Browser Engine: Motores e Debugging

Facilitar a compreensão das camadas de back-end e front-end.
Facilitar a compreensão das etapas de renderização do browser.
Facilitar a compreensão e a utilização do Dev Tools

Visão geral sobre a interação cliente e servidor

Apresentar as camadas que separam o front-end e o back-end

Hoje em dias estamos acostumados a abrir um navegador (Chrome, Firefox, Safari, Edge), pesquisar ou entrar como um link nele e entrar em um site para acessarmos algum conteúdo. Este fluxo é feito diariamente por bilhões de pessoas no mundo através de diferentes dispositivos: Smartphones, Notebooks, PC, Televisões e etc. Mas o que muita gente não sabe é como este fluxo funciona realmente.
O fluxo de acessar um website (Figura 1) ou aplicativo via um dispositivo é o que chamamos de relacionamento cliente e servidor. O cliente neste caso é o seu navegador que faz um requisição, no caso o website, para um servidor que possui este website.

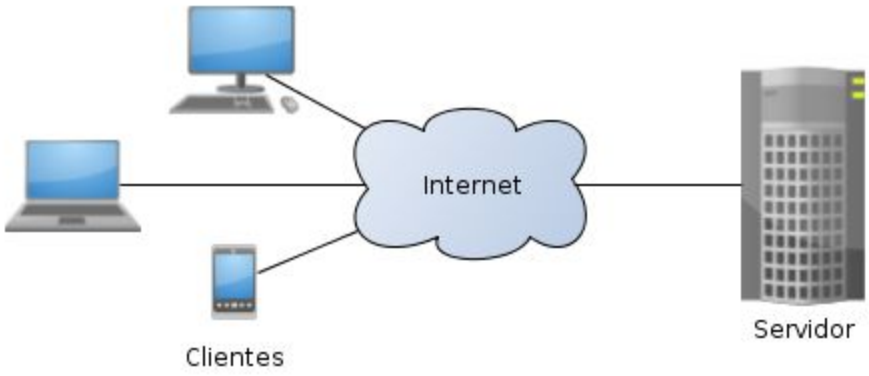

Figura 1 - Relacionamento cliente servidor

O cliente (número 6 na Figura 2), seu navegador, é o responsável por solicitar as informações ao servidor e renderizar este conteúdo para o usuário. Por isso é comum ouvir falar sobre motor de renderização do navegador, Blink para Google Chrome, Gecko para o Firefox, Webkit para Safari. Este motor realiza a transformação do código (HTML, CSS e JavaScript) enviado pelo servidor na página web que estamos acostumados a visualizar.
A comunicação entre cliente e servidor(número 1 na Figura 2) é feita utilizando o protocolo HTTP(Hypertext Transfer Protocol). Ela pode ser feita utilizando outros tipos de protocolos como o FTP(File Transfer Protocol), porém o HTTP é a forma mais comum. Este protocolo possui alguns métodos para identificar o tipo de requisição que o usuário realizou, um destes

métodos é o GET, ele é o método que identifica quando o usuário está pedindo algo ao servidor, como exemplo uma página web.
O servidor é basicamente dividido em algumas partes:

- O Web Server (Servidor Web) que é o programa responsável por receber e responder as requisições HTTP vindas do cliente
- A 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

A requisição chega ao Web Server (número 5 da Figura 2) e este encaminha ao recurso da Web Application responsável por gerá-la (número 2 da Figura 2).
A Web Application pode requisitar alguma informação do Database (número 3 da Figura 2), renderizar com esta informação um página HTML, ou um JSON, ou XML e devolver ao Web Server (número 7 da Figura 2) ou apenas retornar uma um conteúdo HTML (número 4 da Figura 2).

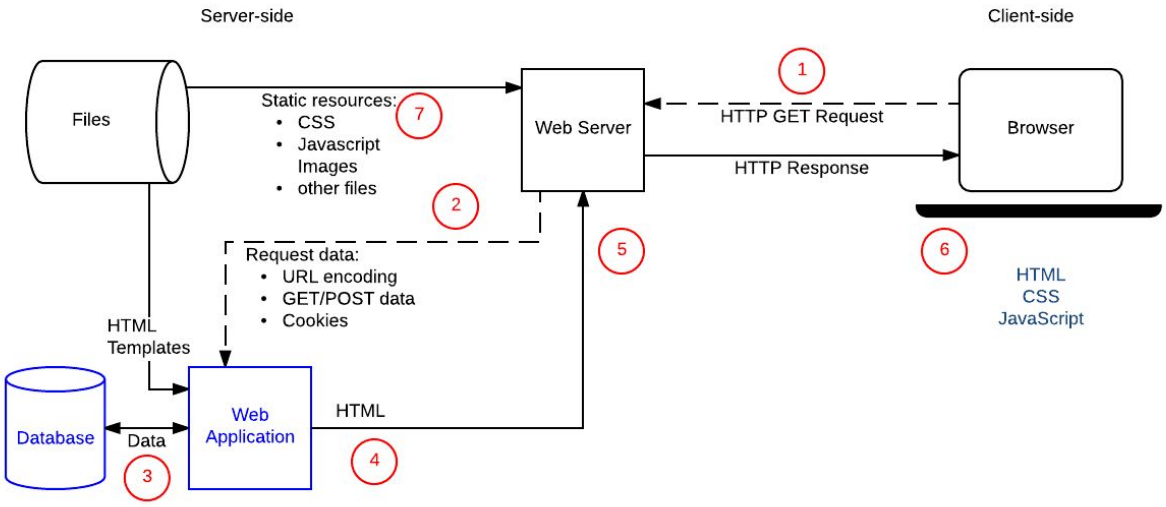

Figura 2 - Arquitetura cliente servidor

Visão geral sobre o HTML e CSS

Dar uma visão sobre HTML e CSS

HTML e CSS são a base para a construção da parte visual de uma página web. Podemos imaginar o HTML como sendo a base de uma construção, vigas, cimento, pilares e o CSS como sendo a parte decorativa e animada, pinturas, fontes de água, portas automáticas.

HTML

HTML é a sigla para HyperText Markup Language, é uma linguagem de marcação usada para descrever uma página web. Usa uma notação para dar informação sobre a página para o browser. Seus elementos possuem abertura e fechamento de tags que envolvem o conteúdo. Salvamos os arquivos com a extensão `.html` e ele irá possuir toda a estrutura da nossa página.

```
   <h1>Top level heading: Maybe a page title</h1>

   <p>A paragraph of text. Some information we would like to communicate to the
viewer. This can be as long or short as we would like.</p>

   <ol>
     <li>Number one on the list</li>
     <li>Number two</li>
     <li>A third item</li>
   </ol>
```

Estrutura básica da página

A estrutura básica de uma página HTML deve seguir a seguinte forma:

```
Doctype
html
 head
 body
html
```

Doctype: Não é uma tag HTML, é uma instrução da página para o navegador sobre qual versão do HTML será utilizada
html: Tag principal, ela engloba todo o conteúdo da página, o browser utiliza ela para identificar que aquele documento será uma página HTML
head: Tag de cabeçalho, todo o conteúdo dentro desta tag não será exibido no navegador, utilizamos ela para os importes dos arquivos CSS, Javascript ou a definição de metadados.
body: Tag de corpo, ela possui todo o conteúdo que será exibido pelo navegador

Cabeçalhos e Parágrafos

As tags de cabeçalhos (headings) são tags responsáveis por definir títulos, subtítulos dos textos da página. São divididas em 6, cada uma possui um tamanho diferente, sendo o maior o h1 e ela vai diminuindo até chegar ao h6

```
<h1>Título h1</h1>
<h2>Título h2</h2>
<h3>Título h3</h3>
<h4>Título h4</h4>
<h5>Título h5</h5>
<h6>Título h6</h6>
```

Para paragrafo temos a tag p

```
<p>Texto dentro de um paragrafo</p>
```

Tags descritivas

São tags que possuem um caráter de acessibilidade, ajudando a pessoas que utilizam leitores de tela a conseguirem encontrar facilmente o conteúdo da página e melhoram o SEO das páginas

```
<main>
<header>
<footer>
<nav>
<video>
<article>
<section>
```

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

CSS

CSS (Cascading Style Sheets) é uma linguagem de estilo, case sensitive, utilizada para aplicar estilo as tags HTML da página web. Os arquivos css salvamos com a extensão `.css` e chamamos estes arquivos de folhas de estilo.
Permite controlar:

- cores
- fontes
- posicionamento
- espaçamento
- tamanho
- decorações
- transições

Três formas de aplicar estilo:

- estilos inline, diretamente na tag HTML usando o stributo `style`
- Adicionando tags `style` no seu html e colocando as regras CSS dentro
- Escrever num arquivo css separado e importá-lo no arquivo HTML

Mudando a cor do texto

```
<h2 style="color: red;">CatPhotoApp</h2>
```

Seletores CSS

```
<style>
 h2 {
   color: red;
 }
</style>
```

Seleção por classes. Elas são estilo reutilizáveis que podem ser adicionados os elementos HTML

```
<style>
 .blue-text {
   color: blue;
 }
</style>
<h2 class="blue-text">CatPhotoApp</h2>
```

Fontes

Você pode alterar o tamanho da fonte com a propriedade `font-size`

```
<style>
 h1 {
   font-size: 30px;
 }
</style>
```

Você pode alterar a família da fonte com a propriedade `font-family`

```
<style>
 h2 {
   font-family: sans-serif;
 }
</style>
```

Tamanho

A propriedade `width` controla a largura de um elemento.

```
<style>
 .larger-image {
   width: 500px;
 }
</style>
```

Bordas

As bordas possuem propriedades como cor, largura e estilo que podem ser modificados

```
<style>
 .thin-red-border {
   border-color: red;
   border-width: 5px;
   border-style: solid;
 }
</style>
```

Você pode adicionar cantos com o `border-radius` e passando o valor em pixels

```
<style>
 .larger-image {
```

```
  width: 500px;
  border-radius: 10px
 }
</style>
```

Background color

O cor do background(plano de fundo) de um elemento pode ser alterada pela propriedade `background-color`

```
<style>
 .green-background {
  background-color: green;
 }
</style>
```

ID de um elemento

Além de classes cada elemento HTML também pode ter um atributo `id`
Diferente de classes ids devem ser únicos. São geralmente usados com JavaScript

```
<h2 id="cat-photo-app">
```

E utilizamos na nossa folha de estilo

```
#cat-photo-element {
 background-color: green;
}
```

Espacamento do elemento

Três propriedades controlam o espaço em volta de cada elemento HTML.

- padding
- margin
- border

```
<script>
 .red-box {
  background-color: crimson;
  color: #fff;
  padding: 20px;
  margin: 20px;
 }
```

```
</script>
```

A diferença básica das 3 propriedades é como elas afetam o componente:

- O padding expande o tamanho interno do componente
- O border é uma camada externa ao elemento logo após ao padding
- O margin é a camada mais externa, ele basicamente cria uma área invisível em volta do elemento onde nenhum outro elemento pode ocupar

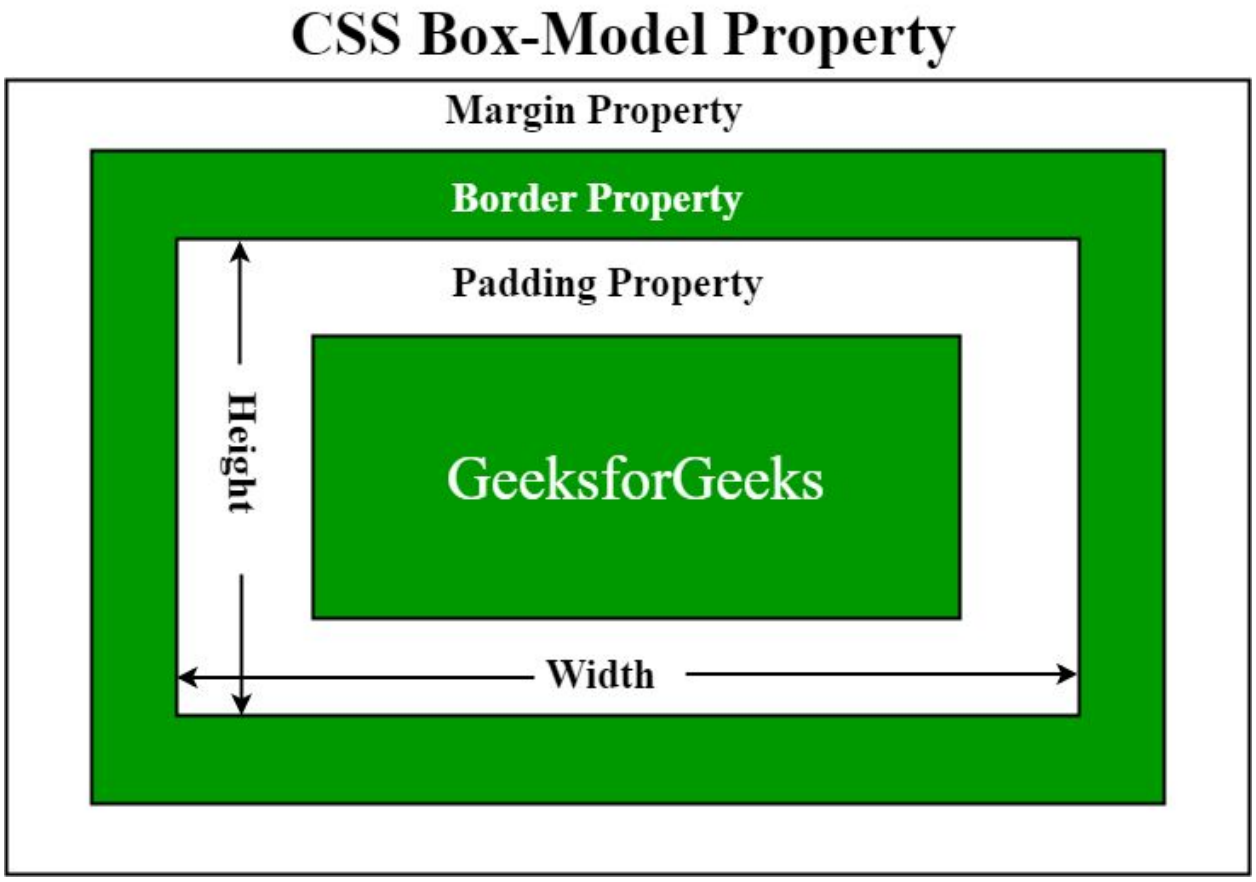

Figura 3 - CSS Box Model

FlexBox

O módulo Flexbox é uma nova recomendação da W3C que especifica como um layout é alinhado dentro da página.
Para utilizarmos devemos apenas definir `display: flex` dentro do nosso elemento. Esta propriedade define o elemento como um container flex e tornando seus filhos flex-itens
As propriedades mais importantes são:

- flex-direction: definida no container flex, ela define a direção que os items terão, podendo ser row(padrão) ou column

- justify-content: definida no container flex, ela define o alinhamento dos items horizontalmente quando o direction é row, ou verticalmente, quando o direction é column
- align-items: definida no container flex, ela faz o inverso do justify-content
- flex-wrap: ela define se os elementos devem quebrar ou não de linha quando chegam ao final da página.

Para outras propriedades aconselho uma lida no Guia completo FlexBox Origamid

Browser rendering

Apresentar as ferramentas de monitoramento de performance, profiling, e renderização do browser, para identificar gargalos e processos custosos durante a fase de renderização

Como vimos no tópico anterior o servidor pode devolver ao cliente um conteúdo HTML e este conteúdo quando chega ao navegador é transformado em uma página web para o usuário utilizar. Neste tópico vamos ver como o navegador funciona para poder transformar HTML e CSS em páginas web.

Parsing (construção do DOM)

O Document Object Model (Modelo de Objeto de Documentos) é a representação do código HTML da nossa página que o navegador gera na memória.

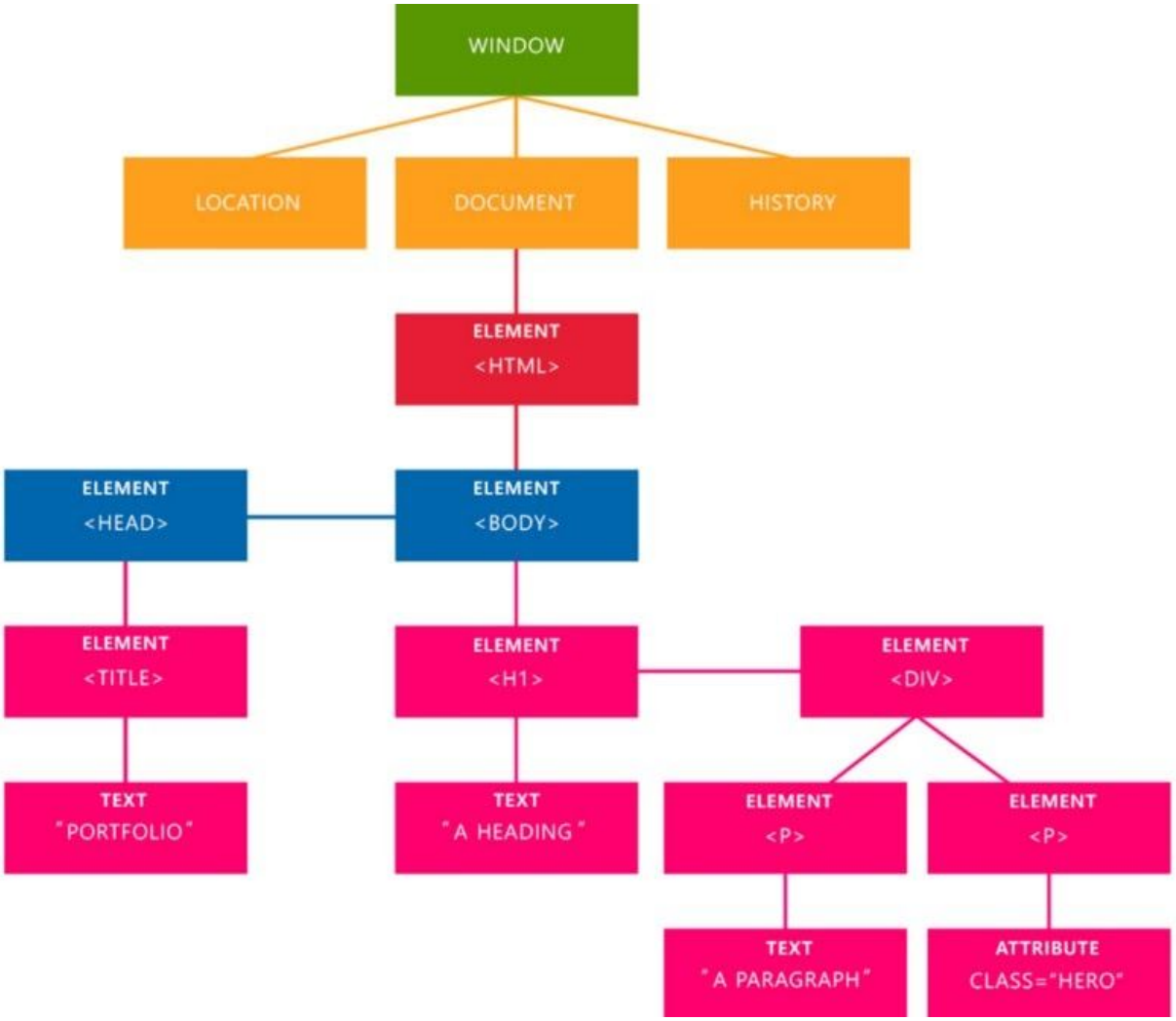

Figura 4 - Representação do DOM

Sua estrutura é a de uma árvore, onde o nó raiz é a janela do navegador e seu filhos são o document que é a representação da página HTML e seu filhos são todas as tags da página.
A construção da página no DOM parte da hierarquia das tags, por exemplo a tag <html>, como engloba todas as tags, é pai de todas as tag. Seus nós filhos são a tag <head> e <body> e assim o DOM vai sendo construído.

Processamento de estilo (CSS)

O CSSDOM é o Objeto que o navegador utiliza para representar os estilos css na memória. Ele é uma representação em árvore assim como o DOM

Composição

- O navegador carrega o HTML do servidor.

- Converte o HTML no DOM
- Baixa todos os recursos lincados na página (CSS, imagens, vídeos, JavaScript)
- Converte os estilos CSS no CSSDOM
- Constroi a Render Tree, que são os estilos css lincados aos nós do elementos do DOM
- Exibe a página na tela.

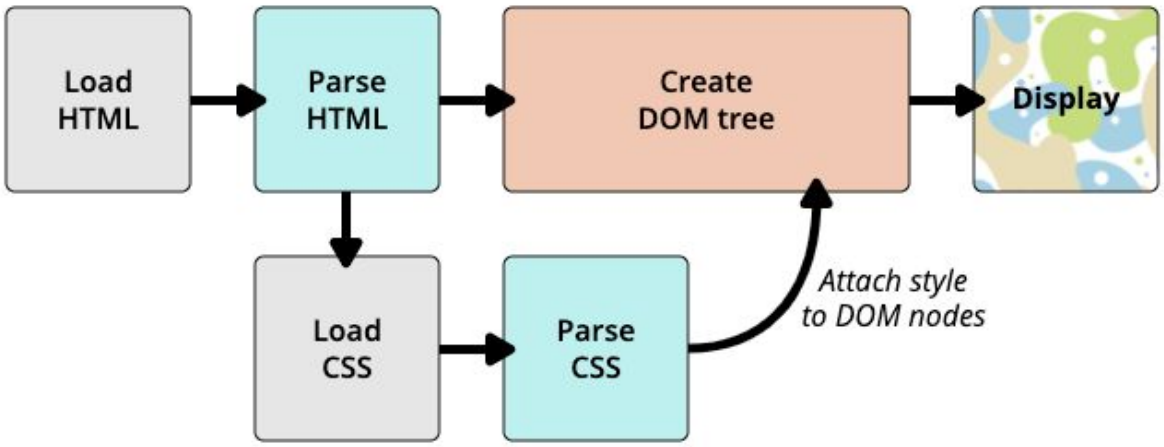

Figura 5 - Fluxo da criação da página

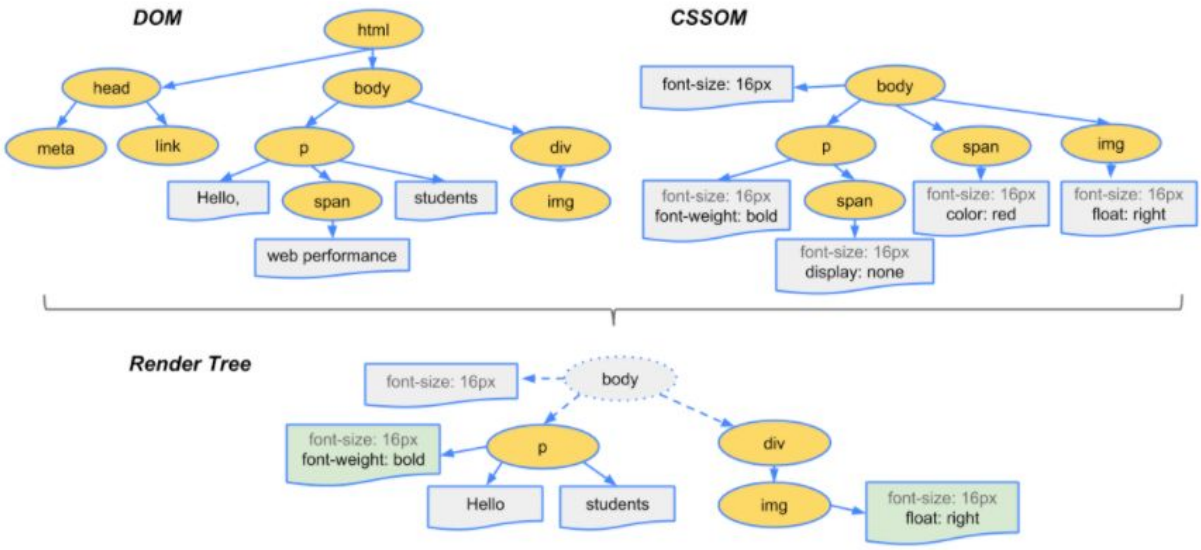

Figura 6 - Representação do DOM, CSSDOM e Render Tree

Desafio: https://github.com/thecodenation/challenges/tree/master/react-11