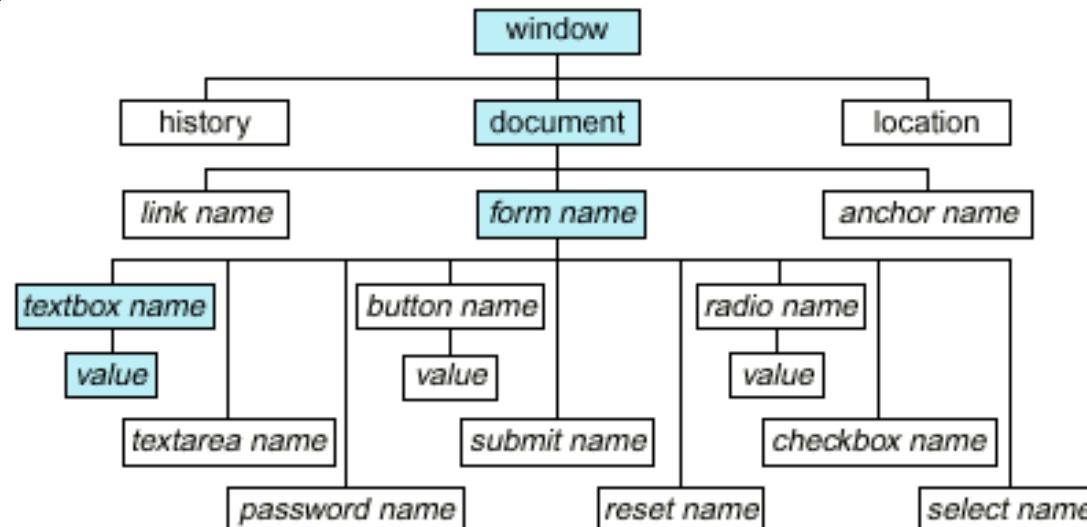


Document Object Model

The True power of dynamic web pages



- ◆ Document Object Model (DOM)
- ◆ The DOM API
- ◆ Selecting DOM elements
 - ◆ `getElementsByXXX()`
 - ◆ `querySelector()`
- ◆ NodeLists
 - ◆ `LiveNodeList`
 - ◆ `StaticNodeList`



The JavaScript Object Model

Document Object Model (DOM)

Document Object Model

- ◆ The Document Object Model is an API for HTML and XML documents
 - ◆ Provides a structural representation of the document
 - ◆ Enables developers to modify the content and visual presentation of a web page

Document Object Model (2)

- ◆ The Document Object Model consists of many objects to manipulate a web page
 - All the properties, methods and events are organized into objects
 - Those objects are accessible through programming languages and scripts
- ◆ How to use the DOM of an HTML page?
 - Write JavaScript to interact with the DOM
 - JavaScript uses the DOM API (native implementation for each browser)

DOM API



- ◆ The DOM API consist of objects and methods to interact with the HTML page
 - ◆ Can add or remove HTML elements
 - ◆ Can apply styles dynamically
 - ◆ Can add and remove HTML attributes
- ◆ DOM introduces objects that represent HTML elements and their properties
 - ◆ `document.documentElement` is <html>
 - ◆ `document.body` is the body of the page

- ◆ Each of the HTML elements have corresponding DOM object types
 - ◆ **HTMLLIElement** represents
 - ◆ **HTMLAudioElement** represents <audio>
- ◆ Each of these objects have the appropriate properties
 - ◆ **HTMLAnchorElement** has **href** property
 - ◆ **HTMLImageElement** has **src** property
- ◆ The document object is a special object
 - ◆ It represents the entry point for the DOM API

- ◆ HTML elements have properties that correspond to the their attributes
 - ◆ id, className, draggable, style, onclick, etc...
- ◆ Different HTML elements have their specific attributes
 - ◆ HTMLImageElement has property src
 - ◆ HTMLInputElement has property value
 - ◆ HTMLAnchorElement has property href

HTML Elements (2)

- ◆ HTML elements also have properties corresponding to their content
 - ◆ **innerHTML**
 - ◆ Returns as a string the content of the element, without the element
 - ◆ **outerHTML**
 - ◆ Returns as a string the content of the element, with the element
 - ◆ **innerText / textContent**
 - ◆ Returns as a string the text content of the element, without the tags

DOM Objects

Live Demo

Selecting DOM Elements

```
<!DOCTYPE html>
▼<html lang="bg" style>
► <head>...</head>
▼<body>
    ▼<div id="Wrapper">
        ► <header id="MainHeader">...</header>
        ► <nav class="kendo-style-black">...</nav>
        ▼<div id="MainContainer">
            ▼<div id="ImportantMessages">
                ► <div class="importantMessageWarning">...</div>
            </div>
            ► <section id="MainContent">...</section>
            </div>
        ► <footer id="MainFooter">...</footer>
        </div>
    </body>
</html>
```

Selecting HTML Elements

- ◆ HTML elements can be found and cached into variables using the DOM API
 - ◆ Select single element

```
var header = document.getElementById("header");
var nav = document.querySelector("#main-nav");
```

- ◆ Select a collection of elements

```
var inputs = document.getElementsByTagName("li");
var radiosGroup = document.getElementsByName("genders[]");
var header = document.querySelectorAll("#main-nav li");
```

- ◆ Using predefined collections of elements

```
var links = document.links;
var forms = document.forms;
```

Using getElementsBy Methods

- ◆ DOM API contains methods for selecting elements based on some characteristic

- ◆ By Id

```
var header = document.getElementById("header");
```

- ◆ By Class

```
var posts = document.getElementsByClassName("post-item");
```

- ◆ By Tag Name

```
var sidebars = document.getElementsByTagName("sidebar");
```

- ◆ By Name

```
var gendersGroup = document.getElementsByName("genders[]");
```

document.getElementsByTagName...

Live Demo

```
var nav = document.getElementById("main-nav");
var navItems = nav.getElementsByClassName("nav-item");
var navButtons = [];
for (var i = 0, len = navItems.length; i < len; i++) {
    navButtons[i] = navItems[i].getElementsByTagName("a")[0];
}
for (var i = 0; i < navButtons.length; i++) {
    navButtons[i].onclick = onNavButtonClick;
}
```

```
<nav id="main-nav">
  <ul>
    <li><a href="#home">Home</a></li>
    <li><a href="#home">News</a></li>
    <li><a href="#home">Careers</a></li>
    <li><a href="#home">Contacts</a></li>
    <li><a href="#home">About us</a></li>
  </ul>
</nav>
```

QuerySelector

- ◆ The DOM API has methods that use CSS-like selectors to find and select HTML elements
 - ◆ `querySelector` returns the first element that matches the selector
 - ◆ `querySelectorAll` returns a collection of all elements that match the selector
 - ◆ Not supported in older browsers (below IE 8)
- ◆ Both methods take as a string parameter
 - ◆ The CSS selector of the element

```
var header = document.querySelector("#header");
//the element with id="header"

var navItems = document.querySelectorAll("#main-nav li");
//li elements contained in element with id=main-nav
```

QuerySelector

Live Demo

Selecting Elements Inside Other Elements

- ◆ The DOM API can be used to select elements that are inside other elements
 - ◆ Select all divs that are inside an element with id "wrapper"

```
var wrapper = document.getElementById("wrapper");
var divsInWrapper = wrapper.getElementsByTagName("div");
```

- ◆ All methods can be used on HTML elements
 - ◆ Except getElementById()

Selecting Inner Elements

Live Demo

NodeLists

- ◆ A NodeList is a collection returned by the DOM selector methods:
 - ◆ **getElementsByTagName(tagName)**
 - ◆ **getElementsByName(name)**
 - ◆ **getElementsByClassName(className)**
 - ◆ **querySelectorAll(selector)**

```
var divs = document.getElementsByTagName("div");
var queryDivs = document.querySelectorAll("div");
for(var i=0; i< divs.length; i++){
    //do stuff with divs[i]...
}
```

- ◆ NodeList looks like an array, but is not
 - ◆ It's an object with properties similar to array
 - ◆ Has length and indexer
 - ◆ Traversing an array with for-in loop works unexpected:

```
for (var i in divs) {  
    console.log("divs[" + i + "] = " + divs[i]);  
}  
//divs[0] = ...  
//divs[1] = ...  
//divs[length] = ...  
//divs[item] = ...
```

NodeList

Live Demo

Static NodeList and Live NodeList

Static and Live NodeLists

- ◆ There are two kinds of NodeLists
 - ◆ `getElementsBy...()` return a `LiveNodeList`
 - ◆ `querySelectorAll()` returns a `StaticNodeList`
- ◆ Live lists keep track of the selected elements
 - ◆ Even when changes are made to the DOM
 - ◆ While static list contain the elements as they were at the execution of the method
- ◆ Keep in mind that `LiveNodeList` is slower than regular array
 - ◆ Need to cache its length for better performance

Static NodeList and Live NodeList

Live Demo

DOM and DOM Manipulation

Questions?

1. Write a script that selects all the div nodes that are directly inside other div elements
 - Create a function using querySelectorAll()
 - Create a function using getElementsByTagName()
2. Create a function that gets the value of <input type="text"> and prints its value to the console
3. Create a function that gets the value of <input type="color"> and sets the background of the body to this value
4. *Write a script that shims querySelector and querySelectorAll in older browsers