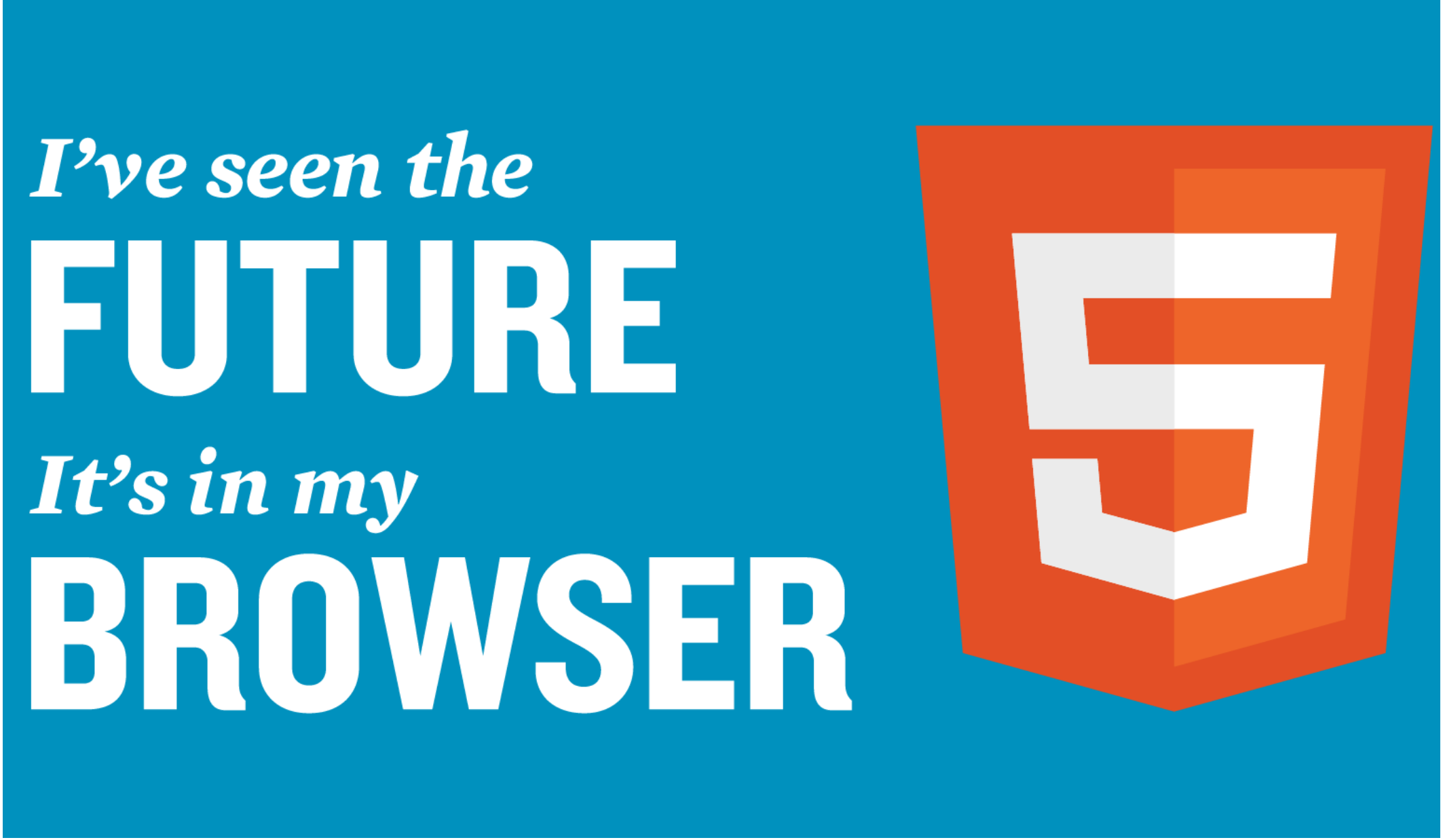


HTML5

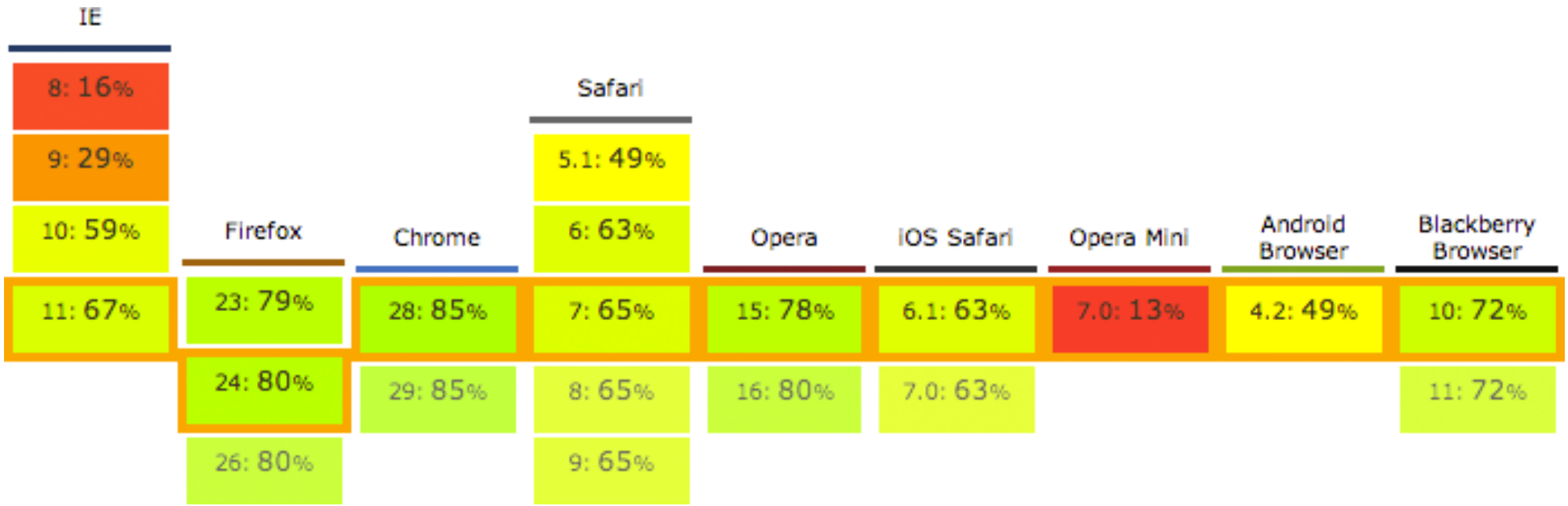
Aula 3

Programa Aula 3

- Novas APIs de JavaScript;
- Aceder ao DOM;
- Geolocation;
- Canvas;
- Web Storage;

Novas APIs de JavaScript

Visão geral de JavaScript



Visão geral - Novas APIs

- Contacts
  - Aceder aos contactos;
- Web Workers
  - Permite inicializar um *worker*, para processamento independente da UI;
- Web Sockets
  - Transmissão de mensagens entre o cliente e o servidor;
- Server Sent Events
  - Permite ao servidor enviar mensagens diretamente para a página;

Visão geral - Novas APIs

- XmlHttpRequest2
  - Funcionalidades avançadas de XHR;
- Micro data
  - Permite embeber informação semântica na página, de forma a ser lida por *máquinas*;
- Media API
  - Acesso aos mecanismos de media do dispositivo (microfone, camera, etc);
- Web Messaging
  - Permite a partilha de dados entre diversos documentos (*iframes* p.e.);

Visão geral - Novas APIs

- Forms
  - Programação dos novos tipos de inputs;
- File API
  - Forma segura de manipular ficheiros no browser;
- WebGL
  - OpenGL no browser - contexto 3D do canvas;

APIs que vamos ver hoje

- Selection
  - Acesso ao DOM e manipulação do mesmo;
- Geolocation
  - Acesso à localização do dispositivo;
- Canvas 2D
  - Manipulação de gráficos em contexto 2D;
- Web Storage
  - Persistir dados no cliente;

APIs da próxima aula

- IndexedDB
  - Base de dados do lado do cliente;
- Drag and Drop
  - Drag and drop nativo;
- App Cache
  - Permite utilizar aplicações em offline;
- Media API (?)
  - Acesso aos mecanismos de media do dispositivo (microfone, camera, etc);

Selection - Aceder ao DOM

Selection - Revisão

- getElementById(*id*)
  - Devolve o primeiro elemento com o id passado por parâmetro;
- getElementsByClassName(class)
  - Devolve uma lista de elementos que tenham a class passada por parâmetro;
- getElementsByTagName(tag)
  - Devolve uma lista de elementos que sejam do tipo paassado por parâmetro;

Selection - traversing properties

- childNodes
  - Devolve todos os filhos desse elemento;
- nextSibling
  - Devolve o elemento imediatamente a seguir no DOM;
- parentElement
  - Devolve o elemento pai;

Selection - Novos métodos

- querySelector(selector)
  - Devolve o primeiro elemento que valida o selector CSS passado por parâmetro;

- querySelectorAll(selector)
  - Devolve todos os elementos que validem o selector CSS passado por parâmetro;

Ambos podem ser usados no *document*, ou num elemento em particular

Selection - Novos métodos - Exemplo

- document.querySelector(‘header’)
  - Devolve o primeiro elemento <*header*> da página;
- document.querySelectorAll(‘.simple-item’)
  - Devolve todos os elementos que tenham a class *simple-item*;
- element.querySelectorAll(‘a’)
  - Devolve todos os elementos <*a*> que sejam filhos de *element*;

Selection - Novos métodos - Exercício

- Pesquisar por todas as *div*'s da página;
- Pesquisar pela *nav* que está no *header*;
- Pesquisar por todos os *a* que estão no *footer*;
- Pesquisar pelos *a* que estão no *footer*, dentro de um *li* que contenha a class *footer-item*;
- Pesquisar pelo primeiro li do footer;

Base: http://bit.ly/VC_HTML5_Base21 Exercício 21

Geolocation

Geolocation - Contexto

- Permite determinar a localização do dispositivo;
- Dispara um evento sempre que a posição muda;
- O utilizador tem que autorizar a disponibilização da sua posição geográfica;

Geolocation - Suporte

- Internet Explorar 9+;
- Chrome;
- Firefox;
- Safari;
- Opera;
- iOS & Android;

Geolocation - Como funciona

- Através de satélites, caso o dispositivo tenha GPS;
- Através das redes wi-fi:
  - Google Location Services;
  - Skyhook Wireless,
  - Outros serviços semelhantes;

Geolocation - Verificar disponibilidade

```
if (“geolocation” in navigator) {

    /* geolocalização disponível */

} else {

    /* geolocalização não disponível */

}
```

Base: http://bit.ly/VC_HTML5_Base22 Demo 22

Geolocation - Obter posição actual

```
navigator.geolocation.getCurrentPosition(function (position) {

    // position.coords.latitude

    // position.coords.longitude

});
```

Base: http://bit.ly/VC_HTML5_Base23 Exercício 23

Geolocation - Alterações da localização

```
var watcher = navigator.geolocation.watchPosition(function
(position) {

    // position.coords.latitude

    // position.coords.longitude

});

navigator.geolocation.clearWatch(watcher);  // parar escuta
```

Geolocation - Exercício watchPosition

- Escutar posição do dispositivo;
- Actualizar latitude e longitude;
- Actualizar número de updates;
- Limpar *watch*;

Base: http://bit.ly/VC_HTML5_Base24 Exercício 24

Dica: *melhores* resultados em dispositivos móveis;

Geolocation - Demo mapas

Ver localização no mapa

Demo: http://bit.ly/VC_HTML5_Demo25

Geolocation - A reter

- API simples de usar;
- getCurrentPosition - posição actual;
- watchPosition - escutar alterações na posição;
- clearWatch - parar escuta;
- GPS consome bastante bateria;
- É possível obter localização via Wi-Fi.

Canvas

Canvas - O que é

- Permite desenhar elementos gráficos através de JavaScript;
- Permite animar os elementos gráficos através de JavaScript;
- Tem um contexto 2D e outro 3D (WebGL);

Canvas - Como usar

<canvas id="my-canvas"></canvas>

Aceder ao canvas

```
var canvas = document.getElementById('my-canvas'),
    contexto = canvas.getContext('2d');
```

Demo 26

Canvas - Linhas e estilos

- beginPath()
  - inicializar nova linha
- moveTo(x, y)
  - mover lápis para ponto inicial
- lineTo(x, y)
  - definir ponto final da linha
- closePath()
  - fechar linha
- stroke()
  - criar linha
- fill()
  - preencher linha (caso forma seja fechada)
- *lineWidth*
  - definir largura da linha
- *strokeStyle*
  - definir cor da linha
- fillStyle
  - definir cor do recheio

Canvas - Exercício de linhas

- Criar uma linha com início no ponto [10, 10] e fim em [50, 75];

- Criar uma linha com início no ponto [200, 200] e fim em [160, 10], com 10px de largura e cor vermelha;

Base: http://bit.ly/VC_HTML5_Base27 Exercício 27

Canvas - Formas

- strokeRect(origin-x, origin-y, width, height)
  - rectângulo com origem [*origin-x*, *origin-y*], largura *width* e altura *height*;
- fillRect(origin-x, origin-y, width, height)
  - rectângulo preenchido origem [*origin-x*, *origin-y*], largura *width* e altura *height*;
- arc(center-x, center-y, radius, startAngle, endAngle)
  - círculo com centro [*center-x*, *center-y*], radio *radius* e ângulo inicial *startAngle* e desenhado até ao ângulo *endAngle*;

Canvas - Exercícios de formas

- Criar círculo com centro em [150, 150], raio 80, de 0 a 360 (dica: radianos, *Math.PI*);
- Criar rectângulo com origem em [115, 180], largura 70 e altura 12;
- Criar círculo azul com centro em [120, 120], raio 10, de 0 a 360 graus;
- Criar círculo azul com centro em [180, 120], raio 10, de 0 a 360 graus;

Base: http://bit.ly/VC_HTML5_Base28 Exercício 28

Canvas - Animações

- requestAnimFrame
- Permite-nos executar uma determinada função tantas vezes quanto necessário para atingir uma frame rate estável;
- Proporciona animações bastante mais fluídas (60FPS);

Demo 29 setInterval

Demo 29 requestAnimFrame

Download: http://bit.ly/VC_HTML5_CanvasAnim

Web Storage

Web Storage - O que é

- Permite guardar informação dentro do browser de forma pouco estruturada;
- Serve para substituir as cookies, para dados que não precisam de ser enviados para o servidor;
- Funciona com uma relação chave/valor;
- Muito útil para aplicações que queiram manter estado do lado do cliente, especialmente para aplicações que funcionem offline;

Web Storage - Sessão

- Definido pelo objeto *sessionStorage*;
- Mantém dados enquanto a página estiver aberta (numa tab / janela);
- Mantém dados caso a tab / janela seja refrescada (F5);
- É automaticamente removida pelo navegador quando a tab / janela for fechada;

Web Storage - Persistente

- Definido pelo objeto *localStorage*;
- Mantém os dados mesmo que a janela seja fechada;
- Apenas removido programaticamente ou através da acção “Limpar cookies” do navegador (diferentes implementações de navegador para navegador);
- Caso o utilizador abra uma nova janela em modo anónimo para a mesma página, essa janela usa um localStorage novo, removido quando a sessão termina;

Web Storage - Métodos setItem e getItem

.setItem (key, value)

- Guarda o value *value* na chave *key;*
- *Exemplo: localStorage.setItem(“foo”, “bar”);*

.getItem (key)

- Devolve o value guardado na chave *key*;
- Caso a chave não exista, devolve *null*;
- *Exemplo: localStorage.getItem(“foo”);*

Web Storage - Exercício sessionStorage

- Abrir http://bit.ly/VC_HTML5_WebStorage
- Abrir a consola do navegador;
- Guardar o valor *rocks* na chave *html5 em sessionStorage*;
- Refrescar a página;
- Ler o valor da chave *html5*;
- Fechar a janela;
- Voltar a abrir;
- Tentar aceder ao valor da chave *html5*;

Web Storage - Exercício localStorage

- Abrir http://bit.ly/VC_HTML5_WebStorage
- Abrir a consola do navegador;
- Guardar o valor *rocks* na chave *html5 em localStorage*;
- Refrescar a página;
- Ler o valor da chave *html5*;
- Fechar a janela;
- Voltar a abrir;
- Tentar aceder ao valor da chave *html5*;

Web Storage - Métodos removeItem e clear

.removeItem (key)

- Remove a chave *key*;
- *Exemplo: localStorage.removeItem(“foo”);*

.clear ()

- Remove todas as chaves e valores associados;
- *Exemplo: localStorage.clear();*

Web Storage - Exercício removeItem

- Abrir http://bit.ly/VC_HTML5_WebStorage
- Abrir a consola do navegador;
- Tentar aceder ao valor da chave *html5*;
- Apagar o valor da chave *html5*;
- Tentar aceder ao valor da chave *html5*;

Web Storage - Exercício

- Criar um formulário com dois campos de texto (para nome e número de telefone);
- Adicionar um botão para guardar;
- Programar o botão para quando se clicar nele guardar na chave pessoa o número de telefone;
- http://bit.ly/VC_HTML5_Base30

Base: http://bit.ly/VC_HTML5_Base30 Exercício 30

Web Storage - Detectar funcionalidade

```
var hasLocalStorage = (function() {
    try {
        localStorage.setItem(mod, mod);     // sessionStorage
        localStorage.removeItem(mod);       // sessionStorage
        return true;
    } catch(e) {
        return false;
    }
}());
```

Web Storage - A reter

- API de simples acesso;
- Útil para qualquer aplicação (especialmente offline);
- Tem limite de tamanho (2.5MB Chrome, 5MB FF, 10MB IE, iOS Safari 5MB);
- Seguro;