# Angular Font Awesome

Installare i seguenti componenti :

npm install @fortawesome/fontawesome-svg-core

npm install @fortawesome/free-brands-svg-icons

npm install @fortawesome/free-regular-svg-icons

npm install @fortawesome/free-solid-svg-icons

npm install @fortawesome/angular-fontawesome

in **app.module.ts**

import {FaIconLibrary, FontAwesomeModule} from '@fortawesome/angular-fontawesome';

in **app.module.ts**

@NgModule({

declarations: [AppComponent, HomePage],

entryComponents: [],

imports: [BrowserModule,

**FontAwesomeModule**,

in **app.module.ts**

import {

faAsterisk, faBell as faSolidBell, faCircle, faCloud,

faCog, faEnvelopeOpen, faHandPointLeft, faMobile, faMoon,

faPlay, faSkating, faSkiing, faSkiingNordic,

faSmileBeam as faSmileBeanSolid, faSmileWink as faSmileWinkSolid,

faSnowboarding, faSpinner, faSquare, faStar, faSun,

faSwimmer, faSync, faBan, faCamera

} from '@fortawesome/free-solid-svg-icons';

import {faBell as faRegularBell, faSmileBeam, faSmileWink} from '@fortawesome/free-regular-svg-icons';

Nel construttore iniettare la FaIconeLibrary ed aggiungere quindi le librerie di icone (sempre in app.module.ts):

export class AppModule {

constructor(library: FaIconLibrary) {

// library.addIconPacks(fas);

library.addIcons(faSmileWinkSolid, faSmileBeanSolid,

faSmileWink, faSmileBeam, faEnvelopeOpen, faCloud,

faMobile, faSquare, faSpinner, faCircle,

faSync, faPlay, faSun, faMoon, faStar,

faHandPointLeft, faAsterisk, faCog, faSkating,

faSkiing, faSkiingNordic, faSnowboarding, faSwimmer,

faSolidBell, faRegularBell, faCamera, faBan);

}

}

Nei componenti, accedi a queste icone per nome. Si noti che il nome non è lo stesso del nome dall'istruzione import. È possibile derivare il nome rimuovendo il prefisso fa e sostituendo camelCase con trattini. Ad esempio: faSmileWink diventa smile-wink.

<fa-icon icon="smile-wink"></fa-icon>

Per impostazione predefinita, specificando solo il nome si fa riferimento alle icone della libreria solida (fas). Per fare riferimento alle icone di altre librerie devi specificare il prefisso.

fa-icon [icon]="['far', 'smile-wink']"></fa-icon>

Prefissi supportati :

* far = regular
* fas = solid
* fal = light
* fab = brands

È possibile modificare il prefisso predefinito (fas), se è necessario fare riferimento a più icone da un set di icone diverso rispetto ai fas predefiniti. Per modificare il valore predefinito, iniettare FaConfig nel costruttore e assegnare il nuovo valore predefinito alla proprietà defaultPrefix.

import { FaConfig } from '@fortawesome/angular-fontawesome';

export class AppComponent {

constructor(faConfig: FaConfig) {

faConfig.defaultPrefix = 'far';

}

}

La libreria Angular FontAwesome supporta molte opzioni per trasformare e combinare icone.

Il codice seguente mostra come ridimensionare, ruotare e capovolgere le icone:

<fa-icon [icon]="['fas', 'skiing']" size="lg"></fa-icon>

<fa-icon [icon]="['fas', 'skiing']" size="2x"></fa-icon>

<fa-icon [icon]="['fas', 'skiing']" size="3x"></fa-icon>

<fa-icon [icon]="['fas', 'skiing']" size="4x"></fa-icon>

[home.page.html](https://github.com/ralscha/blog2019/blob/master/fa/src/app/home/home.page.html#L30-L33)

<fa-icon [icon]="['fas', 'snowboarding']" size="3x"></fa-icon>

<fa-icon [icon]="['fas', 'snowboarding']" size="3x" rotate="90"></fa-icon>

<fa-icon [icon]="['fas', 'snowboarding']" size="3x" rotate="180"></fa-icon>

<fa-icon [icon]="['fas', 'snowboarding']" size="3x" rotate="270"></fa-icon>

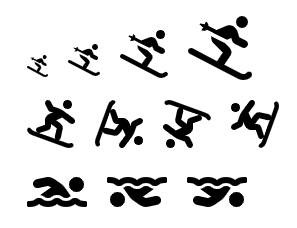
[home.page.html](https://github.com/ralscha/blog2019/blob/master/fa/src/app/home/home.page.html#L36-L39)

<fa-icon [icon]="['fas', 'swimmer']" size="3x" flip="horizontal"></fa-icon>

<fa-icon [icon]="['fas', 'swimmer']" size="3x" flip="vertical"></fa-icon>

<fa-icon [icon]="['fas', 'swimmer']" size="3x" flip="both"></fa-icon>

[home.page.html](https://github.com/ralscha/blog2019/blob/master/fa/src/app/home/home.page.html#L42-L44)

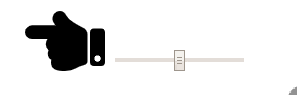


È anche facile aggiungere collegamenti a qualsiasi direttiva e apportare dinamicamente trasformazioni.

<fa-icon size="5x" [icon]="['fas', 'hand-point-left']" transform="rotate-{{magicLevel}}"></fa-icon>

<input type='range' min="0" max="360" [value]="magicLevel" (input)="magicLevel=$event.target.value"/>

[home.page.html](https://github.com/ralscha/blog2019/blob/master/fa/src/app/home/home.page.html#L138-L139)



Se è necessario allineare verticalmente un gruppo di icone, è possibile impostare la direttiva fixedWidth su true e la libreria impone che tutte le icone abbiano la stessa larghezza.

<ul>

<li>

<fa-icon [icon]="['fas', 'swimmer']" [fixedWidth]="true" size="2x"></fa-icon>

Swimmer

</li>

<li>

<fa-icon [icon]="['fas', 'snowboarding']" [fixedWidth]="true" size="2x"></fa-icon>

Snowboarding

</li>

<li>

<fa-icon [icon]="['fas', 'skiing']" [fixedWidth]="true" size="2x"></fa-icon>

Skiing

</li>

<li>

<fa-icon [icon]="['fas', 'skiing-nordic']" [fixedWidth]="true" size="2x"></fa-icon>

Skiing Nordic

</li>

</ul>

L'immagine seguente mostra la differenza. Le prime 4 icone usano la loro larghezza normale. Le altre 4 icone hanno la larghezza allineata con fixedWidth.



The library also supports a spin and pulse [animation](https://fontawesome.com/how-to-use/on-the-web/styling/animating-icons" \t "_blank). These work especially well with icons from the [spinners category](https://fontawesome.com/icons?c=spinners" \t "_blank)

<fa-icon [icon]="['fas', 'asterisk']" size="6x" [spin]="true"></fa-icon>

<fa-icon [icon]="['fas', 'cog']" size="6x" [pulse]="true"></fa-icon>

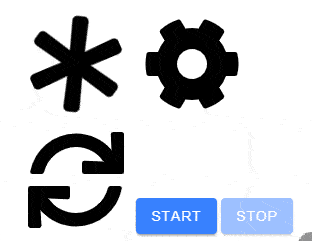
[home.page.html](https://github.com/ralscha/blog2019/blob/master/fa/src/app/home/home.page.html#L47-L48)

<fa-icon size="6x" [icon]="['fas', 'sync']" [spin]="syncRunning"></fa-icon>

<ion-button (click)="syncRunning = true" [disabled]="syncRunning">Start</ion-button>

<ion-button (click)="syncRunning = false" [disabled]="!syncRunning">Stop</ion-button>

[home.page.html](https://github.com/ralscha/blog2019/blob/master/fa/src/app/home/home.page.html#L52-L54)



The [border](https://fontawesome.com/how-to-use/on-the-web/styling/bordered-pulled-icons" \t "_blank) directive draws a border around the icon. If you need to wrap text around icons, the [pull](https://fontawesome.com/how-to-use/on-the-web/styling/bordered-pulled-icons) directive helps with that use case.

<ion-card>

<ion-card-content>

<fa-icon [icon]="['fas', 'asterisk']" [border]="true" size="2x" pull="left"></fa-icon>

Lorem ipsum dolor sit amet, ei cum possit denique, debet mundi sit ne.

</ion-card-content>

</ion-card>

<ion-card>

<ion-card-content>

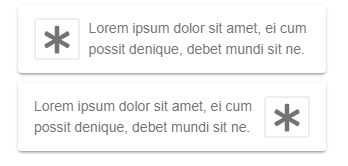
<fa-icon [icon]="['fas', 'asterisk']" [border]="true" size="2x" pull="right"></fa-icon>

Lorem ipsum dolor sit amet, ei cum possit denique, debet mundi sit ne.

</ion-card-content>

</ion-card>

[home.page.html](https://github.com/ralscha/blog2019/blob/master/fa/src/app/home/home.page.html#L100-L112)



With [Mask](https://fontawesome.com/how-to-use/on-the-web/styling/masking" \t "_blank), an application can combine two icons and create one single-color shape.  
[Transform](https://fontawesome.com/how-to-use/on-the-web/styling/power-transforms) allows you to scale, position, flip, and rotate icons arbitrarily.

<fa-icon size="6x" [icon]="['fas', 'swimmer']" transform="shrink-10 down-4" [mask]="['fas', 'circle']"></fa-icon>

[home.page.html](https://github.com/ralscha/blog2019/blob/master/fa/src/app/home/home.page.html#L58)



[Stacking](https://fontawesome.com/how-to-use/on-the-web/styling/stacking-icons) allows you to easily stack multiple icons on top of each other.

<fa-stack size="5x">

<fa-icon [icon]="['fas', 'camera']" stackItemSize="1x"></fa-icon>

<fa-icon [icon]="['fas', 'ban']" style="color:Tomato" stackItemSize="2x"></fa-icon>

</fa-stack>

[home.page.html](https://github.com/ralscha/blog2019/blob/master/fa/src/app/home/home.page.html#L143-L146)



While stacking simply puts the icons on top of each other, [layering](https://fontawesome.com/how-to-use/on-the-web/styling/layering" \t "_blank) gives you even more control how to place multiple icons and text on top of each other.

<fa-layers size="6x">

<fa-icon [icon]="['fas', 'play']" transform="rotate--90 grow-2"></fa-icon>

<fa-icon [icon]="['fas', 'sun']" [inverse]="true" transform="shrink-10 up-2"></fa-icon>

<fa-icon [icon]="['fas', 'moon']" [inverse]="true" transform="shrink-11 down-4.2 left-4"></fa-icon>

<fa-icon [icon]="['fas', 'star']" [inverse]="true" transform="shrink-11 down-4.2 right-4"></fa-icon>

</fa-layers>

[home.page.html](https://github.com/ralscha/blog2019/blob/master/fa/src/app/home/home.page.html#L115-L120)



<fa-layers size="5x">

<fa-icon [icon]="['fas', 'cloud']"></fa-icon>

<fa-layers-text content="Cloud" style="color: white;" transform="shrink-12"></fa-layers-text>

</fa-layers>

[home.page.html](https://github.com/ralscha/blog2019/blob/master/fa/src/app/home/home.page.html#L124-L127)



<fa-layers size="5x">

<fa-icon [icon]="['fas', 'envelope-open']"></fa-icon>

<fa-layers-counter content="5"></fa-layers-counter>

</fa-layers>

[home.page.html](https://github.com/ralscha/blog2019/blob/master/fa/src/app/home/home.page.html#L131-L134)



With the [[styles]](https://github.com/FortAwesome/angular-fontawesome/blob/master/docs/usage/features.md#default-style) directive you can control all kind of properties of the SVG. This directive works similar to the [ngStyle] directive and allows you to dynamically bind properties from the TypeScript code.

<fa-icon size="5x" [icon]="['fas', 'bell']" [styles]="dynamicStyle()"></fa-icon>

<fa-icon size="5x" [icon]="['far', 'bell']"

[styles]="{'color': color, 'stroke': stroke, 'stroke-width': strokeWidth + 'px', 'opacity': opacity}"></fa-icon>

[home.page.html](https://github.com/ralscha/blog2019/blob/master/fa/src/app/home/home.page.html#L150-L152)

color = 'red';

stroke = 'red';

strokeWidth = 0;

opacity = 1;

private interval: number;

ngOnDestroy(): void {

clearInterval(this.interval);

}

ngOnInit(): void {

this.interval = setInterval(() => {

this.color = this.randomColor();

this.stroke = this.randomColor();

this.strokeWidth = this.randomWidth();

this.opacity = Math.random();

}, 1000);

}

dynamicStyle() {

return {

opacity: Math.random(),

color: this.randomColor(),

stroke: this.randomColor(),

'stroke-width': this.randomWidth() + 'px'

};

}

private randomWidth(): number {

return Math.floor(Math.random() \* 30) + 1;

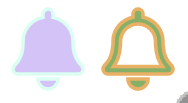
}

private randomColor(): string {

return '#' + (0x1000000 + (Math.random()) \* 0xffffff).toString(16).substr(1, 6);

}

[home.page.ts](https://github.com/ralscha/blog2019/blob/master/fa/src/app/home/home.page.ts#L16-L51)



This concludes this tutorial about FontAwesome and the Angular library. Visit the [project page](https://github.com/FortAwesome/angular-fontawesome) for more in-depth and up to date information.

The source code for the examples in this blog post is hosted on GitHub:  
<https://github.com/ralscha/blog2019/tree/master/fa>