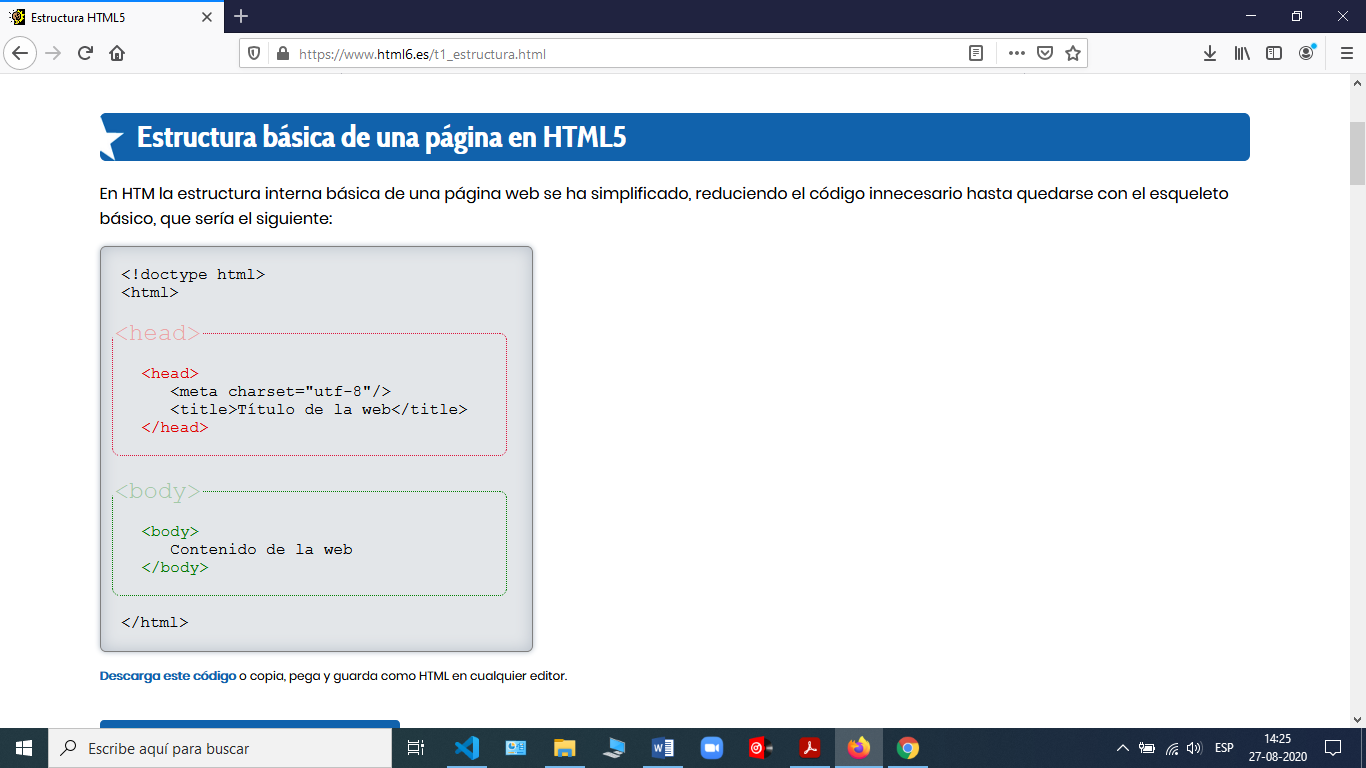
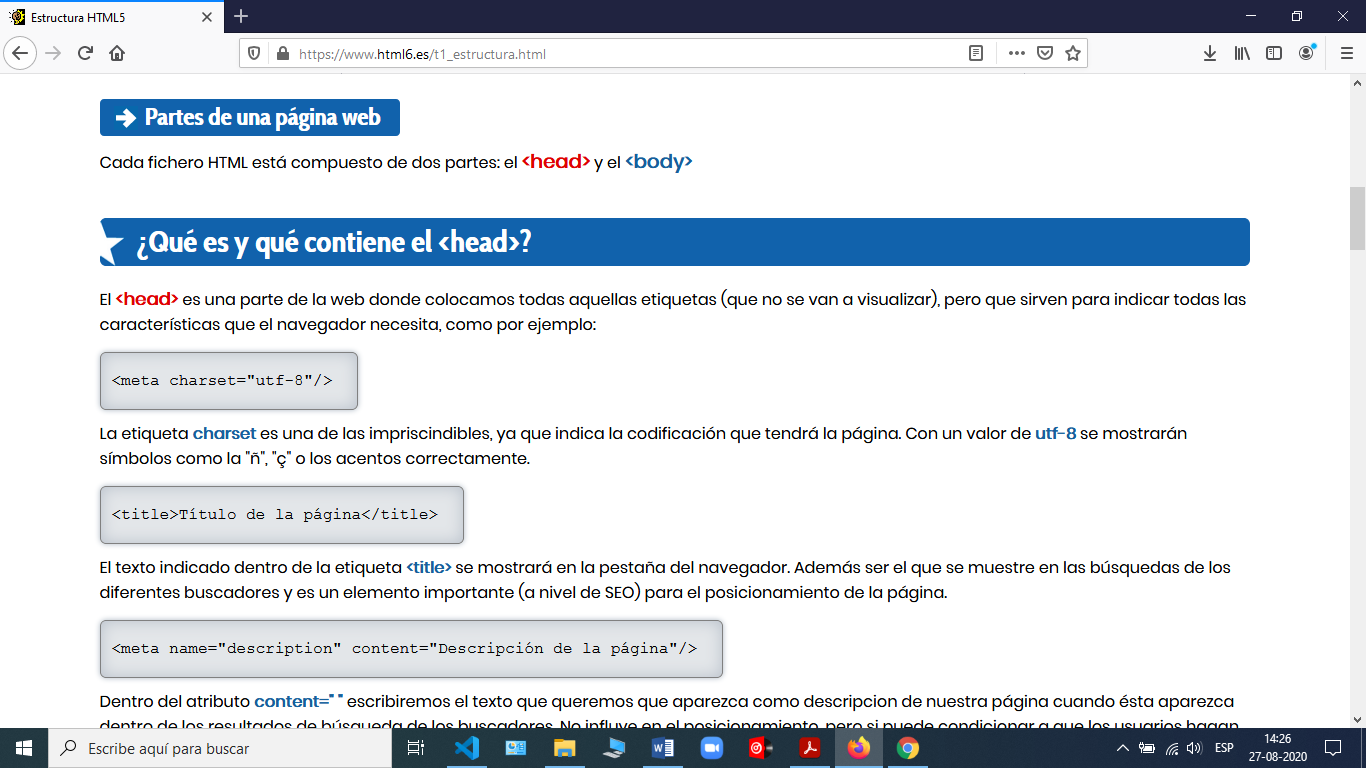
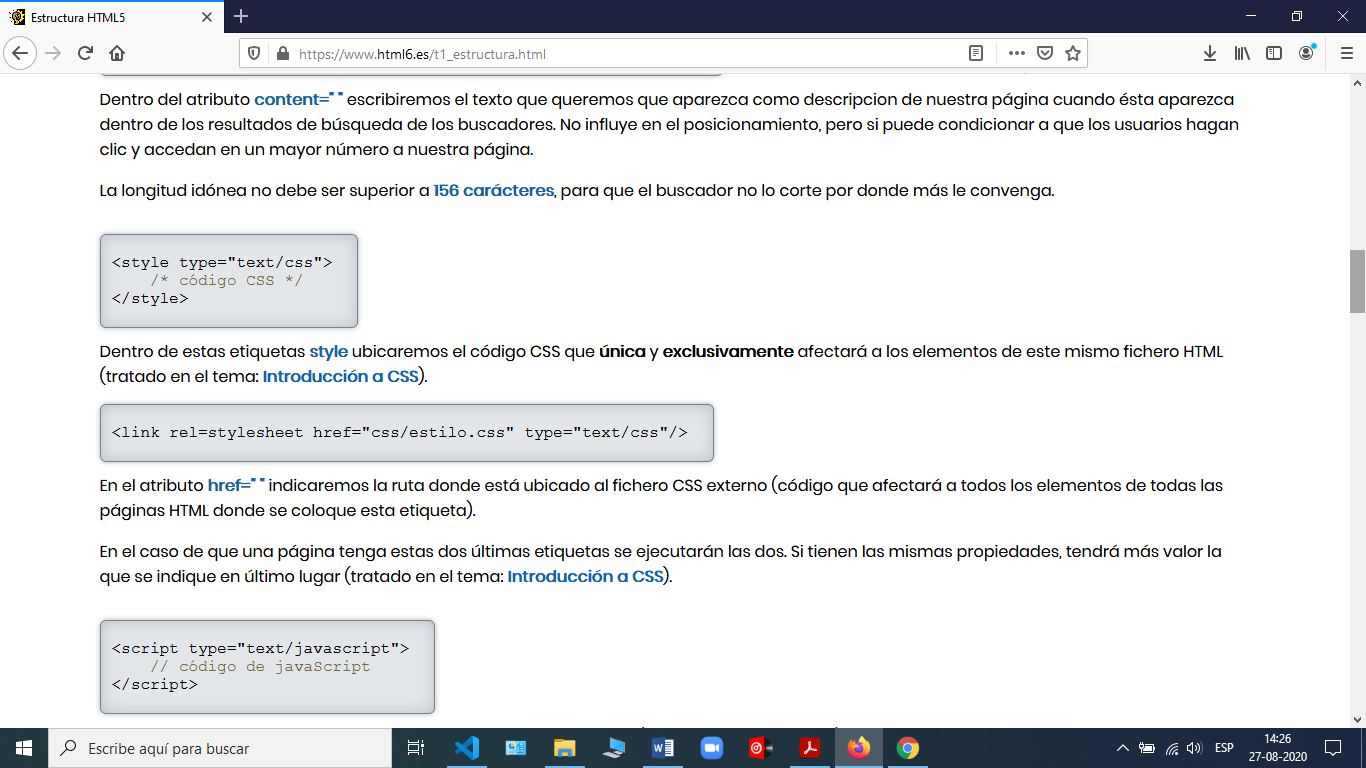
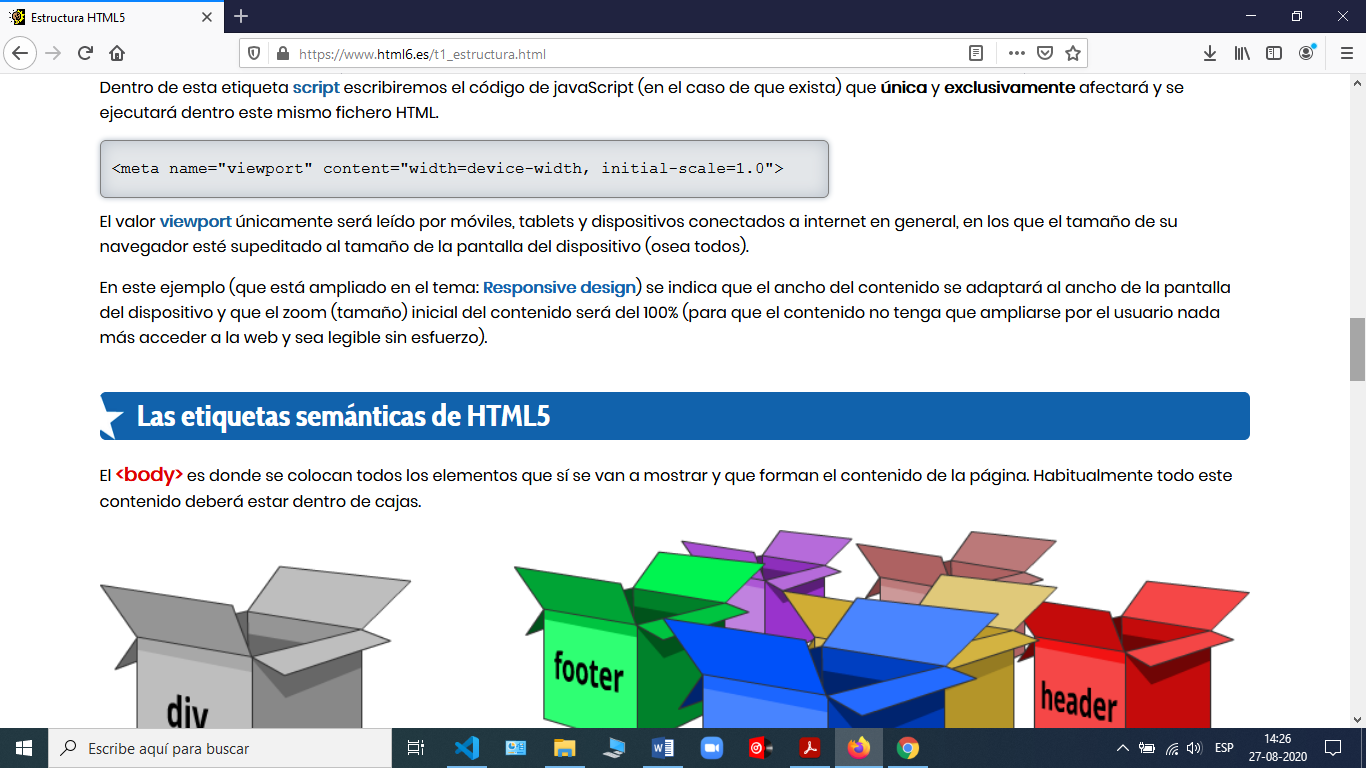
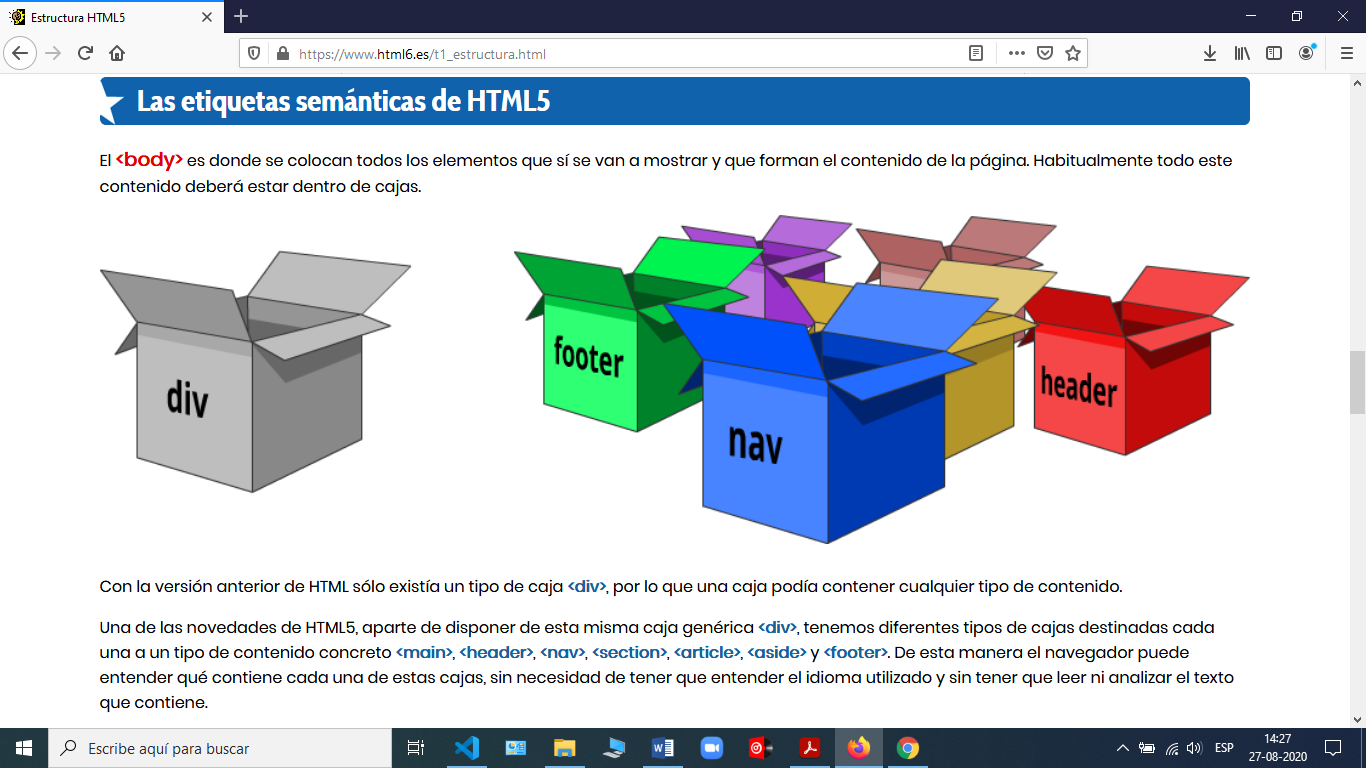
APUNTES ESTRUCTURA O MAQUETACION HTML 5

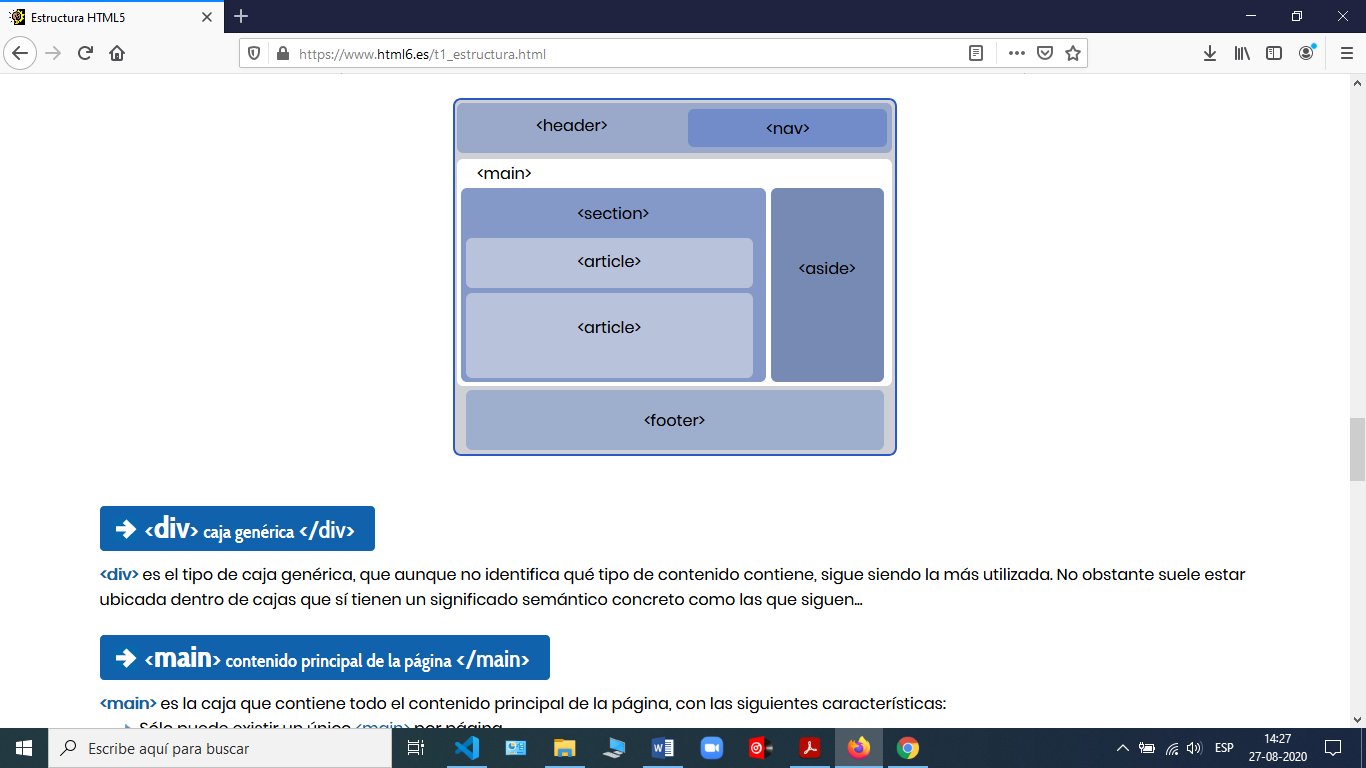


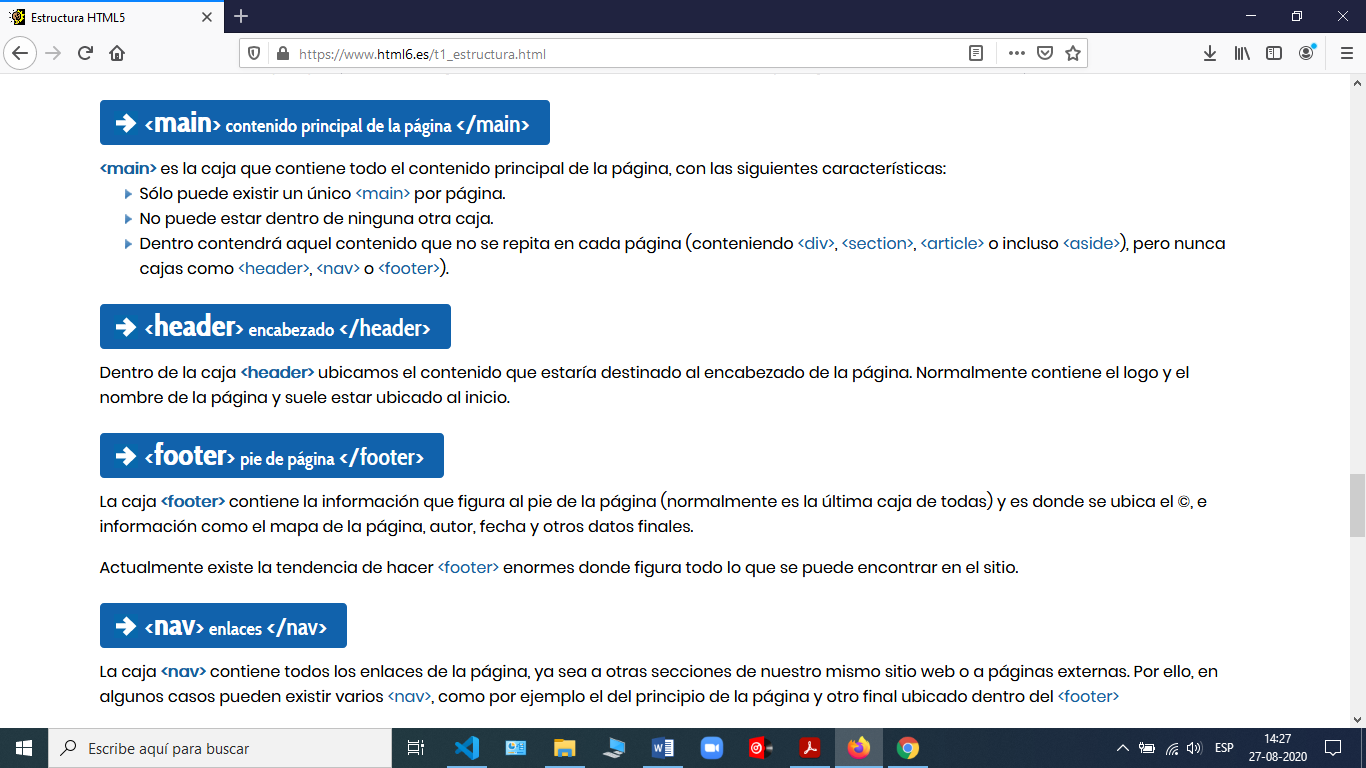


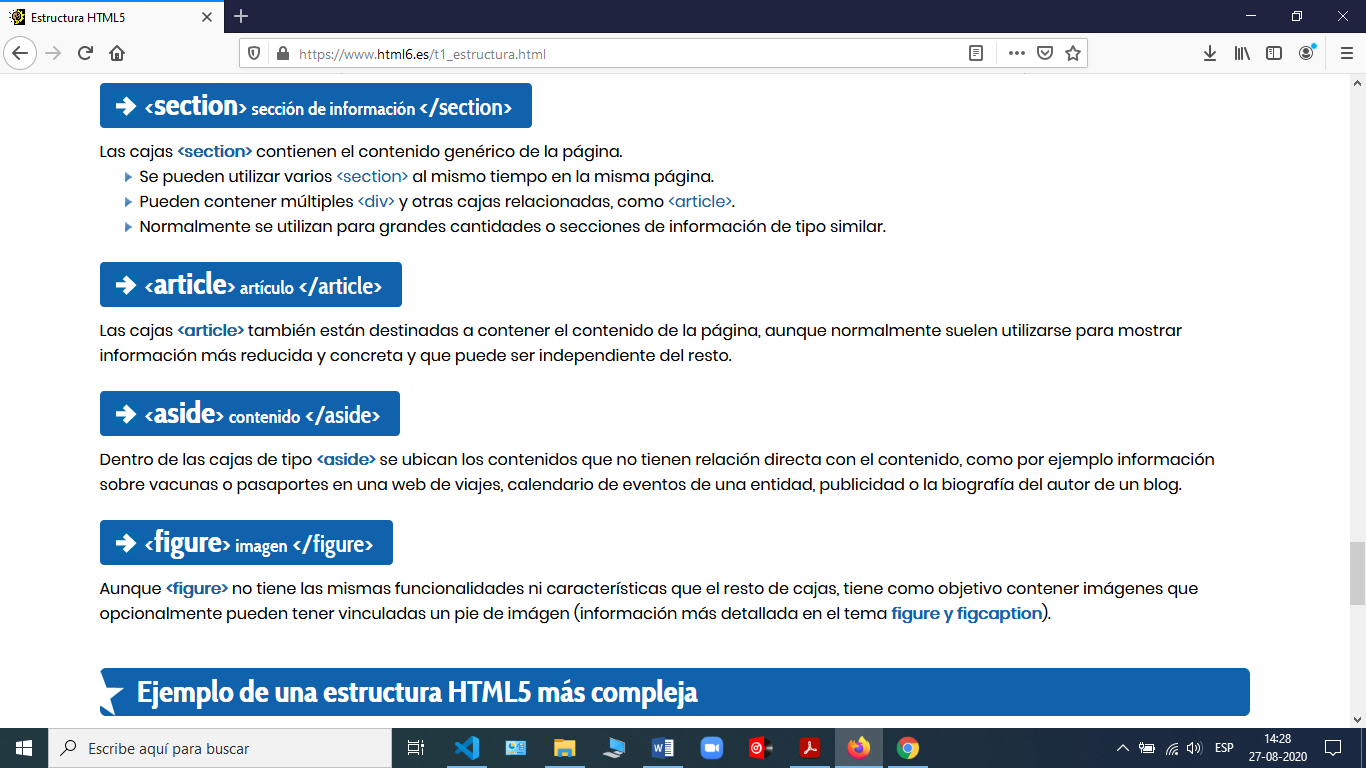


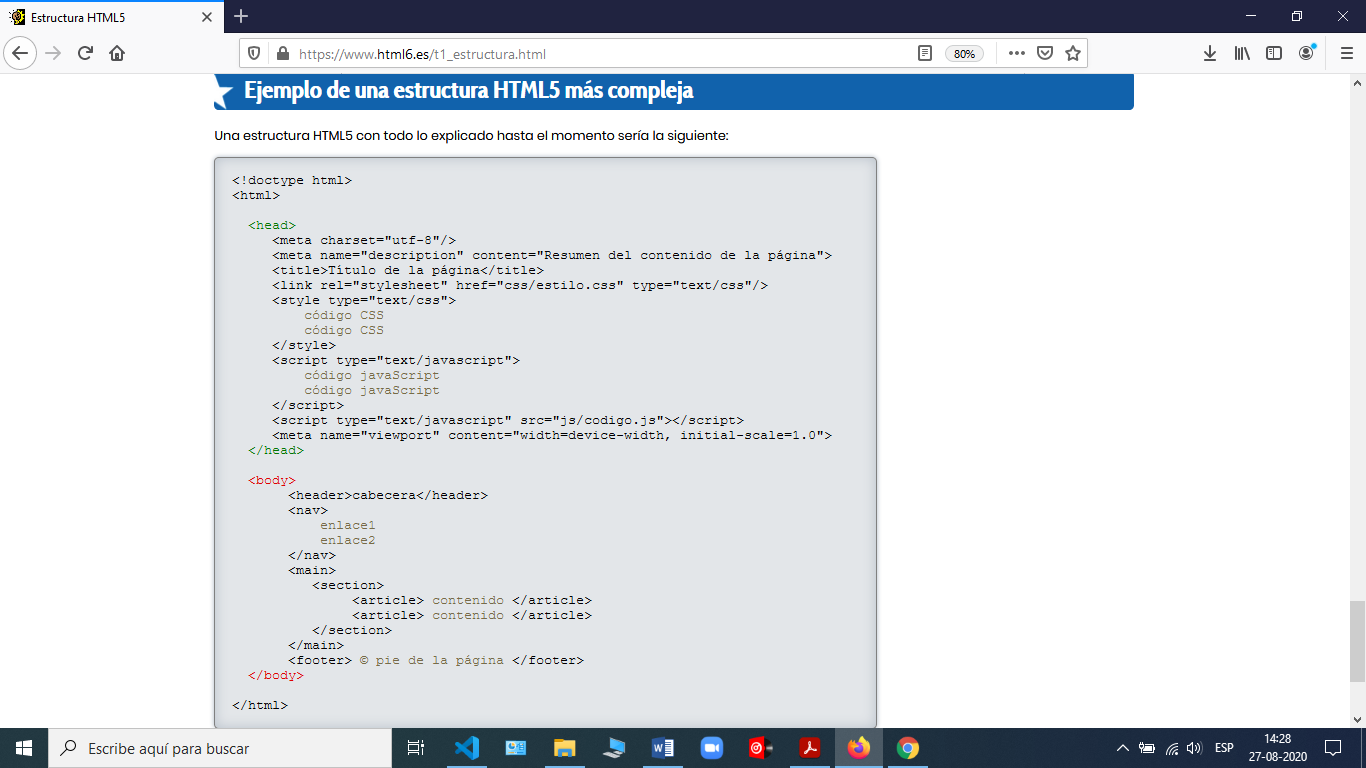












Una estructura HTML5 con todo lo explicado hasta el momento sería la siguiente:

<!doctype html>

<html>

<head>

<meta charset="utf-8"/>

<meta name="description" content="Resumen del contenido de la página">

<title>Título de la página</title>

<link rel="stylesheet" href="css/estilo.css" type="text/css"/>

<style type="text/css">

código CSS

código CSS

</style>

<script type="text/javascript">

código javaScript

código javaScript

</script>

<script type="text/javascript" src="js/codigo.js"></script>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<header>cabecera</header>

<nav>

enlace1

enlace2

</nav>

<main>

<section>

<article> contenido </article>

<article> contenido </article>

</section>

</main>

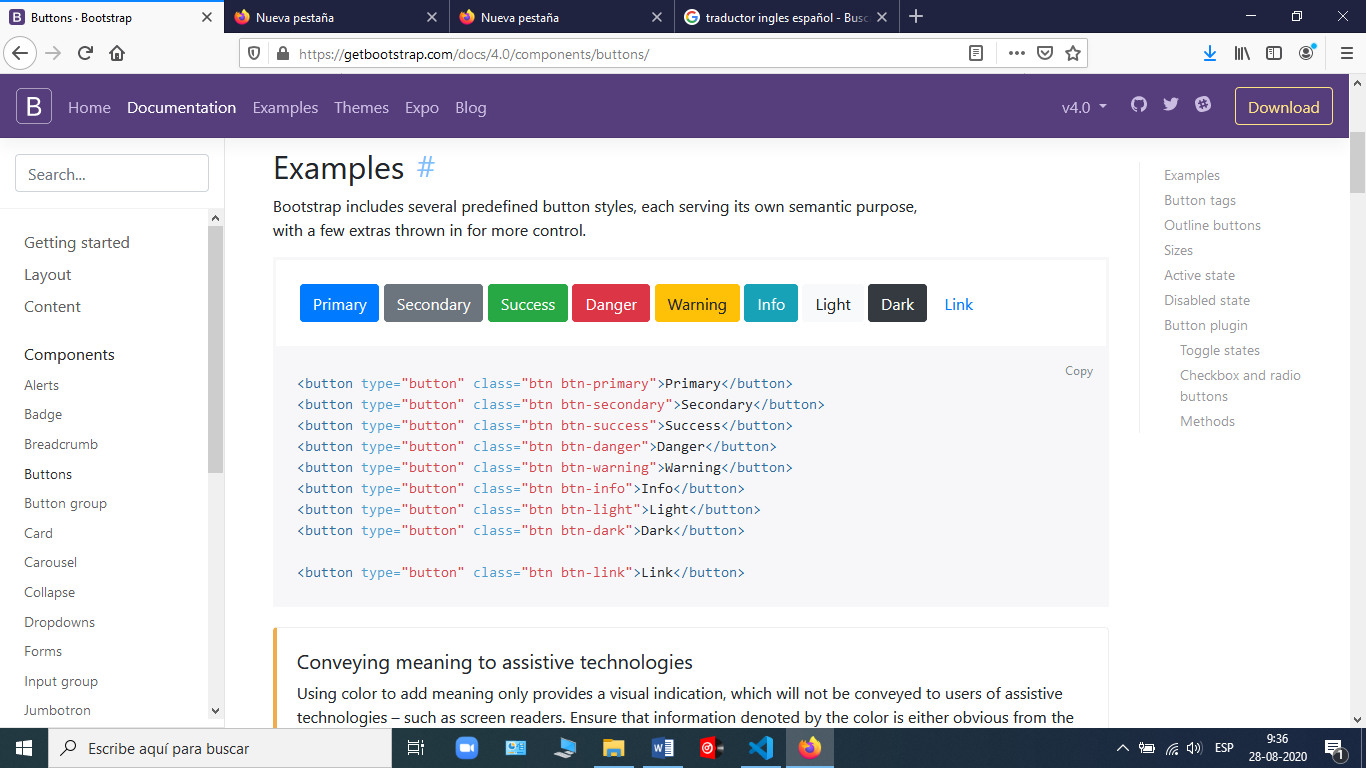
<footer> © pie de la página </footer>

</body>

</html>

USO DE BOTONES PREDEFINIDOS DE BOOTSRTAP

Bootstrap incluye varios estilos de botones predefinidos, cada uno de los cuales tiene su propio propósito semántico, con algunos extras para un mayor control.



<button type="button" class="btn btn-primary">Primary</button>

<button type="button" class="btn btn-secondary">Secondary</button>

<button type="button" class="btn btn-success">Success</button>

<button type="button" class="btn btn-danger">Danger</button>

<button type="button" class="btn btn-warning">Warning</button>

<button type="button" class="btn btn-info">Info</button>

<button type="button" class="btn btn-light">Light</button>

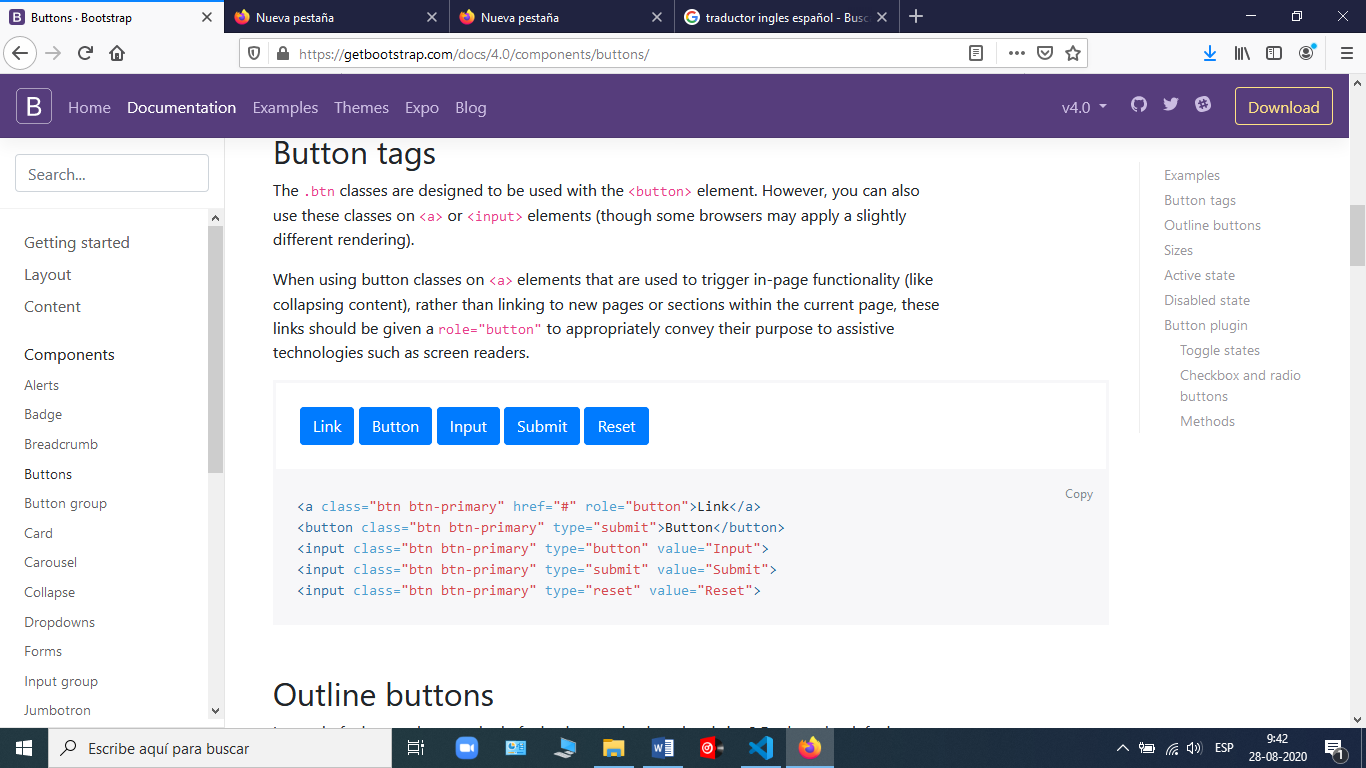
<button type="button" class="btn btn-dark">Dark</button>

<button type="button" class="btn btn-link">Link</button>

**Button tags**

Las clases .btn están diseñadas para usarse con el elemento <button>. Sin embargo, también puede usar estas clases en elementos <a> o <input> (aunque algunos navegadores pueden aplicar una representación ligeramente diferente).

Cuando se utilizan clases de botones en elementos <a> que se utilizan para activar la funcionalidad en la página (como el contenido que se contrae), en lugar de vincular a nuevas páginas o secciones dentro de la página actual, estos vínculos deben tener un rol = "botón" para transmitir su propósito a las tecnologías de asistencia, como los lectores de pantalla.



<a class="btn btn-primary" href="#" role="button">Link</a>

<button class="btn btn-primary" type="submit">Button</button>

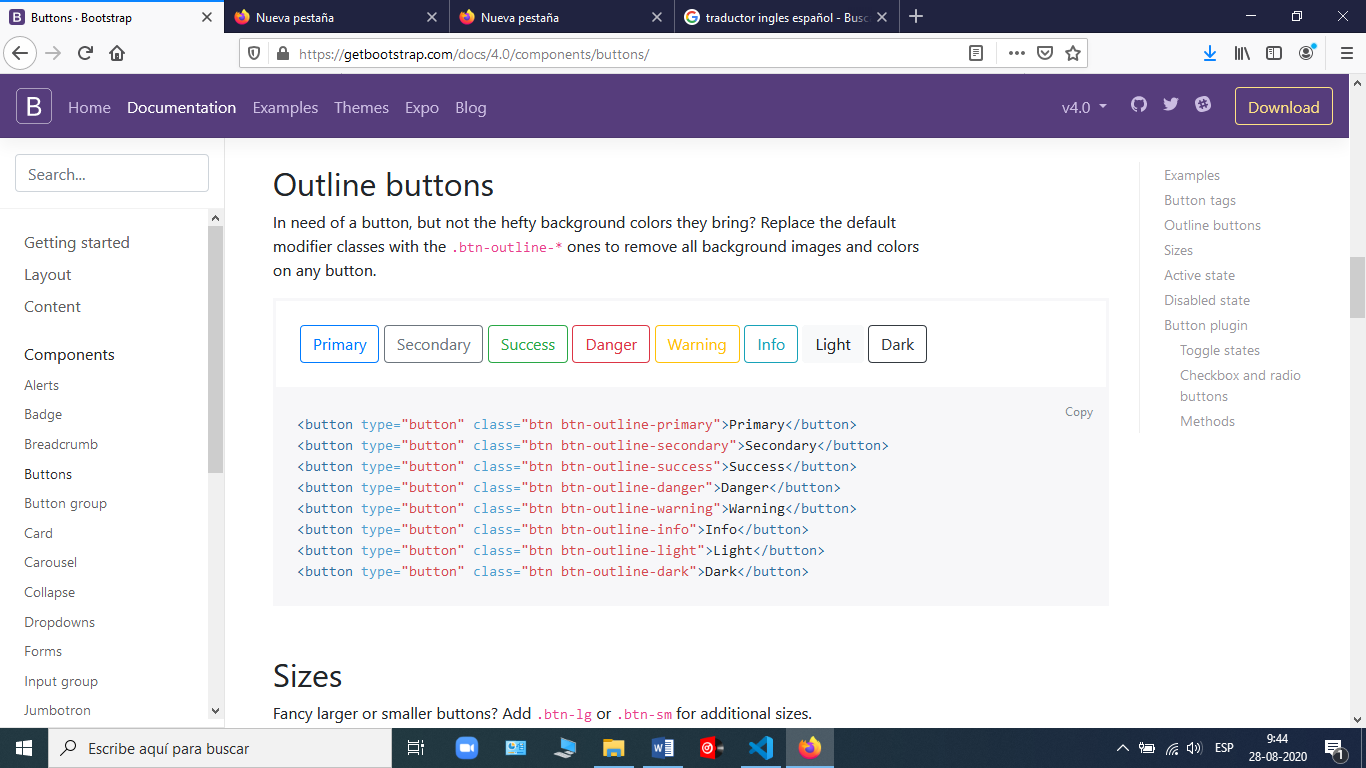
<input class="btn btn-primary" type="button" value="Input">

<input class="btn btn-primary" type="submit" value="Submit">

<input class="btn btn-primary" type="reset" value="Reset">

## Outline buttons

¿Necesitas un botón, pero no los colores de fondo que traen? Reemplace las clases de modificador predeterminadas con las .btn-outline- \* para eliminar todas las imágenes y colores de fondo en cualquier botón.



<button type="button" class="btn btn-outline-primary">Primary</button>

<button type="button" class="btn btn-outline-secondary">Secondary</button>

<button type="button" class="btn btn-outline-success">Success</button>

<button type="button" class="btn btn-outline-danger">Danger</button>

<button type="button" class="btn btn-outline-warning">Warning</button>

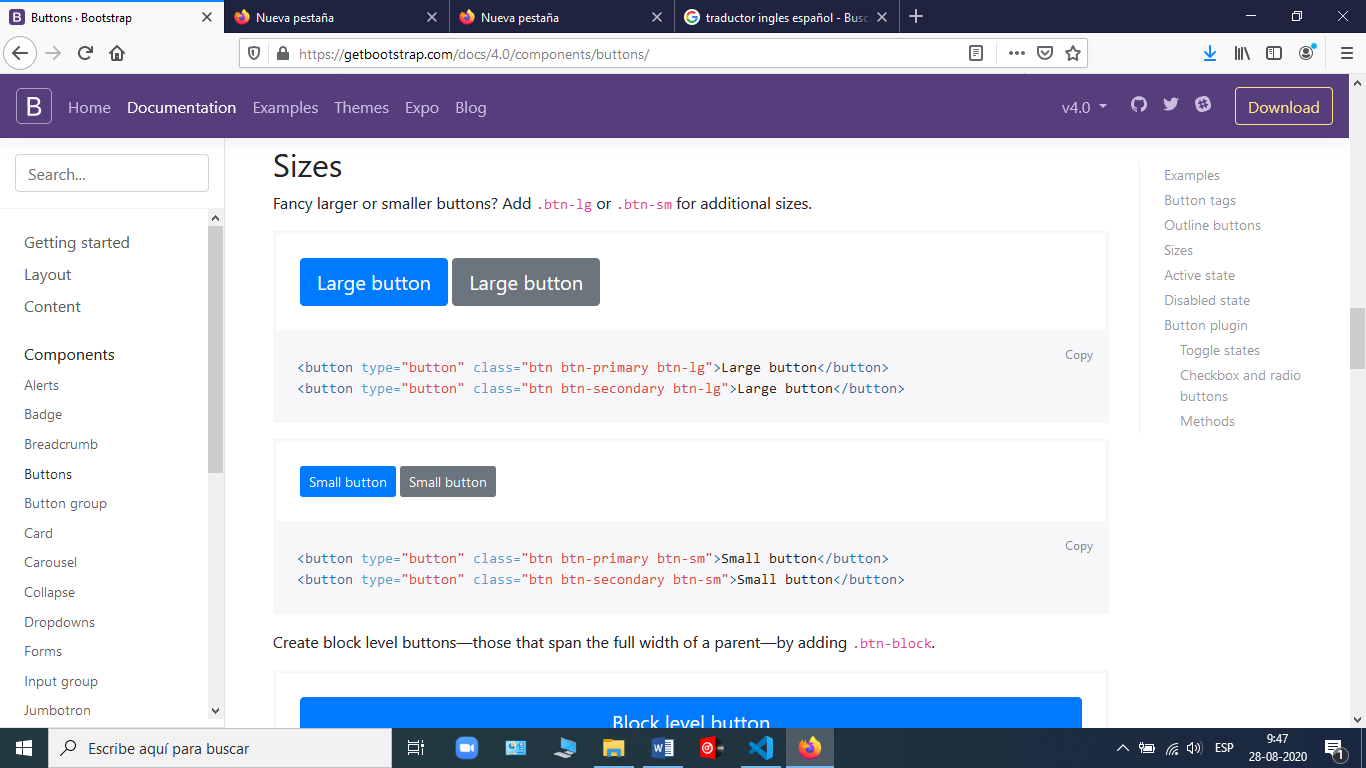
<button type="button" class="btn btn-outline-info">Info</button>

<button type="button" class="btn btn-outline-light">Light</button>

<button type="button" class="btn btn-outline-dark">Dark</button>

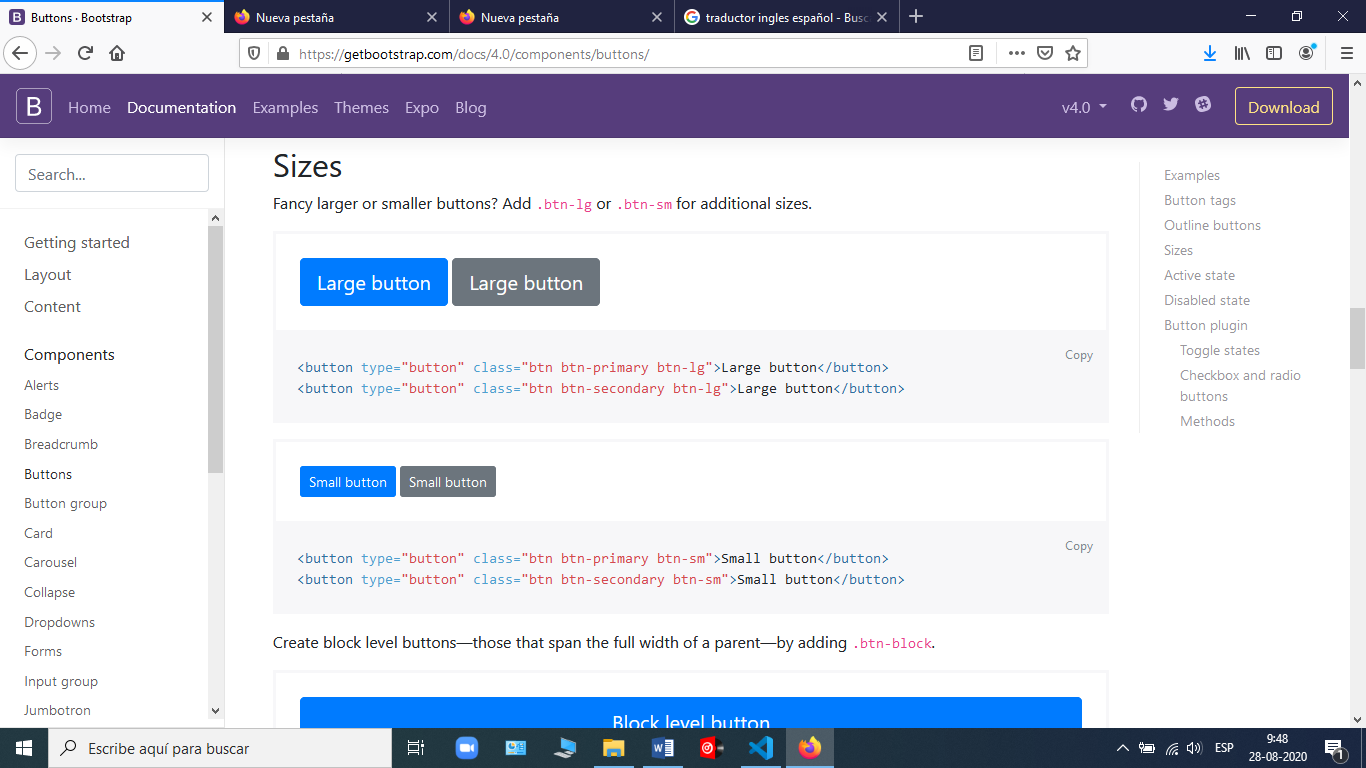
## Sizes

Podrias cambiar el tamaño del botton



<button type="button" class="btn btn-primary btn-lg">Large button</button>

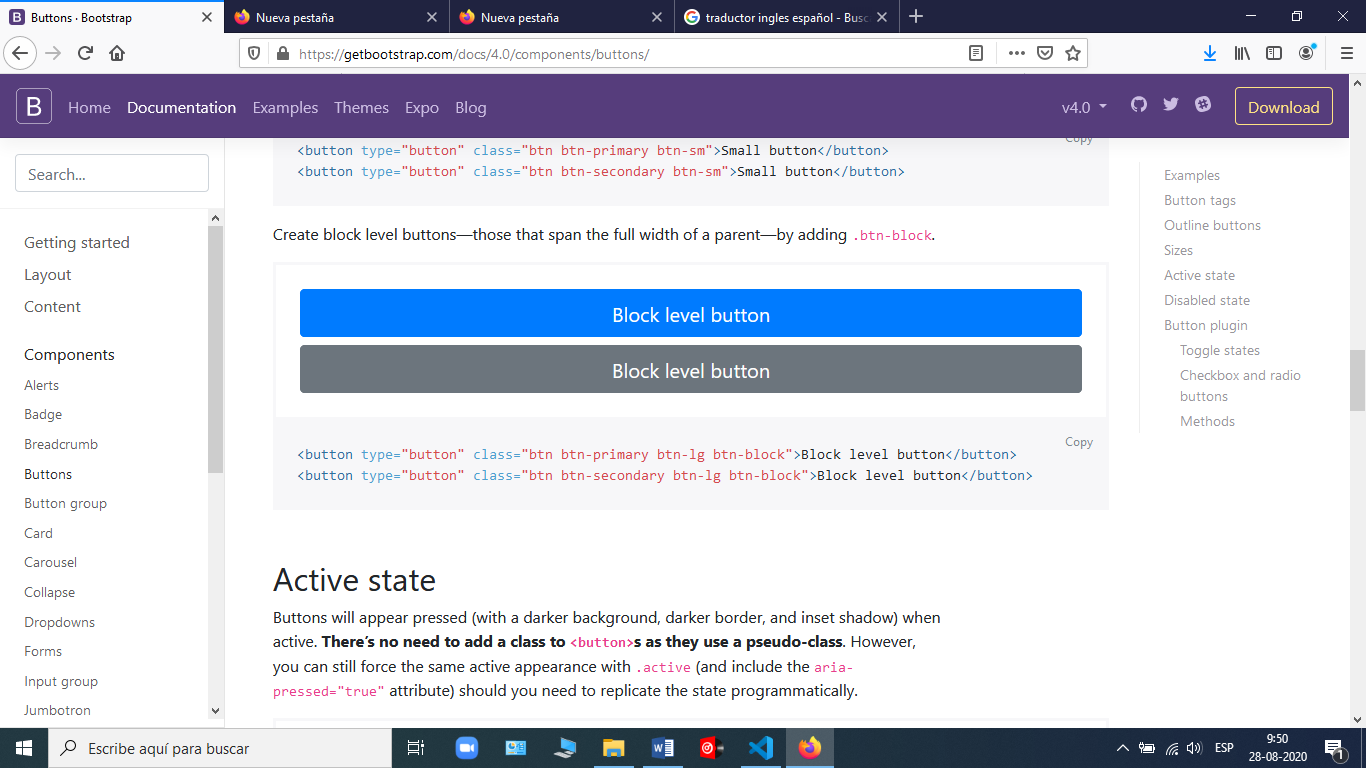
<button type="button" class="btn btn-secondary btn-lg">Large button</button>



<button type="button" class="btn btn-primary btn-sm">Small button</button>

<button type="button" class="btn btn-secondary btn-sm">Small button</button>

Cree botones a nivel de bloque, los que abarcan todo el ancho de un padre, agregando .btn-block.



<button type="button" class="btn btn-primary btn-lg btn-block">Block level button</button>

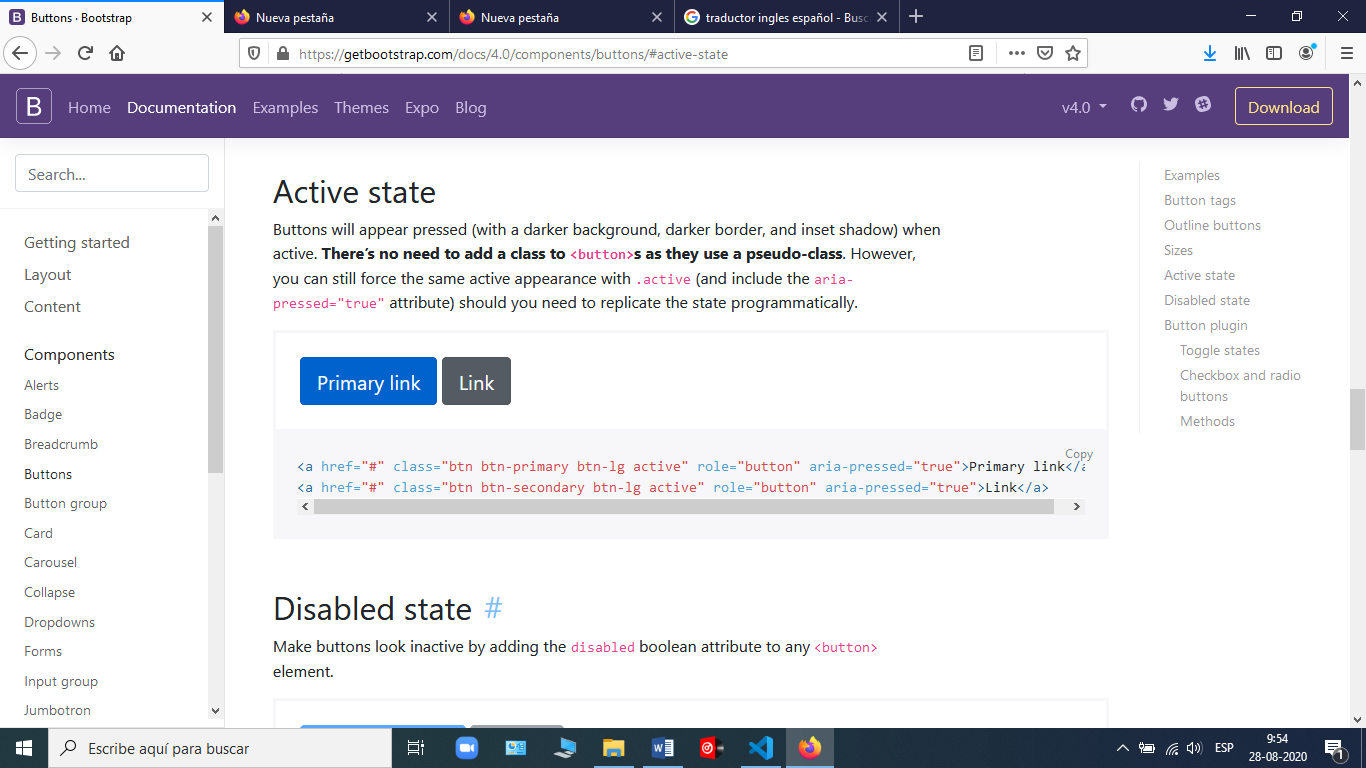
<button type="button" class="btn btn-secondary btn-lg btn-block">Block level button</button>

## Active state

Los botones aparecerán presionados (con un fondo más oscuro, un borde más oscuro y una sombra insertada) cuando estén activos. No es necesario agregar una clase a los <button> s ya que usan una pseudoclase.

Sin embargo, aún puede forzar la misma apariencia activa con .active

(and include the aria-pressed="true" attribute) si necesita replicar el estado mediante programación.

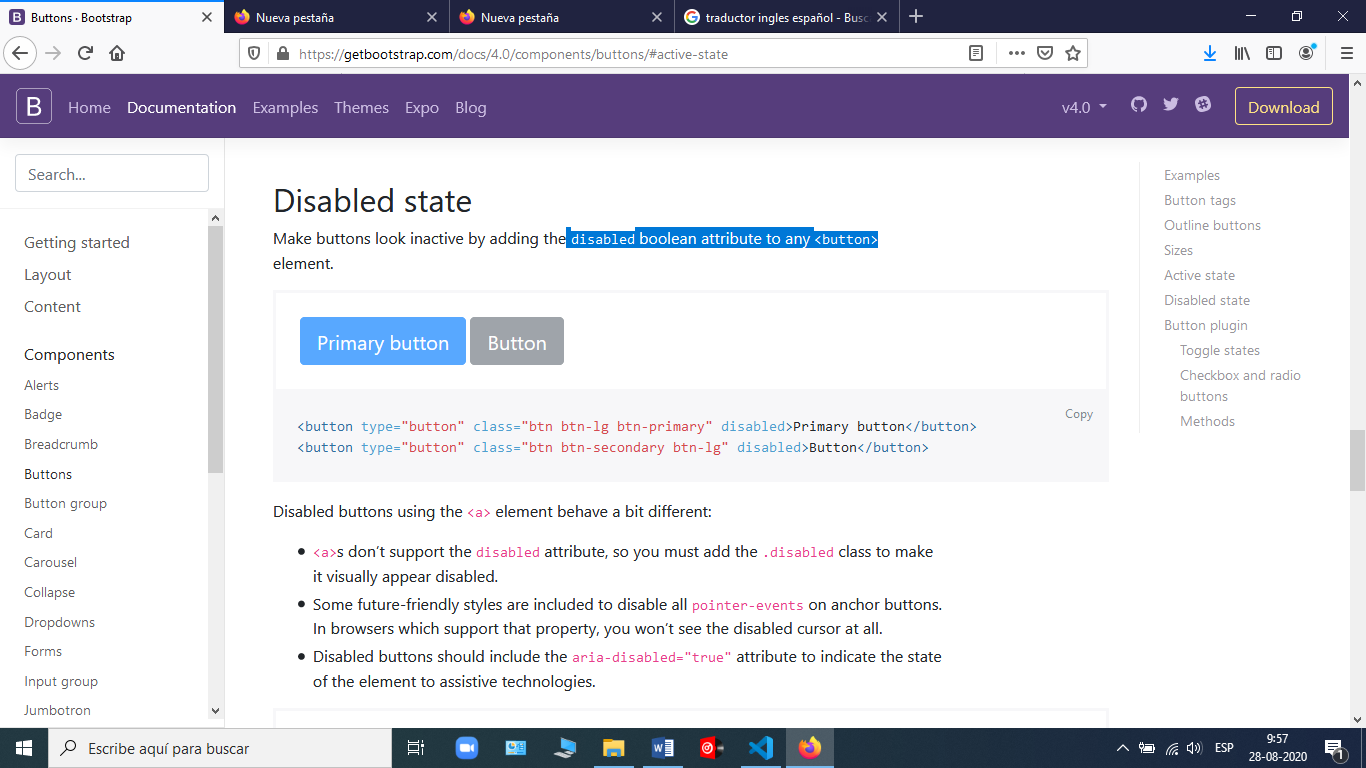


<a href="#" class="btn btn-primary btn-lg active" role="button" aria-pressed="true">Primary link</a>

<a href="#" class="btn btn-secondary btn-lg active" role="button" aria-pressed="true">Link</a>

## Disabled state

Hacer que los botones se vean inactivos agregando la disabled boolean attribute to any <button>



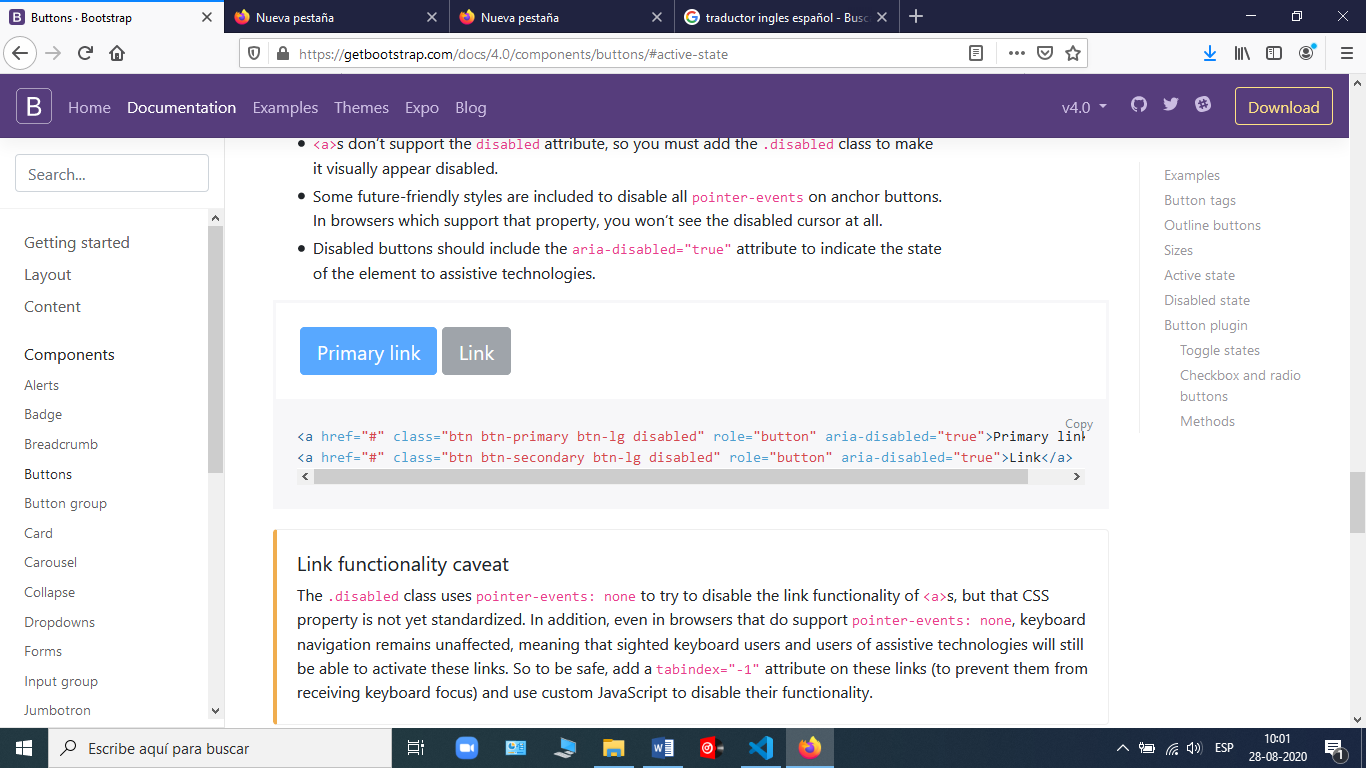
<button type="button" class="btn btn-lg btn-primary" disabled>Primary button</button>

<button type="button" class="btn btn-secondary btn-lg" disabled>Button</button>

Notas:

<a> no admiten el atributo disabled, por lo que debe agregar la clase .disabled para que aparezca visualmente deshabilitado.

Los botones deshabilitados deben incluir el atributo aria-disabled="true" atributo para indicar el estado del elemento a las tecnologías de asistencia.



<a href="#" class="btn btn-primary btn-lg disabled" role="button" aria-disabled="true">Primary link</a>

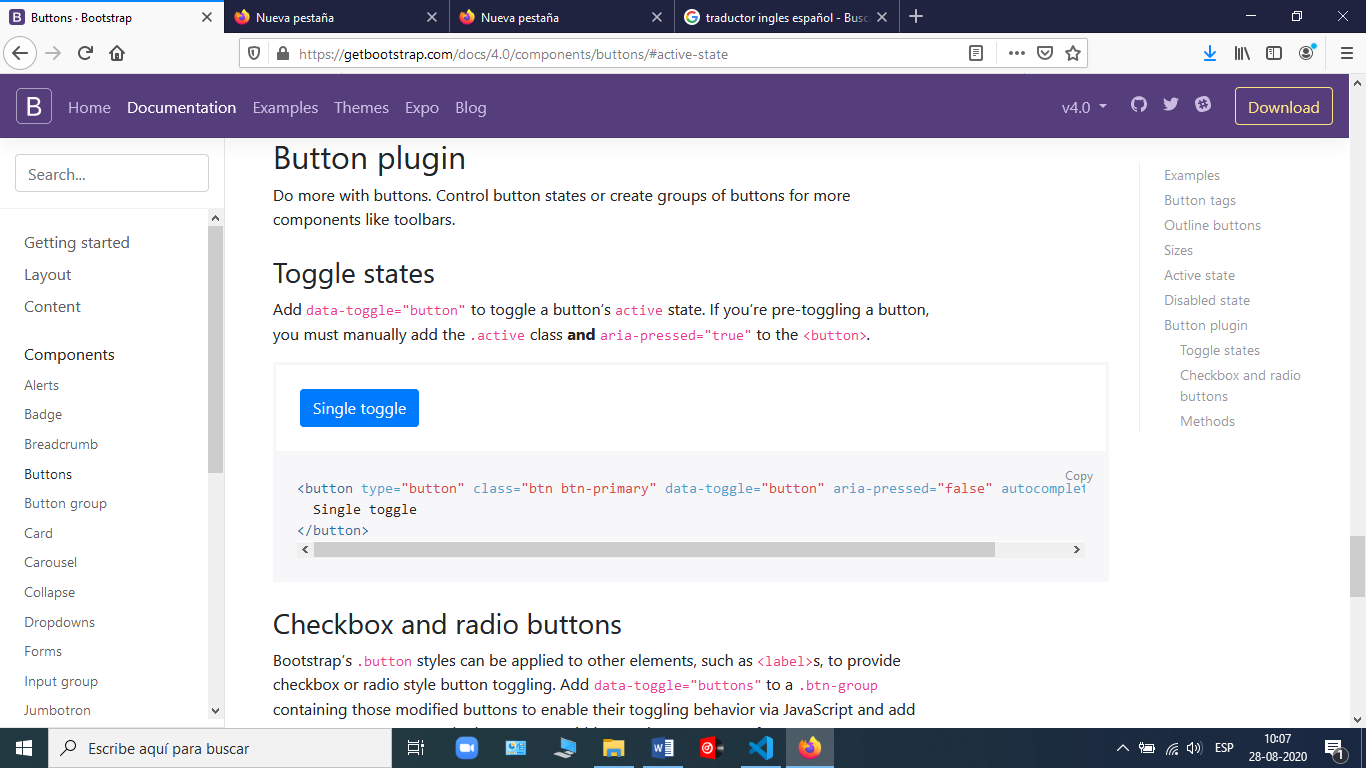
<a href="#" class="btn btn-secondary btn-lg disabled" role="button" aria-disabled="true">Link</a>

## Button plugin

Haz más con los botones. Controle los estados de los botones o cree grupos de botones para más componentes como barras de herramientas.

## Togle States

**Agrega** data-toggle="button" para alternar un botón. Si está activando previamente un botón, debe agregar manualmente el .active class **and** aria-pressed="true" to the <button>.



<button type="button" class="btn btn-primary" data-toggle="button" aria-pressed="false" autocomplete="off">

Single toggle

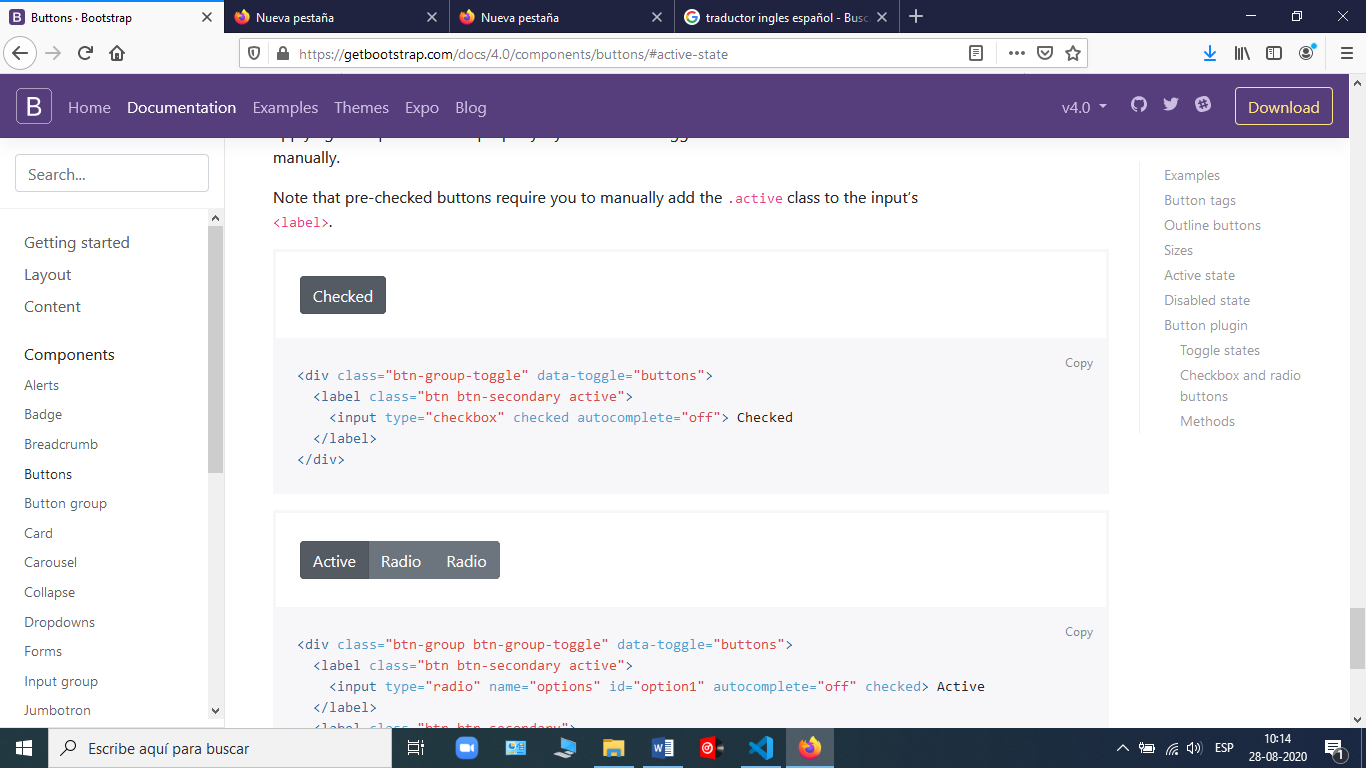
</button>

### **Checkbox and radio buttons**

Los estilos .button de Bootstrap se pueden aplicar a otros elementos, como etiquetas, para proporcionar la alternancia de botones de casilla de verificación o estilo de radio. Agrega data-toggle="buttons" to a .btn-group que contiene esos botones modificados para habilitar su comportamiento de alternancia a través de JavaScript y agregar .btn-group-toggle to style the <input>s dentro de tus botones.

Tenga en cuenta que puede crear botones de entrada única o grupos de ellos.

Tenga en cuenta que los botones previamente marcados requieren que agregue manualmente el .active class to the input’s <label>.



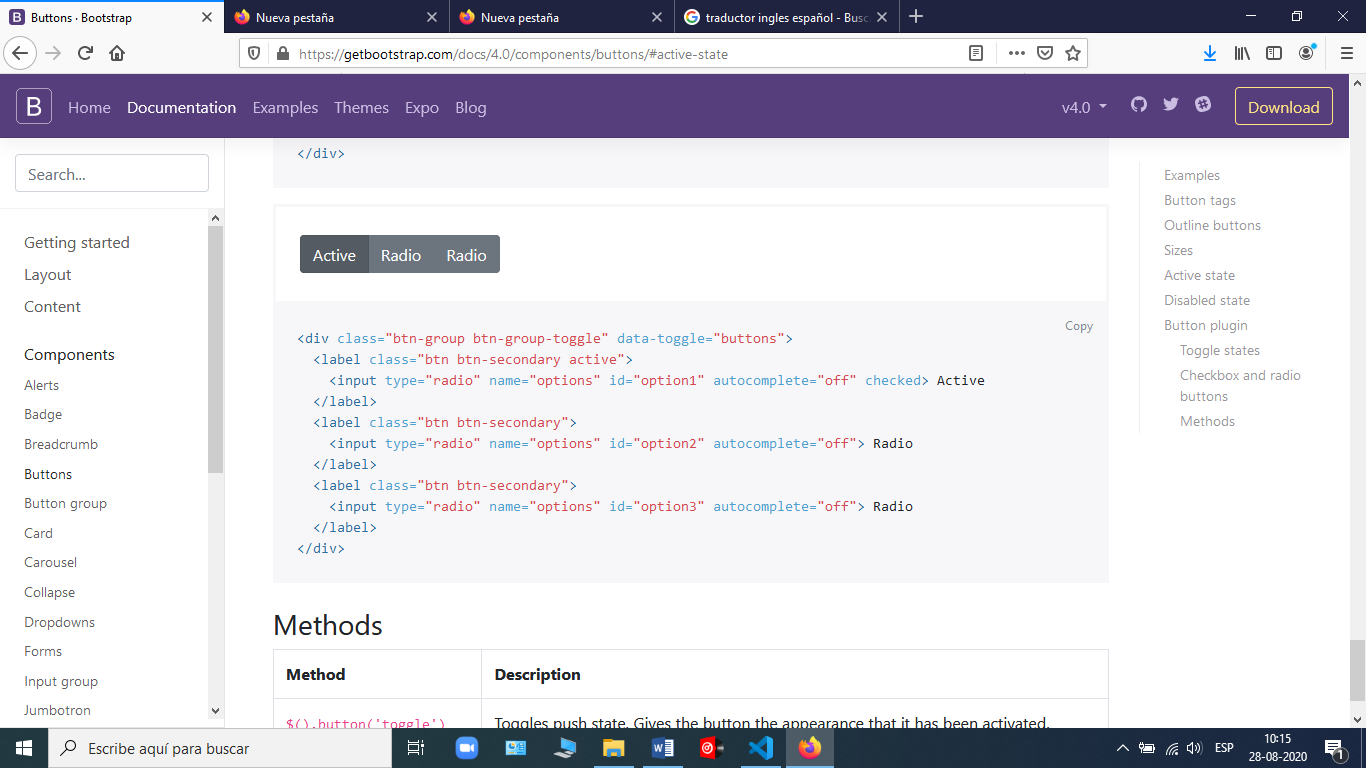
<div class="btn-group-toggle" data-toggle="buttons">

<label class="btn btn-secondary active">

<input type="checkbox" checked autocomplete="off"> Checked

</label>

</div>



<div class="btn-group btn-group-toggle" data-toggle="buttons">

<label class="btn btn-secondary active">

<input type="radio" name="options" id="option1" autocomplete="off" checked> Active

</label>

<label class="btn btn-secondary">

<input type="radio" name="options" id="option2" autocomplete="off"> Radio

</label>

<label class="btn btn-secondary">

<input type="radio" name="options" id="option3" autocomplete="off"> Radio

</label>

</div>

### Methods

$().button('toggle') : Da al botón la apariencia de que ha sido activado.

$().button('dispose') : Destruye el botón de un elemento.

## Introducción

Este artículo es un resumen de algunos métodos DOM nivel 1 poderosos y fundamentales así como una descripción de cómo utilizarlos utilizando Javascript.  Aprenderás a crear, accesar, controlar, y remover elementos HTML dinámicamente.  Los métodos DOM presentados aquí no son específicos de HTML; también aplican para XML. Las demostraciones aquí proporcionadas funcionarán en cualquier navegador moderno, incluyendo todas las versiones de Firefox e IE 5+.

## Ejemplo: Crear una tabla HTML dinámicamente (Ejemplo1.html)

### Contenido HTML

<input type="button" value="Genera una tabla" onclick="genera\_tabla()">

### JavaScript Content

function genera\_tabla() {

// Obtener la referencia del elemento body

var body = document.getElementsByTagName("body")[0];

// Crea un elemento <table> y un elemento <tbody>

var tabla = document.createElement("table");

var tblBody = document.createElement("tbody");

// Crea las celdas

for (var i = 0; i < 2; i++) {

// Crea las hileras de la tabla

var hilera = document.createElement("tr");

for (var j = 0; j < 2; j++) {

// Crea un elemento <td> y un nodo de texto, haz que el nodo de

// texto sea el contenido de <td>, ubica el elemento <td> al final

// de la hilera de la tabla

var celda = document.createElement("td");

var textoCelda = document.createTextNode("celda en la hilera "+i+", columna "+j);

celda.appendChild(textoCelda);

hilera.appendChild(celda);

}

// agrega la hilera al final de la tabla (al final del elemento tblbody)

tblBody.appendChild(hilera);

}

// posiciona el <tbody> debajo del elemento <table>

tabla.appendChild(tblBody);

// appends <table> into <body>

body.appendChild(tabla);

// modifica el atributo "border" de la tabla y lo fija a "2";

tabla.setAttribute("border", "2");

}

Observa cuidadosamente el orden en el que se crearon los elementos en el nodo de texto:

1. Primero se crea el elemento <table>.
2. Posteriormente, creamos el elemento <tbody> , que es el hijo del elemento <table> .
3. Después, utilizamos ciclos para crear los elementos <tr>, que son hijos del elemento <tbody>.
4. Para cada elemento <tr>, utilizamos nuevamente ciclos para generar los elementos <td> que son hijos de los elementos <tr>.
5. Para cada elemento <td>, creamos nodos de texto con el contenido de cada celda.

Una vez creados los elementos <table>, <tbody>, <tr>, y <td> así como los nodos de texto, adicionamos a cada hijo bajo su padre en el órden opuesto:

1 Primero, anexamos cada nodo de texto a su elemento padre <td> :

celda.appendChild(textoCelda);

2 . Posteriormente, anexamos cada elemento <td> a su elemento padre <tr> :

hilera.appendChild(celda);

3 Posteriomente, anexamos cada elemento <tr> a su elemento padre <tbody>:

tblBody.appendChild(hilera);

4 Después, anexamos el elemento <tbody> a su elemento padre <table>:

tabla.appendChild(tblBody);

5 Finalmente, anexamos el elemento <table> a su elemento padre <body>:

body.appendChild(tabla);

Recuérda esta técnica. Te será muy útil en la programación bajo el estándar W3C DOM. Primero, creas los elementos de arriba a abajo; posteriormente adicionas los hijos a los padres de abajo a arriba.

A continuación, aparece el código HTML generado por el código JavaScript:

<table border="2">

<tbody>

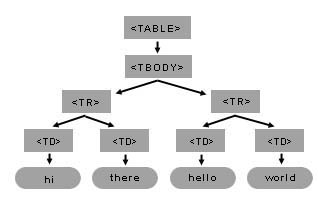
<tr><td>celda en la hilera 0, columna 0</td><td>celda en la hilera 0, columna 1</td></tr>

<tr><td>celda en la hilera 1, columna 0</td><td>celda en la hilera 1, columna 1</td></tr>

</tbody>

</table>

Aquí está el árbol de objetos DOM generado por el código del elemento <TABLE> :



Tú puedes construir esta tabla y sus elementos internos utilizando sólo algunos de los varios métodos del DOM. Recuerda tener en mente el modelo de la estructura que planeas crear; esto hará mucho más fácil la escritura del código necesario.

En el árbol del elemento <table> de la Figura 1, el elemento <table> tiene solamente un hijo mientras que <tbody> tiene dos.  A su vez, cada hijo de <tbody> tiene dos hijos. Finalmente, cada elemento <td> tiene sólo uno, el nodo de texto.