1 - Estruturas de Página

Para iniciar os conhecimentos em desenvolvimento para softwares na Web responsiva devemos ter um conhecimento sólido nas partes fundamentais de código que funcionam para o **cliente** e esta são: HTML, CSS e JavaScript. Durante nossas carreiras como desenvolvedores iremos esbarrar constantemente nessas linguagens e elas são de extrema importância para criarmos páginas, software e aplicações cada vez mais robustas.

### Introdução ao HTML:

O HTML está intimamente ligado ao desenvolvimento Web, pois surgiu paralelamente com o surgimento da Internet (em seu formato como conhecemos hoje) e houve a necessidade de transmitir as informações de uma maneira maximizada, com prioridade em ajudar no compartilhamento dessas informações. O **HTML** é uma sigla para **Hype-Text Markup Language** e, ao contrário do que muitos pensam, **o HTML não é uma linguagem de programação, mas sim de marcação!** O html funciona como o esqueleto de uma página Web. Hoje estamos na versão **HTML 5**, ‘nascida’ em 2014, e que possibilitou o modelo mais atual de veículo de informação como conhecemos hoje, aprimorando o uso de vídeos e áudios em páginas sem que elas tenham que estar ligadas a algoritmos próprios.

O **HTML**, funciona como estrutura primária de informações e possui diversas **tags** representativas para marcar as informações que queremos transmitir. Essas tags são utilizadas como **critério avaliativo** de **Search Engines (SE)** para categorização nas áreas de busca. Decidimos pelo uso do HTML como essa linguagem de marcação (ao inveś de um texto puro) pois, com o uso da internet, seria interessante veiculamos diversos formatos de arquivos pela rede, sem que elas fossem direcionadas com o mesmo ‘peso’ (leia peso como o tamanho de armazenamento em MB) que eles possuem em nossos computadores.

A primeira página da história foi a <https://www.info.cern.ch/hypertext/WWW/TheProject.html> e nela, o criador do HTML explica um pouco sobre o que foi o projeto, o porque da criação da lignuagem de marcação e como ela funciona e se transforma de ‘código’ para a renderização em nossos navegadores clientes!

### Primeiras tags

A estrutura básica das páginas em HTML se dá pela estruturação e marcação de frases, parágrafos, listas, títulos e cabeçalhos, imagens e assim por diante. Existe uma estrutura de tags padrão para o HTML: elas podem ou não conter o fechamento (**</>**), podem possuir atributos após o seu nome, podem conter mais de um atributo por vez, devem possuir um conteúdo e as tag que precisam ser fechadas, **devem ser fechadas!**

Dentro do **VS Code** temos que ter a estrutura live server como plugin, para visualizarmos em que estamos trabalhando de forma síncrona em nosso navegador.

Primeiramente, iniciamos a estrutura do **HTML** com a **tag** <**!DOCTYPE html>** e é a informação que passa ao navegador que o tipo de documento que estamos trabalhando (no caso, é = html).

Depois, utilizamos a tag **<html>** e ela indica que tudo que contém nessa página está marcado pela linguagem html e suas atribuições dadas a ele (como língua nativa, no caso do “pt-br” por exemplo) ao navegador que recebe esse pacote de informações.

O **<head> e o <body>** são as duas principais tags dentro de nosso escopo.

O **<head>** não possui informações visíveis em nossas páginas, mas contém informações cruciais de configuração da página, como o <**meta**> com atributo **charset** (que é o conjunto de caracteres utilizado na construção. O **UTF-8** é o conjunto alfabético que estamos familiarizados aqui no ocidente, por exemplo) e o <**title**> (que é o nome literal - aquele que aparece na aba do navegador - de nossas páginas). É também dentro do <head> onde colocamos algum caminho para a estilização com algum **CSS** ou outros diversos tipos de **scripts**. É nos <head> que ficam as **configurações** ou **dados sobre os dados** de nossas páginas!

Já o **<body>** é o corpo da página e é lá que fazemos as marcações nela. Como títulos **(<h1> à <h6>)**, parágrafos (**<p>**), imagens (**<img src>)**, listas ordenadas (**<ol>**), listas desordenadas (**<ul>**) e seus itens de lista (**<li>**) dentre várias outras! Para evidenciarmos a importância da página dentro da web, existe uma ordenação dada pelo Search Engine dentro de cada tag (dentro da <h1> é onde o Google analisa o assunto principal do site, por exemplo)!

Para editar nossos textos, existem as tags de quebra de linha (**<br>** que, por ser **sem conteúdo, não exige um fechamento**), de **bold** (**<strong>**), de *italic* (**<em>**) e outras diversas marcações para os textos de nossas páginas!

Note sempre que **tags** que não precisam de fechamento são as que não exigem conteúdo entre as **<>**, como o **<br>** e **<img>**.

### Links externos e internos

O **HTML** nos permite ter links externos e internos. Os links **internos** são os que fazem referência à sua própria página, enquanto os **links externos** referenciam outras páginas. Essa característica é muito importante pois dá a habilidade de referenciarmos outros documentos ou pontos de um mesmo documento, deixando o nível de informação dentro de uma página web mais rica e diversa!

As tags para referenciarmos links diversos é a **<a>** que tem abertura e fechamento, deve possuir conteúdo e pode acomodar diferentes atributos, sendo os mais comuns o **href** (que é o direcionamento de homepage, onde colocamos a URL desejada) e o **target** (que é o direcionamento alvo de onde abriremos esse link. O **“\_blank”** abre o link em outra aba, por exemplo).

Para obter o comportamento de **"âncora" em links internos** com a tag <a>, precisamos criar um atributo em forma de **id=””** dentro da tag aonde queremos direcionar a tag <a> para e, dentro da nossa tag de links internos, colocar o **id** desejado dentro do **href** em questão. Assim, ao clicarmos no link gerado, ele faz o direcionamento ao local do id que colocamos dentro do href do link!

2 - Primeiros Passos em CSS

### Introdução ao CSS:

O CSS é o **Cascade Styling Sheet** e a linguagem responsável por estilizar as **tags** e marcações de páginas comumente feitas em HTML. Em questões de programação, ela é feita em estilo cascada, sinalizando que **o computador lê a linguagem de baixo para cima,** e ela é programada seguindo as regras de marcação da página, referenciando os locais que queremos alterar o design.

Para que o computador leia o que está dentro do *“mundo CSS”* precisamos referi-lo dentro do HTML dentro das tags **<style></style>** ou em um arquivo separado com formato **.css** e referenciado através de **<link>** no *“mundo HTML”*.

Dentro da codificação, o **<style>** fica dentro da parte **<head>** do HTML, para alterar as partes que estão entre as chaves do **<body>.** Para estilizar (por exemplo) um **<h2>** é necessário referenciar essa tag seguida de colchetes, como **h2 { $CSS }** sendo o $css uma referência para as regras que ficam entre estes colchetes.

Para alterar as cores dos elementos das páginas podemos trabalhar com alguns parâmetros. “De fábrica" o CSS já possui algumas cores catalogadas e que podemos acessá-las simplesmente escrevendo **green, white, gray** (por exemplo). Podemos também utilizar a escala **rgb ($niveldevermelho, $niveldeverde, $niveldeazul);** que devem ser referenciadas explicitamente como escrito acima, ou no padrão **hexadecimal**, que é um padrão de cores muito utilizado no mundo web.

E se quisermos alterar somente um conteúdo específico, sem

Alterar seus semelhantes marcados pelo texto? Geralmente utilizamos o atributo **id=””** dentro da tag designada. O **id** tem uma atribuição única e que referencia somente aquele elemento. Para alterar o **“id”** dentro do CSS, utilizamos a sintaxe **#$nomedoid { $CSS }** e assim conseguimos manipular somente essa tag designada.

O conceito de **class** funciona como um atributo bem semelhante ao **id,** porém,podemos atribuí-la a diferentes elementos do HTML, manipulando diversas tags que desejamos referenciar, de uma só vez. Para manipular o elemento **class** no CSS, utilizamos a sintaxe **.$nomedaclass { $CSS }**.

### Propriedade de texto no CSS:

Podemos alterar diversas modalidades dentro de nossos textos, como fontes, tamanhos de letras, espaçamento entre elas e muito mais! Pelo **font-family** conseguimos alterar a fonte das tags que estamos especificando, mas **cuidado**, pois a família da fonte deverá existir dentro da fonte do cliente ou deve ir com o pacote de informações do site (**o que deixa ele bem mais pesado**). Para enviar essas fontes dentro do pacote que enviamos do servidor para o cliente, devemos especificar o <**link>** designado da fonte no **<head>** da fonte, especificando sua finalidade dentro dos atributos da tag. Podemos também **utilizar “fontes reservas”** paracaso a fonte não exista ou não possa ser visualizada, o CSS passa automaticamente para a outra escolhida!

Conseguimos alterar o tamanho da fonte com a propriedade **font-size : ;** e a unidade mais comum de medidas para os elementos do CSS é o **pixel (px)**, mas existem diversas outras unidades de medida para elementos, como o **vh, em e várias outras!**

Podemos alterar também a **tag <body>**, atribuindo propriedades diversas para todo o corpo do site, estilizando-o por inteiro e mantendo um padrão desejado dentro dele.

Dentro das estilizações de fonte, podemos colocá-las em **negrito,** *itálico,* underlined e diversas outras através da propriedade **font-weight : ;** ou **font-style : ; ,** sendo que ambas absorvem mais de uma propriedade por vez.

Para mudar o espaçamento entre as letras, utilizamos a propriedade **line-height : ;** , que também absorve unidades de medida para estilizar as letras e linhas. Podemos alinhar o texto com o **text-align : ;** , no qual podemos escolher se ele fica centralizado, mais a esquerda, mais à direita ou mais outras diversas propriedades, no box de estilos do CSS. Com o **text-decoration : ;** podemos também alterar as propriedades de design das letras.

Existe um site muito bom para aprimorar os conhecimentos e visualizar as funções de propriedade do CSS, que se chama [www.csszengarden.com](http://www.csszengarden.com) e nele conseguimos alterar o mesmo site com diversos estilos de CSS diferentes!

### CSS inline e externo:

Podemos alterar o design de nossos sites pelo CSS - além de como vimos, com as propriedades de CSS dentro da tag **<style>** - de maneira **inline**, dentro das próprias tag do HTML, formando atributos para essas propriedades como se fossem as próprias propriedades do CSS. A modalidade **CSS inline** é, quase sempre considerada uma má prática dentro da codificação pois pode ser um empecilho muito grande para sites que possuem ramificações dentro da sua *homepage*.

Já o método de estilização **externa** do CSS é uma boa prática e pode ser feita apenas criando um novo arquivo com **extensão .css** (é uma prática comum para intercalar vários tipos de linguagem de programação). Após criar o arquivo, devemos utilizar o CSS por lá, referenciando as mesmas tags do nosso HTML principal e criar uma tag **<link>, dentro do <head> da página**. Devemos colocar os seguintes atributos para que tudo ocorra dentro do padrão:

**<link rel=”stylesheet” type=”text/css” href=”$NOMEDOARQUIVO”>**

Sendo que o $nomedoarquivo deve ser exatamente igual ao nome do arquivo que desejamos direcionar o .css, além de que eles devem estar dentro de um mesmo diretŕoio (ou pasta).

3 - Seletores e Posicionamento

Para fazer bons desenvolvimentos na Web, precisamos saber efetivamente como funcionam os elementos para sua marcação em HTML e seu design em CSS e suas regras de seleção. Saber criar regras específicas que selecionam os elementos de maneira apropriada e posicioná-los na página permite uma melhor organização e propagação da informação que queremos transmitir. Os seletores de pseudo classe permite que adicionemos lógica condicional à estilização.

**Dicas da aula ao vivo:**

Margin -> usados em tags irmãs (que estão no mesmo container)

Padding -> em tags separadas (que não estão no mesmo bloco)

Quando precisamos conversar apenas com uma tag dentro de uma regra (por exemplo, **id**), seguimos uma seguinte regra no CSS.

Exemplo:

**.html** **.css**

<h2>Cachorro</h2> #market h2 {

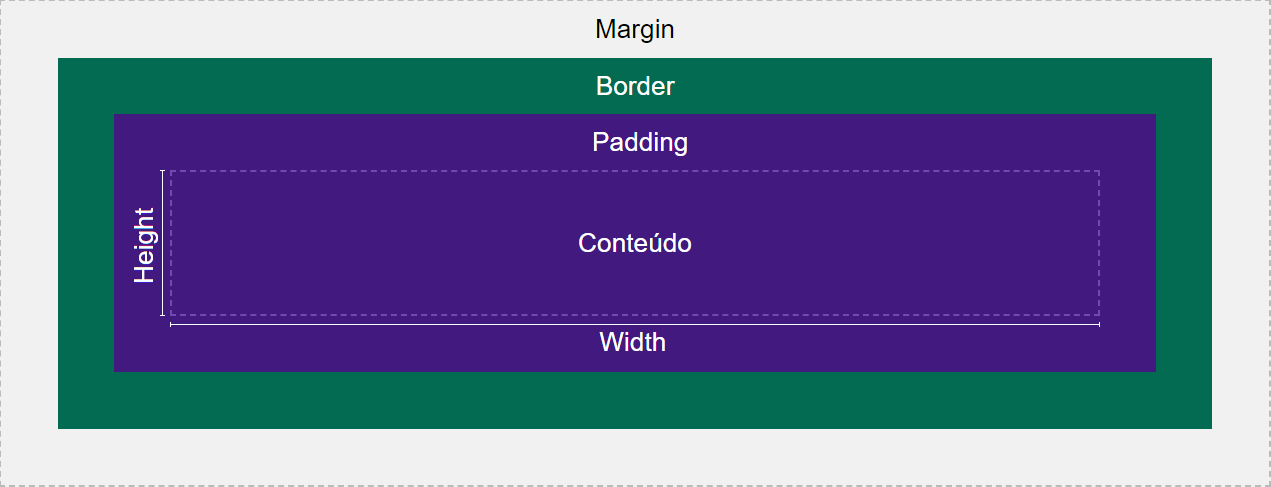
<div id=”market”> text-color: green;

<h2>Preto</h2> }

O CSS vai alterar apenas o h2 que está dentro da tag div, pois direciona a alteração apenas para para o texto que está dentro da tag com id=”market”.

### Box Model - Model de caixas no CSS

O CSS divide a página em um modelo de *“caixas”* , que é uma forma de renderização própria da linguagem e é assim que ela dispõe algumas propriedades de layout. Esse modelo funciona assim, para cada elemento tageado que criamos no HTML:



#### width e height:

O **width** é o atributo responsável pela **largura** e o **height** é o atributo responsável pela **altura** da área de conteúdo do box. Aqui desconsideramos o restante do *box*, visando somente o **content**.

#### padding:

Uma analogia interessante que podemos fazer com o **padding** é que ele pode ser comparado com o *"plástico bolha”* que fica ao redor do conteúdo e *“enche a caixa”* quanto mais for utilizado, pois quanto mais plástico bolha usamos em um conteúdo, maior precisa ser a caixa que ela está. Certo?

A tradução literal de padding é **enchimento** e controla a quantidade de espaço vazio entre o conteúdo bruto e a borda da caixa.

#### border:

É, literalmente, a borda da caixa! A propriedade **border** é uma *shorthand (ou atalho)* que serve para controlar um conjunto de outras propriedades que poderiam ser definidas individualmente, como o **border-width, border-style e border-color**.

O **border-width** é que define a largura da borda e seu *preset value* é definido como *medium.* O **border-style** é quem controla o estilo da linha da borda. Com seu *preset value* definido como *none*, é o que define que ela está transparente, podendo ser alterada com atributos para dar um ar de design para ela! Já o **border-color** é quem controla a cor da borda em si e como seu *preset value* é definido como *currentcolor*, fica “já de fábrica" com a definição do *background* da caixa.

#### margin:

A margem pode ser definida como uma espécie de campo de

força da caixa. A sua ação é definida além das dimensões visíveis da caixa e, além de **não ser afetada pela estilização do elemento**, ela serve para **manejar o afastamento e distanciamento entre caixas**. Tendo como principal utilidade o dimensionamento de elementos na página.

### Agrupamento de elementos:

Possuímos algumas propriedades de agrupamento , tanto no CSS quanto no HTML, e precisamos conhecê-los muito bem para manipularmos os dados conforme desejamos com o fim de criar belas peças para internet! A tag **<span>** vem do *‘mundo html’* e tem como tradução **período.** É uma tag para indicar um container genérico que é utilizado para agrupar elementos para fins de estilo (podemos gerar atributos nele, como class ou id). Já a tag do HTML **<div>** também possui *fechamento* e **divide** o texto em containers que podem ser espaçados (como um parágrafo inteiro, ou um conjunto de informações que queremos deixar em um lugar específico, em nossos sites) - o **<span> e <div>** possuem funções parecidas, mas não são iguais. O **<span> é um elemento inline** quando o **<div> é um elemento em bloco**. Ou seja, devemos usar o *div* para destacar parágrafos e o *span* para destacar somente uma frase dentro desse parágrafo, por exemplo.

Podemos alterar os elementos **width e height** dos nossos **divs**, criando um limite de largura e altura para a nossa divisória, no qual o browser respeita o dimensionamento, independente da tela que está sendo exibida. Já o **overflow** vem com o parâmetro ***visible*** já “de fabrica”, mas podemos alterá-la conforme quisermos! Atribuído o parâmetro ***hidden*** no elemento, o browser esconde tudo que estiver fora dos parâmetros de altura e largura. Já o ***auto***estabelece o acesso da informação que excede o box através de um scroll ao lado dele, conseguimos atribuir medidas horizontais e verticais para o **overflow,** através do **overflow-y**, que estabelece as medidas no eixo cartesiano *y* *(vertical)* e o **overflow-x** que estabelece o mesmo parâmetro para a horizontal *que é o eixo cartesiano x*. Quando o tamanho do box estiver excedendo o tamanho que mostramos ao div pelos elementos, no eixo *x*, também conseguimos visualizar as informações através de um scroll horizontal, se o atributo escolhido for o *auto.*

### Posicionamento de Elementos:

Podemos alterar a posição dos elementos contidos em nosso site através do CSS! O **position** com o atributo ***absolute*** deixa a posição dos elementos em definição absoluta *(dentro da página. Quando scrollamos pela página, os elementos ficam* ***absolutamente no lugar****)* posicionando o elemento em uma camada superior aos outros *(o que define este padrão é a ordem em que estão dispostos no CSS)*. Quando utilizamos o atributo **fixed**, o elemento fica *fixado em seu lugar*, ou seja, quando scrollamos a página, ele fica fixo onde o colocamos (o que é ideal para alguns tipos de cabeçalho de site!).

Utilizamos a propriedade **float** para fazermos as coisas flutuarem! É uma propriedade que permite posicionarmos um objeto para que ele se encaixe dentro do box que está definido.

A propriedade **top** define o seu dimensionamento conforme a

parte superior do site, enquanto a **left** é definida a partir da parte esquerda do site e assim, o righté definido a partir da parte direita do site e o **bottom** da parte de baixo.

O **z-index** é onde colocamos ordem nas camadas de imagem! Conseguimos posicionar essas camadas utilizando os números por ordem das imagens: **sim, é questão de numeração 1, 2, 3 … mesmo!**

### Agrupamento de Seletores e Pseudoclasses:

Podemos economizar diversas linhas de código e deixar tudo mais limpo e repleto de boas práticas com esses métodos! Imagine que você precisa fazer uma versão *dark-theme* do seu site e precisasse editar cada item da página, um por um? Com o agrupamento não precisamos fazer isso, pois colocamos todas as propriedades agrupadas, dentro de um mesmo elemento do CSS e modificamos todas ao mesmo tempo.

Para que isso seja possível, colocamos os elementos que desejamos agrupar em um mesmo parâmetro (como por exemplo: **p, div, header { $CSS }** - mas cuidado! Elementos que são separados por **vírgula** são tratados como agrupamento e cada uma tem sua propriedade respeitada, mas quando estão **sem vírgula** são tratadas como descendentes).

As pseudoclasses sempre vêm com o **nome do elemento, dois pontos (:) e o que vem após é o que nomeamos de pseudoclasse** - como no exemplo a seguir: **li:hover { $CSS}**. Alias, cada pseudoclasse também tem suas propriedades especificas. O **:hover** tem uma função que permite deixar as propriedades que listamos aparecem apenas quando passamos o *mouse* por cima delas. O **:active** só renderiza a propriedade do CSS quando clicamos com o *mouse*por cima deles, assim ativamos a função e deixamos o site dinâmico!

### Combinações de descendentes:

Também conseguimos combinar classes e subclasses que estão dentro do *‘mundo html’* diretamente no CSS, sem precisar voltar até o arquivo HTML que já trabalhamos para marcar a nossa página. Como já vimos, existe a possibilidade de utilizarmos **class e id** em nossos textos para ancorarmos essas tags. É uma boa maneira e faz parte das boas práticas, porém, podemos fazer essas alterações de maneira mais fácil!

Já vimos que, ao realizarmos a ancoragem pela tag **<class>** precisamos formatar o CSS assim **.$nomedaclasse,** mas é de extrema **importância** saber que todas as tags do HTML podem possuir mais de uma classe, simplesmente formatando o atributo da tag da seguinte maneira: **class=”$classe1 $classe2 $classe3 ”** apenas utilizando a barra de espaço.

Para limitar uma classe específica dentro de uma tag que você utilizou muito no HTML, simplesmente indicamos a **$tag.$classe** (por exemplo **p.perigo**) e assim, conseguimos editar somente um elemento desejado.

Ao trabalharmos com descendência, fazemos a analogia da hereditariedade, ou seja, são filhos herdando coisas dos pais! Por exemplo, se possuimos as tags:

**<ul>**

**<li></li>**

**<p></p>**

**</ul>**

Podemos listá-las no CSS como **ul li p{}** pois a linguagem compreende o código como “***p*** *que é filho de* ***li*** *que é filho de* ***ul****”.*

Para selecionar tags dentro do *‘mundo html’* sem termos que

colocar **class ou id** dentro das tags, utilizamos a pseudoclasse **:nth-to-child()** ou **:nth-to-type()** enele utilizamos as palavras **(odd) ou (2n+1)** para seletores ímpares dentro da gama search que estamos implementando às tags, **(even) ou (2n)** para as linhas pares. Para selecionar uma tag em específica, basta colocar o número de sua posição dentro como filho de uma tag. Para selecionarmos uma tag filha dentro de uma tag mãe - como por exemplo: **<div> seguido de <h3> -**, precisamos utilizar a pseudoclasse **:nth-to-child** dentro do CSS, da seguinte maneira **div:nth-to-child(n) h3{}.** Assim, o CSS consegue encontrar, através de hereditariedade de tags, o nosso **h3** dentro da nossa divisão de blocos **<div>!**

4 - HTML Semantico:

Conseguimos diferenciar os significados reais das tag usadas em HTML associando o elemento ao seu significado, porém não seria melhor se conseguíssemos estruturar uma marcação adequada que todos pudessem associar os conteúdos da página, somente olhando para as tags utilizadas? De certo, o google já pensou nisso e consegue buscar um site de maneira mais fácil (favoritando este site e o hankeando melhor, btw) através do HTML semântico.

Já conhecemos as tags **html, head, body e form**, mas sabemos realmente o que elas querem dizer?

* **html** representa o tipo do documento em questão, que nesse caso é ***HTML*** ;
* **head** representa os metadados do documento ***HTML*** , tais como o título do documento, links para arquivos ***CSS*** e ***JavaScript*** de que o documento precisa;
* **body** representa todo o conteúdo visual do documento ***HTML*** ;
* **form** representa um formulário para que o usuário consiga inserir dados.

Existem diversas formas de fazer a mesma coisa no mundo da programação e, de certo, esbarramos nisso durante nossas vidas profissionais. Porém existe um conjunto de boas práticas, para que nossas atividades ocorram de maneira mais efetiva e, utilizando as tag designadas para os locais designados, conseguimos ser mais pontuais em nossos códigos e mais efetivos durante nossa jornada.

A semântica, na língua portuguesa, falamos sobre o conteúdo da temática das frases (ou até de símbolos!). No HTML, o estudo da semântica se dá para obter uma tag com um conteúdo específico para a finalidade que precisamos transmitir assim, em contextos específicos elas possuem atributos específicos para a tarefa!

O **footer,** por exemplo, é a tag que especifica a criação do bloco “rodapé” dentro de um site. Obviamente, podemos utilizar um **<div id=”rodapé”>**, porém esse modo por confundir diversos usuários em uma escala global. Já imaginou as diversas maneiras que possuímos para escrever *rodapé* ao redor do mundo e quantas tags diferentes elas poderiam gerar?

O principal problema para as páginas escritas de maneiras não semântica é a acessibilidade do conteúdo. Um leitor de telas têm dificuldades para associar informações de *“código sujo”* e assim, impedem diversas pessoas que possuem alguma deficiência a acessar as informações de nossos sites.

O **HTML5** foi quem trouxe essa noção de semântica para dentro das marcações ao longo da web, deixando-os mais legíveis e assim podemos, além de ler códigos de outras pessoas de maneira mais simples, conseguimos fazer uma plataforma no qual todos têm acesso!

A tag **header** substitui a **<div>** que costumamos utilizar para montar o bloco do cabeçalho de um site,já a **nav** é utilizada para agrupar os links de navegação de um site, que geralmente é onde ancoramos seções internas do site em uma espécie de *menu*. Já a tag **main** é utilizada quando queremos unir diversos box de nosso site para construir uma informação principal, que antes era feita através de <div>.O **article** é desenvolvido para conteúdos auto-contidos e faz um efeito maior se o conteúdo for um pedaço de informação que faz sentido individualmente e se refere ao pŕoprio artigo que queremos transmitir. O **section** e o **aside** vem como criação de box para a complementação de conteúdo, sendo que na tag **section** colocamos complementos de texto que julgamos importantes para a suplementação de conteúdo e diferentes seções relacionadas a ele. Já na **aside** é onde colocamos o souvenir de nosso site, como bibliografias utilizadas para construção de conteúdo ou links externos.Como já foi dito, o **footer** é a analogia para rodapé da página.

**O HTML semântico** não diz respeito a estrutura de página e suas marcações em si. Ele vem como uma marcação com significado. É como falar com um vocabulário melhor e mais assertivo!

Não quer dizer que, com o HTML semântico vamos parar de utilizar a tag **<div>**, mas que vamos utilizá-la de maneira mais pontual e relevante para nossos códigos! O ideal é sempre que precisarmos utilizar uma tag <div>, façamos uma pesquisa dentro das tags semânticas, para saber se existe uma específica para o que queremos!

### Elementos de bloco e elementos inline

Existem os valores **block e inline**, utilizados no elemento **display** e esse valor influencia bastante na disposição da estrutura de uma página! O atributo **block** vem direto “*de fábrica"* quando um browser renderiza sua página (portanto é bem importante estruturamos essa parte do site!) e ele significa a disposição dos elementos em bloco e, por padrão, ocupa 100% da largura do elemento *pai* (sendo o pai o body, ou seja, ocupa todo o tamanho da página), ocupam a pŕopira linha, sendo posicionados abaixo do elemento anterior, têm a altura padrão definida pelo tamanho do conteúdo, e também é possível alterar suas propriedades de tamanho e largura e todas as suas propriedades de margem. Dentro do atributo **inline** não conseguimos alterar sua **heidth** (altura) e **width** (largura), mas na modalidade **block** conseguimos apenas alterar suas laterais.

Dentro do elemento **inline** as propriedades de texto copulam a largura do pŕopiro conteúdo, dois ou mais elementos inline em sequência são posicionados lado a lado, somente é possível alterar suas margens horizontais e se comporta como um texto redigido aqui no bloco de notas!

Já vimos a diferença entre as tags <span> e <div>, e que a sua principal diferença é que o **<span>** atual inline e para marcar apenas trechos de texto. Já o **<div>** atual como marcador em forma de container, armazenando blocos de textos e parágrafos inteiros.

Quando aplicamos a propriedade **float** em um elemento do tipo **inline,** ele automaticamente absorve a propriedade **block**, pois estará *flutuando* e conseguimos alterar sua posição dentro da página. Existe uma mistura de modalidade, que são do tipo **inline-block** e eles, por padrão, ocupam a largura do próprio conteúdo e também , dois ou mais elementos inline-block em sequência, são posicionados lado a lado. É possível alterar suas propriedades de tamanho e largura, diferente dos outros padrões **inline** e **block** separados. Também é possível alterar todas as propriedades de margem, respeitando as todas as dimensões de suas respectivas “*divs*” pré-definidas.