**Day 7**

**HTML Events**

An HTML event can be something the browser does, or something a user does.

Here are some examples of HTML events:

* An HTML web page has finished loading
* An HTML input field was changed
* An HTML button was clicked

Often, when events happen, you may want to do something.

JavaScript lets you execute code when events are detected.

HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.

**<!DOCTYPE html>**

**<html>**

**<body>**

**<p>Click the button to display the date.</p>**

**<button onclick="displayDate()">The time is?</button>**

**<script>**

**function displayDate() {**

**document.getElementById("demo").innerHTML = Date();**

**}**

**</script>**

**<p id="demo"></p>**

**</body>**

**</html>**

## Common HTML Events

Here is a list of some common HTML events:

**Event**  **Description**

onchange An HTML element has been changed

onclick The user clicks an HTML element

onmouseover The user moves the mouse over an HTML element

onmouseout The user moves the mouse away from an HTML element

onkeydown The user pushes a keyboard key

onload The browser has finished loading the page

## What can JavaScript Do?

Event handlers can be used to handle, and verify, user input, user actions, and browser actions:

* Things that should be done every time a page loads
* Things that should be done when the page is closed
* Action that should be performed when a user clicks a button
* Content that should be verified when a user inputs data

Many different methods can be used to let JavaScript work with events:

* HTML event attributes can execute JavaScript code directly
* HTML event attributes can call JavaScript functions
* You can assign your own event handler functions to HTML elements
* You can prevent events from being sent or being handled

HTML events can be suppressed as well.

# preventDefault() Event Method

**<!DOCTYPE html>**

**<html>**

**<body>**

**<a id="link" href="http://www.google.com/">Google</a>**

**<script>**

**document.getElementById("link ").addEventListener("click", function(event){**

**event.preventDefault()**

**});**

**</script>**

**</body>**

**</html>**

# stopPropagation() Event Method

The stopPropagation() method prevents propagation of the same event from being called.

Propagation means bubbling up to parent elements *or* capturing down to child elements.

## HTML Web Storage

With web storage, web applications can store data locally within the user's browser.

Before HTML5, application data had to be stored in cookies, included in every server request. Web storage is more secure, and large amounts of data can be stored locally, without affecting website performance.

Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.

Web storage is per origin (per domain and protocol). All pages, from one origin, can store and access the same data.

**HTML Web Storage Objects**

HTML web storage provides two objects for storing data on the client:

* **window.localStorage** - stores data with no expiration date
* **window.sessionStorage** - stores data for one session (data is lost when the browser tab is closed)

Before using web storage, check browser support for localStorage and sessionStorage:

### **WebService**

Web Services is the mechanism or the medium of communication through which two applications / machines will exchange the data irrespective of their underline architecture and the technology.

### **Why is WebService Needed?**

In general, software applications are developed to be consumed by the human beings, where a person sends a request to a software service which in-turn returns a response in human readable format.

In the modern era of technology if you want to build a software application you don't need to build each and everything from scratch. There are lots of readymade services available which you can plug into your application and you can start providing those services in your application.

For example, you want to display weather forecast information you don't need to collect, process and render the data in your application. You can buy the services from the people who already well-established in processing and publishing such kind of data.

Web services allow us to do these kinds of implementations.

WebServices can be called by a Software Application using **SOAP** or **HTTP** protocol.

Web Services can be implemented in different ways, but the following two are the popular implementations approaches.

1. SOAP (Simple Object Access Protocol)
2. REST (Representational State Transfer architecture)

## SOAP

* SOAP is a standard protocol defined by the W3C Standard for sending and receiving web service requests and responses.
* SOAP uses the **XML format to send and receive the request** and hence the data is platform independent data. SOAP messages are exchanged between the provider applications and receiving application within the SOAP envelops.
* As SOAP uses the simple http transport protocol, its messages are not got blocked by the firewalls.

## REST

* REST means REpresentational State Transfer; it is an architecture that generally runs over HTTP.
* The REST style emphasizes the interactions between clients and services, which are enhanced by having a limited number of operations.
* REST is an alternative to SOAP (Simple Object Access Protocol) and instead of using XML for request REST uses simple URL in some cases.
* Unlike SOAP, RESTFUL applications uses HTTP build in headers to carry meta-information.

There are various code that REST use to determine whether user has access to API or not like code **200** or **201** indicates successful interaction with response body while 400 indicates a bad request or the request URI does not match the APIs in the system. All API request parameters and method parameters can be sent via either **POST** or **GET** variables.

## WSDL

WSDL (Web Services Description Language) is an XML based language which will be used to describe the services offered by a web service.

WSDL describes all the operations offered by the web service in the XML format. It also defines how the services can be called,

i.e what input value we must provide and what will be the format of the response it is going to generate for each kind of service.

**What is Web Service Testing?**

Web Services Testing is testing of Web services and its Protocols like SOAP & REST. To test a Webservice you can

1. Test Manually
2. Create your own Automation Code
3. Use an off-the shelf automation tool like SoapUI.

WebService Testing involves following steps -

1. **Understand the WSDL file**
2. **Determine the operations that web service provides**
3. **Determine the XML request format which we need to send**
4. **Determine the response XML format**
5. **Using a tool or writing code to send request and validate the response**

**Differences between Webservices and WEB API (REST full Web service)**

**Web Service**

* Defined by W3C, all communication & data exchange is based on XML
* It has defined standards – WSDL
* You cannot compress the data but you can compress the HTML request
* Example: SOAP

**Web API**

* Web API communication & data exchange could be XML, JSON or plain data
* No defined standard
* You can compress the data
* Example: REST

The SOAP extension can be used to write SOAP Servers and Clients. The SoapServer class provides a server for the [» SOAP 1.1](http://www.w3.org/TR/soap11/) and [» SOAP 1.2](http://www.w3.org/TR/soap12/) protocols.

**<form>**

**<fieldset>**

**<legend>Contact Details</legend>**

**<label for="name">Name:</label>**

**<input id="name"><br />**

**<label for="telephone">Telephone:</label>**

**<input id="telephone"><br />**

**<label for="user-email">Email:</label>**

**<input id="user-email">**

**</fieldset>**

**<fieldset>**

**<legend>User info</legend>**

**<label for="username">Username:</label>**

**<input id="username"><br />**

**<label for="password">Password:</label>**

**<input id="password"><br />**

**</fieldset>**

**<input type="submit" value="Submit now" />**

**</form>**