01

Mais Design Fluído

Transcrição

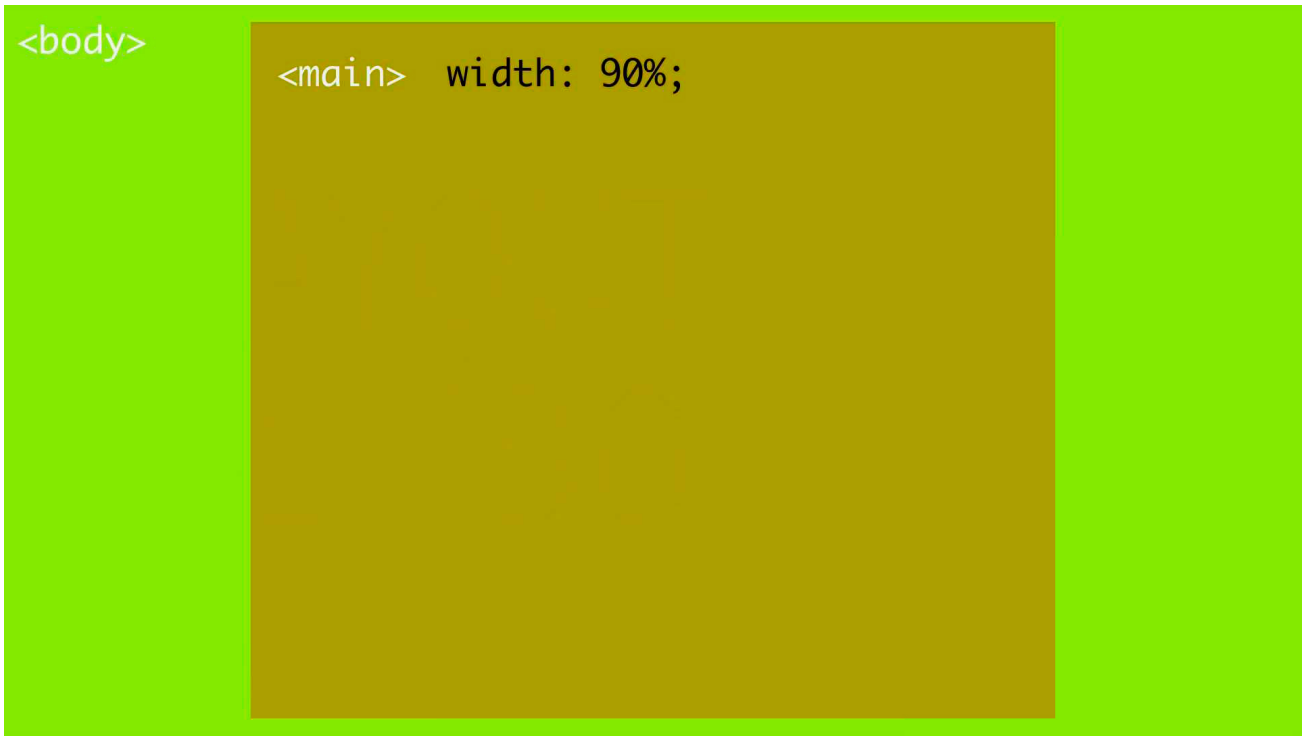
LAYOUT FLUÍDO

Continuando nossos estudos, agora de design responsivo, a gente vai se aprofundar um pouco nas questões do layout fluído, que é justamente o grande pilar da responsividade de nossas [páginas],certo? Para isso, peguei alguns exemplos pra gente dar uma estudada.

MEDIDAS FLEXÍVEIS

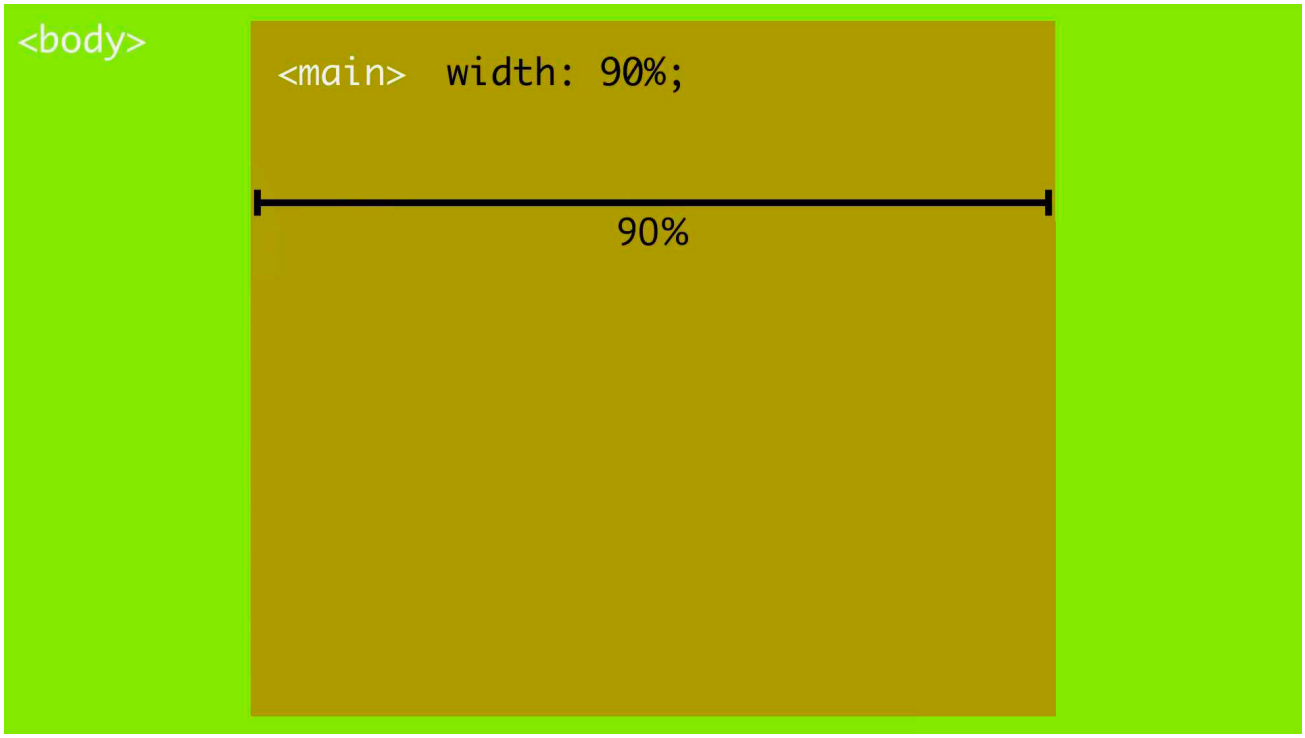
Primeiro a gente tem que lembrar o seguinte: que o ponto central do layout fluído são as medidas flexíveis. Então vamos ver várias medidas agora e como elas ajudam a implementar esse tipo de coisa. Lembrando que a porcentagem é a principal.

Mas como funcionam as porcentagens na web exatamente? Bom, pegue esse exemplo:

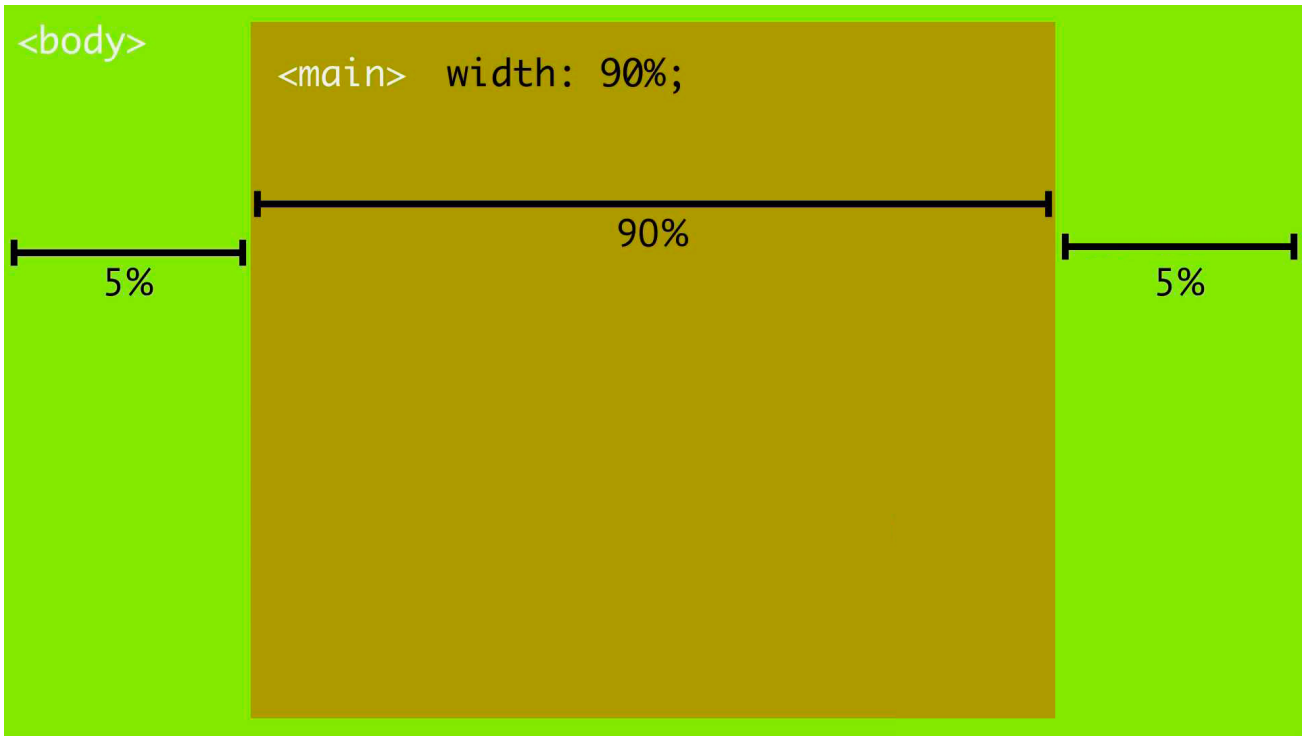


Imaginemos uma página com um `body` e dentro dessa página eu criei um elemento chamado de `main`. E aí, no `CSS`, estou encurtando o código aqui, eu coloco que a largura dele é de 90%. O que isso significa na prática? É 90% do que? Do `body`, da página inteira.

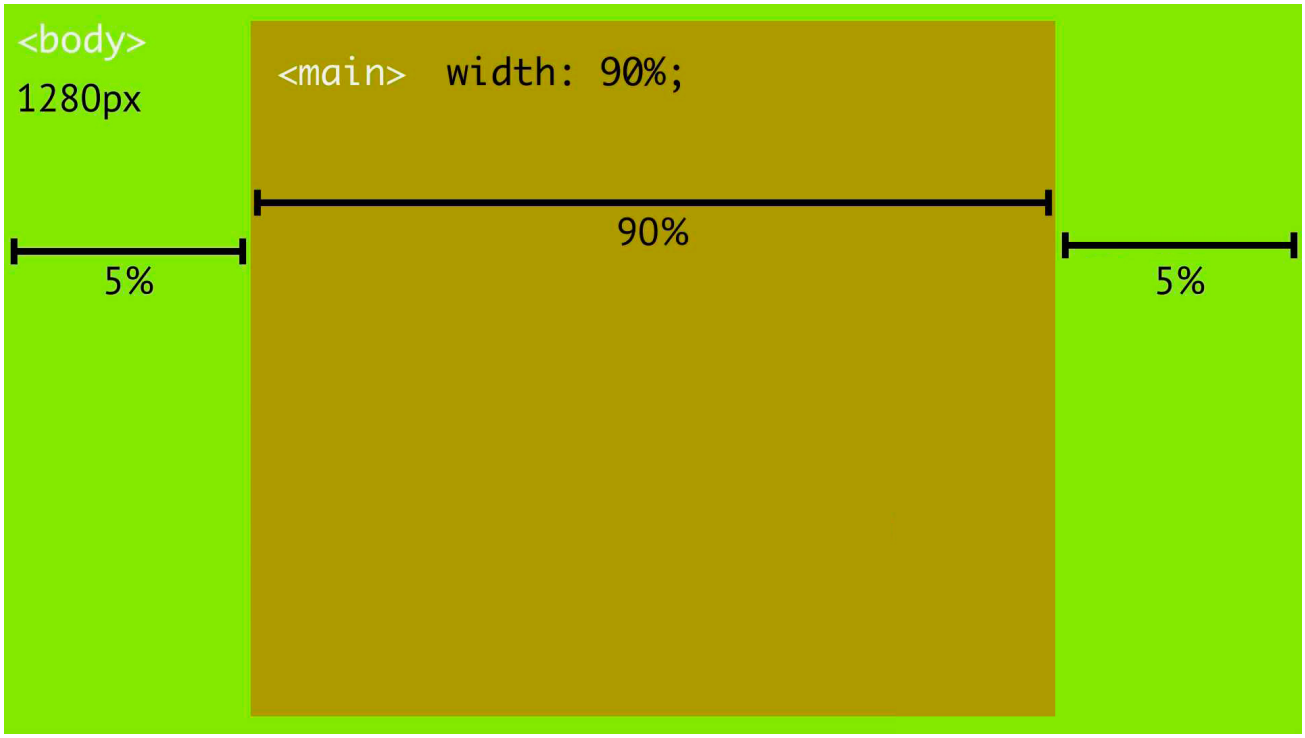
Se fôssemos colocar isso graficamente, significa, se eu centralizar esse elemento na tela que ele ocupa 90%,



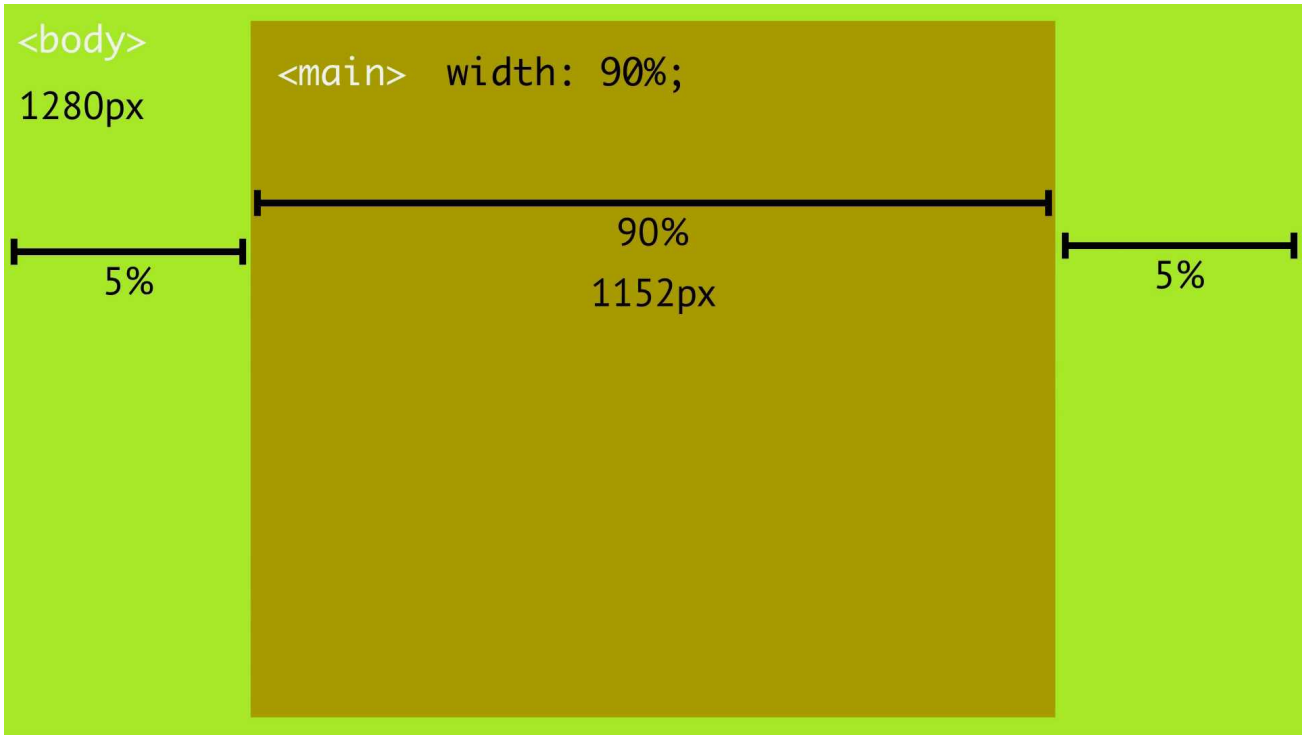
e tem margens de 5% dos dois lados.



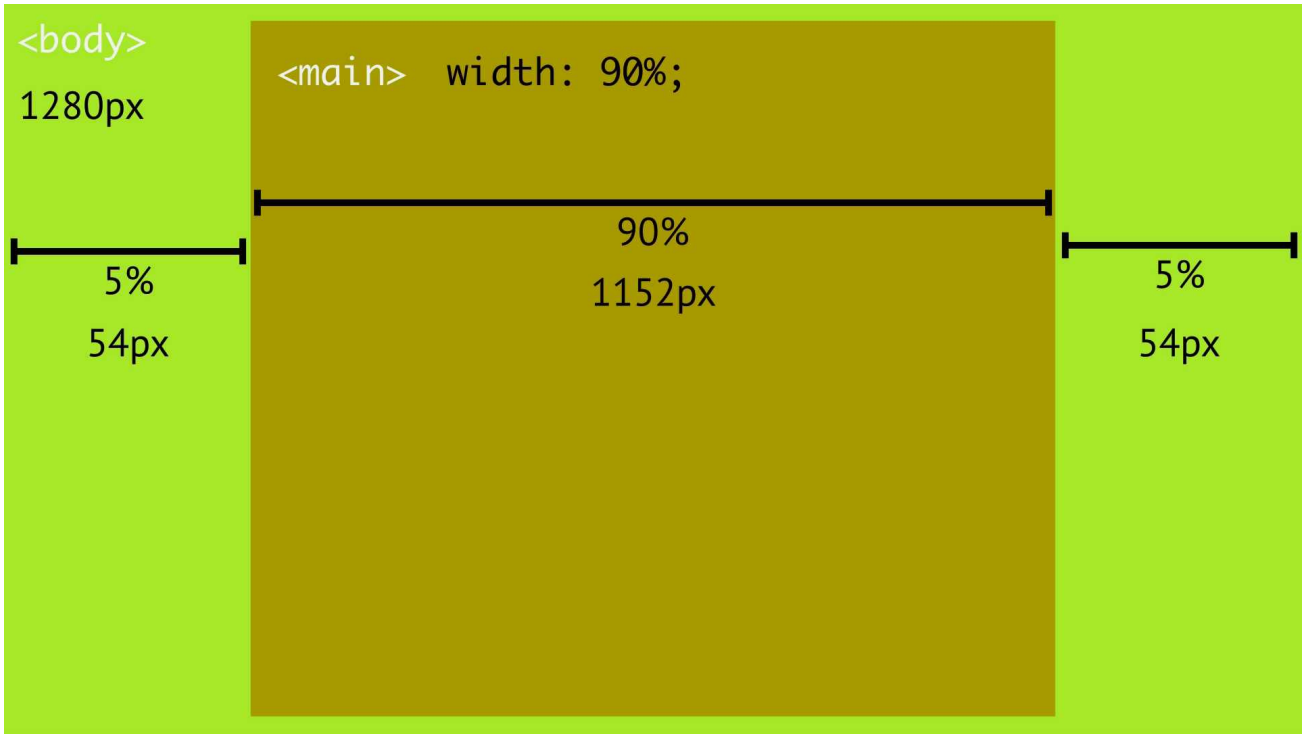
Em números, o que isso significa? Se abrimos em um `desktop`, por exemplo, com resolução de 1280 pixels,



isso significa que esse nosso `main` vai ter 1152 pixels, 90%,

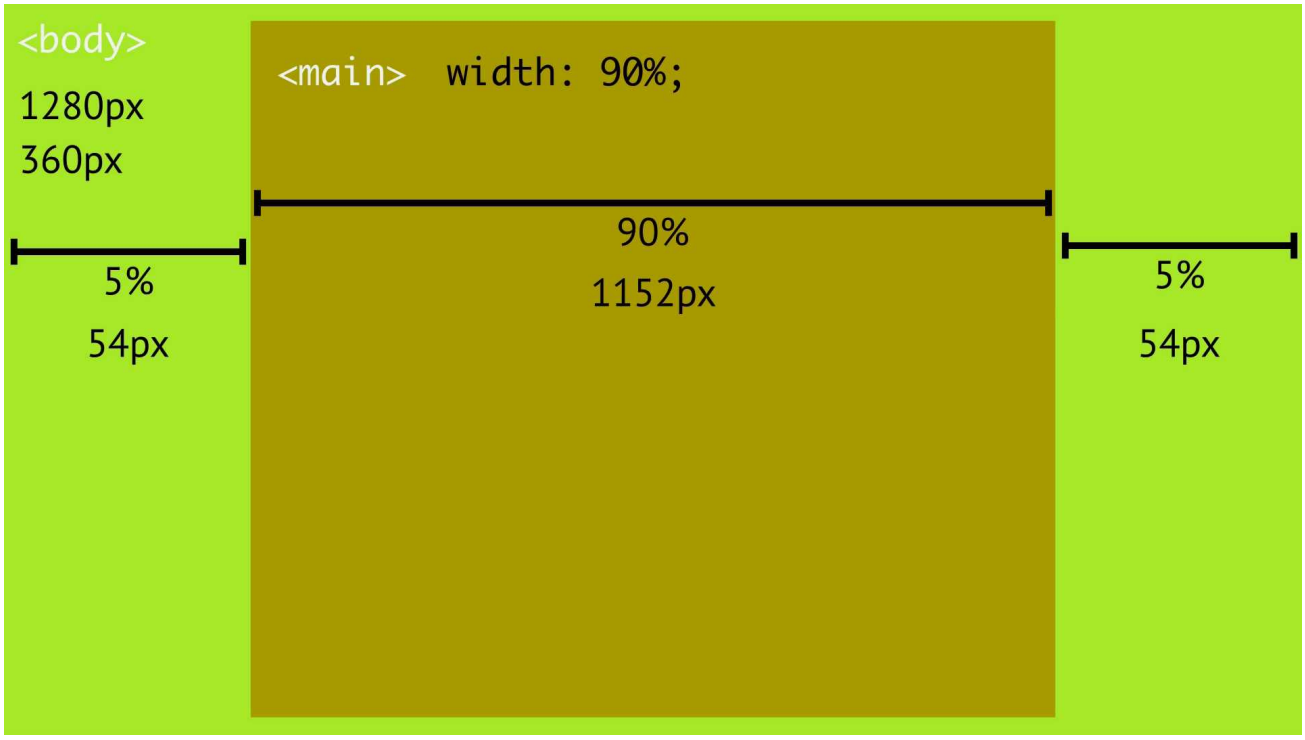


e cada uma das duas margens com 54 pixels:

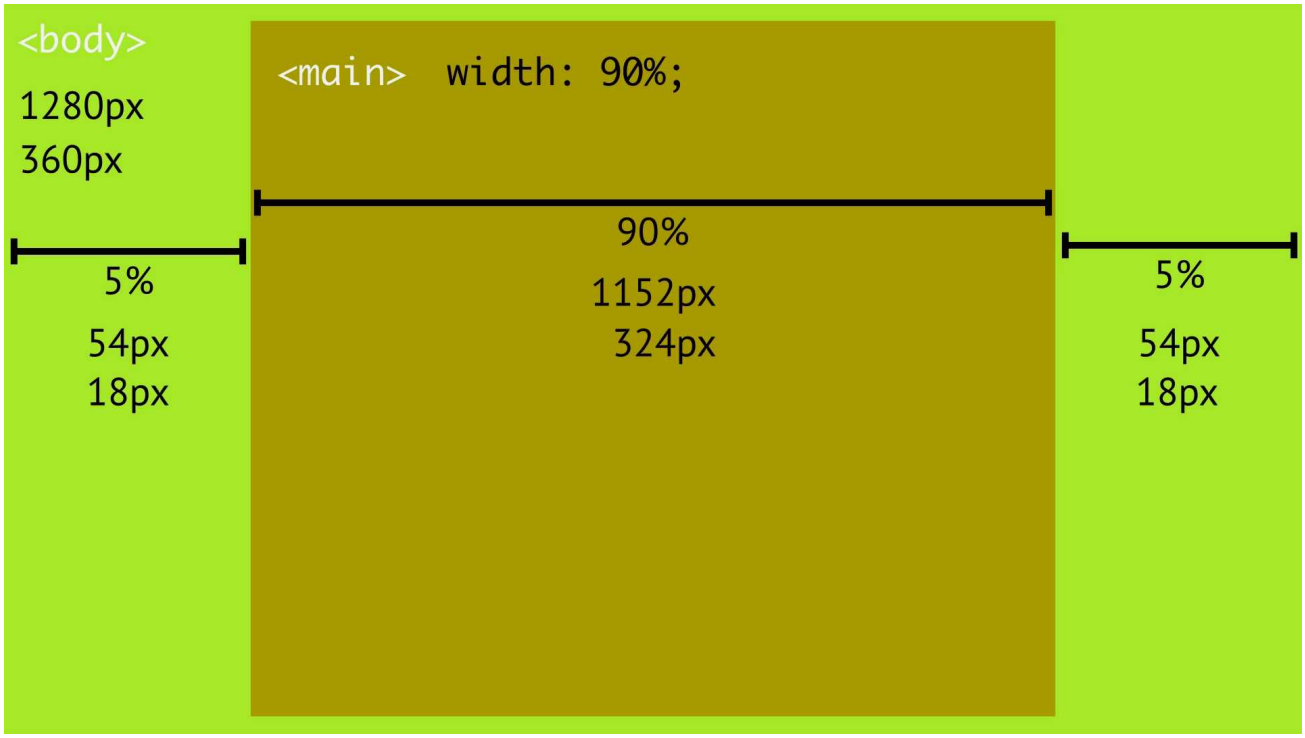


Mas como estamos trabalhando com porcentagens, a grande questão é que isso deve se adaptar a diferentes resoluções.

Então, se abrirmos em uma resolução menor, de 360 pixels, por exemplo,



Isso significa que o `main`, vai ter 324 pixels e as margens terão 18 pixels, cada.

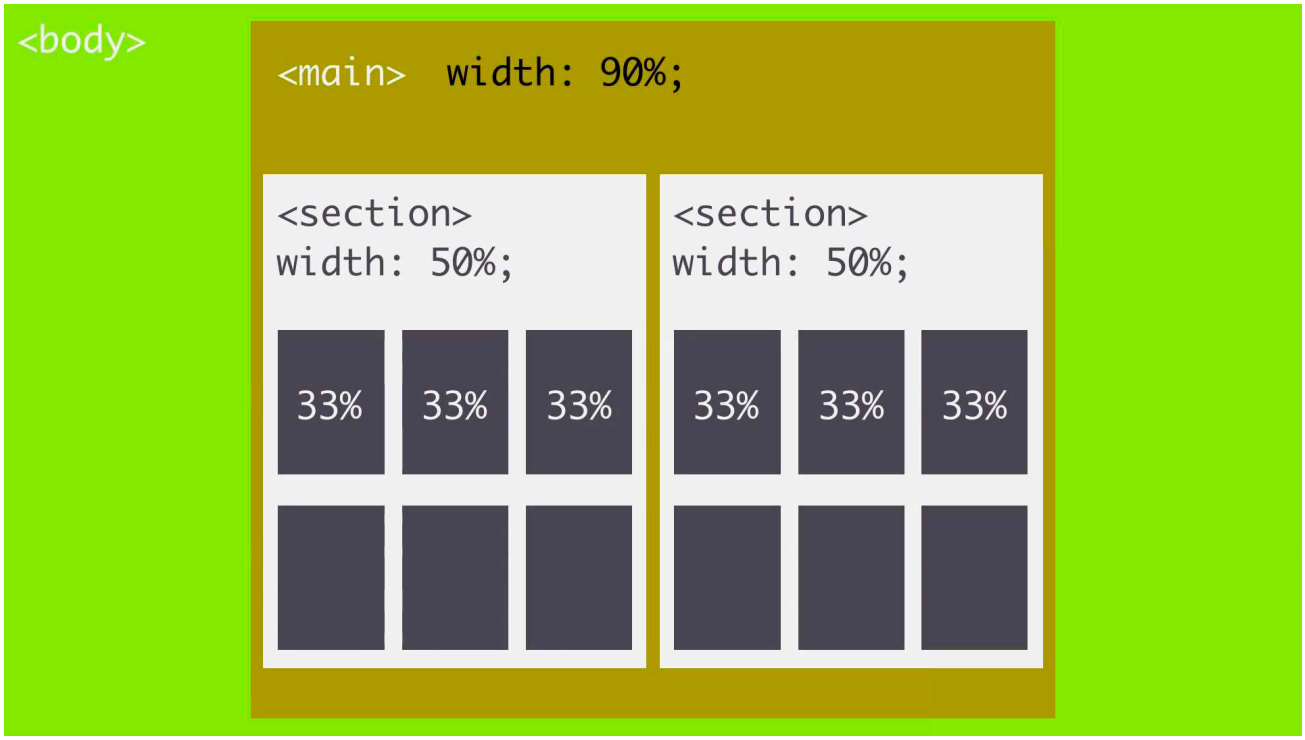


Ou seja, tudo é `pixel`, claro, a quantidade de pixels que será desenhada na tela. No entanto, por estarmos usando a porcentagem na definição do elemento, teremos esse cálculo feito automaticamente, conforme a resolução que estiver sendo usada.

Voltando para o `main`, vamos colocar dois elementos para definir duas colunas diferentes. Por exemplo,

Para definir duas colunas, fácil, 50%. Mas aqui tem um ponto importante: cuidado ao definir dois elementos dentro do `main`, dividindo o meio, é 50% do `main` e não 50% do `body`. O que se quer dizer com isso? Não se coloca, por exemplo, 45%, o que seria a metade de 90%, mas sim, coloca-se 50% do `body`. Isso quer dizer que, para se chegar ao tamanho final da `section`, o navegador vai calcular o tamanho do `body`, o tamanho do `main` e o tamanho da `section`, multiplicando as porcentagens de cada um deles.

Por isso é fácil, se eu quiser dentro de cada uma dessas `sections` um `grid` de três colunas, que estão dentro de um elemento composto de duas colunas, dentro de outro elemento que é 90%, parece meio confuso, mas não, é fácil: três colunas, 33%:



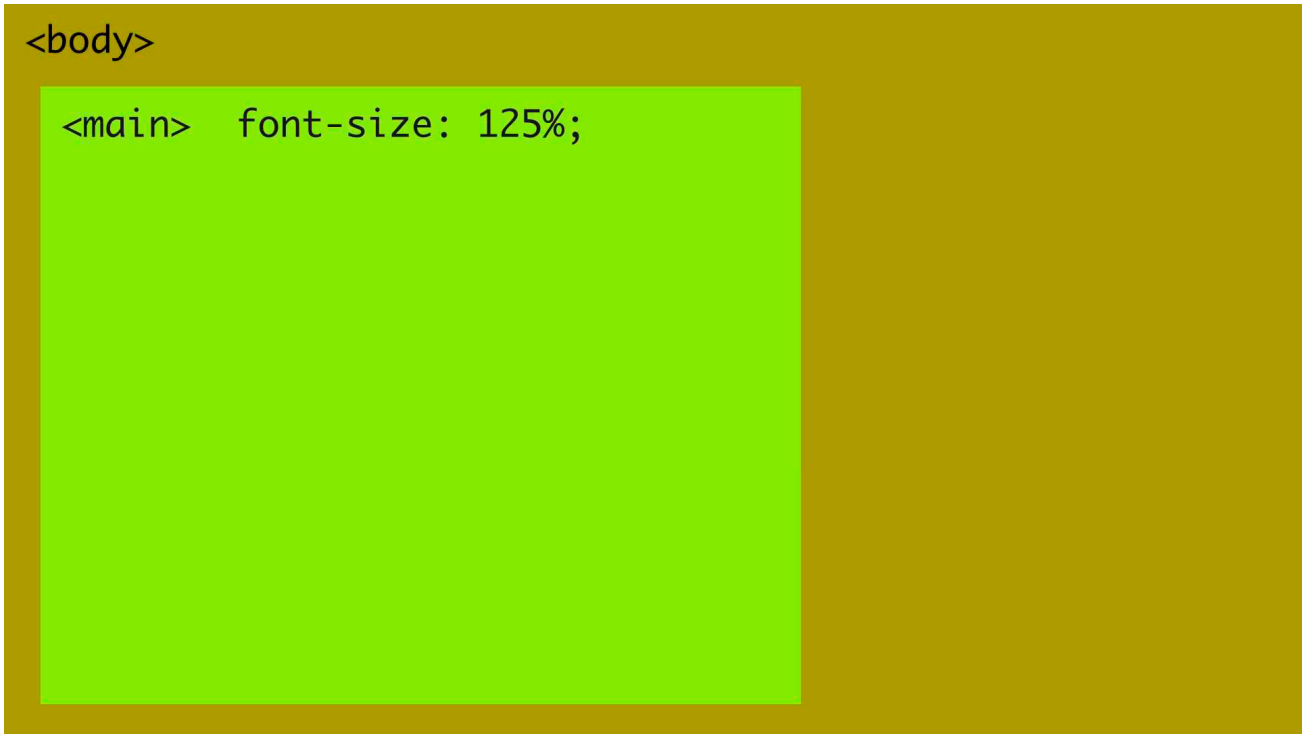
Então, divide-se em 33% cada um desses elementos. Ou seja, estamos usando porcentagens para definir os elementos trabalhados.

As porcentagens podem ser utilizadas também em outros contextos, em especial, o `font-size` .

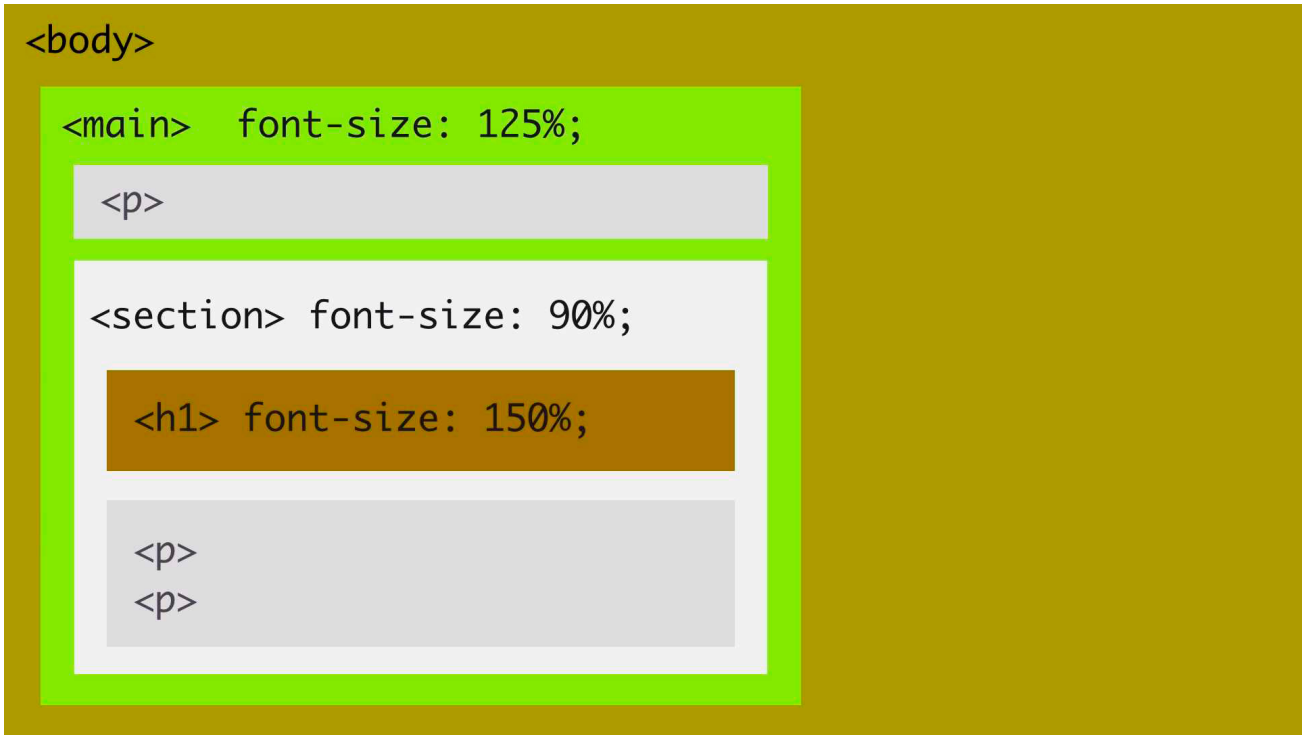
FONT SIZE %

No `font-size` as porcentagens têm outro significado. Antes falávamos de significado da largura dos elementos, agora tem-se uma `font-size` ou porcentagem que quer dizer outra coisa.

Tem-se aqui um exemplo:



É o mesmo `body` com alguns outros elementos lá dentro. Imagine um elemento `main` onde se define um `font-size` de 125%. Depois se adiciona um parágrafo, sem nenhum `font-size`, ele vai herdar o `font-size`. Depois coloca-se uma `section` com `font-size` de 90% e um título com `font-size` de 150%, lá dentro da `section`, e mais dois parágrafos sem `font-size`:

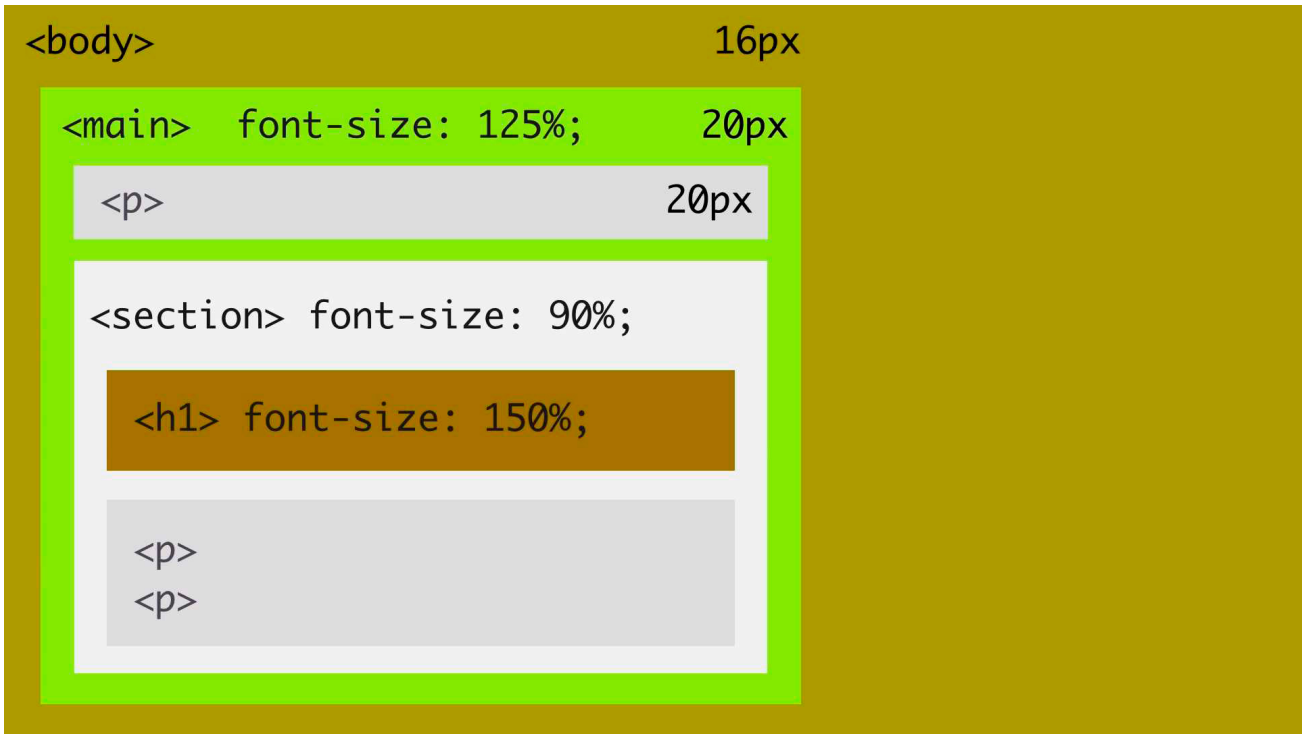


Partindo deste caso, a questão é: qual será o tamanho das fontes de cada um desses elementos? Quando se fala em 125%, isto é em relação a quê?

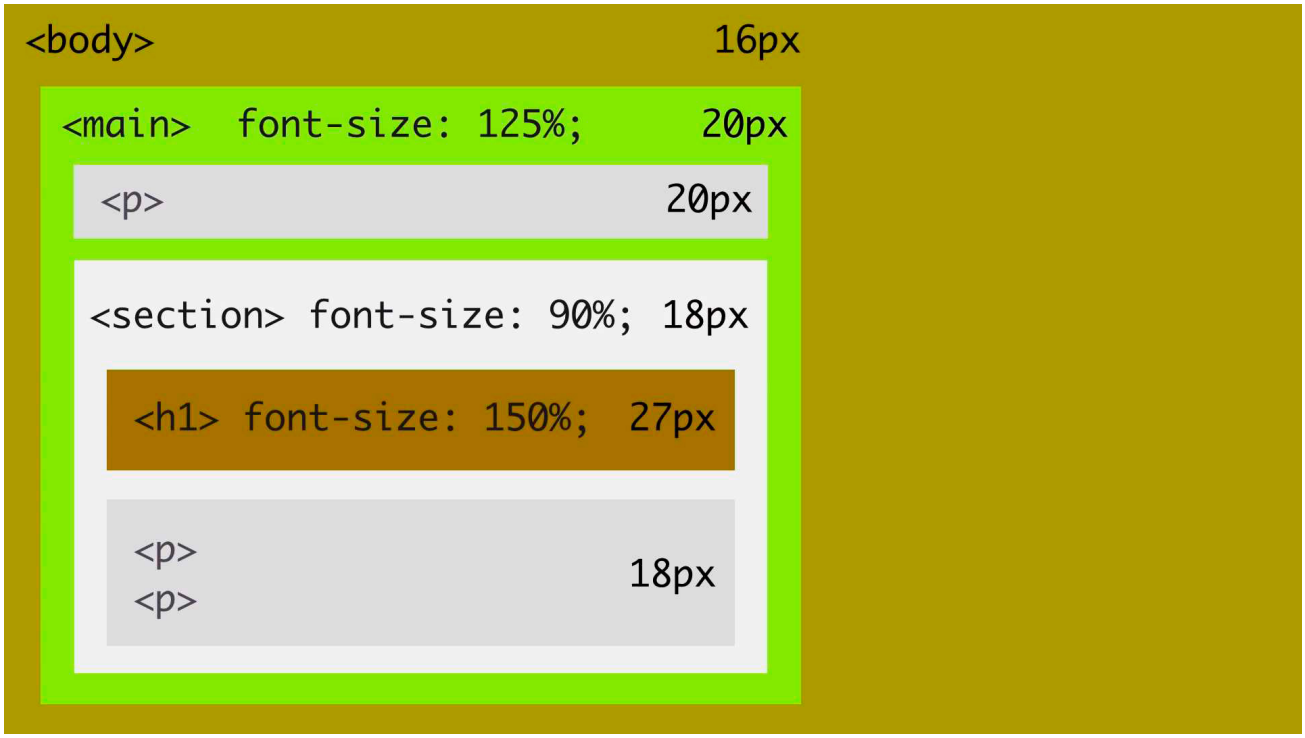
O grande ponto é o seguinte: quando se usa porcentagem nos `font-size`, está se usando uma multiplicação com relação ao `font-size` do elemento `pai`, ou seja, se colocarmos 125% no `main`, estamos dizendo que ele é 125% maior do que a fonte do `body`.

Mas aí, pode-se argumentar que o `body` não teve fonte. Se não se colocar nenhuma fonte no `body`, assume-se que fonte-padrão do navegador, que vale na `tag HTML`, na `tag] body`, é 16 pixels.

E então faremos as contas levando isso em consideração: 1,25 vezes 16 chega-se em 20 pixels e o parágrafo herda esses 20 pixels, porque não tem nenhuma definição de fonte.



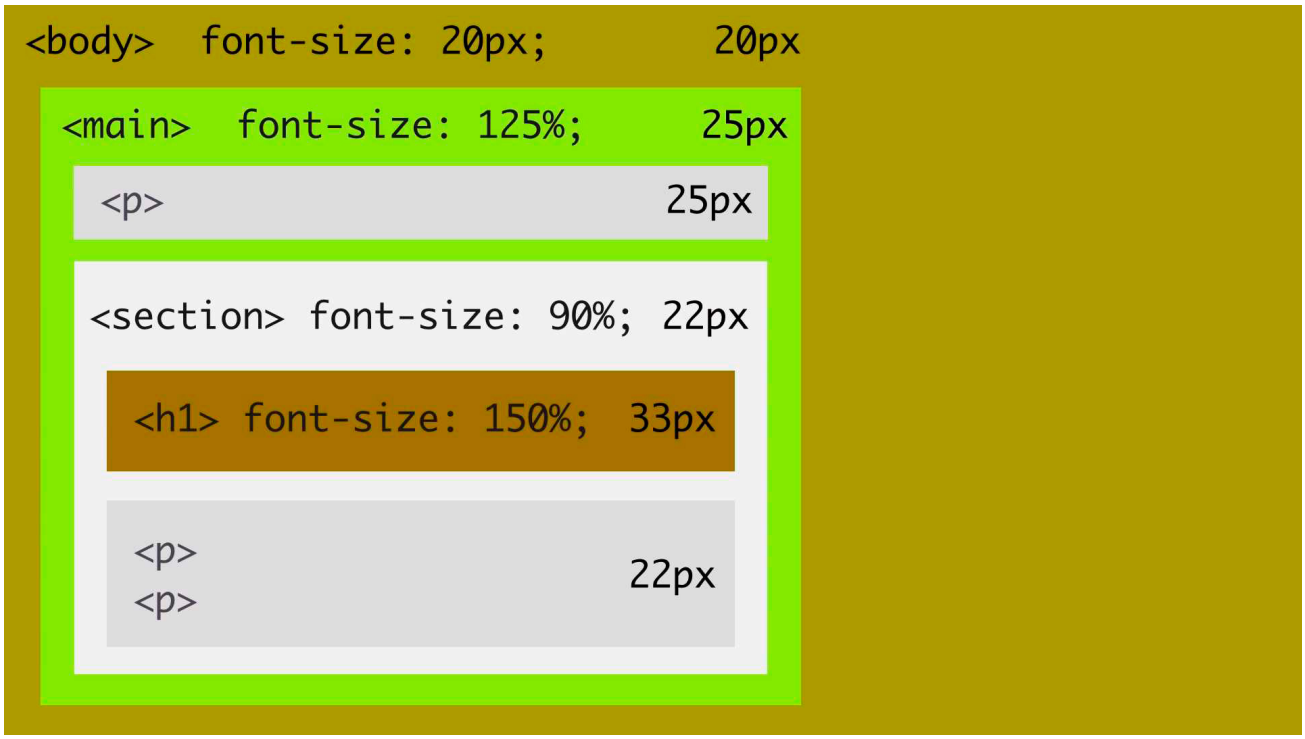
Aqui, uma definição importante ao chegar na `section`: é 90% do valor que temos no elemento-pai, ou seja, 90% dos 20 pixels, daí o resultado de 18 pixels. E aí, novamente, quando se vai calcular no `h1`, é 150 vezes o elemento-pai imediato, que é 18 pixels, resultando nos 27 pixels, e o parágrafo herdará os 18 pixels.



Então, estamos trabalhando com porcentagem, multiplicando os elementos-pai até chegar a um valor em `pixel`, no valor real, que será desenhada aquela fonte na tela.

Um ponto importante: por que falamos que a porcentagem, nesse caso das fontes, é uma medida flexível? Porque aqui tem o seguinte detalhe: se mudamos a fonte inicial de 16 pixels lá no `body`, todo o cálculo que é feito em porcentagem embaixo, mudará também. Ou seja, é muito fácil se alterar algum elemento e causar uma alteração em cascata na página. Isso é bem interessante em uma página responsiva, por exemplo, para evitar a redefinição do tamanho de cada um dos elementos o tempo todo.

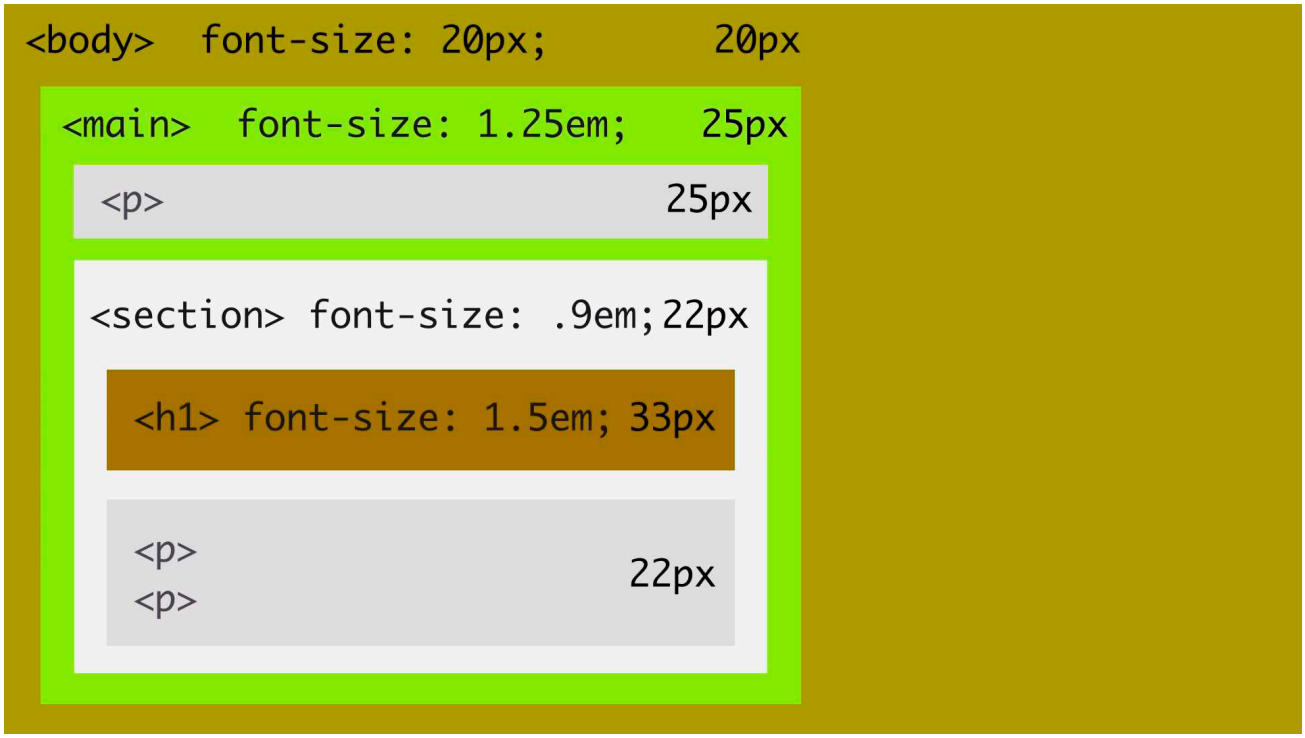
Vamos ver um caso na figura seguinte:



Se trocarmos para 20 pixels lá no `body`, pode-se perceber que todos os elementos trocaram: o `main` fica com 25 pixels, a `section` fica com 22 pixels, o `h1` com 33 pixels e assim por diante. Tudo é recalculado automaticamente.

Por isso, fala-se que a porcentagem na fonte facilita a mudança em um lugar somente, a qual será replicada em cascata, aumentando-se proporcionalmente todas as fontes da página, alterando-se apenas a fonte do `body`.

Outro ponto interessante. Veja-se a figura a seguir:



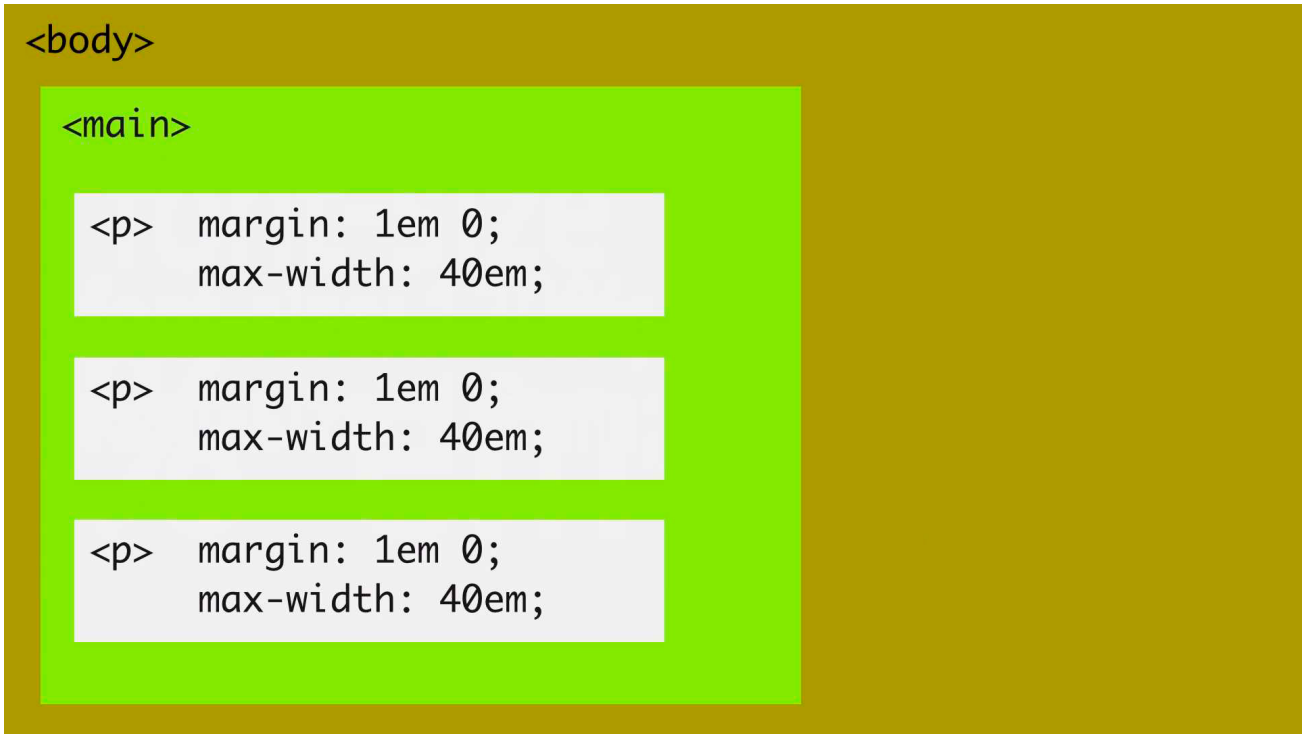
Substitui-se as porcentagens para `em`, que é outro tipo de medida utilizada nas fontes. Repare que foi mudado apenas um detalhe no `main`: em vez de 125%, foi utilizado 1.25em. O `font-size` da `section`, transformou-se em .9em e no `h1`, 1.5em. O mesmo valor usado em porcentagem, é utilizado em `em`. Notando que o cálculo final em pixels é o mesmo.

FONT-SIZE: 120% = FONT-SIZE: 1.2em

Isso quer dizer que a porcentagem e o `em` têm o mesmo significado, ou seja, 120% e 1.2em querem dizer a mesma coisa.

Esse raciocínio vale quando se está utilizando `font-size`, ou seja, nas fontes a porcentagem e o `em` têm o mesmo significado. Isso é importante porque se a porcentagem é utilizada nas larguras, como visto anteriormente, está claro que seu significado será diferente, quando, então, indicará a largura do elemento-pai.

Interessante é que o `em` pode ser utilizado fora dos `font-size`, quando então pode-se ver a diferença entre as duas medidas.



Por exemplo, na Fig.20, onde se tem um `body`, um `main` e alguns parágrafos lá dentro. Repare que as margens são definidas e o `width` do parágrafo utilizando `em`.

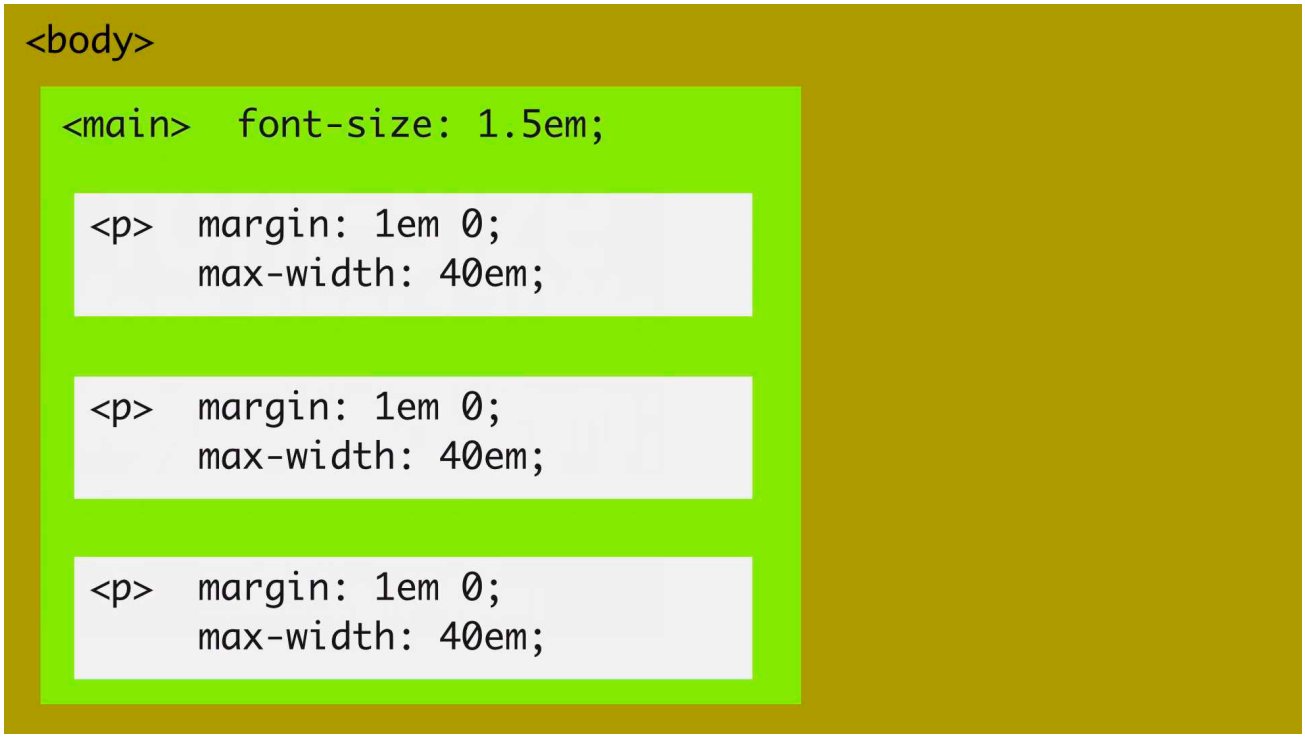
O que significa trabalhar com `em`? Isso quer dizer que uma margem de 1em é proporcional ao texto, ao tamanho da fonte, que estiver sendo utilizada no elemento em determinado momento.

Lembre-se que o valor-padrão é 16 pixels, no caso de não haver nada definido.

Assim, tem-se uma margem proporcional ao tamanho da fonte. A mesma coisa ao `width`: se o `max-with` é 40em, seu aumento será proporcional ao aumento da fonte.

Isso é interessante porque, se repararmos, a margem entre os parágrafos deve mudar proporcionalmente ao tamanho do texto, afinal é uma medida textual. Se tivermos uma fonte muito grande, a margem será maior. O [`line-date`] deverá aumentar também, assim como o tamanho da linha. Ou seja, sempre que tiver coisas que são relacionadas ao tamanho do texto, à proporção que tem a ver com tamanho do texto, o `em` é uma boa medida para isso.

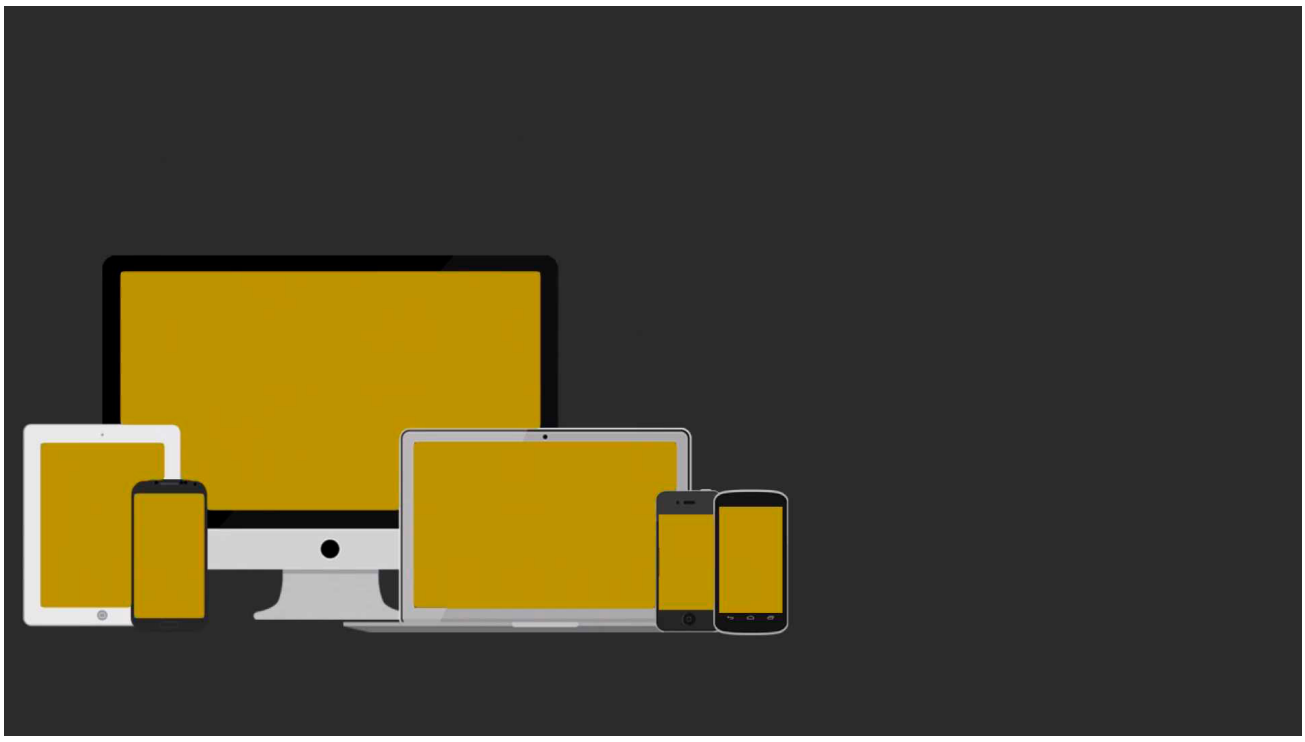
E teremos o mesmo efeito de antes:



Se trocarmos o `em` do elemento-pai, por exemplo, o `main` para 1.5em, o tamanho de todos os elementos seguintes é afetado internamente. Nota-se que o cálculo do `em` é proporcional, como visto antes. A grande diferença é que o `em`, nesse caso, está sendo utilizado em elementos que não são apenas `font-size`, mas são propriedade, como margem e `width`, que devem ser proporcionais ao parágrafo e ao texto que se está utilizando. Isso é o nosso `em`.

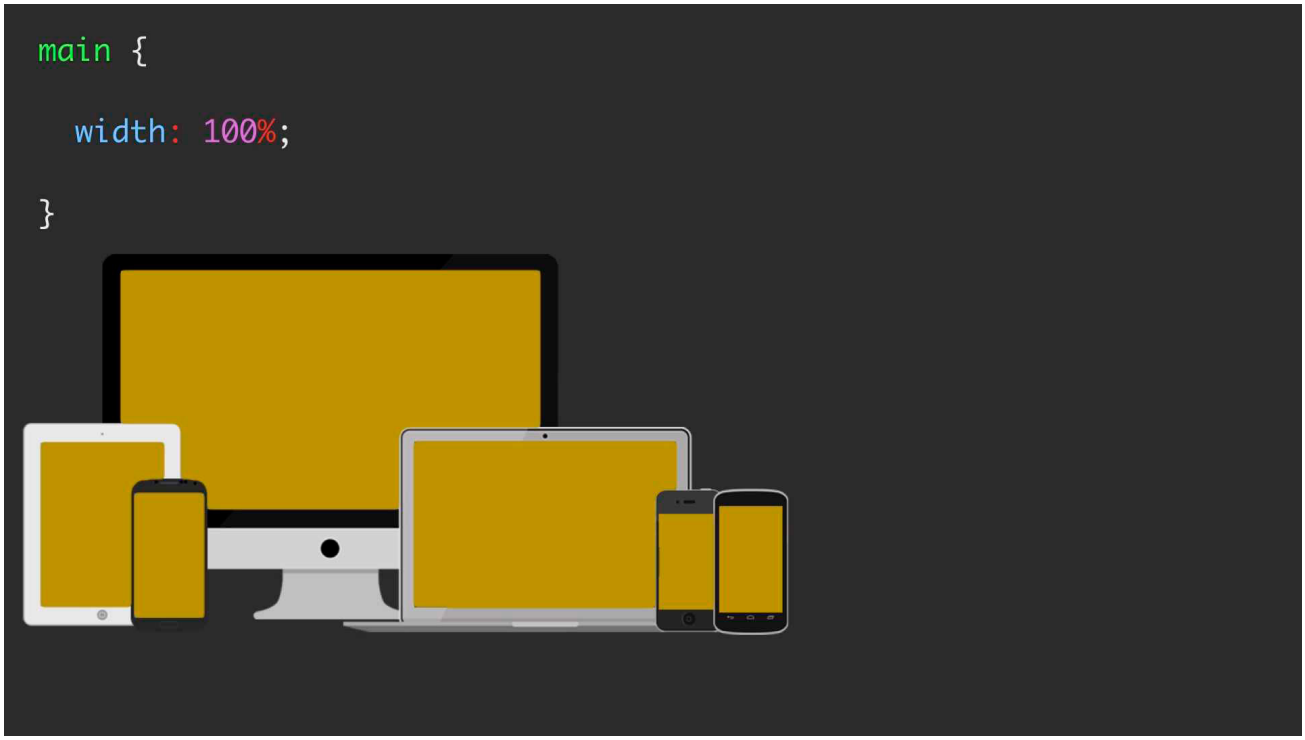
A aplicação de tudo isso na prática, porcentagem, `em`, exige alguns detalhes a mais, os quais serão demonstrados a seguir, que devem ser tratados antes de partirmos para os exercícios, aplicando as porcentagens em alguns cenários mais interessantes.

TÉCNICAS DE DESIGN FLEXÍVEL



Imagine, por exemplo, uma página já construída, a qual se aplica a vários dispositivos, desde o pequeno celular até um computador, um notebook ou um computador maior, tudo isso funcionando com `width` de 100%, para ocupar largura

completa.



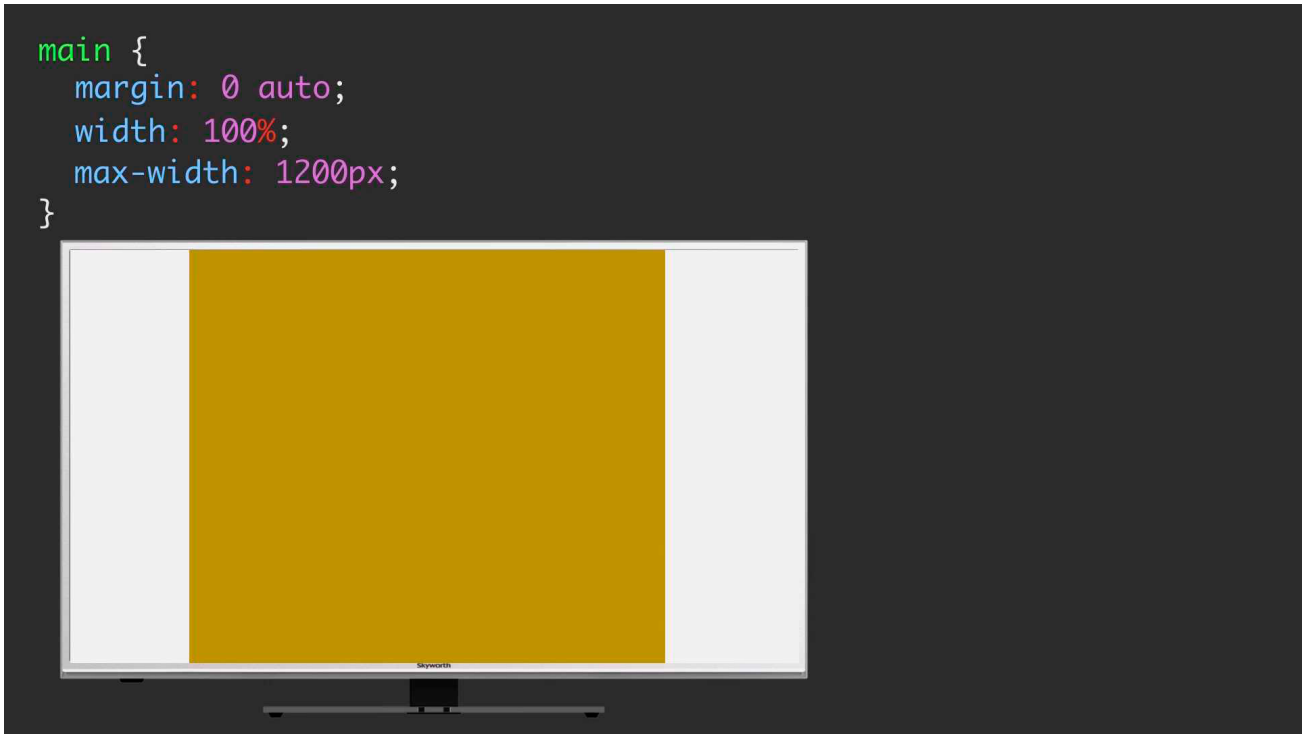
Porém, pretende-se colocar a página em uma TV gigantesca, com 4K, com 3800 pixels de largura.

O `width` de 100% vai funcionar, ocupará tudo, mas talvez não fique tão interessante, tão bonito, por quê? Ficará meio largo demais, pois a TV não é tão otimizada para o tamanho do `site` que fizemos.

Está claro que é possível ajustar o site para ser utilizado em todos os meios possíveis, desde um minúsculo `smart` em relógios, até uma TV gigantesca, enfim, porém isso nem sempre é prático. Pode ser que se queira limitar até onde o site fique responsivo, isto é, a face [...] possui uma variedade de resoluções, mas quando chegar em um limite muito grande, por exemplo, uma TV gigantesca, decide-se parar [...]. Como isso ocorre? Imagine-se que o site responsivo, quando chegar na TV grande, deve ficar centralizado no meio de uma largura fixa. Coloca-se um limite no responsivo.

Isso é possível, bastando trabalhar com `max-width`, que é onde se define o tamanho máximo que a página terá, [...] sem ultrapassá-lo. Ele será responsivo para valores abaixo do valor máximo fixado, do qual não passará.

Coloca-se margens zero auto para o site ficar centralizado.

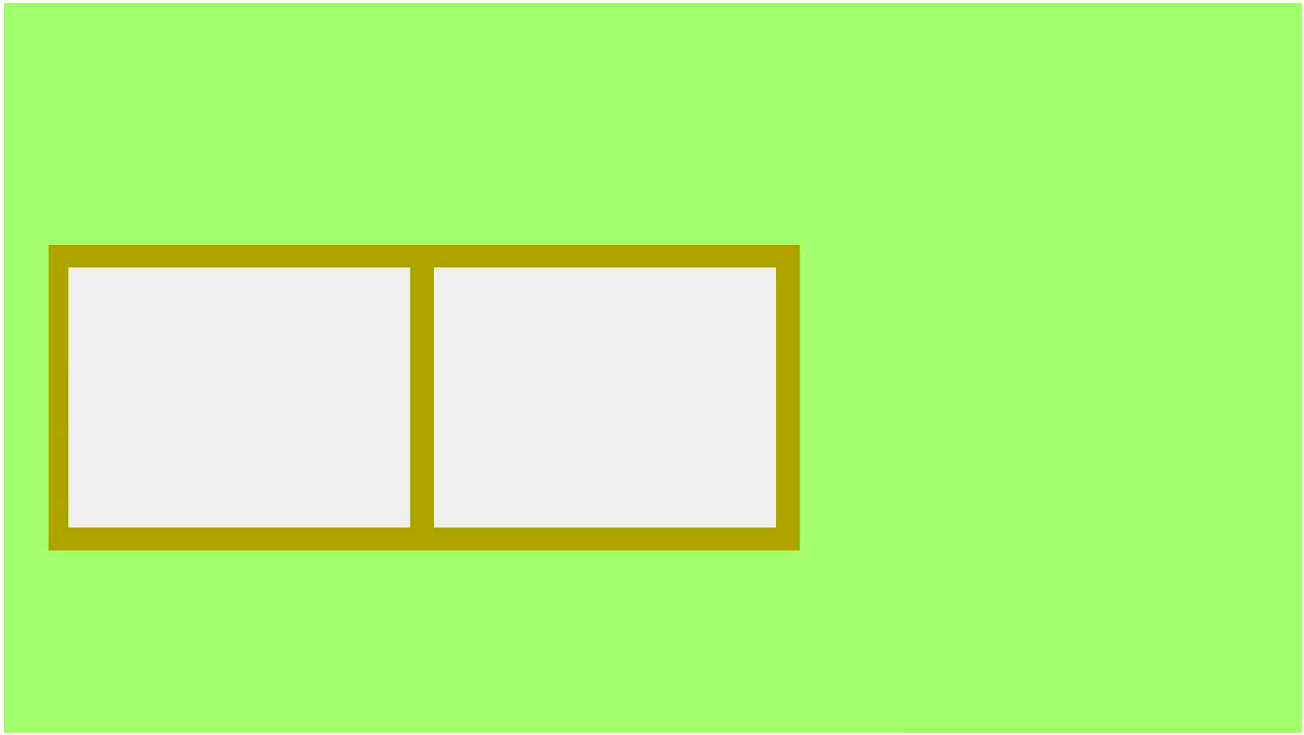


Através do `max-width`, e também do `min-width` (que funciona ao contrário), limita-se as bordas até quanto se entender necessário para o objetivo que se deseja para o [design] responsivo funcionar.

Isso seria interessante também, porque em termos práticos é impossível atingir todos os dispositivos existentes, por isso a necessidade de limitar para os dispositivos mais comuns.

Nesse caso, talvez 1200 ou 1400 pixels seja uma medida suficiente para 99.9% dos dispositivos que existem atualmente. É claro que, futuramente, tais números possam ser revistos, mediante o avanço tecnológico.

Agora, vejamos este caso.

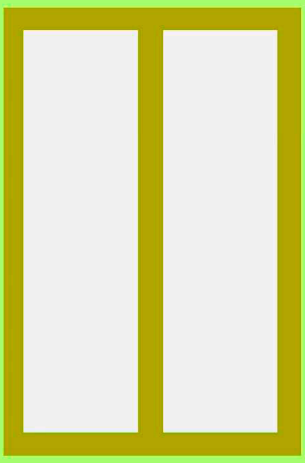


Como se implementa um design com duas colunas em CSS? Coloca-se uma [classe painel], um `width` de 50%, um `float left`, e os painéis ficarão flutuando lado a lado, formando duas colunas.

```
.painel {
  width: 50%;
  float: left;
}
```

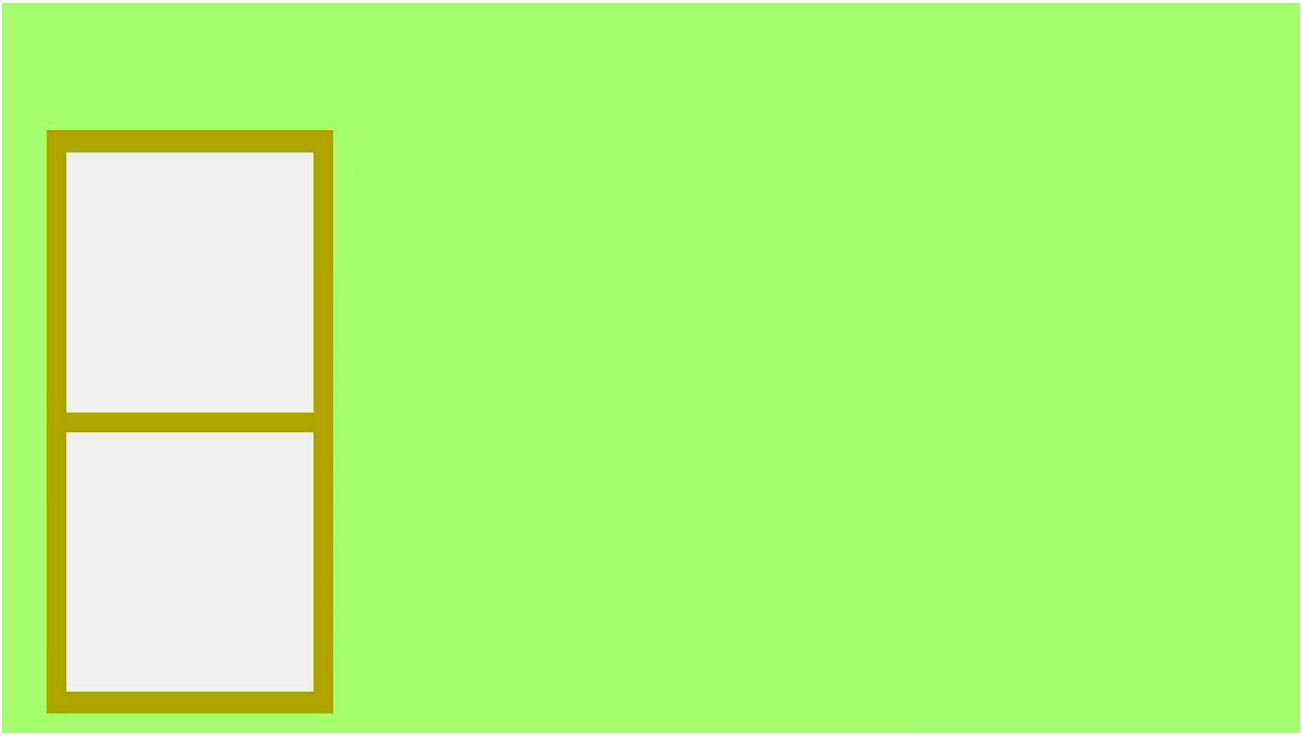
A discussão gira em torno da seguinte questão: quando se diminui a resolução para um tamanho minúsculo, a `width` de 50% será respeitada, mas o tamanho talvez fique pequeno demais, muito apertado.

```
.painel {
  width: 50%;
  float: left;
}
```

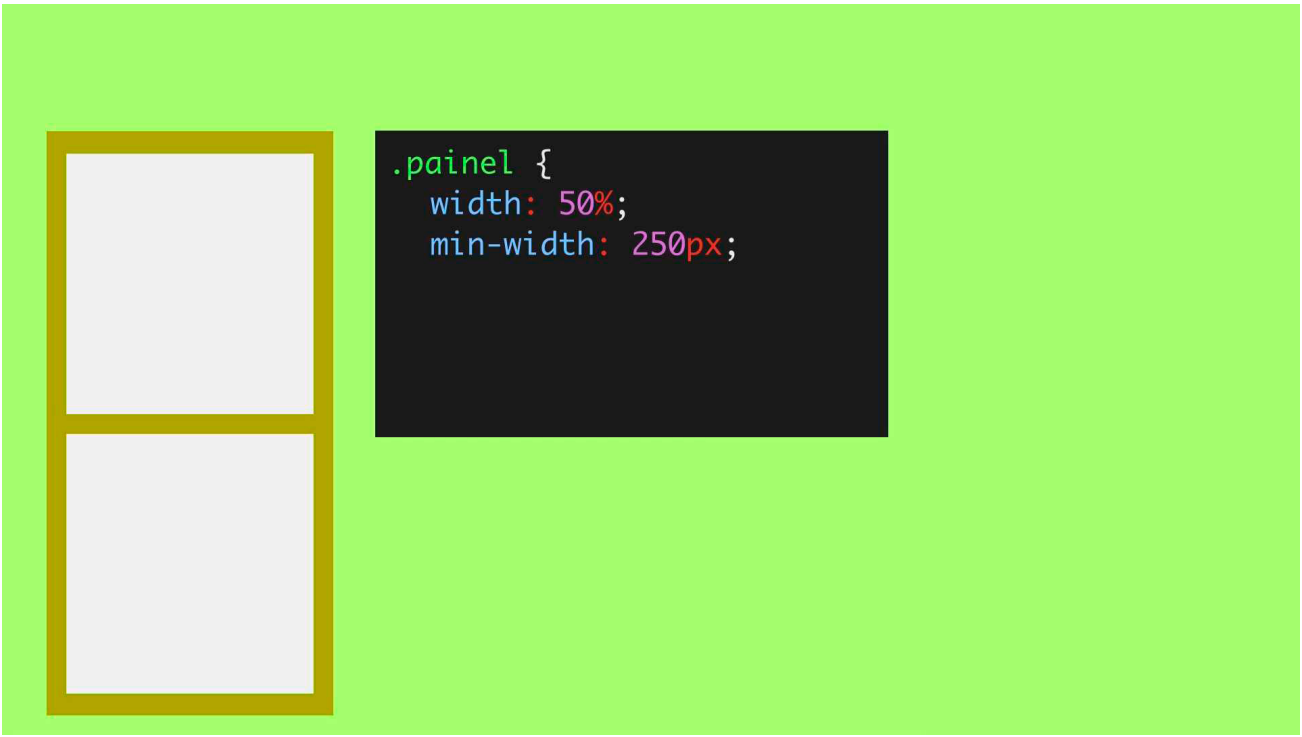


O que se pode fazer para resolver o problema? De fato, o site deve continuar ocupando os 50%, isto é, que ele seja flexível de acordo com a resolução, mas, ao chegar em resoluções muito pequenas, queremos que o site fique com um tamanho mínimo, impedindo que fique muito pequeno.

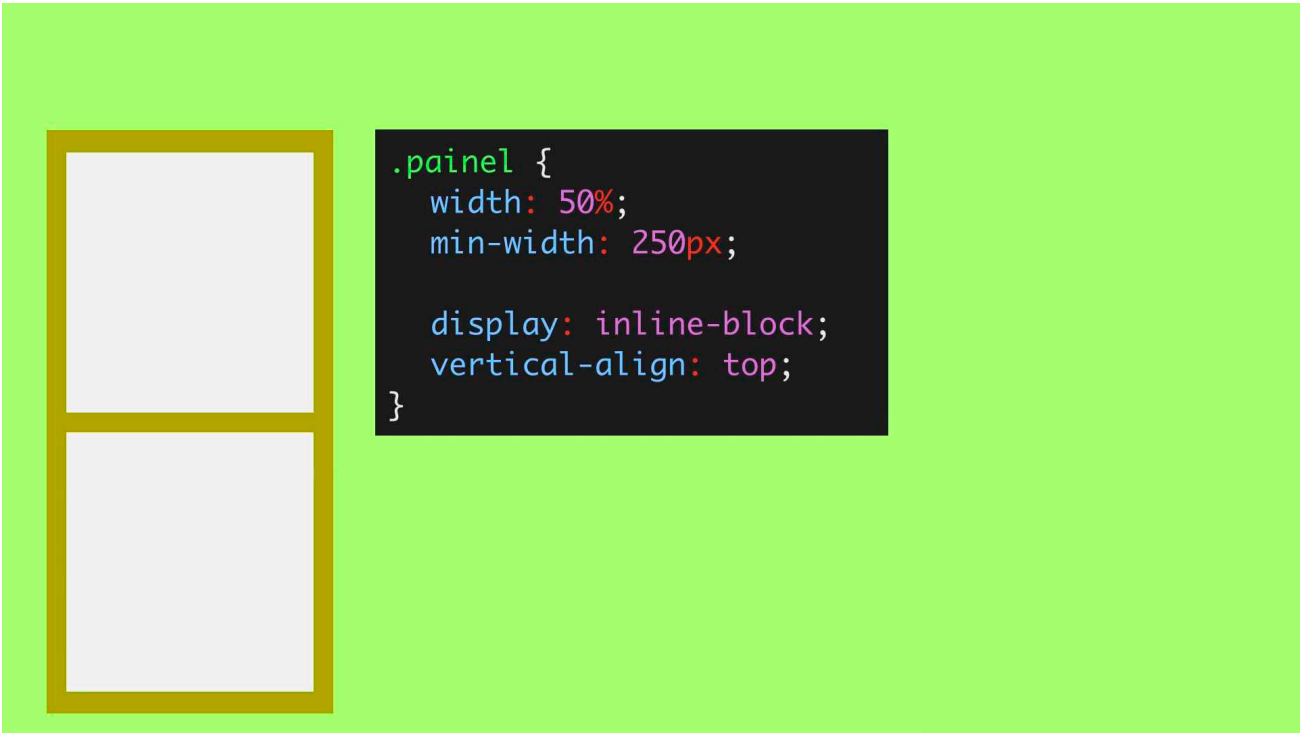
Graficamente seria o seguinte: coloca-se duas colunas lado a lado e quando o site atingir o tamanho mínimo, uma coluna escorrega para baixo da outra sem ficar muito apertado.



Essa solução pode ser implementada, em `CSS` da seguinte forma: utililza-se classe painel, `width` 50% e um `min-width` de 250 pixels. Isso significa que o site vai ficar com 50% em várias resoluções, porém nunca abaixo de 250 pixels, se ficar abaixo disso, uma das colunas escorrega para a linha de baixo.



O comportamento de escorregar uma das colunas para a linha de baixo, pode ser feito de diferentes formas, como, por exemplo, um `inline block` , alinhado no topo.



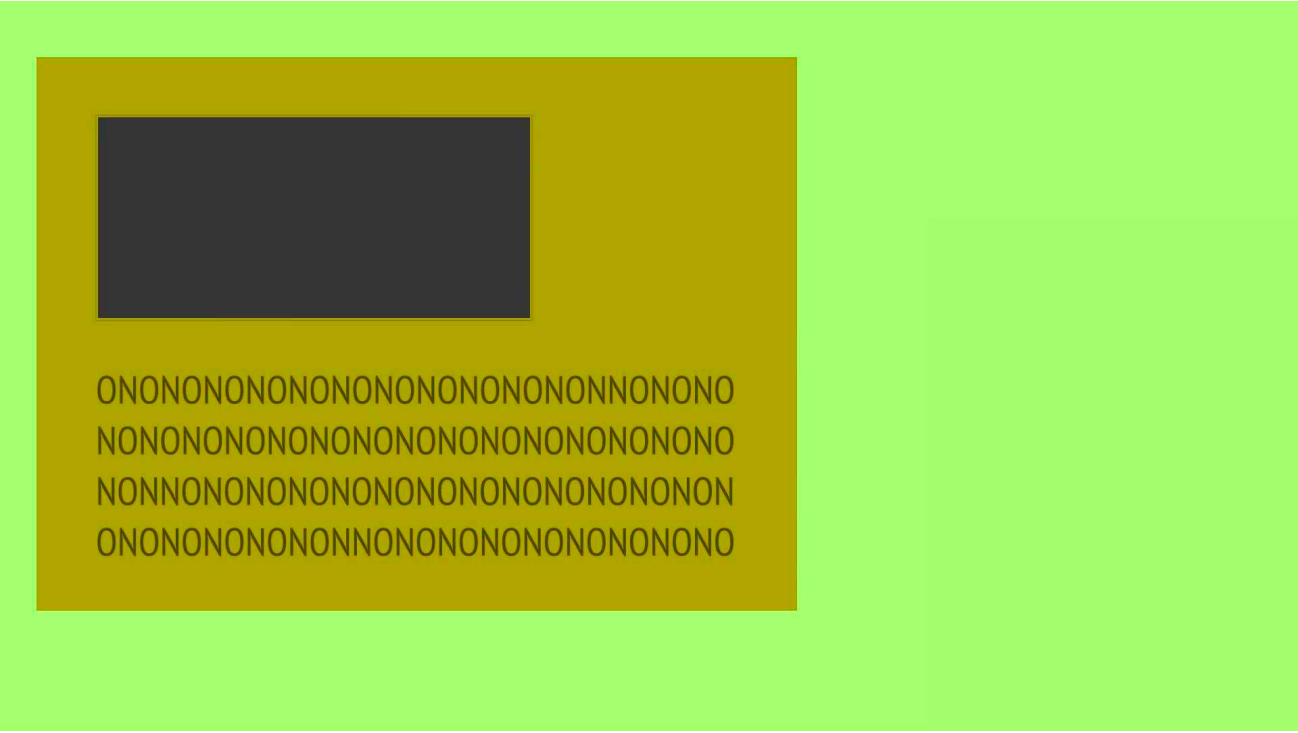
Aqui, a ideia é demonstrar que tanto o `min-width`, quanto o do `max-width`, como no exemplo anterior, podem ser boas ferramentas para ajudar a limitar a responsividade do site, nos extremos onde eles não fazem mais sentido.

IMAGENS FLUÍDAS

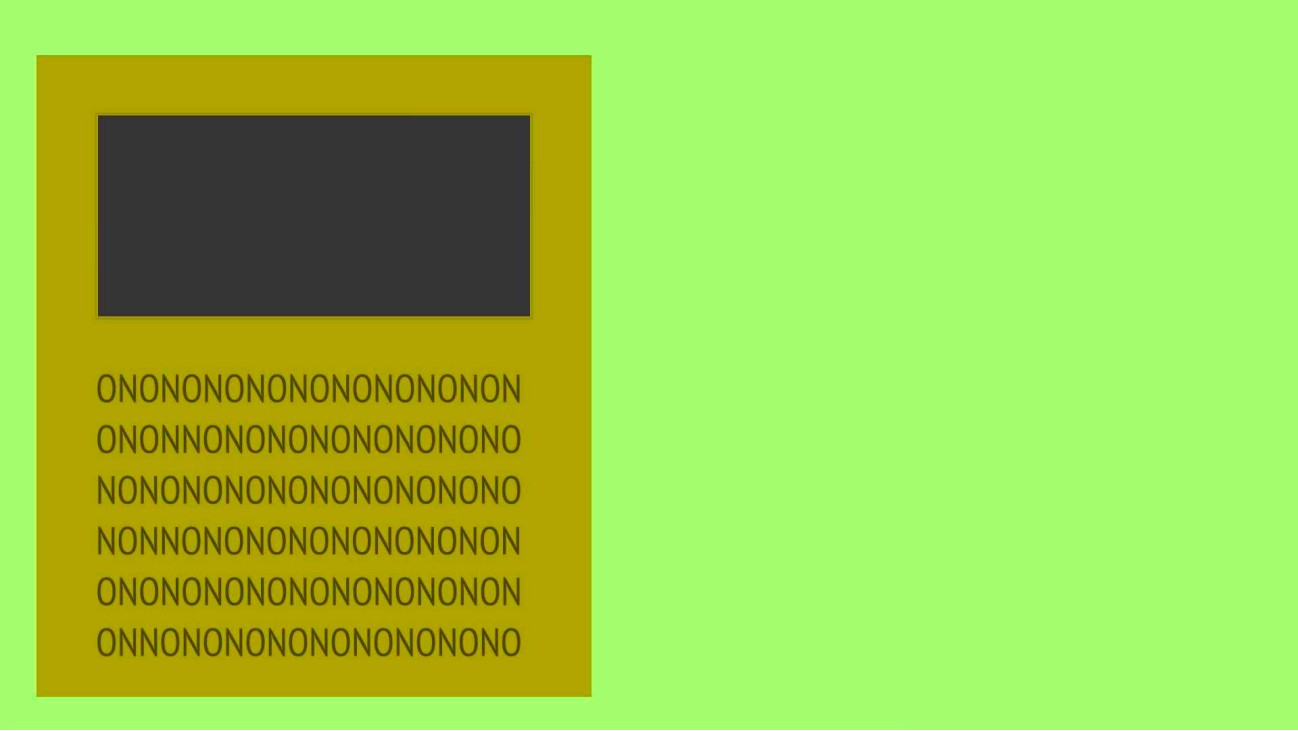
Por último, para continuarmos essa parte básica de layout fluído, vamos falar de imagens fluídas.

Pode-se colocar imagens na página criada, mas estas costumam ser o grande inimigo do design responsivo, porque se pensarmos bem, considerando que todos os elementos possuem medidas flexíveis, a imagem é uma coisa que basicamente se constitui de pixels, mas como estamos tratando de design responsivo, não é possível usar pixels fixos, porém somente percentuais flexíveis. E as imagens não tem como: são transportadas do Photoshop, por exemplo, em pixels e como poderia ser transportada a imagem em pixels para o mundo das formas flexíveis do design responsivo?

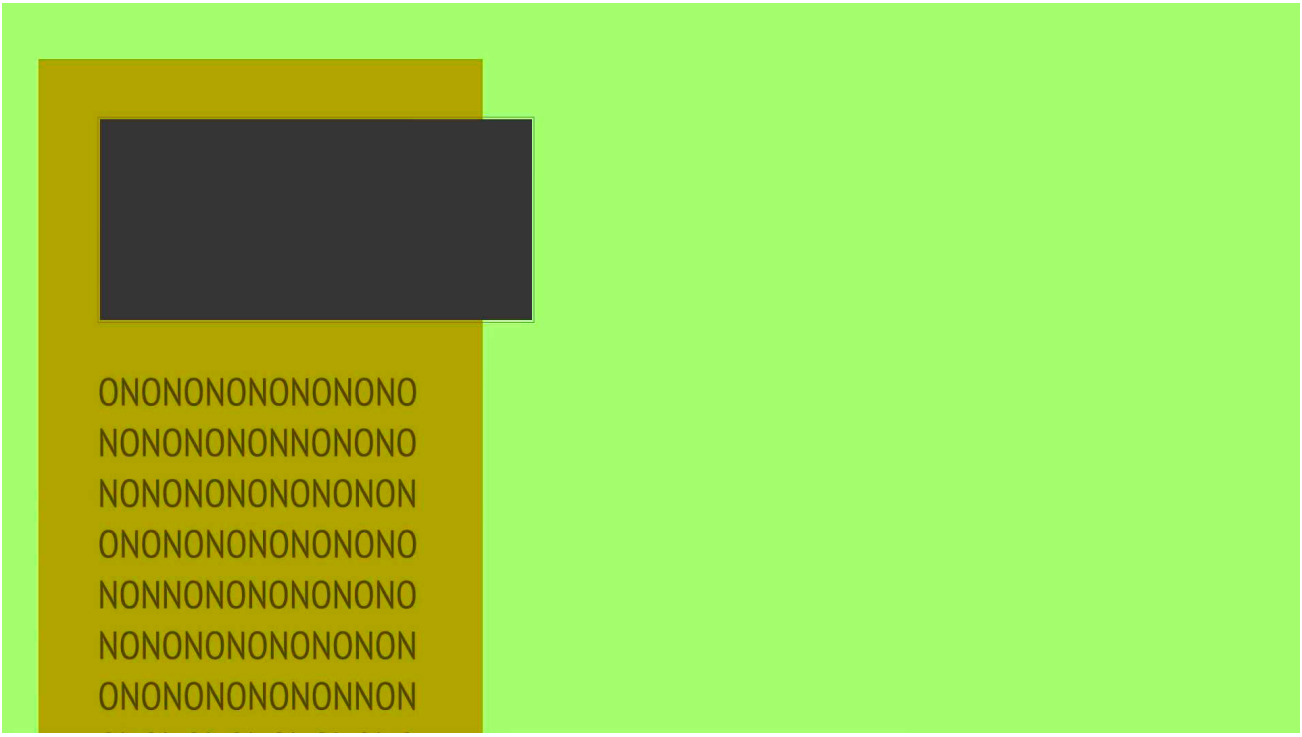
Mais adiante veremos detalhadamente como lidar com imagens, mas por agora vamos trabalhar com o cenário apresentado na Fig. 35:



Temos uma imagem que ocupa determinado espaço, no texto de um elemento, a qual é pouco menor que o texto, isto é, digamos que a imagem tenha 400 pixels e o elemento laranja tenha 600 pixels. Em [...] ela ocupa o seu espaço sem qualquer problema. Tudo está definido como porcentagem, o tamanho, a largura do elemento, o que significa que se o elemento for diminuído, os seus componentes vão se ajustando.

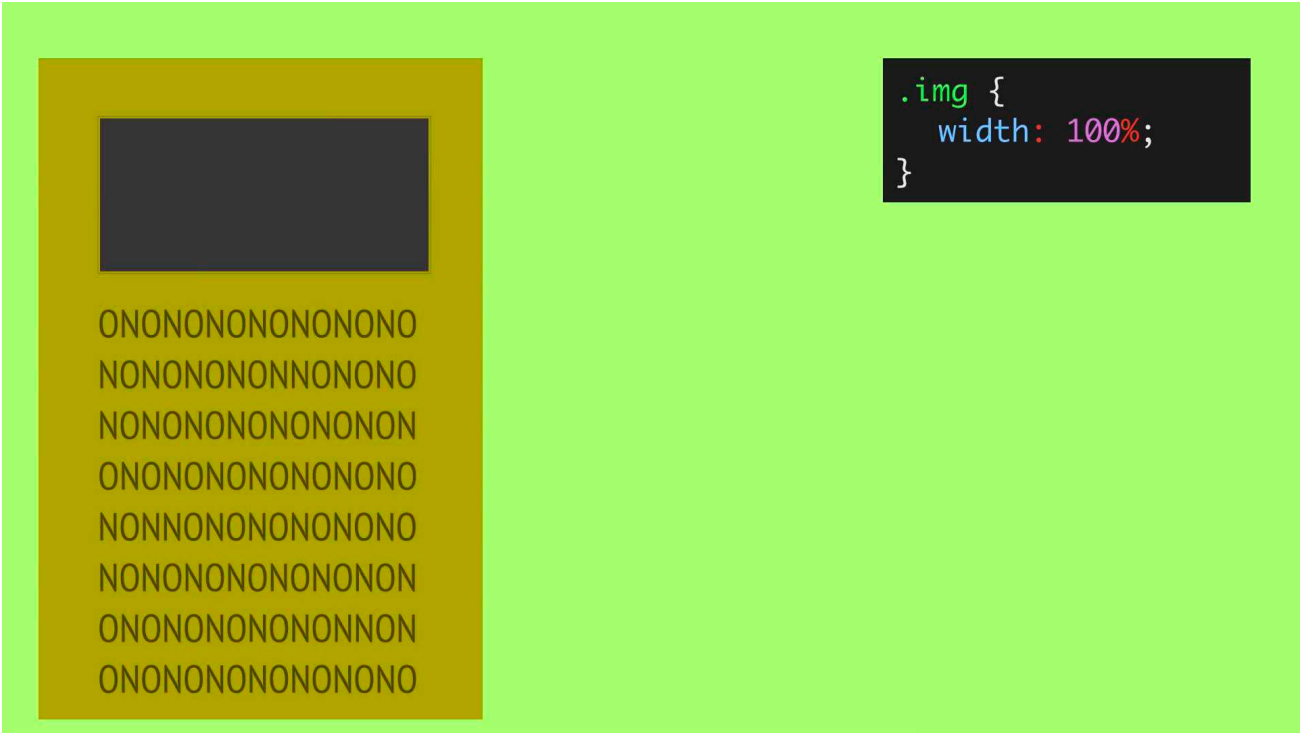


Porém, lembre-se de que a imagem é constituída de pixels fixos e se o elemento for reduzido um pouco mais, a imagem começará a vazar, `estourando` tamanho do elemento principal.

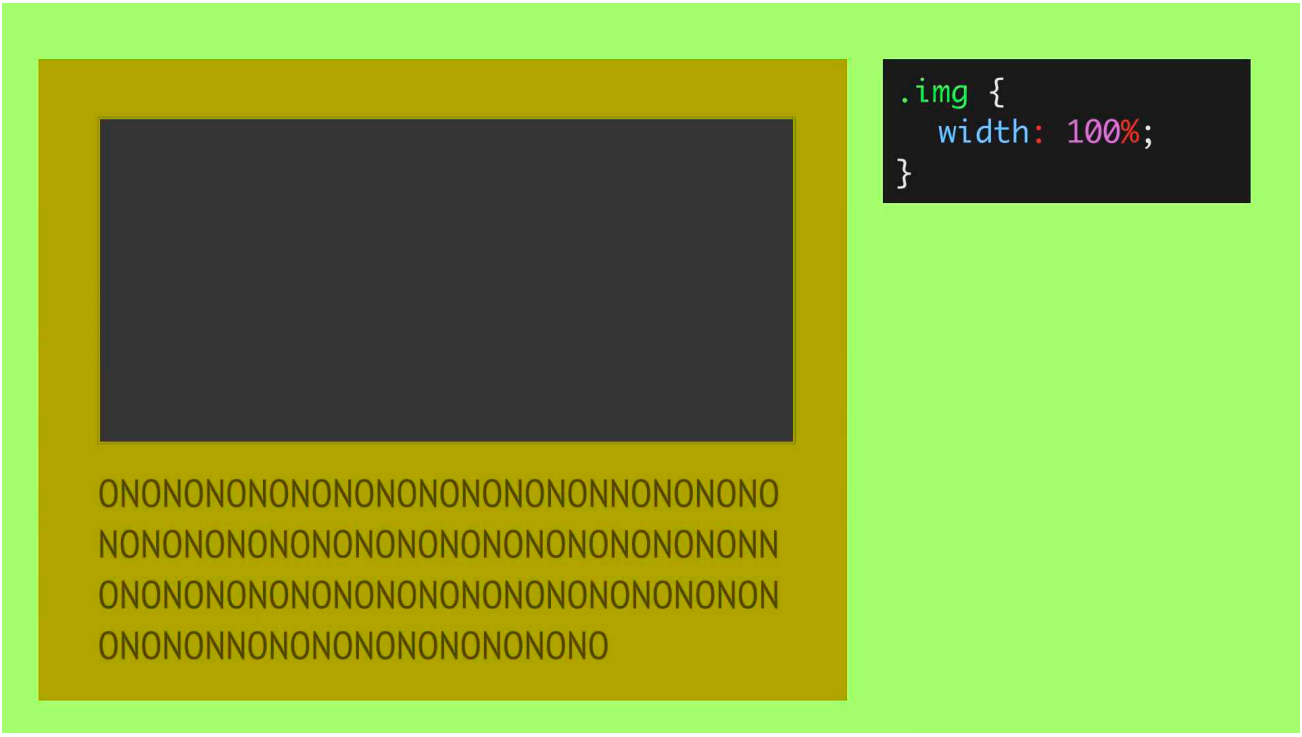


Isso ocorre porque a imagem é fixa e o elemento ficou menor que aquilo. Contudo, é fácil corrigir o problema, ou seja, o que se quer é que a imagem fique grudada dentro do elemento.

Usa-se a classe da imagem, com `width` de 100% (lembre-se que a porcentagem é sempre relativa ao elemento-pai) ao que se tem no conteúdo laranja. Na prática significa que a imagem vai ficar contida ali dentro, como se vê na Fig. 38:

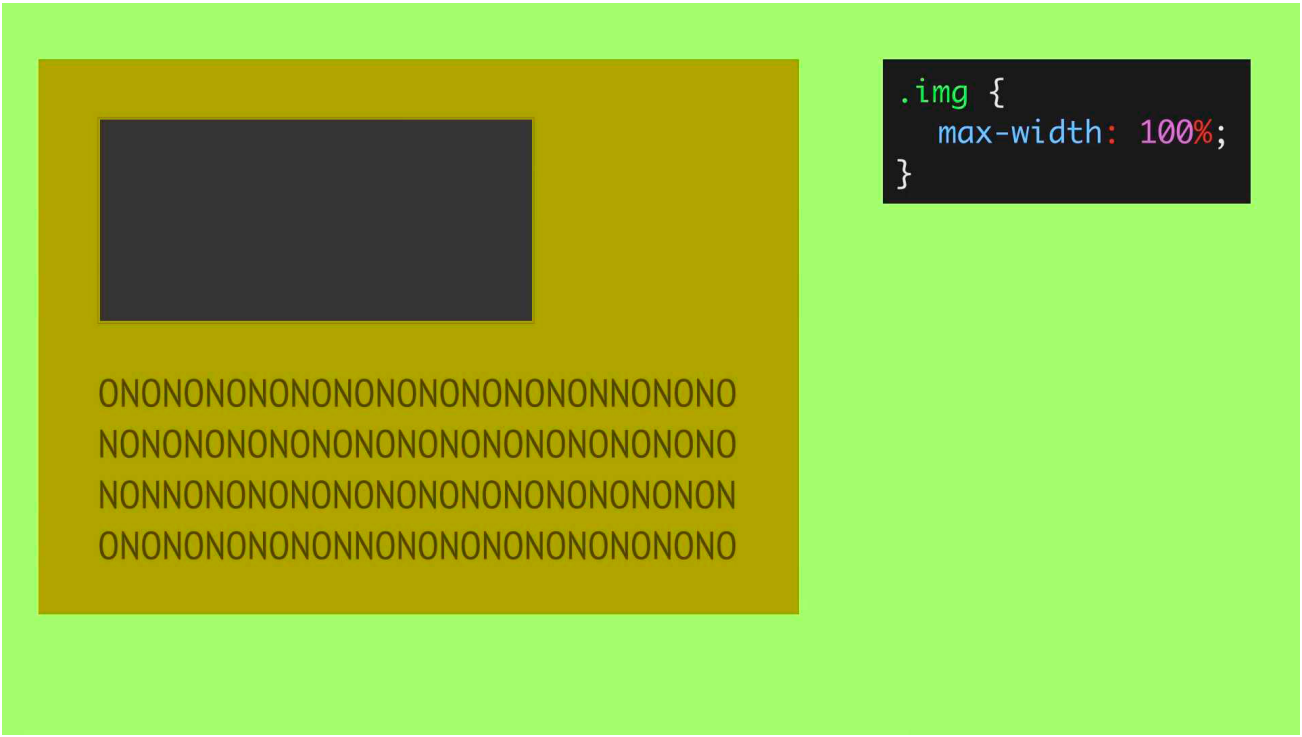


Há um detalhe importante: resolvemos o caso da imagem estourar o layout, porém no momento em que aumentar novamente, a imagem aumenta proporcionalmente, afinal a largura foi definida como 100% e assim sempre será.



Repare que a imagem vai ocupar 100%, não interessando se a tela aumentar até 1000 pixels, acompanhando sempre o tamanho total da tela. Porém, lembremos que a imagem inicial tinha 400 pixels de largura e se for colocada em uma tela de 1000 pixels, vai aumentar e esticar, se tornando `pixelizada` , aquele efeito de zoom em uma foto digital em que se enxerga todos os pixels. A qualidade não fica boa.

O que se pretende é que nesse layout maior a imagem fique do tamanho certo, não estoure o seu maior tamanho e, por outro lado, se ajuste a um layout pequeno, sem qualquer vazamento. Isto é obtido utilizando-se o `max-width` de 100%.



Assim a imagem não vai estourar o tamanho do contêiner, sem forçar a imagem a ficar sempre do tamanho do contêiner. Ela ficará em seu tamanho natural quando em situações grandes e do tamanho máximo para encaixar dentro do contêiner, quando precisar ser reduzida.

Essa regra da imagem com `max-width` 100% é clássica em sites responsivos, e na criação de sites deve ser um dos primeiros comandos a serem escritos, justamente para evitar o problema das imagens ficarem estourando.

Mais adiante veremos mais detalhes de imagens responsivas.

RESUMO

1) Sobre o layout fluído, trabalhamos com porcentagens e `em` , analisando o funcionamento de ambos os tipos de medida na prática e os cálculos, como o navegador chega naqueles valores;

2) Vimos também como faz diferença o uso de porcentagens e `em` nas fontes e nas medidas, como `width` , margem e tudo mais;

3) Foi mostrado como se trabalha com `inline-block` e também com `max` e `min-width` , para criar aquelas ideias de limitar o design responsivo, colocar uma coluna embaixo da outra, o que for necessário;

4) Por fim, demonstramos um pouco com a ideia de imagens fluídas trabalhando com `max-width` de 100%.

Agora vamos aplicar vários desses conceitos na prática em um exercício que temos no curso, continuando na próxima aula.