Basic web page structure:

```
<!doctype html>
<!-- Name, date, citations, and assign # -->
<html>
 <head>
   <title> Title for FRAME </title>
   <!--function definitions -->
   <script>
       // JS code here
   </script>
 </head>
 <body>
   Stuff that appears in the page.
 </body>
</html>
```

Some HTML Elements (**/>** implies no end tag)

Tag	**Reqd Attribs**	**Other Attribs**
<p>		id, style
Basic text formatting, including text-align		
<br />		
Start on a new line (of text)		
<h1>		
A heading (title). <h1>…<h6>		
<div>		id, style
Create a vertical "block" on the page. Often used to apply text-align, color, or other styles.		
<span>		id, style
A "horizontal <div>". May apply to a word.		
<a>	**href**	target
A hyperlink. "http://" reqd for non-local link.		
<img />	**src, alt**	height, width, align
Insert an image. Src may be local or full URL. In_line unless aligned (affects text wrapping).		
<ul>		style
Bulleted list. Style sets type of bullet.		
<ol>		style
Numbered list. Styles sets type of numbers (iii)		
<li>		
One list item in <ul> or <ol>. May include an entire sub-list inside <li> … </li>		
<table>	**summary**	style, align
Contains <tr>'s. Summary is for screen readers. Align controls where table appears horizontally. Borders are set using styles.		
<tr>		style, align
Contains <td>'s. Defines one row in table. Number of <tr>'s defines number of rows in table.		
<td>		style, align
Contains contents of one cell in the table. Max number of <td>'s in any row sets number of columns in table. Acts like a "web page in a box".		
<input>	**type**	id, width, value
Types: text & button. Generally used for input, but may also be used for display of output.		
textarea	Id, rows, cols	</textarea>

<b>, <*i*>, <big>, <small>,

Special Characters:

&nbsp;   a *space* that can "glue" words together
&apos; ' a single quote (apostrophe)
&mdash;   – a long hyphen
&amp; & an ampersand

Event handling spec'd as attribute of an element:
<… onclick="// JS code run when event occurs">

onclick	user must click (mouse down-up) on the element
onmouseover	move cursor over the element
onmouseout	move cursor off the element
onload onunload()	Attribute of <body>, invoked as soon as page finishes "loading." Often used to set a timer with setInerval("jsFunct()", numMsec);
onfocus onblur()	A textbox become the typing focus (by selecting it)
onchange	The value of an attribute is changed by the user

Event Handlers often use:

this.*attrib*	an attribute of the same element
document.getElementById('*label*').*attrib*	an attribute of the element with id="*label*"
amountBox.value img1.height	Doc-Object model naming

Event Handlers often access or change:

innerHTML	replace all of the HTML in the targeted element with whatever is spec'd
src	change the image file displayed
height	change height of an image
width	change width of an image

JavaScript

Defined as a function within a <script> element in the <head>.

```
<head><script>
 function myFunct() {
   alert("hi"); }
</script></head>
```

And/or, included within quotes and assigned as the event handler for an attribute of an element.
<img src="a.jpg" onclick="**myFunct(*args*);**">

Variables (local/global) *vs.* parameters

- Retain value until explicitly changed with another assignment.

- May be accessed directly by name.
- May only hold one value at a time.
- Are always stored as a specific *type* (number, string, Boolean, etc).

```
var1 = value;  var2 = expression;
// evaluate expression to a value, then assign
// Expressions may include: + - * /
// or functions: Math.random()
```

JS Functions

Name	Argument(s)
alert()	Text to be displayed in a pop-up. A numeric variable's value is automatically converted to text.
parseFloat()	Text that you want to convert to a number. (Often the contents of a text box.)
parseInt()	Return the (leading) integer value of its argument which is a string or number " 9.9" and 9.9 both → 9
x=setInterval()	First parameter is a (user defined) JS function that is invoked everytime the timer goes off. 2nd parameter is the amount of time in msecs (1000 → 1 second).
clearInterval(x)	Stops an interval timer
setTimer()	One shot timer
isNaN(arg)	True if arg is not a number or it is a string that does not begin with a number, false otherwise

document.getElementById('*label*') is also a JS function that accesses (finds) an element (object) in the current page.

User defined functions
Defined with a <script> element in the <head>.
Each function must have a unique name, followed by parenthesis and the code enclosed in curly braces.

```
<script>
  function abcd(opt_arg_list) {
    var local_temp = whatever;
    // do stuff
    // using the args, or why are they there
   return value; // optional return statement
   }
```

Math Library Functions: Math.*xxx*

sqrt(4)	2 // result * result → original
abs(-3)	3 // result is always positive
ceil(1.02)	2 // closest int greater than
floor(1.99)	1 // closest int less than
round(1.99)	2 // usual meaning
random()	0.1234 // [0,1)
max(a, b)	a, if a>b; otherwise b
min(a, b)	a, if a<b; otherwise b
pow(3, 2)	9 // 3^2

Random # between min & max (inclusive):
```
Math.floor(Math.random()*(max-min+1)) + min;

num += Math.random() // rounded to two places…
alert(num.toFixed(2));
num2 = Math.round(num * 100)/100;
```

Libraries – loading JS code
```
<script src="dir/fileName"></script>
```

Conditionals:
```
if (anum % 2 == 0) alert("even");
 // else dealWithOddNums(anum);

if (anum < 12.34 && name != "Susan") {
   // block of stmts executed if true
   }
else {
    // block of stmts executed if false
    }

if (size < 1) alert("petite");
else if (size < 2) alert("small");
else if (size < 3) alert("medium");
else alert("large");
```

==	!=	<	>	<=	>=	&&	||	!

```
switch(code) {
  case 0: //code block
           break;
  case 1: // code block
           break;
  default: // code block
  }
```

Repetition:
```
anum = 5;
while (anum > 0) {
  alert(anum);
  anum -= 1;
  }

for (anum=5; anum>0; anum--)
  alert(anum);

anum = 5;
recFunct(anum);
          function recFunct(x) {
             if (anum > 0) {
                alert(anum);
                recFunct(x-1);
                }
          }
```

Strings:
JS strings (and primitive numbers) are immutable
S3 = "ab" + " cd"; // creates the new string "ab cd"
S3 = S1 + S2; // creates a new string S2 is concatenated
Mixing strings and numbers is confusing – use with care

s.length	A property of the string
s.toLowerCase()	Returns a lowercase version of s
s.toUpperCase()	→ uppercase version of s
s.charAt(indx)	Returns the single character at location indx in s or undefined (0…)
s.substring(start, end+1)	→string of characters from start to end in s
s.search('xx')	Index of first occurrence of the substring ‘xx’ in s or -1
s.search(/[aeiouAEIOU]/)	Index of first vowel in s
s.search(/[0-9]/)	Index of first digit in s
s.split(' ');	Returns an array of substrings of s split on spaces (spaces are eaten)

Arrays:
```
mix = [1, 1.234, "huh", false, 2*3];
alert(mix[1]);
x = mix[0] + 1;
if (!mix[3]) mix[2] = mix[0] + 1;
mix[9] = "wtf";
alert(mix[8]);
alert(mix);

for(indx=0; indx<counters.length; indx++)
   counters[indx] = 0; // reset values to zero
```

Data Representation
No integers & floats/doubles, just numbers
Numerical representation errors: 1/3 or 1/10
Everything (eventually) represented in binary
Strings are a sequence of characters (indexed 0…)
Characters are “integer” codes (Unicode/ASCII),
most with natural ordering, but ‘a’ > ‘A’
Hexidecimal (base 16 – 0..9A..F) are 4 bits each
Images represented as pixels
Colors represented as RGB, usually in hex (x00-xFF),

#000000	Black	#FFFFFF	White
#CCCCCC	Gray shade	#FF0000	Red
#00FF00	Green	#0000FF	Blue