ANNOUNCEMENTS

- Midterm #2
ONLOAD/ONUNLOAD

- **ONLOAD** → Allow us to execute code when the page is loaded.
- **Example**: clock.html, clock.js
- **ONUNLOAD** → Allow us to execute code when the user navigates away from a web page.
You use the . (period) operator to access an object’s properties

- `<OBJECT>..<PROPERTY>`

A property value can be any data type we have seen including objects.

You can create your own objects by either:

```javascript
var myObj = {};
var myOtherObj = new Object();
```

You can create properties by assigning a value to it (we do not use `var`).

```javascript
myObj.created = "Monday";
```

You can update the property by assigning a new value.

You can delete a property with the delete operator.

```javascript
delete myObj.created;
```

You can check for the existence of a property using the “in” operator.

**Example:** ObjectEx.java
FOR/IN

- General form

  ```
  for (propertyName in object)
  statement
  ```

- Can be used to display the properties of an object (Example: forIn.html).
- for/in does not specify the order in which properties of an object are visited.
- **Example:** ObjectEx.java
- The for/in does not loop through all the possible properties as some properties are considered non-enumerable.
- User-defined properties are enumerable.
We can also view an object as an entity that associates values with strings. How? Let’s first see how we can use the [ ] operator to access properties.

You can use [ ] operator instead of . (period) operator

myObj.created → myObj[“created”]

IMPORTANT: Notice that we have a string on the right side (“created”) whereas on the left side it is a property (variable)

Using [ ] operator can provide a nice alternative to add properties to an object dynamically (when the program is executing).

Example: AddingProperties.html
Global object – created by JavaScript interpreter when it starts up.
Interpreter initializes the Global object with predefined values and functions. For example, parseInt, Infinity, etc.

Top-level code – JavaScript code that does not belong to a function.

Global variables – variables in top-level code.
Global variables are properties of the Global object. When you define a variable outside any function you are defining a global variable (a property of the global object).

You should avoid using global variables in your code.
In client-side JavaScript the Window object (window) represents the global object for all JavaScript code present in the browser window.

You can use the keyword this to refer to the Global object. Keep in mind that inside a function this does not refer to the global object.

Example: GlobalObject.html
SESSIONS

- Session - time period during which a person views a number of different web pages in a browser and then quit.
- What would you like
  - To keep track of information throughout the session.
  - For example, keeping track of color preferences, usernames, data selection, etc.
- What is the problem?
  - http (the protocol that makes possible the communication between browsers and web servers) is stateless (it has no memory)
  - Stateless - every page request is independent
- One Possible Solution
  - Cookies
COOKIES

- Cookie - small piece of information sent by a server and stored either in the browser’s memory or as a small file in the hard drive. Acceptance of the cookie depends on the client.
- Browser sends the cookie back with every request to the server that sent the cookie.
- Cookie - contains a name/value pair.
- Setting a cookie - associating a value with a name.
- Getting a cookie - getting the value associated with a name.
- Constrains:
  - Browser typically accept only 20 cookies per domain before dropping old cookies
  - 4KB per cookie
  - 300 cookies per domain
Each cookie consists of name, value, expiration date, host, and path information.

This is how the cookie information may look like when sent by the server in the http header:

```
Set-Cookie: automobile=nelyota; path=/;
domain=notRealCars.com
```

If no expiration date is set for a cookie, the cookie expires when the user's session expires (i.e., when the user closes the browser).

If the user accesses any page matching the path and domain of the cookie, the browser will resend the cookie to the server.

Let's see cookies in our browser.
Setting/Reading Cookies

- **Setting cookies**
  - We can set a cookie by using `document.cookie`
    ```javascript
    document.cookie = "school=UMCP";
    document.cookie = "mascot=terp";
    ```
  - **Example:** setCookie.html

- **Reading cookies**
  - `document.cookie` has a string with all the cookies
  - You must extract from the string each cookie
  - Cookies are separated by `;`
  - **Example:** readCookie.html
COOKIES WITH AN EXPIRATION DATE

- Cookies without an expiration date will expire when the browser is closed.
- Specify expiration date using “expires” and date in GMT
- GMT (Greenwich mean time)
  Wdy, DD-Mon-YYYY HH:MM:SS GMT
  Sun, 15-Apr-2007 11:29:00 GMT
- **Example:** setCookieExpiration.html
  - Syntax is very strict (you must have space after semicolon)
  - When updating a cookie make sure use the same features (expires, path, etc.)
  - To delete a cookie set the expiration time to some point in the past.