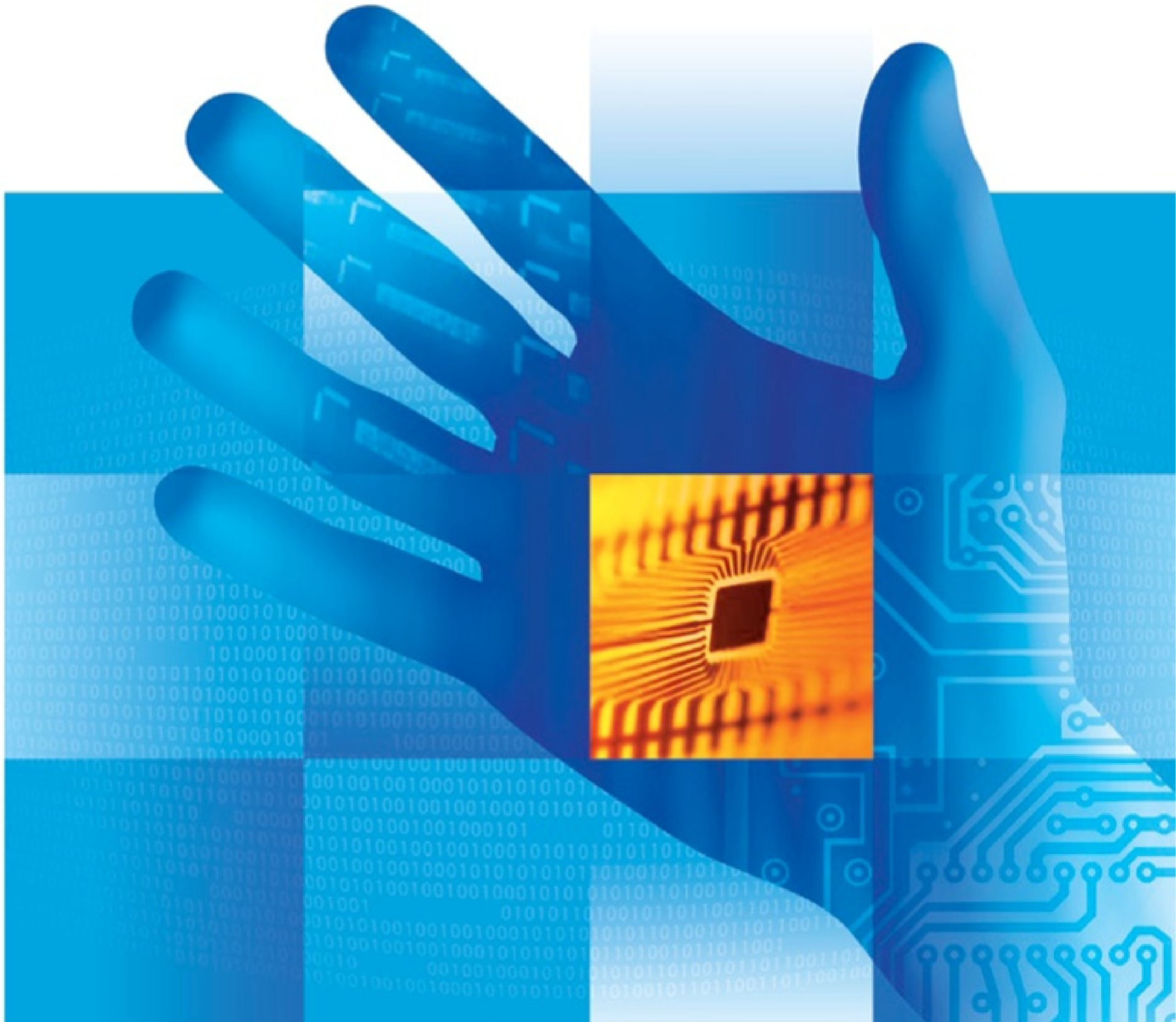
Basic Storage
Offline applications

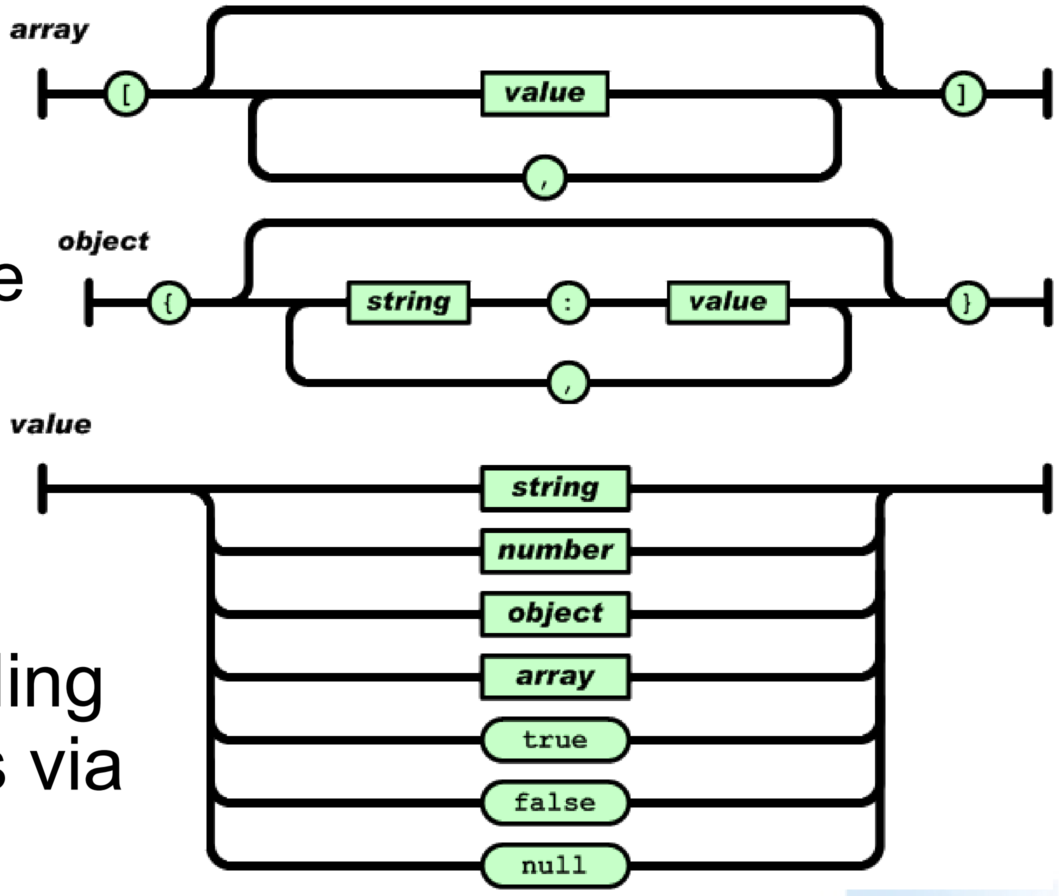


TAMZ 1

Lab 3

JS Object Notation (JSON)

- Object is stored as a string using JavaScript syntax
  - eval can be used, insecure
    eval('(' + myJSONString + ')');
- Javascript-enabled browsers following ECMAScript standard (2009) offer a way of encoding and decoding these objects via **JSON** object
  - JSON.**parse**(myString[, rev]);
    - reviver function as 2nd arg.
  - JSON.**stringify**(myObject[, rep]);
    - replacer function as 2nd arg.



Images courtesy of: http://json.org

Web Storage

- Abused nowadays for usertracking, no easy way to display stored data, but main idea was to persist data inside a session and between sessions
  - `localStorage` – data without expiration date
  - `sessionStorage` – data for current browser session (lost after browser restart)

- Everything stored as string key-value pairs in an object (must be converted to strings and back for non-string values – esp. objects and arrays)

```
// Really basic use for objects, using JSON
if (typeof(Storage)!=="undefined" && localStorage != null) {…}

localStorage.conversions = JSON.stringify(conversions);
conversions = JSON.parse(localStorage.conversions);
```

Storage interface

Storage interface

- Attributes
  - length – number of elements in storage (readonly)
- Methods
  - key(*idx*) – returns string key for *idx*
  - getItem(*key*) – returns string value (*data*) for *key*
  - setItem(*key*, *data*) – stores *data* for *key* in the storage
  - removeItem(*key)* – deletes *data* for *key* from the storage
  - clear() – clears the whole storage (removes everything)
- Storage event in DOM
  - fired on each setItem(), removeItem() and clear()
  - *storage* event (key, oldvalue, newvalue, url, storage area)
- Storage attributes in HTML document
  - localStorage, sessionStorage
- QUOTA_EXCEEDED_ERR – storing to already full storage

Web Storage examples

- Persistent counter with local storage

```
<div>Page display count in this browser: <span id="cnt">?</span></div>
<script>
   if (!localStorage.pageShows)  { localStorage.pageShows = 0; }
   localStorage.pageShows = parseInt(localStorage.pageShows) + 1;
   document.getElementById('cnt').textContent = localStorage.pageShows;
</script>
```

- Removing an item from local storage

```
localStorage.removeItem('pageShows');
```

- Removing all items session storage

```
sessionStorage.clear();
```

- Accessing all items in session storage

```
for (var i = 0; i < sessionStorage.length; i++) {
    var key=sessionStorage.key(i);
    document.writeln("K: " +key+ " V: "+sessionStorage.getItem(key));
}
```

Web Storage closing remarks

- There are no methods to filter/sort/iterate the storage objects directly, but we can write our own
  - use .length and .key(id) in a *for* cycle like in the example
  - write a filter function used within the cycle
    - e.g. myfilter(key) → true/false (include the item?)
  - create an output array with the filtered result
  - provide a sort function which will be used to sort the array
    - e.g. mysort(a, b) → returns: -1 (a<b), 0 (a==b), 1 (a>b)
- Same origin policy is applied
  - Scripts having same protocol, host and port share storage
    - We can use the data from other pages on server
    - The ~5 megabyte limit is shared by these scripts!
    - Security considerations, race conditions
      - we should use prefixing, e.g. **app.record**, encrypt data
- Two basic approaches for storing all application data
  - Store everything in single key × each item has its own key

Application cache

"Offline" or pre-cached pages defined in application manifest

- Inside HTML
  `<html manifest="demo.appcache">`
- Cache manifest file MIME type `(e.g. demo.appcache)`
  `Content-Type: text/cache-manifest`
- Contains `CACHE MANIFEST`, `NETWORK:`, `FALLBACK:` and newer `SETTINGS:` section
- By default `SETTINGS` is `fast` (use cached data when on-line)
- Comments in manifest
  - start with # (not first line!)
  - may be used to change the manifest & force reload of pages, unless they load with an error
- Cache size limit (~5MB)

```
CACHE MANIFEST
# File ver. 1.2.1
/theme.css
/logo.gif
/main.js

NETWORK:
* #File names or *

FALLBACK:
/html/ /offline.html

SETTINGS:
prefer-online
```

Task (1 point)

Create an application containing persistent data, which will be saved and restored each time we will get back to the page

- Possible topics:
  - Persistent textarea with history
    - contents of the textarea stored in sessionStorage
    - the history entries are stored in localStorage
  - Configuration form or Personal information
    - password stored in sessionStorage (unsafe)
    - single/multi choice selections, text, slider, date stored in localStorage
  - …