
iscte
INSTITUTO
UNIVERSITÁRIO
DE LISBOA
emprego
digital
UPskill

Javascript

JavaScript

- Programming language for the web
- High-level, dynamic and interpreted
  - Suitable for object-oriented and functional programando styles
- Different from Java
  - Except for a superficial syntactic resemblance

How to include a Javascript file in a html page?

1.Add code to event handler attributes in a tag

```
<input type="button" value="click" onClick='alert("btn clicked");' />
```

2. Nested in a script element

```
<script type="text/javascript" >alert("Give the user some info!");</script>
```

3. Specifying a js code file as a script element's source (src attribute)

```
<script src='https://upskill.pt/wp-includes/js/jquery/jquery.js?ver=1.12.4-wp'></script>
```

The basics

- JavaScript is case-sensitive
- Ignores whitespaces
    - E.g., Tabs and spaces
- A script is made up of **statements**
- Comments
    - Single-line: **//**
    - Multi-line: **/* */**

JavaScript statements and variables

- A **statement** is a command that tells the browser what to do
  - E.g., alert(“One Message”);

- A **variable** is like an information container – give it a name and assign it a value
  - E.g., let x = 5;
  - Variable names must start with a letter or underscore, cannot contain whitespaces or special characters (e.g., ! . , / \ + * =)

Data types

- Undefined
  - Variable is not defined
- Null
  - Variable is defined, although has no inherent value
- Numbers
- Strings
- Booleans
  - true or false
- Arrays
  - Group of multiple values that can be assigned to a single variable
- Object

Comparison operators

==	Is equal to
!=	Is not equal to
===	Is identical to (equal to and of the same data type)
!==	Is not identical to
>	Is greater than
>=	Is greater than or equal to
<	Is less than
<=	Is less than or equal to

Mathematical operators

+=	Adds the value to itself
++	Increases the value of a number (or a variable containing a number value) by 1
--	Decreases the value of a number (or a variable containing a number value) by 1

Conditional statements

- It tells the browser

  “if this condition is met, then execute the commands listed between the curly brackets {}”

```
if( test == "testing" ) {
        alert( "You haven't changed anything." );
} else {
     alert( "You've changed something!" );
}
```

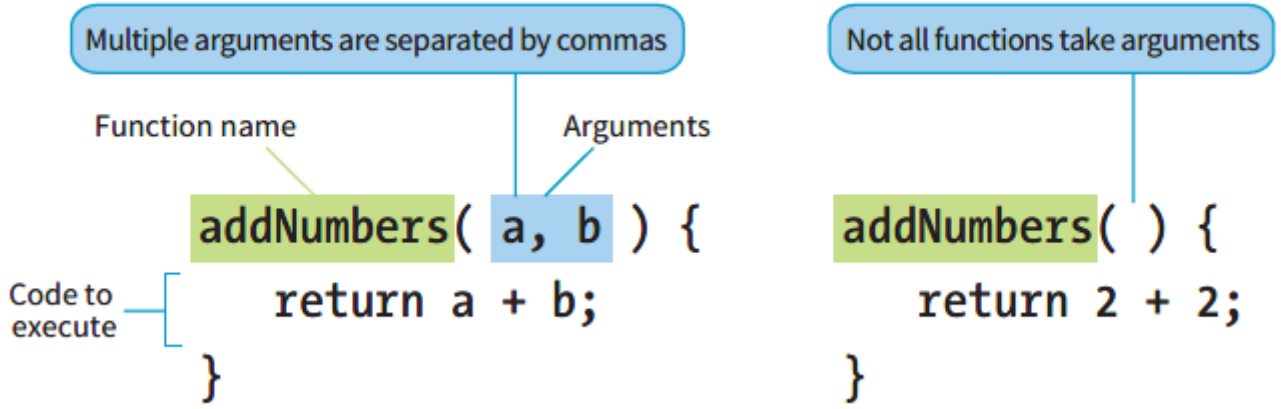
Loops

- How to iterate over an array?
  - **length**: instead of limiting the iterations, use the number of elements it contains
  - **[i]**: access an element in the *i*th position

```
for( initialize the variable;  test the condition;  alter the value; ) {
    // do something
}
```

Functions

- A bit of code for performing a task that doesn't run until it is referenced or called
- Allows code reuse instead of repetition
- Can have arguments – a value that a function uses when it runs
- Can return a value which is the result of such a function



Types of functions

- Native
  - Functions that are built into Javascript
  - E.g., alert(), confirm(), prompt(), Date(), parseInt(“123”)

- Custom functions
  - Our own functions

```
function name() {
    // Our function code goes here.
}
```

JavaScript events

- Action that can be detected with JavaScript
- Event handlers "listen" for certain document, browser, or user actions and bind scripts to those actions

Event handler	Event description
onblur	An element loses focus.
onchange	The content of a form field changes.
onclick	The mouse clicks an object.
onerror	An error occurs when the document or an image loads.
onfocus	An element gets focus.
onkeydown	A key on the keyboard is pressed.
onkeypress	A key on the keyboard is pressed or held down.
onkeyup	A key on the keyboard is released.
onload	A page or an image is finished loading.
onmousedown	A mouse button is pressed.
onmousemove	The mouse is moved.
onmouseout	The mouse is moved off an element.
onmouseover	The mouse is moved over an element.
onmouseup	A mouse button is released.
onsubmit	The submit button is clicked in a form.

JavaScript events

```
<body onclick="myFunction();"> /* myFunction will now run when the user
clicks anything within 'body' */
```

```
window.onclick = function() {
    /* Any code placed here will run when the user clicks anything
within the browser window */
};
```

```
window.addEventListener("click", function(e) {

});
```

Test yourself

Given the following array

```
var myArray = [1, "two", 3, "4"]
```

write what the alert message will say for each of these examples:

a. `alert( myArray[0] );`

b. `alert( myArray[0] + myArray[1] );`

c. `alert( myArray[2] + myArray[3] );`

d. `alert( myArray[2] - myArray[0] );`

Test yourself

What will each of these alert messages say?

a.
```
var foo = 5;
foo += 5;
alert( foo );
```

b.
```
i = 5;
i++;
alert( i );
```

c.
```
var foo = 2;
alert( foo + " " + "remaining");
```

d.
```
var foo = "Mat";
var bar = "Jennifer";
if( foo.length > bar.length ) {
  alert( foo  + " is longer." );
} else {
  alert( bar + " is longer." );
}
```

e.
```
alert( 10 === "10" );
```

Test yourself

Match each event handler with its trigger.

a. `onload`	1. The user finishes a form and hits the submit button.
b. `onchange`	2. The page finishes loading.
c. `onfocus`	3. The pointer hovers over a link.
d. `onmouseover`	4. A text-entry field is selected and ready for typing.
e. `onsubmit`	5. A user changes her name in a form field.

Document Object Model (DOM)

- A way of accessing and manipulating the contents of a document
  - It represents the page so that programs can change the document structure, style, and content
- Is a programming interface (API) for HTML and XML pages
- Provides:
  - A structured map of the document
  - Set of methods to interface with the contained elements
- Translates markup into a format that JavaScript “understands”

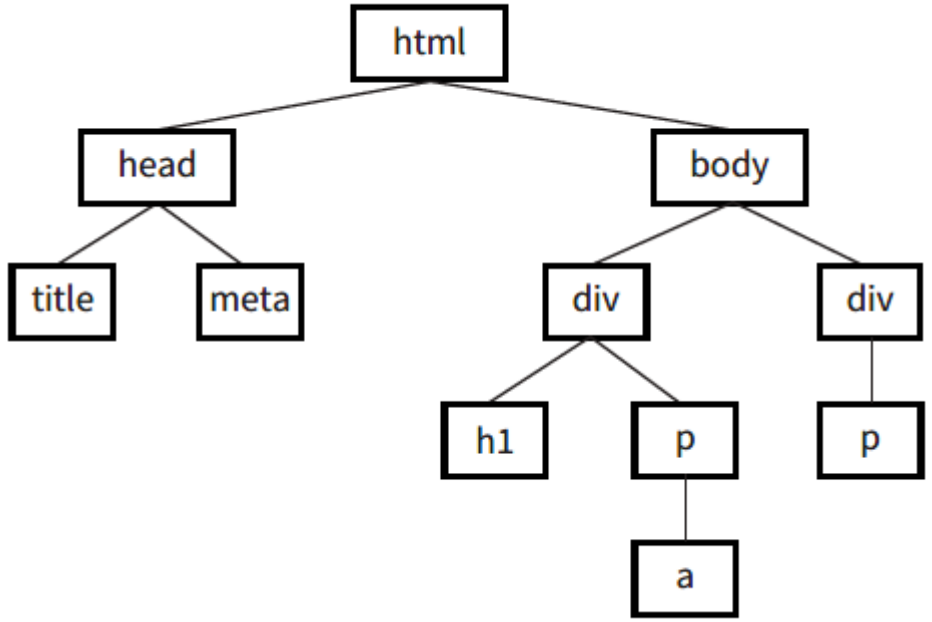
https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

Document Object Model (DOM)

```
<!DOCTYPE html>
<html>
<head>
  <title>Document title</title>
  <meta charset="utf-8">
</head>
<body>
  <div>
    <h1>Heading</h1>
    <p>Paragraph text with a <a href="foo.html">link</a> here.</p>
  </div>
  <div>
    <p>More text here.</p>
  </div>
</body>
</html>
```

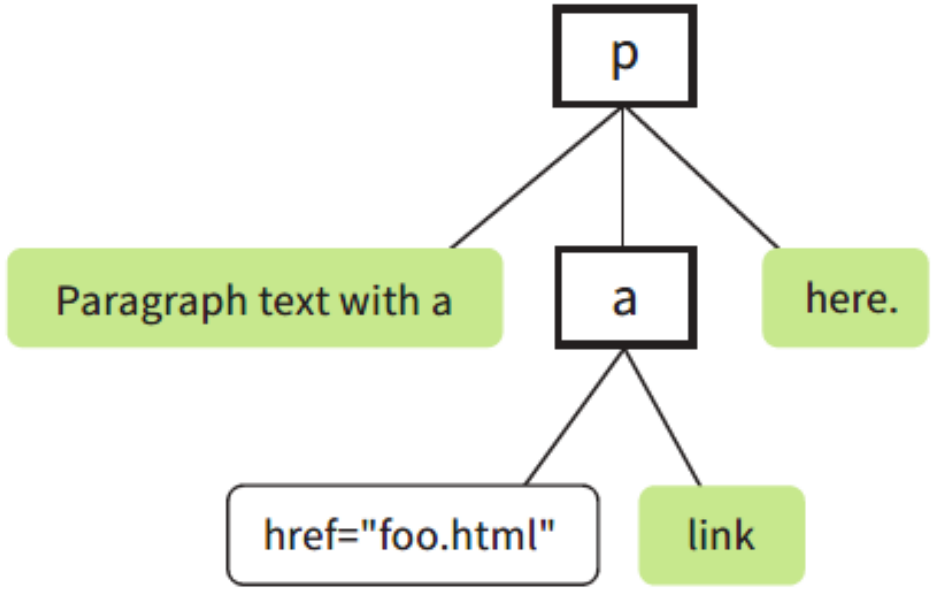
Document Object Model (DOM)

```
<!DOCTYPE html>
<html>
<head>
  <title>Document title</title>
  <meta charset="utf-8">
</head>
<body>
  <div>
    <h1>Heading</h1>
    <p>Paragraph text with a <a href="foo.html">link</a> here.</p>
  </div>
  <div>
    <p>More text here.</p>
  </div>
</body>
</html>
```



Document Object Model (DOM)

Paragraph text with a link here.



Accessing the DOM

```
1  <html>
2    <head>
3      <script>
4         // run this function when the document is loaded
5         window.onload = function() {
6
7           // create a couple of elements in an otherwise empty HTML page
8           const heading = document.createElement("h1");
9           const heading_text = document.createTextNode("Big Head!");
10          heading.appendChild(heading_text);
11          document.body.appendChild(heading);
12        }
13     </script>
14   </head>
15   <body>
16   </body>
17 </html>
```

```
document.querySelector(selector)
document.querySelectorAll(name)
document.createElement(name)
parentNode.appendChild(node)
element.innerHTML
element.style.left
element.setAttribute()
element.getAttribute()
element.addEventListener()
window.content
window.onload
window.scrollTo()
```

Fundamental data types

Data type (Interface)	Description
`Document`	When a member returns an object of type `document` (e.g., the `ownerDocument` property of an element returns the `document` to which it belongs), this object is the root `document` object itself. The DOM `document` Reference chapter describes the `document` object.
`Node`	Every object located within a document is a node of some kind. In an HTML document, an object can be an element node but also a text node or attribute node.
`Element`	The `element` type is based on `node`. It refers to an element or a node of type `element` returned by a member of the DOM API. Rather than saying, for example, that the `document.createElement()` method returns an object reference to a `node`, we just say that this method returns the `element` that has just been created in the DOM. `element` objects implement the DOM `Element` interface and also the more basic `Node` interface, both of which are included together in this reference. In an HTML document, elements are further enhanced by the HTML DOM API's `HTMLElement` interface as well as other interfaces describing capabilities of specific kinds of elements (for instance, `HTMLTableElement` for `<table>` elements).
`NodeList`	A `nodeList` is an array of elements, like the kind that is returned by the method `document.querySelectorAll()`. Items in a `nodeList` are accessed by index in either of two ways: • list.item(1) • list[1] These two are equivalent. In the first, `item()` is the single method on the `nodeList` object. The latter uses the typical array syntax to fetch the second item in the list.
`Attribute`	When an `attribute` is returned by a member (e.g., by the `createAttribute()` method), it is an object reference that exposes a special (albeit small) interface for attributes. Attributes are nodes in the DOM just like elements are, though you may rarely use them as such.
`NamedNodeMap`	A `namedNodeMap` is like an array, but the items are accessed by name or index, though this latter case is merely a convenience for enumeration, as they are in no particular order in the list. A `namedNodeMap` has an `item()` method for this purpose, and you can also add and remove items from a `namedNodeMap`.

Useful JavaScript libraries - jQuery

- Written in 2005 by John Resig
- If a site uses a JS library, 97% chance that it's jQuery
- Free, open source and employs and easy to use syntax
- Other libraries include:
  - MooTools
  - Dojo
  - Prototype

https://jquery.com/

Useful JavaScript libraries - jQuery

A Brief Look

DOM Traversal and Manipulation

Get the `<button>` element with the class 'continue' and change its HTML to 'Next Step...'

```
$( "button.continue" ).html( "Next Step..." )
```

Event Handling

Show the `#banner-message` element that is hidden with `display:none` in its CSS when any button in `#button-container` is clicked.

```
var hiddenBox = $( "#banner-message" );
$( "#button-container button" ).on( "click", function( event ) {
  hiddenBox.show();
});
```

Ajax

Call a local script on the server `/api/getWeather` with the query parameter `zipcode=97201` and replace the element `#weather-temp`'s html with the returned text.

```
$.ajax({
  url: "/api/getWeather",
  data: {
    zipcode: 97201
  },
  success: function( result ) {
    $( "#weather-temp" ).html( "<strong>" + result + "</strong> degrees" );
  }
});
```

How to use jQuery?

- Download:
  - https://jquery.com/download/
- Documentation:
  - https://api.jquery.com/
- Does my browser support it?
  - https://jquery.com/browser-support/

Asynchronous JavaScript and XML (AJAX)

- Term coined by Jesse James Garrett
- Is not a single technology, but a combination of HTML, CSS, DOM, and JavaScript – including **XMLHttpRequest** object, which allows data to be transferred asynchronously
  - Ajax may use XML for data, but normally uses JSON (JavaScript Object Notation)

Example – Google Books API

- Google Books API v1 gives you programmatic access to many of the operations available on Google Books website
- Some of the main features that the API provides are:
  - search and browse through the list of books that match a given query.
  - view information about a book, including metadata, availability and price, links to the preview page.
  - manage your own bookshelves.

https://developers.google.com/books/docs/overview

Operation	Description	REST HTTP mappings
list	Lists a specified subset of resources within a collection.	GET on a collection URI.
insert	Inserts a new resource into a collection (creating a new resource).	POST on a collection URI, where you pass in data for a new resource.
get	Gets a specific resource.	GET on resource URI.
update	Updates a specific resource.	PUT on resource URI, where you pass in data for the updated resource.
delete	Deletes a specific resource.	DELETE on resource URI, where you pass in data for the resource to be deleted.

https://developers.google.com/books/docs/v1/getting_started

Example usage

Perform a search for quilting:

```
GET https://www.googleapis.com/books/v1/volumes?q=quilting
```

References

- David Flanagan, **“JavaScript: The Definitive Guide”**, O'Reilly Media, Inc., 2011
- Douglas Crockford, **“JavaScript: The Good Parts”**, O'Reilly Media, Inc., 2008
- Robbins, Jennifer, **“Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics”,** O'Reilly Media, Inc., 2018

Thank you

O futuro profissional começa aqui

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

UPskill