Aufgabe 1

In den Cookies werden einerseits die Sessiondaten gespeichert, sprich die ID unter der die Session auf dem Server gespeichert ist. In dieser wird der aktuelle Zustand der Applikation gespeichert. Es werden auch dauerhafte Einstellungen wie die Sprache oder ähnliches in den Cookies gespeichert.

Der LocalStorage wird für Nutzerspezifische Daten verwendet, sprich cachingspezifische Daten(Beispsielsweise können Fonts oder Sounds gut damit gecached werden) und anderes, was über ein Schließen der Webseite hinaus verfügbar sein soll.

Der SessionStorage wird für das Caching der aktuell aufgerufenen Seiten verwendet (zumindest bei Amazon.de) und generell für Daten die nach dem Schließen nicht mehr relevant sind. Hier wird bei Facebook auch ein Log über die Navigationspunkte geführt.

Generell ist anzumerken, dass der WebStorage bisher noch unbegrenzt ist, es aber für Cookies eine Maximalgröße gibt.

Aufgabe 2

Cookies sind an das document Objekt im DOM gebunden und einfach als Semikolonseparierter String angebunden. Die Key-Value Paare sind per = getrennt, also kann man per Substring Methode die entsprechenden Einträge finden. Um Werte zu löschen wird das Expire-Datum in die Vergangenheit gesetzt und um welche hinzuzufügen wird document.cookie einfach als key + ' ' + value gesetzt. Eine komplette Implementierung des Interface findet sich unten und unter `http://jsfiddle.net/BigDane/ZGZnE/2/`.

```
var cookieStorage = {
    set: function (key, value) {
        document.cookie = key + "=" + escape(value);
    },
    get: function (key) {
        var pos = this.find(key);
        if (pos.start !== undefined && pos.end !== undefined) {
            return unescape(document.cookie.substring(pos.start,
                pos.end));
        } else {
            return undefined;
        }
```

```
    },
    del: function (key) {
        document.cookie = key + '=; expires=Thu, 01 Jan 1970 00:00:01
            GMT;';
    },
    find: function (key) {
        // Returns start and end of key value pair
        var pos = {};
        var cookies = document.cookie;
        pos.start = cookies.indexOf(" " + key + "=");
        if (pos.start === -1) {
            pos.start = cookies.indexOf(key + "=");
        }
        if (pos.start === -1) {
            return {}; // Well, wo don't know about that
        }

        pos.end = cookies.indexOf(";", pos.start);
        if (pos.end === -1) {
            pos.end = cookies.length;
        }

        return pos;
    }
};


cookieStorage.set('A', '55');

console.log('A is set to:' + cookieStorage.get('A'));

cookieStorage.del('A');
console.log('A is deleted and now:' + cookieStorage.get('A'));
```

Die WebStorages hingegen sind deutlich moderner und lassen sich direkt als Key-Value Store nutzen. Folgendes Objekt abstrahiert den Webstorage:

```
var Storage = (function() {
  function Storage(storage) {
    this.storage = storage;
  }
```

```
  return {
        set: function (key, value){
              return this.storage.setItem(key, value);
        },
        get: function (key) {
              return this.storage.getItem(key);
        },
        del: function (key){
              return this.storage.removeItem(key);
        }
  };


})();

var session = new Storage(sessionStorage);
var local = new Storage(localStorage);
```