实战：仅用18行JavaScript构建一个倒数计时器

爱前端不爱恋爱
微信公众号：web前端学习圈，关注领取全套50G前端系统教程
关注她

有时候，你会需要构建一个JavaScript倒计时时钟。你可能会有一个活动、一个销售、一个促销或一个游戏。你可以用原生的JavaScript构建一个时钟，而不是去找一个插件。尽管有很多很棒的时钟插件，但是使用原生JavaScript可以带来以下好处：

- 你的代码将是轻量级的，因为它将具有零依赖性。
- 你的网站将表现得更好。你不需要加载外部脚本和样式表。
- 你将拥有更多的控制权。你将会建立一个完全按照你的意愿来表现的时钟。

所以，废话不多说，下面是如何在短短的18行JavaScript中制作自己的倒计时钟。

0.演示和代码

本示例代码和演示地址：https://coding.zhangbing.site

1.基本时钟：倒数到特定的日期或时间

以下是创建基本时钟所需步骤的简要概述：

- 设置有效的结束日期。
- 计算剩余时间。
- 将时间转换为可用格式。
- 将时钟数据输出为可重复使用的对象。
- 在页面上显示时钟，并在时钟为零时停止时钟。

首先，你需要设置一个有效的结束日期。这应该是JavaScript的 Date.parse() 方法可以理解的任何格式的字符串。例如：

ISO 8601格式：

```
const deadline = '2015-12-31';
```

简短格式：

```
const deadline = '31/12/2015';
```

或者，长格式：

```
const deadline = 'December 31 2015';
```

这些格式中的每一种都允许你指定一个准确的时间和一个时区（或者在ISO日期的情况下指定一个与UTC的偏移）。例如：

```
const deadline = 'December 31 2015 23:59:59 GMT+0200';
```

3.计算剩余时间

下一步是计算剩余时间。我们需要编写一个函数，该函数需要一个表示给定结束时间的字符串（如上所述）。然后，我们计算该时间与当前时间之间的时差。看起来像这样：

```
function getTimeRemaining(endtime){
  const total = Date.parse(endtime) - Date.parse(new Date());
  const seconds = Math.floor( (total/1000) % 60 );
  const minutes = Math.floor( (total/1000/60) % 60 );
  const hours = Math.floor( (total/(1000*60*60)) % 24 );
  const days = Math.floor( total/(1000*60*60*24) );

  return {
    total,
    days,
    hours,
    minutes,
    seconds
  };
}
```

首先，我们要创建一个变量 `total` 以保留到截止日期为止的剩余时间。 `Date.parse()` 函数将时间字符串转换为毫秒值，这样我们就可以将两次相减，得到中间的时间量。

```
const total = Date.parse(endtime) - Date.parse(new Date());
```

4.将时间转换为可用格式

现在我们要将毫秒转换为天，小时，分钟和秒。让我们以秒为例：

```
const seconds = Math.floor( (t/1000) % 60 );
```

让我们分解一下这里发生的事情。

1. 将毫秒除以1000可转换为秒： `(t/1000)`
2. 将总秒数除以60，然后取余数。你不需要所有的秒，只需要计算分钟数后剩下的那些：
   `(t/1000) % 60`

重复此逻辑，将毫秒转换为分钟，小时和天。

5.将时钟数据输出为可重复使用的对象

在准备好几天，几小时，几分钟和几秒钟之后，我们现在可以将数据作为可重复使用的对象返回：

```
return {
  total,
  days,
  hours,
  minutes,
  seconds
};
```

这个对象允许你调用你的函数，并获得任何计算值。这是如何获取剩余分钟数的示例：

```
getTimeRemaining(deadline).minutes
```

方便吧?

6.在页面上显示时钟，并在时钟为零时停止时钟

现在我们有了一个可以吐出剩余天数、小时、分钟和秒数的函数，我们就可以建立我们的时钟了。首先，我们将创建以下HTML元素来保存时钟：

```
<div id="clockdiv"></div>
```

然后，我们将编写一个函数，在新的div中输出时钟数据：

```
function initializeClock(id, endtime) {
  const clock = document.getElementById(id);
  const timeinterval = setInterval(() => {
    const t = getTimeRemaining(endtime);
    clock.innerHTML = 'days: ' + t.days + '<br>' +
                      'hours: '+ t.hours + '<br>' +
                      'minutes: ' + t.minutes + '<br>' +
                      'seconds: ' + t.seconds;
    if (t.total <= 0) {
      clearInterval(timeinterval);
    }
  },1000);
}
```

该函数有两个参数，这两个参数是包含时钟的元素的id，以及倒计时的结束时间。在函数内部，我们将声明一个 `clock` 变量，并使用它来保存对时钟容器div的引用。这意味着我们不必一直查询DOM。

接下来，我们将使用 `setInterval` 每秒执行一个匿名函数。此功能将执行以下操作：

- 计算剩余时间。
- 将剩余时间输出到我们的div。
- 如果剩余时间为零停止计时。

此时，剩下的唯一步骤是像这样运行时钟：

```
initializeClock('clockdiv', deadline);
```

恭喜你！现在，你仅用18行JavaScript就拥有了一个基本时钟。

7.准备展示你的时钟

在设置时钟样式之前，我们需要进行一些改进。

- 消除初始延迟，使你的时钟立即显示。
- 让时钟脚本更有效率，这样它就不会连续重建整个时钟。
- 根据需要添加前导零。

7.1 消除初始延迟

在时钟中，我们使用 `setInterval` 每秒更新一次显示。多数情况下，这很好，除非在开始时会有一秒钟的延迟。要消除此延迟，我们必须在间隔开始之前更新一次时钟。

让我们将传递给 `setInterval` 的匿名函数移到其自己的单独函数中，我们可以将此函数命名为 `updateClock` 。在 `setInterval` 外调用一次 `updateClock` 函数，然后在 `setInterval` 内再次调用。

在你的JavaScript中，替换这个

```
const timeinterval = setInterval(() => { ... },1000);
```

新代码

```
function updateClock(){
  const t = getTimeRemaining(endtime);
  clock.innerHTML = 'days: ' + t.days + '<br>' +
                    'hours: '+ t.hours + '<br>' +
                    'minutes: ' + t.minutes + '<br>' +
                    'seconds: ' + t.seconds;
  if (t.total <= 0) {
    clearInterval(timeinterval);
  }
}

updateClock(); // 首先运行一函数能以避免延迟
var timeinterval = setInterval(updateClock,1000);
```

7.2 避免不断重建时钟

我们需要使时钟脚本更高效，我们只想更新时钟中的数字，而不是每秒重建整个时钟。实现此目的的一种方法是将每个数字放在 `span` 标签内，并仅更新这些 `span` 的内容。

这是HTML：

```
<div id="clockdiv">
    Days: <span class="days"></span><br>
    Hours: <span class="hours"></span><br>
    Minutes: <span class="minutes"></span><br>
    Seconds: <span class="seconds"></span>
</div>
```

现在让我们参考这些元素。在定义 `clock` 变量的位置之后添加以下代码

```
const daysSpan = clock.querySelector('.days');
const hoursSpan = clock.querySelector('.hours');
const minutesSpan = clock.querySelector('.minutes');
const secondsSpan = clock.querySelector('.seconds');
```

接下来，我们需要修改 `updateClock` 函数，使其只更新数字。新的代码是这样的：

```
    daysSpan.innerHTML = t.days;
    hoursSpan.innerHTML = t.hours;
    minutesSpan.innerHTML = t.minutes;
    secondsSpan.innerHTML = t.seconds;

    ...
}
```

7.3 添加前导0

现在时钟不再每秒都在重建，我们还有另一件事要做：添加前导零。例如，不是让时钟显示7秒，而是显示07秒。一种简单的方法是在一个数的开头加上一串“0”，然后切掉最后两个数字。

例如，要在“seconds”值上添加前导零，你可以更改以下设置：

```
secondsSpan.innerHTML = t.seconds;
```

为

```
secondsSpan.innerHTML = ('0' + t.seconds).slice(-2);
```

如果你愿意，你也可以在分钟和小时的前面加零。如果你已经走到这一步，恭喜你！你的时钟现在已经可以显示了。

8.更进一步

以下示例演示了如何为某些用例扩展时钟。它们都是基于上面的基本例子。

8.1 自动调节时钟

假设我们想让时钟在特定的日子出现，而不是在其他的日子。例如，我们可能有一系列事件即将发生，而不希望每次都手动更新时钟。以下是如何提前安排事情的方法。

通过在CSS中将其 `display` 属性设置为 `none` 来隐藏时钟，然后将以下内容添加到 `initializeClock` 函数中（以 `var clock` 开头的行之后）。这将导致只有在调用 `initializeClock` 函数后才会显示时钟：

```
clock.style.display = 'block';
```

接下来，我们可以指定显示时钟的日期。这将替换截止日期变量（ `deadline` ）：

```
const schedule = [
    ['Jul 25 2015', 'Sept 20 2015'],
    ['Sept 21 2015', 'Jul 25 2016'],
    ['Jul 25 2016', 'Jul 25 2030']
];
```

`Schedule` 数组中的每个元素代表一个开始日期和一个结束日期。如上所述，它可以包含时间和时区，但我在这里使用了普通的日期，以保持代码的可读性。

最后，当用户加载页面时，我们需要检查是否在指定的时间范围内。此代码应替换先前对 `initializeClock` 函数的调用：

```
// 遍历schedule中的每个元素
schedule.forEach(([startDate, endDate]) => {
  // 以毫秒为单位放置日期以便于比较
  const startMs = Date.parse(startDate);
  const endMs = Date.parse(endDate);
```

```
  // 如果当前日期在开始日期和结束日期之间，则显示时钟
  if (endMs > currentMs && currentMs >= startMs ) {
    initializeClock('clockdiv', endDate);
  }
});
```

现在，你可以提前安排你的时钟，而不必手动更新它。如果你愿意，你可以缩短代码。为了便于阅读，我把我的代码写得很啰嗦。

8.2 从用户到达起将计时器设置为10分钟

用户到达或开始特定任务后，有必要在给定的时间内设置倒计时。我们将在此处将计时器设置为10分钟，但是你可以使用任意时间。

我们需要做的就是用以下命令替换 `deadline` 变量：

```
const timeInMinutes = 10;
const currentTime = Date.parse(new Date());
const deadline = new Date(currentTime + timeInMinutes*60*1000);
```

这段代码以当前时间为基准，增加10分钟。这些值将转换为毫秒，因此可以将它们加在一起并变成新的截止日期。

现在我们有一个时钟，从用户到达时开始倒计时十分钟，你可以自由发挥，尝试不同的时间长度。

8.3 跨页面保持时钟进度

有时，除了当前页面外，还需要保留时钟状态。如果我们想在整个网站上设置10分钟的计时器，则我们不希望在用户转到其他页面时重置该计时器。

一个解决方案是将时钟的结束时间保存在一个cookie中。这样一来，导航到一个新的页面就不会把结束时间重置到十分钟以后。

这是逻辑：

1. 如果Cookie中记录了截止日期，使用该截止日期。
2. 如果不存在Cookie，请设置一个新的截止日期并将其存储在Cookie中。

要实现这一点，请使用以下命令替换 `deadline` 变量：

```
let deadline;
// 如果有一个名为myClock的cookie，则使用该值作为截止日期
if(document.cookie && document.cookie.match('myClock')){
  // 从Cookie获取截止日期值
  deadline = document.cookie.match(/(^|;)myClock=([^;]+)/)[2];
} else {
  // 否则，请设置从现在开始10分钟的截止日期，
  // 将其保存在具有该名称的cookie中

  // 创建从现在开始10分钟的截止日期
  const timeInMinutes = 10;
  const currentTime = Date.parse(new Date());
  deadline = new Date(currentTime + timeInMinutes*60*1000);

  // 将截止日期存储在cookie中以供将来引用
  document.cookie = 'myClock=' + deadline + '; path=/; domain=.yourdomain.com';
}
```

需要注意的是，你需要将 `.yourdomain.com` 改为你的实际域名。

JavaScript日期和时间是从用户的计算机上获取的，这意味着用户可以通过更改计算机上的时间来影响JavaScript时钟。在大多数情况下，这并不重要，但在一些超级敏感的情况下，就需要从服务器上获取时间。可以使用一些Node.js或Ajax来完成，这两者都超出了本教程的范围。

从服务器获取时间后，我们可以使用本教程中的相同技术来使用它。

10.总结

在完成本文中的示例之后，你现在知道了如何使用几行简单的JavaScript代码创建自己的倒计时计时器！我们已经了解了如何制作一个基本的倒计时时钟并有效地显示它。我们还介绍了添加一些有用的附加功能，包括日程安排、绝对时间与相对时间，以及在页面和网站访问之间用cookie保存状态。

下一步是什么?

试着添加一些创意风格，或者新的功能（比如暂停和恢复按钮）。之后，如果你想出了任何很酷的时钟例子，你想分享，让我们在评论区见。

> 原作者姓名：杜尼卜
> 原出处：SegmentFault
> 原文链接：人类身份验证 - SegmentFault

发布于昨天 09:45

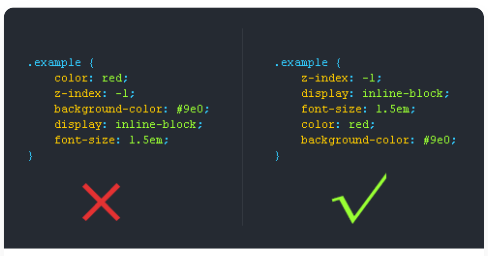
前端工程师　前端开发　程序员

文章被以下专栏收录

web前端学习圈
web前端学习方法/知识干货/实战案例等，每天更新

关注专栏

推荐阅读



web前端达到什么水平，才能找到工作?

晴空万里g...　发表于零基础学习...

50道JavaScript基础面试题（附答案）

1 介绍JavaScript的基本数据类型 Number、String 、Boolean 、Null、Undefined Object 是JavaScript 中所有对象的父对象 数据封装类对象：Object、Array、Boolean、Number 和 String 其...

爱前端不爱...　发表于web前端...

36 * N 个工 JavaScript

最近在思否看中常用的Java 坑爹面试官（ 有一个答案呢 重方案一：Se noRepeat(ar

爱前端不爱...

2 条评论　切换为时间排序

写下你的评论...

夜子　昨天 10:45

setInterval做定时器并不准确，建议使用requestAnimationFrame。

赞

严寒　昨天 20:25

赞