INLEDANDE WEBBPROGRAMMERING MED JAVASCRIPT

INTRODUCTION TO WEB PROGRAMING USING JAVASCRIPT

ME152A

L4: 1. HIGHER ORDER FUNCTIONS
2. REGULAR EXPRESSIONS
3. JAVASCRIPT - HTML
4. DOM AND EVENTS

OUTLINE

- What did we learn so far?
- Higher order functions
- Regular expressions
- HTML with JavaScript
- DOM
- Events

WHAT DID WE LEARN SO FAR?

- Objects
- Three different ways of creating objects?
- JavaScript prototype?

HIGHER-ORDER FUNCTIONS

- Functions that operate on other functions, either by taking them as arguments or by returning them, are called *higher-order functions*.

- Hides the details (complexity)
- Enables composition
  - We can focus on the process (not details)

HIGHER-ORDER FUNCTIONS (EXAMPLES)

functions that create new functions.

```
1  function greaterThan(n) {
2    return function(m) { return m > n; };
3  }
4  var greaterThan10 = greaterThan(10);
5  console.log(greaterThan10(11));
6  // → true
```

functions that change other functions.

```
1  function noisy(f) {
2    return function(arg) {
3      console.log("calling with", arg);
4      var val = f(arg);
5      console.log("called with", arg, "- got", val);
6      return val;
7    };
8  }
9  noisy(Boolean)(0);
10 // → calling with 0
11 // → called with 0 - got false
```

JSON

- JavaScript Object Notation (JSON) - pronounced “Jason”
- Widely used as a data storage and communication format on the Web.
- JSON is similar to JavaScript’s way of writing arrays and objects, with a few restrictions.
- All property names have to be surrounded by double quotes, and only simple data expressions are allowed
- —no function calls, variables, or anything that involves actual computation.
- Comments are not allowed in JSON.

```
[
  {"name": "Emma de Milliano", "sex": "f",
   "born": 1876, "died": 1956,
   "father": "Petrus de Milliano",
   "mother": "Sophia van Damme"},
  {"name": "Carolus Haverbeke", "sex": "m",
   "born": 1832, "died": 1905,
   "father": "Carel Haverbeke",
   "mother": "Maria van Brussel"},
  … and so on
]
```

JSON (CONT'D)

- JavaScript gives us two functions
  - `JSON.stringify` - takes a JavaScript value and returns a JSON-encoded string
  - `JSON.parse` takes such a string and converts it to the value it encodes.

```
1 var string = JSON.stringify({name: "X", born: 1980});
2 console.log(string);
3 // → {"name":"X","born":1980}
4 console.log(JSON.parse(string).born);
5 // → 1980
```

HIGHER-ORDER FUNCTION: 'FILTER' EXAMPLE

- *filter* in this example is a *so-called higher-order function.*

```
1  var animals = [
2    { name: 'Rocky',  type: 'dog', age: 11 },
3    { name: 'Ginger',   type: 'cat', age: 13 },
4    { name: 'Lola', type: 'dog', age: 5 },
5    { name: 'Luna',     type: 'dog', age: 12 },
6    { name: 'Rayas',     type: 'cat', age: 6 },
7  ];
8
9  //filter old dogs and log as object
10 var dogs = animals.filter(function(animal) {
11   return animal.age > 10 && animal.type === 'dog';
12 });
13
14
15 console.log (dogs);
```

HIGHER-ORDER FUNCTION: 'MAP' EXAMPLE

- *map* is a higher-order function just like *filter* is

```
1  var animals = [
2    { name: 'Rocky',  type: 'dog', age: 11 },
3    { name: 'Ginger',   type: 'cat', age: 13 },
4    { name: 'Lola', type: 'dog', age: 5 },
5    { name: 'Luna',     type: 'dog', age: 12 },
6    { name: 'Rayas',     type: 'cat', age: 6 },
7  ];
8
9
10
11 //filter old dogs names and map their name
12 var oldDogNames = animals.filter(function(animal) {
13     return animal.age > 10 && animal.type === 'dog';
14   })
15   .map(function(animal) {
16     return animal.name;
17   });
18 console.log (oldDogNames);
```

REGULAR EXPRESSIONS

- A regular expression is a sequence of characters that forms a **search pattern**.
- When you search for data in a text, you can use this search pattern to describe what you are searching for.
- A regular expression can be a single character, or a more complicated pattern.
- Regular expressions can be used to perform all types of **text search** and **text replace** operations.

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

REGULAR EXPRESSIONS

Syntax

```
/pattern/modifiers;
```

Example

```
var pattern = /javascript/i;
```

Example explained:
/javascript/i is a regular expression.
javascript/i is a pattern (to be used in a search).
i is a modifier (modifies the search to be case-insensitive).

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

REGULAR EXPRESSIONS: USING STRING METHODS

- **The search() method** uses an expression to search for a match, and returns the position of the match.

```
1   var str = "Introduction to javascript!";
2   var n = str.search(/javascript/i);
3      console.log (n);
```

- **The replace() method** returns a modified string where the pattern is replaced.

```
1  var str ="Introduction to java";
2  var txt = str.replace(/java/i,"javascript");
3      console.log (txt);
```

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

REGULAR EXPRESSIONS (CONT'D)

Modifiers

Modifiers are used to perform case-insensitive and global searches:

Modifier	Description
i	Perform case-insensitive matching
g	Perform a global match (find all matches rather than stopping after the first match)
m	Perform multiline matching

Brackets

Brackets are used to find a range of characters:

Expression	Description
[abc]	Find any character between the brackets
[^abc]	Find any character NOT between the brackets
[0-9]	Find any digit between the brackets
[^0-9]	Find any digit NOT between the brackets
(x|y)	Find any of the alternatives specified

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

REGULAR EXPRESSIONS (CONT'D)

RegExp Object Methods

Method	Description
compile()	Deprecated in version 1.5. Compiles a regular expression
exec()	Tests for a match in a string. Returns the first match
test()	Tests for a match in a string. Returns true or false
toString()	Returns the string value of the regular expression

There are also

- RegExp Object properties
- Quantifiers
- Metacharacters

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

JAVASCRIPT AND HTML

BROWSER

- A web browser is an application that retrieves, presents and travers information resources on WWW (World Wide Web)
- Without web browsers, there would be no JavaScript.
- Various browser vendors
- The *World Wide Web* (not to be confused with the Internet as a whole) is a set of protocols and formats that allow us to visit web pages in a browser.
- *Hypertext Transfer Protocol* (HTTP), allows computers to request documents over the network
- Each document on the Web is named by a *Uniform Resource Locator* (URL), which looks something like this:

```
 http://eloquentjavascript.net/12_browser.html
|      |                          |                 |
protocol       server                path
```

HTML

- HTML, which stands for *Hypertext Markup Language*, is the document format used for web pages. An HTML document contains text, as well as *tags* that give structure to the text, describing things such as links, paragraphs, and headings.

```
<!doctype html>
<html>
  <head>
    <title>My home page</title>
  </head>
  <body>
    <h1>My home page</h1>
    <p>Hello, this is my home page.</p>
    <p>The book! Read it
      <a href="http://eloquentjavascript.net">here</a>.</p>
  </body>
</html>
```

HTML AND JAVASCRIPT

- the most important HTML tag is `<script>`

```
1 <h1>Testing alert</h1>
2 <script>alert("hello!");</script>
```

```
1 <h1>Testing alert</h1>
2 <script src="code/hello.js"></script>
```

`<button>` tag has an `onclick` attribute

```
1 <button onclick="alert('Boom!');">DO NOT PRESS</button>
```

THE DOCUMENT OBJECT MODEL (DOM)

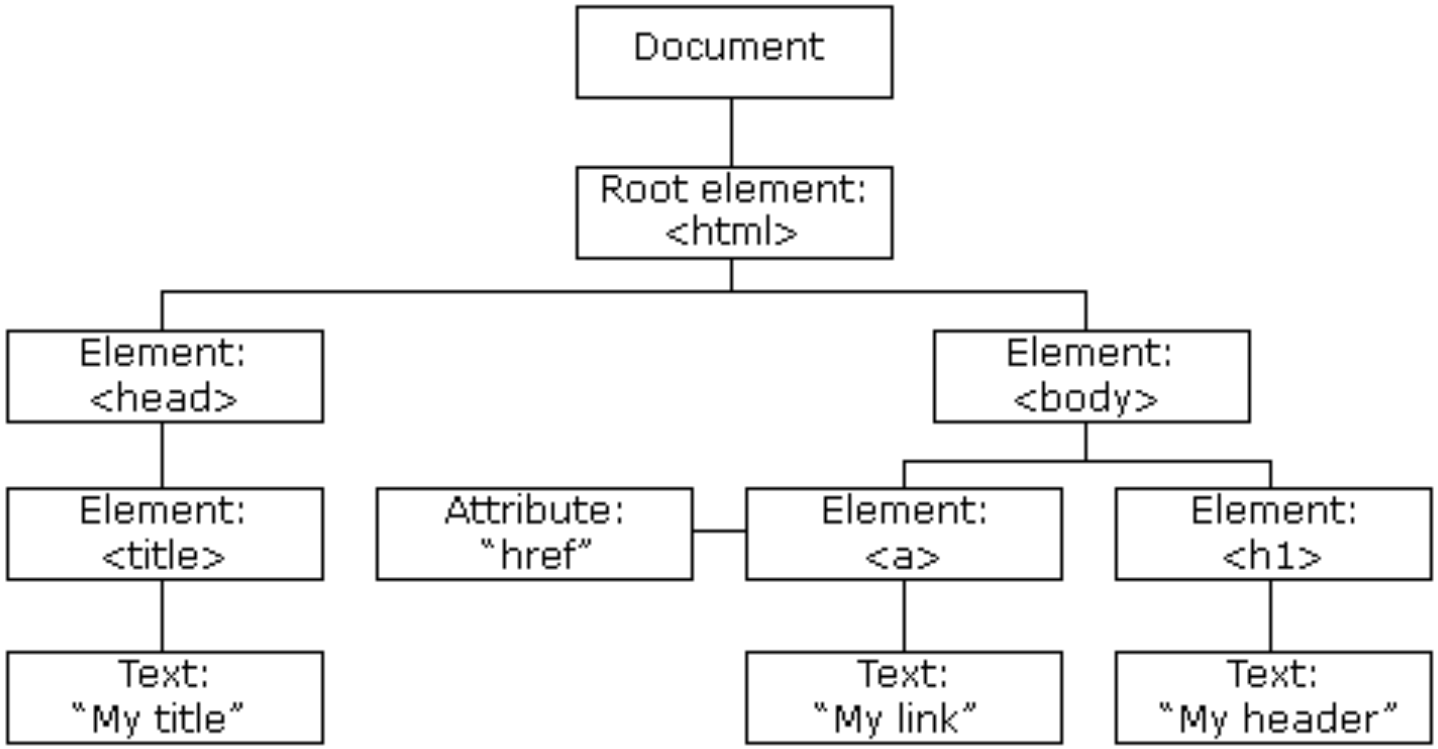
- The DOM is a W3C (World Wide Web Consortium) standard.
- *"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*
- What is the HTML DOM?

  is a standard **object** model and **programming interface** for HTML. It defines:

  The HTML elements as **objects**
  The **properties** of all HTML elements
  The **methods** to access all HTML elements
  The **events** for all HTML elements

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

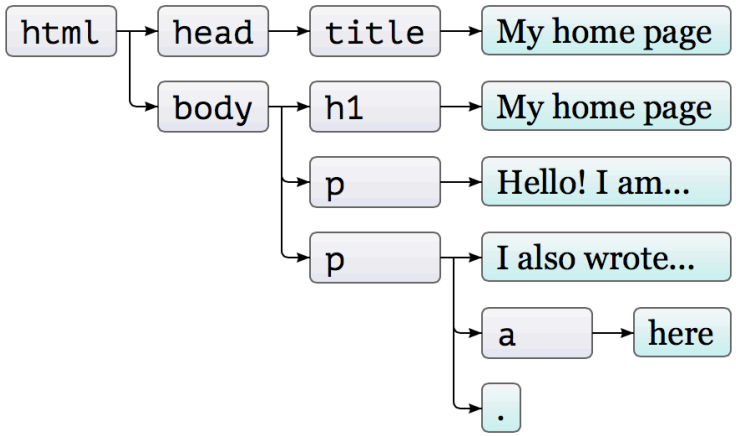
THE HTML DOM TREE OF OBJECTS

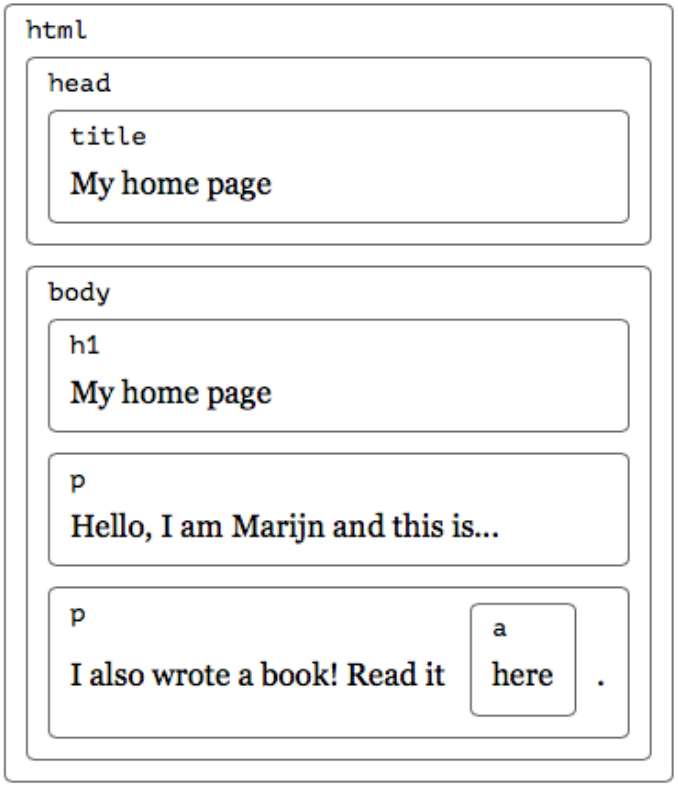


http://www.w3schools.com/js/js_htmldom.asp

DOCUMENT STRUCTURE

```
1  <!doctype html>
2  <html>
3    <head>
4      <title>My home page</title>
5    </head>
6    <body>
7      <h1>My home page</h1>
8      <p>Hello, I am Marijn and this is my home page.</p>
9      <p>I also wrote a book! Read it
10       <a href="http://eloquentjavascript.net">here</a>.</p>
11   </body>
12 </html>
```





HTML DOM METHODS

- The HTML DOM can be accessed with JavaScript (and with other programming languages).
- In the DOM, all HTML elements are defined as **objects**.
- A **property** is a value that you can get or set (like changing the content of an HTML element).
- A **method** is an action you can do (like add or deleting an HTML element).

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

EXAMPLE: “getElementById ”

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4
5  <h1>My First Page</h1>
6
7  <p id="demo"></p>
8
9  <script>
10 document.getElementById("demo").innerHTML = "Hello World!";
11 </script>
12
13 </body>
14 </html>
```

getElementById is a **method**, while innerHTML is a **property**.

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

FINDING HTML ELEMENTS

Method	Description
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

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

CHANGING HTML ELEMENTS

Method	Description
element.innerHTML = *new html content*	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.setAttribute*(attribute, value)*	Change the attribute value of an HTML element
element.style.*property = new style*	Change the style of an HTML element

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

ADDING AND DELETING ELEMENTS

Method	Description
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element
document.replaceChild(*element*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

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

ADDING EVENTS HANDLERS

Method	Description
document.getElementById(*id*).onclick = function(){*code*}	Adding event handler code to an onclick event

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

FINDING HTML OBJECTS

Property	Description	DOM
document.anchors	Returns all <a> elements that have a name attribute	1
document.applets	Returns all <applet> elements (Deprecated in HTML5)	1
document.baseURI	Returns the absolute base URI of the document	3
document.body	Returns the <body> element	1
document.cookie	Returns the document's cookie	1
document.doctype	Returns the document's doctype	3
document.documentElement	Returns the <html> element	3
document.documentMode	Returns the mode used by the browser	3
document.documentURI	Returns the URI of the document	3
document.domain	Returns the domain name of the document server	1
document.domConfig	Obsolete. Returns the DOM configuration	3
document.embeds	Returns all <embed> elements	3
document.forms	Returns all <form> elements	1
document.head	Returns the <head> element	3
document.images	Returns all <img> elements	1
document.implementation	Returns the DOM implementation	3
document.inputEncoding	Returns the document's encoding (character set)	3
document.lastModified	Returns the date and time the document was updated	3
document.links	Returns all <area> and <a> elements that have a href attribute	1
document.readyState	Returns the (loading) status of the document	3
document.referrer	Returns the URI of the referrer (the linking document)	1
document.scripts	Returns all <script> elements	3
document.strictErrorChecking	Returns if error checking is enforced	3
document.title	Returns the <title> element	1
document.URL	Returns the complete URL of the document	1

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

FINDING HTML ELEMENTS

- Finding HTML elements by id

```
var myElement = document.getElementById("intro");
```

- Finding HTML elements by tag name

```
var x = document.getElementsByTagName("p");
```

- Finding HTML elements by class name

```
var x = document.getElementsByClassName("intro");
```

- Finding HTML elements by CSS selectors

```
var x = document.querySelectorAll("p.intro");
```

- Finding HTML elements by HTML object collections

```
var x = document.forms["frm1"];
var text = "";
var i;
for (i = 0; i < x.length; i++) {
    text += x.elements[i].value + "<br>";
}
document.getElementById("demo").innerHTML = text;
```

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

CHANGING HTML CONTENT

```
<html>
<body>

<p id="p1">Hello World!</p>

<script>
document.getElementById("p1").innerHTML
= "New text!";
</script>

</body>
</html>
```

changes the content of a <p> element

```
<!DOCTYPE html>
<html>
<body>

<h1 id="header">Old Header</h1>

<script>
var element =
document.getElementById("header");
element.innerHTML = "New Header";
</script>

</body>
</html>
```

changes the content of an <h1> element

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

CHANGING THE VALUE OF AN ATTRIBUTE

```
<!DOCTYPE html>
<html>
<body>

<img id="myImage" src="smiley.gif">

<script>
document.getElementById("myImage").src = "landscape.jpg";
</script>

</body>
</html>
```

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

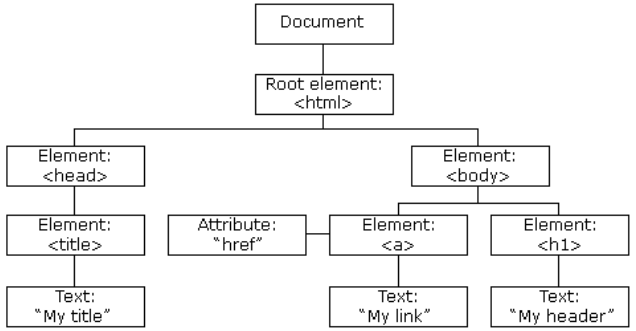
CHANGING THE STYLE

```
<html>
<body>

<p id="p2">Hello World!</p>

<script>
document.getElementById("p2").style.color = "blue";
</script>

<p>The paragraph above was changed by a script.</p>

</body>
</html>
```

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

HTML DOM NAVIGATION

- According to the W3C HTML DOM standard, everything in an HTML document is a node:
  - The entire document is a document node
  - Every HTML element is an element node
  - The text inside HTML elements are text nodes
  - Every HTML attribute is an attribute node
  - All comments are comment nodes



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

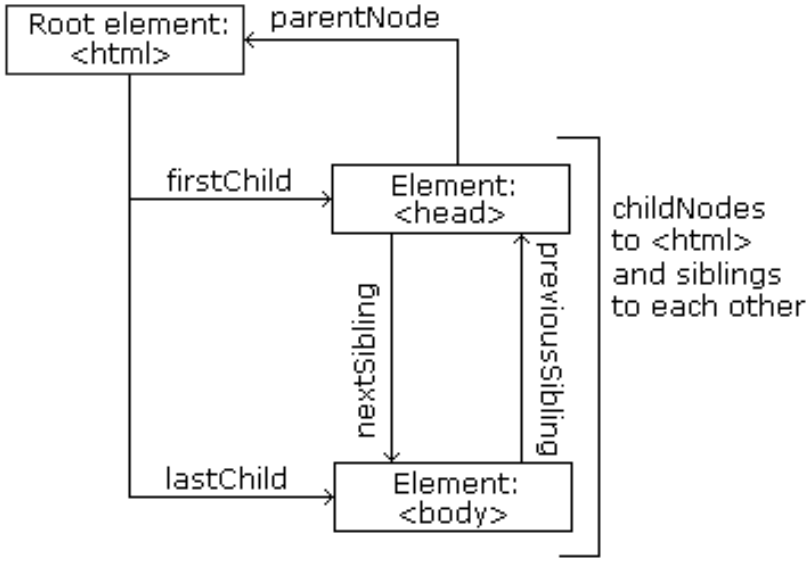
NODE RELATIONSHIPS

- The nodes in the node tree have a hierarchical relationship to each other.
- The terms parent, child, and sibling are used to describe the relationships.
  - In a node tree, the top node is called the root (or root node)
  - Every node has exactly one parent, except the root (which has no parent)
  - A node can have a number of children
  - Siblings (brothers or sisters) are nodes with the same parent

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

NODE RELATIONSHIPS (CONT'D)

```
<html>

  <head>
      <title>DOM Tutorial</title>
  </head>

  <body>
      <h1>DOM Lesson one</h1>
      <p>Hello world!</p>
  </body>

</html>
```



From the HTML above you can read:

- <html> is the root node
- <html> has no parents
- <html> is the parent of <head> and <body>
- <head> is the first child of <html>
- <body> is the last child of <html>

- <head> has one child: <title>
- <title> has one child (a text node): "DOM Tutor
- <body> has two children: <h1> and <p>
- <h1> has one child: "DOM Lesson one"
- <p> has one child: "Hello world!"
- <h1> and <p> are siblings

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

NAVIGATING BETWEEN NODES

- You can use the following node properties to navigate between nodes with JavaScript:
  - parentNode
  - childNodes[*nodenumber*]
  - firstChild
  - lastChild
  - nextSibling
  - previousSibling

Child Nodes and Node Values

```
<html>
<body>

<h1 id="intro">My First Page</h1>

<p id="demo">Hello!</p>

<script>
var myText =
document.getElementById("intro").childNodes[
0].nodeValue;
document.getElementById("demo").innerHTML =
myText;
</script>

</body>
</html>
```

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

DOM ROOT NODES

- There are two special properties that allow access to the full document:
  - document.body - The body of the document
  - document.documentElement - The full document

```
<html>
<body>

<p>Hello World!</p>
<div>
<p>The DOM is very useful!</p>
<p>This example demonstrates the
<b>document.body</b> property.</p>
</div>

<script>
alert(document.body.innerHTML);
</script>

</body>
</html>
```

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

nodeName + nodeValue + nodeType

The **nodeName** property specifies the name of a node.

- nodeName is read-only
- nodeName of an element node is the same as the tag name
- nodeName of an attribute node is the attribute name
- nodeName of a text node is always #text
- nodeName of the document node is always #document

The **nodeValue** property specifies the value of a node.

- nodeValue for element nodes is undefined
- nodeValue for text nodes is the text itself
- nodeValue for attribute nodes is the attribute value

The **nodeType** property returns the type of node. **nodeType** is read only.

The most important **nodeType**'s are:

Element type	NodeType
Element	1
Attribute	2
Text	3
Comment	8
Document	9

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

CREATING NEW HTML ELEMENTS (NODES)

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4
5  <div id="div1">
6  <p id="p1">This is a paragraph.</p>
7  <p id="p2">This is another paragraph.</p>
8  </div>
9
10 <script>
11 var para = document.createElement("p");
12 var node = document.createTextNode("This is new.");
13 para.appendChild(node);
14 var element = document.getElementById("div1");
15 element.appendChild(para);
16 </script>
17
18 </body>
19 </html>
```

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

REMOVING EXISTING HTML ELEMENTS

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4
5  <div id="div1">
6  <p id="p1">This is a paragraph.</p>
7  <p id="p2">This is another paragraph.</p>
8  </div>
9
10 <script>
11 var parent = document.getElementById("div1");
12 var child = document.getElementById("p1");
13 parent.removeChild(child);
14 </script>
15
16 </body>
17 </html>
```

To replace an element to the HTML DOM, use the replaceChild() method

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

HTML DOM NODE LIST

- The getElementsByTagName() method returns a **node list**. A node list is an array-like collection of nodes.

The following code selects all <p> nodes in a document:

```
<!DOCTYPE html>
<html>
<body>

<p>Hello World!</p>

<p>The DOM is very useful!</p>

<p id="demo"></p>

<script>
var myNodelist = document.getElementsByTagName("p");
document.getElementById("demo").innerHTML =
"The innerHTML of the second paragraph is: " +
myNodelist[1].innerHTML;
</script>

</body>
</html>
```

The length property defines the number of nodes in a node list:

```
<!DOCTYPE html>
<html>
<body>

<p>Hello World!</p>

<p>How many paragraphs in this document?</p>

<p>This example demonstrates the length property of a nodelist.</p>

<p id="demo"></p>

<script>
var myNodelist = document.getElementsByTagName("p");
document.getElementById("demo").innerHTML = myNodelist.length;
</script>

</body>
</html>
```

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

USING EVENTS

- The HTML DOM allows you to execute code when an event occurs.
- Events are generated by the browser when "things happen" to HTML elements:
  - When a user clicks the mouse
  - When a web page has loaded
  - When an image has been loaded
  - When the mouse moves over an element
  - When an input field is changed
  - When an HTML form is submitted
  - When a user strokes a key

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

USING EVENTS: changing style by onclick

```
<!DOCTYPE html>
<html>
<body>

<h1 id="id1">My Heading 1</h1>

<button type="button"
onclick="document.getElementById('id1').style.color = 'red'">
Click Me!</button>

</body>
</html>
```

USING EVENTS: a function is called from the event handler

```
<!DOCTYPE html>
<html>
<body>

<h1 onclick="changeText(this)">Click on this text!</h1>

<script>
function changeText(id) {
    id.innerHTML = "Ooops!";
}
</script>

</body>
</html>
```

HTML EVENT ATTRIBUTES

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4
5  <p>Click the button to display the date.</p>
6
7  <button onclick="displayDate()">The time is?</button>
8
9  <script>
10 function displayDate() {
11     document.getElementById("demo").innerHTML = Date();
12 }
13 </script>
14
15 <p id="demo"></p>
16
17 </body>
18 </html>
```

THE onchange EVENT

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
    var x = document.getElementById("fname");
    x.value = x.value.toUpperCase();
}
</script>
</head>
<body>

Enter your name: <input type="text" id="fname" onchange="myFunction()">
<p>When you leave the input field, a function is triggered which transforms the input text to upper case.</p>

</body>
</html>
```

THE onmouseover AND onmouseout EVENTS

```
<!DOCTYPE html>
<html>
<body>

<div onmouseover="mOver(this)" onmouseout="mOut(this)"
style="background-color:#D94A38;width:120px;height:20px;padding:40px;">
Mouse Over Me</div>

<script>
function mOver(obj) {
    obj.innerHTML = "Thank You"
}

function mOut(obj) {
    obj.innerHTML = "Mouse Over Me"
}
</script>

</body>
</html>
```

THE onmousedown, onmouseup AND onclick EVENTS

```
<!DOCTYPE html>
<html>
<body>

<div onmousedown="mDown(this)" onmouseup="mUp(this)"
style="background-color:#D94A38;width:90px;height:20px;padding:40px;">
Click Me</div>

<script>
function mDown(obj) {
    obj.style.backgroundColor = "#1ec5e5";
    obj.innerHTML = "Release Me";
}

function mUp(obj) {
    obj.style.backgroundColor="#D94A38";
    obj.innerHTML="Thank You";
}
</script>

</body>
</html>
```

HANDLING EVENTS: addEventListener

- The addEventListener() method attaches an event handler to the specified element.
- The addEventListener() method attaches an event handler to an element without overwriting existing event handlers.
- You can add many event handlers to one element.
- You can add many event handlers of the same type to one element, i.e two "click" events.
- You can add event listeners to any DOM object not only HTML elements. i.e the window object.
- The addEventListener() method makes it easier to control how the event reacts to bubbling.
- When using the addEventListener() method, the JavaScript is separated from the HTML markup, for better readability and allows you to add event listeners even when you do not control the HTML markup.
- You can easily remove an event listener by using the removeEventListener() method.

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

addEventListener: SYNTAX

```
element.addEventListener(event, function, useCapture);
```

- The first parameter is the type of the event (like "click" or "mousedown").
- The second parameter is the function we want to call when the event occurs.
- The third parameter is a boolean value specifying whether to use event bubbling or event capturing. This parameter is optional.

```
element.addEventListener("click", function(){ alert("Hello World!"); });
```

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

addEventListener: ADD MANY EVENT HANDLERS TO THE SAME ELEMENT

```
<!DOCTYPE html>
<html>
<body>

<p>This example uses the addEventListener() method to add many events on the same button.</p>

<button id="myBtn">Try it</button>

<p id="demo"></p>

<script>
var x = document.getElementById("myBtn");
x.addEventListener("mouseover", myFunction);
x.addEventListener("click", mySecondFunction);
x.addEventListener("mouseout", myThirdFunction);

function myFunction() {
    document.getElementById("demo").innerHTML += "Moused over!<br>";
}

function mySecondFunction() {
    document.getElementById("demo").innerHTML += "Clicked!<br>";
}

function myThirdFunction() {
    document.getElementById("demo").innerHTML += "Moused out!<br>";
}
</script>

</body>
</html>
```

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

addEventListener: KEY PRESS

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4
5  <p>This page turns violet when you hold the V key.</p>
6
7  <script>
8    addEventListener("keydown", function(event) {
9      if (event.keyCode == 86)
10       document.body.style.background = "violet";
11   });
12   addEventListener("keyup", function(event) {
13     if (event.keyCode == 86)
14       document.body.style.background = "";
15   });
16 </script>
17
18 </body>
19 </html>
```

addEventListener: DEBOUNCING

```
<textarea>Type something here...</textarea>
<script>
  var textarea = document.querySelector("textarea");
  var timeout;
  textarea.addEventListener("keydown", function() {
    clearTimeout(timeout);
    timeout = setTimeout(function() {
      console.log("You stopped typing.");
    }, 500);
  });
</script>
```

```
<script>
  function displayCoords(event) {
    document.body.textContent =
      "Mouse at " + event.pageX + ", " + event.pageY;
  }

  var scheduled = false, lastEvent;
  addEventListener("mousemove", function(event) {
    lastEvent = event;
    if (!scheduled) {
      scheduled = true;
      setTimeout(function() {
        scheduled = false;
        displayCoords(lastEvent);
      }, 250);
    }
  });
</script>
```

REFLECTION

- Higher order functions?
- Regular expressions?
- DOM?
- EVENTS?

THANK YOU

QUESTIONS?

Literature:

Haverbeke, M. (2014). *Eloquent JavaScript: A Modern Introduction to Programming*. No Starch Press.