Learning Goals

There are many tools available to help people create visualizations.

What are the strengths and weaknesses of current tools?

What trade-offs must be considered when designing a tool for creating visualizations?
How do people create visualizations?

**Chart Typology**
- Pick from a stock of templates
- Easy-to-use but limited expressiveness
- Prohibits novel designs, new data types

**Component Architecture**
- Permits more combinatorial possibilities
- Novel views require new operators, which requires software engineering
Graphics APIs
Canvas, OpenGL, Processing
Eye(int tx, int ty, int ts) {
    x = tx;
    y = ty;
    size = ts;
}

void update(int mx, int my) {
    angle = atan2(my-y, mx-x);
}

void display() {
    pushMatrix();
    translate(x, y);
    fill(255);
    ellipse(0, 0, size, size);
    rotate(angle);
    fill(153, 204, 0);
    ellipse(size/4, 0, size/2, size/2);
    popMatrix();
}
Graphics APIs
Canvas, OpenGL, Processing
Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Canvas, OpenGL, Processing
Data State Model

[Chi 98]
Prefuse & Flare

Operator-based toolkits for visualization design

Vis = (Input Data -> Visual Objects) + Operators

Prefuse (http://prefuse.org)  Flare (http://flare.prefuse.org)
Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Canvas, OpenGL, Processing
Chart Typologies
Excel, Google Charts

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Canvas, OpenGL, Processing
Chart Typologies
## Data Sets: State Quick Facts

 Uploaded By:  zinggoat                                      Created at:  Friday May 18, 3:08 PM
 Data Source:  US Census Bureau                                Description:  
 Tags:  people census

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Alabama</td>
<td>4557808</td>
<td>0.03</td>
<td>4447100</td>
<td>0.1</td>
<td>0.07</td>
<td>0.24</td>
<td>0.13</td>
</tr>
<tr>
<td>2. Alaska</td>
<td>663661</td>
<td>0.06</td>
<td>626932</td>
<td>0.14</td>
<td>0.08</td>
<td>0.29</td>
<td>0.06</td>
</tr>
<tr>
<td>3. Arizona</td>
<td>5939292</td>
<td>0.16</td>
<td>5130632</td>
<td>0.4</td>
<td>0.08</td>
<td>0.27</td>
<td>0.13</td>
</tr>
<tr>
<td>4. Arkansas</td>
<td>2779154</td>
<td>0.04</td>
<td>2673400</td>
<td>0.14</td>
<td>0.07</td>
<td>0.25</td>
<td>0.14</td>
</tr>
<tr>
<td>5. California</td>
<td>36132147</td>
<td>0.07</td>
<td>33871648</td>
<td>0.14</td>
<td>0.07</td>
<td>0.27</td>
<td>0.11</td>
</tr>
<tr>
<td>6. Colorado</td>
<td>4665177</td>
<td>0.08</td>
<td>4301261</td>
<td>0.31</td>
<td>0.07</td>
<td>0.26</td>
<td>0.1</td>
</tr>
<tr>
<td>7. Connecticut</td>
<td>3510297</td>
<td>0.03</td>
<td>3405565</td>
<td>0.04</td>
<td>0.06</td>
<td>0.24</td>
<td>0.14</td>
</tr>
<tr>
<td>8. Delaware</td>
<td>843524</td>
<td>0.08</td>
<td>783600</td>
<td>0.18</td>
<td>0.07</td>
<td>0.23</td>
<td>0.13</td>
</tr>
<tr>
<td>9. Florida</td>
<td>17789864</td>
<td>0.11</td>
<td>15982378</td>
<td>0.24</td>
<td>0.06</td>
<td>0.23</td>
<td>0.17</td>
</tr>
<tr>
<td>10. Georgia</td>
<td>9072576</td>
<td>0.11</td>
<td>8186453</td>
<td>0.26</td>
<td>0.08</td>
<td>0.26</td>
<td>0.1</td>
</tr>
<tr>
<td>11. Hawaii</td>
<td>1275194</td>
<td>0.05</td>
<td>1211537</td>
<td>0.09</td>
<td>0.07</td>
<td>0.24</td>
<td>0.14</td>
</tr>
<tr>
<td>12. Idaho</td>
<td>1429096</td>
<td>0.1</td>
<td>1293953</td>
<td>0.29</td>
<td>0.07</td>
<td>0.27</td>
<td>0.11</td>
</tr>
<tr>
<td>13. Illinois</td>
<td>12763371</td>
<td>0.03</td>
<td>12419293</td>
<td>0.09</td>
<td>0.07</td>
<td>0.26</td>
<td>0.12</td>
</tr>
</tbody>
</table>
Choosing a visualization type for State Quick Facts

Analyze a text

**Tag Cloud**
How are you using your words? This enhanced tag cloud will show you the words popularity in the given set of text.
Learn more

**Wordle**
Wordle is a toy for generating ‘word clouds’ from text that you provide. The clouds give greater prominence to words that appear more frequently in the source text.
Learn more

**Word Tree**
See a branching view of how a word or phrase is used in a text. Navigate the text by zooming and clicking.
Learn more

Compare a set of values

**Bar Chart**
How do the items in your data set stack up? A bar chart is a simple and recognizable way to compare values. You can display several sets of bars for multivariate comparisons.
Learn more

**Block Histogram**
This versatile chart lets you get a quick sense of how a single set of data is distributed. Each item in the data is an individually identifiable block.
Learn more
Every Wednesday, when I get home from school, I have a piano lesson. My teacher is a very strict house. Her name is Hillary Clinton. Our piano is a Steinway Concert tree and it has 88 cups. It also has a soft pedal and a/an Smiley pedal. When I have a lesson, I sit down on the piano Alberto and play for 16 minutes. I do scales to exercise my cats, and then I usually play a minuet by Johann Sebastian Washington. Teacher says I am a natural Haunted House and have a good musical leg. Perhaps when I get better I will become a concert vet and give a recital at Carnegie hospital.
[M]ost charting packages channel user requests into a **rigid array of chart types**. To atone for this lack of flexibility, they offer a kit of post-creation editing tools to return the image to what the user originally envisioned. **They give the user an impression of having explored data rather than the experience.**

Leland Wilkinson

*The Grammar of Graphics, 1999*
Chart Typologies
Excel, Many Eyes, Google Charts

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Canvas, OpenGL, Processing
Chart Typologies
Excel, Many Eyes, Google Charts

Visual Analysis Grammars
VizQL, ggplot2

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Canvas, OpenGL, Processing
ggplot(diamonds, aes(x=price, fill=cut)) + geom_bar(position="dodge")
ggplot(diamonds, aes(x=price, fill=cut)) + geom_bar(position="dodge")
qplot(long, lat, data = expo, geom = "tile", fill = ozone, 
facets = year ~ month) +
scale_fill_gradient(low = "white", high = "black") + map
```
Plot.plot({
  grid: true,
  facet: {
    data: athletes,
    y: "sex"
  },
  marks: [
    Plot.rectY(athletes, Plot.binX({y: "count"}, {x: "weight", fill: "sex"})),
    Plot.ruleY([0])
  ]
})
```
Chart Typologies
Excel, Many Eyes, Google Charts

Visual Analysis Grammars
VizQL, ggplot2

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Canvas, OpenGL, Processing
Chart Typologies
Excel, Many Eyes, Google Charts

Visual Analysis Grammars
VizQL, ggplot2

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Canvas, OpenGL, Processing

Ease-of-Use
Chart Typologies
Excel, Many Eyes, Google Charts

Visual Analysis Grammars
VizQL, ggplot2

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Canvas, OpenGL, Processing
Chart Typologies
Excel, Many Eyes, Google Charts

Visual Analysis Grammars
VizQL, ggplot2

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Canvas, OpenGL, Processing
Chart Typologies
Excel, Many Eyes, Google Charts

Visual Analysis Grammars
VizQL, ggplot2

Visualization Grammars
Protovis, D3.js

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Canvas, OpenGL, Processing
Today's first task is not to invent wholly new [graphical] techniques, though these are needed. Rather we need most vitally to recognize and reorganize the essential of old techniques, to make easy their assembly in new ways, and to modify their external appearances to fit the new opportunities.

J. W. Tukey, M. B. Wilk

*Data Analysis & Statistics, 1965*
Visualization Grammar
Visualization Grammar

Data

Input data to visualize
Visualization Grammar

Data
Input data to visualize

Transforms
Group, aggregate, stats, layout
Visualization Grammar

Data: Input data to visualize
Transforms: Group, aggregate, stats, layout
Scales: Map data values to visual values
Visualization Grammar

Data
Input data to visualize

Transforms
Group, aggregate, stats, layout

Scales
Map data values to visual values

Guides
Axes & legends visualize scales
Visualization Grammar

Data
Input data to visualize

Transforms
Group, aggregate, stats, layout

Scales
Map data values to visual values

Guides
Axes & legends visualize scales

Marks
Data-representative graphics
Protovis: A Grammar for Visualization

A graphic is a composition of data-representative marks.

with Mike Bostock & Vadim Ogievetsky
MARKS: Protovis graphical primitives
<table>
<thead>
<tr>
<th>MARK</th>
<th>$\lambda : D \rightarrow R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>visible</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>left</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>bottom</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>width</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>height</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>fillStyle</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>strokeStyle</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>lineWidth</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>...</td>
<td>$\lambda$</td>
</tr>
</tbody>
</table>
### RECT

<table>
<thead>
<tr>
<th>data</th>
<th>1</th>
<th>1.2</th>
<th>1.7</th>
<th>1.5</th>
<th>0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>visible</td>
<td>true</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>left</td>
<td>$\lambda$: index * 25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bottom</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>width</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>height</td>
<td>$\lambda$: datum * 80</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fillStyle</td>
<td>blue</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>strokeStyle</td>
<td>black</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lineWidth</td>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### RECT

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1.2</th>
<th>1.7</th>
<th>1.5</th>
<th>0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>visible</td>
<td>true</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>left</td>
<td>0 * 25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bottom</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>width</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>height</td>
<td>1 * 80</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fillStyle</td>
<td>blue</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>strokeStyle</td>
<td>black</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lineWidth</td>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \lambda : D \rightarrow R \]
<table>
<thead>
<tr>
<th>data</th>
<th>1</th>
<th>1.2</th>
<th>1.7</th>
<th>1.5</th>
<th>0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>visible</td>
<td>true</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>left</td>
<td>1 * 25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bottom</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>width</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>height</td>
<td>1.2 * 80</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fillStyle</td>
<td>blue</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>strokeStyle</td>
<td>black</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lineWidth</td>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1.2</td>
<td>1.7</td>
<td>1.5</td>
<td>0.7</td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
<td>1.2</td>
<td>1.7</td>
<td>1.5</td>
<td>0.7</td>
</tr>
<tr>
<td>visible</td>
<td>true</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>left</td>
<td>2 * 25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bottom</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>width</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>height</td>
<td>1.7 * 80</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fillStyle</td>
<td>blue</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>strokeStyle</td>
<td>black</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lineWidth</td>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### RECT

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1.2</th>
<th>1.7</th>
<th>1.5</th>
<th>0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>visible</td>
<td>true</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>left</td>
<td>3 * 25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bottom</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>width</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>height</td>
<td>1.5 * 80</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fillStyle</td>
<td>blue</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>strokeStyle</td>
<td>black</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lineWidth</td>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECT</td>
<td>λ : D → R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-----------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>data</td>
<td>1</td>
<td>1.2</td>
<td>1.7</td>
<td>1.5</td>
<td>0.7</td>
</tr>
<tr>
<td>visible</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>true</td>
</tr>
<tr>
<td>left</td>
<td></td>
<td></td>
<td></td>
<td>4 * 25</td>
<td></td>
</tr>
<tr>
<td>bottom</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>width</td>
<td></td>
<td></td>
<td></td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>height</td>
<td></td>
<td></td>
<td></td>
<td>0.7 * 80</td>
<td></td>
</tr>
<tr>
<td>fillStyle</td>
<td></td>
<td></td>
<td></td>
<td>blue</td>
<td></td>
</tr>
<tr>
<td>strokeStyle</td>
<td></td>
<td></td>
<td></td>
<td>black</td>
<td></td>
</tr>
<tr>
<td>lineWidth</td>
<td></td>
<td></td>
<td></td>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
**RECT**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>1</td>
</tr>
<tr>
<td>visible</td>
<td>true</td>
</tr>
<tr>
<td>left</td>
<td>$\lambda$: index * 25</td>
</tr>
<tr>
<td>bottom</td>
<td>0</td>
</tr>
<tr>
<td>width</td>
<td>20</td>
</tr>
<tr>
<td>height</td>
<td>$\lambda$: datum * 80</td>
</tr>
<tr>
<td>fillStyle</td>
<td>blue</td>
</tr>
<tr>
<td>strokeStyle</td>
<td>black</td>
</tr>
<tr>
<td>lineWidth</td>
<td>1.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
var vis = new pv.Panel();
vis.add(pv.Bar)
.data([1, 1.2, 1.7, 1.5, 0.7])
.visible(true)
.left((d) => this.index * 25);
.bottom(0)
.width(20)
.height((d) => d * 80)
.fillStyle("blue")
.strokeStyle("black")
.lineWidth(1.5);
vis.render();
var army = pv.nest(napoleon.army, "dir", "group");
var vis = new pv.Panel();

var lines = vis.add(pv.Panel).data(army);
lines.add(pv.Line)
   .data(() => army[this.idx])
   .left(lon).top(lat).size((d) => d.size/8000)
   .strokeStyle(() => color[army[paneIndex][0].dir]);

vis.add(pv.Label).data(napoleon.cities)
   .left(lon).top(lat)
   .text((d) => d.city).font("italic 10px Georgia")
   .textAlign("center").textBaseline("middle");
PRELUDE NO.1 IN C MAJOR, BWV 846
(FROM WELL-TEMPERED CLAVIER, BOOK 1)

BY J.S. BACH

Bach’s Prelude #1 in C Major | Jieun Oh
d3.js  Data-Driven Documents

with Mike Bostock, Jason Davies & Vadim Ogievetsky
Protovis

*Specialized mark types*
- Streamlined design
- Limits expressiveness
- More overhead (slower)
- Harder to debug
- Self-contained model

*Specify a scene (nouns)*
- Quick for static vis
- Delayed evaluation
- Animation, interaction are more cumbersome
### Protovis

*Specialized mark types*
- Streamlined design
- Limits expressiveness
- More overhead (slower)
- Harder to debug
- Self-contained model

*Specify a scene (nouns)*
- Quick for static vis
- Delayed evaluation
- Animation, interaction are more cumbersome

### D3

*Bind data to DOM*
- Exposes SVG/CSS/…
- Less overhead (faster)
- Debug in browser
- Use with other tools

*Transform a scene (verbs)*
- More complex model
- Immediate evaluation
- Dynamic data, anim, and interaction natural
D3 Selections

The core abstraction in D3 is a selection.
The core abstraction in D3 is a **selection**.

```javascript
// Add and configure an SVG element (<svg width="500" height="300">)
var svg = d3.append("svg")
  .attr("width", 500) // set SVG width to 500px
  .attr("height", 300); // set SVG height to 300px
```
The core abstraction in D3 is a selection.

```
// Add and configure an SVG element (<svg width="500" height="300">)
var svg = d3.append("svg")
  .attr("width", 500)
  .attr("height", 300);
// Select & update existing rectangles contained in the SVG element
svg.selectAll("rect")
  .attr("width", 100)
  .style("fill", "steelblue");
```

D3 Selections
Data Binding

Selections can **bind** data and DOM elements.

```javascript
var values = [ {…}, {…}, {…}, ... ]; // input data as JS objects
```
Data Binding

Selections can **bind** data and DOM elements.

```javascript
var values = [ {…}, {…}, {…}, … ]; // input data as JS objects
// Select SVG rectangles and bind them to data values.

var bars = svg.selectAll("rect.bars").data(values);
```
Data Binding

Selections can *bind* data and DOM elements.

```javascript
var values = [ {…}, {…}, {…}, ... ]; // input data as JS objects
// Select SVG rectangles and bind them to data values.

var bars = svg.selectAll("rect.bars").data(values);
// What if the DOM elements don’t exist yet? The `enter` set represents data
// values that do not yet have matching DOM elements.

bars.enter().append("rect").attr("class", "bars");
```
Data Binding

Selections can **bind** data and DOM elements.

```javascript
var values = [ {…}, {…}, {…}, … ]; // input data as JS objects
// Select SVG rectangles and bind them to data values.

var bars = svg.selectAll(“rect.bars”).data(values);
// What if the DOM elements don’t exist yet? The enter set represents data
// values that do not yet have matching DOM elements.

bars.enter().append(“rect”).attr(“class”, “bars”);
// What if data values are removed? The exit set is a selection of existing
// DOM elements who no longer have matching data values.

bars.exit().remove();
```
The Data Join

**ENTER**
Data values without matching DOM elements.

**UPDATE**
Existing DOM elements, bound to valid data.

**EXIT**
DOM elements whose bound data has gone “stale”.
The Data Join

```javascript
var s = d3.selectAll(...).data(...)
```

**DATA VALUES**
- **ENTER**
  - Data values without matching DOM elements.
  - `s.enter().append(...)`

**ELEMENTS**
- **UPDATE**
  - Existing DOM elements, bound to valid data.
  - `s`
- **EXIT**
  - DOM elements whose bound data has gone "stale".
  - `s.exit()`
Data Binding

Selections can **bind** data and DOM elements.

```javascript
var values = [ {…}, {…}, {…}, ... ]; // input data as JS objects
// Select SVG rectangles and bind them to data values.
var bars = svg.selectAll("rect.bars").data(values)
  .join(
    enter => enter.append("rect"), // create new
    update => update, // update current
    exit => exit.remove() // remove outdated
  )
```
D3 Modules

Data Parsing / Formatting (JSON, CSV, …)
Shape Helpers (arcs, curves, areas, symbols, …)
Scale Transforms (linear, log, ordinal, …)
Color Spaces (RGB, HSL, LAB, …)
Animated Transitions (tweening, easing, …)
Geographic Mapping (projections, clipping, …)
Layout Algorithms (stack, pie, force, trees, …)
Interactive Behaviors (brush, zoom, drag, …)

Many of these correspond to future lecture topics!
Chart Typologies
Excel, Many Eyes, Google Charts

Visual Analysis Grammars
VizQL, ggplot2

Visualization Grammars
Protovis, D3.js

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Canvas, OpenGL, Processing
Administrivia
A2: Deceptive Visualization

Design **two** static visualizations for a dataset:

1. An **earnest** visualization that faithfully conveys the data
2. A **deceptive** visualization that tries to mislead viewers

Your two visualizations need to address different questions.

Try to design a deceptive visualization that appears to be earnest. Can you trick your classmates and course staff?

You are free to choose your own dataset, but we have also provided some preselected datasets for you.

Submit two images and a brief write-up on Gradescope.

Due by **Wed 10/19 11:59pm**.
A2 Peer Reviews

On Friday 4/21 you will be assigned two peer A2 submissions to review. For each:

• Try to determine which is earnest and which is deceptive
• Share a rationale for how you made your determination
• Share feedback using the “I Like / I Wish / What If” rubric

Assigned reviews will be posted on the A2 Peer Review page on Canvas, along with a link to a Google Form. You should submit two forms: one for each A2 peer review. Due by Mon 4/24 11:59pm.
Assignment Regrades

Students can request a regrade through Gradescope but must justify the reasons for the regrade.

Timeline: 72 hours after the grade is released

We will review the entire assignment in more detail, which could result in a higher OR lower grade.
A3: Interactive Prototype

Create an interactive visualization. Choose a driving question for a dataset and develop an appropriate visualization + interaction techniques, then deploy your visualization on the web.

Due by 11:59pm on Monday, Nov 7. We encourage you to form teams of 3-4 people.
Form a **team of 3-4** for A3 and the Final Project.  
(Start thinking about your Final Project, too!)

A3 is open-ended. You can use it to start exploring your FP topic if you like, or expand on A2.

Submit signup form by **Fri 10/28, 11:59pm**.  
**If you do not have team mates**, you should:  
- Post on Ed about your interests/project ideas
Team Member Roles

We encourage you to structure team responsibilities!

**Coordinator**: Organize meetings, track deadlines, etc.

**Data Lead**: Data wrangling, management, distillation

**Tech Lead**: Manage code integration, GitHub repo

**UX Lead**: Visualization/interaction design & evaluation

One may have multiple roles, share work across roles…
Requirements

**Interactive.** You must implement interaction methods! However, this is not only selection / filtering / tooltips. Also consider annotations or other narrative features to draw attention and provide additional context.

**Web-based.** D3 is encouraged, but not required. Deploy visualization with GitHub pages or Observable.

**Write-up.** Provide design rationale on your web page.
Interactive Prototype Tips

Start now. It will take longer than you think.

Keep it simple. Choose a *minimal* set of interactions that enables users to explore and generate interesting insights. Do not feel obligated to convey *everything* about the data: focus on a compelling subset.

Promote engagement. How do your chosen interactions reveal interesting observations?
Two Tutorials This Week

**D3.js Deep Dive:** Thursday 10/27 during lecture, Led by Vishal and Tukey

Be sure to read the D3, Part 1 notebook ahead of time. We’ll work through Part 2 in class. Also read the JS/Observable primer if you’re new to this!

**Web Publishing:** Friday 10/28 4:30-6pm in G20, Led by Aakash and Wei Jun
A Visualization Tool Stack
Chart Typologies
Excel, Many Eyes, Google Charts

Visual Analysis Grammars
VizQL, ggplot2

Visualization Grammars
Protovis, D3.js

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Canvas, OpenGL, Processing
Chart Typologies
Excel, Many Eyes, Google Charts

Visual Analysis Grammars
VizQL, ggplot2

Visualization Grammars
Protovis, D3.js

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Canvas, OpenGL, Processing

Charting Tools

Declarative Languages

Programming Toolkits
Chart Typologies
Excel, Many Eyes, Google Charts

Visual Analysis Grammars
VizQL, ggplot2

Visualization Grammars
Protovis, D3.js

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Canvas, OpenGL, Processing
What is a Declarative Language?

Programming by describing what, not how

Separate **specification** *(what you want)* from **execution** *(how it should be computed)*

In contrast to **imperative programming**, where you must give explicit steps.
What is a Declarative Language?

Programming by describing \textit{what}, not \textit{how}

Separate \textbf{specification} (\textit{what you want}) from \textbf{execution} (\textit{how it should be computed})

In contrast to \textbf{imperative programming}, where you must give explicit steps.

d3.selectAll("rect")
  .data(my_data)
  .join("rect")
  .attr("x", d => xscale(d.foo))
  .attr("y", d => yscale(d.bar))
SELECT customer_id, customer_name, 
COUNT(order_id) as total 
FROM customers 
INNER JOIN orders ON 
customers.customer_id = orders.customer_id 
GROUP BY customer_id, customer_name 
HAVING COUNT(order_id) > 5 
ORDER BY COUNT(order_id) DESC
Why Declarative Languages?

- Faster iteration, less code, larger user base?
- Better visualization. *Smart defaults.*
- **Reuse.** *Write-once, then re-apply.*
- **Performance.** *Optimization, scalability.*
- **Portability.** *Multiple devices, renderers, inputs.*
- **Programmatic generation.** *Write programs which output visualizations.*
- **Automated search & recommendation.**
Chart Typologies
Excel, Many Eyes, Google Charts

Visual Analysis Grammars
VizQL, ggplot2

Visualization Grammars
Protovis, D3.js

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Processing, OpenGL, Java2D
Chart Typologies
Excel, Many Eyes, Google Charts

Visual Analysis Grammars
VizQL, ggplot2, **Vega-Lite**

Visualization Grammars
Protovis, D3.js, **Vega**

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Processing, OpenGL, Java2D
Chart Typologies
Excel, Many Eyes, Google Charts

Visual Analysis Grammars
VizQL, ggplot2, **Vega-Lite**

Visualization Grammars
Protovis, D3.js, **Vega**

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Processing, OpenGL, Java2D

Charting Tools
Declarative Languages
Programming Toolkits
Visual Analysis Grammars
VizQL, ggplot2, **Vega-Lite**

Visualization Grammars
Protovis, D3.js, **Vega**

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Processing, OpenGL, Java2D
Interactive Data Exploration
  Tableau, Lyra, Voyager

Visual Analysis Grammars
  VizQL, ggplot2, Vega-Lite

Visualization Grammars
  Protovis, D3.js, Vega

Component Architectures
  Prefuse, Flare, Improvise, VTK

Graphics APIs
  Processing, OpenGL, Java2D

Graphical Interfaces

Declarative Languages

Programming Toolkits
The Lyra Visualization Design Environment (VDE) alpha
Arvind Satyanarayan, Kanit "Ham" Wongsuphasawat, Jeffrey Heer

William Playfair’s classic chart comparing the price of wheat and wages in England recreated in the Lyra VDE.

See also: Charticulator, Data Illustrator
Lyra  A Visualization Design Environment

Driving Shifts into Reverse by Hannah Fairfield, NYTimes
Lyra  A Visualization Design Environment

by William Playfair
Lyra  A Visualization Design Environment

based on the Railway Timetable by E. J. Marey
Lyra - A Visualization Design Environment

ZipScribble by Robert Kosara
Voyager. Wongsuphasawat et al. InfoVis’15, CHI’17
**Key Idea:** Augment manual exploration with visualization recommendations sensitive to the user’s current focus.

The goal is to support systematic consideration of the data, without exacerbating false discovery.

To model a user’s search frontier, we enumerate related Vega-Lite specifications, seeded by the user’s current focus.

Candidate charts are pruned and ranked using models of estimated perceptual effectiveness.
Interactive Data Exploration
   Tableau, **Lyra, Voyager**

Visual Analysis Grammars
   VizQL, ggplot2, **Vega-Lite**

Visualization Grammars
   Protovis, D3.js, **Vega**

Component Architectures
   Prefuse, Flare, Improvise, VTK

Graphics APIs
   Processing, OpenGL, Java2D
Summary

There is no one-size-fits-all tool for visualization.

Instead, visualization tools fall along a spectrum ranging from graphical interfaces to advanced programming toolkits.

Visualization tools make deliberate tradeoffs between ease of use and expressiveness, placing them at specific points along the spectrum.

Users often select and switch between various tools to meet their current needs.