Módulo 11

PROG. WEB 2 - JAVASCRIPT

Programar o comportamento da UI

Client-side scripting

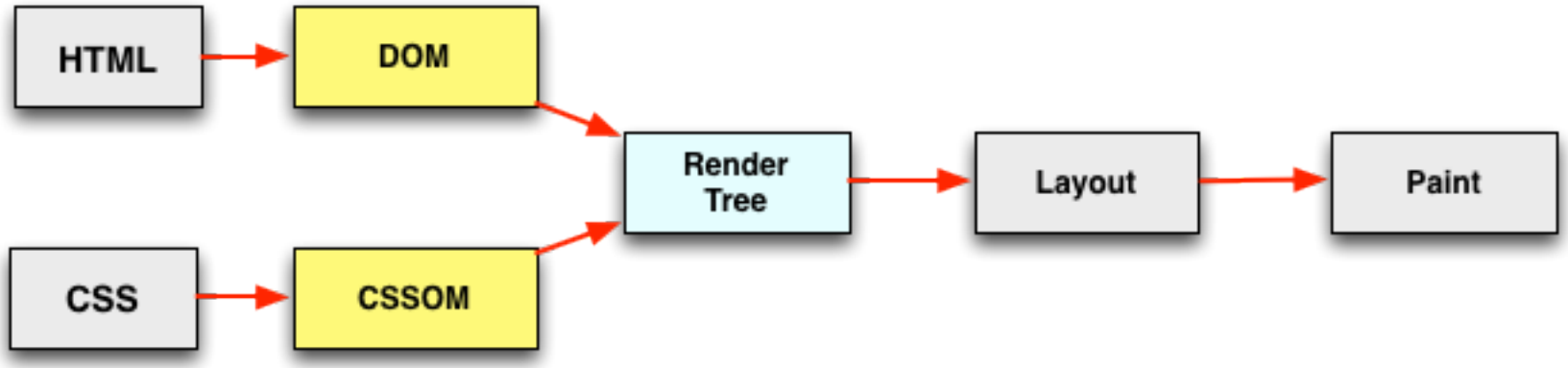
- Código corre no *frontend* (browser)
- Código fonte é visível (ofuscação?!)
- Validação e tratamento de eventos na interface
- Páginas alteradas dinamicamente (interfaces mais reactiva)

Server-side scripting

- Código corre no *backend* (servidor)
- Código fonte não é visível (mais seguro)
- Regras de negócio e acesso a dados
- Páginas criadas dinamicamente

Client-side scripting

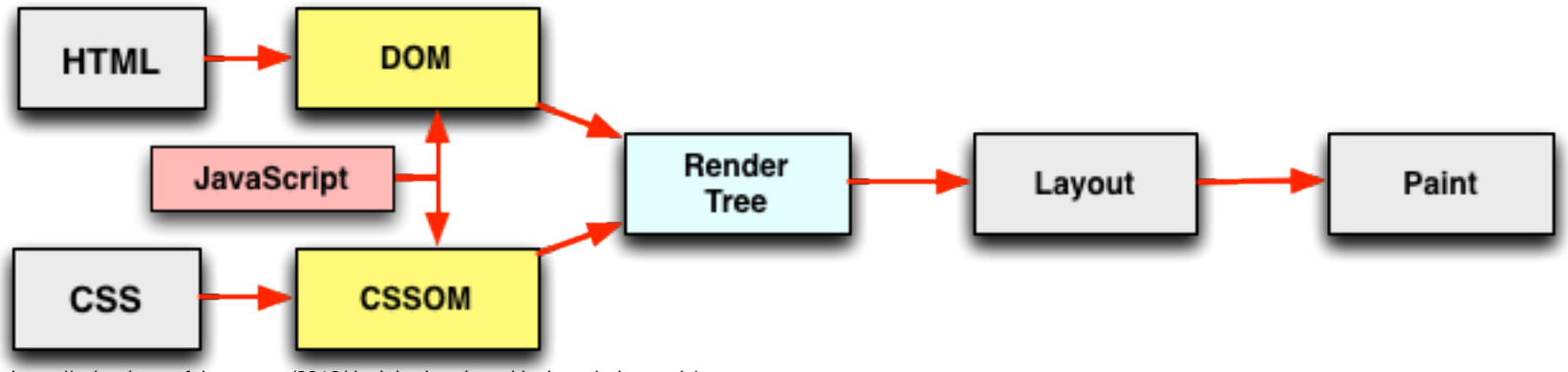
- Conteúdo dos documentos guardado num Document Object Model (**DOM**)
- **Browser** faz o rendering do DOM
  - De acordo com CSS
- DOM manipulado por código **JavaScript**
  - Código JavaScript consegue alterar o conteúdo e visualização do documento



https://calendar.perfplanet.com/2012/deciphering-the-critical-rendering-path/

Client-side scripting

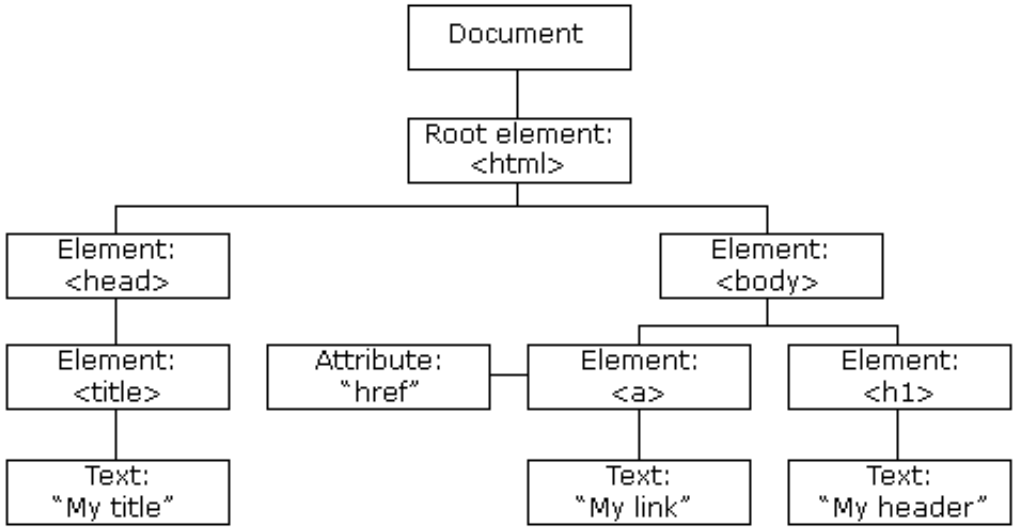
- Conteúdo dos documentos guardado num Document Object Model (**DOM**)
- **Browser** faz o rendering do DOM
  - De acordo com CSS
- DOM manipulado por código **JavaScript**
  - Código JavaScript consegue alterar o conteúdo e visualização do documento



https://calendar.perfplanet.com/2012/deciphering-the-critical-rendering-path/

DOM – Document Object Model

- API para aceder e manipular documentos XML / HTML
  "The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document." (w3schools)
- Representa a estrutura numa árvore



DOM levels

- 3 níveis de DOM:
  - Core DOM
    - standard model for any structured document
  - XML DOM
    - standard model for XML documents
  - HTML DOM
    - standard model for HTML documents

- Cada nível define:
  - objectos
  - propriedades
  - métodos

Core DOM – objecto Node

- O tipo base para todo o DOM
  - `x.nodeName` – the name of x
  - `x.nodeValue` – the value of x
  - `x.parentNode` – the parent node of x
  - `x.childNodes` – the child nodes of x
  - `x.appendChild(node)`
    - insert a child node into x
  - `x.removeChild(node)`
    - remove a child node from x
  - …

http://www.w3.org/TR/DOM-Level-2-Core/core.html#ID-1950641247

HTML DOM – objecto document

- A raíz da árvore que representa o documento
  - `document.getElementsByTagName(name)`
    - get all elements with a specified tag name
  - `document.getElementById(id)`
    - get element with a specified id
  - …

http://www.w3schools.com/jsref/dom_obj_document.asp

Javascript

- alias ECMAScript
- Uma linguagem de Scripting
  - Interpretada pelo browser
- **Não** é Java!!
  - mas sintaxe semelhante
- Nem sequer da Sun
  - Netscape

Javascript – syntax Java

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Exemplo de Javascipt 1</title>
        <script text="text/javascript">
        visitas = new Object();
        function ola(form) {
            var nome = form.nome.value;
            if (visitas[nome]) {
                visitas[nome]++;
            } else {
                visitas[nome] = 1;
            }
            document.getElementById("dados").innerHTML =  "Olá "+nome+".";
            document.getElementById("dados").style.color = visitas[nome]>5?"green":"black";
            form.nome.value = "";
            return false;  // prevent navigation
        }
        </script>
    </head>
    <body>
        <form onsubmit="return ola(this)">
            Nome: <input type="text" name="nome" /><input type="submit" value="OK" />
        </form>
        <p id="dados"></p>
    </body>
</html>
```

Dynamically typed.
var defines function scope
(use let for block scope)

if, while, do/while, for,
switch

event-based
programming model

Event-based programming model

- Browser has control of execution loop
- Application registers event handlers
  - Functions to be executed when specific events are detected by the browser
- Two classes of events
  - Browser events — e.g. a page being loaded, the browser gaining internet connection
  - User actions (related events) — e.g. clicking a button, moving the mouse to an element

Javascript – events

- load / unload — o browser terminou de carregar a página/saiu da página

```
<body onLoad="setup()">
```

- focus / blur / change — um elemento ganhou / perdeu foco / foi alterado

```
<input type="text" name="nome" onChange="upperCase()"/>
```

- submit — um formulário foi submetido

```
<form onSubmit="return ola(this)">
```

- mouseover / mouseout — o rato entra / sai de um elemento

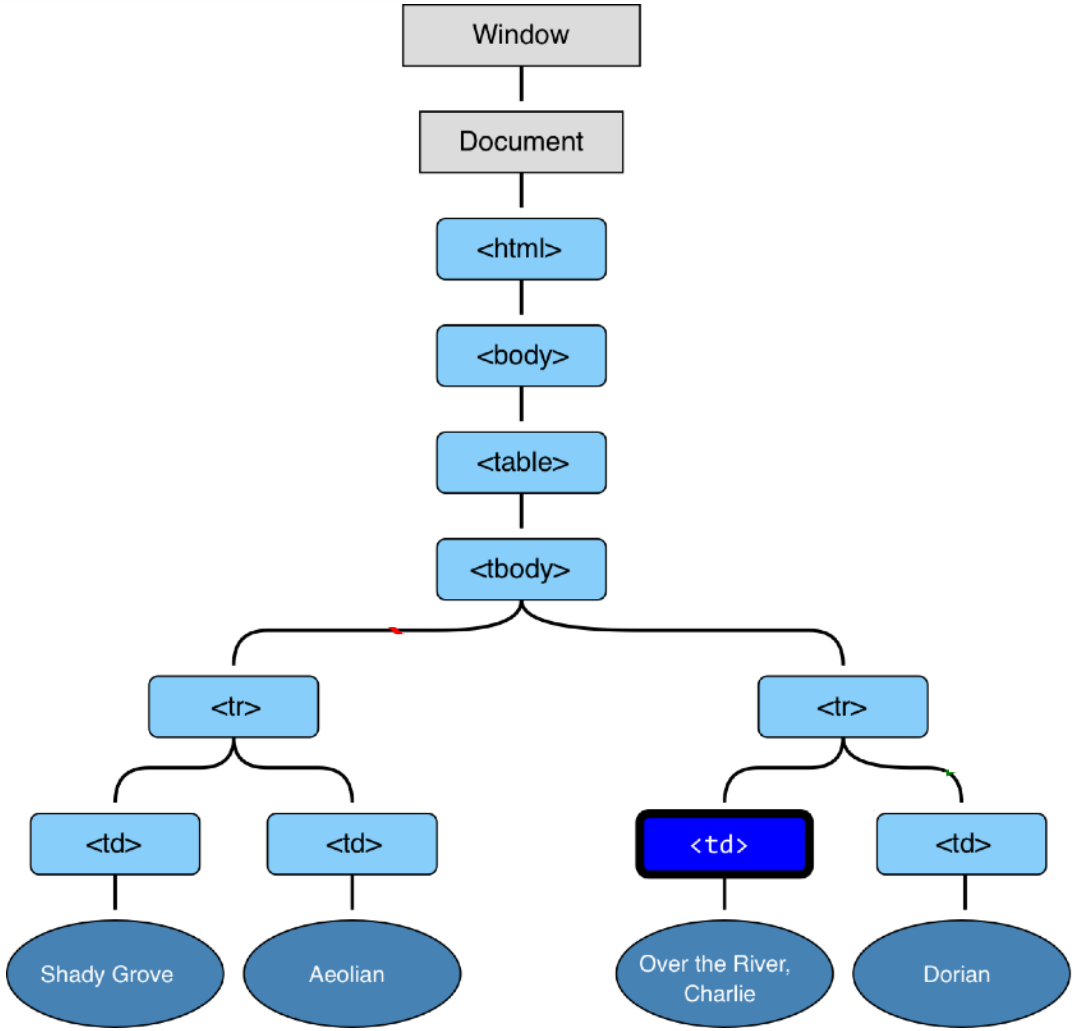
```
<input type="submit" value="OK" onMouseOver="…'" onMouseOut="…" />
```

- click — o elemento é clicado

```
<div onClick="alert('DIV clicado!')">
    <em>Clica-me!<em>
</div>
```

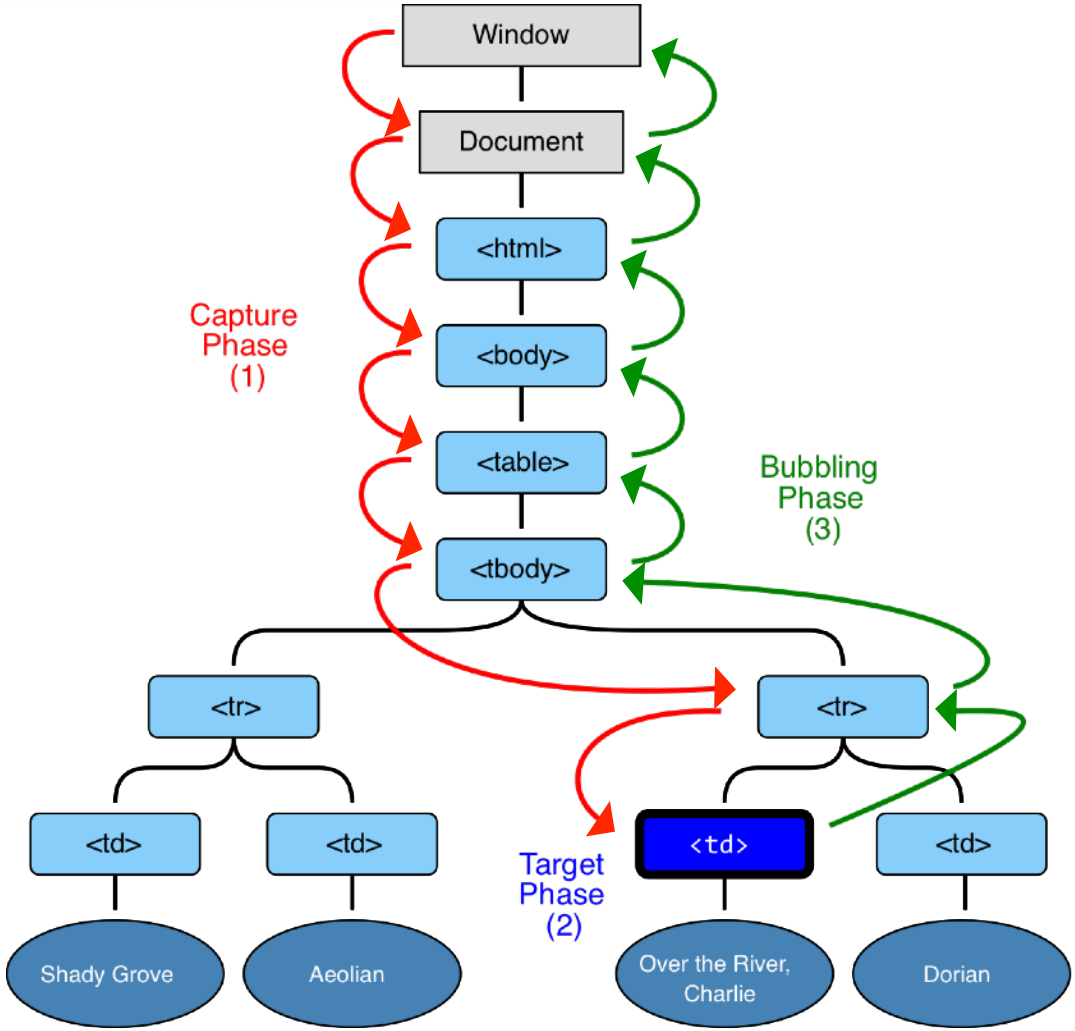
- …

Events' processing phases



https://javascript.info/bubbling-and-capturing

Events' processing phases



https://javascript.info/bubbling-and-capturing

Javascript - classes

```
class Game {
     //constructor
    constructor() {
        this._name = "Oxenfree";
        this._year = 2017;
    }

    //getter for _nome
    get name() {
        return this._name;
    }

    //setter for _nome
    set name(v) {
        this._name = v;
    }

    //example method
    toString() {
         return "Games(" + this._name + ", " + this._year + ")";
    }
}
```

Properties implicitly defined

getters and setters
No enforced encapsulation.
Use of "_" for private properties is only a convention (# prefix will become the norm).

JQuery


```
document.getElementById("dados").innerHTML =  "Olá "+nome+".";
document.getElementById("dados").style.color = visitas[nome]>5?"green":"black";
```

```
$("#dados").html("Olá "+nome+".").css("color", visitas[nome]>5?"green":"black");
```

- Ver tutorial…

JSON

- JavaScript Object Notation

- Ver tutorial…

https://code.google.com/p/google-gson/