# DOM Operations

## Creating dynamic pages

# Table of Contents

* Traversing the DOM
* Parents, Children and Siblings
* DOM manipulation
* Adding, Removing and Altering Elements
* Static and Live NodeLists

# DOM Elements - What is the DOM built from?

# DOM Elements

* А **DOM element** is a JavaScript object that represents an element from the HTML
  + **Selected** using any of the DOM selectors
  + **Created** dynamically from code
* DOM elements can be changed
  + This changes are **immediately** applied to the DOM, and the HTML page

//changes the content of the div

selectedDiv.innerHTML = "changed";

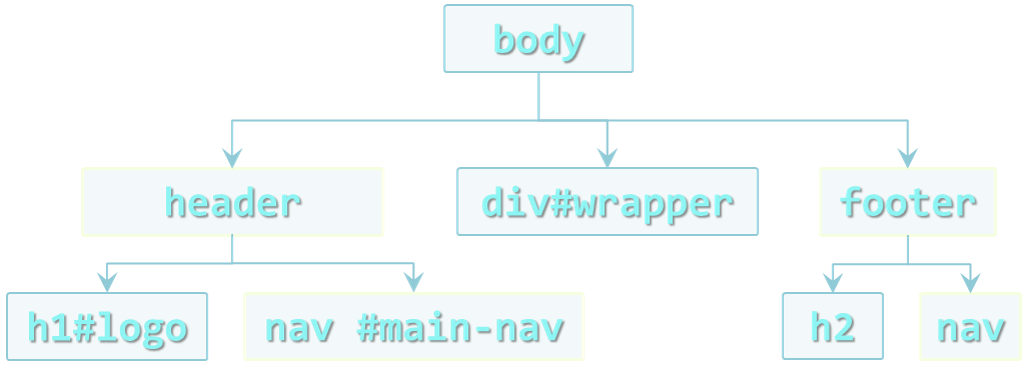
//changes the background of the div to "#456"

selectedDiv.style.background = "#456";

var div = document.createElement("div");

# *DOM Elements - Live Demo*

# Traversing the DOM



# Traversing the DOM

* DOM elements have properties about their position in the DOM:
  + Their parent
  + Their children
  + Their Siblings
    - Elements immediatelly before and after the element
* These properties can be used to traverse through the DOM
* element.parentNode
  + Returns the direct parent of the element
  + The parent of document is null
* element.childNodes
  + Returns a nodeList of all the child nodes
    - Including the text nodes (whitespaces)

# Traversing the DOM - Example

* Traverse a <UL> with <LI>s:

function iterateList(listId){

var trainersList = document.getElementsById(listId);

var parent = trainersList.parentNode;

log("parent of trainers-list: " + parent.nodeName +

" with id: " + parent.id);

var children = trainersList.childNodes;

log("elements in trainers-list: " + children.length);

log("element in trainers-list");

for (var i = 0, len = children.length; i < len; i++) {

var subItem = children[i];

log(subItem.nodeName + " content: " +

subItem.innerText);

}

# *Traversing the DOM - Live Demo*

# Using the Named Elements in DOM Objects

* DOM elements have some properties for special elements around them:
  + **First** and **last** child nodes
  + The element **before**/**after** the current node
* The named elements are:
  + firstChild and lastChild
  + nextSibling / nextElementSibling
  + previousSibling / previousElementSibling

# *Using the Named Elements in DOM Objects - Live Demo*

# Manipulating the DOM - Making a web page dynamic

# Manipulating the DOM

* DOM can be manipulated dynamically with JS
  + HTML elements can be created
  + HTML elements can be removed
  + HTML elements can be altered
    - Change their content
    - Change their styles
    - Change their attributes

# Creating HTML Еlements

* The document object has a method for creation of HTML elements
  + document.createElement(elementName)
  + Returns an object with the corresponding HTML element type

var liElement = document.createElement("li");

console.log(liElement instanceof HTMLLIElement); //true

console.log(liElement instanceof HTMLElement); //true

console.log(liElement instanceof HTMLDivElement); //false

# Creating HTML Еlements

* After an HTML element is created it can be treated as if it was selected from the DOM
* When HTML elements are created dynamically they are just JavaScript objects
  + They are still not in the DOM (the web page)
  + New HTML elements must be appended to DOM

var studentsList = document.createElement("ul");

var studentLi = document.createElement("li");

studentsList.appendChild(studentLi);

document.body.appendChild(studentsList);

# *Appending Elements to the DOM - Live Demo*

# Inserting Elements Before/After Other Element

* The DOM API supports inserting a element before or after a specific element
  + appendChild() inserts the element always at the end of the DOM element
  + parent.insertBefore(newNode, specificElement)

# *Inserting Elements After/Before Other Elements - Live Demo*

# Removing Elements

* Elements can be removed from the DOM
  + Using element.removeChild(elToRemove)
  + Pass the element-to-remove to their parent

var trainers = document.getElementsByTagName("ul")[0];

var trainer = trainers.getElementsByTagName("li")[0];

trainers.removeChild(trainer);

//remove a selected element

var selectedElement = //select the element

selectedElement.parentNode.removeChild(selectedElement);

# *Removing Elements - Live Demo*

# Altering the Elements

* DOM elements can be remove and/or changed
  + Both the node's children and the node itself
* With the DOM API each DOM element node can be altered
  + Change its properties
  + Change its appearance
* Keep in mind that each HTML element is unique in the DOM
  + If JavaScript changes its appearance or its position, it is still the same element object

<div id="f"><p id="the-p">text</p></div>

<div id="s"></div>

…

var second = document.getElementById("s");

var theP = document.getElementById("the-p");

second.appendChild(theP);

…

//the DOM is:

<div id="f"></div>

<div id="s"><p id="the-p">text</p></div>

# *Altering HTML Elements - Live Demo*

# Altering the Style

* The style of each HTML element can be altered using JavaScript
  + Meaning changing the style attribute
    - The inline styles, not CSS

var div = document.getElementById("content");

div.style.display = "block";

div.style.width = "123px";

// Do not forget the unit

# *Altering the Style - Live Demo*

# DOM Optimizations - Everybody likes it fast, right?

# Appending DOM Elements

* The DOM API provides a method for appending DOM elements to a element
  + The appendChild() method
* parentNode.appendChild(node) appends the DOM element node to the DOM element parentNode
  + If parentNode is appended to the DOM, the node is also appended

# Optimizing the Appending of Elements

* Appending elements to the DOM is a **very slow operation**
  + When an elements is appended to the DOM, the DOM is **rendered anew**
  + All newly created elements must be appended together
* Here comes the DocumentFragment element
  + It is a **minimal DOM element**, with no parent
  + It is used to **store ready-to-append** elements and append them at once to the DOM
* Using DocumentFragment
  + Append the elements to a DocumentFragment
  + Appending DocumentFragment to the DOM appends only its child elements
  + <http://jsperf.com/append-doc-fragment/2>

dFrag = document.createDocumentFragment();

dFrag.appendChild(div);

//appending more elements

document.body.appendChild(dFrag);

# *Working with DocumentFragment - Live Demo*

# Faster Creation of Elements

* **Creating a DOM element** is a slow operation
  + Create the element
  + Set its **content**
  + Set its **style**
  + Set its **attributes**
* This is an issue when creating many elements that have a common structure
  + Only **one or two** things are different for all elements
* Creating a **dynamic list** of elements
  + All of the LI elements have the **same classes**, **styles**, **attributes**
  + Only the innerHTML is different
* DomElement.cloneNode(true) can be used
  + Creates a full copy (deep copy) of the element

var clonedNode = someElement.cloneNode(true)

# *Faster Creation of Elements - Live Demo*