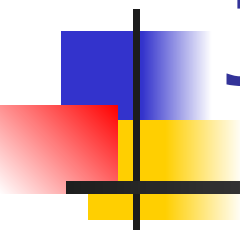
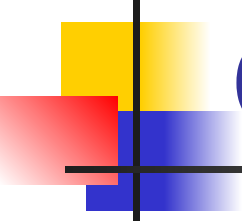


HTML Documents and JavaScript

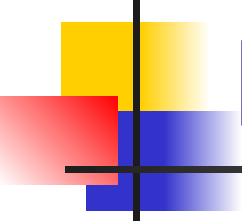


Tom Horton
Alfred C. Weaver
CS453 Electronic Commerce



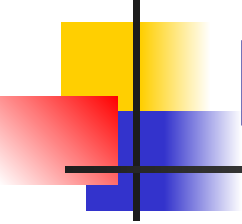
Overview

- Some basic HTML
 - And principles and issues
- W3C Standards that are relevant
 - DOM, XML, XHTML, ECMAScript
- JavaScript introduction
- Your tasks:
 - HTML, JavaScript exercises in VirtualLabs
 - Homework 2 on JavaScript



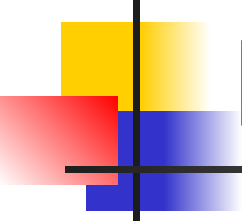
Readings

- Many on-line tutorials
 - www.w3schools.com/Xhtml
 - Other on-line references (report!)
- ~~Our textbook~~
 - ~~Chap. 12 on HTML~~
- Virtual Lab exercises
 - On HTML, JavaScript



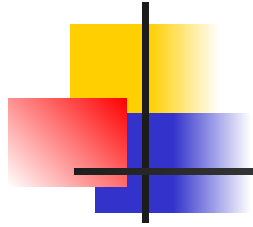
HTML Background

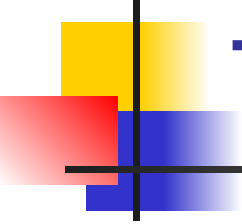
- Many “markup” languages in the past
- SGML: Standard Generalized Markup Language
 - HTML (Hypertext Markup Language) based on SGML
- XML (eXtensible Markup Language) “replaces” SGML
 - XHTML is replacing HTML



Principles

- Distinguish structure from presentation
 - Presentation based on structure
 - Presentation may vary, perhaps based on display characteristics, user-preference, etc.
- People like to ignore this idea
 - E.g. use `<B>` vs. `<EM>`
 - `<font>` tag?
- XML and CSS or XSL

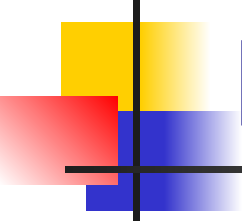




Tags and Elements

- Example of an element:
 <name attr1="attrval">content</name>
- Begin and end tags set off a section of a document
 - Has a semantic property by tag-name
 - Modified by attributes
- "content" can contain other elements
 - Elements nest, don't "overlap"
- Empty-elements: no end tag
 -

 - Note space before />



Basic HTML Structure

- Comments:

`<!-- ... -->`

- Example:

`<html>`

`<head>`

`...`

`</head>`

`<body>`

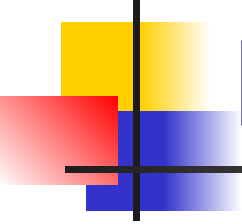
`....`

`</body>`

`</html>`

`<--- title, meta-tags,
etc. (not displayed)`

`<--- main content
(displayed)`



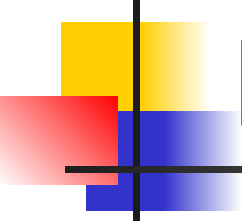
Larger Example

```
<html>
<head>
<title>An Example</title>
</head>
<body>
<h3><hr>An Example</h3>
<p align="left">
  <font face="Comic Sans MS"
    size="4"><b>
    Hello World!</b></font>
</p>
<p align="right">
<font size="5"><u>I am
  21.</u></font>
</p>
<!-- see next column -->
```

```
<p>
  <ol type="I" start=7>
    <li><font
      color=#00FF00>Green</font><
    /li>
    <li>Yellow</li>
    <ul type=square>
      <li>John</li>
      <li>Mike</li>
    </ul>
  </ol>
</p>
</body>
</html>
```

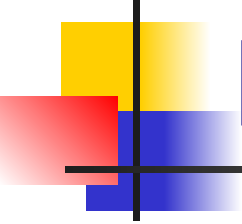
Displays As...





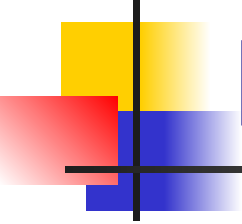
Basic Tags

- Text display:
 - `<em>`, `<strong>`, `<em>`
- Structure:
 - `<h1>`, `<h2>`, `<h3>`
 - `<p>`
 - `<ul>`, `<ol>`, `<blockquote>`
- Attributes:
 - `Align`, `text`, `bgcolor`, etc.



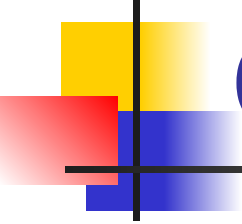
Basic Tags (2)

- Links:
`<a href="...">...</a>`
- Images:
 - `` an empty tag
- Tables
 - Use an editor!
- Forms: later



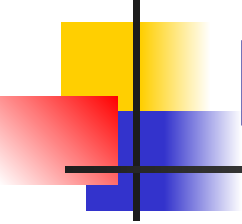
More HTML

- Learn on your own
- You may never code in “raw” HTML
- You may need to tweak HTML files created by a tool
- You will need to understand HTML to code in JavaScript etc.
- You will need to understand HTML to know limitations on how docs on the web can be structured

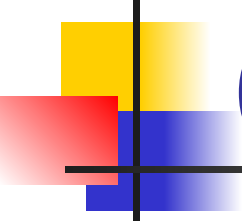


Question:

- You're writing software to process an HTML page
 - A web-browser engine, for example
- What data structure would best represent an HTML document?
 - Why?



Discuss and give me details

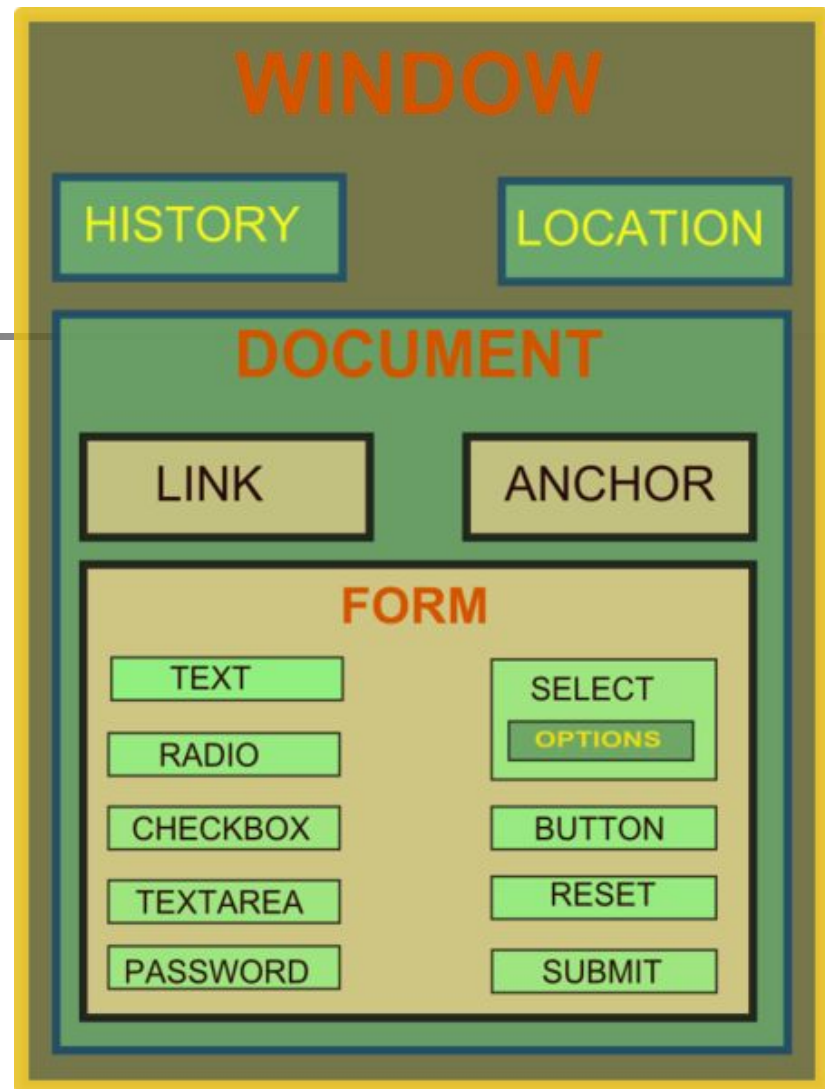


Document Object Model (DOM)

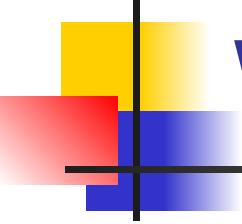
- An model for describing HTML documents (and XML documents)
 - A standard (ok, standards)
 - Independent of browser, language
 - (ok, mostly)
 - A common set of properties/methods to access everything in a web document
- APIs in JavaScript, for Java, etc.

DOM

- You get anything you want from...

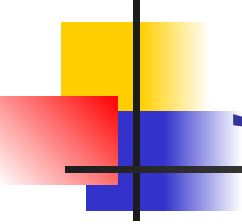


- More info:
http://en.wikipedia.org/wiki/Document_Object_Model



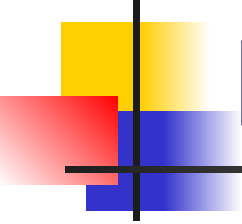
W3C Standards

- XML, XHTML
- CSS, XSL
- XSLT
- DOM
- ECMAScript
- etc



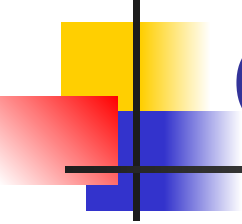
JavaScript

- An example of a “scripting” language that is embedded in HTML documents
 - The browser’s display engine must distinguish from HTML and Script statements
- Others like this:
 - PHP (later in the course)



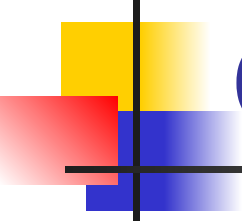
History

- JavaScript created by Netscape
- JScript created by Microsoft
- IE and Netscape renderings are slightly different
- Standardized by European Computer Manufacturers Association (ECMA)
- <http://www.ecma-international.org/publications/standards/Ecma-262.htm>



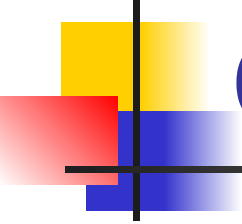
General Format

- `<!doctype ...>`
- `<html>`
- `<Head>`
- `<Title>` Name of web page `</title>`
- `<script type="text/javascript">`
- ...script goes here
- `</script>`
- `</head`
- `<body>`
- ...page body here: text, forms, tables
- ...more JavaScript if needed
- ...onload, onclick, etc. commands here
- `</body>`
- `</html>`



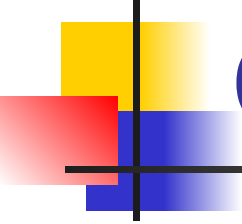
Characteristics

- Case sensitive
- Object oriented
- Produces an HTML document
- Dynamically typed
- Standard operator precedence
- Overloaded operators
- Reserved words



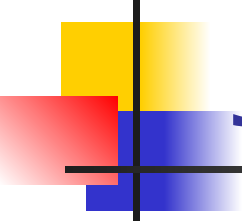
Characteristics

- Division with `/` is not integer division
- Modulus (`%`) is not an integer operator
- `5 / 2` yields `2.5`
- `5.1 / 2.1` yields `2.4285714285714284`
- `5 % 2` yields `1`
- `5.1 % 2.1` yields `0.8999999999999999`



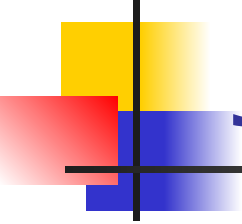
Characteristics

- " and ' can be used in pairs
- Scope rules for variables
- Strings are very common data types
- Rich set of methods available
- Arrays have dynamic length
- Array elements have dynamic type
- Arrays are passed by reference
- Array elements are passed by value



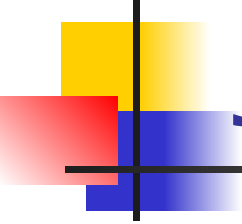
JavaScript Topics

- code placement
- document.writeln
- document tags
- window.alert
- user input/output
- parseInt and parseFloat
- arithmetic
- arithmetic comparisons
- for loops
- while loops
- do-while loops
- if-else
- variable values in tags
- math library
- switch
- break
- labeled break
- continue
- Booleans



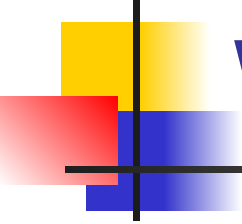
JavaScript Topics

- functions
- random numbers
- rolling dice
- form input
- form output
- submit buttons
- games
- arrays
- searching
- strings
- substrings
- string conversions
- markup methods



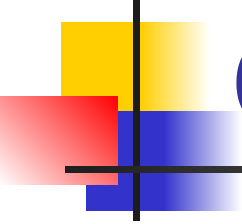
JavaScript's Uses Include:

- “Dynamic” web-pages
 - What’s DHTML? (in a second)
- Image manipulation
 - Swapping, rollovers, slide shows, etc.
- Date, time stuff (e.g. clocks, calendars)
- HTML forms processing
 - Verifying input; writing output to fields
- Cookies



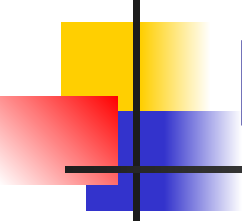
What's DHTML?

- Purpose: make dynamic / interactive web-pages on the client side
- Use of a collection of technologies together to do this, including
 - Markup language (HTML, XML, etc.)
 - Scripting language (JavaScript, etc.)
 - Presentation language (CSS etc.)



Other References

- CS453 Virtual Lab exercises
- *The Web Wizard's Guide To JavaScript*, Steven Estrella, Addison-Wesley
- *JavaScript for the World Wide Web*, Gesing and Schneider, Peachpit Press
- <http://www.w3schools.com/js/>
- www.javascript.com
- E-books in UVa's Safari On-line Books:
<http://proquest.safaribooksonline.com/search>



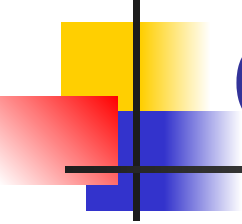
Browser Compatability

- Use of:

```
<script type="text/javascript"  
language="javascript" >  
<!--
```

```
// ends script hiding -->  
</script>
```

- "language=" for pre IE5 and NS6
- Comment for very old browsers (e.g. IE2)
 - BTW, comments in HTML vs. in JavaScript



Organization of JavaScript

- Create functions (non-OO style)

- Define in header
- Or load a .js file in header:

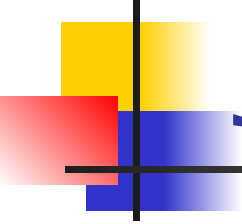
```
<script type="text/javascript"  
language="javascript" src="mylib.js">
```

- Functions called in <BODY>

- Often in response to events, e.g.

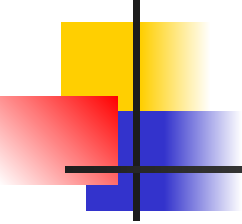
```
<input type="button"... onclick="myFunc (...) ; ">
```

- Global variables



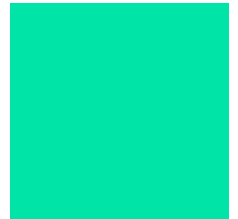
JavaScript

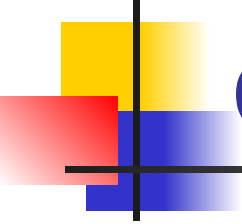
- Programming by example



document.writeln

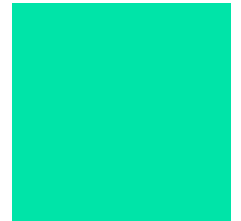
```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<!-- Welcome to JavaScript -->
<HEAD>
<TITLE> Welcome to JavaScript </TITLE>
<SCRIPT TYPE="text/javascript">
    document.writeln( "<FONT COLOR='magenta'><H1>Welcome to ",
        "JavaScript Programming!</H1></FONT>" );
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

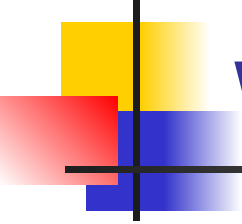




document.write

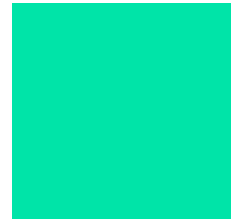
```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE> Using document.write </TITLE>
<SCRIPT TYPE="text/javascript">
    document.write ( "<H1>Welcome to ");
    document.writeln( "JavaScript Programming!</H1>" );
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```





window.alert

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE> Using window.alert </TITLE>
<SCRIPT TYPE="text/javascript">
    window.alert( "Welcome to\nJavaScript\nProgramming!" );
</SCRIPT>
</HEAD>
<BODY>
<P>Click Refresh (or Reload) to run this script again.</P>
</BODY>
</HTML>
```

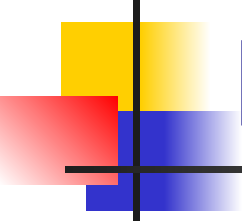




User input/output

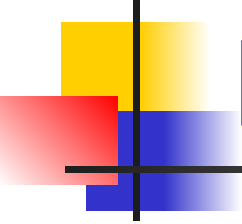
```
<SCRIPT TYPE="text/javascript">
    var firstNumber, // first string entered by user
        secondNumber, // second string entered by user
        number1,      // first number to add
        number2,      // second number to add
        sum;          // sum of number1 and number2
    // read in first number from user as a string
    firstNumber = window.prompt("Enter first integer", "0" );
    // read in second number from user as a string
    secondNumber = window.prompt( "Enter second integer", "0" );
    // convert numbers from strings to integers
    firstNumber = parseInt(firstNumber);
    number2 = parseInt( secondNumber );
    // add the numbers
    sum = firstNumber + number2;
    // display the results
    document.writeln( "<H1>The sum is " + sum + "</H1>" );
</SCRIPT>
```





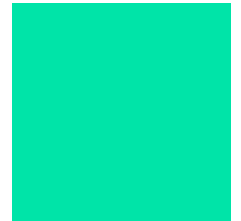
Functions

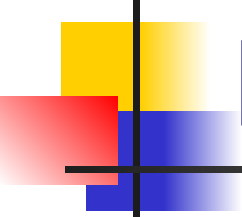
```
<SCRIPT TYPE = "text/javascript">
    var input1 = window.prompt( "Enter first number", "0" );
    var input2 = window.prompt( "Enter second number", "0" );
    var input3 = window.prompt( "Enter third number", "0" );
    var value1 = parseFloat( input1 );
    var value2 = parseFloat( input2 );
    var value3 = parseFloat( input3 );
    var maxValue = maximum( value1, value2, value3 );
    document.writeln( "First number: " + value1 +
        "<BR>Second number: " + value2 +
        "<BR>Third number: " + value3 +
        "<BR>Maximum is: " + maxValue );
    // maximum method definition (called from above)
    function maximum( x, y, z ) {
        return Math.max( x, Math.max( y, z ) );
    }
</SCRIPT>
```



Random Numbers

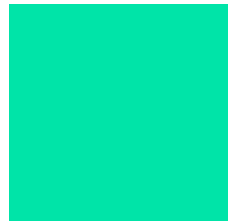
```
<SCRIPT TYPE="text/javascript">
  var value;
  document.writeln( "<H1>Random Numbers</H1>" +
    "<TABLE BORDER = '1' WIDTH = '50%'><TR>" );
  for ( var i = 1; i <= 20; i++ ) {
    value = Math.floor( 1 + Math.random() * 6 );
    document.writeln( "<TD>" + value + "</TD>" );
    if ( i % 5 == 0 && i != 20 )
      document.writeln( "</TR><TR>" );
  }
  document.writeln( "</TR></TABLE>" );
</SCRIPT>
```

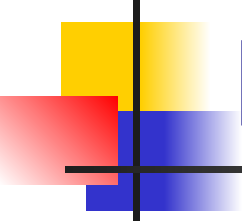




Roll the Die

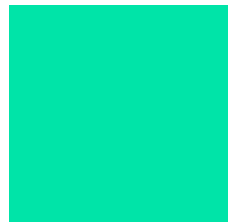
```
<SCRIPT TYPE="text/javascript">
  var frequency1 = 0, frequency2 = 0,
  frequency3 = 0, frequency4 = 0,
  frequency5 = 0, frequency6 = 0, face;
  // summarize results
  for ( var roll = 1; roll <= 6000; ++roll ) {
    face = Math.floor( 1 + Math.random() * 6 );
    switch ( face ) {
      case 1: ++frequency1; break;
      case 2: ++frequency2; break;
      case 3: ++frequency3; break;
      case 4: ++frequency4; break;
      case 5: ++frequency5; break;
      case 6: ++frequency6; break;
    }
  }
  document.writeln( "<TABLE BORDER = '1' WIDTH = '50%'">" ); .....
```

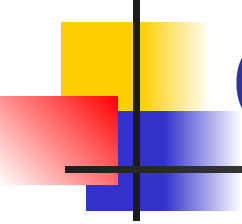




Rules of Craps

- First roll:
 - 7 or 11 is a win
 - 2, 3, or 12 is a lose
 - otherwise, roll becomes your point
- Subsequent rolls:
 - rolling your point is a win
 - 7 or 11 is a lose
 - otherwise continue to roll





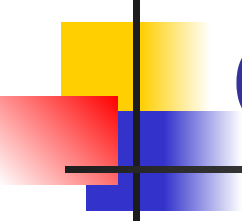
Craps

```
<SCRIPT TYPE="text/javascript">  
// variables used to test the state of the game  
var WON = 0, LOST = 1, CONTINUE_ROLLING = 2;  
// other variables used in program  
var firstRoll = true,    // true if first roll  
    sumOfDice = 0,       // sum of the dice  
    myPoint = 0,         // point if no win/loss on first roll  
    gameStatus = CONTINUE_ROLLING; // game not over yet
```



Craps

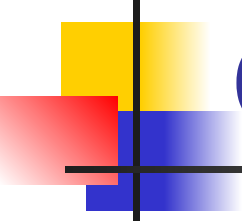
```
// process one roll of the dice
function play() {
  if ( firstRoll ) {
    // first roll of the dice
    sumOfDice = rollDice();
    switch ( sumOfDice ) {
      case 7: case 11:
        // win on first roll
        gameStatus = WON;
        document.craps.point.value = ""; // clear point field
        break;
      case 2: case 3: case 12:
        // lose on first roll
        gameStatus = LOST;
        document.craps.point.value = ""; // clear point field
        break;
```



Craps

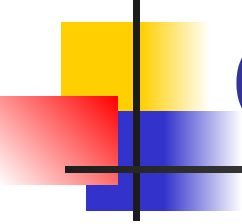
default:

```
// remember point
gameStatus = CONTINUE_ROLLING;
myPoint = sumOfDice;
document.craps.point.value = myPoint;
firstRoll = false;
}
}
else {
    sumOfDice = rollDice();
    if ( sumOfDice == myPoint ) gameStatus = WON;
    else if ( sumOfDice == 7 ) gameStatus = LOST;
}
```



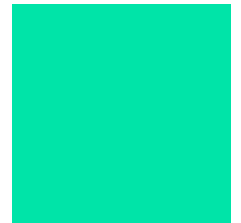
Craps

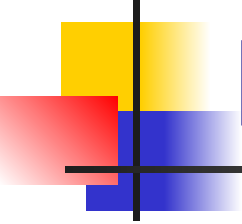
```
if ( gameStatus == CONTINUE_ROLLING ) window.alert ("Roll again");
else {
    if ( gameStatus == WON ) {
        window.alert ("Player wins. " + "Click Roll Dice to play again.");
        document.craps.point.value = " ";
    }
    else {
        window.alert ("Player loses. " + "Click Roll Dice to play again.");
        document.craps.point.value = " ";
    }
    firstRoll = true;
}
}
```



Craps

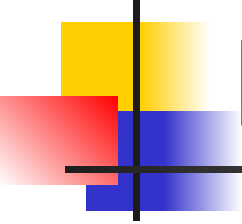
```
// roll the dice
function rollDice() {
  var die1, die2, workSum;
  die1 = Math.floor( 1 + Math.random() * 6 );
  die2 = Math.floor( 1 + Math.random() * 6 );
  workSum = die1 + die2;
  document.craps.firstDie.value = die1;
  document.craps.secondDie.value = die2;
  document.craps.sum.value = workSum;
  return workSum;
}
</SCRIPT>
```





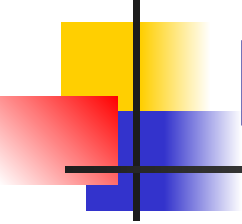
Poker Hand

```
<SCRIPT TYPE="text/javascript">
function rand1toN(N) {
    return Math.floor( 1+Math.random()*N );
}
function dealcard(card) {
    var rank = new Array(0,"A","2","3","4","5","6","7",
        "8","9","T","J","Q","K");
    var suit = new Array(0, "Spades", "Hearts", "Diamonds", "Clubs");
    card[0] = rank[rand1toN(13)];
    card[1] = suit[rand1toN(4)];
}
```



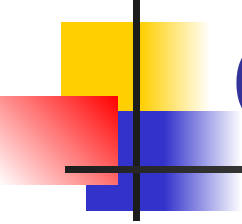
Poker Hand

```
var card = new Array(2);
var player = new Array(10);
var dealer = new Array(10);
for (var i=0; i<=4; i++) {
    dealcard(card);
    player[i*2] = card[0];
    player[i*2+1] = card[1];
    dealcard(card);
    dealer[i*2] = card[0];
    dealer[i*2+1] = card[1];
}
```



Poker Hand

```
document.writeln("<H1> PLAYER </H1>");
document.writeln("<TABLE BORDER='1' >");
for (var i=0; i<=4; i++) {
    document.writeln("<TR><TD><P>" + player[i*2] + "</TD>"
        + "<TD><P>" + player[i*2+1] + "</TD></TR>");
}
document.writeln("</TABLE> </HTML>");
</SCRIPT>
```



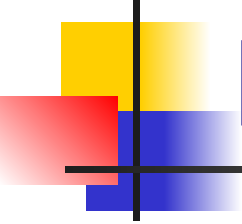
Character Processing

```
<SCRIPT TYPE="text/javascript">
var s = "ZEBRA";
var s2 = "AbCdEfG";
document.writeln( "<P> Character at index 0 in '" +
    s + "' is " + s.charAt( 0 ) );
document.writeln( "<BR>Character code at index 0 in '" +
    s + "' is " + s.charCodeAt( 0 ) + "</P>" );
document.writeln( "<P>" + String.fromCharCode( 87, 79, 82, 68 ) +
    "' contains character codes 87, 79, 82 and 68</P>" );
document.writeln( "<P>" + s2 + "' in lowercase is '" +
    s2.toLowerCase() + "'" );
document.writeln( "<BR>" + s2 + "' in uppercase is '" +
    s2.toUpperCase() + "'</P>" );
</SCRIPT>
```



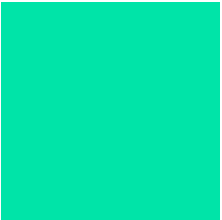
Dates and Times

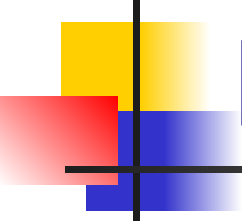
```
<SCRIPT LANGUAGE = "JavaScript">
var current = new Date();
document.writeln(current);
document.writeln( "<H1>String representations and valueOf</H1>" );
document.writeln( "toString: " + current.toString() +
    "<BR>toLocaleString: " + current.toLocaleString() +
    "<BR>toUTCString: " + current.toUTCString() +
    "<BR>valueOf: " + current.valueOf() );
document.writeln( "<H1>Get methods for local time zone</H1>" );
document.writeln( "getDate: " + current.getDate() +
    "<BR>getDay: " + current.getDay() + "<BR>getMonth: " +
    current.getMonth() + "<BR>getFullYear: " + current.getFullYear() +
    "<BR>getTime: " + current.getTime() + "<BR>getHours: " +
    current.getHours() + "<BR>getMinutes: " + current.getMinutes() +
    "<BR>getSeconds: " + current.getSeconds() + "<BR>getMilliseconds: " +
    current.getMilliseconds() + "<BR>getTimezoneOffset: " +
    current.getTimezoneOffset() );
```



Dates and Times

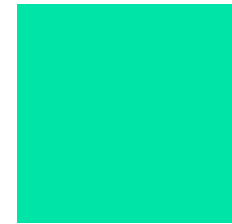
```
document.writeln( "<H1>Specifying arguments for a new Date</H1>" );  
var anotherDate = new Date( 1999, 2, 18, 1, 5, 3, 9 );  
document.writeln( "Date: " + anotherDate );  
document.writeln( "<H1>Set methods for local time zone</H1>" );  
anotherDate.setDate( 31 );  
anotherDate.setMonth( 11 );  
anotherDate.setFullYear( 1999 );  
anotherDate.setHours( 23 );  
anotherDate.setMinutes( 59 );  
anotherDate.setSeconds( 59 );  
document.writeln( "Modified date: " + anotherDate );  
</SCRIPT>
```

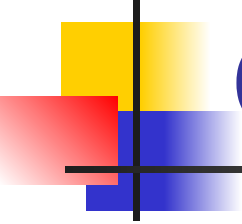




Radio buttons

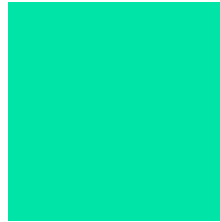
- Assure that at least one radio button is clicked before taking action

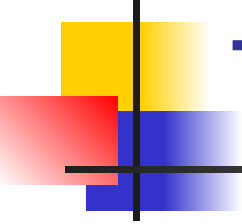




Checkboxes

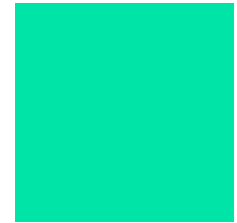
- Respond to selections made with checkboxes

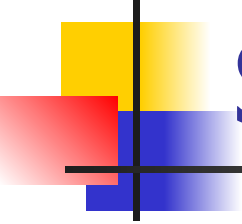




Textboxes

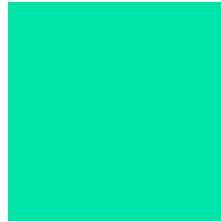
- Detecting an empty textbox

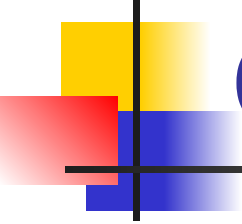




Self-grading Tests

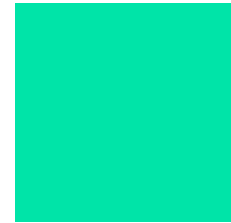
- Collecting and evaluating answers to questions

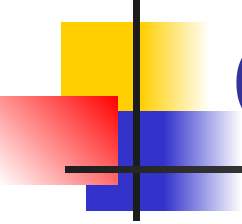




Character String Processing

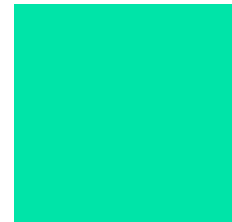
- Validate an email address

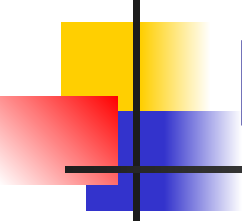




Cookies

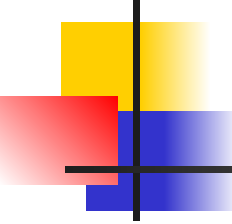
- Write a cookie on the client's device





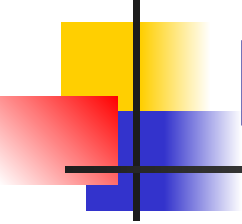
Events

- JavaScript can execute a statement (typically, call a function) when an event occurs
- `<... oneventname="javascript stmt;">`
- `<BODY ... ONLOAD="func();">`
- `<INPUT TYPE="submit" ... ONSUBMIT="f();">`



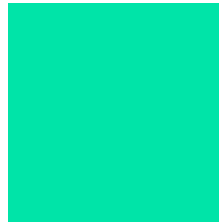
Events

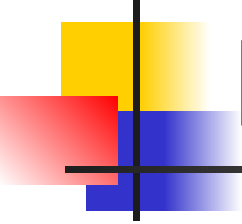
- *onsubmit* - call when submit button is clicked
- *onclick* - call when this button is clicked
- *onreset* - call when the reset button is clicked
- *onload* - call after page loads
- *onmouseover* - call when mouse pointer enters image area
- *onmouseout* - call when mouse pointer leaves image area
- *onfocus* - call when control receives focus
- *onblur* - call when a control loses focus
- *onchange* - call when a control loses focus and the value of its contents has changed
- many more



Mouse Events

- Illustrate *onmouseover* and *onmouseout*

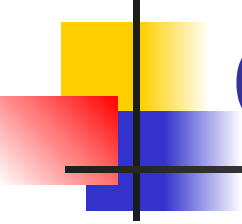




Handling Time

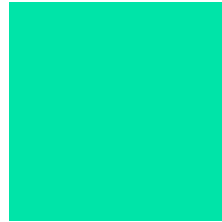
- Create a simple JavaScript clock

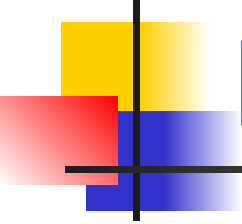




Controlling Time

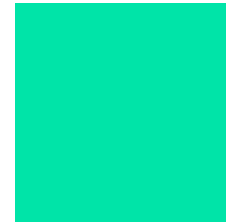
- Turn a clock on and off and format the time string

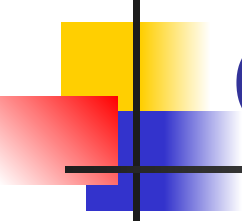




Handling Images

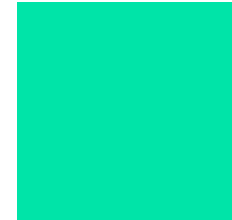
- Create a slide show

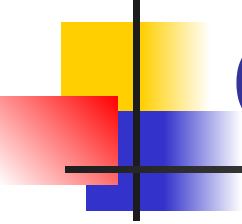




Generate Real-Time Data

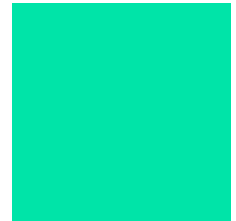
- Simulate monitoring real-time information from a device

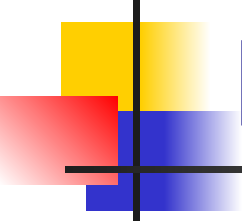




Continuous Update

- Gather data synchronously using the clock as the event generator





End of Examples
