Today's Goals

Today's lecture will cover:
- More on `new` and `objects`
- Built in objects Image, String, Date, Boolean, and Number
- The `getElementById()` method
- Layers

More on `new`

- In the exercise from last class period, we created elements of an array using the keyword `new`. Let's look deeper.
- The `new` operator is used to create an instance of a pre-defined object. (Remember that instances are to objects as proper nouns are to nouns.)
- If an object has a constructor function, that function is executed when an instance of the object is created.

Creating/Defining Objects

- A user can define an object.
- In JavaScript, an object is defined by defining the constructor function.
- A constructor function is defined just like a function.
- The name of the constructor function defines the name of the object.
- The properties and methods of the object are defined and initialized within the constructor function.
- The `new` operator is the only way to call a constructor.

Creating/Defining Objects (continued)

- For example, the following function defines the object instructor:

  ```javascript
  function instructor(name, phone, email)
  {
    this.name = name;
    this.phone = phone;
    this.email = email;
  }
  ```

- To create an instance of instructor, simply initialize a var with the constructor containing the appropriate arguments.

  ```javascript
  var tarnoff = new instructor("David Tarnoff", "423.439.6404", "tarnoff@etsu.edu");
  ```

Creating/Defining Objects (continued)

- The keyword `this` is used to identify the current instance being referenced by the function.
- Remember that objects can be embedded into hierarchies, i.e., an object can become the property of an object.
- For example, the instructor object defined above could become a property of a course_section object.
Creating/Defining Objects (continued)

```javascript
function course_section(course_title, section_number, assigned_instructor) {
    this.title = course_title;
    this.section_number = section_number;
    this.instructor = assigned_instructor;
}
```

An instance could then be created:

```javascript
var CSCI2910_001 = new course_section("Client/Server-Side Programming", "001", tarnoff);
```

Creating/Defining Objects (continued)

- To create a function for an object, we used the keyword "prototype".
- Within the constructor function, insert the code:
  ```javascript
  this.prototype.myfunction = function(args) {
      // insert myfunction code here
  }
  ```
- Can also define outside constructor function:
  ```javascript
  obj_name.prototype.myfunction = function(args) {
      // insert myfunction code here
  }
  ```

Image Object

- There are a number of pre-defined JavaScript objects such as the Image object.
- Properties of the Image object include:
  - `border` - Contains the width of the border in pixels (read only)
  - `complete` - Boolean value indicating whether the browser has finished loading the image. (read only)
  - `height` - The height of the image in pixels (read only)
  - `lowsrc` - Specifies the URL of a low-resolution replacement of the image which is loaded and displayed before the high-resolution image is loaded and displayed
  - `name` - This is the name/id property of the image
  - `src` - Specifies the URL of the image
  - `width` - The width of the image in pixels (read only)

String Object

- The constructor for a new String object takes as its argument the initial string:
  ```javascript
  myString = new String("This is great!");
  ```
- The property length returns the length of the string. For the example below, `mylength` would equal 14.
  ```javascript
  mylength = myString.length;
  ```

String Object Methods

- `charAt(index)` - returns the character at the position in the string referred to by index
- `charCodeAt(index)` - returns the Unicode value of the character at the position in the string referred to by index.
- `fromCharCode(num1,...,numN)` - creates a string from the sequence of Unicode values passed to it as arguments.
- `toLowerCase()` - converts all of the characters in the string to lower case.
- `toUpperCase()` - converts all of the characters in the string to upper case.
- `indexOf(character[, start_index])` - returns the index of the first occurrence of the specified character. If `start_index` is used, search begins from that point in the string.
- `lastIndexOf(character[, start_index])` - returns the index of the first occurrence of the specified character. If `start_index` is used, search begins from that point in the string.
- `split(delimiter)` - splits a string into substrings and returns an array that contains the resulting substrings.

Formatting Methods of String

- There are some methods of the object String that when used in conjunction with an output method will create HTML like formatting. For example, the method `sub()` will cause the text to be outputted as a subscript:
  ```javascript
  var subscript = "24";
  document.write("A" + subscript.sub());
  ```
- outputs the following to the HTML screen:
  \[ A_{24} \]
**Formatting Methods of String**

(continued)

- anchor("name") – creates an HTML anchor.
- blink() – makes the displayed string blink. (Of course you know the warnings about blink in HTML, right?)
- fixed() – makes the displayed string appear as if contained within \(<tt>...</tt>\) tags.
- strike() – makes the displayed string appear as if contained within \(<strike>...</strike>\) tags. (strike through)
- sub() – makes the displayed string appear as if contained within \(<sub>...</sub>\) tags. (subscript)
- sup() – creates an HTML link pointing to URL.
- link("URL") – returns a four-digit absolute number that represents the year, according to universal time, for the supplied date.
- dateString – a string that represents the date in a format that is recognized by the Date.parse method.
- milliseconds – an integer that represents the number of milliseconds since January 01 1970 00:00:00.
- yr_num, mo_num, day_num, hr_num, min_num, sec_num, ms_num – a set of integers that represent the year, month, and day of the date.
- hr_num, min_num, sec_num, ms_num – a set of integers that represent the hours, minutes, seconds, and milliseconds.

**In-Class Exercise**

- Divide into teams.
- Using the printout of FormChek.js, pick a procedure and discuss within your team how the procedure works. Pick an interesting one.

**Date Object**

- There are a number of constructors that can be used to create a new Date object.
  - new Date()
  - new Date(milliseconds)
  - new Date(dateString)
  - new Date(yr_num, mo_num, day_num [, hr_num, min_num, sec_num, ms_num])

- Sometimes, the arguments to these constructors may be confusing.
  - milliseconds – an integer that represents the number of milliseconds since 01/01/70 00:00:00.
  - dateString – a string that represents the date in a format that is recognized by the Date.parse method.
  - yr_num, mo_num, day_num – a set of integers that represent the year, month, and day of the date.
  - hr_num, min_num, sec_num, ms_num – a set of integers that represent the hours, minutes, seconds, and milliseconds.

**Date Object Methods (continued)**

(Source: www.devguru.com)

- getMonth() – returns an integer (0 for January thru 11 for December) that represents the month for the specified date.
- getSeconds() – returns an integer between 0 and 59 that represents the seconds (local time) for the specified date.
- getTime() – returns a numeric value representing the number of milliseconds since midnight 01/01/1970 for the specified date.
- getTimezoneOffset() – returns the difference in minutes between local time and Greenwich Mean Time. This value is not a constant, as you might think, because of the practice of using Daylight Saving Time.
- getUTCDay() – returns an integer between 1 and 6 that represents the day of the week, according to universal time, for the specified date.
- getUTCDate() – returns an integer (0 for Sunday thru 6 for Saturday) that represents the day of the week, according to universal time, for the specified date.
- getUTCFullYear() – returns a four-digit absolute number that represents the year, according to universal time, for the supplied date.
- getUTCHours() – returns an integer between 0 and 23 that represents the hour (local time) for the specified date.
- getUTCMilliseconds() – returns an integer between 0 and 999 that represents the milliseconds (local time) for the specified date.
- getUTCMinutes() – returns an integer between 0 and 59 that represents the minutes (local time) for the specified date.
- getUTCMonth() – returns an integer, 0 for January thru 11 for December, according to universal time, for the specified date.
- getUTCSeconds() – returns an integer between 0 and 59 that represents the seconds, according to universal time, for the specified date.
- parse(dateString) – takes a date string and returns the number of milliseconds since January 01 1970 00:00:00.
Date Object Methods (continued)
(Source: www.devguru.com)
• `setDate(dateVal)` – used to set the day of the month using an integer for the supplied date according to local time. (1 to 31)
• `setFullYear(yearVal[, monthVal, dayVal])` – used to set the full year for the supplied date according to local time.
• `setHours(hoursVal[, minutesVal, secondsVal, msVal])` – used to set the hours for the supplied date according to local time.
• `setMilliseconds(millisecondsVal)` – used to set the milliseconds for the supplied date according to local time. (0 to 999)
• `setMinutes(minutesVal[, secondsVal, msVal])` – used to set the minutes for the supplied date according to local time.
• `setMonth(monthVal[, dayVal])` – used to set the month for the supplied date according to local time.
• `setSeconds(secondsVal[, msVal])` – used to set the seconds for the specified date according to local time.
• `setTime(timeVal)` – used to set the time of a Date object according to local time. The timeVal argument is an integer that represents the number of milliseconds elapsed since 1 January 1970 00:00:00.
• `setUTC???( )` – there are similar functions for setting UTC date
• `toGMTString()` – converts a local date to Greenwich Mean Time.
• `toLocaleString()` – uses the relevant locale’s date conventions when converting a date to a string.
• `toString()` – returns a string representing a specified object.
• `toUTCString()` – uses the universal time convention when converting a date to a string.
• `UTC(year, month, day[, hours, minutes, seconds, ms])` – returns the number of milliseconds from the date in a Date object since January 1, 1970 00:00:00 according to universal time. This is a static method of Date so the format is always `Date.UTC()` as opposed to `objectName.UTC()`.

Boolean Object
- A number of methods such as isNaN() return true/false values
- Programmers can create their own true/false values using Boolean elements.
- Can create objects explicitly using new along with constructor (constructor takes as argument initial value – default is "false")
  ```javascript
  var b_val = new Boolean("true");
  ```
- Supports toString() method.

Number Object
- There is an object allowing programmers to create variables to hold numeric constants.
- Primarily used to access Number methods.
  ```javascript
  const_val = new Number(24); 
  ```
- Number properties:
  - `MAX_VALUE` – property that represents the largest possible JavaScript value (approx. 1.79769e+308)
  - `MIN_VALUE` – property that represents the smallest possible positive JavaScript value. (5e-324)

Number Object Methods
- `toExponential(num_digits)` – returns a string containing the number in exponential form with the number of digits following the decimal point defined by num_digits.
- `toFixed(num_digits)` – returns a string containing the number represented in fixed-point notation with the number of digits following the decimal point defined by num_digits.
- `toString([radix])` – returns a string representing the Number object. If used, "radix" indicates the base to be used for representation. "radix" can be between 2 and 36.

Accessing HTML Elements as Objects
- In order to have access to the object properties and methods inherent to an HTML element, we have to declare an object instance to refer to the HTML element.
- This is done with the `getElementById()` method.
  ```javascript
  var html_obj = document.getElementById("test");
  ```
- This code will create the object `html_obj` that points to the tag that used the name/id "test".
Modifying the Style of HTML Objects

• One of the most common uses for `getElementById()` is to create an HTML object in order to modify its style or change one of its attributes.
• Style typically uses hyphens, e.g., font-size.
• JavaScript replaces hyphens by capitalizing next character, e.g., fontSize replaces font-size.

```javascript
html_obj.style.fontSize = "16px";
html_obj.setAttribute("align", "center");
```

Layers

• In HTML, the elements are displayed in the order that they are encountered in the source.
• With CSS, positioning became easier, but elements still fought in shared space with margins and padding.
• The concept of layers is one that has been used in graphics for a long time, i.e., the concept that groups of elements can reside on different planes in the z-direction, not just the x- and y- directions.

Layers (continued)

A number of benefits come with this capability:
– Elements can be placed at exact X and Y positions without fighting for space with elements on other layers.
– Elements can overlap.
– Transparent elements can have other elements showing through.

Layer Attributes

• Layer element is define using the HTML `<layer>...</layer>` tags
• Attributes of the `<layer>` tag:
  – name/id = "layername" – same as name and id for other HTML elements
  – left = "pixels" – number of pixels from the left edge of the browser window
  – top = "pixels" – number of pixels from the top edge of the browser window
  – z-index = "integer" – an integer specifying the position of the layer with respect to the other layers. The higher numbers are stacked on top of the lower ones.
  – above = "layername" – this attribute allows the programmer to indicate the name/id of a layer above which this layer is to be placed. (Used instead of z-index)
  – below = "layername" – this attribute allows the programmer to indicate the name/id of a layer below which this layer is to be placed.
  – visibility = "show | hide | inherit" – determines whether the layer is displayed or not. Can be changed real-time to create certain effects such as swapping text.
  – bgcolor = "rgbColor" – specifies background color of layer.
  – background ="imageURL" – specifies background image of layer.

JavaScript Control of Layers

• In JavaScript, the layers appear in an array called "layers".
• Can access these layers in one of three ways:
  – `document.layerName`
  – `document.layers[index]`
  – `document.layers["layerName"]`
• A layer's properties can be accessed with the syntax:
  `layerName.propertyName`
Layers also have some methods:

- **offset(x,y)** – Changes a layer's position by using the x and y values as offsets from the current position.
- **moveTo(x,y)** – Changes a layer's position by moving its upper left corner to the position specified by x, y.
- **resize(width,height)** – Changes a layer's size.
- **moveAbove(layerName)** – Moves layer to position immediately above layer referred to by layerName.
- **moveBelow(layerName)** – Moves layer to position immediately below layer referred to by layerName.