Timer-Based Animation

Jerry Cain
CS 106J
April 24, 2017

Timer Events

- The programs you saw on Friday responded to mouse events by adding an event listener to the GWindow object.
- JavaScript also allows you to listen for timer events, which occur after a specified time interval.
- As with mouse events, you specify the listener for a timer event in the form of a callback function that is automatically invoked at the end of the time interval.
- You can add animation to a JavaScript program by setting a timer for a short interval and having the callback function make small updates to the graphical objects in the window.
- If the time interval is short enough (typically between 20 and 30 milliseconds), the animation will appear smooth to the human eye.

Timeouts

- JavaScript supports two kinds of timers. A one-shot timer invokes its callback function once after a specified delay. You create a one-shot timer by calling

  ```javascript
  setTimeout(function, delay);
  ```

  where `function` is the callback function and `delay` is the time interval in milliseconds.

- An interval timer invokes its callback function repeatedly at regular intervals. You create an interval timer by calling

  ```javascript
  setInterval(function, delay);
  ```

  The `setInterval` function returns a numeric value that you can later use to stop the timer by calling `clearTimeout` with that numeric value as an argument.

A Simple Example of Animation

```javascript
function AnimatedSquare() {
  var gw = GWindow(GWINDOW_WIDTH, GWINDOW_HEIGHT);
  var dw = (gw.getWidth() - SQUARE_SIZE) / N_STEPS;
  var dy = (gw.getHeight() - SQUARE_SIZE) / N_STEPS;
  var square = GRect(0, 0, SQUARE_SIZE, SQUARE_SIZE);
  square.setFillStyle(0);
  gw.add(square);
  var stepCount = 0;
  var step = function() {
    square.move(dw, dy);
    stepCount++;
    if (stepCount == N_STEPS) setTimeout(timer);
  }
  gw.add(step());
  gw.add(stepCount);
  gw.add(step);
  gw.add(step);
}
```

A Simple Example of Animation