INF 272
jQuery

Write Less, Do More

Session 05

INFORMATIKA • DEPARTEMENT • INFORMATICS • DEPARTMENT
Universiteit van Pretoria
University of Pretoria

The contents of this lecture are as follows:

Overview

- **jQuery** is a JavaScript Framework designed to simplify tasks regarding the HTML DOM, Event Handling and AJAX/AJAJ (which we will discuss in much greater detail later).
- Always start your jQuery with: $(document).ready(function(){});
- jQuery usages follow a general trend:
  - $(selector).operation()
  - $: is the jQuery keyword used to indicate the use of the framework
  - Selector: jQuery makes use of the CSS selector syntax for selecting objects
  - Operation: the different operations that can be performed on a specific element
- Example: $(“#myDiv”).addClass()

Syntax: Events

- With jQuery you can attach event listeners to elements with ease with the following syntax:
  - $(“#myDiv”).on(“click”, function() {});

    OR
  - $(“#myDiv”).click(function() {});
- Use the same event names that you would use in Vanilla JS

DOM Manipulation

- Getting and Setting Content of Elements:

GET	SET
var pText = $(“p”).text()	$(“p”).text(“lorem ipsum”)
var dHtml = $(“div”). html()	$(“p”). html(“<p>lorem</p>”)
var inputVal = $(“input”). val()	$(“input”). val(“John Doe”)

- You can also get and set the value of any attribute as follows:
  - Set: $(“div”).attr(“id”, “myDiv”)
  - Get: $(“div”).attr(“id”)

DOM Manipulation

- You can add HTML elements dynamically as follows:
  - First create the element by defining a self closing tag: ***var newDiv = $(“<div/>”)***
  - Give it some attributes: ***var newDiv = $(“<div/>”, { id: “myDiv”, class: “container” })***
  - Then perform the actual insertion:
    - $(“body”).append(newDiv)
    - $(“body”).prepend(newDiv)
    - $(“h1”).after(newDiv)
    - $(“h1”).before(newDiv)
    - There are many other functions like insertAfter, insertBefore, appendTo, prependTo, etc.
- You can remove elements dynamically as follows:
  - $(“#myDiv”).remove()
  - $(“body”).empty()

CSS Manipulation

- The most efficient way to work with jQuery and CSS is to dynamically add and remove (toggle) classes to apply/remove styling from an element:
  - $(“#myDiv”).addClass(“active”)
  - $(“#myDiv”).removeClass(“active”)
  - $(“#myDiv”).toggleClass(“active”)
- Otherwise, you can also specify specific styles to be added to the element like you would in vanilla JS:
  - $(“#myDiv”).css(“display”, “none”)
  - $(“#myDiv”).css({ “display”: “block”, “color”: “red” })

Animations

- jQuery makes hiding and showing with (and without) animates very easy:
  - $(“#myDiv”).hide() and $(“#myDiv”).show() (this does not animate the hide/show)
  - $(“#myDiv”).fadeIn(500) and $(“#myDiv”).fadeOut(“slow”)
  - $(“#myDiv”).slideUp(500) and $(“#myDiv”).slideDown(“slow”)
- Or you can define your own specific animations with the animate() function:
  - $(“#myDiv”).animate({ max-height: “300px” }, 500, “swing”, function() {});

    OR
  - $(“#myDiv”).animate({ max-height: “300px” }, { duration: 500, easing: “swing”, queue: false, complete: function() {} });

DOM Traversal

- jQuery provides a number of functions to help with traversing the DOM relative to a specific element.
- Moving Up the Tree:
  - To get an element’s parent use: *$(“#myDiv”).parent()*
  - To get a list of ancestors all the way to the <html> tag, use: *$(“#myDiv”).parents()*
- Moving Down the Tree:
  - To get a list of an element’s immediate children use: *$(“#myDiv”).children()*
  - To find a specific element in an element’s list of descendants, use: *$(“#myDiv”).find(“p”)*

DOM Traversal

- Moving Side to Side in the Tree:
  - To get a list of a specific element’s sibling: *$(“#myDiv”).siblings()*
  - To get the element immediately before the current element: *$(“#myDiv”).prev()*
  - To get the element immediately after the current element: *$(“#myDiv”).next()*

Video

- This week's video is available at

  https://www.youtube.com/watch?v=BWXggB-T1jQ

- However, the video has been loaded onto ClickUP as well. Please note that additional notes are posted if and when required.

Conclusion

- By using jQuery functions, we are able to essentially Do More, while Working Less. jQuery is mainly used for DOM Manipulation and makes those specific operations a lot easier, smoother, and more intuitive than vanilla JS.
- Any questions? ^^