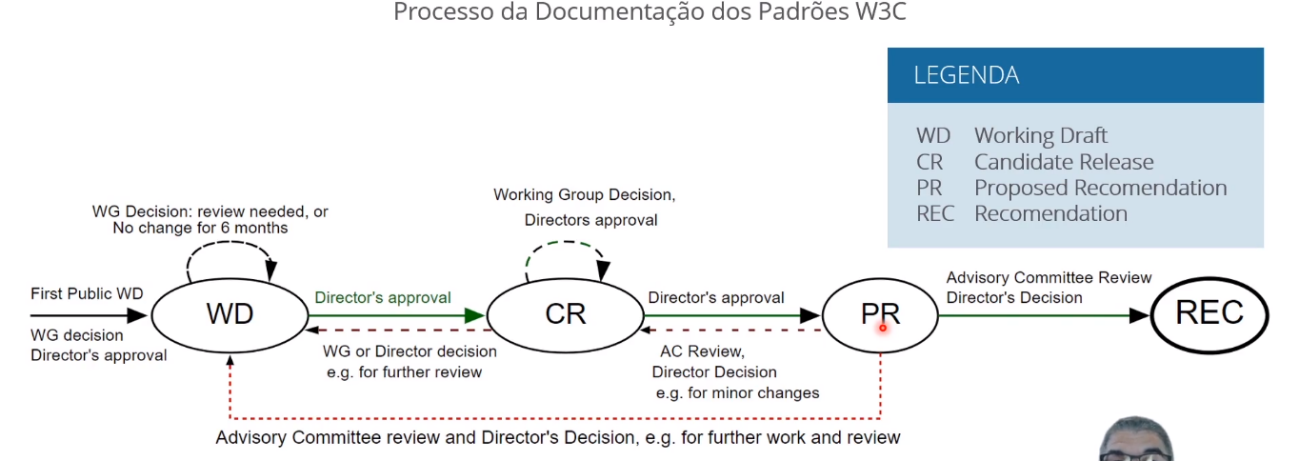
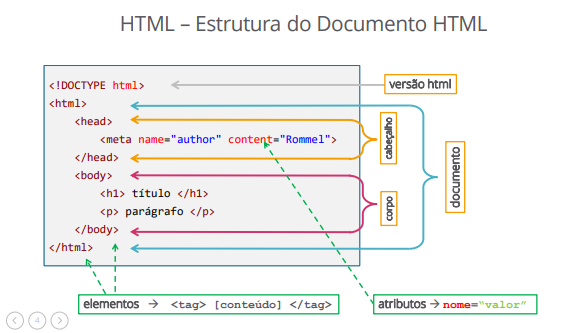
Aula 1.1

* ***Evolução da Web***
  + Web: teia mundial de informações
  + World Wide Web: Sistema de informações interligados por meio de hiper textos cujo acesso de da por meio de browsers
  + A web 1.0 – leitura
  + A web2.0 – leitura e escrita(colaboração) – pessoas participam na web
  + A web 3.0 – leitura, escrita e entendimento (caráter compreensível) – começa a ser entendido por algoritmos.
  + A web 4.0 leitura, escrita, entendimento e agentes inteligentes
  + Existe um problema de compatibilidade dos browsers com alguns sites da web – solução: Criação do World Wide Web Consortium(W3C) que foi responsável por manter e evoluir padrões e tecnologias da web
    - Processo de documentação dos Padrões W3C:
  + É necessário a padronização para melhorar compatibilidade e estabilidade.

Aula 1.2

* ***Tecnologias HTML e CSS***
  + Hyper text markup language
  + Linguagem de marcação de hypertextos – combinação imagens, vídeos e áudios no mesmo conjunto de informações
  + Estrutura do documento HTML:
  + Tag é um código usado para marcar o início e, onde for requerido, o fim de um elemento HTML (abertura e de fechamento de um elemento)
  + Os Atributos servem para definir uma propriedade de um elemento HTML
  + HTML é usado para estruturar o conteúdo, já o CSS para modificação e formatação da página.
  + Como funciona:
    - 1- Carregamento do HTML
    - 2- Interpretação do HTML
    - 3- Identificação da necessidade do CSS e seu carregamento
    - 4- Interpretação CSS
    - 5- Criação arvore DOM (junção das duas tecnologias)
    - 6- Display
  + Como associar:
    - Nome do elemento
    - Id do elemento – uso do #
    - Classe do elemento – uso do .

Aula 1.3

* ***Sintaxe da Linguagem HTML***
  + Cada elemento é marcado por uma tag de abertura e fechamento

<body> [conteúdo] <body>

* + Formatos de elementos:
    - Elementos filhos
    - Elementos com texto
    - Elementos vazios – não há necessidade de fechamento
    - Elementos de conteúdo misto
  + Atributos
    - São definidos pelo par nome/valor
    - Alteram o funcionamento dos elementos HTML
    - Cada elemento possui um conjunto de próprio de atributos.
    - Os atributos possuem valores livres ou pré-definidos
    - Exemplo: 
  + Comentários
    - <!—e -->
    - Exemplo:
  + Doctype
    - Indica a versao do HTML e a linguagem usada
    - Quirks mode : modo estranho – sem reconhecimento da linguagem – identificacao do proprio modo
    - Strict mode: modo estrito – reconhecimento da linguagem – linguagem já especificada
  + Cabeçalho(head)
    - Primeira parte do arquivo representada pela tag <head>.
    - Metadados da página – dados de dados
  + Corpo(body)
    - Segunda parte do arquivo representada pela tag <body>.
    - Todo conteúdo da página.

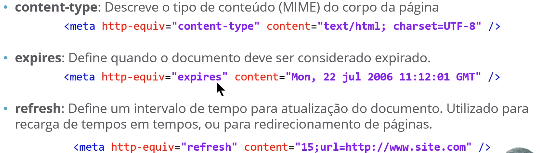
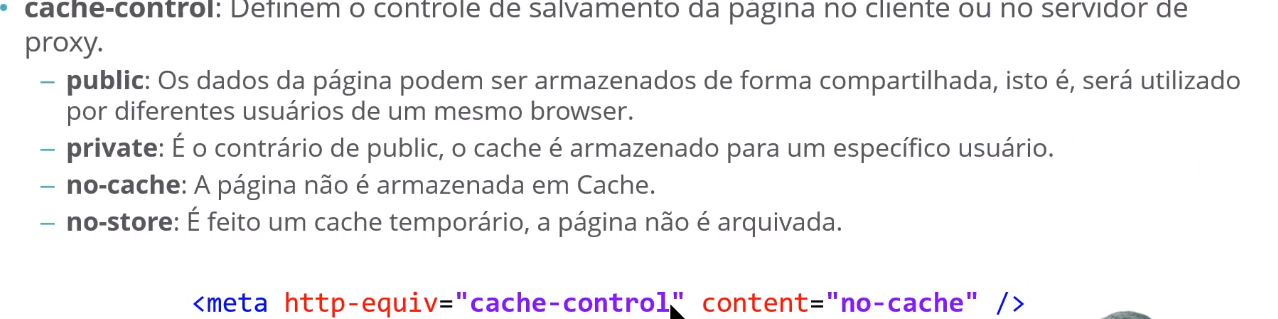
Aula 1.4

* ***Elementos HTML***
* Tipos de elementos quanto ao fluxo na página
  + Elementos Inline: Se disponha em linhas até a falta de espaço

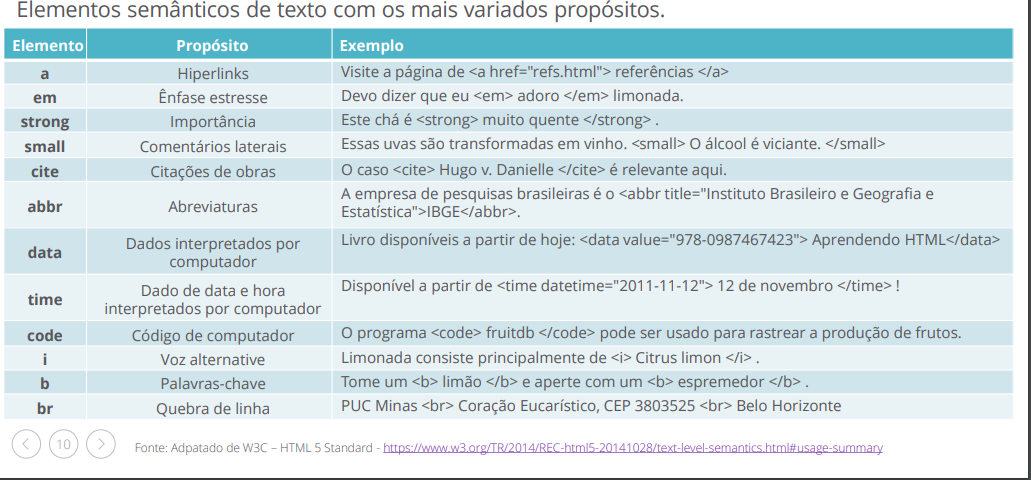
Ex: <img> , <span> <a> <button>

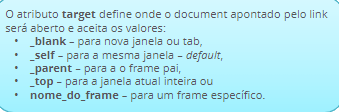
* + Elementos Block: Iniciam uma nova linha sempre – ocupa todo o espaço

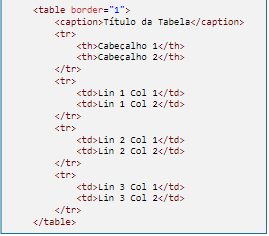
Ex: <div><canvas>

* Tipos:
  + **Metadados**
    - Informações sobre a página
    - <title> : Título da aba
    - <style> : Adiciona estilo para algum elemento
    - <link> : Definem ligações com outros arquivos
    - <meta> : define qualquer informação de metadados que não podem ser definidos por outros elementos HTML
    - <meta charset>: Define a codificação de caracteres utilizada no documento. – UTF-8

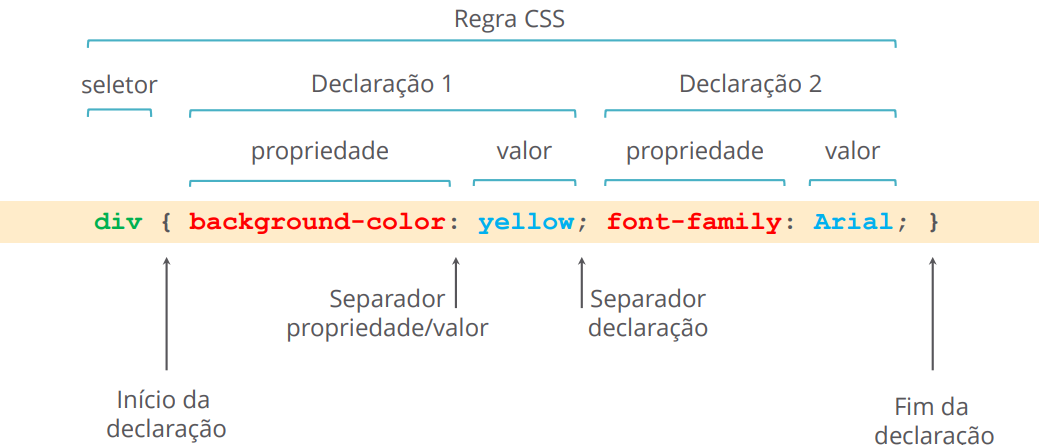
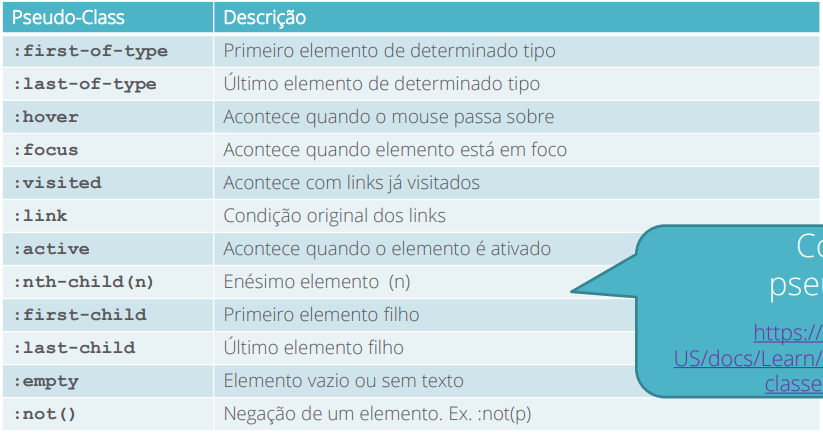
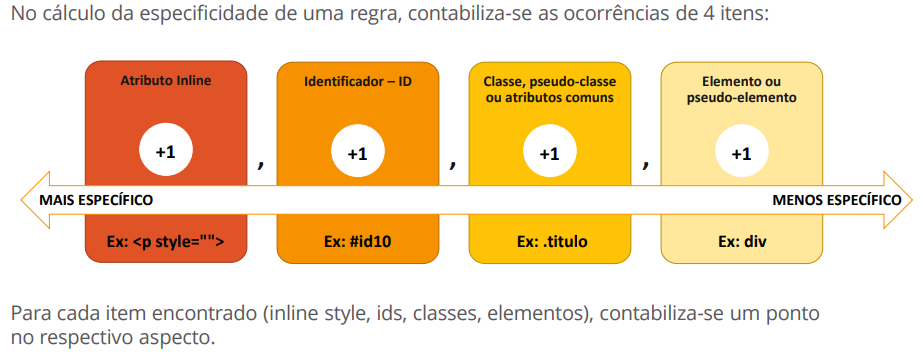
**Textuais**

* + - <h1> : Títulos
    - <p>: Parágrafos
    - <br> : Quebra de linha
    - <ul> : Lista sem ordenação
    - <ol> : Lista com ordenação
    - <dl>: Lista
    - <a>: Links

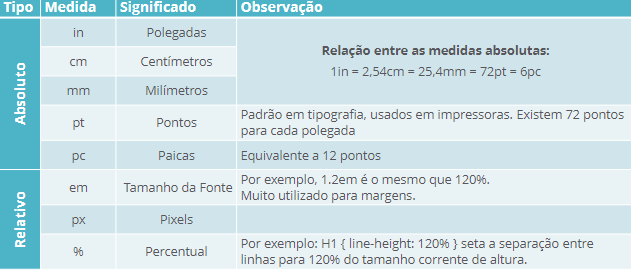
OBS 

* + **Multimídia** 
    - <img> : Inserir uma imagem – (Inline)
  + **Tabela**
    - <table>: Criação de uma tabela
    - <caption>: Título da tabela
    - <tr> Linha da tabela
    - <th> Coluna
    - <td rowspan> Agrupar
  + **Estrutura**
    - <div>: Separação de partes
    - <nav> <section> <article> <header><footer> : Significado
  + **Avançados**
    - <form>: formulário
    - <input>: campo
    - <label> : controlar o checkbox – juntamente com o input.
    - <script>: Referência para arquivo script
    - <iframe>: Fração de conteúdo de outro local- Conteúdo embutido

***Aula 1.5 - Sintaxe da Linguagem CSS***

* OBS (Como linkar um arquivo CSS ao HTML) 
* **Regras**:
  + Composta por seletor + declarações
  + Seletor: especifica quais elementos de uma página irão receber transformações
  + Declaração: Propriedade + valor
  + Exemplo:
* **Tipos de seletores**:
  + Elemento – especifica qual elemento a regra se aplicará
  + Identificador(id) – Se usa a **#** para declarar
  + Classe(class) – Se usa um **.** para declarar
  + Atributo gerais – Se usa entre **[]**
  + Pseudo-Classe – Classes não explicitas , se usa **:**
    - OBS: Hover – Quase passa o mouse
    - Outros tipos de pseudoclasse:
  + Pseudo-Elemento – Elemento não explicito , se usa **::**
  + Universal : Todos elementos, se usa **\***
* **Combinação de seletores**:
* **Comentários**:
  + Utiliza-se /\* para criar comentários ou desativa uma regra
* **Prioridade de Seletores**:
  + Ordem de leitura:
    - A última regra irá prevalecer
  + Especificidade:
  + Importância:
    - !important – se torna prioritária sob as demais.

**Aula 1.6 - Formatação de Páginas com CSS**

* ****Elementos inline irá dispor em uma linha, já os elementos blocks gastam a linha toda
* A propriedade margin serve para centralizar algum elemento. Ex: **margin: 0 auto;**
* O CSS pode transformar um elemento através da propriedade **display**:
  + Display:none
  + Display:block
  + Display:inline
  + Display:inline-block
* **Box model**:
  + height e width

Tamanho básico do elemento

* Forma para aplicação de tamanho – *ABCD*(ordem horaria )
  + margin

Espaçamento ao redor do elemento

* + border

Borda em torno do elemento

* + padding

Espaçamento entre a borda e o conteúdo

* + background permite controlar:

cor de fundo do elemento – imagem como fundo

* **Tipografia**
  + **font-family**: define o tipo da letra (fonte) empregada no texto– font-family: arial, helvetica, sans-serif
  + **font-size**: especifica o tamanho da letra– font-size: xx-small | x-small | small | medium | large | x-large | xx-large | smaller | larger |

10px | 80%

* + **font-style**: define o estilo da fonte– font-style: normal | italic | oblique
  + **font-weigth**: indica a largura dos caracteres– font-weigth: normal | bold | bolder | lighter | 100 | 200
  + **font-variant**: indica se os caracteres minúsculos devem ser mostrados como maiúsculas– font-variant: normal | small-caps
  + **line-height**: define a altura da linha– line-height: normal | 1.6 | 80% | 200%
  + **color**: altera a cor do texto
  + **text-align** – Define o alinhamento do texto – text-align: left | right | center | justify
  + **text-shadow** – Define se o texto terá sombra – text-shadow: none | x y [cor]
  + **text-decoration** – Define como o texto será decorado.– text-decoration-line: none | underline | overline | line-through – text-decoration-style: solid | wavy | dashed – text-decoration-color: [cor]
  + **letter-spacing** | word-spacing – Espaçamento entre letras e entre palavras em um bloco de texto – leter-spading: normal | 2px | 0.1em
  + **text-transform** – Define se o texto deve vir em maiúscula, minúscula ou capitalizado. – text-transform: none | capitalize | uppercase | lowercase
* **Backgrounds**
  + **background-color**: cor de fundo do elemento- background-color: yellow;
  + **background-image**: imagem ou gradiente de fundo do elemento. – Background-image: url(<http://imagens.com/fundo.jpg>); – background-image: linear-gradient (red, blue);
  + • **background-repeat**: indica como repetir a imagem no fundo caso essa seja menor que o elemento. – background-repeat: no-repeat | repeat | repeat-x | repeat-y | space | round
  + **background-position**: posição inicial do fundo em relação ao elemento HTML. – background-position: top | left | center | right | bottom
* **Posicionamento**
  + **position: static**

– Valor padrão para os elementos.

– O elemento é considerado como não posicionado.

– O elemento segue o fluxo de renderização da página normalmente.

– Não é afetado pelas propriedades: top, bottom, left, right e z-index

* + **position: relative**

O elemento segue o fluxo de renderização da página normalmente.

– O elemento é considerado como posicionado.

– É afetado pelas propriedades: top, bottom, left, right e z-index

– Estabelece como ponto zero o canto superior esquerdo

* + **position: fixed**

O elemento NÃO segue o fluxo de renderização da página

– O elemento é considerado como posicionado.

– É afetado pelas propriedades: top, bottom, left, right e z-index

– O elemento se mantém fixo na posição estabelecida sem refletir à

rolagem da tela

* + **position: absolute**

– O elemento NÃO segue o fluxo de renderização da página

– O elemento é considerado como posicionado.

– É afetado pelas propriedades: top, bottom, left, right e z-index

– Estabelece como ponto zero o canto superior esquerdo do objeto .

**Aula 2.1 Layouts Web/Mobile e Responsividade + Apresentação**

* **Tipos de layout**
  + Liquido/adaptativo
    - Utilizam valores em % para se ajustar na tela
    - Se **encaixa** redimensionando o conteúdo em qualquer dispositivo
  + Fixo
    - Construído para um tamanho especifico em pixels
    - Pode gerar barras de rolagem ou re-escalar a página toda para caber. (Zoom)
    - Quando a tela é maior que o conteúdo gera a centralização.
    - Quando a tela é menor que o conteúdo gera a barra de rolagem.
    - Mais barato.
  + Responsivo
    - Construção baseada em porcentagens e alguns conteúdos em pixels (60% para 40%)
    - Layout inteligente – principal diferença com o liquido.
    - Mais caro
    - Usa CSS e JavaScript
    - Trabalho pesado para seu desenvolvimento.
    - O que é responsividade? – Comportamento inteligente para se adaptar a diferentes tipos de telas. Para isso, elementos podem ser escondidos, priorizados ou inseridos.
  + Exemplos
    - 1. Tela do gmail – Liquido
      2. Tela do facebook – Fixo
      3. Tela do Globo.com – Responsivo
* **Mobile First**
  + Projeto que faz a criação para dispositivos moveis primeiro e depois as adaptações para o desktop.
  + Vantagens
    - Acesso via smarthphone é predominante – prioriza o UX
    - Menos gambiarras
    - Ranqueamento no Google(SEO)
    - Otimização no carregamento.
  + Importância
    - Mapas conceituais
    - Mapa do site
    - Wireframe
    - Taskflow

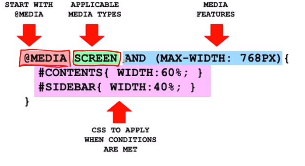
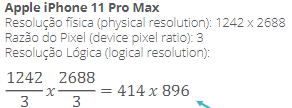
**Aula 2.2 CSS Viewport e Media Queries**

* **Viewport**
  + É a área visível do usuário de uma página da web
  + Varia de acordo com o dispositivo
  + Era comum que as páginas tivessem um tamanho fixo
  + É necessário que se especifique o viewport para que o couteudo seja adaptado para certo dispositivo
  + ****Para definir a viewport – (tag meta):

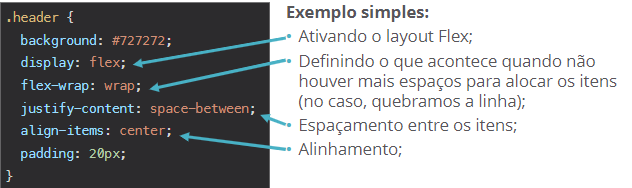
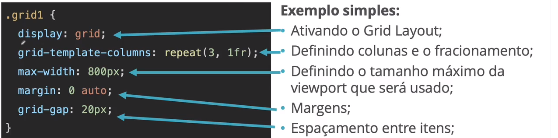
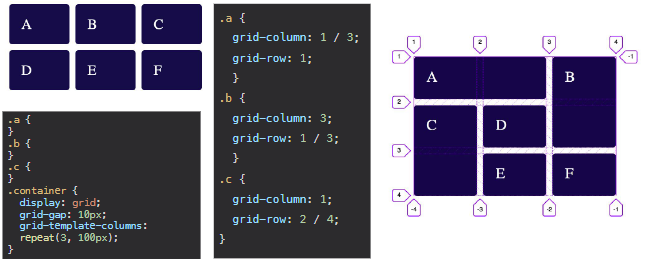
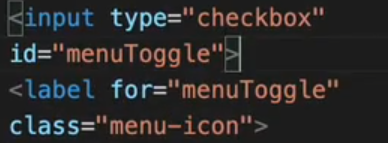
X = valor fixo em PX ou device-width

Y = initial-scale=1

Z = minimum-scale / maximum-scale / user-scalable

* + Para adaptar seu site se usa o Media Queries
* **Media Queries**
  + Seletores CSS que irão operar apenas em determinados tamanhos.
  + Sintaxe:
  + ****Importante saber o breakpoints(tamanho especifico).
  + Em monitores 1px = 1px, porem em dispositivos com telas pequenas é diferente – é preciso saber a razão de pixel.
  + Ex:
  + Existem tabelas feitas na internet com os breakpoints.
  + ****Tamanhos:

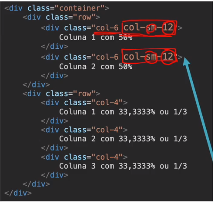
**Aula 2.3 Layout Flex e Grid**

* **O que são?**
  + São estruturas CSS que trabalham contensão de elementos baseados em posição, tamanho e alinhamento.
  + Recomendação para a construção de layouts responsivos.
  + As duas técnicas podem trabalhar em conjunto
  + Flexbox- Uma dimensão
  + Grid – Duas dimensões
* **Layout Flex**
  + Usado para alinhar componentes em 1 dimensão (linha ou coluna)
  + Espaçamento calculado automaticamente.
  + Não permite a interação entre linha e coluna.
  + ****Sintaxe:
* **Layout grid**
  + Multidimensional
  + Indicado para layouts mais complexos onde os elementos tem interação entre a linha e coluna.
  + É mais dificl de ser implementado – necessário o fracionamento das div
  + ****Sixtaxe:
  + Disposição dos objetos:
  + ****
* Video aula Mao na massa- Menu mobile
  + <label> ~e usada para controle do input.
  + Ex:
  + Para fixar o menu basta usar – position: fixed

**Aula 2.4 Framework Bootstrap**

* Framework – abstração que une códigos comuns entre vários projetos de software promovendo uma funcionalidade genérica.
* Quando existe uma grande intercessão entre os componentes é justificável criar um framework.
* Biblioteca – componentes que trabalham de forma independentes.
* Blocos necessários para o funcionamento do Bootstrap:
* Bootsnipp.com – templates

**Aula 2.5 Framework Bootstrap – Grid**

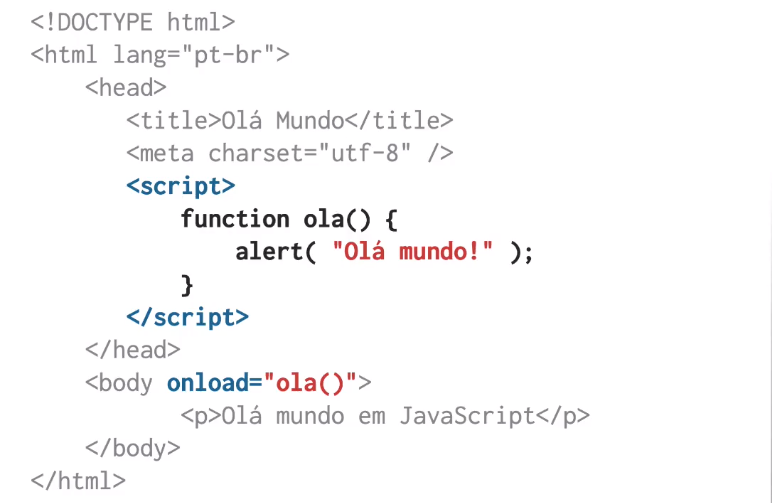
* Sistema de grid do Bootstrap:
  + Dividido em 12 colunas.
  + ****Para a construção de coluna é necessário estar dentro da class container, depois é necessário a criação de linhas ( “row”).
  + ****Tornando responsável usando Bootstrap:
  + Divisão:
    - <header>
    - <main>
    - <footer>
  + Durante a programação do HTML é necessário ir já criando a responsividade

**Aula 2.6 - Framework Bootstrap – Componentes**

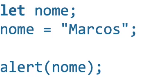
* São as estruturas de UI(user interface) oferecidas pelo bootstrap para a montagem dos layouts.
* Get started – componentes.
* São utilizados para aprimorar seu layout, além de implementar funções para sua página web.
* ****Efeito quando o mouse está em cima:
* Figma.com – criar layouts.

**Unidade 3 - Tornando o site dinâmico com JavaScript**

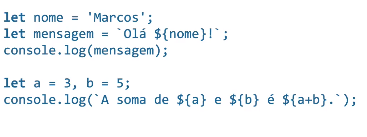
**Aula 3.1 – JavaScript**

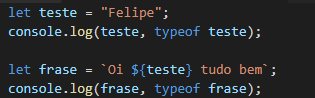
* A linguagem JS permite que transforme a página web interativa
* HTML – estrutura CSS – apresentação – JS – comportamento
* Controle de formulário é a aplicação mais básica- poupa o tempo de espera e a sobrecarga do servidor.
* Feita para criar apenas scripts e não um aplicativo todo.
* Manipulação personalização ou automação de recursos existentes.
* Execução no navegador:
  + Acelera interação.
  + Limitações de segurança.
* JS vem da ECMAScript e não tem nada a ver com java.
* Como colocar JS:
* ****Todo código fica dentro da tag <script>, porém é possível linkar um arquivo completo.
* Elemento onload – quando a página for carregada.

**Aula 3.2 – Variáveis**

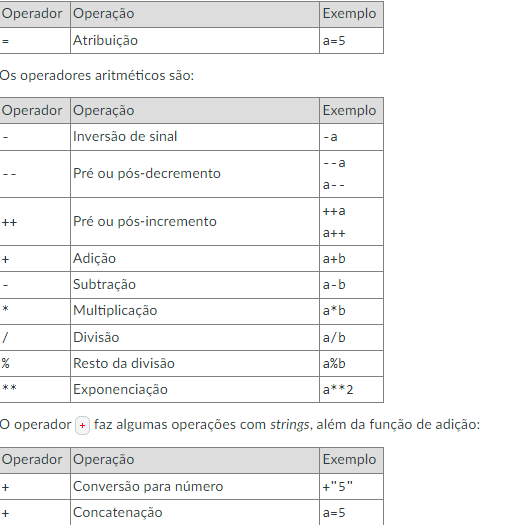
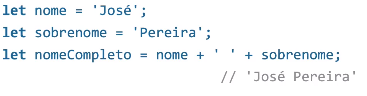
* Variável:
  + Espaço temporária para armazenamento de dados.
  + Poderá ser recuperada em qualquer momento.
  + Como declarar:
  + Uma variável simples só pode conter um valor de cada vez.
  + Tipos:
    - String
    - Inteiro
    - Booleando
  + Fracamente tipada – o tipo pode ser mudado.
  + Declarar:
    - 1- Por meio da instrução LET
      * Declara variável que só existira dentro do bloco que foi declarada
    - 2- Por meio da instrução VAR
      * Declara variável cujo escopo é global.
    - 3- Por meio de nenhuma instrução
      * Teoricamente é a mesma coisa que VAR, porém é uma má pratica.
    - 4- Por meio da instrução CONST
      * Constante e não é variável e só fica no bloco.

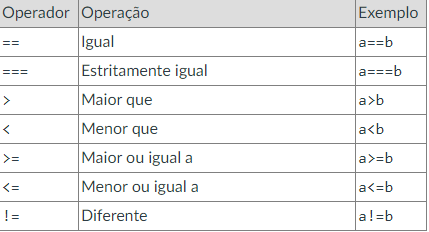
**Aula 3.4 – Tipos de Dados**

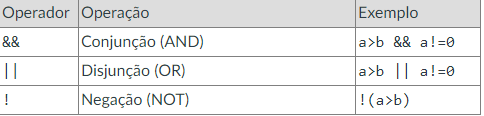
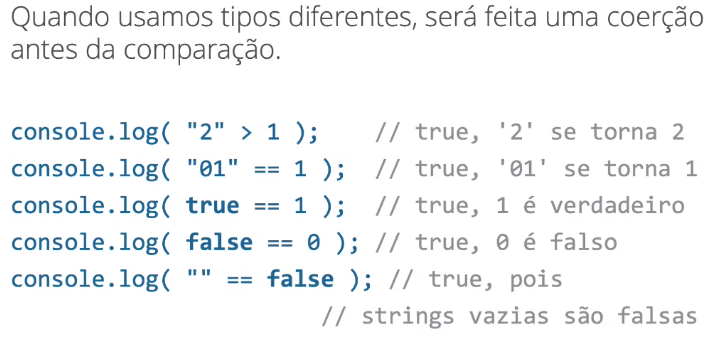
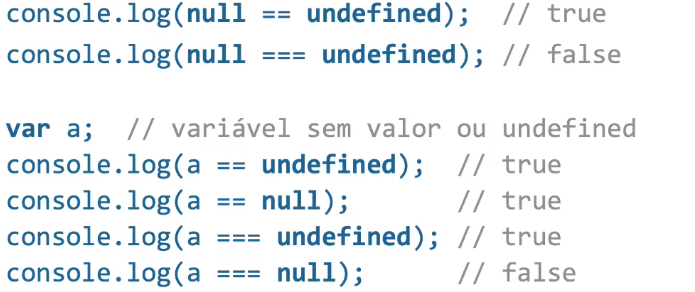
* Permite alterações – TOMAR CUIDADO
* Números
  + Todos os tipos são **number**
  + Valores especiais:
    - Infinity
    - -Infinity
    - NaN
* BigInt
  + Números muito grandes
  + Usado em criptografia
  + Coloca-se um n no final do número
* String
  + Texto e caractere
  + Existem 3 formar de delimitação de uma string:
  + Uma única linha
* Template Strings
  + Criação de uma string de várias linhas.
  + Representar código HTML e CSS
  + Permitir incorporações – por meio de ${x}
  + EX:
* Lógico(booleando)
  + Dois valores: True e False
  + Boolen
* Null e Undefined
  + Valor vazio e desconhecido
* Para testar tipo de variável usa-se **typeof** além de comparações**.**
* Exemplo da diferença entre string e template string:



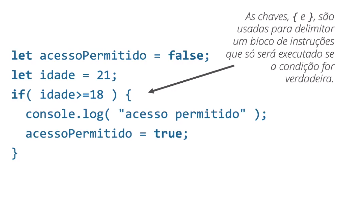
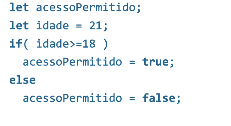
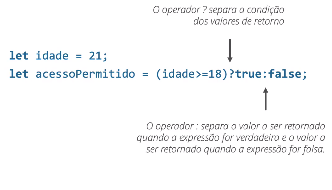
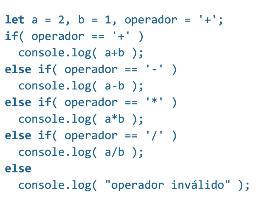
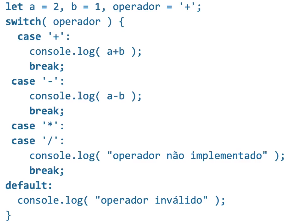
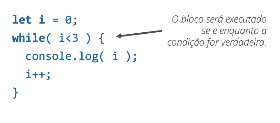
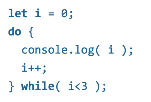
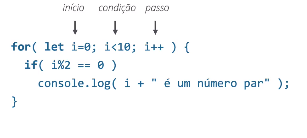
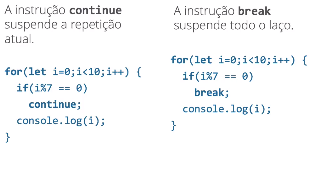
**Aula 3.5 – Operadores aritméticos e de concatenação**

* Operador unário = apenas um operando.
* Operador bínario = dois operandos
* Tabela:
* Não faz diferença se o sinal de decremento vem antes ou depois, porém em uma expressão sim.
* Concatenação de strings :
  + Junção de strings
  + Ex:

**Aula 3.6 – Operadores relacionais e lógicos**

* Tabela:
* Comparação de strings:
  + Maior ou menor, mesmo tamanho = ordem alfabeto
  + Maior ou menor, tamanho diferente = ordem de quantidade de letras.
  + Maior ou menor, mesmo tamanho e tipo diferente (a , A) = ordem de bits(tabela de caracteres ASCII)
* Comparacao de tipos diferentes:
* Usa-se === para comparar valor e tipo.
* Comparação de null e undefined:

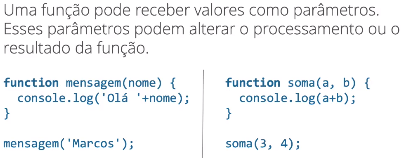
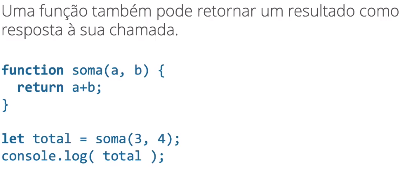
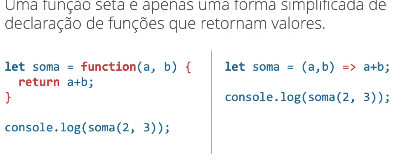
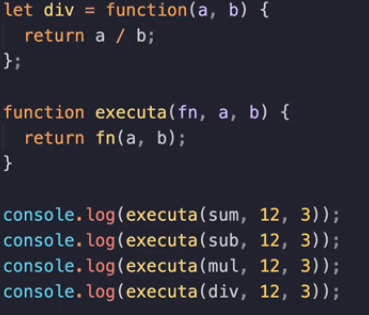
**Aula 3.7 – Controle do fluxo**

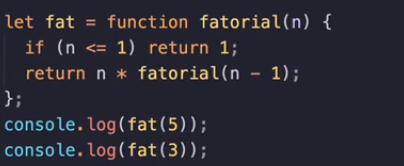
* É a determinação de qual instrução deve ser executada a seguir.
* Estrutura IF:
* Estrutura IF/ELSE:
* Expressão usando?:
* Estrutura if/else/elseif
*  Estrutura switch
* Estrutura while
* Estrutura do/while
* Estrutura for
* Instrução continue e break:

**Aula 3.8 – Funções**

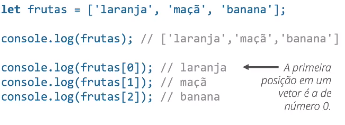
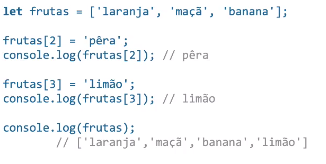
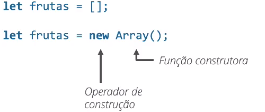
* Mecanismo de controle para facilitar o entendimento do código e separa-los.
* Conjunto de comandos.
* Não está no fluxo normal do código e sim quando for chamado.
* Como declarar:
  + **Function nome\_funcao(){CODIGO}**

**Nome\_funcao();**

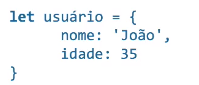
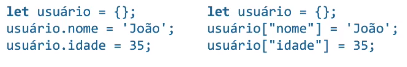
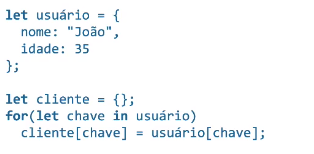
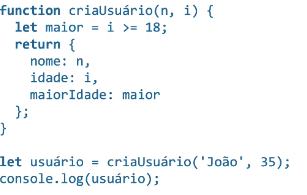
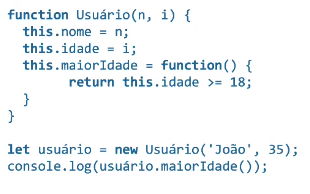
* Parametros:
* Valor de retorno:
* Variavel criada na função só existira dentro dela.
* Uma variável pode conter uma função 🡪JS.
* Possibilidade de passar uma função como parâmetro para uma outra função.
* Arrow functions:
* Expressao de função guardada em uma variável(div), o qual é parâmetro para outro função(executa)
* Recursividade usando funções: 🡪 precisa ser nomeada



**Aula 3.9 – Vetores**

* Vetores permitem armazenar um conjunto de valores a variáveis.
* Possibilidade de usar a sequencia toda ou cada elemento.
* Pode conter dados de diferentes tipos.
* EX:
* ****Vetores podem ser alterados da seguinte forma:
* Tamanho do vetor:
  + Usa-se o comando lenght
  + Ex:
  + Para adicionar certo 🡪 frutas[frutas.length] = ‘x’;
* ****Vetores vazios:
* Métodos:
  + .sort() 🡪 Age sobre o vetor, ordenar dados sobre o vetor.
  + .pop()🡪 Remoção do fim do vetor, e retorno o mesmo.
  + .push() 🡪 Acrescenta no fim do vetor.
  + .shift()🡪 Remoção do inicio do vetor, e retorno o mesmo.
  + .unshift()🡪Acresenta no ínicio do vetor.

**Aula 3.10 – Objetos**

* São estruturas similares ao vetores , porém são identificados por meio de um nome.
* Uso de chaves{}
* Exemplos:
* Como declarar:
  + Feita por uma literal de objeto.
  + EX:
  + É possivel declarar um objeto vazio, por meio de {} ou uma funcao construtora.
* ****Acrescimo de propriedades:
* Operador **in** permite nos testar se uma propriedade existe no objeto.
* Quando copiamos um objeto, estamos apenas criando uma nova referencia e não um novo objeto.
* ****Quando precisarmos de uma cópia é necessário fazer uma clonagem.
* Função para criar um objeto:
* ****Operador **this** 🡪 referencia a propriedade do objeto.
* ****Função Construtora:

**Aula 4.1 – Janelas modais**

* Todo código JS fica entre elementos <script> no cabeçalho.
* Eventos **onload 🡪** Quando terminar de carregar a página.
  + Ex <body onload = “função()”>
* Janelas modais travam a interação, e permitem uma interação com o usuário.
* Tipos de janela
  + Alert.
  + Prompt.
  + Confirm

**Aula 4.2 – Eventos**

* Onload 🡪 Quando termina de carregar todo conteúdo
* Para visualizar todos eventos ir em MDN events reference.
* Eventos são acontecimentos que irá executar um código, assim misturando JS, CSS e JS.
* Principais:
  + Onclick . EX: 
    - Não precisa ser necessariamente em um botão.
  + Onload
    - Window.onload 🡪 Janela toda.

**Eventos de Mouse**

* click – disparado quando o mouse é clicado em um elemento da página
* mouseover – disparado quando o mouse entre sobre a área de um elemento
* mouseout – disparado quando o mouse deixa a área de um elemento
* mousemove – disparado quando o mouse é movimentado
* contextmenu – disparado quando o botão direito do mouse é clicado
* mousedown – disparado quando o botão do mouse é pressionado
* mouseup – disparado quando o botão do mouse é solto
* Eventos de formulário e seus campo.
* submit – disparado quando um formulário é enviado
* focus – disparado quando um campo do formulário é ativado
* blur – disparado quando o usuário deixa um campo do formulário
* change – disparado quando o valor de um campo é alterado
* select – disparado quando é feita uma escolha em uma caixa de seleção

**Eventos de teclado**

* keydown – disparado quando uma tecla é pressionada
* keyup – disparado quando a tecla é liberada

**Eventos de janela**

* load – disparado quando a página termina de ser carregada
* unload – disparado quando o usuário deixa a página ou fecha o navegador
* rezise – disparado quando a janela do navegador é redimensionada
* Inserçao Js : 

**Aula 4.3 – Objetos do navegador**

* Window 🡪 representa TODAS informações do usuário e janela.
  + Tudo declarado fará parte do objeto window.
  + Let🡪variável local
  + Var 🡪variável global
* Navigator 🡪representar o navegador em si.
  + Navigator.userAgent = qual aparelho você está usando.
* Document 🡪Documento em si. Interatividade com usuário.

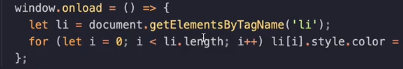
**Aula 4.4 – Document Object Model (DOM)**

* Recursos para manipular o documento.
* API = Application Programing Interface
* Tudo disponível nos objetos vem do DOM.
* O código fonte não será alterado a partir de interações, e sim será criado um documento alterado.
* Alterando o texto pelo Document:

OBS:

1. Document.getElemntbyId
2. Innertext é difenrete de InnerHTML

* Alterando a formatação:
* Alterando usando getelmentbytag:



Obs

1. Retornar um vetor, então tem que trabalhar como se fosse um vetor.

* **Document.querySelectorAll** 🡪 Utiliza a linguagem CSS.

**Aula 4.5 – Formulários**

* Uso do elemento <form> action = login.php e metohd post🡪 será enviado para esse arquivo.
* Construção de um formulário em HTML:
* A maioria dos comandos ficará dentro da função window.onload().
* Eventos dentro de formulários:
  + Change 🡪onchange🡪Quando houver uma mudança.
  + Input 🡪oninput🡪Qualquer **MÍNIMA** alteração gera um evento.
  + Focus 🡪onfocus 🡪Quando receber o foco, ou seja, for selecionado.
  + Blur 🡪onblur🡪Quando sair do foco.
  + Submit 🡪onsubmit 🡪 Quando submit for clicado.
* É possível trabalhar com campos sem estar usando o formulário.
* Caixas de seleção 🡪 devolve on ou não aparece
* Uso de **nome.checked** e uso de **nome.value**.
* Servico para interpretação de dados = postman-echo.com
* Métodos 🡪 **POST E GET**
* ****Exemplo validação campos/formulários:
* PreventDefault() 🡪bloqueia ação do evento.
* Post 🡪 Não aparece na URL.
* Get🡪Aparece na URL.

**Aula 4.6 - O objeto evento**

* Picsum.photos – biblioteca de fotos.
* Tratamento de eventos:

let trataEvento = (evento) => {

console.log(`${evento.type} em ${evento.target.id}`);

};

* forEach 🡪Passa por todos elementos do vetor.

**Aula 5.1 – JSON**

* JSON 🡪 JavaScript Object Notation
* Troca de informações na internet via objetos JS.
* Todas informações tem que ser **strings.**
* Não aceita funções.
* Para converter tudo em string 🡪**JSON.stringify**
* Para voltar de uma string para objeto 🡪**JSON.parse**

**Aula 5.2 – Armazenamento Local**

* Para armazenar:
  + Window.localStorage 🡪 Permanente
  + Window.sessionStorage 🡪Temporario
* Metodos
  + SetItem(chave,valor)
  + GetItem(chave)
  + RemoveItem(chave)
  + Clear

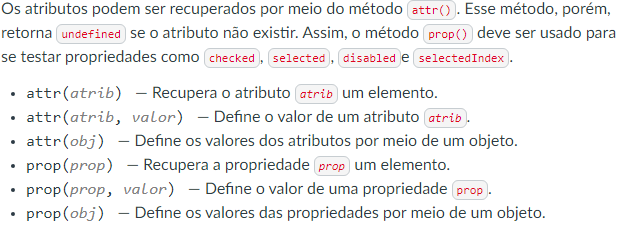
**Aula 6.1 Biblioteca jQuery**

* Biblioteca JS , que auxilia as operações mais comuns
* Versionamento Semântico
  + Tres números 2.0.0
  + 2🡪Grande mudança que causa incompatibilidade
  + 0🡪Mudança na API porem não causa incompatibilidade
  + 0🡪patch , pequena mudança para corrigir bugs.
* Para incorporar:
* Usa-se $ para selecionar algo na página = Doccumentselect.
* Window.onload = . ready
* As principais funcionalidades da jQuery estão organizadas em 8 grupos:
  + Núcleo - Funções do núcleo da jQuery, além de alguns utilitários mais usados;
  + Seleção e percurso - Funções para localização de conteúdo e para navegação entre eles;
  + HTML e CSS - Funções para edição e alteração de conteúdo e de formatação;
  + Eventos - Funções para gerenciamento de eventos de interface;
  + Efeitos - Funções para animações básicas;
  + AJAX - Funções para acesso assíncrono a dados e para manipulação de objetos JSON;
  + Interface - Plugin para controles avançados de interface com o usuário;
  + Extensibilidade - Base para a construção de plugins.

**Aula 6.2 Seletores jQuery**

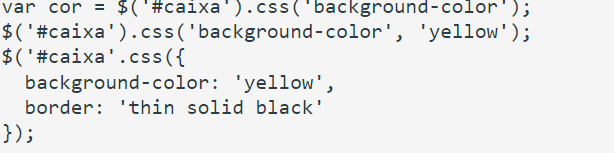
* Tudo começa a partir de uma seleção da página.
* **$(‘SELETOR CSS’)**
* Alterando propriedades css : 
* <https://api.jquery.com/category/selectors/>

**Aula 6.3 Manipulação do conteúdo com JQuerry**

* Metodos:
  + html() — Recupera o conteúdo HTML de um elemento.
  + html(valor) — Define o conteúdo HTML de um elemento.
  + text() — Recupera o conteúdo textual de um elemento (ignorando as marcações, isto é, os filhos).
  + text(valor) — Define o conteúdo textual de um elemento. Como ignora as marcações, qualquer tag será convertida em caracteres (ex.: '<b>' à '&lt;b&gt;').
  + val() — Retorna o valor do campo de formulário.
  + val(valor) — Estabelece o valor para o campo de formulário.
* Exemplo de contador usando JQuerry:
* .append($(CSS)) 🡪adicionar um elemento na página via JS.
* Metodos:
  + prepend() — Insere conteúdo como o primeiro filho do elemento.
  + append() — Insere conteúdo como o último filho do elemento.
  + before() — Insere conteúdo antes do elemento (como irmão).
  + after() — Insere conteúdo após o elemento (como irmão).
  + prependTo() — Insere conteúdo como o primeiro filho do elemento indicado.
  + appendTo() — Insere conteúdo como o último filho do elemento indicado.
  + insertBefore() — Insere conteúdo antes do elemento indicado (como irmão).
  + insertAfter() — Insere conteúdo após o elemento indicado (como irmão)
* Prop 🡪Representa o documento
* Attr🡪Representa o elemento do código fonte

**Aula 6.4 - Manipulação da formatação com jQuery**

* Ex:



* addClass() — Adiciona uma classe ao elemento.
* hasClass() — Testa se o elemento possui uma determinada classe.
* removeClass() — Remove uma classe do elemento.
* toggleClass() — Adiciona a classe se o elemento não a tiver ou retira a classe se o elemento já a tiver.

**Aula 6.5 - Filtros das seleções jQuery**

* filtrar conjuntode seleções:
  + filter() — Reduz o conjunto para aqueles que atendem a um seletor ou passem em uma função de teste.
  + is() — Testa se pelo menos um objeto atende ao seletor e, nesse caso, retorna verdadeiro.
  + not() — Elimina do conjunto os elementos que se enquadrem no segundo seletor.
  + has() — Reduz o conjunto àquele elementos que possuam um determinado descendente.
  + eq() — Reduz o conjunto para apenas um elemento do índice especificado.
  + first() — Reduz o conjunto ao seu primeiro elemento.
  + last() — Reduz o conjunto ao seu último elemento.
  + slice() — Reduz o conjunto a um subconjunto especificado por uma faixa de índices.
  + map() — Transforma o conjunto em um outro objeto contendo novos valores.
* Usamos o método get() para retornar o resultado como um objeto JavaScript comum.
* Add() 🡪 Para ir juntando filtragens

**Aula 6.6 - Tratamento de eventos com jQuery**

EVENTOS DE MOUSE

* click() — Clique do mouse sobre o elemento.
* dblclick() — Duplo clique do mouse sobre o elemento.
* mousedown() — Botão do mouse é pressionado sobre o elemento.
* mouseup() — Botão do mouse é solto sobre o elemento.
* mouseover() — Cursor do mouse entra na área do elemento, excluindo a área dos descendentes.
* mouseenter() — Cursor do mouse entra na área do elemento, incluindo a área dos descendentes.
* mouseout() — Cursor do mouse sai da área do elemento, inclusive quando indo para a área dos descendentes.
* mouseleave() — Cursor do mouse sai da área do elemento, independentemente da área dos descendentes.
* mousemove() — Cursor do mouse se move sobre o elemento.
* hover(fn1,fn2) — Combinação dos eventos mouseenter e mouseleave.
* contextmenu() — Clique com o botão direito do mouse.

EVENTOS DE TECLADO

* keypress() — Tecla do teclado foi pressionada e solta com o elemento em foco.
* keydown() — Tecla do teclado foi pressionada com o elemento em foco. Inclui teclas como Shift, Esc e Control.
* keyup() — Tecla do teclado foi solta com o elemento em foco.

EVENTOS DE FORMULÁRIO

* focus() — O campo do formulário (ou link) recebeu o foco.
* blur() — O campo do formulário (ou link) perdeu o foco.
* focusin() — O elemento ou algum descendente dele recebeu o foco.
* focusout() — O elemento ou algum descendente dele perdeu o foco.
* change() — O valor do campo input, textarea ou select foi alterado.
* select() — O usuário fez alguma seleção de texto em um campo input do tipo texto ou em um campo textarea.
* submit() — O usuário tentou enviar o formulário.

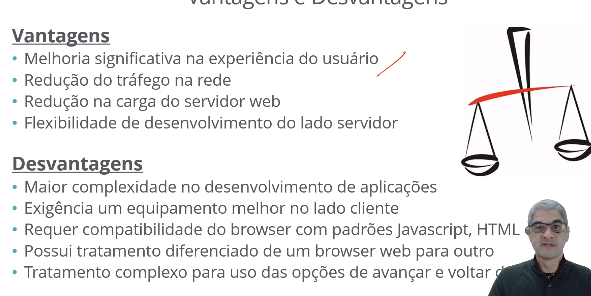
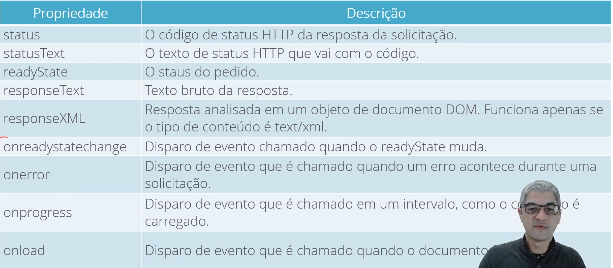
EVENTOS DE JANELA

* resize() — A janela do navegador foi redimensionada.
* scroll() — A janela do navegador ou algum elemento com a propriedade overflow definida para scroll, foi rolado.

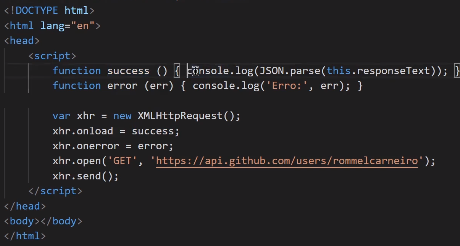
EVENTOS DE DOCUMENTO

* load() — O documento HTML (ou outro recurso específico) foi carregado.
* ready() — Toda a página foi carregada (HTML e recursos).
* unload() — O usuário saiu da página (navegando outra página ou, em alguns navegadores, fechando a janela).

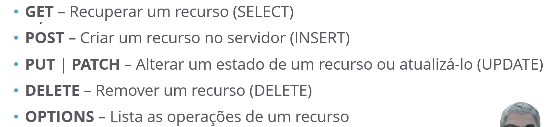
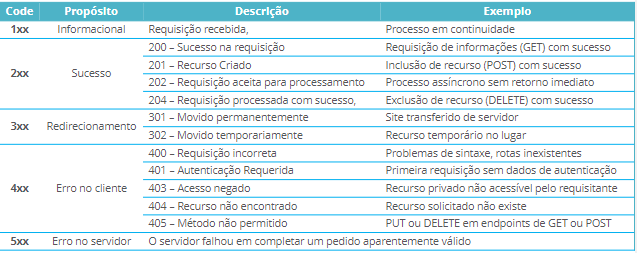
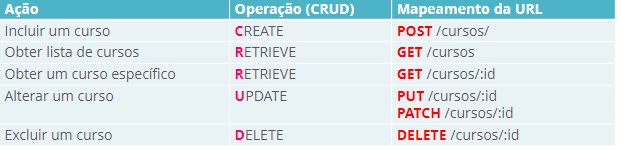
**Aula 7.1 Introdução ao AJAX**

* AJAX 🡪 Asynchronus Javascript and XML
  + Combinação de varias tecnologias
  + Padrões 🡪XHTML e CSS
  + Exibicao dinâmica atrasves do DOM
  + Recuperação assincrônica de dados com XMLHttpRequest
* Gmail
  + Navegação entre pastas e tags
  + Carga automática de mensagens
  + Salvamento automático
  + Sugestão de contatos
* Método clássico
  + Usuario entra com uma requisição e retorna uma pagina , depois mais uma e assim por diante
  + Logica toda no servidor
* Método AJAX
  + URL 🡪primeira requisição
  + Grande parte de código JS e faz requisições menores.
  + Diversas requisições para uma página
  + Logica dividida entre servidor e cliente
  + AJAX esta na base de SPA.
* Vantagens e Desvantagens:
* API XMLHttpRequest
  + Criado pela Microsoft
  + Base pra programação em AJAX
  + Troca de dados em vários formatos
  + Propriedades:
* API Fetch
  + Mais nova
* Cross-Origin Resource Sharing(CORS)
  + Browser corta a conexão com sites que estão saindo e indo para outro servidor, chamadas que não são seguras.

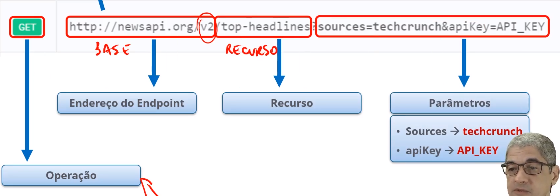
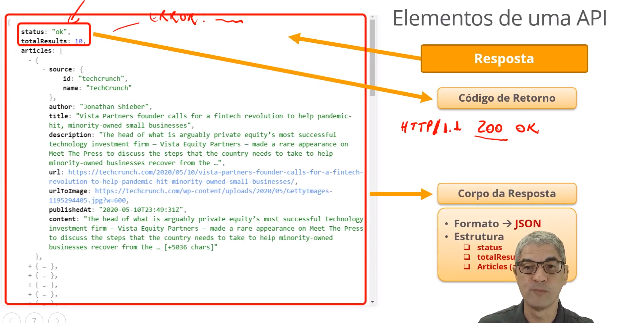
**Aula 7.2 API e Web Services**

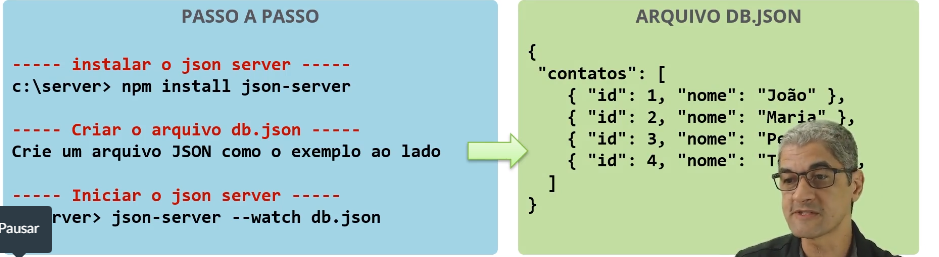
* API:
  + Application Programing Interface
  + É uma interface de uma aplicação que é voltada a outras aplicações, integrando sistemas.
  + Conversa de aplicações
  + Tipos:
    - Web API ou Web Services
    - API Legadas
* Web Services
  + Sistema de software projetado para suportar comunicação maquina a máquina através de uma rede.
  + Usam protocolo HTTP.
  + ****Exemplo de uso da api do git hub:
  + Podem ser utilizados com ambientes protegidos com firwaal sem abertura de portas adicionais.
  + Padrões W3C
  + Permite compatibilidade entre diversas plataformas e linguagens
  + Baixo acoplamento das aplicações.

**Aula 7.3 - RESTful API**

* Transferência de Estado Representacional
* Tudo que o servidor troca é uma representação do dado.
* RESTful 🡪uso rigoroso das regras
* RESTlike 🡪uso parcial das regras
* Prima pela simplicidade e eficiência 🡪 separação do cliente e servidor web
* Termos:
  + Recurso 🡪
    - O alvo conceitual de uma referencia de hipertexto.
    - Entidades expostas por um sistema web como usuários, posts, fotos, localidades...
    - Possuem um identificador baseado no padrão URI
  + Identificador do recurso 🡪URL,URN
  + Representação 🡪Documento HTML, Imagem JPEG
* Principios importantes de desing de uma API RESTful:
  + **Uso das definições do protocolo HTTP**
    - Utiliza métodos HTTP em função semântica
    - Uso do código de status:
  + **Estrutura uniforme de Endpoint**
    - Entrada da API
    - Crud de recursos
    - URL utilizada
    - EX:
  + Uso da abordagem stateless (sem estado)
  + **Uso da abordagem HATEOAS**
    - Hipertexto como mecanismo de estado da aplicação.
  + Controle do versionamento da API
  + Controle adequado do cache
  + Documentação clara e adequada da API

**Aula 7.4 - Usando APIs RESTful**

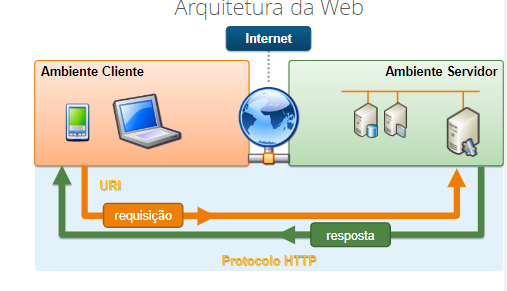
* Elementos de uma API
  + Agente envia uma requisição para a API.
  + Primeiro ponto 🡪Segurança
  + Segundo ponto 🡪Endpoint
  + Requisição:
    - Endereço do Endpoint
    - Operação
    - Recurso
    - Parâmetro
    - EX:
  + Reposta:
    - Codigo de Retorno e Corpo da Reposta
    - EX:

**Aula 7.5 - API RESTful com JSON Server**

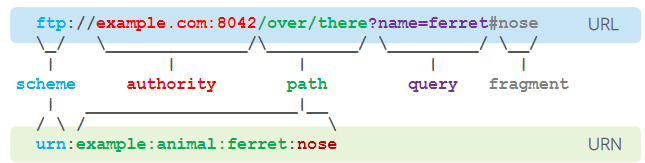
**Aula 8.1 – Introdução à arquitetura da Web**

* Arquitetura da Web
  + Ambiente cliente se conecta com o ambiente servidor 🡪envia requisições através de uma URI e recebe uma reposta do servidor.
  + Cliente web 🡪 é um programa ou aplicação específica, na maioria das vezes um Navegador ou Browser, também conhecido como agente de usuário (user-agent) que envia requisições via protocolo HTTP(S) a uma outra aplicação, o Servidor Web através de uma rede de computadores como a Internet.
  + Servidor web 🡪É um software que recebe requicoes e envia respostas.
  + Endereço ip 🡪identificador de cada máquina(host). (DNS traduz nome para número)
  + Sistema cliente-servidor utiliza um protocolo (linguagem o qual se comunica) HTTP.

**Aula 8.2 - Servidores Web**

* Diagrama arquitetura da web:
* Ambiente Servidor:
  + Servidor web 🡪 Papel fundamental para comunicacao entre o ambiente servidor e usuario.
  + Servidores de aplicacoes:
    - Processar os programas de backend
    - Devolve esses recursos para o servidor web, que envia para o usuario.
    - Podem requisitar um banco de dados ou API
* Funções do servidor web:
  + Atender requisioes HTTP e processar respostas.
  + Gerenciar múltiplos sites
  + Gerenciar arquivos dos sites
  + Integrar mecanismos de scripts
  + Autenticar usuários.
  + Implementar a criptografia na comunicação.
  + Cache de recursos
  + Auditoria.
* Servidores Web Locais:
  + XAMPP
  + IIS
* Hospedagem de Sites na Internet:
  + Servidores de nuvem:
    - Microsoft Azure
    - Heroku
    - Amazon AWS
    - Google
  + Editores Online: 🡪Edicao colaborativa + web server
    - Repl.it
    - CodeSandBox
    - Glitch
  + Outras opções:
    - GitHubPages

**Aula 8.3 URI, URL e URN**

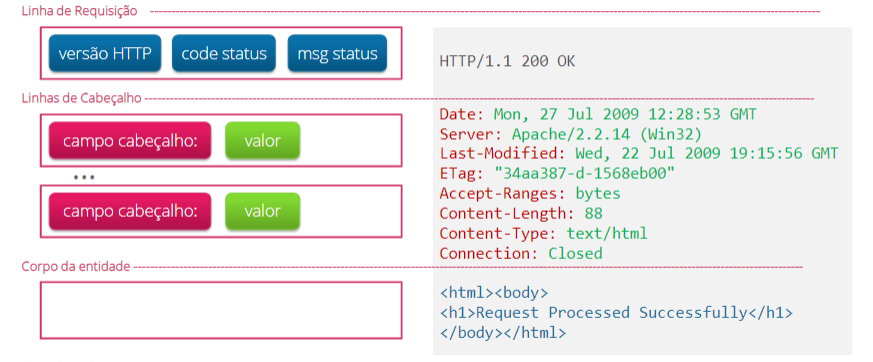
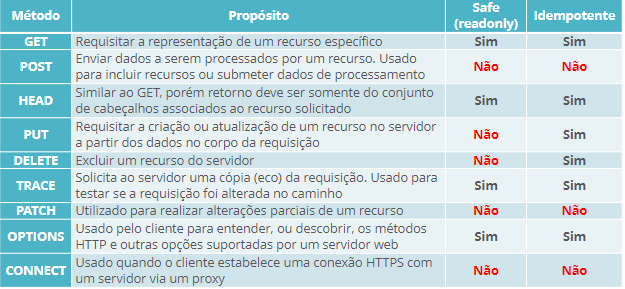
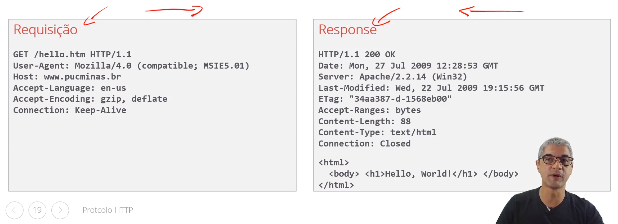
* Significado:
  + URI – Uniform Resource Identifier
  + URL – Uniform Resource Locator
  + URN – Uniform Resource Name
* URI (Uniform Resource Identifier) é um padrão para o endereçamento de recursos disponíveis na rede que engloba os conceitos de URL (Uniform Resource Locator) e URN (Uniform Resource Name)
* Recurso 🡪 Identifica qualquer coisa na internet
* URL:
  + Padrao de Uri que serve para referenciar um recurso e sua localização
  + Esquema 🡪protocolo (HTTP)
  + Hostname 🡪servidor
  + Para informar usuário e senha usa-se @
  + Porta 🡪 qual software será necessário 🡪Quando a porta é 80 não precisa ser informada
  + Caminho ou Path 🡪 Onde o recurso se encontra
  + Querry 🡪 Inicio = ? Identifica o recurso especifico dentro do path.

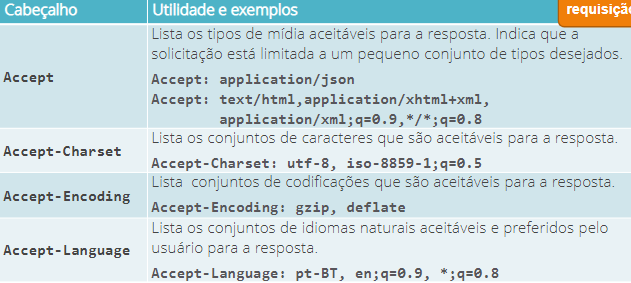
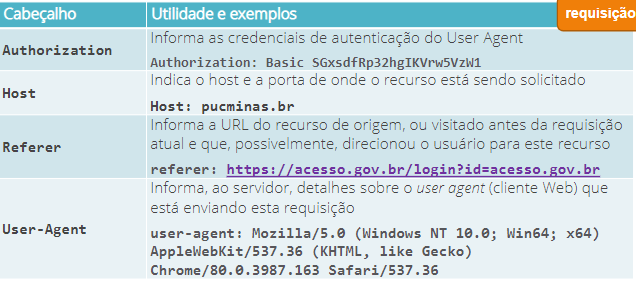
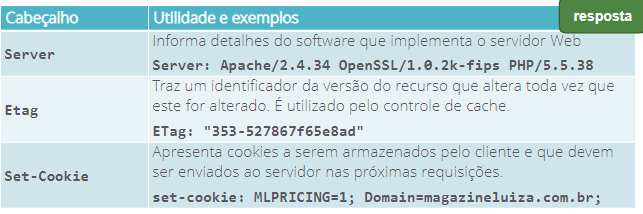
Exemplo google:

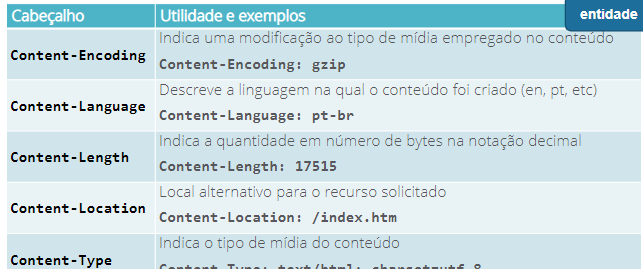
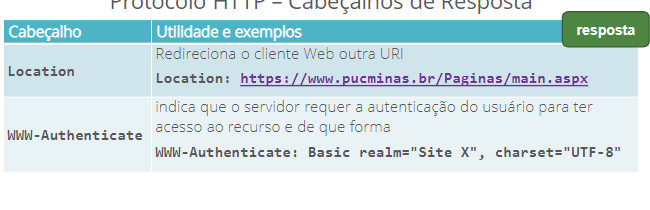
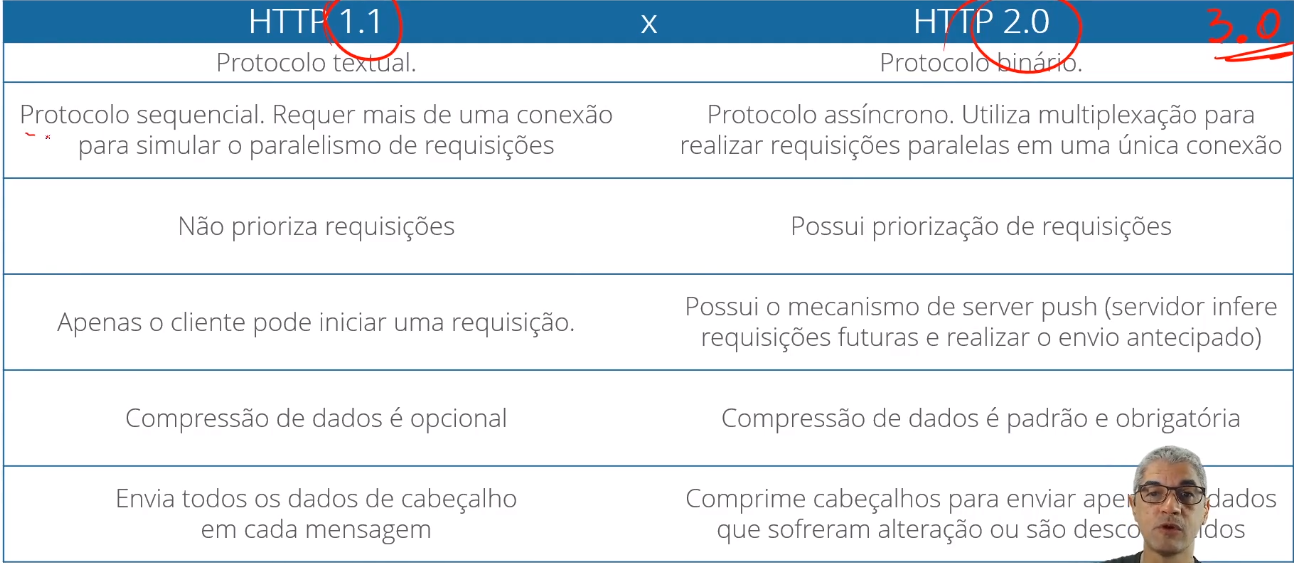
* Detalhando elementos de uma URL:
* Esquema – identifica a forma de interação entre um cliente e um servidor, como User:pass – informações de usuário
  + Host – nome ou número IP onde se encontra a aplicação servidor
  + Porta – identifica a porta TCP/IP associada ao servidor. A porta padrão do HTTP (80) pode ser omitida
  + Caminho – indica o local exato onde o recurso se encontra
  + Query – dados não hierárquicos, detalhando a consulta normalmente sob a forma de pares nome e valor
  + Fragmento – identifica uma seção no recurso
* URN
  + Só identifica o nome e não a localização
  + NID 🡪Namespace Identifier
  + NSS 🡪Namespace Specific String

**Aula 8.4 - Protocolo HTTP**

HyperText Transfer Protocol(HTTP) é um protocolo da camalida de aplicação de sistemas distribuídos e colaborativos no formato de hipertextos

* + Porta 80 🡪 padrão
  + Protocolo não guarda estado do cliente(stateless)
  + Utiliza TCP.
  + Continua em evolução.
  + Cliente 🡪Cria uma requisição para solicitar algum serviço.
  + Srvidor 🡪Compreende a solicitação e monta uma resposta.
  + Linguagem utilizada entre cliente e servidor
* **Requisições e Respostas**
  + Requisição:
  + Resposta:
  + Codigos:
* **Métodos:**
  + GET
    - Requisitar um recurso especifico do servidor web.
    - Não altera o estado do servidor(safe)
    - Pode ser mantidos em cache
    - Envia dados ao servidor via parâmetros de querry.
    - Tem restrições quanto ao tamanho e ao formato.
    - Ex
  + POST
    - Enviar dados para o servidor para ser processado
    - Alteram o servidor – not safe
    - Podem enviar dados via query ou corpo da requisição
    - Não possuem restrições de tamanho e tipo.
  + HEAD
    - Similar ao GET, porem o seridor deve enviar apenas o conjunto de cabeçalhos.
  + PUT
    - Similar ao POST, porém atualiza o recurso do servidor.
  + DELETE
    - Solicita a exclusão de dados.
  + TRACE
    - Usado para ecoar o conteúdo de uma requisição HTTP ao servidor. Usado para verificar se a requisição é alterada no caminho por agentes intermediários (servidores de cache ou proxy).
  + OPTIONS
    - Usado quando o cliente quer saber os tipos de métodos que existe.
  + CONNECT
    - Utilizado na conexão com o protocolo seguro 🡪 HTTPS.
* **Cabeçalhos**
  + Ocorrem após a primeira linha.
  + Tipos:
    - Request header: informações sobre a requisição feita ou sobre o cliente Web.
    - Response header: informações sobre a resposta encaminhada ou sobre o servidor Web.
    - Entity header: informações sobre o conteúdo da entidade trocada como tamanho e tipo.
    - General header: Usado tanto em requisições quanto em respostas.



* Novas versões:
  + HTTP 3 – UDP