

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Using JavaScript for Client-Side Behavior

Internet Applications, ID1354

Contents

- The Document Object Model, DOM
- The Browser Object Model, BOM
- The jQuery JavaScript Library
- AJAX
- The Knockout JavaScript Framework
- Long Polling
- WebSocket

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Section

- The Document Object Model, DOM
- The Browser Object Model, BOM
- The jQuery JavaScript Library
- AJAX
- The Knockout JavaScript Framework
- Long Polling
- WebSocket

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The Document Object Model, DOM

- ▶ The W3C Document Object Model, DOM, is an API that allows programs to **access and update document content**.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The Document Object Model, DOM

- ▶ The W3C Document Object Model, DOM, is an API that allows programs to **access and update document content**.
- ▶ Defines **objects** representing HTML elements, **methods** to access HTML elements, and **events** generated by HTML elements.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The Document Object Model, DOM

- ▶ The W3C Document Object Model, DOM, is an API that allows programs to **access and update document content**.
- ▶ Defines **objects** representing HTML elements, **methods** to access HTML elements, and **events** generated by HTML elements.
- ▶ The best that can be said about **browser support** is that it varies.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The Document Object Model, DOM

- ▶ The W3C Document Object Model, DOM, is an API that allows programs to **access and update document content**.
- ▶ Defines **objects** representing HTML elements, **methods** to access HTML elements, and **events** generated by HTML elements.
- ▶ The best that can be said about **browser support** is that it varies.
 - ▶ Try the features you want to use in all relevant browsers, check **caniuse.com**, etc

DOM

BOM

jQuery

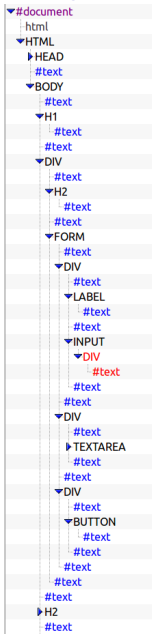
AJAX

Knockout

Long Polling

WebSocket

The DOM Tree



- ▶ The DOM objects are organized in a **tree**.

DOM

BOM

jQuery

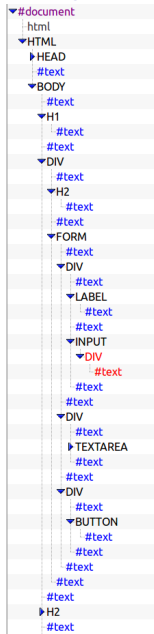
AJAX

Knockout

Long Polling

WebSocket

The DOM Tree



- ▶ The DOM objects are organized in a **tree**.
- ▶ The picture to the left is a part of the DOM tree for the course's **chat program**.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The DOM JavaScript API

- ▶ All HTML elements are represented by **objects**.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The DOM JavaScript API

- ▶ All HTML elements are represented by **objects**.
- ▶ The HTML objects have **properties** you can get or set, to read or update the objects.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The DOM JavaScript API

- ▶ All HTML elements are represented by **objects**.
- ▶ The HTML objects have **properties** you can get or set, to read or update the objects.
- ▶ The HTML objects have **methods**, for example for adding and deleting elements.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The DOM JavaScript API

- ▶ All HTML elements are represented by **objects**.
- ▶ The HTML objects have **properties** you can get or set, to read or update the objects.
- ▶ The HTML objects have **methods**, for example for adding and deleting elements.
- ▶ An example is the JavaScript statement

```
document.getElementById("demo").innerHTML =  
    "Hello World!";
```

that uses the method **getElementById** to find the HTML object for the element with id **demo**, and sets the HTML of that object to **"Hello World!"**, using the **innerHTML** property.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The **document** Object

- ▶ The object **document** can be accessed by any JavaScript code. It **represents the entire web page** and is the entry point to the DOM API.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The **document** Object

- ▶ The object **document** can be accessed by any JavaScript code. It **represents the entire web page** and is the entry point to the DOM API.
- ▶ Sample methods in **document** are
Find HTML elements `getElementById`,
`getElementsByTagName`,
`getElementsByClassName`

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The **document** Object

- ▶ The object **document** can be accessed by any JavaScript code. It **represents the entire web page** and is the entry point to the DOM API.
- ▶ **Sample methods in **document** are**

Find HTML elements `getElementById,`
`getElementsByTagName,`
`getElementsByClassName`

Properties of HTML elements `innerHTML, attribute`

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The **document** Object

- ▶ The object **document** can be accessed by any JavaScript code. It **represents the entire web page** and is the entry point to the DOM API.
- ▶ **Sample methods in **document** are**

Find HTML elements `getElementById,`
`getElementsByTagName,`
`getElementsByClassName`

Properties of HTML elements `innerHTML, attribute`

Add or delete elements `createElement,`
`removeChild, appendChild`

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The `document` Object

- ▶ The object `document` can be accessed by any JavaScript code. It represents the entire web page and is the entry point to the DOM API.
- ▶ Sample methods in `document` are

Find HTML elements `getElementById`,
`getElementsByTagName`,
`getElementsByClassName`

Properties of HTML elements `innerHTML`, `attribute`

Add or delete elements `createElement`,
`removeChild`, `appendChild`

Collections of HTML elements `cookie`, `URL`, `elements`,
`forms`

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Change CSS Rules

- ▶ To change CSS rules use the following type of statement:

```
document.getElementById(id).style.<property>  
    = <some style>
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Change CSS Rules

- ▶ To change CSS rules use the following type of statement:

```
document.getElementById(id).style.<property>
    = <some style>
```

- ▶ For example, the statement

```
document.getElementById("p2").style.color =
    "blue";
```

changes the **font color of element with id p2** to blue.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Events

- ▶ The DOM defines many **events**, for example **onClick**, which is fired by an element when the user clicks on it.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Events

- ▶ The DOM defines many **events**, for example **onClick**, which is fired by an element when the user clicks on it.
- ▶ To react to an event, add JavaScript code to the **event handler attribute** of the element that generates the event.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Events

- ▶ The DOM defines many **events**, for example **onClick**, which is fired by an element when the user clicks on it.
- ▶ To react to an event, add JavaScript code to the **event handler attribute** of the element that generates the event.
- ▶ For example, to **change text** in the paragraph when the user clicks it:

```
<p onclick="clicked(this)">Click Me!</p>
```

```
function clicked(source) {  
    source.innerHTML = "You clicked";  
}
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Examples of Events

Mouse events **onclick**, **ondblclick**,
onmousedown, **onmouseover**

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Examples of Events

Mouse events **onclick**, **ondblclick**,
onmousedown, **onmouseover**

Keyboard events **onkeydown**, **onkeypress**,
onkeyup, fired in that order.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Examples of Events

Mouse events **onclick**, **ondblclick**,
onmousedown, **onmouseover**

Keyboard events **onkeydown**, **onkeypress**,
onkeyup, fired in that order.

Object events **onload**, **onunload**

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Examples of Events

Mouse events **onclick**, **ondblclick**,
onmousedown, **onmouseover**

Keyboard events **onkeydown**, **onkeypress**,
onkeyup, fired in that order.

Object events **onload**, **onunload**

Form events **onchange**, **onselect**

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Event Listeners

- ▶ The JavaScript function **addEventListener()** attaches an event listener to the specified element.

```
<button id="myBtn">Try it</button>  
<p id="demo"></p>
```

```
document.getElementById("myBtn").  
    addEventListener("click", displayDate);  
  
function displayDate() {  
    document.getElementById("demo").  
        innerHTML = Date();  
}
```

DOM

BOM

jQuery

X

jQuery

Long Polling

WebSocket

Event Listeners

- ▶ The JavaScript function `addEventListener()` attaches an event listener to the specified element.

```
<button id="myBtn">Try it</button>  
<p id="demo"></p>
```

```
document.getElementById("myBtn").  
    addEventListener("click", displayDate);  
  
function displayDate() {  
    document.getElementById("demo").  
        innerHTML = Date();  
}
```

- ▶ **Multiple** event listeners, even of the same type, can be attached to the same element.

DOM

BOM

jQuery

X

jQuery

Long Polling

WebSocket

Event Listeners

- ▶ The JavaScript function `addEventListener()` attaches an event listener to the specified element.

```
<button id="myBtn">Try it</button>
<p id="demo"></p>
```

```
document.getElementById("myBtn") .
    addEventListener("click", displayDate);

function displayDate() {
    document.getElementById("demo") .
        innerHTML = Date();
}
```

- ▶ **Multiple** event listeners, even of the same type, can be attached to the same element.
- ▶ **Event listeners** is preferred over `onEvent` attributes since it separates JavaScript from HTML, thereby increasing **cohesion**.

DOM

BOM

jQuery

X

jQuery

Long Polling

WebSocket

Passing Parameters to Event Listeners

- ▶ The following code illustrates how to pass **parameters** to event listeners.

```
<button id="myBtn">Try it</button>  
<p id="demo"></p>
```

```
document.getElementById("myBtn").  
  addEventListener("click", function() {  
    showLabel(this);  
  });  
  
function showLabel(source) {  
  document.getElementById("demo").innerHTML  
    = source.innerHTML;  
}
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Question 1

Event Bubbling

- ▶ When an element fires an event, also the event handlers of its **parents** are invoked.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Event Bubbling

- ▶ When an element fires an event, also the event handlers of its **parents** are invoked.
- ▶ An event first triggers the **deepest** possible element, **then its parents** in nesting order.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Event Bubbling Example

▶ HTML:

```
<div onclick="show(1)">1
  <div onclick="show(2)">2
    <div onclick="show(3)">3
      <p id="event-log"></p>
    </div>
  </div>
</div>
```

JavaScript:

```
function show(sourceNo) {
  var curr = document.
    getElementById("event-log").innerHTML;
  document.getElementById("event-log").
    innerHTML = curr + " " + sourceNo;
}
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Event Bubbling Example

▶ HTML:

```
<div onclick="show(1)">1
  <div onclick="show(2)">2
    <div onclick="show(3)">3
      <p id="event-log"></p>
    </div>
  </div>
</div>
```

JavaScript:

```
function show(sourceNo) {
  var curr = document.
    getElementById("event-log").innerHTML;
  document.getElementById("event-log").
    innerHTML = curr + " " + sourceNo;
}
```

- ▶ When clicking the **innermost** div (number 3), the output is **3 2 1**

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Stopping the Bubbling

- ▶ Bubbling is prevented by calling **stopPropagation()** on the **event** object.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Stopping the Bubbling

- ▶ Bubbling is prevented by calling `stopPropagation()` on the `event` object.

```
<div onclick="show(1)">1
  <div onclick="show(2)">2
    <div onclick='show(3, event) '>3
      <p id="log"></p>
    </div>
  </div>
</div>
```

```
function show(sourceNo, event) {
  var curr = document.
    getElementById("log").innerHTML;
  document.getElementById("log").
    innerHTML = curr + " " + sourceNo;
  event.stopPropagation();
}
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Stopping the Bubbling

- ▶ Bubbling is prevented by calling `stopPropagation()` on the `event` object.

```
<div onclick="show(1)">1
  <div onclick="show(2)">2
    <div onclick='show(3, event)'>3
      <p id="log"></p>
    </div>
  </div>
</div>
```

```
function show(sourceNo, event) {
  var curr = document.
    getElementById("log").innerHTML;
  document.getElementById("log").
    innerHTML = curr + " " + sourceNo;
  event.stopPropagation();
}
```

- ▶ When clicking the **innermost** div, output is now 3

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Event Capturing

- ▶ Before bubbling, the event goes the other way, from **outermost to innermost** element. This is called **capturing**.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Event Capturing

- ▶ Before bubbling, the event goes the other way, from **outermost to innermost** element. This is called **capturing**.
- ▶ The capturing phase is ignored by all **onEvent** attributes and event listeners, except listeners with the **useCapture** argument **set to true**:

```
document.getElementById("myId").  
  addEventListener("click", handler, true);
```

DOM

BOM

jQuery

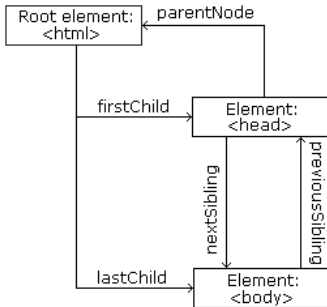
AJAX

Knockout

Long Polling

WebSocket

Navigating the DOM Tree



- ▶ The image to the left illustrates **parent**, **child** and **sibling** relationships between nodes in the DOM tree.

Image from

http://www.w3schools.com/js/js_htmlDOM_navigation.asp

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Navigating the DOM Tree

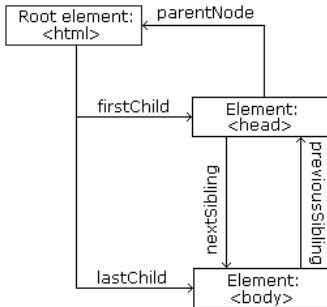


Image from
http://www.w3schools.com/js/js_htmlDOM_navigation.asp

- ▶ The image to the left illustrates **parent**, **child** and **sibling** relationships between nodes in the DOM tree.
- ▶ The DOM tree can be **navigated** with the node properties **parentNode**, **childNodes**, **firstChild**, **lastChild**, **nextSibling**, and **previousSibling**

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Navigating the DOM Tree (Cont'd)

- ▶ Note that, in the code below, the `<p>` node contains a **child text node** with the value **the text**

```
<p>the text</p>
```

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Navigating the DOM Tree (Cont'd)

- ▶ Note that, in the code below, the `<p>` node contains a **child text node** with the value **the text**

```
<p>the text</p>
```

- ▶ Text content can be accessed with the **innerHTML** and **nodeValue** properties.

```
<p id="demo">text content</p>
```

Using **innerHTML**:

```
var text = document.getElementById("demo").  
    innerHTML;
```

Using **nodeValue**:

```
var text = document.getElementById("demo").  
    childNodes[0].nodeValue;
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Adding Elements

- ▶ To add a new element, first **create** it, then **insert** it in the DOM tree.

```
<div id="target"></div>
```

```
var elem = document.createElement("p");  
var text =  
    document.createTextNode("added element");  
elem.appendChild(text);  
document.getElementById("target").  
    appendChild(elem);
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Removing Elements

- ▶ To remove an element, use the **removeChild** method.

```
<div id="parent">  
  <p>To be removed</p>  
</div>
```

```
var parent =  
  document.getElementById("parent");  
var child =  
  parent.getElementsByTagName("p")[0];  
parent.removeChild(child);
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Question 2

Section

- The Document Object Model, DOM
- **The Browser Object Model, BOM**
- The jQuery JavaScript Library
- AJAX
- The Knockout JavaScript Framework
- Long Polling
- WebSocket

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The Browser Object Model, BOM

- ▶ While the DOM provides an API for accessing the current document, the **Browser Object Model, BOM**, provides an API that gives **access to the browser**.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The Browser Object Model, BOM

- ▶ While the DOM provides an API for accessing the current document, the **Browser Object Model, BOM**, provides an API that gives **access to the browser**.
- ▶ The BOM is **not standardized**, but more or less the same methods are implemented in all modern browsers.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The Browser Object Model, BOM

- ▶ While the DOM provides an API for accessing the current document, the **Browser Object Model, BOM**, provides an API that gives **access to the browser**.
- ▶ The BOM is **not standardized**, but more or less the same methods are implemented in all modern browsers.
- ▶ The following slides contain a **short overview of major objects** and methods, to give an idea of what can be done with the BOM.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

BOM Objects

- ▶ The **window** object has:
 - ▶ Properties for **height and width** of the browser window.
 - ▶ Methods to **open, close, move and resize** the browser window.
 - ▶ Methods to execute some code at specified **time-intervals**.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

BOM Objects

- ▶ The **window** object has:
 - ▶ Properties for **height and width** of the browser window.
 - ▶ Methods to **open, close, move and resize** the browser window.
 - ▶ Methods to execute some code at specified **time-intervals**.
- ▶ The **location** object has:
 - ▶ Properties that gives information about the **current URL**.
 - ▶ The **assign** method that **loads a new document**.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

BOM Objects (Cont'd)

- ▶ The **navigator** object can give information about **browser type and browser features**.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

BOM Objects (Cont'd)

- ▶ The **navigator** object can give information about **browser type and browser features**.
- ▶ The **screen** object has properties for **height, width and pixel depth** of the user's screen.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

BOM Objects (Cont'd)

- ▶ The **navigator** object can give information about **browser type and browser features**.
- ▶ The **screen** object has properties for **height, width and pixel depth** of the user's screen.
- ▶ The **document** object has the **cookie** property, which is used to get and set **cookies**.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Section

- The Document Object Model, DOM
- The Browser Object Model, BOM
- **The jQuery JavaScript Library**
- AJAX
- The Knockout JavaScript Framework
- Long Polling
- WebSocket

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The jQuery Library

- ▶ jQuery provides an API that **simplifies** many common JavaScript tasks, like DOM manipulation, CSS manipulation, event handling, effects and animation.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The jQuery Library

- ▶ jQuery provides an API that **simplifies** many common JavaScript tasks, like DOM manipulation, CSS manipulation, event handling, effects and animation.
- ▶ There are **many jQuery plugins** that provide more features.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The jQuery Library

- ▶ jQuery provides an API that **simplifies** many common JavaScript tasks, like DOM manipulation, CSS manipulation, event handling, effects and animation.
- ▶ There are **many jQuery plugins** that provide more features.
- ▶ jQuery **hides cross-browser issues**, all jQuery code will work the same way in all browsers supporting jQuery.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The jQuery Library

- ▶ jQuery provides an API that **simplifies** many common JavaScript tasks, like DOM manipulation, CSS manipulation, event handling, effects and animation.
- ▶ There are **many jQuery plugins** that provide more features.
- ▶ jQuery **hides cross-browser issues**, all jQuery code will work the same way in all browsers supporting jQuery.
- ▶ jQuery is **very commonly used**.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Installing jQuery

- ▶ jQuery is a **JavaScript file**, to use it you just have to **provide a link** to that file.

Installing jQuery

- ▶ jQuery is a [JavaScript file](#), to use it you just have to [provide a link](#) to that file.
- ▶ The jQuery library file comes in two versions:
 - ▶ A [development version](#), which is uncompressed and therefore readable.
 - ▶ A [live website version](#), which has been minified and compressed and therefore is not readable. Instead it is shorter and thereby faster to download.

Installing jQuery (Cont'd)

- ▶ Either you download it from **jquery.com** and place it on **your server**, or you provide a link to a Content Delivery Network, **CDN**, as follows:

```
<script src="https://cdnjs.cloudflare.com/  
ajax/libs/jquery/1.10.2/jquery.min.js">  
</script>
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Installing jQuery (Cont'd)

- ▶ Either you download it from **jquery.com** and place it on **your server**, or you provide a link to a Content Delivery Network, **CDN**, as follows:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.10.2/jquery.min.js">
</script>
```

- ▶ Using a CDN is normally faster, since:

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Installing jQuery (Cont'd)

- ▶ Either you download it from **jquery.com** and place it on **your server**, or you provide a link to a Content Delivery Network, **CDN**, as follows:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.10.2/jquery.min.js">  
</script>
```

- ▶ Using a CDN is normally faster, since:
 - ▶ The file is delivered from the CDNs server **closest to the user**.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Installing jQuery (Cont'd)

- ▶ Either you download it from **jquery.com** and place it on **your server**, or you provide a link to a Content Delivery Network, **CDN**, as follows:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.10.2/jquery.min.js">
</script>
```

- ▶ Using a CDN is normally faster, since:
 - ▶ The file is delivered from the CDNs server **closest to the user**.
 - ▶ Many users already have downloaded jQuery from the CDN when visiting another site. As a result, it is **loaded from browser cache**.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The jQuery Function

- ▶ Very central to jQuery is the [jQuery function](#), which has two names, **jQuery** and the commonly used **\$**.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The jQuery Function

- ▶ Very central to jQuery is the [jQuery function](#), which has two names, **jQuery** and the commonly used **\$**.
- ▶ Remember that **\$** is a perfectly legal JavaScript identifier, there is nothing magic about that name.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The jQuery Function

- ▶ Very central to jQuery is the [jQuery function](#), which has two names, `jQuery` and the commonly used `$`.
- ▶ Remember that `$` is a perfectly legal JavaScript identifier, there is nothing magic about that name.
- ▶ The jQuery function normally takes one parameter, which is either a [CSS selector](#) or a [reference to an object](#) in the document, and returns a jQuery object wrapping all HTML element(s) corresponding to the search criteria.

```
$(document) //The document object.  
$(this)     //The current element.  
$("#someId") //All elements with id "someId"  
$(div)      //All div elements.
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The jQuery Function

- ▶ Very central to jQuery is the [jQuery function](#), which has two names, `jQuery` and the commonly used `$`.
- ▶ Remember that `$` is a perfectly legal JavaScript identifier, there is nothing magic about that name.
- ▶ The jQuery function normally takes one parameter, which is either a [CSS selector](#) or a [reference to an object](#) in the document, and returns a jQuery object wrapping all HTML element(s) corresponding to the search criteria.

```
$(document) //The document object.  
$(this)    //The current element.  
$("#someId") //All elements with id "someId"  
$(div)     //All div elements.
```

- ▶ Any CSS selector can be used as search criteria.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The jQuery Object

- ▶ The **jQuery object** also has two names, **jQuery** and the commonly used **\$**.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The jQuery Object

- ▶ The `jQuery` object also has two names, `jQuery` and the commonly used `$`.
- ▶ The `jQuery` object contains many methods that operate on the wrapped HTML element. For example the `html` method that gets or sets the HTML content of the wrapped element:

```
/* Store the HTML of the element with
id "someId" in the variable "content". */
var content = $("#someId").html();

/* Set the HTML of the element with
id "someId" to "content<br/>". */
$("#someId").html(content + "<br/>");
```

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The jQuery Object (Cont'd)

- ▶ The jQuery object supports **array subscripting** via brackets:

```
$("h1")[0]; //The first h1 element.
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The jQuery Object (Cont'd)

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

- ▶ The jQuery object supports **array subscripting** via brackets:

```
$("h1")[0]; //The first h1 element.
```

- ▶ The jQuery object also has **utility methods** that are not related to a HTML element:

```
// Returns the string "extra whitespace"  
$.trim( "  extra whitespace  " );
```

Event Handlers

- ▶ In jQuery, an event handling function is passed as **argument to a method with the event name** in the jQuery object wrapping the desired event source.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Event Handlers

- ▶ In jQuery, an event handling function is passed as **argument to a method with the event name** in the jQuery object wrapping the desired event source.
- ▶ The following code adds an event handler to all **<p>** elements. The event handler will change the paragraph's text to **You clicked!** when the user clicks it.

```
$( "p" ).click( function () {  
    $( this ).html( "You clicked" );  
} );
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The Document Ready Event

- ▶ jQuery defines the [document ready event](#), which is fired when the DOM has been constructed.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The Document Ready Event

- ▶ jQuery defines the [document ready event](#), which is fired when the DOM has been constructed.
- ▶ It is usually best to [wait for this event](#) before running JavaScript code, to avoid operating on elements that have not been defined.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The Document Ready Event

- ▶ jQuery defines the [document ready event](#), which is fired when the DOM has been constructed.
- ▶ It is usually best to [wait for this event](#) before running JavaScript code, to avoid operating on elements that have not been defined.
- ▶ It is normally not necessary to wait for the JavaScript load event, which fires when everything, including images, is loaded and rendered.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The Document Ready Event

- ▶ jQuery defines the [document ready event](#), which is fired when the DOM has been constructed.
- ▶ It is usually best to [wait for this event](#) before running JavaScript code, to avoid operating on elements that have not been defined.
- ▶ It is normally not necessary to wait for the JavaScript load event, which fires when everything, including images, is loaded and rendered.
- ▶ Therefore, JavaScript code is normally written like this:

```
$(document).ready(function () {  
    // JavaScript code here...  
});
```

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

CSS Rules

- ▶ When passed a **CSS property name**, the **css** method returns the value of that property.

```
$ ("body" ).css ("background-color" );
```

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

CSS Rules

- ▶ When passed a **CSS property name**, the **css** method returns the value of that property.

```
$ ("body" ).css ("background-color" );
```

- ▶ When passed one or more **property:value pairs**, those rules are set for the specified element(s).

```
$ ("body" ).css ("background-color", "yellow" );
```

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Effects

- ▶ There are lots of **effects** provided by jQuery, here are some examples.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Effects

- ▶ There are lots of **effects** provided by jQuery, here are some examples.
 - ▶ The **hide** and **show** methods can **hide/show** elements. It is also possible to specify the speed of the (dis)appearance.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Effects

- ▶ There are lots of **effects** provided by jQuery, here are some examples.
 - ▶ The **hide** and **show** methods can **hide/show** elements. It is also possible to specify the speed of the (dis)appearance.
 - ▶ Various **fade** methods causes an element to **fade in/out**.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Effects

- ▶ There are lots of **effects** provided by jQuery, here are some examples.
 - ▶ The **hide** and **show** methods can **hide/show** elements. It is also possible to specify the speed of the (dis)appearance.
 - ▶ Various **fade** methods causes an element to **fade in/out**.
 - ▶ Various **slide** methods causes an element to **slide up/down**.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Effects

- ▶ There are lots of **effects** provided by jQuery, here are some examples.
 - ▶ The **hide** and **show** methods can **hide/show** elements. It is also possible to specify the speed of the (dis)appearance.
 - ▶ Various **fade** methods causes an element to **fade in/out**.
 - ▶ Various **slide** methods causes an element to **slide up/down**.
 - ▶ The **animate** method is used to create custom **animations**.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Effects (Cont'd)

- ▶ Effects can have **callback functions** that are executed when the effect is done.

```
$("#button").click(function() {  
    $("#p").hide("slow", function() {  
        alert("The paragraph is now hidden");  
    });  
});
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

jQuery Method Chaining

- ▶ Many of the element manipulation methods of the jQuery object **return the jQuery object** itself.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

jQuery Method Chaining

- ▶ Many of the element manipulation methods of the jQuery object **return the jQuery object** itself.
- ▶ This means it is possible to create **chains** of such methods.

```
$( "button" ).click( function () {  
    $( "#p1" ).css( "color", "blue" )  
        .slideUp( 3000 )  
        .slideDown( 2000 );  
});
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Element Content and Element Attributes

- ▶ The **text** method is used to access [text content](#) of an HTML element, the **html** method is used for text content with [HTML tags](#), the **val** method is used for [form field values](#), and **attr** is used for an element's [attributes](#).

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Element Content and Element Attributes

- ▶ The `text` method is used to access [text content](#) of an HTML element, the `html` method is used for text content with [HTML tags](#), the `val` method is used for [form field values](#), and `attr` is used for an element's [attributes](#).
- ▶ If called [without arguments](#), these methods return the current value. If called [with arguments](#) they set a new value.

```
$("#btn").click(function() {  
    var current = $("#test").html();  
});
```

```
$("#btn").click(function() {  
    $("#textField").val("New value");  
});
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Element Content and Element Attributes (Cont'd)

- ▶ The **text**, **html**, **val**, and **attr** methods can have **callback functions**.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Element Content and Element Attributes (Cont'd)

- ▶ The `text`, `html`, `val`, and `attr` methods can have [callback functions](#).
- ▶ The callback function takes two parameters, the [index](#) of the current element in the list of elements selected and the [original value](#).

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Element Content and Element Attributes (Cont'd)

- ▶ The `text`, `html`, `val`, and `attr` methods can have [callback functions](#).
- ▶ The callback function takes two parameters, the [index](#) of the current element in the list of elements selected and the [original value](#).
- ▶ The [return value](#) of the callback function becomes the new text of the element.

```
$("#btn").click(function() {  
    $("p").text(function(i, oldText) {  
        return "Old text: " + oldText +  
            ", index: " + i;  
    });  
});
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

To Add or Remove Elements

- ▶ The **append**, **prepend**, **before**, and **after** methods are used to **add elements**.

```
// Append a list item to the ordered list
// with id "someList".
$("#someList").
    append("<li>Appended item</li>");
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

To Add or Remove Elements

- ▶ The **append**, **prepend**, **before**, and **after** methods are used to **add elements**.

```
// Append a list item to the ordered list
// with id "someList".
$("#someList").
    append("<li>Appended item</li>");
```

- ▶ The **remove** and **empty** methods are used to **remove elements**.

```
// Remove the element with id "#menu".
$("#menu").remove();
```

DOM

BOM

jQuery

AJAX

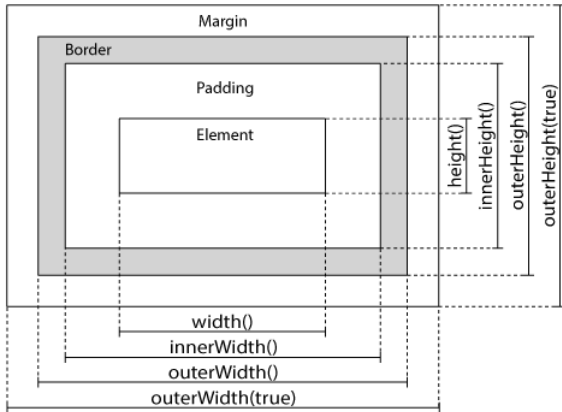
Knockout

Long Polling

WebSocket

Dimensions

The following image, taken from http://www.w3schools.com/jquery/jquery_dimensions.asp, illustrates the methods used to set or get **element dimensions**.



DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Traversing the DOM Tree

Here are samples of jQuery methods used to traverse the **DOM** tree.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Traversing the DOM Tree

Here are samples of jQuery methods used to **traverse the DOM** tree.

parent Returns the parent on the nearest higher level.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Traversing the DOM Tree

Here are samples of jQuery methods used to **traverse the DOM** tree.

parent Returns the parent on the nearest higher level.

parents Returns all parents all the way up to the html element.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Traversing the DOM Tree

Here are samples of jQuery methods used to **traverse the DOM** tree.

parent Returns the parent on the nearest higher level.

parents Returns all parents all the way up to the html element.

children Returns all children on the nearest lower level.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Traversing the DOM Tree

Here are samples of jQuery methods used to traverse the **DOM** tree.

parent Returns the parent on the nearest higher level.

parents Returns all parents all the way up to the html element.

children Returns all children on the nearest lower level.

find Returns all descendants on all lower levels.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Traversing the DOM Tree

Here are samples of jQuery methods used to traverse the **DOM** tree.

parent Returns the parent on the nearest higher level.

parents Returns all parents all the way up to the html element.

children Returns all children on the nearest lower level.

find Returns all descendants on all lower levels.

siblings Returns all siblings.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Traversing the DOM Tree

Here are samples of jQuery methods used to traverse the **DOM** tree.

parent Returns the parent on the nearest higher level.

parents Returns all parents all the way up to the html element.

children Returns all children on the nearest lower level.

find Returns all descendants on all lower levels.

siblings Returns all siblings.

filtering The **first**, **last**, **eq**, and **filter** methods can be used to filter the search results of the methods above.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Question 3

Section

- The Document Object Model, DOM
- The Browser Object Model, BOM
- The jQuery JavaScript Library
- **AJAX**
- The Knockout JavaScript Framework
- Long Polling
- WebSocket

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Loading an Entire Page

- ▶ Traditionally, an **entire page is loaded** when the user clicks a link or a button.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Loading an Entire Page

- ▶ Traditionally, an **entire page is loaded** when the user clicks a link or a button.
- ▶ Here, to load an entire page means that **all HTML** in the page is read from the server.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Loading an Entire Page

- ▶ Traditionally, an **entire page is loaded** when the user clicks a link or a button.
- ▶ Here, to load an entire page means that **all HTML** in the page is read from the server.
- ▶ Dynamic **data is included on the server**, before the HTML is sent to the client, for example using a PHP program.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Loading an Entire Page

- ▶ Traditionally, an **entire page is loaded** when the user clicks a link or a button.
- ▶ Here, to load an entire page means that **all HTML** in the page is read from the server.
- ▶ Dynamic **data is included on the server**, before the HTML is sent to the client, for example using a PHP program.
- ▶ This behavior is appropriate if the entire page content really must change, but that is often **not the case**.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Loading an Entire Page



Consider for example the sample chat application. **All that happens** when the user clicks **Send** is that the new entry is added, the rest of the page is untouched.

DOM

BOM

jQuery

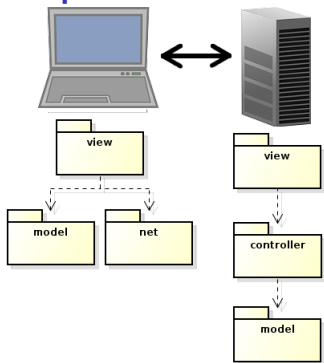
AJAX

Knockout

Long Polling

WebSocket

Repetition: The MVVM Pattern



- ▶ The philosophy behind Model-View-ViewModel, MVVM, is to send **only state changes** from server to client.

DOM

BOM

jQuery

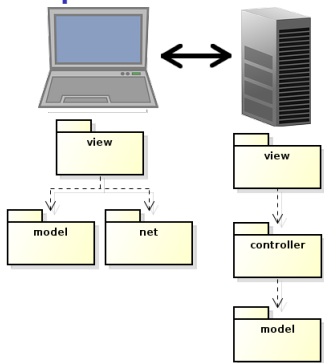
AJAX

Knockout

Long Polling

WebSocket

Repetition: The MVVM Pattern



- ▶ The philosophy behind Model-View-ViewModel, MVVM, is to send **only state changes** from server to client.
- ▶ State changes, which means new data, are **stored in the viewmodel**.

DOM

BOM

jQuery

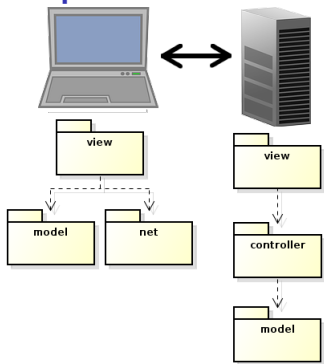
AJAX

Knockout

Long Polling

WebSocket

Repetition: The MVVM Pattern



- ▶ The philosophy behind Model-View-ViewModel, MVVM, is to send **only state changes** from server to client.
- ▶ State changes, which means new data, are **stored in the viewmodel**.
- ▶ Therefore, the viewmodel will always contain the **current state** of the application.

DOM

BOM

jQuery

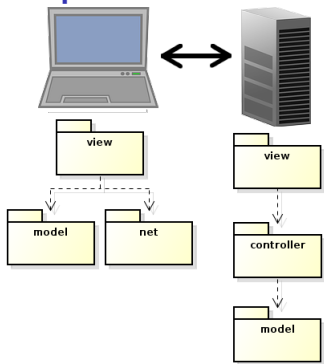
AJAX

Knockout

Long Polling

WebSocket

Repetition: The MVVM Pattern



- ▶ The philosophy behind Model-View-ViewModel, MVVM, is to send **only state changes** from server to client.
- ▶ State changes, which means new data, are **stored in the viewmodel**.
- ▶ Therefore, the viewmodel will always contain the **current state** of the application.
- ▶ The browser view must **reflect the viewmodel** state, preferably using the observer pattern.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

AJAX: To Load Only Data

- ▶ The dominating method to request data from the server, **without reloading** the web page, is **A**synchronous **J**avaScript **A**nd **X**ML, **AJAX**.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

AJAX: To Load Only Data

- ▶ The dominating method to request data from the server, **without reloading** the web page, is **A**synchronous **J**avaScript **A**nd **X**ML, **AJAX**.
- ▶ AJAX is basically a way to use **existing technologies**, such as JavaScript, HTTP and XML.
 - ▶ No new language or markup.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

AJAX: To Load Only Data

- ▶ The dominating method to request data from the server, **without reloading** the web page, is **A**synchronous **J**avaScript **A**nd **X**ML, **AJAX**.
- ▶ AJAX is basically a way to use **existing technologies**, such as JavaScript, HTTP and XML.
 - ▶ No new language or markup.
- ▶ The only thing specific for AJAX is a **JavaScript object**, called **XMLHttpRequest**, which is standardized by W3C.

DOM

BOM

jQuery

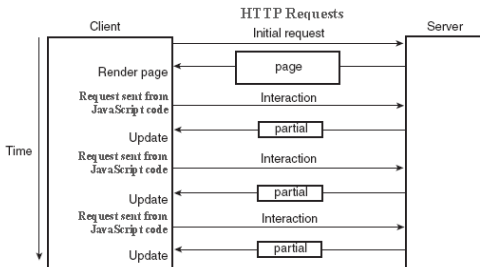
AJAX

Knockout

Long Polling

WebSocket

How Does It Work?



DOM

BOM

jQuery

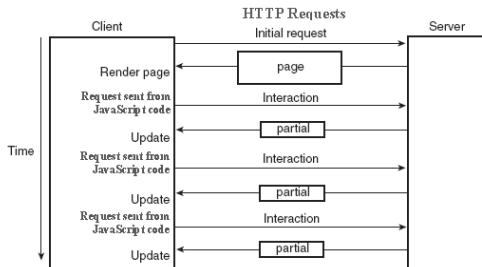
AJAX

Knockout

Long Polling

WebSocket

How Does It Work?



- ▶ Web page is loaded only on **first** request.

DOM

BOM

jQuery

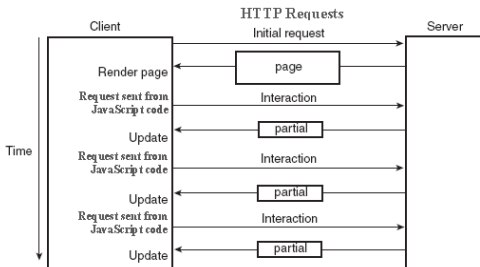
AJAX

Knockout

Long Polling

WebSocket

How Does It Work?



- ▶ Web page is loaded only on **first** request.
- ▶ **Subsequent requests** come from JavaScript code, using **XMLHttpRequest**.

DOM

BOM

jQuery

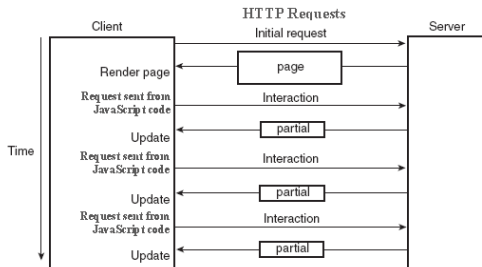
AJAX

Knockout

Long Polling

WebSocket

How Does It Work?



- ▶ Web page is loaded only on **first** request.
- ▶ **Subsequent requests** come from JavaScript code, using **XMLHttpRequest**.
- ▶ The server **returns only data**, no markup.

DOM

BOM

jQuery

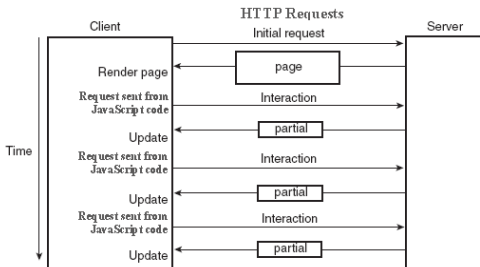
AJAX

Knockout

Long Polling

WebSocket

How Does It Work?



- ▶ Web page is loaded only on **first** request.
- ▶ **Subsequent requests** come from JavaScript code, using **`XMLHttpRequest`**.
- ▶ The server **returns only data**, no markup.
- ▶ Returned data is available to JavaScript code, and is used to **update the web page**, by updating the DOM.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

How Does It Work? (Cont'd)

- ▶ Note that AJAX requests are **ordinary HTTP GET or HTTP POST requests.**

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

How Does It Work? (Cont'd)

- ▶ Note that AJAX requests are **ordinary HTTP GET or HTTP POST requests**.
- ▶ The server directs the request to the **resource specified in the URL**, just as when loading a HTML document.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

How Does It Work? (Cont'd)

- ▶ Note that AJAX requests are **ordinary HTTP GET or HTTP POST requests**.
- ▶ The server directs the request to the **resource specified in the URL**, just as when loading a HTML document.
- ▶ An AJAX request is normally **handled by a program**, for example PHP, which generates a response containing the new data.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Data Format

- ▶ Client and server need to **agree on the format** of the data included in the HTTP response.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Data Format

- ▶ Client and server need to **agree on the format** of the data included in the HTTP response.
- ▶ **XML** is an obvious option, but it has some **drawbacks**:

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Data Format

- ▶ Client and server need to **agree on the format** of the data included in the HTTP response.
- ▶ **XML** is an obvious option, but it has some **drawbacks**:
 - ▶ Interpreting an XML document requires **extra code**.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Data Format

- ▶ Client and server need to **agree on the format** of the data included in the HTTP response.
- ▶ **XML** is an obvious option, but it has some **drawbacks**:
 - ▶ Interpreting an XML document requires **extra code**.
 - ▶ Using a XSLT stylesheet to generate a view is a bit **tricky**.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Data Format

- ▶ Client and server need to **agree on the format** of the data included in the HTTP response.
- ▶ **XML** is an obvious option, but it has some **drawbacks**:
 - ▶ Interpreting an XML document requires **extra code**.
 - ▶ Using a XSLT stylesheet to generate a view is a bit **tricky**.
 - ▶ XML documents are quite **long and wordy**.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Data Format

- ▶ Client and server need to **agree on the format** of the data included in the HTTP response.
- ▶ **XML** is an obvious option, but it has some **drawbacks**:
 - ▶ Interpreting an XML document requires **extra code**.
 - ▶ Using a XSLT stylesheet to generate a view is a bit **tricky**.
 - ▶ XML documents are quite **long and wordy**.
- ▶ Therefore, **JavaScript Object Notation, JSON**, is normally used instead of XML.
 - ▶ **Compact and easy to translate** to JavaScript objects.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

JSON

- ▶ The **JSON syntax** is very simple:

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

JSON

- ▶ The **JSON syntax** is very simple:
 - ▶ **Data** is name/value pairs:

```
"firstName": "Olle"
```

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

JSON

- ▶ The **JSON syntax** is very simple:
 - ▶ **Data** is name/value pairs:
`"firstName": "Olle"`
 - ▶ Data is **separated** by commas.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

JSON

- ▶ The **JSON syntax** is very simple:

- ▶ **Data** is name/value pairs:

```
"firstName": "Olle"
```

- ▶ Data is **separated** by commas.
- ▶ **Objects** are denoted with { and }:

```
{"firstName": "Olle", "lastName": "Olsson"}
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

JSON

- ▶ The **JSON syntax** is very simple:

- ▶ **Data** is name/value pairs:

```
"firstName": "Olle"
```

- ▶ Data is **separated** by commas.
- ▶ **Objects** are denoted with { and }:

```
{"firstName": "Olle", "lastName": "Olsson"}
```

- ▶ **Arrays** are denoted with [and]:

```
"employees": [  
  {"firstName": "Olle", "lastName": "Olsson"},  
  {"firstName": "Stina", "lastName": "Nilsson"}  
]
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

JSON

- ▶ The **JSON syntax** is very simple:

- ▶ **Data** is name/value pairs:

```
"firstName": "Olle"
```

- ▶ Data is **separated** by commas.
- ▶ **Objects** are denoted with { and }:

```
{"firstName": "Olle", "lastName": "Olsson"}
```

- ▶ **Arrays** are denoted with [and]:

```
"employees": [  
  {"firstName": "Olle", "lastName": "Olsson"},  
  {"firstName": "Stina", "lastName": "Nilsson"}  
]
```

- ▶ Data types are JavaScript types, for example **string**, `"abcd"`; **integer**, `123`; **boolean**, `false`

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

JSON is not an Alternative to XML

- ▶ Note that JSON is **not a general alternative** to XML. There is nothing like namespace, DTD, Schema, XSLT or anything else of all the XML standards.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

JSON is not an Alternative to XML

- ▶ Note that JSON is **not a general alternative** to XML. There is nothing like namespace, DTD, Schema, XSLT or anything else of all the XML standards.
- ▶ JSON is just a format suitable for transferring **JavaScript values**.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The Client: jQuery AJAX Methods

- ▶ Instead of covering the **XMLHttpRequest** object, we will look at some convenient **jQuery functions**.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The Client: jQuery AJAX Methods

- ▶ Instead of covering the **XMLHttpRequest** object, we will look at some convenient **jQuery functions**.
- ▶ **getJSON** sends a HTTP GET request. Data can be included as a **query string** and the response is **parsed as JSON data**.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The Client: jQuery AJAX Methods

- ▶ Instead of covering the `XMLHttpRequest` object, we will look at some convenient `jQuery` functions.
- ▶ `getJSON` sends a HTTP GET request. Data can be included as a `query string` and the response is `parsed as JSON data`.
- ▶

```
$.getJSON(url, "reqData=" + someVariable, function(returnedData) {  
    //Handle returnedData, which is  
    //the received JSON data, parsed  
    //to a JavaScript variable.  
});
```

An HTTP `GET` request is sent to the URL specified in `url`. The request has the `query string` `reqData=<value of someVariable>` and the anonymous `callback function is executed` when the server's response arrives.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

jQuery AJAX Methods (Cont'd)

- ▶ **post** sends data with a **HTTP POST request**.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

jQuery AJAX Methods (Cont'd)

- ▶ `post` sends data with a **HTTP POST request**.
- ▶ `$.post(url, "data=" + someVariable);`

An HTTP **POST request** is sent to the URL specified in `url`. The request has the **body data=<value of someVariable>**.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

jQuery AJAX Methods (Cont'd)

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

- ▶ **post** sends data with a **HTTP POST request**.
- ▶ `$.post(url, "data=" + someVariable);`

An HTTP **POST request** is sent to the URL specified in `url`. The request has the **body data=<value of someVariable>**.

- ▶ jQuery has **many more** AJAX methods.

The Server: JSON handling in PHP

- ▶ Remember that an AJAX request is a normal HTTP request.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The Server: JSON handling in PHP

- ▶ Remember that an AJAX request is a normal HTTP request.
- ▶ Therefore, to handle an AJAX request is **no different** from other request handling.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The Server: JSON handling in PHP

- ▶ Remember that an AJAX request is a **normal HTTP request**.
- ▶ Therefore, to handle an AJAX request is **no different** from other request handling.
- ▶ What is specific for AJAX interaction, is that we have to generate a **JSON response**.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The Server: JSON handling in PHP

- ▶ Remember that an AJAX request is a **normal HTTP request**.
- ▶ Therefore, to handle an AJAX request is **no different** from other request handling.
- ▶ What is specific for AJAX interaction, is that we have to generate a **JSON response**.
- ▶ `json_encode($aPhpObject)`

The **json_encode** PHP method encodes the PHP object in **aPhpObject** to JSON representation.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

JSON handling in PHP (Cont'd)

```
class SomeClass implements \JsonSerializable {
    private $some_var;
    ...
    public function jsonSerialize() {
        $json_obj = new stdClass;
        $json_obj->someVar = $this->some_var;
        ...
        return $json_obj;
    }
}
```

- ▶ The object that shall be JSON encoded must be of a class that **implements `JsonSerializable`**.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

JSON handling in PHP (Cont'd)

```
class SomeClass implements \JsonSerializable {
    private $some_var;
    ...
    public function jsonSerialize() {
        $json_obj = new stdClass;
        $json_obj->someVar = $this->some_var;
        ...
        return $json_obj;
    }
}
```

- ▶ The object that shall be JSON encoded must be of a class that **implements `JsonSerializable`**.
- ▶ That class must have a **method** called **`jsonSerialize`**, which **returns an object** containing all relevant fields.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

JSON handling in PHP (Cont'd)

```
class SomeClass implements \JsonSerializable {
    private $some_var;
    ...
    public function jsonSerialize() {
        $json_obj = new stdClass;
        $json_obj->someVar = $this->some_var;
        ...
        return $json_obj;
    }
}
```

- ▶ The object that shall be JSON encoded must be of a class that **implements `JsonSerializable`**.
- ▶ That class must have a **method** called **`jsonSerialize`**, which **returns an object** containing all relevant fields.
- ▶ That returned object must be **possible to encode** with **`json_encode`**.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Question 4

Section

- The Document Object Model, DOM
- The Browser Object Model, BOM
- The jQuery JavaScript Library
- AJAX
- **The Knockout JavaScript Framework**
- Long Polling
- WebSocket

DOM

BOM

jQuery

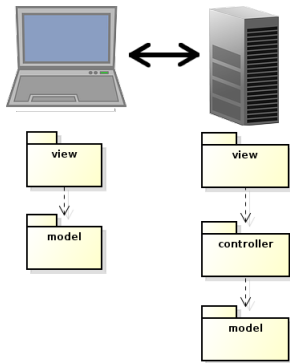
AJAX

Knockout

Long Polling

WebSocket

Reminder: The MVVM Pattern



- ▶ The MVVM pattern introduces a **client-side model** which **reflects the server-side model** and is responsible for notifying the view of updates.

DOM

BOM

jQuery

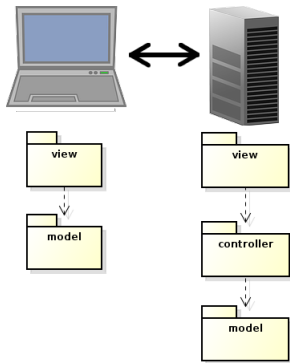
AJAX

Knockout

Long Polling

WebSocket

Reminder: The MVVM Pattern



- ▶ The MVVM pattern introduces a **client-side model** which **reflects the server-side model** and is responsible for notifying the view of updates.
- ▶ The server-side view is **relieved from creating the HTML**. There will not be PHP in the HTML files!

DOM

BOM

jQuery

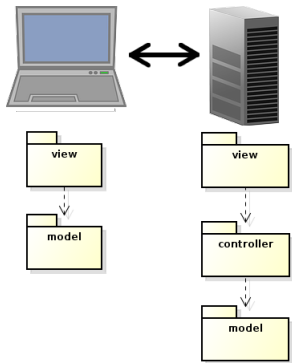
AJAX

Knockout

Long Polling

WebSocket

Reminder: The MVVM Pattern



- ▶ The MVVM pattern introduces a **client-side model** which **reflects the server-side model** and is responsible for notifying the view of updates.
- ▶ The server-side view is **relieved from creating the HTML**. There will not be PHP in the HTML files!
- ▶ Also, network **communication is reduced**, since only model updates are fetched from the server. There is no need to reload the entire web page at each user action.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

MVVM JavaScript Frameworks

- ▶ The code for implementing the Observer pattern to have the **view reflect changes in the viewmodel** (and viewmodel reflect changes in the view), will be the same for more or less all applications.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

MVVM JavaScript Frameworks

- ▶ The code for implementing the Observer pattern to have the **view reflect changes in the viewmodel** (and viewmodel reflect changes in the view), will be the same for more or less all applications.
- ▶ Also the code for **viewmodel-to-server communication** will be quite similar in all applications.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

MVVM JavaScript Frameworks

- ▶ The code for implementing the Observer pattern to have the [view reflect changes in the viewmodel](#) (and viewmodel reflect changes in the view), will be the same for more or less all applications.
- ▶ Also the code for [viewmodel-to-server communication](#) will be quite similar in all applications.
- ▶ This calls for a [client-side framework](#), since we do not want to rewrite the same code for each new application. This is exactly the purpose of [Knockout](#).

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

MVVM JavaScript Frameworks

- ▶ The code for implementing the Observer pattern to have the [view reflect changes in the viewmodel](#) (and viewmodel reflect changes in the view), will be the same for more or less all applications.
- ▶ Also the code for [viewmodel-to-server communication](#) will be quite similar in all applications.
- ▶ This calls for a [client-side framework](#), since we do not want to rewrite the same code for each new application. This is exactly the purpose of [Knockout](#).
- ▶ There are also many alternative frameworks, perhaps the most commonly used is [Backbone](#). Backbone is more used and more powerful than Knockout, but too complicated for this course.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The Knockout JavaScript Framework

- ▶ Like jQuery, Knockout is a [JavaScript file](#) that can be linked from a CDN, for example:

```
<script src="https://cdnjs.cloudflare.com/
  ajax/libs/knockout/3.1.0/knockout-min.js">
</script>
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The Knockout JavaScript Framework

- ▶ Like jQuery, Knockout is a [JavaScript file](#) the can be linked from a CDN, for example:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/knockout/3.1.0/knockout-min.js">
</script>
```

- ▶ Knockout should be used [together with jQuery](#), which handles low-level DOM interaction.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The Knockout JavaScript Framework

- ▶ Like jQuery, Knockout is a [JavaScript file](#) the can be linked from a CDN, for example:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/knockout/3.1.0/knockout-min.js">
</script>
```

- ▶ Knockout should be used [together with jQuery](#), which handles low-level DOM interaction.
- ▶ Knockout implements the [MVVM pattern](#), by managing [View-to-Viewmodel bindings](#).

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The Knockout JavaScript Framework

- ▶ Like jQuery, Knockout is a [JavaScript file](#) the can be linked from a CDN, for example:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/knockout/3.1.0/knockout-min.js">
</script>
```

- ▶ Knockout should be used [together with jQuery](#), which handles low-level DOM interaction.
- ▶ Knockout implements the [MVVM pattern](#), by managing [View-to-Viewmodel bindings](#).
- ▶ The following slides contain a brief introduction to Knockout. For a more extensive guide, see <http://knockoutjs.com/documentation/introduction.html>

DOM

BOM

jQuery

AJAX

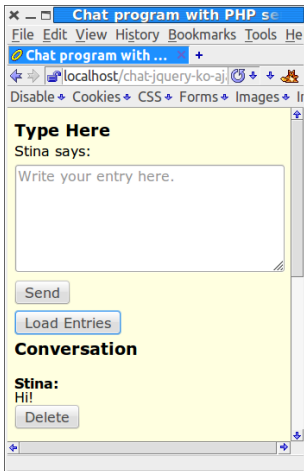
Knockout

Long Polling

WebSocket

The Chat Application

- ▶ To illustrate this Knockout tutorial, we will use the [chat application](#).



DOM

BOM

jQuery

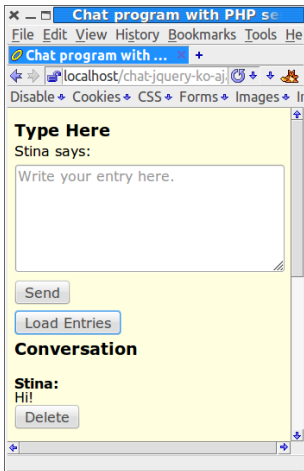
AJAX

Knockout

Long Polling

WebSocket

The Chat Application



- ▶ To illustrate this Knockout tutorial, we will use the **chat application**.
- ▶ We will **create a viewmodel** that holds the current conversation.

DOM

BOM

jQuery

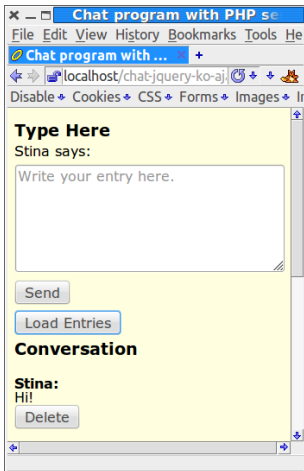
AJAX

Knockout

Long Polling

WebSocket

The Chat Application



- ▶ To illustrate this Knockout tutorial, we will use the **chat application**.
- ▶ We will **create a viewmodel** that holds the current conversation.
- ▶ The view shall be **updated** as soon as the **viewmodel changes state**.

DOM

BOM

jQuery

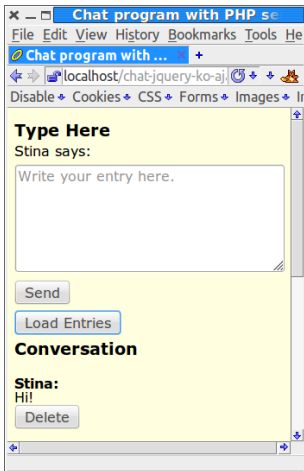
AJAX

Knockout

Long Polling

WebSocket

The Chat Application



- ▶ To illustrate this Knockout tutorial, we will use the **chat application**.
- ▶ We will **create a viewmodel** that holds the current conversation.
- ▶ The view shall be **updated** as soon as the **viewmodel changes state**.
- ▶ The viewmodel can change state either because the **user wrote an entry** or because another user wrote an entry, which was **loaded from the server** to this user's viewmodel.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The Knockout Viewmodel

- ▶ The viewmodel is an ordinary **JavaScript object**, but to make use of Knockout's viewmodel-to-view binding (observer pattern), the properties must be declared as **observables**.

```
function Person(name, age) {  
    var self = this;  
    self.name = ko.observable(name);  
    self.age = ko.observable(age);  
}
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The Knockout Viewmodel

- ▶ The viewmodel is an ordinary **JavaScript object**, but to make use of Knockout's viewmodel-to-view binding (observer pattern), the properties must be declared as **observables**.

```
function Person(name, age) {  
    var self = this;  
    self.name = ko.observable(name);  
    self.age = ko.observable(age);  
}
```

- ▶ To read or write a property value, call the property as a function.

```
var olle = new Person("Olle", 35);  
olle.name(); // Returns "Olle"  
olle.age(36); // Sets the age to 36.
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The Chat Application's Viewmodel

- ▶ The Chat's viewmodel has two objects.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The Chat Application's Viewmodel

- ▶ The Chat's viewmodel has two objects.
 - ▶ **EntryToAdd** represents a newly written entry that shall be sent to the server.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The Chat Application's Viewmodel

- ▶ The Chat's viewmodel has two objects.
 - ▶ **EntryToAdd** represents a newly written entry that shall be sent to the server.
 - ▶ **Conversation** represents the entire conversation.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The Chat Application's Viewmodel

- ▶ The Chat's viewmodel has two objects.
 - ▶ **EntryToAdd** represents a newly written entry that shall be sent to the server.
 - ▶ **Conversation** represents the entire conversation.
- ▶ The **EntryToAdd** object has two properties, **nickName** and **msg**

```
function EntryToAdd() {  
    var self = this;  
    self.nickName = ko.observable();  
    self.msg = ko.observable("");  
    ...  
}
```

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The Chat Application's Viewmodel (Cont'd)

```
var entryToAdd = new EntryToAdd();  
ko.applyBindings(entryToAdd,  
                 document.getElementById('new-entry'));
```

- ▶ The viewmodel must be [registered with Knockout](#) to enable notifying the observers in the view.

The Chat Application's Viewmodel (Cont'd)

```
var entryToAdd = new EntryToAdd();  
ko.applyBindings(entryToAdd,  
                 document.getElementById('new-entry'));
```

- ▶ The viewmodel must be [registered with Knockout](#) to enable notifying the observers in the view.
- ▶ After a JavaScript object has been registered with knockout using the function **applyBindings**, it will always have [the same state](#) as the elements in the DOM.

The Chat Application's Viewmodel (Cont'd)

```
var entryToAdd = new EntryToAdd();  
ko.applyBindings(entryToAdd,  
                 document.getElementById('new-entry'));
```

- ▶ The viewmodel must be [registered with Knockout](#) to enable notifying the observers in the view.
- ▶ After a JavaScript object has been registered with knockout using the function **applyBindings**, it will always have [the same state](#) as the elements in the DOM.
 - ▶ When the object changes state, the DOM will change state. When the DOM changes state, the object will change state.

The Chat Application's Viewmodel (Cont'd)

```
var entryToAdd = new EntryToAdd();  
ko.applyBindings(entryToAdd,  
                 document.getElementById('new-entry'));
```

- ▶ The viewmodel must be [registered with Knockout](#) to enable notifying the observers in the view.
- ▶ After a JavaScript object has been registered with knockout using the function `applyBindings`, it will always have [the same state](#) as the elements in the DOM.
 - ▶ When the object changes state, the DOM will change state. When the DOM changes state, the object will change state.
- ▶ The second parameter tells to which element in the DOM `entryToAdd` shall be bound. There can only be at most one bound object per element.

Data Bindings

- ▶ A HTML element in the view is connected to a viewmodel property with a **binding**.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Data Bindings

- ▶ A HTML element in the view is connected to a viewmodel property with a [binding](#).
- ▶ A binding is declared by adding the **data-bind** attribute to the HTML element.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Data Bindings

- ▶ A HTML element in the view is connected to a viewmodel property with a [binding](#).
- ▶ A binding is declared by adding the **data-bind** attribute to the HTML element.
- ▶ There are many different types of bindings, like:

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Data Bindings

- ▶ A HTML element in the view is connected to a viewmodel property with a [binding](#).
- ▶ A binding is declared by adding the **data-bind** attribute to the HTML element.
- ▶ There are many different types of bindings, like:

text The property value is inserted to the HTML element.

```
The message is:  
<span data-bind="text: msg"></span>
```

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Data Bindings

- ▶ A HTML element in the view is connected to a viewmodel property with a **binding**.
- ▶ A binding is declared by adding the **data-bind** attribute to the HTML element.
- ▶ There are many different types of bindings, like:

text The property value is inserted to the HTML element.

```
The message is:  
<span data-bind="text: msg"></span>
```

visible Decides if the element is rendered.

```
<div data-bind=  
  "visible: shouldShowMessage">
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Data Bindings

- ▶ A HTML element in the view is connected to a viewmodel property with a **binding**.
- ▶ A binding is declared by adding the **data-bind** attribute to the HTML element.
- ▶ There are many different types of bindings, like:

text The property value is inserted to the HTML element.

```
The message is:  
<span data-bind="text: msg"></span>
```

visible Decides if the element is rendered.

```
<div data-bind=  
  "visible: shouldShowMessage">
```

css Adds or removes CSS classes. The following binding adds the class **warning** if the **profit** property is negative.

```
<div data-bind=  
  "css: {warning: profit () < 0 }">
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Form Field Bindings

There are also bindings for form elements, such as:

click Specifies a method that is called when the element is clicked.

```
<button data-bind="click: clickHandler">Click me</button>
```

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Form Field Bindings

There are also bindings for form elements, such as:

click Specifies a method that is called when the element is clicked.

```
<button data-bind="click: clickHandler">Click me</button>
```

textInput Binds a text field or text area to a viewmodel property

```
<input type="text" data-bind="textInput: username"/>
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Form Field Bindings

There are also bindings for form elements, such as:

click Specifies a method that is called when the element is clicked.

```
<button data-bind="click: clickHandler">Click me</button>
```

textInput Binds a text field or text area to a viewmodel property

```
<input type="text" data-bind="textInput: username"/>
```

enable The element is enabled only when the value is **true**

```
Your cellphone number:  
<input type='text' data-bind="textInput: cellphoneNumber, enable: hasCellphone"/>
```

DOM

BOM

jQuery

AJAX

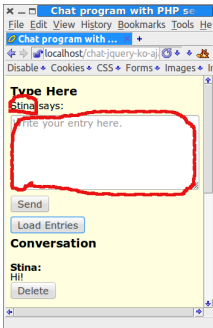
Knockout

Long Polling

WebSocket

Value Data-Bindings

- ▶ Now let us create the data bindings for the **nick name** text and **msg** text area.



```

<label id="nickNameLabel" for="entry">
  <span data-bind="text: nickName">
    </span>
  says:
</label>

...
<textarea id= "entry" rows = 5
  data-bind="textInput: msg"
  placeholder="Write entry here.">
</textarea>

```

DOM

BOM

jQuery

AJAX

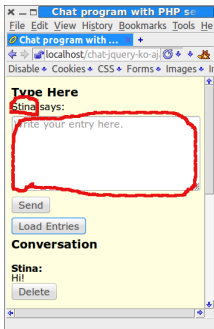
Knockout

Long Polling

WebSocket

Value Data-Bindings

- ▶ Now let us create the data bindings for the **nick name** text and **msg** text area.



```

<label id="nickNameLabel" for="entry">
  <span data-bind="text: nickName">
    </span>
  says:
</label>
...
<textarea id= "entry" rows = 5
  data-bind="textInput: msg"
  placeholder="Write entry here.">
</textarea>

```

- ▶ Now the **nickname** element and the **nickName** property in the **EntryToAdd** object will **always** have the same value.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Value Data-Bindings

- ▶ Now let us create the data bindings for the **nick name** text and **msg** text area.

```

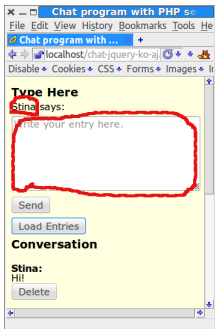
<label id="nickNameLabel" for="entry">
  <span data-bind="text: nickName">
    </span>
  says:
</label>

...
<textarea id="entry" rows=5
  data-bind="textInput: msg"
  placeholder="Write entry here.">
</textarea>

```

- ▶ Now the **nickname** element and the **nickName** property in the **EntryToAdd** object will **always** have the same value.

- ▶ Also, the text in the textarea will **always be the same** as the value of the **msg** property



DOM

BOM

jQuery

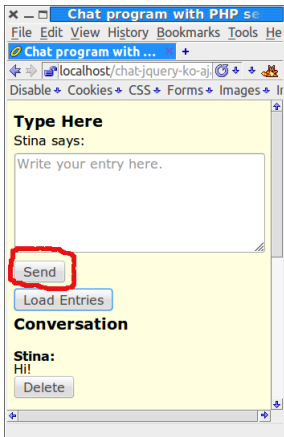
AJAX

Knockout

Long Polling

WebSocket

Click Data-Bindings



```
<button data-bind="click: sendEntry">  
  Send  
</button>
```

- ▶ Then let us create the data-binding for the **Send** button.

DOM

BOM

jQuery

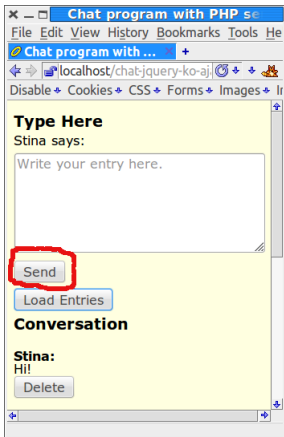
AJAX

Knockout

Long Polling

WebSocket

Click Data-Bindings



```
<button data-bind="click: sendEntry">  
  Send  
</button>
```

- ▶ Then let us create the data-binding for the **Send** button.
- ▶ Here we specified that the **sendEntry** method is called when the user clicks the button.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The `sendEntry` Method

```
1 self.sendEntry = function () {
2     if (self.msg() !== "") {
3         $.post(writeUrl,
4             "msg=" + ko.toJSON(self.msg));
5         self.msg("");
6     }
7 };
```

- ▶ Line 2 checks that the user has typed a message.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The `sendEntry` Method

```
1 self.sendEntry = function () {  
2     if (self.msg() !== "") {  
3         $.post(writeUrl,  
4             "msg=" + ko.toJSON(self.msg));  
5         self.msg("");  
6     }  
7 };
```

- ▶ Line 2 checks that the user has typed a message.
- ▶ Lines 3-4 sends the new entry to the server in a HTTP **post** request.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The `sendEntry` Method

```
1 self.sendEntry = function () {
2     if (self.msg() !== "") {
3         $.post(writeUrl,
4             "msg=" + ko.toJSON(self.msg));
5         self.msg("");
6     }
7 };
```

- ▶ Line 2 checks that the user has typed a message.
- ▶ Lines 3-4 sends the new entry to the server in a HTTP `post` request.
- ▶ The call to `ko.toJSON` on line 4 converts the message to JSON format.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The `sendEntry` Method

```
1 self.sendEntry = function () {
2     if (self.msg() !== "") {
3         $.post(writeUrl,
4             "msg=" + ko.toJSON(self.msg));
5         self.msg("");
6     }
7 };
```

- ▶ Line 2 checks that the user has typed a message.
- ▶ Lines 3-4 sends the new entry to the server in a HTTP `post` request.
- ▶ The call to `ko.toJSON` on line 4 converts the message to JSON format.
- ▶ Line 5 clears the text area.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Read Nickname From Server

- ▶ One question remains, how did the nickname show up correctly in the view? Note that it is not inserted in php code on the server (very good!):

```
<label id="nickNameLabel" for="entry">  
  <span data-bind="text: nickName">  
  </span>  
  says:  
</label>
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Read Nickname From Server

- ▶ One question remains, how did the nickname show up correctly in the view? Note that it is not inserted in php code on the server (very good!):

```
<label id="nickNameLabel" for="entry">
  <span data-bind="text: nickName">
  </span>
  says:
</label>
```

- ▶ The answer is that the nickname was read in an extra AJAX call from the **EntryToAdd** constructor.

```
function EntryToAdd() {
  var self = this;
  self.nickName = ko.observable();
  ...
  $.getJSON(nickNameUrl, function (username) {
    self.nickName(username);
  });
}
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Read Nickname From Server (Cont'd)

- ▶ The ajax call on the previous slide is handled by the **GetUsername** request handler (code below is not complete).

```
class GetUsername extends AbstractRequestHandler {  
  
    protected function doExecute() {  
        $this->addVariable('jsonData',  
                           $contr->getUsername());  
        return 'json-view';  
    }  
  
}
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Read Nickname From Server (Cont'd)

- ▶ The ajax call on the previous slide is handled by the **GetUsername** request handler (code below is not complete).

```
class GetUsername extends AbstractRequestHandler {  
  
    protected function doExecute() {  
        $this->addVariable('jsonData',  
                           $contr->getUsername());  
        return 'json-view';  
    }  
  
}
```

- ▶ **json-view.php** just echos the value of **\$jsonData**.

```
echo \json_encode($jsonData);
```

The **C**onversation object in the viewmodel

- ▶ Now we have covered everything in the **EntryToAdd** object in the viewmodel.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The **C**onversation object in the viewmodel

- ▶ Now we have covered everything in the **E**ntry**T**o**A**dd object in the viewmodel.
- ▶ There is one more object in the viewmodel, **C**onversation, which contains the entire chat conversation.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The **C**onversation object in the viewmodel

- ▶ Now we have covered everything in the **EntryToAdd** object in the viewmodel.
- ▶ There is one more object in the viewmodel, **C**onversation, which contains the entire chat conversation.
- ▶ **C**onversation has a property, **e**ntries that holds an array of all conversation entries.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

The **C**onversation object in the viewmodel

- ▶ Now we have covered everything in the **EntryToAdd** object in the viewmodel.
- ▶ There is one more object in the viewmodel, **C**onversation, which contains the entire chat conversation.
- ▶ **C**onversation has a property, **e**ntries that holds an array of all conversation entries.
- ▶ To understand how that works, we must look at how knockout binds arrays to elements in the DOM.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Observable Arrays

- ▶ To observe an [array](#), use **observableArray**.

```
var myObservableArray = ko.observableArray();
```

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Observable Arrays

- ▶ To observe an [array](#), use `observableArray`.

```
var myObservableArray = ko.observableArray();
```

- ▶ The `entries` property in `Conversation` is an `observableArray` and holds an array of objects having the two properties `nickName` and `msg`

```
function Conversation(entryToAdd) {  
    var self = this;  
    ...  
    self.entries = ko.observableArray();  
    ...  
}
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Flow Control Data-Bindings

- ▶ When binding **entries** to the DOM, we must use the **flow control data bindings** **if** and **foreach**. Let's first have a look at those.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Flow Control Data-Bindings

- ▶ When binding **entries** to the DOM, we must use the **flow control data bindings** **if** and **foreach**. Let's first have a look at those.
- ▶ **foreach** binds each element to in an array to the DOM.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Flow Control Data-Bindings

- ▶ When binding **entries** to the DOM, we must use the **flow control data bindings** **if** and **foreach**. Let's first have a look at those.
- ▶ **foreach** binds each element to in an array to the DOM.
- ▶ Whenever elements are **added, removed, or re-ordered** in a bound observable array, the DOM will be updated to reflect the new array contents.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

foreach Data-Binding (Cont'd)

Assuming **people** is a JavaScript array of objects with **firstName** and **lastName** properties, the following generates a table with one object per row.

```
<table>
  <thead>
    <tr><th>First name</th>
    <th>Last name</th></tr>
  </thead>
  <tbody data-bind="foreach: people">
    <tr>
      <td data-bind="text: firstName"></td>
      <td data-bind="text: lastName"></td>
    </tr>
  </tbody>
</table>
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

foreach Data-Binding (Cont'd)

- ▶ The current array element is referred using **\$data**

```
<ul data-bind="foreach: months">
  <li>
    <span data-bind="text: $data"></span>
  </li>
</ul>
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

foreach Data-Binding (Cont'd)

- ▶ The current array element is referred using `$data`

```
<ul data-bind="foreach: months">
  <li>
    <span data-bind="text: $data"></span>
  </li>
</ul>
```

- ▶ **as** gives an alias to the current array element.

```
<ul data-bind=
  "foreach: { data: categories, as: 'category' }">
  <li>
    <ul data-bind=
      "foreach: { data: items, as: 'item' }">
      <li>
        <span data-bind="text: category.name">
        </span>:
        <span data-bind="text: item"></span>
      </li>
    </ul>
  </li>
</ul>
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Containerless Control Flow Syntax

- ▶ If there is no containing element for the data-binding, use the [containerless syntax](#), based on HTML comment tags.

```
<!-- ko foreach: {data:items, as:'item'} -->  
  <p>  
    <span data-bind="text:item.price">  
      </span>  
  </p>  
<!-- /ko -->
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Containerless Control Flow Syntax

- ▶ If there is no containing element for the data-binding, use the **containerless syntax**, based on HTML comment tags.

```
<!-- ko foreach: {data:items, as:'item'} -->  
  <p>  
    <span data-bind="text:item.price">  
    </span>  
  </p>  
<!-- /ko -->
```

- ▶ An **if** data-binding displays the enclosed elements **only** if the condition is true.

```
<!-- ko if: item.isOffer -->  
  <p>  
    Offer!  
  </p>  
<!-- /ko -->
```

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

The Conversation Data-Binding

- Now we can create the data binding for the **conversation part** of the chat application (code is not complete).

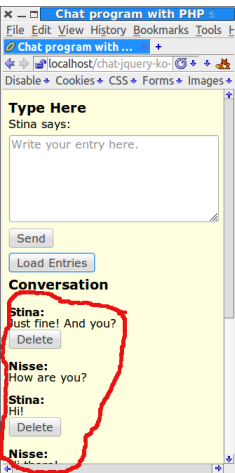
DOM

BOM

jQuery

AJAX

Knockout



```

<!-- ko foreach: {data: entries, as: 'entry'} -->
  <p>
    <span data-bind='text: entry.nickName'></span>:
  </p>
  <!-- ko foreach: entry.msg -->
    <p>
      <span data-bind='text: $data'></span>
    </p>
  <!-- /ko -->
  <!-- ko if: entry.iWroteThisEntry -->
    <p>
      <button data-bind=
        'click: $parent.deleteEntry'>
        Delete
      </button>
    </p>
  <!-- /ko -->
<!-- /ko -->

```

The Conversation Data-Binding

- ▶ Now we can create the data binding for the **conversation part** of the chat application (code is not complete).
- ▶ The inner loop is needed for multi-line messages.

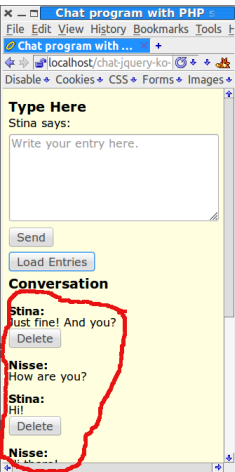
DOM

BOM

jQuery

AJAX

Knockout



```

<!-- ko foreach: {data: entries, as: 'entry'} -->
  <p>
    <span data-bind='text: entry.nickName'></span>:
  </p>
  <!-- ko foreach: entry.msg -->
    <p>
      <span data-bind='text: $data'></span>
    </p>
  <!-- /ko -->
  <!-- ko if: entry.iWroteThisEntry -->
    <p>
      <button data-bind=
        'click: $parent.deleteEntry'>
        Delete
      </button>
    </p>
  <!-- /ko -->
<!-- /ko -->

```

No PHP! (Almost)

- ▶ That was all, now we have seen an **overview of some features** in Knockout, and also created the chat application.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

No PHP! (Almost)

- ▶ That was all, now we have seen an **overview of some features** in Knockout, and also created the chat application.
- ▶ As a result, there is now **no PHP code** left in the html files!

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

No PHP! (Almost)

- ▶ That was all, now we have seen an **overview of some features** in Knockout, and also created the chat application.
- ▶ As a result, there is now **no PHP code** left in the html files!
 - ▶ Except for including fragments and creating link paths.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Without JavaScript framework

Without a JavaScript framework that handles viewmodel-to-DOM bindings, where would we store the nickname and chat conversation? There are three options, each with serious drawbacks.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Without JavaScript framework

Without a JavaScript framework that handles viewmodel-to-DOM bindings, where would we store the nickname and chat conversation? There are three options, each with **serious drawbacks**.

1. **Write the JavaScript code that is now in the framework.** This means quite a lot of quite tricky handling of the DOM, knockout.js contains thousand of lines of JavaScript code.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Without JavaScript framework

Without a JavaScript framework that handles viewmodel-to-DOM bindings, where would we store the nickname and chat conversation? There are three options, each with **serious drawbacks**.

1. **Write the JavaScript code that is now in the framework.** This means quite a lot of quite tricky handling of the DOM, knockout.js contains thousand of lines of JavaScript code.
2. **Do not create a viewmodel,** but store the data only in the DOM. This is the case in the chat example without knockout on the course web. Even for this small program it means quite a lot of searching through HTML elements and parsing string.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Without JavaScript framework

Without a JavaScript framework that handles viewmodel-to-DOM bindings, where would we store the nickname and chat conversation? There are three options, each with **serious drawbacks**.

1. **Write the JavaScript code that is now in the framework.** This means quite a lot of quite tricky handling of the DOM, knockout.js contains thousand of lines of JavaScript code.
2. **Do not create a viewmodel,** but store the data only in the DOM. This is the case in the chat example without knockout on the course web. Even for this small program it means quite a lot of searching through HTML elements and parsing string.
3. **Store data only on the server.** This means there is no JavaScript client at all, and we are back to reloading the entire page whenever the user does something.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Section

- The Document Object Model, DOM
- The Browser Object Model, BOM
- The jQuery JavaScript Library
- AJAX
- The Knockout JavaScript Framework
- **Long Polling**
- WebSocket

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Long Polling

- ▶ Long Polling, also called Comet, is a programming technique that enables web servers to **push data to a client** even if the client has not explicitly requested that data.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Long Polling

- ▶ Long Polling, also called Comet, is a programming technique that enables web servers to **push data to a client** even if the client has not explicitly requested that data.
1. The browser makes an **Ajax request** to the server, no matter whether there is data to fetch or not.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Long Polling

- ▶ Long Polling, also called Comet, is a programming technique that enables web servers to **push data to a client** even if the client has not explicitly requested that data.
1. The browser makes an **Ajax request** to the server, no matter whether there is data to fetch or not.
 2. The request is **left unanswered by the server, until there is data** to send to the browser.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Long Polling

- ▶ Long Polling, also called Comet, is a programming technique that enables web servers to **push data to a client** even if the client has not explicitly requested that data.
1. The browser makes an **Ajax request** to the server, no matter whether there is data to fetch or not.
 2. The request is **left unanswered by the server, until there is data** to send to the browser.
 3. When receiving the server response, the browser **immediately makes a new request** in order to obtain the *next* data set.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Long Polling

- ▶ Long Polling, also called Comet, is a programming technique that enables web servers to **push data to a client** even if the client has not explicitly requested that data.

1. The browser makes an **Ajax request** to the server, no matter whether there is data to fetch or not.
2. The request is **left unanswered by the server, until there is data** to send to the browser.
3. When receiving the server response, the browser **immediately makes a new request** in order to obtain the *next* data set.
4. Start over from point one again.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Long Polling JavaScript Client

```
$(document).ready(function () {  
    function fetchFromServer() {  
        $.getJSON("get-msg.php",  
            function (response) {  
                $("#fromServer").  
                    prepend("<p>" + response +  
                        "</p>");  
                fetchFromServer();  
            });  
    }  
  
    fetchFromServer();  
});
```

Long polling is achieved by the call to **fetchFromServer** in red. This way a new call is made as soon a response has been handled.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Long Polling PHP Server

```
1  define("ONE_SEC", 1);
2  define("FILE_PATH", "msg-to-client.txt");
3
4  while (TRUE) {
5      $msg = \file_get_contents(FILE_PATH);
6      if ($msg !== FALSE) {
7          \file_put_contents(FILE_PATH, '');
8          echo \json_encode($msg);
9          return;
10     }
11     \sleep(ONE_SEC);
12 }
```

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

Long Polling PHP Server

```
1  define("ONE_SEC", 1);
2  define("FILE_PATH", "msg-to-client.txt");
3
4  while (TRUE) {
5      $msg = \file_get_contents(FILE_PATH);
6      if ($msg !== FALSE) {
7          \file_put_contents(FILE_PATH, '');
8          echo \json_encode($msg);
9          return;
10     }
11     \sleep(ONE_SEC);
12 }
```

- ▶ Line six checks if the file is empty. The condition `$msg !== FALSE` is required since `$msg` would be false if the file contained for example the string `0`.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Long Polling PHP Server

```
1  define("ONE_SEC", 1);
2  define("FILE_PATH", "msg-to-client.txt");
3
4  while (TRUE) {
5      $msg = \file_get_contents(FILE_PATH);
6      if ($msg !== FALSE) {
7          \file_put_contents(FILE_PATH, '');
8          echo \json_encode($msg);
9          return;
10     }
11     \sleep(ONE_SEC);
12 }
```

- ▶ Line six checks if the file is empty. The condition `$msg !== FALSE` is required since `$msg` would be false if the file contained for example the string `0`.
- ▶ Line seven empties the file.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Long Polling PHP Server

```
1  define("ONE_SEC", 1);
2  define("FILE_PATH", "msg-to-client.txt");
3
4  while (TRUE) {
5      $msg = \file_get_contents(FILE_PATH);
6      if ($msg !== FALSE) {
7          \file_put_contents(FILE_PATH, '');
8          echo \json_encode($msg);
9          return;
10     }
11     \sleep(ONE_SEC);
12 }
```

- ▶ Line six checks if the file is empty. The condition `$msg !== FALSE` is required since `$msg` would be false if the file contained for example the string `0`.
- ▶ Line seven empties the file.
- ▶ Line eleven pauses execution one second.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

A Word of Warning

- ▶ The server will not handle multiple simultaneous requests in the same session. If one request is being handled, **other requests in the same session are blocked.**

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

A Word of Warning

- ▶ The server will not handle multiple simultaneous requests in the same session. If one request is being handled, **other requests in the same session are blocked**.
- ▶ Therefore, if the server has started a session, it is best to **stop the session before calling `sleep`**.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

A Word of Warning

- ▶ The server will not handle multiple simultaneous requests in the same session. If one request is being handled, **other requests in the same session are blocked**.
- ▶ Therefore, if the server has started a session, it is best to **stop the session before calling `sleep`**.

```
\session_write_close();  
\sleep(self::ONE_SEC);  
\session_start();
```

`session_write_close` saves session data and closes the session. This is what happens every time a response has been sent, if a session was started.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

Section

- The Document Object Model, DOM
- The Browser Object Model, BOM
- The jQuery JavaScript Library
- AJAX
- The Knockout JavaScript Framework
- Long Polling
- **WebSocket**

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

WebSocket

- ▶ WebSocket is a W3C specification.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

WebSocket

- ▶ WebSocket is a W3C specification.
- ▶ Specifies a **full-duplex, persistent, TCP connection** between browser and server.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

WebSocket

- ▶ WebSocket is a W3C specification.
- ▶ Specifies a **full-duplex, persistent, TCP connection** between browser and server.
- ▶ Endpoints are identified by URIs.
 - ▶ `ws://host:port/path` (plain websocket connection)
 - ▶ `wss://host:port/path` (encrypted websocket connection)

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

WebSocket

- ▶ WebSocket is a W3C specification.
- ▶ Specifies a **full-duplex, persistent, TCP connection** between browser and server.
- ▶ Endpoints are identified by URIs.
 - ▶ `ws://host:port/path` (plain websocket connection)
 - ▶ `wss://host:port/path` (encrypted websocket connection)
- ▶ Web sockets are message based, endpoints exchange messages (text or binary).

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

WebSocket

- ▶ WebSocket is a W3C specification.
- ▶ Specifies a **full-duplex, persistent, TCP connection** between browser and server.
- ▶ Endpoints are identified by URIs.
 - ▶ `ws://host:port/path` (plain websocket connection)
 - ▶ `wss://host:port/path` (encrypted websocket connection)
- ▶ Web sockets are message based, endpoints exchange messages (text or binary).
- ▶ A **generic transport service**, like TCP. Just like many connection-oriented protocols are built on top of TCP (HTTP, FTP, POP3, IMAP, etc), there are many message-oriented protocols on top of web sockets.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

A New Kind of Web?

- ▶ The W3C specification defines a JavaScript endpoint, and is implemented by all major browsers.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

A New Kind of Web?

- ▶ The W3C specification defines a JavaScript endpoint, and is implemented by all major browsers.
- ▶ There are endpoints for all major server-side technologies, e.g., PHP, Java, .NET, Python, Ruby.

[DOM](#)[BOM](#)[jQuery](#)[AJAX](#)[Knockout](#)[Long Polling](#)[WebSocket](#)

A New Kind of Web?

- ▶ The W3C specification defines a JavaScript endpoint, and is implemented by all major browsers.
- ▶ There are endpoints for all major server-side technologies, e.g., PHP, Java, .NET, Python, Ruby.
- ▶ In conclusion, **any browser or server can have a full-duplex connection with any other browser or server.**

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket

A New Kind of Web?

- ▶ The W3C specification defines a JavaScript endpoint, and is implemented by all major browsers.
- ▶ There are endpoints for all major server-side technologies, e.g., PHP, Java, .NET, Python, Ruby.
- ▶ In conclusion, **any browser or server can have a full-duplex connection with any other browser or server.**
- ▶ The browser is no longer a just a user interface. It becomes a **node in a network**, that can be programmed (in JavaScript) to do anything, and that can communicate (over websockets) with any other node.

DOM

BOM

jQuery

AJAX

Knockout

Long Polling

WebSocket