

# Logic Programming

## *Reactive Worksheets*

Michael Genesereth  
Computer Science Department  
Stanford University

# Worksheets

TurboTax Free Edition 2007

https://qtwu4.turbotaxonline.intuit.com/secure/ttonline.htm?uid=120515092K3A0&src=44

TurboTax Free Edition

Home Personal Info Federal Taxes Federal Review State Taxes Print & File

You & Your Family

Tell Us About You

First Name: [ ] Middle Initial: [ ]

Last Name: [ ] Jr., Sr., etc.: [ ]

Social Security Number: [ ]

Birth Date (mm/dd/yyyy): [ ]

Occupation: [ ]

Best Contact Phone: [ ] Extension: [ ]

State of Residence on December 31, 2007: [ ]

State: [ ]

Do Any of These Apply?

A parent or someone else can claim this person as a dependent on their return.

This person served in the U.S. Armed Forces in 2007 (active, reserve, or National Guard).

This person was legally blind as of December 31, 2007. [Explain This](#)

This person passed away before filing this return.

None of the above

Continue

Round-trip
  One-way
  Multi-city

From City name or [airport](#): 
 To City name or [airport](#):

Leave: 
 Return:

Anytime 
 Anytime

Travelers ([Children or seniors?](#))

Adult (18-64)

Flight preference

I prefer non-stop flights

[Expand search options](#) (Preferred airlines, first/business class, etc.)  
[Flexible dates](#)



DEPARTMENT OF COMPUTER SCIENCE

MSCS Program Sheet (2010-11)

Artificial Intelligence Primary Specialization

Name: Charles Parnell Naut Advisor: [ ] Proposed date for degree conferral: [ ] Date: 10/8/2010

Student ID #: [ ] Email: cnaut@stanford.edu [ ] [ ] HCP?  Coterm?

GENERAL INSTRUCTIONS

Before the end of your first quarter, you should complete the following steps. Detailed instructions are included in the [Guide to the MSCS Program Sheet](#) in your orientation packet (an online version is available at [cs.stanford.edu/degrees/mscs/programsheets/](#)):

- Complete this program sheet by filling in the number, name and units of each course you intend to use for your degree.
- Create a course schedule showing the year and quarter in which you intend to take each course in your program sheet.
- Meet with your advisor and secure the necessary signatures on the program sheet.

FOUNDATIONS REQUIREMENT

You must satisfy the requirements listed in each of the following areas; all courses taken elsewhere must be approved by your advisor on a foundation course waiver form. Required documents for waiving a course include course descriptions, syllabi, and textbook lists. These documents can be organized here: [cs.stanford.edu/degrees/mscs/waivers/](#). Do not enter anything in the "Units" column for courses taken elsewhere.

Note: If you are amending an old program sheet, enter "on file" in the approval column for courses that have already been approved.

Required:	Equivalent elsewhere (course number/title/institution)	Approval	Grade	Units
Logic, Automata and Complexity (✓ CS 103)	<input type="text"/>	<input type="button" value="v"/>	<input type="text"/>	4
Probability ( <input type="checkbox"/> CS 109, <input type="checkbox"/> STATS 116, <input type="checkbox"/> CME 106, or <input type="checkbox"/> MS&E 220)	<input type="text"/>	<input type="button" value="v"/>	<input type="text"/>	
Algorithmic Analysis (✓ CS 161)	<input type="text"/>	<input type="button" value="v"/>	<input type="text"/>	5
Computer Organization and Systems (✓ CS 107)	<input type="text"/>	<input type="button" value="v"/>	<input type="text"/>	5
Principles of Computer Systems (✓ CS 110)	<input type="text"/>	<input type="button" value="v"/>	<input type="text"/>	5

TOTAL UNITS USED TO SATISFY FOUNDATIONS REQUIREMENT: **10**

Note: This total may not exceed 10 units.

7 Requirements Left Total Units: 10 Status: Draft

# Characteristics

## **Meaningful Data Display**

All data readily accessible

Tables, Charts, Graphs

## **Modifiability**

What-you-see-is-what-you-get

Random access - data can be changed in any order

## **Constraint Checking**

Completeness and Consistency

Problem alerting and Guidance in solving

## **Automatic Computation of Results**

Consequences computed

Presentation automatically updated

# Assignment - Academic Program Sheet

**DEPARTMENT OF COMPUTER SCIENCE**  
**MSCS Program Sheet (2010-11)**

Artificial Intelligence Primary Specialization

---

Name: **Charles Parnell Naut**    Advisor:    Proposed date for degree conferral:    Date: 10/8/2010  
 Student ID #:    Email: **cnaut@stanford.edu**     HCP?     Coterm?

---

**GENERAL INSTRUCTIONS**

Before the end of your first quarter, you should complete the following steps. Detailed instructions are included in the **Guide to the MSCS Program Sheet** in your orientation packet (an online version is available at [cs.stanford.edu/degrees/mscs/programsheets/](http://cs.stanford.edu/degrees/mscs/programsheets/)):

- Complete this program sheet by filling in the number, name and units of each course you intend to use for your degree.
- Create a course schedule showing the year and quarter in which you intend to take each course in your program sheet.
- Meet with your advisor and secure the necessary signatures on the program sheet.

---

**FOUNDATIONS REQUIREMENT**

You must satisfy the requirements listed in each of the following areas; all courses taken elsewhere must be approved by your adviser on a foundation course waiver form. Required documents for waiving a course include course descriptions, syllabi, and textbook lists. These document can be organized here: [cs.stanford.edu/degrees/mscs/waivers/](http://cs.stanford.edu/degrees/mscs/waivers/). Do not enter anything in the "Units" column for courses taken elsewhere.

Note: If you are amending an old program sheet, enter "**on file**" in the approval column for courses that have already been approved.

Required:	Equivalent elsewhere (course number/title/institution)	Approval	Grade	Units
Logic, Automata and Complexity ( <input checked="" type="checkbox"/> CS 103)				4
Probability ( <input type="checkbox"/> CS 109, <input type="checkbox"/> STATS 116, <input type="checkbox"/> CME 106, or <input type="checkbox"/> MS&E 220)				
Algorithmic Analysis ( <input checked="" type="checkbox"/> CS 161)				5
Computer Organization and Systems ( <input checked="" type="checkbox"/> CS 107)				5
Principles of Computer Systems ( <input checked="" type="checkbox"/> CS 110)				5

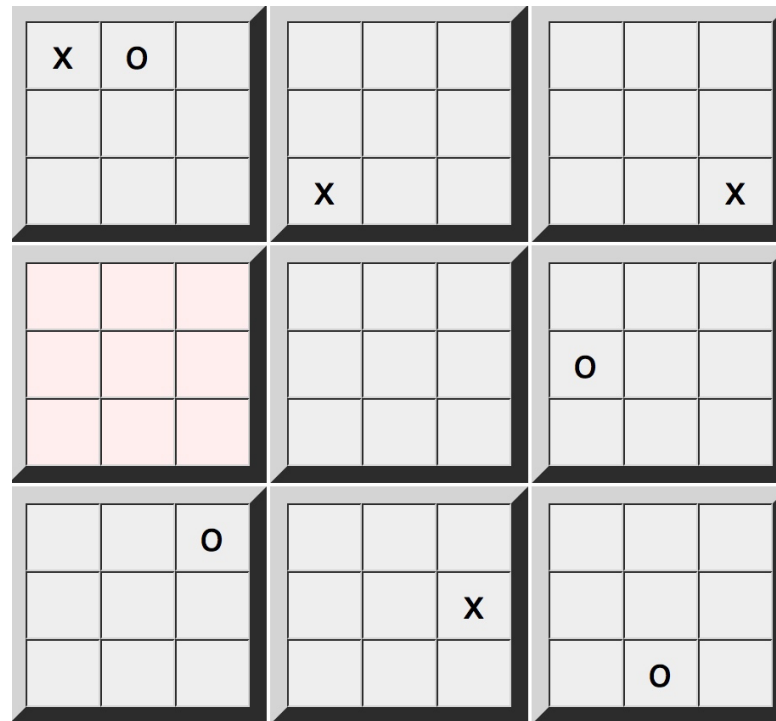
TOTAL UNITS USED TO SATISFY FOUNDATIONS REQUIREMENT: **10**

Note: This total may not exceed 10 units.

✘ 7 Requirements Left    Total Units: 10    Status: Draft

Demonstration

# Assignment - Nineboard Tic Tac Toe



Demonstration

# Example - Connect Four

Demonstration

# Example - Solar System

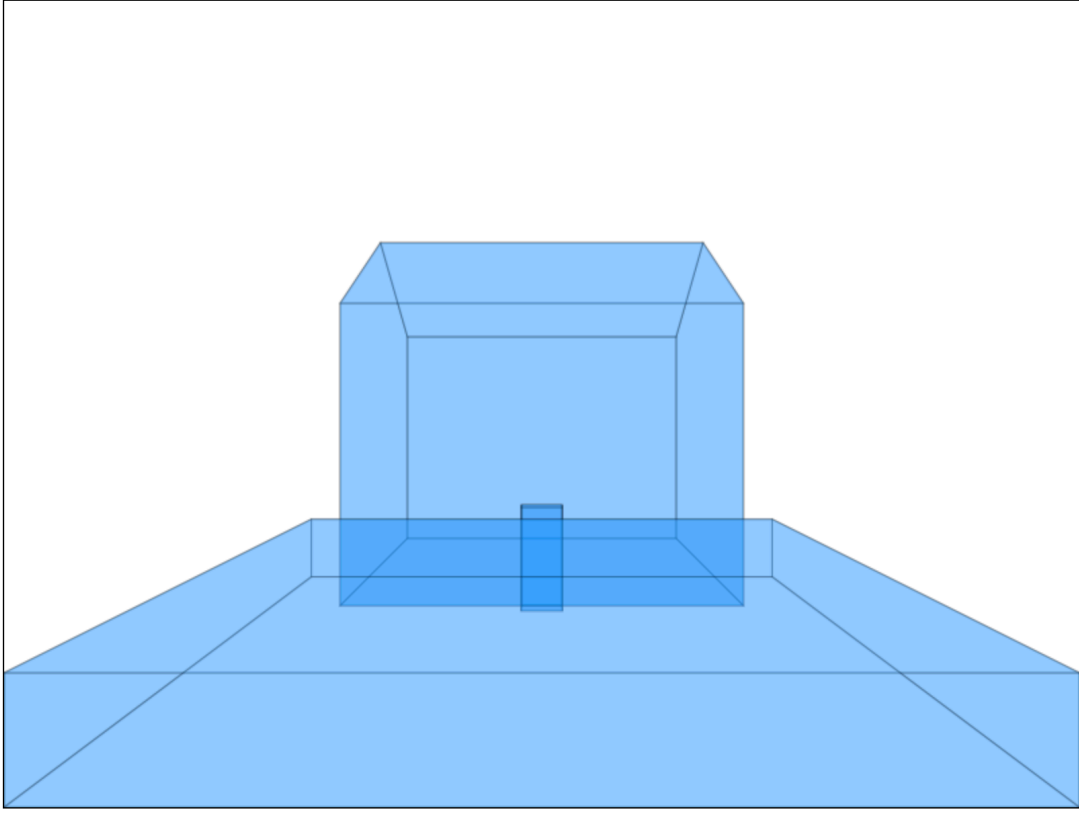
Demonstration

# Assignment - Portico

Not Secure — complaw.stanford.edu

## Portico

Use sliders to adjust view. Click and drag to move building. Click Larger, Smaller, Taller, Shorter to adjust size.



Larger Smaller Turn Stop Taller Shorter

Item	Data
Zone	R-1

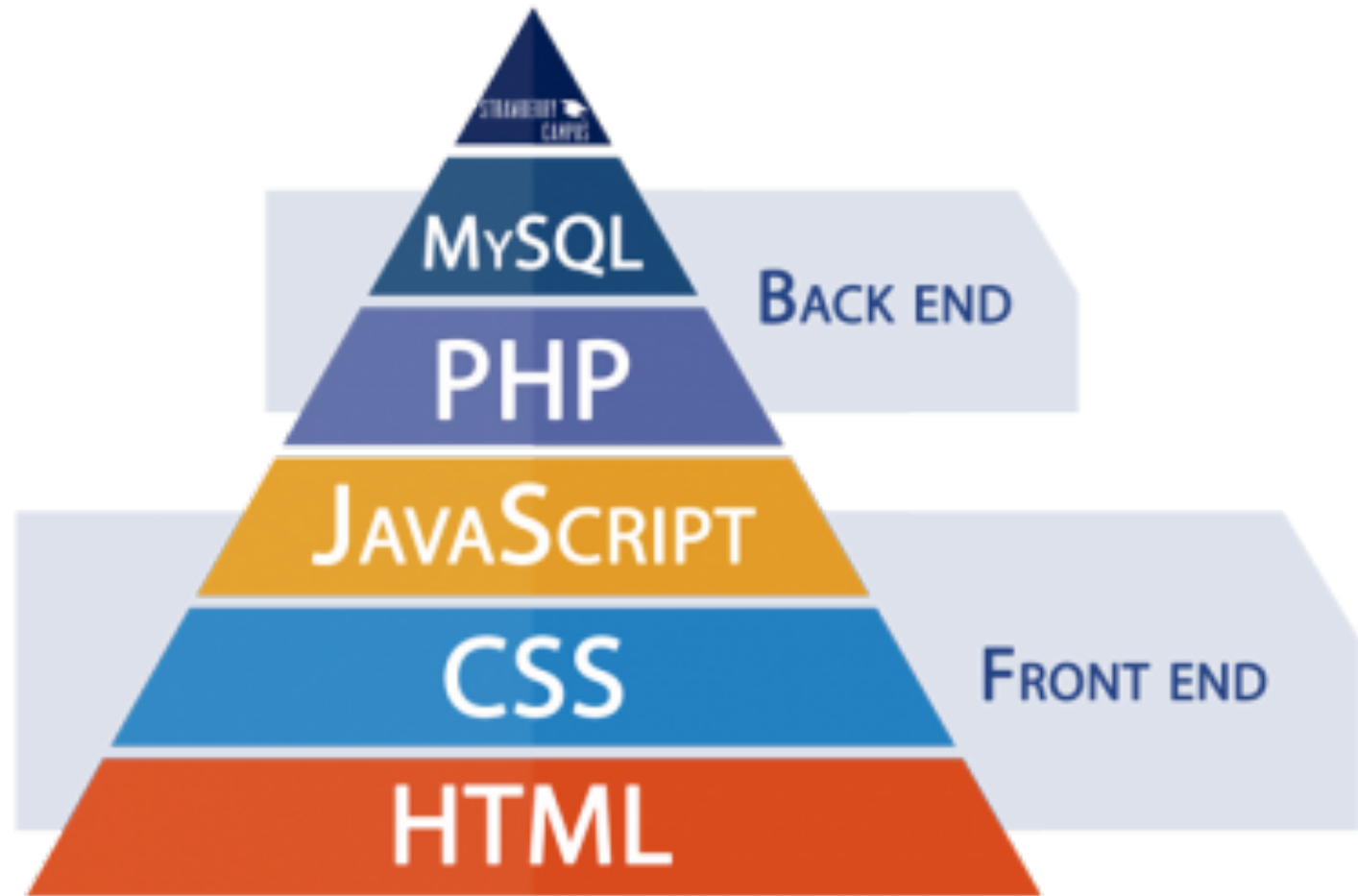
Standard	Actual	Allowed	Status
Footprint	160000	168000	✓

Item	Min	Max
Home x	200	600

Demonstration



# Current Approach



THE BIG 5

DO YOU MASTER THEM ALL?

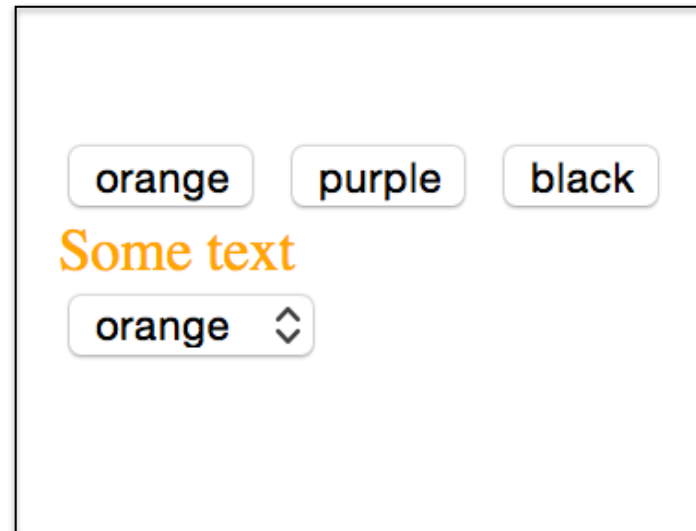
# DIY Manifesto

*Do It Yourself!*

Worksheets :: Spreadsheets

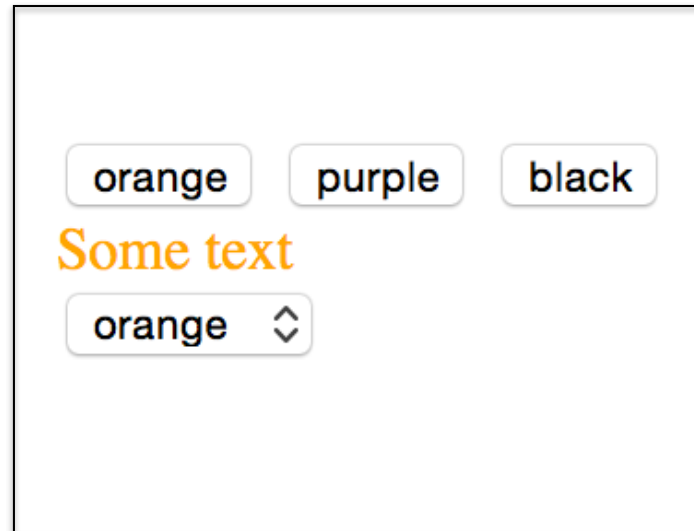
# Web Pages

# Webpage



Demonstration

# HTML Representation



orange purple black

Some text

orange ▾

```
<html>
  <body>
    <input id='o' type='button' value='orange' />
    <input id='p' type='button' value='purple' />
    <input id='b' type='button' value='black' />
    <p id='text' color='orange'>Some text.</p>
    <select id='s'>
      <option>orange</option>
      <option>purple</option>
      <option>black</option>
    </select>
  </body>
</html>
```

# "Mirror" Semantics

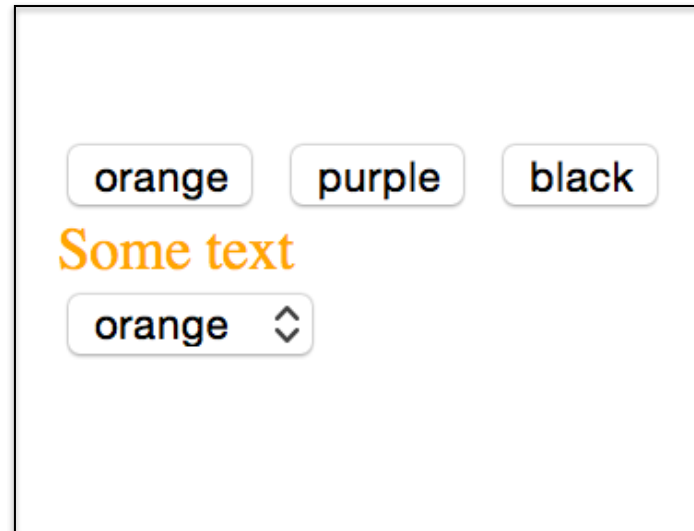
Web browsers read HTML, create internal representation called the Document Object Model (DOM), and render page.

## Dynamics

User gestures change DOM

Changes to DOM are reflected in visible web page

# Dataset Representation



```
value(o,orange)
value(p,purple)
value(b,black)
value(s,orange)

style(o,color,black)
style(p,color,black)
style(b,color,black)
style(text,color,orange)
style(s,color,black)
...
```

# Our "Mirror" Semantics

Web browsers read HTML, create internal representation called the Document Object Model (DOM) **and create dataset**, and render page.

## Dynamics

User gestures translated to actions

**Actions change the dataset**

**Changes to dataset reflected in DOM**

Changes to DOM are reflected in visible web page



# Dataset Representation

## DOM:

```
<center>  
  <input id='mynode'  
        type='text'  
        value='hello'  
        size='30'  
        style='color:black' />  
</center>
```

# Dataset Representation

## DOM:

```
<center>  
  <input id='mynode'  
        type='text'  
        value='hello'  
        size='30'  
        style='color:black' />  
</center>
```

## Dataset:

```
value(mynode,hello)  
attribute(mynode,size,30)  
style(mynode,color,black)
```

# Widget Predicates

`value(widget, value)` - true whenever the value associated with *widget* is *value*. The widget here may be a text field, selector, checkbox, radio button field, slider, and so forth.

`valuelist(widget, list)` - true whenever *list* contains the values associated with the multi-valued node *widget*. The widget in this case is typically a multi-valued selector or a checkbox field.

`options(selector, list)` - true whenever *list* contains the options for *selector*.

# Node Predicates

`rows (table, list)` - true whenever *list* contains the rows of *table*.

`innerHTML (node, string)` - true whenever the innerHTML associated with *node* is *string*.

`attribute (node, property, value)` - true whenever the *property* attribute of *node* is *value*.

`style (node, property, value)` - true whenever the *property* style of *node* is *value*.

# Actions

Gestures performed by the user:

- Making a selection from drop-down list

- Changing value of text field

- Clicking a button

Automatic Actions:

- Loading a page

- Clock tick

# Example

## DOM:

```
<input id='orange'  
      type='button'  
      value='orange' />
```

*-> user clicks*

## Action:

```
click(orange)
```

# Example

## DOM:

```
<select id='pagecolor'>  
  <option>orange</option>  
  <option>purple</option> -> user selects  
  <option>black</option>  
</select>
```

## Action:

```
select(pagecolor, purple)
```

# Example

## DOM:

```
<center>
  <input id='mynode'
        type='text'
        value='hello' -> user enters "goodbye"
        size='30'
        style='color:black' />
</center>
```

## Action:

```
select(mynode, "goodbye")
```



# Operations

`click(widget)`: This action occurs when the user clicks on *widget*.

`select(selector, value)`: This action occurs when the user enters or selects *value* as the value of *widget*.

`multiselect(multiselector, list)`: This action occurs when the user erases or deselects a value of *multiselector*. Here *list* is a list of selected values.

# Operations

`click(widget)`: This action occurs when the user clicks on *widget*.

`select(selector, value)`: This action occurs when the user enters or selects *value* as the value of *widget*.

`multiselect(multiselector, list)`: This action occurs when the user erases or deselects a value of *multiselector*. Here *list* is a list of selected values.

`tick`: This action occurs periodically (when a page contains a timer and the timer is activated). By default, it happens once per second.

`load`: This occurs when a page is first loaded.

# Buttons

orange

blue

purple

black

```
click(orange) :: style(page,color,orange)
```

```
click(blue) :: style(page,color,blue)
```

```
click(purple) :: style(page,color,purple)
```

```
click(black) :: style(page,color,black)
```

```
click(X) :: style(page,color,X)
```

# Buttons

orange

blue

purple

black

```
click(orange) :: style(page,color,orange)
```

```
click(blue) :: style(page,color,blue)
```

```
click(purple) :: style(page,color,purple)
```

```
click(black) :: style(page,color,black)
```

```
click(X) :: style(page,color,X)
```

```
click(X) ::
```

```
  style(page,color,Y) ==> ~style(page,color,Y)
```

# Selectors

orange
blue
purple
black

```
select(pagecolor,X) :: style(page,color,X)
```

# Selectors

orange
blue
purple
black

`select(pagecolor,X) :: style(page,color,X)`

`select(pagecolor,X) ::`

`style(page,color,Y) ==> ~style(page,color,Y)`

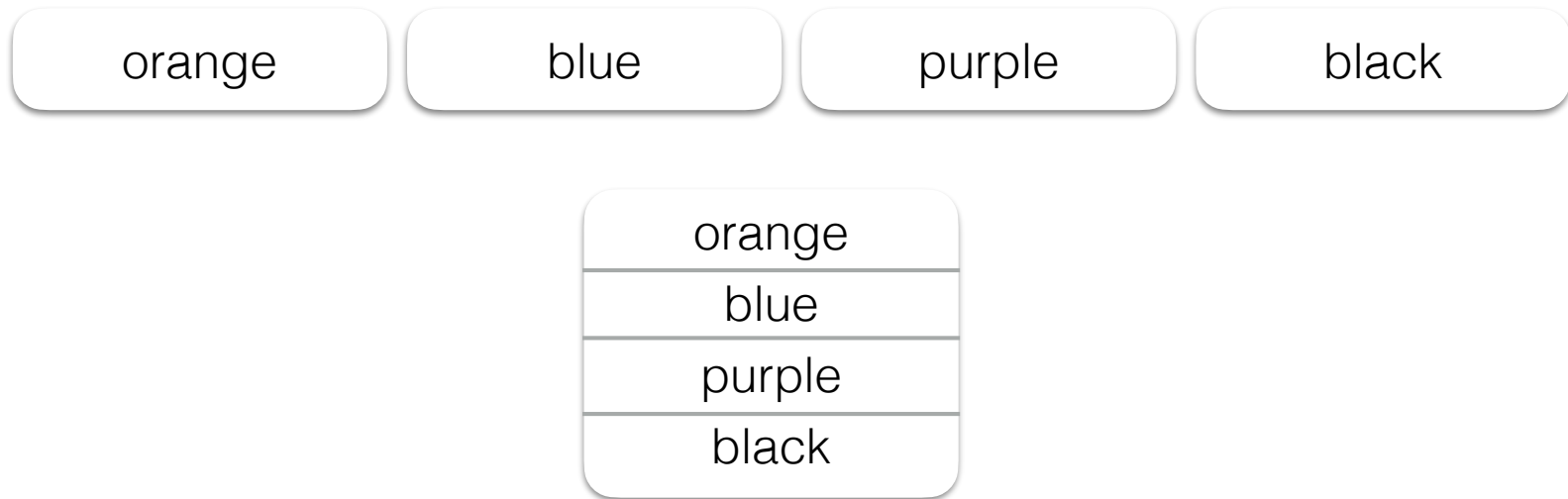
# Selectors

orange
blue
purple
black

```
select(pagecolor,X) :: style(page,color,X)
select(pagecolor,X) ::
  style(page,color,Y) ==> ~style(page,color,Y)
```

```
select(pagecolor,X) :: value(pagecolor,X)
select(pagecolor,X) ::
  value(pagecolor,Y) ==> ~value(pagecolor,Y)
```

# Interaction Between Buttons and Selectors



```
click(X) :: style(page,color,X)
```

```
click(X) ::
```

```
  style(page,color,Y) & distinct(X,Y)
```

```
  ==> ~style(page,color,Y)
```

```
click(X) :: value(pagecolor,X)
```

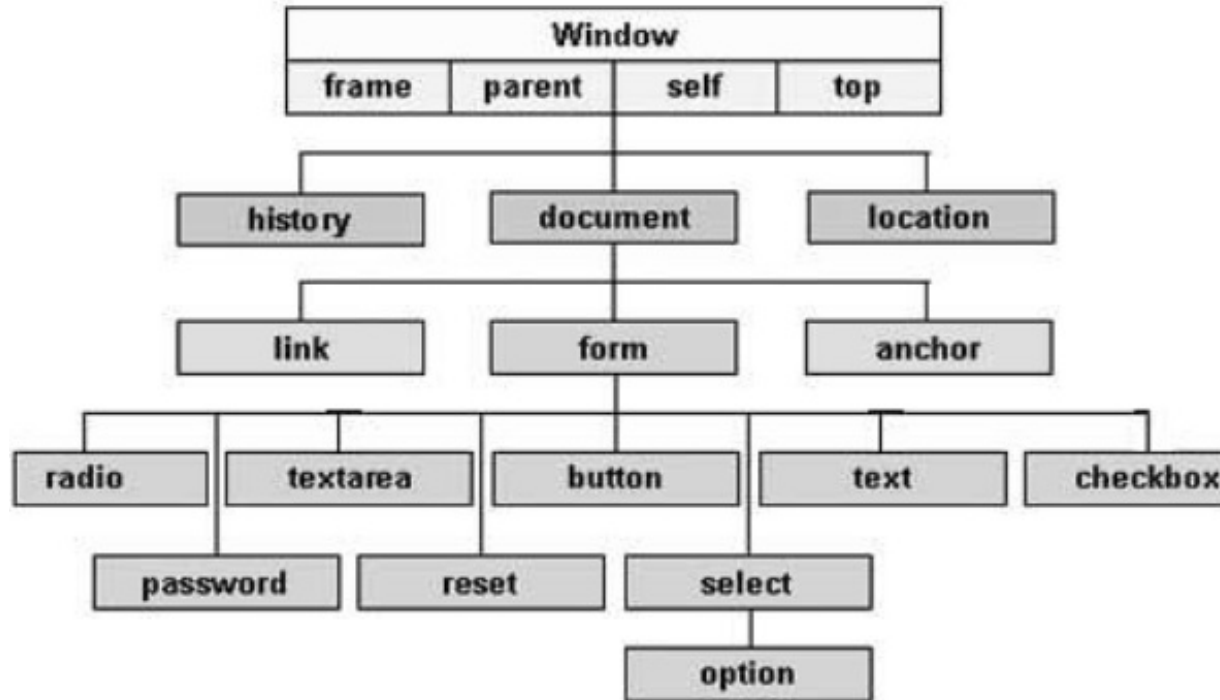
```
click(X) ::
```

```
  value(pagecolor,Y) ==> ~value(pagecolor,Y)
```



# Representational Alternatives

# Document Object Model (DOM)



NB: The DOM is a tree (not a graph).

# Term Representation

Idea - Represent DOM as a term

```
<center>
  <input id='mynode'
        type='text'
        value='hello'
        size='30'
        style='color:black' />
</center>
```

```
node(center,
     [],
     [node(input,
           [[id,mynode],
            [type,text],
            ...,
            [style,stylenode([color,black])]])])])
```

# Analysis

## Advantages

Conceptually simple and appealing

## Disadvantages

Rules are messy

Computational cost - Term update, DOM update

# Full Dataset Representation

## Idea

represent *entire* DOM

in dataset and view definitions

use operator definitions to update dataset

```
<center>                                attribute(mynode,value,hello)
  <input id='mynode'                    attribute(mynode,size,30)
    type='text'                         style(mynode,color,black)
    value='hello'                       style(mynode,"font-family",courier)
    size='30'                           style(mynode,"font-size",12px)
    style='color:black' /> ...
</center>
```

# Analysis

Advantage - conceptually simple and flexible

"Mirror semantics"

state of DOM and dataset synchronized  
changing either one changes the other

Possible to define some features as views  
(but then must define DOM gestures as operators)

Disadvantages - computational cost and coverage

Entire DOM must be updated on each cycle  
(less problematic if concentrate on nodes w/ ids)

Must ensure that the *entire* DOM is captured

# Relevant Dataset Representation

## Idea

represent *relevant* portion of DOM as dataset  
use operator definitions to update dataset

## Inertial / differential

Anything not in the dataset closure remains same

# Analysis

Disadvantages - *\*not\** mirror semantics

- Things with no ids do not change

- Cannot create new nodes without update problems

Advantage - conceptually simple

- Focussed

- Deals nicely with *\*numerous\** DOM features and updates

- Low computation cost



# Authoring

# Augmented HTML

*Augmented* HTML is plain HTML with augmentations that allow authors to use logic programs to control the *appearance* and the *behavior* of the web page.

Essentials:

Representation of the state of the page as a dataset

Values, attributes, styles via *relations*

Behavior via *operation definitions*

# Converting HTML Pages to Worksheets

Start with an HTML page.

(1) Add worksheets code.

(2) Initialize.

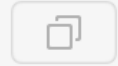
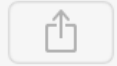
(3) Add identifiers and event handlers.

(4) Add Data and Rules.

Done.



file:///Users/mrg/Desktop/example.html



orange

purple

black

Some text

orange

# Raw HTML

```
<html>
  <head>
</head>
  <body>
    <input type='button' value='orange' />
    <input type='button' value='purple' />
    <input type='button' value='black' />
    <p color='orange'>Some text.</p>
    <select>
      <option>orange</option>
      <option>purple</option>
      <option>black</option>
    </select>
  </body>
</html>
```

# Load Worksheets Code

```
<html>
  <head>
    <script type='text/javascript'
      src='http://epilog.stanford.edu/javascript/epilog.js'></script>
    <script type='text/javascript'
      src='http://worksheets.stanford.edu/javascript/worksheets.js'></script>
  </head>
  <body>
    <input type='button' value='orange' />
    <input type='button' value='purple' />
    <input type='button' value='black' />
    <p color='orange'>Some text.</p>
    <select>
      <option>orange</option>
      <option>purple</option>
      <option>black</option>
    </select>
  </body>
</html>
```

# Initialize

```
<html>
  <head>
    <script type='text/javascript'
      src='http://epilog.stanford.edu/javascript/epilog.js'></script>
    <script type='text/javascript'
      src='http://worksheets.stanford.edu/javascript/worksheets.js'></script>
  </head>
  <body onload='initialize()'>
    <input type='button' value='orange' />
    <input type='button' value='purple' />
    <input type='button' value='black' />
    <p color='orange'>Some text.</p>
    <select>
      <option>orange</option>
      <option>purple</option>
      <option>black</option>
    </select>
  </body>
</html>
```

# Add Identifiers and Event Handlers

```
<html>
  <head>
    <script type='text/javascript'
      src='http://epilog.stanford.edu/javascript/epilog.js'></script>
    <script type='text/javascript'
      src='http://worksheets.stanford.edu/javascript/worksheets.js'></script>
  </head>
  <body id='page' onload='initialize()'>
    <input type='button' value='orange' id='orange' onclick='handle(this)' />
    <input type='button' value='purple' id='purple' onclick='handle(this)' />
    <input type='button' value='black' id='black' onclick='handle(this)' />
    <p color='orange'>Some text.</p>
    <select id='pagecolor' onchange='handle(this)'>
      <option>orange</option>
      <option>purple</option>
      <option>black</option>
    </select>
  </body>
</html>
```



# Add Data and Rules

```
<html>
  <head>
    <script type='text/javascript'
      src='http://epilog.stanford.edu/javascript/epilog.js'></script>
    <script type='text/javascript'
      src='http://minimal.stanford.edu/worksheets/javascript/worksheets.js'></
script>
  </head>
  <body id='page' onload='initialize()'>
    <input type='button' value='orange' id='orange' onclick='handle(this)'/>
    <input type='button' value='purple' id='purple' onclick='handle(this)'/>
    <input type='button' value='black' id='black' onclick='handle(this)'/>
    <p color='orange'>Some text.</p>
    <select id='pagecolor' onchange='handle(this)'\>
      <option>orange</option>
      <option>purple</option>
      <option>black</option>
    </select>
  </body>
  <textarea id='lambda' style='display:none'></textarea>
  <textarea id='library' style='display:none'>...</textarea>
</html>
```

# Documentation

<http://worksheets.stanford.edu/worksheets/introduction/index.html>

