

**فریم ورک تست جاوا اسکریپت QUnit:** انتخاب و استفاده از یک فریم ورک برای تست کدهای جاوا اسکریپت، قطعاً نتیجه بهتری را به همراه خواهد داشت. من در این جا از [QUnit](#) که یکی از بهترین‌های تست واحد است، استفاده می‌کنم. برای این کار فایل‌های qunit.css و qunit.js را دانلود و مانند زیر برای تست واحد آماده کنید:

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Refactored date examples</title>

  <link rel="stylesheet" href="../qunit.css">
  <script src="../qunit.js"></script>
  <script src="prettydate.js"></script>

  <script>
    test("prettydate basics", function() {
      var now = "2013/01/28 22:25:00";
      equal(prettyDate(now, "2013/01/28 22:24:30"), "just now");
      equal(prettyDate(now, "201308/01/28 22:23:30"), "1 minute ago");
      equal(prettyDate(now, "2013/01/28 21:23:30"), "1 hour ago");
      equal(prettyDate(now, "2013/01/27 22:23:30"), "Yesterday");
      equal(prettyDate(now, "2013/01/26 22:23:30"), "2 days ago");
      equal(prettyDate(now, "2012/01/26 22:23:30"), undefined);
    });
  </script>
</head>
<body>

<div id="qunit"></div>

</body>
</html>
```

در کد بالا ابتدا فایل‌های فریم ورک و فایل prettydate.js را اضافه کردیم. برای نمایش نتیجه تست، یک تگ div با نام qunit در بین تگ body اضافه می‌کنیم.

#### تابع test:

این تابع برای تست توابع نوشته شده، استفاده می‌شود. ورودی‌های این تابع، یکی عنوان تست و دومی یک متود دیگر، به عنوان ورودی دریافت می‌کند که در آن بدنه تست نوشته می‌شود.

#### تابع equal:

اولین تابع برای سنجش تست واحد equal است و در آن، تابعی که می‌خواهیم تست کنیم با مقدار خروجی آن مقایسه می‌شود. فایل را با نام test.htm ذخیره و آن را در مرورگر خود باز نمایید. خروجی در شکل آورده شده است:

# Prettydate tests

Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_6\_8) AppleWebKit/535.11  
(KHTML, like Gecko) Chrome/17.0.963.56 Safari/535.11

1. prettydate basics (0, 6, 6)

Tests completed in 26 milliseconds.  
6 tests of 6 passed, 0 failed.

همین طور که در تصویر بزرگ می‌بینید اطلاعات مرورگر، زمان تکمیل تست و تعداد تست، تعداد تست پاس شده و تعداد تست شکست خورده، نشان داده شده است.

# Prettydate tests

Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_6\_8) AppleWebKit/535.11  
(KHTML, like Gecko) Chrome/17.0.963.56 Safari/535.11

## 1. prettydate basics (1, 5, 6)

1. undefined, expected: "just now"
2. undefined, expected: "1 minute ago"
3. undefined, expected: "1 hour ago"
4. undefined, expected: "Yesterday"
5. undefined, expected: "2x days ago" result: "2 days ago", diff: "2x  
"2 days ago"
6. undefined, expected: undefined

Tests completed in 27 milliseconds.  
5 tests of 6 passed, 1 failed.

اگر یکی از تست‌ها با شکست روبرو شود رنگ پس زمینه قرمز و جزئیات شکست نمایش داده می‌شوند.

**بهینه سازی، مرحله اول:**

در حال حاضر تست ما کامل نیست زیرا امکان تست n weeks ago یا تعداد هفته پیش میسر نیست. قبل از آنکه این را به آزمون اضافه کنیم، تغییراتی در تست می‌دهیم

```
test("prettydate basics", function() {
  function date(then, expected) {
    equal(prettyDate("2013/01/28 22:25:00", then), expected);
  }
  date("2013/01/28 22:24:30", "just now");
  date("2013/01/28 22:23:30", "1 minute ago");
  date("2013/01/28 21:23:30", "1 hour ago");
  date("2013/01/27 22:23:30", "Yesterday");
  date("2013/01/26 22:23:30", "2 days ago");
  date("2012/01/26 22:23:30", undefined);
});
```

تابع prettyDate را در تابع دیگری به نام date قرار می‌دهیم. این تغییر سبب می‌شود تا امکان مقایسه زمان ورودی تست جاری با تست قبلی فراهم شود.

## تست دستکاری عناصر DOM:

تا اینجا با تست توابع آشنا شدید، حالا می‌خواهیم تغییراتی در prettyDate دهیم تا امکان انتخاب عناصر DOM و به روزرسانی آن نیز وجود داشته باشد. فایل prettyDate2.js در زیر آورده شده است:

```
var prettyDate = {
  format: function(now, time){
    var date = new Date(time || "");
    diff = ((new Date(now)).getTime() - date.getTime()) / 1000,
    day_diff = Math.floor(diff / 86400);

    if ( isNaN(day_diff) || day_diff < 0 || day_diff >= 31 )
      return;

    return day_diff === 0 && (
      diff < 60 && "just now" ||
      diff < 120 && "1 minute ago" ||
      diff < 3600 && Math.floor( diff / 60 ) +
        " minutes ago" ||
      diff < 7200 && "1 hour ago" ||
      diff < 86400 && Math.floor( diff / 3600 ) +
        " hours ago" ) ||
    day_diff === 1 && "Yesterday" ||
    day_diff < 7 && day_diff + " days ago" ||
    day_diff < 31 && Math.ceil( day_diff / 7 ) +
      " weeks ago";
  },
  update: function(now) {
    var links = document.getElementsByTagName("a");
    for ( var i = 0; i < links.length; i++ ) {
      if ( links[i].title ) {
        var date = prettyDate.format(now, links[i].title);
        if ( date ) {
          links[i].innerHTML = date;
        }
      }
    }
  }
};
```

prettyDate شامل دو تابع، یکی format که weeks ago به آن اضافه گردیده و تابع update که با انتخاب تگ‌ها، مقدار title را به تابع فرمت و خروجی آن را در HTML هر عنصر قرار می‌دهد. حال یک تست واحد می‌نویسیم:

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Refactored date examples</title>
  <link rel="stylesheet" href="../qunit.css">
  <script src="../qunit.js"></script>
  <script src="prettydate2.js"></script>
  <script>
    test("prettydate.format", function() {
      function date(then, expected) {
        equal(prettyDate.format("2013/01/28 22:25:00", then),
          expected);
      }
      date("2013/01/28 22:24:30", "just now");
      date("2013/01/28 22:23:30", "1 minute ago");
      date("2013/01/28 21:23:30", "1 hour ago");
      date("2013/01/27 22:23:30", "Yesterday");
      date("2013/01/26 22:23:30", "2 days ago");
      date("2012/01/26 22:23:30", undefined);
    });

    function domtest(name, now, first, second) {
      test(name, function() {
        var links = document.getElementById("qunit-fixture")
          .getElementsByTagName("a");
        equal(links[0].innerHTML, "January 28th, 2013");
        equal(links[2].innerHTML, "January 27th, 2013");
        prettyDate.update(now);
        equal(links[0].innerHTML, first);
        equal(links[2].innerHTML, second);
      });
    }
  </script>
</head>
</html>
```

```

    }
    domtest("prettyDate.update", "2013-01-28T22:25:00Z",
        "2 hours ago", "Yesterday");
    domtest("prettyDate.update, one day later", "2013/01/29 22:25:00",
        "Yesterday", "2 days ago");
    </script>
</head>
<body>

<div id="qunit"></div>
<div id="qunit-fixture">

<ul>
  <li id="post57">
    <p>blah blah blah...</p>
    <small>
      Posted <span>
        <a href="/2013/01/blah/57/" title="2013-01-28T20:24:17Z">
          >January 28th, 2013</a>
        </span>
        by <span><a href=""></a></span>
      </small>
    </li>
    <li id="post57">
      <p>blah blah blah...</p>
      <small>
        Posted <span>
          <a href="/2013/01/blah/57/" title="2013-01-27T22:24:17Z">
            >January 27th, 2013</a>
          </span>
          by <span><a href=""></a></span>
        </small>
      </li>
    </ul>

</div>

</body>
</html>

```

همین طور که مشاهده می‌کنید در تست واحد اول خود تابع `prettyDate.format` را تست نموده ایم. در تست بعدی عناصر DOM نیز دستکاری و تست شده است. تابع `domtest` با جستجوی تگ `qunit-fixture` و تگ‌های `a` درون آن، مقدار نهایی `html` آن با مقدار داده شده، مقایسه شده است.

## Refactored date examples

- noglobals
- notrycatch

☐ Hide passed tests

Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)  
 Chrome/28.0.1500.71 Safari/537.36

Tests completed in 268 milliseconds.  
 12 tests of 14 passed, 2 failed.

1. prettydate.format (0, 6, 6) Rerun

2. prettyDate.update (0, 4, 4) Rerun

3. prettyDate.update, one day later (2, 2, 4) Rerun

1. okay

2. okay

3. failed

Expected: "Yesterday"

Result: "22 hours ago"

Diff: "Yesterday" "22 hours ago"

Source: at Object.<anonymous>

4. failed

Expected: "2 days ago"

Result: "Yesterday"

Diff: "2 days ago" "Yesterday"

Source: at Object.<anonymous>

در شکل بالا نتیجه تست واحد نشان داده شده است.