

# PRIMEIRO MÓDULO

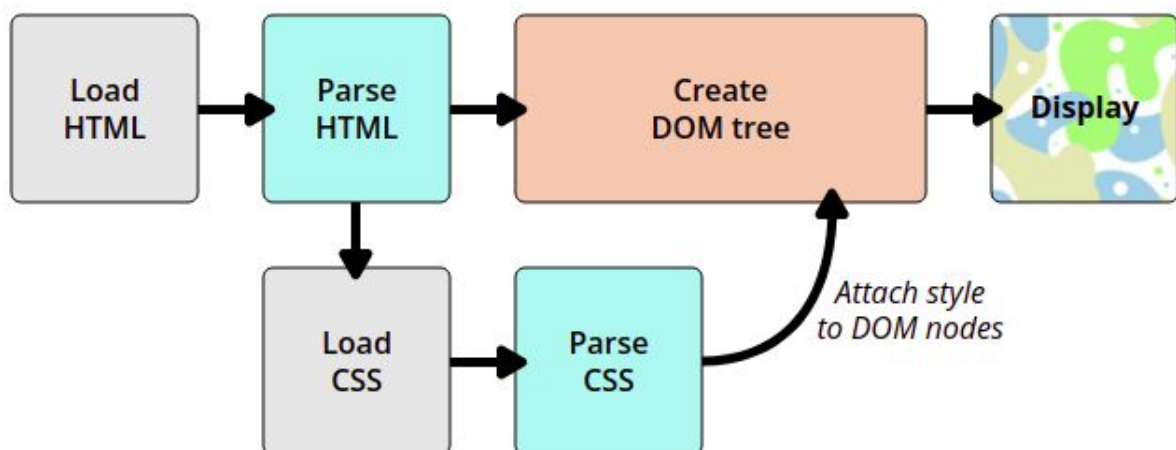
Visão geral sobre interação cliente e servidor

Visão geral sobre HTML e CSS

Hora do Código

Bom exemplo para estruturação do desafio

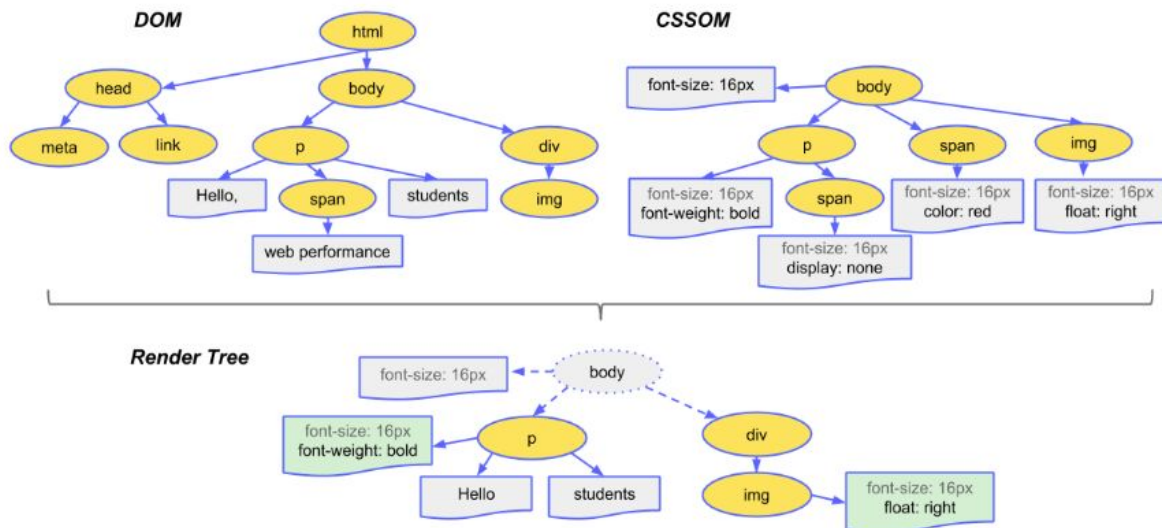
How CSS Works



Quando uma regra CSS é inválida, nada acontece. Aquele estilo simplesmente é ignorado  
Por isso, faz sentido aplicar fallbacks em alguns casos, para features novas que nem todos os navegadores suportam. Ex:

```
.box {  
  width: 500px;  
  width: calc(100% - 50px);  
}
```

## Construção, layout e gravação da árvore de renderização

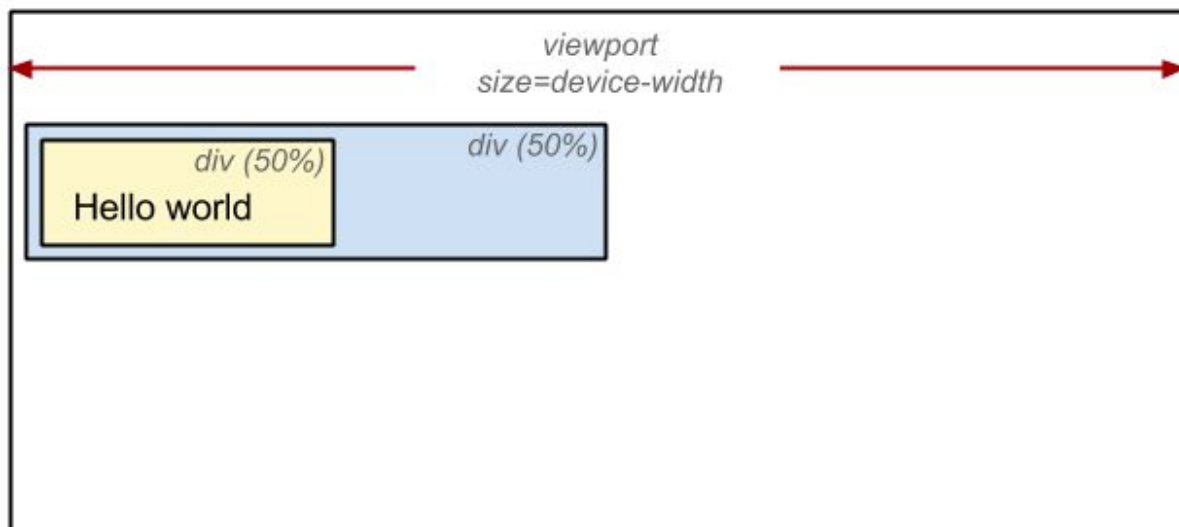


Para construir a árvore de renderização, em termos gerais, o navegador executa as seguintes atividades:

- A partir da raiz da árvore DOM, percorre cada nó visível.
  - Alguns nós não são visíveis (por exemplo, tags script, tags meta e assim por diante) e são omitidos, pois não são refletidos no resultado da renderização.
  - Alguns nós foram ocultados via CSS e também são omitidos da árvore de renderização, como por exemplo, o nó "span"---do exemplo acima---não está presente na árvore de renderização porque temos uma regra explícita que define a propriedade "display: none" nela.
- Para cada nó visível, encontre as regras do CSSOM correspondentes adequadas e aplique-as.
- Para cada nó visível, encontre as regras do CSSOM correspondentes adequadas e aplique-as.

Até agora, calculamos que nós devem ser visíveis e seus estilos processados. Mas ainda não calculamos a posição e o tamanho exatos na janela de visualização do dispositivo---essa é a fase do layout, também conhecida como **"reflow"**.

Para determinar o tamanho e a posição exatos de cada objeto, o navegador começa na raiz da árvore de renderização e passa por toda ela



O resultado do processo de layout é um "modelo de caixa" que captura a posição e o tamanho exatos de cada elemento dentro da janela de visualização. Todas as medições relativas são convertidas em pixels absolutos na tela.

Agora podemos finalmente passar essas informações para a última fase, que converte cada nó da árvore de renderização em pixels reais na tela. Essa etapa é frequentemente chamada de **"gravação" ou "rasterização"**.

Vamos recapitular as etapas do navegador:

1. Processar a marcação HTML e criar a árvore do DOM.
2. Processar a marcação CSS e criar a árvore do CSSOM.
3. Combinar o DOM e o CSSOM em uma árvore de renderização.
4. Executar o layout na árvore de renderização para calcular a geometria de cada nó.
5. Pintar os nós individuais na tela.

A otimização do caminho crítico de renderização é o processo de minimizar o total de tempo gasto nas etapas 1 a 5 da sequência acima. Isso permite renderizar conteúdo na tela o mais cedo possível, além de reduzir o tempo entre as atualizações da tela após a renderização inicial, ou seja, atingir uma taxa de atualização mais alta para conteúdo interativo.

## Browser Engine: Motores e Debugging

O servidor é basicamente dividido em algumas partes:

- Web Server (Servidor Web) que é o programa responsável por receber e responder as requisições HTTP vindas do cliente
- Web Application (Aplicação Web) que é o programa que tem a lógica de negócio da aplicação
- Database (Banco de Dados) que é onde as informações são armazenadas

Tags descritivas:

**main:** utilizamos apenas uma vez na página, ela deve englobar o conteúdo principal da página

**header:** engloba conteúdos de cabeçalho de algum texto

**footer:** também geralmente utilizamos uma vez na página, sendo a tag que engloba o conteúdo do final da página

**nav:** tag que engloba o principal menu de navegação da página

**video:** tag que engloba conteúdo de vídeo

**article:** tag que engloba um conteúdo auto contido

**section:** tag que engloba vários conteúdos que se relacionam

## QUIZZ

### QUESTÃO 1 DE 3

Quantas colunas o sistema de grid do Bootstrap tem?

☐ 14

☐ 12

☐ 10

☐ 6

☐ 8

ENVIAR RESPOSTA

R: 12

QUESTÃO 2 DE 3

Quais são componentes do Bulma?

- ☐ Breadcrumb, Message e Tabs
- ☐ Tabs, Panel e Popover
- ☐ Popover Pagination, Navbar
- ☐ Carousel Dropdown e Breadcrumb
- ☐ Tabs, Carousel e Nav

ENVIAR RESPOSTA

R: Breadcrumb, Message e Tabs

QUESTÃO 3 DE 3

No componente Card do Bootstrap, qual classe é usada para adicionar padding em uma seção?

- ☐ .card-body
- ☐ .card-content
- ☐ .card-box
- ☐ .card-padding
- ☐ .card-item

ENVIAR RESPOSTA

R: .card-body

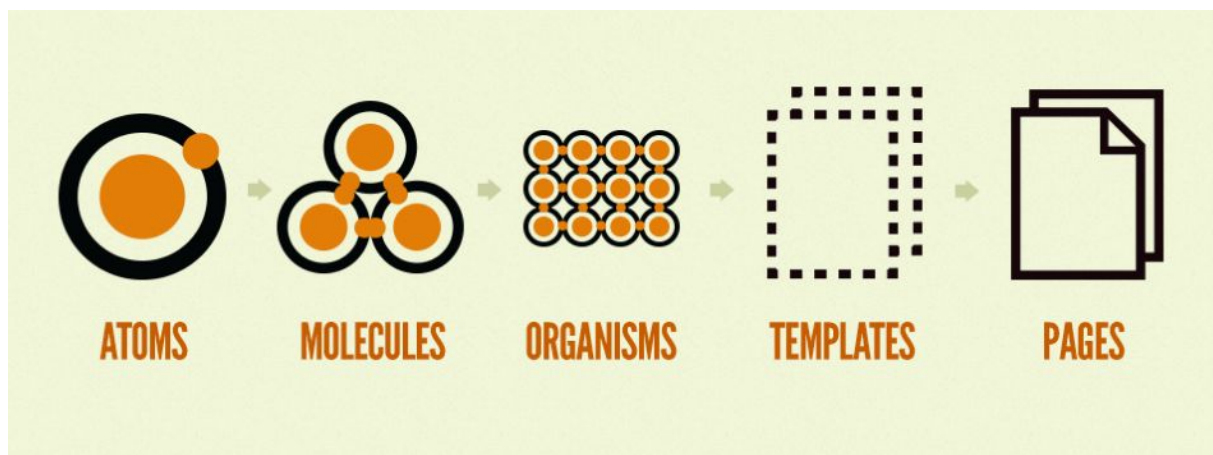
# SEGUNDO MÓDULO

BEM (Block, Element, Modifier)

sintaxe: **.block\_\_element--modifier**

exemplo: **.header\_\_logo--dark**, **.card\_\_item--active**

ATOMIC DESIGN



Átomos:

SEARCH THE SITE	LABEL
ENTER KEYWORD	INPUT
SEARCH	BUTTON

Molécula

SEARCH THE SITE
ENTER KEYWORD
SEARCH

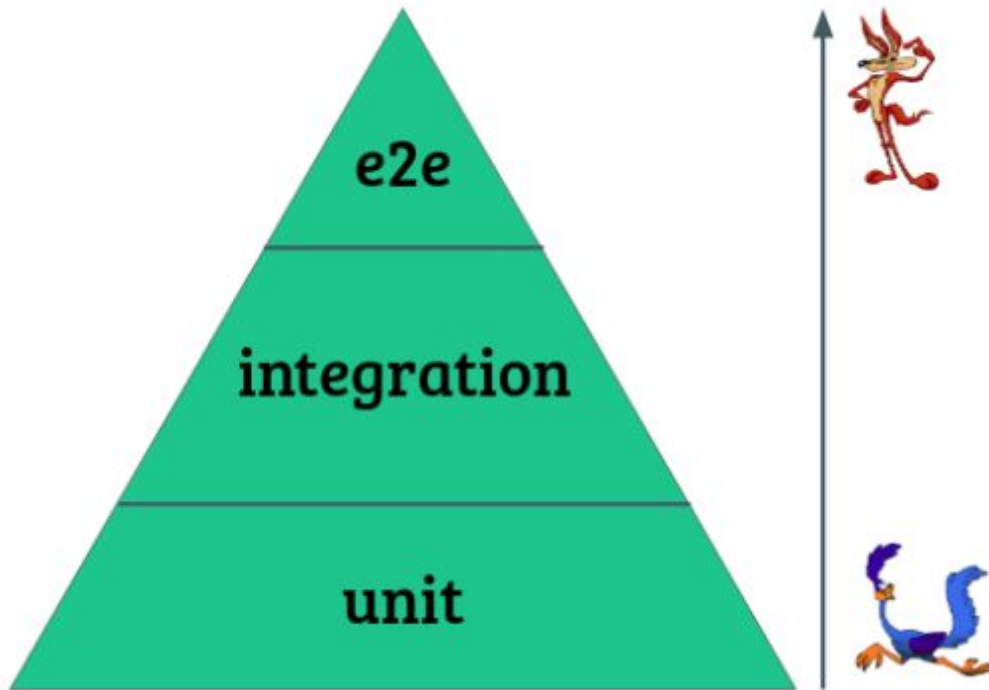
Organismo

	Home About Blog Contact	SEARCH THE SITE
		ENTER KEYWORD
		SEARCH

# QUARTO MÓDULO

Learning JS by doing it: <https://jskatas.org/>

A pirâmide de testes:



Funções puras -> sem side effects. Mesma entrada gerada mesma saída SEMPRE. Com isso, são mais fáceis de testar

Currying: A técnica de transformar uma função com múltiplos parâmetros em uma sequência de funções que aceitam apenas um parâmetro é chamada de Currying. Grande vantagem: transformar 0 código em pequenos pedaços mais expressivos e com maior reuso.

Exemplo:



```
1 var add = function(x, y) {  
2   return x + y;  
3 };  
4  
5 add(1, 2) // 3
```

noCurrying.js hosted with ♥ by GitHub

[view raw](#)

Nisso:

```
1 var add = function(x) {  
2   return function(y) {  
3     return x + y;  
4   };  
5 };  
6  
7 add(1)(2); // 3
```

currying.js hosted with ♥ by GitHub

[view raw](#)

Compose:

```
1 const compose = (f, g) => x => f(g(x));  
2  
3 const toUpperCase = x => x.toUpperCase();  
4 const exclaim = x => x + '!!';  
5  
6 const angry = compose(toUpperCase, exclaim);  
7  
8 angry('ahhh'); // AHHH!
```

composeES6.js hosted with ♥ by GitHub

[view raw](#)