JAVASCRIPT – AN INTRODUCTION

Kiet Bui

: AGENDA

- History
- The Three Layers of the Web
- Programming with JavaScript
- Document Access
- Browser Access
- Events
- Animation
- Forms
- Go Pro
  - Errors and Debugging
  - Ajax
  - CoffeeScript
  - Knockout.js

: HISTORY

- Created by Brendan Eich in 1995 for Netscape Navigator 2 release. Called Mocha and later LiveScript.
- On release of 1.1 name was changed to JavaScript.
- In 1997 JavaScript 1.1 was submitted to ECMA as a proposal. TC39 was assigned to standardize the syntax and semantics of the language.
- TC39 released ECMA-262 known as ECMAScript.
- ECMAScript is basis for all the JavaScript implementations.

: THE THREE LAYERS OF THE WEB

- HTML for Content
  - There is no download link on this page.
- CSS for Presentation
  - .warning { color: red; }
- JavaScript for Behavior
  - <script type="text/javascript">
    window.alert(document.getElementsByClassName("warning")[0].innerHTML);
    </script>

LIBRARIES

JavaScript can be used to create reusable libraries.

Some are:

- Prototype
- Script.aculo.us
- Yahoo! User Interface Library
- Dojo
- jQuery
- MooTools
- Knockout

: PROGRAMMING WITH JAVASCRIPT

Running a JavaScript program/script.

There are two ways:

- <script type="text/javascript">
  window.alert("JavaScript");
  </script>
- <script type="text/javascript" src="myScript.js"></script>
- <script type="text/javascript" src="myScript.js" defer></script>

There was once a language attribute also in the script tag. i.e., language="javascript 1.5" or whatever

STATEMENTS, COMMENTS AND VARIABLES

- Statements: Can be separated by new line or semicolon.
- Comments: 2 types.
  - Single line: // Author: Manvendra SK
  - Multi line: /* Perhaps some lengthy
    license. */
- Variables: Overwrites when redefined. 2 types.
  - Local: Declared using var keyword.
  - Global: Declared without var keyword or attached to **window** object directly. i.e., window.someVar = “value”;
  - We use **assignment opertor (=)** to assign values.

VARIABLE DATA TYPES

JavaScript is **loosely typed** language. While Java is **strictly typed**.

- Numbers: 3, -3, 3.33, -3.33. These all are numbers in JS, whether it's some **integer** or **floating point number**.
  - Operations: **addition (+), subtraction (-), division (/), multiplication (*), modulo division (%)**
  - Result is always promoted to float whenever possible. i.e., 5 / 3, 3 / 5, 0.5 * 5, 0.5 + 5
  - Assignment operators: **+=, -=, /=, *=, %=**
  - Special operators: **increment (++), decrement (–)**

STRINGS

- Strings: Series/sequence of characters from zero to infinity. Can contain any character. Can be created using single or double quotes.
- Use backslash (\) to escape special characters.

  i.e. “Hi this is \“special\” word here.”

  “Hi this is **t**tab**t** here.“
- Operations: Concatenation.

  i.e., “Some ” + “string.”

  “Some ” + someVar + “.”

  var name = “Manvendra ”;

  name = name + “SK”; or name += “SK”;

BOOLEANS

- Booleans: Are you saying **true** or **false**.
  - Cases for true
    - “<1 space here>”, “string”, “undefined”, 1, 2, 3, -3, -2, -1, true; all represents **true** value.
  - Cases for false
    - “<empty string>”, undefined, null, void(0), 0, false; all represents **false** value.
  - Universal checking program:
    - <any value> ? “True” : “False”;

ARRAYS

- Arrays: To hold a collection of items together. Can store any data types together in single array.

  i.e., var array = [1, {“k”: “v”}, 2.3, [“another”, “array”]];

- 2 types: Single dimensional and Multi dimensional (array of array(s)).

  i.e., var array = [“1st”, “2nd”, “3rd”];

  var arrayM = [[“1st”], [“2nd”], [“3rd”]];

- Accessing values stored in array: We use what is called index which is some integer ranges from 0 to array’s length - 1.

ACCESSING ARRAYS

- Accessing Single dimensional array:

  i.e., array[0]; // 1st

  array[1]; // 2nd

  array[2]; // You guess here.

- Accessing Multi dimensional array:

  i.e., arrayM[0]; // [“1st”]

  arrayM[0][0]; // 1st.

ASSOCIATIVE ARRAYS

- Associative Arrays: Kind of Java Maps. Remember the Key-Value pairs?

  i.e., var pincodes = [];

  pincodes[“khanpur”] = 110062;

  pincodes[“badarpur”] = 110044;

  pincodes[“saket”] = 110017;

- Accessing Associative arrays:

  i.e., pincodes[“saket”]; // 110017

  pincodes[“badarpur”]; // You guess here.

- Associative arrays are actually **Object**s!

CONTROLLING PROGRAM FLOW

- Conditions: Making decisions.
- if statement: Expects a boolean expression.

> Expression is combination of values, variable references, function calls and operators that evaluates to some value.

- Some comparison operators that we can use are: **less than (<), greater than (>), less than equal to (<=), greater than equal to (>=), equals to (==), not equals to (!=), not (!)**
- Multiple conditions operators: **and (&&), or (||)**

IF STATEMENTS

- if-else statement: if condition is false then execute the else block.
- else-if statements: It's not reverse of the if-else. It just says if condition false then check this (else-if) condition.
- How it looks like:

  i.e., if (condition) {

  // Some true code

  } else if (condition) {

  // Some 2nd true code

  } else {

  // Some false code

  }

LOOPS

- Loops: Minimizing repetition
- while loop: Simplest loop. While condition is true run the code. Condition can be any expression.
- do-while loop: Runs at least once, as condition is evaluated after running the loop body. As always condition can be any expression.
- for loop: Succinct all the above!
- All the loops must consist three things: Incrementer, conditional expression, logic to increase the incrementer – so that condition eventually turns to false.

LOOPS FACES

- How these look like?:
- while loop: i.e., while (condition) {

```
                    // Run the code
            }
```

- do-while loop: i.e., do {

```
                    // Run the code
        } while (condition)
```

- for loop: i.e., for (declaration; condition; action) {

```
                    // Run the code
    }
```

FUNCTIONS: WRITING CODE FOR LATER

- If we want to re-run some code again and again from different places, then put that code in a function and call this function.
- Functions are like little packages of JavaScript code waiting to be called into action.
- Some predefined functions are **alert(), prompt() and confirm()**. These all are available on the top level **window** object.
- Functions can return values. Values returned by predefined **window** functions are: **undefined**, **user input string** or **empty string** or **null**, **true** or **false**.

MY FUNCTIONS

- Writing your own functions:

  i.e., function sayHi() {

  alert(“Hi”);

  }
- Calling functions: i.e., sayHi();

Parentheses are needed to call the functions.

ARGUMENTS: PASSING DATA TO FUNCTIONS

- Arguments or parameters: When a function expects something then it's called a **parameters**. While we pass the expected data to a function then it's called **arguments**.
- Declaring parameters:

  i.e., function sayHi(name) {

  alert("Hi " + name);

  }
- Calling function with arguments:

  i.e., sayHi("Manvendra");
- Functions can contain any number of parameters.

A SECRET ARRAY

- arguments array: Functions have one secret. They contain the **arguments** array. This array contains all the arguments passed to the function.

  i.e., function poll() {

  ```
  var affirmative = arguments[0];
  var negative = arguments[1];
  }

  // Calling the function
  poll("affirmative", "negative");
  ```

RETURN AND SCOPE

- Returning: As we know functions can return the values. We use **return** statement to return something.

  i.e., function sayHi(name) {

  return “Hi ” + name;

  }

- Scope: Can be local or global. Ensure functions always declare variables using the var keyword or else they will look for it in global scope and will modify them.

I HAVE ANOTHER FACE

- Alternate function syntax:

```
i.e., var sayHi = function() {
        alert("Hi");
      }
```

OBJECTS

- Objects exist as a way of organizing **variables** and **functions** into logical groups. If we create objects to organize some variables and functions then the terminology changes slightly, now they are called **properties** and **methods** respectively.
- We have used the objects in this presentation. Where? Didn't you use arrays? **Array** is an object of JavaScript. Same array can be created as follows: i.e., var array = **new** Array(1, 2, 3);
- This is called instantiation of object using the **new** keyword.
- Built-in objects: **String, Date, Math, Boolean, Number, RegExp, Object, Array**

CREATE YOUR OWN OBJECTS

- We can create our own objects to encapsulate the data and logic. i.e.,

```
var Person = new Object();
Person.name = “Manvendra SK”;
Person.age = 23;
Person.speakName = function() {
alert(“Hello, I’m ” + this.name);
};

Person.speakName();
```

ALTERNTAE SYNTAX

- Alternate objects syntax: i.e.,

```
 var Person = {
name: “Manvendra SK”,
age: 23,
speakName: function() {
        alert(“Hello, I’m ” + this.name);
}
 };

 Person.speakName();
```

It’s JSON (JavaScript Object Notation) syntax

CREATING REAL OBJECTS

- We can create objects those can be instantiated using the **new** keyword.

```
var Person = function(name, age) {
   this.name = name;
   this.age = age;
};

Person.prototype.speakName =  function() {
   alert("Hello, I'm " + this.name);
};
```

USING REAL OBJECTS

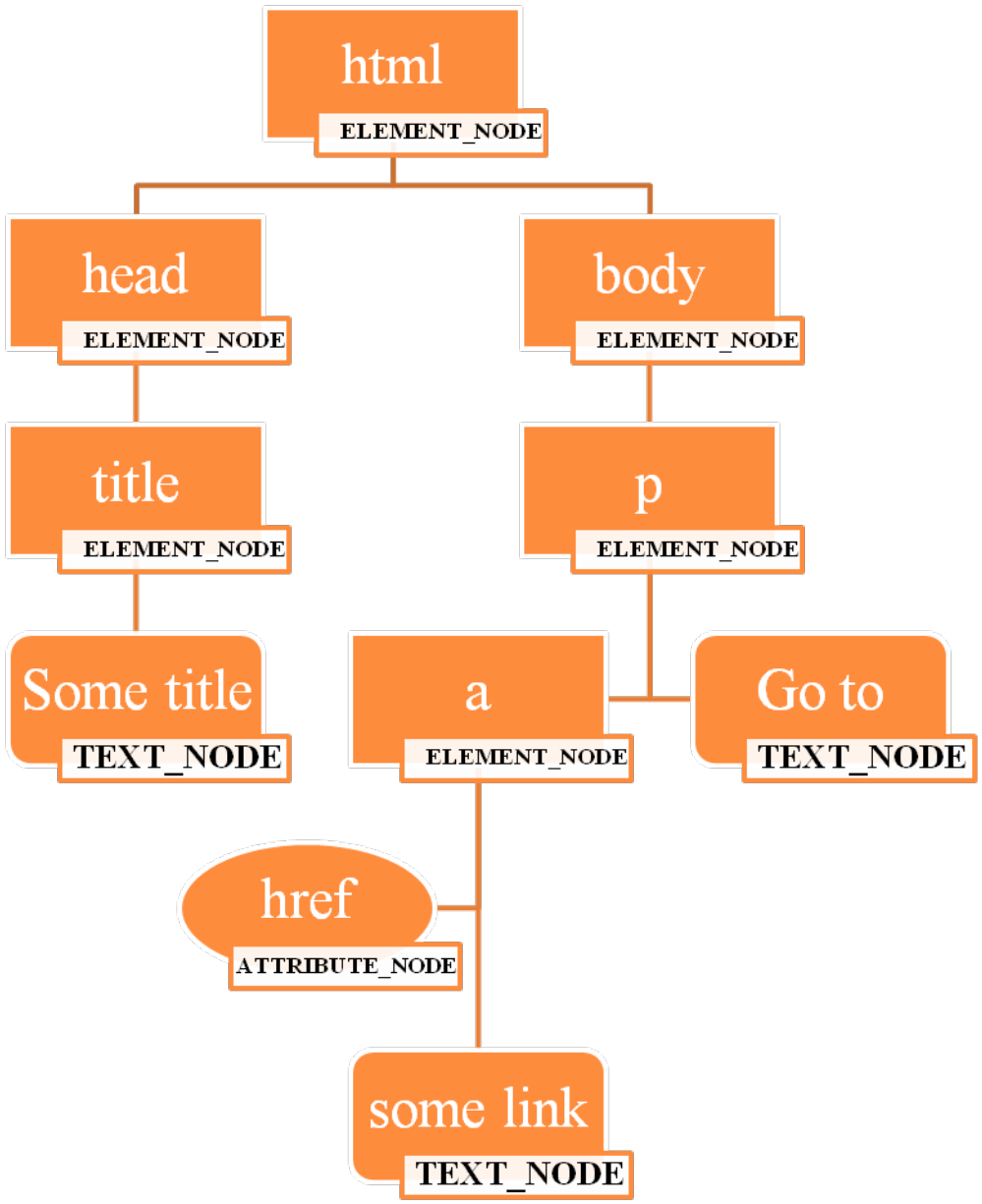
```
var manvendra = new Person("Manvendra", 23);

manvendra.speakName();
```

- This method of creating objects is called Prototype pattern.

: DOCUMENT ACCESS

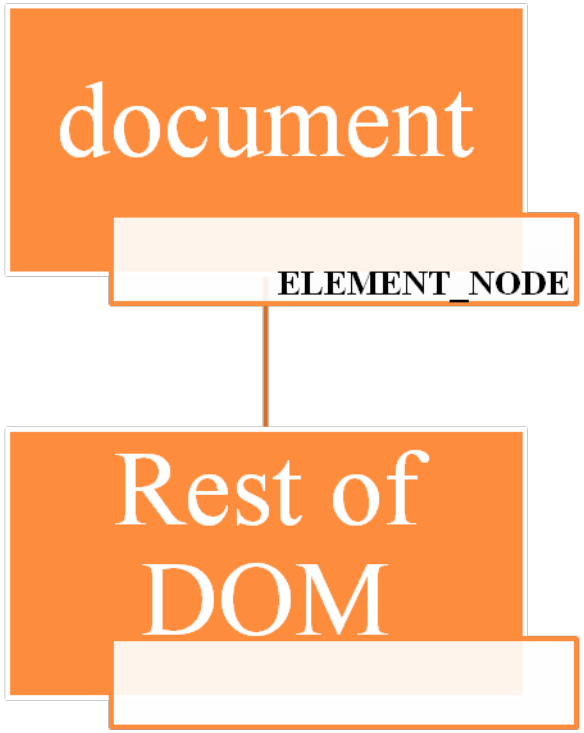
- Document Object Model: Mapping your HTML. The browser stores its interpreted HTML code as a structure of JavaScript objects, called **DOM**.
- DOM consists of Nodes. There are currently 12 types of nodes. Most of which we use are **ELEMENT_NODE (1), ATTRIBUTE_NODE (2), TEXT_NODE (3)**.
- We can use the **nodeType** property to check the type of the node. i.e., document.getElementsByTagName("h1") [0].nodeType; // 1

- DOM Structure:



GRAND NODE

- In fact there is an universal node. **document** node. This node exists even if the document is blank.



DOM MANIPLUATION

- Access elements: Use **document.get*()** methods.
- Alter properties. It depends on the type of node.
  - If it's some html node like div, h1, span, label then you would set its **innerHTML** property or sometimes **textContent** property, if element is **li**.
  - If it's some input element then you would set its **value** property.
- Traversal: We use **nextSibling, previousSibling, parent, children** etc. properties.

MANIPLUATING STYLE AND CLASSES

- Altering style: We can use the **style** property to alter the style of any element. i.e.,

  document.getElementsByTagName (“body”) [0].style.background = “#ddd”;

  document.getElementByTagName(“label”) [0].style.position = “absolute”;

  document.getElementByTagName(“label”) [0].style.left = “300px”;

- We can also use **className** property to access or set a CSS class directly. It’s directly available on the element.

: BROWSER ACCESS

- Browser Object Model: It targets the browser environment rather than the document. It offers some objects allow us to handle the browser by our script.
- window
- location
- navigator
- screen
- history

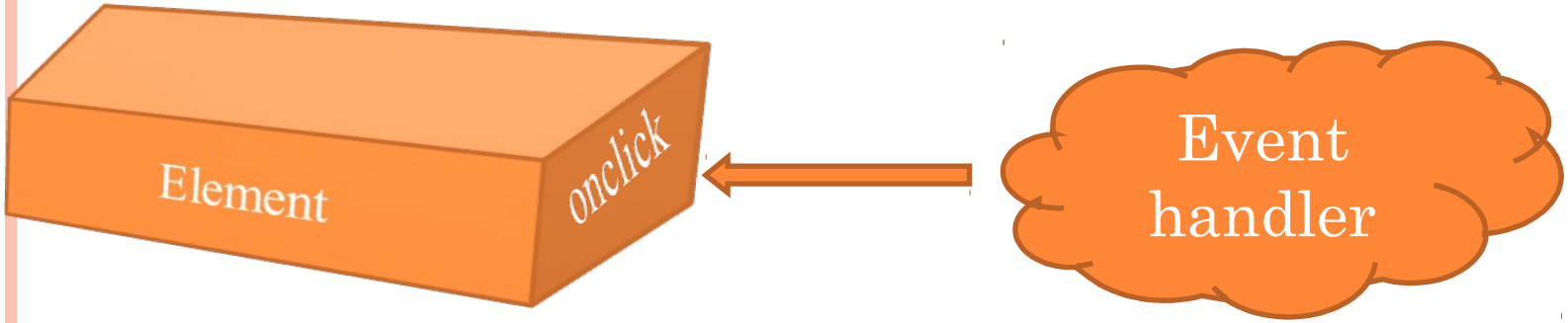
: EVENTS

- JS enables us to write scripts that are triggered by **events** that occur during the user's interaction with the page, like clicking a hyperlink, scrolling the browser's viewport, typing a value into a form field or submitting a form.
- DOM Levels:
  - DOM Level 1
  - DOM Level 2

There is also DOM Level 3

EVENT HANDLERS

- Simplest way to run JS code in response to an event is to use an **event handler** (function).



- How to attach?: element.on*event* = eventHandler

Note: I didn't provide parentheses to eventHandler

DEFAULT ACTIONS AND THIS

- Default actions: Things the browser normally does in response to events.
- How to prevent?: return **false** or **true** to cancel or let the default action follow.
- this keyword. **this** is an object reference that you get when executing a method on an object. When browser invokes the event handler on some event for an element, then within the event handler **this** points to the element on which event is fired.

We can't set the value of **this** object.

EVENT LISTENERS

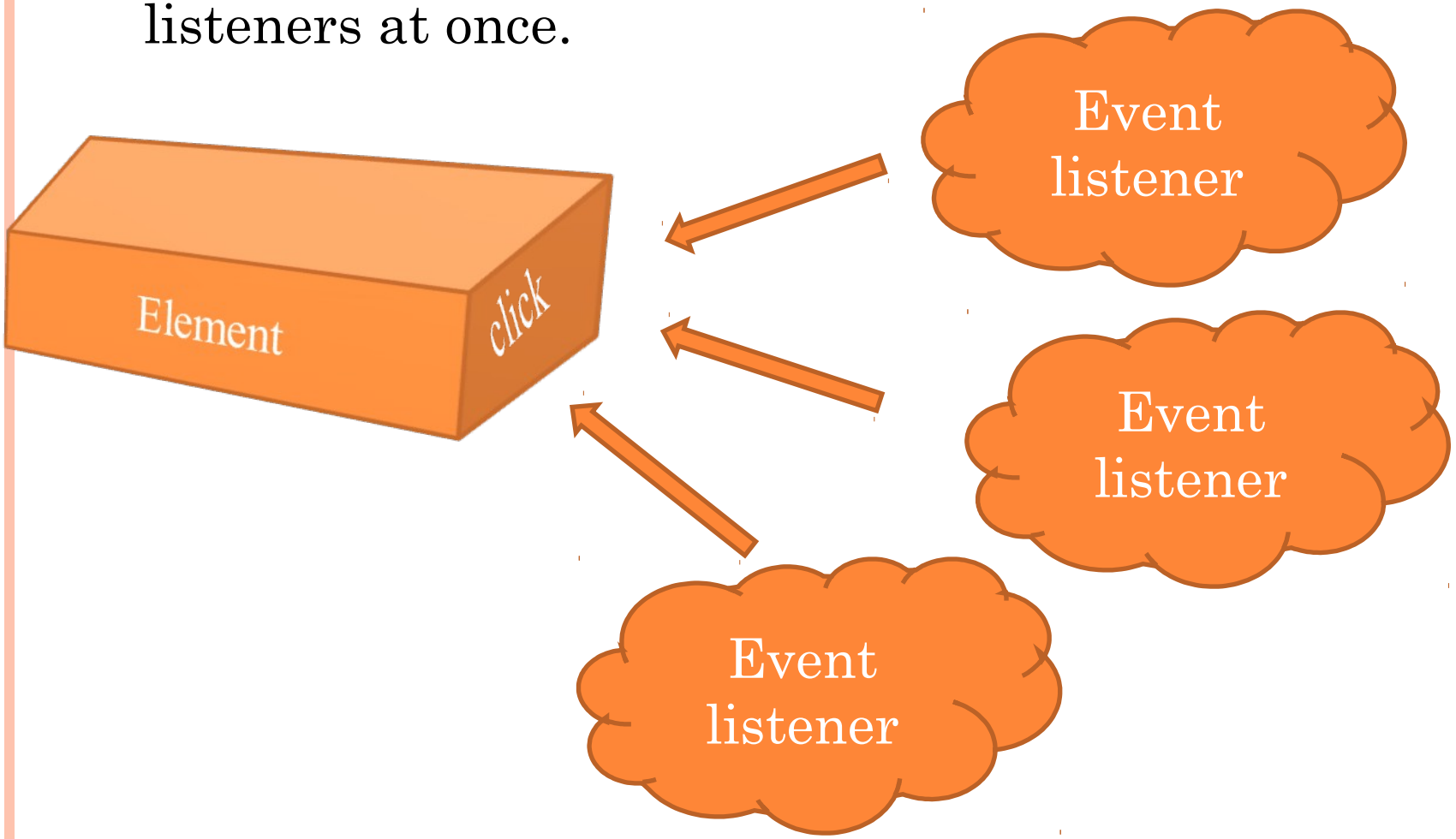
- What's wrong with event handlers? You can't assign multiple event handlers to an event. i.e.,

  element.onclick = firstHandler;

  element.onclick = secondHandler;
- Now only secondHandler will be called, as it has replaced the firstHandler.
- However we can assign as many event listeners as we want to an event of an element, and all of them will be called.

IT'S LIKE HUB

- As we can see we can plugin many event listeners at once.



ATTACHING EVENT LISTENERS

- How to attach event listener?: i.e.,

  element.addEventListener("event", eventListener, false);

  for IE, element.attachEvent("onevent", eventListener);

Object detection:
typeof element.property != "undefined"

EVENT PARAMETER

- Event parameter of the listener: Browser automatically passes the **event** parameter to the event listener function. i.e.,

  ```
  function someClickEventListener (event) {
        // Use event argument's methods
  }
  ```

- **event** parameter has some important methods. Some of them are: **preventDefault()** and **stopPropagation()**.

EVENT OBJECT

- IE has its own way. If you remember IE doesn't expect third argument. It means it also doesn't pass **event** parameter to the event listener function. Then how to access it?
- In IE there is a global event object, which is **window.event**. And it has equivalent properties like the event parameter which are: **returnValue** which we can set to **false**, and **cancelBuble** which we can set to **true**. These two settings achieves the same effect as its event parameter counterpart's methods.

DETACHING EVENT LISTENERS

- How to detach event listener: i.e.,

  element.removeEventListener(“event”, eventListener, false);

  for IE, element.detachListener(“onevent”, eventListener);

EVENT PROPAGATION

- What is event propagation?
- Event propagation runs in three phases:
  - Capture phase: Document to element.
  - Target phase: Element which triggered the event.
  - Bubbling phase: Element to Document.

: ANIMATION

- Animation is nothing but the frames those comes one by one and complete in some time from start to end. We need some way to schedule the frames to come one after other.
- JavaScript provides two methods **setTimeout** and **setInterval** both of which are available on **window** object. These two methods comes with their companions **clearTimeout** and **clearInterval** respectively.
- **setTimeout** calls the task only once, while **setInterval** calls the task repeatedly.

USING SET* METHODS

- How to use?: Both methods have same syntax. i.e.,

  window.setTimeout(callback, timeout);

- Both methods return a **timer**'s id which (timer) they have created. It's where we use the **clear*** methods, passing them this **timer**'s id.

Callback is a function that is called by the script that expect it as a parameter upon its task completion. It means functions can be passed as parameters like normal variables. Just pass function name without parentheses.

: FORMS

- Forms are there to collect the data. And JavaScript is known for its form validations. But as we know JavaScript is more than that. However forms are still integral part. Whenever there is a single input box it must be contained inside a form.
- Some DOM Methods for HTML Form Controls

Method	Element(s)	Description
blur	input, select, textarea	Removes keyboard focus
click	input	Simulates a mouse click
focus	input, select, textarea	Gives keyboard focus

- …Continued

reset	form	Reset all control's value to default
select	input, textarea	Selects all the contents
submit	form	Submits the form without triggering submit event

- Some DOM Properties for HTML Form Controls

Property	Element(s)	Description
elements	form	A node list containing all the form controls

- …Continued

checked	input	True only for checkbox and radio inputs if checked else false
disabled	button, input, optgroup, option, select, textarea	While true controls is unavailable to user
form	button, input, option, select, textarea	A reference to the form containing this control
index	option	Index of this option control within the select control that contains it (0 for first)

- …Continued

options	select	A node list containing all the options controls
selected	option	True if this option is selected else false.
selectedIndex	select	Index of the currently selected option (0 for the first)
value	button, input, option, select, textarea	Current value of this control, as it would be submitted to the server

- Some DOM Events for HTML Form Controls

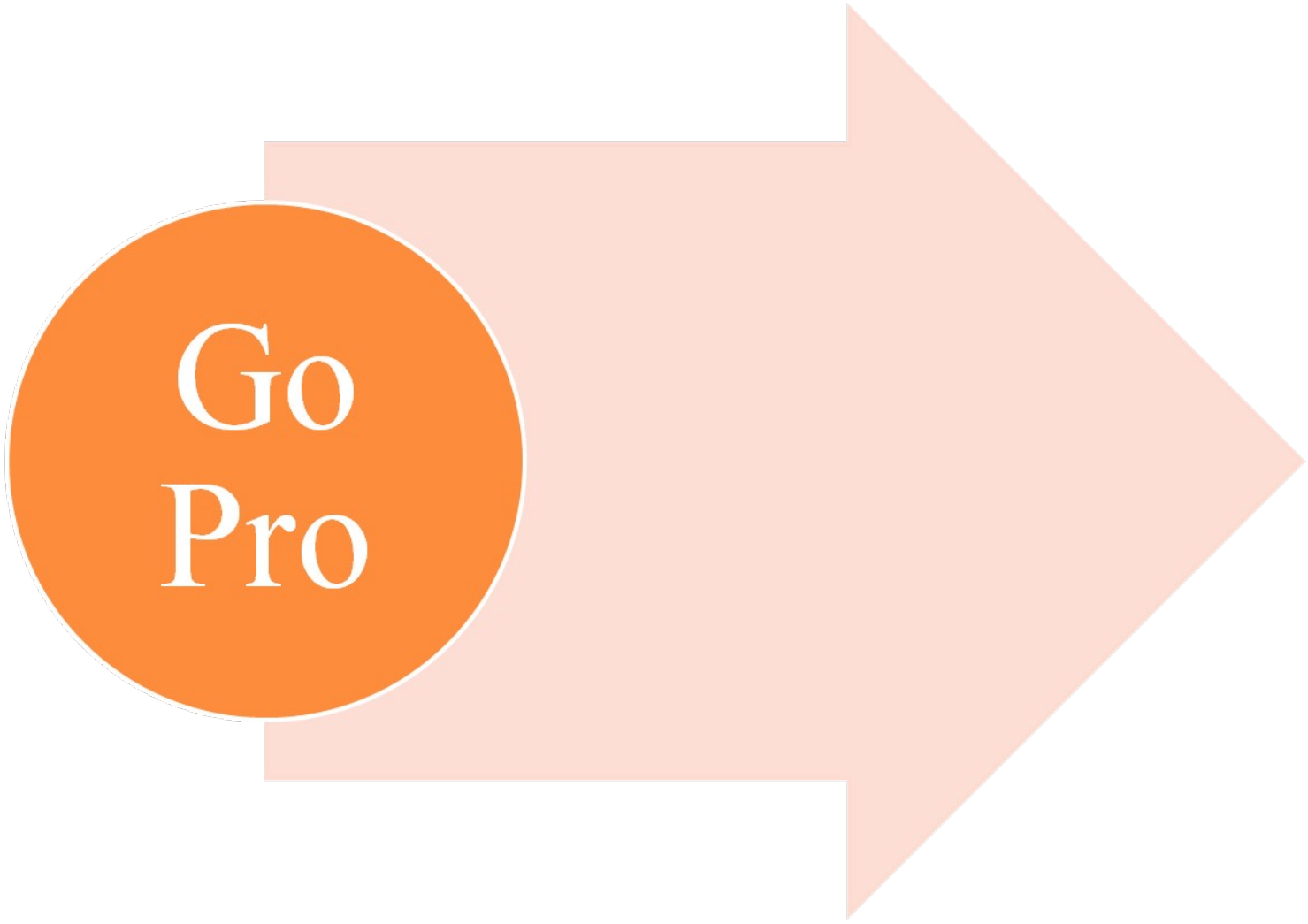
Event	Element(s)	Triggered when…
change	input, select, textarea	Control lost focus after its value has changed
select	input, textarea	User has selected some text
submit	form	User has requested to submit the form

HANDLING THE SUBMIT EVENT

- How to handle the submit event of the form?: i.e.,

  form.addEventListener("submit", submitListener, false);

  For IE, form.attachEvent("onsubmit", submitListener)


Go
Pro

: Errors and Debugging

- Every browser has its own way for JavaScript error messages. As each browser has different implementation of the language. Most sensible are Firefox's.
- Three kinds of error can occur in JavaScript.
  - Syntax Errors: Occur when your code violates the fundamental rules (or syntax) of the language.
  - Runtime Errors: Occur when a valid piece of code tries to do something that it's not allowed to do, or that is impossible.
  - Logic Errors: Occur when there are bugs in our code. And we don't get intended results from the script.

DEBUGGING TOOLS

- There are JavaScript debugging tools built right into the browsers.
- Some are:
  - Firefox: Ctrl+Shift+I
  - Chrome: Ctrl+Shift+I
  - Opera: Ctrl+Shift+I
  - Safari: Ctrl+Alt+I
  - IE: F12
  - Firefox's Firebug: F12
- Inside these tools we can execute our script step by step, watch for variables values and more.

: AJAX

- AJAX acronym for Asynchronous JavaScript And XML.
- This technology is used for **asynchronous** information transfer between browser and server in **bite-size** chunks.
- It is supported using the **XMLHttpRequest** object built right into the browser.
- Firstly implemented by IE 5 and 6 and they did using the **ActiveX** object.
- IE 7+ don't use ActiveX object, instead they use XMLHttpRequest.

INITIALIZE

- How to initialize?: i.e.,

  var requester = new XMLHttpRequest();

  For IE,

  var requester = new ActiveXObject(“Microsoft.XMLHTTP”);

PROPER INSTANTIATION

- Example of instantiating the XMLHttpRequest:

```
try {
    var requester = new XMLHttpRequest();
} catch (error) {
    try {
        var requester = new
    ActiveXObject("Microsoft.XMLHTTP");
    } catch (error) {
        var requester = null;
    }
}
```

USING THE XHR

- Using the XMLHttpRequest:

```
requester.setRequestHeader("Content-Type",
    "application/x-www-form-urlencoded");  //
    Optional
requester.open("GET", "/url.xml", true);
requester.send(null);
requester.onreadystatechange = function() {
    // Use requester.readyState property, which is 0
    to 4, uninitialized, loading, loaded, interactive,
    complete.
    // Also now check requester.state property, which
    contains HTTP codes of response. For success use
    200 or 304, other are failures.
```

READ THE DATA FROM XHR

- Where is the data:
  - responseXML: If the server responsed with **content-type** set to **text/xml** then we have DOM. We can traverse it as usual to get the data. It also populates the **responseText** property, just for an alternative.
  - responseText: If the server responsed with the **content-type** set to **text/plain** or **text/html** then we have single string as the data. But now the **responseXML** will be empty.

: COFFEESCRIPT

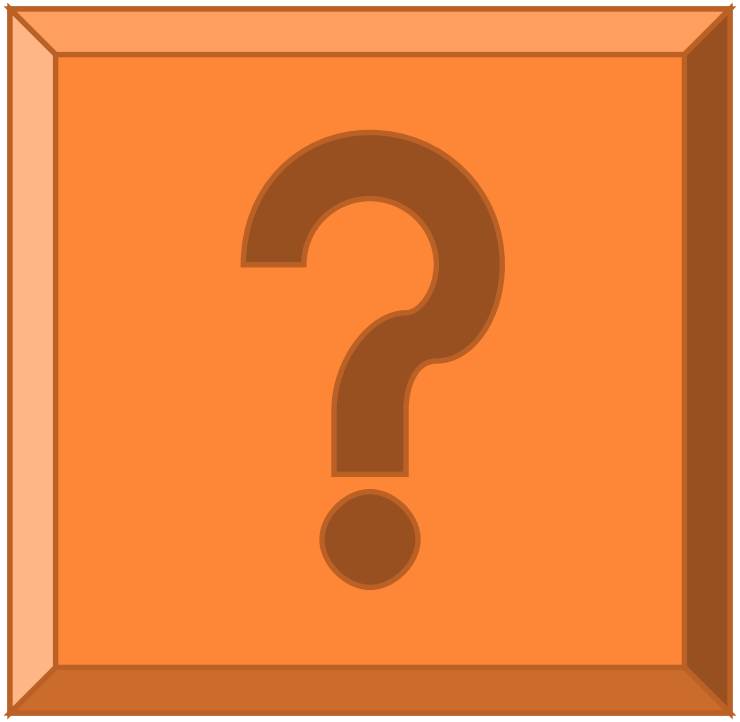
- CoffeeScript is a new way to write tricky and lengthy JavaScript easily and intuitively.

ttp://coffeescript.org/

: KNOCKOUT.JS

- Knockout is a MVVM (Model-View-ViewModel) library. MVVM is an extension of MVC, originally created by Microsoft. Knockout allows us to do UI bindings declaratively rather than using JS.

http://knockoutjs.com/



: EXERCISES

- Prompt for amount, interest rate and number of years. Now calculate the Simple Interest using the values, and show in alert box.
- Check if any given string is palindrome. Use input element to get the string.
- Calculate the area of circle.
- On click of input button ask the user name and display it inside a input text box.
- Copy text of first text box to second text box on every change of value in first text box.

- …Continued
- Allow submission of form only if the user has entered his name (should not be blank) and age (must be greater than or equals to 18).
- Change the color of a div element on mouse over and restore it on mouse out.
- Create a Clock object and encapsulate methods like start and stop for starting and stopping the clock. Implementation must use Prototype pattern and event listener mechanism. Display the clock in some div or span or p element.