

## Resource Hints

### Resource Hints: DNS- PREFETCH and PRECONNECT

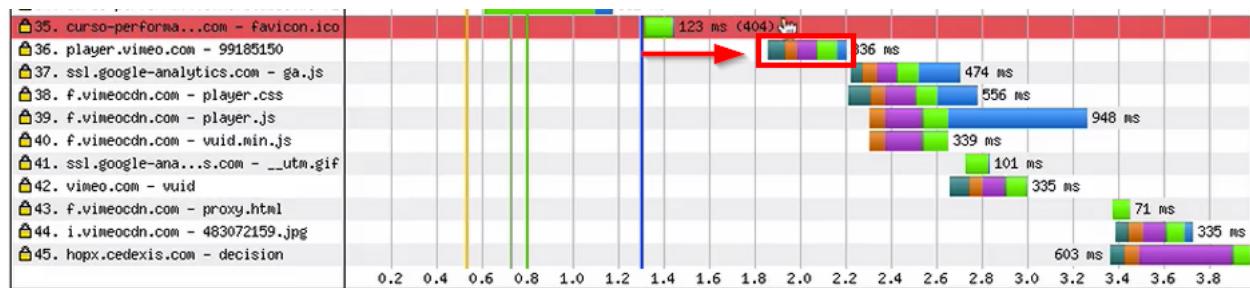
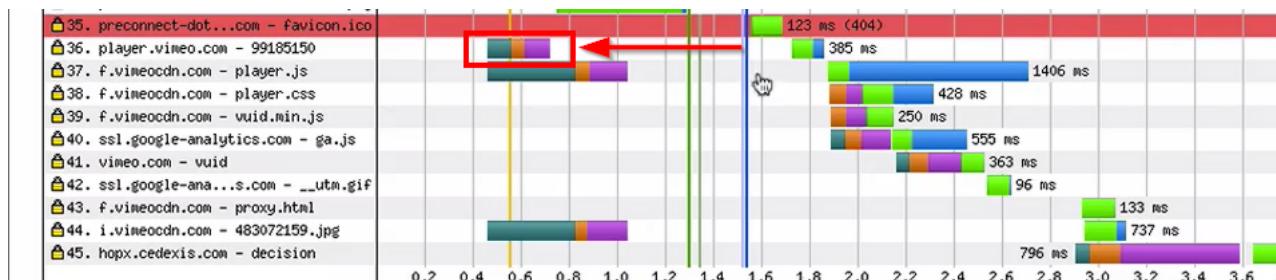
Vamos tentar entender como funcionam os `rel` e `preloads`. Queremos abordar também os *Resource Hints*. O primeiro *Resource* que foi inventado foi o *DNS-PREFETCH* que permite pedir ao navegador para resolver o DNS de um certo recurso antecipando que precisaremos dele mais adiante. Podemos utilizar isso em nosso `index.html`, criando uma tag "Link" e usando:

```
<link rel="dns-prefetch" href="/player.vimeo.com/">
```

Com isso, a resolução do DNS é antecipada para o começo. Quando ele fizer o *parcing* do HTML. A resolução do DNS é feita totalmente em *background* o que não significa que isso não travará a renderização ou o download das demais coisas. Também podemos fazer o mesmo com outras coisas e podemos utilizar esse recurso quantas vezes fizer sentido para o site.

Mostramos ele pois é o que possui melhor suporte, funciona em todos os navegadores. Foi um dos primeiros a ser inventado desses *Resource Hints*. Logo, se percebeu que podia-se dar passos além, não apenas resolver o DNS.

Um segundo *Resource Hint* é o *preconnected*. Além de antecipar o *DNS* podemos antecipar, também, a conexão. Para usar isso podemos escrever `link rel=preconnect`. Ele inclui a resolução do DNS, faz a conexão TCP e se for uma conexão segura ele inclusive faz a negociação CSL para nós. Você pode se perguntar se isso realmente impacta em algo, repare nos gráficos. O primeiro diz respeito as mudanças que acabamos de fazer:



O que fizemos foi antecipar a conexão. Não podemos utilizar isso com diversas conexões, pois pode acabar atrapalhando as conexões que já temos. Podemos fazer isso com algum recurso adicional, por exemplo, um vídeo. Coisas que você gostaria de carregar mais rapidamente e que devem ser indicadas para o navegador.

### PREFETCH

O `prefetch` serve para passarmos para ele alguma "url" ou arquivo para que faça o download e guardemos isso no seu `cache`. O `prefetch` antecipa o download imaginando que esse download será utilizado no futuro. Ele serve para navegações secundárias, ele possui menos prioridade do que qualquer outro `request` que tenhamos feito na página, por isso, ele é indicado para ser feito na página seguinte.

Podemos antecipar o download do "categorias.js" digitando:

```
<link rel="prefetch" href="pagina-seguinte/categoria.js">
```

Assim, antecipamos esse download imaginando que a chance de ele clicar em alguma categoria da **Alura** é alta. Assim, já deixando isso no `cache`.

É preciso tomar cuidado, se abusarmos do uso dessas táticas é possível que começemos a baixar coisas que não são tão úteis, coisas a mais. Temos que utilizar esses recursos com certa esperteza. Outra sugestão é também não fazer o `prefetch` de um vídeo de 2G, pois pode acontecer um potencial desperdício. O `prefetch` deve ser utilizado quando se visualiza uma chance alta de ser necessário.

O `preload` possui um uso similar, mas ele possui uma semântica distinta:

```
<link rel="preload" href="arquivo.css">
```

A diferença é que o `prefetch` possui uma prioridade baixa, está direcionado a uma navegação futura. O `preload` está direcionado para o momento, para a navegação atual.

O `preload` pode ser utilizado quando temos um carrossel de imagens e vários banners passando, colocamos o primeiro banner na imagem, mas sabendo que devemos antecipar o segundo banner, podemos pedir que o segundo banner seja antecipando o seguinte, pois na hora que o carrossel rodar o segundo banner já irá aparecer. Digitaremos o seguinte nesse caso:

```
<link rel="preload" href="banner2.jpg">
```

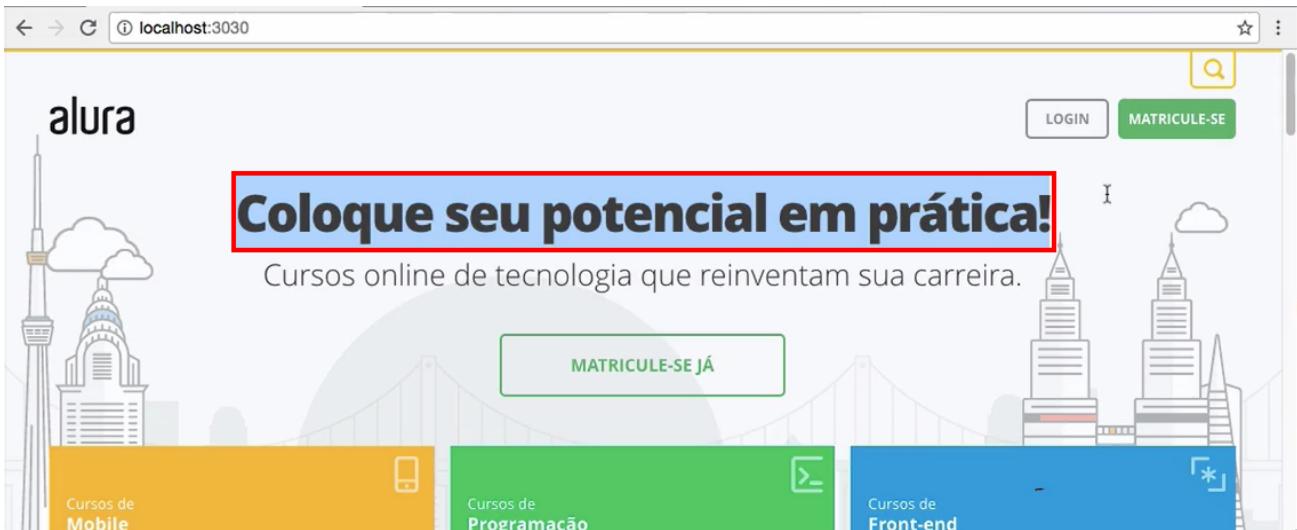
Já utilizamos o `preload` antes com coisas que queríamos que tivessem alta prioridade. Não poderíamos, nesse caso, usar o `prefetch`, pois desejamos uma prioridade alta.

Imagine que temos diversos `preloads` possíveis, para saber qual deve ser o `preload` a ser utilizado devemos introduzir um atributo, o `as`, ele na verdade comunica a prioridade para o download do navegador:

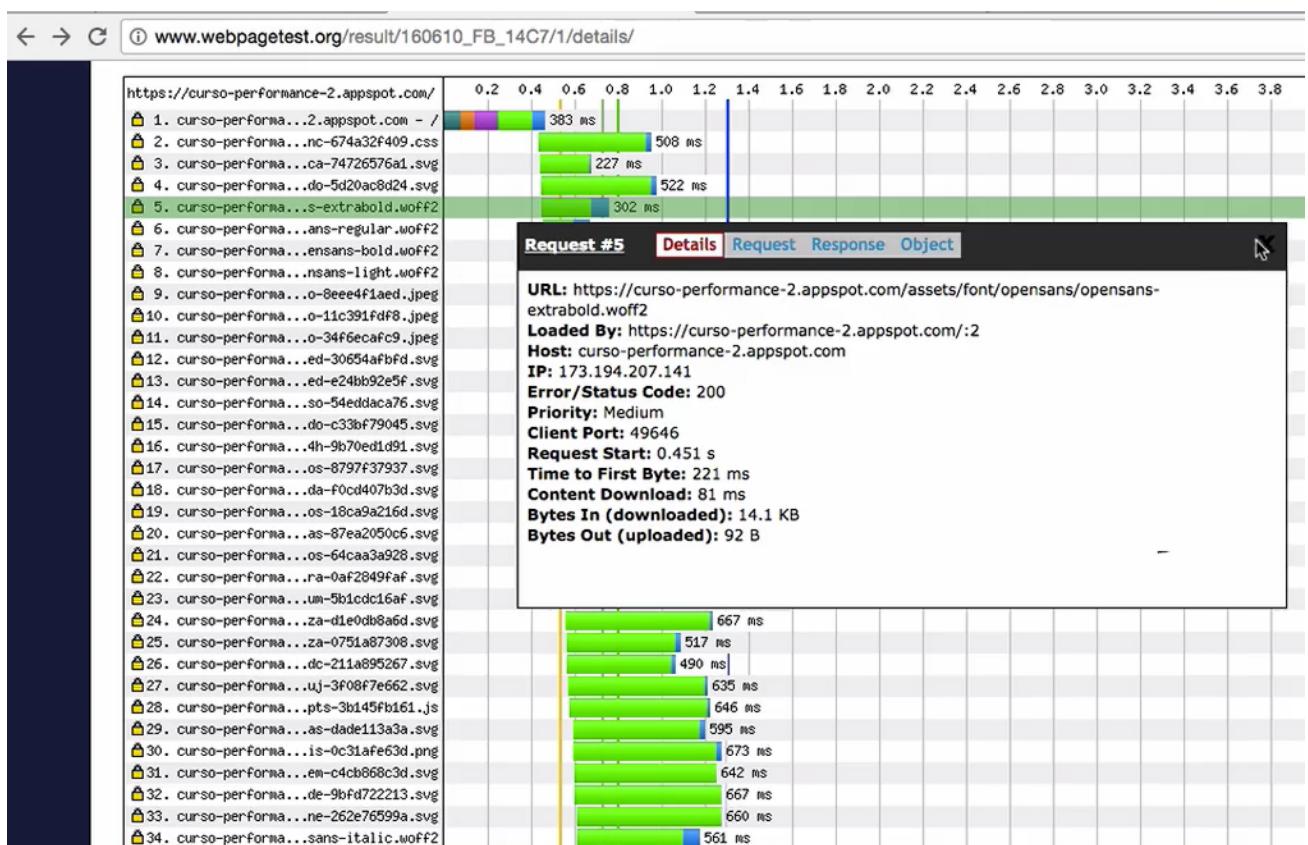
```
<link rel="preload" href="arquivo.css" as="style">
```

Ele lê o seguinte interpretando que o `style` possui prioridade, se fosse `as="image"` o navegador lê isso como algo com menor prioridade, pois sabe que isso é uma característica das imagens. Se você deixar sem o `as` isso indica que a prioridade é baixa.

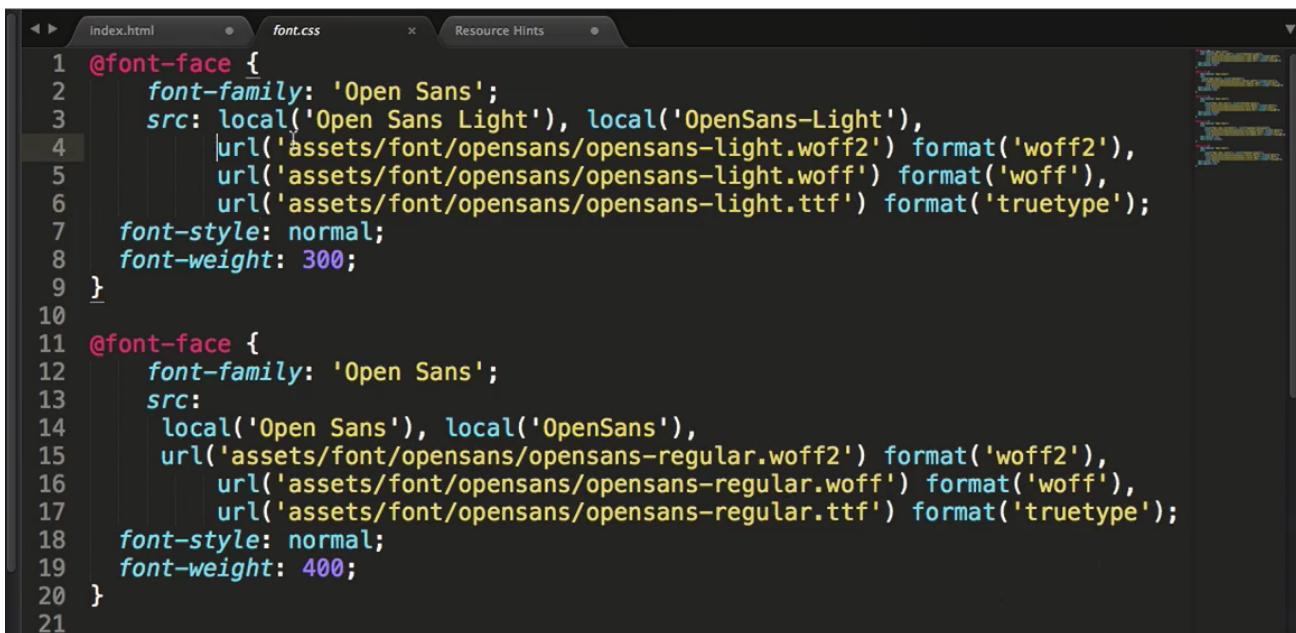
Vamos pensar onde caberia o uso do `preload` no site, um recurso que costuma atrasar bastante a renderização do site é o seguinte:



A fonte! As fontes são alguns dos recursos que mais tardam em serem baixados na página, podemos, inclusive, reparar pela faixa em azul no gráfico "WaterFall":



Vamos relembrar, as fontes são declaradas no CSS:



```

1 @font-face {
2     font-family: 'Open Sans';
3     src: local('Open Sans Light'), local('OpenSans-Light'),
4          url('assets/font/opensans/opensans-light.woff2') format('woff2'),
5          url('assets/font/opensans/opensans-light.woff') format('woff'),
6          url('assets/font/opensans/opensans-light.ttf') format('truetype');
7     font-style: normal;
8     font-weight: 300;
9 }
10
11 @font-face {
12     font-family: 'Open Sans';
13     src:
14         local('Open Sans'), local('OpenSans'),
15         url('assets/font/opensans/opensans-regular.woff2') format('woff2'),
16         url('assets/font/opensans/opensans-regular.woff') format('woff'),
17         url('assets/font/opensans/opensans-regular.ttf') format('truetype');
18     font-style: normal;
19     font-weight: 400;
20 }
21

```

E a "url" dessa fonte só será baixada na hora em que ela for realmente utilizada. O navegador adia o download da fonte o máximo possível. Ele precisa primeiro baixar o CSS, "parsear" o CSS, baixar o HTML "parseá-lo" e encontrar algum nó que use a fonte e, então, inicia o *download*. E isso, pode ser demorado. E, assim, podemos fazer o `preload` disso, escrevendo:

```
<link rel="preload" href="assets/font/opensans-extrabold.woff2" as="font" type="font/woff2">
```

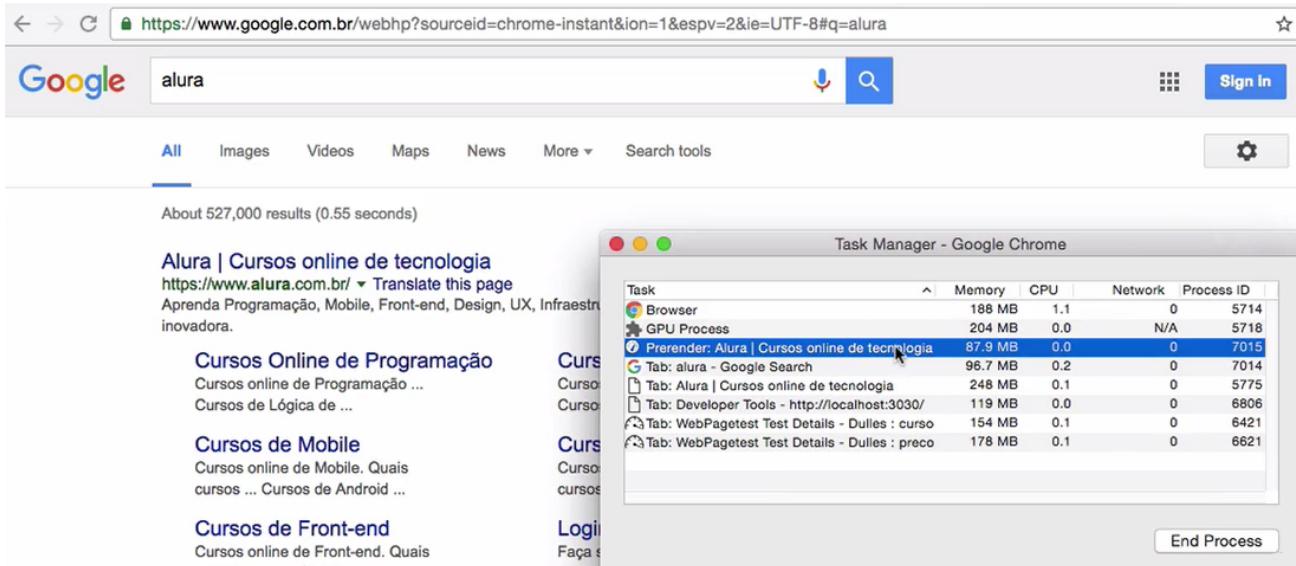
Agora, faremos esse download mais rapidamente. O `type="font/woff2"` serve para indicar que o download só vai ser feito se o navegador entender o `woff2` e você pode usar o `woff1` para os navegadores que entendem o `woff1`. O `preload` é bem útil para baixar recursos que são tardivamente descobertos, seja por ser assíncrono, secundário, por algum motivo que não é possível que ele seja descoberto logo de cara.

## PRERENDER

Para fecharmos o tema de *Resources Hint* vamos finalizar abordando o `prerender`.

Isso pode parecer estranho, mas na verdade já utilizamos o `prerender` antes. Imagine, quando realizamos uma busca no **Google** ele nos mostra os resultados relacionados as palavras-chaves que usamos. Se buscarmos **Alura** a chance de clicarmos no primeiro site que aparece é muito grande e existe tanto essa certeza que o primeiro link já começa a ser renderizado antes mesmo de ser clicado, é isso que faz o `prerender`.

No **Chrome** se formos em "More Tools > Tool Manager" podemos verificar todas as abas que estão abertas e nessa janela podemos visualizar a aba "Prerender". Essa página já foi renderizada. É como se tivesse uma aba escondida:



The screenshot shows a Google search results page for the query "alura". The search bar contains "alura". Below the search bar, there are filters for "All", "Images", "Videos", "Maps", "News", and "More". The search results list several links related to Alura's courses. To the right of the search results, a "Task Manager - Google Chrome" window is open, showing a list of browser processes. The "Prerender" tab is selected, showing a list of prerendered tabs. The table in the Task Manager window is as follows:

Task	Memory	CPU	Network	Process ID
Browser	188 MB	1.1	0	5714
GPU Process	204 MB	0.0	N/A	5718
Prerender: Alura   Cursos online de tecnologia	87.9 MB	0.0	0	7015
Tab: alura - Google Search	96.7 MB	0.2	0	7014
Tab: Alura   Cursos online de tecnologia	248 MB	0.1	0	5775
Tab: Developer Tools - http://localhost:3030/	119 MB	0.0	0	6806
Tab: WebPageTest Test Details - Dulles : curso	154 MB	0.1	0	6421
Tab: WebPageTest Test Details - Dulles : preco	178 MB	0.1	0	6621

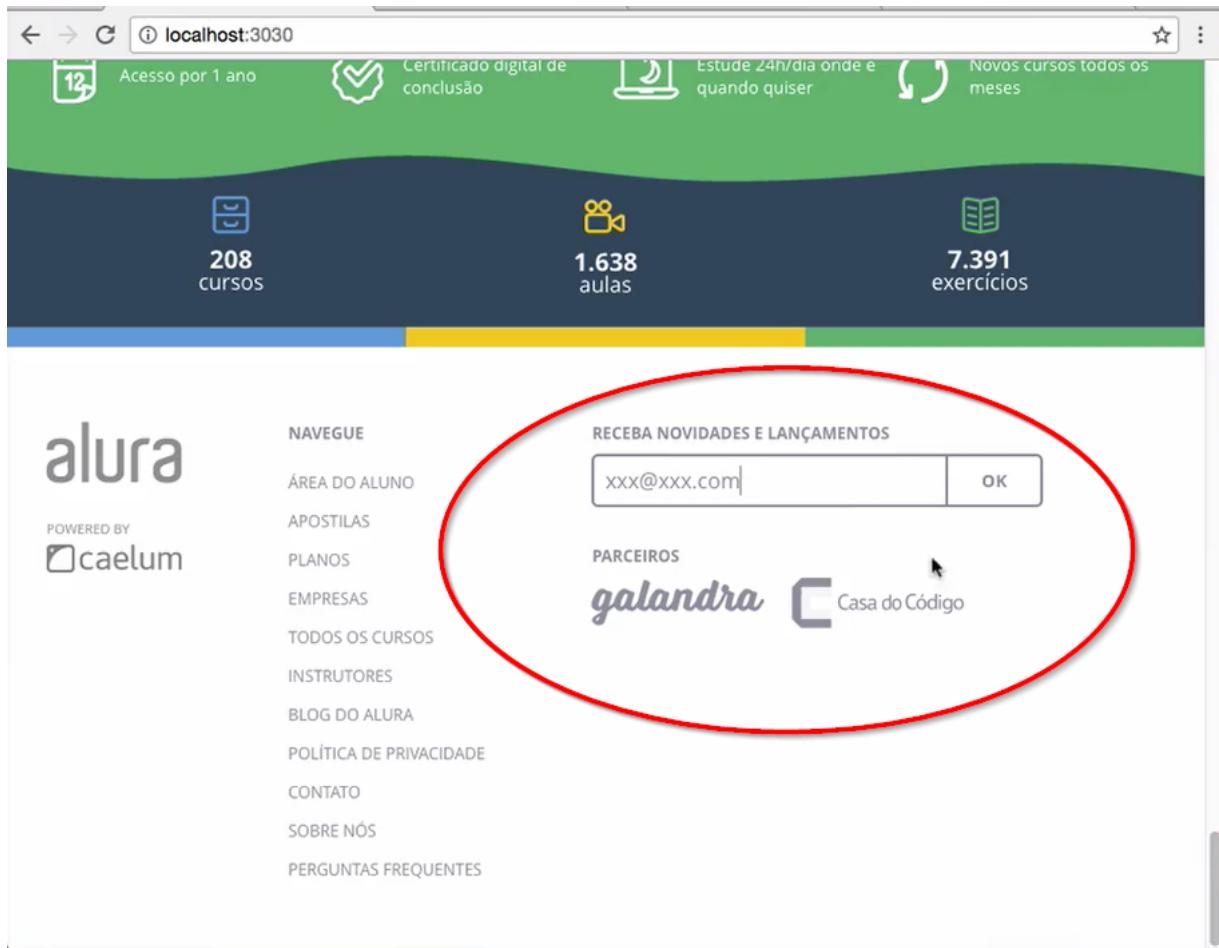
No momento em que clicamos no primeiro link que aparece na lista ele abre quase que instantaneamente. Isso acontece quando fazemos uma busca e o **Google** acredita que a possibilidade de ter acertado é grande. A ação do usuário é antecipada.

Para fazer isso é bastante fácil. Vamos voltar ao código, na "index.html" e acrescentamos o `link rel="prerender"` e a página que deve ser pré-renderizada `href="pagina.html"` :

```
<link rel="prerender" href="pagina.html">
```

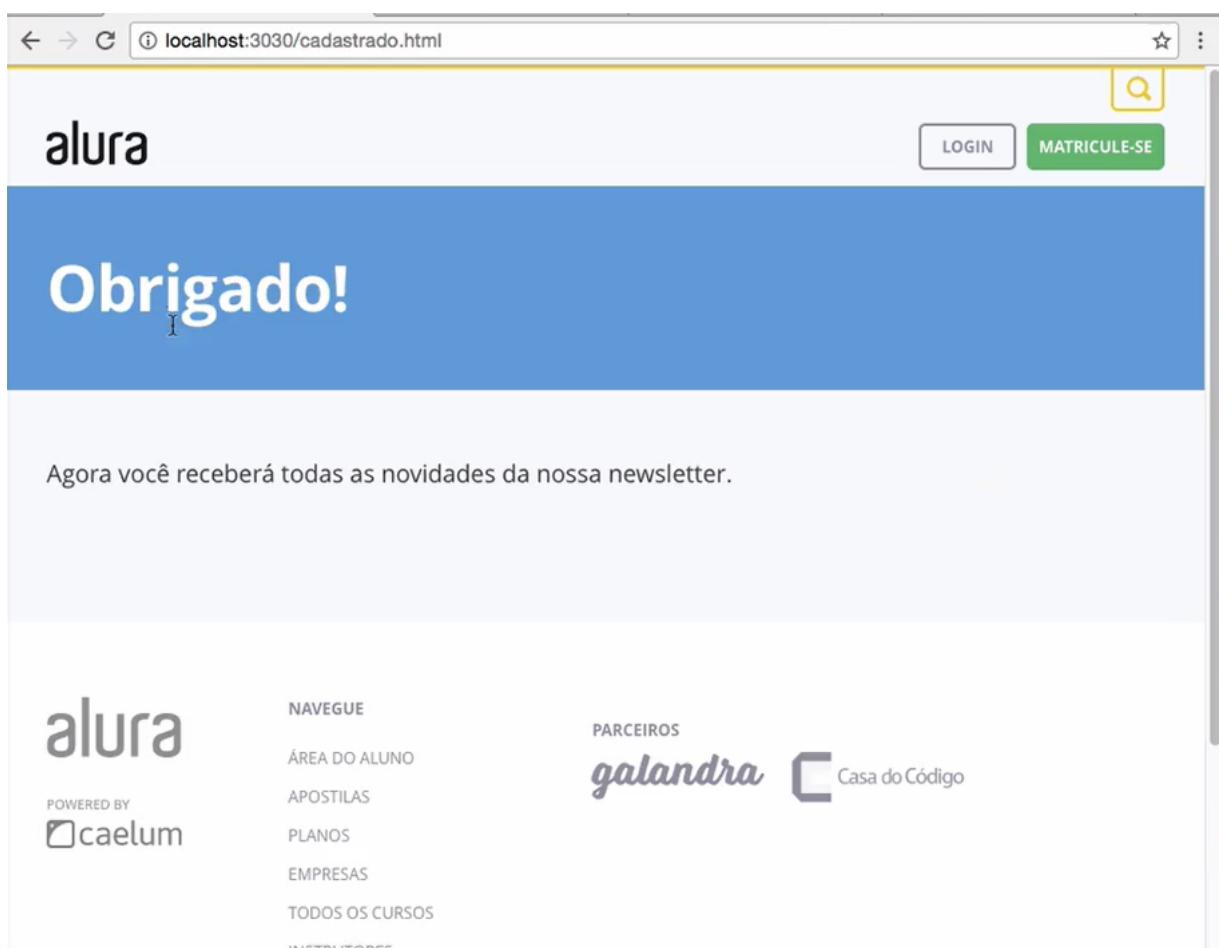
É importante usar esse recurso se você tiver uma razoável certeza de que isso vai ser utilizado, isso é muito útil quando temos os casos de *work flow*. Por exemplo, no caso da leitura de um texto que possui mais de uma página, é provável que a notícia siga sendo lida, a probabilidade da pessoa continuar até as próximas páginas é grande.

No site do **Alura** existe um cenário interessante, no final da página existe um campo para se cadastrar e receber novidades e lançamentos:



The screenshot shows the Alura website homepage. At the top, there are several promotional banners: "Acesso por 1 ano" (Access for 1 year), "Certificado digital de conclusão" (Digital certificate of completion), "Estude 24h/dia onde e quando quiser" (Study 24h/day where and when you want), and "Novos cursos todos os meses" (New courses every month). Below these are three main statistics: "208 cursos" (208 courses), "1.638 aulas" (1.638 lessons), and "7.391 exercícios" (7.391 exercises). On the left, the "alura" logo is displayed, followed by "POWERED BY" and the "caelum" logo. A sidebar on the left lists navigation links: NAVEGUE (ÁREA DO ALUNO, APOSTILAS, PLANOS, EMPRESAS, TODOS OS CURSOS, INSTRUTORES, BLOG DO ALURA, POLÍTICA DE PRIVACIDADE, CONTATO, SOBRE NÓS, PERGUNTAS FREQUENTES). On the right, there is a "RECEBA NOVIDADES E LANÇAMENTOS" (Receive news and releases) form with an input field containing "xxx@xxx.com" and an "OK" button. Below this, there is a "PARCEIROS" (Partners) section featuring the logos for "galandra" and "Casa do Código". A red oval highlights the newsletter sign-up area.

Quando terminamos de preencher com nossas informações esse campo e damos um "ok" somos redirecionados para uma página que diz "Obrigada!":



The screenshot shows a confirmation page with a large blue header containing the text "Obrigado!". Below the header, a message says "Agora você receberá todas as novidades da nossa newsletter." (Now you will receive all the news from our newsletter.). The page layout is identical to the homepage, with the "alura" logo, "POWERED BY" "caelum" logo, and a sidebar with navigation links: NAVEGUE (ÁREA DO ALUNO, APOSTILAS, PLANOS, EMPRESAS, TODOS OS CURSOS, INSTRUTORES). On the right, there is a "PARCEIROS" (Partners) section featuring the logos for "galandra" and "Casa do Código".

A implementação disso é curioso, no "index.html" temos o *input* com o cadastro:

E na hora que clicamos nisso podemos observar que ele não dispara nenhum formulário, isso foi implementado utilizando o *javascript*. Isso está no "footer.js":

A regra é que quando a pessoa clicar no botão de `Newsletter` ele cadastrá a newsletter fazendo uma validação, chamando a `API` e ao terminar de fazer isso somos redirecionados para a página do cadastrado. É uma `redirect`. Esse cenário é um forte candidato para fazermos um `prerender`. Escreveremos o seguinte no `"index.html"`:

```
<link rel="prerender" href="cadastrado.html">
```

Fazendo um `gulp` e carregando a página o que veremos é um `prerender` do "cadastrado":

Coloque seu potencial em prática!

Cursos online de tecnologia que reinventam sua carreira.

**MATRICULE-SE JÁ**

Cursos de Mobile: iOS, Android, PhoneGap, e mais...

Cursos de Programação: Java, C#, Ruby, PHP, Python e mais...

Cursos de Front-end: HTML, CSS, Angular, JavaScript e mais...

Task Manager - Google Chrome

Task	Memory	CPU	Network	Process ID
Browser	188 MB	8.5	0	5714
GPU Process	212 MB	0.0	0	5718
Tab: Alura   Cursos online de tecnologia	246 MB	12.2	0	5775
Tab: uol - Google Search	94.7 MB	0.0	0	7031
Tab: WebPagetest Test Details - Dúlles : curso	154 MB	0.1	0	6421
Tab: WebPagetest Test Details - Dúlles : preço	179 MB	0.1	0	6621

End Process

Outra técnica que podemos utilizar, já que o usuário só vai chegar ao `prerender` se ele preencher o campo das *Newsletter* é criar o `link rel="prerender"` de maneira dinâmica. Imagine que o usuário está na página e começa a preencher o campo, nesse caso, a chance de ele apertar o "ok" é altíssima. Podemos pegar esse evento para disparar o download. Vamos no "footer.js" e diremos que quando a pessoa der foco, aí se criará o elemento do `prerender`:

```
inputEmail.onfocus = function() {
  var prerender = document.createElement('link');
  prerender.rel = 'prerender';
  prerender.href = 'cadastrado.html';

  document.head.appendChild(prerender);
}
```

O ponto é tentar descobrir qual o melhor momento para fazer isso. E isso dependerá do cenário. Vamos testar isso! Antes ele dava `prerender` apenas de abrirmos a página, agora, ele disparará o `prerender` apenas quando dermos foco no campo do e-mail:

Alura - Cursos online de tecnologia

Prerender: Cadastrado na Newsletter do Alura

Task Manager - Google Chrome

Task	Memory	CPU	Network	Process ID
Browser	188 MB	3.8	0	5714
GPU Process	212 MB	0.0	0	5718
Tab: Alura   Cursos online de tecnologia	246 MB	10.0	36 B/s	5775
Tab: uol - Google Search	94.7 MB	0.0	0	7031
Tab: WebPagetest Test Details - Dúlles : curso	154 MB	0.1	0	6421
Tab: WebPagetest Test Details - Dúlles : preço	179 MB	0.1	0	6621

End Process

Alura - Cursos online de tecnologia

Prerender: http://localhost:3030/cadastrado.html

Task Manager - Google Chrome

Task	Memory	CPU	Network	Process ID
Browser	188 MB	3.2	0	5714
GPU Process	212 MB	0.0	0	5718
Tab: Alura   Cursos online de tecnologia	246 MB	9.5	0	5775
Tab: uol - Google Search	94.7 MB	0.0	0	7031
Tab: WebPagetest Test Details - Dúlles : curso	154 MB	0.1	0	6421
Tab: WebPagetest Test Details - Dúlles : preço	179 MB	0.1	0	6621

End Process

