Contenteditable and Rich Text Editing

a short story by Eric Wood

eric@ericwood.org | @eric_b_wood

So…what is it?

- What You See Is What You Get (WYSIWYG)
- Think Microsoft Word, but in the browser

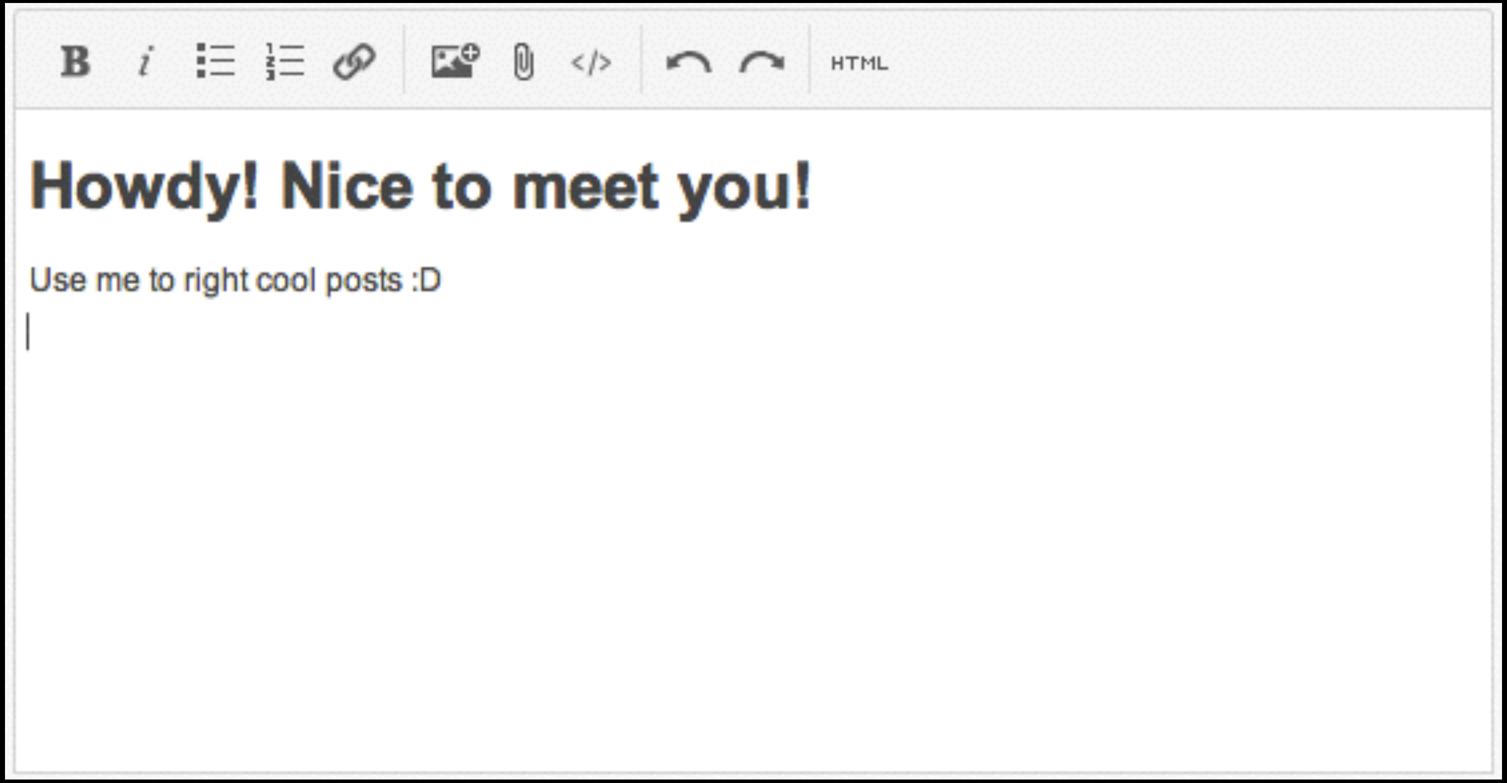
```html
<div contenteditable="true">
  <h1>Edit me!</h1>
</div>
```

Brief History

- First appearance in IE 5.5
- Reverse-engineered and supported by other browsers
- Barely any spec, mostly based off reverse-engineering of original IE stuffs
- For more info: http://blog.whatwg.org/the-road-to-html-5-contenteditable

The Twilight Zone

Introducing The Spiceworks WYSIWYG

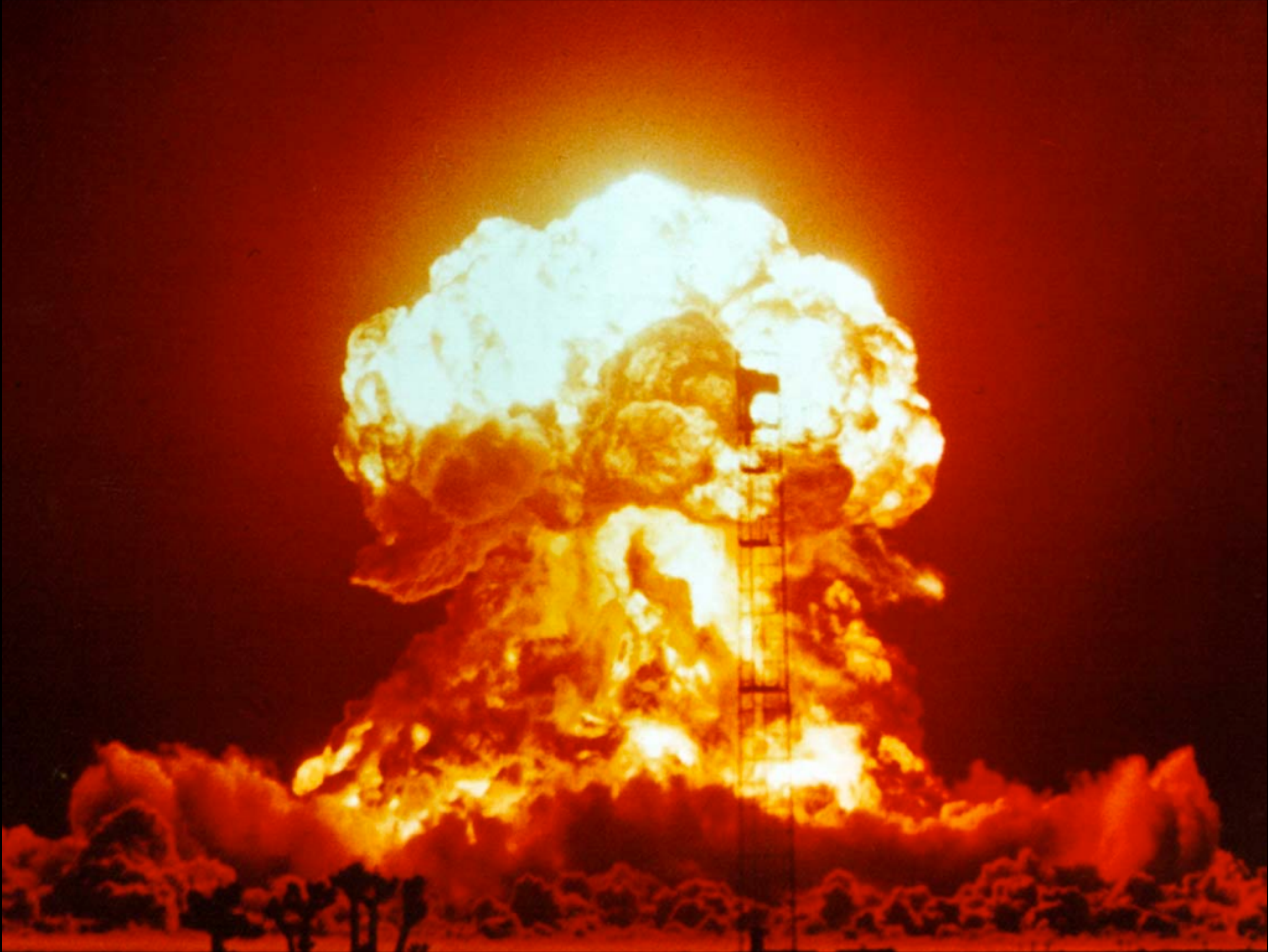


Howdy! Nice to meet you!

Use me to right cool posts :D

We needed a better WYSIWYG editor

- CKEditor was holding us back!
- We picked wysihtml5 as a starting point
  - It’s fairly small, the source is readable, etc.
- All is well in the universe…?



The system clipboard

- The clipboard can have all kinds of stuff in it (different formats, even)
- Most common:
  - text/plain
  - text/html (rich text)
- Word (and like everything else) use rich text :\
- Every application has its own way of formatting rich text

Rich Text Examples

- A Word document that only says “wtf”
  - http://pastebin.com/DX7wwU85

So…what now?

Well…onpaste looks promising, let’s check it out

The onpaste Event

http://jsfiddle.net/eric_wood/ZW5sF/

Caveats

- That approach is great!
- IE doesn’t care, though…
- Firefox is the only browser that does text/plain right
- window.clipboardData.getData(“Text”);
  - Prompts the user for permission **EVERY TIME**
- Back to the drawing board!

A hack!

Well, textareas do what we want…

Wait! I have an idea!

It’s crazy, though…

http://jsfiddle.net/eric_wood/LkGv5/

Caveats

- Browser support is iffy (at best)
- Timing is SUPER important! (it’s flaky)
- Only works if you use cmd+v or ctrl+v (boooo)

I’ve lost my mind.
Let’s try something really dumb.

- What if we just nuked everything?
- onpaste =>
    - Save cursor state, clear editor
    - [pasted text appears!]
    - Use editor’s parser to clean it
    - Save contents temporarily
    - Restore previous contents and selection
    - Call insertHTML and add the pasted text!

Caveats

- The one thing browsers can’t agree on is the API for working with the caret (rangy isn’t helpful)
- Works in Chrome!
- Huge UX failure in every browser
    - Weird stuff happens in each one

Back to that first solution…

- Fine, it’s not 100% effective, but it covers a huge number of users (internal ones in particular)
- We can get the HTML from the clipboard, but what about converting it to plaintext?

HTML => plaintext
Let the browser do the work!

```
var wrapper = $('<div style="display: none;" />');
wrapper.html(html);
wrapper.appendTo('body');
var contents = wrapper.find('*');
if(!contents.length) {
  return wrapper.text();
}
contents.parent().find('br').replaceWith('\n');
```

```
var text = [];
contents.each(function() {
  var $el = $(this);
  var display = $el.css('display');

  if(display === 'block') {
    text.push($el.text());
  }
});
```

```
wrapper.remove();

text = _.filter(text, function(t) {
  return /\S/.test(t);
});

return text.join('\n');
```

```
if(event.clipboardData) {
  event.preventDefault();

  var text;
  var html = event.clipboardData.getData('text/html');

  if(html === '') {
    // sometimes there's plaintext instead (which is good news!)
    text = event.clipboardData.getData('text/plain');
  } else {
    text = wysihtml5.dom.htmlToText(html);
  }

  // wrap everything in <p> tags...
  var result = _.map(text.split('\n'), function(line) {
    return '<p>' + line + '</p>';
  });

  result = result.join('\n');

  // throw it into the composer, bam!
  _this.commands.exec('insertHTML', result);
}
```

Questions?