



Bootstrap

Front-End Basics

Table of Content

1. JavaScript
 - What is JavaScript?
 - Functions
 - Objects
2. Bootstrap
3. Grid system
4. Bootstrap components



JavaScript

Dynamic Programming Language

- JavaScript is a dynamic programming language
 - Operations otherwise done at compile-time can be done at run-time
- It is possible to change the type of a variable or add new properties or methods to an object while the program is running
- In static programming languages, such changes are normally not possible

Data Types

- Seven data types that are primitives
 - **String** - used to represent textual data
 - **Number** - a numeric data type
 - **Boolean** - a logical data type
 - **Undefined** - automatically assigned to variables
 - **Null** - represents the **intentional absence** of any object value
 - **BigInt** - represent integers with **arbitrary precision**
 - **Symbol** - **unique** and **immutable** primitive value
- Data structures
 - **Array**
 - **Object**

Variable Values

- **let**, **const** and **var** are used to declare variables
 - **let** - for reassigning a variable

```
let name = "George";
name = "Pesho";
```
 - **const** - once assigned it cannot be modified

```
const name = "George";
name = "Pesho"; // TypeError
```
 - **var** - defines a variable in the lexical scope **regardless** of block scope

```
var name = "George";
name = "Pesho";
```

Dynamic Typing

- Variables in JavaScript are not directly associated with any particular value type
- Any variable can be assigned (and re-assigned) values of all types

```
let foo = 42; // foo is now a number
foo = 'bar'; // foo is now a string
foo = true;  // foo is now a boolean
```

Comparison Operators

```
console.log(1 == '1'); // true
console.log(1 === '1'); // false
console.log(3 != '3'); // false
console.log(3 !== '3'); // true
console.log(5 < 5.5); // true
console.log(5 <= 4); // false
console.log(2 > 1.5); // true
console.log(2 >= 2); // true
console.log(5 ? 4 : 10); // 4
```

Operator	Notation in JS
EQUAL value	=
EQUAL value and type	===
NOT EQUAL value	!=
NOT EQUAL value/type	!==
Greater than	>
Greater than OR EQUAL	>=
LESS than	<
LESS than OR EQUAL	<=

Functions

- Function - named list of instructions (statements and expressions)
- Can take parameters and return result
 - Function names and parameters use camel case
 - The '{' stays on the same line

```
function printStars(count) {
  console.log("*".repeat(count));
}

printStars(10);
```

Declaring Functions

- Function declaration

```
function walk() {
  console.log("walking");
}
```

- Function expression

```
let walk = function () {
  console.log("walking");
}
```

- Arrow functions

```
let walk = () => {
  console.log("walking");
}
```

Parameters

- You can initialize parameters with no value

```
function foo(a, b, c) {  
  console.log(a);  
  console.log(b);  
  console.log(c); // undefined  
}  
foo(1, 2);
```

- The unused parameters are ignored

```
function foo(a, b, c) {  
  console.log(a);  
  console.log(b);  
  console.log(c);  
}  
foo(1, 2, 3, 5, 8);
```

Hoisting

- Variable and function declarations are put into memory during the compile phase, but stay exactly where you typed them in your code
- Only declarations are hoisted

```
console.log(num); // Returns undefined  
var num;  
num = 6;
```

Hoisting Variables

```
num = 6;  
console.log(num); // Returns 6  
var num;
```

```
num = 6;  
console.log(num); // ReferenceError: num is not defined  
let num;
```

```
console.log(num); // ReferenceError: num is not defined  
num = 6;
```

Hoisting Functions

```
run(); // running  
function run() {  
  console.log("running");  
};
```

```
walk(); // ReferenceError: walk is not defined  
let walk = function() {  
  console.log("walking");  
}
```

```
console.log(walk); // undefined  
walk(); // TypeError: walk is not a function  
var walk = function() {  
  console.log("walking");  
};
```

What is an Object?

- An object is a collection of fields, and a field is an association between a name (or key) and a value
- Objects are a reference data type
- You define (and create) a JavaScript object with an object literal:

```
let person = {  
  firstName: "John",  
  lastName: "Doe",  
  age: 42  
};
```

Variables Holding References

- The in-memory value of a reference type is the reference itself (a memory address)
- ```
let x = {name: 'John'};
let y = x;
```

```
y.name = "Pesho";
console.log(x.name); // Pesho
```

## Object Properties

- A property of an object can be explained as a variable that is attached to the object
- Object properties are basically the same as ordinary JavaScript variables, except for the attachment to objects

| Property Name | Property Value |
|---------------|----------------|
| firstName     | John           |
| lastName      | Doe            |
| age           | 42             |

## Object Keys and Values

```
let course = { name: 'JS Core', hall: 'Open Source' };
let keys = Object.keys(course);
console.log(keys); // ['name', 'hall']
if (course.hasOwnProperty('name')) {
 console.log(course.name); // JS Core
}
```

```
let values = Object.values(course);
console.log(values); // ['JS Core', 'Open Source']
if (values.includes('JS Core')) {
 console.log("Found 'JS Core' value");
}
```

## For... in Loop

- for... in - iterates a specified variable over all the enumerable properties of an object

```
let obj = {a: 1, b: 2, c: 3};
for (const key in obj) {
 console.log(`obj.${key} = ${obj[key]}`);
}
// Output:
// "obj.a = 1"
// "obj.b = 2"
// "obj.c = 3"
```

## For... of Loop

- The for.. of statement creates a loop iterating over iterable objects

```
let obj = { a: 1, b: 2, c: 3};
for (const key of Object.keys(obj)) {
 console.log(`obj.${key} = ${obj[key]}`);
}
// "obj.a = 1"
// "obj.b = 2"
// "obj.c = 3"

for (const val of Object.values(obj)) {
 console.log(val);
}
// 1
// 2
// 3
```



Bootstrap

## What is a Responsive Design?

- Presentation layers that adjust according to the screen size of the different devices



## Bootstrap

- World's most popular front-end component library
- Open source toolkit for developing with HTML, CSS and JS
- Works with
  - Responsive **grid system**
  - Extensive prebuilt **components**
  - Powerful plugins built on **jQuery**

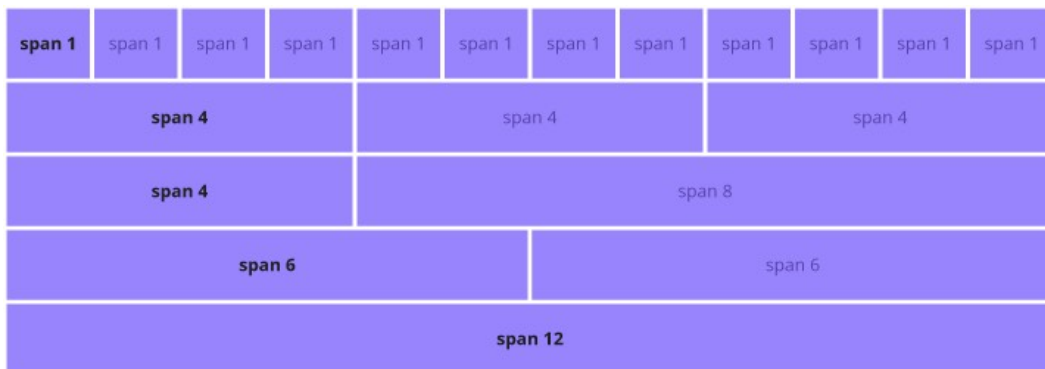
## Include from a BootstrapCDN – JS

- Be sure to place jQuery and Popper first, as the Bootstrap code depends on them  
jQuery CDN, bootstrap CDN

```
<script src="https://code.jquery.com/jquery3.3.1.slim.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>
```

## Bootstrap Grid System

### Build Layouts with Grid – Twelve Column System



### Bootstrap Grid System Demo

#### index.html

```
<div class="container">
 <div class="row">
 <div class="col-xs m-3">Column one</div>
 <div class="col-xs m-3">Column two</div>
 <div class="col-xs m-3">Column three</div>
 </div>
</div>
```

### Bootstrap Containers

- Rows must be placed in containers
  - .container has one fixed width for each screen size in bootstrap (xs, sm, md, lg)
  - .container-fluid expands to fill the available width

### Column Classes

- Determines how many columns to use on different screen sizes

#### index.html

```
<div class="col-sm-8 col-lg-4">Column one</div>
<div class="col-sm-2 col-lg-4">Column two</div>
<div class="col-sm-2 col-lg-4">Column three</div>
```

- **.col-xs**: width less than 768px
- **.col-sm**: width between 768px and 992px
- **.col-md**: width between 992px and 1200px
- **.col-lg**: width over 1200px

### Color

- Handful of color utility classes

#### index.html

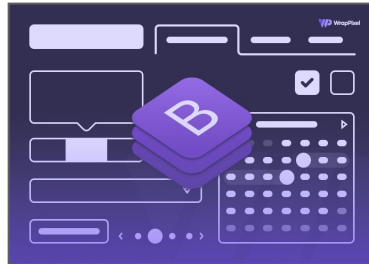
```
<p class="text-primary">.text-primary</p>
<p class="text-secondary">.text-secondary</p>
<p class="text-success">.text-success</p>
<p class="text-danger">.text-danger</p>
<p class="text-warning">.text-warning</p>
<p class="text-info">.text-info</p>
<p class="text-light bg-dark">.text-light</p>
<p class="text-dark">.text-dark</p>
<p class="text-muted">.text-muted</p>
<p class="text-white bg-dark">.text-white</p>
```

## Background Color

- Easily set the background of an element to any contextual class

index.html

```
<p class="bg-primary text-white">.bg-primary</p>
<p class="bg-secondary text-white">.bg-secondary</p>
<p class="bg-success text-white">.bg-success</p>
<p class="bg-danger text-white">.bg-danger</p>
<p class="bg-warning text-dark">.bg-warning</p>
<p class="bg-info text-white">.bg-info</p>
<p class="bg-light text-dark">.bg-light</p>
<p class="bg-dark text-white">.bg-dark</p>
```



## Bootstrap Components

### Button Groups

- Custom button styles with support for multiple sizes, states, and more

index.html

```
<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>
...
Documentation: https://getbootstrap.com/docs/4.0/components/buttons
```

### Alerts

- Provide contextual feedback message for typical user actions with the handful of flexible alert messages

index.html

```
<div class="alert alert-success alert-dismissible">
 x
 Success!
 This alert box could indicate a successful or positive action.
</div>
...
```

### Nav and Navbar

- Require a wrapping .navbar
- Responsive by default
- Come with built-in support for a handful of sub-components
  - **.navbar-brand** for your company, product, or project name
  - **.navbar-nav** for a full-height and lightweight navigation
  - **.nav-item** for every item in navigation

See more at: <https://getbootstrap.com/docs/4.0/components/navbar>

## Forms

- Form control styles, layout options and custom components for creating a wide variety of forms
- Use type attribute on all inputs to take advantage of newer input controls
  - Email verification
  - Number selection

See more at: <https://getbootstrap.com/docs/4.0/components/forms>

## Tables

```
<table class="table">
 <thead class="thead-dark">
 <tr> <th scope="col">#</th>
 <th scope="col">First</th>
 <th scope="col">Last</th>
 <th scope="col">Handle</th> </tr>
 </thead>
 <tbody>
 <tr> <th scope="row">1</th>
 <td>Mark</td>
 <td>Otto</td>
 <td>@mdo</td> </tr>
 <tr> . . . </tr>
 <tr> . . . </tr>
 </tbody>
 </table>
```

## Jumbotron

- Lightweight, flexible component for showcasing hero unit style content

```
<div class="jumbotron">
 <h1 class="display-4">Hello, world!</h1>
 <p class="lead">This is a ...</p>
 <hr class="my-4"><p>It uses ...</p>
 <p class="lead">
 Learn more
 </p>
</div>
```

## Summary

- JS is a dynamic programming language
- Functions in JS
- JS objects hold key-value pairs
- Bootstrap is the most popular front-end component library

NodeJS: This package has installed:  
Node.js v18.14.1 to /usr/local/bin/node  
pm v9.3.1 to /usr/local/bin/npm  
Make sure that /usr/local/bin is in your SPATH.