HTML5 APIs by @naoisegolden

Based on @soyjavi's "HTML5 WTF!"

Naoise* Golden Santos

Frontend Developer at XING

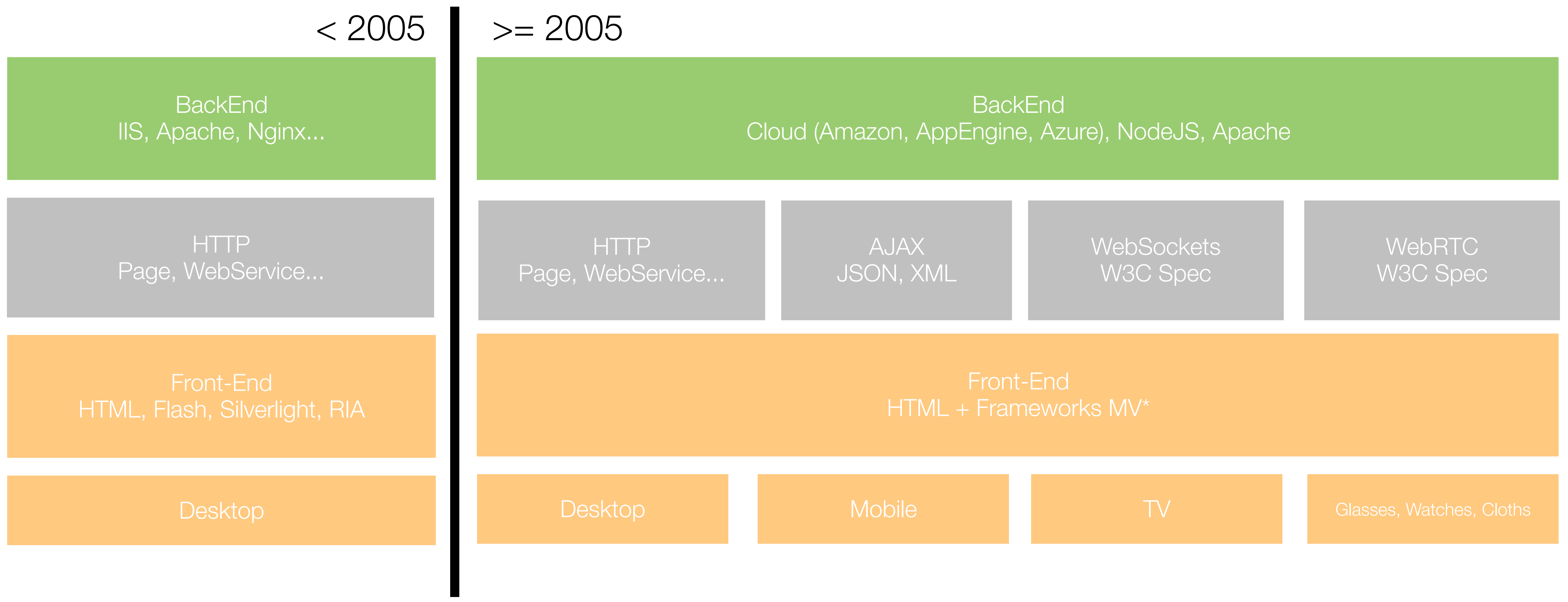
naoisegolden @ (gmail, twitter, github,…)

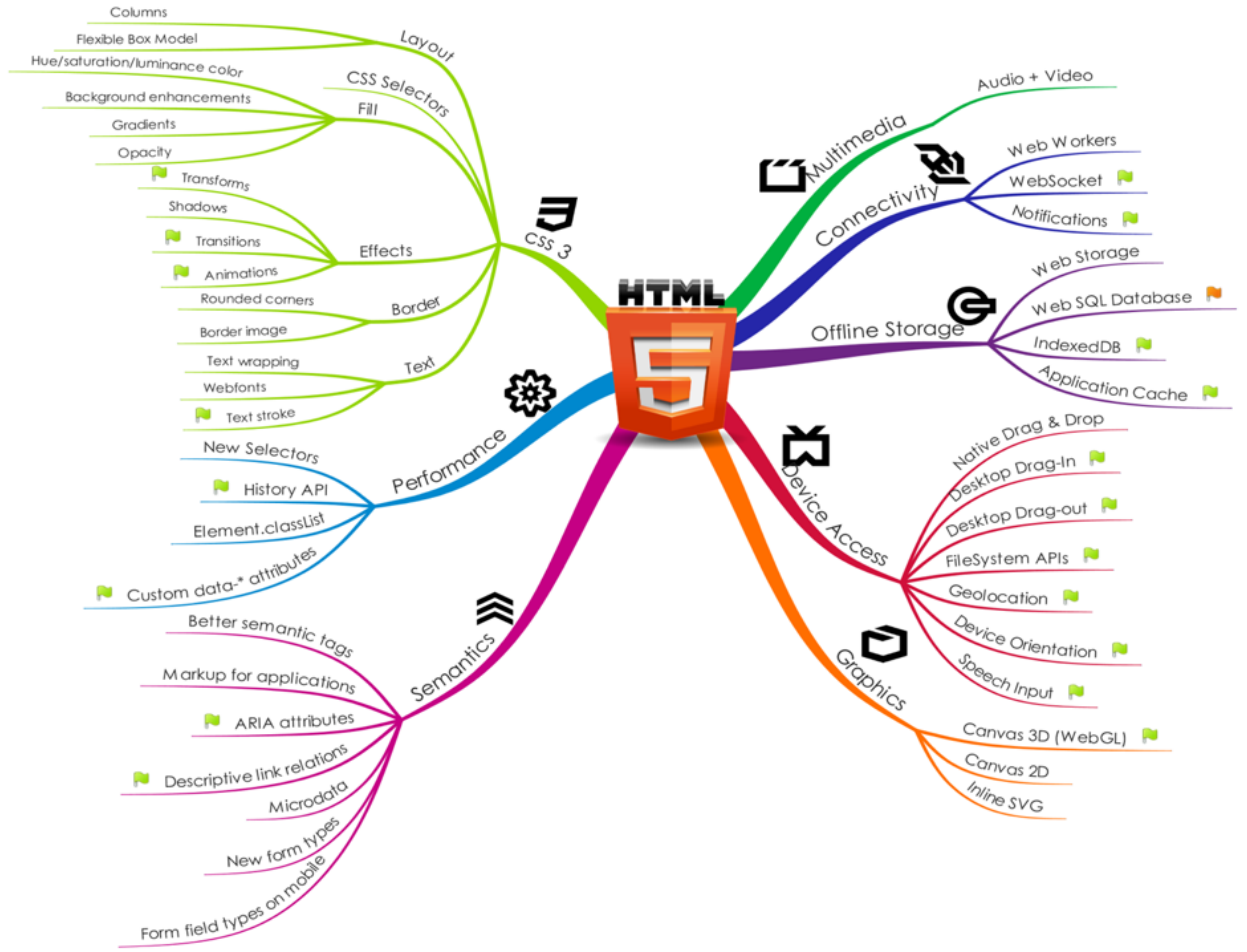
*pronounced *nisha*

Web App != Web Site

Web App != Web Site

The web project architecture has changed




HTML
CSS 3
Layout
Columns
Flexible Box Model
CSS Selectors
Fill
Hue/saturation/luminance color
Background enhancements
Gradients
Opacity
Effects
Transforms
Shadows
Transitions
Animations
Border
Rounded corners
Border image
Text
Text wrapping
Webfonts
Text stroke
Performance
New Selectors
History API
Element.classList
Custom data-* attributes
Semantics
Better semantic tags
Markup for applications
ARIA attributes
Descriptive link relations
Microdata
New form types
Form field types on mobile
Multimedia
Audio + Video
Connectivity
Web Workers
WebSocket
Notifications
Offline Storage
Web Storage
Web SQL Database
IndexedDB
Application Cache
Device Access
Native Drag & Drop
Desktop Drag-In
Desktop Drag-out
FileSystem APIs
Geolocation
Device Orientation
Speech Input
Graphics
Canvas 3D (WebGL)
Canvas 2D
Inline SVG

Device Orientation

Device Orientation `SPEC`

Many new computers, mobile phones and other devices are now have accelerometers, gyroscopes, compasses and other hardware designed to determine motion and orientation. Modern web browsers allow access to this data. One example of that are the new HTML5 DeviceOrientation and DeviceMotion events. These events provide developers with information about the orientation, motion and acceleration of the device.

http://dev.w3.org/geo/api/spec-source-orientation.html

Device Orientation `Code`

1. Check for API Support

```
if (window.DeviceOrientationEvent) {
  console.log("DeviceOrientation is supported!");
}
```

2. Subscribe to event

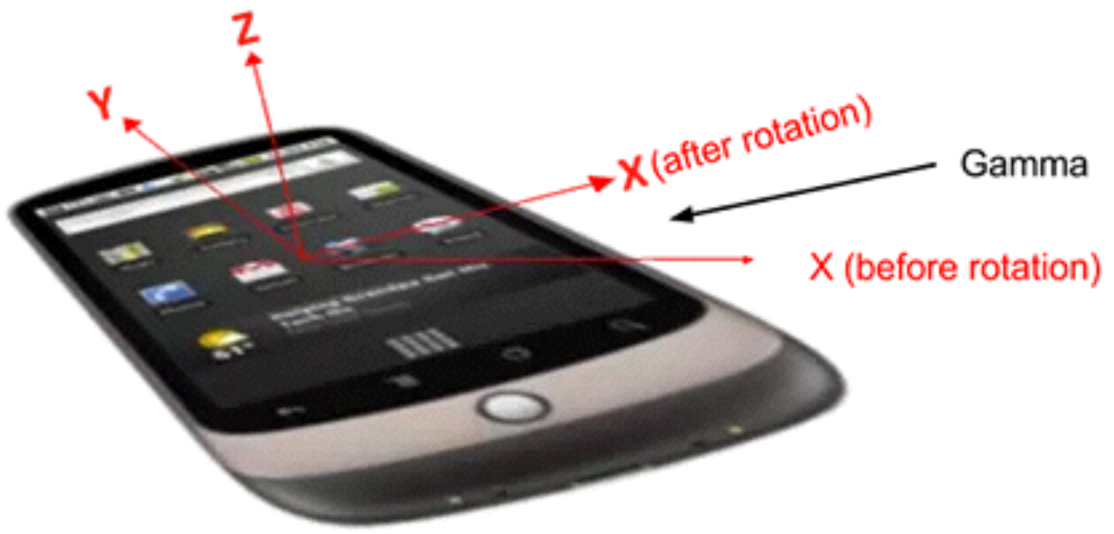
```
window.addEventListener('deviceorientation', onChange);
```

3. Handle Event

```
function onChange(eventData) {
    // gamma is the left-to-right tilt in degrees, where right is positive
    var tiltLR = eventData.gamma;
    // beta is the front-to-back tilt in degrees, where front is positive
    var tiltFB = eventData.beta;
    // alpha is the compass direction the device is facing in degrees
    var dir = eventData.alpha;
}
```

Device Orientation Hack

- Draw HTML5 logo in your Web App
- Rotate in 3D with your device's Orientation
- Move your computer/phone!



Geolocation

Geolocation Spec

The Geolocation API lets you find out where the user is and keep tabs on them as they move around, always with the user's consent.

The API is device-agnostic; it doesn't care how the browser determines location, so long as clients can request and receive location data in a standard way.

http://dev.w3.org/geo/api/spec-source

IRON HACK

Geolocation Code

1. Check for API Support

```
if (navigator.geolocation) {
  console.log("Geolocation is supported!");
}
```

2. Get current Permission

```
navigator.geolocation.getCurrentPosition(onSuccess, onError);
function onSuccess(position){
    console.log(position.coords.latitude, position.coords.longitude);
}
function onError(error){
  alert(‘Error en GPS: ‘ + error);
}
```

3. Create a watcher position

```
var options = {enableHighAccuracy: true, maximumAge:30000, timeout: 60000}
window.navigator.geolocation.watchPosition(onSuccess, onError, options);
```

Geolocation `Hack`

- Get current user's geolocation
- Draw a Google Static Map with the location

IRON HACK

Notifications

Notifications `Spec`

The Notifications API allows you to display notifications to the user for given events, both passively (new emails, tweets or calendar events) and on user interactions regardless of which tab has focus. There is draft spec but the W3C is currently working on a new SPEC.

http://www.w3.org/TR/notifications/

Notifications Code

1. Check for API Support

```
if (window.Notification) {
  console.log("Notifications are supported!");
}
```

2. Request Permisions

```
if (Notification.permission !== 'denied') {
  Notification.requestPermission(function (permission) {
    // If the user is okay, let's create a notification
    if (permission === "granted") {
      console.log(“Permission granted!”);
    }
  });
}
```

3. Create Notification

```
var notification = new Notification("Hi there!");
```

Notifications `Hack`

- Request permission for API
- Create a notification with the location
- Show a Google Static Map as an icon

WebStorage
Past, Present & Future

Why Store Data on Client?

The main reason is practicality. JavaScript code running on the browser does not necessarily need to send all information to the server. There are several use cases:

1. You want to increase **performance**. You can cache data client-side so it can be retrieved without additional server requests.

2. You have a significant quantity of **client-side-only data**, e.g. HTML strings or widget configuration settings.

3. You want you make your application work **off-line**.

Web SQL Database (past)

The Web SQL Database was an initial attempt by vendors to bring SQL-based relational databases to the browser. It has been implemented in Chrome, Safari and Opera 15+, but was opposed by Mozilla and Microsoft.

The **advantages**:

- designed for robust client-side data storage and access
- it uses SQL like many server side applications

The **disadvantages**:

- SQL never seemed appropriate for client-side development
- the W3C specification was abandoned in 2010

Web Storage (present & future)

Web Storage provides two objects with identical APIs: localStorage to retain persistent data and sessionStorage to retain session-only data which is lost when the tab is closed.

Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.

The **advantages**:

- easy to use with simple name/value pairs
- session and persistent storage options are available
- wide support on desktop and mobile browsers including IE8+

The **disadvantages**:

- string values only — serialization may be necessary
- unstructured data with no transactions, indexing or searching facilties
- may exhibit poor performance on large datasets

IRON HACK

WebStorage Code

1. Save a item

```
window.localStorage.setItem(KEY, VALUE);
```

2. Get a item

```
window.localStorage.getItem(KEY);
```

3. Remove a item

```
window.localStorage.removeItem(KEY);
```

WebStorage `Hack`

- Try out LocalStorage and Session storage and see the difference
- Store and retrieve objects

IndexedDB (Future)

IndexedDB provides a structured, transactional, high-performance NoSQL-like data store with a synchronous and asynchronous API. The API permits you to create databases, data stores and indexes, handle revisions, populate data using transactions, run non-blocking queries...

The **advantages** of IndexedDB:

- designed for robust client-side data storage and access
- Support in modern desktop browsers: IE10+, Firefox 23+, Chrome 28+

The **disadvantages**:

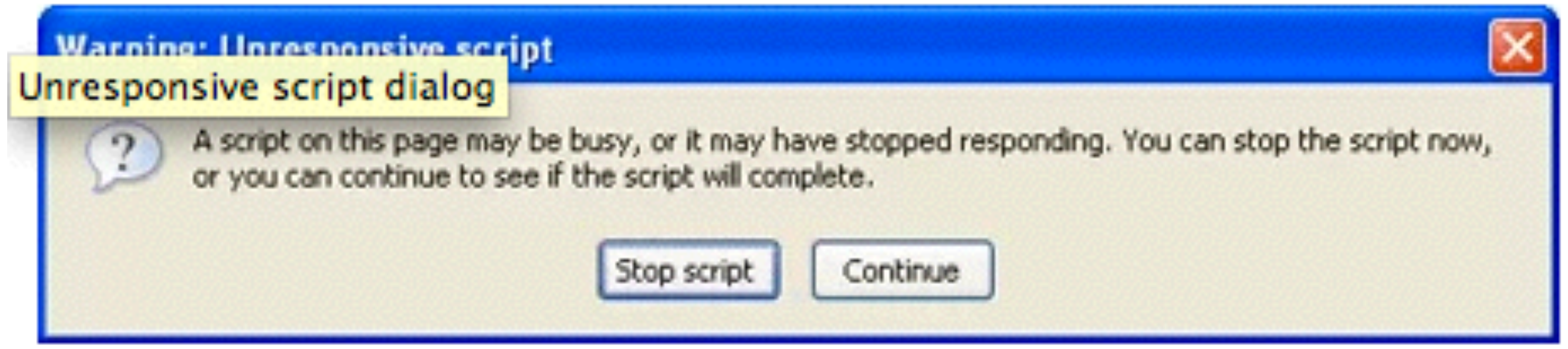
- the API is very new and subject to revision
- little support in older and mobile browsers
- like any NoSQL store, data is unstructured which can lead to integrity issues

WebWorkers

IRON HACK

WebWorkers Spec

The Web Workers specification defines an API for spawning background scripts in your web application. Web Workers allow you to do things like fire up long-running scripts to handle computationally intensive tasks, but without blocking the UI or other scripts to handle user interactions. They're going to help put and end to that nasty 'unresponsive script' dialog that we've all come to love:


Warning: Unresponsive script

Unresponsive script dialog

A script on this page may be busy, or it may have stopped responding. You can stop the script now, or you can continue to see if the script will complete.

http://www.w3.org/TR/workers/

WebWorker Code

Web Workers run in an isolated thread and the code that they execute needs to be contained in a separate file:

```
var worker = new Worker('task.js');
```

After creating the worker, register the callback for on message (2 options):

```
worker.onmessage = function(event) { /* Do Something */ };
worker.addEventListener(“message”, function(event) { /* Do Something */});
```

Then, start the worker by calling the postMessage() method:

```
worker.postMessage('data');
```

task.js

```
self.onmessage = function(event) {
  // Do some work!!!
  self.postMessage("received: " + event.data);
};
```

WebWorker `Hack`

- Create an event system to get the Google Static Map URL passing a set of coordinates.

Application Cache

Cache Spec

Developers can use the Application Cache (AppCache) interface to specify resources that the browser should cache and make available to offline users. Applications that are cached load and work correctly even if users click the refresh button when they are offline.

Using an application cache gives an application the following benefits:

- Offline browsing: users can navigate a site even when they are offline.
- Speed: cached resources are local, and therefore load faster.
- Reduced server load: the browser only downloads resources that have changed from the server.

http://www.w3.org/TR/offline-webapps/#offline

Cache Code

1. Cache resources files in the browser.

```
<html manifest="cache.appcache">
  ...
</html>
```

2. cache.appcache file:

```
CACHE MANIFEST
# version 1.0.0

CACHE:
/html5/src/logic.js
/html5/src/style.css
/html5/src/background.png

NETWORK:
*
```

Cache `Hack`

- Create a cache manifest and specify static assets

Websockets

Websockets Spec

WebSockets is an advanced technology that makes it possible to open an interactive communication session between the user's browser and a server. With this API, you can send messages to a server and receive event-driven responses without having to poll the server for a reply.

Example: collaborative text editing

Uses ws:// protocol

http://www.w3.org/TR/websockets/

Websockets Code

1. Check functionality

```
if(window.WebSocket) { … }
```

2. Create a communication with localhost

```
var ws = new WebSocket('ws://localhost:3000');

ws.onopen = function() { alert('Connected'); };

ws.onclose = function() { alert('Disconnected'); };

ws.onmessage = function(e) { alert('Message received: ' + e.data); };

ws.send('Hi there, websockets!');
```

Websockets `Hack`

- Try out web sockets in localhost
- Create a push notification system when somebody is geolocalised

Forms

Forms Spec

Not many people get excited about forms, but HTML5 brings some big improvements, both for the developers creating them and for the users filling them out. New form elements, attributes, input types, browser-based validation, CSS3 styling techniques, and the FormData object make it easier and hopefully more enjoyable to create forms.

http://www.w3.org/TR/html5/forms.html

Forms Code

New Elements

```
<progress /> /*Represents completion of a task.*/

<meter /> /*Represents a scalar measurement within a known range..*/

<datalist /> /*Represents a set of option elements.*/

<keygen /> /*A control for key-pair generation.*/

<output /> /*Displays the results of a calculation.*/
```

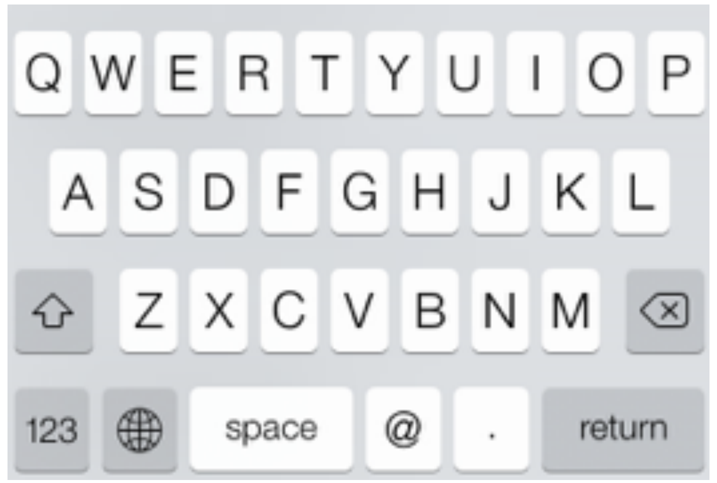
Example

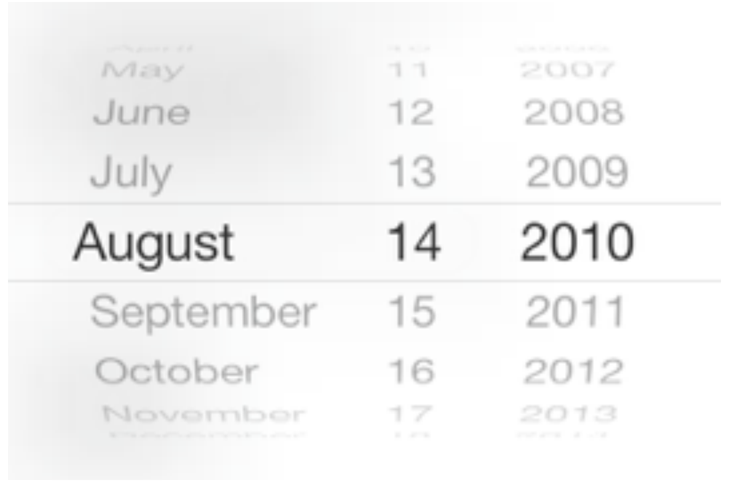
```
<input type="text" list="sources" />

<datalist id="sources">
   <option>Professor</option>
   <option>Master</option>
</datalist>
```

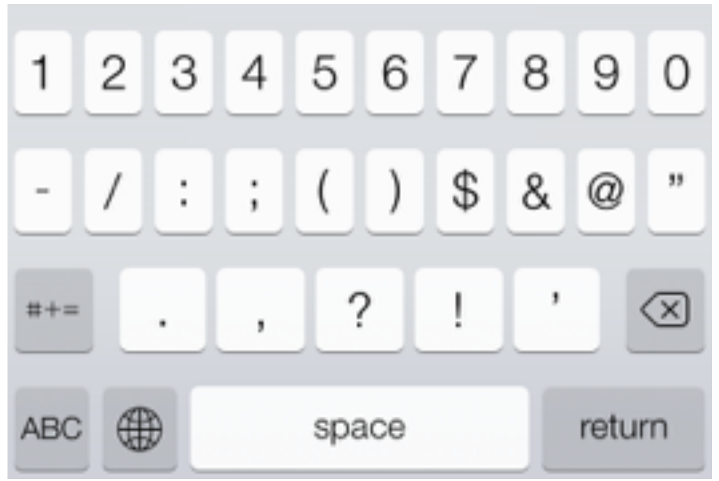
Forms Code

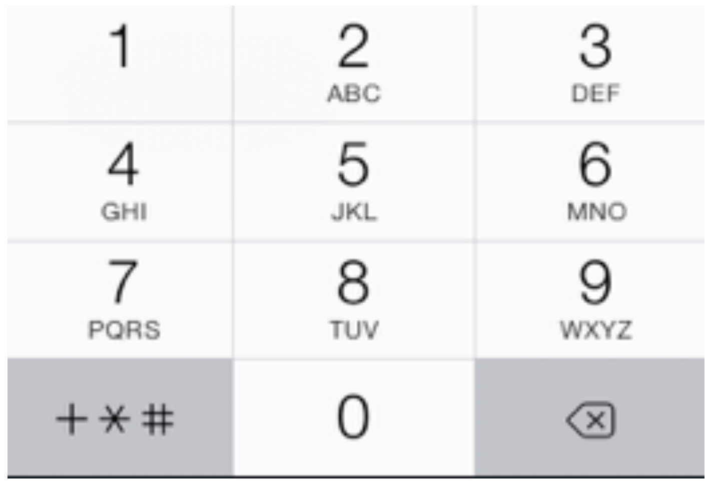
HTML5 introduces 13 new input types. When viewed in a browser that doesn't support them, these input types fall back to text input.

```
tel
search
url
email
datetime
date
month
week
time
datetime-local
number
range
color
speech
```









Forms Code

HTML5 also introduces several new attributes for the input and form elements.

```
autofocus
placeholder
form
required
autocomplete
pattern
dirname
novalidate
formaction
formenctype
formmethod
formnovalidate
```

Example

```
<input type="search" placeholder="Search..." required />
```

Forms Code

With the FormData object, you can create and send a set of key/value pairs and, optionally, files using XMLHttpRequest. When using this technique, the data is sent in the same format as if you'd submitted it via the form's submit() method with the encoding type of multipart/form-data.

```
var formData = new FormData();
formData.append("part_num", "123ABC");
formData.append("part_price", 7.95);
formData.append("part_image", somefile)

var xhr = new XMLHttpRequest();
xhr.open("POST", "http://some.url/");
xhr.send(formData);
```

You can also use FormData to add additional data to an existing form:

```
var formElement = document.getElementById("someFormElement");
var formData = new FormData(formElement);
formData.append("part_description", "The best part ever!");

var xhr = new XMLHttpRequest();
```

Forms Hack

- Create a form with required attributes
- Create two “regexp” inputs: ZIP code and URL
- Create a search input using “speech to text” feature