Desenvolvimento Web I

Curso Livre

Um pouco de nós

Desenvolvedor Front-end na MJV Technology & Innovation

Desenvolvedor Front-end na MJV Technology & Innovation

2º Ciclo / Análise e Desenvolvimento de Sistemas

ABRICK

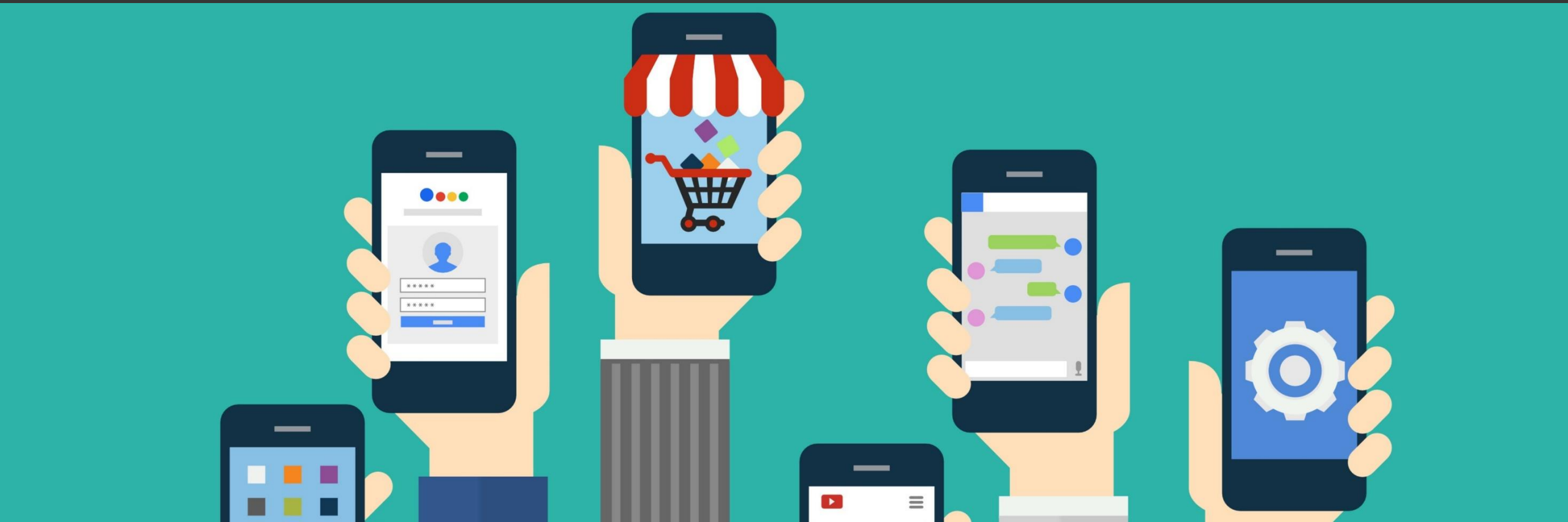
Sobre o curso livre

- Objetivo
- Conteúdo
- Certificado
- Presenças
- Avaliações

<DIA1>

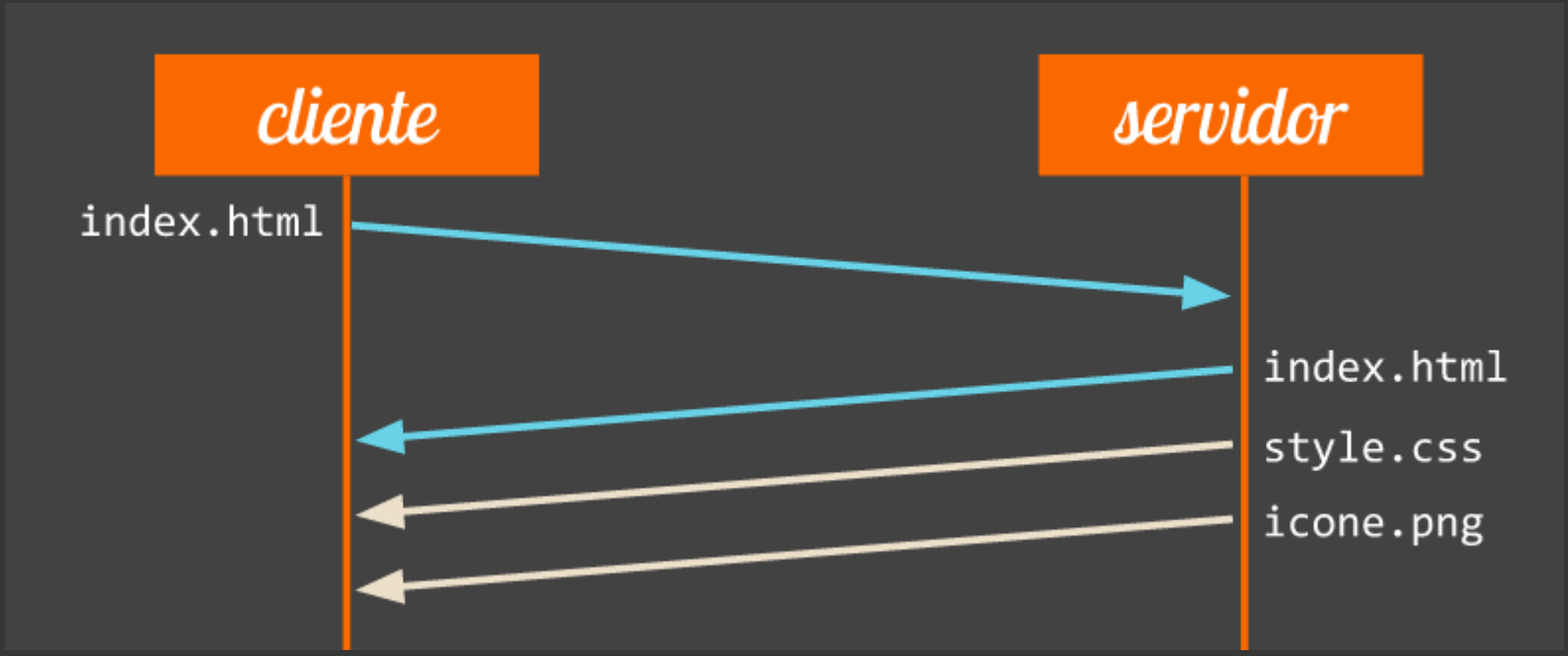
WEB

- Função da web e o consumo de informação
- Consumo de informação em diversos dispotivos



Arquitetura Cliente-Servidor

- Requisição x Resposta



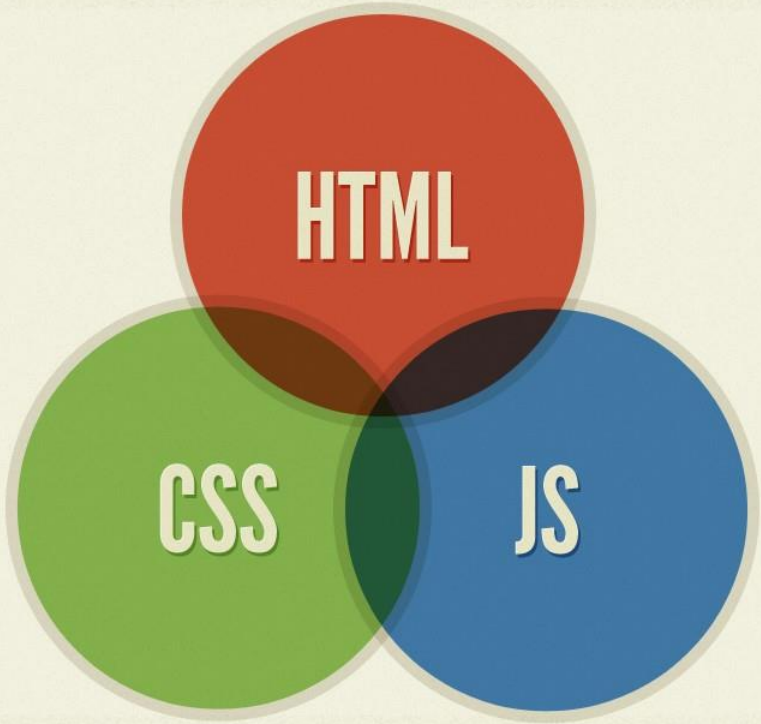
FRONT-END

BACK-END

Linguagens de programação

- Front-end
  - HTML
  - CSS
  - JavaScript
- Back-end
  - PHP
  - Python
  - Ruby
  - Java
  - NodeJS
  - ASP

Marcação



Estilização

Interação

O que preciso pra começar?

Somente um editor de texto




HTML

- Estrutura
  - Tags
  - Doctype
  - Head
  - Body
  - Metatags

HTML5

```
<!DOCTYPE html>
```

XHTML 1.0 Transitional

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.0 Transitional//EN" "http://www.w3.org/
TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

HTML

```
<!DOCTYPE html>
<html>
        <head>

        </head>
        <body>

        </body>
</html>
```

HTML

<TAG> </TAG>

<TAG/>

HTML

Algumas Tags

<meta charset="utf-8">
paragrafo
divisão
<form>formulário</form>
lista não ordenada
lista ordenada
item de lista
<img src="caminho-da-imagem">
Link
<h1>cabeçalho de nível 1</h1>

HTML

<TAG ATRIBUTOS DATA-ATRIBUTOS>

<TAG ATRIBUTOS DATA-ATRIBUTOS>

HTML

Alguns Atributos

```
<img src="caminho-da-imagem">
<a href="http://link">Link</a>
<div data-div="1">divisão</div>
<p class="paragraph">texto</p>
<form id="formulario"></form>
<div style="font-size:16px;" onclick="funcao()">divisão</div>
```

HTML

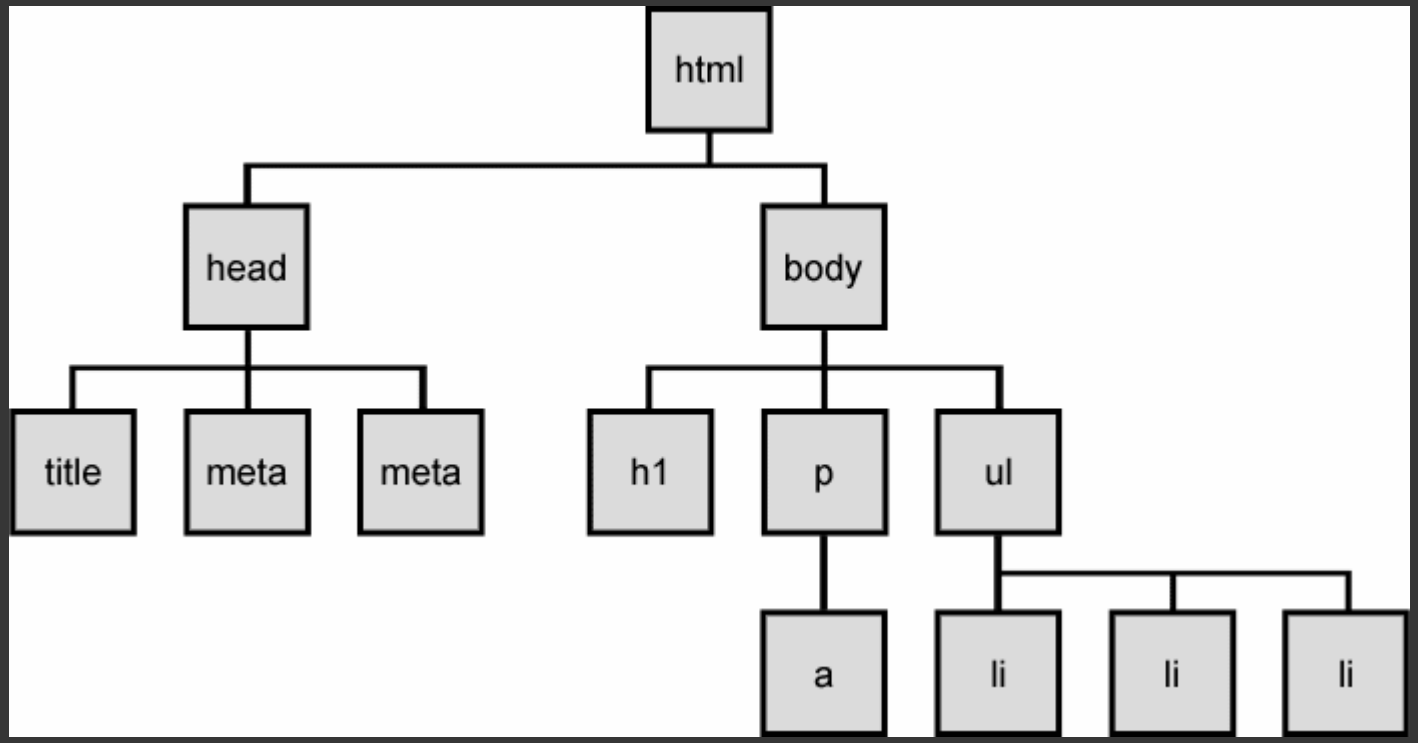
- Toda tag é um bloco
- Renderização
- Cross-browser
- Classes, id e data-*

Navegadores mais usados (e o internet explorer)

NAVEGADORES MODERNOS

HTML

- DOM (Document Object Model)



- Alterações no DOM afetam a performance

HTML

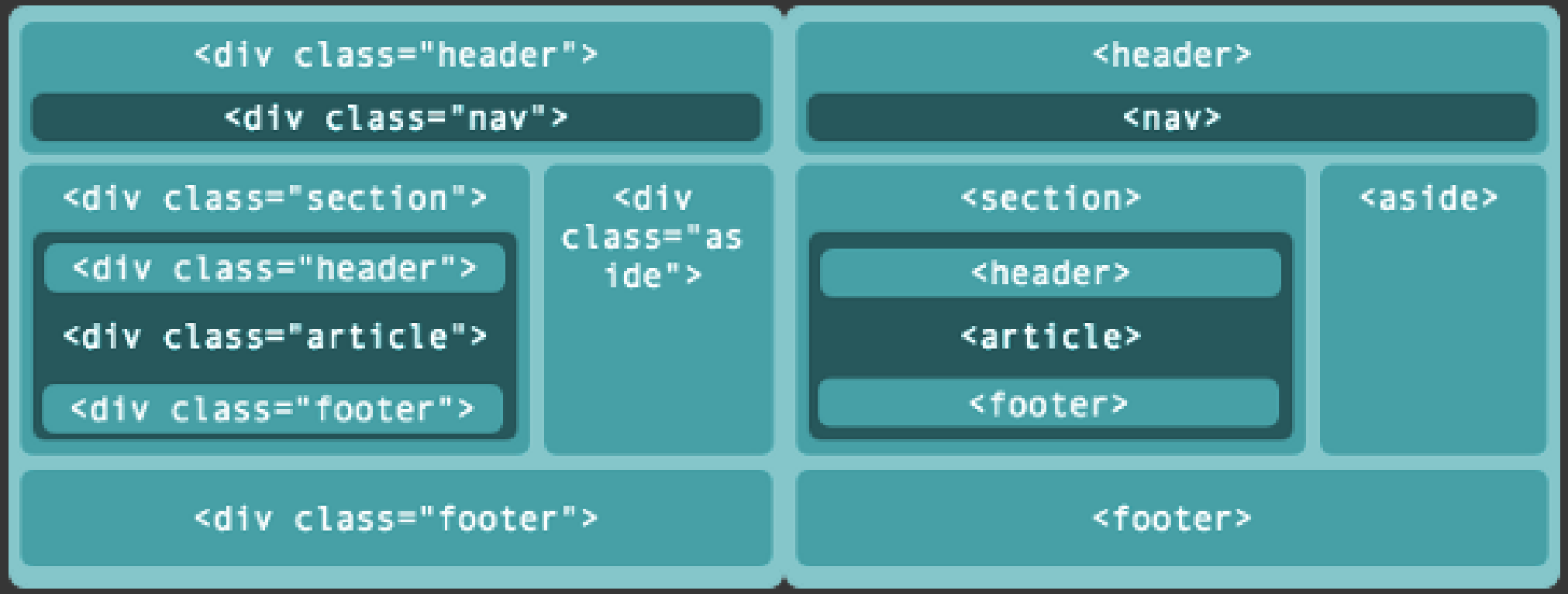
- Semântica
  - Web 1.0 (sintática) , 2.0 (social) e 3.0 (semântica)

```
<div>
        Parágrafo de texto
        institucional
</div>
```

```
<p>
        Parágrafo de texto
        institucional
</p>
```

HTML

- Semântica



HTML

- Semântica
  - SEO e posicionamento em motores de busca
  - Boas práticas
  - Acessibilidade

Inspecionar elementos

</DIA1>

<DIA2>

Proposta de código

Montar a estrutura HTML de 3 páginas:

- Cadastro de conteúdo
- Listagem de conteúdo
- Visualização de conteúdo

As páginas devem possuir a estrutura HTML do cabeçalho, do menu (com links para as demais páginas), do rodapé (com direitos autorais do desenvolvedor) e do conteúdo interno.

OBS.: NÃO UTILIZAR CSS E JS

</DIA2>

<DIA3>

CSS

- Folhas de Estilo em Cascata

CSS

- Estilização
- Efeitos
- Animações
- Micro interações

CSS

- Estrutura:

```
seletor {
        regra: valor ;
        regra: valor ;
        regra: valor ;
}
```

```
h1 {
        color: red ;
        background: white;
        padding: 15px ;
}
```

CSS

- Métodos de utilização:
  - Inline
  - Embutido
  - Externo

CSS

- Métodos de utilização:
  - Inline

```
<p style=“color:red;”>
        Meu texto
</p>
```

CSS

- Métodos de utilização:
  - Embutido

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

CSS

- Métodos de utilização:
  - Externo

No arquivo html:

```
<link rel="stylesheet" href="caminho-da-pasta/style.css">
```

No arquivo css:

```
p {
        color: red;
}
```

CSS

- Benefícios e malefícios de cada método de utilização:
  - Inline : reescrita de código, sem reutilização, restrição, poluição
  - Embutido : permite reutilização, mas ainda polui o principal html
  - Externo : permite reutilização , sem poluição

CSS

- Seletores:
  - elemento-pai elemento-filho
  - .elemento-com-classe
  - #elemento-com-id
  - [atributo-de-elemento=“valor-do-atributo”]
  - elemento.restricao-por-classe#restricao-por-id

CSS

- Regras reduzidas

```
.elemento {
    margin-top:5px;
    margin-right:10px;
    margin-bottom:5px;
    margin-left:10px;
}
```

```
.elemento {
    margin:5px 10px 5px 10px;
}
```

```
.elemento {
    margin:5px 10px;
}
```

CSS

- Regras reduzidas

```
.elemento {
    margin-top:5px;
    margin-right:10px;
    margin-bottom:8px;
    margin-left:10px;
}
```

```
.elemento {
    margin:5px 10px 8px;
}
```

CSS

- Regras reduzidas

```
.elemento {
    margin-top:10px;
    margin-right:10px;
    margin-bottom:10px;
    margin-left:10px;
}
```

```
.elemento {
    margin: 10px;
}
```

CSS

- Regras reduzidas : regras que aplicam seus resultados mediante a quantidade de valores passados:
  - 4 valores: aplica seu resultado seguindo a ordem TOPO, DIREITA, BAIXO, ESQUERDA
  - 3 valores: aplica seu resultado seguindo a ordem TOPO, DIREITA-ESQUERDA e BAIXO
  - 2 valores: aplica seu resultado seguindo a ordem TOPO-BAIXO e DIREITA-ESQUERDA
  - 1 valor: aplica seu resultado seguindo a ordem TOPO-BAIXO-DIREITA-ESQUERDA

CSS

- Regras prefixadas: prefixos utilizados para melhorar a usabilidade no quesito cross-browser, pelos diferentes motores de renderização

```
.elemento {
        border-radius:5px;
}
```

```
.elemento {
        border-radius:5px;
        -webkit-border-radius:5px;
        -moz-border-radius:5px;
        -o-border-radius:5px;
        -ms-border-radius:5px;
}
```

- -webkit- : para Chrome/Opera
- -ms- : IE
- -o- : Opera (versões antigas)
- -moz- : Mozilla Firefox

CSS

- Seletores “Especiais”:
  - Pseudo-elementos: elementos criados apenas no css
    - elemento:before , elemento:after
  - Pseudo-classes: classes ativadas mediante “ações” do usuário
    - elemento:hover , elemento:active

CSS

- Especificidade: regras CSS mais ESPECÍFICAS fazem com que suas regras sejam IGNORADAS ou PREVALECIDAS

```
.p {
        color: red;
}
.alunos-container .p {
        color: blue;
}
```

https://specificity.keegan.st/

CSS

- Transições: define um tempo para que alterações de css ocorram mediante ações do usuário ocorram. É definida pela regra de transition e utiliza um tempo e uma “curva” (cubic bezier) para execução da animação

```
.p {
        color: red;
        transition: .2s ease-in-out;
}
.p:hover {
        background: darkred;
        color: white;
}
```

CSS

- Animações: utilizam dos mesmos princípios das transições (tempo e curva de execução), porém, devem ter seus frames definidos

```
.p {
        animation: colorAnimation infinite 5s;
}
```

```
@keyframes colorAnimation {
    from { background: black; }
    to { background: red; }
}
```

```
@keyframes colorAnimation {
    0% { background: black; }
    50% { background: red; }
}
```

CSS

RESPONSIVIDADE

CSS

- Nova meta tag no html, para que os elementos se adaptem ao “viewport” , área visualizada no dispositivo acessado

```
<meta name="viewport" content="width=device-width" />
```

CSS

- Media queries: blocos de regra CSS que são utilizados para SOBRESCREVER regras previamente definidas, mediante determinada largura/altura/dados do viewport do usuário

```
.p{
        background: red;
}

@media (max-width: 800px) {
        .p {
                background:black;
        }
}
```

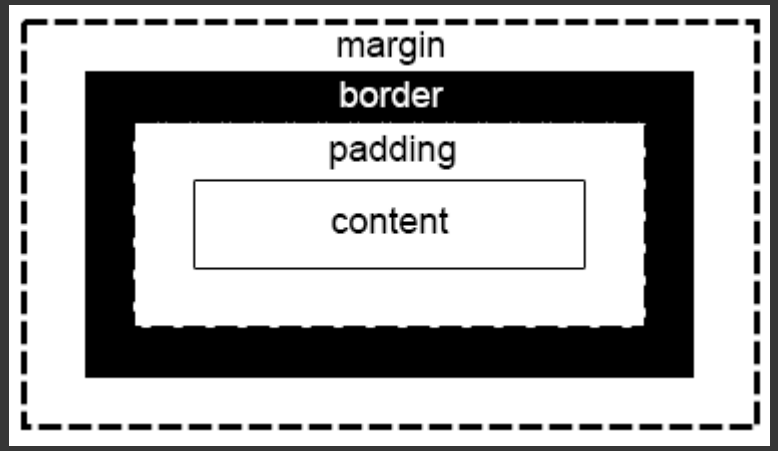
CSS

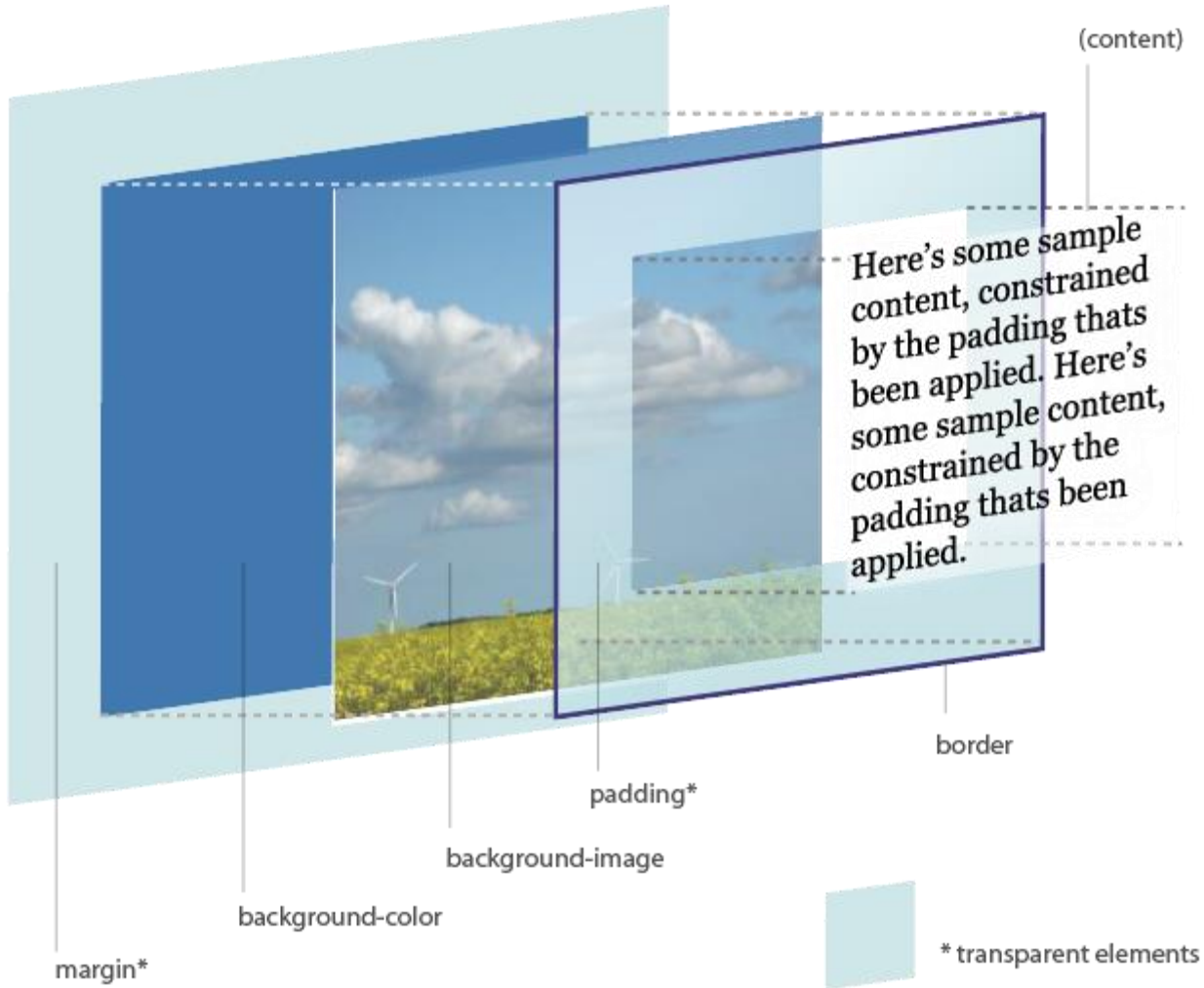
- RESET: regra inicial CSS utilizada para “resetar” , limpar , todos os elementos CSS , pois os elementos já possuem certos valores por padrão

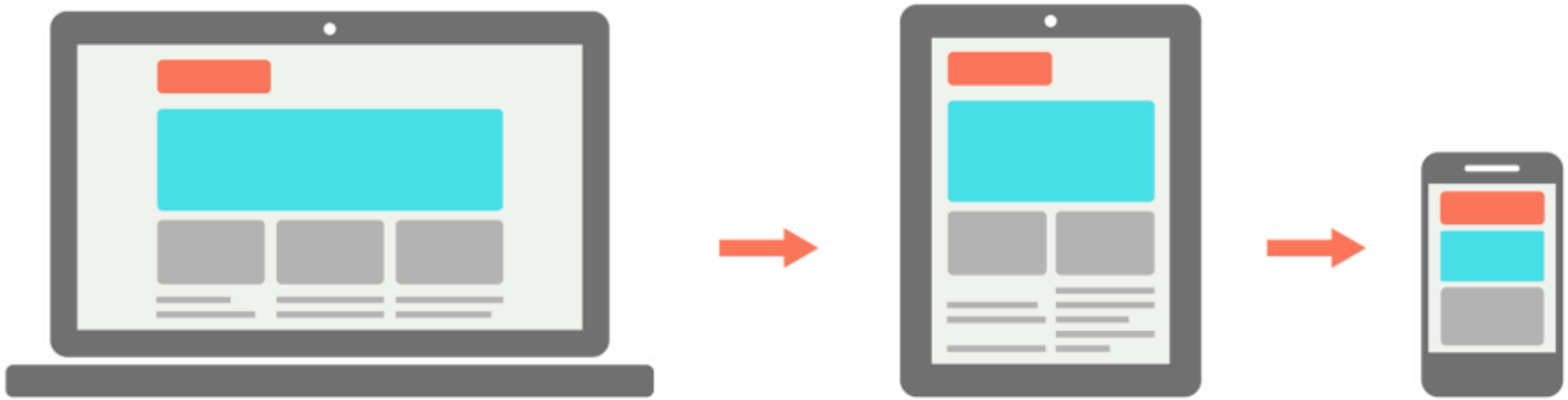
```
* {
        margin: 0;
        padding: 0;
        border: 0;
        text-decoration: none;
        list-style: none;
        vertical-align: base-line;
        box-sizing: border-box;
        outline: none;
}
```

CSS

- BOX MODEL: “caixa modelo” , estrutura fundamental a qual TODOS os elementos na página obedecem



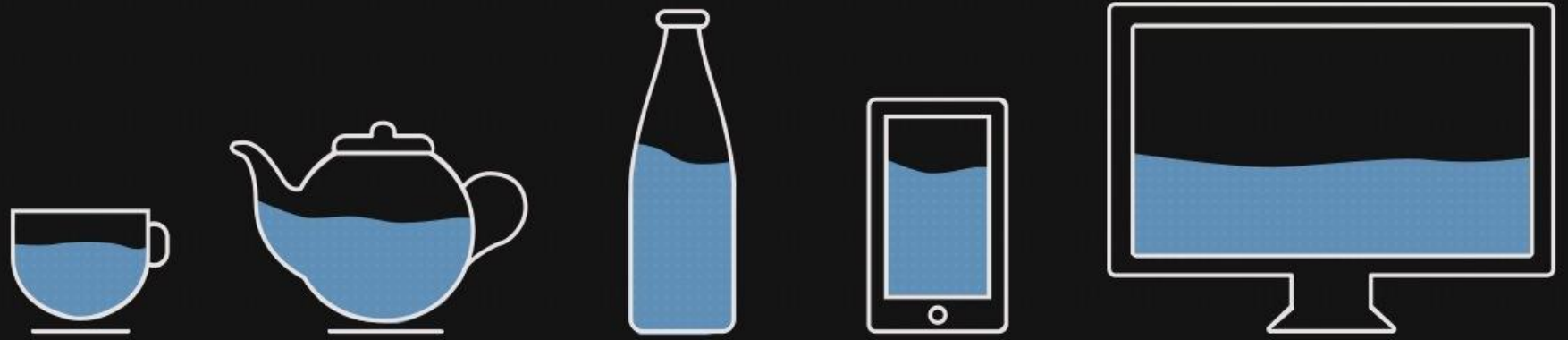

(content)
Here's some sample content, constrained by the padding thats been applied. Here's some sample content, constrained by the padding thats been applied.
border
padding*
background-image
background-color
margin*
* transparent elements



Responsive Web Design

Mobile First Web Design

CONTENT IS LIKE WATER



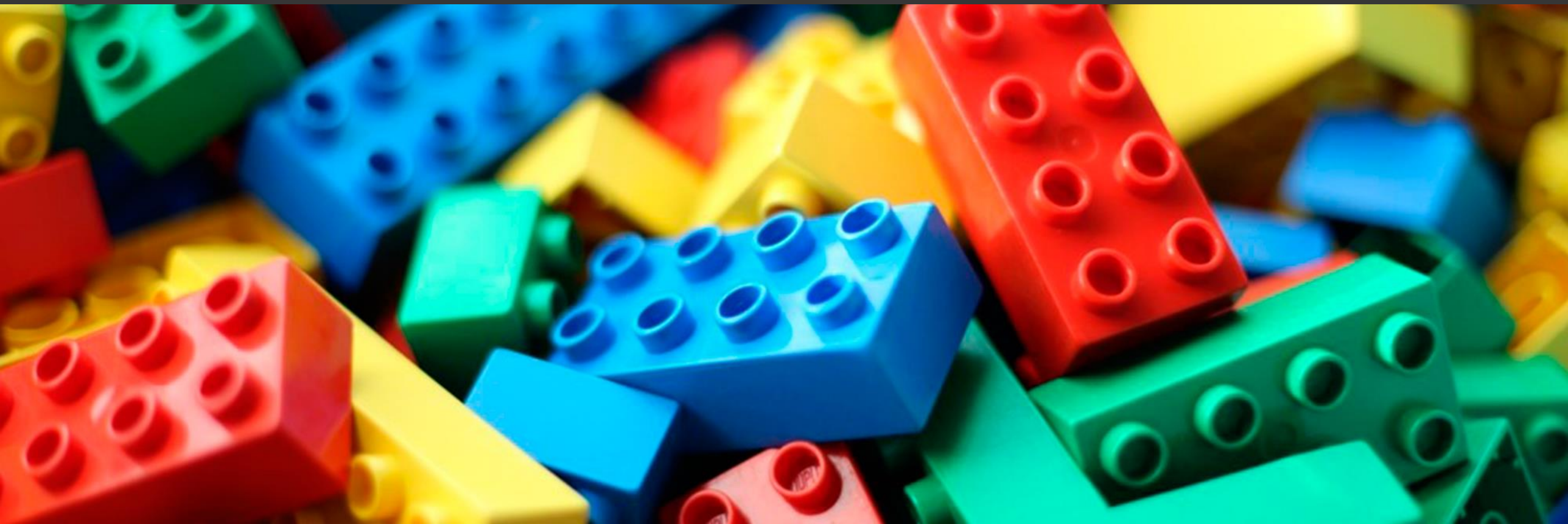
"You put water into a cup it becomes the cup.
You put water into a bottle it becomes the bottle.
You put it in a teapot, it becomes the teapot."

Josh Clark *(originally Bruce Lee)* - Seven deadly mobile myths

Illustration by Stéphanie Walter

CSS

PRINCÍPIO DA RESPONSABILIDADE ÚNICA



CSS

Mais um pitaco sobre boas práticas:

Utilize o princípio da responsabilidade única e crie padrões/nomenclaturas para seu css

```
.alunos-list{
}
.alunos-item{
}
.alunos-link{
}
```

```
.lista-de-alunos{
}
.aluno{
}
.link-de-aluno{
}
```

Inspecionar elementos

<DIA4>

Proposta de código

Implementar CSS nas páginas HTML já criadas, estilizando seus elementos.

Alterações de estilo em elementos clicáveis, responsividade com quebras em 800px e 500px.

Utilizar design flat e minimalista.

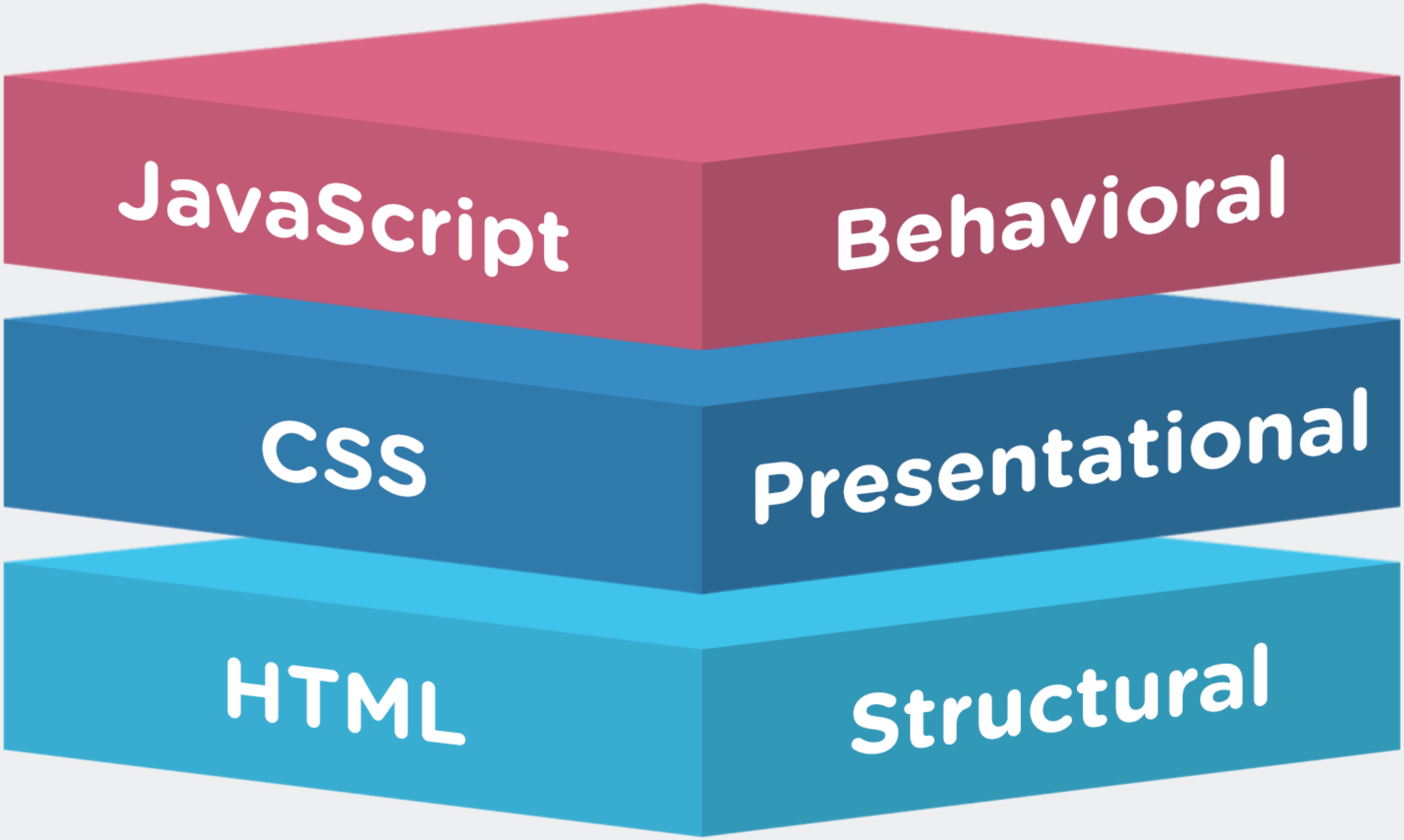
Utilizar paleta de cores coerente com seu conteúdo.
http://flatcolors.net/palettes.php

</DIA4>

<DIA5>

JAVASCRIPT

- É aqui que separamos os Web Designers dos Desenvolvedores Front-end :)


JavaScript
Behavioral
CSS
Presentational
HTML
Structural

JAVASCRIPT

- Dinamismo
- Validações
- Conexões (Requisições/Respostas)
- Armazenamento
- Controle total sobre o HTML e o CSS

- Açúcar
- Tempero
- TUDO QUE HÁ DE BOM


SUGAR

JAVASCRIPT

- Métodos de utilização:
  - Inline
  - Embutido
  - Externo

JAVASCRIPT

- Métodos de utilização:
  - Inline

```
<p onclick="alert('ola');">
        Meu texto
</p>
```

JAVASCRIPT

- Métodos de utilização:
  - Embutido

```
<script>
        alert( ‘ola’ );
</ script >
```

JAVASCRIPT

- Métodos de utilização:
  - Externo

No arquivo html:

```
<script src="caminho-da-pasta/script.js">
```

No arquivo js:

```
alert( 'ola' );
```

JAVASCRIPT

- Benefícios e malefícios de cada método de utilização:
  - Inline : reescrita de código, sem reutilização, restrição, poluição
  - Embutido : permite reutilização, mas ainda polui o principal html
  - Externo : permite reutilização , sem poluição

JAVASCRIPT

- Instâncias de variáveis: utilização da palavra reservada var

```
var variavelA;
var variavelInteira = 10;
var variavelString = ‘sim’;
var variavelArray = [ 0 , 1  , 2 , ‘sim’ , ‘nao’ ];
var variavelBool = true;
var variavelObjeto = { tipo : ‘curso’ , nome : ‘Desenvolvimento Front-end’ };
```

JAVASCRIPT

- Fraca tipagem

```
var variavelString = ‘sim’;
console.log( ‘valor da variavel string: ‘ + variavelString );
variavelString = 5;
console.log( ‘valor da variavel string: ‘ + variavelString );
```

JAVASCRIPT

- E se definirmos a variável sem o var?

```
<script>
    var variavelInteira = 10;
</script>
```

```
<script>
    variavelInteira = 10;
</script>
```

JAVASCRIPT

- E se definirmos a variável sem o var?

Podemos resultar em uma grande poluição do escopo global da aplicação, podendo, futuramente, acarretar alguns problemas como conflito entre variáveis, bugs e dificuldade de manutenção.

JAVASCRIPT

- E por falar em ESCOPO...

  Uma breve introdução sobre funções

JAVASCRIPT

Uma breve introdução sobre funções

```
<script>
        function criaVariavel(){
                a = 5;
        }
        function alertaVariavel(){
                alert( a );
        }
</script>
```

A aplicação inteira tem acesso à variável a, após a chamada da função criaVariavel, fazendo com que o valor 5 seja exibido ao chamar a função alertaVariavel

JAVASCRIPT

Uma breve introdução sobre funções

```
<script>
        function criaVariavel(){
                var a = 5;
        }
        function alertaVariavel(){
                alert( a );
        }
</script>
```

A aplicação não tem mais acesso à variável a, após a chamada da função criaVariavel, fazendo com que o valor e a variável existam somente no escopo da função criaVariavel, resultando em um erro pois não existe a no escopo da segunda função.

JAVASCRIPT

- Acessando elementos no DOM

```
<div id="minha-div">
        Essa é minha divisão
</div>
<script>
        var minhaDiv = document.getElementById( 'minha-div' );
        var texto = minhaDiv.innerHTML;
        alert( texto );
        minhaDiv.innerHTML = "O texto da minha div foi alterado :)";
</script>
```

JAVASCRIPT

- Acessando elementos no DOM: não precisamos necessariamente acessar os elementos somente através de ID

```
<div id="minha-div" class="minha-div" data-div="1">
        Essa é minha divisão
</div>
<form class="minha-div" name="formulario" data-div="form">
        <input type="text" name="nome" value="Gabriel Luiz Ramos">
</form>
<script>
        var primeiraDivPorId = document.getElementById( 'minha-div' );
        var primeiraDivPorClasse = document.querySelector( '.minha-div' );
        var todasDivPorClasse = document.querySelectorAll( '.minha-div' );
        var primeiraDivPorAttr = document.querySelector( '[data-div]' );
        var todasDivPorAttr = document.querySelectorAll( '[data-div]' );
        var elementoDoFormPorNome = formulario.nome;
</script>
```

JAVASCRIPT

- Por que aprender JS se existem diversas ferramentas atuais que fazem o que eu preciso de uma maneira mais simples?

Porque entender o que ocorre por trás de qualquer ferramenta faz com que você, como desenvolvedor, entenda a real necessidade de cada uma delas.

Seja um desenvolvedor sem vícios e independente

Inspecionar elementos

</DIA5>

<DIA6>

JAVASCRIPT

- Funções

JAVASCRIPT

- Podemos dividir a criação de funções JS em, basicamente, 3 tipos:
  - Funções nativas
  - Funções declaradas
  - Funções encapsuladas

JAVASCRIPT

- Funções nativas: são as funções que já existem, nativamente, na linguagem, como as definidas no console, window, document, de conversão etc.

```
<script>
        var url = window.location.href;
        console.log( url );
        document.getElementById( ‘qualquer-id’ );

        var numero = '684359145';
        numero = numero + 10;
        console.log( 'numero string + 10 = ' + numero );

        numero = parseInt( '684359145' );
        numero = numero + 10;
        console.log( 'numero int + 10 = ' + numero );
</script>
```

JAVASCRIPT

- Funções declaradas: são as funções declaradas ao longo do desenvolvimento utilizando a palavra reservada function.

```
<script>
        function falar(){
                alert( 'Olá!' );
        }
</script>
```

JAVASCRIPT

- Funções encapsuladas: são as funções declaradas ao longo do desenvolvimento e encapsuladas dentro de uma variável.

```
<script>
        var falar = function(){
                alert( 'Olá!' );
        }
</script>
```

JAVASCRIPT

- Funções encapsuladas e declaradas podem definir a forma como você escreverá seu código

```
<script>
        console.log( typeof falar );

        function falar(){
                alert( 'Olá!' );
        }

        console.log( typeof falar );
</script>
```

```
<script>
        console.log( typeof falar );

        var falar = function(){
                alert( 'Olá!' );
        }

        console.log( typeof falar );
</script>
```

Em ambos os casos acima poderemos utilizar nossas funções. Porém, ao utilizarmos a notação encapsulada, nossas funções só existirão e só poderão ser executadas após a página interpretar o trecho de código necessário. Ou seja: no primeiro exemplo, poderíamos executar a função antes de declará-la, já no segundo, caso fizessemos isso, teríamos um erro como retorno.

JAVASCRIPT

- Parâmetros: valores passados a funções, definidos dentro do parenteses da função e passados em sua chamada

```
<script>
        function falar( nome ){
                alert( "Olá, " + nome );
        }
        falar( "Fulano de tal" );
</script>
```

JAVASCRIPT

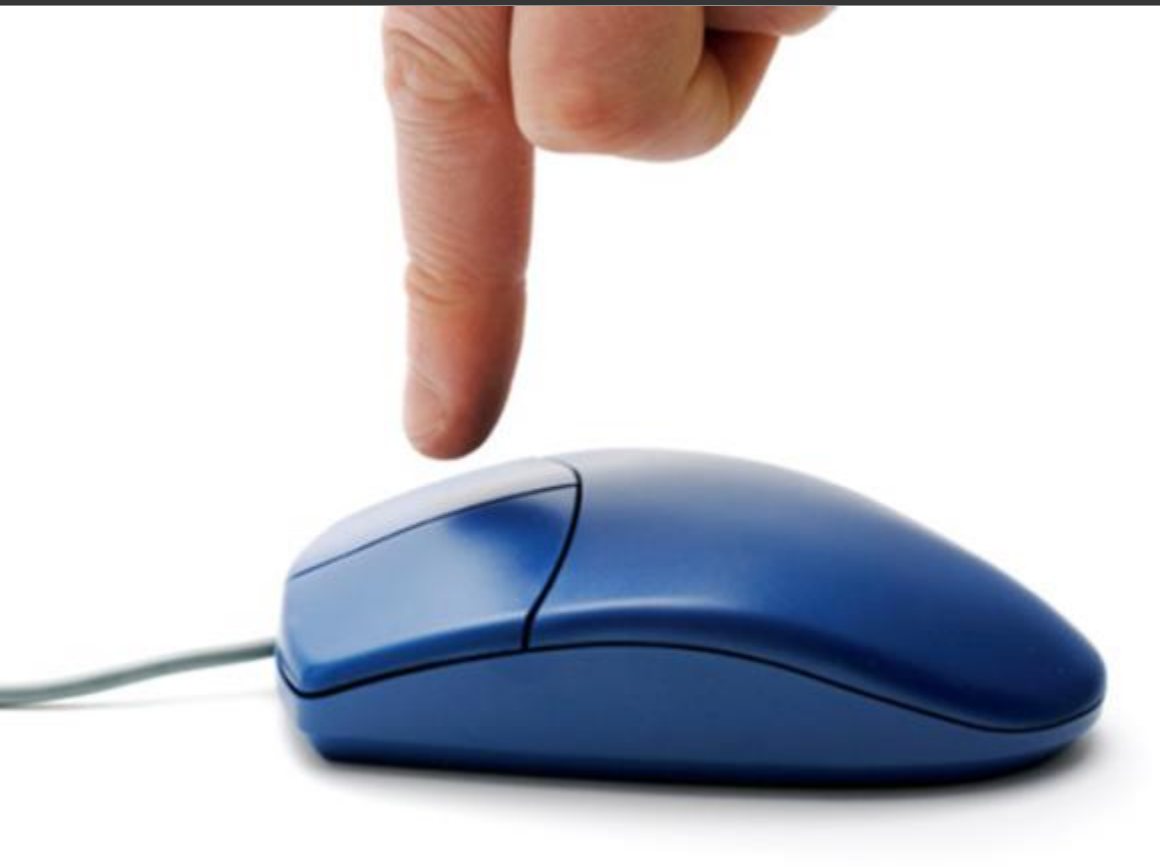
- Retornos: valores retornados (ou recebidos) de funções

```
<script>
        function getNome(){
                return "Fulado de tal";
        }
        function getIdade(){
                return 20;
        }
        function falar( nome ){
                alert( "Olá, " + nome );
        }

        var nome = getNome();
        var idade = getIdade();
        falar( nome );
</script>
```

JAVASCRIPT

- Eventos



JAVASCRIPT

- No JS podemos atribuir (atachar) funções às ações, dependentes de determinados eventos a serem executados pelo usuário.

```
<script>
        var clicado = 0;
        document.querySelector( '[data-button]' ).addEventListener( 'click' , function(){
                clicado++;
                document.querySelector( '[data-value]' ).innerHTML = "Você já clicou no botão " + clicado + "
                vezes";
        });
</script>
```

JAVASCRIPT

- O ato de “atachar” funções a determinados eventos também pode ser chamado de *bind* e pode ser inseridos e removidos de diversas formas, conforme necessidade.

```
<script>
        var botao = document.querySelector( ‘[data-button]’ );
        botao.addEventListener( 'click' , function(){
                alert( ‘clicado’ );
        });
        botao.removeEventListener( ‘click’ );
</script>
```

- *Bind* inserido e removido por eventListener

JAVASCRIPT

- O ato de “atachar” funções a determinados eventos também pode ser chamado de *bind* e pode ser inseridos e removidos de diversas formas, conforme necessidade.

```
<script>
        var botao = document.querySelector( ‘[data-button]’ );
        botao.onclick = function(){
                alert( ‘clicado’ );
        };
        botao.onclick = null;
</script>
```

- *Bind* inserido e removido diretamente por evento

JAVASCRIPT

- preventDefault: função que anula a função original de algum elemento. Para isso, o parâmetro de evento deve ser INJETADO ao declarar a função (comumente utiliza-se o nome e ou event para realizar essa injeção).

```
<a href="http://google.com" target="_blank" data-unbind>click-me!</a>
<script>
        var element = document.querySelector( '[data-unbind]' );
        element.onclick = function( e ){
                e.preventDefault();
        }
</script>
```

Essa abordagem é muito utilizada em formulários para garantir que, antes de seu disparo, eles tenham todos seus dados validados.

JAVASCRIPT

- Validações e tratamento de formulários

JAVASCRIPT

- Formulários são elementos do HTML que visam DISPARAR uma quantidade de dados à alguma página através de seu METHOD (forma de disparo) e ACTION (página que receberá os dados).

```
<form method=“get” action=“pagina-que-salva-formulario.php” >
        <input  type=“text” name=“nome_completo”>
        <button>Enviar</enviar>
</form>
```

Exemplo de formulário utilizando o método GET para enviar somente um campo (nome_completo) para a página “pagina-que-salva-formulario.php”

JAVASCRIPT

- Method: refere-se à forma como a requisição será feita através do disparo de dados, que são subdivididos em
  - GET: ação que visa RETORNAR dados e/ou que permite que dados sejam trafegados pela url. (http://teste.com/usuario.php?nome_completo=curso)
  - POST: ação que visa ENVIAR dados e/ou que não permite que dados sejam trafegados pela url. (http://teste.com/usuario.php)

Dados enviados via GET também podem ser acessados utilizando as propriedades de window dentro de seu sistema (por exemplo: window.location.hash)

JAVASCRIPT

- A validação de formulário ocorre ao impedir que o próprio formulário dispare ao clicar no botão

```
<form method="get" action="pagina-que-salva-formulario.php"  name="meu_form">
        <input  type="text" name="nome_completo">
        <button>Enviar</enviar>
</form>
<script>
        meu_form.onsubmit = function( e ){
                e.preventDefault();
        }
</script>
```

Exemplo de formulário utilizando o método GET para enviar somente um campo (nome_completo) para a página "pagina-que-salva-formulario.php"

JAVASCRIPT

- Validações consistem em funções que verificam se nossos valores foram preenchidos corretamente.

```
<form method=“get” action=“pagina-que-salva-formulario.php”  name=“meu_form”>
        <input  type=“text” name=“nome_completo”>
        <button>Enviar</enviar>
</form>
<script>
        meu_form.onsubmit = function( e ){
                e.preventDefault();
                var nome = this.nome_completo.value;
                if( !nome_completo.trim() ){
                        alert( “Digite o nome corretamente” );
                }
        }
</script>
```

COMPARTILHAMENTO DE DADOS

COMPARTILHAMENTO DE DADOS

- Em diversos sistemas, aplicações e sites, muitas vezes é interessante a troca de informações (sendo elas sigilosas ou não). Existem basicamente 2 formas de escrever dados a serem compartilhados e elas são utilizando:
  - JSON: sigla para Javascript Object Notation (notação de objetos javascript).
  - XML: sigla para Extensible Markup Language (linguagem de marcação extensiva).

COMPARTILHAMENTO DE DADOS

- JSON x XML

XML

```
... >
  <name>Barry & Associates, Inc.</name>
  <phone>612-321-8156</phone>
  <street1>14597 Summit Shores Dr</street1>
  <street2></street2>
  <city>Burnsville</city>
  <state>MN</state>
  <postalcode>55306</postalcode>
  <country>United States</country>
< ...
```

JSON

```
{
  "name"        : "Barry & Associates, Inc.",
  "phone"       : "612-321-8156",
  "street1"     : "14597 Summit Shores Dr",
  "street2"     : "",
  "city"        : "Burnsville",
  "state"       : "MN",
  "postalcode"  : "55306",
  "country"     : "United States"
}
```

COMPARTILHAMENTO DE DADOS

- JSON, por ser uma “extensão” do Javascript tem sua manipulação muito mais facilitada. Podemo escrever objetos javascript como variáveis ou como arquivos.

No arquivo.js

```
<script>
        var meuObjeto = {
                nome : “Curso” ,
                tipo : “Desenvolvimento web”
        };
</script>
```

Em arquivo a parte

```
{
        “nome” : “curso” ,
        “tipo” : “Desenvolvimento web”
}
```

Exemplo de escrita em JSON em variáveis e em arquivos.

COMPARTILHAMENTO DE DADOS

- Utilizando JSON podemos transformar nossos objetos em objetos interpretáveis pelo Javascript, ou em strings JSON para serem tratados (JSON.stringify / JSON.parse).

No arquivo.js

```
<script>
        var meuObjeto = {
                nome : “Curso” ,
                tipo : “Desenvolvimento web”
        };
        var meuObjetoString = JSON.stringify( meuObjeto );
        console.log( meuObjetoString );
        var meuObjetoNatural = JSON.parse( meuObjetoString );
        console.log( meuObjetoNatural );
</script>
```

ARMAZENAMENTO DE DADOS

ARMAZENAMENTO DE DADOS

- É possível realizar o armazenamento de alguns dados em nosso navegador utilizando a api WebStorage, que é dividida em:
  - LocalStorage: possui um princípio de armazenamento local, sendo necessário, para exclusão, um procedimento manual.
  - SessionStorage: possui um princípio de armazenamento em sessão, tendo seus dados apagados ao encerrar a janela do navegador.

A plataforma WebStorage baseia-se no padrão chave – valor para seu armazenamento, similar à uma chave simples de um array. Por exemplo:

["nome"] = "Curso"

["tipo"] = "Desenvolvimento Web"

ARMAZENAMENTO DE DADOS

- Para manipularmos a plataforma WebStorage, utilizamos Javascript, utilizando notações de objetos.

```
<script>
        localStorage.nome = “Curso”;
        localStorage[ “tipo” ] = “Desenvolvimento Web”;

        sessionStorage.nome = “Curso”;
        sessionStorage[ “tipo” ] = “Desenvolvimento Web”;
</script>
```

ARMAZENAMENTO DE DADOS

- Exitem também algumas funções, que auxiliam no desenvolvimento com WebStorage, tais como:
  - .setItem(key, value)
  - .getItem(key)
  - .removeItem(key)
  - .clear()
  - .key(index)
  - .length

Inspecionar elementos

</DIA6>

<DIA7>

Proposta de código

Finalização do sistema,
Seguindo as seguintes etapas:

- Identificar a “chave primária” de acordo com o sistema que estiver realizando
- Validar ao menos 2 de seus campos de cadastro, exibindo alerta de erro
- Exibir na página de listagem e de conteúdo detalhado, seus respectivos dados

REVIEW DO CURSO

- Web
  - Funções
  - Arquitetura cliente x servidor
- HTML
  - Marcação
  - Estrutura
  - Tags
  - Atributos
  - Cross Browser e renderizações
  - DOM
  - Semântica
  - Acessibilidade
- CSS
  - Estrutura e seletores
  - Métodos de Utilização (inline, embutido e externo)
  - Regras reduzidas
  - Regras prefixadas
  - Seletores especiais (pseudo-seletores , pseudo-classes)
  - Transições
  - Animações
  - Responsividade
  - Reset
  - Box-model
- JS
  - Métodos de utilização (inline, embutido e externo)
  - Instâncias, tipos de variáveis e programação
  - Escopo global e local
  - Acessando elementos no DOM
  - Funções (nativas, declaradas e encapsuladas)
  - Eventos (binds ou atachamentos e preventDefault)
  - Validações
  - Método de disparo de formulários
  - JSON x XML
  - Armazenamento de dados (webstorage)

</DIA7>

Referências

- Can I Use? http://caniuse.com
- Stackoverflow:
  - EN: http://stackoverflow.com/
  - PT: http://pt.stackoverflow.com/
- W3C: http://www.w3c.br/
- W3School: http://www.w3schools.com/
- Calculadora de Especificidade CSS: https://specificity.keegan.st/
- Javascript.com: http://javascript.com