DOM & jQuery

Role of JavaScript on the Client Side

browser

create/modify/delete ←

data exchange ←

Page

document

Root element:
<html>

DOM
Document Object Model

Element:
<head>

Element:
<title>

Text:
"My title"

Element:
<body>

Element:
<h1>

Text:
"A heading"

Element:
<a>

Attribut:
href

Text:
"Link text"

JavaScript

json

Services @
Server Side
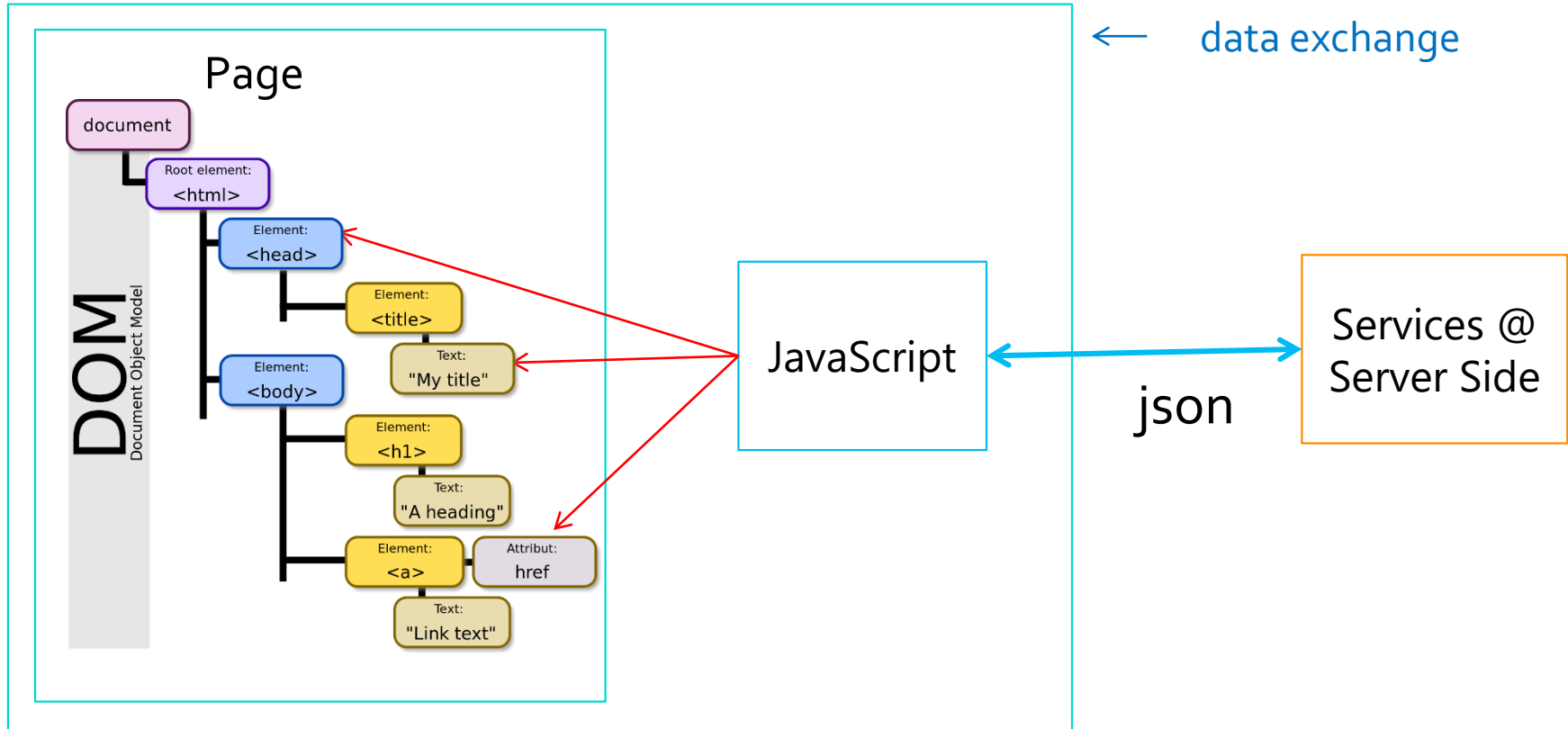
Document Object Model (DOM)

- Every HTML element can be modified via the JavaScript DOM API to:

  o Create interactive pages

  o Updates the objects of a page without reloading it

  o Listen to and handle events when the user does something on the page
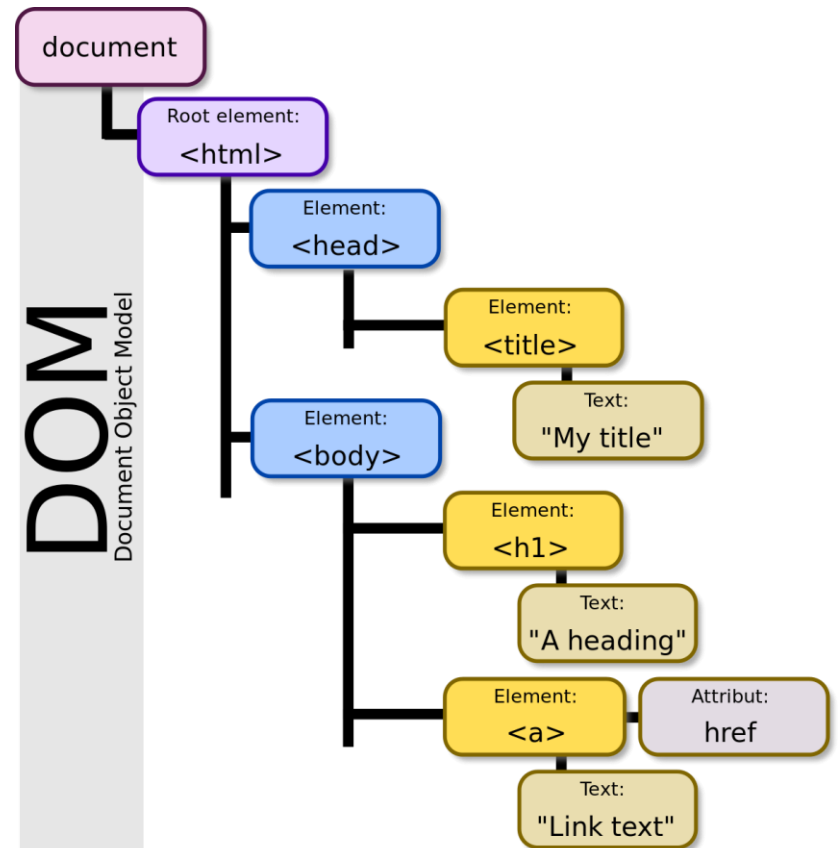
Example DOM Element

HTML

```
<p>
  Look at this octopus:
  <img src="octopus.jpg" alt="an octopus" id="icon01" />
  Cute, huh?
</p>
```

DOM Element Object

Property	Value
tagName	"IMG"
src	"octopus.jpg"
alt	"an octopus"
id	"icon01"

JavaScript

```
var icon = document.getElementById("icon01");
icon.src = "kitty.gif";
```

DOM **querySelector**() Method

- **querySelector**() returns the first element that matches a specified *CSS selector* in the document

- **querySelectorAll**() returns all elements in the document that matches a specified CSS selector

Example CSS selectors:

1. By tag name: document.querySelector("p")
2. By id : document.querySelector("#id")
3. By class: document.querySelector(".classname")
4. By attribute: document.querySelector("a[target]")

- Examples
- https://www.w3schools.com/jsref/met_document_queryselector.asp
- https://www.w3schools.com/jsref/met_document_queryselectorall.asp

5

Accessing Elements – old way!

- Access elements via their ID attribute

```
let element = document.getElementById("some-id")
```

- Via the **name** attribute

```
let elArray = document.getElementsByName("some-name")
```

- Via tag name

```
let imgTags = document.getElementsByTagName("img")
```

  o Returns array of `<img>` elements

DOM Manipulation

- Once we access an element, we can read and write its attributes

```
function change(state) {
  let lampImg = document.querySelector("#lamp")
  lampImg.src = `lamp_${state}.png`
  let statusDiv =
    document.querySelector("#statusDiv")
  statusDiv.innerHTML = `The lamp is ${state}`
}
…
<img src="test_on.gif"  id="lamp"
  onmouseover="change('off')"
  onmouseout="change('on')" />
```

Common Element Properties

- innerHTML – get/set the HTML content of an element

- className – the class attribute of the tag

User Chrome Dev Tool to see the Properties of Page element
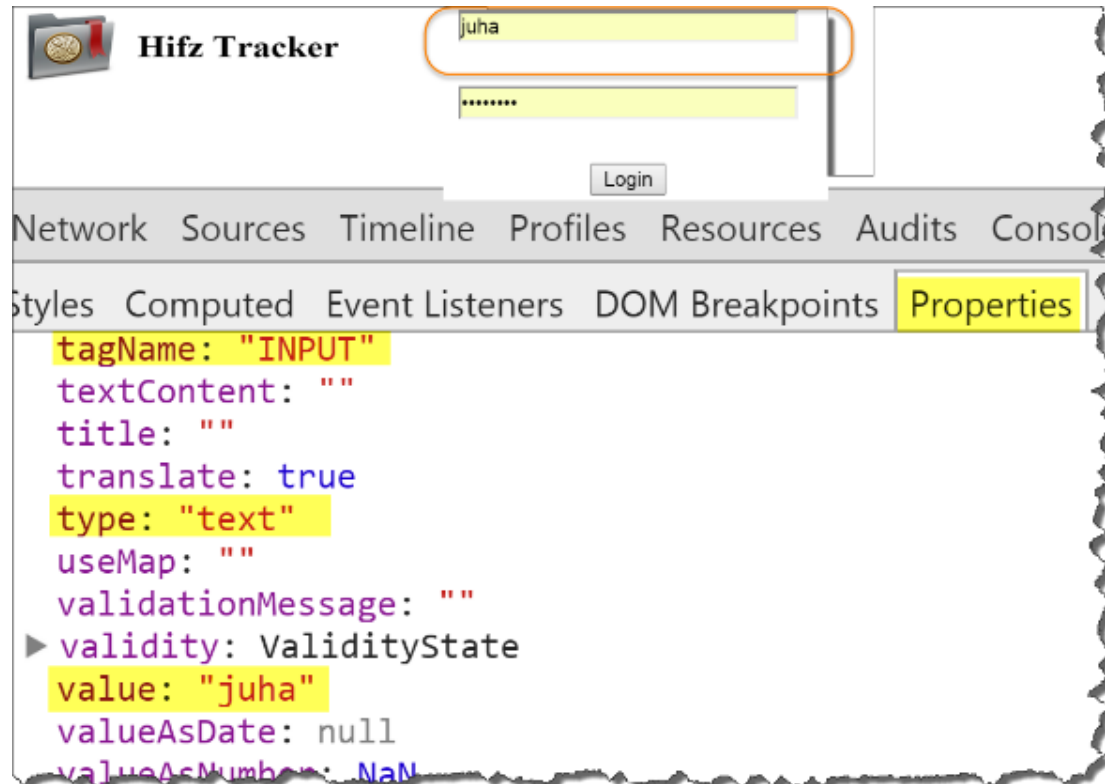
Event Handlers

- JavaScript can register event handlers
  - Events are fired by the Browser and are sent to the specified JavaScript event handler function
  - Can be set with HTML attributes:

```
<img src="test.gif" onclick="imageClicked()" />
```

  - Can be set through the DOM:

```
let img = c("#myImage")
img.addEventListener('click', imageClicked)
```

Common DOM Events

- Mouse events:
  - onclick, onmousedown, onmouseup
  - onmouseover, onmouseout, onmousemove

- Key events:
  - onkeypress, onkeydown, onkeyup
  - Only for input fields

- Interface events:
  - onblur, onfocus, onscroll

- Form events
  - onsubmit: allows you to cancel a form submission if some input fields are invalid

Event Handler

```
<script>
document.querySelector("dateBtn").
    addEventListener("click", displayDate)

function displayDate() {
    document.querySelector("#date").innerHTML =
        Date()
}
</script>
```

Try it @
http://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_displaydate

DOMContentLoaded

- DOMContentLoaded is fired when the DOM tree is built, but external resources like images and stylesheets may be not yet loaded

  o Best event for adding event listeners to page elements

```js
//When the document is loaded in the browser then listen to studentsDD on change event
document.addEventListener("DOMContentLoaded", () => {
    console.log("js-DOM fully loaded and parsed");
    document.querySelector('#studentsDD').addEventListener("change", onStudentChange)
})
```

Commonly used methods

- ## Add Element

```
document.body.appendChild(document.createElement('p'))
```

- ## Add div element and assign it *foo* css class

```
let newDiv = document.createElement('div')
newDiv.classList.add('foo')
```

- ## Navigate elements

```
let parent = document.querySelector('#about').parentNode
let children = document.querySelector('#about').children
```

- ## Hide & Show

```
document.querySelector('myDiv').style.display = 'none';
document.querySelector('myDiv').style.display = '';
```

https://jquery.com/

jQuery

- jQuery is a fast, small and feature-rich JavaScript library that works across a multitude of browsers

- Simplifies HTML document traversing, event handling and animation

- To include jQuery in your website, all you need is a script tag with its *src* pointed to the hosted location

```
<script
src="https://code.jquery.com/jquery-3.1.1.slim.min.js">
</script>
```

jQuery Syntax

- You can use the **$()** function to **select** HTML elements and perform some **action** on the element(s)

- Basic syntax is: **$(** *selector* **).** *action* **()**
  - A **$** sign to access jQuery
  - A (*selector*) to find HTML elements
  - A *action*() to be performed on the element(s)

jQuery Selectors

jQuery supports CSS selectors:

1. By tag name: `$("p")`

2. By id : `$("#id")`

3. By class: `$(".classname")`

4. By attribute: `$("a[href]")`

5. ...

DOM method	jQuery equivalent
`querySelector("#id")`	`$("#id")`
`querySelector("tag")`	`$("tag")`
`querySelector(".classname")`	`$(".classname")`

jQuery Syntax

- Examples:

  - $("p").hide() - hides all <p> elements

  - $(".test").hide() - hides all elements with class="test"

  - $("#test").hide() - hides the element with id="test"

  - $('div').css('background', 'blue') - Make all DIVs blue

.ready() event

- jQuery provides a *ready* event that is fired when the document is ready to be manipulated

- You'll put most of your code in this method

```
$(document).ready( () => {
   // Your code here e.g.,
   alert("Ok document is ready…")
})
```

Creating Elements

- Creating new elements is also easy

```
let divElement = $('<div>')
let paragraph = $('<p>Some text</p>')
```

- Creating complex node

```
$("<p>", {
    "id": "myParagraph",
    "class": "special",
    "text": "My paragraph is awesome!"
})
```

Adding Elements

- Adding elements can be done on the fly

  - **jQuery.appendTo() / jQuery.prependTo()**
  - **jQuery.append() / jQuery.prepend()**

```
<h2>QU Greetings</h2>
<div id="wrapper">
  <div>Hello, student</div>
  <div>Goodbye, student</div>
</div>
```

```
<h2>QU Greetings</h2>
<div id="wrapper">
  <div>Hello, student<p>Test</p></div>
  <div>Goodbye, student<p>Test</p></div>
</div>
```
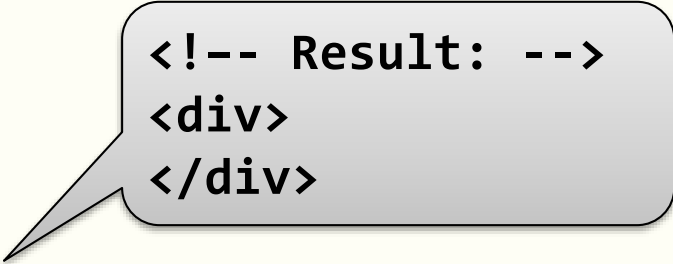
```
$('#wrapper div').append('<p>Test</p>')
```

```
<div>First</div>
<h2>QU Greetings</h2>
<div id="wrapper">
  <div>Hello, studentdiv>
  <div>Goodbye, student<div>
</div>
```

```
$('<div>First</div>').prependTo('body')
```

Removing Elements

- You can also easily remove elements from the DOM

```
<div>
  <p>Red</p>
  <p>Green</p>
</div>

<script>
  $('p').remove() // Remove all paragraphs
</script>
```

<!-- Result: -->
<div>
</div>

jQuery Events

- jQuery has a convenient way for attaching and detaching events Using methods **on()** and **off()**

```
$('#button').on('click', onButtonClick)

function onButtonClick() {
  $(this).hide()
  // "this" is the event source (the button clicked)
}
```

Looping over the DOM

- Using the DOM

```
let elems = document.querySelectorAll("li")
for ( let e of elems )
    // do stuff with e
}
```

- Using jQuery

```
let elems = $("li")
for ( let e of elems )
    // do stuff with e
}
```

jQuery css method parameters

- **Getter:**

  ```
  $("#myid").css(propertyName)
  ```

- **Setter:**

  ```
  $("#myid").css(propertyName, value)
  ```

- **Multi-setter:**

  ```
  $("#myid").css({
          'propertyName1': value1,
          'propertyName2': value2,
          ...
          })
  ```

More node manipulation with jQuery

jQuery method	functionality
.hide()	Hides an element
.show()	Shows an element
.empty()	remove the content of an element, equivalent to innerHTML = ""
.html()	get/set the content of an element in HTML format
.text()	get/set the content of an element as plain text
.val()	get/set the value of a form input, select, textarea, ...

Summary

- jQuery – the most popular client-side JS library

- Select DOM elements with jQuery
  - `$([selector])`

- DOM Traversal:
  - `$([selector]).next() / parent()`

- Altering the DOM:
  - `$([selector]).html(…) / append(…)`

- jQuery Events
  - `$([selector]).on([event], [callback])`

Resources

- W3C School:

  http://www.w3schools.com/jquery/

- Code School:

  http://www.codeschool.com/courses/jquery-air-first-flight