Jurnal Teknik Informatika (JUTIF)
Vol. 3, No. 2, April 2022, hlm. 387-394

DOI: https://doi.org/10.20884/1.jutif.2022.3.2.212
p-ISSN: 2723-3863
e-ISSN: 2723-3871

IMPLEMENTATION OF WEBIX DYNAMIC SCRIPTING FOR WEB-BASED FORM APP DEVELOPMENT AS AN ALTERNATIVE TO DATA-COLLECTING APPLICATIONS

Cahyo Nugroho[*1], Yerymia Alfa Susetyo[2]

[1, 2]Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana
Email: [1]672018039@student.uksw.edu, [2]yeremia.alfa@uksw.edu

(Naskah masuk: 14 Maret 2022, Revisi: 18 Maret 2022, Diterbitkan: 25 April 2022)

Abstract

In the 4th Industrial revolution, science and technology became increasingly advanced and more evolved. Besides that, data also became a new commodity that can be used to accomplish various things. One of the Indonesian retail companies that used data to improve the quality of the product is PT. XYZ. The oracle form is an application of form that used by PT XYZ for data collection. But the oracle form can no longer complied the company's needs, so the company needs an alternative form application that can be used to replace the oracle form. Therefore, a web-based form application was created and combined with dynamic scripting. Dynamic scripting was a proggraming technique to avoid writing the same code repeatedly. In this study, dynamic scripting's application was combined with the webix framework, resulting in a new framework that can shorten the code line and increase the time efficiency of the building of the form application.

Keywords: *data, dynamic scripting, form, framework, webix*

IMPLEMENTASI WEBIX DYNAMIC SCRIPTING DALAM PEMBUATAN APLIKASI FORM BERBASIS WEB SEBAGAI ALTERNATIF APLIKASI PENGUMPULAN DATA

Abstrak

Pada era perkembangan industri 4.0, ilmu pengetahuan dan tekonolgi menjadi semakin maju dan berkembang selain itu data menjadi sebuah komoditas baru yang dapat dimanfaatkan untuk mencapai berbagai hal. Salah satu perusahaan ritel di Indonesia yang memanfaatkan data untuk dapat meningkatkan kualitas produk yang diberikan adalah PT. XYZ. *Oracle Form* merupakan sebuah aplikasi *form* yang digunakan oleh PT. XYZ untuk menjadi solusi pengumpuluan data. Namun *Oracle Form* sudah tidak dapat lagi memenuhi kebutuhan perusahaan, sehingga perusahaan memerlukan sebuah aplikasi *form* alternatif yang dapat digunakan untuk menggantikan *Oracle Form*. Oleh karena itu, dibuatlah sebuah aplikasi *form* berbasis *web* yang memanfaatkan *dynamic scripting*. *Dynamic scripting* merupakan sebuah teknik pemrograman untuk menghindari penulisan *code* yang sama secara berulang. Dalam penelitian ini, penerapan *dynamic scripting* dipadukan dengan *framework webix* sehingga menghasilkan sebuah *framework* baru yang dapat mempersingkat baris *code* dan meningkatakan efisiensi waktu dalam pembuatan aplikasi *form*.

Kata kunci: *data, dynamic scripting, form, framework, webix*

1. PENDAHULUAN

Pada era digital saat ini, data dipandang sebagai komoditas baru yang dapat memberikan pengaruh besar dalam perkembangan teknologi yang ada. Di era ini perusahaan berusaha mengembangkan sebuah teknologi untuk dapat melakukan pengolahan data [1]. Tujuannya adalah mendapatkan informasi yang berguna untuk meningkatkan mutu dan kualitas pelayanan perusahaan. Pengolahan data juga dapat menghasilkan suatu informasi yang dapat membantu perusahaan dalam melakukan pengambilan keputusan [2]. Pada tahun 2017, data yang ada di dunia sudah mencapai 1.8 *Zetta Byte* atau setara dengan 18×10^8 *TeraByte* [3]. Semakin besar dan kompleks sebuah data, semakin sulit bagi perusahaan untuk melakukan pengolahan data. Sehingga perusahaan memerlukan sebuah aplikasi yang dapat melakukan penginputan data. *Form* merupakan salah satu aplikasi yang diperlukan

dalam internal perusahaan [4][5]. Aplikasi *form* dapat membantu perusahaan dalam melakukan penginputan data dikarenakan *form* terdiri dari beberapa *field* yang dapat dikustomisasi sesuai dengan kebutuhan yang ada [6].

PT. XYZ merupakan salah satu perusahaan ritel terkemuka di Indonesia. Perusahaan ini bergerak pada bidang industri penyedia kebutuhan sehari-hari melalui pengelolaan waralaba *minimarket*. Saat ini, PT. XYZ sudah memakai *Oracle Form* untuk menjadi solusi kebutuhan penginputan data. *Oracle Form* merupakan aplikasi *form* dari *Oracle* yang berguna untuk melakukan penginputan data [7]. *Oracle Form* memanfaatkan *database Oracle* untuk dapat berfungsi dengan baik [8]. Namun dalam perjalanannya, perusahaan memerlukan *database* lain selain *Oracle* untuk memenuhi kebutuhannya. Sehingga perusahaan tidak dapat lagi menggunakan *Oracle Form*. Hal ini menuntut perusahaan untuk menggunakan alternatif teknologi lain untuk menggantikan *Oracle Form* serta melakukan proses migrasi pada aplikasi *Oracle Form* yang masih digunakan.

Webix merupakan sebuah *framework Javascript* yang dikembangkan oleh XBSoftware. *Framework* ini digunakan untuk pembuatan *user interface* pada aplikasi berbasis *web*. *Webix* memungkinkan *developer* untuk membuat tampilan aplikasi *web* yang kompleks menjadi lebih mudah [9]. *Webix* sangat memudahkan *developer* dalam melakukan pembuatan *user interface*. Namun dalam pengembangan aplikasi *form* akan ada banyak *element* yang digunakan lebih dari sekali misalnya *text field*. Untuk membuat *element text field* lain dengan *label* yang berbeda, *developer* perlu menulis ulang *syntax* yang sama seperti element *text field* sebelumnya dengan perbedaan hanya ada pada *label* saja. Hal ini membuat *code* menjadi semakin panjang dan sulit dipahami. Oleh karena itu, *dynamic scripting* digunakan untuk mengatasi masalah ini. *Dynamic Scripting* merupakan teknik pemrograman dengan memanfaatkan fungsi secara *reusable* untuk menghindari penulisan *code* yang sama secara berulang sehingga proses pembuatan aplikasi menjadi lebih cepat [10]. Lebih sedikit *code* juga berarti lebih mudah untuk melakukan *debug* saat terjadi *error*.

Pembuatan *framework* dalam aplikasi *form* berbasis web ini dilakukan dengan memanfaatkan kombinasi antara *dynamic scripting* dan *webix* sehingga dapat meningkatkan efisiensi waktu dan mempersingkat baris *code*. *Framework* ini diharapkan dapat membantu perusahaan dalam melakukan proses migrasi dari *Oracle Form* ke aplikasi *form* berbasis *web*.

Berdasarkan latar belakang masalah yang ada maka dibuatlah sebuah rumusan masalah bagaimana melakukan implementasi *Webix Dynamic Script*ing dalam pembuatan aplikasi *Form* berbasis *web*. Dalam penelitian ini juga diberikan batasan masalah yaitu penelitian ini hanya berfokus pada pembuatan *dynamic scripting* di sisi *user interface* saja.

2. TINJAUAN PUSTAKA

Pada penelitian yang berjudul "Aplikasi *E-Learning* Siswa SMK berbasis *Web*", peneliti mengimplementasikan *form* ke dalam aplikasi *E-Learning*. Sebelumnya para guru melakukan pendataan identitas siswa, nilai, materi, dan data kelas secara manual. Hal ini menyebabkan terjadinya data yang tidak sinkron antara guru satu dengan yang lainnya [11]. Implementasi *form* dalam aplikasi *E-Learning* mempermudah para guru dalam melakukan *input* data. Selain itu implementasi *form* dalam *E-Learning* juga dapat mengatasi masalah data yang tidak sinkron antara guru satu dengan yang lainnya. Penelitian ini memiliki kesamaan dengan penelitian yang akan dilakukan, yaitu mengimplementasikan *form* dalam aplikasi berbasis *web*. Hal yang menjadi pembeda adalah penelitian ini hanya menggunakan *Javascript* dalam pembangunan *user interface*, sementara penelitian yang akan dilakukan menggunakan salah satu *framework Javascript* yaitu *Webix*.

Pada penelitian yang dilakukan oleh Taewook Oh, et.al yang berjudul *"A Generalized Framework for Automatic Scripting Language Parallelization"* pada tahun 2017, *data scientist* biasanya perlu membuat banyak *script* dalam melakukan pengolahan data. Hal ini terjadi karena banyaknya data yang harus diolah [12]. *Dynamic Scripting* dimanfaatkan untuk membantu *data scientist* dalam melakukan pembuatan *script*. Sehingga *data scientist* tidak perlu membuat banyak *script* untuk melakukan pengolahan data. Konsep pada penelitian ini tidak hanya dapat digunakan dalam bidang *data science*, tapi juga dapat diimplementasikan pada penelitian yang akan dilakukan. Melalui pemanfaatan *Dynamic Scripting*, *developer* tidak perlu melakukan penulisan *code* yang panjang. Sehingga dapat meningkatkan efisiensi waktu.

Kemudian pada penelitian yang berjudul "Integrasi *Framework Kivy* dan *Webix* pada Pembangunan *Framework Mobile Web Easy Development System*" oleh Vio Ayu Oktavia Putu Warisman dan Nina Setiyawati, peneliti melakukan pembuatan sebuah *framework* yang memanfaatkan *Kivy* dan *Webix* [13]. Dengan menggunakan *Webix* peneliti dapat membuat sebuah tampilan dengan desain yang menarik dan memberikan kebebasan dalam melakukan kustomisasi. Hal ini bisa dilakukan karena *Webix* memiliki banyak *element* yang dapat dimanfaatkan dalam melakukan pengembangan aplikasi berbasis *web*. Penelitian ini memanfaatkan *Webix* dalam pembuatan aplikasi pada *platform mobile*. Hal ini sedikit berbeda dengan penelitian yang akan dilakukan, dimana penelitian lebih menitikberatkan pada penggunaan *Webix* dalam pengembangan aplikasi pada *platform web*.

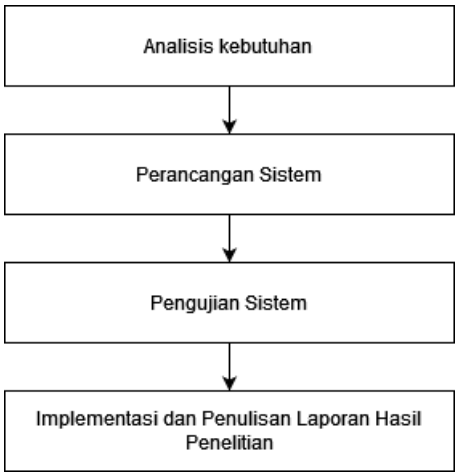
Berdasarkan hasil dan kesimpulan dari penelitian–penelitian terdahulu tentang penerapan *form*, *Dynamic Scripting* dan penerapan *Webix*, maka dilakukanlah penelitian untuk mengimplementasikan *Webix* dan *Dynamic Scripting* dalam pembuatan aplikasi *form* berbasis *web*.

Javascript merupakan bahasa pemrograman yang paling banyak digunakan dalam pembuatan aplikasi berbasis *web* berdasarkan survey *Stack Overflow* pada tahun 2018. Selain sebagai bahasa pemrograman untuk membangun aplikasi *web*, *Javascript* juga digunakan untuk membangun aplikasi *server side* dan aplikasi *mobile* [14]. *Javascript* menjadi bahasa dasar untuk pembuatan banyak *framework* seperti *Webix, React.js, React Native, Vue, Angular* dan masih banyak lagi.

Webix merupakan sebuah *framework* yang memanfaatkan bahasa pemrograman *Javascript* sebagai dasarnya. *Webix* bersifat *cross-platform* sehingga *Webix* dapat berjalan di *mobile* maupun *web*. *Framework* ini memungkinkan *developer* untuk membuat sebuah aplikasi kompleks namun dengan *code* yang lebih sedikit [15].

3. METODE PENELITIAN

Ketika penelitian ini dilakukan, perusahaan masih menggunakan *Oracle Form* dan banyak karyawan sudah sangat terbiasa dengan *workflow* yang ada di dalam *Oracle Form*. Hal ini menjadi tantangan tersendiri, mengingat pembuatan metode penelitian harus sesuai dengan kebutuhan perusahaan. Metode penelitian yang dibuat dapat dilihat dari diagram pada Gambar 1.

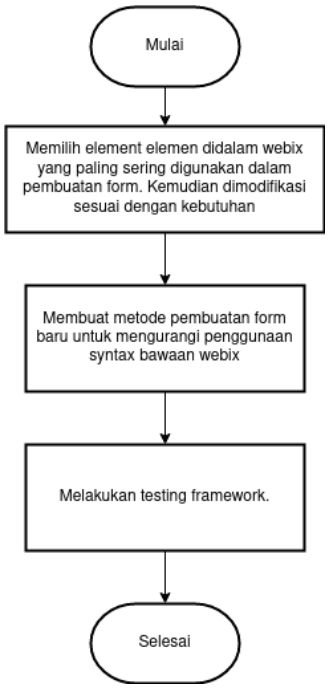


Gambar 1. Tahapan Penelitian

Tahap pertama adalah melakukan analisis kebutuhan. Tahapan ini dilakukan untuk menganalisis permasalah yang ada dan menentukan solusi yang diperlukan. Analisis kebutuhan dilakukan dengan cara mengamati fitur yang ada pada aplikasi *Oracle Form*. Selain itu juga dilakukan diskusi dengan praktisi *IT* (*Information Technology*) perusahaan. Tahap kedua adalah melakukan perancangan sistem. Perancangan sistem dilakukan sesuai dengan hasil dari analisis yang sebelumnya sudah dilakukan. Tahap ketiga adalah melakukan pengujian sistem. Aplikasi yang sudah dibuat akan dilakukan pengujian apakah sudah sesuai dengan kebutuhan atau belum. Tahap terakhir adalah melakukan implementasi dan pembuatan laporan tentang aplikasi yang sudah dibuat.

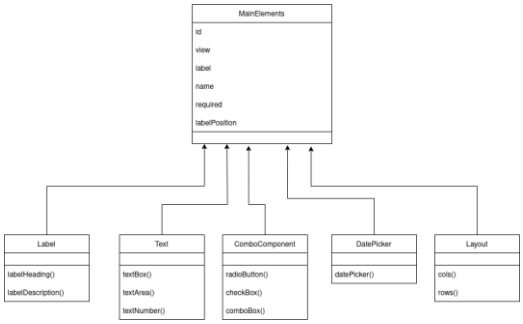
4. HASIL DAN PEMBAHASAN

Penelitian ini menghasilkan *framework* yang dapat dimanfaatkan untuk membangun sebuah aplikasi *form* berbasis *web* dengan memanfaatkan *webix* dan *dynamic scripting*. Teknologi yang digunakan dalam melakukan pembuatan *framework webix dynamic scripting* memanfaatkan bahasa pemrograman *Javascript* dan *framework webix 9.1 open source edition*. *Framework* yang dihasilkan dapat mempersingkat jumlah baris *code* dan mengurangi waktu yang diperlukan untuk membuat aplikasi *form*. Pembuatan framework dapat dilihat dari diagram alir pada Gambar 1.



Gambar 2. Diagram alir pembuatan *framework*

Langkah awal yang dilakukan untuk melakukan pembuatan *framework* adalah melakukan pemilihan *element* dalam *webix* yang biasa digunakan untuk pembuatan *form*. *Element* yang dipilih kemudian dimodifikasi dan disesuaikan dengan *element* yang ada pada *oracle form* untuk mempermudah *user* dalam melakukan pembuatan *form*. Secara *default webix* menggunakan *format syntax json* untuk melakukan pembuatan *element*. Modifikasi dilakukan dengan cara mengubah setiap *element* yang dibutuhkan menjadi bentuk kelas dan fungsi. Susunan setiap fungsi dan kelas dapat dilihat pada Gambar 3.



Gambar 3. *Class diagram* element *webix*

Setiap *element* akan dibuat menjadi sebuah fungsi. Fungsi-fungsi tersebut kemudian diletakan di dalam kelas yang sesuai dengan element yang ada. Pada *framework* ini dibagi menjadi 5 *element* besar yaitu *Label*, *Text*, *ComboComponent*, *DatePicker*, dan *Layout*. Setiap *element* besar kemudian akan memiliki *element* yang lebih mendetail lagi dalam bentuk fungsi yang ada dalam kelas tersebut. Dapat dilihat dari diagram pada kelas *Label*, kelas ini hanya akan terdiri dari element yang digunakan untuk membuat *label* begitu juga dengan kelas kelas yang lainnya. Setiap kelas merupakan *child class* dari *parent class* yaitu *MainElements*. Pada *MainElements* terdapat atribut yang sering digunakan ketika membuat element. Atribut atribut ini kemudian akan diwariskan kepada masing masing *child class* dengan tujuan untuk mengurangi penulisan *code* secara berulang. Sehingga ketika ditemukan atribut yang sama ketika melakukan pembuatan *element* lain maka atribut yang ada di *MainElements* akan di-*reuse*. Hasil dari modifikasi *element* dapat dilihat pada potongan kode program 1.

Kode program 1. Hasil modifikasi *element webix*

```
class MainElements {
  baseElements(id, view,
    label, required = false) {
      let baseElements = {
       id: id,
       view: view,
       label: label,
       name: id,
       required: required,
       labelPosition: "top"
      }
    return baseElements;
  }
}
class Text extends MainElements {
  textBox(id, label, required) {
  let textBox = super.baseElements(
    id,"text", label, required);
     return textBox;
   }
}
```

Modifikasi *element* ini menghasilkan *element webix* yang memiliki jumlah baris *code* yang lebih sedikit jika dibandingkan dengan *element webix* tanpa modifikasi. Perbandingan antara sebelum modifikasi dengan sesudah modifikasi, dapat dilihat pada potongan kode program 2 dan 3

Kode program 2. Sebelum modifikasi *element webix*

```
var text = webix.ui({
    id:"id",
    view:"text",
    value:"Your name",
    label:"Name",
    required:true,
    labelPosition:"top",
});
```

Kode program 3. Sesudah modifikasi *element webix*

```
textBox(id, label, required) {
 let textbox = super.baseElements(id,
    "text",label,required)
    return textBox;
}
```

Dapat dilihat pada potongan kode program 3, jumlah baris *code* yang dibutuhkan untuk membuat *element webix* tanpa me-*reuse* atribut yang ada adalah 8 baris. Sedangkan dapat dilihat juga pada potongan kode program 4, hasil dari modifikasi *element webix* hanya membutuhkan 4 baris saja. Hal ini dapat dicapai karena atribut *element* yang sering dipakai dalam *webix* seperti *id*, *view*, *label*, *name*, *required*, dan *labelPostion* tidak perlu lagi ditulis pada saat pembuatan fungsi pada *child class*.

Dalam pembuatan *element webix* terkadang ada beberapa *element webix* yang memilki atribut unik dan belum ada pada *MainElements*. Untuk dapat menambahkan atribut unik ini maka digunakanlah fungsi bawaan *Javacript* yaitu *spread*. Dengan menggunakan *spread* maka penambahan atribut unik yang belum ada pada kelas *parent* dapat dilakukan tanpa mengubah atribut yang ada dalam kelas *parent*. Hal ini dapat dilihat pada baris 3 pada potongan kode 4.

Kode program 4. *Element* modifikasi yang memerlukan atribut unik

```
radioButton(id, label, options, required){
 let radioButton = super.baseElements(id,
  "radio",label,required);
 radioButton={...radioButton,...{options:
  options}};
 return radioButton;
}
```

Langkah selanjutnya setelah melakukan modifikasi *element* menjadi berbentuk fungsi dan kelas adalah membuat metode pembuatan *form* yang baru. Hal ini dilakukan karena *element* yang telah dimodifikasi tidak dapat dipanggil dengan menggunakan *syntax* bawaan *webix*. Maka diperlukan sebuah metode baru untuk memanggil *element element* yang sudah di modifikasi.

Pembuatan *form* baru memerlukan sebuah *file .js* baru yang terdiri dari dua bagian utama yaitu *headerForm* dan *bodyForm*. Kedua *element* ini merupakan sebuah *array* yang menampung *element element* yang sudah dimodifikasi. Susunan *file .js* dapat dilihat pada potongan kode 5.

Kode program 5. Metode pembuatan *form* baru

```
let headerForm = [
 label.labelHeading("Form Title", "left"),
 label.labelDescription("Lorem ipsum dolor
  sit amet, adipiscing elit."),
]
let bodyForm = [
 textBox.textBox("first_name", "First
Name",
  true),
 textBox.textBox("last_name", "Last
Name"),
 textBox.textNumber("number_field",
"Number
  Field", true),
]
```

Pada potongan kode 5 baris 1 terdapat *variable headerForm*. *Variable* merupakan bagian dari *form* yang digunakan untuk menampung *element* judul dan deskripsi. *HeaderForm* merupakan sebuah *array* yang berisi *element labelHeading* dan *labelDecription*. Kedua element ini digunakan untuk membuat judul dan deskripsi *form*.

Kemudian pada potongan kode 5 baris 5 terdapat *variable bodyForm*. *Variable* ini merupakan bagian utama dari *form*. Karena pada bagian ini menampung *element element* utama dari sebuah form seperti *textBox*, *textArea*, *radioButton*, dan lain lain. *BodyForm* tidak memiliki batasan berapa banyak *element* yang ditampung. Banyaknya *element* disesuaikan dengan kebutuhan pembuatan *form*.

Kode program 6. File *baseHTML*

```
<head>
   <script src="form.js"></script>
</head>
<body>
  <script type="text/javascript"
charset="utf-8">
    webix.ui({
        id: "formLayout",
        type: "space",
        rows:
            [
                {
                 id: "header",
                 view: "form",
                 elements: headerForm
                },
                {
                 id: "body",
                 view: "form",
                 scroll: true,
                 elements: bodyForm
                },
            ]
       });
   </script>
</body>
```

Potongan kode 6 merupakan potongan kode *html* yang digunakan sebagai *base* dari *form* yang dibuat. *File .js* yang sebelumnya sudah dibuat berisi *element* pada potongan kode 5 kemudian akan dihubungkan dalam *file baseHtml*. Hal ini dapat dilihat pada baris 2.

Pada *file .js* yang sebelumnya dibuat sudah dijelaskan bahwa terdapat 2 *element* utama yaitu *headerForm* dan *bodyForm*, kedua *element* ini akan dipanggil kembali di dalam *file baseHtml* pada baris 14 dan 19. Dengan begini pembuatan *form* tidak perlu menggunakan banyak *file html* karena pemanggilan *file .js* dapat dilakukan secara dinamis. Sehingga hanya membutuhkan 1 *file baseHtml* untuk dapat membuat banyak *form*.

Langkah terakhir adalah melakukan *testing framework*. *Testing* dilakukan dengan 2 cara. Cara pertama dengan melakukan *black box testing* pada *framework* yang sudah dibuat. *Black box testing* merupakan teknik pengujian perangkat lunak dengan cara melakukan pengamatan terhadap *input* dan *output* tanpa mempedulikan struktur kode [16][17]. *Testing* ini dilakukan untuk mengetahui apakah perangkat lunak dapat berfungsi dengan baik atau tidak. Cara kedua adalah dengan membandingkan pembuatan *form* menggunakan *webix* tanpa *dynamic scripting* dan pembuatan *framework* menggunakan *webix dynamic scripting*. Dari perbandingan yang dilakukan dapat dilihat apakah dengan menggunakan *webix dynamic scripting* jumlah baris kode benar benar berkurang. Table 1 merupakan hasil dari *black box testing* yang telah dilakukan.

Tabel 1. Hasil pengujian *blackbox*

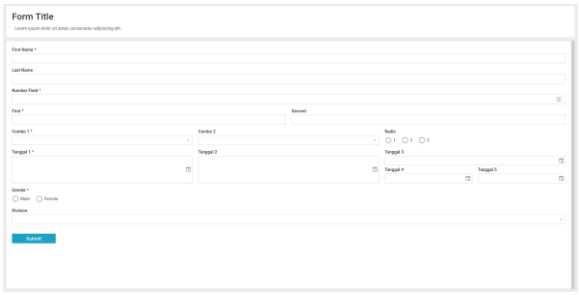
No	Pengujian	Hasil yang diharapkan	Hasil Pengujian	Kesimpulan
1.	Membuat element labelHeading menggunakan webix dynamic scripting.	Teks judul muncul pada bagian atas form.	Teks judul berhasil muncul pada bagian atas form.	Valid.
2.	Membuat element labelDescription menggunakan webix dynamic scripting.	Teks deskripsi muncul pada bagian atas form, dibawah judul form.	Teks deskripsi berhasil muncul dibawah teks judul form.	Valid.
3.	Membuat element textBox menggunakan webix dynamic scripting.	Sebuah text box muncul pada bagian body form.	Text box berhasil muncul di bagian body form.	Valid.
4.	Membuat element textArea menggunakan webix dynamic scripting.	Sebuah text area muncul pada bagian body form.	Text area berhasil muncul di bagian body form.	Valid.
5.	Membuat element textNumber menggunakan webix dynamic scripting.	Sebuah text box yang hanya bisa diisi dengan angka muncul pada bagian body form.	Text box berhasil muncul dan hanya bisa diisi dengan angka saja	Valid.
6.	Membuat element radioButton menggunakan webix dynamic scripting.	Sebuah combo element berbentuk radio button muncul pada bagian body form.	Combo dalam bentuk radio button berhasil muncul pada bagian form.	Valid.
7.	Membuat element checkBox menggunakan webix dynamic scripting.	Sebuah checkbox muncul pada bagian body form.	Checkbox berhasil muncul pada bagian body form.	Valid.
8.	Membuat element comboBox menggunakan webix dynamic scripting.	Sebuah combo element berbentuk combo box muncul pada bagian body form.	Combo box berhasil muncul pada bagian body form.	Valid.
9.	Membuat element datePicker menggunakan webix dynamic scripting.	Sebuah element date picker untuk pemilihan tanggal muncul pada bagian body form.	Element date picker berhasil muncul pada bagian body form.	Valid.
10.	Membuat element cols menggunakan webix dynamic scripting.	Element element yang diletakan dalam element cols posisinya akan berjejer kesamping.	Element element dalam cols posisinya berhasil menjadi berjejer kesamping.	Valid.
11.	Membuat element rows menggunakan webix dynamic scripting.	Element element yang diletakan dalam rows posisinya akan menurun.	Element element dalam rows posisinya berhasil menjadi menurun	Valid.

Pada tabel 1 dapat dilihat *framework* dapat digunakan dengan baik sebagaimana fungsinya. Kemudian untuk hasil perbandingan dapat dilihat dari tabel 2.

Tabel 2. Hasil perbandingan *webix* tanpa *dynamic scripting* dan dengan *dynamic scripting*.

Percobaan	Pembuatan form webix tanpa dynamic scripting.	Pembuatan form webix dengan dynamic scripting.
Pembuatan radioButton	`{ view:"radio", label:"Branch", value:1, options:[ {"Master"} ] }`	`comboComponent.radioButton("radio","Branch",["Master"])`
Pembuatan texBox	`{ view:"text", id:"first_name, label:"Name", required:true }`	`textBox.textBox("first_name","First Name",true)`
Pembuatan comboBox	`{ view:"combo", id:"combo", label: 'Combo', value:"One", options:["One", "Two"] }`	`comboComponent.comboBox("combo","Combo",["One", "Two")`

Hasil dari *form* yang sudah dibuat menggunakan *framework* dapat dilihat pada Gambar 4.



Gambar 4. *Screenshot hasil pembuatan form menggunakan framework yang telah dibuat*

5. KESIMPULAN DAN SARAN

Berdasarkan penelitian yang sudah dilakukan, peneliti menyimpulkan bahwa penerapan *Webix Dynamic Scripting* dapat mempersingkat jumlah baris kode yang diperlukan dalam melakukan pembuatan aplikasi *form* berbasis *web*. Hal ini dikarenakan *dynamic scripting* mengurangi pemanggilan atribut secara berulang. Atribut yang sama akan dibuat dalam sebuah kelas *parent* yang kemudian diwariskan ke kelas *element child*. Selain itu pemisahan kode yang bersifat statis dan yang bersifat dinamis juga berkontribusi dalam pengurangan baris kode. Dengan berkurangnya baris kode maka dapat membantu *developer* dalam melakukan pengembangan aplikasi *form* berbasis *web*. Saran untuk penelitian selanjutnya adalah mencoba untuk menerapkan dynamic scripting pada *framework user interface javascript* lainnya misalnya *angular*, *vue*, ataupun *react*.

DAFTAR PUSTAKA

[1] K. Vassakis, E. Petrakis, and I. Kopanakis, "Big data analytics: Applications, prospects and challenges," *Lect. Notes Data Eng. Commun. Technol.*, vol. 10, pp. 3–20, 2018, doi: 10.1007/978-3-319-67925-9_1.

[2] M. M. Gultom and Maryam, "Sistem Informasi Penjualan Material Bangunan Pada Toko Bangunan Berkah," *J. Tek. Inform.*, vol. 1, no. 2, pp. 79–86, 2020, doi: 10.20884/1.jutif.2020.1.2.19.

[3] R. Dautov and S. Distefano, "Quantifying volume, velocity, and variety to support (Big) data-intensive application development," *Proc. - 2017 IEEE Int. Conf. Big Data, Big Data 2017*, vol. 2018-Janua, pp. 2843–2852, 2017, doi: 10.1109/BigData.2017.8258252.

[4] B. Mulatsih, "Penerapan Aplikasi Google Classroom , Google Form , Dan Quizizz Dalam Pembelajaran Kimia Di Masa Pandemi Covid-19 Application of Google Classroom , Google Form and Quizizz in Chemical Learning During the Covid-19 Pandemic," *ideguru J. Karya Ilm. Guru*, vol. 5, no. 1, pp. 16–26, 2020, [Online]. Available: https://jurnal-dikpora.jogjaprov.go.id/index.php/jurnalideguru/article/view/129.

[5] P. D. A. Pamungkas, "ISO 9126 Untuk Pengujian Kualitas Aplikasi Perpustakaan Senayan Library Management System (SLiMS)," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 2, no. 2, pp. 465–471, 2018, doi: 10.29207/resti.v2i2.398.

[6] N. Ihsan, V. A. Tiwow, and M. Saleh, "Pemanfaatan Aplikasi Google Form dalam Monitoring Kegiatan Kuliah pada Program Studi Fisika Universitas Negeri Makassar," *Semin. Nas. Fis. PPs UNM*, vol. 2, pp. 21–24, 2020, [Online]. Available: http://ojs.unm.ac.id/semnasfisika.

[7] "Oracle Forms." https://www.oracle.com/application-development/technologies/forms/forms.html (accessed Feb. 22, 2022).

[8] K. Garcés *et al.*, "White-box modernization of legacy applications: The oracle forms case study," *Comput. Stand. Interfaces*, vol. 57, no. December 2016, pp. 110–122, 2018, doi: 10.1016/j.csi.2017.10.004.

[9] "Webix: JavaScript UI Library; HTML5 JS framework, 102 UI Web Widgets." https://webix.com/ (accessed Nov. 04, 2021).

[10] S. Ben Sassi and G. Goos, *Reuse in Emerging Software Engineering Practices*. 2020.

[11] Marlina, Masnur, and M. Dirga.F, "Aplikasi E-Learning Siswa Smk Berbasis Web," *J. SINTAKS Log. Vol.*, vol. 1, no. 1, pp. 2775–412, 2021.

[12] T. Oh, S. R. Beard, N. P. Johnson, S. Popovych, and D. I. August, "A Generalized Framework for Automatic Scripting Language Parallelization," *Parallel Archit. Compil. Tech. - Conf. Proceedings, PACT*, vol. 2017-Septe, pp. 356–369, 2017, doi: 10.1109/PACT.2017.28.

[13] N. Setiyawati and V. A. O. P. Warisman, "Integrasi Framework Kivy dan Webix pada Pembangunan Framework Mobile Web Easy Development System," *JUITA J. Inform.*, vol. 8, no. 2, p. 253, 2020, doi: 10.30595/juita.v8i2.7464.

[14] A. Wirfs-Brock and B. Eich, "JavaScript: The first 20 years," *Proc. ACM Program. Lang.*, vol. 4, no. HOPL, 2020, doi:

10.1145/3386327.

[15] Z. Gao *et al.*, "Construction practice of student evaluation system based on JFinal + webix integrated framework and Baidu AI platform," *MATEC Web Conf.*, vol. 336, p. 05016, 2021, doi: 10.1051/matecconf/202133605016.

[16] A. Cabana, C. Charrier, and A. Louis, "Mono and multi-modal biometric systems assessment by a common black box testing framework," *Futur. Gener. Comput. Syst.*, vol. 101, pp. 293–303, 2019, doi: 10.1016/j.future.2019.04.053.

[17] E. Viglianisi, M. Dallago, and M. Ceccato, "RESTTESTGEN: Automated Black-Box Testing of RESTful APIs," *Proc. - 2020 IEEE 13th Int. Conf. Softw. Testing, Verif. Validation, ICST 2020*, pp. 142–152, 2020, doi: 10.1109/ICST46399.2020.00024.