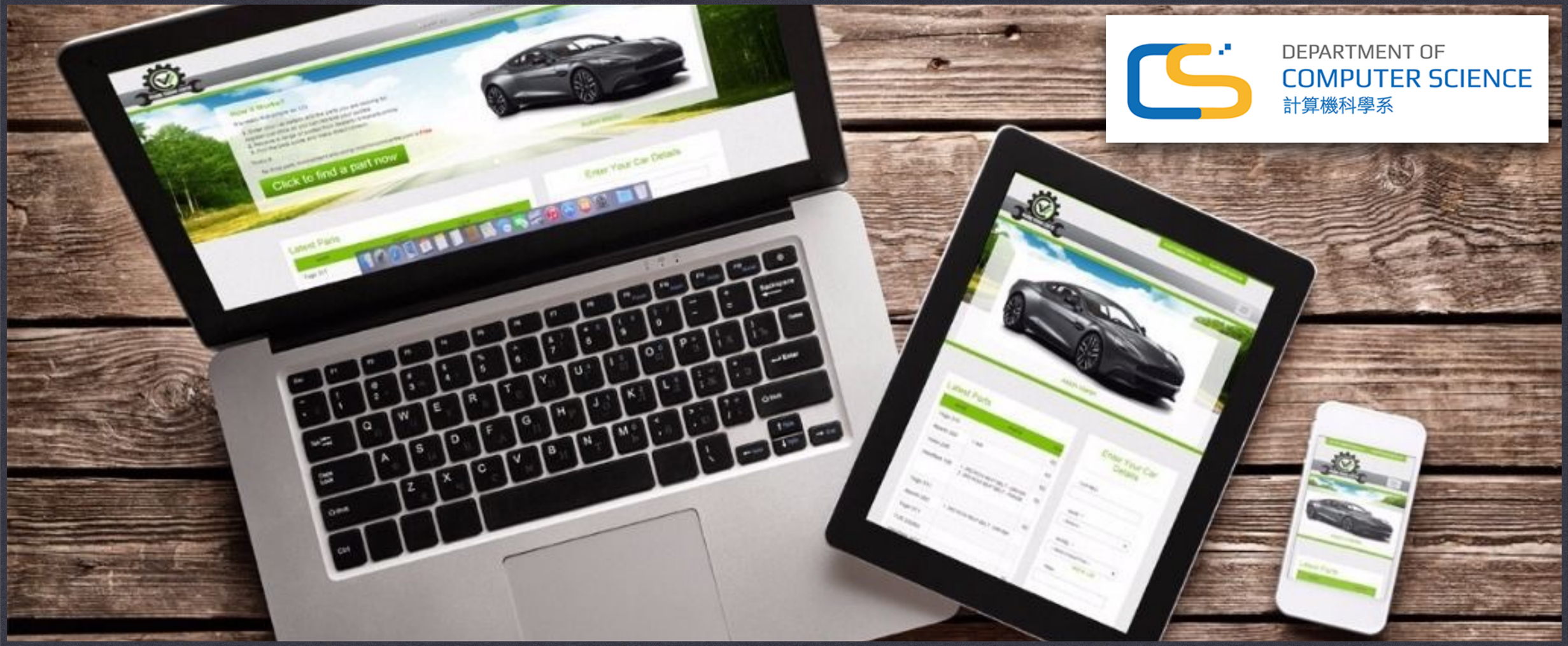



DEPARTMENT OF
COMPUTER SCIENCE
計算機科學系

# MORE ON WEB FRONTEND DEVELOPMENT

## COMP7980 - WEB AND MOBILE PROGRAMMING

CHAPTER 2 - 2022 - HKBU

MARTIN & KENNY

# Agenda

- CSS Layout and **Responsive Web Design**
  - Flexbox, Bootstrap
- Client-side **JavaScript**
- **Form** Elements

# RESPONSIVE WEB DESIGN

# Responsive Web Design

- **Responsive Web Design** (RWD) is an approach to web design aimed at crafting sites to provide an **optimal viewing** and interaction experience **across a wide range of devices**

# Why RWD?

- It is becoming more important as the **amount of mobile traffic now accounts for more than half of total internet traffic**.
- This trend is so prevalent that Google has begun to **boost the ratings of sites that are mobile friendly** if the search was made from a mobile device.
  - This has the net effect of **penalizing sites that are not mobile friendly**.

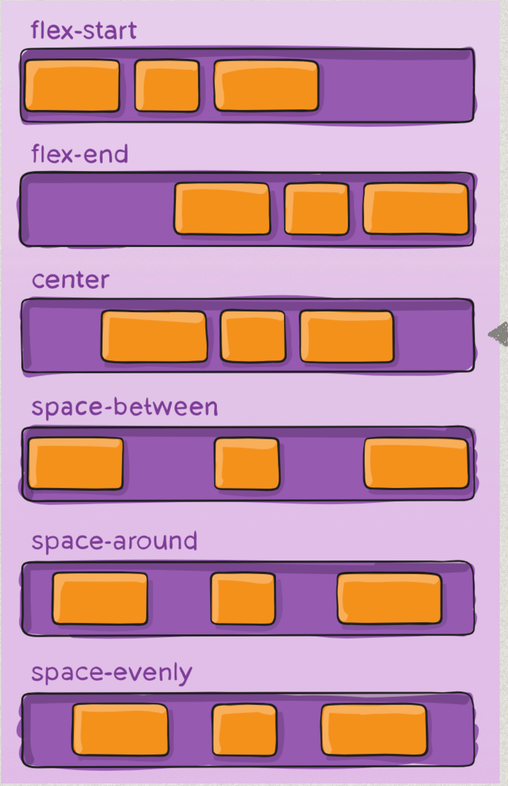
# CSS Flexbox

- CSS Flexbox (flexible box) provides a more efficient way to **lay out, align and distribute space** among items in a **container**, even when their size is unknown and/or dynamic.
- A **flex container** **expands its items** to fill available free space or shrinks them to prevent overflow

https://css-tricks.com/snippets/css/a-guide-to-flexbox/

# Flexbox Container

- **nowrap** may shrink the items
- **wrap** will move the overflow items to next row line




**flex-container properties**

**display**
flex · inline-flex

**flex-wrap**
*nowrap* · wrap · wrap-reverse

**flex-direction**
*row* · row-reverse · column · column-reverse

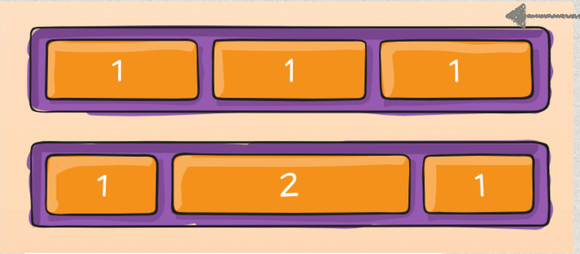
**justify-content**
*flex-start* · flex-end · center · space-between · space-around

**align-items**
flex-start · flex-end · center · *stretch* · baseline

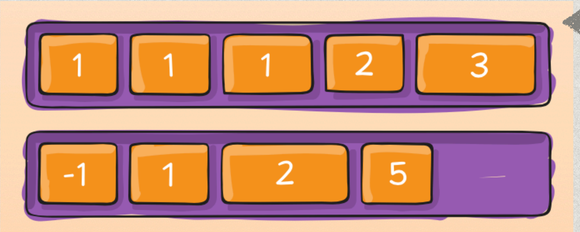
**align-content**
flex-start · flex-end · center · *stretch* · space-between · space-around

https://www.alsacreations.com/xmedia/guidelines/flexbox-cheatsheet.pdf

# Flexbox Items




For the child with flex-grow value of 2, the remaining space would take up twice as much space as the others




**flex-item properties**

**flex-grow**
*0* · 1 · 2 · …

**flex-shrink**
0 · *1* · 2 · …

**flex-basis**
*auto* · value · fill · min-content · …

**flex** (= flex-grow flex-shrink flex-basis)
*flex: initial; ⇢ flex: 0 1 auto;*
flex: auto; ⇢ flex: 1 1 auto;
flex: none; ⇢ flex: 0 0 auto;
flex: value; ⇢ flex: value 1 0;

**order**
*0* · 1 · 2 · …

**align-self**
*auto* · flex-start · flex-end · center · stretch · baseline

# RESPONSIVE WEB DESIGN WITH BOOTSTRAP

# Bootstrap

- Originally named **Twitter Blueprint**.
- Bootstrap is a free, open source framework based on **Flexbox**.
- It provides a **grid system** to support responsive web design.
- **Containers** are the most basic layout element in Bootstrap and are **required when using our default grid system**.

```
<div class="container my-4">
  <!-- Content here -->
</div>
```

# Responsive grid system

- The **responsive grid system** is formed by .row and .col classes.
- The column elements will have **equal width**.

```
<div class="row">
    <div class="col card">
        Card 1
    </div>
    <div class="col card">
        Card 2
    </div>
    <div class="col card">
        Card 3
    </div>
    <div class="col card">
        Card 4
    </div>
</div>
```

# Breakpoints

Bootstrap includes six default breakpoints, sometimes referred to as *grid tiers*, for building responsively. These breakpoints can be customized if you're using our source Sass files.

| Breakpoint | Class infix | Dimensions |
|---|---|---|
| Extra small | *None* | <576px |
| Small | `sm` | ≥576px |
| Medium | `md` | ≥768px |
| Large | `lg` | ≥992px |
| Extra large | `xl` | ≥1200px |
| Extra extra large | `xxl` | ≥1400px |

# Grid Classes

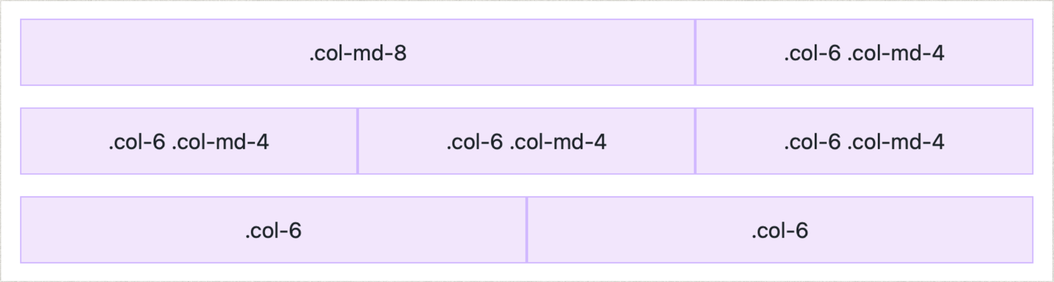
- **Grid classes apply to devices with screen widths greater than or equal to the breakpoint sizes**, and override grid classes targeted at smaller devices.
- Therefore, e.g. applying any **.col-md-*** class to an element will not only affect its styling on medium devices but **also on large devices if a .col-lg-* class is not present**.

# Columns

- Bootstrap includes a **responsive, mobile first** fluid grid system that appropriately scales up to **12 columns** as the device or viewport size increases.
  - Developers can specify **how many columns are needed** for each component under different screen sizes.
  - Bootstrap will then **re-arrange the components** to fit each screen size.

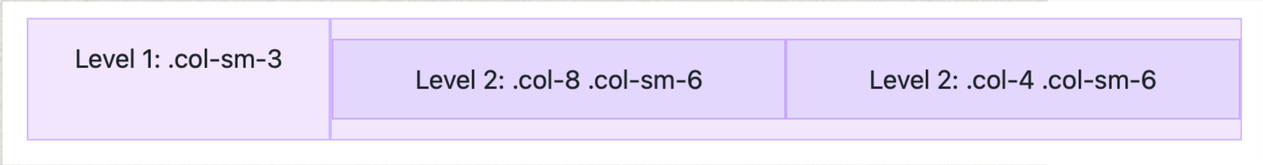
```html
<div class="container text-center">
   <div class="row">
    <div class="col-md-8">.col-md-8</div>
    <div class="col-6 col-md-4">.col-6 .col-md-4</div>
  </div>

   <div class="row">
    <div class="col-6 col-md-4">.col-6 .col-md-4</div>
    <div class="col-6 col-md-4">.col-6 .col-md-4</div>
    <div class="col-6 col-md-4">.col-6 .col-md-4</div>
  </div>

   <div class="row">
    <div class="col-6">.col-6</div>
    <div class="col-6">.col-6</div>
  </div>
</div>
```

Being displayed on desktop




What if it is being displayed on mobile phone?

# Nesting




```html
<div class="container text-center">
  <div class="row">
    <div class="col-sm-3">
      Level 1: .col-sm-3
    </div>
    <div class="col-sm-9">
      <div class="row">
        <div class="col-8 col-sm-6">
          Level 2: .col-8 .col-sm-6
        </div>
        <div class="col-4 col-sm-6">
          Level 2: .col-4 .col-sm-6
        </div>
      </div>
    </div>
  </div>
</div>
```

# Margin & Padding

The classes are named using the format `{property}{sides}-{size}` for `xs` and `{property}{sides}-{breakpoint}-{size}` for `sm`, `md`, `lg`, `xl`, and `xxl`.

Where *property* is one of:

- `m` - for classes that set `margin`
- `p` - for classes that set `padding`

Where *sides* is one of:

- `t` - for classes that set `margin-top` or `padding-top`
- `b` - for classes that set `margin-bottom` or `padding-bottom`
- `s` - (start) for classes that set `margin-left` or `padding-left` in LTR, `margin-right` or `padding-right` in RTL
- `e` - (end) for classes that set `margin-right` or `padding-right` in LTR, `margin-left` or `padding-left` in RTL
- `x` - for classes that set both `*-left` and `*-right`
- `y` - for classes that set both `*-top` and `*-bottom`
- blank - for classes that set a `margin` or `padding` on all 4 sides of the element

Where *size* is one of:

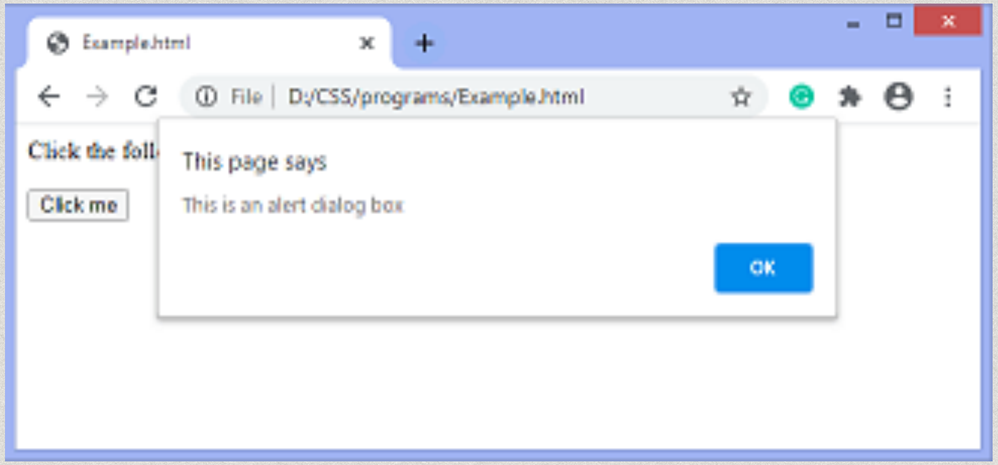
- `0` - for classes that eliminate the `margin` or `padding` by setting it to `0`
- `1` - (by default) for classes that set the `margin` or `padding` to `$spacer * .25`
- `2` - (by default) for classes that set the `margin` or `padding` to `$spacer * .5`
- `3` - (by default) for classes that set the `margin` or `padding` to `$spacer`
- `4` - (by default) for classes that set the `margin` or `padding` to `$spacer * 1.5`
- `5` - (by default) for classes that set the `margin` or `padding` to `$spacer * 3`
- `auto` - for classes that set the `margin` to auto

**Spacer = 1rem**
**= font-size**
**= 16px by default**

# CLIENT-SIDE SCRIPTING WITH JAVASCRIPT

# JavaScript

- JavaScript was designed to **add interactivity** to HTML pages
- Note that **JavaScript** and **Java** are two completely **different languages** in both concept and design!
- We use JavaScript to define the **behavior** of a webpage.



# What can JavaScript do?

- **React to events**
  - Execute when something happens, like when a page has finished loading or when a user clicks on an HTML element.
  - e.g., onchange, onclick, etc

```
<script>
    document.getElementById("id").value = "audi"
</script>
```

- **Read and write HTML elements**
  - Read and change the content of an HTML element

# Access HTML Element in DOM Tree

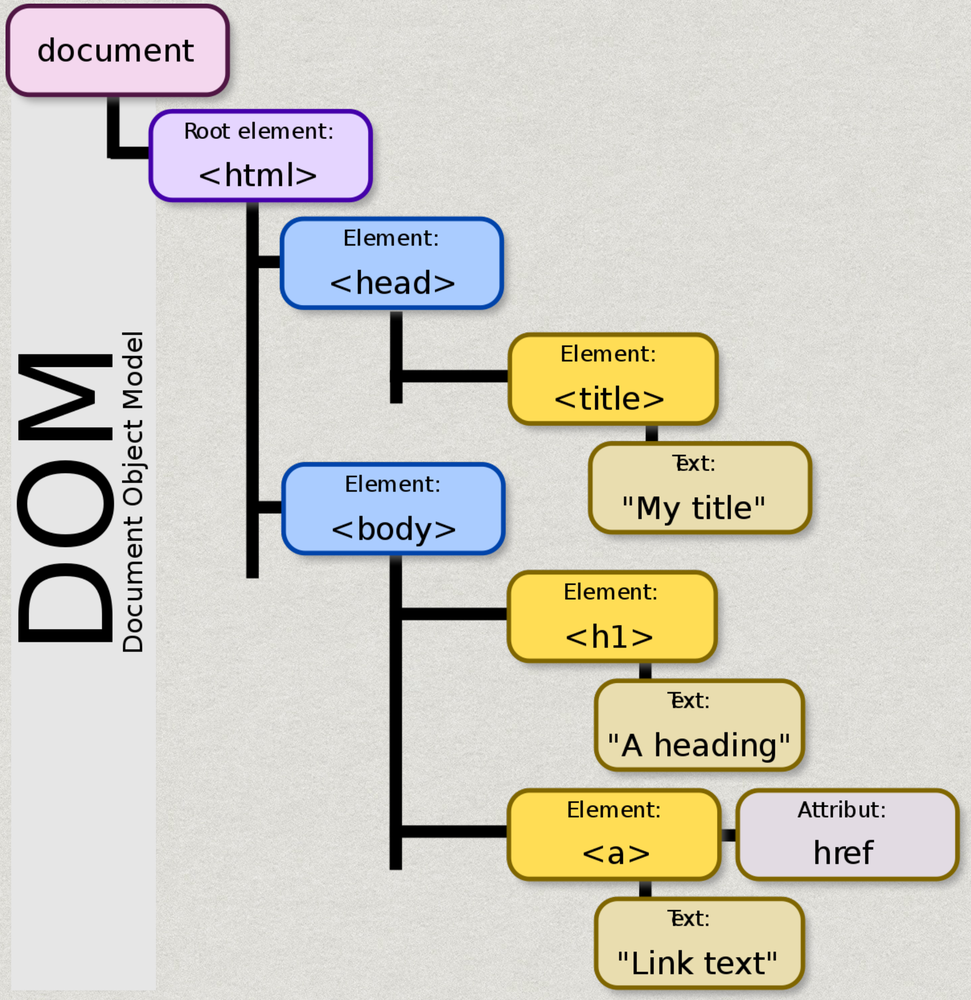
- To get an element **with a specified id**

```
document.getElementById(id)
```

- When a web page is loaded, the browser creates a **Document Object Model (DOM tree)** of the page.

- To access the **parent element** of an arbitrary node ( `elem` ) in the DOM tree, use:

```
elem.parentNode
```



# Read and write HTML elements

- To assign a new **class name** to a particular ID element:

```
document.getElementById("myid").classList.add("active");
```

- To **edit a style property** of a particular ID element:

```
document.getElementById("myid").style.display = "none";
```

- To **modify the inner HTML**

```
document.getElementById("myid").innerHTML = "Inner content";
```

# QuerySelector & QuerySelectorAll

- These methods return element(s) that **match a specified CSS selector(s)**.

| | |
|---|---|
| Element Selector | `document.querySelector('td');` |
| Id Selector | `document.querySelector('#id2');` |
| Class Selector | `document.querySelectorAll('.sundays');` |
| Attribute Selector | `document.querySelectorAll('ol[start="5"]');` |

# QuerySelector & QuerySelectorAll

- The document method querySelector() returns **the first element** within the document that matches the specified selector, or group of selectors.
  - If no matches are found, null is returned.
- The document method querySelectorAll() returns **a collection of elements** within the document that matches the specified selector, or group of selectors.

# JavaScript Object

- **Objects** can be seen as **a collection of properties**
  - Properties are identified using **key** values
  - **Property values** can be values of any type, including other objects
  - Also sometimes referred as **key-value pairs** or **dictionary**.
- car is an object:

```
var car = { type: "Volks", model: "500", color: "white" };
```

  - To access the **car's color value**, we can use car.color

# The for/in Statement

- The **for/in** statement can loop through the **properties of an object**

```
var person = {fname: "John", lname: "Doe", age: 25};

for (var key in person) {

    document.getElementById("id1").innerHTML
                                    += key + ": " + person[key] + "<br>";

}
```

fname: John
lname: Doe
age: 25

# Array

- An **array** is a **special kind of object**, with a collection of **elements**.
  - An array can **hold many values under a single name**, and you can access the values by **referring to an index number.**
- vehicles is the array:

```
var vehicles = ["minibus", "taxi", "motorcycle"];
```

  - To access a particular element of the vehicles, we use vehicles[1]

# The for/of Statement

- The **for/of** statement can loop through **the elements of an array**

```
var persons = ["John", "Jean"];

for (var person of persons) {

    document.getElementById("id2").innerHTML += person + "<br>";

}
```

John
Jean

# FORM ELEMENTS

# HTML Form Elements

- <input> with different types
  - text, number, url, date, checkbox

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| button | date | email | image | password | reset | tel | url |
| checkbox | datetime | file | month | radio | search | text | week |
| color | datetime-local | hidden | number | range | submit | time | |

- <select> to give a combo box for selection.

# HTML5 Form Input Validation

- These **input validations** are supported by HTML5
  - Input types number and email.
  - Ensure **mandatory input**, use element attribute required.
  - Specify a **valid range** for number field, use max and min.