Universidade Federal
de Mato Grosso
Campus Rondonópolis

Programação em Ambiente Web I

Aula 7 – HTML5 (2)

Prof. Msc Cleyton Slaviero

cslaviero@gmail.com

Agradecimentos ao Prof. Jivago Medeiros (UFMT-Cba)
Fabiano Taguchi (UFMT-Roo)

HTML5 – API's

- APIs de storage:
  - **Web Storage,** (**localStorage** e **sessionStorage**)
  - Indexed Database (Indexed DB) API
  - Web SQL Database (Web SQL) (Abandonado pela W3C)
- Offline Web (Application Cache) *(Work in Progress)*
- **Canvas**
- **Navigator**
  - **Geolocation API**
- WebSockets

Universidade Federal
de Mato Grosso
Campus Rondonópolis

WebStorage

- Porque armazenar algo?
  - Funcionamento offline
  - Armazenar dados frequentemente usados
- Antigamente... Cookies
  - Problemas
    - Segurança (acesso por subdomínios)
    - Tamanho do cookie (4kb*20 = 80kb)
- A API de *Web Storage* do HTML5 inclui os objetos ***sessionStorage*** e ***localStorage*** que fornecem aos sites / sistemas de um determinado domínio um espaço gerenciado pelo navegador para o armazenamento de dados.
  - **Importante:** ao contrário dos *cookies*, os dados da API de Web Storage não são enviados ao servidor, ao menos que o desenvolvedor faça uma requisição enviando os dados.

sessionStorage e localStorage

- ***sessionStorage***: armazena os dados no navegador durante uma sessão, quando a sessão é encerrada, geralmente com o fechamento do navegador, os dados são apagados.
- ***localStorage***: ao contrário do sessionStorage, os dados são armazenados "permanentemente".

Importante 1: Apesar dos dados não serem enviados ao servidor, o usuário tem acesso aos dados e pode modificá-los ou mesmo apagá-los.

Importante 2: Os navegadores impõem e gerenciam um limite de armazenamento (por domínio), geralmente é de pelo menos 5Mb, no firefox são 10Mb.

Universidade Federal de Mato Grosso
Campus Rondonópolis

Utilização

- Tanto o objeto ***sessionStorage*** quanto o objeto ***localStorage*** possuem os mesmo métodos principais para armazenar, retornar e excluir dados, sendo eles:
- ***`setItem(chave,valor)`***
  - armazena o valor no navegador tendo como índice para acesso a chave passada
- ***`getItem(chave)`***
  - retorna o valor armazenado referente a chave passada como parâmetro
- ***`removeItem(chave)`***
  - remove do armazenamento o item referente a chave

Universidade Federal de Mato Grosso
Campus Rondonópolis

Exemplo

```
localStorage.setItem("professor",”Cleyton Slaviero");
sessionStorage.setItem("aluno",”Fulano de Tal");
console.log("Professor -> "+localStorage.getItem("professor"));
console.log("Aluno -> "+sessionStorage.getItem("aluno"));
```

Canvas

- A especificação do HTML5 inclui a *tag <canvas>*
  - ”Tela" para que, via linguagem de *script* (geralmente JavaScript), sejam feitos desenhos, edições de imagens, animações e até jogos eletrônicos

Utilização

- Para utilizarmos o canvas, primeiramente devemos incluir o elemento em nosso documento HTML, ex:

```
<canvas id="tela"
style="border: 1px solid black; width: 350px; height:
350px;">
</canvas>
```

- Canvas tem apenas dois atributos: width e height
  - Se não iniciados, o tamanho será de 300x150 pixels
- É possível redimensionar via CSS, mas durante a renderização a imagem é escalonada para caber no tamanho do layout

Utilização

- É possível adicionar borda, margem, fundo, etc
  - Isso não afeta o desenho do canvas
  - Caso nada seja informado, o canvas começa transparente
- É possível adicionar uma informação adicional (texto ou código html), para o caso do browser não suportar canvas

Utilização

- <canvas> cria uma superfície de desenho de tamanho fixo, que expõe um ou mais contextos de renderização
  - Contextos podem ser 2D ou 3D

Utilização

- Devemos "pegar o contexto" do canvas e a partir dele, por exemplo, desenhar objetos:

```
var canvas = document.getElementById("tela");
var ctx = canvas.getContext("2d");
ctx.fillStyle = "rgb(6,125,43)";
ctx.fillRect (10, 10, 155, 50);
ctx.fillStyle = "rgba(255, 255, 0, 0.7)";
ctx.fillRect (30, 30, 155, 50);
```

Nesse exemplo desenharemos na nossa tela dois retângulos “preenchidos”, um sobre o outro, sendo o segundo com uma pequena transparência

Utilização

- E se o browser não suporta?
  - Podemos verificar getContext do canvas
    - O que fazemos é verificar se a função existe para aquele elemento
    - Caso exista, é sinal que o browser sabe renderizar aquele elemento, e ele retorna a própria função, o que num if equivale a ”true”

Utilização

- Diferente do SVG que vimos anteriormente, o <canvas> suporta somente formas primitivas: retângulos.
  - Outras formas partem da combinação de um ou mais caminhos (paths)
- Exemplo:
  - Desenha um triângulo →

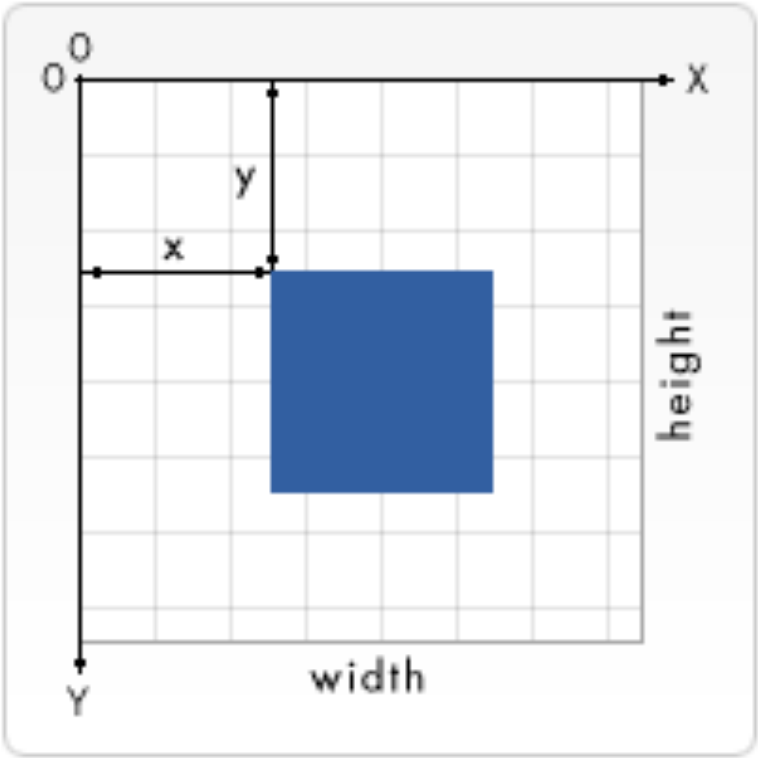
```
function draw() {
        var canvas = document.getElementById('canvas');
        if (canvas.getContext){
                var ctx = canvas.getContext('2d');
                ctx.beginPath();
                ctx.moveTo(75,50);
                ctx.lineTo(100,75);
                ctx.lineTo(100,25);
                ctx.fill();
        }
}
```

Utilização

- Também é possível
  - Desenhar linhas
  - Desenhar arcos
  - Desenahr curvas quadráticas/Bezier
  - Mover o ponto de escrita

Funções para desenho (de retângulos)

- fillRect(x,y,width,height)
  - Desenha um retângulo preenchido (*filled rectangle*)
- *strokeRect(x,y,width,height)*
  - Desenha um retângulo não preenchido (somente a borda)
- clearRect(x,y,widht,height)
  - Limpa a área definida, deixando-a transparente
- O ponto de origem pode ser modificado, se necessário



Canvas

- Mais informações sobre canvas
  - Tutorial de Canvas da MDN

http://developer.mozilla.org/pt-BR/docs/Web/Guide/HTML/Canvas_tutorial

Objeto navigator

- Representa o estado e identidade do ”user-agent”
- Fornece uma série de informações, tais como
  - ***navigator.product*** engine de renderização
  - ***navigator.language*** idioma do navegador
  - ***navigator.onLine*** status da conexão com a internet
  - ***navigator.oscpu*** SO no qual está rodando o navegador (ex: Linux e Windows)
  - ***navigator.userAgent*** string com informações do navegador, ex: versão , plataforma, entre outras. Essa informação é incluída no cabeçalho das requisições HTTP
  - ...

Geolocation

- `navigator.geolocation`
  - Fornece informações sobre localização
    - **latitude** e **longitude**
- Podemos usar essas informações para criar websites/webapps com resultados personalizados pela localização do usuário

Geolocation

- É possível testar a existência dos serviços de geolocalização

```
if ("geolocation" in navigator)
        { /* geolocation está disponível */ }
 else
        { /* geolocation NÃO está disponível */ }
```

Utilizando geolocation

- Para retornamos a localização (latitude e longitude) do usuário por meio do objeto `geolocation` devemos utilizar o `método` ***`getCurrentPosition()`***, que possui a seguinte sintaxe:

(opcional)
Vetor com opções:
- enableHighAccuracy
- timeout
- maximumAge

Função a ser executada caso a localização seja retornada com **sucesso**

```
navigator.geolocation.getCurrentPosition(funcaoSucesso[,funcaoErro[,opcoes]]);
```

Função a ser executada caso haja **erro** retornando a localização (opcional)

Exemplo

```
navigator.geolocation.getCurrentPosition(
        function(pos) {
                console.log("Lat: "+pos.coords.latitude+”/"+"Long:"+pos.coords.longitude);
});
```

Importante:

1.Ao invés de passarmos o nome de uma função para o método ***getCurrentPosition()***, nós utilizamos uma função anônima.

2.É passado como parâmetro para a função de ***callback*** em caso de sucesso o objeto **position**, que nos permite retornar as coordenadas

Exemplo

- Mas se o usuário está se locomovendo?
  - `getCurrentPosition()` não é suficiente
  - Utiliza-se o método `watchPosition()`

```
var watchID =  navigator.geolocation.watchPosition(funcaoSucesso[,funcaoErro[,opcoes]]);
```

  - Esse método irá fazer um 'callback' toda vez que a posição do usuário mudar
  - Como parar?
    - O método retorna um valor que identifica o "watcher"
      - Podemos usar o método `clearWatch(watchID)` para parar de chamar a função de callback

Informações recuperadas

- coords.latitude
- coords.longitude
- coords.altitude (metros)
- coords.accuracy (metros)
- coords.altitudeAccuracy (metros)
- coords.heading (norte verdadeiro)
- coords.speed (m/s)

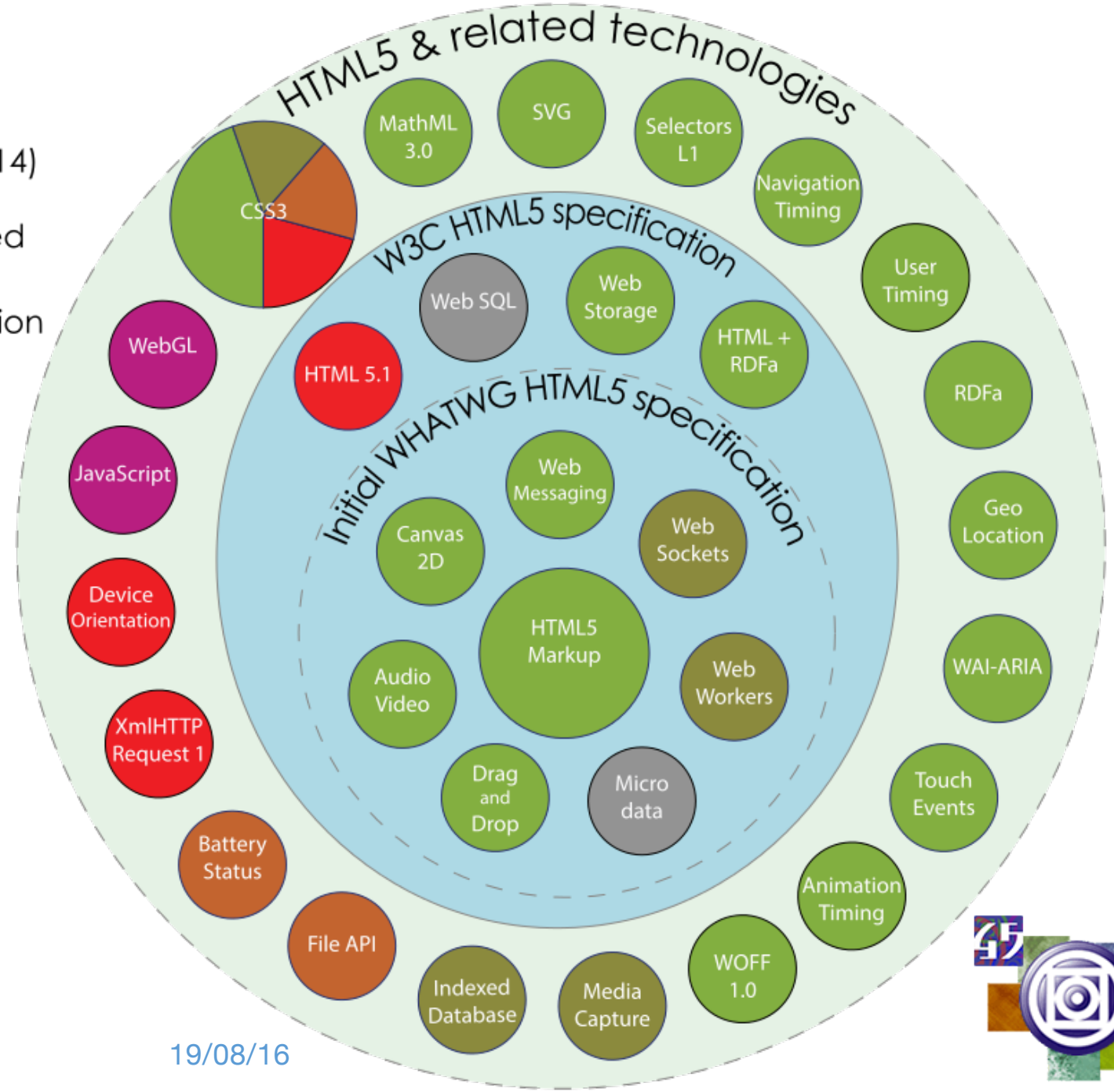
Dicas finais

- Como utilizar essas informações?
  - Há na internet algumas ferramentas que retornam o nome do país, estado, cidade, e até mesmo bairro, rua, CEP, etc, a partir da latitude e longitude fornecidas
  - Mais conhecido: Google Maps API:
    - Recuperar informações de endereço (json): http://maps.googleapis.com/maps/api/geocode/json?latlng=-16.464601,-54.578470
    - Recuperar uma imagem estática do endereço: https://maps.googleapis.com/maps/api/staticmap?center=-16.464601,-54.578470&zoom=18&size=300x300&sensor=false
    - Outras informações
- Mais informações
  - https://diveintohtml5.com.br/geolocation.html

HTML5

Taxonomy & Status (October 2014)

- Recommendation/Proposed
- Candidate Recommendation
- Last Call
- Working Draft
- Non-W3C Specifications
- Deprecated or inactive



Universidade Federal de Mato Grosso
Campus Rondonópolis