HTML5 - API

HTML5 is the latest version of HMTL used for presenting and structuring the content. HTML5 provides API's for to develop dynamic and interactive web pages. HTML5 API's can be used to incorporate enhanced functionalities in HTML web page(s).

HTML5 API's are used to cache the application data to access it while offline, store the data at the client side temporarily. HTML5 API's provides geolocation support and helps in creating vector graphics dynamically. HTML5 API's improves the user interaction and resolves cross-browser compatibility issues. HTML5 API's performs validations effectively, implements reusability and handles file system and so on.

At the end of this course, participant will be able to:

Create graphics on webpage using Canvas and SVG APIs

Improve user-interactivity of web-app using Drag-and-Drop API

Use Geolocation API on webpages

Create webpages having offline browsing capability

Create web-apps with storage features

Improve performance of web-apps using Web Worker and Navigation Timing APIs

Perform communication over web using Web Messaging and WebRTC APIs

Implement server-sent events through HTML5 API

Create semantics webpages using MathML API and Microdata

Handle file-system using File API

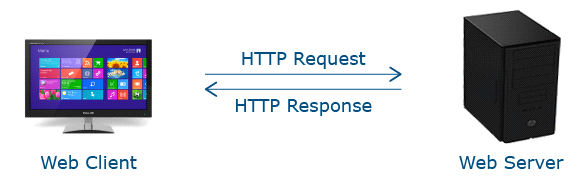
Perform validations using Validation API

Implement reusability using Shadow DOM

Tailor web-app UX depending on browser's capability using Modernizr

Web Storage API

Need of Web Storage API



Web uses HTTP request / response model for all client-server communication.

Whenever user wants new / updated data, it has to send HTTP request to web server.

Web server sends HTTP response and it is rendered by user’s browser.

HTTP request / response model have following drawbacks:

1) Increased network traffic

2) Increased server load.

3) Increased user waiting time

These drawbacks can be overcome by storing data at client side.

For this, Web Storage API is used.

Web Storage API stores data in client machine.

Storing data on client’s machine has following advantages:

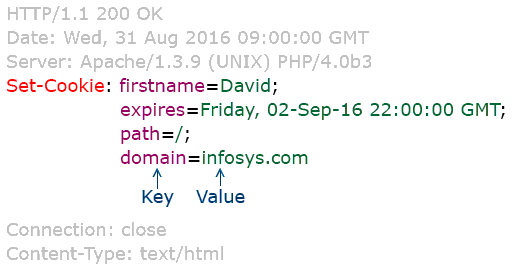
1) Reduces server load – as browser only needs to download updated resources.

2) Improves website performance – as cached resources loads faster.

3) Allows offline browsing – user can access web resources without Internet.

Before Web Storage API, Cookies were used to store web data locally.

Below code-snippet demonstrates how cookie is set using HTTP header.



However, cookies has following drawbacks:

1) Small amount of data (upto 4 KB) can be stored

2) Increased HTTP request-response cycle

3) Security issues

To overcome these drawbacks, Web Storage API is used.

Advantages of Web Storage API

1) Large amount of data can be stored – Atleast 5MB data can be stored in client machine.

2) Reduced network overhead – Stored data is never send back to web server.

Hence, there are no additional HTTP request-response cycle.

3) Secure – Each domain is given a part of memory in client's machine.

Webpages from same domain can share data.

One domain cannot override / access data of another domain.

Hence, it is ensures data security.

Web Storage Objects

You can store data in 2 ways:

1) For particular session i.e. Session Storage

2) Across sessions i.e. Local Storage

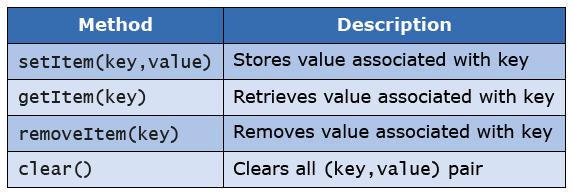
For session storage, sessionStorage object is used.

For local storage, localStorage object is used.

Both these objects are created by Javascript runtime engine of browser.

Web Storage Methods

Web Storage API provides following 4 methods for managing data.

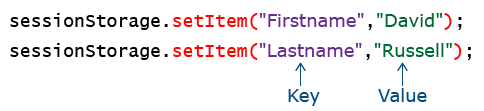


Data is stored in the form of (key, value) pair.

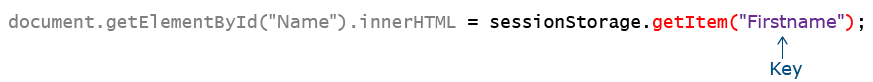
Both key and value are stored as String.

Let’s see how to handle session-related data in detail.

Below code-snippet stores username for a session:

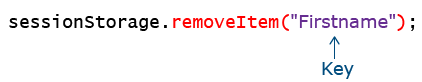


Below code-snippet retrieves username and display it in textbox:



Username is shown in textbox having id "Name".

Below code-snippet removes firstname value:



Session data gets cleared when user closes browser window.

However, you can also explicitly clear data using below code-snippet.

Image



Web Worker API

Need of Web Worker API

You would have observed that its not possible to run two task simultaneously in Javascript.

This is because Javascript executes in a single-threaded environment.

Such serial execution impacts website’s performance when you want to do multiple tasks simultaneously such as manipulating DOM, event-handling and UI rendering.

This can be overcome by executing Javascript in background mode.

Web Worker API allows us to execute Javascript in background mode.

Web Worker API runs Javascript in background while keeping HTML webpage responsive.

That is, while Javascript is running, user can still click on button, select things, etc.

Let’s see how to run Javascript in background by using Web Worker API.

Running Javascript in background is a 3 - step process.

Step 1: Create an external Javascript file which contains code which should run in background.

Step 2: Create web worker object that executes an external Javascript file.

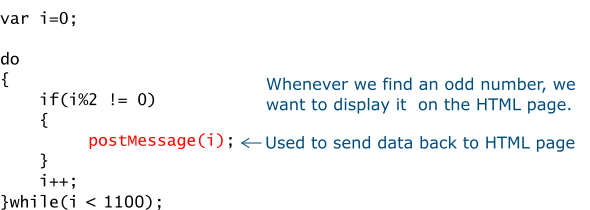
Step 3: Display data sent by an external Javascript on HTML page.

Let’s implement these steps.

Step 1: Create an external Javascript file which contains code to run in background.

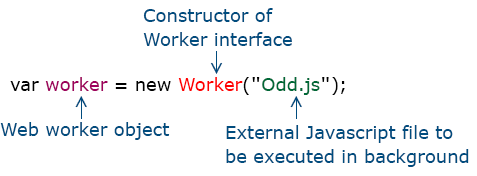
For e.g.: We want to find odd numbers between 1 to 1100.

For this, we have written below code in an external Javascript file (i.e. Odd.js).



Step 2: Create web worker object.

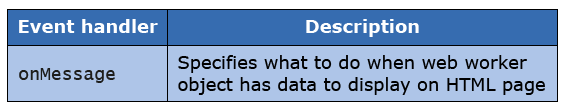
Code-snippet:



Above code-snippet will start executing Odd.js in background.

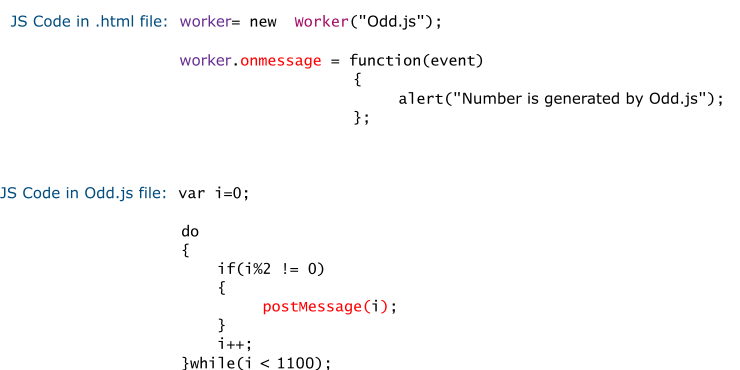
Step 3: Display output sent by an external Javascript on HTML page.

For this, web worker object provide following event handler.



Code-snippet:

Image



Use case scenario for Web Worker API

Web Worker API should not be used for user-input validation scripting.

It should only be used for CPU-oriented scripts.

As a web developer whenever you have to do any CPU oriented scripting, you should always use Web Worker API.

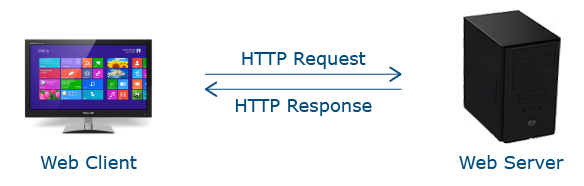
Web Socket API

Need of Web Socket API

Web uses HTTP request / response model for all client-server communication.

When user clicks on link, next webpage is requested from web server.

Web server sends requested webpage and it is rendered by client’s browser.



HTTP request / response cycle is an overhead.

Such an overhead is not acceptable for latency-critical applications.

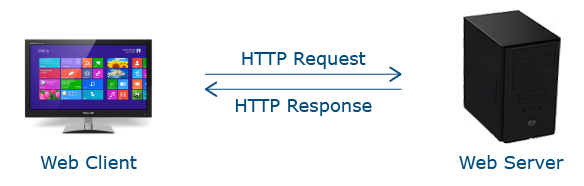
To address this issue, Web Socket API is used.

Introduction to Web Socket API

Web socket is a persistent connection between client and web server.

Both parties can send data any time.

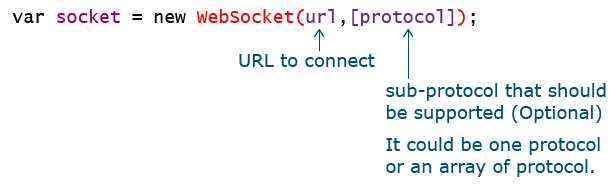
It allows bi-directional communication over a single socket.



Object

For establishing socket connection, we need to create an object of Web Socket Interface.

Code-snippet:

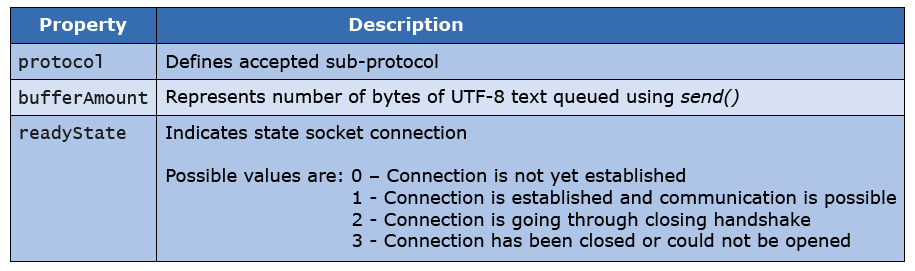


For establishing socket connection, ws (Web Socket) protocol is used.

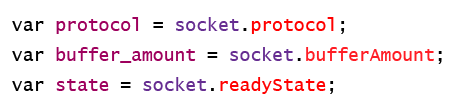
For e.g.: ws://echo.websocket.org

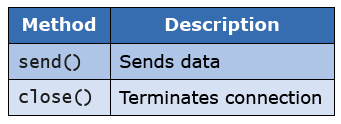
For establishing secure-socket connection, wss (Web Socket Secure) protocol is used.

For e.g.: wss://echo.websocket.org

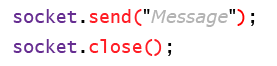


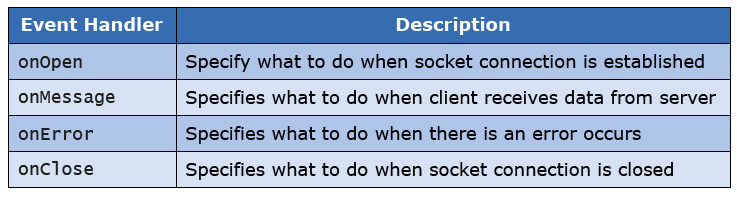
Code-snippet:



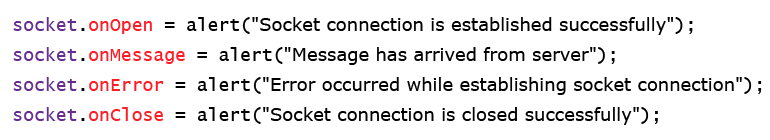


Code-snippet:





Code-snippet:



Code Snippet for Web Socket API



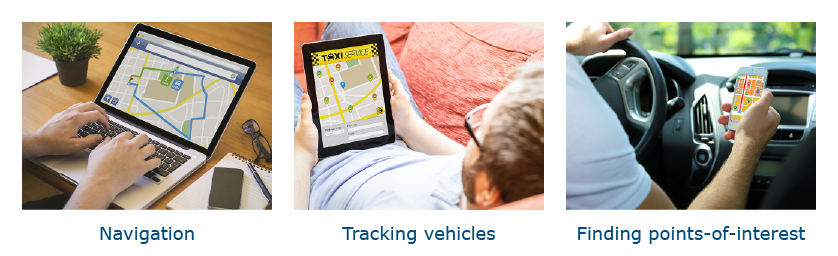
Advantages of Web Socket API

1) Reduced delay (i.e. latency) associated with HTTP request-response model.

2) Reduced network overhead – as whenever web server have updated information, it’ll automatically send it to client. It will not wait for client’s request.

Geolocation API

Need of Geolocation API



All these applications requires information about geographical location of user.

You can build such applications using Geolocation API.

Introduction to Geolocation API

Geolocation API consists of 3 main interfaces called Navigator, Position and PositionError.

Objects of all three interfaces are automatically created by Javascript runtime engine of browser.

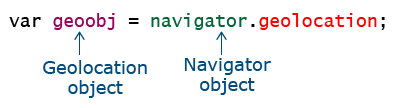
Object

navigator object represents information about user’s browser.

It has many properties and methods.

One of the property is geolocation – It is used to get geolocation object.

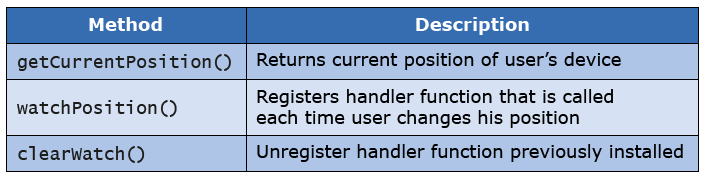
Code-snippet:



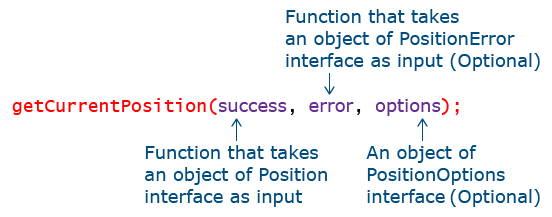
Geolocation object (i.e. geoobj) has methods to collect location-based information.

Method

Geolocation object supports following methods to collect location-based information.



Let’s see syntax of getCurrentPosition() method in detail.



Code Snippet for Geolocation API



Use - Case scenario for Geolocation API

As a web developer, whenever you are working on cab / rickshaw-booking application you should use Geolocation API to provide location-based information / service to your user.

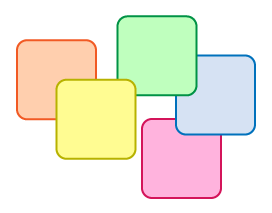
Canvas API

Need of Canvas API

Suppose you want to add this animation to your website. How will you do it?

By using Adobe Flash.

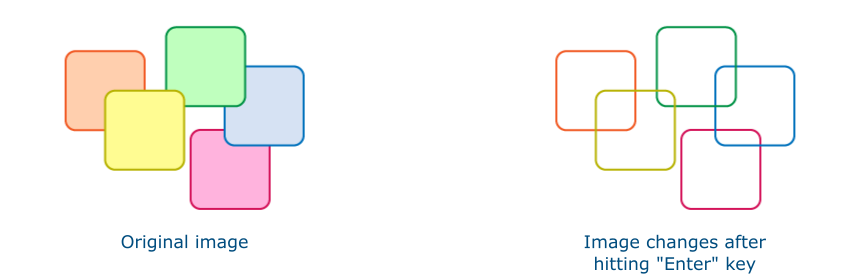
But Flash is not supported by all browsers and also it has become obsolete.



Suppose you want to add this graphics in your webpage. How will you do it?

By using image.

But images need separate HTTP request / response cycle, this increases network overhead.



Suppose you want to change graphics upon user-interaction. How will you do it?

Using Javascript DOM manipulation methods.

But again images increase network overhead and also its JavaScript dependent.

Introduction to Canvas API

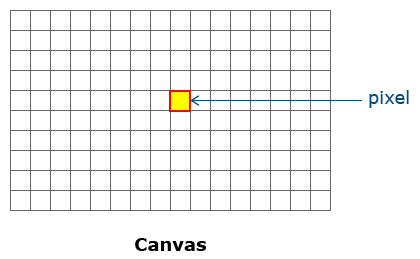
Canvas API has made graphics an intrinsic part of Web.

Hence, using it we can make user-interactive graphics directly on web.

It allows us to draw graphics on the fly by using scripting language (usually Javascript).

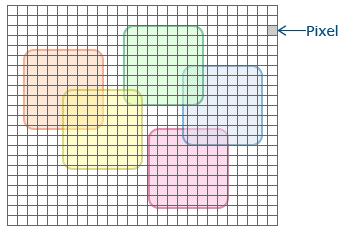
For drawing graphics, it provides us a drawing surface called Canvas.

Canvas is a grid of pixels.



It supports only raster graphics – means all drawings are pixel-based.

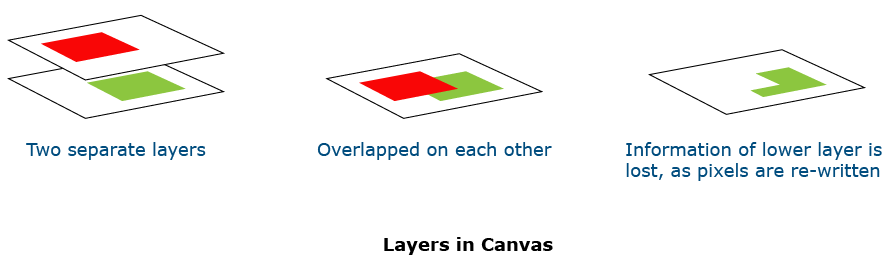
Hence, upon scaling the graphic you can notice individual pixels.



It does not support layers.

Hence, its not possible to overlap one graphical element over another.

If you are drawing one element on top of another, information of below element is lost.



Canvas element provides canvas on webpage.

By default, canvas is rectangular in shape.

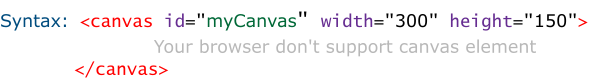
It has no content and no border.

It has 3 attributes.

1) id – Used to refer to canvas element in script. It is mandatory.

2) width – Specifies width of canvas in pixel.

3) height – Specifies height of canvas in pixel.

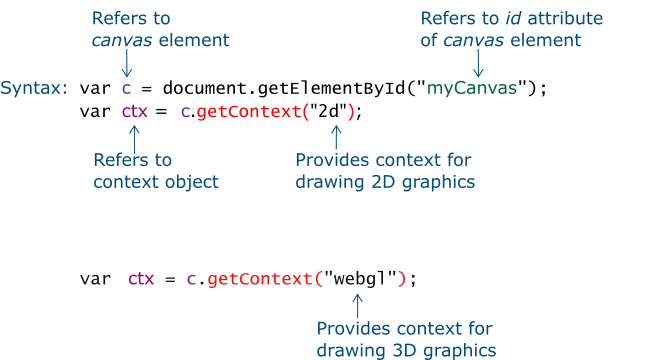


Above code adds blank Canvas on webpage having width 300px and height 150px.

Context object

To draw graphics on canvas, we need a context.

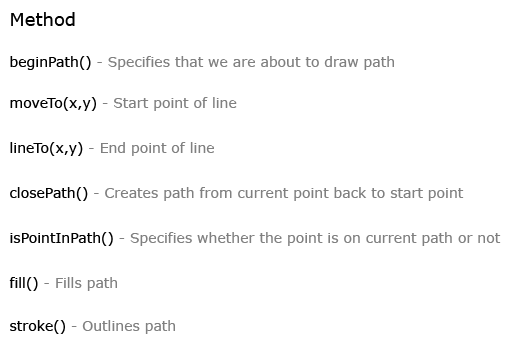
To obtain context, we need to invoke getContext() method on canvas element.



Property and Method

Context object has many properties and methods for drawing graphics on canvas.

Let’s see them in detail.



Line / Path

Make below observation(s) in given code-snippet:

beginPath() starts drawing path

moveTo() specifies starting point of line

lineTo() specifies ending point of line

stroke() outlines drawn line

closePath() closes open path

Carry out below action-item(s) in given code-snippet and observe change(s):

Comment stroke()

Comment closePath()

Change co-ordinates of line

<html>

<body>

<canvas id="myCanvas" width="300" height="500">

Your browser don't support canvas element

</canvas>

</body>

</html>

**SVG API**

To overcome drawbacks of Canvas API, SVG API is used.

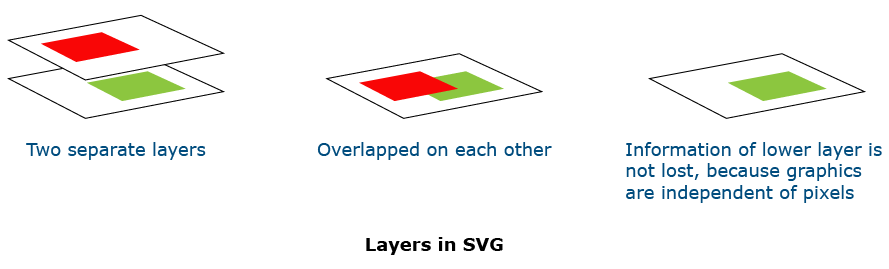
SVG is vector-based – means graphical elements are pixel-independent.

Hence, you can scale your graphical elements to any resolution, without quality degradation.

Need of SVG API

It supports layers.

Hence, you can overlap, one graphical element on top of another without losing details of bottom element.



Introduction to SVG API

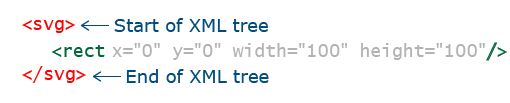
SVG is an XML (Extensible Markup Language) language for describing 2D graphics.

It provides XML elements, to draw graphical elements.

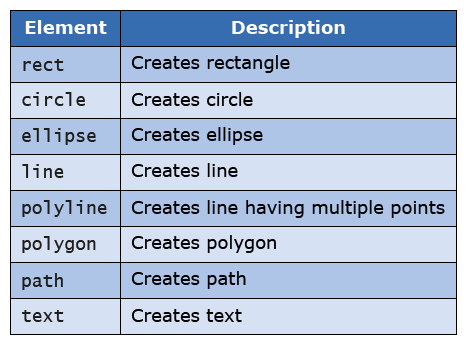
For eg: To draw rectangle, rect element is used.

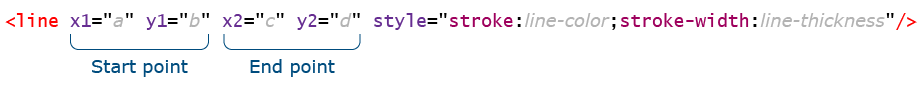
Image

rect element should be enclosed within <svg>…</svg> tags.



Below are few widely used SVG elements:





ProblemStatement:

Make below observations in given code-snippet:

Usage of <line /> element to draw line

Carry out below action-item(s) in given code-snippet and observe change(s):

Change co-ordinates of line

<html>

<body>

<svg width="200" height="400" >

Sorry, your browser does not support inline SVG.

<line x1="10" y1="10" x2="100" y2="10" style="stroke:rgb(0,0,0);stroke-width:2"/>

</svg>

</body>

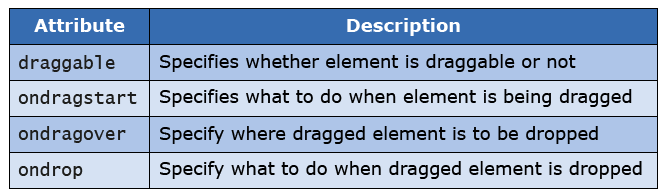
</html>

Drag and Drop API

**Introduction to Drag and Drop API**

Drag and Drop API allows user to drag an object and drop it at different location.

It consists of following 4 attributes:



How to use Drag and Drop API

Making an element draggable is 4 - step process:

Step 1: Make an element draggable.

Step 2: Specify what to do when element is being dragged.

Step 3: Specify where dragged element is to be dropped.

Step 4: Specify what to do when dragged element is dropped.

Step 1: To make an image draggable, set draggable=true.

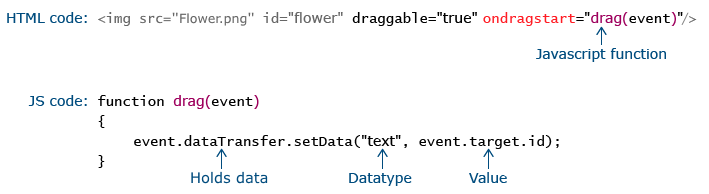
Code-snippet:

Image

Step 2: Specify what to do when element is being dragged.

For this, ondragstart attribute is used.

Code-snippet:



Step 3: Specify where dragged element is to be dropped.

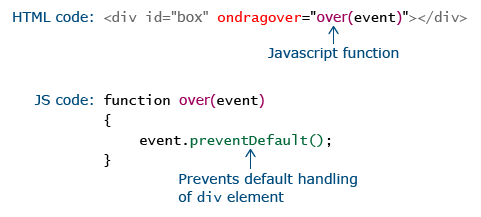
For this, ondragover attribute is used.

By default, data / elements cannot be dropped in other elements.

To allow a drop, we must prevent default handling of an element.

This is done by calling preventDefault() method.

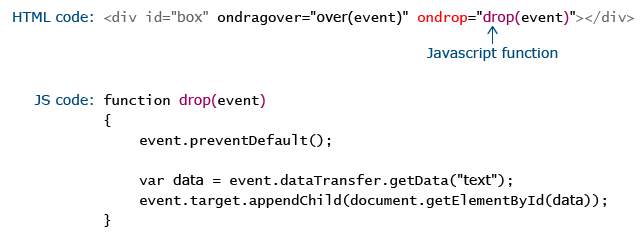
Code-snippet:



Step 4: Specify what to do when dragged element is dropped.

For this, ondrop attribute is used.

Code-snippet:



Code snippet for Drag and Drop API



Use - case scenario for Drag and Drop API

Drag and Drop API is a good tool to create interactive UI.

You would have experienced it while re-arranging / deleting apps on your mobile phone.

Hence, you should consider it, while designing UI of your website.

Offline Browsing API

Need of Offline Browsing API

In absence of Internet access would you like to access website?

Yes. Its possible using Offline Browsing API.

Introduction to Offline Browsing

In offline browsing, user can access cached state of webpage, even when he is not connected to Internet.

It does not require separate download and management of downloaded files.

Let’s see how to implement Offline Browsing API.

Offline Browsing Process

Implementation of Offline Browsing API is a 3 - step process.

Step 1: Create cache-manifest file.

Step 2: Add manifest attribute in an html element of webpage that should be cached.

Step 3: Configure web server.

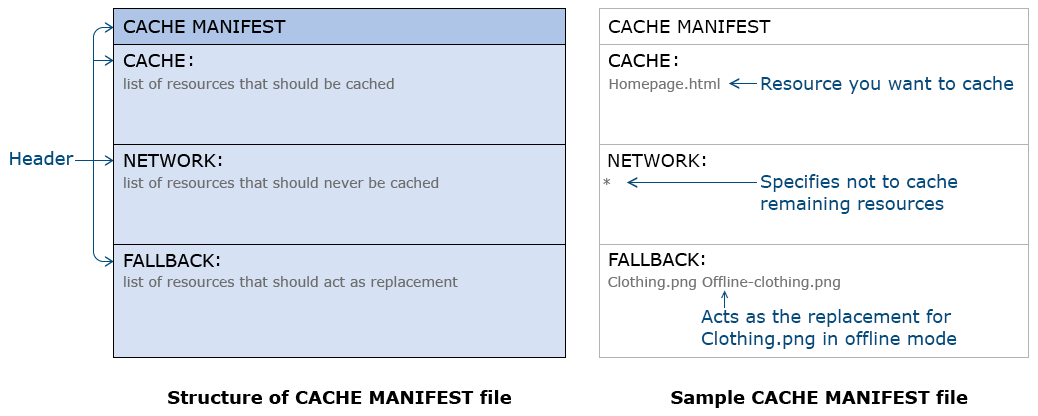
Step 1: Create cache-manifest file.

It's a simple text file.

It can have any extension.

It begins with header CACHE MANIFEST.

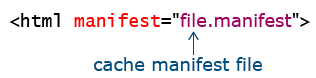
It consists of 3 parts. Each part is identified by header.



Step 2: Add manifest attribute in an html element of webpage that should be cached.

manifest attribute refers to cache-manifest file.

Code-snippet:



Step 3: Configure web server to serve cache manifest file with right MIME type.

MIME type of the cache-manifest file is "text/cache-manifest".

Most of the time, web server will be configured. If not, you need to configure it.

Cautions

1. Only resources which don’t depend on session tracking (like profile, etc.) or that does not need any database resource should be cached.

2. When user reconnects to Internet, he may still be accessing offline resources instead of online resources. Hence, a separate code needs to be written to check for connectivity and update pages. This requires multiple HTTP request - response cycles which leads to network overhead.

Concluding Offline Browsing API

It always a good option, to allow your user to have some level of access to your website while he / she is offline.

This enhances usability of your website.

IndexedDB API

Drawbacks of Web Storage API

Searching for a particular key-value pair is inefficient.

Does not allow to store duplicate values for a key.

Doesn't provide true search capabilities as data is only retrieved via keys.

Not suitable for storing "database".

To overcome these drawbacks, IndexedDB API is used.

Introduction to IndexedDB API

IndexedDB API is another storage API.

It is Javascript-based Object Oriented DBMS.

It uses transactional database to store keys and their corresponding values (one or more values per key).

Features

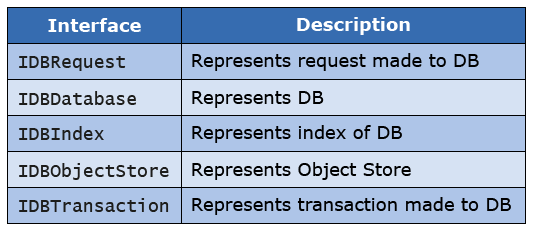
1) Asynchronous operation – Do not make webpage unresponsive, while it is performing its DB operations.

2) Does use SQL – It queries index. It produces cursor which is used to iterate across result set.

3) Secure – Webpages from same domain can only share data.

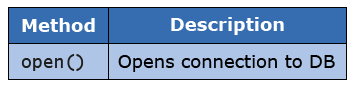
Interfaces

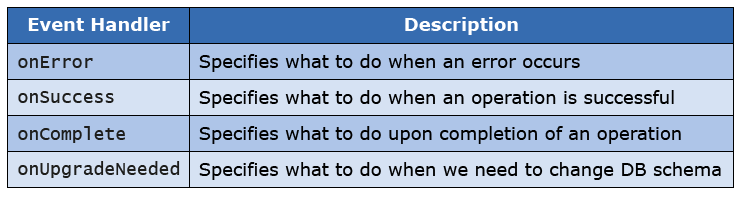
IndexedDB API consists of following 5 interfaces.



Method and Event handler

IndexedDB API consists of following method and event handlers.





How to use IndexedDB API

Using IndexedDB for storing and retrieving data is 5 - step process:

Step 1: Create DB.

Step 2: Create an Object Store.

Step 3: Create Transaction object

Step 4: Connect Transaction object to Object Store.

Step 5: Access data via Transaction object.

Step 1: Create DB.

To create DB, we need to invoke open() method on IndexedDB property of window object.

Code-snippet:



DB version allows us to update DB schema.

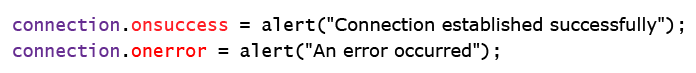
It first checks whether DB exists or not.

If it exists, it establishes a connection to it. Else it creates DB.

open() can throw 2 events: 1) success (Upon establishing connection successfully)

2) error (Upon encountering any error)

To handle these events, onsuccess and onerror event handlers are invoked on IDBRequest object (i.e.: connection variable).



Step 2: Create Object Store.

Data is stored in Object Store.

Suppose we want to store below customer data.



For this, we’ll create an Object Store named "Customers".

To create a new object source, we have to use createObjectStore() method of IDBDatabase Interface.

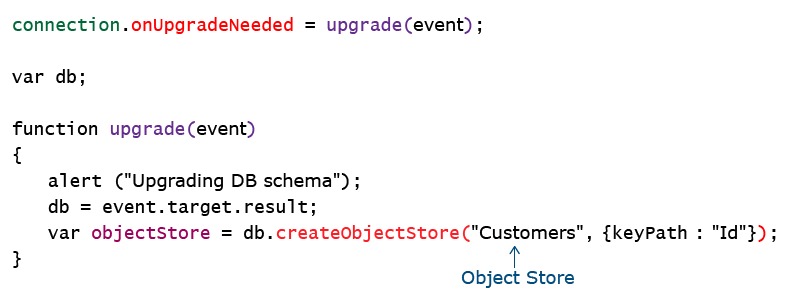
This method returns an object of IDBObjectStore interface.

Creation of new Object Store, modifies DB schema.

Hence, it throws an upgradeNeeded event.

To handle this event, we need to use onUpgradeNeeded event handler.

Code snippet:



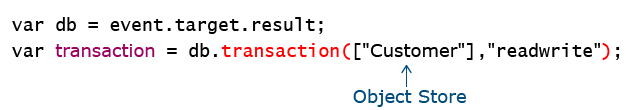
This creates an Object Store named "Customers" with index key as "Id".

Step 3: Create Transaction object.

Now, to add data in an "Customer" Object Store, we need transaction object, with read / write permission.

To create Transaction object, transaction() method of IDBDatabase interface is used.

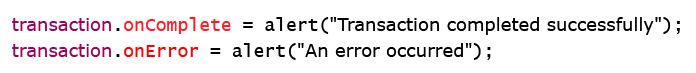
transaction() method returns an object of IDBTransaction interface.



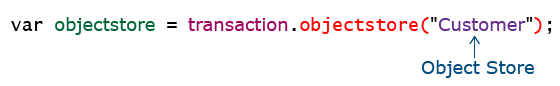
transaction() can throw 2 events: 1) Complete (Upon creating transaction successfully)

2) Error (Upon encountering an error).

To handle these events onComplete and onError, event handlers are used respectively.



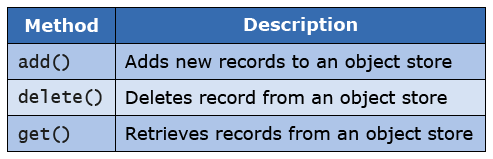
Step 4: To access Object Store using Transaction object, we need to invoke objectStore() method of IDBTransaction interface.

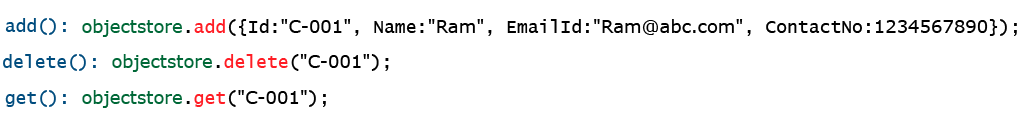


objectstore() method returns an object store that has already been added to the scope of this transaction.

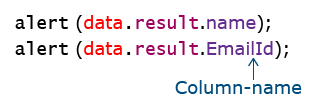
Step 5: Accessing data using Transaction object.

For accessing data, IDBObjectSource provides us following methods:





Below code-snippet demonstrates accessing values of retrieved data:





Use - case scenario for IndexedDB API

As a web developer whenever you are working on realtime application like banking, which demands storage of large amount of structured data at client side, better go for IndexedDB API.

Navigation Timing API

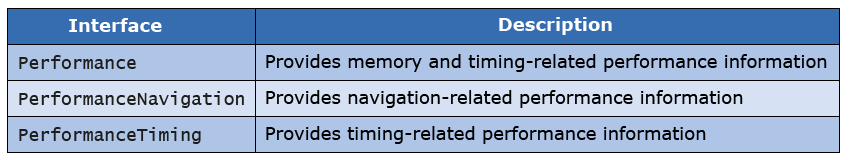
Need of Navigation Timing API

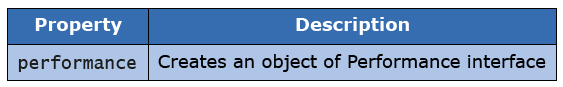
Have you ever wondered why your webpage takes so long to load?

Would you like to know what is causing slow loading of your webpage?

Yes. For this Navigation Timing API is used.

Interfaces and Property





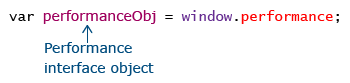
Performance interface

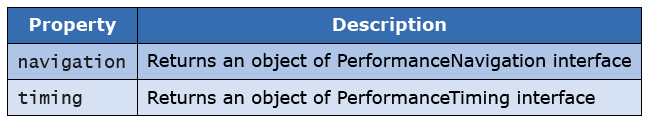
Performance interface provides memory and timing-related performance data.

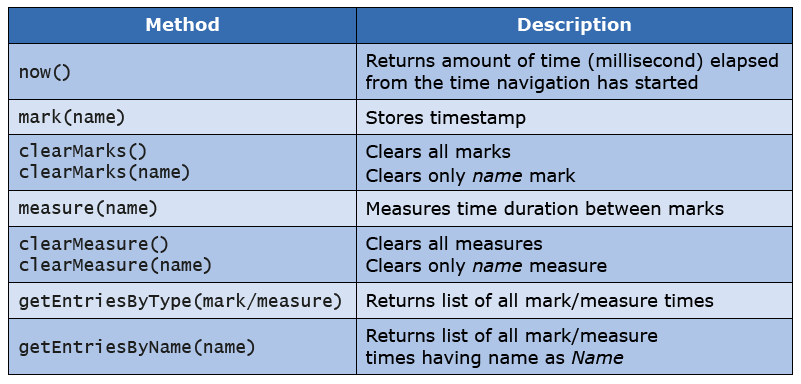
For eg: How much time has elapsed since navigation has started?

How long it takes to reach at particular point while loading webpage?

Below code-snippet demonstrates how to obtain an object of Performance interface.



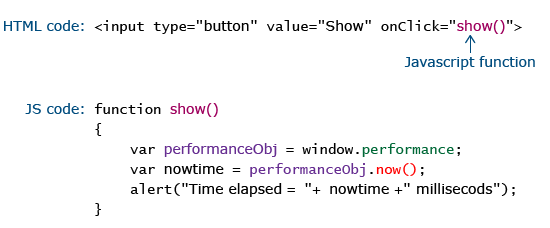




Setting marks at multiple places in website helps you to identify how long it takes to hit that particular mark.

Hence, you can identify where is a loophole in your website.

Below code-snippet demonstrates time elapsed from the point you started navigating webpage.



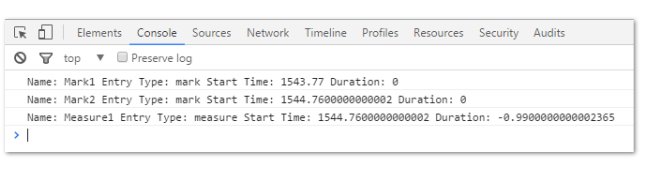
Below code-snippet demonstrates how to establish marks and calculate duration between marks:



Write this code in Notepad. Save it as Mark-Measure-Demo.html.

Run it on Google Chrome browser.

Press F12 -> Select Console tab. You will see below output.



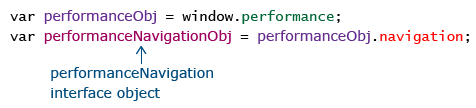
performance Navigation interface

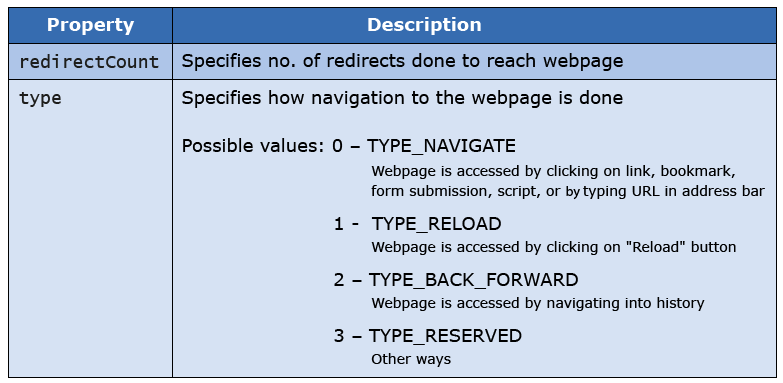
It provides information about how navigation to webpage was done.

For e.g.: How many redirects it took to reach the current webpage?

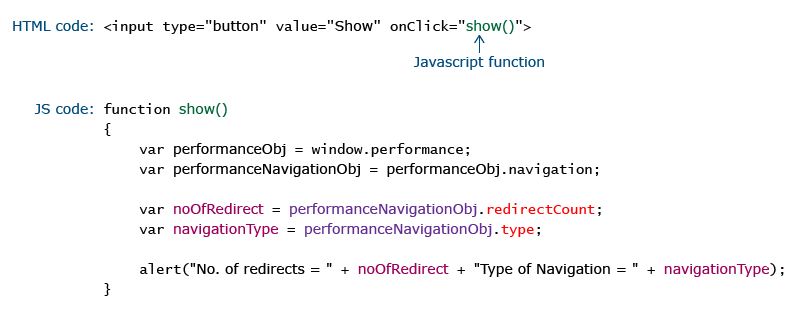
Was webpage accessed by clicking on hyperlink?

Code-snippet for obtaining object of PerformanceNavigation interface:





Below code-snippet demonstrates use of redirectCount and type property for webpage:



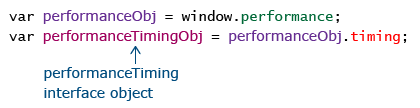
PerformanceTiming interface

It provides information about how long it took for current webpage to load.

For e.g.: When did first redirect to current webpage was started?

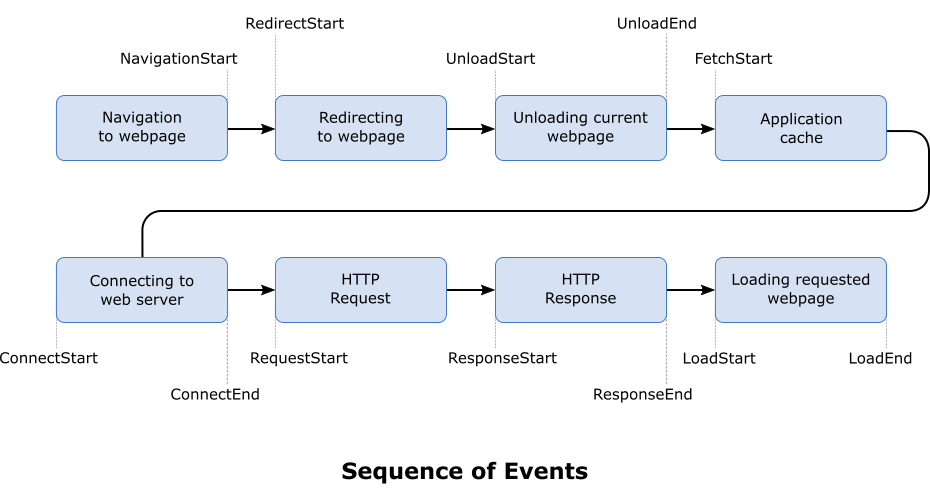
When did last redirect to current webpage ended?

Code-snippet for obtaining object of PerformanceTiming interface:

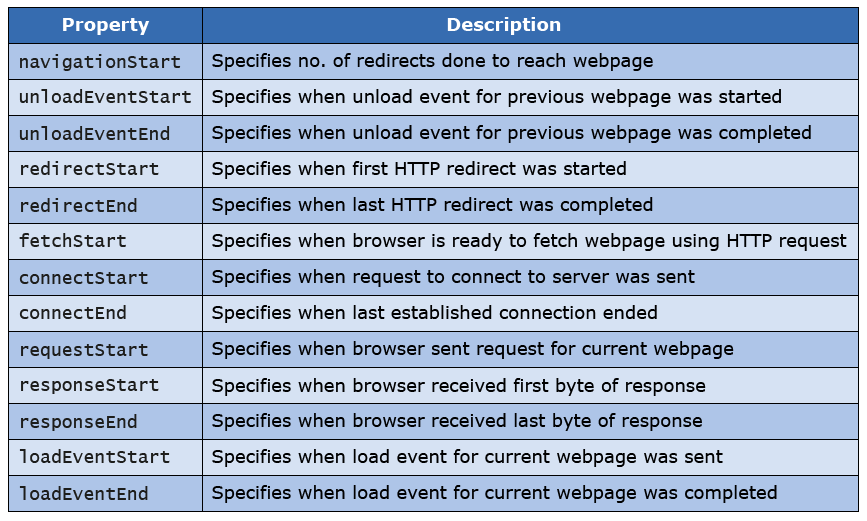


For loading a webpage, no. of events occurs at a backend.

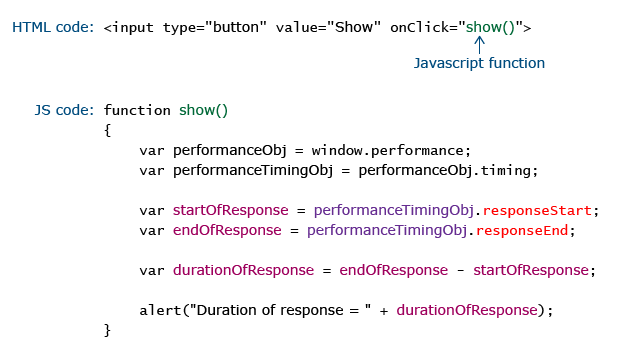
Figure below shows those events in sequence of their occurrence.



PerformanceTiming interface provides readonly property to access amount of time it takes for each event to start and end.



Below code-snippet demonstrates calculating time-duration between starting and completion of HTTP-response:



Concluding Navigation Timing API

Performance of your website is a key factor in determining its success.

High-speed website, always leads to enhanced user-experience which will earn a customer to you.

Hence, while developing website, you should use Navigation Timing API, to optimize performance of your website.

Web Messaging API

Need of Web Messaging API

On Facebook you do real-time text-chatting.

However, you can chat only with other Facebook user.

What if you want to chat with someone who don’t have Facebook account?

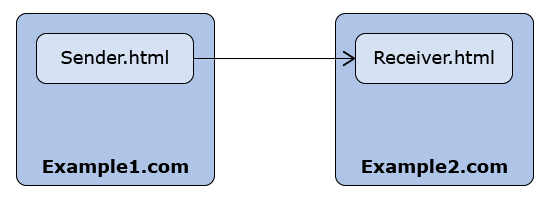
To address this issue, Web Messaging API is used

Introduction to Web Messaging API

Web Messaging API allows webpages to communicate with each other regardless of their domain.

Suppose, there are 2 webpages – Sender.html and Receiver.html in different domains.

Sender.html wants to send message to Receiver.html.



How to use Web Messaging API

Steps for implementation:

For Sender.html

Step 1: Open Receiver.html in new window / tab.

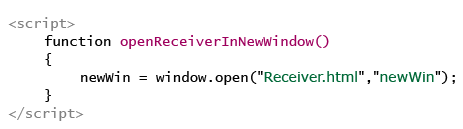
Step 2: Send message to Receiver.html.

Step 3: Implement event handler for handling response received from Receiver.html.

For Receiver.html

Step 1: Implement event handler for handling message send by Sender.html.

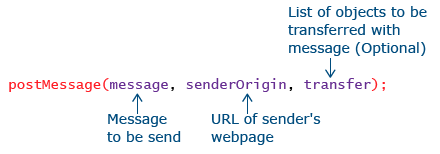
Step 1: Open Receiver.html page in new window / tab.



Step 2: Send message to receiver page.

To send message postMessage() method is invoked on window object.

Syntax

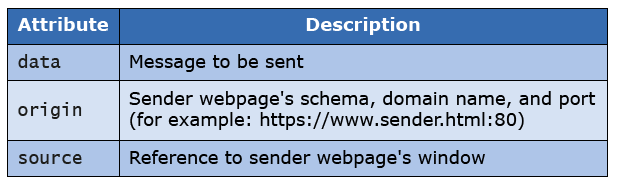


Step 3: Implement event handler for handling response received from Receiver.html.

Code-snippet:



Received message has following 3 readonly attributes.



Step 1: Implement event handler for handling message send by Sender.html.

Code-snippet:



Code Snippet for Web Messaging API



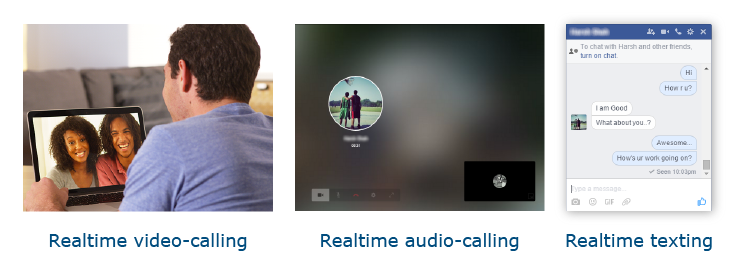
WebRTC API

You do real-time video calling using Skype.

You do real-time audio calling using Facebook.

You also do real-time texting using Facebook.

Need of WebRTC API



These applications have following drawbacks:

Needs to be downloaded and installed in user’s device.

Needs native app or plugin support.

Downloading and installing native app / plugin is a pain, as they needs to be updated every now and then.

Also, deploying, debugging, troubleshooting and maintaining them is difficult.

Is it possible to develop such an application directly on web?

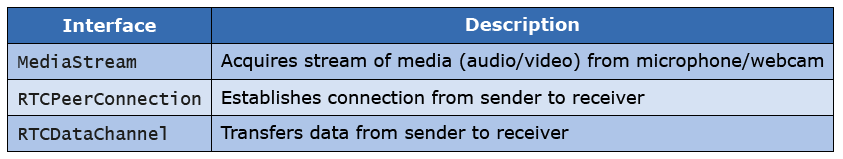
Yes. By using WebRTC API.

Introduction to WebRTC API

WebRTC stands for Web Real Time Communication.

WebRTC API allows us to do real-time communication directly on Web, without any download or native app / plugin support.

Interface



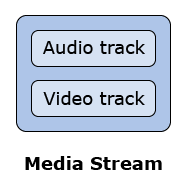
Introduction

MediaStream Interfaces represents "stream" of media.

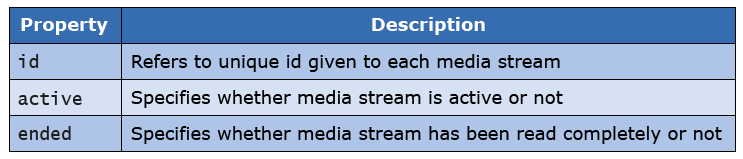
Each "stream" of media is made up of tracks.

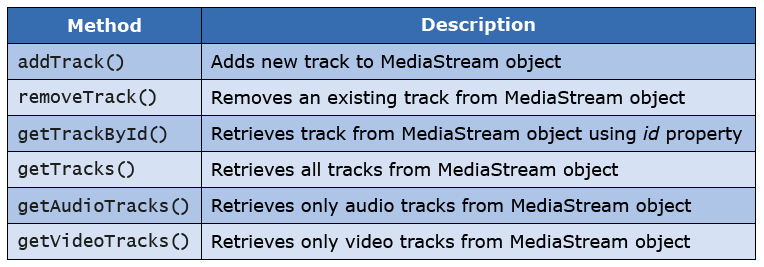
There are 2 types of tracks: 1) Audio track

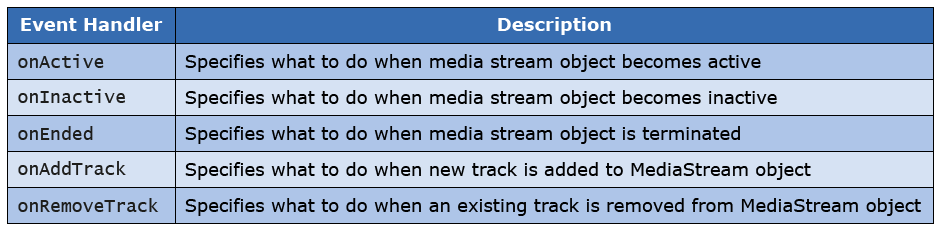
2) Video track



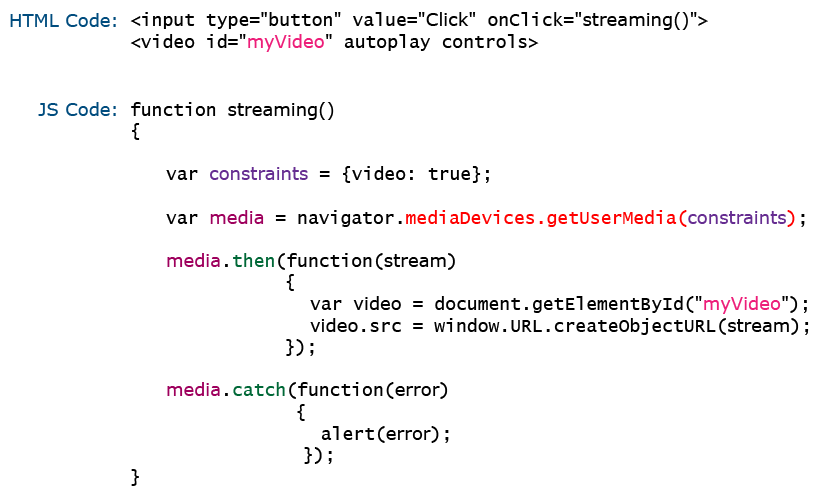
Media stream interface







Below code-snippet demonstrates capturing video using webcam and playing it in browser:



RTC peer connection interface

Using MediaStream interface, we can capture and play media stream locally.

But, what if we want to play captured media stream on remote system?

For this, RTCPeerConnection interface is used.

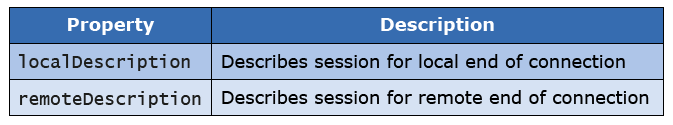
It is used to establish connection between sender and receiver.

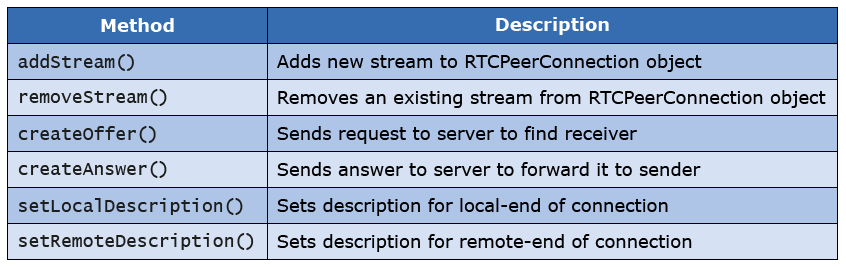
Sender starts connection by calling receiver.

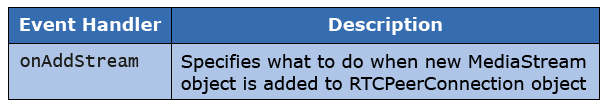
Receiver can either accept or reject the call.

If receiver accepts the call, connection is established.

Both sender and receiver needs to setup the connection.





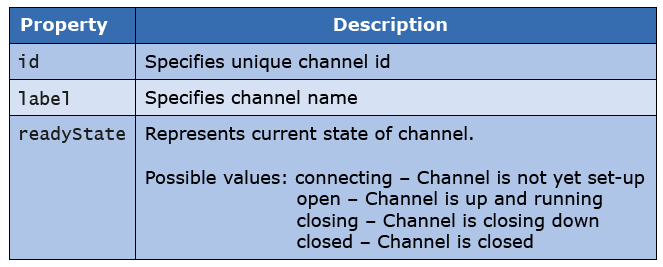


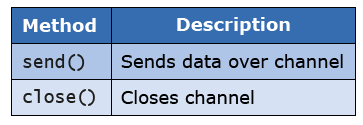
RTC data channel interface

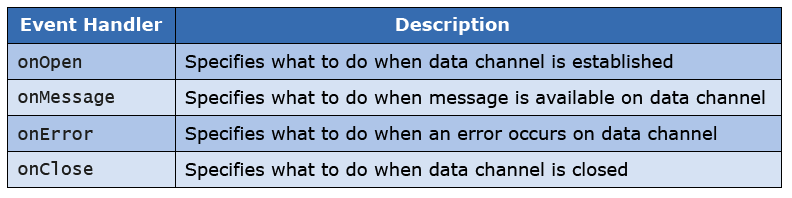
We have learned, how to capture and play real-time media stream both locally and remotely.

Now, let’s see how to communicate real-time text.

For this, RTCDataChannel interface is used.







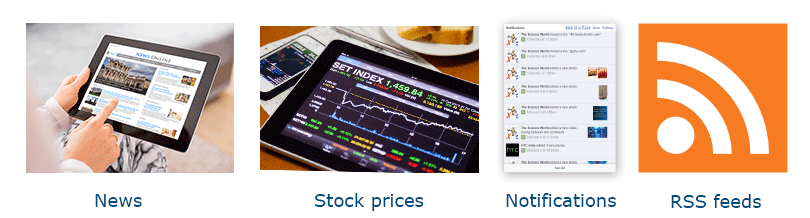
EventSource API

Need of EventSource API

You keep on getting updated news.

You keep on getting updated stock prices.

You keep on getting updated notifications on social networking sites.



In all these applications, server sends data to you as and when updates are available.

Need of EventSource API

In a typical client-server model, client sends request to server by clicking on hyperlink or by submitting form.

Thus event flows from client to server.

Such events are called "client-sent events".



But in all these applications, server is supposed to send data to client automatically.

Hence, event is flowing from server to client.

Such events are called "Server-sent events".

EventSource API is used to implement server-sent events.

How to use EventSource API

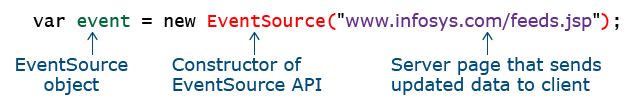
Implementing server-sent events is a 2 - step process:

Step 1: Create an object of EventSource API.

Step 2: Specify what to do when server-sent event occurs.

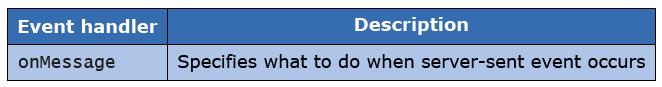
Step 1: Create an object of EventSource API.

Code-snippet:



Step 2: Specify what to do when a server-sent event occurs.

For this, the EventSource object provides following event handler:



code snippet:

Image

Code snippet for EventSource API



Notification API

Need of Notification API

Whenever you receive new email, you get notification.

To provide such desktop notifications Notification API is used.

Introduction to Notification API

Showing desktop notification involves 2 things.

1) User needs to grant permission to website for displaying notifications.

2) Create new notification.

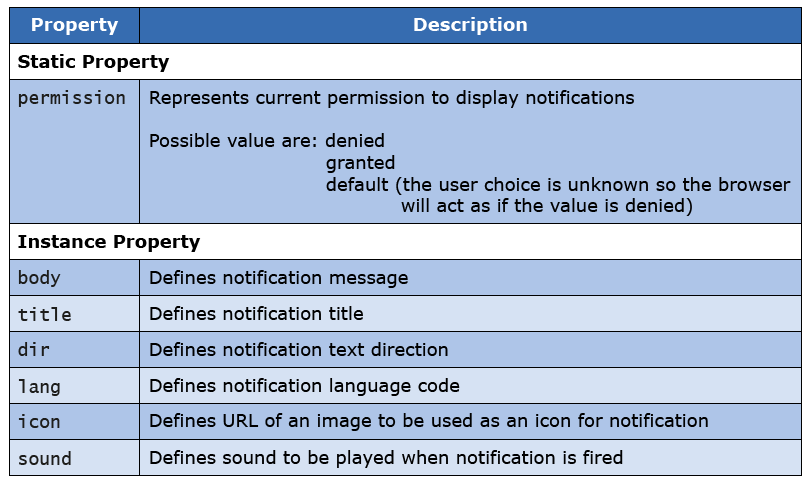
Notification interface

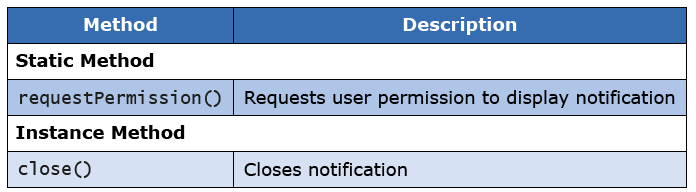
Notifications API provides an interface called "Notification".

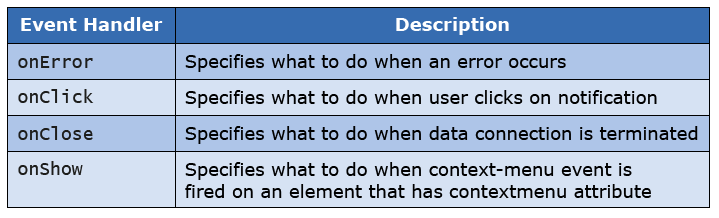
To create notification, we need an object of Notification interface.

Syntax:

Image







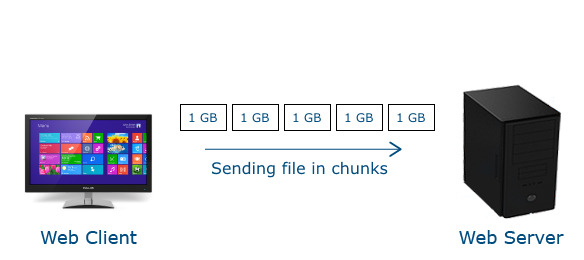
Code snippet for Notification API



File API

Need of File API

Due to unreliable network, it may not be a good option to upload an entire file to web server.



Better choice would be to split file in chunks and upload chunks to web server.

For this, File API is used.

file input element allows you to upload your file on web server.

But you cannot read file’s content / metadata locally.

File API allows you to read file’s content / metadata and render it on browser.

Thus, you can interact with your file locally.

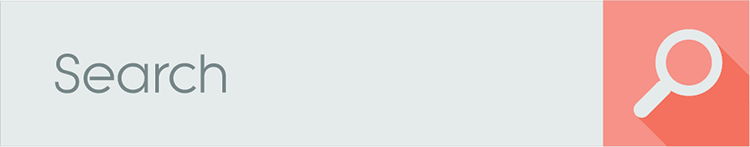
Web components

Suppose in your website you are providing search-box on every page.

Coding same search-box for all webpages is a non-productive task.

Better way is to make search-box reusable i.e. you write code for it once, but you can use it multiple times.

Such reusable widgets can be made using Web components.



Introduction to Web components

Web components consists of 4 technologies: 1) Custom Elements

2) HTML Templates

3) Shadow DOM

4) HTML Imports

Modernizr

Need of Modernizr

Activity: Open <www.bostonglobe.com> in Chrome, IE, Mozilla, Opera and Safari browser

Observe webpage in each browser.

You will find that same webpage appears different in different browser.

This happens because each browser has its own capabilities.

Also, latest version of browser have some additional features which are absent in older versions.

These differing capabilities leads to different output of same HTML code.

If before rendering webpage, you can determine capabilities of your user’s browser, then you can tailor user’s experience of your website accordingly.

For this, Modernizr is used.

Introduction to Modernizr

Modernizr is a JavaScript library.

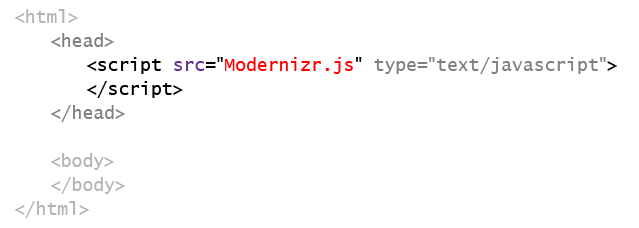
It is used to detect which HTML features are supported by your user’s browser.

It does this by examining how your user's browser responds to series of tests.

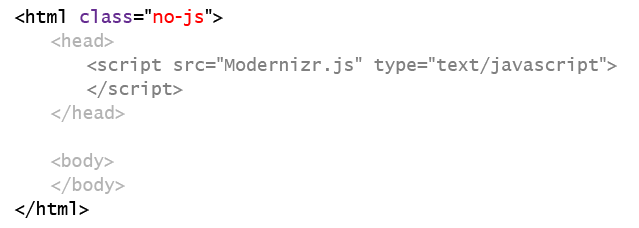
Tests are conducted at the time of page load.

Code snippet for Modernizr

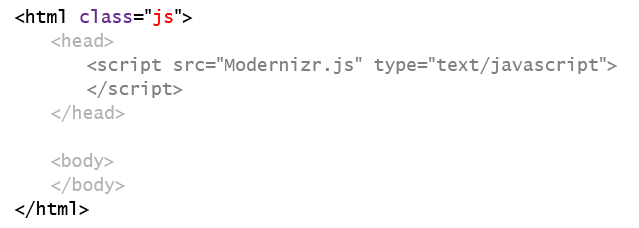
To use Modernizer, you need to include it in your HTML page.



Next, include class="no-js" in html element.



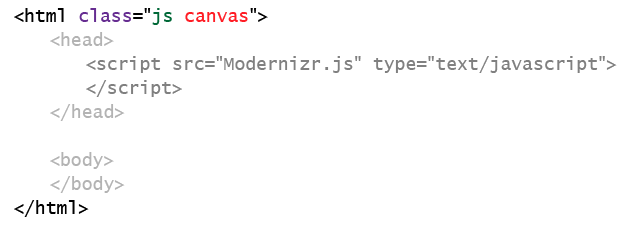
When Modernizr runs, it replaces class="no-js" with class="js".



Then, as and when it detects feature, it adds its corresponding class-name in class attribute.

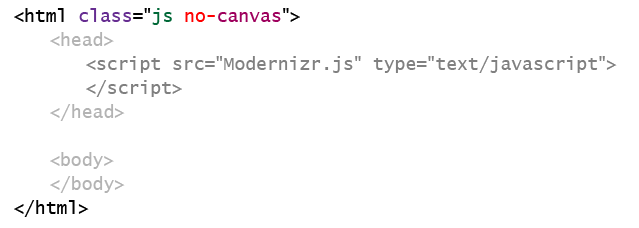
Suppose, it detects canvas element in .html file.

Then, Modernizr will append canvas class-name to value of class attribute.



Suppose, user’s browser does not supports canvas element.

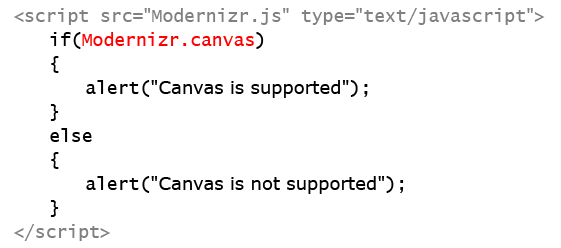
Then, Modernizr will first prefix it by "no-" and then will append it to value of class attribute.



Result of test is in boolean value.

It is stored in an object called Modernizr.

Hence, by accessing value of Modernizr object, you can tailor user-experience of your website.



Microdata

Need of Microdata

Suppose you want to provide following content in your website.



Which HTML elements you’ll use to provide semantic to it?



Here, by using p element you have provided structure to our content.

But, while reading this code-snippet, computer programs cannot make out that the list-items are information about Titanic movie.

To overcome this drawback of HTML elements, Microdata is used.

Introduction to Microdata

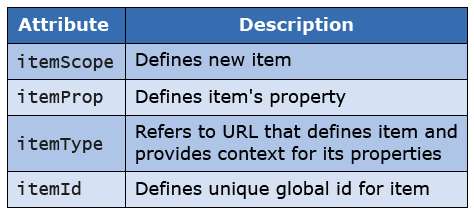
Microdata is a small, targeted vocabulary used to add semantic to HTML markup.

It enhances machine readability of webpages.

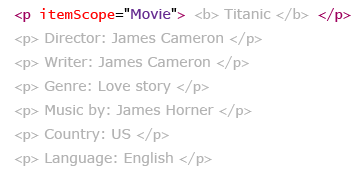
It do not add any visual element on webpage.

It only helps computer programs and assistive technologies to better understand and handle content of your webpage.

Attributes



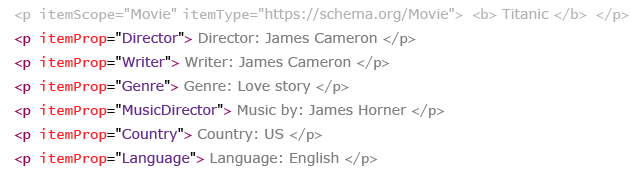
Our content starts from p element. So, we'll create an item within p element using itemScope attribute.



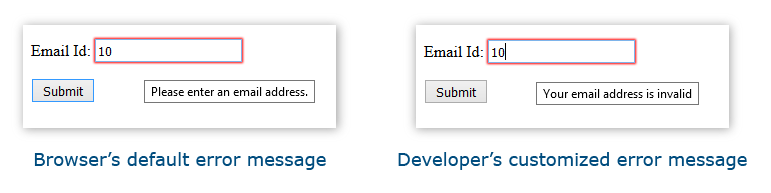
Next, we’ll provide link of webpage that defines item "Movie" and provides context for its properties (such as – Director, Writer, etc.) using itemType attribute.



Next, let's add semantic to each information using itemProp attribute.

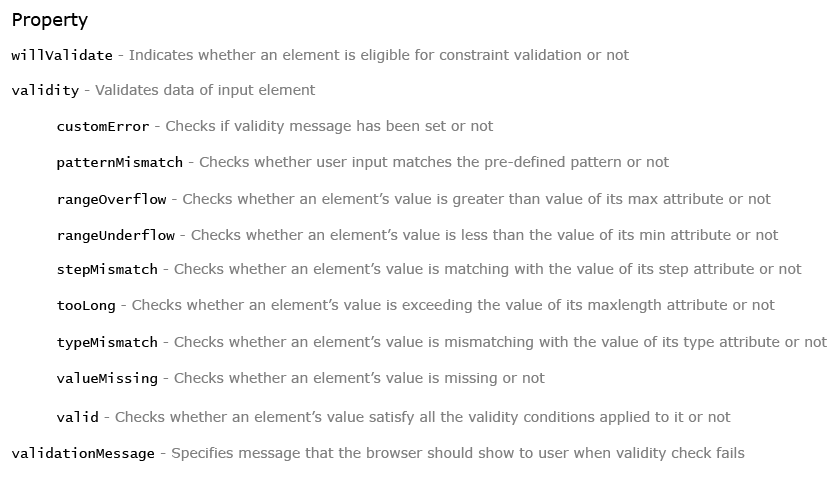


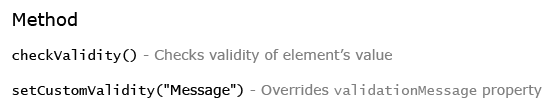
Constraint Validation API

 Writing Javascript code from scratch for complex validations is painful.

Constraint Validation API allows us to do complex validations easily.

Property





willValidate – Indicates whether an element is eligible for constraint validation or not.

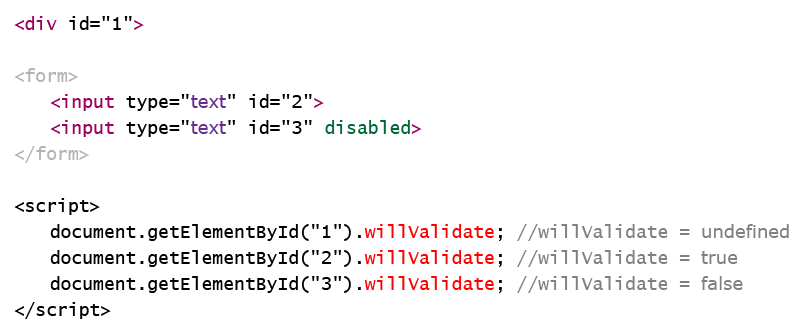
Possible values:

true – Makes an element eligible for validation.

Default value for all form elements (unless an element is disabled using disabled attribute).

false – Makes an element ineligible for validation.

undefined – Default value for all elements whose value cannot be submitted to server. For e.g.: div

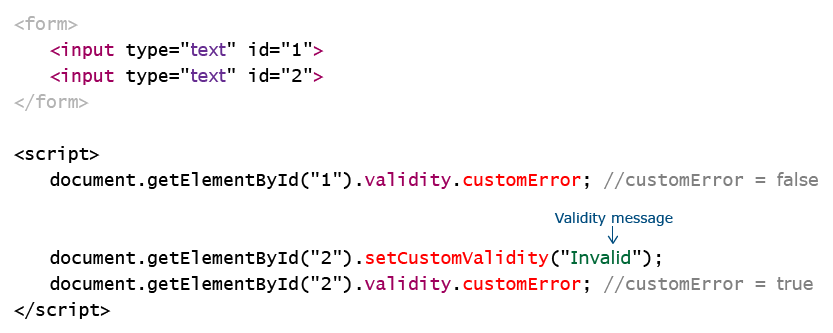


customError – Checks if validity message has been set or not.

Possible values:

true – If validity message is set using setCustomValidity("Message") method

false – Otherwise

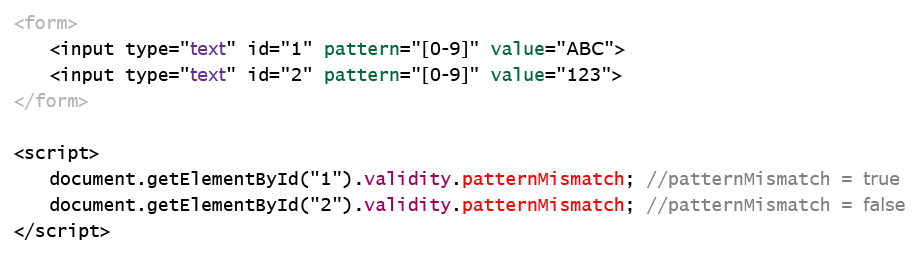


patternMismatch – Checks whether user input matches pre-defined pattern or not.

Possible values:

true – If an element’s value does not matches with pattern defined in pattern attribute.

false – If an element’s value matches with pattern defined in pattern attribute.



rangeOverflow – Checks whether an element’s value is greater than value of its max attribute or not.

Possible values:

true – If an element’s value is greater than value of its max attribute

false – Otherwise

