UNIT V

CLIENT SIDE ESSENTIALS

JAVA SCRIPT OBJECT AND FUNCTIONS :

What is JavaScript ?

- JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.
- JavaScript was first known as **LiveScript,** but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java. JavaScript made its first appearance in Netscape 2.0 in 1995 with the name **LiveScript**. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

The ECMA-262 Specification defined a standard version of the core JavaScript language.

- JavaScript is a lightweight, interpreted programming language.
- Designed for creating network-centric applications.
- Complementary to and integrated with Java.
- Complementary to and integrated with HTML.
- Open and cross-platform

Client-Side JavaScript

- Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser.
- It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content.
- The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts. For example, you might use JavaScript to check if the user has entered a valid e-mail address in a form field.

- The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server.
- JavaScript can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly.

Advantages of JavaScript

The merits of using JavaScript are −

- **Less server interaction** − can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
- **Immediate feedback to the visitors** − don't have to wait for a page reload to see if they have forgotten to enter something.
- **Increased interactivity** − can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
- **Richer interfaces** − can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

JavaScript - Functions

- A function is a group of reusable code which can be called anywhere in your program. Functions allow a programmer to divide a big program into a number of small and manageable functions.
- Like any other advanced programming language, JavaScript also supports all the features necessary to write modular code using functions. functions like **alert()** and **write()** .

Function Definition

- Use a function, we need to define it. The most common way to define a function in JavaScript is by using the **function** keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

Syntax

The basic syntax is shown here.

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

```
function functionname(parameter-list) {
statements
}
//-->
</script>
```

Example

Try the following example. It defines a function called sayHello that takes no parameters −

```
<script type = "text/javascript">
  <!--
    function sayHello() {
      alert("Hello there");
    }
  //-->
</script>
```

Calling a Function

To invoke a function somewhere later in the script, you would simply need to write the name of that function .

<html>

```
  <head>
    <script type = "text/javascript">
      function sayHello() {
        document.write ("Hello there!");
      }
    </script>

  </head>

  <body>
    <p>Click the following button to call the function</p>
    <form>
      <input type = "button" onclick = "sayHello()" value = "Say Hello">
```

```
    </form>
    <p>Use different text in write method and then try...</p>
  </body>
</html>
```

Output

Click the following button to call the function

Use different parameters inside the function and then try...

Say Hello

Hello!.....

The return Statement

- A JavaScript function can have an optional **return** statement. This is required if you want to return a value from a function. This statement should be the last statement in a function.
- For example, you can pass two numbers in a function and then you can expect the function to return their multiplication in your calling program.

Example

Try the following example. It defines a function that takes two parameters and concatenates them before returning the resultant in the calling program.

```
<html>
  <head>
    <script type = "text/javascript">
      function concatenate(first, last) {
        var full;
        full = first + last;
        return full;
      }
```

```
      function secondFunction() {
        var result;
        result = concatenate('Zara', 'Ali');
        document.write (result );
      }
    </script>
  </head>

  <body>
    <p>Click the following button to call the function</p>
    <form>
      <input type = "button" onclick = "secondFunction()" value = "Call Function">
    </form>
    <p>Use different parameters inside the function and then try...</p>
  </body>
</html>
```

Output

Click the following button to call the function

Use different parameters inside the function and then try...

AzaraAli

JavaScript - Objects

JavaScript is an Object Oriented Programming (OOP) language. A programming language can be called object-oriented if it provides four basic capabilities to developers −

- **Encapsulation** − the capability to store related information, whether data or methods, together in an object.

- **Aggregation** − the capability to store one object inside another object.
- **Inheritance** − the capability of a class to rely upon another class (or number of classes) for some of its properties and methods.
- **Polymorphism** − the capability to write one function or method that works in a variety of different ways.

Objects are composed of attributes. If an attribute contains a function, it is considered to be a method of the object, otherwise the attribute is considered a property.

Object Properties

Object properties can be any of the three primitive data types, or any of the abstract data types, such as another object. Object properties are usually variables that are used internally in the object's methods, but can also be globally visible variables that are used throughout the page.

The syntax for adding a property to an object is −

```
objectName.objectProperty = propertyValue;
```

For example − The following code gets the document title using the **"title"** property of the **document** object.

```
var str = document.title;
```

Object Methods

Methods are the functions that let the object do something or let something be done to it. There is a small difference between a function and a method – at a function is a standalone unit of statements and a method is attached to an object and can be referenced by the **this** keyword.

Methods are useful for everything from displaying the contents of the object to the screen to performing complex mathematical operations on a group of local properties and parameters.

For example − Following is a simple example to show how to use the **write()** method of document object to write any content on the document.

```
document.write("This is test");
```

User-Defined Objects

All user-defined objects and built-in objects are descendants of an object called **Object**.

The new Operator

The **new** operator is used to create an instance of an object. To create an object, the **new** operator is followed by the constructor method.

In the following example, the constructor methods are Object(), Array(), and Date(). These constructors are built-in JavaScript functions.

```
var employee = new Object();
var books = new Array("C++", "Perl", "Java");
var day = new Date("August 15, 1947");
```

The Object() Constructor

A constructor is a function that creates and initializes an object. JavaScript provides a special constructor function called **Object()** to build the object. The return value of the **Object()** constructor is assigned to a variable.

The variable contains a reference to the new object. The properties assigned to the object are not variables and are not defined with the **var** keyword.

Example 1

Try the following example; it demonstrates how to create an Object.

```
<html>
  <head>
    <title>User-defined objects</title>
    <script type = "text/javascript">
      var book = new Object();  // Create the object
      book.subject = "Perl";     // Assign properties to the object
      book.author  = "Mohtashim";
    </script>
  </head>

  <body>
    <script type = "text/javascript">
      document.write("Book name is : " + book.subject + "<br>");
      document.write("Book author is : " + book.author + "<br>");
    </script>
  </body>
```

```
</html>
```

Output

```
Book name is : Perl
Book author is : Mohtashim
```

JQUERY

jQuery is an open source JavaScript library that simplifies the interactions between an HTML/CSS document, or more precisely the Document Object Model (DOM), and JavaScript.

Elaborating the terms, jQuery simplifies HTML document traversing and manipulation, browser event handling, DOM animations, Ajax interactions, and cross-browser JavaScript development.

jQuery is widely famous with its philosophy of **"Write less, do more."** This philosophy can be further elaborated as three concepts:

- Finding some elements (via CSS selectors) and doing something with them (via jQuery methods) i.e. locate a set of elements in the DOM, and then do something with that set of elements.
- Chaining multiple jQuery methods on a set of elements
- Using the jQuery wrapper and implicit iteration

Using jQuery (JS) library on HTML page

There are several ways to start using jQuery on your web site.

1. Use the Google-hosted/ Microsoft-hosted content delivery network (CDN) to include a version of jQuery.
2. Download own version of jQuery from jQuery.com and host it on own server or local filesystem.

Basic syntax for any jQuery function is:

$(selector).action()

- A $ sign is to define/access jQuery
- A (selector) is to "query (or find)" HTML elements in html page
- A jQuery action() is the action to be performed on the selected element(s)

Example:

```
<!DOCTYPE html>
<html>
    <head>
        <script src=
"https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js">
        </script>
        <script>
            $(document).ready(function () {
                $("h2").click(function () {
                    $(this).hover();
                });
            });
        </script>
    </head>
    <body>
        <center>
            <h2 style="color: green;">
                SNSRCAS
            </h2>
        </center>
    </body>
</html>
```

Output:

Advantages:

- Wide range of plug-ins. jQuery allows developers to create plug-ins on top of the JavaScript library.
- Large development community
- It has a good and comprehensive documentation
- It is a lot more easy to use compared to standard javascript and other javascript libraries.
- JQuery lets users develop Ajax templates with ease, Ajax enables a sleeker interface where actions can be performed on pages without requiring the entire page to be reloaded.
- Being Light weight and a powerful chaining capabilities makes jQuery more strong.

Disadvantages:

- While JQuery has an impressive library in terms of quantity, depending on how much customization you require on your website, the functionality may be limited thus using raw javascript may be inevitable in some cases.
- The JQuery javascript file is required to run JQuery commands, while the size of this file is relatively small (25-100KB depending on the server), it is still a strain on the client computer and maybe your web server as well if you intend to host the JQuery script on your own web server.

JavaScript Events

- The change in the state of an object is known as an **Event**. In html, there are various events which represents that some activity is performed by the user or by the browser. When javascript code is included in HTML, js react over these events and allow the execution. This process of reacting over the events is called **Event Handling**. Thus, js handles the HTML events via **Event Handlers**.
- **For example**, when a user clicks over the browser, add js code, which will execute the task to be performed on the event.

Some of the HTML events and their event handlers are:

Mouse events:

Event Performed	Event Handler	Description
Click	onclick	When mouse click on an element
Mouseover	onmouseover	When the cursor of the mouse comes over the element
Mouseout	onmouseout	When the cursor of the mouse leaves an element
Mousedown	onmousedown	When the mouse button is pressed over the element
Mouseup	onmouseup	When the mouse button is released over the element
Mousemove	onmousemove	When the mouse movement takes place.

Keyboard events:

Event Performed	Event Handler	Description
Keydown & Keyup	onkeydown & onkeyup	When the user press and then release the key

Form events:

Event Performed	Event Handler	Description
Focus	onfocus	When the user focuses on an element
Submit	onsubmit	When the user submits the form
Blur	onblur	When the focus is away from a form element
Change	onchange	When the user modifies or changes the value of a form element

Window/Document events

Event Performed	Event Handler	Description
Load	onload	When the browser finishes the loading of the page
Unload	onunload	When the visitor leaves the current webpage, the browser unloads it
Resize	onresize	When the visitor resizes the window of the browser

Click Event

```
<html>
<head> Javascript Events </head>
<body>
<script language="Javascript" type="text/Javascript">
   <!--
   function clickevent()
   {
      document.write("This is Java");
   }
   //-->
</script>
<form>
<input type="button" onclick="clickevent()" value="Who's this?"/>
</form>
</body>
</html>
```

OUTPUT

Javascript Events
Who is this ?
This is a java

MouseOver Event

```
<html>
<head>
<h1> Javascript Events </h1>
</head>
```

```
<body>
<script language="Javascript" type="text/Javascript">
   <!--
   function mouseoverevent()
   {
      alert("This is JavaTpoint");
   }
   //-->
</script>
<p onmouseover="mouseoverevent()"> Keep cursor over me</p>
</body>
</html>
```

XML DOM

What is XML DOM

- DOM is an acronym stands for Document Object Model. It defines a standard way to access and manipulate documents. The Document Object Model (DOM) is a programming API for HTML and XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated.
- As a W3C specification, one important objective for the Document Object Model is to provide a standard programming interface that can be used in a wide variety of environments and applications. The Document Object Model can be used with any programming language.
- XML DOM defines a standard way to access and manipulate XML documents.

What does XML DOM

- The XML DOM makes a tree-structure view for an XML document.
- can access all elements through the DOM tree.
- We can modify or delete their content and also create new elements. The elements, their content (text and attributes) are all known as nodes.
- For example, consider this table, taken from an HTML document:

```
<TABLE>
<ROWS>
<TR>
<TD>A</TD>
<TD>B</TD>
</TR>
<TR>
<TD>C</TD>
```

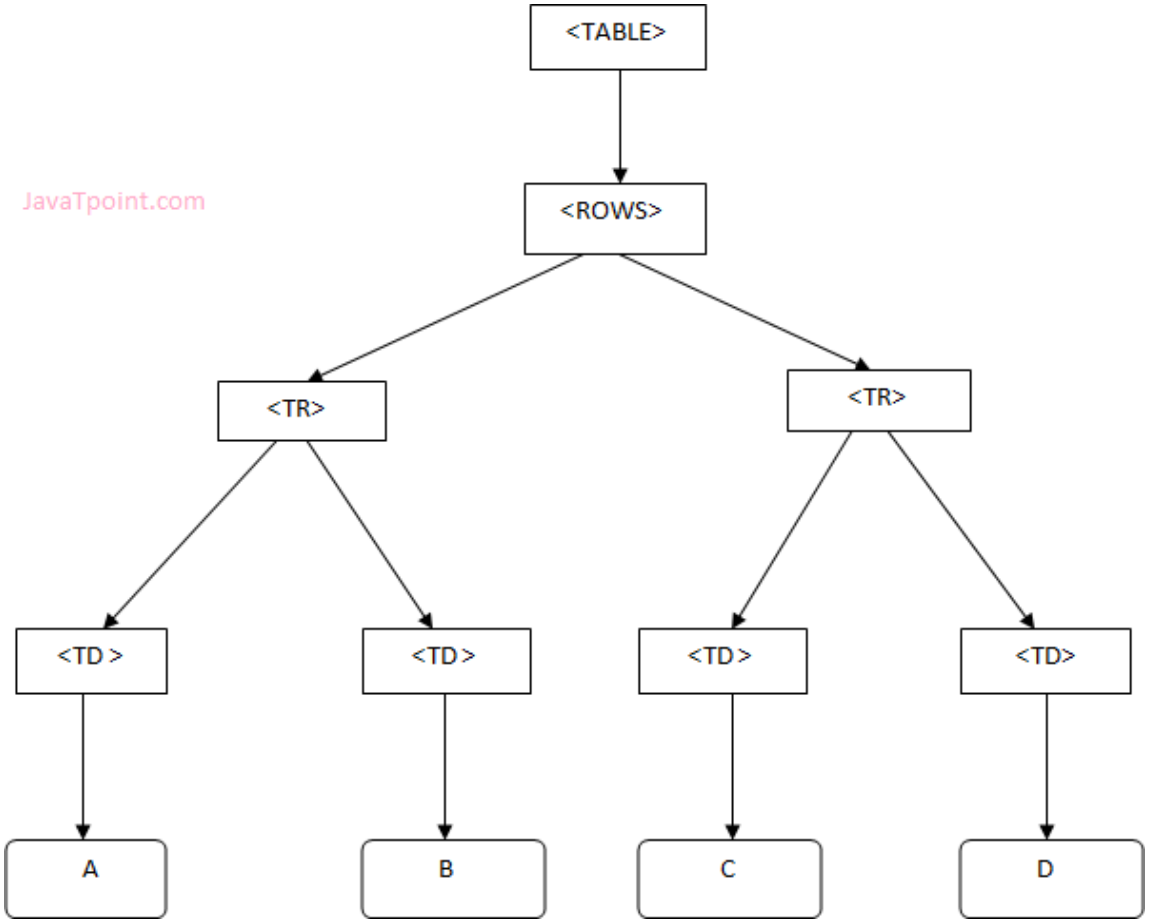
<TD>D**</TD>**

</TR>

</ROWS>

</TABLE>

1. The Document Object Model represents this table like this:



XML DOM Example : Load XML File

- Let's take an example to show how an XML document ("note.xml") is parsed into an XML DOM object.
- This example parses an XML document (note.xml) into an XML DOM object and extracts information from it with JavaScript.
- Let's see the XML file that contains message.

note.xml

<?xml version="1.0" encoding="ISO-8859-1"**?>**

<note>

<to>sonoojaiswal@javatpoint.com**</to>**

<from>vimal@javatpoint.com**</from>**

<**body**>Hello XML DOM</**body**>

</**note**>

What is AJAX?

- AJAX stands for **A**synchronous **Ja**vaScript and **X**ML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script.
    - Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display.
    - XML is commonly used as the format for receiving server data, although any format, including plain text, can be used.
    - AJAX is a web browser technology independent of web server software.
    - A user can continue to use the application while the client program requests information from the server in the background.
    - Data-driven as opposed to page-driven.

Rich Internet Application Technology

- AJAX is the most viable Rich Internet Application (RIA) technology so far. It is getting tremendous industry momentum and several tool kit and frameworks are emerging. But at the same time, AJAX has browser incompatibility and it is supported by JavaScript, which is hard to maintain and debug.

AJAX is Based on Open Standards

AJAX is based on the following open standards −

- Browser-based presentation using HTML and Cascading Style Sheets (CSS).
- Data is stored in XML format and fetched from the server.
- Behind-the-scenes data fetches using XMLHttpRequest objects in the browser.
- JavaScript to make everything happen.

Manipulating the Web using Jquery

- The DOM, a structure the web browser is built upon to relate with webpages. It generates information about your webpage by interacting with the HTML file.

JQUERY?

- As stated earlier, it is a JavaScript library and the most popular one, designed to simplify workflow for web developers. As written on the official website

"jQuery is a fast, small, and feature-rich JavaScript library."

- This method helps developers to Select Element, Manipulate Element, Create Element, Add Event Listener, Animate Elements, Add Effects, and Make HTTP Request.

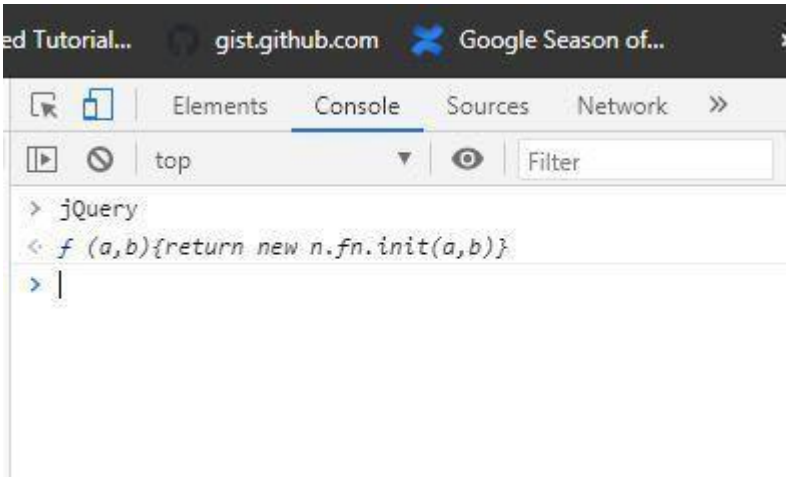
WHY USE JQUERY?

- JQuery is a great tool because if it will make your workflow faster then go ahead including it.
- Selecting an element with “document.querySelectorAll()” is considered to be lengthy, with JQuery this can be fixed just by using the “$(dollar sign)” to select elements. It also makes your code shorter and clear, for example, check out the block of code with and without Jquery.

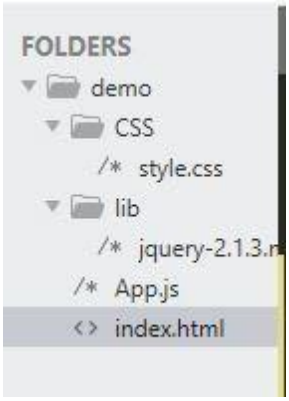
```
// hide and show element h1 with Jquery
$(h1).hide()

$(h1).show()

//hide and show element h1 without Jquery
var h1 = document.getElementById("h1");;

h1.style.display = "none";

h1.style.display = "";
```

USING JQUERY TO MANIPULATE A WEB APP?

- Just added the mini version of JQuery to our project, so time to check if JQuery is actually included. And we added it in our head tag so it runs first before the App runs.



To wrap this up on how to include JQuery for our project, either way, works, but we need a script tag and a source attribute with value such as device directory or CND link.



- The Web Speech API aims to enable web developers to provide, in a web browser, speech-input and text-to-speech output features that are typically not available when using standard speech-recognition or screen-reader software.

- The API itself is agnostic of the underlying speech recognition and synthesis implementation and can support both server-based and client-based/embedded recognition and synthesis.

- The API is designed to enable both brief (one-shot) speech input and continuous speech input. Speech recognition results are provided to the web page as a list of hypotheses, along with other relevant information for each hypothesis.

- All portions of that report may be considered informative with regards to this document, and provide an informative background to this document. This specification is a fully-functional subset of that report

This section is non-normative.

- Voice Web Search
- Speech Command Interface
- Domain Specific Grammars Contingent on Earlier Inputs
- Continuous Recognition of Open Dialog
- Domain Specific Grammars Filling Multiple Input Fields
- Speech UI present when no visible UI need be present
- Voice Activity Detection
- Temporal Structure of Synthesis to Provide Visual Feedback
- Hello World
- Speech Translation
- Speech Enabled Email Client
- Dialog Systems
- Multimodal Interaction
- Speech Driving Directions

- Multimodal Video Game
- Multimodal Search

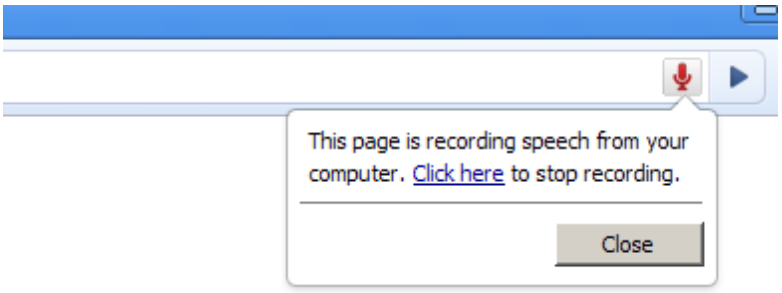
To keep the API to a minimum, this specification does not directly support the following use case. This does not preclude adding support for this as a future API enhancement, and indeed the Incubator report provides a roadmap for doing so.

- Rerecognition

Note that for many usages and implementations, it is possible to avoid the need for Rerecognition by using a larger grammar, or by combining multiple grammars — both of these techniques are supported in this specification.

3. Security and privacy considerations

1. User agents must only start speech input sessions with explicit, informed user consent. User consent can include, for example:
    - User click on a visible speech input element which has an obvious graphical representation showing that it will start speech input.
    - Accepting a permission prompt shown as the result of a call to SpeechRecognition.start.
    - Consent previously granted to always allow speech input for this web page.
2. User agents must give the user an obvious indication when audio is being recorded.
    - In a graphical user agent, this could be a mandatory notification displayed by the user agent as part of its chrome and not accessible by the web page.
    - This UI element must also allow the user to stop recording.



    - In a speech-only user agent, the indication may for example take the form of the system speaking the label of the speech input element, followed by a short beep.
3. The user agent may also give the user a longer explanation the first time speech input is used, to let the user know what it is and how they can tune their privacy settings to disable speech recording if required.

Implementation considerations

This section is non-normative.

1. Spoken password inputs can be problematic from a security perspective, but it is up to the user to decide if they want to speak their password.

2. Speech input could potentially be used to eavesdrop on users. Malicious webpages could use tricks such as hiding the input element or otherwise making the user believe that it has stopped recording speech while continuing to do so.

SpeechRecognition Methods

***start()* method**

- When the start method is called it represents the moment in time the web application wishes to begin recognition.
- When the speech input is streaming live through the input media stream, then this start call represents the moment in time that the service must begin to listen and try to match the grammars associated with this request.
- Once the system is successfully listening to the recognition the user agent must raise a start event. If the start method is called on an already started object (that is, start has previously been called, and no error or end event has fired on the object), the user agent must throw an "InvalidStateError" DOMException and ignore the call.

***stop()* method**

- The stop method represents an instruction to the recognition service to stop listening to more audio, and to try and return a result using just the audio that it has already received for this recognition.
- A typical use of the stop method might be for a web application where the end user is doing the end pointing, similar to a walkie-talkie. The end user might press and hold the space bar to talk to the system and on the space down press the start call would have occurred and when the space bar is released the stop method is called to ensure that the system is no longer listening to the user.

***abort()* method**

The abort method is a request to immediately stop listening and stop recognizing and do not return any information but that the system is done. When the abort method is called, the speech service must stop recognizing.

SpeechRecognition Events

- The DOM Level 2 Event Model is used for speech recognition events. The methods in the EventTarget interface should be used for registering event listeners. The SpeechRecognition interface also contains convenience attributes for registering a single event handler for each event type. The events do not bubble and are not cancelable.

- For all these events, the timeStamp attribute defined in the DOM Level 2 Event interface must be set to the best possible estimate of when the real-world event which the event object represents occurred.
- This timestamp must be represented in the user agent's view of time, even for events where the timestamps in question could be raised on a different machine like a remote recognition service (i.e., in a speechend event with a remote speech endpointer).

***audiostart* event**

Fired when the user agent has started to capture audio.

***soundstart* event**
Fired when some sound, possibly speech, has been detected. This must be fired with low latency, e.g. by using a client-side energy detector. The audiostart event must always have been fired before the soundstart event.

***speechstart* event**

Fired when the speech that will be used for speech recognition has started. The audiostart event must always have been fired before the speechstart event.

***speechend* event**

Fired when the speech that will be used for speech recognition has ended. The speechstart event must always have been fired before speechend.

***soundend* event**

Fired when some sound is no longer detected. This must be fired with low latency, e.g. by using a client-side energy detector. The soundstart event must always have been fired before soundend.

***audioend* event**

Fired when the user agent has finished capturing audio. The audiostart event must always have been fired before audioend.

***result* event**

Fired when the speech recognizer returns a result. The event must use the SpeechRecognitionEvent interface. The audiostart event must always have been fired before the result event.

***nomatch* event**

Fired when the speech recognizer returns a final result with no recognition hypothesis that meet or exceed the confidence threshold. The event must use the SpeechRecognitionEvent interface. The results attribute in the event may contain speech recognition results that are below the confidence threshold or may be null.
The audiostart event must always have been fired before the nomatch event.

***error* event**

Fired when a speech recognition error occurs. The event must use the SpeechRecognitionErrorEvent interface.

***start* event**

Fired when the recognition service has begun to listen to the audio with the intention of recognizing.

***end* event**
Fired when the service has disconnected. The event must always be generated when the session ends no matter the reason for the end.

Speech Synthesis Markup Language (SSML)

```
<speak>
  Here are <say-as interpret-as="characters">SSML</say-as> samples.
  I can pause <break time="3s"/>.
  I can play a sound
  <audio src="https://www.example.com/MY_MP3_FILE.mp3">didn't get your MP3 audio file</audio>.
  I can speak in cardinals. Your number is <say-as interpret-as="cardinal">10</say-as>.
  Or I can speak in ordinals. You are <say-as interpret-as="ordinal">10</say-as> in line.
  Or I can even speak in digits. The digits for ten are <say-as interpret-as="characters">10</say-as>.
  I can also substitute phrases, like the <sub alias="World Wide Web Consortium">W3C</sub>.
  Finally, I can speak a paragraph with two sentences.
  <p><s>This is sentence one.</s><s>This is sentence two.</s></p>
</speak>
```

using SSML

Depending on your implementation, you may need to escape quotation marks or quotes in the SSML payload that you send to Text-to-Speech. The following example shows how to format SSML input included within a JSON object.

```
"{
   'input':{
    'ssml':'<speak>The <say-as interpret-as=\"characters\">SSML</say-as>
        standard <break time=\"1s\"/>is defined by the
        <sub alias=\"World Wide Web Consortium\">W3C</sub>.</speak>'
   },
   'voice':{
     'languageCode':'en-us',
     'name':'en-US-Standard-B',
     'ssmlGender':'MALE'
   },
   'audioConfig':{
     'audioEncoding':'MP3'
   }
 }"
```

Reserve characters

Avoid using SSML reserve characters in the text that is to be converted to audio. When you need to use an SSML reserve character, prevent the character from being read as code by using its escape code. The following table shows reserved SSML characters and their associated escape codes.

Character	Escape code
"	&quot;
&	&amp;
'	&apos;
<	&lt;
>	&gt;

Select a voice

Support for SSML elements

The following sections describe the SSML elements and options that can be used in your Actions.

Note that not all of the elements and options described in the W3 SSML specification are currently supported by Text-to-Speech. This page details which elements and options are available for your application.

<speak>

The root element of the SSML response.

To learn more about the speak element, see the W3 specification.

Example

```
<speak>
  my SSML content
</speak>
```

<break>

An empty element that controls pausing or other prosodic boundaries between words. Using <break> between any pair of tokens is optional. If this element is not present between words, the break is automatically determined based on the linguistic context.

To learn more about the break element, see the W3 specification.

Attributes

Attribute	Description
Time	Sets the length of the break by seconds or milliseconds (e.g. "3s" or "250ms").

Strength	Sets the strength of the output's prosodic break by relative terms. Valid values are: "x-weak", weak", "medium", "strong", and "x-strong". The value "none" indicates that no prosodic break boundary should be outputted, which can be used to prevent a prosodic break that the processor would otherwise produce.

Example

The following example shows how to use the <break> element to pause between steps:

```
<speak>
  Step 1, take a deep breath. <break time="200ms"/>
  Step 2, exhale.
  Step 3, take a deep breath again. <break strength="weak"/>
  Step 4, exhale.
</speak>
```