Tutorial 1: What can you do now?

We're not expecting anyone to be able to just start building HTML5 apps, but it is worthwhile starting off with this small real-world exercise, which is about finding out where your starting position is.

The Task: Build an egg-timer

Technically this is not a big job, but it can incorporate as much (or as little) of your prior knowledge as you like. There is one feature of Javascript that you need to know to get the core of this working, which is the setTimeout() function. It operates like this:

```
//***  setTimeout( <A Javascript function>, a number of milliSeconds); ***
// The nominated Javascript function will now be called after the given
// number of milliSeconds.  It will return a timer identifier so that you can
// later cancel the timeout.  For example…
function timesUp(){
    alert("Time's up!");
}
var t = setTimeout(timesUp, 30000);          // Alert will show in 30sec.
// unless this is called before then…
function cancel(){
    clearTimeout(t);
}
```

There are limitations to setTimeout() – it is all or nothing, in that having called it, the only options are to wait for it or cancel it, and having started, you have no way of knowing how far through the time period you are. Because of this, developers often use a slightly different function:

```
//*** setInterval(<A Javascript function>, <milliSeconds>); ***
// In this case, the function will be called repeatedly at the given interval.
// To make that into a countdown timer….
var seconds, timer;
function countdown(){
    seconds = seconds – 1;
    if(seconds === 0) {
        alert("Time's up");
    }
    // We can do whatever we like here – typically, display the
    // time left on a web-page, something like…
    document.getElementById("count").innerHTML = seconds + "S";
}

// Now to make a 30 second countdown…
function startCountdown(){
    seconds = 30;
    timer = setInterval(countdown, 1000);   // 1 second intervals
}
function cancel(){
    clearInterval(timer);
}
```

In this tutorial, you should work on the **design** of a web-application that uses these Javascript features to create an egg timer. Obviously there is a range of levels of polish you can add to this,

depending on your background experience etc., but you can expect help in getting the Javascript code tied up with any HTML-based user-interface (if you need it). If you want to use graphics and CSS to make it look polished, that's fine, but primarily your job is to figure out how you would put a web app together to do this job. Note that with a few very minor changes (adding a manifest to the HTML of the web app), you would also be able to deploy this as an app for your mobile phone (provided it is reasonably up to date and has a web browser).

Some things to consider…

1. How will you package this app? It could all be done in a single HTML file with the script (and CSS if you're using it) embedded, but is this the best approach?
2. What will the user-interface need to include. If your timer is always a 30 second timer, you could get away with a blank page that pops up an alert box, but to make a fairly functional timer, it might need more than that. For example:
    a. Say I wanted a timer to time any cooking process – I would want to be able to set the amount of time.
    b. What if I wanted to time a number of things simultaneously, I might need to set more than one timeout.
    c. Is an alert() box the best way to attract the user's attention?
3. Some of the new HTML5 mark-up may work very nicely for this – e.g. you can define a slider as an input control to indicate the number of seconds or minutes (<input type="range" min="1" max="10">). Ideally, you would want to also put the time period that had been selected by attaching a function to the control's onchange event, something like:

```
<input type="range" id="range" min="1" max="10" value="1"><span id="value">1</span>
… and then in script …
document.getElementById("range").onclick = function(){
    document.getElementById("value").textContent = this.value;
}
```

4. Usually simple web-apps succumb to 'feature creep', which describes when an application that was once very specific in its objectives grows all sorts of new facilities as new versions appear (eventually to become so bloated that people stop using them). Try to think of a number of features you could legitimately add to this application (obviously once you got the core functionality working).

It would be useful if you could prepare at least some of this app for next week – we can use the tutorial space to get any remaining questions you have answered. Next week's tutorial will use a similar core feature to what this app does as the next target.