The l3pdffield-checkbox module
Commands to create checkbox fields
LaTeX PDF management testphase bundle

The LaTeX Project*

Version 0.00a, released 0000-00-00

1 l3pdffield-checkbox Documentation

The implementation of form fields in hyperref has some bugs, see for example https://github.com/latex3/hyperref/issues/94. This package is a first step towards the goal to review and improve the code of form fields.

Like the pdfmanagement-testphase package itself it is a temporary package: the definite home of the code is not yet decided.

It handles for now *only* checkboxes, other form fields like radio buttons or text fields will follow later. As no fonts are used, it doesn't relies on the hyperref code to initialize the form, but it can be used with hyperref.

The code requires the new PDF management.

Click me: 

Please keep in mind

- Not every PDF viewer supports checkboxes.
- The handling can depend on settings in the PDF viewer. In adobe reader for example I had to disable an option to avoid that it tries to create an appearance it self
- Standards like pdf/A disable form fields too (as you typically can't change the PDF.

If hyperref is loaded before the package will suppress the deprecated `/NeedAppearances` setting. If hyperref is loaded later you should do it in the `\Form` options.

So a typical use together with hyperref could look like this

```
\RequirePackage{pdfmanagement-testphase}
\DeclareDocumentMetadata{uncompress}
\documentclass{article}
\usepackage{hyperref}
```

*E-mail: latex-team@latex-project.org

```
\usepackage{l3pdffield-checkbox-testphase}
\begin{document}
\Form
```

1.1 Commands

`\pdffield_checkbox:n` `\pdffield_checkbox:n{`⟨*key val list*⟩`}`

This creates a checkbox to check and uncheck. The list of allowed keys is described below. Typically the ⟨*key val list*⟩ should at least set the name. Checkboxes with the same name belong to the same field and are checked and unchecked together. The default appearance is a quadratic frame with a `\texttimes` in it for the checked case.

`\pdffield_setup:nn` `\pdffield_setup:nn{`⟨*field type*⟩`}{`⟨*key val list*⟩`}`

This allows to setup up values for following fields. `{`⟨*field type*⟩`}` should be a field, currently the only allowed value is `checkbox`.

`\pdffield_store_appearance:nn` `\pdffield_store_appearance:nn{`⟨*name/state*⟩`}{`⟨*content*⟩`}`

This is a small wrapper around `\pdfxform_new:nn` and store appearances for the fields. ⟨*state*⟩ should be for checkboxes either `Yes` or `Off` (and typically you should define both). ⟨*name*⟩ is the name that is used in the `appearance` key, `checkbox/default` is predefined and so can't be used. ⟨*content*⟩ is arbitrary content. The dimensions should fit to the planed size of the checkbox.

1.2 Keys

The new checkbox commands accepts following keys:

`name` (*checkbox key*) This sets the (internal) name of the field. It shouldn't contain a period, be not empty and sensibly consist of simple chars. Checkboxes instances with the same name belong to the same field and are checked and unchecked together.

`altname` (*checkbox key*)
`mappingname` (*checkbox key*) This sets an alternative name for user interaction. This name can only be set at the first checkbox instance, when the field is initialized. This sets an alternative name for export. This name can only be set at the first checkbox instance, when the field is initialized.

`width` (*checkbox key*)
`height` (*checkbox key*)
`depth` (*checkbox key*) These keys allow to set the dimensions of checkbox instance. The value should be a command that expands to a dimension expression. By default `width` and `height` use `\normalbaselineskip`, the `depth` is zero.

`appearance` (*checkbox key*) This key sets the normal appearance. It takes as value a ⟨*name*⟩ and expects that the two appearances ⟨*name*⟩`/Yes` and ⟨*name*⟩`/Off` has been created with the command described below. The initial value is `checkbox/default` and shows a `\texttimes`.

`rollover-appearance` (*checkbox key*) This key sets the rollover appearance (when the mouse hovers over the checkbox). It takes as value a ⟨*name*⟩ and expects that the two appearances ⟨*name*⟩`/Yes` and ⟨*name*⟩`/Off` has been created with the command described below. Initially this is not set. An empty value removes the entry.

`down-appearance` (*checkbox key*) This key sets the down appearance (when the mouse clicks). It takes as value a ⟨*name*⟩ and expects that the two appearances ⟨*name*⟩`/Yes` and ⟨*name*⟩`/Off` has been created with the command described below. Initially this is not set. An empty value removes the entry.

`checked` (*checkbox key*)

This is a boolean key which allows to set if the checkbox should be initially checked or not. It sets the `/V` and `/DV` key of the field and the `/AS` key of the annotation instance. It is possible to use different values for different instances, if one wants to confuse the user.

`setfieldflags` (*checkbox key*)
`unsetfieldflags` (*checkbox key*)

These keys allow to set or unset the field flags. They expect a comma lists of flag names. Allowed names `ReadOnly`, `Required`, `NoExport`, `Multiline`, `Password`, `NoToggleToOff`, `Radio`, `Pushbotton`, `Combo`, `Edit`, `Sort`, `FileSelect`, `MultiSelect`, `DoNotSpellCheck`, `DoNotScroll`, `Comb`, `RadiosInUnison`, `RichText`, `CommitOnSelChange`.

From these `Radio`, `Pushbotton` are set automatically automatically by the code as this is required for a checkbox. Not every one from the rest makes sense for checkboxes but the same key will be used for other fields too. Check the PDF reference to decide which one to set or unset.

`keystroke` (*checkbox key*)
`format` (*checkbox key*)
`validate` (*checkbox key*)
`calculate` (*checkbox key*)

These keys add the `/K`, /F, `/V`, `/C` key to the `/AA` dictionary of the field object. Their value should be javascript code. The `/AA` dictionary is suppressed if a pdf/A standard is set. These keys are probably not of much used with a checkbox.

`onfocus` (*checkbox key*)
`onblur` (*checkbox key*)
`onmousedown` (*checkbox key*)
`onmouseup` (*checkbox key*)
`onenter` (*checkbox key*)
`onexit` (*checkbox key*)

These keys adds the `/F`, /Bl, `/D`, `/U`, `E` and `X` key to the `/AA` dictionary of the widget annotation (the checkbox instance) object. Their value should be javascript code. The `/AA` dictionary is suppressed if a pdf/A standard is set.

For example

```
onenter={app.alert('Hello');}
```

1.3 Using with hyperref

The `\CheckBox` command from hyperref also prints a label, something that the command here don't do. A redefinition like the following should allow to `\CheckBox` to use the commands of this module. Be aware that the behaviour will not be identical! Not every setting and key from hyperref has been copied.

```
\ExplSyntaxOn\makeatletter
\def\@CheckBox[#1]#2{\LayoutCheckField{#2}{\pdffield_checkbox:n {name=#2,#1}}}
\ExplSyntaxOff\makeatother
```

1.4 Some background

Form fields consist of a field object and number of instances of the field: A checkbox can appear on more than one page or location and if one instance is checked all other instances follows and are checked too.

All instances are in this case widget annotations and are referenced in the Kid array of the field object[1]. This means that the code has to collect all the children and write out the field object at the end of the document.

If a field has only one children the content of the field dictionary and the widget annotation dictionary can be merged—some examples in the PDF reference show such merged dictionaries—but the code here keeps them separate, at the end this is clearer.

[1] Fields can actually build a tree: between the root field and widget annotations there can be more fields. The last one before the widget is the *terminal* field, but unless a sensible use case comes up, I will assume that the widget annotations are direct children of the root and that the root field is the terminal field.

All the root field objects must be referenced in the AcroForm dictionary in the Fields entry. This can be done with

```
\pdfmanagement_add:nnx{Catalog/AcroForm}{Fields}{<obj ref>}
```

A checkbox has two different looks: checked and unchecked. The current hyperref implementation uses symbolic names for the two states and and adds some values with the /MK key and lets the PDF viewer create a look from them. But this doesn't work reliably. Also newer PDF versions deprecate the /NeedAppearances setting and require that such a look, an "appearance", is given as form XObjects: such form XObjects are like small pictures stored in the PDF and can be referenced in various part of the PDF. They can be created with the commands of the l3pdfxform package.

The checkbox instances—the widget annotations—cover a rectangular area on the page, the XObjects are squeezed into this rectangle. So for the best result both should have the same ratio of width and height. XObjects used as appearances can not be rotated, if needed one has to create a new appearance.

1.4.1 The field dictionary

The field dictionary shall or can have the following entries

FT required for terminal fields (here for the root field), the value is always `Btn`, so this entry is set by the code.

Parents currently irrelevant as we don't have a field hierarchy.

Kids an array. Contains references to the children, in our case to the widget annotations. The array is build by the code.

T required, the name, a (unique) text string without a period. This field is a mandatory argument which must be given by the user. The value should be passed through a suitable string conversion and checked if it contains a period (which is not allowed).

TU,TM optional, alternative names for user messages (TU) and export (TM). As these fields are optional they should be set by some key-val option.

Ff A bitset, two flags must be unset for a checkbox (Radio and Pushbotton), for the rest we need a keyval interface.

V describes the initial value, for checkboxes is should be either `/Yes` or `/Off`. The initial value will be `/Yes`, keys are need to set both the local and the default value.

DV optional, the default value after a reset. Should like `V` be either `/Yes` or `/Off`.

AA An action dictionary. For this we need a special command to setup such dictionaries, so that they then can be used in various places.

1.4.2 The widget annotation dictionary

Type Value: `/Annot`, set automatically

Subtype Value: `/Widget` this is added automatically. We use an internal dictionary which is locally copied over the one from l3pdfannot. It can be filled with keyval options.

Parent The reference to the field, automatically added.

Rect the size, calculated from the box size

Contents a text string, not really needed but an optional key should allow to set it,

AP the appearance dictionary. It should look like this `/AP <</N <</Yes 17 0 R/Off 15 0 R>>>>`. I need to test if it makes sense here to have a `/R` and `/D` entry too. The objects refer to suitable xforms.

AS should be either `/Yes` or `/Off`, and sensibly by default be the same as the V entry in the field dictionary. If they differ the AS entry wins.

A Action, this must be checked.

AA additional actions. This must be checked too.

Border, C, OC, AF, BM, Lang, P, NM, M, F, BS, H : These are not specifically needed for checkbox. An interface to add something to the used annot dictionary is needed (or already there), but probably no special key-val support for now.

MK this is what hyperref uses to set the appearance, but I explicitly leave it out and use `AP`.

Q (alignment), used by hyperref but not relevant as we don't have variable text here.

2 l3pdffield-checkbox Implementation

```
⟨*package⟩
⟨@@=pdffield⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesExplPackage{l3pdffield-checkbox}{0000-00-00}{v0.00a}{form field checkbox}%
```

2.1 hyperref specific command

hyperref sets NeedAppearances by default. As this is deprecated we disable this.

```
\csname HyField@NeedAppearancesfalse\endcsname % suppress NeedAppearances
% values from hyperref:
%\def\DefaultOptionsofCheckBox{print}
%\def\DefaultHeightofCheckBox{\normalbaselineskip}
%\def\DefaultWidthofCheckBox{\normalbaselineskip}
```

2.2 local variables

```
\str_new:N \l__pdffield_field_name_str
\str_new:N \l__pdffield_tmpa_str
\str_new:N \l__pdffield_name_tmpa_str
\tl_new:N \l__pdffield_keys_tmpa_tl
```

2.3 messages

```
\msg_new:nnn {pdffield}{no-period}
  {
    The~field~name~`#1`~contains~a~period. \\
    This~is~not~allowed. `
```

```
  }
\msg_new:nnn {pdffield}{empty-name}
  {
    The~field~name~is~empty. \\
    This~is~not~allowed. ‘
  }
\msg_new:nnn {pdffield}{appearance-missing}
  {
    The~appearance~‘#1‘~is~missing~for~the~#2~appearance.
  }
\msg_new:nnn {pdffield}{field-keys-ignored}
  {
    The~field~‘#1‘~is~already~initialized\\
    The~field~keys~‘#2‘~are~ignored.
  }
```

2.4 bitsets

A bitset for the field flag Ff and an internal copy of the annot bitset.

```
\bitset_new:Nn \l__pdffield_Ff_bitset
 {
    ReadOnly          = 0,
    Required          = 1,
    NoExport          = 2,
    Multiline         = 12,%Tx
    Password          = 13,
    NoToggleToOff     = 14,%Btn, radio button
    Radio             = 15,%Btn: Radio:    15=1, 16=0
    Pushbutton        = 16,%Btn: Checkbox: 15=0, 16=0
                           %Btn: Pushbutton: 16=1
    Combo             = 17,%Ch: Combo=1 List=0
    Edit              = 18,%Ch, Combo=1 -> + edit field
    Sort              = 19,%Ch, not relevant for view...
    FileSelect        = 20,%Tx
    MultiSelect       = 21,%Ch
    DoNotSpellCheck   = 22,%Tx, Ch (if Combo + Edit set)
    DoNotScroll       = 23,%Tx
    Comb              = 24,%Tx, requires MaxLen in dict
    RadiosInUnison    = 25,%Btn Radio
    RichText          = 25,%Tx
    CommitOnSelChange = 26
  }

\bitset_new:Nn \l__pdffield_F_bitset
  {
    Invisible      = 1,
    Hidden         = 2,
    Print          = 3,
    NoZoom         = 4,
    NoRotate       = 5,
    NoView         = 6,
    ReadOnly       = 7,
    Locked         = 8,
    ToggleNoView   = 9,
```

```
    LockedContents = 10
  }
```

2.5 The field dictionary

The field dictionary is the main object. It references the actual widget annotations as kids. It is created at the first checkbox with a specific name. To be able to set values from the outside it will use a dictionary which can be filled by key-val.

```
\pdfdict_new:n   {l__pdffield/checkbox/field}
\pdfdict_put:nnn {l__pdffield/checkbox/field}{FT}{/Btn}
```

We need to check if the name contains a dot. But we will do this in an external command to avoid to have it twice. Here we assume that the name is already converted and safe.

We also assume that values that can be changed by the user are set outside in the dictionary. If the field object already exists nothing is done.

__pdffield_checkbox_field_add:n

__pdffield_checkbox_field_add:n{⟨*name*⟩}

⟨*name*⟩ should be a PDF text string without period. It identifies the field.

```
\cs_new_protected:Npn \__pdffield_checkbox_field_new:n #1
  {
      \group_begin:
      \pdf_object_new:nn {__pdffield_checkbox/field/#1}      {dict}
      \pdf_object_new:nn {__pdffield_checkbox/field/#1/Kids} {array}
      \seq_new:c {g__pdffield_checkbox/field/#1/Kids_seq}
      \hook_gput_code:nnn {shipout/lastpage}{pdffield} %xetex needs this ...
        {
          \pdf_object_write:nx {__pdffield_checkbox/field/#1/Kids}
            {
              \seq_use:cn{g__pdffield_checkbox/field/#1/Kids_seq}{~}
            }
        }
      \pdfdict_put:nnn {l__pdffield/checkbox/field}{T}{(#1)}
     % V,DV are names describing the appearance. With checkboxes
     % the values /Yes and /Off are used.
     % this values are taken from the outside
      \pdfdict_put:nnx {l__pdffield/checkbox/field}
         {Kids}
         {
           \pdf_object_ref:n {__pdffield_checkbox/field/#1/Kids}
         }
      \bitset_set_false:Nn \l__pdffield_Ff_bitset  {Radio}
      \bitset_set_false:Nn \l__pdffield_Ff_bitset  {Pushbutton}
      \pdfdict_put:nnx {l__pdffield/checkbox/field}
        {Ff}
        {\bitset_to_arabic:N \l__pdffield_Ff_bitset }
      \pdfdict_if_empty:nF{l__pdffield/checkbox/field/AA}
        {
          \pdfmeta_standard_verify:nT
            {annot_widget_no_AA}
            {
              \pdfdict_put:nnx
                {l__pdffield/checkbox/field}
                {AA}
                {<<\pdfdict_use:n {l__pdffield/checkbox/field/AA}>>}
```

```
            }
        }
      \pdf_object_write:nx {__pdffield_checkbox/field/#1} { \pdfdict_use:n {l__pdffield/check
      \pdfmanagement_add:nnx
        { Catalog / AcroForm }
        { Fields }
        {\pdf_object_ref:n {__pdffield_checkbox/field/#1} }
      \group_end:
  }

```

(*End definition for* `\__pdffield_checkbox_field_add:n`.)

2.6 The annot dictionary

We assume that the annotation should really occupy space on the page and leave vertical mode. We also assume that keys like AP, AS are added before through keys to the dictionary. We use a local dictionary which is copied into `l__pdfannot/widget` in the code.

```
\pdfdict_new:n   {l__pdffield/checkbox/annot}
\pdfdict_put:nnn {l__pdffield/checkbox/annot}{Subtype}{/Widget}
```

`\__pdffield_checkbox_annot_add:nnnn`

```
\cs_new_protected:Npn \__pdffield_checkbox_annot_add:nnnn #1 #2 #3 #4 %name, wd, ht, dp,
  {
    \group_begin:
```

Copy the internal dictionary to the pdfannot dictionary. This perhaps need a function in l3pdfannot, as it actually uses an internal name of another module.

```
    \pdfdict_put:nnx {l__pdffield/checkbox/annot}{AP}{<<\pdfdict_use:n{l__pdffield/checkbox/a
    \pdfmeta_standard_verify:nF
      {annot_flags}
      {
        \bitset_set_true:Nn  \l__pdffield_F_bitset {Print}
        \bitset_set_false:Nn \l__pdffield_F_bitset {Hidden}
        \bitset_set_false:Nn \l__pdffield_F_bitset {Invisible}
        \bitset_set_false:Nn \l__pdffield_F_bitset {NoView}
      }
    \pdfdict_if_empty:nF{l__pdffield/checkbox/annot/AA}
      {
        \pdfmeta_standard_verify:nT
         {annot_widget_no_AA}
         {
            \pdfdict_put:nnx
              {l__pdffield/checkbox/annot}
              {AA}
              {<<\pdfdict_use:n {l__pdffield/checkbox/annot/AA}>>}
         }
      }
    \pdfdict_put:nnx {l__pdffield/checkbox/annot}{F}{ \bitset_to_arabic:N \l__pdffield_F_bits
    \pdfdict_set_eq:nn {l__pdfannot/widget}{l__pdffield/checkbox/annot}
    \pdfannot_dict_put:nnx {widget}{Parent}{\pdf_object_ref:n{__pdffield_checkbox/field/#1}}
    \mode_leave_vertical:
    \hbox_to_wd:nn
```

```
      { #2  }
      {
        \rule [-#4]{0pt}{\dim_eval:n{#3+#4} }
        \pdfannot_widget_box:nnn
           { #2 }
           { #3 }
           { #4 }
         \hfill
      }
    \seq_gput_right:cx {g__pdffield_checkbox/field/#1/Kids_seq}{ \pdfannot_box_ref_last:}
    \group_end:
  }

```

(*End definition for* `\__pdffield_checkbox_annot_add:nnnn`.)

2.7 Appearances

We don't try to force a size or content here. The default appearances are a cross (`\texttimes`), Every appearance should have two versions and follow the naming checkbox/⟨*name*⟩/Yes and checkbox/⟨*name*⟩/Off. TODO check if one delay the creation to a sensible place (and don't forget that the appearance key sets an initial value)

`\pdffield_store_appearance:nn`
`\__pdffield_store_default_appearances:`

```
\cs_new_protected:Npn \pdffield_store_appearance:nn #1 #2
  {
     \pdfxform_new:nnn {__pdffield_#1}{}{#2}
  }

\cs_new_protected:Nn \__pdffield_store_default_appearances:
  {
     \pdffield_store_appearance:nn {checkbox/default/Yes}
       {
         \normalsize
         \fboxsep 0pt
         \framebox
           [ \dim_eval:n { \box_ht:N\strutbox+\box_dp:N\strutbox } ]
           { \texttimes \strut }
       }
     \pdffield_store_appearance:nn {checkbox/default/Off}
       {
         \normalsize
         \fboxsep 0pt
         \framebox
           [ \dim_eval:n { \box_ht:N\strutbox+\box_dp:N\strutbox } ]
           { \phantom{\texttimes} \strut }
       }
  }

\__pdffield_store_default_appearances:
```

(*End definition for* `\pdffield_store_appearance:nn` *and* `\__pdffield_store_default_appearances:`. *This function is documented on page 2.*)

We define a dictionary for the AP content, so that we can add R and D too

```
\pdfdict_new:n   {l__pdffield/checkbox/annot/AP}
```

2.8 Assembling the checkbox

__pdffield_checkbox_add:n

```
\cs_new_protected:Npn \__pdffield_checkbox_add:n #1
  {
    \group_begin:
    \keys_set_filter:nnnN {pdffield / checkbox }{field}{#1}\l__pdffield_keys_tmpa_tl
    \str_if_empty:NT \l__pdffield_field_name_str
      {
        \msg_error:nn {pdffield}{empty-name}
      }
    \exp_args:Nx
      \pdf_object_if_exist:nTF {__pdffield_checkbox/field/\l__pdffield_field_name_str}
       {
          \tl_if_empty:NF \l__pdffield_keys_tmpa_tl
           {
             \msg_warning:nnxx
               {pdffield}
               {field-keys-ignored}
               {\l__pdffield_field_name_str}
               {\l__pdffield_keys_tmpa_tl}
           }
       }
       {
         \keys_set:nV { pdffield/checkbox } \l__pdffield_keys_tmpa_tl
         \exp_args:No
         \__pdffield_checkbox_field_new:n {\l__pdffield_field_name_str}
       }
    \exp_args:No
      \__pdffield_checkbox_annot_add:nnnn
        {\l__pdffield_field_name_str}
        {\l__pdffield_annot_wd_tl }
        {\l__pdffield_annot_ht_tl }
        {\l__pdffield_annot_dp_tl }
    \group_end:
  }
```

(*End definition for* `\__pdffield_checkbox_add:n`.)

2.9 Keys

The size of the checkbox

```
\tl_new:N \l__pdffield_annot_ht_tl
\tl_new:N \l__pdffield_annot_wd_tl
\tl_new:N \l__pdffield_annot_dp_tl

\keys_define:nn { pdffield / checkbox }
  {
    ,width  .tl_set:N = \l__pdffield_annot_wd_tl
    ,height .tl_set:N = \l__pdffield_annot_ht_tl
    ,depth  .tl_set:N = \l__pdffield_annot_dp_tl
    ,width  .initial:n = \normalbaselineskip
    ,height .initial:n = \normalbaselineskip
```

```
    ,depth  .initial:n = 0pt
  }
```

The names. The main name should not be empty, it is added to the dictionary when the field is created. A new name means a new field. The other names can only be set when the field is created, so we put them in the field group.

```
\keys_define:nn { pdffield / checkbox }
  {
    ,name .code:n =
      {
        \pdf_string_from_unicode:nnN {utf8/string-raw}{#1}\l__pdffield_field_name_str
        \str_if_in:NnT \l__pdffield_field_name_str {.}
          {
            \msg_error:nnx {pdffield}{no-period}{\l__pdffield_field_name_str}
          }
        \str_if_empty:NT\l__pdffield_field_name_str
          {
            \msg_error:nn {pdffield}{empty-name}
          }
      }
    ,name .value_required:n = true
    ,name .initial:n = checkbox
    ,altname .code:n =
      {
        \pdf_string_from_unicode:nnN {utf8/string}{#1}\l__pdffield_name_tmpa_str
        \pdfdict_put:nnx { l__pdffield/checkbox/field }{TU}{\l__pdffield_name_tmpa_str}
      }
    ,altname .groups:n = {field}
    ,mappingname .code:n =
      {
        \pdf_string_from_unicode:nnN {utf8/string}{#1}\l__pdffield_name_tmpa_str
        \pdfdict_put:nnx { l__pdffield/checkbox/field }{TM}{\l__pdffield_name_tmpa_str}
      }
    ,mappingname .groups:n = {field}
  }
```

A key to decide if the Box is initially checked or not

```
\keys_define:nn { pdffield / checkbox }
 {
   ,checked .choice:
   ,checked / false .code:n =
     {
       \pdfdict_put:nnn {l__pdffield/checkbox/field}{V}{/Off}
       \pdfdict_put:nnn {l__pdffield/checkbox/annot}{AS}{/Off}
       \pdfdict_put:nnn {l__pdffield/checkbox/field}{DV}{/Off}
     }
   ,checked / true .code:n =
     {
       \pdfdict_put:nnn {l__pdffield/checkbox/field}{V}{/Yes}
       \pdfdict_put:nnn {l__pdffield/checkbox/annot}{AS}{/Yes}
       \pdfdict_put:nnn {l__pdffield/checkbox/field}{DV}{/Yes}
     }
   ,checked .default:n = {true}
   ,checked .initial:n = {false}
 }
```

Flags. We don't add lots of individual keys but mapped the key names directly

```
\keys_define:nn { pdffield / checkbox }
  {
    ,setfieldflags .code:n =
      {
          \clist_map_inline:nn {#1}
           {
             \bitset_set_true:Nn \l__pdffield_Ff_bitset {##1}
           }
      }
    ,setfieldflags .groups:n = {field}
    ,unsetfieldflags .code:n =
      {
          \clist_map_inline:nn {#1}
           {
             \bitset_set_false:Nn \l__pdffield_Ff_bitset {##1}
           }
      }
    ,unsetfieldflags .groups:n = {field}
    ,setannotflags .code:n =
      {
          \clist_map_inline:nn {#1}
           {
             \bitset_set_true:Nn \l__pdffield_F_bitset {##1}
           }
      }
    ,unsetannotflags .code:n =
      {
          \clist_map_inline:nn {#1}
           {
             \bitset_set_false:Nn \l__pdffield_F_bitset {##1}
           }
      }
  }

\keys_define:nn { pdffield / checkbox }
  {
    appearance .code:n = %value is a name of an appearance
      {
        \pdfxform_if_exist:nTF {  __pdffield_#1/Yes }
          {
            \pdfdict_put:nnn {l__pdffield/checkbox/annot/AP}
              {N}
              {
                 <<
                    /Yes ~ \pdfxform_ref:n { __pdffield_#1/Yes}
                    /Off ~ \pdfxform_ref:n { __pdffield_#1/Off}
                 >>
              }
          }
          {
            \msg_error:nnnn{pdffield}{appearance-missing}{#1}{normal}
          }
      },
```

```
    appearance .initial:n = checkbox/default,
  }

\keys_define:nn { pdffield / checkbox }
  {
    rollover-appearance .code:n = %value is a name of an appearance
      {
       \tl_if_empty:nTF {#1}
         {
           \pdfdict_remove:nn {l__pdffield/checkbox/annot/AP} {R}
         }
         {
           \pdfxform_if_exist:nTF {  __pdffield_#1/Yes }
             {
               \pdfdict_put:nnn {l__pdffield/checkbox/annot/AP}
                 {R}
                 {
                    <<
                       /Yes ~ \pdfxform_ref:n { __pdffield_#1/Yes}
                       /Off ~ \pdfxform_ref:n { __pdffield_#1/Off}
                    >>
                 }
             }
             {
               \msg_warning:nnnn{pdffield}{appearance-missing}{#1}{rollover}
             }

         }
      },
  }

\keys_define:nn { pdffield / checkbox }
  {
    down-appearance .code:n = %value is a name of an appearance
      {
       \tl_if_empty:nTF {#1}
         {
           \pdfdict_remove:nn {l__pdffield/checkbox/annot/AP} {D}
         }
         {
           \pdfxform_if_exist:nTF {  __pdffield_#1/Yes }
            {
              \pdfdict_put:nnn {l__pdffield/checkbox/annot/AP}
                {D}
                {
                   <<
                      /Yes ~ \pdfxform_ref:n { __pdffield_#1/Yes}
                      /Off ~ \pdfxform_ref:n { __pdffield_#1/Off}
                   >>
                }
            }
            {
              \msg_warning:nnnn{pdffield}{appearance-missing}{#1}{down}
            }
```

```
          }
      },
 }
```

Keys for the AA dictionary. They all trigger javascript option.

__pdffield_define_AAaction_key:nnn

```
\pdfdict_new:n {l__pdffield/checkbox/annot/AA}
\pdfdict_new:n {l__pdffield/checkbox/field/AA}

\cs_new_protected:Npn \__pdffield_define_AAaction_key:nnn #1 #2 #3 %#1 key, #2 pdf, #3 dict
  {
    \keys_define:nn { pdffield / checkbox }
      {
         #1 .code:n =
           {
             \pdf_string_from_unicode:nnN {utf8/string}{##1}\l__pdffield_tmpa_str
             \str_if_empty:NTF \l__pdffield_tmpa_str
               {
                 \pdfdict_remove:nn {l__pdffield/checkbox/#3/AA}{#2}
               }
               {
                 \pdfdict_put:nnx {l__pdffield/checkbox/#3/AA}
                  {#2}
                  {<</S/JavaScript/JS\l__pdffield_tmpa_str>>}
               }
           },
        #1 .groups:n = {#3}
      }
  }
\__pdffield_define_AAaction_key:nnn {keystroke}{K}{field}
\__pdffield_define_AAaction_key:nnn {format}   {F}{field}
\__pdffield_define_AAaction_key:nnn {validate} {V}{field}
\__pdffield_define_AAaction_key:nnn {calculate}{C}{field}
\__pdffield_define_AAaction_key:nnn {onfocus}  {Fo}{annot}
\__pdffield_define_AAaction_key:nnn {onblur}   {Bl}{annot}
\__pdffield_define_AAaction_key:nnn {onmousedown}{D}{annot}
\__pdffield_define_AAaction_key:nnn {onmouseup}{U}{annot}
\__pdffield_define_AAaction_key:nnn {onenter}  {E}{annot}
\__pdffield_define_AAaction_key:nnn {onexit}   {X}{annot}
```

(*End definition for* `\__pdffield_define_AAaction_key:nnn`.)

2.10 user commands

\pdffield_checkbox:n
\pdffield_setup:nn

```
\cs_set_eq:NN \pdffield_checkbox:n \__pdffield_checkbox_add:n

\cs_new_protected:Npn \pdffield_setup:nn #1 #2
 {
   \keys_set:n {pdffield / #1 } {#2}
 }
⟨/package⟩
```

(*End definition for* `\pdffield_checkbox:n` *and* `\pdffield_setup:nn`. *These functions are documented on page 2.*)