JavaScript

- Popular scripting language:
  - Dynamic and **loosely typed** variables.
  - **Functions** are now first-class citizens.
  - Supports **OOP**.

```javascript
var simple = 2;          // Simple variables.
simple = "I'm text now!"; // Changing type.

var array = new Array( 1, 2, "hey" ); // Associative array
array[3] = new Array( "a", "b" );    // with more complex content.

var obj = { "some-text": "This is a point" }; // An object
obj.x = -1.2;                // and its members.
obj.y = +3.4;
```
What does the following function do?

```javascript
function whatDoIDo( x )
{
    // If the input argument is invalid, throw an exception!
    if (x < 0) throw new Error( "x must not be negative" );

    // Otherwise, compute a value and return normally.
    for( var f = 1; x > 1; f *= x, x-- ) /* empty */ ;
    return f;
}
```
// Store the start and end points (state) of this new range object.
function Range(from, to) {
    this.from = from;
    this.to = to;
}

// All Range objects inherit from this object.
// Note that the property name must be "prototype" for this to work.
Range.prototype = {
    constructor: Range, // Explicitly set the constructor back-reference.

    includes: function(x) {
        return this.from <= x && x <= this.to;
    },

    toString: function() {
        return "(" + this.from + "..." + this.to + ")";
    }
};

- What about OOP in JavaScript?
```javascript
var r = new Range(1, 3);

r.includes(2)  
true

console.log(r.toString())
(1...3)
```
var or not var...  
that's the question...

```javascript
var foo = 10;

myFunction = function() {
  var foo = 20;

  (function() {
    var wibble = 1;
    foo = 30;
  })();
  console.log(foo); // What is foo?
}();

myFunction();
console.log(foo); // What is foo?
```
How to include JavaScript in a webpage?

- Inline, between a pair of `<script>` and `</script>` tags.
- From an external `.js` file specified by the `src` attribute of a `<script>` tag.
- In an HTML event handler attribute, such as `onclick` or `onmouseover`.
- In a URL that uses the special `javascript:` protocol.

```html
<a href="javascript:alert( new Date().toLocaleTimeString() );">Check the time!</a>
```
What does this script do?

```javascript
window.onload = function() {
  var elements = document.getElementsByClassName( "reveal" );
  for( var i = 0; i < elements.length; i++ ) {
    var elt = elements[i];
    var title = elt.getElementsByClassName( "handle" )[0];
    title.onclick = function() {
      if( elt.className == "reveal" )
        elt.className = "revealed" ;
      else
        if( elt.className == "revealed" )
          elt.className = "reveal" ;
    }
  }
};
</script>
Asynchronous Javascript And XML.
- An architecture for web applications that features scripted HTTP.
- Avoids page reloads.
- Makes web applications look and feel like desktop applications.

The XMLHttpRequest object defines the API for scripted HTTP.
- We have to use it in the same order as the HTTP requests parts:
  - HTTP method and URL – request.open( "GET", "/your/url" );
  - Optional headers - request.setRequestHeader( "Content-Type", "text/plain" );
  - Optional request body – request.send( optionalBody );
What does this function do? Is anything missing?

```javascript
function postMessage( msg )
{
    var request = new XMLHttpRequest(); // New request.
    request.open( "POST", "/log.php" ); // POST to a server script.

    // Send the message, in plain-text, as the request body.
    request.setRequestHeader( "Content-Type",
        "text/plain;charset=UTF-8" );

    request.send( msg );
}
```
Retrieving the server’s response (from GET request)

```javascript
function getText( url, callback )
{
    var request = new XMLHttpRequest();  // Create new request.
    request.open( "GET", url );          // Specify url to fetch.
    request.onreadystatechange = function()
    {
        // If the request is complete and was successful.
        if( request.readyState === 4 && request.status === 200 )
        {
            var type = request.getResponseHeader( "Content-Type" );
            if( type.match( /^text/ ) ) // Is response text?
                callback(request.responseText);  // Pass it to callback
        }
    }
    request.send( null );                 // Why null here?
}
```
function postData( url, data, callback )
{
    var request = new XMLHttpRequest();
    request.open( "POST", url );

    request.onreadystatechange = function()
    {
        if( request.readyState === 4 && callback )
            callback( request );
    };
    request.setRequestHeader( "Content-Type", "application/x-www-form-urlencoded" );

    request.send( encodeFormData( data ) );
}
JavaScript Object Notation

```
    o = {x:1, y:{z:[false,null,""]}};  // Define a test object.
    s = JSON.stringify(o);  // s = '{"x":1,"y":{"z":false,null,""}}'.
    p = JSON.parse(s);  // p is a deep copy of o.
```

How do we use eval() instead of parse()?
HTML5

- The new `<canvas>` tag:
  - Works with JavaScript to generate computer graphics in your browser.

```html
...<body>
<canvas id="myCanvas" width="800" height="600"></canvas>
<script>
  var canvas = document.getElementById( 'myCanvas' );
  var context = canvas.getContext( '2d' );
  context.font = '40pt Calibri';
  context.fillStyle = '#ffffff';
  context.fillText( 'Hello World!', 150, 100 );
...
```
Cookies

- A small amount of named data stored by the web browser, and associated with a particular website.
Cookies

- Cookies are strings containing
  - `name = value;`
  - `path = path;`
  - `domain = domain;`
  - `max-age = seconds;`
  - `secure;`

- To delete them we set their **max-age to zero** and their **value to “”** by using the same `name, path, and domain`.
In JavaScript

```javascript
Cookie c = new Cookie( "color", "#abcdef" );
c.setPath( "/" );
c.setMaxAge( 24*60*60 );
response.addCookie( c );
```

In a Java Servlet

```java
Cookie c = new Cookie( "color", "#abcdef" );
c.setPath( "/" );
c.setMaxAge( 24*60*60 );
response.addCookie( c );
```

In JavaScript

```javascript
document.cookie = "color=red; domain=localhost; path=/; max-age=1000";
```