br为什么能换行

背景

在做富文本编辑器基础研究时，研究到DOM节点的`offsetLeft`属性，发现当我们不对浏览器默认样式进行如下`reset`时，该属性的值会比期望的多出`8px`的宽度（`chrome浏览器`）：

```
* {
  margin: 0;
}
```

同时，我们也知道，这肯定是浏览器默认样式（`user agent stylesheet`）在作祟，可以从控制台看到浏览器默认样式如下：

```
// user agent stylesheet
body {
  display:block;
  margin: 8px;
}
```

于是，就想着看看其它HTML标签的默认样式是什么，搜索到了`chrome内核源码`的默认CSS样式表

通过查阅该文档，理解了很多之前无法理解的内容，比如，为什么`div`是块级元素而`span`就是内联元素，而为什么我们又可以通过`css`规则将`div`改变为内联元素或者把`span`元素转变为块级元素。

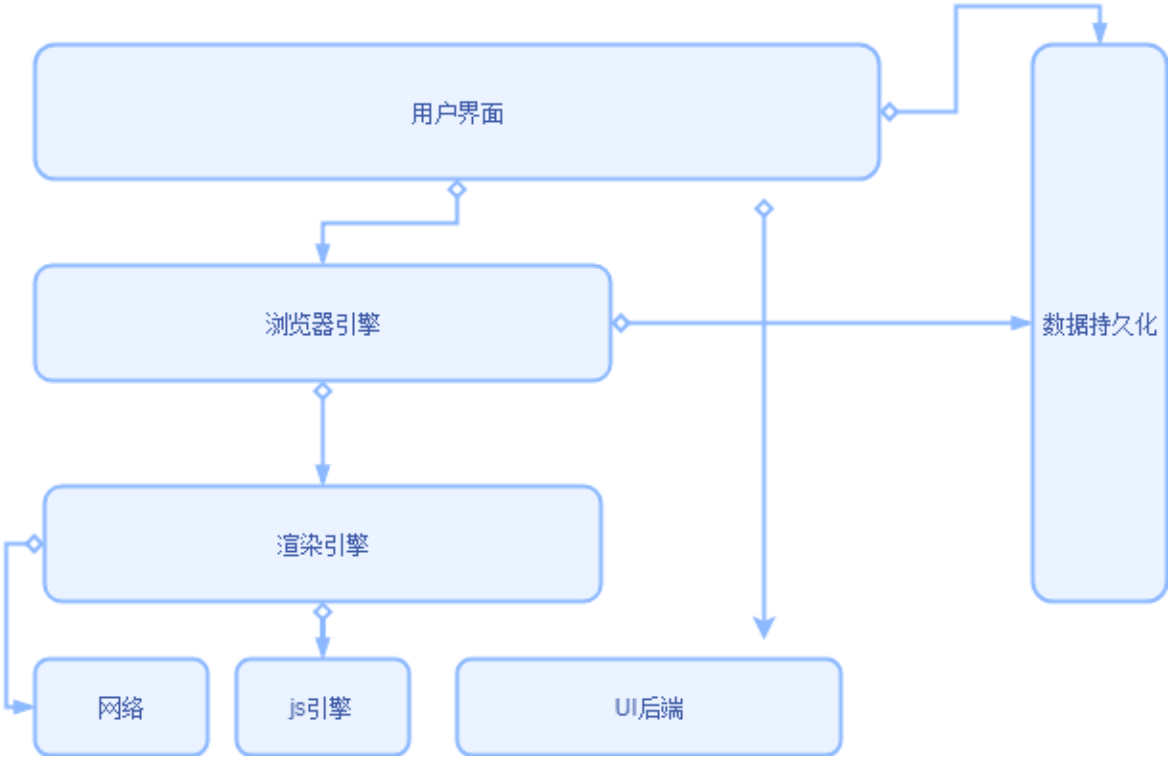
但是在这份文档里，我却没有看到`br`标签的默认样式，于是就产生了好奇，是什么样式设置让`br`可以实现换行的呢？

经过一番搜索，终于有了答案。

这个故事，得从头讲起。

浏览器的组成

我们常说的浏览器，实际上由以下几部分组成，先看图：



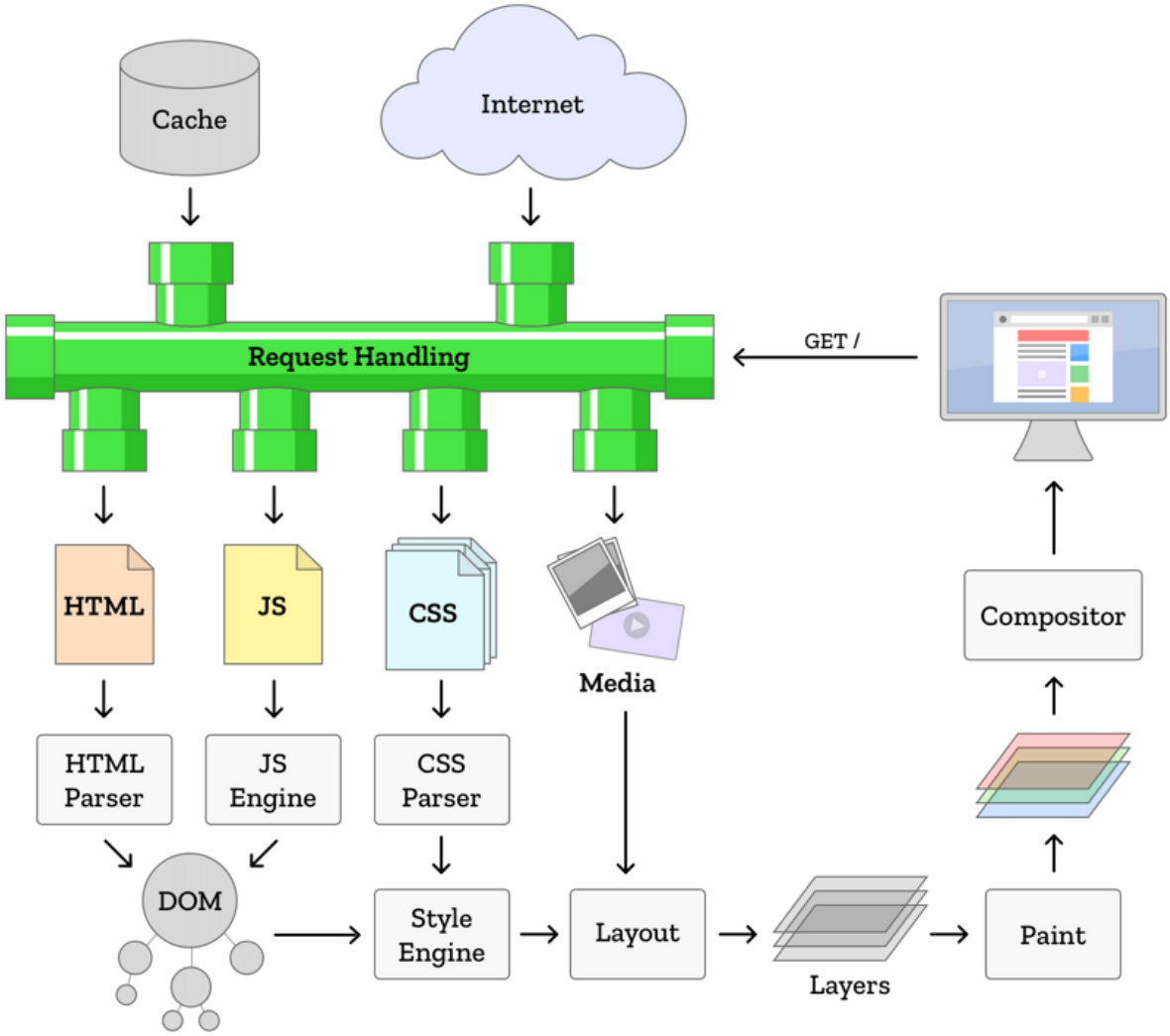
> 上图箭头表示各个组成部分之间的通讯

- 用户界面：包括地址输入框、前进返回按钮、书签、历史记录等用户可见和可操作性的区域；
- 浏览器引擎：负责在用户界面和渲染引擎之间传送指令，或者在本地缓存中读写数据，相当于在各个模块之间跑腿送货的。
- 渲染引擎：即我们说的浏览器内核。负责解析 `DOM` 文档(`HTML` 等)和 `CSS` 规则并将内容排版到浏览器中显示有样式的界面，这个是我们下面要讲的。
- 网络模块： 负责开启网络县城发送请求和下载资源
- `JS` 引擎： 负责解释和执行 `JS` 脚本，典型的如： `V8` 引擎
- `UI` 后端： 负责绘制基本的浏览器窗口内控件，如按钮、输入框等
- 数据持久化：负责 `cookie/localStorage` 等本地缓存的存储

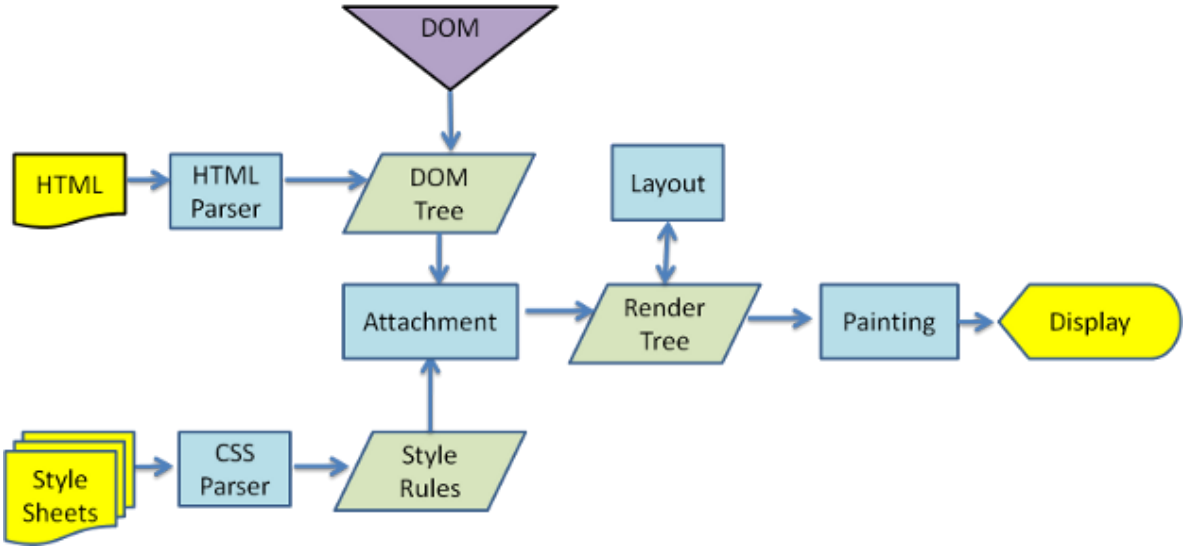
渲染引擎

我们的网页之所以能够有各种各样的布局与样式，全依赖于渲染引擎对我们所写 `HTML` 与 `CSS` 的解析渲染。

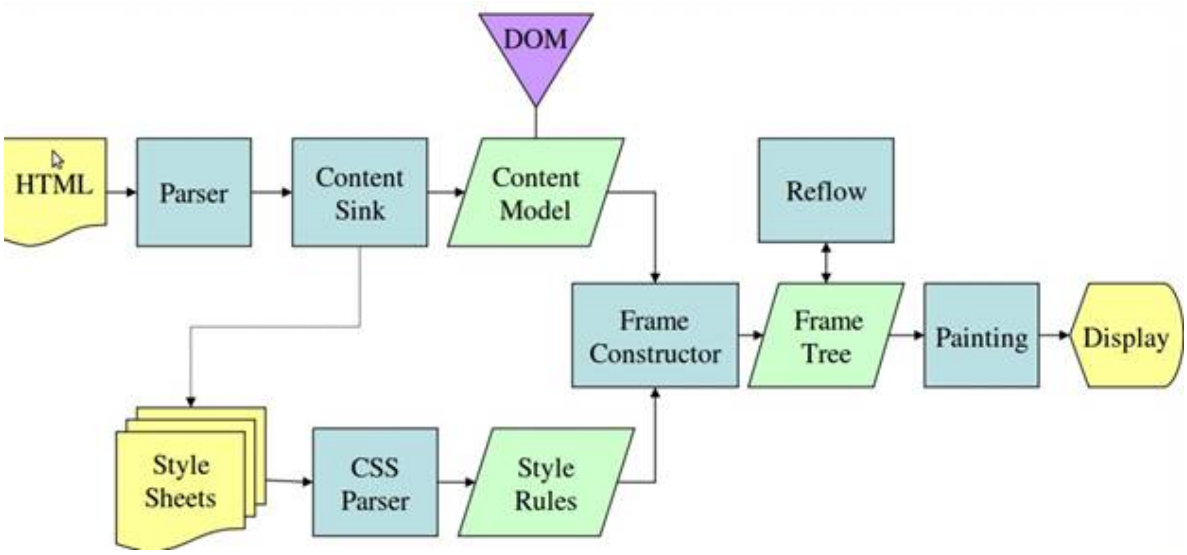
网页呈现流程：



webkit渲染流程：



gecko渲染流程：



从这3张图我们得到以下信息点：

- 网络模块负责从网络获取得到 `HTML` 、 `JS` 、 `CSS` 等资源
- 渲染引擎中的 `HTML` 解析器与 `JS` 引擎分别解释执行 `HTML` 内容与 `JS` 脚本，生成 `DOM` 树。
- `CSS` 解析器解析 `CSS规则` 生成了样式规则集（即 `CSSOM` 树）
- 渲染引擎根据 `DOM` 树与 `CSSOM` 树生成样式树
- 再根据样式树与布局（gecko成为重排，主要是计算后的元素位置）生成渲染树
- 最后渲染引擎将渲染树绘制到显示设备上。

既然 `HTML` 的作用只是用来构建 `DOM` 树，体现结构与语义，那么同理 `<br>` 标签也就只是表示文档某处有一个换行，但是却无法真正实现换行的效果。

那么这个真正的换行效果是谁实现的呢？答案是—— `CSS` .

CSS的五种来源

有这样一段代码：

```
<html>
  <body>
    <h1>静夜思</h1>
    <p>
      <span>窗前明月光</span><br>
      <span>疑是地上霜</span><br>
      <span>举头望明月</span><br>
      <span>低头思故乡</span><br>
    </p>
  </body>
</html>
```

我们不给它设置任何 `CSS` 样式，看看效果：

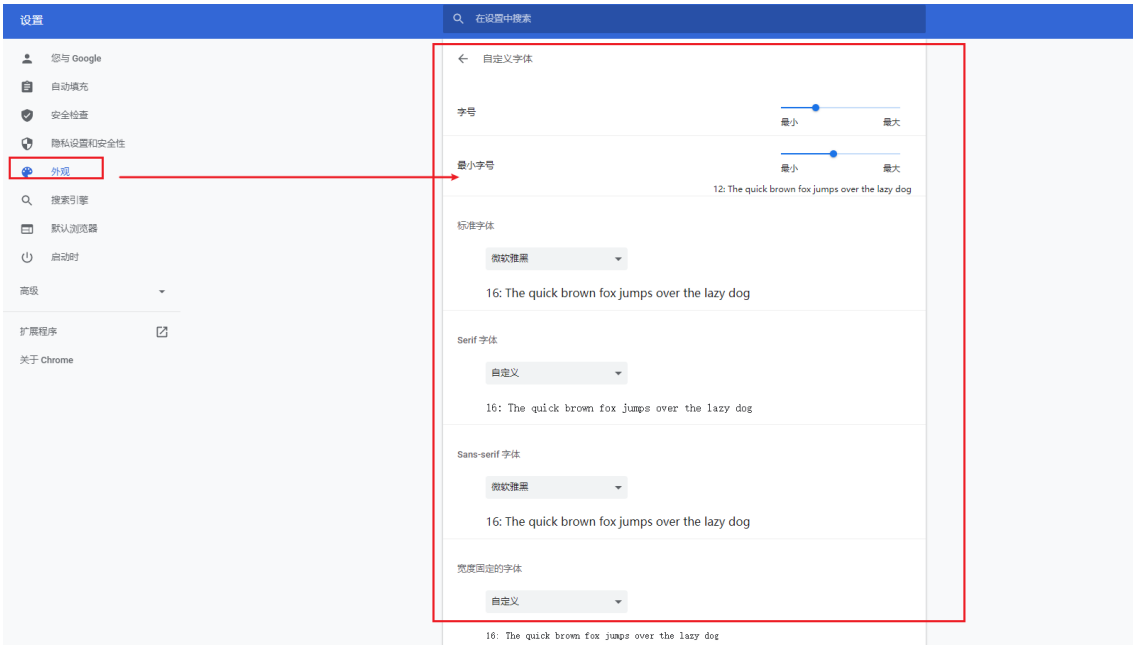
在没有设置任何样式的情况下，标题依旧加粗加大，内容依旧有换行。

而这些效果，全部来自于浏览器的默认样式：`user agent stylesheet`

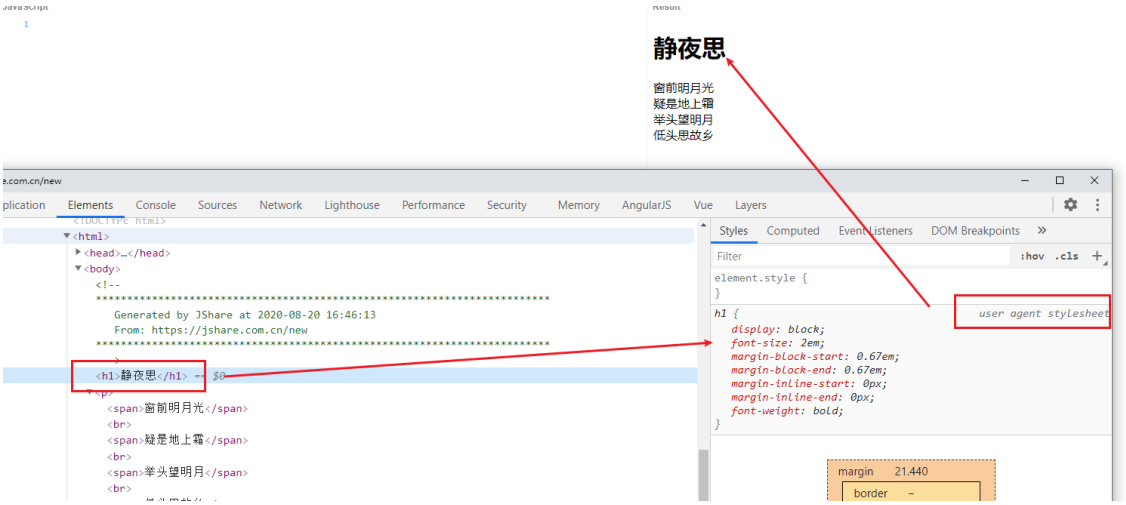
说到这里，我们就要说一下网页中`CSS`的五种来源了：

上面三种样式来源，就是我们在前端开发时经常要编写的部分，而下面两种，则是浏览器内置的，而由下到上，上面一级的样式可以覆盖下面一级的，其中：

- 浏览器用户自定义样式，是指用户可以通过浏览器选项中提供的功能来设置浏览器默认的一些样式，比如字体，字号等；



- 浏览器默认样式，就是我们上面说的 `user agent stylesheet`，是在我们没有在程序中编写样式，也没有设置浏览器用户自定义样式时，浏览器会去读取一份内置的样式表，利用这份样式表里的样式来渲染页面。



可是在检查元素这里却看不到 `<br>` 标签对应的浏览器默认样式，我们查阅浏览器源码中的 默认CSS样式表，也没有发现有 `<br>` 标签的样式，那它的换行是如何实现的呢？

br的CSS实现

几经周折，我们在HTML规范的HTML标签示例中找到了它：

```
hr              { border: 1px inset }
ol, ul, dir,
menu, dd        { margin-left: 40px }
ol              { list-style-type: decimal }
ol ul, ul ol,
ul ul, ol ol    { margin-top: 0; margin-bottom: 0 }
u, ins          { text-decoration: underline }
br:before       { content: "\A"; white-space: pre-line }
center          { text-align: center }
:link, :visited { text-decoration: underline }
:focus          { outline: thin dotted invert }

/* Begin bidirectionality settings (do not change) */
BDO[DIR="ltr"]  { direction: ltr; unicode-bidi: bidi-override }
BDO[DIR="rtl"]  { direction: rtl; unicode-bidi: bidi-override }
```

原来 `<br>` 标签的换行是这几条 `css` 规则

```css
br:before       {
    content: "\A";
    white-space: pre-line
}
```

为什么这几条就能实现换行呢？

我们知道 `white-space:pre-line` 的作用是去除空格，保留换行，但它本身并不会添加换行效果，那剩下的就是要弄清楚 `content:  '\A'` 是什么意思就知道换行是如何实现的了。

通过查看规范， `CSS` 的 `content` 属性只能用于 `:after` 和 `:before` 伪元素，它接受以下类型的值：

- 普通字符串，如 `content: '你好'`
- 元素属性，如 `content: attr(alt)`
- 外部资源：如 `content:url(http://www.baidu.com/picture/a.jpg)`
- 调用计数器： 如 `content: counter(dd) ' '`
- `unicode` 字符： 如 `content:'\21e0'`
- 引号标志：如 `content: open-quote | close-quote | no-open-quote | no-close-quote`

！！！恍然大悟，原来 `\A` 是 `unicode` 字符，那么 `\A` 在 `unicode` 字符中表示什么呢？

转义字符	意义	ASCII码值（十进制）
\a	响铃(BEL)	007
\b	退格(BS)，将当前位置移到前一列	008
\f	换页(FF)，将当前位置移到下页开头	012
\n	换行(LF)，将当前位置移到下一行开头	010
\r	回车(CR)，将当前位置移到本行开头	013
\t	水平制表(HT)（跳到下一个TAB位置）	009
\v	垂直制表(VT)	011

众所周知，`unicode` 是 `ascii` 的超级，并且 `unicode` 码是16进制的，那16进制的 `A` 正好是十进制的 `010`，我们终于在这张表里找到了十进制 `010` 的 `ascii` 码对应的字符：`\n` —— 换行符！！！

现在我们知道 `br` 为什么可以换行了：

- 利用伪元素创建了一个内容为换行符的元素
- 利用 `white-space` 属性保留了这个换行符

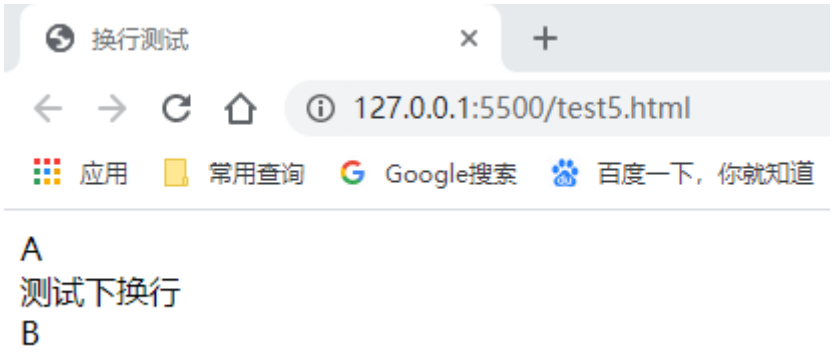
扩展1：自己实现换行

我们完全可以不用 `br` 标签，自己实现换行：

```
<style>
span:before {
            content:'\A';
            white-space: pre-line;
        }
        span:after {
            content: '\A';
            white-space: pre-line;
        }
</style>

<body>
  A<span>测试下换行</span>B
</body>
```

我们看看效果：



扩展2： 小写的a行不行

由于 `unicode` 码是不区分大小写的，所以把 `\A` 替换成 `\a` 也是完全没有问题的，读者可自行测试。

扩展3：只能使用 pre-line吗

使用 `white-space` 的目的就是保留我们利用 `content` 创建的换行符，所以所有可以保留换行的值都是可以的：

下面的表格总结了各种 white-space 值的行为：

	换行符	空格和制表符	文字换行	行尾空格
normal	合并	合并	换行	删除
nowrap	合并	合并	不换行	删除
pre	保留	保留	不换行	保留
pre-wrap	保留	保留	换行	挂起
pre-line	保留	合并	换行	删除
break-spaces	保留	保留	换行	换行

由上表知，可以保留换行符的 `pre,pre-wrap,pre-line,break-spaces` 都是可以的，读者可自行测试

扩展4：换行换成回车可以吗？

我们知道，要生成换行的效果，除了换行符，回车也可以达到相同效果，那把这里的换行符换成回车符（ `\d` ）可以吗?

不可以。

这涉及到回车和换行的区别，简而言之，这两个词是借鉴于老式打字机：

*"回车"，告诉打字机把打印头定位在左边界

"换行"，　告诉打字机把纸向下移一行。

对应到计算机输入中，回车只是代表把光标定位到行首，并不能实现换行，真正要移动到下一行，要使用换行符

扩展5：能不能覆盖br标签的样式让她不换行

既然我们前面讲 `CSS` 五个来源时，说上一级的会覆盖下一级，我们可以利用这个特性，将 `span` 变成块级元素，将 `p` 变成内联元素，那我们在自己的 `CSS` 中重写 `br` 的伪元素样式，可以改变它的行为吗？

```
br::before {
            content: '\d';
            white-space: nowrap;
        }
        br::after {
            content: '\d';
            white-space: nowrap;
        }
```

经过测试，不可以。

原因暂时未知，如果读者中有研究出结果的，欢迎分享出来。

经过对规范文档一番搜索，发现一些蛛丝马迹：

首先，在 `HTML` 规范中关于 `br` 标签的说明部分， 我们可以找到以下说明：

> This element has rendering requirements involving the bidirectional algorithm.
>
> 该元素渲染时需要符合双向演化算法的要求

好，先暂停一下，啥是个双向演化算法？

我们都知道，一般来说，我们接触的书写语言，无论是汉语还是英语还是德语法语，它的书写顺序都是从左至右的，可是世界是多样的，就有那么一些语言，它的书写方向偏偏就是从右至左的，比如阿拉伯语等，这还不是问题，关键是，这些语言中的数字却又是从左至右书写的，这样就导致这种语言实质上同时包含了两种书

写方向，而在计算机中，这些语言的文字都是使用 `unicode` 编码来实现的，所以，聪明的大神们设计了针对 `unicode` 字符集的 `双向演化算法` 来解决这类问题。

好巧不巧，任何语言的书写都需要换行（这不废话吗），所以 `br` 的渲染也要符合双向演化算法的要求。

知道了这个，我们再来看双向演化算法的要求：

> The mapping of HTML to the Unicode bidirectional algorithm must be done in one of three ways. Either the user agent must implement CSS, including in particular the CSS **unicode-bidi**, **direction**, and **content** properties, and must have, in its user agent style sheet, the rules using those properties given in this specification's **rendering** section, or, alternatively, the user agent must act as if it implemented just the aforementioned properties and had a user agent style sheet that included all the aforementioned rules, but without letting style sheets specified in documents override them, or, alternatively, the user agent must implement another styling language with equivalent semantics. [**CSS-WRITING-MODES-3**] [**CSS3-CONTENT**]
>
> HTML到Unicode双向算法的映射必须通过以下三种方式之一完成。 用户代理必须实现CSS，特别是CSS unicode-bidi，direction和content属性，并且必须在其用户代理样式表中具有使用本规范呈现部分中给出的那些属性的规则，或者，** 用户代理必须充当仅实现上述属性的行为，并且具有包括所有上述规则的用户代理样式表，但又不能让文档中指定的样式表覆盖它们，或者，用户代理必须实施另一种样式语言 具有相同的语义 **

看到上面我用一对星号括起来的内容没，就是说作为用户代理，浏览器有两种选择：

- 实现规范给出的规则，并且不允许文档中的样式表覆盖他们。【吼吼，被我抓到了】
- 试试另一种样式语言，具有相同的语义【也可能是这种】

那我们看看规范给出的规则：

```
@namespace url(http://www.w3.org/1999/xhtml);

[dir]:dir(ltr), bdi:dir(ltr), input[type=tel i]:dir(ltr) { direction: ltr; }
[dir]:dir(rtl), bdi:dir(rtl) { direction: rtl; }

address, blockquote, center, div, figure, figcaption, footer, form, header, hr,
legend, listing, main, p, plaintext, pre, summary, xmp, article, aside, h1, h2,
h3, h4, h5, h6, hgroup, nav, section, table, caption, colgroup, col, thead,
```

```
tbody, tfoot, tr, td, th, dir, dd, dl, dt, ol, ul, li, bdi, output,
[dir=ltr i], [dir=rtl i], [dir=auto i] {
  unicode-bidi: isolate;
}

bdo, bdo[dir] { unicode-bidi: isolate-override; }

input[dir=auto i]:matches([type=search i], [type=tel i], [type=url i],
[type=email i]), textarea[dir=auto i], pre[dir=auto i] {
  unicode-bidi: plaintext;
}
/* see prose for input elements whose type attribute is in the Text state */

/* the rules setting the 'content' property on <br> and <wbr> elements also has
bidi implications */
```

看到第22行的注释了吗？

它的意思是 `br` 标签的 `content` 属性可以设置类似以上的规则。

但同时， `br` 标签又属于段落文本类型，它同样可以实现段落文本的规则：

```
//……
br { display-outside: newline; } /* this also has bidi implications */
//……
```

遗憾的是，这些都只是规范推荐的规则，浏览器可以不按这些规则实现，只要实现相同的语义就可以。

那浏览器还有什么选择呢？

这又引出了另一对概念：替换元素与非替换元素

说到 `CSS` 的模型，我们都知道块级元素和内联元素的区分，但却很少了解，其实还有一种维度的区分方式，就是替换元素和非替换元素。

所谓替换元素，就是在渲染时，直接用其它内容将它替换掉，所以它并不属于 `CSS` 格式化模型，而是独立计算渲染的，但是我们可以通过 `CSS` 样式设置它的大小和位置。（这个具体比较复杂，有兴趣可以自行研究下）

比如， `img` 标签就是典型的替换元素，浏览器在渲染它的时候，实际上是直接拿它的 `src` 属性所指向的图片对象将它替换了。 尽管我们查看源码可以看到它的位置仍旧是一个 `img` 标签，但是实际上在渲染层面，它已经被替换，

在 `MDN` 文档中，我们也可以找到这样的话：

> 由 `::before` 和 `::after` 生成的伪元素 包含在元素格式框内， 因此不能应用在 *替换元素上*， 比如 `<img>` 或 `<br>` 元素。

就是这么奇怪，在`HTML`规范中，替换元素并不包含`<br>`，但在这里，又说`<br>`是替换元素，我们只能暂时认为，在浏览器内核实现时，开发人员可能会选择使用这种方式来实现换行：

——直接将`br`标签替换为一个换行符。

根据可以查阅到的资料，目前就只有这两种猜测：

- 浏览器实现了规范推荐的规则，这些规则被隐藏起来了，所以查看元素时看不到它的默认规则，同时根据双向演化算法要求禁止我们去覆盖它的规则
- 浏览器选择了将它实现为替换元素，因为替换元素不属于`CSS`格式化模型，所以我们无法通过自己编写的CSS规则改变它的行为