2.– 5. September 2013
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Offline. Na und?

Strategien für offlinefähige Applikationen in HTML5

Stephan Hochdörfer
bitExpert AG

Offline. Na und?

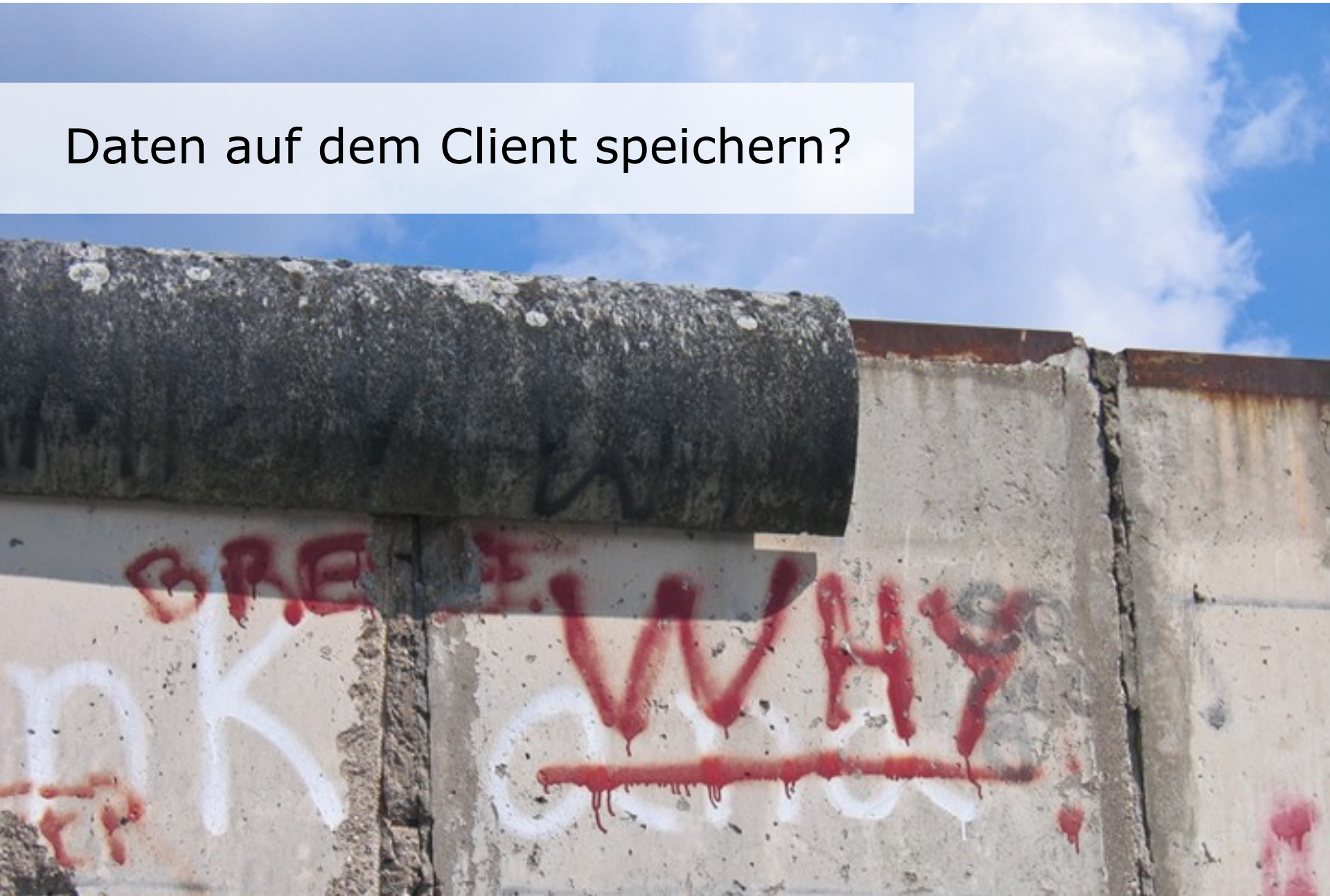
Strategien für offlinefähige Applikationen in HTML5

Stephan Hochdörfer, bitExpert AG

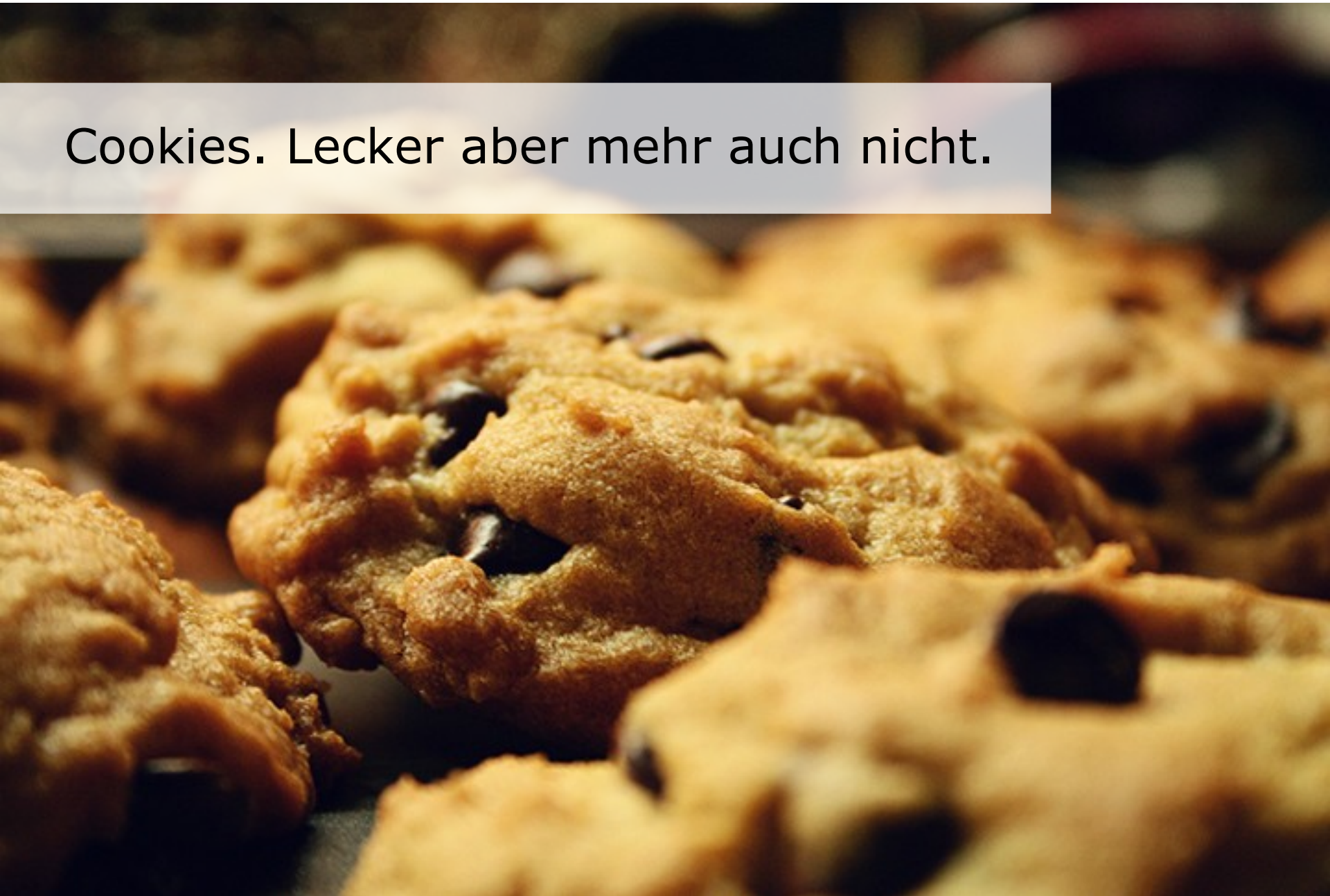
Über mich

- Stephan Hochdörfer
- Head of IT der bitExpert AG, Mannheim
- S.Hochdoerfer@bitExpert.de
- @shochdoerfer

Daten auf dem Client speichern?



Wie wurde dies in der Vergangenheit gelöst?


Cookies. Lecker aber mehr auch nicht.

IE DHTML Behaviours


Flash Cookies schmecken auch lecker.


Google Gears mit neuem Ansatz.


Gibt es nichts wirklich sinnvolles?

als Lösung...

[...] we take the next step, announcing 2014 as the target for Recommendation.

Jeff Jaffe, Chief Executive Officer, World Wide Web Consortium

Offline. Na und? - Strategien für offlinefähige Applikationen

Mark Zuckerberg: Our Biggest Mistake Was Betting Too Much On HTML5 | TechCrunch - Chromium

Mark Zuckerberg: Our Big

techcrunch.com/2012/09/11/mark-zuckerberg-our-biggest-mistake-with-mobile-was-betting-too-much-on-html5/

TechCrunch HOME STARTUPS MOBILE GADGETS EUROPE VIDEO MORE SEARCH

Mark Zuckerberg: Our Biggest Mistake Was Betting Too Much On HTML5

DREW OLANOFF

Tuesday, September 11th, 2012 **75 Comments**

Today, Mark Zuckerberg revealed that Facebook's mobile strategy relied too much on HTML5, rather than native applications.

Not only was this a big mistake with mobile, but Zuckerberg says that its biggest mistake period was the focus on HTML5. This is the first time that the Facebook CEO has openly admitted this, but things are looking good for the new iOS native app. According to Zuckerberg, people are consuming twice as many feed stories since the update to the new iOS app, which is great.

> *The first half year has been a little bit slow on product, but for the next six months I expect a lot of really cool stuff.*

This "really cool stuff" will probably have monetization in mind, as it's very clear that **mobile is the path to ad revenue** for the company.

It's extremely difficult for a company to nose-dive into an adoption of a particular set of tools and then quickly change course. I suspect that this is exactly what happened with Facebook and things are at least looking up.

Last year, Facebook's CTO at the time, **Bret Taylor, discussed the future of mobile**, which at the time very much included HTML5. Here's a snippet from that piece, where Taylor discusses the platform:

> *Does that mean an evolution away from Flash? After all, Flash dominates the market for the types of HTML5 games that Facebook is talking about. "Well it's hard," Taylor said about Flash specifically. When I laughed and noted he was giving the diplomatic answer, he assured me that it is something they think about a lot. "We want to be ahead of the curve and fill in the gaps when possible," is how he ended up putting it.*

Offline. Na und? - Strategien für offlinefähige Applikationen

The Making of Fastbook: An HTML5 Love Story | Blog | Sencha - Chromium

The Making of Fastbook: A

www.sencha.com/blog/the-making-of-fastbook-an-html5-love-story

Sencha

Products | Support | Training | Company | Blog | Contact | Store

Home / Blog

Blog

Search

The Making of Fastbook: An HTML5 Love Story

December 17, 2012 | Jamie Avins and Jacky Nguyen

+1 813 Tweet 4,245 Like 2.6k

When we started what became Sencha, we made a bet on the web: a bet that modern application development didn't need anything except the browser, a great set of frameworks and a great set of tools. With those three weapons in hand, we knew developers could build applications that would delight users. The advent of HTML5 upped the game and it gave developers even more tools to let them treat the browser as an application development platform and not a page rendering engine. Developers sprang at the opportunity and unleashed a torrent of apps — on both desktop and mobile — that leveraged the new HTML5 capabilities to build amazing applications using web standards.

So, when Mark Zuckerberg said HTML5 wasn't ready, we took a little offense to the comment.

144 comments
HTML5
RSS | Responses

Sencha Newsletter
Sign up for exclusive Sencha news!
Email Sign up

Twitter Facebook
Tumblr LinkedIn
RSS Feed Vimeo
Google+

Facebook Activity

Registrieren Erstelle ein Konto oder melde dich an, um zu sehen, was deine Freunde machen.

Productive Enterprise Web Development with Ext JS and Clear Data Builder | Blog | Sencha
25 Personen recommended das.

Soziales Plug-in von Facebook

Categories

Was bedeutet „offline“?

Was bedeutet „offline“?

Applikation vs. Content

Was bedeutet „offline“?

Application Cache vs. Offline Storage

App Cache für statische Ressourcen

HTML Page:

```
<!DOCTYPE html>
<html lang="en">
```

App Cache für statische Ressourcen

HTML Page:

```
<!DOCTYPE html>
<html lang="en" manifest="cache.manifest">
```

cache.manifest (Content-Type: text/cache-manifest):

```
CACHE MANIFEST

js/app.js
css/app.css
favicon.ico
http://someotherdomain.com/image.png
```

App Cache für statische Ressourcen

```
CACHE MANIFEST
# 2012-09-16

NETWORK:
data.php

CACHE:
/main/home
/main/app.js
/settings/home
/settings/app.js
http://myhost/logo.png
http://myhost/check.png
http://myhost/cross.png
```

App Cache für statische Ressourcen

```
CACHE MANIFEST
# 2012-09-16

FALLBACK:
/ /offline.html

NETWORK:
*
```

App Cache Scripting

```
// events fired by window.applicationCache
window.applicationCache.onchecking = function(e)
{log("Checking for updates");}
window.applicationCache.onnoupdate = function(e)
{log("No updates");}
window.applicationCache.onupdateready = function(e)
{log("Update ready");}
window.applicationCache.onobsolete = function(e)
{log("Obsolete");}
window.applicationCache.ondownloading = function(e)
{log("Downloading");}
window.applicationCache.oncached = function(e)
{log("Cached");}
window.applicationCache.onerror = function(e)
{log("Error");}

// Log each file
window.applicationCache.onprogress = function(e) {
  log("Progress: downloaded file " + counter);
  counter++;
};
```

App Cache Scripting

```
// Check if a new cache is available on page load.
window.addEventListener('load', function(e) {
  window.applicationCache.addEventListener('updateready',
  function(e) {

    if(window.applicationCache.status ==
        window.applicationCache.UPDATEREADY) {
      // Browser downloaded a new app cache.
      // Swap it in and reload the page
      window.applicationCache.swapCache();
      if (confirm('New version is available. Load it?)) {
        window.location.reload();
      }
    } else {
      // Manifest didn't change...
    }
  }, false);

}, false);
```

App Cache – Einige Fallstricke!

App Cache – Einige Fallstricke!

1. Dateien werden immer(!) vom lokalen Cache ausgeliefert.

App Cache – Einige Fallstricke!

2. Der lokale Cache wird nur dann aktualisiert wenn sich die manifest Datei geändert hat.

App Cache – Einige Fallstricke!

3. Nicht ladbare Dateien aus der CACHE Sektion führen dazu dass der Cache invalide ist.

App Cache – Einige Fallstricke!

4. Kann die manifest Datei nicht geladen werden, erfolgt kein Caching!

App Cache – Einige Fallstricke!

5. Nicht gecachte Ressourcen werden auf einer gecachten Seite nicht angezeigt.

App Cache – Einige Fallstricke!

6. Nach Aktualisierung des Caches muss die Seite neu geladen werden!

App Cache – Einige Fallstricke!

7. Mit expires Header arbeiten um das Cachen des manifests zu verhinden!

App Cache – Cache Manifest Validator

http://manifest-validator.com

App Cache – Was darf gecacht werden?

App Cache – Was darf gecacht werden?

Ja:

- Schriften
- Startbild
- Applikationsicon
- Einstiegsseite
- Fallbackseite

Nein:

- CSS
- HTML
- Javascript

App Cache – Was darf gecacht werden?

Den App Cache nur für
„statischen Content“ verwenden!

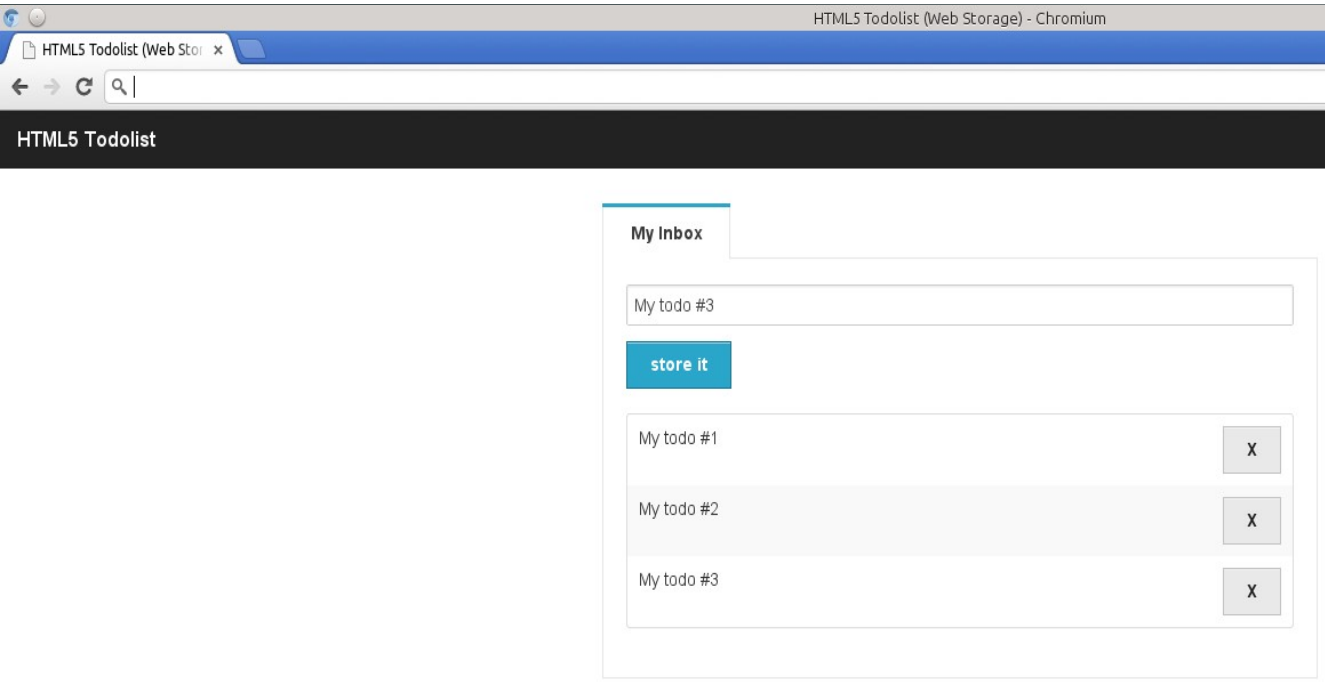
Ausflug: Data URI Schema

Ausflug: Data URI Schema

```
<!DOCTYPE HTML>
<html>
 <head>
  <title>The Data URI scheme</title>
  <style type="text/css">
  ul.checklist li {
    margin-left: 20px;
    background: white
url('
AFCAYAAACNbyblAAAAHElEQVQI12P4//8/w38GIAXDIBKE0DHxgljNBAA
O9TXL0Y4OHwAAAABJRU5ErkJggg==') no-repeat scroll left
top;
}
  </style>
 </head>
 <body>
  <img
src="
AFCAYAAACNbyblAAAAHElEQVQI12P4//8/w38GIAXDIBKE0DHxgljNBAA
O9TXL0Y4OHwAAAABJRU5ErkJggg==" alt="Red dot">
 </body>
</html>
```

Dynamische Daten lokal speichern...

Offline. Na und? - Strategien für offlinefähige Applikationen



Beispiel: Todoliste

Dynamische Daten lokal speichern...

Sourcen:

github.com/bitExpert/html5-offline

Dynamische Daten lokal speichern...

Web Storage, Web SQL Database,
IndexedDB, File API

Web Storage

Web Storage

Komfortable Art Daten offline zu speichern: Key/Value Speicher

Web Storage: 2 Arten

localStorage vs. sessionStorage

Web Storage: Datensatz hinzufügen

```
function add(item) {
    try {
        // for a new item set id
        if((typeof item.id === "undefined")
            || (null == item.id) || ("" == item.id)) {
            item.id = get_lastIndex() + 1;
        }

        // store object as string
        localStorage.setItem(item.id,
            JSON.stringify(item)
        );

        // update the index
        set_lastIndex(item.id);
    }
    catch(ex) {
        console.log(ex);
    }
}
```

Web Storage: Datensatz ändern

```
function modify(item) {
    try {
        // store object as string
        localStorage.setItem(item.id,
            JSON.stringify(item)
        );
    }
    catch(ex) {
        console.log(ex);
    }
}
```

Web Storage: Datensatz löschen

```
function remove (id) {
     try {
          localStorage.removeItem(id);
     }
     catch(ex) {
          console.log(ex);
     }
}
```

Web Storage: Datensätze auslesen

```
function read() {
	try {
		var lastIdx = get_lastIndex();
		for(var i = 1; i <= lastIdx; i++) {
			if(null !== localStorage.getItem(i)) {
				// parse and render item
				var item = JSON.parse(
					localStorage.getItem(i)
				);
			}
		}
	}
	catch(ex) {
		console.log(ex);
	}
}
```

Web Storage: Wie sessionStorage nutzen?

Web Storage: Wie sessionStorage nutzen?

Einfach: Ersetze „localStorage“
durch „sessionStorage“

Web Storage: Datensatz hinzufügen

```
function add(item) {
    try {
        // for a new item set id
        if((typeof item.id === "undefined")
            || (null == item.id) || ("" == item.id)) {
            item.id = get_lastIndex() + 1;
        }

        // store object as string
        sessionStorage.setItem(item.id,
            JSON.stringify(item)
        );

        // update the index
        set_lastIndex(item.id);
    }
    catch(ex) {
        console.log(ex);
    }
}
```

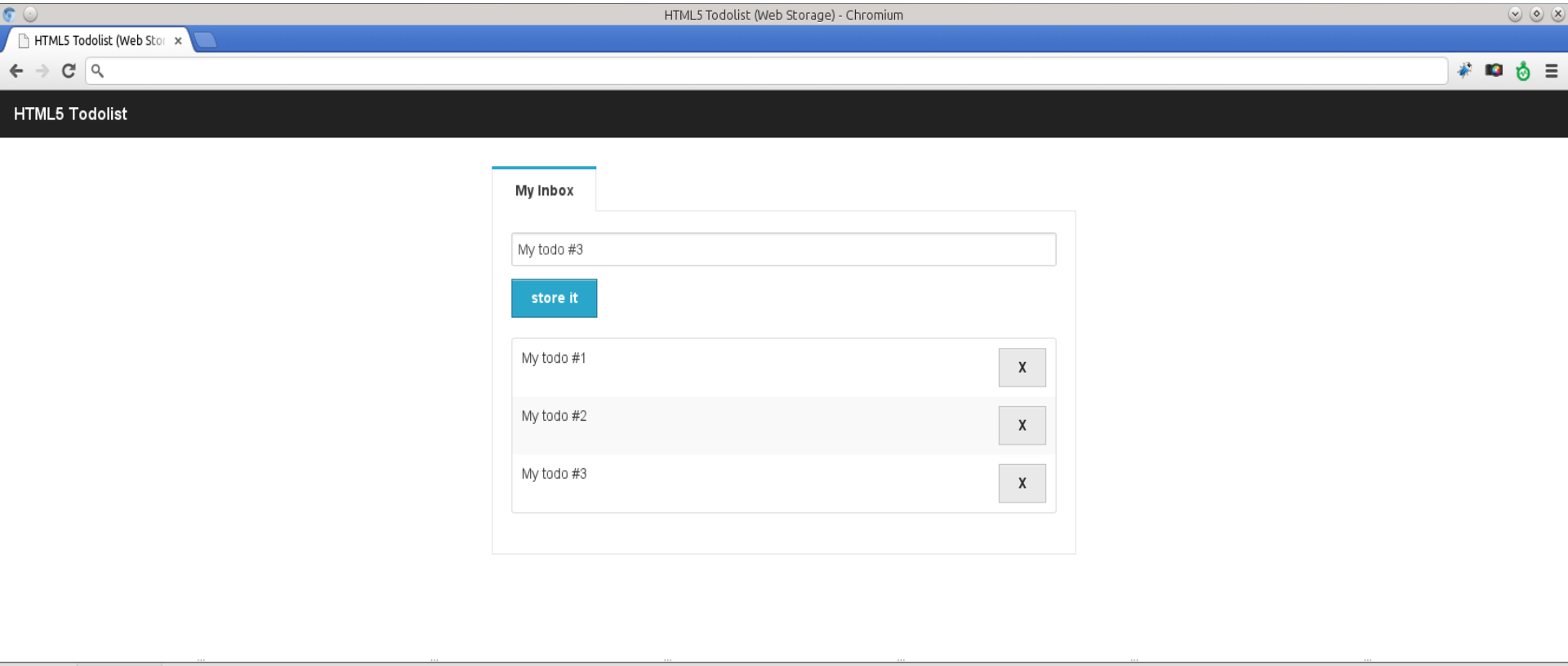
Web Storage: sessionStorage Fallstrick

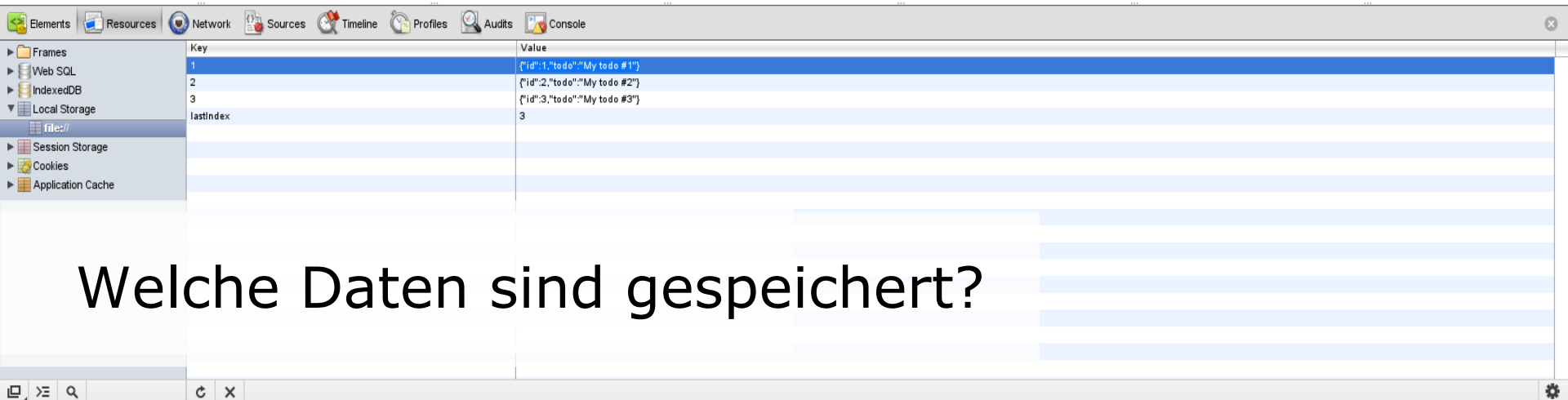
Jedes Tab / Fenster bringt
eigenen sessionStorage mit sich!

Web Storage: Alternative Zugriffsmöglichkeiten

```
var value = "my value";

// method call
localStorage.setItem("key", value);

// Array accessor
localStorage[key] = value;

// Property accessor
localStorage.key = value;
```

Offline. Na und? - Strategien für offlinefähige Applikationen





Welche Daten sind gespeichert?

Web Storage: Vorteile

Die meisten der aktuellen Browser „können“ Web Storage.

Web Storage: Vorteile

Funktioniert sogar im IE8 :)

Web Storage: Vorteile

Kein HTTP Overhead
wie bsp. bei Cookies.

Web Storage: Nachteile

Daten werden unstrukturiert gespeichert.

Web Storage: Nachteile

Nicht transaktionsfähig!

Web Storage: Nachteile

Nicht asynchron!

Web Storage: Nachteile

Daten können nicht automatisch verfallen. Manueller Aufwand nötig.

Web Storage: Nachteile

Unzureichende Informationen wie voll der lokale Cache wirklich ist.

Web Storage: Nachteile

http:// und https:// definieren unterschiedliche Storages.

Web SQL Database

Web SQL Database

Eine lokale SQL Datenbank
auf SQLite Basis.

Web SQL Database: Callbacks

```
var onError = function(tx, ex) {
    alert("Error: " + ex.message);
};

var onSuccess = function(tx, results) {
    var len = results.rows.length;

    for(var i = 0; i < len; i++) {
        // render found todo item
        render(results.rows.item(i));
    }
};
```

Web SQL Database: Datenbank erzeugen

```
// initalize the database connection
var db = openDatabase('todo', '1.0', 'Todo Database',
    5 * 1024 * 1024 );

db.transaction(function (tx) {
     tx.executeSql(
          'CREATE TABLE IF NOT EXISTS todo '+
          '(id INTEGER PRIMARY KEY ASC, todo TEXT)',
          [],
          onSuccess,
          onError
     );
});
```

Web SQL Database: Datenbankmigration

```
function m1(t){ t.executeSql("create table tbl1...") }
function m2(t){ t.executeSql("alter table tbl1...") }
function m3(t){ t.executeSql("alter table tbl1...") }

if(db.version == "") {
  db.changeVersion("", "1", m1, null, function() {
    db.changeVersion("1", "2", m2, null, function() {
      db.changeVersion("2", "3", m3);
    });
  });
}

if(db.version == "1") {
  db.changeVersion("1", "2", m2, null, function() {
    db.changeVersion("2", "3", m3);
  });
}

if(db.version == "2") {
  db.changeVersion("2", "3", m3);
}
```

Web SQL Database: Datensatz hinzufügen

```
function add(item) {
    db.transaction(function(tx) {
        tx.executeSql(
            'INSERT INTO todo (todo) VALUES (?)',
            [
                item.todo
            ],
            onSuccess,
            onError
        );
    });
}
```

Web SQL Database: Datensatz verändern

```
function modify(item) {
    db.transaction(function(tx) {
        tx.executeSql(
            'UPDATE todo SET todo = ? WHERE id = ?',
            [
                item.todo
                item.id
            ],
            onSuccess,
            onError
        );
    });
}
```

Web SQL Database: Datensatz löschen

```
function remove(id) {
    db.transaction(function (tx) {
        tx.executeSql(
            'DELETE FROM todo WHERE id = ?',
            [
                id
            ],
            onSuccess,
            onError
        );
    });
}
```

Web SQL Database: Datensätze auslesen

```
function read() {
    db.transaction(function (tx) {
        tx.executeSql(
            'SELECT * FROM todo',
            [],
            onSuccess,
            onError
        );
    });
}
```

Web SQL Database: Vorteile

Eine SQL Datenbank im Browser!

Web SQL Database: Nachteile

Eine SQL Datenbank im Browser!

Web SQL Database: Nachteile

SQLite kann seeeehr langsam sein!

Web SQL Database: Nachteile

Nicht länger Teil der
HTML5 Spezifikation!

IndexedDB

IndexedDB

Kompromiss aus Web Storage
und Web SQL Database.

IndexedDB

Die NoSQL Datenbank für
den Browser.

Web SQL Database vs. IndexedDB

Kategorie	Web SQL	IndexedDB
Speicherart	Tabellen mit Spalten und Zeilen	Objectstore mit Javascript Objekten und Keys
Abfrage mechanismus	SQL	Cursor APIs, Key Range APIs und Applicationslogik
Transaktionalität	Lock für Databanken, Tabellen oder Zeilen bei READ_WRITE Transaktionen	Locks für Datenbanken (VERSION_CHANGE Transaktion) und Objectstores (READ_ONLY, READ_WRITE Transaktion).
Transaktions-Commits	Transaktionen werden explizt erzeugt. Standard: Rollback, außer es wird explizit ein commit ausgeführt.	Transaktionen werden explizt erzeugt. Standard: Committen sofern kein Fehler aufgetreten ist.

IndexedDB: Vorarbeiten

```
// different browsers, different naming conventions
var indexedDB = window.indexedDB ||
   window.webkitIndexedDB || window.mozIndexedDB ||
   window.msIndexedDB;

var IDBTransaction = window.IDBTransaction ||
   window.webkitIDBTransaction;

var IDBKeyRange = window.IDBKeyRange ||
   window.webkitIDBKeyRange;
```

IndexedDB: Objektspeicher erzeugen

```
var db = null;
var request = indexedDB.open("todo");
request.onfailure = onError;
request.onsuccess = function(e) {
    db = request.result;
    var v = "1.0";
    if(v != db.version) {
        var verRequest = db.setVersion(v);
        verRequest.onfailure = onError;
        verRequest.onsuccess = function(e) {
            var store = db.createObjectStore(
                "todo",
                {
                    keyPath: "id",
                    autoIncrement: true
                }
            );
            e.target.transaction.oncomplete =
                function() {};
        };
    }
};
```

IndexedDB: Datensatz hinzufügen

```
function add(item) {
    try {
        var trans = db.transaction(["todo"],
            IDBTransaction.READ_WRITE);

        var store   = trans.objectStore("todo");
        var request = store.put({
            "todo": item.todo,
        });
    }
    catch(ex) {
        onError(ex);
    }
}
```

IndexedDB: Datensatz verändern

```
function modify(item) {
    try {
        var trans = db.transaction(["todo"],
            IDBTransaction.READ_WRITE);

        var store   = trans.objectStore("todo");
        var request = store.put(item);
    }
    catch(ex) {
        onError(ex);
    }
}
```

IndexedDB: Datensatz löschen

```
function remove(id) {
    try {
        var trans = db.transaction(["todo"],
            IDBTransaction.READ_WRITE);

        var store   = trans.objectStore("todo");
        var request = store.delete(id);
    }
    catch(ex) {
        onError(ex);
    }
}
```

IndexedDB: Datensätze auslesen

```
function read () {
    try {
        var trans = db.transaction(["todo"],
            IDBTransaction.READ);

        var store = trans.objectStore("todo");
        var keyRange = IDBKeyRange.lowerBound(0);
        var cursorRequest = store.openCursor(keyRange);

        cursorRequest.onsuccess = function(e) {
            var result = e.target.result;
            if(!!result == false) {
                return;
            }
            // @TODO: render result.value
            result.continue();
        };
    }
    catch(ex) {
        onError(ex);
    }
}
```

IndexedDB: Datensatz mittels Index auslesen

```
try {
    var index = db.openIndex('todo');
    var item  = index.get(1337);
}
catch(ex) {
    onError(ex);
}
```

IndexedDB: Index erzeugen

```
try {
    // create non-unique index
    db.createIndex("Created", "createdDate",
        {"unique": false});

    // create unique index
    db.createIndex("OtherKey", "otherField",
        {"unique": true});

    // create multi-column index (not working in IE10!)
    db.createIndex("MultiIndex", ["field1", "field2"]);
}
catch(ex) {
    onError(ex);
}
```

IndexedDB: Zu kompliziert?

IndexedDB: Zu kompliziert?

db.js als Wrapper nutzen

IndexedDB: db.js als Wrapper

```
var server;
db.open( {
    server: 'todo',
    version: 1,
    schema: {
        todo: {
            key: { keyPath: 'id' , autoIncrement: true},
            indexes: {
                Created: { }
            }
        }
    }
}).done(function(s) {
    server = s;
});
```

IndexedDB: db.js als Wrapper

```
server.todo.add( {
      todo: 'This is a sample todo item'
}).done(function(item) {
      // item stored
});
```

IndexedDB: db.js als Wrapper

```
server.todo.query()
    .execute()
    .done(function(results) {
          // do something with the results
    });
```

IndexedDB: Vorteile

Neuer Standard der künftig durch viele Browser unterstützt wird.

IndexedDB: Nachteile

Nicht alle mobilen Browser untersützen IndexedDB.

IndexedDB Polyfill

IndexedDB Emulation via WebSQL:
http://nparashuram.com/IndexedDBShim

File API

File API

FileReader API und FileWriter API

File API: Vorarbeiten

```
var onError = function(e) {
    var msg = '';

    switch(e.code) {
        case FileError.QUOTA_EXCEEDED_ERR:
            msg = 'QUOTA_EXCEEDED_ERR'; break;
        case FileError.NOT_FOUND_ERR:
            msg = 'NOT_FOUND_ERR'; break;
        case FileError.SECURITY_ERR:
            msg = 'SECURITY_ERR'; break;
        case FileError.INVALID_MODIFICATION_ERR:
            msg = 'INVALID_MODIFICATION_ERR'; break;
        case FileError.INVALID_STATE_ERR:
            msg = 'INVALID_STATE_ERR'; break;
        default:
            msg = 'Unknown Error'; break;
    };

    alert("Error: " + msg);
};
```

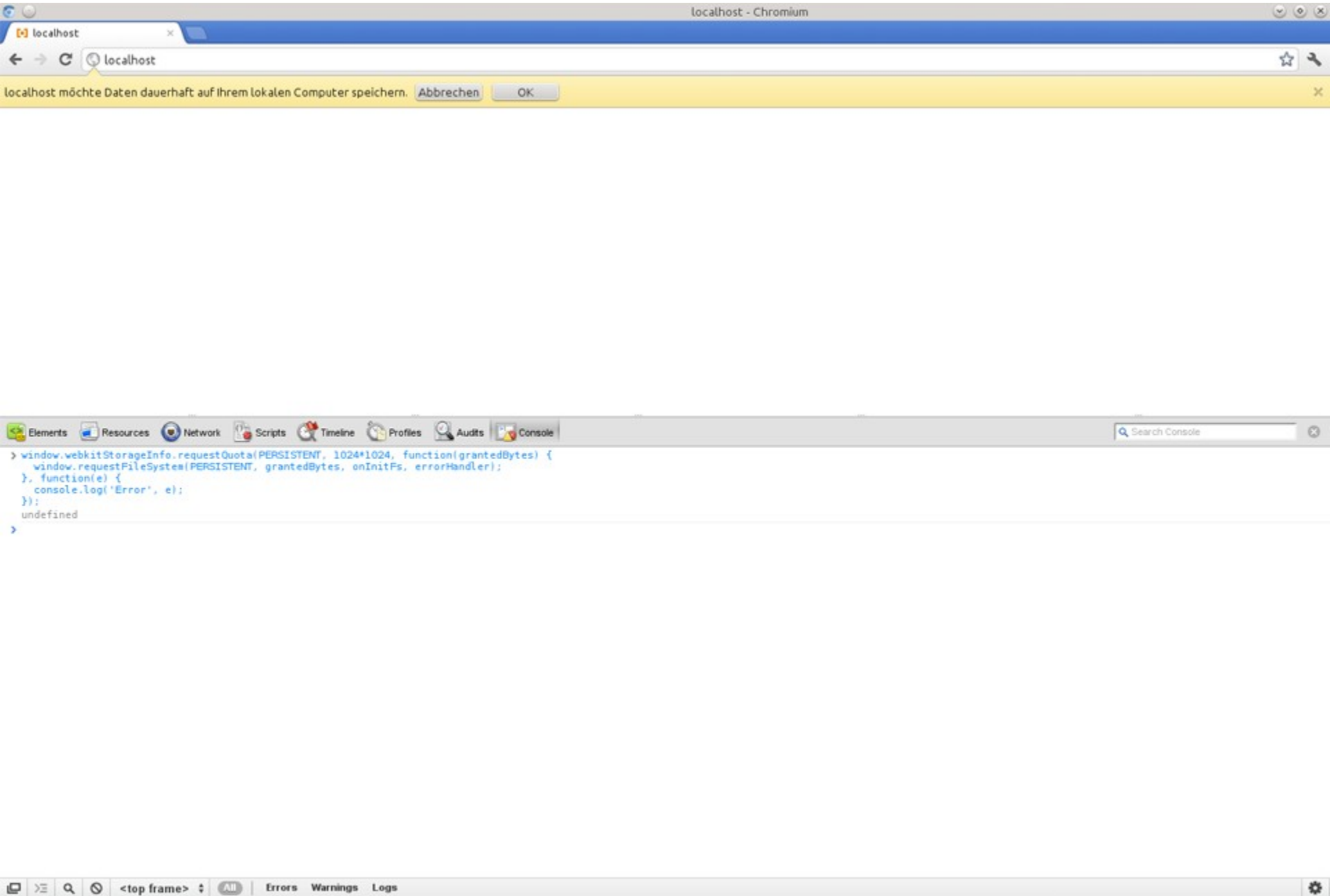
File API: Vorarbeiten II

```
// File system has been prefixed as of Google Chrome 12
window.requestFileSystem = window.requestFileSystem ||
    window.webkitRequestFileSystem;

window.BlobBuilder = window.BlobBuilder ||
    window.WebKitBlobBuilder;

var size = 5 * 1024*1024; // 5MB
```

File API: Quota anfordern

```
// request quota for persistent store
window.webkitStorageInfo.requestQuota(
     PERSISTENT,
     size,
     function(grantedBytes) {
          window.requestFileSystem(
               PERSISTENT,
               grantedBytes,
               function(fs) {
                    // @TODO: access filesystem
               }
          )
     }
}
```

Offline. Na und? - Strategien für offlinefähige Applikationen



File API: Daten hinzufügen

```
function add(item) {
      window.webkitStorageInfo.requestQuota(
          PERSISTENT,
          size,
          function(grantedBytes) {
              window.requestFileSystem(
                  PERSISTENT,
                  grantedBytes,
                  function(fs){
                      writeToFile(fs, item);
                  },
                  onError
              );
          },
          function(e) {
              onError(e);
          }
      );
  }
```

File API: Daten hinzufügen II

```
function writeToFile(fs, item) {
    fs.root.getFile(
        'todo.txt',
        {
            create: true
        },
        function(fileEntry) {
            fileEntry.createWriter(
                function(fileWriter) {
                    var bb = new window.BlobBuilder();
                    bb.append(JSON.stringify(item)+
                        "\n");

                    fileWriter.seek(fileWriter.length);
                    fileWriter.write(
                        bb.getBlob('text/plain'));
                }, onError
            );
        }, onError
    );
}
```

File API: Daten hinzufügen II

```
function writeToFile(fs, item) {
    fs.root.getFile(
        'todo.txt',
        {
            create: true
        },
        function(fileEntry) {
            fileEntry.createWriter(
                function(fileWriter) {
                    var bb = new window.BlobBuilder();
                    bb.append(JSON.stringify(item)+
                        "\n");

                    fileWriter.seek(fileWriter.length);
                    fileWriter.write(
                        bb.getBlob('text/plain'));
                }, onError
            );
        }, onError
    );
}
```

Deprecated!

bitExpert

File API: Daten hinzufügen II

```
function writeToFile(fs, item) {
    fs.root.getFile(
        'todo.txt',
        {
            create: true
        },
        function(fileEntry) {
            fileEntry.createWriter(
                function(fileWriter) {
                    var blob = new Blob([
                            JSON.stringify(item)+"\n"
                    ]);

                    fileWriter.seek(fileWriter.length);
                    fileWriter.write(blob);
                }, onError
            );
        }, onError
    );
}
```

File API: Daten auslesen

```
function read() {
        window.webkitStorageInfo.requestQuota(
            PERSISTENT,
            size,
            function(grantedBytes) {
                window.requestFileSystem(
                    PERSISTENT,
                    grantedBytes,
                    function(fs){
                        readFromFile(fs);
                    },
                    onError
                );
            },
            function(e) {
                onError(e);
            }
        );
}
```

File API: Daten auslesen II

```
function readFromFile(fs) {
    fs.root.getFile(
        'todo.txt',
        {
            create: true
        },
        function(fileEntry) {
            fileEntry.file(function(file){
                var reader = new FileReader();
                reader.onloadend = function(e) {
                    if (evt.target.readyState ==
                        FileReader.DONE) {
                        // process this.result
                    }
                };
                reader.readAsText(file);
            });
        }, onError
    );
}
```

Bin ich online?

Bin ich online?

```
document.body.addEventListener("online", function () {
  // browser is online!
}

document.body.addEventListener("offline", function () {
  // browser is not online!
}
```

Bin ich online? Andere Vorgehensweise...

```
$.ajax({
  dataType: 'json',
  url: 'http://myappurl.com/ping',
  success: function(data){
    // ping worked
  },
  error: function() {
    // ping failed -> Server not reachable
  }
});
```

Datensynchronisation

Datensynchronisation

PouchDB, the JavaScript Database that syncs!

Datensynchronisation

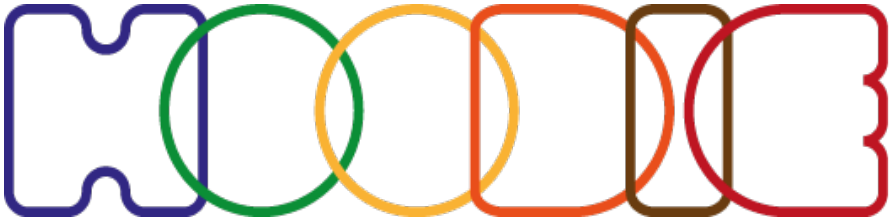
```
var db = new PouchDB('todo');

db.put({
 _id: 1,
 todo: 'Get some work done...',
});

db.replicate.to('http://example.com/mydb');
```

NoBackend Bewegung

NoBackend Bewegung

Entkopplung von Frontend und Backend!

NoBackend Bewegung: Hood.ie



„Hoodie is an architecture for frontend-only web apps“

Browserunterstützung?

Browserunterstützung?

	App Cache	Web Storage	WebSQL	IndexedDB	File API	Data URI
IE	10.0	8.0	10.0	10.0	-	8.0*
Firefox	11.0	11.0	11.0	11.0	19.0	16.0
Chrome	18.0	18.0	18.0	18.0	18.0	18.0
Safari	5.1	5.1	5.1	-	-	5.1
Opera	12.1	12.1	12.1	-	-	12.1
iOS Safari	3.2	3.2	3.2	-	-	3.2
Android	2.1	2.1	2.1	-	-	2.1

Quelle: http://caniuse.com

Speicherplatzbeschränkung?

Speicherplatzbeschränkung?

Alle vorgestellten Technologien sind durch Quotas beschränkt.

Speicherplatzbeschränkung?

	App Cache	Web Storage	WebSQL	IndexedDB	File API
iOS 5.1	10 MB	5 MB	5 MB	5 MB	
Android 4	unlimited	5 MB	?	?	
Safari 5.2	unlimited	5 MB	5 MB	5 MB	
Chrome 18	5 MB	5 MB	unlimited	unlimited	unlimited
IE 10	50 MB	10 MB	500 MB	500 MB	
Opera 11	50 MB	5 MB	5 MB	5 MB	
Firefox 11	unlimited	10 MB	50 MB	50 MB	

Minimumwerte, je nach Konfiguration ist mehr möglich.

Vielen Dank!