Web toolkit

Release

MIT - Agenzia per l'Italia Digitale

21 set 2017

Indice

CAPITOLO 1

Web toolkit

Implementazione di riferimento delle “Linee guida di design per i servizi web della PA”

L’implementazione di riferimento (web toolkit) delle “Linee guida di design“ fornisce componenti *open source* da incorporare nei siti web delle Pubbliche Amministrazioni (PA) per costruire un’interfaccia grafica usabile, accessibile e consistente:

- fogli di stile CSS
- componenti interattivi (Javascript)
- snippet e template HTML

Contribuire al progetto

Se hai suggerimenti o vuoi contribuire direttamente allo sviluppo del toolkit puoi farlo utilizzando il repository GitHub:

https://github.com/italia/ita-web-toolkit

Se hai domande sull’utilizzo o bisogno di aiuto per l’installazione puoi aprire una nuova richiesta.

Obiettivi

- presentare pattern e template in modo da poter essere discussi, condivisi e migliorati recependo i contributi della community e i risultati dei test di usabilità
- chiarire quale aspetto deve avere il sito web per risultare conforme alle indicazioni sul visual design riportate nelle linee guida
- fornire un tool che faciliti una rapida prototipazione dei siti in modo da poter coinvolgere gli utenti fin dalle prime fasi di progettazione
- mettere a disposizione di fornitori e PA materiale riutilizzabile nell’implementazione del frontend del sito

- suggerire un set di componenti dinamici (widget javascript) che è possibile utilizzare per soddisfare i requisiti di accessibilità richiesti dalla normativa
- convogliare, in un unico framework di sviluppo, i contributi della community di design.italia.it, l'esperienza sul campo dei fornitori della PA e le competenze tecniche degli sviluppatori

CAPITOLO 2

Come iniziare

La styleguide

I componenti grafici del toolkit sono presentati tramite una styleguide; puoi visualizzarli navigando dal menu "Componenti":

- **Components**: elementi autonomi dell'interfaccia (form, tipografia, tabelle, griglia responsive, ...)
- **Modules**: elementi che dipendono da altri componenti (header, footer, ...)
- **Templates**: template HTML di pagine intere o parti del layout
- **Utils**: componenti di utilità (margini, padding, colori, tipografia, ...)
- **Icons**: icone personalizzate in formato SVG, PNG e icon font

Nella pagina che illustra ogni elemento è visibile in calce il template HTML che lo realizza.

Utilizzo del toolkit

Per utilizzare i componenti all'interno del tuo sito è necessario includere, oltre l'HTML, i fogli di stile (CSS) e i Javascript (JS) necessari.

Puoi scaricare i file CSS / JS già compilati da incorporare in ogni pagina: **CSS / Javascript già compilati**

In produzione vanno inclusi i file:

```
.
- build.css
- vendor/modernizr.js
- vendor/jquery.min.js
- IWT.min.js
```

facendo riferimento al **template generico di esempio**

File Javascript di componenti opzionali

I file contenuti nella directory `build` del tipo

```
- 0.chunk.js
- 1.chunk.js
- ...
```

sono Javascript associati a componenti dell'interfaccia opzionali; vengono **caricati automaticamente** a runtime solo dove il componente (es. carousel) viene effettivamente utilizzato nell'HTML.

Questo meccanismo permette di incorporare esempi e widget Javascript complessi nel toolkit, senza che ciò impatti sulla dimensione finale del file `IWT.min.js` riducendo quindi i tempi di download e parsing durante il rendering delle pagine web.

Tail file devono esser presenti nella stessa directory dove è contenuto il file `IWT.min.js`; se il percorso di questa directory differisce da quello di default (`/build`), è necessario specificare il percorso alternativo, **relativo alla radice del sito web**, come visibile nel **template generico di esempio**:

```
<!-- sostituire questo percorso con quello
     degli assets javascript nel proprio sito web:
     è il percorso, relativo alla webroot, della directory
     che contiene il file IWT.min.js e i file *.chunk.js -->

<script>__PUBLIC_PATH__ = '/assets/javascript/'</script>
```

Altri contenuti della directory build

- `src/icons`: icone personalizzate in formato SVG, PNG e *icon font*.
- `*.map`: file utili solo in fase di debug e possono essere omessi in produzione.
- `*.styleguide.*`: file che definiscono l'aspetto della styleguide e possono essere omessi in produzione.

Dipendenze esterne

Nella directory `vendor` si trovano i *polyfill* che è consigliabile includere per garantire la compatibilità con i browser obsoleti (IE8/9):

```
.
- polyfill.min.js
- rem.min.js
- respond.min.js
- selectivizr.js
- slice.js
```

La directory `vendor` Contiene inoltre la libreria jQuery, una dipendenza necessaria per alcuni componenti del toolkit.

Molti CMS o framework CSS vengono già distribuiti con una loro versione della libreria `jQuery`; in questo caso non è necessario utilizzare quella del toolkit, a patto che venga inclusa nell'HTML prima del file `IWT.min.js`.

I componenti del toolkit lavorano con versioni di jQuery >= 1.11.x, tuttavia se vuoi garantire la compatibilità con IE8 ricorda di utilizzare versioni di jQuery inferiori alla 2.x.

Personalizzare lo stile (CSS)

Se vuoi personalizzare gli elementi grafici (es. colori) utilizzando i file già compilati dovrai necessariamente **sovrascrivere** le classi del CSS di base (`build.css`) includendo un tuo ulteriore CSS contenente le direttive *custom*; questa metodologia non è ottimale se le personalizzazioni sono particolarmente complesse: in questo caso è consigliato seguire la procedura descritta in "Utilizzare l'ambiente di sviluppo" agendo direttamente sul codice sorgente dei fogli di stile del toolkit per realizzare una *build* personalizzata, prima di integrare il CSS nel layout.

CAPITOLO 3

Tecnologie utilizzate

Il toolkit è basato su alcuni software open source che svolgono diversi task e di cui è bene possedere una conoscenza anche sommaria prima di procedere a modificare i sorgenti.

npm

Il tool npm è utilizzato per la gestione delle dipendenze necessarie sia alla fase di sviluppo / compilazione che per i componenti Javascript integrati lato client (nel browser).

La lista delle librerie utilizzate è visibile nel file `package.json`.

Prima di incorporare nuove dipendenze è buona pratica verificare se siano già presenti nel registro npm in modo da poterle integrare più agevolmente.

SUIT CSS

SUIT CSS è una metodologia di implementazione per i fogli di stile CSS a corredo di un insieme di utilità che ne facilitano la manutenzione.

Relativamente al toolkit:

1. vengono adottate le convenzioni di nomenclatura SUIT CSS nella stesura del codice dei fogli di stile
2. vengono utilizzate le classi di utilità fornite dalla libreria. Si consiglia in particolare di far riferimento alla documentazione online delle classi SUIT CSS con particolare riferimento alla griglia responsive
3. viene utilizzato il preprocessore CSS di SUIT integrato da un insieme di plugin *ad-hoc*; è possibile visualizzare la lista dei plugin nel file `.postcss.js` contenuto nella directory radice del *repository*.

È possibile consultare l' elenco e la documentazione di tutte le classi SUIT CSS.

PostCSS

PostCSS è un tool che permette di manipolare i CSS tramite javascript.

I fogli di stile del toolkit vengono trasformati tramite PostCSS: in questo modo è possibile usufruire di alcuni costrutti non standard che agevolano il mantenimento del codice. Puoi far riferimento alla documentazione online dei singoli plugin:

Verifica della sintassi

Per il *linting* (ovvero, l'analisi del codice in cerca di errori e/o costrutti con sintassi errata) vengono utlizzati i due tool:

- stylelint - per il codice CSS
- ESLint - per il codice Javascript

Generazione della styleguide e dei moduli CSS / JS

Questi due tool sono attivati dagli *script npm* secondo le modalità descritte nel capitolo successivo: "Utilizzare l'ambiente di sviluppo":

- fractal è utilizzato per generare la *styleguide* (l'elenco navigabile dei componenti grafici)
- webpack organizza i moduli CSS / Javascript in modo da poter esser utilizzati in produzione

Non è necessario conoscere il funzionamento di questi software che vengono qui citati per completezza.

CAPITOLO 4

Utilizzare l'ambiente di sviluppo

La procedura qui descritta è rivolta sia a chi vuole **contribuire allo sviluppo del toolkit** sia a chi vuole **utilizzare il CSS personalizzato all'interno di un sito web** per realizzare quindi un "tema" grafico specifico.

Ottenere i sorgenti

Per ottenere i sorgenti puoi scaricare direttamente l'archivio compresso da GitHub.

Un'alternativa migliore è il download dei sorgenti tramite il software di versionamento Git; dopo aver installato Git puoi effettuare una copia locale del repository digitando da linea di comando:

```
git clone https://github.com/italia/ita-web-toolkit
```

L'utilizzo di Git permette di mantenere i sorgenti sincronizzati con i nuovi rilasci del toolkit nonché di contribuire al progetto proponendo di incorporare le proprie modifiche nella linea principale di sviluppo, vedi anche:

Installare il software richiesto

Per poter compilare i sorgenti (CSS / JS) è richiesta l'installazione di Node.js / npm.

Terminata l'installazione, la seguente sequenza di comandi eseguiti all'interno della directory del toolkit completerà il download e l'installazione delle librerie richieste per lo sviluppo:

```
npm install
```

Compilare i sorgenti

Dopo aver eseguito il comando

```
npm run build
```

la directory `build` conterrà i file CSS e Javascript compilati.

Modificare i sorgenti

Per poter visualizzare la styleguide in locale (i template HTML con i diversi componenti grafici) puoi sostituire come ultimo passaggio (al posto di `npm run build`) il comando

```
npm run watch
```

e visitare con il browser la pagina http://localhost:1310

A questo punto puoi modificare i sorgenti: qualsiasi modifica effettuata ai fogli di stile CSS, Javascript e/o template HTML mentre `npm run watch` rimane in esecuzione risulterà immediatamente visibile nel browser **senza dover ricaricare manualmente la pagina o lanciare il comando di build**.

CAPITOLO 5

Elementi dell'interfaccia

Gli elementi dell'interfaccia (componenti), contenuti nella directory `src/components` sono costituiti da:

1. uno (o più) file **CSS**
2. uno (o più) file **Javascript**
3. uno (o più) snippet HTML di esempio (file *.tmpl)

All'interno della directory di ogni componente deve esser presente almeno una tra queste tre componenti.

Il **codice CSS** va collocato in un file:

`src/components/<nome-componente>/index.css`

e verrà automaticamente incluso nella build.

Il **codice Javascript** va collocato in un file

`src/components/<nome-componente>/index.js`

e verrà automaticamente incluso nella build.

Puoi includere ulteriori **assets** (CSS / JS / immagini) contenuti nella directory del componente utilizzando le direttive `require` (nell'index.js) o `@import` (nell' index.css).

Il nome dei file con gli snippet HTML *deve* iniziare con quello del componente, ad esempio:

- `src/components/<nome-componente>/<nome-componente>.tmpl`
- `src/components/<nome-componente>/<nome-componente>--2.tmpl`

Negli snippet HTML è possibile utilizzare i costrutti del linguaggio di templating nunjucks.

I CSS vengono processati tramite PostCSS (e i relativi plugin presentati nella sezione Tecnologie) al momento dell'esecuzione di `npm run build` (o `npm run build:css`).

Nei file Javascript è possibile utilizzare la sintassi ECMAScript 2015.

Regole generali nello sviluppo di nuovi componenti

- i componenti devono essere quanto più possibile indipendenti tra loro
- nei CSS si possono utilizzare le classi di utilità (quelle con prefisso `u-*`), ma è sconsigliato incorporare selettori di altri moduli
- le regole di nomenclatura sono quelle di SUIT CSS
- i componenti Javascript devono dichiarare tutte le loro dipendenze nell'`index.js` (es. tramite i costrutti `import` o `require`)
- è incoraggiato l'utilizzo del costrutto PostCSS `@extend`, ma **esclusivamente** per estendere le classi di utilità (`u-*`)

CAPITOLO 6

Compatibilità con altri framework

Se utilizzi già un framework CSS all'interno del tuo progetto, ma hai intenzione di incorporare *anche* uno o più moduli del toolkit, puoi creare una *build* personalizzata volta a risolvere eventuali incompatibilità che potrebbero incorrere includendo il CSS/JS *as-is*.

Scegliendo selettivamente i singoli moduli prima della fase di compilazione puoi escludere dall'output il codice di quelli che non vuoi utilizzare.

In questo modo si evitano eventuali conflitti con i plugin Javascript e i nomi delle classi CSS e al contempo si riduce la dimensione dei file che il browser deve scaricare per visualizzare le pagine.

Configurazione dinamica dei moduli da includere o escludere

Per escludere dai CSS/JS generati alcuni moduli e/o includerne selettivamente altri edita il file config.js contenuto nella directory radice del toolkit prima di effettuare una nuova `build`.

Il file contiene due liste di **espressioni regolari**:

- l'array `CONFIG.excludes` è la *blacklist* con i percorsi su filesystem dei moduli da escludere
- l'array `CONFIG.includes` è la *whitelist* con i percorsi su filesystem dei moduli da includere

Esempi di utilizzo di blacklist / whitelist

Per includere solo il modulo che realizza la visualizzazione ad albero (*treeview*) più uno stile di base per la lista dei link (*linklist*) utilizza la seguente whitelist:

```
CONFIG.includes = [
    '(.*)theme(.*)',
    '(.*)utils(.*)',
    '(.*)linklist(.*)',
    '(.*)treeview(.*)',
]
```

Per escludere i moduli che non verrebbero utilizzati (es. carousel, spid), puoi utilizzare una blacklist:

```
CONFIG.excludes = [
    '(.*)carousel(.*)',
    '(.*)spid(.*)',
]
```

Per escludere `normalize.css` (perché eventualmente già incluso dal tuo framework):

```
CONFIG.excludes = [
    '(.*)normalize(.*)',
]
```

Dopo aver modificato il file `config.js`:

- eseguendo `npm run build:css` si ottiene in output il file `build/build.css`
- eseguendo `npm run build:js` si ottiene in output il file `build/IWT.min.js`

L'output conterrà esclusivamente il CSS e il Javascript dei moduli che hai configurato.

CAPITOLO 7

Struttura del filesystem

La directory `src` contiene le sotto-directory:

- **components** - CSS/Javascript/HTML dei componenti autonomi dell'interfaccia
- **modules** - CSS/Javascript/HTML degli elementi complessi dell'interfaccia
- **fonts** - il font Titillium Web e le direttive CSS @font-face relative
- **icons** - le icone svg / png utilizzate nel progetto e il CSS per l'*icon font*
- **legacy** - CSS per supportare i vecchi browser
- **scripts** - Javscript "globali" (non legati a una particolare componente dell'interfaccia)
- **templates** - template HTML per elementi del layout e pagine web
- **themes** - il foglio di stile con le personalizzazioni per uno specifico tema (es. colore principale dal quale viene declinata l'intera *palette*)
- **utils** - CSS contenenti classi di utilità generica (margini, padding, tipografia responsive, ...)
- **vendor** - Javascript da incorporare per garantire un minimo grado di compatibilità con i browser obsoleti (IE8/9)

```
.
- docs
- src
    - components
    |   - ...
    - fonts
    |   - titillium
    |       - font
    - icons
    |   - ita
    |       - font
    |       - png
    |       - svg
    - legacy
    - modules
```

```
|   - ...
- scripts
- templates
|   - layout
|   - pages
|   - ...
- themes
|   - pac
- utils
|   - borders
|   - colors
|   - ...
- vendor
```

CAPITOLO 8

Sviluppare un tema personalizzato

All'interno della directory `src/themes` è presente un CSS di esempio in cui viene mostrato come sovrascrivere le variabili utilizzate all'interno del toolkit:

```
/* src/themes/pac/index.css */
:root {
  --Color-primary: #a8336c;

  --Pac-font-family-sans: Titillium Web, HelveticaNeue-Light, Helvetica Neue Light,
                          Helvetica Neue, Helvetica, Arial, Lucida Grande, sans-serif;
}

.t-Pac {
  font-family: var(--Pac-font-family-sans);
}
```

Similmente è possibile sovrascrivere qualsiasi variabile contenuta nei moduli CSS, modificando ad esempio i colori che caratterizzano l'aspetto grafico dei vari elementi.

Variabili CSS

Alcune variabili che è possibile sovrascrivere:

```
--Color-focus
--Color-primary

--Button-default-bg
--Button-default-color
--Button-default-border
--Button-danger-bg
--Button-danger-color
--Button-danger-border
--Button-info-bg
```

```
--Button-info-color
--Button-info-border

--Table-border-color
--Table-stripe-color

--Prose-color-link
--Prose-color-link-hover
--Prose-color-link-visited

--Form-input-border-color
--Form-state-invalid-border-color
--Form-state-invalid-label-color
--Form-state-invalid-message-color
--Form-state-warning-border-color
--Form-state-warning-label-color
--Form-state-warning-message-color

--Alert-error-bg
--Alert-error-color
--Alert-error-border
--Alert-error-background-image
--Alert-warning-bg
--Alert-warning-color
--Alert-warning-border
--Alert-warning-background-image
--Alert-success-bg
--Alert-success-color
--Alert-success-border
--Alert-success-background-image
--Alert-info-bg
--Alert-info-color
--Alert-info-border
--Alert-info-background-image

--Offcanvas-width
--Offcanvas-max-width

--Linklist-border-color
--Linklist-link-padding
--Linklist-link-arrow-width

--Treeview-link-padding
--Treeview-link-arrow-width
--Treeview-link-arrow-up
--Treeview-link-arrow-down

--Hero-height
--Hero-background-img
  ...
```

Personalizzare i colori

La palette di colori è generata automaticamente a partire da un colore di base (`--Color-primary`) (`src/utils/colors/index.css`) applicando tinte e tonalità secondo quanto descritto nel relativo paragrafo delle Linee Guida;

in questo modo è sufficiente sovrascrivere la variabile `--Color-primary` per ottenere l'**intera palette**.

Per applicare font e colori personalizzati bisogna:

1. modificare il file del tema `src/themes/pac/index.css` sovrascrivendo le variabili CSS
2. effettuare una nuova *build* del toolkit tramite `npm run build` (vedi "Utilizzare l'ambiente di sviluppo")
3. incorporare i CSS generati (contenuti nella directory `build`) nei template HTML

In alternativa, è possibile modificare i CSS *mentre* il comando `npm run watch` è in esecuzione (senza effetuare quindi una nuova build per ogni modifica); in questo caso è possibile visualizzare immediatamente il risultato delle modifiche accedendo con il browser all'indirizzo della styleguide (`http://localhost:1310`) e navigando tra i componenti dell'interfaccia.

Nomenclatura delle regole CSS personalizzate

Tutte le modifiche che riguardano uno specifico **tema** vanno applicate assegnando al tag `body` la **classe** utilizzata in `src/themes/*/index.css` (nel caso specifico `t-Pac`, ma può esser scelta arbitrariamente).

```
<!-- nel template HTML -->
<body class="t-Pac"> ...
```

Ciò significa che tale classe deve comparire **necessariamente** come prefisso nei selettori di ogni nuova regola CSS introdotta dal tema (e/o qualsiasi modifica ai moduli effettuata sovrascrivendo le regole originali).

Ad esempio, all'interno di `src/themes/index.css`, per applicare un `padding` alla classe `.Alert` dovrai utilizzare:

```
.t-Pac .Alert {
  padding-right: 1em;
}
```

e **non** quindi:

```
/* No !!! Mai senza prefisso ! */
.Alert {
  padding-right: 1em;
}
```