

HTML5 JavaScript Storage for Structured Data

Andy Gup, Esri

www.andygup.net

@agup



IndexedDB

and the

mobile web

Why attend this session?

Have storage needs > 5MBs

Want to store data types other than Strings

Don't want to manually serialize/deserialize

Looking into offline JavaScript

Agenda

Intro to IndexedDB coding patterns

Fitting IndexedDB into overall application

Performance

Wet your appetite! Can't cover it all

Who am I?

Andy Gup, Esri

Sr. Developer – JS and native Android



www.andygup.net

github.com/andygup

agup@esri.com

[@agup](#)



Structured data?

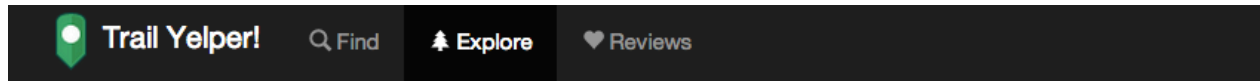
JSON Objects (not serialized)

Complex Objects (difficult serialize/deserialize)

Binary data (e.g. images, files)

Arrays

Offline JavaScript Demo



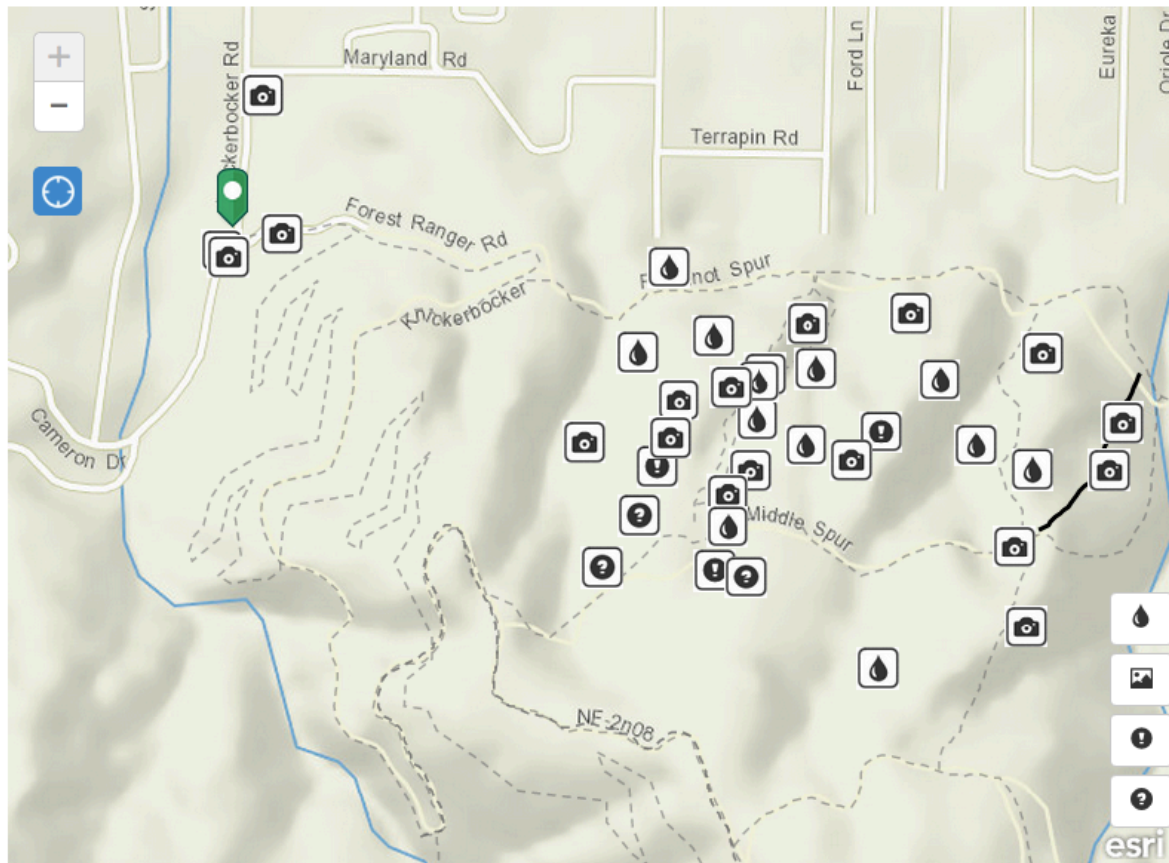
Step 2: Explore your trail

Locate your trail and take it with you!

Locate Trail

Save to Phone

Rate Trail



IndexedDB browser support?

► Show options

■ = Supported

■ = Not supported

■ = Partially supported

■ = Support unknown

IndexedDB - Candidate Recommendation

Method of storing data client-side, allows indexed database queries. Previously known as WebSimpleDB API.

Usage stats:

Global

Support:	55%
Partial support:	10.86%
Total:	65.86%

Show all versions

	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
								2.1		
								2.2		
						3.2		2.3		
						4.0-4.1		3.0		
	8.0					4.2-4.3		4.0		
	9.0	28.0	33.0			5.0-5.1		4.1		
	10.0	29.0	34.0			6.0-6.1		4.2-4.3	7.0	
Current	11.0	30.0	35.0	7.0	22.0	7.0-7.1	5.0-7.0	4.4	10.0	10.0
Near future		31.0	36.0	8.0	23.0	8.0		4.4.3		
Farther future		32.0	37.0		24.0					
3 versions ahead		33.0	38.0							

Notes

Known issues (0)

Resources (5)

Feedback

Edit on GitHub

Partial support in IE 10 & 11 refers to a number of subfeatures not being supported. Partial support in BB10 refers to an outdated specification being implemented. Code targeting the current state of the specification might not work.

IndexedDB browser support?

► Show options

■ = Supported

■ = Not supported

■ = Partially supported

■ = Support unknown

IndexedDB - Candidate Recommendation

Method of storing data client-side, allows indexed database queries. Previously known as WebSimpleDB API.

Usage stats:

Global

Support:	55%
Partial support:	10.86%
Total:	65.86%

Show all versions

	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	BlackBerry Browser	IE Mobile
								2.1		
								2.2		
						3.2		2.3		
						4.0-4.1		3.0		
						4.2-4.3		4.0		
	8.0					5.0-5.1		4.1		
	9.0	28.0	33.0			6.0-6.1		4.2-4.3	7.0	
	10.0	29.0	34.0							
Current	11.0	30.0	35.0	7.0	22.0	7.0-7.1	5.0-7.0	4.4	10.0	10.0
Near future		31.0	36.0	8.0	23.0	8.0		4.4.3		
Farther future		32.0	37.0		24.0					
3 versions ahead		33.0	38.0							

Notes

Known issues (0)

Resources (5)

Feedback

Edit on GitHub

Partial support in IE 10 & 11 refers to a number of subfeatures not being supported. Partial support in BB10 refers to an outdated specification being implemented. Code targeting the current state of the specification might not work.

How does IndexedDB work?

Key-Value pairs

Search Indexes

NoSQL

Cursors

Asynchronous via callbacks

Notifications via DOM events

Key/Value pairs?

Key

String, Number, Date, Array

Value (incomplete list)

String

Object

Array

Blob

ArrayBuffer

Uint8Array

File

Will IndexedDB work offline?

YES!

Can also be used with Application Cache.

How to use? Six basic steps

1. Include shim (if necessary)
2. Set vendor prefixes
3. Validate functionality
4. Open (or upgrade)
5. Add, update, retrieve or delete data
6. Capture events via callbacks

How to use? IndexedDBShim.js

Required for Safari 7 on iOS and Mac

Safari 7 only comes with Web SQL

```
<script src="IndexedDBShim.js"></script>
```

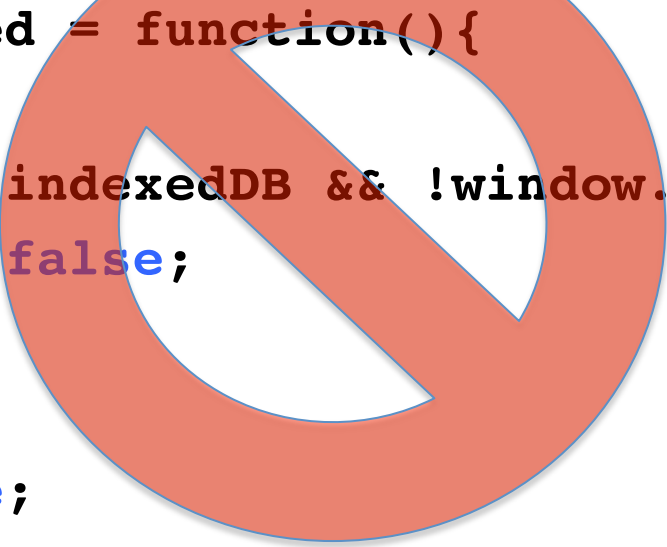
How to use? Set Prefixes

```
window.indexedDB = window.indexedDB ||  
    window.mozIndexedDB ||  
    window.webkitIndexedDB ||  
    window.msIndexedDB;
```

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


How to use? Validate functionality

```
this.isSupported = function(){  
    if(!window.indexedDB && !window.openDatabase){  
        return false;  
    }  
  
    return true;  
};
```



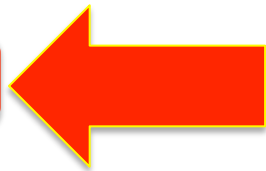
How to use? Validate functionality

```
this.isSupported = function(callback){  
  
    if(!window.indexedDB && !window.openDatabase){  
        return callback(false);  
    }  
    else{  
        testFunctionality(function(success,result){  
            return callback(success,result);  
        });  
    }  
};
```



How to use? Validate functionality

```
function testFunctionality(callback){  
    . . .  
    var objectStore = transaction.objectStore("table1");  
    var request = objectStore.put(myBlob,"test1key");  
    request.onsuccess = function(event)  
    {  
        callback(true,event.target.result);  
    };  
  
    request.onerror = function(error)  
    {  
        callback(false,error);  
    };  
}
```



How to use? Open database

```
var db = null;
```

```
var request =  
    indexedDB.open("customerDB", /*version*/ 2);
```



STEP 1

```
request.onsuccess = function(event){  
    db = event.target.result;  
    callback(true);  
}
```



STEP 2

```
request.onerror = function(error){  
    callback(false, error);  
}
```

How to use? Transactions

Handles ALL reading and writing

```
var transaction =  
    db.transaction([ "customerDB" ], "readwrite");
```

Provides events

Three modes

“readonly”

“readwrite”

“versionchange”

How to use? Transaction events

```
var transaction =  
    db.transaction([ "customerDB" ], "readwrite");  
  
transaction.onerror = function(error){ . . . }  
  
//  
// Careful! This always returns  
//  
transaction.oncomplete = function(event){  
    . . .  
}
```

How to use? Transaction requests

```
var transaction =  
    db.transaction([ "customerDB" ], "readwrite");
```



STEP 1

```
var objectStore =  
    transaction.objectStore("customerDB");
```



STEP 2

```
var request =  
    objectStore.put({test:1}, "myKey");
```



STEP 3

```
// Did transaction request work or not?  
request.onerror = function(error){ . . . }  
request.onsuccess = function(event){ . . . }
```

How to use? Write data

add(anyValue, /*optional*/ key)

Write only unique values

Duplicate entries fail with error

put(anyValue, /*optional*/ key)

Overwrites existing entries with same key

How to use? Write data

```
request =
```

```
objectStore.put({test:1}, "myKey");
```

```
request.onsuccess = function(event){  
    callback(true, event.target.result);  
}
```

```
request.onerror = function(error){  
    callback(false, error);  
}
```


How to use? Get data

```
request =
```

```
    objectStore.get("aUniqueKey");
```

```
request.onsuccess = function(event){  
    callback(true,event.target.result);  
}
```

```
request.onerror = function(error){  
    callback(false,error);  
}
```

Database size – how big??

General suggestion (smartphones/tablet):

- Less than 100 MB

Why? Depends on:

- Remaining free memory
- How much memory used by browser
- How many tabs are already open
- If other applications already running

Can I use too much memory?

YES!

The device OS will shutdown the browser

Greater potential for data corruption

- If shutdown occurs during a 'write' operation

Read/Write performance

Know thy data!

Not all data types perform equally







Pre- and post-processing is expensive

DEMO

<https://github.com/andygup/indexeddb-typetest-js>







Chrome 35.0.1916 – MacBook

319KB JPEG

Initial Type	Type	Test	Image	Elapsed Time to get()
Simple String	string	test7		2.294000005349517ms
Object	object	test8		38.40400000626687ms
Array	array	test9		24.017999996431172ms
Blob	string	test10		9.407999998074956ms
Uint8Array	uint8array	test11		1.1030000023311004ms
ArrayBuffer	arraybuffer	test12		2.9829999984940514ms
Size	318679 bytes			

Safari 7.0.3 – MacBook (Shim)







319KB JPEG

Initial Type	Type	Test	Image	Elapsed Time to get()
Simple String	string	test7		4ms
Object	object	test8		11ms
Array	array	test9		7ms
Blob	blob	test10		10ms
Uint8Array	arraybuffer	test11		210ms
ArrayBuffer	arraybuffer	test12		10ms
Size	318679 bytes			

Whoa!!


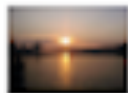
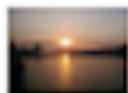

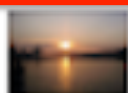

Safari 7.1.1 iPad 3 (Shim)

319KB JPEG

Initial Type	Type	Test	Image	Elapsed Time to get()
Simple String	string	test7		35ms
Object	object	test8		18ms
Array	array	test9		18ms
Blob	blob	test10		30ms
Uint8Array	arraybuffer	test11		54ms
ArrayBuffer	arraybuffer	test12		2ms
Size	318679 bytes			

Safari 8 iPhone 5S (no shim!)

319KB JPEG

Initial Type	Type	Test	Image	Elapsed Time to get()
Simple String	string	test7		27.638375000606175ms
Object	object	test8		21.705125000153203ms
Array	array	test9		63.42466666683322ms
Blob	string	test10		8.576708332839189ms
UInt8Array	uint8array	test11		13.27020833377901ms
ArrayBuffer	arraybuffer	test12		15.185874999588123ms
Size	318679 bytes			

Safari 7 iPhone 5 (Shim)







319KB JPEG

Initial Type	Type	Test	Image	Elapsed Time to get()
Simple String	string	test7		24ms
Object	object	test8		45ms
Array	array	test9		54ms
Blob	blob	test10		64ms
Uint8Array	arraybuffer	test11		973ms
ArrayBuffer	arraybuffer	test12		56ms
Size	318679 bytes			

Whoa!!






Firefox 29.0.1 Desktop – MacBook

319KB JPEG

Initial Type	Type	Test	Image	Elapsed Time to get()
Simple String	string	test7		7.043865999999998ms
Object	object	test8		28.0127150000000014ms
Array	array	test9		14.23775900000000096ms
Blob	blob	test10		10.7641889999999874ms
Uint8Array	uint8array	test11		18.1950860000000006ms
ArrayBuffer	arraybuffer	test12		23.28420900000000033ms
Size	318679 bytes			

Android 4.4.4 – Nexus 4

319KB JPEG

Initial Type	Type	Test	Image	Elapsed Time to get()
Simple String	string	test1		72.63899999816203ms
Object	object	test2		53.3489999994341284ms
Array	array	test3		53.898999999863701ms
Blob				
Uint8Array	uint8array	test5		11.01800000469666ms
ArrayBuffer	arraybuffer	test6		14.100000000325963ms

Pre- and Post-processing

What final data type does your app need?

Pre-process

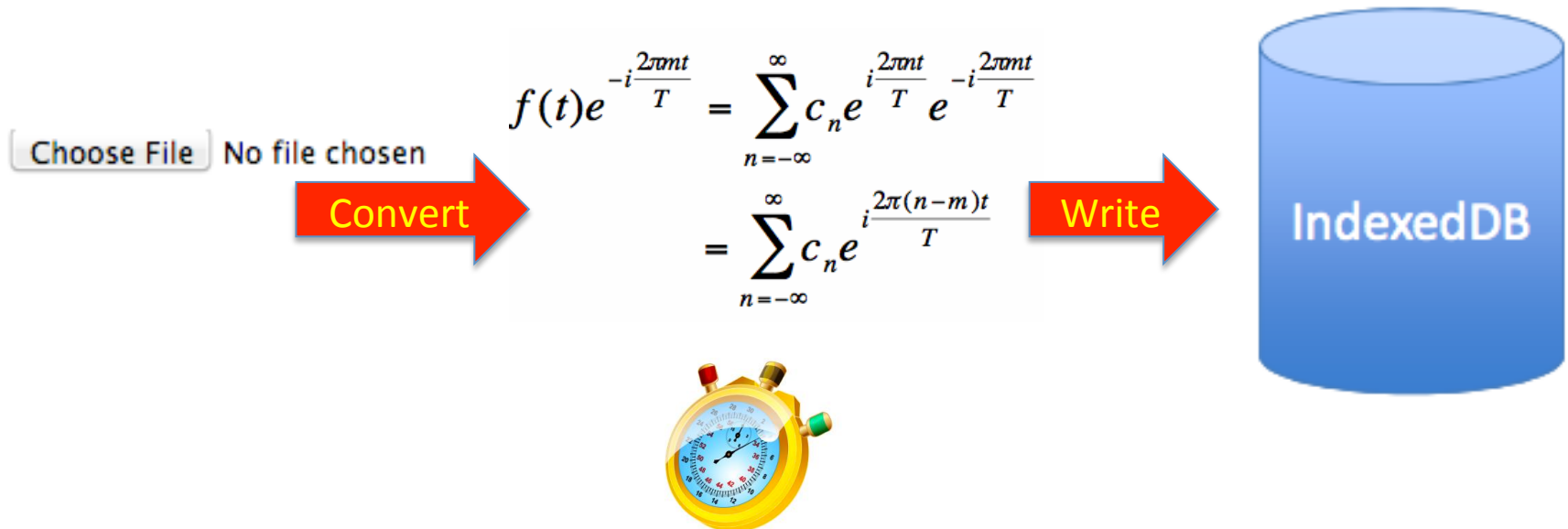
Work done before 'writing' data to DB

Post-process

Work done after 'getting' data from DB

Pre-processing

1. Retrieve data from <input> or server
2. Convert data (if needed)
3. Write to database

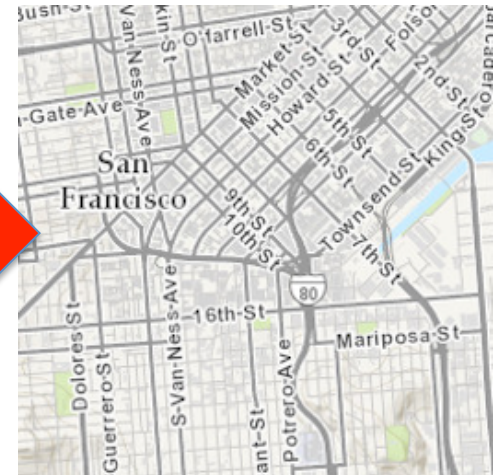


Post-processing

1. Retrieve data from database
2. Convert data (if needed)
3. Display data



```
var uInt8Array = new Uint8Array(result);  
var blob = new Blob([uInt8Array], {type: "image/png"});  
var image = document.createElement("img");  
var urlCreate = window.URL || window.webkitURL;  
var url = urlCreate.createObjectURL(blob);  
image.src = url;  
image.height = 30;  
image.onload = function(e){  
    URL.revokeObjectURL(this.src);  
}  
  
value.appendChild(image)  
break;
```



Pre- and Post-processing

XMLHttpRequest response types

Value	Response data type
""	String (Default)
"arraybuffer"	ArrayBuffer
"blob"	Blob
"document"	Document
"json"	Object
"text"	String

Pre- and Post-processing

Potentially huge performance differences

Examples

- FileReader
- Canvas
- Bitwise conversion
- Blob to objectURL

Pre- and Post-processing (Example)

```
var uInt8Array = new Uint8Array(arrBuff);  
var blob = new Blob(  
    [uInt8Array],  
    {type: "image/jpeg"});  
var image = document.createElement("img");  
var urlCreate = window.URL || window.webkitURL;  
var url = urlCreate.createObjectURL(blob);  
image.src = url;  
image.height = 30;  
image.onload = function(e) {  
    callback(image);  
    URL.revokeObjectURL(this.src);  
}
```

INPUT

OUTPUT



DEMO

<https://github.com/andygup/image-parsing-test-js>

Convert ArrayBuffer to Image*



GIF

Test	FileReader	Canvas	Bitwise
type	image/gif	image/gif	image/gif
Size	402629	402629	402629
Avg. (ms)	5.4146	34.0201	25.9631

PNG

Test	FileReader	Canvas	Bitwise
Type	image/png	image/png	image/png
Size	409317	409317	409317
Avg. (ms)	5.8122	25.5615	23.3734






JPG




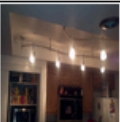

Test	FileReader	Canvas	Bitwise
Type	image/jpeg	image/jpeg	image/jpeg
Size	404699	404699	404699
Avg. (ms)	5.7540	39.6070	27.0127

* Chrome 35

Convert ArrayBuffer to Image*









Type	image_3.9mb.jpg	image_2.1mb.jpg	image_1.5mb.jpg	image_800kb.jpg	image_405kb.jpg
Size	3863500	2763687	1532129	796339	404699
FileReader	15.434999993885867	13.084999998682179	8.228999999118969	3.608999992138706	4.256000000168569
Bitwise	266.4130000048317	118.2589999953052	89.66300000611227	42.74499999883119	20.90400000452064
Canvas	370.2860000048531	202.9599999950733	188.9939999964554	114.50199999671895	45.574999996460974
Images					

Type	image_3.9mb.png	image_2.1mb.png	image_1.5mb.png	image_800kb.png	image_405kb.png
Size	3863521	2144014	1460123	799053	409317
FileReader	15.855999998166226	8.312999998452142	7.406999997328967	3.715999991982244	3.6010000039823353
Bitwise	228.63900000811554	141.73300001129974	90.87899999576621	43.76300000876654	23.245999997016042
Canvas	299.7310000064317	94.530999995186	75.66400000359863	49.11300000094343	25.83400000031665
Images					

* Chrome 35

Convert ArrayBuffer to Image

Type	image_100kb.jpg	image_58kb.jpg	image_27kb.jpg
Size	16984	11603	7332
FileReader	0.7210000039776787	0.6870000070193782	1.4530000044032931
Bitwise	1.141000000643544	0.5819999933009967	0.5549999914364889
Canvas	7.279999990714714	6.057999999029562	7.644999990588985
Images			

Type	image_100kb.png	image_58kb.png	image_27kb.png
Size	98133	57550	27332
FileReader	1.224999999976717	0.9160000045085326	0.9369999897899106
Bitwise	4.690000001573935	2.7799999952549115	1.3840000028721988
Canvas	11.365000013029203	9.47699999960605	8.04000000061933
Images			

DEMO

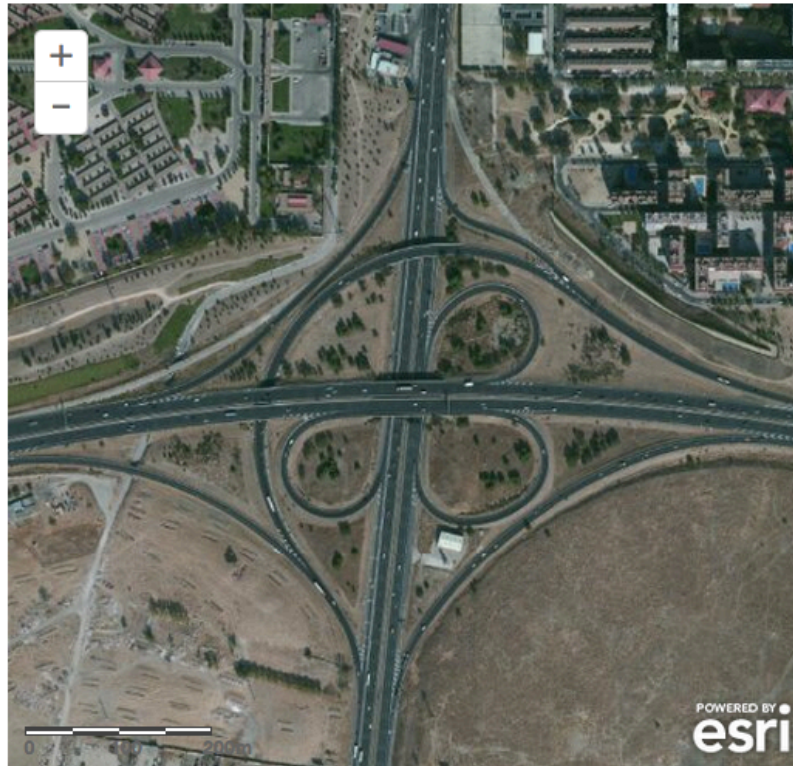
<https://github.com/Esri/offline-editor-js>

Min Zoom Level (farthest from ground)

Current Zoom Level

Max Zoom Level (closer to ground)

Level	Tile Count	Size Mb (aprox.)
13	1	0.03 Mb
14	2	0.05 Mb
15	4	0.11 Mb
16	9	0.24 Mb
17	20	0.53 Mb
18	72	1.89 Mb
19	240	6.32 Mb
Total	348	9.15 Mb

[Prepare for Offline](#)[Delete All Tiles](#)[Go Offline](#)[Usage: 4.16 Mb \(127 tiles\)](#)[Show Stored Tiles](#)[Save to File](#)[Load from File](#)[Using proxy: Yes](#)

1. Navigate to your area of interest

2. Click 'Prepare for Offline' button

3. Go Offline and enjoy!

Summary - IndexedDB

Designed to work with complex data.

High performance/Asynchronous

Compatible with many offline workflows

Support continues to improve

References

<https://github.com/andygup/indexeddb-typetest-js>

<https://github.com/andygup/image-parsing-test-js>

<https://github.com/Esri/offline-editor-js>

<https://github.com/axemclion/IndexedDBShim>

<http://developers.arcgis.com/javascript>

Questions?

Andy Gup, Esri

Sr. Developer – JS and native Android

www.andygup.net

github.com/andygup

agup@esri.com

[@agup](#)

