jQuery Fundamentals

Eng. Niveen Nasr El-Den
iTi

Day 2

The document object in the BOM
is the top level of
the DOM hierarchy.

DOM Relationships

Scripting HTML

HTML DOM

- The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

- It is a hierarchy of data types for HTML documents, links, forms, comments, and everything else that can be represented in HTML code.

- The general data type for objects in the DOM are *Nodes*. They have *attributes*, and some nodes can contain other nodes.

- There are several node types, which represent more specific data types for HTML elements.

DOM

- It allows code running in a browser to access and interact with every node in the document.
- Nodes can be created, moved and changed.
- Node types are represented by numeric constants.
- Event listeners can be added to nodes and triggered on occurrence of a given event.

DOM
is an API that represents and interacts with any **HTML** or **XML** document.

The DOM
is a document model loaded in the browser and representing the document as a node tree, where each node represents part of the document

The DOM
is an application programming interface “API“

⇩

a set of functions or methods used to access some functionality

The DOM

Defines the logical structure of document and the way a document is accessed and manipulated

DOM

connects web pages to scripts or programming lan guages by representing the structure of a document.

HTML DOM

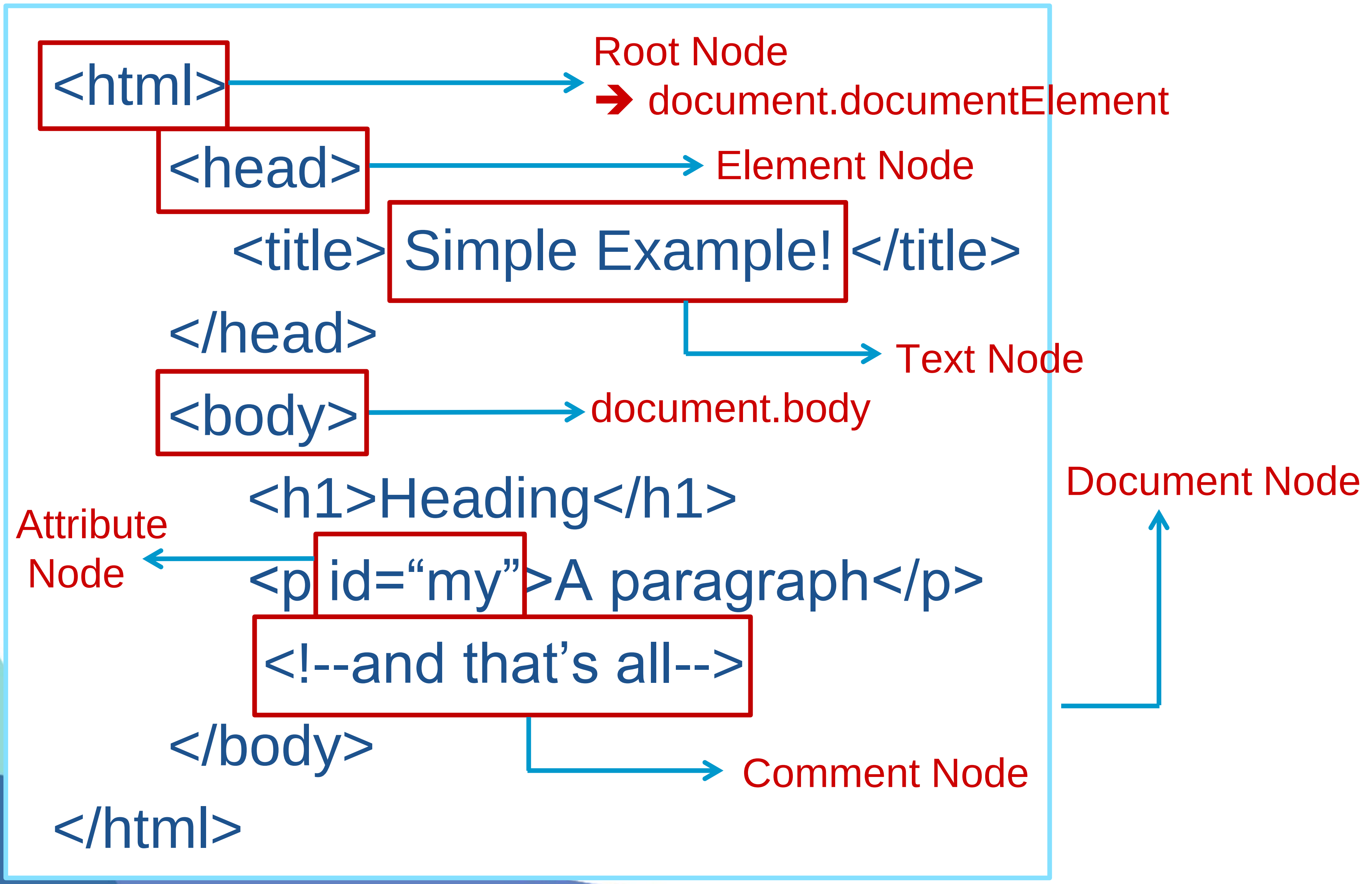
- According to the DOM, everything in an HTML document is a node.

- The DOM says:
  - ▻ The entire document is a document node
  - ▻ Every HTML element is an element node
  - ▻ The text in the HTML elements are text nodes
  - ▻ Every HTML attribute is an attribute node
  - ▻ Comments are comment nodes

- JavaScript is powerful DOM Manipulation

An **element** is a specific type of **node**, one that can be directly specified in the HTML with an **HTML tag**

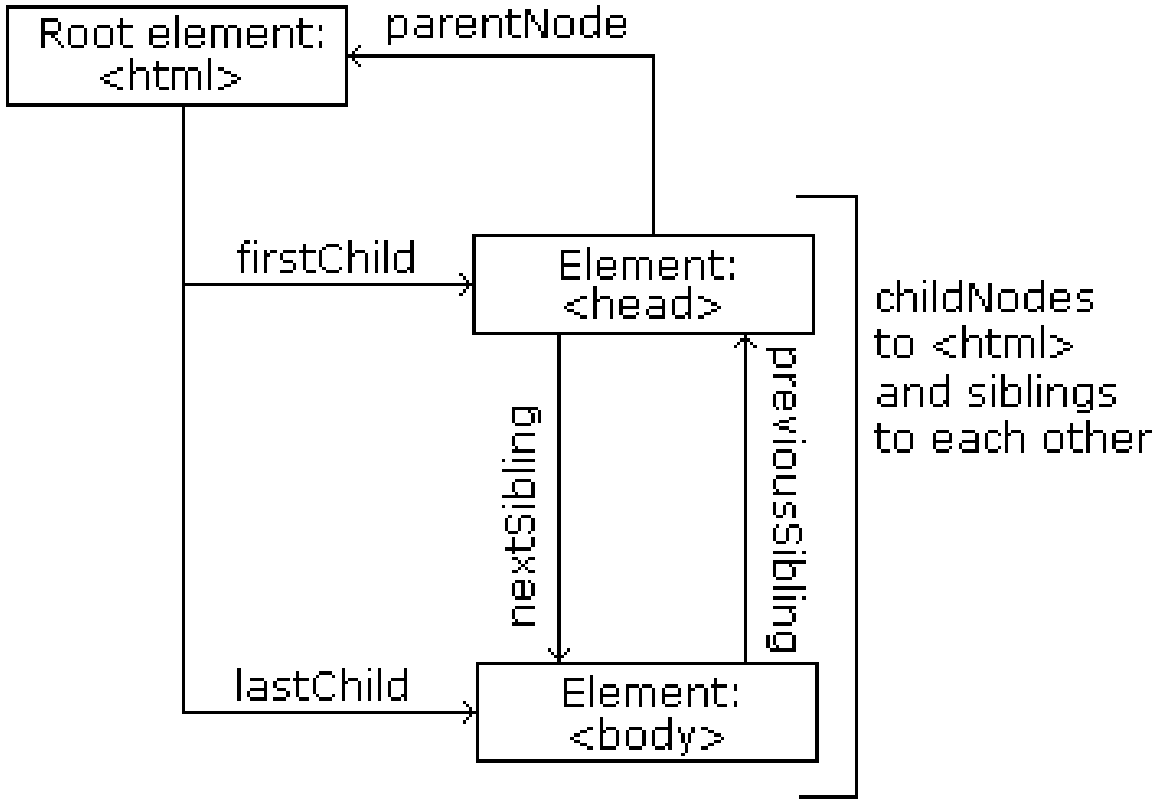
Simple Example!



```
<html>
    <head>
        <title>Simple Example!</title>
    </head>
    <body>
        <h1>Heading</h1>
        <p id="my">A paragraph</p>
        <!--and that's all-->
    </body>
</html>
```

Node Tree

- The HTML DOM views HTML document as a node-tree.
- All the nodes in the tree have relationships to each other.
    - Parent
        - parentNode
    - Children
        - firstChild
        - lastChild
    - Sibling
        - nextSibling
        - previousSibling

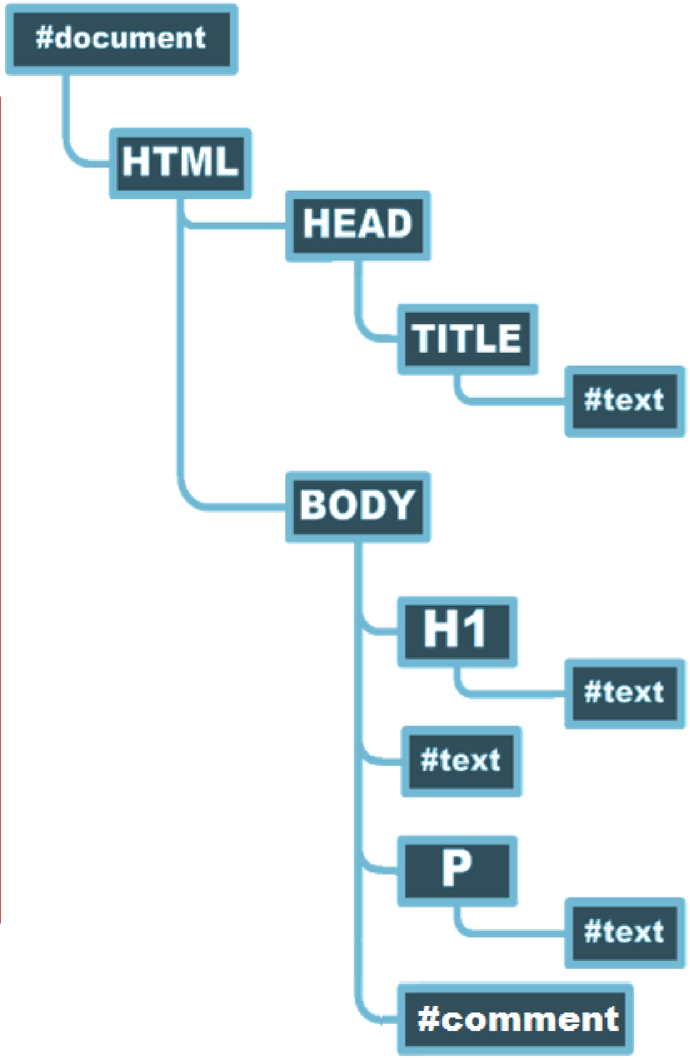


Nodes Relationships

- The terms **parent**, **child**, and **sibling** are used to describe the relationships.
  - Parent nodes have children.
  - Children on the same level are called siblings (brothers or sisters).
- Attribute nodes are not child nodes of the element they belong to, and have no parent or sibling nodes
- In a node tree, the top node is called the root
- Every node, except the root, has exactly one parent node
- A node can have any number of children
- A leaf is a node with no children
- Siblings are nodes with the same parent

Simple Example!

```
<html>
        <head>
            <title>Simple Example!</title>
        </head>
        <body>
            <h1>Greeting</h1>
             Welcome All
            <p>A paragraph</p>
            <!-- and that’s all-->
        </body>
 </html>
```



Node Properties

- All nodes have three main properties

Property	Description
nodeName	Returns HTML Tag name in uppercase display
tagname	
nodeType	returns a numeric constant to determine node type. There are 12 node types.
nodeValue	returns null for all node types except for text and comment nodes.

Using nodeName
If node is text it returns #text
For comment it returns #comment
For document it returns #document

Value	Description
1	**Element Node**
2	**Attribute Node**
3	**Text Node**
8	**Comment Node**
9	**Document Node**

To get the Root Element:
document.documentElement.

Node Collections

- Node Collections have One Property
  - ▻ **length : gives the length of the Collection.**
    - **e.g. childNodes.length:** returns number of elements inside the collection
- We can check if there is child collection using
  - ▻ **hasChildNodes():** Tells if a node has any children
- We can check if there is attribute collection using
  - ▻ **hasAttributes():** Tells if a node has any attributes

Collection	Description	Accessing
childNodes	Collection of element's children	childNodes[ ] childNodes.item()
attributes	Returns collection of the attributes of an element	attributes[ ] attributes.item()

Dealing With Nodes

- **Dealing with nodes fall into four main categories:**
  - ▻ **Accessing Node**
  - ▻ **Modifying Node’s content**
  - ▻ **Adding New Node**
  - ▻ **Remove Node from tree**

Accessing DOM Nodes

- You can access a node in 5 main ways:
  - ▻ [window.]document.getElementById("id")
  - ▻ [window.]document.getElementsByName("name")
  - ▻ [window.]document.getElementsByTagName("tagname")
  - ▻ By navigating the node tree, using the node relationships
  - ▻ New HTML5 Selectors.

Example!

New HTML5 Selectors

- In HTML5 we can select elements by ClassName

```
var elements = document.getElementsByClassName('entry');
```

- Moreover there’s now possibility to fetch elements that match provided CSS syntax

```
var elements =  document.querySelectorAll(".someClasses)");
```

```
var elements = document.querySelectorAll("div,p");
```

```
var elements = document.querySelector("#someID");
```

```
var first_td =   document.querySelector("span");
```

Accessing DOM Nodes

Navigating the node tree, using the node relationships

firstChild	Move direct to first child node
lastChild	Move direct to last child node
parentNode	To access child's parent node
nextSibling	Navigate down the tree one node step
previousSibling	Navigate up the tree one node step
Using children collection → childNodes[ ]	

Example!

Accessing DOM Elements

Navigating the elements nodes, using the relationships

firstElementChild	Move direct to first Element child
lastElementChild	Move direct to last Element child
parentElement	To access child’s Element parent
nextElementSibling	Navigate down the tree to next Element
previousElementSibling	Navigate up the tree to previous Element

Example!

Modifying Node’s Content

- Changing the Text Node by using

innerHTML	Sets or returns the HTML contents (+text) of an element
textContent	Equivalent to innerText.
nodeValue → with text and comment nodes only	
setAttribute()	Modify/Adds a new attribute to an element
just using attributes as object properties	

Example!

Node’s Class Attribute

- The global class attribute is get and set via className property

- The classList property returns a collection of the class attributes of the caller element, it has the following methods
    - add(“classNm”)
    - remove(“classNm”)
    - toggle(“classNm”)
    - replace(“oldClassNm”,”newClassNm”)

Manipulating Styles

- Modifying style properties of any HTML element is accessed using the style object
- For inline style
    - ▻ Node.style[.prop_name]
    - ▻ Node.style.cssText
- To read internal or external styling in general
    - ▻ document.styleSheets
    - ▻ document.styleSheets[i].cssRules
    - ▻ document.styleSheets[i].cssRules[idx].selectorText
    - ▻ document.styleSheets[i].cssRules[idx].cssText
- To read none inline styling applied for specific element
    - ▻ getComputedStyle(elem).prop_nm
    - ▻ getComputedStyle(elem). getPropertyValue(prop_nm)

Creating & Adding Nodes

Method	Description
createElement()	To create new tag element
createTextNode()	To create new text element
createAttribute()	To creates an attribute element
createComment()	To creates an comment element

Creating & Adding Nodes

Method	Description
cloneNode(true|false)	Creating new node a copy of existing node. It takes a Boolean value true: Deep copy with all its children or false: Shallow copy only the node
b.appendChild(a)	To add new created node "a" to DOM Tree at the end of the selected element "b".
b.append(a)	Experimental function to o add new created node "a" to DOM Tree at the end of the selected element "b".
b.prepend(a)	Experimental function to o add new created node "a" to DOM Tree at the top of the selected element "b".

Example!

Creating & Adding Nodes

https://developer.mozilla.org/en-US/docs/Web/API/Element

Method	Description
insertBefore(a,b)	Similar to appendChild() with extra parameter, specifying before which element to insert the new node. a: the node to be inserted b: where a should be inserted before **document.body.insertBefore(a,b)**
e.insertAdjacentElement(pos,elem)	○ e: represents the target element ○ elem: represents the element to be added ○ pos: represents the position relative to the targetElem • 'beforebegin': Before the targetElement itself. • 'afterbegin': Just inside the targetElement, before its first child. • 'beforeend': Just inside the targetElement, after its last child. • 'afterend': After the targetElement itself.

Example!

Removing DOM Nodes

Method	Description
removeChild()	To remove node from DOM tree
parent.replaceChild(n,o)	To remove node from DOM tree and put another one in its place n: new child o: old child
removeAttribute()	Removes a specified attribute from an element

- To quick replace a node set its outerHTML property

  elem.outerHTML="<div>something</div";

- A quick way to wipe out all the content of a subtree is to set the innerHTML to a blank string. This will remove all of the children of <body>

  document.body.innerHTML="";

Example!

Summary

- Access nodes:
  - ▻ Using parent/child relationship properties parentNode, childNodes, firstChild, lastChild, nextSibling, previousSibling
  - ▻ Using getElementsById(), getElementsByTagName(), getElementsByName()
- Modify nodes:
  - ▻ Using innerHTML or innerText/textContent
  - ▻ Using nodeValue or setAttribute() or just using attributes as object properties
- Remove nodes with
  - ▻ removeChild() or replaceChild()
- And add new ones with
  - ▻ appendChild(), cloneNode(), insertBefore()

Modeling HTML or XML documents as objects are not part of the core JavaScript language.

The DOM
Defines the logical structure of document and the way a document is accessed and manipulated

jQuery Fundamentals

Eng. Niveen Nasr El-Den
iTi

jQuery Selectors & Filters

:contains(txt) Selector

- Used to select an element based on string it contains
- it’s case sensitive.

- Example:
  $('span:contains("A")');

:not() Selectors

- jQuery gives us the :not filter, which you can use in the following way:
  `$('div:not(#content)');`
  → Select all DIV elements except #content

- The selector can be as complex as you like:
  `$('a:not(div.important a, a.nav)');`
  →Selects anchors that do not reside within 'div.important' or have the class 'nav'

siblings( ) Method

- To select and filter all siblings of an element occurring either next or previous.

- You can pass an optional selector expression to filter the selection

- Example:
  $('h1').siblings('h2,h3,p');
  →Selects all H2, H3, and P elements that are siblings of H1 elements..

next( ) Method

- To make the same functionality of the sibling combinator (+) without using it, you can use the next( ) method.
- The next( ) method can make a nice alternative to the selector syntax, especially in a programmatic setting when you're dealing with jQuery objects as variables.
- Example

```
var topHeaders = jQuery('h1');
topHeaders.next('h2').css(‘color',‘red');
```

nextAll( ) Method

- Sometimes you'll want to target siblings dependent on their position relative to other elements
  - for example
    To select all list items beyond the second (after li.selected), you could use the following method
    $('li.selected').nextAll('li');

- The nextAll() method, just like siblings(), accepts a selector expression to filter the selection before it's returned. If you don't pass a selector, then nextAll() will return all siblings of the subject element that exist after the subject element, although not before it.

Selecting Specific Siblings

- If you’re looking to select the adjacent sibling of a particular element.
- Then you can use the adjacent sibling combinator (**+**).
- Similar to the child (>) combinator, the sibling combinator expects a selector expression on each side.
- The righthand expression is the subject of the selector, and the lefthand expression is the sibling you want to match.

children( ) method

- Selecting children in a more programmatic environment should be done using jQuery’s children() method, to which you can pass a selector to filter the returned elements.

- This would select all direct children of the #content element:

```
$('#content').children();
```

- The preceding code is essentially the same as $('#content > *')

jQuery Attributes Methods

Attributes & Properties

- **The basic components we can manipulate when it comes to DOM elements are the *properties* and *attributes* assigned to those elements.**

- **Most of these attributes are available through JavaScript as DOM node properties.**

- **Some of the more common properties are:**
  - ▻ **className**
  - ▻ **tagName**
  - ▻ **id**
  - ▻ **href**
  - ▻ **title**
  - ▻ **src**

Attributes & Properties

- **Getting/setting & removing attributes**
  - ▻ **.attr(attr_nm [,val])**
  - ▻ **.removeattr(attr_nm)**

- **Getting/setting elements content & values**
  - ▻ **.text([“str”])**
  - ▻ **.val([“val”])**
  - ▻ **.html([“str”])**

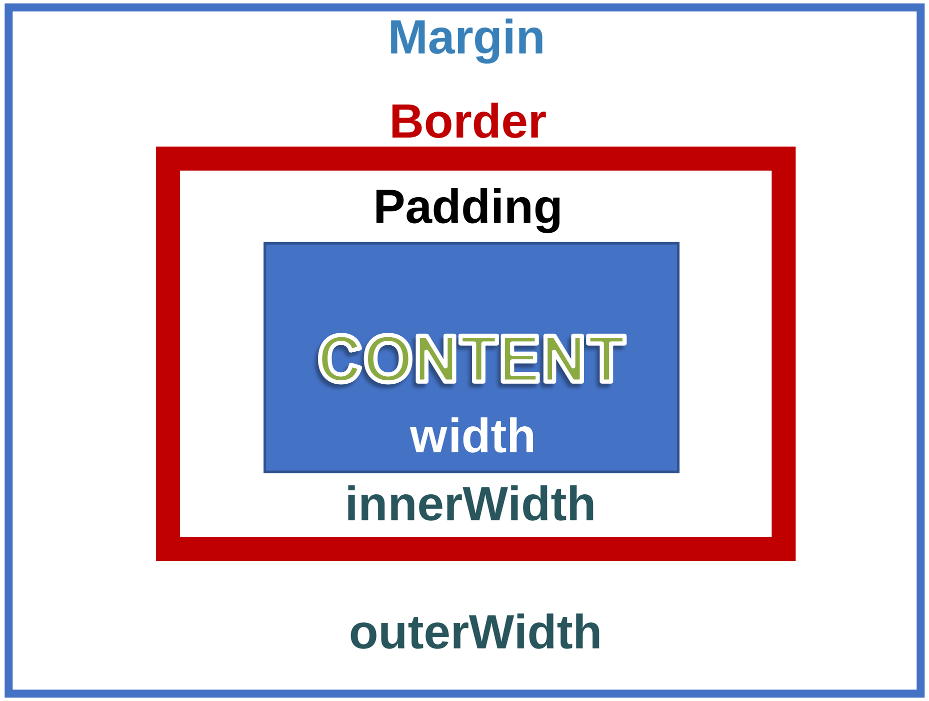
Attributes & Properties

- **Styling methods**
  - ▻ **.addClass()**
  - ▻ **.removeClass()**
  - ▻ **.hasClass()**
  - ▻ **.toggleClass()**
  - ▻ **.css()**

Other CSS functions

- .position()
- .scrollTop([val])
- .scrollLeft([val])

- .height([val])
- .width([val])
- .innerHeight()
- .innerWidth()
- .outerHeight(margin)
- .outerWidth(margin)



Other Functions

- Array Functions
  - ▻ makeArray(coll)
  - ▻ inArray
  - ▻ unique→uniqueSort
  - ▻ merge
  - ▻ map
  - ▻ grep

- TestingFunctions
  - ▻ isArray
  - ▻ isFunction
  - ▻ isNumeric
  - ▻ type
  - ▻ etc..

each() Method

- Iterate over a jQuery object, array, javascript object properties; executing a function for each matched element.
- Syntax:

```
$.each(coll,function(idx,elem){});
```

→where: coll is either an array or javascript object properties..

```
$(selector).each(function(idx,elem){});
```

```
<ul>
    <li>trip</li>
    <li>vacation</li>
    <li>abroad</li>
</ul>
```

```
$(document).ready(function(){
  $('li').each(function(index) {
    alert(index + ': ' + $(this).text());
  });
});
```

DOM Manipulation

DOM Manipulation

- Create Elements
- Insert Elements
    - Element Content
- Remove Elements

Clone Elements

- Clone elements via **.clone()**
  - ▻ Clone selected element
  - ▻ .clone(**true**) to copy events and data, too

- Example:
  $('#myid')**.clone()**

Insert Elements

- add and insert elements
  - inside others with **.append()**, **.appendTo()**, **.prepend()** and **.prependTo()**
  - before and after others with **.after()**, **.insertAfter()**, **.before()** and **.insertBefore()**
  - around others with **.wrap()**, **.wrapAll()** and **.wrapInner()**
  - in place of others with **.replaceWith()** and **.replaceAll()**

.append() & .prepend() Methods

- .append()

  e.g. $('#TestList').append('<li>Appended item</li>')
  The new item will be inserted as the last term

- .prepend()

  e.g. $('#TestList').prepend('<li>Appended item</li>')
  The new item will be inserted as the first item of the list

.append() & .prepend() Methods

- Both append() and prepend() methods take an infinite amount of new elements as parameters.

```
var item1 = $("<li></li>").text("Item 1");
var item2 = "<li>Item 2</li>“;
var item3 = document.createElement("li");

item3.innerHTML = "Item 3“;

$("#TestList").append(item1, item2, item3);
```

.before() & .after() Methods

- Used to insert things before or after one or several elements.
- Both after() and before() allows you to use HTML strings, DOM elements and jQuery objects as parameters and an infinite amount of them as well.

- Example:
  `$('input.test1').before('<i>Before</i>')`
  An italic tag will be inserted before each input element on the page using the "test1" class.

  `$('input.test1').after('<b>After</b>')`
  A bold tag will be inserted after each input element on the page using the "test1" class.

Methods variations

- There are variations of append() and prepend() methods, called appendTo() and prependTo().
  - ▻ `$('#TestList').append('<li>Appended item</li>')`
  - ▻ `$('<li>Appended item</li>').appendTo('#TestList')`

- There are variations of before() and after() methods, called insertBefore() and insertAfter().
  - ▻ `$('#test').before('<i>Before</i>')`
  - ▻ `$('<i>Before</i>').insertBefore('#test')`

.remove() & .empty() Methods

- . empty() method
  - ▻ will only delete all child elements of the selected element(s).
    Example: `$("#test").empty();`
- . remove() method
  - ▻ will delete the selected element(s)
    Example: `$("#test").remove();`
  - ▻ It comes with one optional parameter, allowing to filter the elements to be removed, using any of the jQuery selector syntaxes.
    Example: `$("#test").remove(".bold");`

jQuery Events

Shorthand Syntax

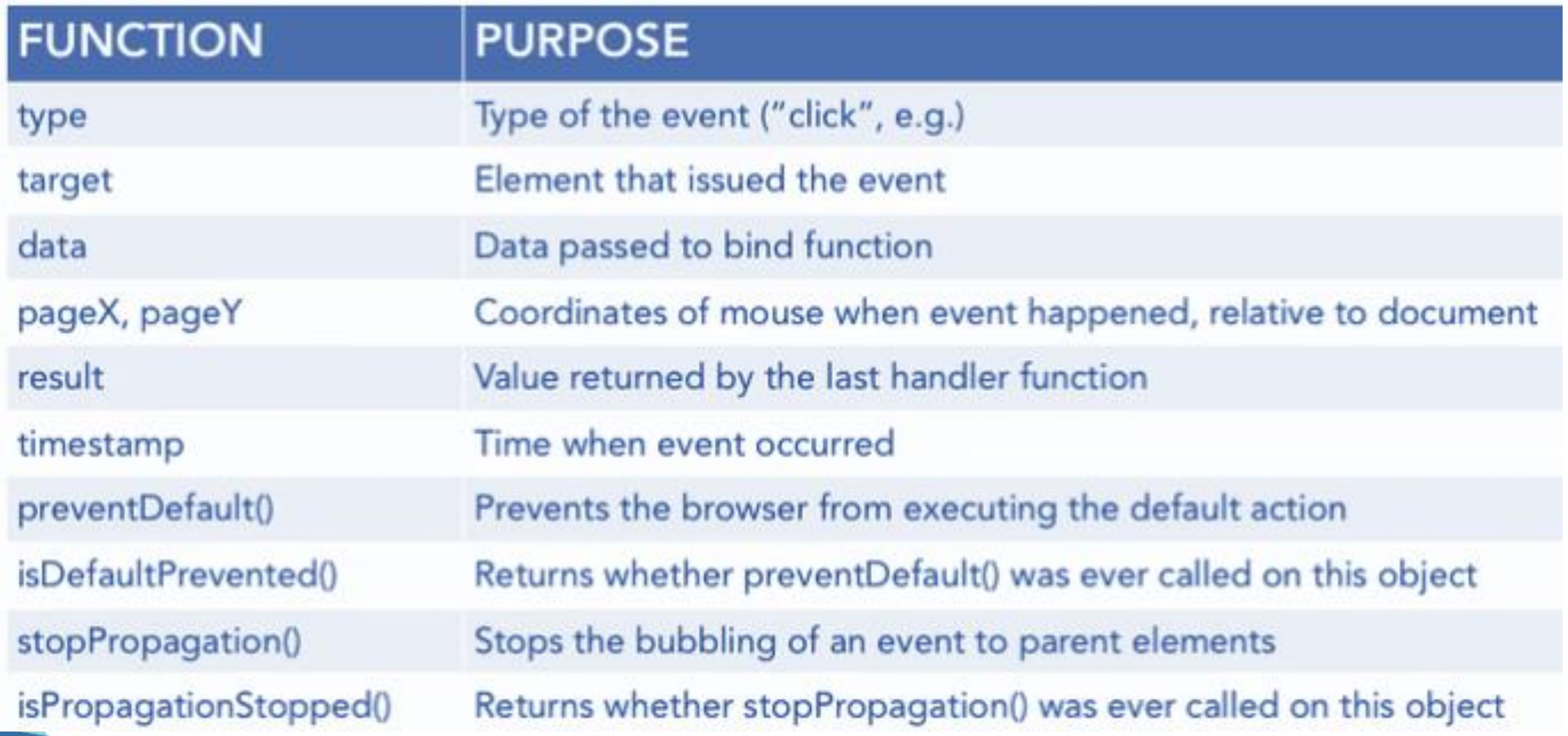
- .submit()
- .change()
- .focus()
- .blur()
- .focusout()

- .keydown()
- .keypress()
- .keyup()
- .scroll()
- .resize()
- .error()

- .mouseover()
- .mouseout()
- .mouseenter()
- .mouseleave()
- .mousemove()
- .dblclick()

jQuery Event Object

- jQuery defines an event object with the most important properties needed cross-browsers



FUNCTION	PURPOSE
type	Type of the event ("click", e.g.)
target	Element that issued the event
data	Data passed to bind function
pageX, pageY	Coordinates of mouse when event happened, relative to document
result	Value returned by the last handler function
timestamp	Time when event occurred
preventDefault()	Prevents the browser from executing the default action
isDefaultPrevented()	Returns whether preventDefault() was ever called on this object
stopPropagation()	Stops the bubbling of an event to parent elements
isPropagationStopped()	Returns whether stopPropagation() was ever called on this object

Event Binding and Delegation

- .on()
  - ▻ new in jQuery 1.7
  - ▻ the future of jQuery event handling
  - ▻ one event handler method to rule them all
  - ▻ use instead of .bind() and .live() and .delegate()

- .off()
  - ▻ unbinds event handlers bound by .on()

Event Binding ".on()" & Removing ".off()"

```
$('button')
.on('click', clickListener)
.on('click', otherListener);

// remove all click listeners
$('button').off('click');

// remove only the specified listener
$('button').off('click', clickListener);
```

Event Delegation

```
$('#wrapper').on('click', 'button', function(event) {
  // button got clicked
});

// remove click listeners
$('#wrapper').off('click', 'button');
```

Miscellaneous Event Methods

- .tigger()
  - ▻ Trigger events programmatically without waiting to user

```
$('button').trigger('click');
```

- .one()
  - ▻ Similar to bind but the event handler works once

```
$('button').one('click‘, function(event) {
  // button got clicked
});
```

jQuery Animations & Effects

jQuery & jQuery UI Effects

- jQuery provides a simple interface for doing various kind of amazing effects.

- jQuery methods allow us to quickly apply commonly used effects, which fall into 2 categories:
    - ▻ jQuery Effect Methods
    - ▻ UI Library Based Effects

jQuery Effects

- jQuery supports us with simple basic Effect methods which can be used in:
    - ▻ Showing and Hiding elements
    - ▻ Toggling the elements
    - ▻ Fading
    - ▻ Sliding
    - ▻ Custom Animation

jQuery UI

- jQuery UI provides a comprehensive set of:
  - ***Effects***
    - Animated transitions and easing for rich interactions.
  - ***Interaction plugins***
    - Complex behaviors like drag and drop, resizing, selection and sorting.
  - ***UI Widgets***
    - Full-featured UI controls, each has a range of options and is fully themeable

jQuery UI Effects

- jQuery supports us with simple basic Effect methods which can be used in:
  - Showing and Hiding elements
  - Fading
  - Sliding
  - etc..

- We can create animated transitions using jQuery UI with these set of pre-built effects
  - **Movement:**
    - Bounce, Scale, Shake, Size
  - **Feedback:**
    - Highlight, Pulsate, Transfer
  - **Show/Hide:**
    - Blind, Clip, Drop, Explode, Fold, Puff, Scale, Slide

jQuery UI Effects

Blind	**Blinds the element away or shows it by blinding it in.**
Bounce	**Bounces the element vertically or horizontally n-times.**
Clip	**Clips the element on or off, vertically or horizontally.**
Drop	**Drops the element away or shows it by dropping it in.**
Explode	**Explodes the element into multiple pieces.**
Fold	**Folds the element like a piece of paper.**
Highlight	**Highlights the background with a defined color.**
Scale	**Shrink or grow an element by a percentage factor.**
Shake	**Shakes the element vertically or horizontally n-times.**
Size	**Resize an element to a specified width and height.**
Slide	**Slides the element out of the viewport.**
Transfer	**Transfers the outline of an element to another.**

jQuery UI Effects “Show/Hide”

[selector].*show* (speed , [callback]);

[selector].*show* (effect, [options], [speed], [callback]);

- ***effect*** → values: 'blind', 'clip', 'drop', 'explode', 'fold', 'puff', 'slide', 'scale', 'size', 'pulsate'.
- ***speed*** → "slow", "normal", or "fast“
- ***callback*** → a function to be executed whenever the animation completed

[selector].*hide* (speed, [callback]);

Using .animate() Method

- **The *animate( )* method performs a custom animation of a set of numeric CSS properties.**
  - **top, left, width, height, opacity, fontSize, borderWidth**

- Properties can be animated
  - by number, percent, etc.
  - relatively ("+=200px", "-=20%", etc.)
  - by keyword: "hide", "show", or "toggle"

Using .animate() Method

```
selector.animate(params, [duration, [easing, [complete]]]);
```

```
selector.animate(params, options);
```

Using .animate() Method

- Example, slowly moving an element 300px to the right

```
$('.toMove').animate({
  left: '+=300px'
}, 800);

// same as
$('.toMove').animate({
  left: '+=300px'
}, {
  duration: 800
});
```

Complete

- Function executed when the animation ends
- Called once for *each* animated element

- Example:

```
$("#divTestArea3").fadeIn(2000, function() {

  $("#divTestArea3").fadeOut(3000);

});
```

Easing

- Changes the velocity at different points in the animation
- A number of standard equations first created by Robert Penner
  - Available with jQuery only
    - linear
    - Swing
  - More easing functions are available within
    - jQuery- UI or
    - stand-alone (plug-in)at http://gsgd.co.uk/sandbox/jquery/easing/

http://easings.net/

Options object allows for fine tuning

- **duration**: A string or number determining how long the animation will run.
- **easing**: A string indicating which easing function to use for the transition. ("linear", "swing". More with plugin.)
- **queue**: A Boolean indicating whether to place the animation in the effects queue. If false, the animation will begin immediately.
- **specialEasing**: A map of one or more of the CSS properties defined by the properties argument and their corresponding easing functions (added 1.4).
- **step**: A function to be called for each step of the animation.
- **progress**: A function to be called after each animation step
- **complete**: A function to call once the animation is complete. (callback function.)
- …

Animation

- Animation can be chained

- By default multiple animations occur:
  - in sequence for the same element(s)
  - simultaneously for different element(s)

- .is(':animated')) identify elements that are currently animated

.queue() & .dequeue()

- .queue()
  - ➢ Introduce function in the middle of animation chain

- .dequeue()
  - ➢ Execute the next function on the queue

Stopping animation

- Stop current animations before finishing.
- It works for all effects related jQuery functions, including sliding, fading and custom animations with the animate() method.
- Two ways to stop current animations before finishing
  - .stop([clearQueue [, gotoEnd]])
  - .finish()
    - Clears the queue of animation after moving the animated object to its final destination

.stop()

- .stop([clearQueue [, gotoEnd]])
    - Stops the current animation, and starts next animation within the chain
    - clearQueue
        - A boolean value that specifies whether the animation queue should be cleared or not
        - true: stops the animation
    - gotoEnd
        - A boolean value that tells jQuery whether you would like for it to just stop where it is, or rush through the animation instead, allowing for it to finish.
        - true: means that the animated object moves to its final destination

Delay

- Delays any further items in the queue from executing for a specified period of time.

- Its value is milliseconds

- Example

```
$('#warning').fadeIn(600).delay(4000).fadeOut();
```

jQuery Interactions

jQuery Interactions

- **Draggable** - Makes items draggable by the mouse
- **Droppable** - Makes drop targets for draggables
- **Sortable** - Makes a list of items mouse sortable
- **Selectable** - Makes a list of items mouse and keyboard selectable
- **Resizable** - Makes an element resizable

jQuery Widgets

jQuery Widgets

- Accordion
- Autocomplete
- Button
- Datepicker
- Dialog
- Progressbar
- Slider
- Tabs
- …

Using Multiple Libraries Problem

- The jQuery library, and virtually all of its plugins are constrained within the *jQuery* namespace.

- We can use multiple libraries all together without conflicting each others.

- For example we can use jQuery and Prototype javascript libraries together.

Using Multiple Libraries Problem

- By default, jQuery uses "$" as a shortcut for "jQuery".
- Many JavaScript libraries use $ as a function or variable name, just as jQuery does.
- In jQuery's case, $ is just an alias for jQuery, so all functionality is available without using $.
- We can override that default by calling *jQuery.noConflict()* at any point after jQuery and the other library have loaded.

Solution (1)

```
<script src="prototype.js"></script>
 <script src="jquery.js"></script>
<script>
        jQuery.noConflict() ; // Use jQuery via jQuery(...)


        jQuery(document).ready(function()
                                    { jQuery("div").hide();
        });
        // Use Prototype with $(...), etc.
         $('someid').hide();
        </script>
```

Solution (2)

```
<script src="prototype.js"></script>
<script src="jquery.js"></script>
<script>
   var $j = jQuery.noConflict(); // Use jQuery via $j(...)


    $j (document).ready(function(){
                         $j(“div").hide(); });


    // Use Prototype with $(...), etc.
   $('someid').hide();


 </script>
```

Assignment