jQuery

write less, do more.

SUJATABATRA

Introduction to jquery

1

SUJATA BATRA

Topics

- Motivation for jQuery
- History Of jQuery
- What is jQuery
- jQuery vs Custom JavaScript
- jQuery vs Other JavaScript Libraries
- Quick Start jQuery

Motivation for jQuery

- JavaScript has regained its prestige as *rich internet applications and* Ajax gaining popularity

- Rapid Application development is made possible by the use of straightforward JavaScript , CSS and HTML.

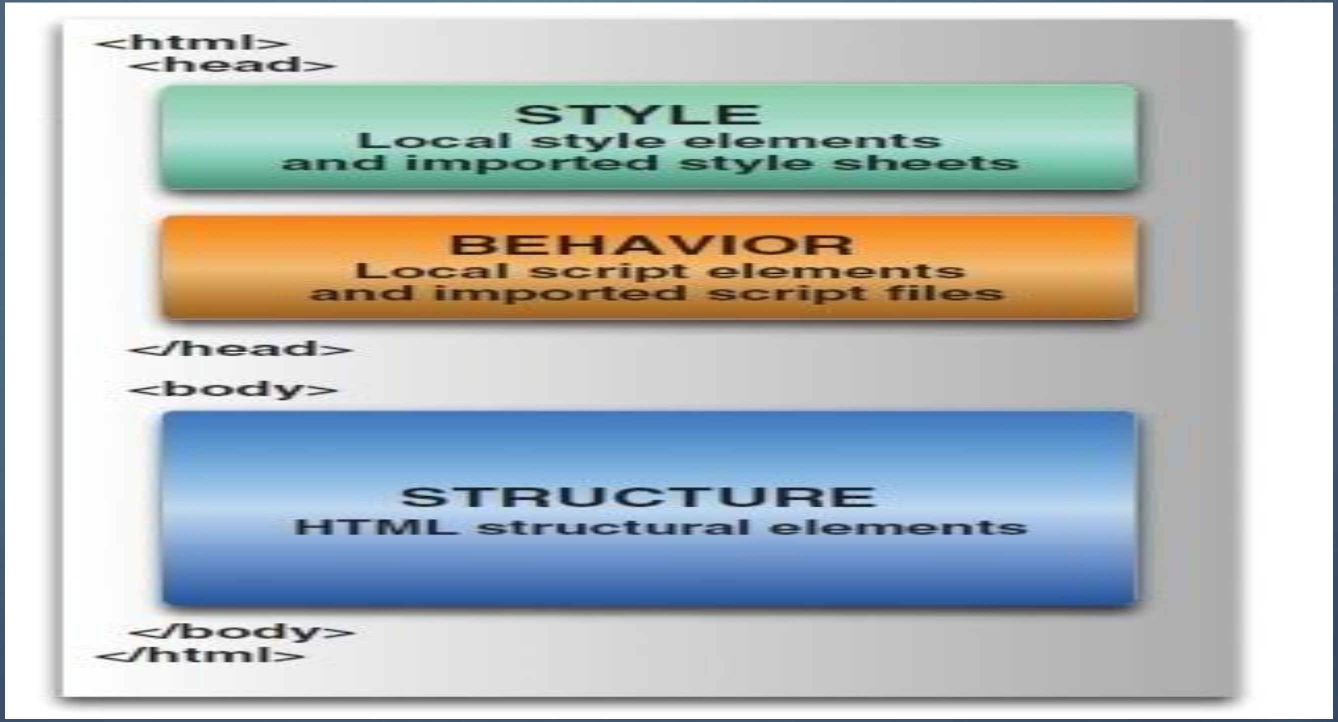
- Replace the cut-and-paste codes by full-featured JavaScript libraries and also to solve difficult cross-browser problems

- Improved patterns for web development.
  - Lazy Initialization,Composite,Façade,Observer etc.,

- *jQuery-driven code is easier to write, simpler to read, and faster to execute than its plain JavaScript equivalent.*

Unobtrusive JavaScript

- Technique to have a clear separation of responsibilities
- A web page with structure, style, *and* behavior each partitioned
- Client-side code with the same level of care and respect as server-side code
- *Extra work—But Not with JQuery.*



<html>
<head>
STYLE
Local style elements and imported style sheets
BEHAVIOR
Local script elements and imported script files
</head>
<body>
STRUCTURE
HTML structural elements
</body>
</html>

What is jQuery

- jQuery is a JavaScript library that simplifies:
  - HTML element selection and document object model (DOM) traversal
  - Element creation, deletion, and modification
  - Event handling
  - Animation
  - Ajax interactions
  - Custom widget integration (date picker, slider, dialogs, tabs, etc…) with jQuery UI
- jQuery is:
  - Free!
  - Open-source
  - lightweight footprint
  - Cross-browser compatible
  - Extensible! can write plugins or pick the one among a large list of existing ones.

History of jQuery

- The First Stable version of jQuery v1.0 was released in 2006 by a team headed by John Resig
- Can be downloaded from the http://jquery.com/download/
- **Minified Version :**
  - compressed version with whitespaces and comments stripped out, shorter variable names in order to preserve bandwidth.
  - Can use normal jquery.js version for Debugging purpose

Adding JQuery

- **Local System**
  - Can add using <script> by downloading the minified version
  - **<script type="text/javascript" src="path/to/jquery.min.js">**
- Adding Jquery with Google CDN
  - https://ajax.googleapis.com/ajax/libs/jquery/3.6.4/jquery.min.js

Adding jQuery with Google CDN

- Google has jQuery libraries on its Content delivery network and allows any website to use it for free.

- **Caching:** The most important benefit is caching. If any previously visited site by user is using jQuery from Google CDN then the cached version will be used. It will not be downloaded again.

- **Reduce Load:** It reduces the load on web server as it downloads from Google server's.

- **Serves fast :** As Google has dozen's of different servers around the web and it will download the jQuery from whichever server closer to the user.

- **Parellel Downloading:** As the js file is on a separate domain, modern browsers will download the script in parallel with scripts on your domain.

The jQuery Function

- $() function returns a JavaScript object containing an array of the DOM elements
- Wraps the collected elements with extended functionality.
- Return group of elements, which can be ready for another action.
- To collect a group of elements, pass the selector to the jQuery function
- **$(selector)**
- Or
- **jQuery(selector)**
- $ is an alias for the jQuery() function

jQuery(callback)

- *Binds a function to be executed when the DOM has finished loading.*

```
$(function() {
    // Document is ready
});
```

- ***Creating a failsafe jQuery code using the $ alias, without relying on the global alias.***

```
jQuery(function( $ ) {

});
```

Document Ready Handler

- The **onload** handler, executes statements after the entire page is fully loaded.
- This delays execution until after the DOM tree is created and all external resources are fully loaded and the page is displayed in the browser window.
- Can trigger the execution of code once the DOM tree has loaded and without waiting for external resources.

```
jQuery(document).ready(function() {
alert(jQuery('p').text());
});

$(function(){alert($ ('p').text());
});
```

First JQUERY-Anonymous Functions

```
<head>
      <script type="text/javascript" src="js/jquery.js"></script>

<script type="text/javascript" >

        $(document).ready(function() {

          $("#msgId").html("<b>Hello From JQuery<b>");
     });

</script>
</head>
<body>
          <div id="msgId"></div>
</body>
```

JQuery with Named Functions

```
<script type="text/javascript" src="js/jquery.js"></script>
 <script type="text/javascript" >
 function setValue()
 {
       $("#msgId").html("Hello From JQuery")
 }

       $(function() {
              $("#echoButton").click(setValue);
                     $("#echoButton").click(setValue());
       });

</script>
<p id="msgId">Example</p>

<input type="button" value="Get TextFld Value" id="echoButton"/>
```

Selection

2

Topics

- Basic Selectors
- Hierarchy Selectors
- Selection By Attribute
- Form Selectors
- Position Filters

Basic Selectors

- Basic jQuery selectors work the same as CSS selectors:

- $("*") - // Selectors return a pseudo-array of jQuery elements

- Selectors start with **$(), representing the global jQuery function**

- It can take following arguments

- **Tag Name:** Represents a tag name available in the DOM.
- **Tag ID:** Represents a tag available with the given ID in the DOM.
- **Tag Class :** Represents a tag available with the given class in the DOM.

- *If more than one element has the same id only the first matched element in the DOM will be returned*

Basic Selectors

- <p id="*id1*">*Java*</p>
- <p id="*id2*" *class*="*style1*">*JEE*</p>
- ***<div>Spring</div>***

=$("#id1").text();

$(".style1").text();

=$(“div“).text();

Controlling the context

- $("", $()) -> function can have a second argument
- First argument -> is a selector,
- Second argument -> denotes the context of the operation.
- To Select the list Item inside a Division Identified with "colLef"
- **$("li", $("#colLef")).css("border", "2px solid red");**

Element to Selection

Context of Selection

Selection By Attribute

- **$('a[href]');**
    - Selects all <a> elements that have an href attribute.
- **$('a[href="Welcome.html"]');**
    - Selects all <a> elements whose href attribute exactly matches Welcome.html.
- **$("div[id^='main']")**
    - Select all div elements Id starting with main
- **$("div[id$='name']")**
    - Select all div elements Id END with name
- **$("a[href*='msdn']")**
    - Select all href elements containing msdn

Working with Attributes

- Can manipulate an element's attributes
- **Set Attribute Value:**
- The **attr(name, value)**
  - set the named attribute onto all elements in the wrapped set using the passed value.
- $("#myimg").attr("src", "/images/jquery.jpg");
- **removeAttr()**

Hierarchy Selectors

- **$('#footer span');**
  - Selects the <span> elements that are descendants of the element with the id footer.
- **$('ul > li');**
  - Selects the <li> elements that are immediate children of <ul> elements.
- **$('h2 + p');**
  - Selects the <p> elements that are **immediately** preceded by an <h2> element.
- **$('h2 ~ p');**
  - Selects **all** <p> elements following an <h2> element that have the same parent as the <h2>element.

Form Selectors

- **:input ->** Input, textarea, select, and button elements
- **:enabled ->** Form elements that are enabled
- **:disabled ->** Form elements that are disabled
- **:checked** -> Radio buttons or checkboxes that are checked
- **:selected ->** Option elements that are selected

- $(":checkbox")implies $("*:checkbox")).

- **var retVal =$("input:text#txtFld1").val();**

- var retVal=$("input:radio[name=rad]:checked").val();

Find Dropdown Selected Item

```
<select name="cities">
   <option value="1">Chennai</option>
   <option value="2">Trichy</option>
   <option value="3">Madurai</option>
</select>
```

```
$("select[name='cities'] option:selected").val()
```

Selecting by Position

- **a:first**
  - This format of selector matches the first <a> element on the page.
- **p:odd**
  - This selector matches every odd paragraph element.
- **p:even**
  - This selector matches every even paragraph element.
- **ul li:last-child**
  - chooses the last child of parent elements. In this example, the last <li> child of each <ul> element is matched
- $('li:nth-child(2)')
  - selects all <li> elements that are the second child of their parent
- $('li:nth-child(2n+1)')
  - (2 x 0) + 1 = 1 = 1st Element
    (2 x 1) + 1 = 3 = 3rd Element
    (2 x 2) + 1 = 5 = 5th Element

Selecting by Position

- Can filter the elements selected based on their position in the collection

- **$('.article:gt(1)');**
  - selects elements with an index number higher than a specified number.
  - List all elements with class article that are greater than index 1

- **$('.article:lt(3)');**
  - From the collection of all elements with the class article, select up to the first three.

Other Selections

- **$(p:contains('java'));**
  - Selects all <p> elements that contain the string “java", either directly or in any of the child elements.
  - The text matching is case sensitive.
- **$('div:has(h2)');**
  - Selects all <div> elements that have at least one <h2> element as a descendant.
- **$('option:not(:selected)');**
  - Selects all <option> elements that are **not** selected.
- **$('p:hidden'); , $('p:visible');**
  - Selects all <p> elements that are hidden/visible.
  - An element is considered hidden if:
    - It has a CSS display value of none
    - It is a form element with type="hidden"
    - Its width and height are explicitly set to 0
    - An ancestor element is hidden, so the element is not shown on the page

jQuery Method Chaining

- Selections return an instance of a jQuery object.
- Returned instance can then invoke methods on the objects to manipulate the HTML elements it represents.
- `$('.status') .css('backgroundColor','yellow') .attr('title', 'Training by Ramesh');`
  - This selects the elements with the class status, sets the background color of the selected elements to yellow, and then sets the title attribute of the selected elements.
- Chaining works with events also
  - `$('#dvContent').addClass('dummy') .css('color', 'red') .fadeIn('slow');  });`

CSS Styling and jQuery

4

Jquery Css

- JQuery CSS selectors can be used to change CSS properties
- CSS Methods do not modify the content of the jQuery object and they are used to apply CSS properties on DOM elements.
- The method for CSS manipulation is css()
- **selector**.css(PropertyName, PropertyValue);
- $("li").css("color", "red")

Css Manipulations

- *To Get the Current Css Style associated with an Element*
- $(“div”).css(“background-color”);

- ***From jQuery 1.9*** : Can have multi-property getter.

- var *propCollection* =
  $("#dvBox").css(["width", "height", "backgroundColor"]);

- *propCollection* will be an array of values.
- { width: "100px", height: "200px", backgroundColor: "#FF00FF" }

Set CSS Properties

- *To Set a New style to an Element*
- $("div").css("float", "left");
- *To Set Multiple style properties*

-
  ```
  $("div").css({                    "color":"blue",
                    "padding": "1em"
                    "margin-right": "0",
                    "marginLeft": "10px"  } );
  ```

Event handling

Event Handling

- The event handling methods are core functions in jQuery.
- Event handlers are method that are called when "something happens" in HTML.
- The term "**triggered (or "fired") by an event**" is often used.

-
```
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide();
  });
});
```

- <p>This is a paragraph.</p>

- <button>Click me</button>

JQuery Events

- **$(*document*).ready(function)**
  - Binds a function to the ready event of a document
- **$(*selector*).click(function)**
  - Triggers, or binds a function to the click event of selected elements
- **$(*selector*).focus(function)**
  - Triggers, or binds a function to the focus event of selected elements
- **$(*selector*).mouseover(function)**
  - Triggers, or binds a function to the mouseover event of selected elements

Animation and effects

Showing and hiding elements

- Showing or hiding elements are simple and pop elements into existence or make them instantly vanish!
- **show()**
  - to show the elements in a wrapped set, and hide() to hide them.
- jQuery hides elements by changing their style.display properties to none.
- If an element in the wrapped set is already hidden, it will remain hidden but still be returned for chaining.

Useful Functions

- .hide()
- .show()
  - `$(“div”).show();`
  - `$(“div”).show(“slow”);`
  - `$(“div”).hide(“fast”);`
- .toggle(func1, func2)
  - first click calls func1, next click executes func2
- .hover(over, out)
  - mouseover, mouseout

Hover Method

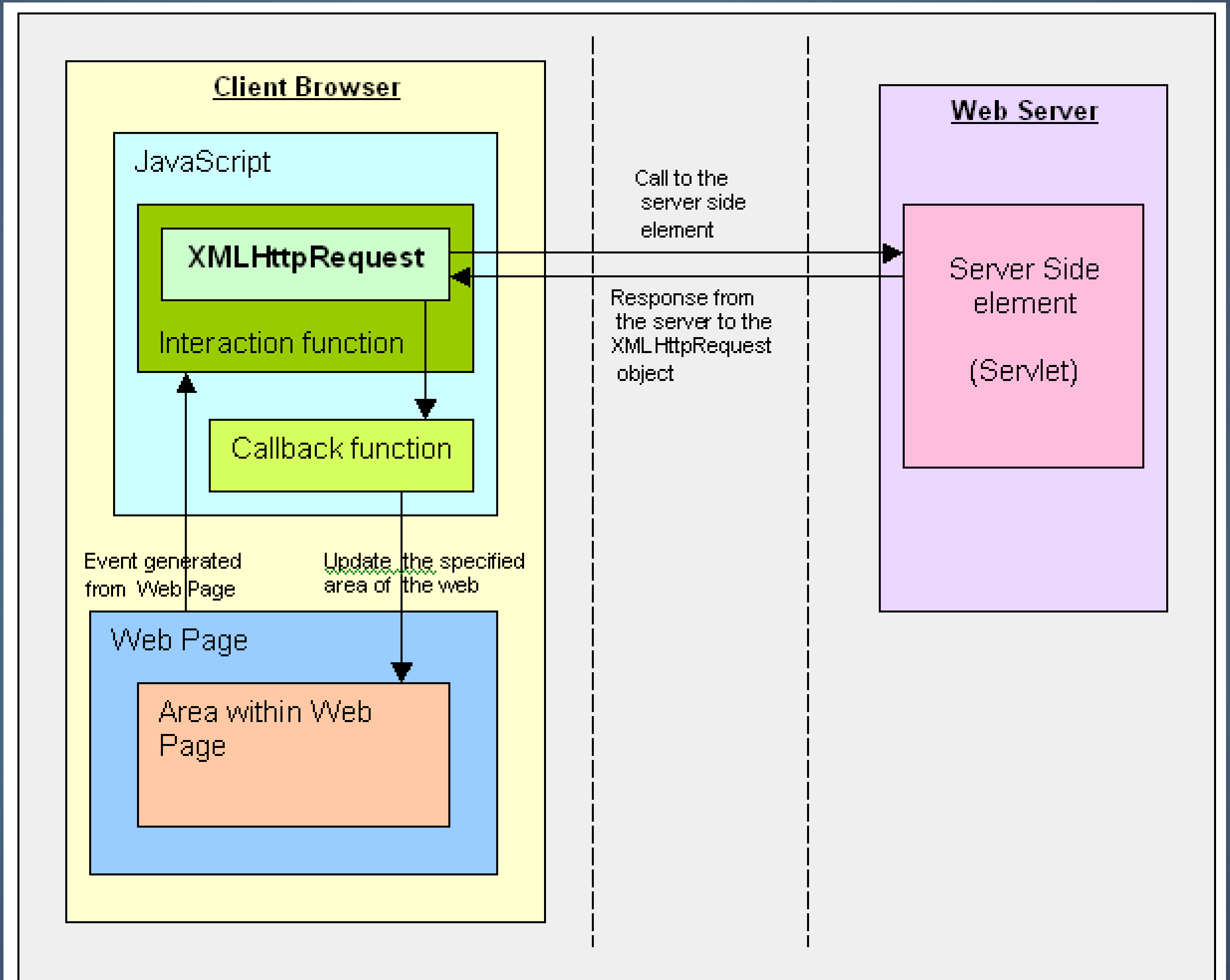
```
<script>
$(document).ready(function(){
  $("p").hover(function(){
    $("p").css("background-color","yellow");
    },function(){
    $("p").css("background-color","pink");
  });
});
</script>
</head>
<body>

<p>Hover the mouse pointer over this paragraph.</p>

</body>
```

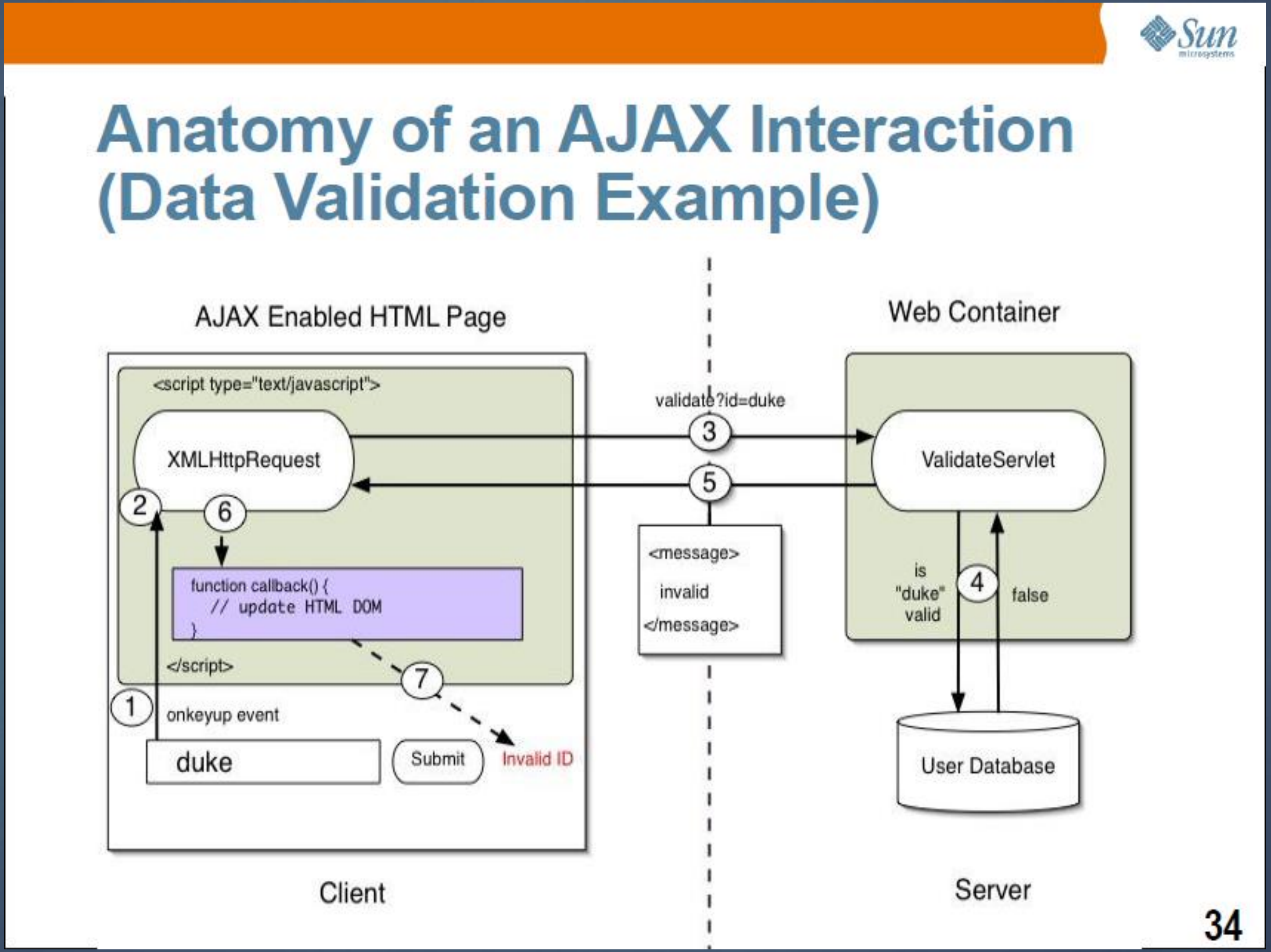
Jquery –ajax interaction

Ajax Interaction


Client Browser
JavaScript
XMLHttpRequest
Interaction function
Callback function
Event generated from Web Page
Update the specified area of the web
Web Page
Area within Web Page
Call to the server side element
Response from the server to the XMLHttpRequest object
Web Server
Server Side element
(Servlet)

AJAX interaction with XMLHttpRequest object

Ajax Interaction



Anatomy of an AJAX Interaction (Data Validation Example)

34

XMLHttpRequest

- JavaScript object
- Adopted by modern browsers
  - Mozilla™, Firefox, Safari, and Opera
- Communicates with a server via standard HTTP GET/POST
- XMLHttpRequest object works in the background for performing asynchronous communication with the backend server
  - Does not interrupt user operation

Server-Side AJAX Request Processing

- Server programming model remains the same
  - It receives standard HTTP GETs/POSTs
  - Can use Servlet, JSP, JSF, ...
- With minor constraints
  - More frequent and finer-grained requests from client
  - Response content type can be
    - **text/xml**
    - **text/plain**
    - **text/json**
    - **text/javascript**

Steps of AJAX Operation

1. A client event occurs
2. An XMLHttpRequest object is created
3. The XMLHttpRequest object is configured
4. The XMLHttpRequest object makes an async. request
5. The Servlet/JSP returns an XML document containing the result
6. The XMLHttpRequest object calls the callback() function and processes the result
7. The HTML DOM is updated

Jquery Ajax

- jQuery.ajax() method. – Used to make Ajax Calls
- It has common options of a request type, url, dataType,error, and success:
- The first option that needs to be addressed when starting an Ajax request is the type of HTTP request
- var options = {
- type: 'GET',
- url: 'hello-ajax.jsp',
- dataType: 'html'
- };

  $.ajax(options);

JQuery Ajax

```
var options = {
        type: 'GET',
        url: 'hello-ajax.html',
        dataType: 'html',
        error: function(xhr, textStatus, errorThrown) {
        alert('An error occurred! ' + errorThrown);
        },
        success: function(data, textStatus) {
        $('body').append( data );
        }
        };

$.ajax( options );
```

Making an Ajax Call

```
$(document).ready(function(){
$("button").click(function(){
  $.ajax({url:"First.Jsp", success:function(result){
    $("div").html(result);
  }});
});});
</script>
</head>
<body>

<div><h2>Let AJAX change this text</h2></div>
<button>Change Content</button>
</body>
</html>
```

JQuery AJAX Events:

- The methods used during the life cycle of AJAX call progress.
- Based on different events/stages following methods are available:
- **ajaxComplete(callback)**
  Attach a function to be executed whenever an AJAX request completes.
- **ajaxStart(callback)**
  - Attach a function to be executed whenever an AJAX request begins and there is none already active
- **.ajaxError(callback)**
  - Attach a function to be executed whenever an AJAX request fails.
- **ajaxSend(callback)**
  - Attach a function to be executed before an AJAX request is sent.
- **ajaxStop(callback)**
  - Attach a function to be executed whenever all AJAX requests have ended
- ajaxSuccess(callback)
  - Attach a function to be executed whenever an AJAX request completes successfully.

Ajax Events

- Within the ready handler on the page, we also establish a number of event handlers as follows:

- $('body').bind(
- 'ajaxStart ajaxStop ajaxSend ajaxSuccess ajaxError ajaxComplete',
- function(event){ say(event.type); }
-);

JQuery AJAX Methods:

- jQuery.ajaxSetup(options) :
  - Load a remote page using an HTTP request. Setup global settings for AJAX requests.
- jQuery.get(url, [data], [callback], [type])
  - Load a remote page using an HTTP GET request.
- jQuery.getJSON(url, [data], [callback]):
  - Load JSON data using an HTTP GET request.
- jQuery.getScript(url, [callback])
  - Loads and executes a JavaScript file using an HTTP GET request
- .jQuery.post(url, [data], [callback], [type])
  - Load a remote page using an HTTP POST request.
- load(url, [data], [callback])
  Load HTML from a remote file and inject it into the DOM

JSON: JavaScript Object Notation.

- Lightweight text-data interchange format
- "self-describing" and easy to understand
- Uses JavaScript syntax for describing data objects
- JSON parsers and JSON libraries exists for many different programming languages
- JavaScript program can use the eval() to get native JS objects.

```
{
    "employees": [
      { "firstName":"John" , "lastName":"Doe" },
      { "firstName":"Anna" , "lastName":"Smith" },
      { "firstName":"Peter" , "lastName":"Jones" }
    ]
}
```

The employees object is an array of 3 employee records (objects).

JSON Syntax Rules

- JSON syntax is a subset of the JavaScript object notation syntax:
  - Data is in name/value pairs
  - Data is separated by commas
  - Curly braces hold objects
  - Square brackets hold arrays

- **JSON Name/Value Pairs**
- JSON data is written as name/value pairs.
- A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value:

- "firstName" : "John“ equivalent to firstName = "John"

Working With Json Data

- Servers return JSON string against a request.
- **getJSON()** parses the returned JSON string and makes the resulting string available to the callback function
- **[selector].**getJSON(URL, [data], [callback]);
- **data:**
  - An object whose properties serve as the name/value pairs used to construct a query string to be appended to the URL, or a preformatted and encoded query string.
- **Callback:**
  - The data value resulting from digesting the response body as a JSON string is passed as the first parameter to this callback, and the status as the second.

Generating JSON Data

```
User user = new User(101,”Ramesh);

ObjectMapper mapper = new ObjectMapper();
 try {
 mapper.writeValue(new File(“user.json"), user);

System.out.println(mapper.writeValueAsString(user));

} catch (Exception e) {

e.printStackTrace();

}
```

Working With Json Data

```
$(document).ready(function() {

  $("#driver").click(function(event){ $.getJSON('user.json', function(jd) {
     $('#id1').html('<p> Name: ' + jd.name + '</p>');
     $('#id1').append('<p>Age : ' + jd.age+ '</p>');
     });
  });
  });

 <div id="id1" > </div>

<input type="button" id="driver" value="Load Data" />
```