# HOORAY ICON FONTS

james williamson | lynda.com

<br/>

## Hello.

**I’m James Williamson**

lynda.com | senior author

**@jameswillweb on the Twitter**

<br/>

## Let's talk about icons

**"Icons are little miracle workers. They circumvent language obstacles, give concise warnings and directions, convey our moods and show which buttons to press."**

-John Hicks

## Icons give us a shared language

## This is extremely valuable in the mobile/responsive context

As screen real estate shrinks, icons give us a way to clearly communicate ideas, actions, and instructions to our users with a minimal footprint.

## How do we display icons?

**Images**
High overhead, painful to maintain, resolution dependent

**CSS Sprites**
Lower overhead, difficult to create, resolution dependent

**SVG**
Scalable, style-able, decent support, higher overhead

We need more responsive solutions

## A more responsive solution

We need icons that scale independently of resolution

We need icons that can be styled through CSS

We need icons that are small in file size

We need icons that can be downloaded in a single file to reduce server requests

We need **ICON FONTS!**

## HOORAY ICON FONTS!

♥

&lt;br/&gt;

# (Actually, this is not exactly a new idea)

# Using fonts for icons in our UI

**Pros:**

Scalable

Single file request

Styles with CSS

Well-supported

**Cons:**

Monochromatic

Tricky to make

Accessibility issues

Can be tricky to control

## Icon font options

Go grab one of the dozens of high-quality, open-source icon fonts available online

Purchase a commercial icon font

Use an icon-font hosting service like Pictos

Build your own

## Using existing icon fonts

Plenty of high-quality, open-source icons available

Many include @font-face kits and code samples

You may not be able to find every icon you need

Dependent upon the icon’s style

Must be careful to avoid bloat

## Building your own

Services like the Noun Project, Icomoon & Pictos allow you to build custom icon fonts

Some allow you to choose the icons you want while others allow you to upload your own artwork

For greater control, you can use programs like Illustrator, Inkscape, Glyphs, FontForge, and FontLab to build your own icon font

## Using icon fonts

There are multiple ways to display icon fonts based on coding standards, font construction, and accessibility considerations

Despite the differences in implementations, best practices are starting to emerge regarding icon font syntax...

## Demo

Basic icon font usage

## Icons mapped to Basic Latin

**HTML**

```
<p><span class=”icon”>q</span> Brad Frost
loves QR Codes!</p>
```

**CSS***

```
.icon {
    font-family: “Your Icon Font”;
}
```

**Result:** Brad Frost loves QR Codes!

<br/>

** @font-face is assumed*

# Icons mapped to Basic Latin

**Pros:**

Easy to use

No complex CSS

Single rule drives all icons

**Cons:**

Accessibility

Confuses the robots

Falls back to a letter that has no business being there

## Demo

Using generated content

## Unicode mapping & generated content

**HTML**

```
<p class=”icon-heart”>I love icons!</p>
```

**CSS***

```
.icon-heart:before {
    font-family: “Your Icon Font”;
    content: “\2661”;
    display: inline-block;
}
```

**Result:** ♥ I love icons!

** @font-face is assumed*

# Unicode mapping & generated content

**Pros:**

Leaves content untouched

You can use common unicode values for symbols as fallbacks

PUA unicode values will fallback to an empty glyph

**Cons:**

Unicode mapping can be hard to remember

Unless you create your own, unicode mapping might not meet your requirements

Making it class-based bloats CSS

## Demo

Using the data-icon attribute

&lt;br/&gt;

## Using the data-icon attribute

**HTML**

```
<p data-icon=”&#x2661;”>I love icons!</p>
```

**CSS***

```
*[data-icon]:before {
    font-family: “Your Icon Font”;
    content: attr(data-icon);
    display: inline-block;
}
```

**Result:** ♥ I love icons!

** @font-face is assumed*

## Using the data-icon attribute

| **Pros:** | **Cons:** |
|---|---|
| Nice and semantic | Not as human readable |
| No need to use extra classes | |

## …hold up though!

Using generated content with data-icon still leaves us with accessibility issues.

Generated content will still be read by screen readers.

Which could be awkward.

## Demo

The still even-more awesome way to accessibly use generated content & data-icon

## Using the aria-hidden attribute

**HTML**

```
<p><span data-icon=”&#xE001;” aria-hidden=”true”>
</span>I love icons!</p>
```

**CSS***

```
*[data-icon]:before {
    font-family: “Your Icon Font”;
    content: attr(data-icon);
    display: inline-block;
    speak: none;
}
```

**Result:** ♥ I love icons!

** @font-face is assumed*

## Using the aria-hidden attribute

**Pros:**

Semantically clean

Well supported

Creates purely visual content

**Cons:**

Requires extra markup

# A quick word about PUA

Unicode includes several Private Use values that are not reserved for specific languages or characters

The Basic Multilingual Plane includes 6,400 code points and is widely used for icon font encoding

PUA encoded glyphs will fallback to an empty glyph if the font fails or if @font-face is not supported

BMP encoding runs from E000 - F8FF

Apple places their logo at F8FF

## Using ligatures for icon fonts

**HTML**

```
<p><span class=”icon”>twitter</span> Tweet
that!</p>
```

**CSS***

```
.icon {
    font-family: “ligature icons”;
    text-rendering:  optimizeLegibility;
}
```

**Result:** 🐦 Tweet that!

** @font-face is assumed*

## Using ligatures for icon fonts

**Pros:**

Easy to use

Falls back to meaningful text

If the font is mapped correctly you can combine techniques

**Cons:**

Must use a ligature-mapped icon font

Extra text in content can be weird

Ligature support is uneven

# Tips for displaying icon fonts

Normalize them

*font-weight, font-style, font-variant, text-transform...*

Using inline-block gives you more control

*also ensures 'click-ability'*

Although scalable, not every size displays great

*try to scale along the font's design grid*

Base64 encode your fonts

*avoids cross-domain Firefox issues*

<br/>

# Tips for displaying icon fonts

Use text-rendering: optimizeLegibility for ligatures

*also enables kerning pairs*

Use -webkit-font-smoothing: antialiased

*makes icons crisper in webkit-based browsers*

Use vertical-align to control baselines on inline icons

*not all icon fonts will align to the baseline the same*

## Taking icon fonts further

You can style font icons using any text styling property

You can animate and transform them as well

*This sometimes gets a little wonky, provide fallbacks*

# Do icons have to be monochrome?

No! You can create multi colored icons using multiple glyphs

Glyphs can be overlapped using absolute positioning

You can also use zero-width characters

*This requires the font to be produced with zero-width characters*

**Icon Fonts are not right for every project.**

Before using icon fonts or an icon font service, make sure you have a strategy in place that matches your code aesthetics.

# Icon Font Resources

Chris Coyier's Big List of Icon Fonts
http://css-tricks.com/flat-icons-icon-fonts/

Interactive Unicode Table
http://unicode-table.com

The Noun Project
http://thenounproject.com/

Github: Building Octicons
https://github.com/blog/1135-the-making-of-octicons

Filament Group's Icon Font Compatibility Table
https://docs.google.com/spreadsheet/ccc?key=0Ag5_yGvxpINRdHFYeUJPNnZMWUZKR2ItMEpRTXZPdUE#gid=0

Want these slides?
http://www.slideshare.net/jameswillweb

## 😁 THANK YOU

**james williamson | lynda.com**

**jwilliamson@lynda.com**
**@jameswillweb on the Twitter**
**www.simpleprimate.com**

Want these slides?
**http://www.slideshare.net/jameswillweb**

<br/>