Web Programming

Week 5

"Unfortunately, JS has a misfeature called Automated Semicolon Insertion. It can fail in bad ways, so write like a professional."

Douglas Crockford, "How JavaScript works."

Storybook (initial)

Drehbuch, Intro, Functions
Scientific foundations
Algebraic Data Types
Applied Science, Snake
Scripting, PWA, Plotter, Excel
Objects
Classes
JS Types, JsDoc
Async Programming
Modules
Data Flow, Excel improved
Iterator Protocol, Sequences
Moves, User Interfaces
Crazy JS

Pair, Product Type

```
const pair = x => y => f => f(x)(y);
const fst  = p => p(T);
const snd  = p => p(F);
```

the basic product type

Either, Co-Product, Sum

```
const Left   = x => f => g => f(x);        // ctor 1
const Right  = x => f => g => g(x);        // ctor 2
const either = e => f => g => e(f)(g);     // accessor
```

the basic sum type

Special Case: Maybe

```
const Nothing  = Left ();
const Just     = Right  ;
const maybe    = either ;

maybe (expressionThatMightGoWrong)
      (handleBad)
      (handleGood);
```

go around null / undefined

New Concepts

```
pair + pair == pair        // monoid
map (f) (pair) == pair     // functor
```

Immutability,
Laziness

We did not

use a build system

depend on libraries, frameworks

use a module system

depend on special IDE features

Today: Scripting

Progressive Web App for Testing

General-purpose function plotter

Excel in the browser

Quiz

What is Scripting?

Evaluating text

Sources: file, URL, DB, User Input, ...

Text can be modified, amended, ect. !

Why Scripting?

Command Line, Automation, Build System, Templating, Code Distribution, Formulae, Business Rules, Smart Configuration, Product Lines, DSL, Self-Modifying Code, ...

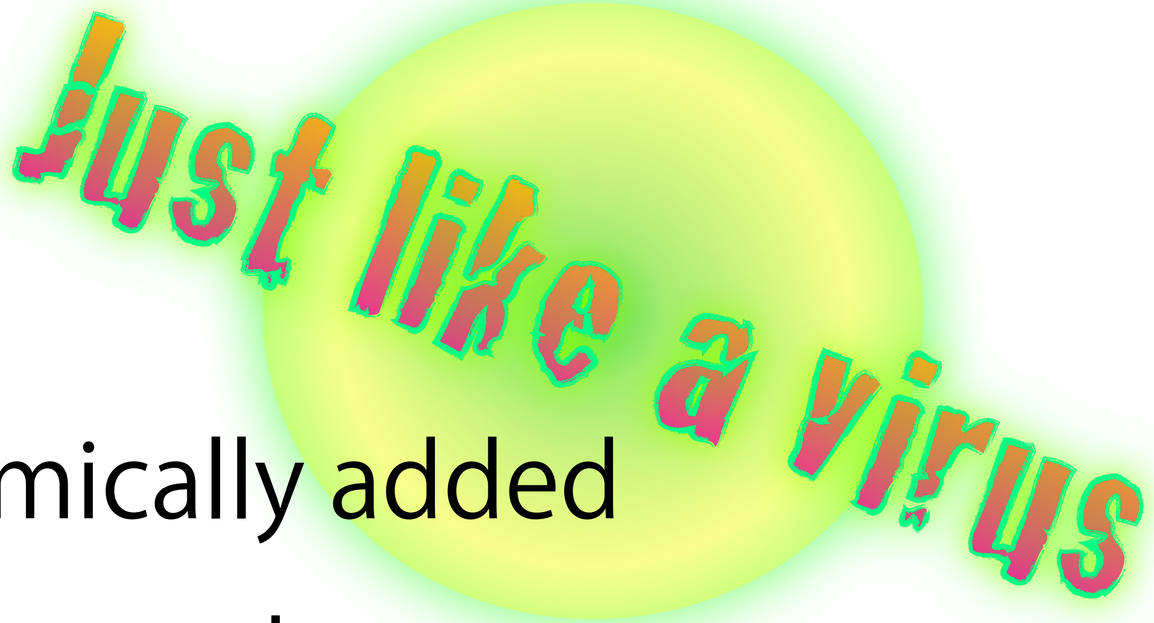
Scripting Characteristics

Interpreted, not compiled (in principle)

Lenient type system

"Best Effort" approach

Progressive Web App



<script> tag static

<script> tag dynamically added

Code, that produces code,
gets interpreted,
and thereby produces code, that ....

Progressive Web App

Example:
Loading test suite dynamically

```
document.write('<script src= …');
```

Function Plotter: eval

eval() works as if the code was copied verbatim in the place of the eval, i.e. you share the scope.

```
eval('some code');
```

side effecting code!

Function Plotter: Function

avoid side effects!

Function() is like eval() but declares parameters and executes in the global scope. It creates a reference.

```
const add = Function('x','y','return x+y');
add(1, 2);
add(2, 3); // no need to re-parse
```

Excel

Note that DOM elements with id="x" appear under the reference x.

Scripting Caution

Especially in JavaScript you cannot exclude possibly harmful side effects from scripts that are loaded from foreign sources.

-> Privacy, Security, Stability

Scripting Caution

"Architecture" of self-modifying code from unreliable sources:

AJAX, PWA, Mashup, "MicroFW", ...

Licenses often require dynamic loading (Google, Facebook, etc.) !

Scripting Caution

"Gives you enough rope to hang yourself."
-- James Strachan

To Do at Home/Kitchen

Complete the Plotter and Excel.

Context in JavaScript (Adam Breindel): https://www.youtube.com/watch?list=PLndbWGuLoHea6b3g3fY77U47RiryT2Sr5 (all 4 parts! total 25 min)