netlink
digital solutions

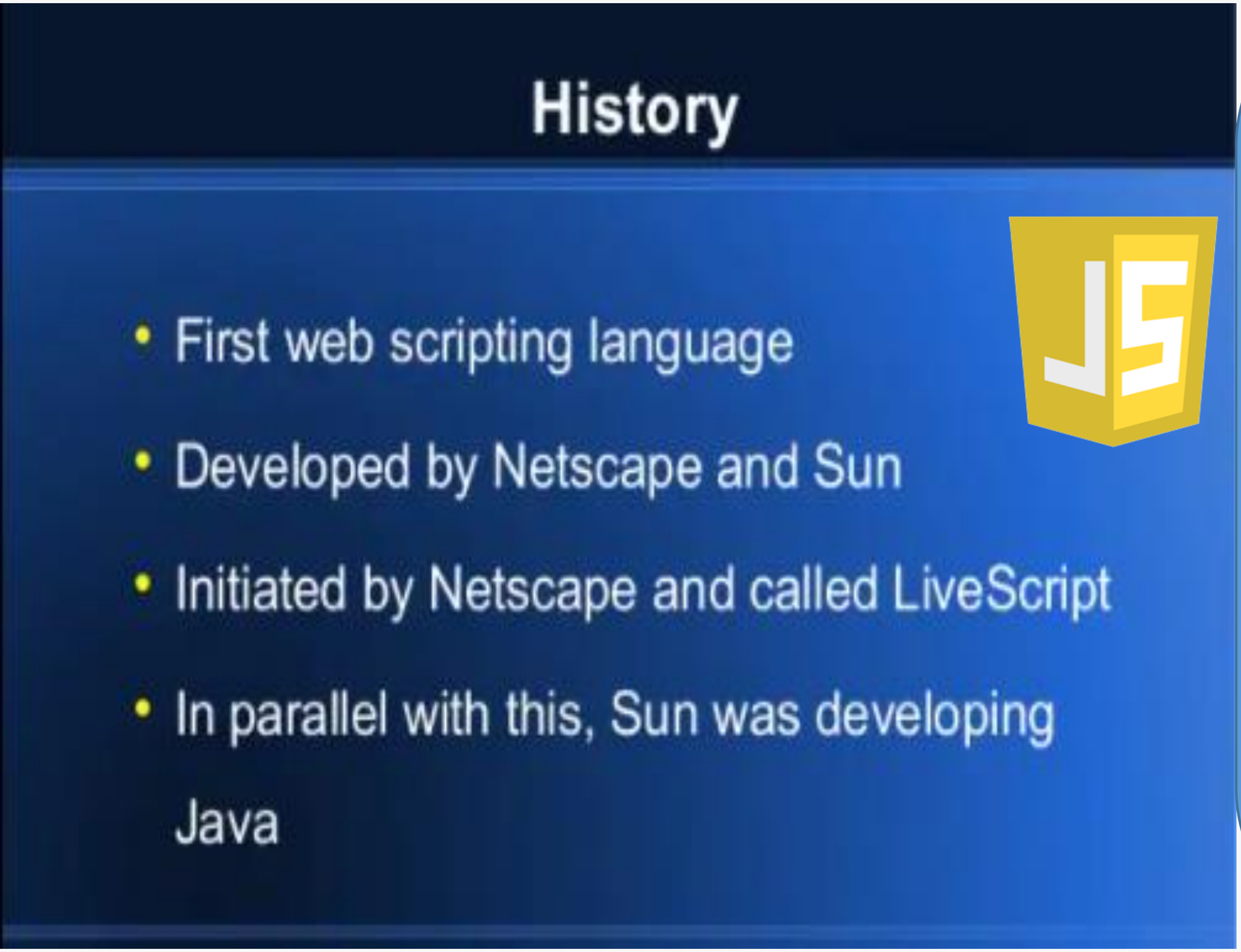
www.netlink.com

JavaScript

Topics to be Covered:

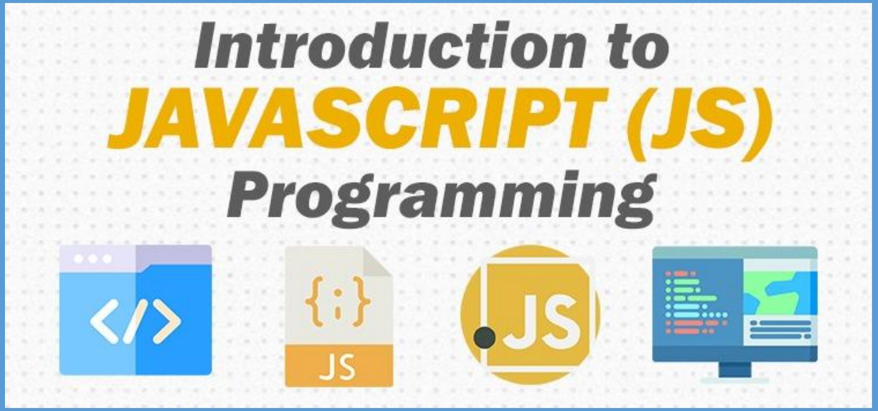
- Introduction of JavaScript
- Variables & Data Type
- Type Coercion
- Variable Mutation
- Operators

JavaScript



History

- First web scripting language
- Developed by Netscape and Sun
- Initiated by Netscape and called LiveScript
- In parallel with this, Sun was developing Java



JavaScript & Java:-

Completely different types of languages that just happen to be similarly named

- ❑ JavaScript - programs are interpreted in the browser
- ❑ Java - programs are compiled and can be run as stand-alone applications

JavaScript

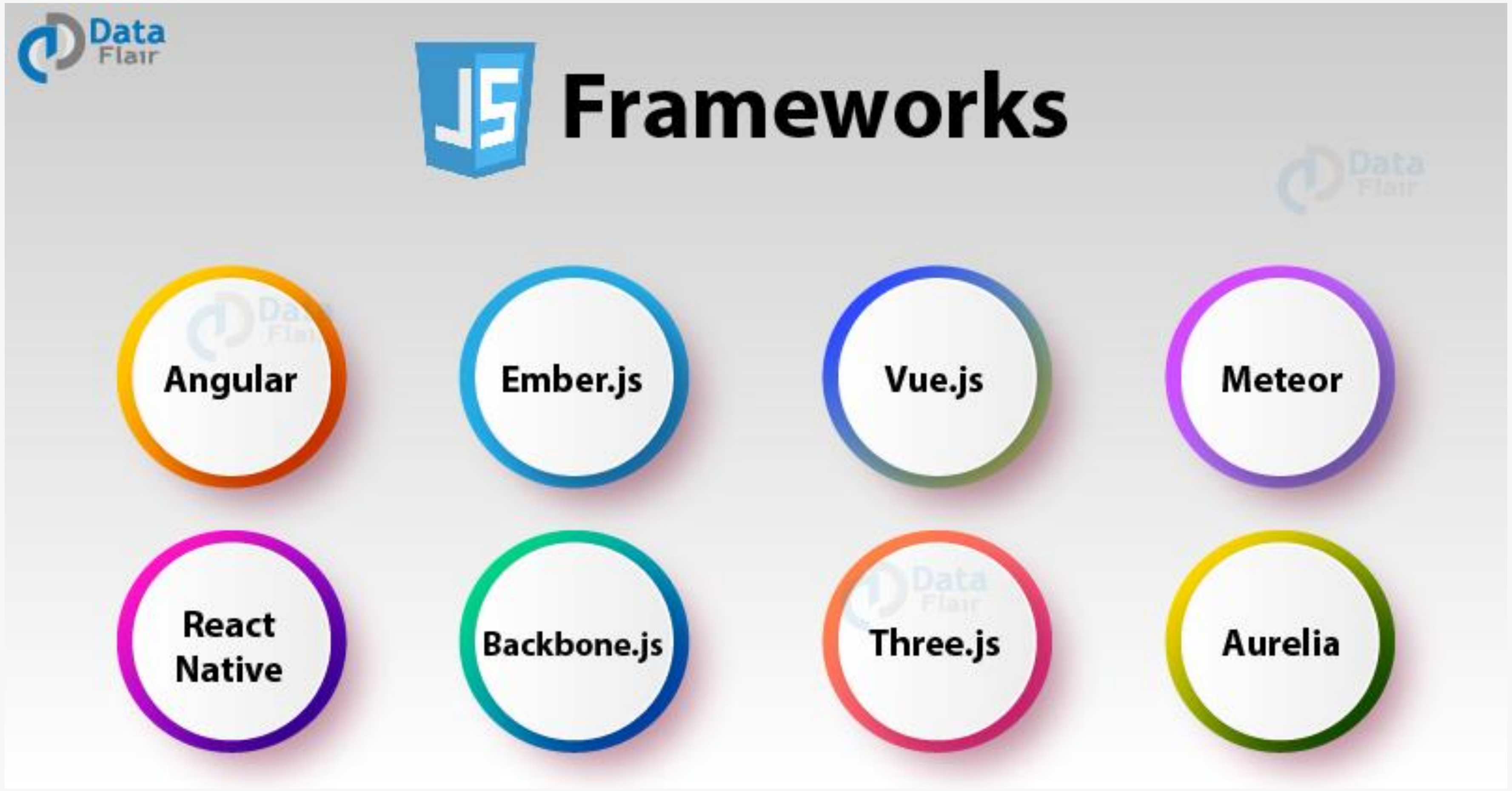
It is the **client-side (browser side)** script-based language that is used to implement business logic, validation, animation, and dynamic design view in web application.

- It is a programming language, used to create functionality in the Web page.

- **Functionality:** Reading inputs, performing process and providing output.
  Ex: Displaying menu when the user clicks on the
  “menu icon”.

- As it is a programming language**, it provides all the programming concepts such as variable, data types, control statements, operators, arrays, functions etc.**

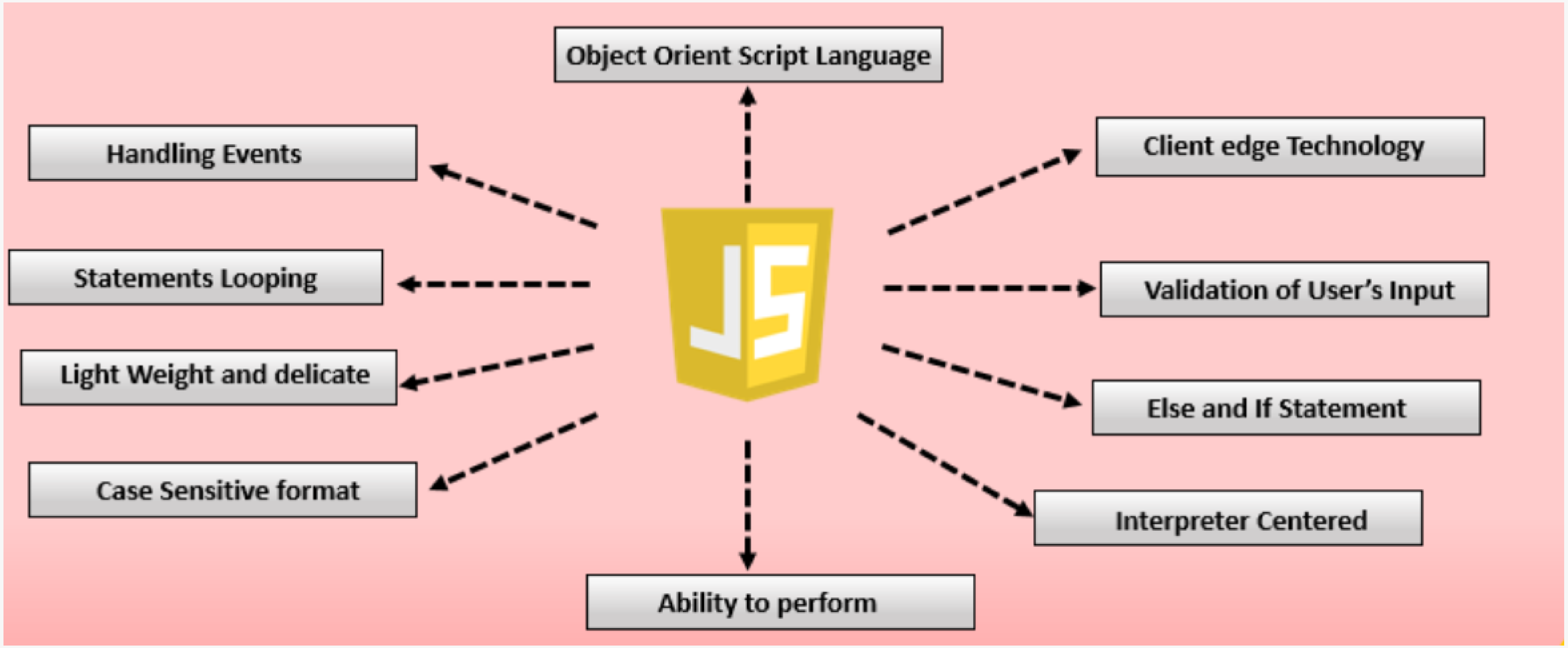
JavaScript


Data Flair
JS Frameworks
Angular
Ember.js
Vue.js
Meteor
React Native
Backbone.js
Three.js
Aurelia

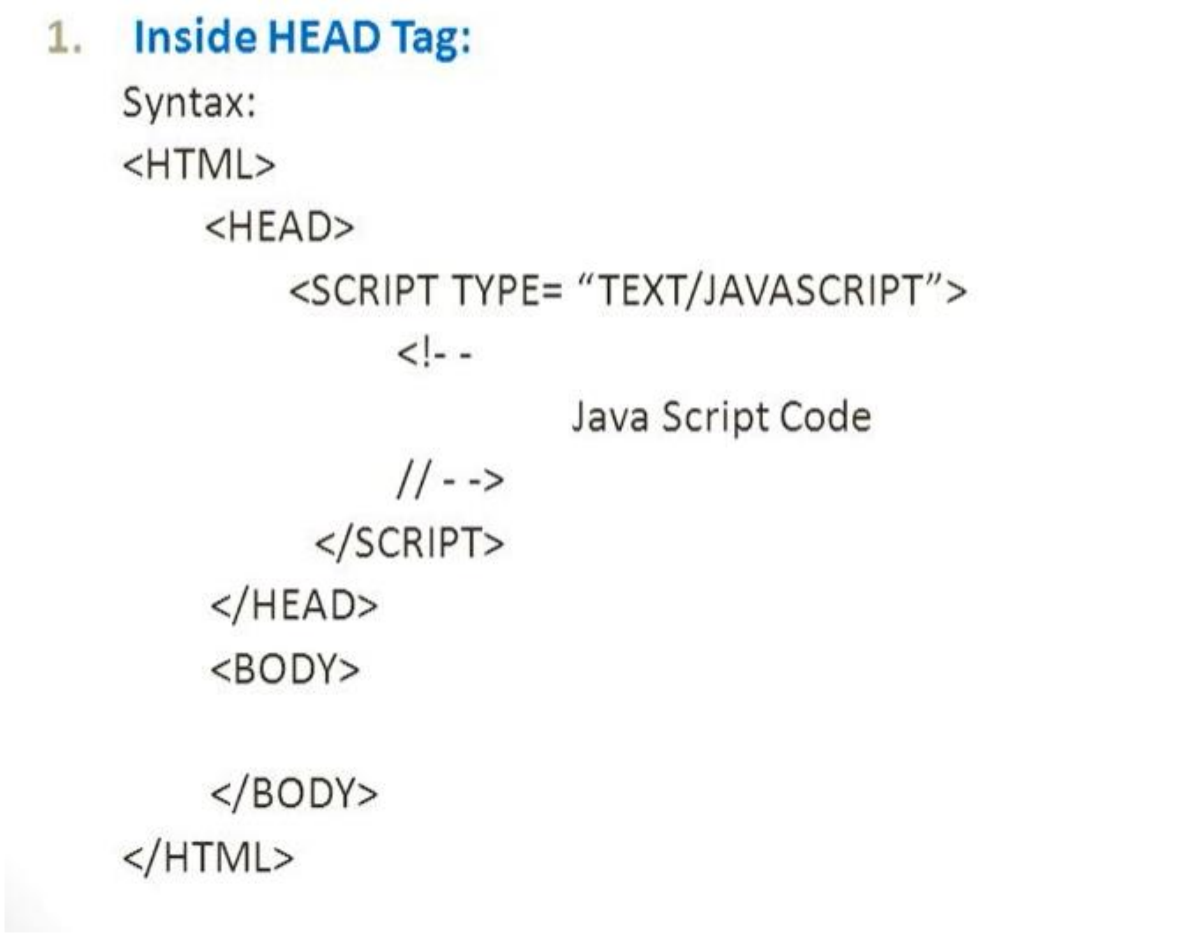
JavaScript Application

- **JavaScript is used to create interactive websites. It is mainly used for:**
  - Client-side validation
  - Dynamic drop-down menus
  - Displaying date and time
  - Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box)
  - Displaying clocks etc.

Features of JS



Syntax of JS

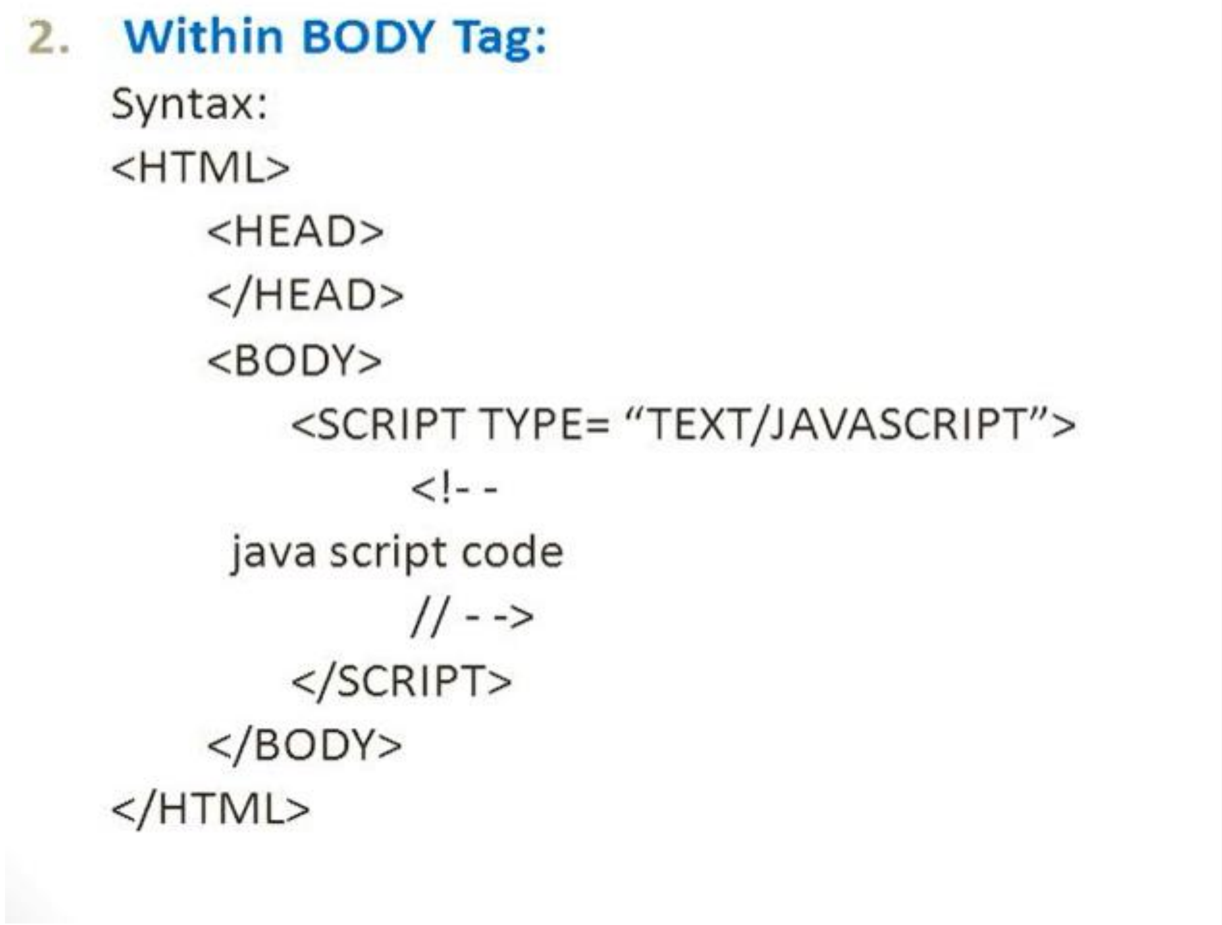


1. **Inside HEAD Tag:**

Syntax:

```
<HTML>
      <HEAD>
            <SCRIPT TYPE= "TEXT/JAVASCRIPT">
                    <!- -
                                Java Script Code
                    // - ->
             </SCRIPT>
      </HEAD>
      <BODY>

      </BODY>
</HTML>
```

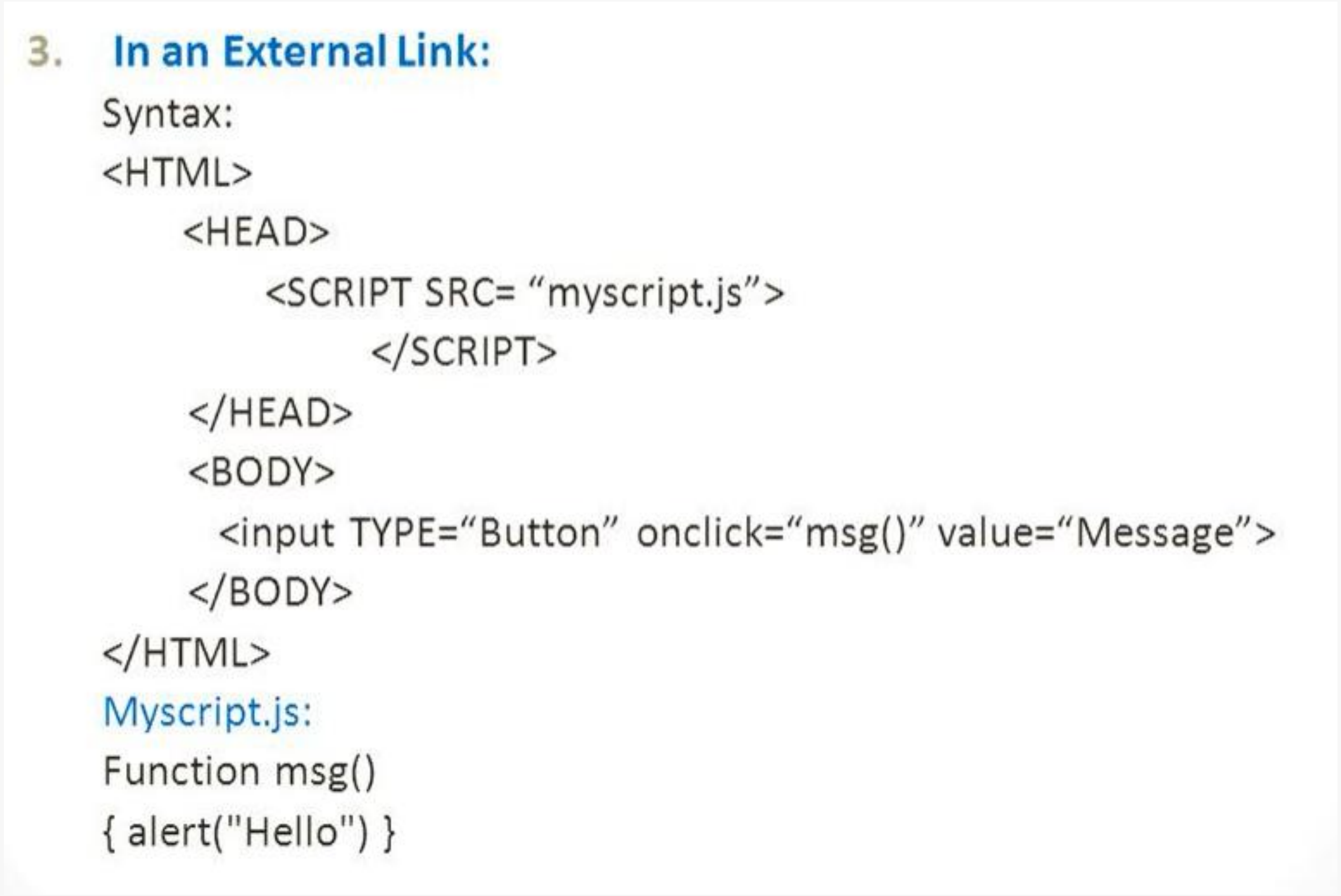


2. **Within BODY Tag:**

Syntax:

```
<HTML>
       <HEAD>
       </HEAD>
       <BODY>
             <SCRIPT TYPE= "TEXT/JAVASCRIPT">
                      <!- -
        java script code
                      // - ->
             </SCRIPT>
       </BODY>
</HTML>
```

JavaScript



3. **In an External Link:**

Syntax:

```
<HTML>
        <HEAD>
                <SCRIPT SRC= “myscript.js”>
                          </SCRIPT>
        </HEAD>
        <BODY>
           <input TYPE=“Button”  onclick=“msg()” value=“Message”>
        </BODY>
</HTML>
```

Myscript.js:

```
Function msg()
{ alert("Hello") }
```

H

Here's what happens when a browser loads a website with a <script> tag on it:

- Fetch the HTML page (e.g., index.html)
- Begin parsing the HTML
- The parser encounters a <script> tag referencing an external script file.
- The browser requests the script file. Meanwhile, the parser blocks and stops parsing the other HTML on your page.
- After some time, the script is downloaded and subsequently executed.
- The parser continues parsing the rest of the HTML document.

- Step #4 causes a bad user experience. Your website basically stops loading until you've downloaded all scripts.
- If there's one thing that users hate, it's waiting for a website to load.

The old approach

- The old approach to solving this problem was to put <script> tags at the bottom of your <body>, because this ensures the parser isn't blocked until the very end.

- This approach has its own problem: the browser cannot start downloading the scripts until the entire document is parsed.

- For larger websites with large scripts and stylesheets, being able to download the script as soon as possible is very important for performance.
- If your website doesn't load within 2 seconds, people will go to another website.

The modern approach

- Today, browsers support the async and defer attributes on scripts. These attributes tell the browser it's safe to continue parsing while the scripts are being downloaded.

async

```
<script src="path/to/script1.js" async></script>
<script src="path/to/script2.js" async></script>
```

- Scripts with the async attribute are executed asynchronously.
- This means the script is executed as soon as it's downloaded, without blocking the browser in the meantime.
- This implies that it's possible that script 2 is downloaded and executed before script 1.
- According to http://caniuse.com/#feat=script-async, 97.78% of all browsers support this.

defer

- `<script src="path/to/script1.js" defer></script>`
- `<script src="path/to/script2.js" defer></script>`

Scripts with the defer attribute are executed in order (i.e. first script 1, then script 2). This also does not block the browser.

- Unlike async scripts, defer scripts are only executed **after the entire document has been loaded**.

- According to http://caniuse.com/#feat=script-defer, 97.79% of all browsers support this. 98.06% support it at least partially.

netlink | digital solutions

JavaScript : Output

JavaScript does not have any built-in print or display functions.

JavaScript can "display" data in different ways:

- Writing into an HTML element, using innerHTML
- Writing into the HTML output using document.write()
- Writing into an alert box, using window.alert()
- Writing into the browser console, using console.log()

JavaScript-Syntax

If we want to access an HTML element using javascript then we can use **two** different syntax

1)document.getElementById("idname").attribute

2)document.getElementByName("elementname").attribute

e.g.:-

```
<input type="text" id="txt"  />

a=document.getElementById("txt").value;
```