Cascading Style Sheets

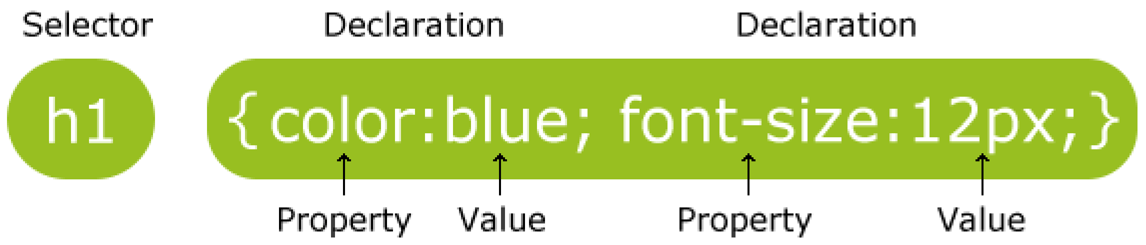
Web Engineering

UNIVERSIDADE
DE TRÁS-OS-MONTES
E ALTO DOURO

What is the purpose of CSS? What alternatives are there for associating styles with HTML documents?

CSS – Cascading Style Sheets

Defines how the elements contained in the code of a web page will be displayed and its biggest advantage is to separate the format and content of a document.



The purpose is…

CSS – Cascading Style Sheets

How to associate with documents…

External

Internal

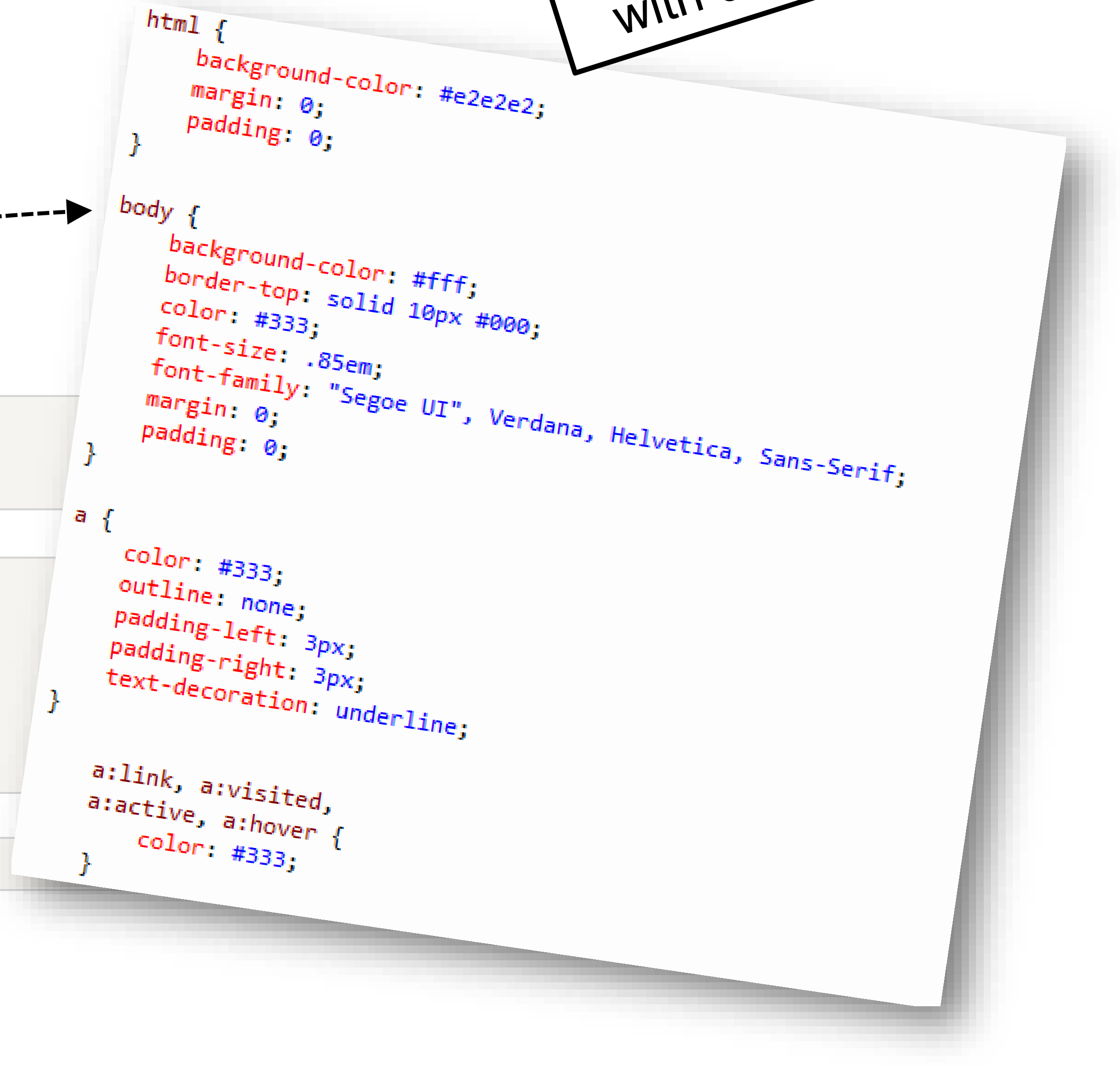
Inline

External file

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

```
<head>
<style>
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("images/back40.gif");}
</style>
</head>
```

```
<p style="color:sienna;margin-left:20px;">This is a paragraph.</p>
```



```
html {
    background-color: #e2e2e2;
    margin: 0;
    padding: 0;
}

body {
    background-color: #fff;
    border-top: solid 10px #000;
    color: #333;
    font-size: .85em;
    font-family: "Segoe UI", Verdana, Helvetica, Sans-Serif;
    margin: 0;
    padding: 0;
}

a {
    color: #333;
    outline: none;
    padding-left: 3px;
    padding-right: 3px;
    text-decoration: underline;
}

a:link, a:visited,
a:active, a:hover {
    color: #333;
}
```

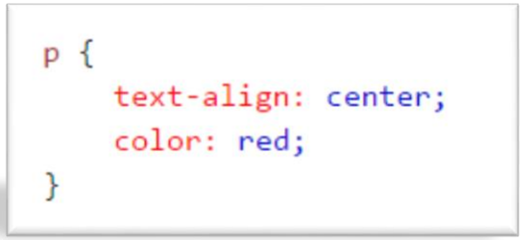
CSS – Cascading Style Sheets

Selectors:

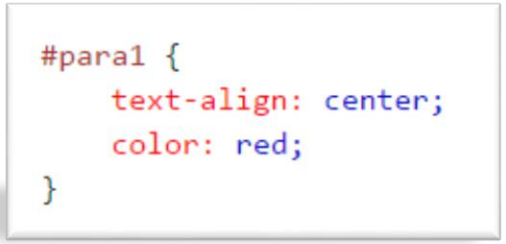
Element, ID, Class, Grouping

(https://www.w3schools.com/cssref/css_selectors.php)

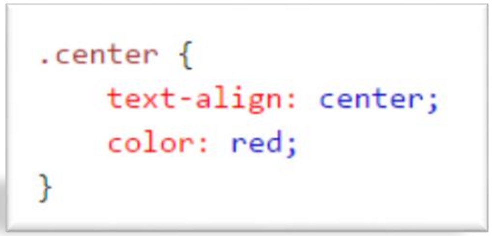
How to associate with elements…



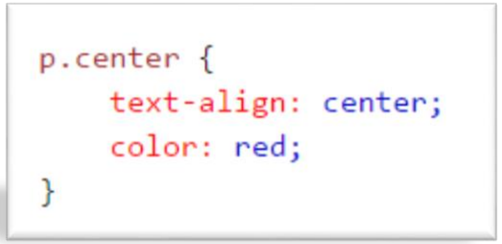
```
p {
    text-align: center;
    color: red;
}
```



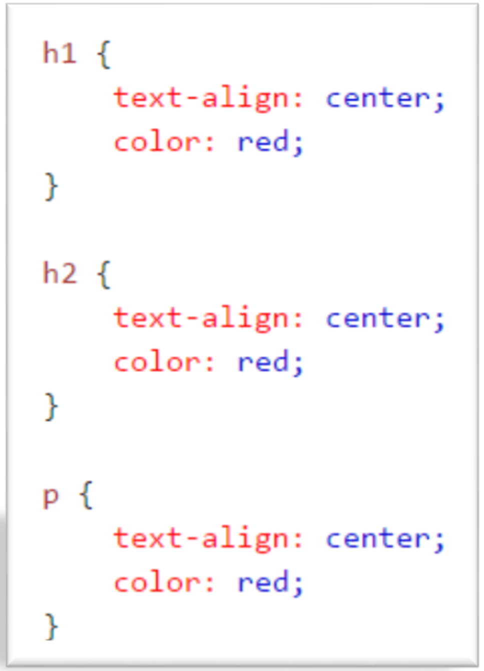
```
#para1 {
    text-align: center;
    color: red;
}
```



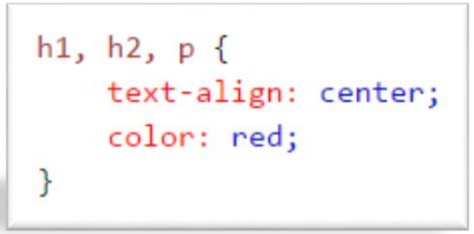
```
.center {
    text-align: center;
    color: red;
}
```



```
p.center {
    text-align: center;
    color: red;
}
```



```
h1 {
    text-align: center;
    color: red;
}

h2 {
    text-align: center;
    color: red;
}

p {
    text-align: center;
    color: red;
}
```



```
h1, h2, p {
    text-align: center;
    color: red;
}
```

CSS – Cascading Style Sheets

syntax

```
selector { property: value }
```

Inheritance

Nested elements inherit the properties of the elements that contain them

```
body {font-family: Verdana, serif;}
h1 {font-family: Georgia, sans-serif;}
p {font-family: Tahoma, serif;}
```

CSS – Cascading Style Sheets

Selector combination

Comma separator allows you to replicate properties across multiple selectors

```
h1, h2, h3, h4, h5, h6 {
		color: #009900;
		font-family: Georgia, sans-serif;
}
```

CSS – Cascading Style Sheets

classes

selector started by a dot (.)

```
.greenboldtext{
        font-size: small;
        color: #008080;
        font-weight: bold;
}

<span class="greenboldtext">my text inside span element</span>
```

CSS – Cascading Style Sheets

ids

seletor started by cardinal (#)

```
#greenboldtext{
        font-size: small;
        color: #008080;
        font-weight: bold;
}

<span id="greenboldtext">my text inside span element</span>
```

CSS – Cascading Style Sheets

divisions vs spans

defines content treated as a block (div), inserting a line break before and after its rendering, or as an in-line element (span) that allows content formatting without breaking it.

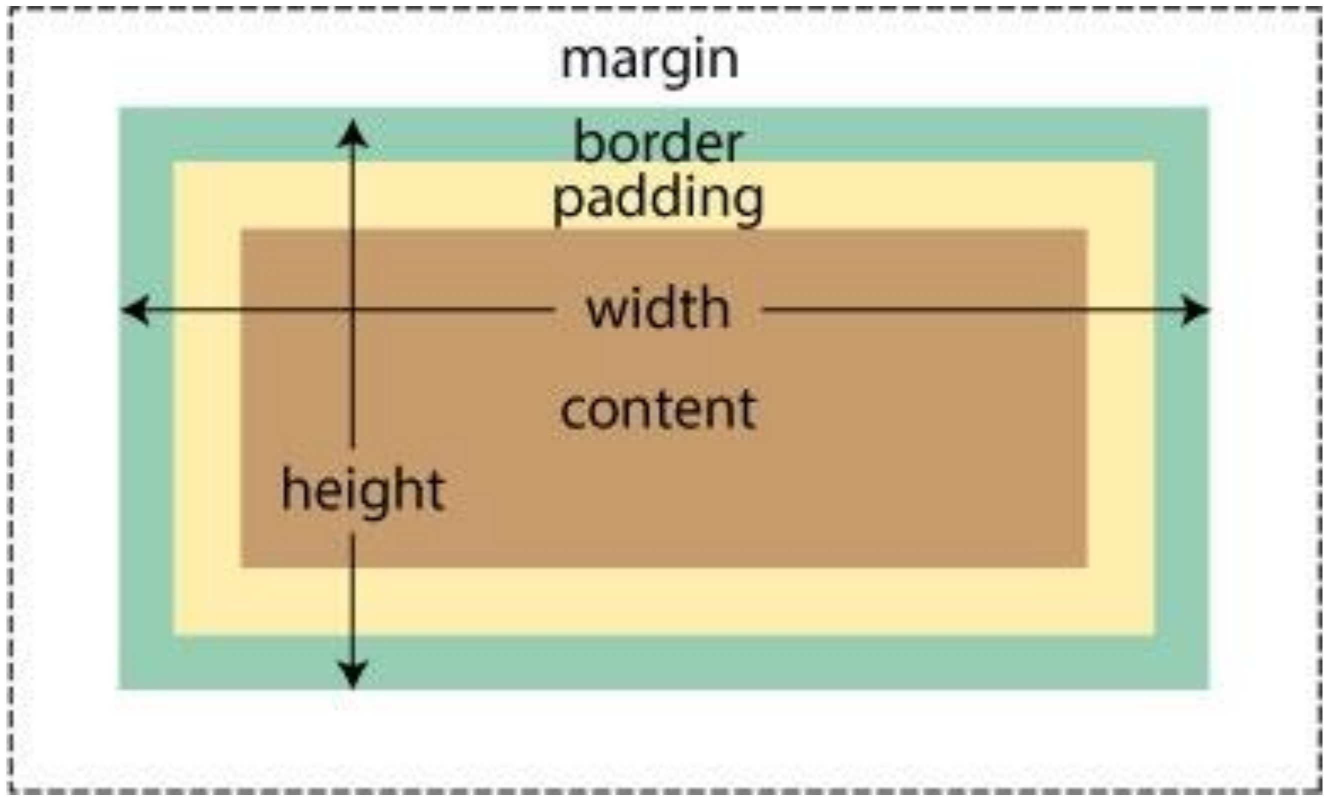
```
#underlined{
		text-decoration: underline;
}
```

<div>the text is <span id="underlined">highlighted</span> from the rest</div>

CSS – Cascading Style Sheets

CSS Box Model

defines the outer and inner spacing of content areas. .



CSS – Cascading Style Sheets

CSS Box Model *(properties without inheritance)*

margin

padding

1. top
2. right
3. bottom
4. left

values in absolute (px) or percentage (%)

```
margin: 10px 10px 10px 10px;
Margin-top: 10px;

padding: 10px 10px 10px 10px;

padding-left: 10px;
padding-right: 10px;
```

CSS – Cascading Style Sheets

text properties

- color
  - color name - example:(red, black...)
  - hexadecimal number - example:(#ff0000, #000000)
  - RGB color code - example: (rgb(255, 0, 0), rgb(0, 0, 0))
- letter-spacing
  - normal
  - length
- text-align
  - left
  - right
  - center
  - justify

CSS – Cascading Style Sheets

text properties

- text-decoration
  - none
  - underline
  - overline
  - line through
  - Blink

- text-indent
  - length
  - Percentage

- text-transform
  - none
  - capitalize
  - lowercase

CSS – Cascading Style Sheets

text font properties

- font-family
  - family-name
  - generic family
- font-size
  - xx-large
  - x-large
  - larger
  - large
  - medium
  - small
  - smaller
  - x-small
  - xx-small
  - length
  - % (percent)

CSS – Cascading Style Sheets

text font properties

- font-style
  - normal
  - italic
  - oblique

- font-variant
  - normal
  - small-caps

CSS – Cascading Style Sheets

text font properties

- font-weight
  - lighter
  - normal
  - 100
  - 200
  - 300
  - 400
  - 500
  - 600
  - 700
  - 800
  - 900
  - bold
  - bolder

CSS – Cascading Style Sheets

text font properties

font

combines several font properties style, weight, variant, size, line height and font-family (in arbitrary order)

```
font: italic bold normal small/1.4em Verdana, sans-serif;
```

CSS – Cascading Style Sheets

CSS Borders

1. top
2. right
3. bottom
4. left

border-color

- color name
- hexadecimal number
- RGB color code
- Transparent

CSS – Cascading Style Sheets

CSS Borders

…

border-style

- dashed
- dotted
- double
- groove
- hidden
- inset
- none
- outset
- ridge
- solid

CSS – Cascading Style Sheets

CSS Borders

…

border-width

- Length
- Thin
- Medium
- Thick

border-left-color
border-top-color
border-right-color
border-bottom-color
…
border-left-width
border-top-width
border-right-width
border-bototm-width
…

CSS – Cascading Style Sheets

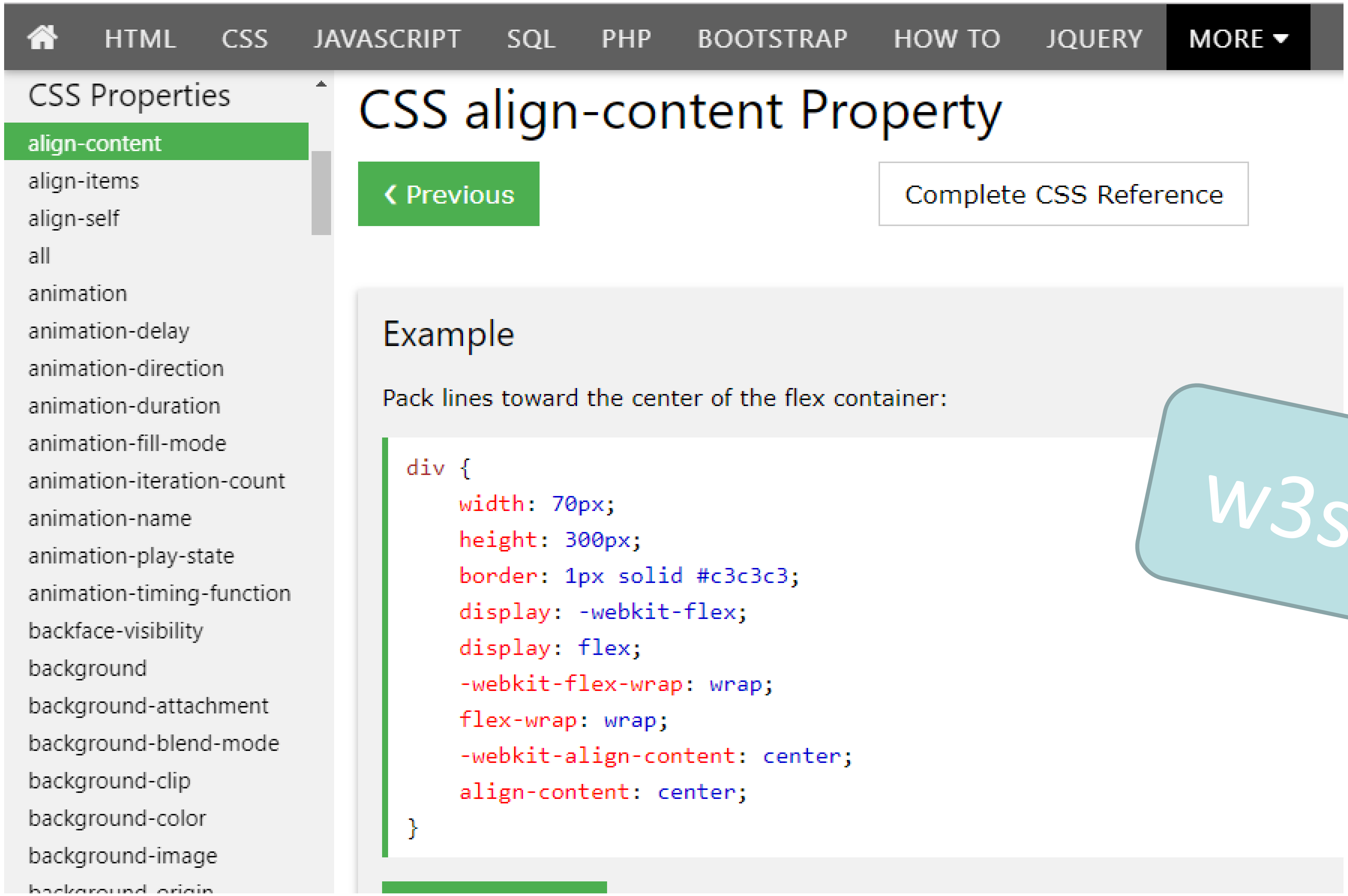
display

- block – line break before and after the block
- Inline – without adding line breaks
- list-item – line break before and after the block plus the list mark
- none – does not render

visibility

- hidden – hide the content
- visible – shows the content

CSS – Cascading Style Sheets



CSS Properties

CSS align-content Property

Example

Pack lines toward the center of the flex container:

```
div {
    width: 70px;
    height: 300px;
    border: 1px solid #c3c3c3;
    display: -webkit-flex;
    display: flex;
    -webkit-flex-wrap: wrap;
    flex-wrap: wrap;
    -webkit-align-content: center;
    align-content: center;
}
```

w3schools

XML vs JSON

Web Engineering

utad UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO

JSON : JavaScript Object Notation

- JSON stands for **J**ava**S**cript **O**bject **N**otation
- JSON is a lightweight format for data exchange.
- JSON is language independent *****
- JSON is self-describing and easy to understand

* JSON uses JavaScript syntax, but the JSON format is text only, such as XML.

Text can be read or used as a data format by any programming language.

JSON : JavaScript Object Notation

XML

```xml
<employees>
    <employee>
        <firstName>John</firstName>
       <lastName>Doe</lastName>
    </employee>
    <employee>
        <firstName>Anna</firstName>
       <lastName>Smith</lastName>
    </employee>
    <employee>
        <firstName>Peter</firstName>
       <lastName>Jones</lastName>
    </employee>
</employees>
```

JSON

```json
{"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]}
```

JSON : JavaScript Object Notation

Main Similarities

- JSON and XML are “self-describing” (human readable)
- JSON and XML are hierarchical (values within values)
- JSON and XML can be parsed and used by many programming languages.
- JSON and XML can be obtained by an XMLHttpRequest

Main differences

- JSON does not use closing tag
- JSON is shorter
- JSON is faster to read and write
- JSON can use arrays

JSON : JavaScript Object Notation

Syntactic rules

JSON syntax derives from syntax for JavaScript object notation:

- Data is name/value pairs
- Data is separated by commas
- Braces define (delimit) objects
- Square brackets define (delimit) arrays

object
 { }
 { *members* **}**
members
 pair
 pair **,** *members*
pair
 string **:** *value*
array
 [ ]
 [*elements* **]**
elements
 value
 value **,** *elements*
value
 string
 number
 object
 array
 `true`
 `false`
 `null`

JSON : JavaScript Object Notation

Values

- A number (integer or floating point)
- A string (enclosed in quotation marks)
- A boolean (true or false)
- An array (in square brackets)
- An object (between braces)
- null

object
    {}
    { *members* **}**
members
    pair
    pair **,** *members*
pair
    string **:** *value*
array
    []
    [*elements* **]**
elements
    value
    value **,** *elements*
value
    string
    number
    object
    array
    `true`
    `false`
    `null`

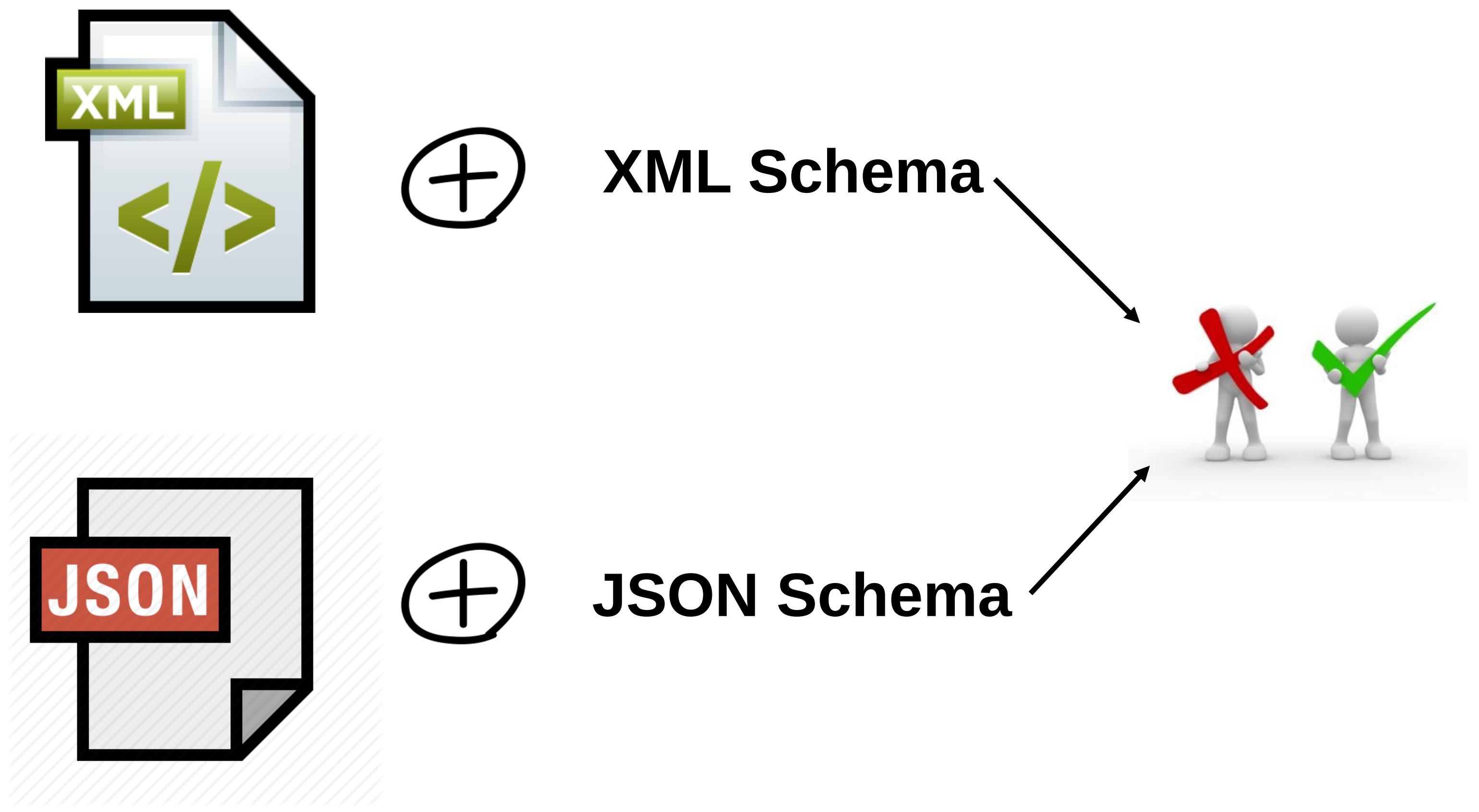
JSON : JavaScript Object Notation

Files

- The extension (type) of the JSON files is ".json“
- The MIME type for JSON text is "application/json”

JSON : JavaScript Object Notation

How to validate a file?



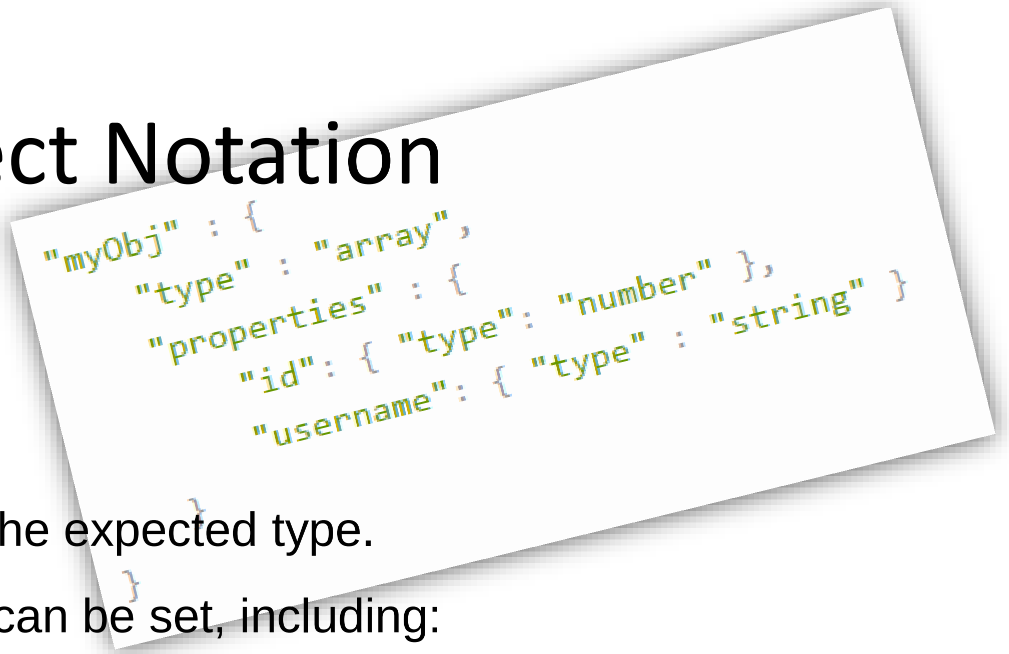
JSON : JavaScript Object Notation

JSON Schema

JSON Schema defines the “application/schema+json” MIME type with the JSON base format to describe the JSON data structure.

JSON Schema defines what a JSON document should look like, ways to extract information from it, and how to interact with it.

JSON : JavaScript Object Notation



```
"myObj" : {
    "type" : "array",
    "properties" : {
        "id": { "type": "number" },
        "username": { "type" : "string" }
    }
}
```

JSON Schema

Schema properties are set with another object containing the expected type.

In addition to providing the required type, other properties can be set, including:

items: This can be a schema or a schema array. When it is a schema/object and the instance value is an array, all items in the array must conform to this schema.

optional: Denotes whether the property should be considered optional.

requires: This indicates that if this property is present on the contained object instance, the property given by the requires attribute must also be present.

maxItems: Defines the maximum number of items in the collection.

JSON : JavaScript Object Notation

```
{
    "users": [
        {"id": 1, "username": "davidwalsh", "numPosts": 404, "realName": "David Walsh" },
        {"id": 2, "username": "russianprince", "numPosts": 12, "realName": "Andrei Arshavin" }
    ]
}
```

What can we see in this document?

- The object has a **users** property.
- The **users** property is an *array*.
- The **users** *array* contains objects
- Each object has an **id** (number), a **username** (string), a **numPosts** (number) and a **realName** (string).

JSON : JavaScript Object Notation

With this structure in mind, we can create a simple schema to validate our expected format:

```
{
    "type" : "object",
    "properties" : {
        "users" : {
            "type" : "array", // remember that arrays are objects
            "items" : { // "items" represents the items within the "users" array
                "type" : "object",
                "properties" : {
                    "id": { "type": "number" },
                    "username": { "type" : "string" },
                    "numPosts": { "type" : "number" },
                    "realName": { "type" : "string", optional: true }
                }
            }
        }
    }
}
```

Javascript

Web Engineering

UNIVERSIDADE
DE TRÁS-OS-MONTES
E ALTO DOURO

JavaScript / jQuery

Why JavaScript?

JavaScript is one of 3 languages that all web programmers (front-end) must learn:

1. **HTML** to define web page content
2. **CSS** to specify web page layout
3. **JavaScript** to program <u>web page</u> behavior

JavaScript / jQuery

What is JavaScript for?

- read/change HTML content
- read/change HTML attributes
- read/change styles applied to HTML (CSS)
- hide/show HTML elements…

JavaScript / jQuery

Where to use JavaScript?

JavaScript code must always be placed inside a SCRIPT element:

- can be placed on **HEAD**
- can be placed on **BODY**
- may be external to document (usually referenced in HEAD)

JavaScript / jQuery

Display (output) possibilities with JavaScript?

JavaScript can display data in the following ways:

- in an alert box: **window.alert()**
- directly in HTML: **document.write()**
- in an HTML element: **innerHTML**
- in the browser console: **console.log()**

JavaScript / jQuery

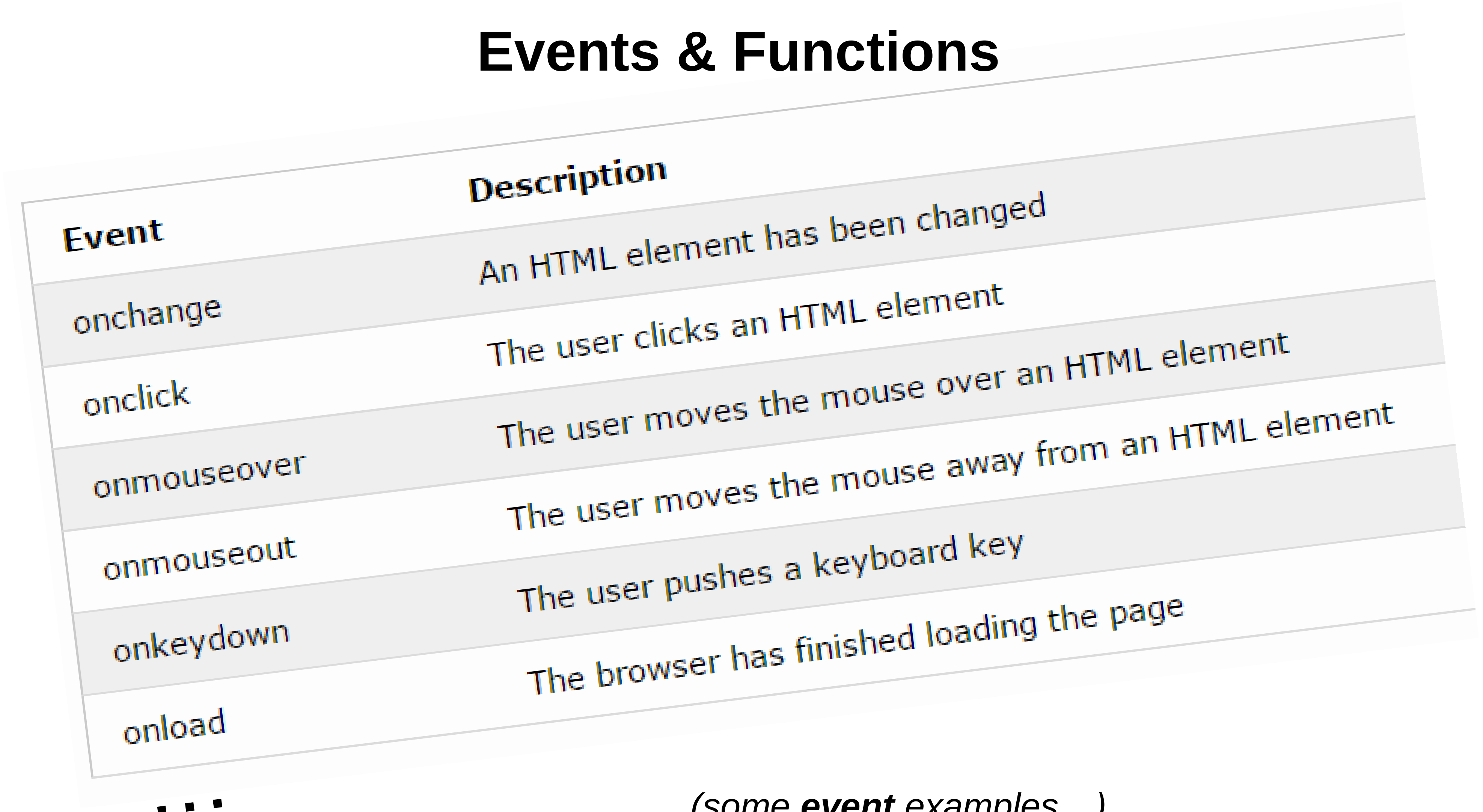
JavaScript Declarations?

They are composed of:

- Values : literals and variables (always declared with **var**)
- Operators : assignment, arithmetic operators, and logical operators
- Expressions
- Keywords
- comments

JavaScript / jQuery

Events & Functions



Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

...

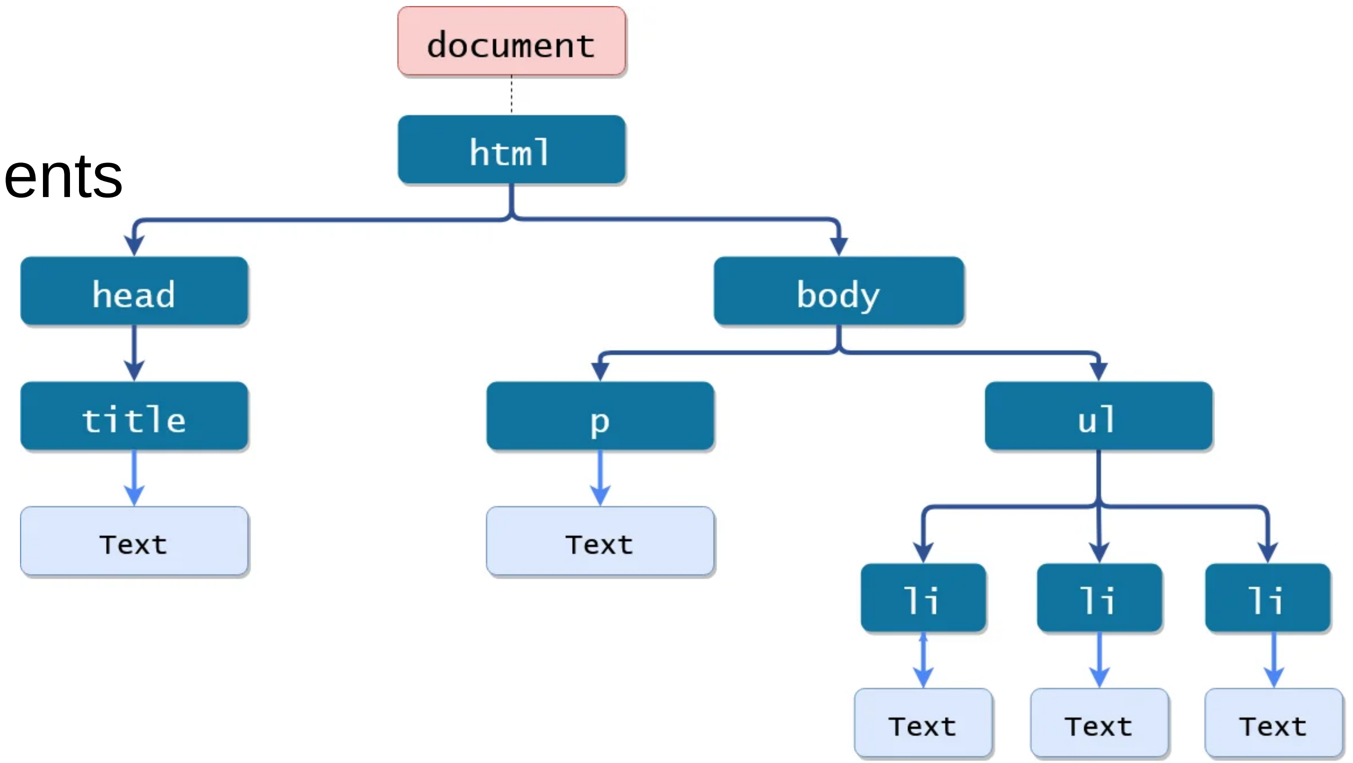
*(some **event** examples…)*

JavaScript / jQuery

JS HTML DOM & JS Browser BOM

Document Object Model (DOM) – Defines:

- HTML elements as **objects**
- Properties of all HTML elements
- Methods to access all HTML elements
- The **events** of all HTML elements

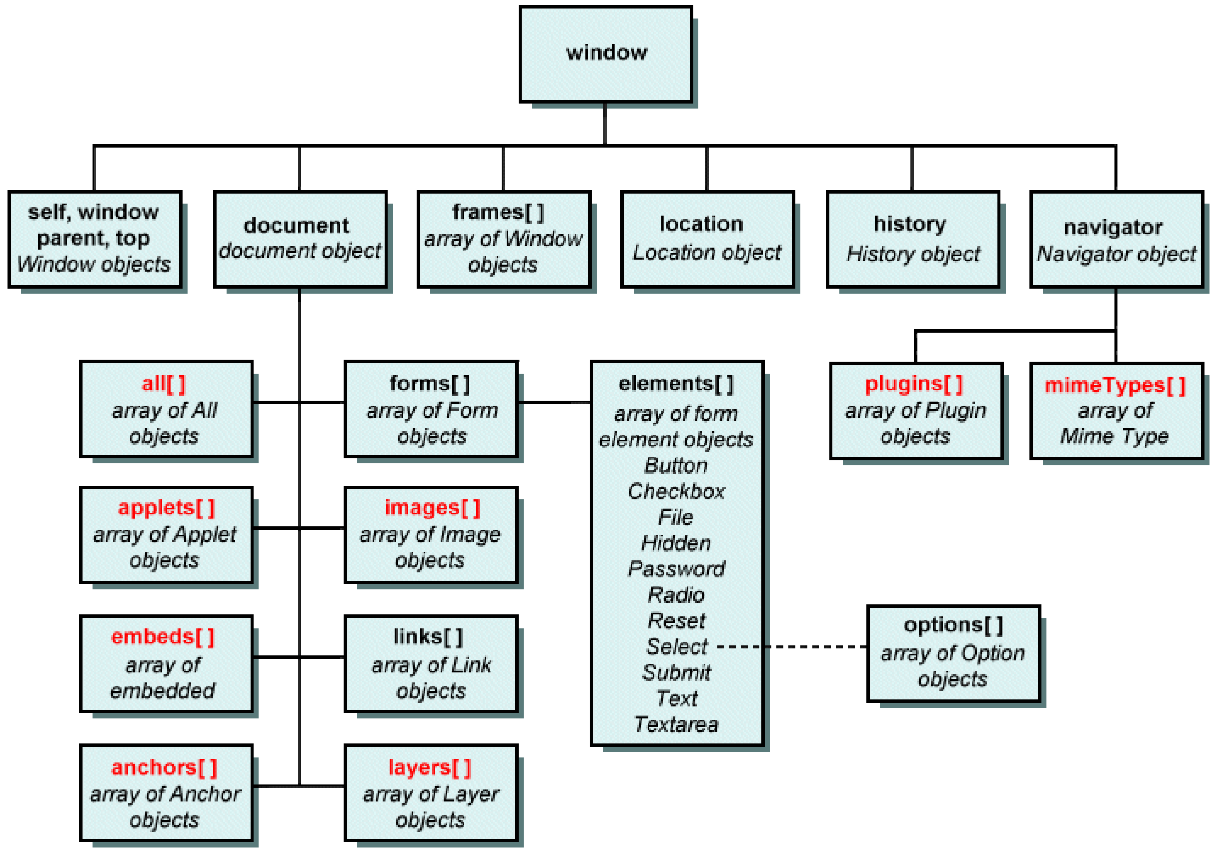


JavaScript / jQuery

JS HTML DOM & JS Browser BOM

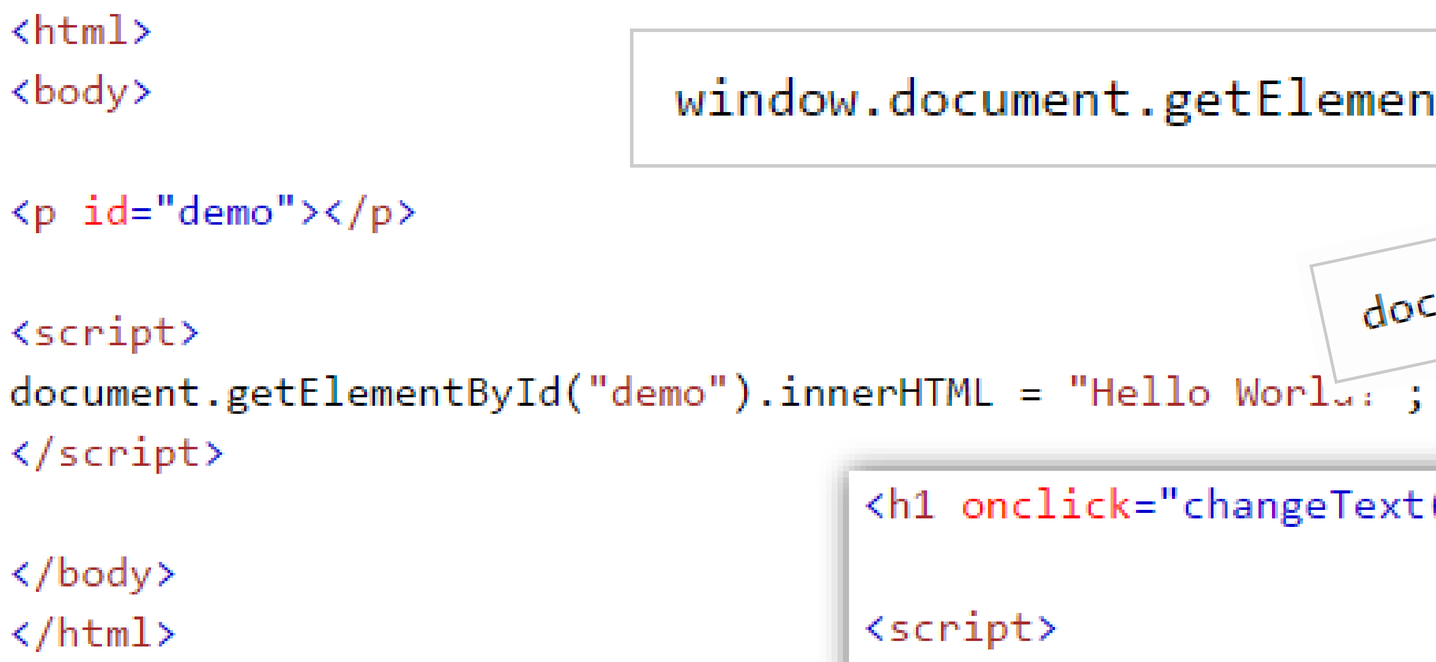
Browser Object Model (BOM) – Defines an object associated with the browser window. Contains:

- document
- screen
- location
- history
- navigator
- popup alert
- timing
- cookies

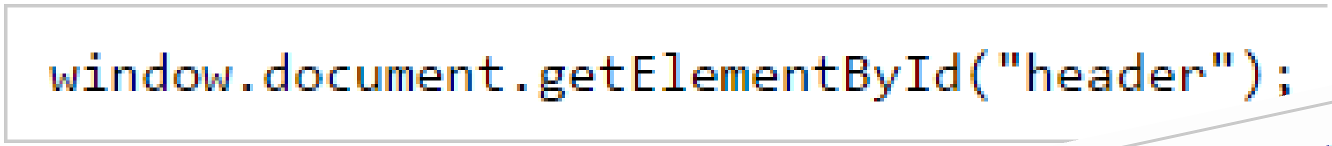

window
self, window parent, top Window objects
document document object
frames[ ] array of Window objects
location Location object
history History object
navigator Navigator object
all[ ] array of All objects
forms[ ] array of Form objects
elements[ ] array of form element objects Button Checkbox File Hidden Password Radio Reset Select Submit Text Textarea
plugins[ ] array of Plugin objects
mimeTypes[ ] array of Mime Type
applets[ ] array of Applet objects
images[ ] array of Image objects
embeds[ ] array of embedded
links[ ] array of Link objects
options[ ] array of Option objects
anchors[ ] array of Anchor objects
layers[ ] array of Layer objects

JavaScript / jQuery

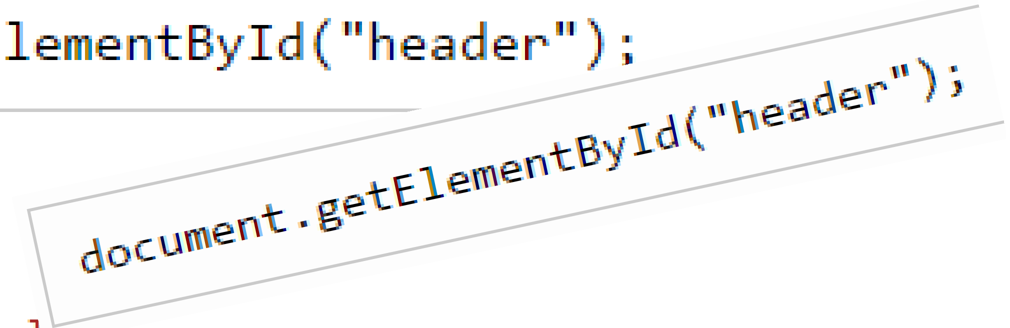
JS HTML DOM & JS Browser BOM



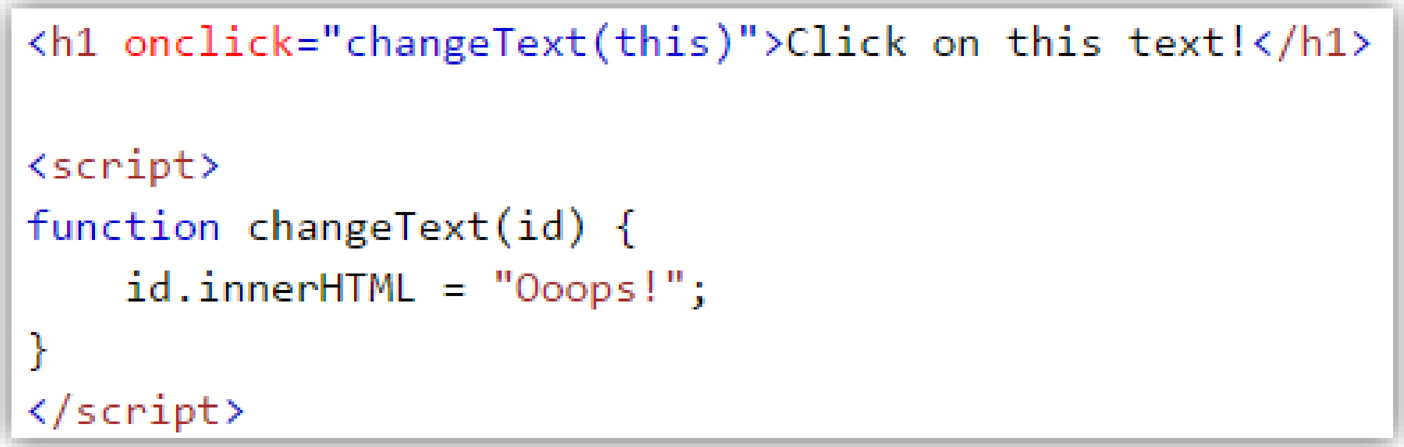
```
<html>
<body>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "Hello Worl  ;
</script>

</body>
</html>
```



```
window.document.getElementById("header");
```



```
document.getElementById("header");
```



```
<h1 onclick="changeText(this)">Click on this text!</h1>

<script>
function changeText(id) {
    id.innerHTML = "Ooops!";
}
</script>
```

JavaScript / jQuery

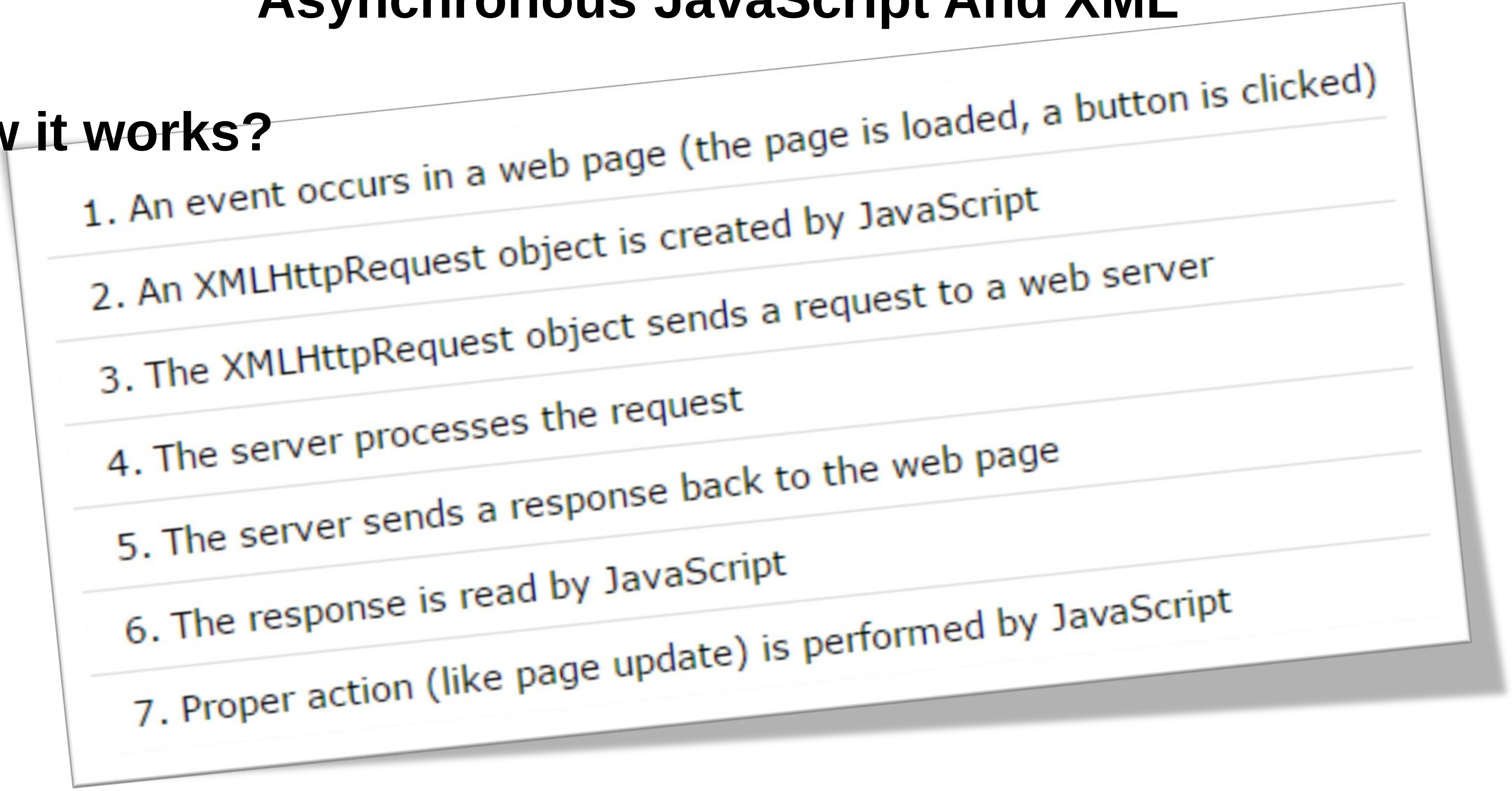
AJAX ?

- AJAX is not a programming language.
- AJAX is a technique for accessing web servers from a web page.
- AJAX stands for Asynchronous JavaScript And XML.

JavaScript / jQuery

Asynchronous JavaScript And XML

How it works?

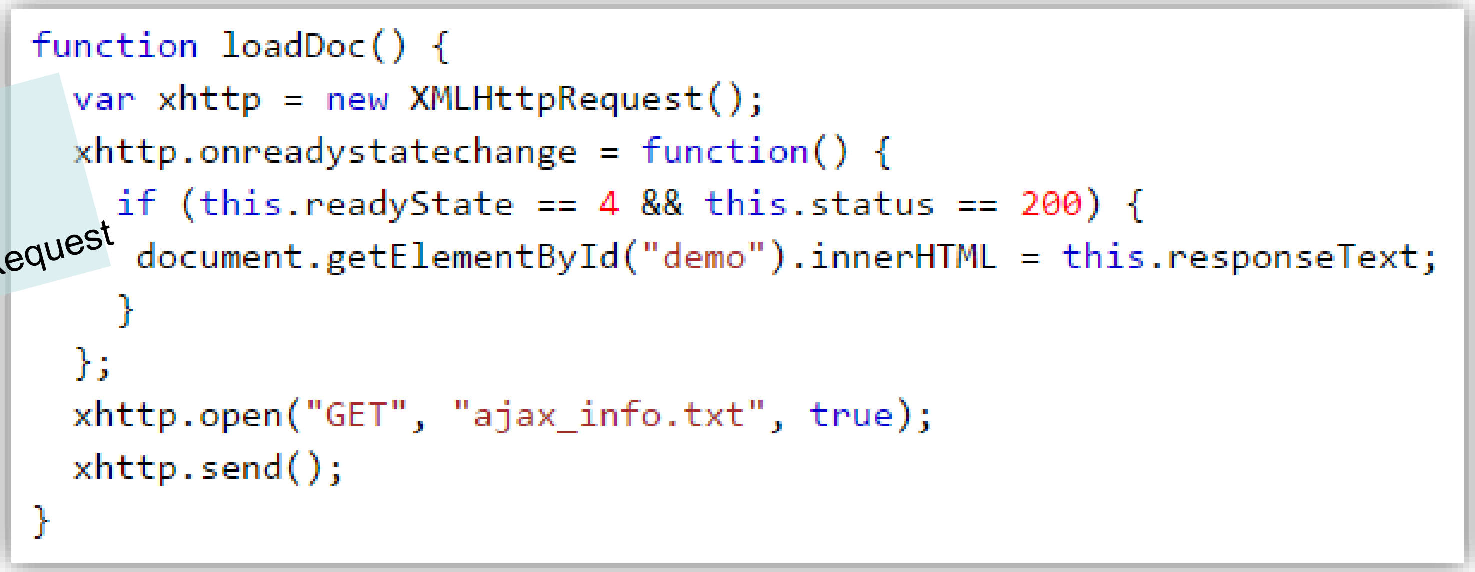


1. An event occurs in a web page (the page is loaded, a button is clicked)
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read by JavaScript
7. Proper action (like page update) is performed by JavaScript

JavaScript / jQuery

Asynchronous JavaScript And XML

```html
<!DOCTYPE html>
<html>
<body>

<div id="demo">
  <h2>Let AJAX change this text</h2>
  <button type="button" onclick="loadDoc()">Change Content</button>
</div>

</body>
</html>
```



```javascript
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
     document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
```

More info at

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

JavaScript / jQuery

A common use of JSON is to exchange data to/from a web server.

When receiving data from a web server, the data is always a string.

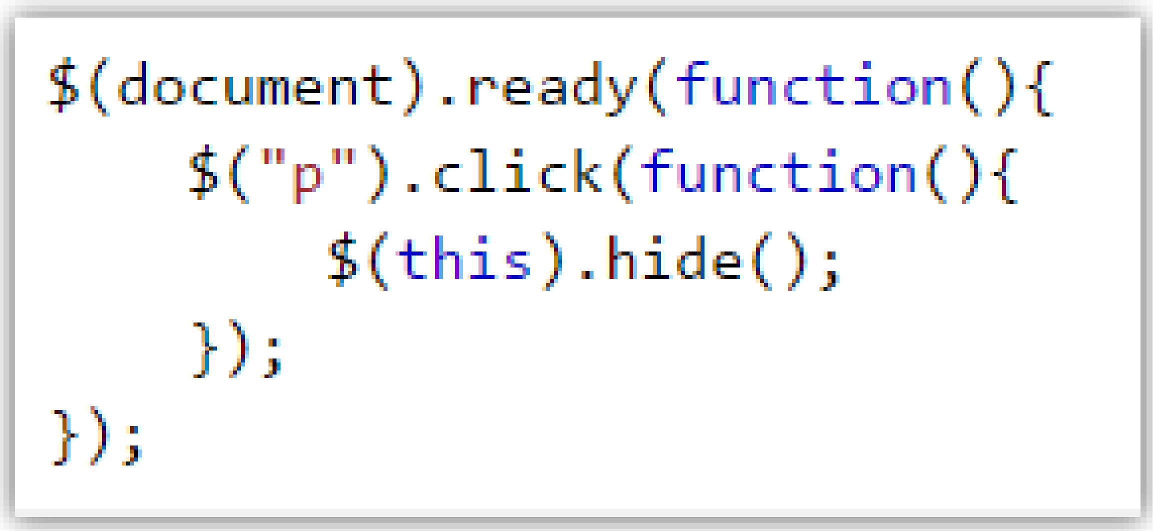
Parse the data with `JSON.parse()`, and the data becomes a JavaScript object.

using JSON in AJAX

JavaScript / jQuery

jQuery

- a JavaScript library;
- simplifies JavaScript programming;
- easy to learn;



```
$(document).ready(function(){
    $("p").click(function(){
        $(this).hide();
    });
});
```

JavaScript / jQuery

jQuery

The jQuery library contains the features:

- HTML/DOM manipulation
- CSS manipulation
- HTML events and methods
- effects and animations
- AJAX
- utilities

JavaScript / jQuery

jQuery - syntax

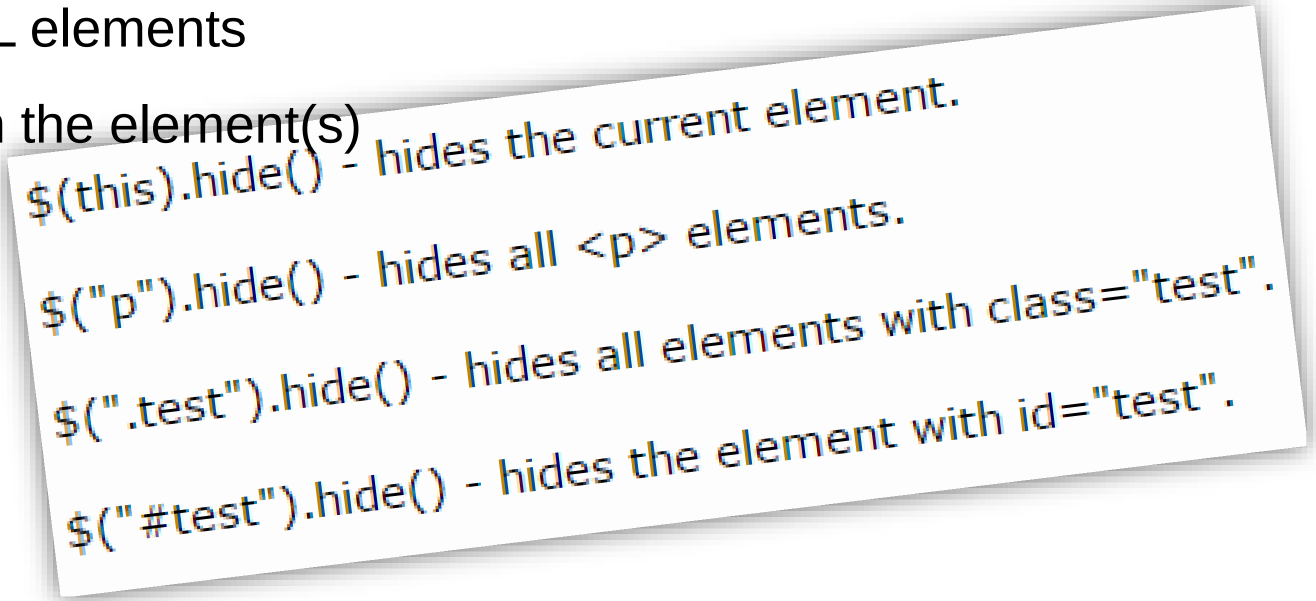
The jQuery syntax is made to select HTML elements and perform some action on the element(s).

The basic syntax is: **$(*selector*).*action*()**

A $ sign to set/access jQuery

A (selector) to "query (or find)" HTML elements

A jQuery action() to be performed on the element(s)



$(this).hide() - hides the current element.

$("p").hide() - hides all <p> elements.

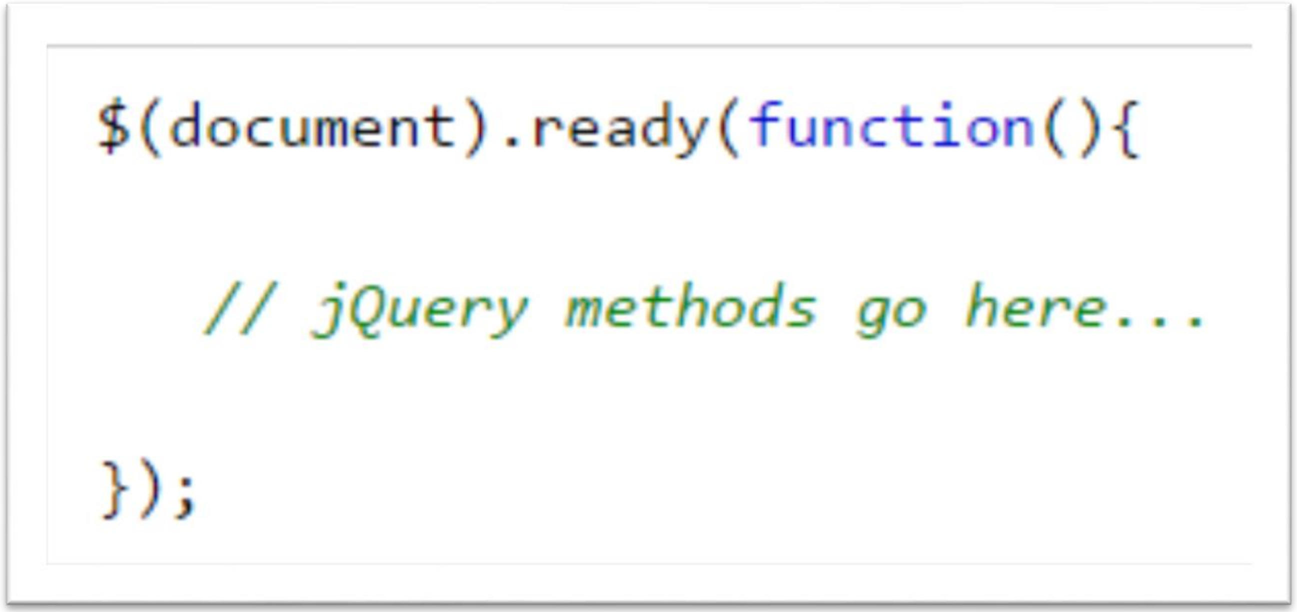
$(".test").hide() - hides all elements with class="test".

$("#test").hide() - hides the element with id="test".

JavaScript / jQuery

jQuery – “ready” event

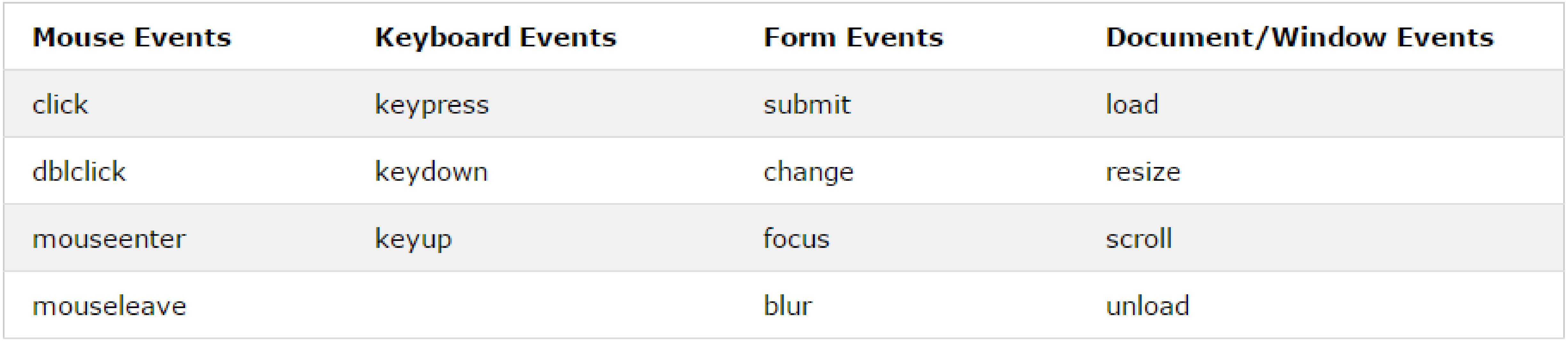
All jQuery methods are within the *ready* event to prevent any jQuery code from executing before the document finishes loading.



```
$(document).ready(function(){

   // jQuery methods go here...

});
```

JavaScript / jQuery

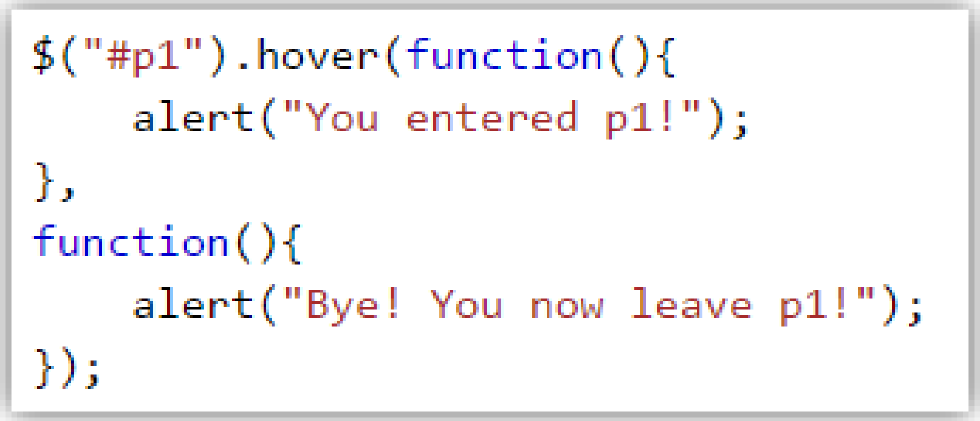
jQuery – common events



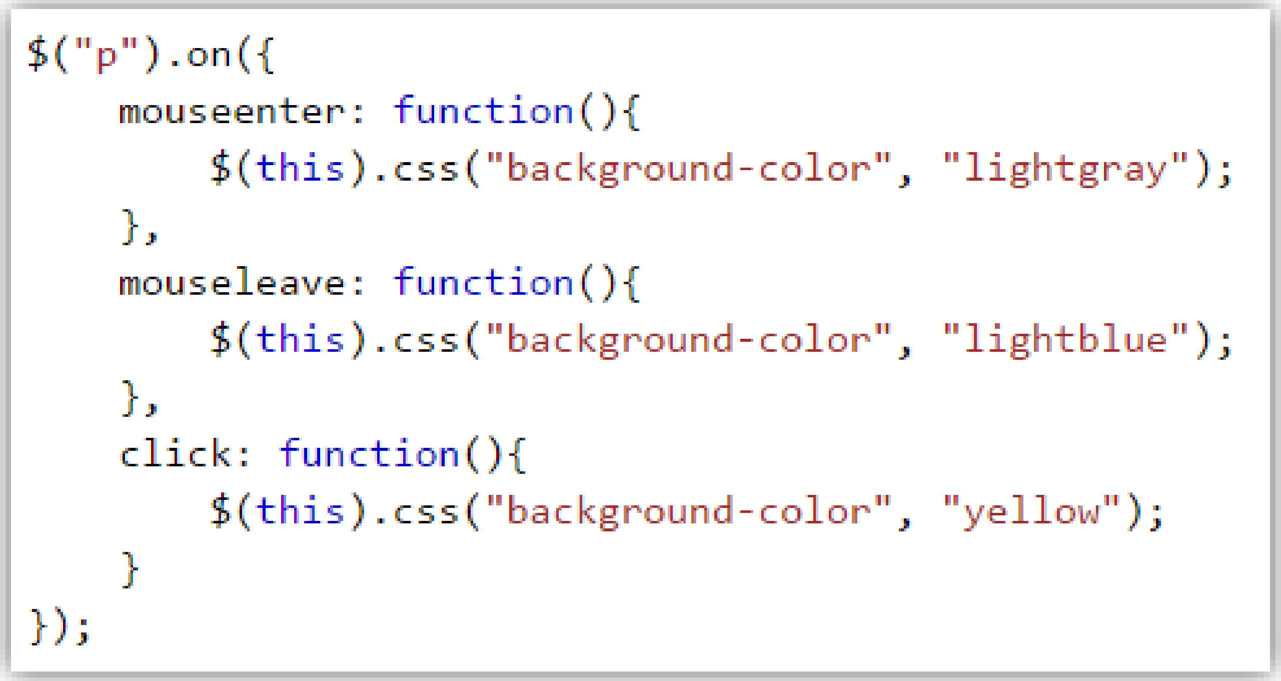
Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

JavaScript / jQuery

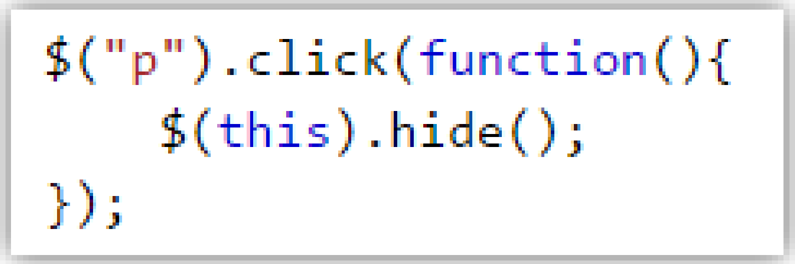
jQuery – examples



```
$("#p1").hover(function(){
    alert("You entered p1!");
},
function(){
    alert("Bye! You now leave p1!");
});
```



```
$("p").on({
    mouseenter: function(){
        $(this).css("background-color", "lightgray");
    },
    mouseleave: function(){
        $(this).css("background-color", "lightblue");
    },
    click: function(){
        $(this).css("background-color", "yellow");
    }
});
```



```
$("p").click(function(){
    $(this).hide();
});
```

JavaScript / jQuery

jQuery – make AJAX requests

$(*selector*).load(*URL,data,callback*);

$.get(*URL,callback*);

$.post(*URL,data,callback*);

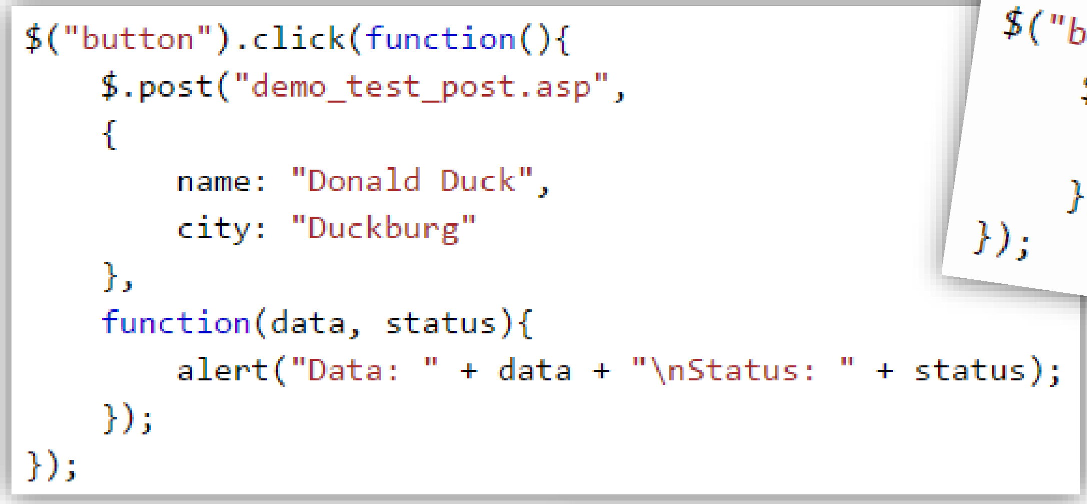
URL – destination of the request

data – additional data for the request (optional)

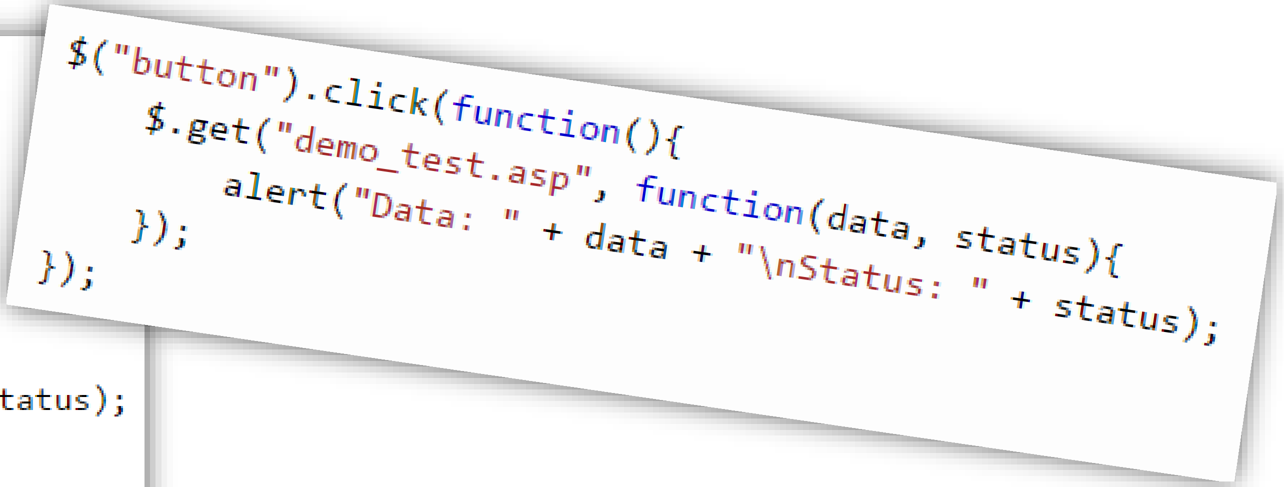
callback – routine to execute when request response arrives (optional)

JavaScript / jQuery

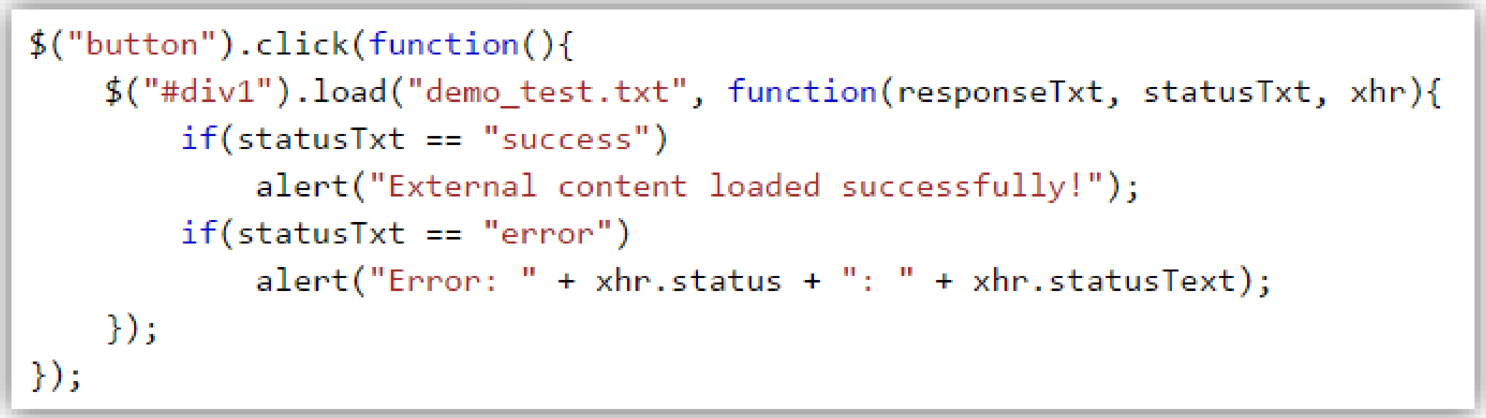
jQuery – make AJAX requests



```
$("button").click(function(){
    $.post("demo_test_post.asp",
    {
        name: "Donald Duck",
        city: "Duckburg"
    },
    function(data, status){
        alert("Data: " + data + "\nStatus: " + status);
    });
});
```



```
$("button").click(function(){
    $.get("demo_test.asp", function(data, status){
        alert("Data: " + data + "\nStatus: " + status);
    });
});
```



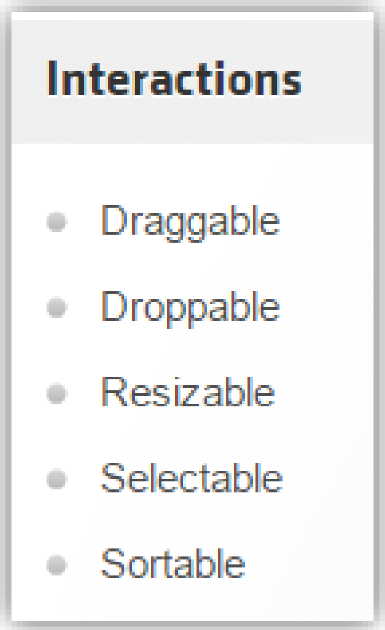
```
$("button").click(function(){
    $("#div1").load("demo_test.txt", function(responseTxt, statusTxt, xhr){
        if(statusTxt == "success")
            alert("External content loaded successfully!");
        if(statusTxt == "error")
            alert("Error: " + xhr.status + ": " + xhr.statusText);
    });
});
```

jQuery UI frameworks

They combine the use of jQuery, CSS and HTML5 for theme development, interaction, data entry and information representation components, effects and animations, among others.

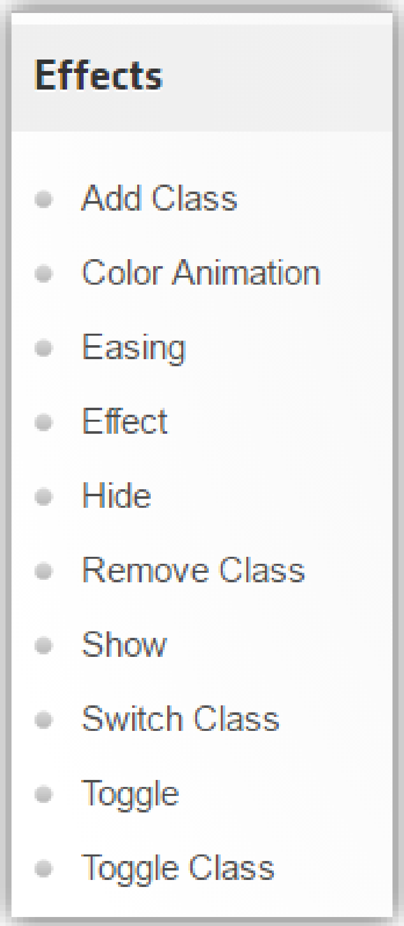
jQuery frameworks

Functionalities (jQuery UI):



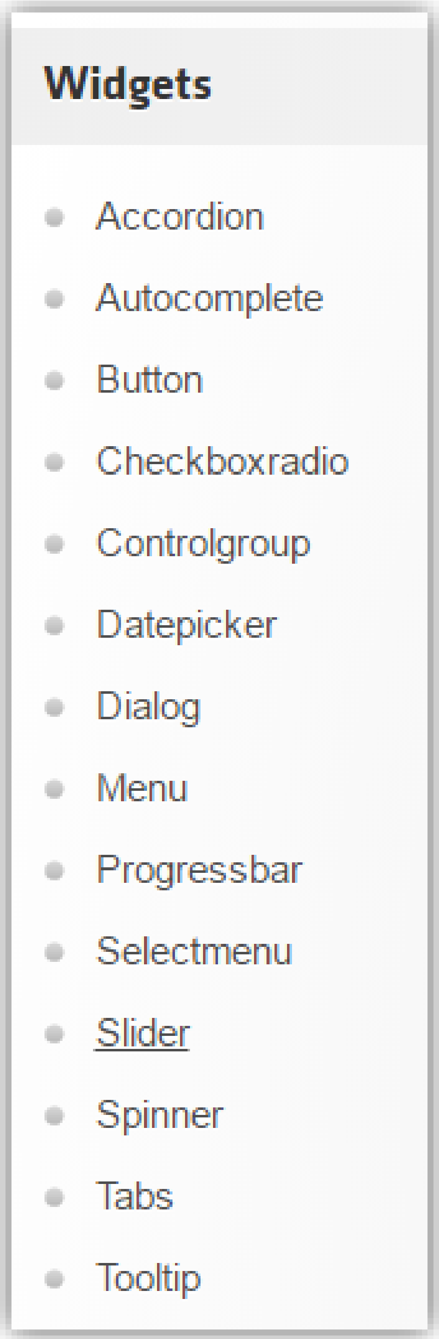
Interactions

- Draggable
- Droppable
- Resizable
- Selectable
- Sortable



Effects

- Add Class
- Color Animation
- Easing
- Effect
- Hide
- Remove Class
- Show
- Switch Class
- Toggle
- Toggle Class



Widgets

- Accordion
- Autocomplete
- Button
- Checkboxradio
- Controlgroup
- Datepicker
- Dialog
- Menu
- Progressbar
- Selectmenu
- Slider
- Spinner
- Tabs
- Tooltip

jQuery frameworks

Bootstrap

Foundation

JQuery UI

AngularJS

…

Bibliography

JavaScript

jQuery

jQuery user interface

w3schools

Google