hits
- GA helps organization to better focus available resources to generate and retain more customers and more business from these customers
- GA enables organizations to "work smarter" and to generate more success from their digital property

Google Analytics (con't)

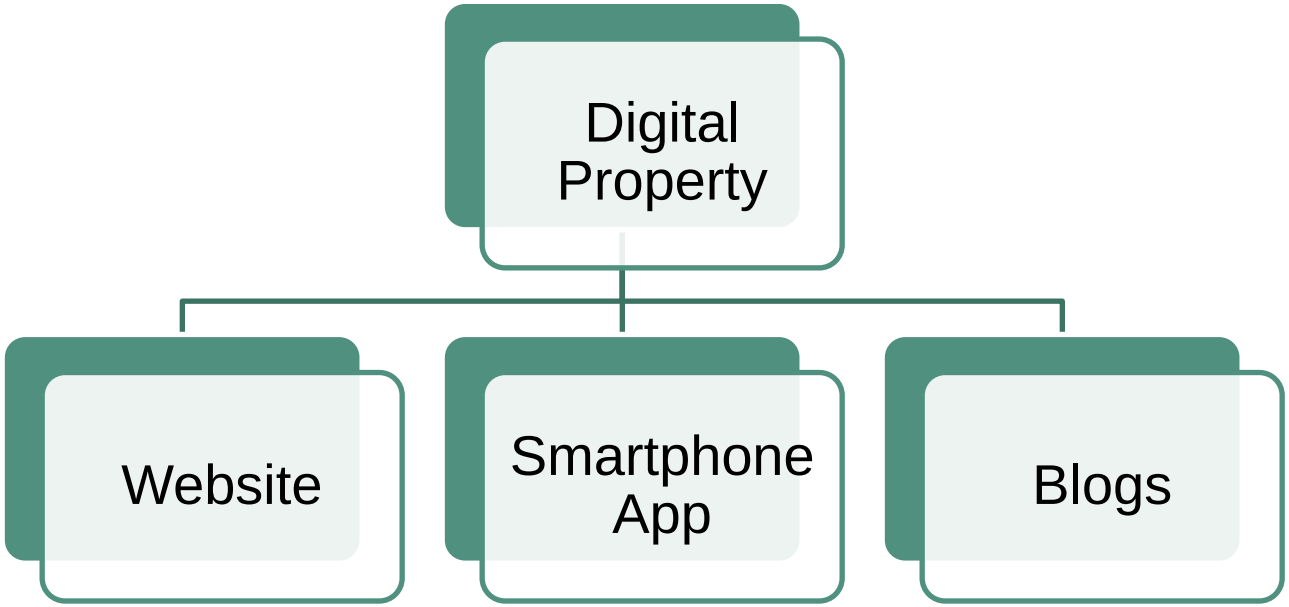
- GA can answer questions such as:
  - How do customers find and enter our website
  - Which content is most traversed, most useful, most consuming
  - How easy is it for visitors to find what they need, to make a purchase, to get support
  - How can visitors' experience be improved
  - How successful are promotions, sales campaigns, etc.
  - How can key customer groups be identified for targeting advertising and promotion **(targeted ads are 3 times more cost effective)**



Google Analytics (con't)

- GA can track the origin of website visitors and relate these origins to page interactions, conversions, purchases, downloads, and other transactions
- GA is free to set up and use as long as the volume level is below a limit (currently 10 million hits per month)
- GA provides for "attribution" where a value is assigned to visitor touchpoints to determine the relative contribution of each touchpoint to a purchase

Google Analytics (con't)



GA can also collect data from any web enabled device including kiosk, point of sale devices, etc. via a standardized API called Measured Protocol

Google Analytics (con't)

- GA uses a "Global Site Tag" which is some **JavaScript** code (gtag.js) that is place in the <HEAD> section of your HTML for each page on your website
- Later more detailed code is inserted in selected pages for more detailed analysis such as purchase conversions

```
<!-- Global site tag (gtag.js) - Google Analytics -->
<script async src="https://www.googletagmanager.com/gtag/js?id=GA_TRACKING_ID"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());

  gtag('config', 'GA_TRACKING_ID');
</script>
```

Organization's GA tracking number

MIS 470 Web Page

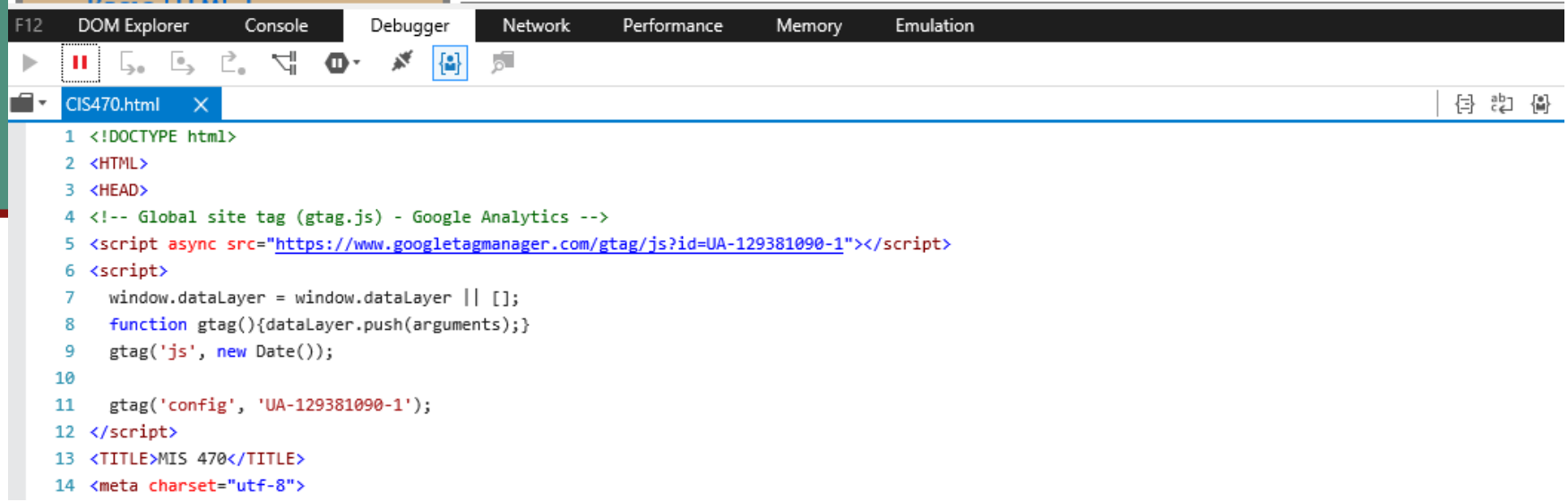
MIS 470

Applications and Internet Programming

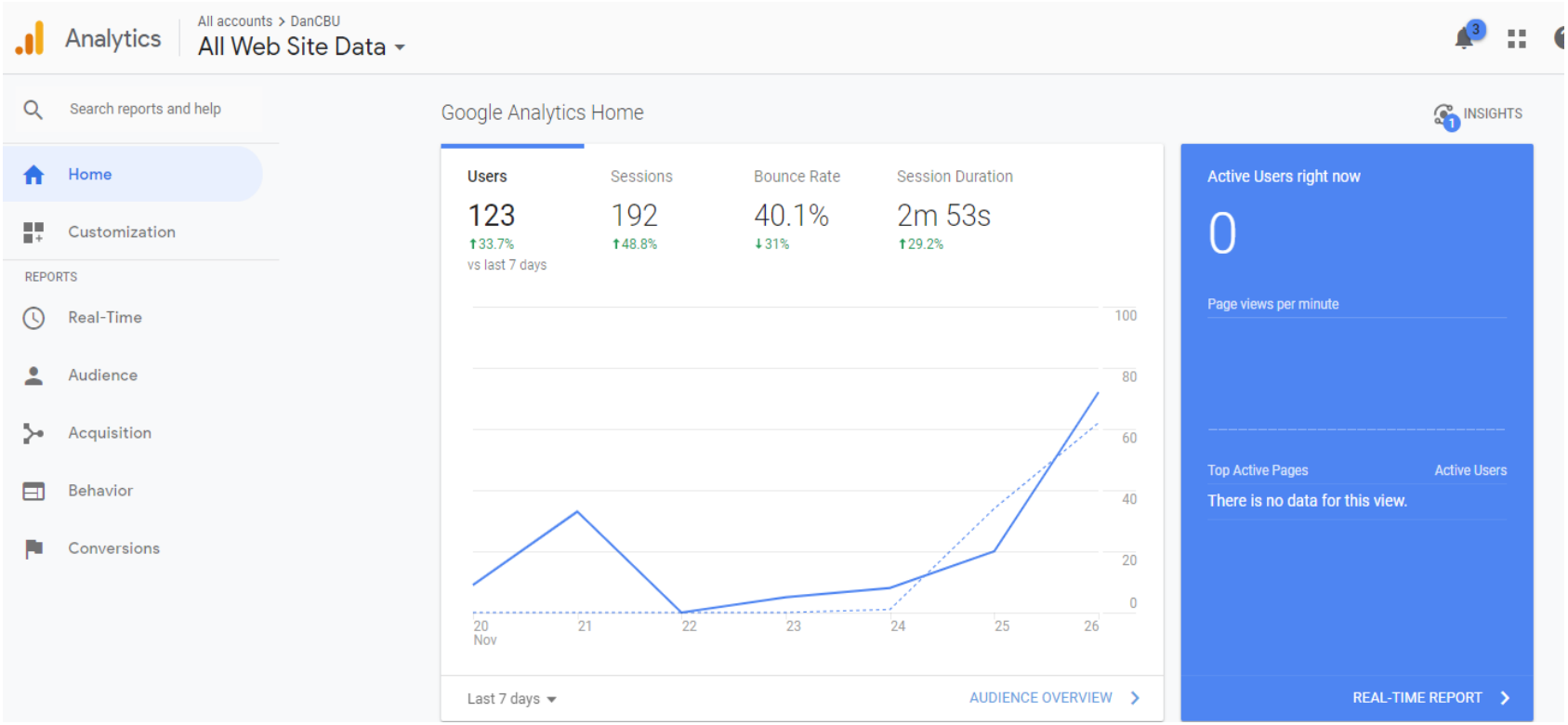
(3 Credits)

Schedule/Content

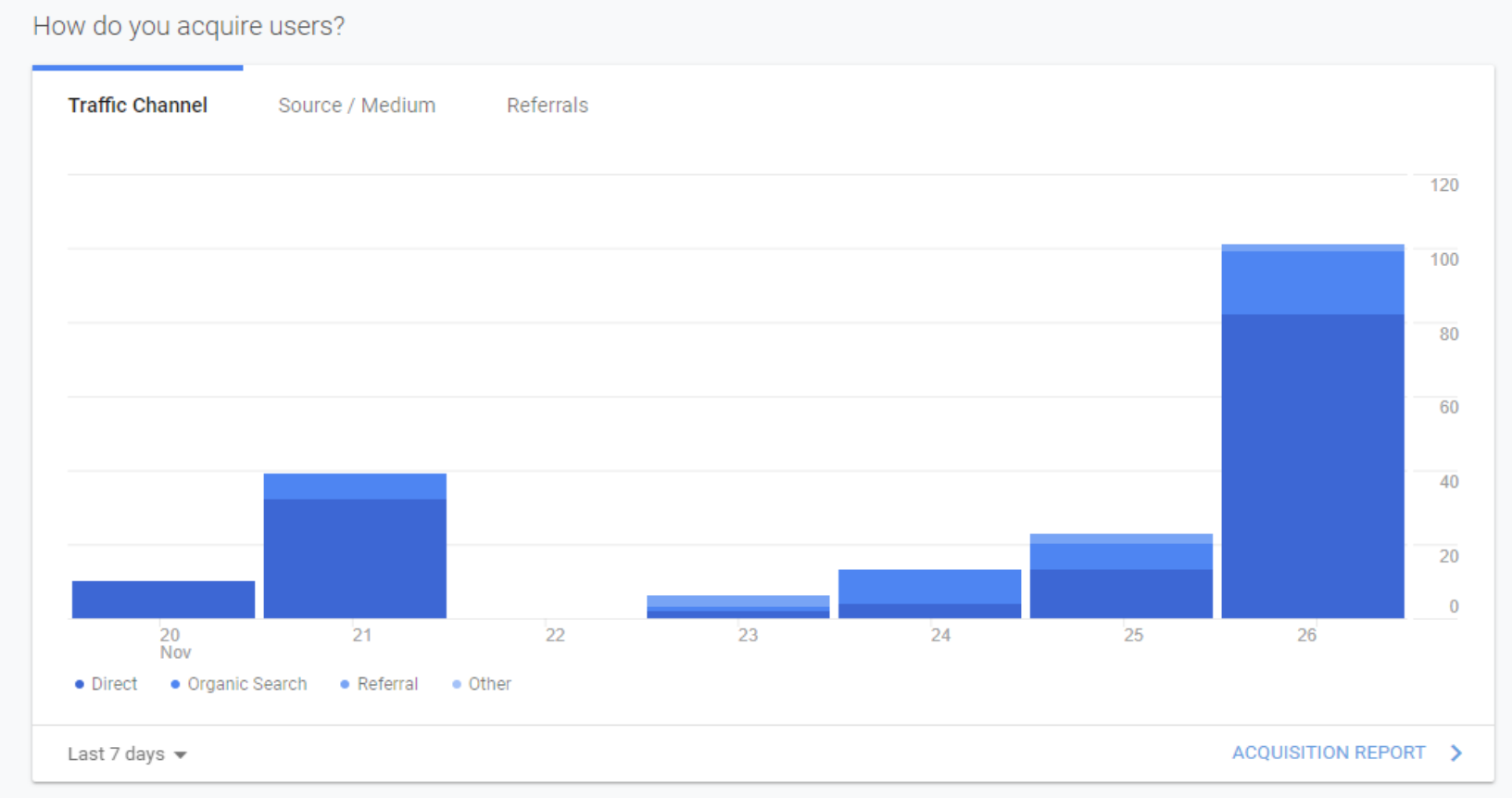
This course introduces modern programming and its application in the internet environment.



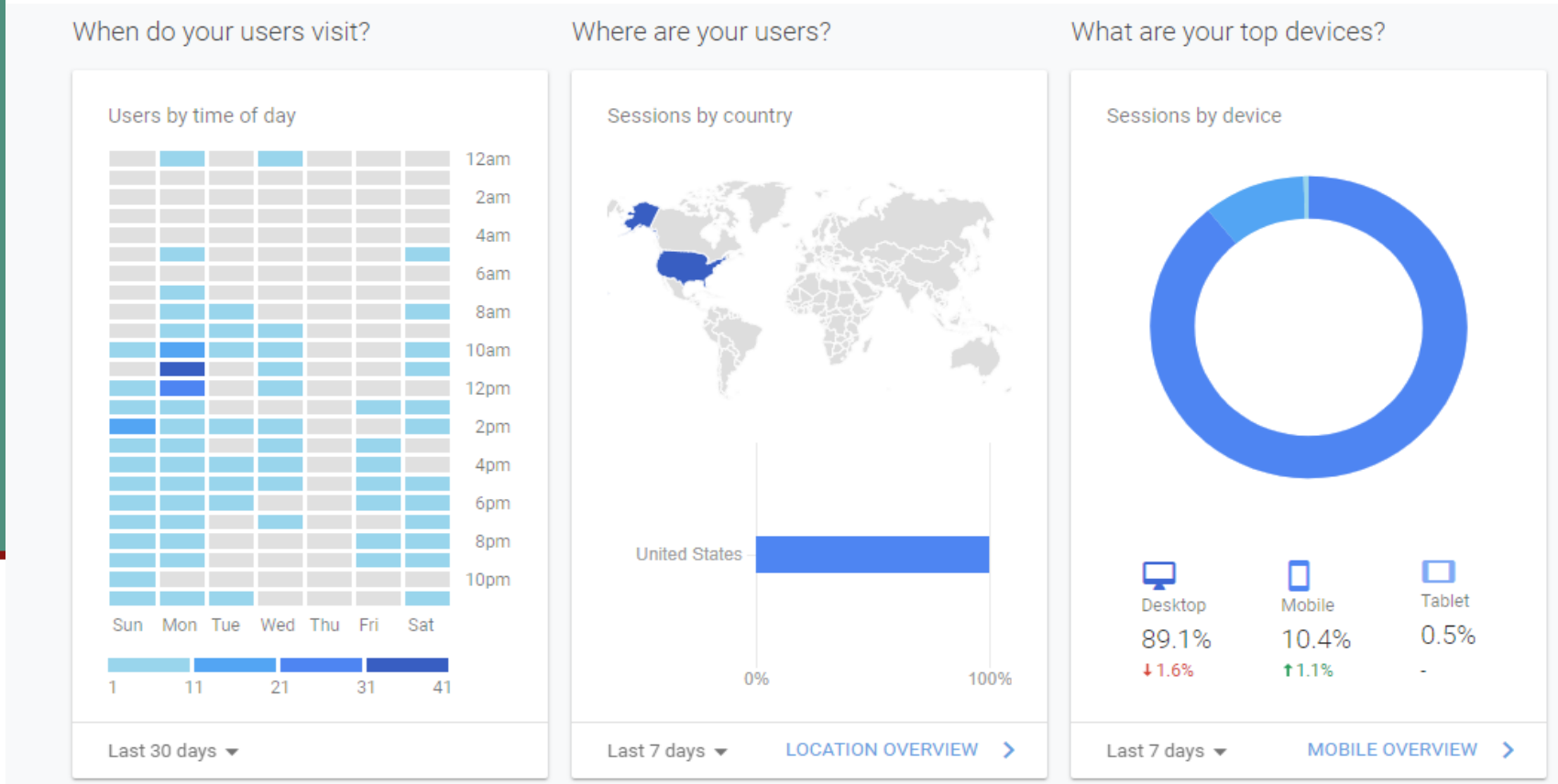
GA Home Page



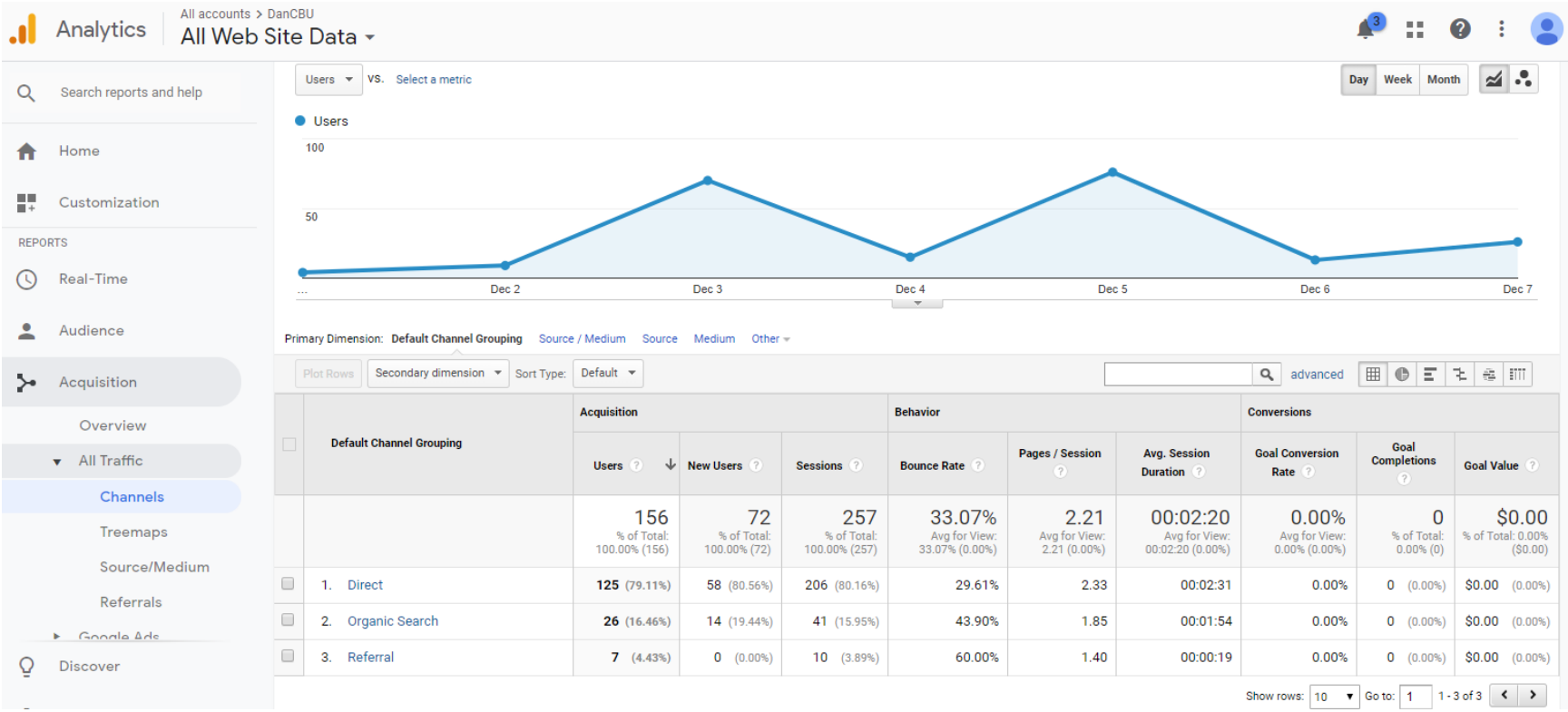
GA Acquire Users [on home page]



GA When/Where/What [on home page]



Acquisitions and Channels

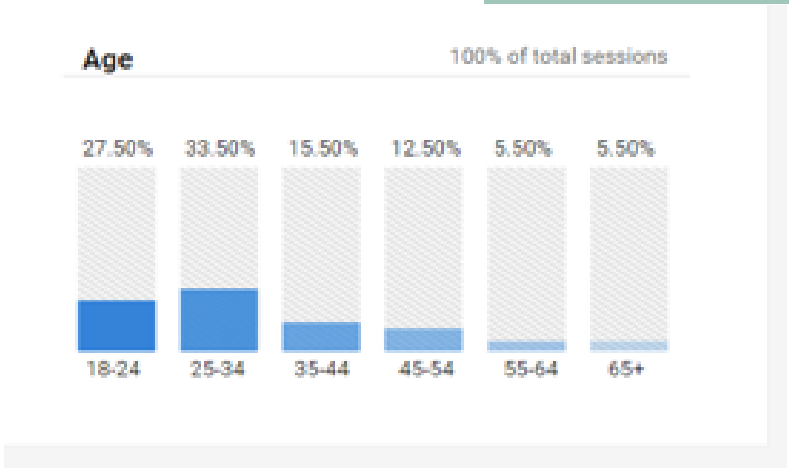


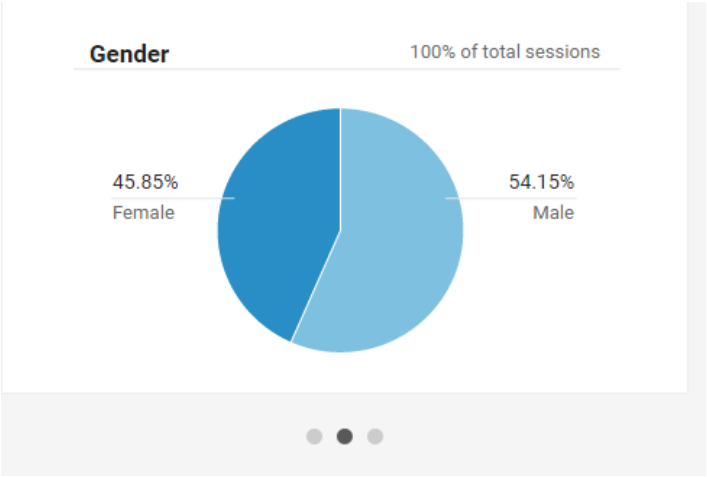
Demographics & Interest

[age, gender, interests]

The Demographics and Interest sections include Overview reports, along with new Age, Gender, and Interest Categories reports.

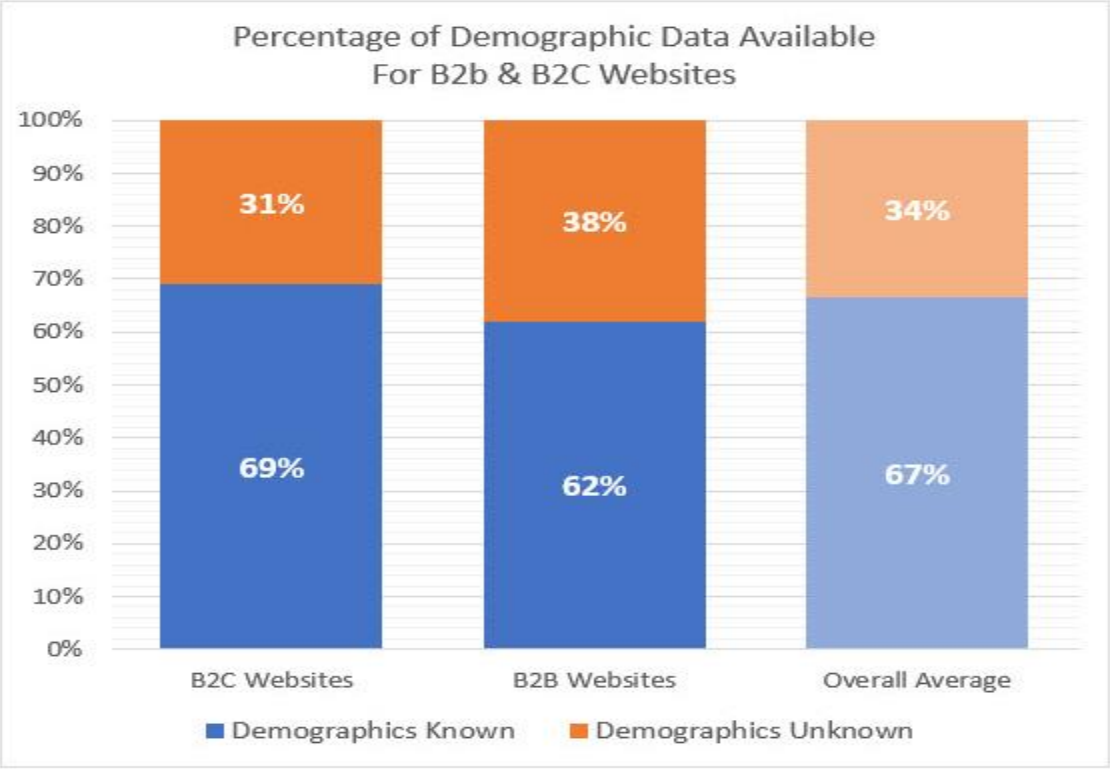
- They allow you to better understand who your users are.
- You can segment the rest of your Analytics data by these same characteristics so you can understand how converting and non-converting users differ (and many other such comparisons).
- These are the same demographics & interest categories used to target ads on the Google Display Network. Use these insights about your users to refine your ad campaign strategies.
- Not all of your users may have demographics associated with them, so these reports may only represent a subset of your users and may not be representative of your overall site composition.



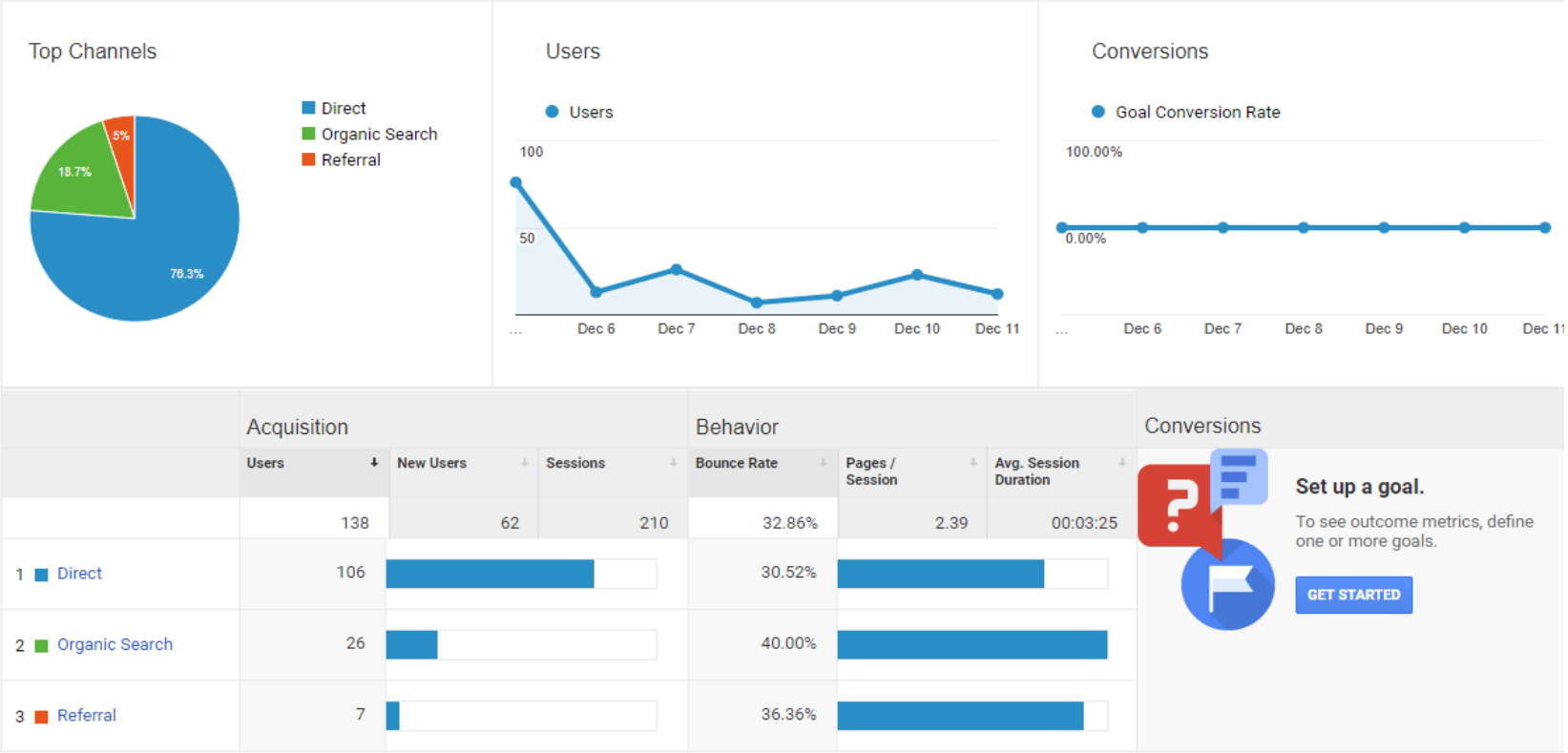


Demographics & Interest (con't)

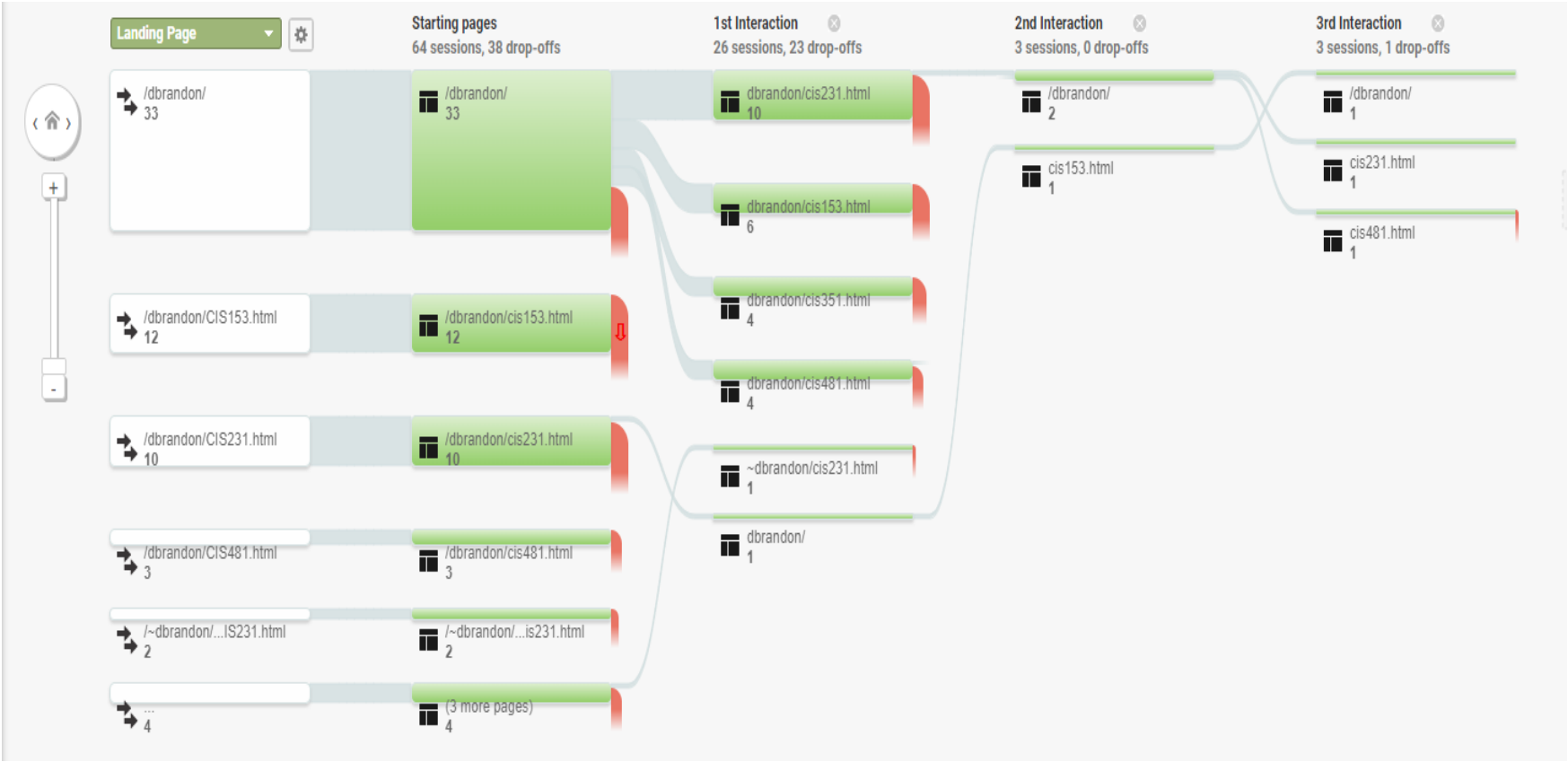
- Estimated accuracy for "known" sites is 80-85%



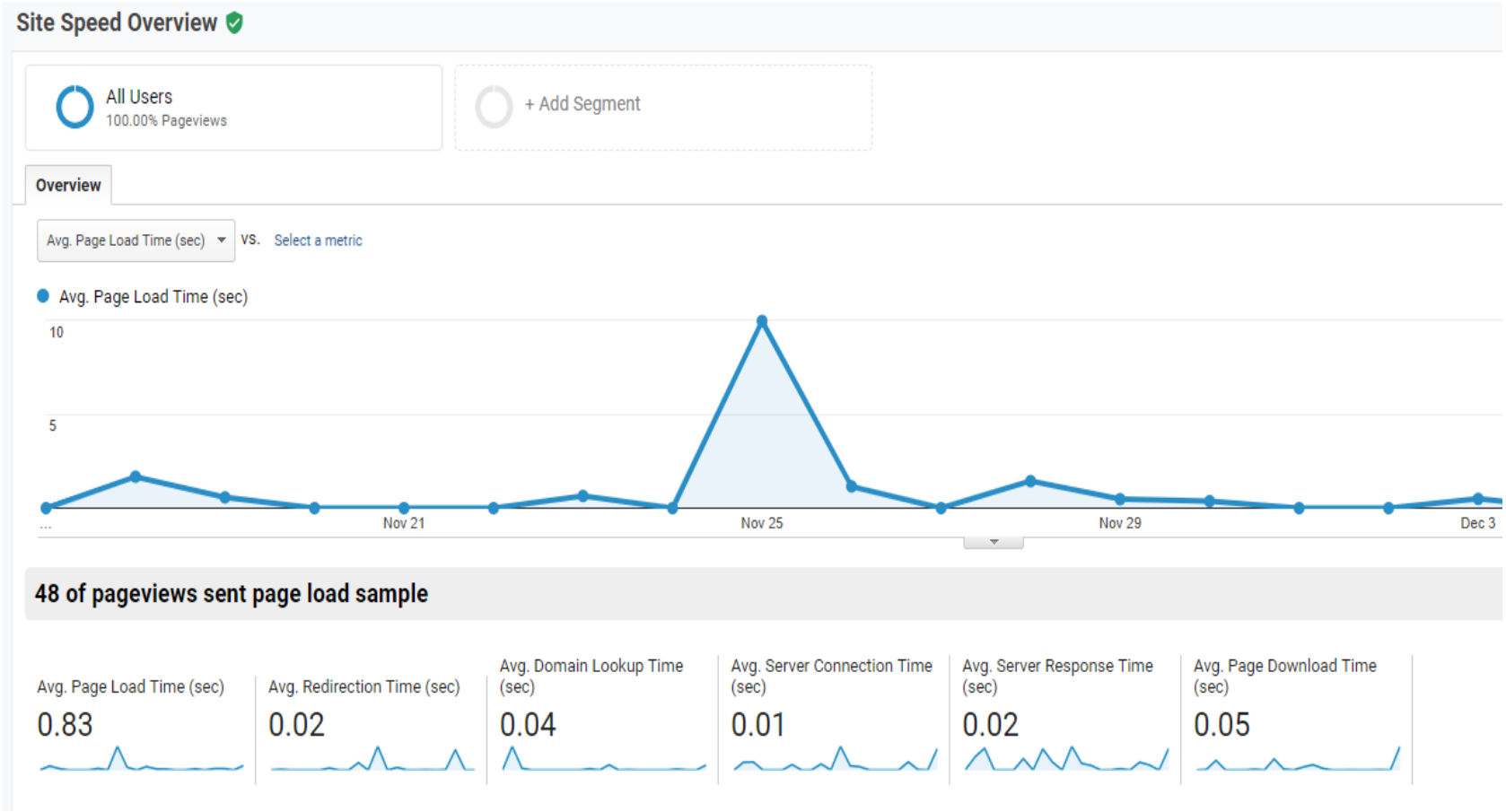
Acquisition



User Flow



Site Speed



Page Timings

Page	Pageviews	Avg. Page Load Time (sec) (compared to site average)
	1,622 % of Total: 100.00% (1,622)	0.83 Avg for View: 0.83 (0.00%)
/dbrandon/	531	10.03%
/dbrandon/CIS153.html	340	-59.33%
/dbrandon/CIS231.html	245	87.76%
/dbrandon/CIS481.html	219	19.28%
/dbrandon/CIS351.html	119	-100.00%
/~dbrandon/CIS231.html	63	36.70%
/~dbrandon/	44	-100.00%
/~dbrandon/CIS153.html	21	-100.00%
/~dbrandon/CIS481.html	19	-100.00%
/	14	-100.00%

Show rows: 10 Go to: 1 1 - 10 of 11

Project Two Deliverables

- Selected industry area – business web site
- Web site design
- Multiple pages
- Common external stylesheet(s) for all pages
- Dynamic effects
- Multimedia
- Rollovers
- JS validation of form data

Sample Student Project

[http://facstaff.cbu.edu/~dbrandon/freight/]

Sample Project

[http://facstaff.cbu.edu/~dbrandon/cleaners/cleaners.html]

Quick Pik & Drop cleaners

Over 10 years of fast and reliable services within the greater Memphis area!!!

Sign up now for FREE Pickup & Delivery

Friday, 07-May-2004 16:02:46 CDT

| Home | Profile | Services | Submit Order | Coupons | Customer

Services | Quick Finds

Welcome to the world of convenience and quality at your front door!!!

Dry cleaning
We are a reliable source for your drycleaning needs.

Laundry
Solutions for busy people who demand fast, reliable and convenience service.

Wash and Fold
We sort it, wash it, dry it, and fold it just like you would do it at home.

Top 10 Reasons to choose Qick Pik & Drop for your drycleaning needs?

1. Convenience, quick and simple
2. Saves you time
3. Our commitment to quality
4. Free pick up and delivery
5. Top-of-the-line service
6. Your satisfactions guarantee
7. We listen
8. Saving money is easy
9. Easy payment
10. Convenience, convenience, convenience!!!

CodeCademy for JavaScript

[http://www.codecademy.com/en/tracks/javascript]

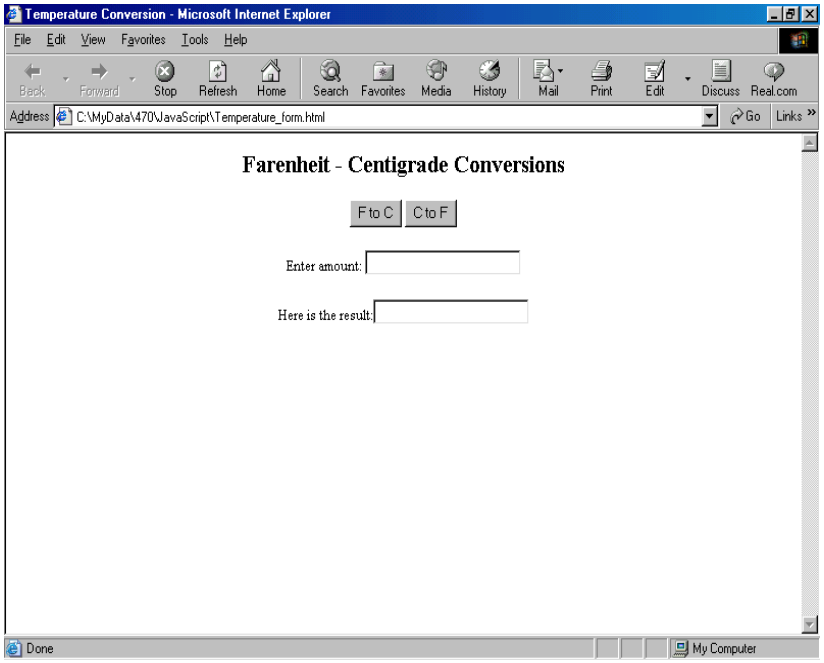
References

- Web Design For Beginners: JavaScript and HTML Form - Vol.2 by V Santhana Krishnan
- Interaction Design with Web Forms: Creating Web and Mobile Environments with HTML5 by Simon Griffin
- Web Forms: The Definitive Guide: Addressing the Challenges of Interactivity in Web and Mobile Environments with HTML5 by Simon Griffin

Homework

[Temperature_form_validity.html]

- Extend your earlier exercise on temperature conversion, to check that the input temperature field is not blank and is numeric ! Also initially set focus to the input field, and reset focus there for errors !
- Hybrid Lesson - Appendix on “cookies”



APPENDIX - Cookies

- Cookies are pieces of data stored in files on users' computers
- They are used to track a users history or "preferences" and/or to maintain "state" between successive pages or CGI calls
- Typical uses are to keep track of items in a "shopping cart", site "dead ends" (where uses lose interest), user preferences for marketing or advertising purposes
- Browsers can be set to block cookies or to accept/reject each cookie

Cookies (con't)

- Cookie location and format is browser dependent; so if a cookie is set by one browser it cannot be read by another
  - In IE, C:\Windows\Cookies\userName@url[x].txt
    - Where x is the number of times visited
- Also a separate directory (or file) is used for each web page so that one web page cannot see another's cookies (note that if you move/rename your web page [on the server] or url, it will not be able to find it's cookies)

Cookies (con't)

- Cookies are pairs of "tags" and "values"; a document has one cookie (document.cookie), with many possible tag/value pairs
- Cookie files can be up to 4K; a defaut limit of 50 cookies per site is normal and 1000 cookies per browser

Cookies (con't)

- Each cookie contains an expiration date, after which the browser will delete it; if no date is set, the cookie is deleted when the session is over
- Dates are expressed in milliseconds from 1/1/1970; the getTime() function of the Date object returns ms past 1/1/1970
- To delete cookies, set the expiration date to a past date

Cookies (con't)

- Cookies can also contain optional information including: path of a CGI program to which cookie contents can be transmitted, the domain to which a cookie can be transmitted, and if this cookie can only be transmitted by HTTPS secure protocol
- Cookies can be set by CGI programs (covered later in course) or JavaScript (covered here)

Creating Cookies

- To create a cookie one needs to write the function that creates the cookie and then the code/event that will call that function. The following function is one way to do it:
  - function setCookie(tag, value, days) {
    - var expireDate = new Date();
    - var expireString = “”;
    - var x = 1000*60*60*24*days;
    - expireDate.setTime(expireDate.getTime()+x);
    - expireString = “expires=“ + expireDate.toGMTString();
    - document.cookie = tag + “=“ + escape(value) + “;” + expireString + “;”
  - }

Getting Cookies

- To read a cookie, one needs to look thru the document.cookie file and find the desired tag, then get the associated value:
  - function getCookie(tag) {
    - var value = null;
    - var c = document.cookie + “;”;
    - var find = tag + “=“;
    - if (c.length > 0) {
      - var begin = c.indexOf(find);
      - if (begin != -1) {
        - begin = begin + find.length;
        - end = c.indexOf(“;”, begin);
        - if (end == -1) end = c.length;
        - value = unescape(c.substring(begin, end));
      - }
    - }
    - return value; }

```
<HTML>
<HEAD><TITLE>Cookies</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
        window.onerror = handleError;

        function getCookie(tag) {
        var value = null;
        var c = document.cookie +";";
        var find = tag + "=";
        if (c.length > 0) {
                var begin = c.indexOf(find);
                if (begin != -1) {
                begin = begin + find.length;
                end = c.indexOf(";", begin);
                if (end == -1)        end = c.length;
                value = unescape(c.substring(begin, end));
                }
        }
        return value;
        }

        function setCookie(tag, value, days) {
        var expireDate = new Date();
        var expString = "";
        var x = 1000*60*60*24*days;
        expireDate.setTime(expireDate.getTime()+x);
        expireString = "expires=" + expireDate.toGMTString();
        document.cookie = tag + "=" + escape(value) + ";" + expireString + ";"
        }
```

```
        function start() {
                var name = getCookie("Name");
                if (name)
                        alert("Hello "  + name + " !");
                else {
                        name = prompt("What is your name ?", "");
                        setCookie("Name", name, 1);
                }
        }
        function handleError (type, url, line) {
                alert("Error: " + type + "At line: " + line);
        }
// -->
</SCRIPT>
</HEAD>
<BODY ONLOAD="start()">
<H1>ABC Company</H1>
</BODY></HTML>
```