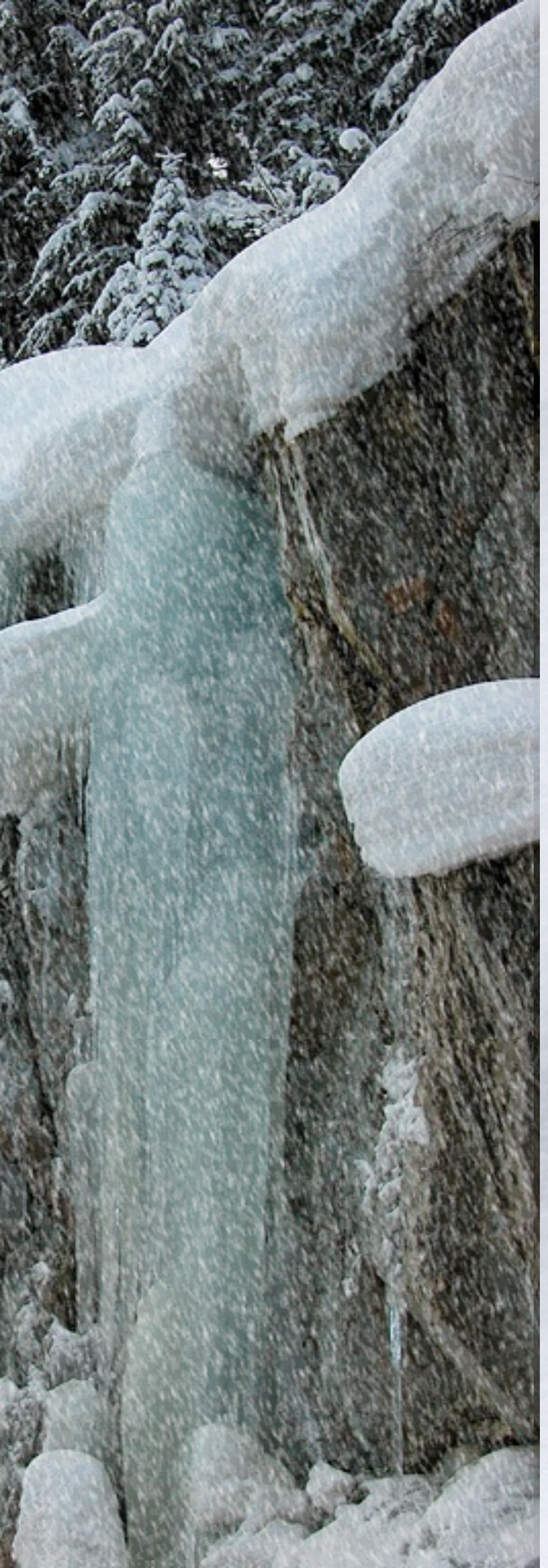
Lecture 6
Introduction to JavaScript

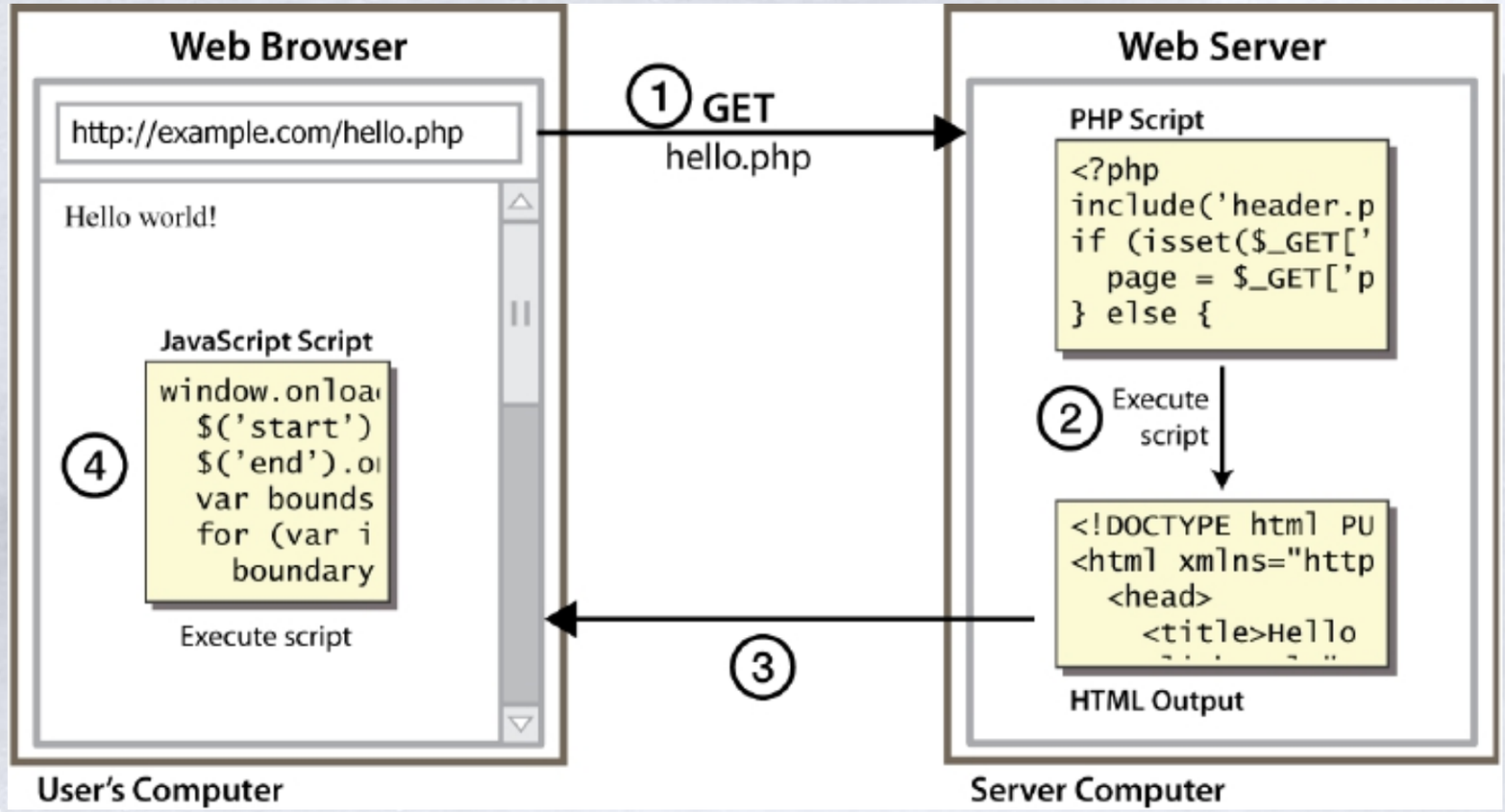


Outline

- *Client Side Basics*
- Overview of JavaScript
- JavaScript Basic Syntax
- The DOM tree
- The Browser Object Model (BOM)

Client-side scripting

- client-side script: code runs in browser after page is sent back from server
  - often this code manipulates the page or responds to user actions



Client-Side Programming

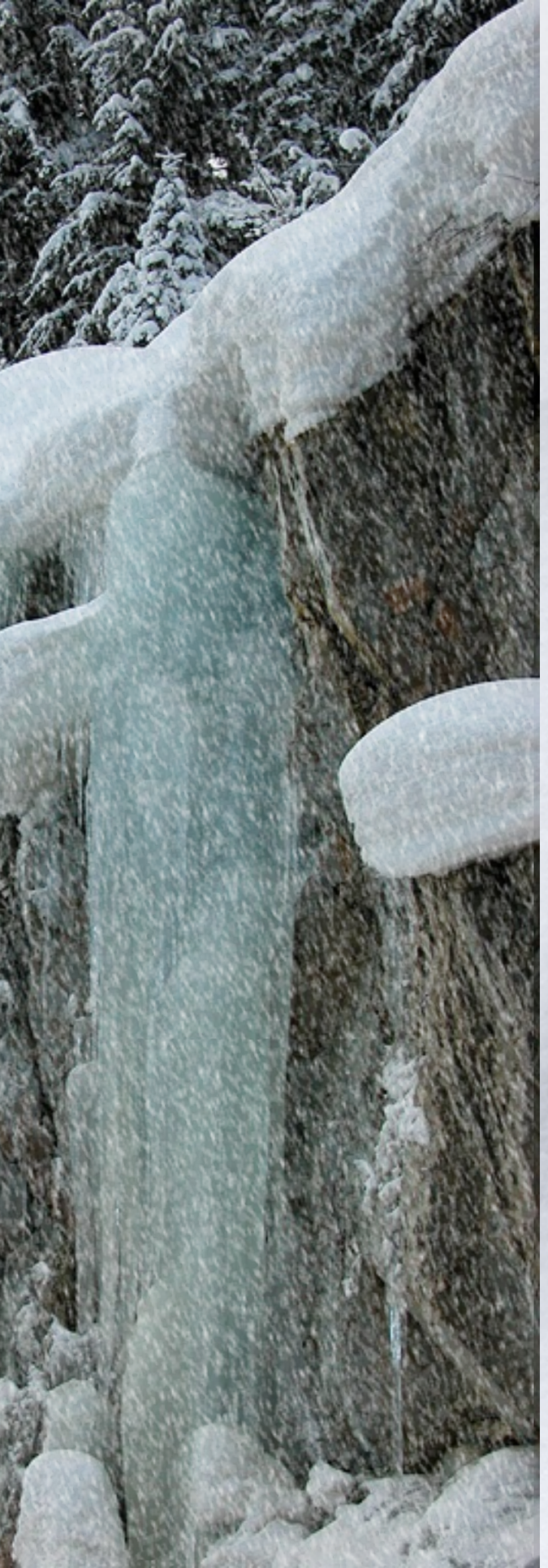
- HTML is good for developing static pages
  - can specify text/image layout, presentation, links, ...
  - Web page looks the same each time it is accessed
  - in order to develop interactive/reactive pages, must integrate programming in some form or another
- client-side programming
  - programs are written in a separate programming (or scripting) language
    - e.g., JavaScript, JScript, VBScript
  - programs are embedded in the HTML of a Web page, with (HTML) tags to identify the program component
    - e.g., <script type="text/javascript"> ... </script>
  - the browser executes the program as it loads the page, integrating the dynamic output of the program with the static content of HTML
  - could also allow the user (client) to input information and process it, might be used to validate input before it's submitted to a remote server

client-side vs. server-side programming

- Client-side scripting (JavaScript) benefits:
  - usability: can modify a page without having to post back to the server (faster UI)
  - efficiency: can make small, quick changes to page without waiting for server
  - event-driven: can respond to user actions like clicks and key presses source code
  - platform-independence: code interpreted by any script-enabled browser
- Server-side programming (PHP) benefits:
  - security: has access to server's private data; client can't see
  - compatibility: not subject to browser compatibility issues
  - power: can write files, open connections to servers, connect to databases, ...

Common Scripting Tasks

- adding dynamic features to Web pages
  - validation of form data (probably the most commonly used application)
  - image rollovers
  - time-sensitive or random page elements
  - handling cookies
- defining programs with Web interfaces
  - utilize buttons, text boxes, clickable images, prompts, etc
- limitations of client-side scripting
  - since script code is embedded in the page, it is viewable to the world
  - for security reasons, scripts are limited in what they can do
    - e.g., can't access the client's hard drive
  - since they are designed to run on any machine platform, scripts do not contain platform specific commands
  - script languages are not full-featured
    - e.g., JavaScript objects are very crude, not good for large project development



Outline

- Client Side Basics
- *Overview of JavaScript*
- JavaScript Basic Syntax
- The DOM tree
- The Browser Object Model (BOM)

Why talk about JavaScript?

- Once thought of as a toy, JavaScript is now the most widely deployed programming language in history.
  - Web pages, AJAX, Web 2.0 / 3.0
  - Increasing number of web-related applications
  - Many operating systems have adopted the open web standards as the presentation layer for native apps, including Windows 8, Firefox OS, Gnome, and Google's Chrome OS.
  - iPhone and Android mobile devices support web views that allow them to incorporate JavaScript and HTML5 functionality into native applications.
  - Moving into the hardware world. Projects like Arduino, Tessel, Espruino, and NodeBots foreshadow a time in the near future where JavaScript could be a common language for embedded systems and robotics.

Advantages of JavaScript

- Performance
  - JIT
  - the overhead of garbage collection and dynamic binding
- Objects
  - JavaScript uses a prototypal inheritance model.
- Syntax
  - familiar to anybody who has experience with C-family languages, such as C++, Java, C#, and PHP.
- First-Class Functions
  - Nearly everything in JavaScript is an object, including functions.
- Events
  - Inside the browser, everything runs in an event loop.
- Reusability
  - the most portable, reusable code
  - JavaScript can be modular and encapsulated

History

- Javascript Developed by Brendan Eich at Netscape
- JScript created by Microsoft
- Standardized by European Computer Manufacturers Association (ECMA)
- http://www.ecma-international. org/ publications /standards/Ecma-262.htm
- ECMAScript 2015
- ECMAScript 2016

Essential of JavaScript

- JavaScript is a script language
- JavaScript programs are evaluated and executed by JavaScript interpreters / engines
  - Rhino, SpiderMonkey, V8, Squirrelfish
- The mainstream purpose and usage: Exposing objects of an application at runtime, for customizing/embedding user logics.
  - Improve the user interface of a website
  - Make your site easier to navigate
  - Replace images on a page without reload the page
  - Form validation
  - Page embellishments and special effects
  - Dynamic content manipulation
  - Emerging Web 2.0: client functionality implemented at client
  - Many others ...

JavaScript vs. Java

- interpreted, not compiled
- more relaxed syntax and rules
  - fewer and "looser" data types
  - variables DON'T need to be declared
  - errors often silent (few exceptions)

- key construct is the function rather than the class
  - "first-class" functions are used in many situations
- contained within a web page and integrates with its HTML/CSS content
  - comparability: browsers may behave differently upon a JavaScript program
    - different dialects/implementations of the standard (ECMAScript)
    - different objects exposed

JavaScript vs. PHP

- similarities:
  - both are interpreted, not compiled
  - both are relaxed about syntax, rules, and types
  - both are case-sensitive
  - both have built-in regular expressions for powerful text processing
- differences:
  - JS is more object-oriented: noun.verb(), less procedural: verb(noun)
  - JS focuses on user interfaces and interacting with a document;PHP is geared toward HTML output and file/form processing
  - JS code runs on the client's browser; PHP code runs on the web server

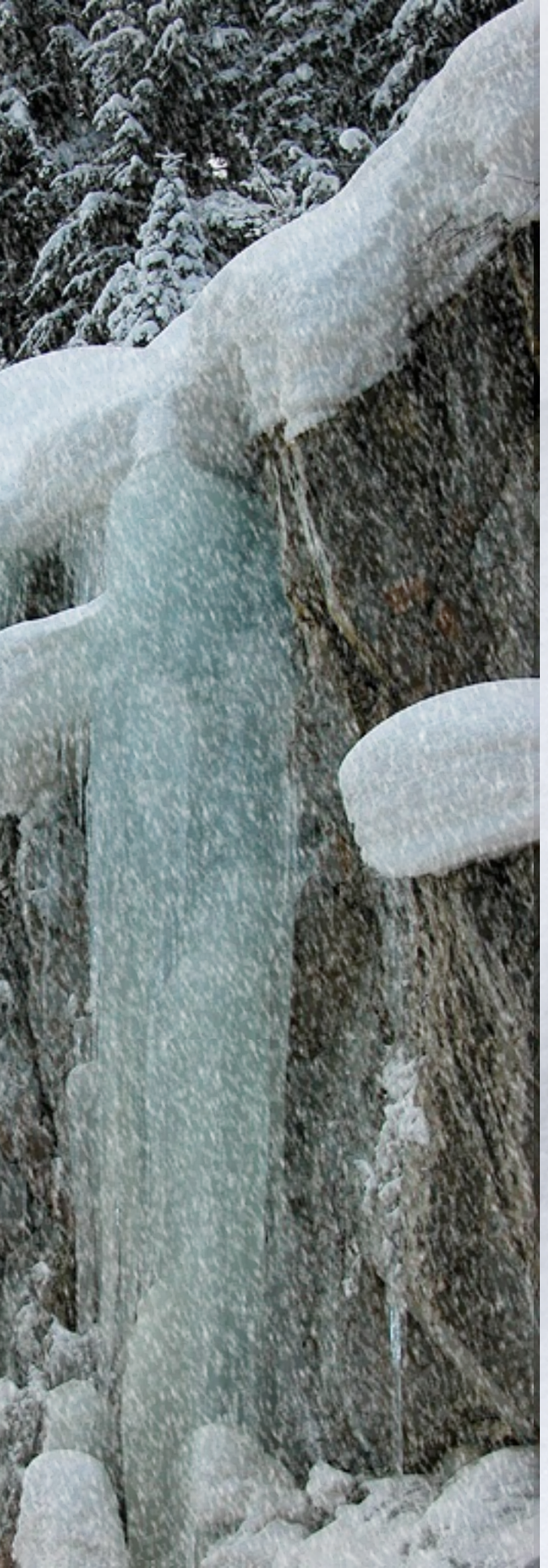
Using the <script> Tag

- To embed a client-side script in a Web page, use the element:

```
<script>
    script commands and comments
</script>
```

- To access an external script, use:

```
<script src="url">
    script commands and comments
</script>
```



Outline

- Client Side Basics
- Overview of JavaScript
- *JavaScript Basic Syntax*
- The DOM tree
- The Browser Object Model (BOM)

Language basics

- JavaScript is case sensitive
  - HTML is not case sensitive; onClick, ONCLICK, ... are HTML
- Statements terminated by returns or semi-colons (;)
  - x = x+1; same as x = x+1
  - Semi-colons can be a good idea, to reduce errors
- "Blocks"
  - Group statements using { ... }
  - Not a separate scope, unlike other languages

Comments (same as Java)

- identical to Java's comment syntax
  - /* multi-line comment */
  - // single-line comment

Objects

- An object is a collection of named properties
  - Simple view: hash table or associative array
  - Can define by set of name:value pairs
    - objBob = {name: “Bob", grade: 'A', level: 3};
  - New members can be added at any time
    - objBob.fullname = 'Robert';
  - Can have methods, can refer to this
- Arrays, functions regarded as objects
  - A property of an object may be a function (=method)

Variables and types

- Variables
  - var, let, const(case sensitive)
  - Define implicitly by its first use, which must be an assignment
    - Implicit definition has global scope, even if it occurs in nested scope?
- types are not specified, but JS does have types ("loosely typed")
  - Number, Boolean, String, Array, Object, Function, Null, Undefined
  - can find out a variable's type by calling typeof

```
var name="Gates", age=60, job="CEO";
var carname="Benz";
var carname;
```

Number type

- integers and real numbers are the same type (no int vs. double)
- same operators:+ - * / % ++ -- etc
- similar precedence to Java
- many operators auto-convert types:
  - "2" * 3 is 6

```
var x=30;
var y=4.2;
var car_count=10;
```

String type

- methods: charAt, charCodeAt,fromCharCode, indexOf, lastIndexOf, replace, split, substring, toLowerCase, toUpperCase
  - charAt returns a one-letter String (there is no char type)
- length property (not a method as in Java)
- Strings can be specified with "" or ''
- concatenation with + :
  - 1+1 is 2, but "1"+1 is "11"

```
var txt="Hello world!";
document.write(txt.length)
```

More about String

- escape sequences behave as in Java: \' \" \& \n \t \\
- converting between numbers and Strings:

```
var x=30;
var s1=“”+x;                    //“30”
var n1=parseInt(“10 oranges”);   //10
var n2=parseFloat(“hello”);      //NaN
```

- accessing the letters of a String:

```
var firstLetter = s[0];                    //fails in IE
var firstLetter = s.charAt(0);
var last Letter = s.charAt(s.length - 1);
```

Old:

```
var message = "The user "+ user.firstName + "" + user.lastName +
" cannot be "+ action + "because "+ validationError;
```

New:

```
var message = `The user ${user.firstName} ${user.lastName} cannot be
  ${action} because ${validationError}` ;
```

Boolean type

- any value can be used as a Boolean
  - false: 0, -0, 0.0, NaN, "", null, and undefined
  - true: anything else
- converting a value into a Boolean explicitly:
  - var boolValue = Boolean(otherValue);
  - var boolValue = !!(otherValue);

```
if (o !== null) ….
```

Special values: null, NaN, undefined

- NaN
  - isNaN()
- undefined
- null
- (-)Infinity

```
var x = 1000 / 0;
isNaN(x);
```

Math object

- methods: abs, ceil, cos, floor, log, max, min, pow, random, round, sin, sqrt, tan
- properties: E, PI

```
var pi_value=Math.PI;
var sqrt_value=Math.sqrt(30);
```

Logical operators

- > < >= <= && || ! == != === !==
- most logical operators automatically convert types:
  - 5<"7" is true
  - 42 == 42.0 is true
  - "5.0" == 5 is true
- === and !== are strict equality tests; checks both type and value
  - "5.0" === 5 is false

if/else statement

- identical structure to Java's if/else statement
- JavaScript allows almost anything as a condition

```
if (condition){
      statements;
} else if (condition) {
    statements;
    } else {
    statements;
    }
```

for loop

- for
- for-in
- for-of

```
var x;
var txt="";
var person={fname:"Bill",lname:"Gates",age:56};
for (x in person){
  txt=txt + person[x];
}
```

while, do/while

- same as Java
- break and continue keywords also behave as in Java

```
cars=["BMW","Volvo","Saab","Ford"];
var i=0;
while (cars[i]){
  document.write(cars[i] + "<br>");
  i++;
}
```

Arrays

- two ways to initialize an array
- length property (grows as needed when elements are added)

```
var empty=[];
var mycars=new Array();
mycars[0]="Saab"
mycars[1]="Volvo"
mycars[2]=“BMW"

var misc=new Array(1.1,true,"BMW");
```

Array methods

- array serves as many data structures: list, queue, stack, ...
- methods: concat, join, pop, push, reverse, shift, slice, sort, splice, toString, unshift
  - push and pop add / remove from back
  - unshift and shift add / remove from front
  - shift and pop return the element that is removed

```
var fruits = ["Banana", "Orange"];
fruits.push("Apple");
fruits.unshift("Mango");
fruits.pop();
fruits.shift();
fruits.sort();
alert(fruits);
```

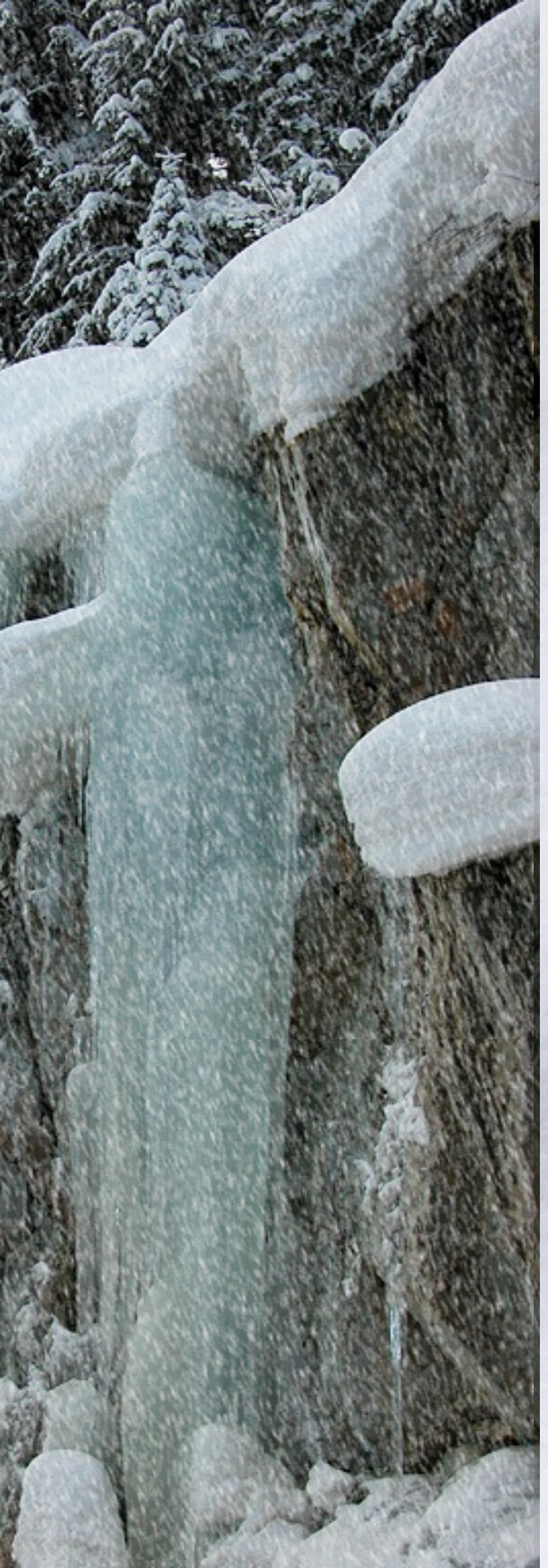
functions

```
function functionName(parameters) {
  //
}
```

- Declarations can appear in function body
  - Local variables, “inner” functions
  - =>
    - (arg1, arg2) => {//something here}
- “Anonymous” functions (expressions for functions)
  - (function (x,y) {return x+y}) (2,3);

Arguments

- Parameter passing
  - Basic types passed by value, objects by reference
- Call can supply any number of arguments
  - functionname.length : # of arguments in definition
  - functionname.arguments.length : # args in call

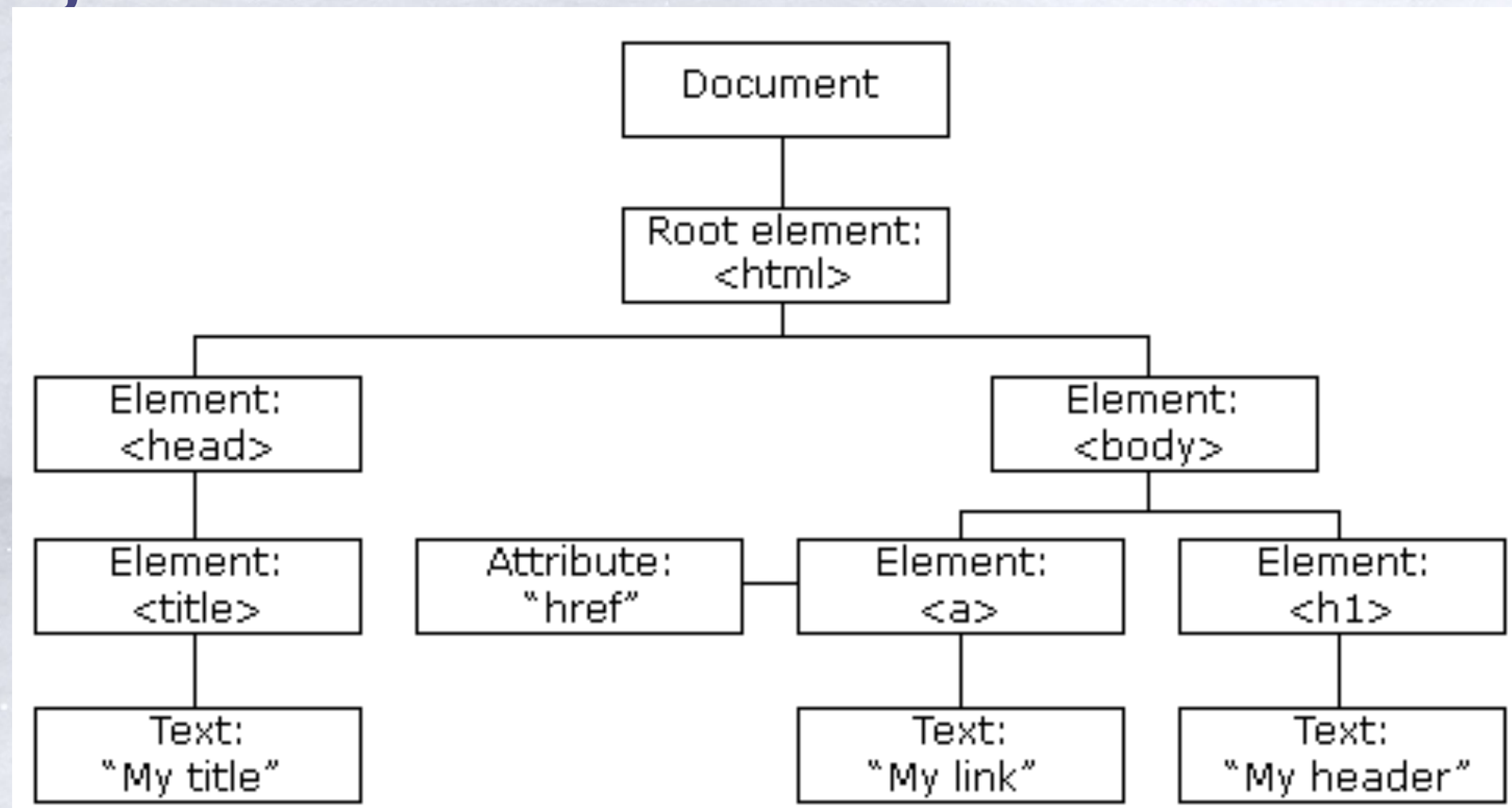


Outline

- Client Side Basics
- Overview of JavaScript
- JavaScript Basic Syntax
- *DOM tree*
- The Browser Object Model (BOM)

Document Object Model (DOM)

- When a web page is loaded, the browser creates a Document Object Model of the page.
- The HTML DOM model is constructed as a tree of Objects:



With the object model, JavaScript gets all the power it needs to create dynamic HTML

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

What is the DOM?

- The DOM is a W3C (World Wide Web Consortium) standard.
- The DOM defines a standard for accessing documents:
  - "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."
- The W3C DOM standard is separated into 3 different parts:
  - Core DOM - standard model for all document types
  - XML DOM - standard model for XML documents
  - HTML DOM - standard model for HTML documents

What is the HTML DOM?

- 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

- In other words: The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

The DOM Programming Interface

- The HTML DOM can be accessed with JavaScript (and with other programming languages).
- In the DOM, all HTML elements are defined as objects.
- The programming interface is the properties and methods of each object.
- 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).

DOM element

- every element on the page has a corresponding DOM object
- access/modify the attributes of the DOM object with objectName.attributeName
- in fact, browsers evaluate a Web page into corresponding DOM objects at runtime

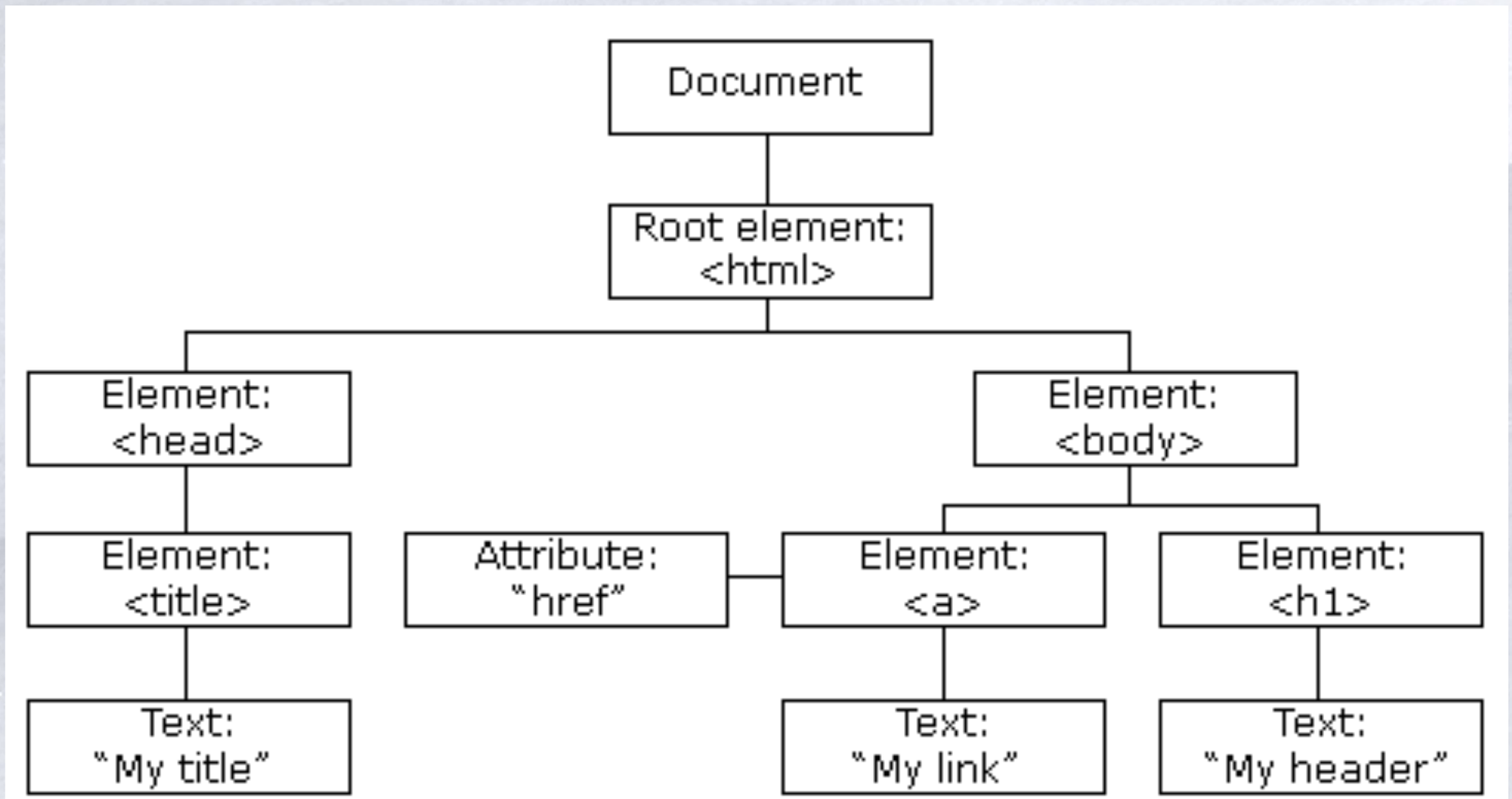
```
<script>
   document.getElementById("demo").innerHTML = "Hello World!";
</script>
```

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"];

JavaScript HTML DOM Navigation

- The elements of a page are nested into a tree-like structure of objects – DOM tree
  - the DOM has properties and methods for traversing this tree

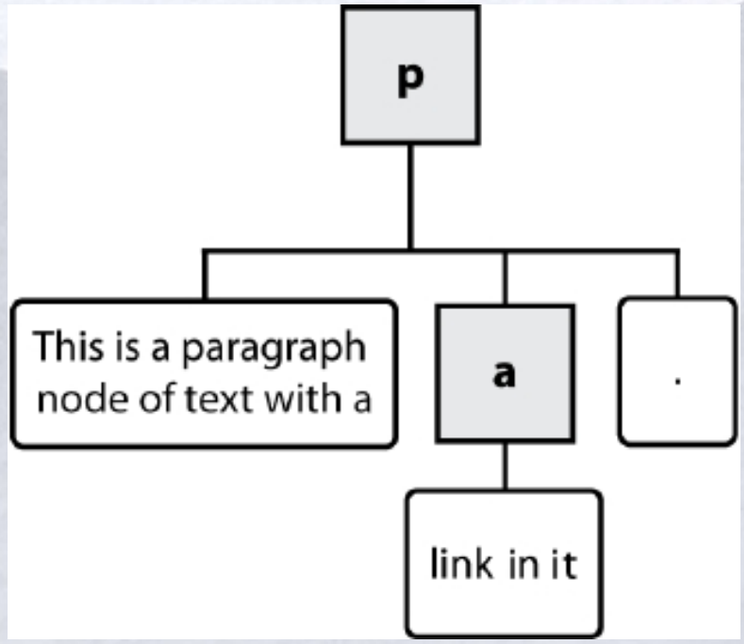


Type of DOM nodes

```HTML
<p>
  This is a paragraph of text with a
  <a href="/path/page.html">link in it</a>.
</p>
```

- element nodes (HTML tag)
  - can have children and/or attributes
- text nodes (text in a block element)
- attribute nodes (attribute/value pair)
  - text/attributes are children in an element node
  - cannot have children or attributes

not usually shown when drawing the DOM tree



Traversing the DOM tree

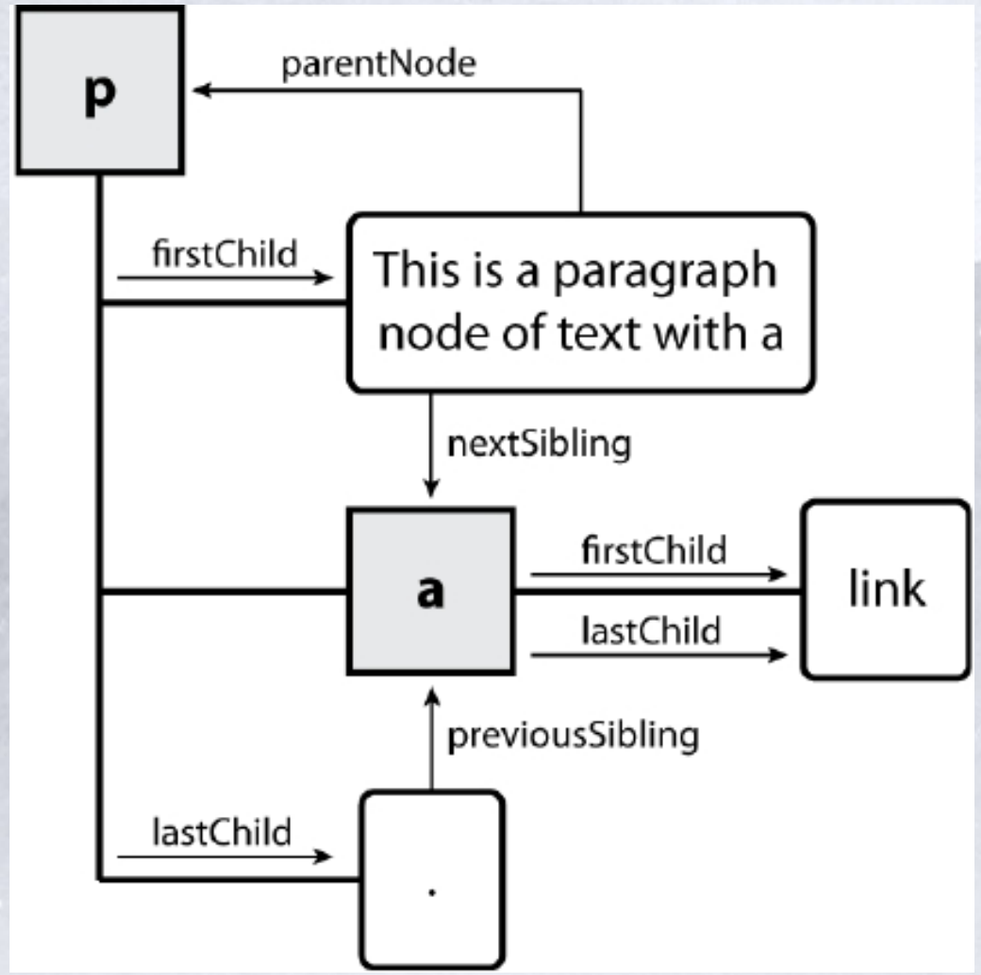
- every node's DOM object has the following properties:

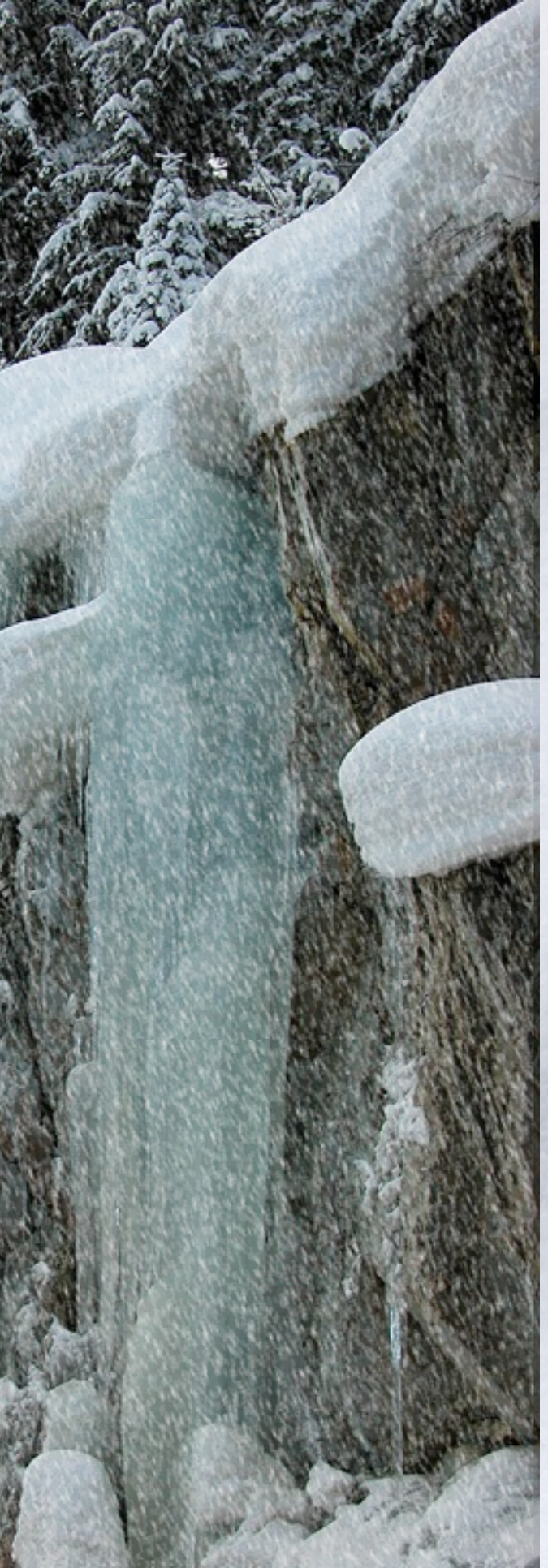
name(s)	description
firstChild, lastChild	start/end of this node's list of children
childNodes	array of all this node's children
nextSibling, previousSibling	neighboring nodes with the same parent
parentNode	the element that contains this node

- browser incompatiblity information (IE sucks)

DOM tree traversal example

```HTML
<p id="foo">This is a paragraph of text with a
  <a href="/path/to/another/page.html">link</a>.</p>
```





Outline

- Client Side Basics
- Overview of JavaScript
- JavaScript Basic Syntax
- DOM tree
- *The Browser Object Model (BOM)*

Global DOM objects

- Every browsers' Javascript program can refer to the following global objects:

name	description
document	current HTML page and its content
history	list of pages the user has visited
location	URL of the current HTML page
navigator	info about the web browser you are using
screen	info about the screen area occupied by the browser
window	the browser window

The window object

- The window object is supported by all browsers. It represents the browser's window.
- All global JavaScript objects, functions, and variables automatically become members of the window object.
- Global variables are properties of the window object.
- Global functions are methods of the window object.
- Even the document object (of the HTML DOM) is a property of the window object:
  - window.document.getElementById("header");
  - document.getElementById("header");

The document object

- the current web page and the elements inside it
- properties:
  - anchors, body, cookie, domain, forms, images, links, referrer, title, URL
- methods:
  - getElementById
  - getElementsByName
  - getElementsByTagName
  - close, open, write, writeln

The location object

- The window.location object can be used to get the current page address (URL) and to redirect the browser to a new page.
- The window.location object can be written without the window prefix.
- Some examples:
  - window.location.href returns the href (URL) of the current page
  - window.location.hostname returns the domain name of the web host
  - window.location.pathname returns the path and filename of the current page
  - window.location.protocol returns the web protocol used (http:// or https://)
  - window.location.assign loads a new document

The navigator object

- The window.navigator object contains information about the visitor's browser.
- The window.navigator object can be written without the window prefix.
- Some examples:
  - navigator.appName
  - navigator.onLine
  - navigator.appCodeName
  - navigator.platform

```
<p id="demo"></p>
 <script>
    document.getElementById("demo").innerHTML =
    "Name is " + navigator.appName + ". Code name is " +
    navigator.appCodeName;
 </script>
```

Warning !!!

- The information from the navigator object can often be misleading, and should not be used to detect browser versions because:
  - Different browsers can use the same name
  - The navigator data can be changed by the browser owner
  - Some browsers misidentify themselves to bypass site tests
  - Browsers cannot report new operating systems, released later than the browser

The screen object

- The window.screen object contains information about the user's screen.
- The window.screen object can be written without the window prefix.
- Properties:
  - screen.width
  - screen.height
  - screen.availWidth
  - screen.availHeight
  - screen.colorDepth
  - screen.pixelDepth

The history object

- The window.history object contains the browsers history.
- The window.history object can be written without the window prefix.
- To protect the privacy of the users, there are limitations to how JavaScript can access this object.
- Some methods:
  - history.back() - same as clicking back in the browser
  - history.forward() - same as clicking forward in the browser

JavaScript Cookies - What are Cookies?

- Cookies let you store user information in web pages.
- Cookies are data, stored in small text files, on your computer.
- Cookies were invented to solve the problem "how to remember information about the user":
  - When a user visits a web page, his name can be stored in a cookie.
  - Next time the user visits the page, the cookie "remembers" his name.
- Cookies are saved in name-value pairs like
  - username=Tom
- When a browser request a web page from a server, cookies belonging to the page is added to the request. This way the server gets the necessary data to "remember" information about users.

Create a Cookie with JavaScript

- JavaScript can create, read, and delete cookies with the document.cookie property.
- With JavaScript, a cookie can be created like this:
  - document.cookie="username=Tom";
- You can also add an expiry date (in UTC time). By default, the cookie is deleted when the browser is closed:
  - document.cookie="username=Tom; expires=Thu, 18 Sep 2015 10:00:00 UTC";
- With a path parameter, you can tell the browser what path the cookie belongs to. By default, the cookie belongs to the current page.
  - document.cookie="username=Tom; expires=Thu, 18 Sep 2015 10:00:00 UTC; path=/";

How to use cookies

- Read a Cookie with JavaScript
  - var x = document.cookie;
  - Note document.cookie will return all cookies in one string much like: cookie1=value1; cookie2=value2; cookie3=value3;
- Change a Cookie with JavaScript
  - With JavaScript, you can change a cookie the same way as you create it
  - document.cookie="username=Tom; expires=Thu, 18 Sep 2015 10:00:00 UTC; path=/";
  - The old cookie is overwritten.
- Delete a Cookie with JavaScript. Deleting a cookie is very simple. Just set the expires parameter to a passed date:
  - document.cookie = "username=; expires=Thu, 01 Jan 1970 00:00:00 UTC";
  - Note that you don't have to specify a cookie value when you delete a cookie.

Example

```
  <script>
function setCookie(cname,cvalue,exdays) {
    var d = new Date();
    d.setTime(d.getTime() + (exdays*24*60*60*1000));
    var expires = "expires=" + d.toGMTString();
    document.cookie = cname+"="+cvalue+"; "+expires;
}
function getCookie(cname) {
    var name = cname + "=";
    var ca = document.cookie.split(';');
    for(var i=0; i<ca.length; i++) {
        var c = ca[i];
        while (c.charAt(0)==' ') c = c.substring(1);
        if (c.indexOf(name) == 0) {
            return c.substring(name.length, c.length);
        }
    }
    return "";
}
function checkCookie() {
    var user=getCookie("username");
    if (user != "") {
        alert("Welcome again " + user);
    } else {
       user = prompt("Please enter your name:","");
       if (user != "" && user != null) {
           setCookie("username", user, 30);
       }
    }
}
</script>
```

References

- JavaScript编程精解（原书第3版） [Eloquent JavaScript: A Modern Introduction to Prog]. [美] 马尔奇·哈弗贝克（Marijn Haverbeke） 著，卢涛　李颖 译. 机械工业出版社, 2020.
- http://www.w3schools.com/
- Javascript权威指南

Thanks!!!